Verhalten, Kommunikation und Interaktion in Virtuellen 3D-Umgebungen

DISSERTATION

der Fakultät für Informatik

der Eberhard-Karls-Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

DIPL.-BIOL. JÜRGEN FECHTER

aus Sigmaringen

Tübingen

1999

Tag der mündlichen Qualifikation: 7. Juli 1999

Dekan: Prof. Dr. Klaus-Jörn Lange

1. Berichterstatter: Prof. Dr. Wolfgang Straßer

2. Berichterstatter: Prof. Dr. Wolfgang Rosenstiel

Zusammenfassung

Zu Beginn der Arbeit 1993 mußten zuerst die Grundlagen und die Basis für den Bereich der Virtuellen Realität erarbeitet werden. Es werden deshalb 3D-Geräte, Grundlagen und Techniken, Formen der Interaktion sowie angrenzende Bereiche, wie beispielsweise Netzwerkprotokolle, behandelt. Im Kern stellt diese Arbeit erstmalig ein generisches Konzept für den Austausch von Nachrichten zwischen 3D-Objekten vor. Es werden die Möglichkeiten und Grenzen dieses Modells, das allgemein Messageconcept genannt wurde, aufgezeigt. Ferner wird die Adaption des Messageconcept auf verschiedene andere Einsatzebenen dargestellt. Besonders wird hier auf eine VRML-basierte Lösung eingegangen, die auch als Vorschlag dem VRML-Konsortium eingereicht wurde. Der Austausch der Nachrichten ist nicht auf eine Applikation beschränkt, sondern kann auch netzwerktransparent erfolgen. Auf die enthaltenen Multi-User-Erweiterungen wird im Zusammenhang mit dieser Arbeit nicht eingegangen.

Für die Verifikation des allgemeinen Messageconcept wurde eine Anwendung aus dem medizinischen Bereich, ein sogenannter Lichtkasten (engl. Alternator), an dem Röntgenaufnahmen befestigt werden, inklusive dazugehörigem Raum, verwendet. Im Gegensatz zu industriellen Lösungen, welche die Umgebung der Radiologen auf die Darstellung der zweidimensionalen medizinischen Bilder beschränken, bildet diese Applikation den dreidimensionalen Arbeitsraum ab. Neben dem Vorteil, daß diese Abbildung aus der realen Welt für einen nicht-computererfahrenen Mediziner intuitiv verständlich ist, steht der Nachteil des höheren Eingabeaufwands. Aufgrund der hohen Leistung, die eine 3D-Anwendung von den Rechnern fordert, mußten die Szenen vereinfacht werden, damit eine adäquate Interaktion möglich ist.

Abstract

In this thesis we show a prototype of a medical application for radiologists which is based on his real working environment. This was realized with help of 3D and VR techniques. Further we have developed a new message-based inter-object communication model which we called message concept. We present a classification of these objects, take a look of their architecture and how they work and can exchange messages. They are able to send and receive messages between applications and among a network. The message concept can be applied to many other areas, some of them are envisaged like help-support or special version for VRML support.

By the way we think that the graphic render performance is still the lack why there are no more Virtual Environment applications available today.

Für den eiligen Leser den Roten Faden auf einen Blick:

Was wurde gemacht:

- Verschiedene 3D-Ein- und Ausgabegeräte und Benutzerführungen (engl. Metaphor) angewendet und kritisch bewertet.
- Graphische 2D-Benutzungsoberflächen, 3D- und VR-Oberflächen gegenübergestellt (Vor- und Nachteile, Restriktionen, Komplexität, Interaktionsaufwand, Informationsgehalt, reale vs. künstliche Objekte).
- Klassifikation und anschließende Standardisierung von 3D-Objekten für Virtuelle Umgebungen ausgearbeitet.
- Ein Kommunikationsmodell (Messageconcept) entworfen, realisiert und dieses an VRML-2.0 angepaßt.
- Verteilung von Nachrichten über ein Netzwerk (Vorstufe zu CSCW und Multi-User-Applikationen).
- Ein Prototyp aus der Radiologie, genannt "Virtual Lightbox", wurde als virtuelle Welt entworfen.
- Entwurf und Realisation von "intelligenten Objekten mit Verhalten".

Was ist neu:

- Messageconcept war das erste echte Ereignismodell für 3D-Objekte.
- Das Ansprechen von Objekten und Objektgruppen über hierarchische Namensgebungen und Unterstützung von dynamisch erzeugten Objekten.
- Der generische Aufbau der Nachrichten (Allgemeines Messageconcept).
- Die Architektur der "Objekte mit Verhalten" (Vorbedingungen, Informationsspeicherung, eigentliche Aktion ausführen, nachfolgende Aktionen).
- Die Fähigkeit, daß diese Objekte in diesem Zusammenhang weitere Abfragen initiieren können.
- Die gleichzeitige Netzwerktransparenz der Nachrichten.
- Die hohe Flexibilität des Messageconcept selbst.

VORWORT

Die vorliegende Arbeit entstand im Rahmen meiner berufsbegleitenden Promotion am Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts für Informatik an der Eberhard-Karls-Universität Tübingen.

Ich möchte mich an dieser Stelle ganz herzlich bei meinem Mentor, Herrn Professor Straßer, bedanken, der mir als Leiter dieses Lehrstuhls die Gelegenheit bot, diese Arbeit anzufertigen.

Mein Dank gilt auch allen Mitarbeitern und Studenten des Arbeitsbereichs Graphisch-Interaktive Systeme für die gute Zusammenarbeit, die Antworten auf meine Fragen, anregende Diskussionen und dem kontroversen Disput. Besonderes erwähnen möchte ich Daniel Schick, welcher durch seine Diplomarbeit einen weiteren Baustein geliefert hat. Ferner Wolfgang Broll, mit dem ich konstruktiv im Bereich Multi-User-VRML zusammenarbeitete sowie Hans-Heino Ehricke, mit dem ich die ersten Schritte in den medizinischen Bereich unternahm. Besonderer Dank gilt L. Miguel Encarnação für die fruchtbare Zusammenarbeit und weil er es immer schaffte, die Deadlines einzuhalten, und Ralf Sonntag, der stets eine Antwort auf meine Fragen parat hatte und den ich als Freund schätzen lernte. Ich möchte auch Frau Ebert für die Korrekturlesung danken.

Special thanks to the Inventor team of Silicon Graphics, especially to Rikk Carey, Gavin Bell, David Mott, Alain Dumesny and Helga Thorvaldsdottir for their support.

And thanks to Bill Gates, chief of Microsoft, for releasing the new operating systems NT and Windows 2000. Which let me spend much time to build up networks and related projects.

Ferner der Firma Ott&Adis, die mir durch die flexible Arbeitszeitgestaltung diese weitere berufliche Qualifikation ermöglichte.

Besonderen Dank gilt auch meiner Mutter und meinen Geschwistern, die mir überhaupt erst die Chance gaben, soweit zu kommen.

Zuletzt richtet sich mein Dank an Gitte, die mich immer wieder antrieb und mir den nötigen Rückhalt gab. Und dem kleinen, süßen, zahnlosen Monster Merlin, der mir gegen Ende der Arbeit den Schlaf raubte.

Tübingen, im Juli 1999 Jürgen Fechter

INHALTSVERZEICHNIS

1 Einleitung

1.1 Motivation

Der Sehsinn des Menschen ist der primäre Sinn für Wahrnehmung. Die visuellen Areale der Großhirnrinde inklusive primärem visuellen Cortex nehmen einen großen Teil des Gehirns ein, und damit werden Leistungen wie das räumliche Sehen durch das Auswerten kleinster Querdisparitäten möglich. Die weitaus größten Informationen nehmen wir über den Lichtsinn auf. Die passive Aufnahme wird auf 10^7 bit/sec angenommen, gefolgt vom Gehör (10^6 bit/sec), dem Tastsinn ($4 \cdot 10^5$ bit/sec), Temperaturempfinden ($5 \cdot 10^3$ bit/sec), den Innenreizen (10^3 bit/sec), dem Geruchs- (20 bit/sec) und Geschmackssinn (13 bit/sec). Doch wird uns nur ein kleiner Teil davon (10^1 - 10^2 bit/sec) bewußt. Die Speicherkapazität des Gehirns wird auf 10^{14} bit geschätzt, unter der Prämisse, daß eine Synapse einem Ein-Aus-Schalter, also 1 bit, entspricht. Doch bereits einfachere Organismen, wie die Fliege *Musca domestica*, zeigen beachtliche Leistungen, was die Orientierung im Raum angeht. Die Kybernetik versucht durch einfache Modellvorstellungen dieses Verhalten nachzubilden. Der Mensch ist vor-programmiert, räumliche Zusammenhänge schnell zu erfassen und kognitiv zu verarbeiten. Aufgrund dieser Fakten ist leicht ersichtlich, warum der Bildschirm als visuelles Ausgabegerät prädestiniert ist, denn er bietet viele Informationen parallel an, welche schnell erfaßt werden können. Allerdings nutzen heutige Oberflächen nicht die Dreidimensionalität, sondern bewegen sich nur in der 2D-Ebene. Forschungen mit computergenerierten 3D-Welten, im folgenden Virtuelle Realität (VR) genannt, sind schon seit den letzten Jahren intensiver Gegenstand der Forschung. Trotzdem blieb bis jetzt ein durchschlagender Erfolg aus, und es stellt sich die Frage, welche Anforderungen gestellt werden, was die Ursache für den ausbleibenden Erfolg sein könnte und welche notwendigen Verbesserungen daraus resultieren. Zu diesem Zweck wurde eine Virtuelle Umgebung für Ärzte zum speziellen Umgang mit Röntgenbildern sowie ein allgemeines Nachrichtenkonzept für 3D-Objekte entwickelt.

1.2 Aufbau der Arbeit

Virtuelle Realität kann nicht in einem Gebiet zusammengefaßt werden, sondern sie setzt sich aus vielen Teilgebieten zusammen. Diese mußten analysiert werden und führten letztlich zu der vorgestellten Lösung. Neben der Definition der Begriffe wird auf die Interaktionstechniken, Benutzungsoberflächen (engl. Graphical User Interfaces kurz: GUI), Eingabegeräte, Ereignismodelle, Netzwerktechniken inkl. Protokolle, CSCW (engl. Computer Supported Cooperative Work) sowie spezielle Sprachen (z. B. VRML 2.0) eingegangen. Ferner soll kurz der Rahmen der möglichen VR Anwendungen auf dem Medizinsektor erörtert werden, da der Autor die Meinung vertritt, daß nur speziell zugeschnittene VR-Programme eine Erfolgsaussicht, sprich Akzeptanz, haben.

2 Stand der Forschung

2.1 Begriffsdefinition

Was ist Virtuelle Realität? Es ist schwierig, eine genaue Definition zu geben, aber allgemein wird darunter verstanden, daß mit Hilfe des Computers der Eindruck von 3D-Umgebungen erzeugt wird, die Objekte mit räumlicher Präsenz enthalten. Die Interaktion des Benutzers erfolgt mit Objekten und nicht, wie bei 2D, mit Bildern von Objekten. Trotz der häufigen Wandlungen, welche der Begriff VR erfährt, sollen repräsentative und akzeptierte Definitionen vorgestellt werden:

The creation of the effect of immersion in a computer-generated three-dimensional environment in which objects have spatial presence. [BF93]

Steve Bryson
Steve Feiner

Virtual Reality is ***the place where humans and computer make contact.*** [CC93]

Ken Pimentel
Kevin Teixeira

Virtual Reality refers to ***immersive, interactive, multi-sensory, viewer-centered, three-dimensional computer generated environments*** *and the combination of* ***technologies*** *required to build these environments.* [CC93]

Carolina Cruz-Neira

Neben dem vor allem im Journalismus häufig benutzten Begriff "Virtual Reality" findet sich in Veröffentlichungen "Virtual Environment", "Synthetic Environment" oder "Virtual World". Da es keine eindeutige Definition dieser Begriffe gibt, werden sie in dieser Arbeit als Synonyme gebraucht.

Warum Virtuelle Realität?

- Bessere Beurteilungen/Schlußfolgerungen aus 3D-Strukturen.
- Überlegene 3D-Interaktion und Kontrolle.
- Bessere Einbindung/Eintauchen in die Umgebung.

- Hohe Informationsdichte.

Wie wird die Virtuelle Realität erreicht?

- Eintauchen

  Darstellung der Umgebung durch Headtracker oder Head-Mounted Display.
- Selbstgegenwärtigkeit

  Großes Sichtfeld durch Bewegung in der Umgebung und Blickwechsel.
- Objektgegenwart

  Gute 3D-Wahrnehmung und Interaktivität mittels Stereo, Head-tracking, schnelles Anwortverhalten.

Schlüsselparameter:

- Hohe Rechnerleistung, um das Eintauchen, die Gegenwärtigkeit und Interaktion zu erreichen.

  durchgehend hohe Bildrate >= 10 Bilder/sec

  alle entstehenden Verzögerungen < 0.1 sec
- Menschliche Faktoren

  neue Interaktions-Techniken/Paradigmas

  Benutzerakzeptanz

Wie kann nun eine Virtuelle Umgebung definiert werden? Nach Ellis [ES94, ES97] besteht die Umgebung aus den folgenden drei abstrakt funktionalen Komponenten Inhalt, Geometrie, und Dynamik:

- **Content**:

  The *objects* and *actors* in the environment are its content. This objects may be described by *vectors* which identify their position, orientation, velocity, and acceleration ... characteristics such color, texture, and energy. This vector is thus a description of the *properties* of the objects. (...The actors) have capacities to initiale interactions with other objects.
- **Geometry:**

  The geometry is a description of the environmental field of action. It has *dimensionality, metrics*, and *extent*.
- **Dynamics**

  The dynamics of an environment are the *rules of interaction* among it contents describing their behaviour as they exchange energy or information.

Die von uns verwendete Sichtweise einer Virtuellen Umgebung nutzt ebenfalls die vorgestellten Komponenten, vereinigt sie jedoch innerhalb eines Objektes. Das heißt konkret, daß ein Objekt ein Aussehen und ein Verhalten haben kann. Wir sprechen in diesem Zusammenhang von einem ***"Objekt mit Verhalten"***. Ein solches Objekt kann eine Geometrie haben, muß aber nicht. Zur stärkeren Begriffsabgrenzung sei noch darauf hingewiesen, daß ein Verhalten nicht notwendigerweise eine Änderung des Objekts selbst oder seiner Position oder Orientierung im Raum zur Folge hat. Es genügt, wenn sich die inneren Zustände des Objekts ändern. Folglich muß ein Objekt mit Verhalten auch nicht sichtbar sein. Auf der anderen Seite gibt es Objekte in der Szene, die statisch sind und kein Verhaltensrepertoire zeigen. Es handelt sich also um Objekte ohne Verhalten. Der Kern unseres Modells sieht vor, daß die Objekte Botschaften austauschen können. Dieser Nachrichtenaustausch kann auch über ein Netzwerk als Medium erfolgen. Diese Integration von Objektverhalten und Nachrichtenaustausch in Virtuellen Umgebungen wird im folgenden durch den Begriff ***Messageconcept*** zusammengefaßt.

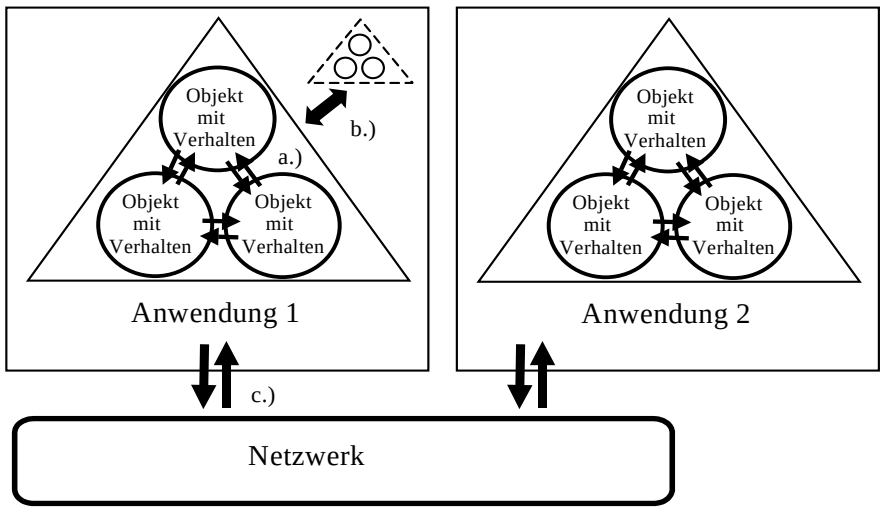


Abbildung 1: Schema Austausch von Nachrichten
a.) Austausch von Nachrichten direkt zwischen den Objekten innerhalb einer Gruppe/Szene.
b.) Nachrichtenaustausch zwischen verschiedenen Gruppen/Szenen.
c.) Netzwerkweiter Austausch von Nachrichten zwischen Applikationen.

Die Abbildung 1 zeigt die möglichen Fälle des Nachrichtenaustausches zwischen Objekten. Ein Objekt mit Verhalten kann eine Nachricht direkt zu einem anderen Objekt senden. Dabei muß das Objekt Mitglied der gleichen Szene bzw. des gleichen Baumes sein. Die Zugehörigkeit

zu solch einer Gruppierung wird durch das Dreieck symbolisiert. Im zweiten Fall findet der Nachrichtenaustausch zwischen verschiedenen Szenen einer Applikation statt. Im dritten Fall schließlich erfolgt der Informationsaustausch über ein Netzwerk. Nicht nur Objekte mit Verhalten können Nachrichten generieren, sondern auch Eingabegeräte. Analog könnten Ausgabegeräte, wie Force-Feedback-Geräte, Nachrichten entgegennehmen. Während die Objekte mit Verhalten einen einheitlichen Nachrichtenaufbau haben, ist dies bei Aus- und Eingabegeräten sehr geräteabhängig. Der Nachrichtenverkehr beschränkt sich im ersten Fall auf die Szene und das Netzwerk, während bei Ein- und Ausgabegeräten auch serielle, parallele oder andere Ports beteiligt sind. Erwähnt sei an dieser Stelle, daß manche Eingabegeräte wie Tracker ihre Informationen auf Netzwerkebene versenden. Auf den genauen Aufbau von Nachrichten wird in Kapitel 8.4.6 noch näher eingegangen.

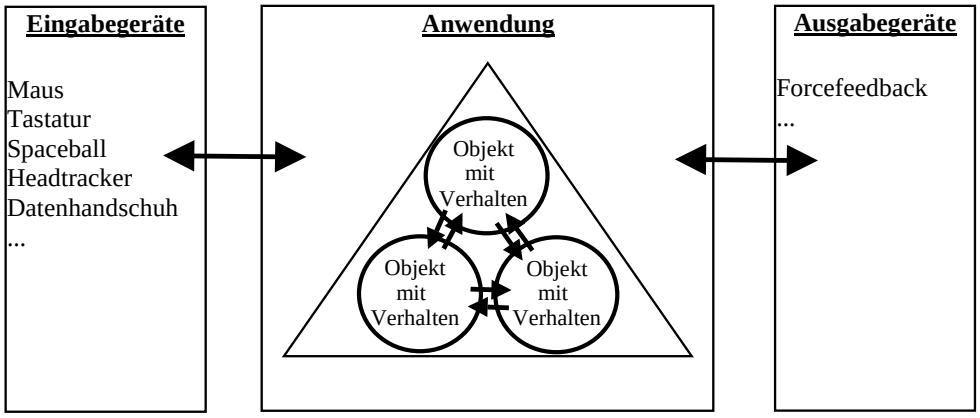


Abbildung 2: Nachrichtenverkehr zwischen Ein- und Ausgabegeräten mit einer Anwendung.

2.2 Präsentation

Die Objekte einer Anwendung können verschieden präsentiert werden. Neben der hier verwendeten 3D-Darstellung gibt es alternative Ansätze, die diese 3D-Darstellung ergänzen beziehungsweise ersetzen können. Einige Ansätze sollen exemplarisch vorgestellt werden.

2.2.1 Natürliche Benutzungsoberfläche

Unter einer natürlichen Benutzungsoberfläche versteht man eine Oberfläche, die einen hohen Wiedererkennungswert hat und eine schnelle Funktionszuordnung zuläßt. Ziel ist die einfache,

intuitive und natürliche Bedienung. Ein Beispiel hierfür ist die Szene eines Schreibtisches mit allen notwendigen Utensilien wie Kalender, Faxgerät, Schreibmaschine, Adreßbuch und Briefkasten. Der Vorteil liegt in der einfachen Erkennung der Dinge, die zur Bewältigung der Arbeit gebraucht werden, da diese Informationen aus dem Alltag schon kognitiv vorliegen. Es ist keine Suche nach irgendwelchen abstrakten Elementen notwendig. Untersuchungen zeigen, daß die mittlere Zugriffsgeschwindigkeit sowohl beim Erst- als auch beim Folgezugriff signifikant kürzer ist [SM97]. Im Gegensatz zu normaler traditioneller Software ist die Oberfläche aufgaben- und nicht programmorientiert aufgebaut. Durch die gegenständliche Visualisierung kommt dieser Oberflächentyp bei Neuanfängern, Kindern und Älteren zum Einsatz und sowie bei PDAs (Persönliche Digitale Assistenten). Ende 1994 führte General Magic ihr kommerzielles Produkt Magic Cap ein, welches die Szene eines einfachen Schreibtischs zeigte. Andere, wie die Oberfläche BOB von Microsoft, bauten auf dieser Arbeit auf und erweiterten das Konzept.

Probleme von normalen Icon-basierten Standardoberflächen:

- Benutzungsoberfläche ist überladen. Der Benutzer braucht die meisten Programme nicht.
- Benutzungsoberfläche ist passiv
- Merkmale/Icons sind versteckt
- Mehrere Programme zusammen werden für eine Aufgabe benötigt (Tabellenkalkulation, Datenbank, Textverarbeitung, Grafikprogramm)
- Terminologie ist oft zu technokratisch

Vorteile einer bildlichen Szenendarstellung gegenüber einer Icon-basierten Standardoberfläche:

- Elemente einfacher zu identifizieren
- Durch die räumliche Anordnung und durch Orientierungsmarken besseres Einprägen der Lage der Elemente, schnellerer Gedächtnisaufruf und Zugriff auf Elemente
- Bessere Gruppierung der verschiedenen Aufgaben innerhalb eines Kontextes und einfachere Bedeutungszuordnung

2.2.2 Soziale Benutzungsoberfläche

1994 definierte Pattie Maes den Begriff "Soziale Benutzungsoberfläche" folgendermaßen: " ... [C]omputer programs that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular application ...The metaphor is that of a personal assistant who is collaborating with the user in the same work environment." [MP94]. Die erste kommerzielle Umsetzung fand 1995 durch die von Clifford Nass und Byron Reevers entwickelte Oberfläche BOB für Microsoft Windows [MS95] statt und wurde speziell für den Heimbereich ausgelegt. Die Arbeit baut auf den natürlichen Oberflächen auf und wird durch soziale Oberflächenkomponenten ergänzt. Der Benutzer soll freundlich mit dem Computer kommunizieren, dabei soll die Interaktion mehr der von Mensch zu Mensch gleichen als von Mensch zu Maschine. Somit gibt es keine Handbücher oder Tutorials, kein umständliches Doppelklicken und keine technokratischen Ausdrücke bzw. "fachchinesische" Erklärungen. Statt dessen stehen Assistenten in vielerlei Gestalten und unterschiedlichen Charakteren zur Verfügung, die den Benutzer durch das Programm begleiten[1]. Diese Assistenten überwachen die Benutzeraktionen und geben bei Bedarf Hilfen und Tips. Andere soziale Aktionen wären u. a. das Begrüßen des Benutzers am Start oder die Verabschiedung am Ende. Kern von BOB

Abbildung 3: Soziale Benutzungsoberfläche BOB

sind 2D-Darstellungen von Räumen, die öffentlich oder nur privat zugänglich sind. BOB

[1] Diese Assistenten kommen auch in der Office97 Version zum Einsatz.

ermöglicht die persönliche Gestaltung dieser Räume mit Elementen, die keine Bedeutung haben, wie Tapeten, Vorhänge und Möbel, aber auch die aufgabentragenden Elemente können individuell angeordnet werden. Dies ermöglicht das grobe Abbild des eigenen Heims oder einer Wunschvorstellung von einem Raum im Computer. Ziel ist, daß sich der Benutzer gerne in seiner Umgebung aufhält; sich emotional wohl fühlt. Bei den Programmen von BOB handelt es sich um Kalender, Notizbuch (To-Do Liste), Checklisten, Adreßbuch, Textverarbeitung, Haushaltsmanager, Spiele, Finanzen und optional Emailprogramm. Diese Programme sind in ihrem Funktionsumfang, im Gegensatz zu den mit dem Betriebssystem Windows mitgelieferten, stark eingeschränkt und haben ansonsten das Erscheinungsbild von normalen fensterbasierten Applikationen.

2.2.3 Oberflächen für abstrakte Datenvisualisierung

2.2.3.1 3D-Informationsräume

Um noch mehr Information den Benutzern zugänglich zu machen, kann die dritte Dimension genutzt werden. Solch ein 3D-Informationsraum aus realen oder künstlichen Gebilden stellen aber weitere Anforderungen an die Benutzer. Einerseits muß die einfache Navigation und Orientierung möglich sein, das heißt, der Benutzer muß zu jeder Zeit wissen, wo er sich befindet. Daraus folgt, daß eine Szene nicht zu komplex werden darf. Dies könnte dadurch erreicht werden, daß die Teile des Raumes, die den Benutzer nicht interessieren, nur grob und detailarm, die interessierenden Teile aber detailliert dargestellt werden. Dies bedingt aber, daß die Darstellung unterschiedlicher Detaillierungsgrade von der Intention des Benutzers und somit dem Kontext der Gesamtaktion abhängen. Zur Orientierung des Benutzers zwischen verschiedenen Sichten sollten deren Übergänge inkrementell und dadurch nachvollziehbar sein. Eventuell können Techniken wie Zoom, Weitwinkel oder Fisheye zum Überblick oder Teilausschnitt eingesetzt werden. Texteinblendungen, wie Hilfetexte oder Anweisungen sollen die bisherige Sicht nicht verdecken. Sie sollten deshalb semitransparent und temporär erfolgen. Damit der Benutzer schnell die räumlichen Zusammenhänge einer Szene erfassen kann, wird die a-priori Kenntnis des Benutzers aufgebaut. Aus Erfahrung weiß er beispielsweise, daß durch Drücken des Schalters das Licht an- oder ausgeht.

2.2.3.2 3D-Oberflächen für virtuelle Landschaften

1992 stellte Silicon Graphics ihr 3D-Dateisystem Navigator (FSN-File System Navigator©) vor. Für die Visualisierung wird eine Landschaft mit 3D-Säulen in der Fischaugenperspektive verwendet, bei der auf einen Blick die Verzeichnishierarchie, die Dateien und Verzeichnisse zu sehen sind. Das Basis-Visualisierungsobjekt ist eine Zelle (Verzeichnis), welche Blöcke in Form von Säulen (Dateien) enthält. Verbindungslinien zwischen den Zellen repräsentieren die Verzeichnishierachie. Die Blöcke wiederum sind folgendermaßen definiert: die Farbe gibt das Alter der Datei an, die Höhe die Volumengröße, ein Bild den Typ, sowie ein eingeblendeter Text den Namen. Durch diese künstliche Landschaft kann der Benutzer hindurchfliegen und Dokumente auswählen. Intuitiv erhält der Benutzer einen Überblick über die Dateistruktur, da diese Präsentation seine natürlichen, kognitiven Fähigkeiten für die Erfassung einer Landschaft

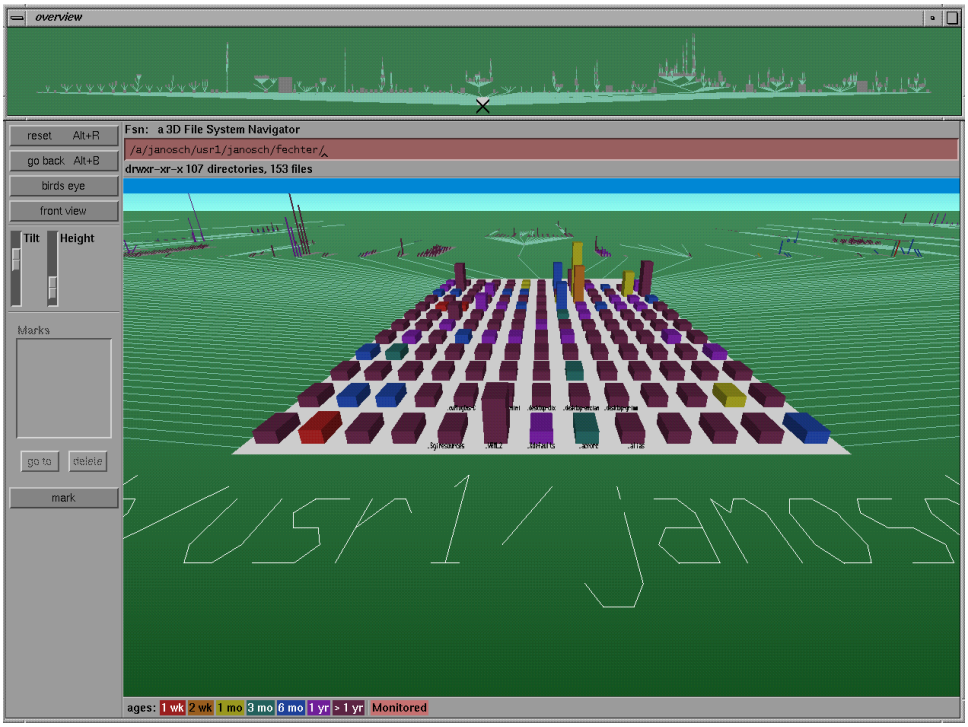


Abbildung 4: Silicon Graphics FSN-File System Navigator

ausnutzt. Dieses Konzept der Visualisierung von Information in Form einer Landschaft eignet sich für jede Art von hierarchischen Datenstrukturen.

2.2.3.3 3D-Oberflächen für Dokumentensuche

Betrachtet man die Evolution von Textdateien, so entwickelten sich aus den reinen Textdokumenten Verbunddokumente mit Tabellen und Grafiken, dann Hypertexte mit Multimediaelementen, und es bleibt abzuwarten, wann die Texte selbst 3D-Objekte enthalten. Auf der anderen Seite steigt die Anzahl der Dokumente kontinuierlich. Die Frage ist, wie man in dieser riesigen Informationsdatenmenge die richtige Information findet. Folglich wird nach neuen Wegen für die Darstellung dieser Information gesucht, inklusive der Teilgebiete Navigation, Herausfiltern und Zugriff auf die Information. Das Hauptproblem liegt in der niedrigen Auflösung des Monitors begründet, wodurch nicht genügend Platz für die gleichzeitige Unterbringung aller Informationen vorhanden ist. In einer wegweisenden Arbeit stellte 1991 die Xerox Parc Gruppe neue Möglichkeiten für die Visualisierung von Dokumenten vor [MR91, RM91]. In einer künstlichen Welt, genauer in 3D-Räumen, wurden neue 3D-Visualisierungstechniken wie die der Informationswand (engl. Perspective Wall) für lineare und die des Kegelbaums (engl. Cone Tree) für hierarchische Daten verwendet. Beide Techniken benutzen interaktive Animationen und der Benutzer kann seine Position wechseln, um eine andere, sich dynamisch ändernde Sicht auf die Informationsstruktur zu bekommen. Ziel ist, die Informationen schneller und effektiver zu durchforsten und aus der Informationsflut die relevanten Informationen zu filtern. Die in einer normalen Anwendung sonst erforderlichen Abfragen werden durch die Navigation ersetzt. Damit dieses Konzept funktioniert, muß die Information (engl. information space) in kontext-orientierte Gruppen eingeteilt werden. Bei der Informationswand wird diese horizontal chronologisch nach Erstellungsdatum oder der letzten Veränderung geordnet. Vertikal erfolgt die Gruppierung nach Dokumentenart und Autor. Ein ausgewähltes Dokument wird durch langsame Animation ins Zentrum bewegt. Der Benutzer kann ferner die Einstellung der Detailgenauigkeit und des gewünschten Kontextumfeldes vornehmen. Beim Kegelbaum wird die Information in dreidimensionale, hierarchische Strukturen in der Art eines Baumes aufgespalten. Auf die Probleme bei der automatischen Generierung der Bäume selbst, wie die allgemein schwierige Klassifikation der Dokumente sowie die Redundanz von Einträgen und den möglichen Verlust von Verbindungen zu ähnlichen Dokumenten, soll nicht weiter eingegangen werden. Der Benutzer kann einen Teilbaum manuell rotieren lassen, um einen Eintrag in den Vordergrund zu bringen. Die einzelnen Elemente der Teilbäume sind halbtransparent, damit auch die dahinterliegenden Elemente wahrgenommen werden können. Damit der Benutzer die Hierachieebenen besser erkennen kann, wird ein Schatten auf die Wände und den Boden des Informationsraums vor den Bäumen projiziert. Eine Weiterentwicklung des Kegelbaum-Modells, basierend auf Inventor, entwickelte die GMD mit LyberWorld [HK94]. Hier gibt es

zwei Arten von Walzen, die sich abwechseln. Aus den roten Walzen mit Dokumenttiteln wachsen bei näherem Interesse für einzelne Titel blaue Kegel, die mit dem Dokument verknüpfte Begriffe anbieten. Ein Klick auf diese Begriffe läßt wiederum einen neuen roten Kegel erscheinen, eben die Dokumente, die diesen Begriff enthalten. Bereits in übergeordneten Kegeln enthaltene Fundstellen sind orange markiert.

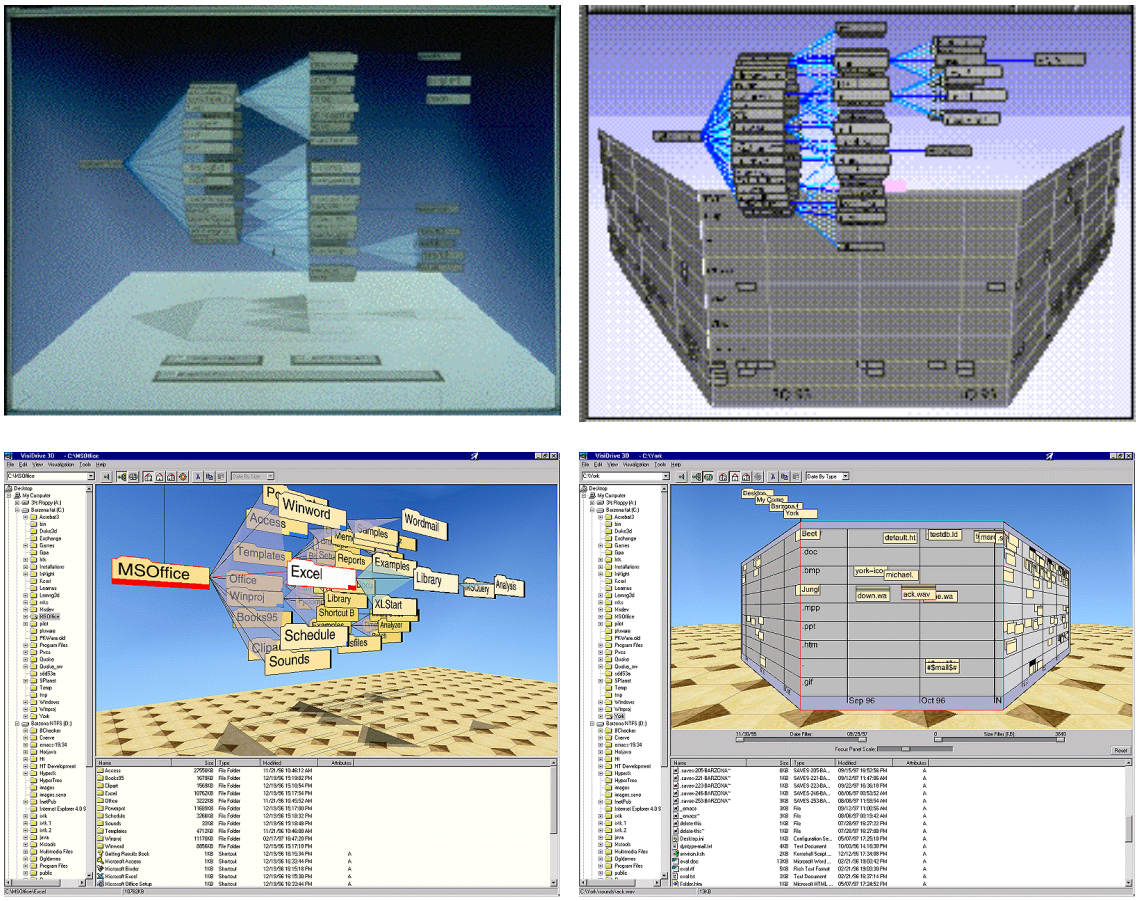


Abbildung 5: Kegelbaum und Pespektivische Wand. Unten die modernere ins Windows-System integrierte Version

Ein weiteres Element, das für 3D-Räume entwickelt wurde, ist das Rondell, welches nach dem Vorbild von drehbaren Kartenständern entstand und der bekannten Litfaßsäule ähnelt. Es entstand im Rahmen einer Diplomarbeit von Ressel [RS95]. Das Rondell eignet sich im Gegensatz zu den Kegelbäumen zur Visualisierung von nicht-hierarchischen Informationen. Es besteht aus oberer und unterer Scheibe und dazwischen ist ein Prismenkörper. Auf den Prismenflächen befindet sich eine Auswahl von Einträgen. Ein Klick auf die obere Scheibe bewirkt ein Drehen um einen Schritt nach rechts und auf die untere einen Schritt nach links.

Dabei wird die Vorderseite parallel zum Betrachter ausgerichtet. Zur besseren Unterscheidung der beiden Drehbewegungen wird ein Pfeil in Rotationsrichtung eingeblendet. Ein Doppelklick führt zur langsamen 360º Rotation, um einen Überblick über die gesamte Informationsmenge zu erhalten. Insgesamt eignet sich das Rondell nur für die Selektion von kleineren Datenbeständen mit maximal zehn Spalten die höchstens 12 Einträge enthalten. Für größere Datenbestände müßte eine Selektion zur Generierung eines weiteren temporären Unterrondells mit der Auflistung der Alternativen führen [PB98].

2.2.3.4 3D-Oberflächen für Datenbanken

Drei Gesichtspunkte müssen bei 3D-Datenbank-Oberflächen berücksichtigt werden [BG95]:

- Die angezeigte Information muß verständlich sein.
- Die Beziehung zwischen den angezeigten Objekten und den Daten, die sie eigentlich repräsentieren, müssen dem Benutzer verständlich sein.
- Der Benutzer sollte möglichst einfach zwischen der speziell gewählten Sicht (Abfrage) und einem allgemeinen Überblick über die Daten umschalten können.

Grundsätzlich können hier zwei Bereiche der Visualisierung unterschieden werden, die man in 3D darstellen kann: die Abfrage selbst und das Ergebnis. Beides wurde schon in Form von künstlichen Landschaften prototypenhaft visualisiert [GC97]. Das Extrahieren von Information aus einer Datenbank durch einfaches Navigieren durch den Benutzer bietet den Vorteil, daß keine Kenntnisse über die Datenabfragesprache selbst nötig sind. Kann das Ergebnis in Form von 3D-Objekten visualisiert werden, so erleichtert dies das Verständnis über die Objekte. Gerade die Bereiche Chemie, Medizin, Statistik, Mathematik und Physik scheinen dazu am geeignetsten zu sein. Es können dabei selbstreplizierende Skripten eingesetzt werden, das heißt, es wird eine neue Abfrage ausgeführt, indem auf die nächste Informationsuntermenge zugegriffen wird. Der Benutzer ‘surft‘ sozusagen durch die Abfragen. Im einfachsten Falle sind Abfrage- und Ergebnisraum in der gleichen Szene visualisierbar. Bildlich könnte zum Beispiel ein Gebäude dargestellt werden, das durch die Navigation mittels Abfragen immer feiner aufgelöst wird. Zuerst in die Bestandteile Gebäudeflügel, Stockwerk, Abteilung und Zimmer, bis schließlich als Ergebnis die darin arbeitenden Leute abgebildet werden.

2.2.3.5 Bewertung

Im Zuge dieser Arbeit wurde der vorgestellte Kegelbaum in eine Szene integriert. Dabei sollten die Teilbäume die Patienten mit den zugehörigen Patientendokumenten/Röntgenbildern

repräsentieren. Die Vermischung von realen, bekannten Objekten mit derart künstlichen Gebilden erwies sich für die Benutzer als so störend, daß auf eine Integration verzichtet wurde. Die interaktive Arbeit mit rotierenden Kegelbäumen stellt ferner hohe Hardwareansprüche an die OpenGL Graphikkarte und führte ebenfalls zur Verwerfung dieses Plans.

2.3 3D-Interaktion

Eine Virtuelle Umgebung ist eine Ansammlung von computergenerierten Objekten, welche mit dem Benutzer oder anderen Objekten interagieren. Speziell in Virtuellen Umgebungen müssen zwei Eigenschaften des Objekts definiert werden: Das Aussehen und sein Verhalten. Sobald die Objekte vom Computer erzeugt worden sind, sorgt ein Code für das Verhalten, indem er die grafische Repräsentation des Objekts ändert. Im allgemeinen werden die auszuführenden Funktionen Änderungen an den dazugehörenden Daten des Objekts vornehmen. Folgende Punkte müssen dabei festgelegt werden:

- Welche Objekte mit welchen Objekten interagieren können.
- Spezifikation der Funktionen für diese Verhalten.
- Änderung der betroffenen Objektdaten.

Mit der Anzahl der Objekte wächst die Menge der möglichen Interaktionen und Wechselbeziehungen stark an und somit auch die Komplexität der Datenpfade. Ebenso nimmt der Aufwand für die Darstellung der Zustände der Objekte sowie die Schwierigkeit einer Vorhersage, welche Objekte von der Interaktion betroffen werden, zu.

2.4 Techniken

2.4.1 Datenfluß-Paradigma

Beim Datenfluß-Paradigma wird eine Pipeline gemäß dem Datenfluß abgebildet, bei der die Komponenten durch eine eindeutige Beziehung miteinander verknüpfbar sind. Deshalb haben die Komponenten für den Datentransfer mindestens einen Port und können in folgende drei Kategorien eingeordnet werden: Die Quelle, auch Erzeuger oder engl. producer genannt, besitzt nur Ausgänge. Ein Eingabegerät wie Spaceball bzw. eine Datei oder Videoquelle erfüllt diese Anforderung. Der Empfänger (engl. consumer, sink) hat analog dazu nur Eingänge. Ein Stellvertreter hierfür wäre beispielsweise ein 3D-Renderer. Folglich gibt es noch einen Umwandler (engl. transformer) mit mindestens einem Ein- bzw. Ausgang. Eine Filterkomponente, ein Modell oder ein Navigator gehört zu dieser Gruppe. Grundsätzlich sind auch Rückkopplungen erlaubt. Der Datenfluß-Baum, wie auch das dynamische Modell, kann

während der Laufzeit geändert werden und das Ergebnis ist sofort sichtbar (z.B. ein Objekt bekommt die Eigenschaft Gewicht und fängt sofort an zu fallen). Die Modellierung des Datenfluß-Ablaufs erfolgt häufig visuell durch ein Werkzeug. Neuere Vertreter für das Datenfluß-Modell sind SUNs JavaBeans oder AVS-Express. Auch bei Multimedia-anwendungen ist die Datenfluß-Sichtweise eine geeignete Darstellungsform. Verschiedene Audio/Videoquellen werden aufbereitet, gemixt und auf unterschiedlichen Ausgabegeräten abgelegt. Als Kontrollelemente gibt es die bekannten Bedienelemente wie Wiedergabe-, Aufnahme- oder Stop-Tasten.
Der erste Vertreter im 3D-Bereich, welcher das Datenfluß-Modell anwandte, war die Firma VPL-Research mit ihrem kommerziellen VR-Produkt RB2 (Reality Built For Two) [BB90]. Hier wurde erstmalig eine Interaktion als Datenpfad einer Bewegung oder Manipulation von Daten aufgefaßt.

Hat man hingegen 3D-Anwendungen, bei denen viele Interaktionen zwischen den unterschiedlichen Objekten vorkommen, so ist die Modellierung des Datenflusses kompliziert und undurchsichtig.

2.4.2 Interaktion durch das Zwei-Punkte-Paradigma

Im Jahre 1991 wurde das Modell des Zwei-Punkte-Paradigma von Steve Bryson vorgeschlagen [BS91]. Es erlaubt dem Benutzer die Ansicht, die Änderung und das Hinzufügen von Interaktionen in Echtzeit. Das Modell kommt aus der physikalischen Welt und versteht eine Interaktion als Summe der einwirkenden Einzelkräfte (Vektoren) auf ein Objekt durch jedes andere Objekt. Beim Zwei-Punkte-Paradigma werden im ersten Schritt die Interaktionen zwischen einem Paar betrachtet. Es werden somit die möglichen Interaktionen in einer Virtuellen Umgebung durch jeweils eine Liste von Interaktionen für jedes Paar von Objekten beschrieben. Diese Liste wird Interaktionsmatrix genannt. Wie aus der Matrix zu entnehmen ist, wird unterschieden, ob Objekt A mit Objekt B interagiert oder Objekt B mit Objekt A. Bei den diagonalen Einträgen interagiert das Objekt mit sich selbst. Hier sind nach der Konvention die Funktionen zu berücksichtigen, welche das Objekt selbst aktualisieren.

<table>
<tr><th></th><th>Objekt A
Boden</th><th>Objekt B
Ball</th></tr>
<tr><td rowspan="2">Objekt A
Boden</td><td rowspan="2">NULL</td><td>Schwerkraft</td></tr>
<tr><td>Rückprall</td></tr>
<tr><td>Objekt B
Ball</td><td>NULL</td><td>Einbeziehung
der</td></tr>
</table>

		Bewegungsgleichung

Tabelle 1: Beispiel einer Interaktionsmatrix mit den zwei Objekten Ball und Boden. Der Ball interagiert mit dem Boden in zweierlei Weise: Die Schwerkraft zieht ihn nach unten und er prallt zurück mit der umgedrehten z-Komponente der Geschwindigkeit. Der Boden interagiert nicht mit dem Ball.

Ein Problem dieses Ansatzes liegt darin, daß die Reihenfolge der Abarbeitung der Interaktionen zu unterschiedlichen Ergebnissen führen kann. Angenommen, das Objekt A wird bewegt und eine zweite Interaktion bewirkt, daß Objekt A ein anderes Objekt B bewegt, dann hängt die Position des Objekts B entscheidend von der Reihenfolge ab, in der die Position berechnet wurde. Ferner, wenn die Interaktionen zu einem nicht linearen Zuwachs führen und somit die Addition der Kräfte nicht kommutativ ist, ist die Reihenfolge der Berechnungen maßgebend. Da ein Objekt durch mehrere Interaktionen innerhalb eines Zeitschrittes beeinflußt werden kann, muß bei der Berechnung auf die alten Werte zum Zeitpunkt t_0 zugegriffen werden. Folglich fordert das Zwei-Punkte-Paradigma, daß die Parameter eines Objekts dupliziert werden, um die zwei Zustände t_0 und t_1 zu halten. Für die Implementation wird eine globale Variable *buffer* benötigt, welche die Zustände 0 und 1 annehmen kann, und bei jeder Abarbeitung der Interaktionsmatrix wird der Wert getauscht.

Notation:

Nach Bryson ist eine Interaktion eine Datenstruktur, die aus mehreren Komponenten besteht. Zuerst die Funktion, die Interaktion selbst, welche als Eingabe ein geordnetes Paar von Objektdaten hat und nach Konvention auf die Daten des zweiten Objektes des Paares wirkt.

Interaction: (O1, O2) ->O2

O1 und O2 stellen einen Verweis auf die Objektdatenstruktur dar. Der Pfeil zeigt an, daß die Daten durch die Interaktion verändert werden. Welche Objektdaten für die Interaktion benötigt werden, wird in der Interaktionsfunktion selbst festgelegt.

Eine einfache Interaktionsfunktion als Zwei-Punkte-Paradigma für eine Echtzeitanwendung hätte folgendes Aussehen in C-Syntax:

```
Interaction(object* O1[2], object* O2[2])
{
O2[(buffer+1)mod2]->position=O1[buffer]->position+O2[buffer]->velocity;
}
```

Von jedem Objekt gibt es zwei vollständige Kopien. Der Zugriff erfolgt mittels der Variable buffer, welche nach Abarbeitung der Interaktionmatrix alternierend die Werte 0 und 1 annimmt.

Neben der oben beschriebenen Einschränkung bleibt noch anzumerken, daß dieses Verfahren nur dann sinnvoll ist, wenn die Interaktion zwischen den Objekten paarweise beschreibbar ist

und nicht mehrere gleichzeitig interagieren. Zu Problemen, wie sie durch das Löschen von Interaktionen hervorgerufen werden, gibt es keine Aussagen.

2.4.3 Objektorientiertes Paradigma

Eine Interaktion als Objekt zu betrachten, welches ein anderes Objekt verändert, entspricht der Weiterentwicklung des Zwei-Punkte-Paradigma. Grundsätzlich gibt es hier zwei Denkansätze: Zum einen ist eine Interaktion ein eigenständiges Objekt, welches anhand der Objekte, die die Änderungen erfahren sollen, weiß, welche Aktion ausgeführt werden soll. Die Interaktion ist vom Objekt losgelöst. Zum anderen kann eine Interaktion als eine Funktion des Objekts betrachtet werden, die Interaktion ist Teil des Objektes. Beim objektorientierten Ansatz werden die lokalen Zustände normalerweise vor der Umwelt versteckt. Folglich kann mit dem Objekt nur mittels Ereignissen (engl. Events) oder Nachrichten (engl. Messages) in Interaktion getreten werden.

2.5 Interaktions-Modelle

Basierend auf dem Konzept des "Interactor" von Faconti und Peterno [FP90] entwickelten Duke und Harrision [DH93] Modelle und die dazugehörende Theorie für Interaktive Systeme. Die formalen Modelle sind abstrakt und basieren auf der Z-Notation [SJ92]. Im Zusammenhang mit dieser Arbeit gibt es interessante Analogien und Betrachtungen. Ein Interaktionsobjekt wird als ein unabhängiges Objekt mit einem lokalen Zustand aufgefaßt, welches durch Ereignisse mit der Umwelt in Beziehung treten kann. Ereignisse werden klassifiziert in *Stimuli-Events*, die extern erzeugt werden und in ihr Gegenteil *Response-Events,* die vom Interaktionsobjekt (engl. *Interactions-Object*) generiert werden. Ein Interaktionsobjekt muß keinem physikalischen Gerät entsprechen. So ist beispielsweise das Pfeilzeichen einer Laufleiste ein Interaktionsobjekt mit den lokalen Zuständen aktiv und inaktiv. Ein Ereignis ist die Abstraktion einer Aktion oder Änderung innerhalb des Systems. Ein Interactor-Objekt ist ein komplettes Informationsprozeß-System, welches Stimuli empfängt, lokale Änderungen der Zustände vornimmt und Response-Events an andere Interaktionsobjekte, auch *Interactoren* genannt, verschickt. Ein interaktives System wird durch das Zusammensetzen von einzelnen Interaktionsobjekten gebildet. Änderungen von lokalen Zuständen des Interactors, welche der Benutzer wahrnehmen kann, werden als Präsentation bezeichnet. Im Grunde ist ein Interactor als Oberflächen-Komponente aufzufassen, welche zwischen dem Benutzer und der unterliegenden Applikation vermittelt. Zwei Architekturen für Graphische Benutzungs-oberflächen (engl. GUIs) welche mit Interactoren realisiert werden können, sind der Model-

View-Controller (abgekürzt: MVC) [KP88] und das Presentation-Abstraction-Control Modell (abgekürzt: PAC) [CJ87].

MVC:

Beim MVC-Modell läßt sich der Interactor in drei Subkomponenten zerlegen: das interne Modell, das Aussehen und die Kontrolleinheit. Die lokalen Zustände und applikationsspezifischen Funktionen werden im Modell abgebildet. Das Modell kann als Abstraktion des Oberflächenobjektes aufgefaßt werden, ohne das eigentliche Aussehen zu beinhalten, denn diese Information steckt in der Sichtkomponente. Die dritte, die Kontrolleinheit, ist für die Ein- und Ausgabe dieses Oberflächenobjekts verantwortlich. Jede dieser drei Subkomponenten ist wiederum als ein Objekt aufzufassen.

Funktionsablauf: Bei der Initialisierung ist normalerweise das Modell für die Erzeugung der eigenen Sicht und der Kontrolleinheit verantwortlich. Die Sicht- und Kontrolleinheit können in diesem Sinne als Kindsobjekte der Modellkomponente aufgefaßt werden. Die Kontrolleinheit erhält die Eingaben des Benutzers (z. B. Maus, Tastatur) und interpretiert sie. Falls eine Eingabe eine Änderung der applikationsspezifischen Daten zur Folge hat, benachrichtigt die

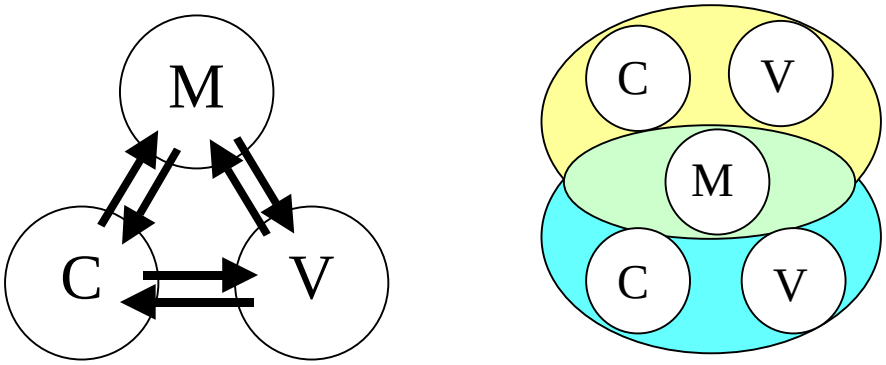


Abbildung 6:
a.) Beispiel für einen MVC-Interactor mit den möglichenNachrichtenaustauschwegen.
b.) Teilen einer Modellkomponente zwischen zwei Interactoren.

Kontrolleinheit das Modell (z. B. Erzeugen eines Popup-Menüs). Die Modellkomponente wiederum sendet bei Änderungen Nachrichten an alle abhängigen Sichteinheiten, damit diese angezeigt werden.

PAC:

Das PAC-Modell ähnelt dem des MVC, kommt aber aus dem Sprachbereich. Bei diesem sprachenbasierten Ansatz wird das Interaktive System in die Komponenten Präsentation,

Applikation und Dialogkontrolle eingeteilt, welche ursprünglich im lexikalischen, semantischen und syntaktischen Teil einer linguistischen Interaktion ihre Entsprechung finden. In der Präsentationskomponente steckt die Information über das Aussehen, während in der Applikationskomponente der Zustand enthalten ist. Nachrichten zwischen den PAC-Interactoren werden über einen Kommunikationskanal der Kontolleinheit ausgetauscht. Somit braucht die Kontrolleinheit auch Wissen über die Applikation und die Präsentations-komponenten. Da ein PAC Interactor keinen separaten Prozeß darstellt, ist volle gleichzeitige Aktion bzw. Interaktion nicht möglich. Allerdings kann ein PAC-Interactor weitere sogenannte Sub-Interactoren enthalten.

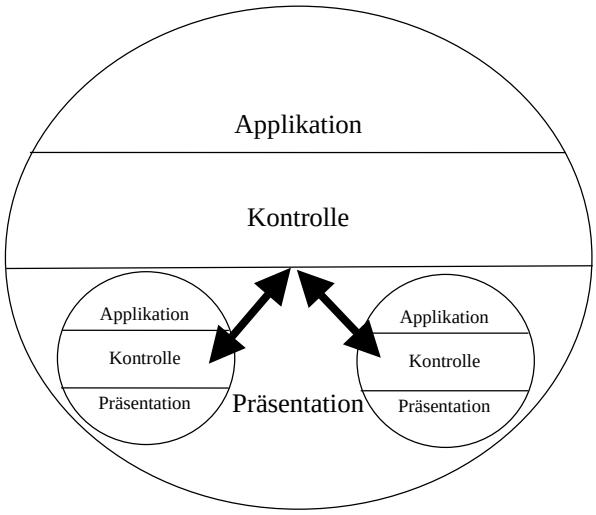


Abbildung 7: Beispiel für einen PAC-Interactor mit zwei Sub-Interactoren und den Benachrichtigungswegen.

2.6 Software

2.6.1 Allgemein

Zur Erstellung von VR-Applikationen müssen neben den Objekten die Interaktion und das Objektverhalten festgelegt werden. Durch die Komplexität der Applikationen werden diese nicht mehr rein durch Verwendung einer Programmiersprache und einer Graphikbeschrei-bungssprache wie Silicon Graphics *OpenGL* realisiert, sondern durch Verwendung spezieller

objektorientierter Bibliotheken. Bei kommerziellen Produkten wie World Toolkit [SE97] sind Treiber für verschiedene Eingabegeräte sowie Basisinteraktionen (z. B. Greifen) vorhanden. Die Funktionen werden durch ein API mittels einer Programmiersprache aufgerufen. Zusätzlich bieten einige VR-Tools eine integrierte Scriptsprache zur Modellierung des Objektverhaltens an. Historisch bedingt wird hier häufig die Programmiersprache C++ verwendet, die sich grundsätzlich portieren läßt. Zur Zeit bauen die meisten VR-Applikationen im Forschungsbereich auf Open Inventor oder VRML auf.

2.6.2 OpenGL

OpenGL ist einerseits eine Architektur und andererseits eine Graphikbeschreibungssprache, die auf viele andere Plattformen (Windows NT, Windows 98, HP-Unix, SUN) portiert wurde [KM97]. OpenGL wurde sehr hardwarenahe konzipiert und arbeitet intern nach dem sogenannten Statemachine Principle. Ein Nachteil des Einsatzes von OpenGL als Graphik-beschreibungsausgabe ist, daß es hohe Hardwareanforderungen verlangt, da der Kern der Renderengine (OpenGL-Machine) implementiert sein muß. Ferner gibt es optionale Erweiterungen, die über die reine Basisrenderingfunktionen hinausgehen [SA93]. Benutzt eine Applikation solch eine erweiterte Funktion, die nicht durch die Graphikhardware unterstützt wird, wird alles, das heißt die gesamte Renderengine[2], durch Software emuliert und nicht nur der Teil, der durch die Hardware nicht abgebildet wurde. Ein Beispiel hierfür wäre Transparenz (Alphablending), Texturemapping und bi- und trilineare Filterung (Mip-Mapping). Der Grund liegt in der notwendigen Rückkopplung. Aber es gibt auch eigenständigere Teile von OpenGL, die wahlweise in Hardware implementiert oder in Software emuliert werden können, ohne daß die gesamte Renderengine bei Gebrauch dieser Funktionen zurück in den Emulationsmodus gehen muß (Depth-Test). Daß softwarebasiertes OpenGL nicht unbedingt sehr langsam ist, bewies SGI durch seine eigene OpenGL Portierung (OpenGL for Windows Silicon, vormals CosmoGL im Gegensatz zu Microsofts OpenGL) auf NT. Um OpenGL attraktiver zu machen, wird es die neue Version 1.2 geben, welche einen verbesserten Funktionsumfang und höhere Geschwindigkeit unter Windows-Betriebssystemen bringen wird. Übergangsweise gibt es die *OpenGL Optimizer* Bibliothek zur schnelleren Darstellung und Bewältigung von großen Datenmengen und schnelleren Rückkopplung von Interaktionen.

Ferner wird angestrebt, unter Windows einheitliche graphische Programmierschnittstellen zu schaffen. Die Fahrenheitarchitektur soll aus drei Komponenten bestehen: Die Fahrenheit Low

[2] Eine eigenständige Regel kann hier nicht angegeben werden, denn dies ist sowohl von der eingesetzten Hardware wie auch von der Intelligenz der Treiber abhängig. Es gibt Fälle, bei denen es besser ist, aus Leistungsgründen ganz auf die Hardwarebeschleunigung zu verzichten und die gesamte OpenGL Pipeline in Software zu emulieren.

Level API soll die Schnittstelle für einfache geometrische Primitive und Darstellungen enthalten und bietet den Funktionsumfang von OpenGL und Direct3D. Die Fahrenheit Szenengraph API soll multiprozessorfähig sein und die Techniken aus den Silicon Graphics Produkten Performer, Open Inventor, Cosmo3D und OpenGL++ enthalten. In der dritten Komponente, Fahrenheit Large Model API, wird das Know-how von OpenGL Optimizer einfließen, um große Datenmengen zu visualisieren [HG98].

2.6.3 Java3D

SUNs junge Programmiersprache JAVA wäre zwar besser für eine Portierung geeignet, zumal sich auch gerade die Benutzungsoberfläche mitportieren ließe, hat aber folgende Schwächen: Die Bibliothek Java3D, verantwortlich für die mathematischen Transformationen, VR-Geräte-einbindung, Szenengraph und Visualisierung, wurde 1997 spezifiziert und erst im Dezember 1998 veröffentlicht [SU97]. Was die 3D- und die Szenengraph-Spezifikation anbelangt, sind diese zwar an VRML97/2.0 angelehnt aber nicht kompatibel. Der Nachrichtenaustausch zwischen den Objekten (engl. Routes), welcher die eigentliche Dynamik der Szenen ausmacht, wurde überhaupt nicht berücksichtigt. Die Java3D-Bibliotheken setzen auf die erst 1997 entwickelte Version JAVA 1.1 auf, welche eine gänzlich andere Ereignisbehandlung hat. Erschwerend kommt hinzu, daß eigenen Messungen zufolge die Ausführungsgeschwindigkeit von JAVA mit der Just-In-Time Option[3] oder in einer native Code-Anwendung um Faktor 10-50 langsamer ist, als eine vergleichbare C++ Implementation, was im Realtime-Bereich von VR-Applikationen nicht akzeptabel ist. Ferner ist die Ausführung von Java Applets mittels Browser sehr von der Implementierungsgüte des integrierten Interpreters (sogenannte Virtual Machine) abhängig. Man kann somit sagen, daß JAVA zwar betriebssystemunabhängig aber hochgradig browserabhängig ist.

2.6.4 Direct3D

Direct3D ist eine Schnittstelle, die sowohl einfache Grafikprimitive wie OpenGL als auch einen Szenengraph enthält [TN96]. Microsofts Direct3D läuft bisher nur auf den Betriebssystemen: Win95/98, Windows NT und Windows 2000. Portierungen auf andere Systeme gibt es nicht. Die Hersteller von Graphikkarten haben die hardwareseitige Unterstützung von Direct3D zugesagt, da im Gegensatz zu OpenGL nur minimale Funktionen oder einzelne Funktionsteile und nicht die ganze Renderengine in Hardware abgebildet werden müssen. Wird eine Funktion nicht hardwaremäßig unterstützt, so wird sie softwareseitig emuliert. Fakt ist aber, daß die

[3] Bei der Just-In-Time Option wird der plattformunabhängige Bytecode während der Laufzeit in prozessorabhängige Instruktionen umgewandelt. Bei native Code wird der Maschinencode direkt beim Compilieren erzeugt, das Programm ist in diesem Falle nicht auf einer anderen Plattform lauffähig, da der Bytecode nicht mehr vorhanden ist.

Hardwarebeschleunigung unter NT erst in der kommenden Version (Windows 2000) unterstützt wird und die Programmierung um einiges aufwendiger als bei OpenGL ist. Direct3D ist vorwiegend für den Spielemarkt gedacht und hat nicht die Darstellungsqualität von OpenGL. Als Resultat kann die Ausgabe derselben Szene auf verschiedenen Rechnern leicht unterschiedlich sein.

2.6.5 Open Inventor

Rikk Carey und Paul Strauss begannen 1989 bei Silicon Graphics mit der Entwicklung von IRIS Inventor. Es kam 1992 auf den Markt und benutzte IRIX GL als Ausgabe [SC92]. Dessen Nachfolger Open Inventor löste IRIX GL ab und setzte auf den offenen Standard OpenGL als 3D-Graphikbeschreibungssprache [WJ94]. Open Inventor ist eine objekt-orientierte Graphik3D-Bibliothek, welche ein schnelles Erstellen von Szenen inklusive einfachen Animationen und Interaktionen erlaubt. Open Inventor enthält deshalb, um die direkte Interaktion im gleichen Fenster mit der Maus zuzulassen, ein einfaches Ereignismodell. Die Inventor-Bibliothek beinhaltet eine Sammlung von unterschiedlichen Objekttypen, graphische Primitive wie Würfel, Kegel und Zylinder, inklusive deren Eigenschaften (Farbe, Transformation), einfache *Manipulators* and *Draggers*, *Sensors* und *Engines*, verschiedene Licht- und Kameratypen. Ferner gibt es Benutzungsoberflächenkomponenten, welche ein einfaches Navigieren mit der Maus erlauben und vordefinierte Widgets, die ein Ändern von Farben und Materialeigenschaften ermöglichen. Darüber hinaus ist die Erstellung von eigenen Objekten möglich. Neben dem objektorientierten Ansatz trugen zwei weitere Eigenschaften zu dem großen Erfolg von Open Inventor bei:

- Szenengraph
  Die Objekte der Szene werden in einem azyklischen hierarchischen Graph abgebildet.
- Szenendatenbank
  Die gesamte Szene kann in ASCII oder binärem Format in einer Datei abgespeichert werden.

Die Objekte im Szenengraphen werden als Knoten (engl. Nodes) bezeichnet und können auf die nachfolgenden Knoten Auswirkungen haben. Ist zum Beispiel der linke Nachbar von einem graphischen Primitiv ein Farbknoten mit dem RGB-Triple 1 0 0, so erscheint das Objekt rot. Man spricht in diesem Zusammenhang davon, daß der hierarchische Graph von oben nach unten und von links nach rechts traversiert wird. Da die Objekte durch Knoten repräsentiert werden, sind sie leicht editierbar. Ferner sind auch andere Operationen, die nichts mit dem

eigentlichen Rendering zu tun haben, wie Selektion, das Berechnen der Bounding Box oder Lesen und Schreiben im Objekt, selbst definiert (gekapselt). Für Objekte in einer höheren Abstraktionsebene gibt es Nodekits. Es handelt sich dabei um Schablonen, die aus mehreren Knoten bestehen, welche applikationsspezifisches Verhalten beinhalten können. Nodekits sind programmiertechnisch eine einzige Klasse, die überladene oder neue Funktionen enthält und deren integrierte Objekte als Szenensubgraph aufgefaßt werden können. Dadurch, daß Nodekits ebenfalls über Methoden für das Laden und Speichern einer Szenenbank verfügen, werden sie persistent.

Die Erleichterung durch Inventor liegt darin, daß keine OpenGL Programmierung nötig ist. Das heißt, der Programmierer braucht sich nicht um Details, wie den Aufbau von Displaylisten oder die Darstellung von Primitiven bzw. Dreiecksnetzen, zu kümmern. Der Modellierer kann sich somit auf den Inhalt konzentrieren und muß sich keine Gedanken machen, wie die 3D-Graphik auf den Bildschirm gebracht wird.

2.6.6 VRML 1.0

Im Mai 1994 präsentierte Mark Pesce auf der ersten internationalen Konferenz über WWW ein Paper, in dem er den ersten Prototypen eines 3D-Web Browsers namens Labyrinth vorstellte. Diesen hatte er zusammen mit Tony Parisi entwickelt [PK94]. Bei den Konferenzteilnehmern herrschte schnell Einigkeit über die Notwendigkeit einer gemeinsamen Sprache für 3D-Szenen und WWW-Hyperlinks, die verschiedene virtuelle Welten miteinander verbinden. Somit war VRML[4] geboren, und Dave Raggett prägte noch auf der Konferenz die Abkürzung Virtual Reality Markup Language[5]. Kurz nach der Konferenz wurde eine Mailingliste von Brian Behlendorf eingerichtet, damit eine weltweite Diskussion und Sammlung von Vorschlägen für eine mögliche VRML-Spezifikation stattfinden konnte. Schnell waren die Anforderungen für VRML klar: Plattformunabhängigkeit, Erweiterbarkeit und die Möglichkeit, mit Verbindungen mit niederer Bandbreite (Modem mit 14.4 KBps) zu arbeiten. Nach sorgfältiger Überlegung wurde das Open Inventor ASCII-Dateibeschreibungsformat von der VRML-Gemeinde als Basis für VRML ausgewählt [BP94]. Die Firma Silicon Graphics Inc. gliederte hierfür einen Teil des Open Inventor-Dateiformats aus und verzichtete durch die Veröffentlichung auf ihre Rechte. Namentlich Rikk Carey und Gavin Bell von Silicon Graphics adaptierten das Format an VRML, das zum Beispiel Erweiterungen für das Laden der Szene über Internet erlaubt (Knoten WWWAnchor und WWWInline). Ein wichtiger Nachteil von

[4] VRML wird „Wörmel“ ausgesprochen.

[5] VRML ist nun die Abkürzung für Virtual Reality Modelling Language

VRML 1.0 ist allerdings, daß nur statische Welten beschrieben werden können. Die Szenenobjekte sind unbeweglich und unveränderbar; die Interaktion beschränkt sich auf reine Bewegung durch die virtuelle Welt. VRML Szenen können durch stand-alone Programme oder durch sogenannte Plugins für Web-Browser angeschaut werden. Letztere ermöglichen das Einbetten von Szenen innerhalb von HTML-Seiten. Die Browser unterscheiden sich hauptsächlich in der Art der angebotenen Navigationstechnik und in der Benutzungsoberfläche. Zum Teil haben die Browser auch firmenspezifische Erweiterungen, wie Netzwerkerweiterungen, um Multi-User-Applikationen zu unterstützen. Leider gibt es Hersteller wie Netscape, die eigenmächtig ihren Browser Live3D durch eigene Szenenknoten erweiterten, was den Kompatibilitätsaspekt ad absurdum führt. VRML kann nicht nur für wissenschaftliche Visualisierungen und Walk-Through-Applikationen, wie Museumsbesuche, eingesetzt werden, sondern hat in der Version 2.0 auch das Potential für Spiele, Unterhaltung und Multimedia Anwendungen.

2.6.7 VRML 2.0

VRML 1.0 hat den entscheidenden Nachteil, daß es nur statische Szenen zuläßt. Deshalb wurde auf der SIGGRAPH-Konferenz im August 1995, die Weiterentwicklung von VRML eingeleitet, damit auch 3D-Animationen möglich sind. Ziel der eigens gegründeten VRML Architecture Group (VAG) war, die Entwicklung eines neuen VRML-Standards zu koordinieren, der dynamische interaktive Welten ermöglicht. Aus den eingehenden Vorschlägen wurde im März 1996 per Abstimmung über das Internet der Vorschlag von Silicon Graphics Inc. "Moving Worlds" ausgewählt [MH95]. Der vom WSI/GRIS miteingereichte Beitrag "The Power of Dynamic Worlds" [BE96a, BE96b] erreichte den zweiten Platz und lag somit vor den Vorschlägen von Apple, Sun oder Microsoft. Im Gegensatz zu den Mitbewerbern zeichnete sich dieser Vorschlag durch ein neues Nachrichtenkonzept/Ereignismodell, eine Multi-User-Unterstützung und dem nahtlosen Übergang von VRML 1.0 aus. Eine Integration dieser Komponenten in den Moving Worlds Vorschlag, wie von Gavin Bell angefragt, hielten die Autoren aufgrund des unterschiedlichen Ansatzes nicht für sinnvoll. Am 4. August 1996 wurde die endgültige Spezifikation von VRML 2.0 auf der SIGGRAPH 96 in New Orleans freigegeben. Die VRML-Sprache wurde im Januar 1998 international standardisiert und erhielt den neuen Namen: VRML97 [VC97]. Das Dokument trägt den Namen ISO/IEC 14772-1:1997. Es wurden hierbei nur kleine Änderungen gegenüber VRML 2.0 vorgenommen. VRML97/2.0 ist nicht mit VRML 1.0 abwärtskompatibel, jedoch kann VRML 1.0 in 2.0 Syntax überführt werden. Um die wesentlichen Erweiterungen von VRML 2.0, nämlich die Animationsfähigkeit, zu unterstützen, wurden mehrere Neuerungen eingeführt: Erstens ist

VRML 2.0 skriptfähig, das heißt, komplizierte Abläufe können in einer Programmiersprache wie JavaScript oder Java gefaßt werden, zweitens gibt es neue Knoten, kombiniert mit einem Event-Routing-Mechanismus, die einfache Animationen mittels Sensoren und Interpolatoren erlauben, und drittens können neue, eigene Knoten aus den existierenden Komponenten aufgebaut werden (genannt Prototypen).
Im Rahmen dieser Arbeit wurde C++ mit Inventor 1.x und 2.x sowie VRML 1.x und 2.x verwendet.

2.7 Netzwerkprotokolle

2.7.1 Grundlagen

Um Nachrichten zwischen Anwendungen auszutauschen, bedarf es der Auswahl des richtigen Protokolls und seiner Verbreitungsart. Zur Verwendung kommen Protokolle aus der TCP/IP-Gruppe (engl. Transmission Control Protocol/Internet Protocol), da Protokolle wie NetBEUI nicht routebar sind oder andere wie IPX/SPX nicht so häufig verbreitet sind. Auf den genauen Aufbau des OSI-Schichtenmodells soll hier nicht eingegangen werden, aber grundsätzlich gilt, daß jede Netzwerkschicht einen Dienst für die nächsthöhere Schicht hat. Dieser Dienst kann mehr oder weniger zuverlässig sein. Die Schicht über einem unzuverlässigen Dienstanbieter kann einen eigenen Fehlerkorrekturalgorithmus verwenden, um ihrerseits einen zuverlässigen Dienst anzubieten. Aufgrund dieser Eigenschaften unterscheidet man zwei Arten von Protokollen: Bei den *verbindungsorientierten* Protokollen wird eine virtuelle Verbindung zwischen Sender und Empfänger aufgebaut, bevor die Daten ausgetauscht werden. Die Verbindung wird abgebaut, sobald ein Teilnehmer die Beendigung wünscht. Vergleichbar ist die verbindungsorientierte Kommunikation mit einem Telefongespräch. Bei den *verbindungslosen* Protokollen überträgt der Sender einfach seine Nachrichten. Ein Beispiel für verbindungslose Protokolle wäre der Einwurf eines Briefes in einen Briefkasten. Dabei sollen die Briefe des Senders so schnell wie möglich an den Empfänger übermittelt werden [TA88]. Verbindungslose Dienste stellen Datenpakete in der Regel unzuverlässig zu[6]. Es gibt keine Garantie dafür, daß ein einzelnes Paket auch sein Ziel erreicht. Ferner stimmt die Reihenfolge des Eintreffens nicht unbedingt mit der des Abschickens überein, auch ein doppeltes Paket kann vorkommen. Demgegenüber garantiert die aufwendigere verbindungsorientierte Kommunikation die

[6] Der verbindungslose Dienst kann neben dem ungeprüften Datagrammdienst, bei dem es keine Empfangsgarantie gibt, auch ein Dienst mit Bestätigung für jedes Paket (ähnlich eines Einschreibens mit Rückschein) oder ein Dienst mit Rückantwort sein, bei dem der Sender eine Anfrage schickt und die Antwort als Quittung bekommt. Wesentlich ist hier im Gegensatz zur verbindungsorientierten Kommunikation, daß es keine virtuelle Verbindung während des Nachrichtenaustausches gibt. [TA88]

geordnete und fehlerfreie Zustellung. Dies wird durch den Einsatz von Mechanismen wie Empfangsbestätigung, Prüfsumme oder Paritätsbits erreicht.

2.7.2 Paketverbreitung

Gemäß der Verbreitungsart werden die paketorientierten Protokolle in die folgenden Datagrammtypen eingeteilt:

- Unicast
- Broadcast
- Multicast

2.7.2.1 Unicast

Bei Unicastverbindungen handelt es sich um Punkt-zu-Punkt Verbindungen zwischen zwei Rechnern. Dabei kann noch unterschieden werden, ob gleichzeitig gesendet und empfangen wird. Man spricht dann von einer full-duplex Verbindung, oder im anderen Fall von einer half-duplex Verbindung, bei der zu einem Zeitpunkt nur in einer Richtung gesendet beziehungsweise empfangen werden kann. Die Unicastverbindung ist am verbreitetsten und wurde in der letzten Zeit durch das Internet weiter ausgebaut, da herkömmliche Internetanwendungen hauptsächlich das Netzwerkprotokoll TCP/IP für die Kommunikation zwischen Server und Client verwenden. Unicastpakete sind für einen Empfänger bestimmt, so daß dieser folglich bekannt sein muß.

2.7.2.2 Broadcast

Beim Broadcast wird im Gegensatz zum Unicast ein Paket an alle empfangsbereiten Rechner gesendet. Je nach Art des Paketes wird dieses bestätigt oder nicht[7]. Ersteres wird zum Beispiel verwendet, um festzustellen, welche Rechner sich im Netz befinden, da diese ein Antwortpaket an den Sender zurückschicken. Zur Auflösung einer Adresse wird ein spezielles Protokoll (ARP-Address Resolution Protocol) verwendet, um den anderen Knoten beim Anschluß eines neuen Rechners die eindeutige Hardware-Adresse zusammen mit der IP-Adresse bekannt zu machen. Ein Broadcast-Paket bleibt normalerweise auf sein Subnetz beschränkt, da Router, Switches oder Bridges dieses nicht nach außen durchlassen. Der Grund liegt darin, daß ein Paket mit der Adresse 255.255.255.255 an sämtliche weltweit aktiven Rechner mit TCP/IP Protokoll geschickt würde. Alle Rechner müßten dann zuerst dieses Paket verarbeiten, was

[7] Unter Windows NT bieten Mailslots einen schnellen Einweg-Kanal durch Verwendung von unbestätigten Broadcast-Datagrammen, während sogenannte Named Pipes und Windows Sockets bidirektionale Kommunikationskanäle mit garantierter Datenzustellung zur Verfügung stellen.[SA97]

einem sinnlosen Ressourcenverlust gleichkommt, und die Rückantworten würden durch die hohe Belastung ein Netzwerk zusammenbrechen lassen.

Die Verwendung von Broadcasting eignet sich also zum Austausch von Nachrichten in kleinen lokalen Netzen, bei denen allen Rechnern dieselbe Information mitgeteilt und dabei die Last klein gehalten werden soll. Ferner kommt es in Frage, um mögliche Kommunikationspartner in Form von Rechnern zu finden.

2.7.2.3 Multicast

Eine Alternative zum Broadcasting bietet Multicasting, bei dem eine Reihe von Rechnern zu einer Gruppe zusammengefaßt wird. Hier wird ein Datagramm zwar von allen Mitgliedern empfangen, aber nicht an das übrige Netzwerk weitergegeben. Es gibt somit keine (CPU-) Belastung von nicht interessierten Rechnern/Knoten.

2.7.3 Netzwerkprotokolle

2.7.3.1 UDP/IP

UDP/IP (User Datagram Protocol) ist ein verbindungsloses Protokoll, welches nur geringe Erweiterungen gegenüber dem IP Protokoll, wie Unterstützung von Portnummern und optionale Prüfsumme zur Überwachung des Inhaltes, hat [PJ80]. Eine sichere Übertragung der Information ist nicht gewährleistet, gegebenenfalls muß die Applikation dies leisten. Aufgrund des niedrigen Overheads eignet sich dieses Protokoll für die schnelle Übertragung von kleinen Datenmengen. Um Datenverluste klein zu halten, sollten die Daten auch nicht kurzfristig in einer hohen Frequenz abgegeben werden, da es sonst zu Verlusten/Kollisionen kommen kann.

2.7.3.2 TCP/IP

Dies ist ein verbindungsorientiertes Protokoll und bietet einen sicheren Datenaustausch zwischen einer Punkt-zu-Punkt Verbindung [PO81]. Da TCP (Transmission Control Protocol) auf dem unsicheren Protokoll IP (Internet Protocol) aufbaut, erhielt dieses im Unix und Internetbereich weit verbreitete Protokoll den zusammengesetzten Namen TCP/IP. Da eine sichere, geordnete und vollduplexfähige Übertragung gewährleistet wird, setzen Applikationen wie FTP oder Telnet auf dieses Protokoll auf. Da aber andererseits der Aufbau einer virtuellen Verbindung zwischen zwei Rechnern mittels eines Handshake-Mechanismus relativ aufwendig ist, lohnt sich dieser Protokollansatz nicht beim Versenden von kleinen Datenmengen.

2.7.3.3 IP-Multicasting

Steve Deering erfand das Multicasting Konzept für Gruppenkommunikation im Rahmen seiner Doktorarbeit an der Stanford Universität [DS88]. Das Wesen von Multicasting besteht darin, daß für eine bestimmte Interessengruppe eine IP-Nummer festgelegt wird. Diese IP-Nummer liegt in einem speziellen Adreßbereich (224.0.0.0 - 239.255.255.255), wobei jedoch manche Bereiche reserviert sind und eine freie Auswahl erst ab 224.2.0.0 erfolgen kann. Ein Rechner kann solch einer Gruppe nun jederzeit beitreten oder sie verlassen. Man spricht in diesem Zusammenhang von einer dynamischen Gruppe. Ferner besteht die Möglichkeit, daß sich ein Rechner an mehreren Gruppen beteiligt. Dies ist sogar sinnvoll, da in einer Gruppe der Ton und in der anderen das dazugehörige Bild übertragen wird, wobei wir beim primären Einsatzbereich von Multicasting sind: Videokonferenzsysteme, Vorträge/Seminare inklusive Tafelanschrieb (engl. Whiteboard) und Internetradio- bzw. Konzertübertragungen [KV96]. Der Vorteil von Multicasting liegt darin, daß gleich viele Pakete versandt werden, unabhängig von der eigentlichen Teilnehmeranzahl. Es wird ersichtlich, daß ein Ausfall eines oder weniger Pakete die Verständlichkeit der Nachricht kaum beeinflußt. Aus diesem Grund ist beim IP-Multicasting weder der sichere Empfang noch die richtige Reihenfolge der Daten sichergestellt, da dies auf Kosten der Übertragungsrate gehen würde. Für andere Anwendungen, wie Simulationen, bei denen dies jedoch eine größere Rolle spielt, gibt es eine Reihe alternativer Ansätze, die mehr Sicherheit auf höherer Protokollebene bieten. Ein guter Überblick über diese Protokollansätze wie RAMP, TRM, MTP-2, RMP, ALF/SRM und DIS findet sich in der Arbeit von Schick [SD97]. Die Anwendung von Multicasting erfordert den Anschluß an ein spezielles Netz, welches MBone[8] genannt wird (siehe Abbildung 8). Dies liegt historisch darin begründet, daß früher Router keine Multicast-Pakete weiterleiten konnten. Die Multicasting-Pakete werden deshalb in reguläre IP-Pakete verpackt und über das Internet zu einem anderen Rechner gesendet. Auf diesem Rechner läuft ein spezielles Programm (mroute - Abkürzung für IP-Multicast Routers), das die Pakete entgegennimmt, entpackt und dem Subnetz zur Verfügung stellt. Das Senden in die andere Richtung funktioniert analog. Die Verbindung zwischen den Rechnern wird als Tunnel beziehungsweise der Vorgang als Tunneln bezeichnet. Durch diese baumartige Struktur ist die Verteilung und Replikation des Multicast Datenstromes in Echtzeit weltweit über das vorhandene Netz möglich. Es ist nun nicht gewollt, daß alle Konferenzen weltweit übertragen werden, und deshalb wurden zwei Mechanismen entwickelt, um die Reichweite der Multicast-Pakete zu begrenzen: Beim älteren

[8] Im Rahmen dieser Arbeit hat der Verfasser Tübingen an das MBone-Netz über die Uni Stuttgart (RUS) angeschlossen. Zuerst wurden die Pakete über einen lokalen Rechner am WSI/GRIS-Institut geroutet. Nach Aufrüstungen des Netzwerks werden nun Multicast-Pakete direkt über die DFN Netzanbindung an das Rechenzentrum (ZDV) geleitet, was eine deutliche Leistungssteigerung brachte.

Verfahren, dem *Truncated Tunnels*, beinhaltet jedes Paket einen Parameter namens TTL (Time-To-Live). Passiert nun ein Paket einen MRouter, so wird ein bestimmter Wert (Kosten) abgezogen und in dem Paket modifiziert. Eine Weiterleitung und Modifizierung des Pakets zu den Nachbarroutern erfolgt nur dann, wenn dieser Wert über dem Kosten-Schwellwert des Routers liegt. Das zweite ergänzende Verfahren heißt *Pruned Tunnels*, in der Literatur auch als *True Multicasting* bekannt. Hierbei werden Multicast-Pakete nicht an andere MRouter oder Teilnehmer weitergeleitet, solange diese nicht explizit Interesse an einer Session angemeldet haben. Diese An- und Abmeldung geschieht mittels eines speziellen Protokolls, des Internet Group Management Protocol (IGMP). Das Routing dieser Multicast-Pakete wird durch das Distance Vector Multicast Routing Protocol (DVMRP) oder alternativ den Multicast Open

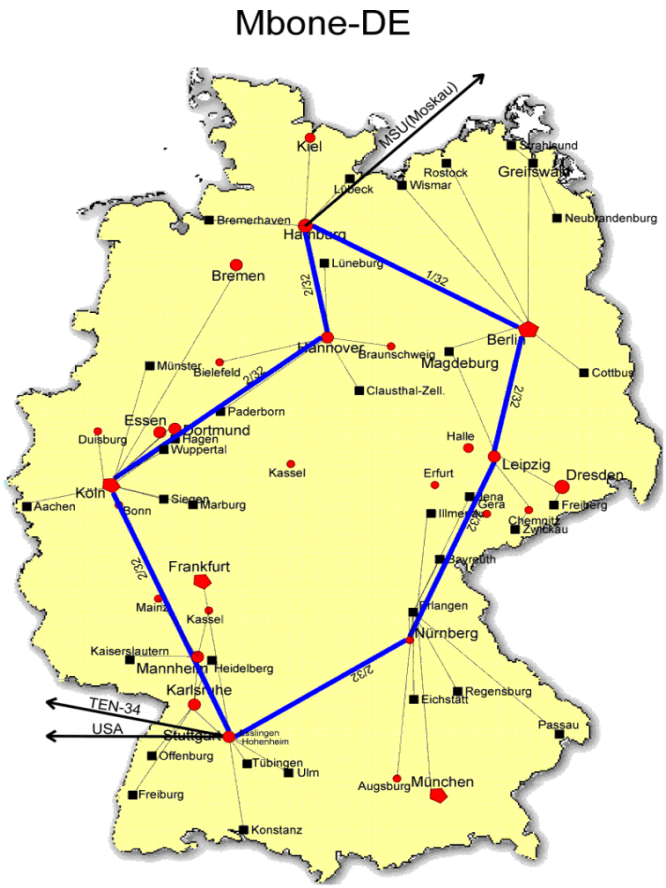


Abbildung 8: Das deutsche MBone-Netz für Multicast-Anwendungen

Shortest Path Factor (MOSPF) vorgenommen. Diese zwei Verfahren finden sich auch in der neuesten Routergeneration (z. B. CISCO) wieder und machen den zuvor angesprochenen mroute-Prozeß auf den Rechnern zum Tunneln immer mehr überflüssig.

2.7.4 Bewertung der Netzwerkprotokolle

Je nach Aufgabengebiet bieten sich folgende aufgeführten Protokolle für den Nachrichtenaustausch an.

Protokoll	Eigenschaften	Einsatzgebiet
TCP/IP	Zuverlässiges Protokoll: Sichere Datenübertragung; Langsame Übertragungsraten	• Versorgung mit wichtigen Daten wie bei Initialisierung • An- oder Abmeldung
UTP	Unzuverlässiges Protokoll: Dateninhalt sicher, aber insgesamt können Pakete verloren gehen oder dupliziert werden; Hohe Datenübertragungsrate	• Bei geringer Teilnehmerzahl für Multimedia (Video, Audio) • Bei hohen Updateraten mit geringen Datenmengen wie Positionsänderungen
IP-Multicasting	Unzuverlässiges Protokoll: Weder Empfang noch richtige Reihenfolge der Pakete sichergestellt	• Bei sehr hoher Teilnehmerzahl • Hervorragend geeignet für Multimedia/Konferenzbereich (Video, Audio)

Die Auswahl des Protokolls hängt ferner ab von:

- Anzahl der zu synchronisierenden (Szenen-)Objekten
- Relevanz der (Update-) Information
- Frequenz der Updatezyklen
- Anzahl der Anwender bei verteilten Umgebungen
- Anzahl der unterschiedlich verteilten Komponenten
  bei zusammengesetzten Umgebungen

2.8 Netzwerkmodelle

2.8.1 Allgemein

Bevor die geeignete Form der Kommunikationstechnik zwischen verschiedenen Teilnehmern/ Rechnern ausgewählt werden kann, muß die Kommunikationsform analysiert werden. Grundsätzlich kann zwischen den zentralen Netzwerkmodellen (Client-Server) und den dezentralen Netzwerkmodellen (engl. Peer-to-Peer) unterschieden werden. Die dezentralen Netzwerkmodelle werden je nach verwendetem Protokoll in einfach verteilte Netzwerkmodelle, Broadcast-Netzwerkmodelle und Multicast-Netzwerkmodelle eingeteilt.

2.8.2 Zentrales Netzwerkmodell

Beim Client-Server-Modell hat mindestens ein Rechner, der Server, eine übergeordnete Rolle beziehungsweise eine Kontrollfunktion. Deshalb spricht man auch von einem zentralen Modell. Dieser eine Server sammelt alle Daten von den anderen Rechnern und speichert die Änderungen in eine zentrale Datenbank ab. Im einfachsten Fall schickt der Server dann die komplette Szenendatenbank an die Teilnehmer. Im anderen Fall werden nur die Änderungsnachrichten an jeden teilnehmenden Rechner geschickt und die lokale Kopie der Teilnehmer auf den neuesten Stand gebracht. In beiden Fällen visualisiert jede Maschine die Szene und verwaltet die lokalen Benutzereingaben. In der Literatur wird dieses Kommunikationsmodell auch als logische Stern-Konfiguration referiert. Je nach Kommunikationsform gibt es einen einzigen Server, der mit den Clients Nachrichten austauscht oder es gibt mehrere Server, bei denen jeder verschiedene Clients bedient und sich die Server wiederum untereinander selbst abgleichen müssen. Damit zeigt sich auch die Hauptschwäche dieses Ansatzes: Er ist nicht beliebig skalierbar, da der zentrale Computer die Hauptlast trägt. Sind viele Prozesse durch die verschiedenen Eingaben der Teilnehmer zu verarbeiten oder nehmen zu viele Teilnehmer an einer Sitzung teil, so kommt es hier schnell zu einem Leistungsengpaß und die Anwendung hat durch die lange Wartezeit nicht mehr den Charakter einer Echtzeitanwendung. Client-Server Architekturen sind leichter zu implementieren, da keine Konsistenzprüfungen und Aktualisierungenmeldungen zwischen mehreren Datenbanken notwendig sind und kommen deshalb für den größten Teil von verteilten Anwendungen zum Einsatz. Dieses Modell wurde auch für die Übertragung von Nachrichten in dieser Arbeit eingesetzt.

2.8.3 Verteiltes Netzwerkmodell

Anders verhält es sich beim Peer-to-Peer Modell, bei dem alle Teilnehmer gleich sind. Das heißt, es gibt keine übergeordnete Instanz und alle Teilnehmer fungieren als Server und Client zugleich. Ein Beispiel dafür wäre ein Videokonferenzsystem. Es handelt sich hier also um ein verteiltes Modell, welches bedingt skalierbar ist. In der Regel sind dabei alle Teilnehmer miteinander verbunden. Jeder Rechner hat lokal eine eigene komplette Kopie der (Szenen-) Datenbank und ist für die Darstellung, Berechnung und Animation der Objekte zuständig. Gibt es Änderungen an der eigenen Datenbank, so muß die Anwendung die Änderungsmeldungen an alle anderen Teilnehmer schicken, damit diese ihre individuellen Datenbanken anpassen können. Aus diesem Grund gibt es auch hier ein Skalierungsproblem. Sind n Clients bei einer virtuellen Simulationsanwendung beteiligt und sendet jeder die neuen Positionsdaten an alle anderen, so werden insgesamt $n \cdot (n-1)$ Nachrichten über das gesamte Netz gesendet.

2.8.3.1 Broadcast-Netzwerkmodell

Um die Anzahl der Nachrichten zu reduzieren, wird in diesem Fall Broadcasting eingesetzt, welches es erlaubt, daß eine einzige Nachricht von allen anderen Rechnern empfangen werden kann. Nachteil dieser Technik ist, daß sämtliche Rechner, auch jene die nicht an der Simulation teilnehmen, diese Nachricht erhalten und damit belastet werden. Ein weiter Nachteil ist, daß sich alle Rechner im selben Subnetz befinden müssen.

2.8.3.2 Multicast-Netzwerkmodell

Multicasting ist ein spezielles Broadcastingverfahren, welches eine Gruppenadressierung erlaubt. Beim Einsatz von Multicasting bekommen nur die registrierten Rechner eine Nachricht zugeschickt und die anderen "uninteressierten" Rechner im Netz bleiben unbelastet. Dazu muß ein Rechner Anschluß an das schon erwähnte MBone-Netz haben, welches das Routen von IP-Multicast-Paketen unterstützt. Es sei nochmals darauf hingewiesen, daß es keine Sicherheit darüber gibt, ob das IP-Multicast-Paket auch wirklich bei einem Rechner ankommt. Durch zusätzliche eigene Mechanismen muß hier Vorsorge getroffen werden.

2.9 Techniken für Multi-User-Systeme

2.9.1 Dead Reckoning

Durch den Einsatz von Broadcast und Multicast zur Paketverbreitung wird, wie im vorangegangenen Kapitel erwähnt, die Anzahl der aufzubauenden Verbindungen und der Pakete, welche gesendet werden müssen, drastisch reduziert. Dennoch stößt solch ein System schnell an seine Grenzen, wenn es sehr viele Teilnehmer (Größenordnung > 1000) gibt. Hier kommt dann ein Algorithmus zum Einsatz, welcher Dead Reckoning genannt wird, und der in dem bekannten Protokoll DIS (Distributed Interactive Simulation Protocol) [IE93] unter SIMNET [PA89] oder NPSNET [MZ94] angewandt wird. Kernpunkt des Verfahrens ist, daß ein (militärisches) Objekt durch die physikalischen Parameter Ort, Bewegungsrichtung, Geschwindigkeit, beziehungsweise Beschleunigung usw. beschrieben wird. Kommt nun ein Netzwerkpaket bei einem Teilnehmer nicht an, so wird die Bewegung mittels der letzten empfangenen Daten interpoliert. Sobald nach dem Ausfall wieder die tatsächliche Position mitgeteilt wurde, wird eine Kurskorrektur durchgeführt (engl. track smoothing).

2.9.2 Heartbeats

Um die Zuverlässigkeit des Systems (verlorene Nachrichten) zu erhöhen und neue Teilnehmer schnell über den Status des Gesamtsystems zu informieren, wird eine Technik eingesetzt, die sich Heartbeats nennt. Dabei sendet jeder Teilnehmer in periodischen Abständen Nachrichten

über seinen augenblicklichen Status. Gleichzeitig weisen diese Nachrichten darauf hin, daß der Teilnehmer noch aktiv an der Sitzung beteiligt ist. Solche Nachrichten werden, falls keine weiteren Informationen enthalten sind, auch als "I-am-alive" Pakete bezeichnet.

2.10 Programmierschnittstellen

2.10.1 Sockets

Die einfachste Art des Nachrichtenaustausches kann durch die Verwendung von Sockets realisiert werden. Bei der Erzeugung eines Sockets wird ein Dateideskriptor zurückgeliefert, mit dessen Hilfe der Auf- bzw. Abbau der Verbindung sowie der Transport von Daten mittels Lese- und Schreibvorgängen bewerkstelligt wird. Bei der Erzeugung des Socket kann festgelegt werden, ob die Verbindung zwischen einem zuverlässigen verbindungsorientierten Bytestrom, einem zuverlässigen verbindungsorientierten Paketstrom oder einer unzuverlässigen Paketübertragung erfolgen soll. In den beiden ersten Fällen wird die Wahl auf TCP/IP fallen und bei der unzuverlässigen Paketübertragung normalerweise auf UDP. Da Socketverbindungen auf relativ niedrigem Abstraktionsniveau ablaufen, sind sie sehr schnell. Sockets wurden beim netzwerkbasierten Nachrichtenaustausch in dieser Arbeit eingesetzt.

2.10.2 Komponentensoftware

Komponentensoftware ist eine konsequente Weiterentwicklung des objektorientierten Ansatzes zur Wiederverwendung von Bibliotheken, speziell für den Einsatz von verteilten Systemen. Alle Ansätze basieren auf dem Client-Server-Modell. Die wichtigsten drei Vertreter dieser Technik sind Common Object Request Broker Architecture (CORBA), Microsofts Distributed Component Object Model (DCOM) und Suns Java Remote Method Invocation (RMI) inklusive der Technik der Object Serialization. Der Ansatz von Sun läuft nur auf Java, das heißt, es handelt sich um eine netzweite Java-zu-Java-Kommunikation auf Objektebene. DCOM und CORBA sind sprachenunabhängig konzipiert. Eine eigene sogenannte Interface Definition Language (IDL) dient zur Beschreibung der Schnittstelle. Ein wesentlicher Unterschied zwischen DCOM und CORBA-Ansatz ist, daß bei CORBA kein Protokoll für die Kommunikation zwischen Server und Client spezifiziert wurde, das Protokoll somit herstellerabhängig ist [CH97]. Die Spezifikation von DCOM wurde auf Binärebene abgefaßt[9]. Eine binäre Komponente kann somit auf Komponenten anderer Hersteller zugreifen, ohne daß, wie bei CORBA, der Quelltext nötig ist. Die Investitionen der Softwareentwicklung

[9] Genau genommen handelt es sich um einen binären Standard je Plattform, das heißt, die binären Komponenten laufen beispielsweise nur auf Intel-Architekturen. Durch das DCOM-Protokoll auf DCE RPC-Basis wird ein Austausch über die Plattformen hinaus möglich.

bleiben hiermit gewahrt. Technisch basiert die DCOM-Schnittstelle auf einer standardisierten Speicheranordnung der Funktionsaufrufe, die den virtuellen Funktionstabellen unter C++ entsprechen [RD96]. Um zum Beispiel die fünfte Funktion mit dem Argument 42 aufzurufen, würde der Funktionsaufruf mittels Zeiger in der Funktionstabelle (*((*p)+5) (p,42)) lauten. Durch den Einsatz von Komponentensoftware wird den Clients erlaubt, Funktionen eines entfernten Servers mittels Remote Procedure Calls (RPC) aufzurufen. Dabei wird in aller Regel eine 3-stufige Architektur (3-Tier) verwendet, bei der die verteilte Umgebung in die Einheiten Präsentation (Client-Benutzer-Schnittstelle), Funktionalität (Verbindung zum Client wie auch zum Datenbankserver) und Daten eingeteilt wird (siehe Abbildung 9). Ein Vorteil bei der Verwendung von Komponentensoftware besteht darin, daß man sich nicht um Punkte wie Marshaling[10] (Remote Zugriff mit Zeigern), Implementierung von Objektserialisierungen oder eigenen Protokollen befassen muß. Der Hauptvorteil ist aber die Austauschbarkeit und Erweiterung von Serverfunktionen unter Beibehaltung der alten Clientschnittstelle. Man

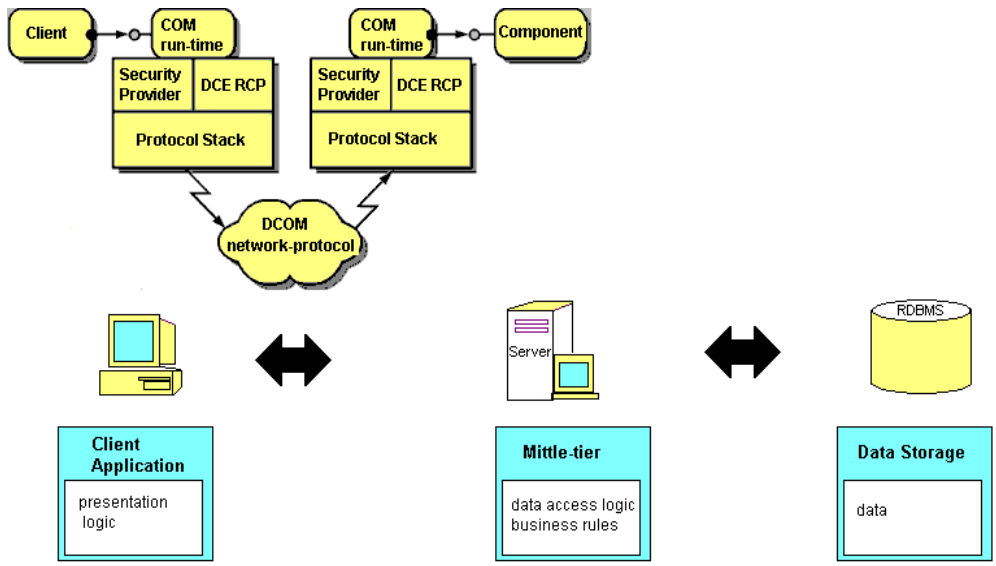


Abbildung 9: Der 3-stufige Aufbau von Applikationen mittels DCOM-Einsatzes

spricht in diesem Zusammenhang auch von der Transparenz der Serverkomponenten für den Client. Dies minimiert den Administrationsaufwand, da nur die serverbasierte Komponente in der Funktionalitätseinheit ausgetauscht werden muß und die Clients unangetastet bleiben. Diese Technik kommt deshalb bei Datenbankanwendungen (z. B. SAP/R3) zum Einsatz. Die Architektur sieht einen/mehrere Server und viele Clients vor und nicht einen Client, der viele

[10] Microsoft versteht darunter die Analyse von Zeigern, das Einpacken in durchgehende Speicherbereiche beim Sender. Dies ermöglicht den Remote-Zugriff der unterschiedlichen Adressräume von Server/Client. Eventuell werden die Bytefolgen noch gedreht. Analog wird das Auspacken als Unmarshaling bezeichnet.

Server beschäftigt, wie bei verteiltem parallelisiertem Rechnen üblich ist. Als Nachteil sind die langsame Übertragungsgeschwindigkeit der Pakete im Vergleich zur Socketimplementierung, die insgesamt niederere Anzahl von Transaktionen pro Sekunde, das Fehlen von Gruppenadressierung (engl. Multicasting), die eingeschränkten Plattformen und die kompliziertere Programmierung zu nennen.

2.11 Anwendungen

Historisch gesehen wurden die ersten VR-Anwendungen bereits entwickelt, bevor es den populären VR-Begriff gab. Primär handelte es sich dabei um Computersimulationen und sekundär um den Forschungs- und Wissenschaftsbereich, bei dem es um eine Verbesserung der VR-Techniken selbst geht. Bei diesen Simulationen werden natürliche und technische Abläufe und Vorgänge mittels eines Modells im Computer nachgebildet. Nach Page et al. kann nach folgenden Zielen unterschieden werden [PB88, KU94]:

- Erforschung komplexer Systeme
- Demonstration und Veranschaulichung komplexer Vorgänge
- Entscheidungshilfe
- Trainings- bzw. Schulungsunterstützung
- Optimierung des Systemverhaltens am Modell

Der Vorteil von Simulationen mit VR-Technik liegt nun darin, daß einerseits interaktiv eingegriffen werden kann und andererseits eine visuelle Rückkopplung erfolgt. Ein komplexes Modell kann somit besser verstanden und die Schlüsselparameter können eingestellt werden. Der Anwender wird durch den Interaktionsprozess aktiv in die Simulationsszene eingebunden. In diesem Zusammenhang wird auch von *graphisch-interaktiven Simulationen* gesprochen. Durch die Nutzung neuerer Ein- und Ausgabegeräte ist dieser Bereich teilweise in den VR-Bereich, bei dem der Benutzer in die Szene eintauchen kann, übergegangen. Als Beispiele für Simulationen im VR-Bereich sind stellvertretend folgende Bereiche zu nennen: Fertigungsabläufe in Fabriken, Werkstoffentwicklung im Automobil-/Flugzeugbau, Design im virtuellen Windkanal, Flug- und Fahrsimulationen, Gestaltung von Innenräumen, Städteplanung und militärische Simulationen.

Ein weiterer Haupteinsatzbereich von VR-Anwendungen liegt in der Medizin. Darunter fallen neben der eigentlichen OP-Planung von Eingriffen auch die Abschätzung von Bestrahlungs-

schäden [KU94], Übungen zur Aus- und Weiterbildung und die Fernoperationen mittels Telemetrie.

Das große Marktpotential von VR-Applikationen liegt im Unterhaltungssektor. Neben dem Erotikbereich ist vor allem der Spielesektor durch 3D-artige Spiele wie DOOM ein wesentliches Marktsegment.

Durch die Nutzung der globalen Vernetzung kommt der Interaktion, Kommunikation und Kooperation zwischen Teilnehmern ein neuer Stellenwert zu. Verteilte kooperative Anwendungen, gekoppelt mit multimedialen Komponenten werden die nächste Generation von VR-Software sein.

2.12 Limitierungen

Im Bereich der 3D-Grafik wurden zwar zahlreiche Graphikbeschreibungssprachen und Algorithmen entwickelt, völlig außer acht gelassen wurde aber der Kommunikationsaspekt. Fragen wie:

- Erfordert die Erweiterung von 2D auf 3D ein anderes Kommunikationsmodell?
- Wie können/sollen 3D-Objekte miteinander kommunizieren können?
- Welche Art von Daten müssen/sollen ausgetauscht werden?
- Aus welchen Bestandteilen soll/muß eine solche Nachricht bestehen?
- Was sind die Vor-/Nachteile einer solchen nachrichtenbasierten Kommunikation?
- Welche Mechanismen für die Implementierung sind nötig?

Diesen und anderen Fragen wird in den folgenden Kapiteln nachgegangen werden.

Ferner wird der Versuch unternommen, ein einfaches Bildverarbeitungsprogramm mit Hilfe von VR abzubilden - speziell für den Bereich von Medizinern, genauer Radiologen. Es geht dabei darum, wie eine Benutzungsoberfläche aussehen könnte, damit ein Mediziner sofort damit arbeiten kann, ohne zuvor ein Handbuch lesen zu müssen. Handelsübliche Programme leisten dies in der Regel nicht[11].

[11] Der Verfasser kann dies aus eigener Erfahrung bestätigen, da er die Leitung des Projekts „Klinik 2000" der Firma Ott&Adis am Klinikum Tübingen innehatte. Es handelt sich dabei um ein Abteilungsinformationssystem für die elektronische Krankenakte auf Basis von relationalen SQL-Datenbanken.

3 Stereoskopie

3.1 Physiologische Grundlagen des räumlichen Sehens

Die Wahrnehmung von räumlicher Tiefe wird primär durch das binokulare stereoskopische Sehen gewonnen. Aber auch monokulare Mechanismen spielen eine Rolle. Folgende Informationsquellen werden hierfür ausgewertet: Größenunterschiede bekannter Gegenstände, Überdeckungen, Schatten, perspektivische Verkürzungen und vor allem die parallaktische Verschiebung der Gegenstände relativ zueinander, die bei einer Bewegung des Kopfes entstehen [ST87]. Das Wesen des binokularen Sehens liegt darin, daß ein Gegenstand der näheren Umgebung aus geometrisch-optischen Gründen auf verschiedene Stellen (Winkel

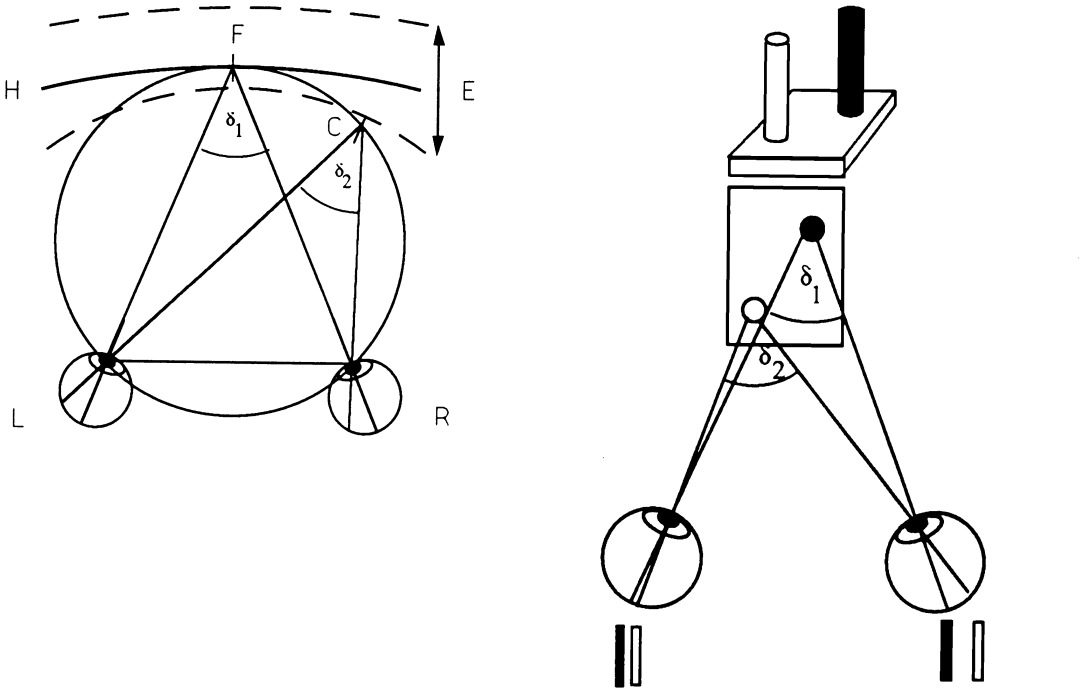


Abbildung 10: a.) Empirischer Horopterkreis K mit Abstand E in dem Objekte noch binokular ohne Doppelbilder fusioniert werden. b.) Entstehen der horizontalen Parallaxe [nach FT92]

gemessen von der Fovea, der Stelle des schärfsten Sehens) jeder Netzhaut fällt. Diese sogenannte Querdisparation oder horizontale Parallaxe ist um so größer, je größer und näher

ein Gegenstand ist (siehe Abbildung 10a.). Wird die Querdisparation zu groß, werden Doppelbilder wahrgenommen (>1,5º). Die Verschmelzung der beiden unterschiedlichen Bilder zu einem einzigen Bild mit stereoskopischer Tiefe nennt man binokulare Fusion. Bei der Fixation eines Punktes bewegen sich die Augen aufeinander zu (sie konvergieren); dies hat zur Folge, daß der abgebildete Punkt genau auf die Mitte der Netzhaut (Fovea) fällt. Ferner werden bei der Fixation alle anderen Gegenstände, welche sich auf einem Kreis befinden auf dem der Knotenpunkt des optischen Systems eines jeden Auges und der Fixationspunkt liegen, auf sogenannten korrespondierenden Netzhautstellen abgebildet. Mathematisch wäre dies der Vieth-Müller-Kreis, experimentell ergeben sich aber Abweichungen und man spricht vom Horopterkreis. Punkte auf dem Horopterkreis werden einfach gesehen, und Punkte innerhalb eines Bereiches um den Horopterkreis können binokular fusioniert werden. Außerhalb dieses Bereiches kommt es zu den Doppelbildern (siehe Abbildung 10 b.).

3.2 Stereomodelle

Um einen stärkeren räumlichen Eindruck zu gewinnen, bietet sich die Stereoprojektion an. In der Computergraphik wird dabei eine Szene aus zwei verschiedenen, horizontal liegenden Kamerapositionen betrachtet. Während das menschliche Sichtvolumen einem irregulären Kegelstumpf entspricht, wird in der Computergraphik ein Pyramidenstumpf verwendet. Ferner wird aus Leistungs- und einfacheren Realisationsgründen ein symmetrisches Viewing Volumen verwendet. Das heißt, die beiden Kameraöffnungswinkel links und rechts des Sehstrahls zum Brennpunkt, sind gleich groß. Dies hat zur Folge, daß es Bereiche gibt, welche jeweils nur von einer Kamera gesehen werden. Bei dem von beiden Kameras gesehenen, überlappenden Bereich spricht man vom *Scheinfenster (Plane of Convergence)*. In der Photographie werden die beiden gewonnenen Stereohalbbilder soweit beschnitten, bis beide möglichst nur noch den gemeinsamen Anteil zeigen. Im folgenden Abschnitt sollen die Stereomodelle vorgestellt werden, welche vorwiegend bei der perspektivischen Projektion verwendet werden. Im Gegensatz zur orthographischen Projektion entspricht die perspektivische mehr dem menschlichen Auge, da weiter entfernte Gegenstände kleiner und nahe größer dargestellt werden.

3.2.1 Rotationsmodell

Das Rotationsmodell kommt dem physiologischen Modell sehr nahe. Die Hauptsehstrahlen treffen sich im fixierten Punkt (engl. Point-of-Interest kurz: POI). Die Augen drehen sich zum Objekt hin, was der physiologischen Konvergenz entspricht. Computertechnisch wird nun für jedes Auge eine Zentralprojektion durchgeführt. Das gewonnene Bild auf der Projektionsebene steht senkrecht auf den Hauptsehstrahlen, folglich sind die beiden Bilder um den Winkel δ_1 gegeneinander rotiert. Dies hat zur Folge, daß die Objekte im Bild für das linke Auge etwas anders rotiert dargestellt werden als im Bild für das rechte Auge. Bei nahen Objekten wirkt sich dieser Sachverhalt, welcher als *vertikale Parallaxe* bezeichnet wird, besonders störend auf die Wahrnehmung aus, da die Bilder nicht mehr binokular fusioniert werden können (siehe Abbildung 11b.).

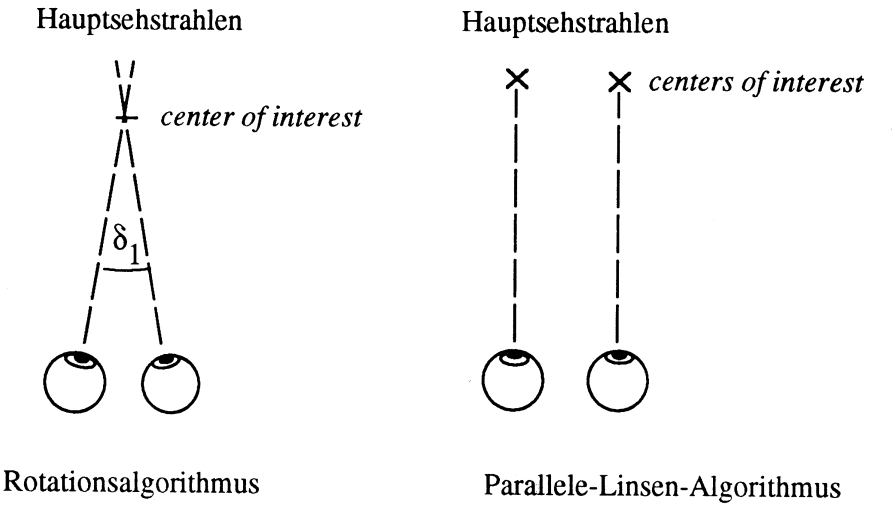
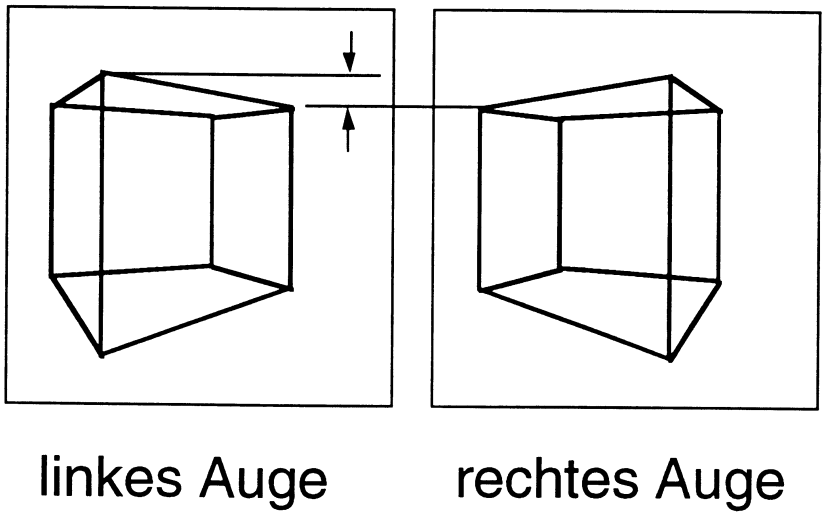


Abbildung 11: a.) Gegenüberstellung des Rotations- und des Parallele-Linsen-Modells



b.) Vertikale Parallaxe durch perspektivische Verzerrung bei Verwendung des Rotationsalgorithmus [nach FT92]

3.2.2 Das Parallele-Linsen-Modell

Eine vertikale Parallaxe wird bei diesem Modell vermieden, da mit zwei Fixationspunkten gearbeitet wird. Weil diese beiden Hauptsehstrahlen parallel zueinander sind, liegen die beiden gewonnenen Bilder auf einer gemeinsamen Projektionsebene. Der eigentliche Blickpunkt (Zyklopenauge) liegt genau zwischen den beiden Fixationspunkten. Das Parallele-Linsen-Modell hat die Eigenschaft, daß Punkte, die bereits vor der Projektion in der Projektionsebene liegen, in sich selbst abgebildet werden. Sie haben daher keine Parallaxe. Die Projektionsebene der stereoskopischen Projektion wird deshalb auch als *zero parallax plane* bezeichnet [LL91, RK92]. Punkte zwischen Blickpunkt und zero parallax plane besitzen negative, Punkte dahinter positive Parallaxe. Eine ausführliche Darstellung des Parallele-Linsen-Algorithmus findet sich in der Arbeit von Fröhlich [FT92].

Bei dieser Arbeit wurde folgendes Stereoprojektionsmodell eingesetzt:
Für die perspektivische Stereoprojektion wurde das Rotationsmodell eingesetzt. Der Point-of-Interest entsprach dem Schwerpunkt des Sichtvolumens. Die Translation entspricht dem normierten Augabstand (63,5 mm). Man spricht hier auch von Basisbreite.
Beim einfacheren Fall der orthogonalen Projektion wurde die Kamera nicht einfach um Δx verschoben, sondern nur um den POI rotiert. Bewährt hat sich hier der Wert von $\pm 3°$. Dieses Verfahren hat zur Folge, daß der Bereich der gemeinsamen Überlappung größer ist.

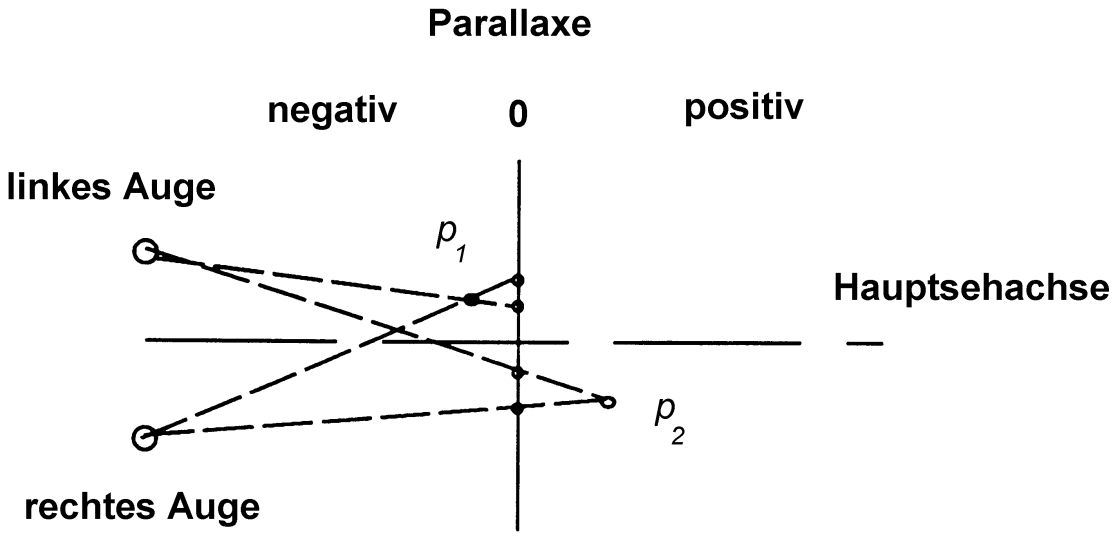


Abbildung 12: Positive und negative Parallaxe beim Parallele-Linsen-Modell

3.2.3 Technik

Es wurden folgende Techniken für die Stereowiedergabe verwendet:

- Anaglyphen (Rot/Blau bzw. Rot/Grün Brille)
- Polarisation (Polarisationsbildschirmvorsatz von Tektronix)

Die im Unterhaltungsbereich vorkommenden Autostereogramme oder auch SIRDS genannt (englische Abkürzung für Single Image Random Dot Stereograms) wurden auf ihre Eignung untersucht. Bei Autostereogrammen wird das Bild für das linke und rechte Auge in einem einzigen Bild zusammengefaßt. Die Tiefeninformation der ursprünglichen Szene wird durch

Abbildung 13: Beispiel eines farbigen SIRDS. Es sind die Buchstaben WSI dargestellt.

Zuordnung einer Farbe oder Helligkeit codiert. Nicht korrespondierende Punkte erhalten eine zufällig ausgewählte Farbe bzw. Helligkeit. Da der Informationsverlust von realer Farbe und Auflösung für den wissenschaftlichen Bereich zu hoch ist, kommt dieses Verfahren nicht zum Einsatz [TG96].

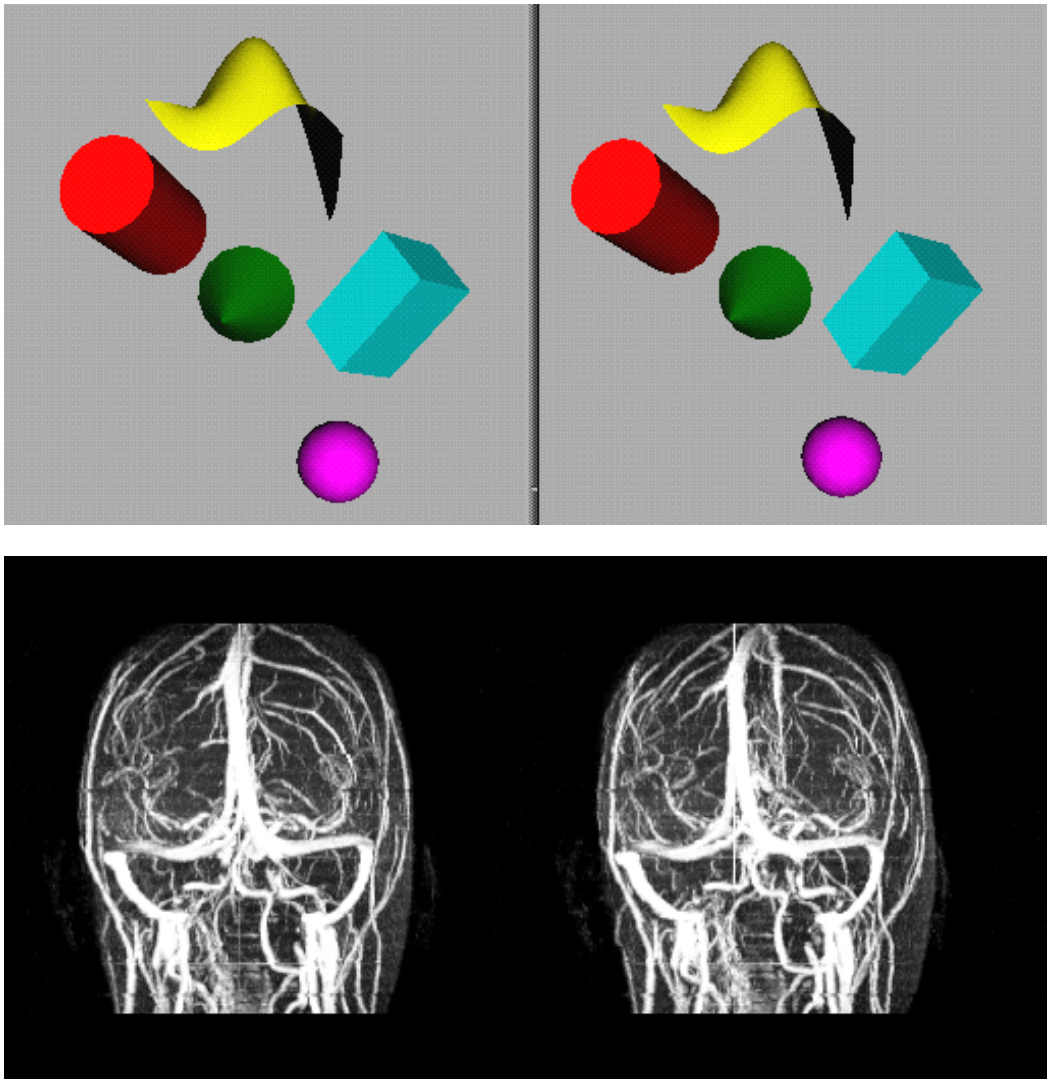


Abbildung 14: Beispiele klassischer Stereobilderpaare
a.) Computergeneriert aus Open Inventor
b.) Aus medizinischen Bilddaten (MRI T1/T2)
Die Bilder sollten aus ca. 40 cm Entfernung betrachtet werden.

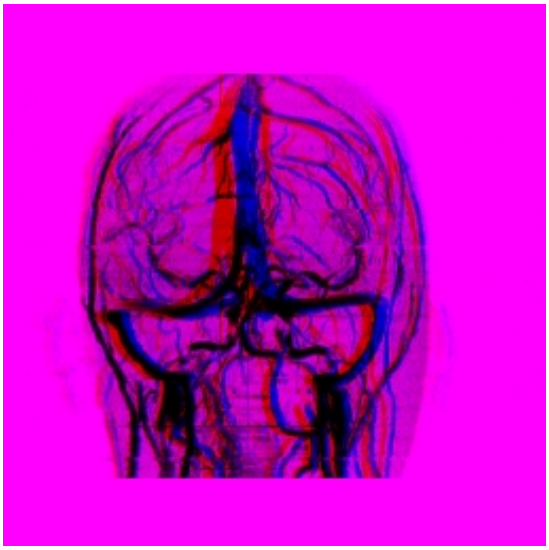


Abbildung 15: Beispiel eines Rot/Blau-Anaglyphenbildes

3.2.4 Bewertung in der Praxis

Die ersten Eindrücke bei den vom Verfasser erzeugten Anaglyphenbildern erbrachten zwar einen Zusatzgewinn in bezug auf die räumliche Tiefe, aber die Überlagerung des blauen und roten Halbbildes führte leider nicht zu einem "sauberen" grauen Vollbild. Dieses Verfahren kann deshalb nicht angewendet werden, wenn der Grauwert eine wesentliche Informationsquelle ist (Beispiel: Schatten im Lungengewebe einer CT-Aufnahme). Bei Angiographieaufnahmen kann es jedoch eingesetzt werden.

Es dürfen (wie beim realen Sehen) keine Gegenstände der Szenen zu nahe im Vordergrund stehen, da das menschliche Auge sonst die Objekte nicht mehr zur Deckung bringen kann und Doppelbilder entstehen. Auf diesen Sachverhalt wurden die Szenen abgestimmt. Trotzdem klagten Probanden, die längere Zeit Stereoszenen betrachteten, über Kopfschmerzen und Überanstrengung durch die rasche Ermüdung der Augenmuskulatur[12]. Dies liegt einerseits am verwendeten symmetrischen Kameraöffnungswinkel, aber auch an der Unnatürlichkeit der Situation. Beim Betrachten einer natürlichen Szene wechselt der Blickpunkt stetig, was jeweils eine Neuberechnung der Szene notwendig macht. Auch der Einsatz eines konventionellen Eyetrackers führt nicht zum Erfolg, da das Auge sprunghafte Bewegungen, sogenannte Saccaden, macht, welche bisher nicht aufgelöst werden, da nur eine gemittelte Augenposition gemessen werden kann. Für alle VR-Systeme, die es heute gibt, gilt, daß die Akkomodation und die Konvergenz entkoppelt sind. Das Bild auf dem Monitor hat einen anderen Abstand,

[12] Auf den Großbildflimmer-Effekt soll im näheren nicht eingegangen werden, da dieser nichts mit der eigentlichen Stereowiedergabe zu tun hat.

als das Gehirn aus dem virtuellen Bild der beiden Augen ermittelt hat. Die Krümmung der Augenlinse (Akkomodation) wird konstant gehalten, während die Blickachsen konvergieren oder divergieren müssen, um die Objekte, die verschiedene räumliche Tiefen besitzen, auf der Netzhaut zu fusionieren. Bei stereoskopischen Darstellungen konvergieren die Augen entsprechend der Darstellungsparallaxe, sie akkomodieren aber auf die Bildschirmoberfläche. Bei monoskopischen Darstellungen ist dies hingegen nicht der Fall, da auf die Bildschirmoberfläche konvergiert und akkomodiert wird. Die Verwendung des Logitech-Headtrackers zusammen mit dem Stereomode änderte wider Erwarten nichts. Es verbessert sich zwar der Eindruck, in einer 3D-Welt zu sein, aber die Symptome verstärken sich. Die Ursachen sind darin zu sehen, daß sich widersprechende Informationen von Kopf- und Augenposition mit dem gesehenen Bild vorliegen. Die Kopfposition entspricht eben nicht der Augenposition und schon gar nicht dem Blickpunkt in der Tiefe. Dazu kommt noch ein zweiter Effekt, der gemeinhin als "Seekrankheit" (engl. Motion Sickness) bezeichnet wird. Er wird dadurch ausgelöst, daß das vestibuläre System (Gleichgewichtssinn) und das visuelle System sich widersprechende Informationen liefern. Bewegt sich ein Benutzer mittels Maus durch die künstliche 3D-Welt, berechnet der Computer eine veränderte Perspektive. Das visuelle System erwartet nun aber eine Körperbewegung, die nicht stattfindet. Zum zweiten kann die Zeitverzögerung zwischen den Kopfbewegungen des Benutzers und der veränderten, vom Computer generierten Perspektive ebenfalls Konflikte auslösen. Die Schwelle liegt bei ca. 50 ms, welche der Mensch noch wahrnehmen kann.

Das Frauenhofer Institut für Arbeitswirtschaft und Organisation in Stuttgart setzt ein modernes CAVE™-System für die 3D-Visualisierung ein, aber auch da gibt es die gleichen Probleme, wie der Verfasser bei einem Besuch selbst miterleben und in Erfahrung bringen konnte.

Nur ein echtes 3D-Szenen generierendes Gerät wie Omniview oder ein zukünftiges, noch in weiter Ferne liegendes, holografisches Ausgabegerät, kann eine längere Betrachtung ohne Nebenwirkungen erlauben. Die folgende Tabelle gibt eine Übersicht über verschiedene Situationen und die dadurch auftretenden Konflikte wieder.

Situation	**Konflikte**
Grundsätzliche Betrachtung mit bisherigen kommerziellen 3D-Geräten	• Akkomodation und Konvergenz entkoppelt

Betrachtung von fester Position aus ohne Navigation in der Szene	• Konvergenzbewegung erwartet neue räumliche Szene • Augen- und Kopfbewegung erwartet neue räumliche Szene
Betrachtung von fester Position aus mit Navigation in der Szene	• Visuelles System und vestibuläres System entkoppelt
Betrachtung mit wechselnder Position mit Navigation in der Szene	• Visuelles System und vestibuläres System teilweise entkoppelt je nach Eingabemedium: Joystick, Gesten: total entkoppelt Headtracker: moderat entkoppelt außer bei ruckhaften Änderungen • Zeitliche Verschiebung zwischen erwartetem und tatsächlich ausgegebenem Bild • Räumliche Positionsungenauigkeit zwischen erwartetem und tatsächlich ausgegebenem Bild

Tabelle 2: Zusammenhang zwischen verschiedenen Situationen und den zu beobachtenden Konflikten.

3.2.5 Integration von Stereo-Texturen und 3D-Volumendaten

In der Medizin spielen räumliche Gegebenheiten zum Beispiel bei der Diagnostik und Operationsplanung eine entscheidende Rolle. Deshalb soll in den nächsten zwei Unterkapiteln kurz aufgezeigt werden, wie Stereo-Texturen oder Volumendaten integriert werden können. Die Quellen von dreidimensionalen medizinischen Daten sind Computertomographen (CT), Kernspintomographen (MRI), Positronen-Emmissions-Tomographen (PET) oder 3D-Ultraschallgeräte.

3.2.5.1 Stereo-Texturen

Eine besondere Behandlung erfordert die Integration von Stereo-Texturen, da für jedes Auge eine extra Textur gehalten werden muß. Zwei Lösungsansätze sind hierfür denkbar:

1.) Es gibt zwei separate 3D-Szenen für die linke bzw. rechte Kamera mit den korrespondierenden Texturen.

2.) Es gibt eine 3D-Szene mit beiden Halbbildern der Textur, welche abwechselnd entsprechend der auszugebenden Kamera angezeigt werden.

Durch den geringen Speicherverbrauch ist der zweiten Methode klar der Vorzug zu geben. Keine konventionelle Rendersoftware sieht bisher die Verwendung von Stereo-Texturen vor.

3.2.5.2 3D-Volumendaten

3D-Volumendaten erfordern einen großen Platzbedarf. Beispielsweise braucht man bei einer Auflösung von 256^3 (=16M Voxel) und bei 32 Bit Farbtiefe (RGBA) schon 64 MB Speicher. Ferner ist hochspezielle Hardware mit 3D-Hardwaretexturing und Alpha-Blending nötig, um diese Daten in Echtzeit ausgeben zu können. Vertreter solcher Hochleistungsgraphikrechner sind beispielsweise die Reality-Engine, die neuere Onyx2 InfiniteReality von Silicon Graphics oder PC-basierte Systeme mit Spezialgrafikhardware, die für reines Volumenrendering ausgelegt sind [KG95]. Enthält nun eine 3D-Szene zusätzlich zum Voxelmodell auch noch Oberflächen oder andere 3D-Objekte, so sind alle Objekte entweder einer einheitlichen Beschreibung zuzuführen und einheitlich, zum Beispiel durch Ray-Casting, zu visualisieren [KA93], oder aber die Objekte werden durch separate Prozesse verarbeitet und zu einem späteren Zeitpunkt zusammengefaßt. Der letztere Ansatz hat das Problem, daß normalerweise keine Wechselwirkungen zwischen Volumen und Oberflächen berücksichtigt werden. Eine Ausnahme hiervon ist der Hybrid-Ray-Tracing Ansatz von Levoy, bei dem ein Strahl gleichzeitig durch den Volumen- und den Oberflächendatensatz geschickt wird, und anschließend die beiden Strahlen kombiniert und gemeinsam ausgewertet werden (ray-merging) [LM90]. An eine direkte Integration von einem oder mehreren Voxeldatensätzen in 3D-Szenen kann durch die hohe Leistungsanforderung zum gegenwärtigen Zeitpunkt nicht gedacht werden, und für eine Stereovisualisierung bräuchte man dazu noch die zweifache Leistung. Denkbar wäre aber spezielle Hardware, die im Hintergrund die zwei Stereoansichten in Form von Texturen von den Volumendaten berechnet und diese an die Szene weitergibt. Die andere Methode, eine Oberflächenextraktion aus den Voxeln durchzuführen und die so gewonnenen triangulierten Polygonnetze in die Szene einzubinden, erfordert einen höheren Vorverarbeitungsaufwand. Sollten die relevanten Daten die der Oberfläche sein, so bietet diese Methode den Vorteil der einfachen Integration, eines geringen Platzbedarfs und einer schnellen Ausgabefähigkeit auf konventionellen Systemen. Auf die Möglichkeit der Anzeige eines Polygonnetzes mit unterschiedlichen Auflösungen[13] von einem Objekt, dessen Auflösung abhängig vom Betrachtungsabstand ist, sei kurz hingewiesen. Mitsubishi entwickelt zur Zeit eine 3D-Volumen Graphikzusatzkarte mit dem speziellen Algorithmus EM-Cube, basierend auf Cube-4. Dieser soll bei 256^3 Auflösung mit 16 Bit Farbtiefe 30 Bilder pro Sekunde liefern. Die Kombination der Pipelines der OpenGL-gerenderten Szene mit der Volumengraphik der EM-Cube Karte ist angedacht [OP97]. Volumengraphik bringt dann Vorteile, wenn die Szene aus sehr vielen kleinen Polygonen (kleiner als 1 Pixel) oder aus sehr irregulären Objekten, die

[13] Verschiedene Techniken wie Level of Detail, Multiresolution oder Progressive Refinement können hierfür verwendet werden.

analytisch schwer zu beschreiben sind, besteht, oder die Szene aus Objekten besteht, die unter der Oberfläche untersucht werden sollen. Auf der anderen Seite ist der Nachteil reiner Volumengraphik der enorme Speicherverbrauch. Angenommen aus einer Szene mit 2048^3 soll nur ein Bereich 64^3, dieser aber hochauflösend, dargestellt werden, so sollte dieses Teilvolumen wiederum hochauflösend hinterlegt sein, damit genauere Details erkennbar sind. Bedingt durch die erwähnten großen Datenmengen eines Volumenmodells sind auch die Anforderungen an die Rechnerleistung gegenüber dem reinen Oberflächenmodell sehr hoch. Einfache Operationen im dreidimensionalen Raum und Interaktionen führen zu langen Verarbeitungszeiten und verhindern den praktikablen Einsatz speziell im Falle von interaktiven Systemen. Ein weiteres Problem, wie Animationen auf Voxelbasis erfolgen sollen, befindet sich erst im Forschungsstadium.

Eine alternative Betrachtungsweise, erweitert in Analogie zu OpenGL, stellte Silicon Graphics mit der OpenGL Volumizer API vor [GH98]. Neben der kleinsten Einheit des Volumens, dem vorgestellten Voxel, wird hier, als nächst höheres fundamentales Volumenprimitiv, der vierseitige, dreidimensionale, massive Tetraederkörper benutzt. Die Tetraeder beschreiben die Geometrie, die Voxel das Erscheinungsbild (engl. Appearance). Dem Tetraeder können Eigenschaften wie Farbe, Normalen oder 3D-Textur zugeordnet werden. Der Hauptunterschied zur oberflächenbasierten Darstellung liegt darin, daß jeder Punkt im eingeschlossenen Volumen des Tetraeders mit den Eigenschaften verknüpft ist und nicht nur jeder Punkt auf der Oberfläche des Dreiecks wie beim oberflächenorientierten Ansatz. Ein beliebiges Objekt wird also in Tetraeder zerlegt und diese wieder in Schichten von transparenten, texturierten Polygonen. Diese Zerlegung ist sichtabhängig. Zur Darstellung eines Volumenobjekts wird nun konventionelles TextureMapping der polygonisierten, texturierten Flächen der einzelnen Schichten eingesetzt. Dieser Ansatz der Benutzung von Tetraedern gestattet eine Deformation der Volumengestalt sowie das Spezifizieren des Interessenbereichs (Volume-of-Interest kurz: VOI) mit willkürlich geformter Flächengeometrie. Und da alle Objekte, egal ob volumetrisch oder geometrisch, in Polygone umgewandelt werden, können beide Objekttypen in derselben Szene gemischt werden.

4 Ein- und Ausgabegeräte

4.1 Allgemein

Um die Interaktion im 3D-Raum zu erweitern und zu verbessern, wurden neue Ein- und Ausgabegeräte entwickelt. Zu Beginn dieser Arbeit wurden Literaturstudien dahingehend betrieben und einige dieser Geräte mit Prototyp-Applikationen auf ihre Tauglichkeit geprüft. Es ging hier nicht um die Eingabe von 3D-Koordinaten, wie beispielsweise bei CAD-Anwendungen, sondern um die Navigation im Raum sowie die Fähigkeit, Interaktionen mit Objekten durchzuführen. Auf die Technik, Anwendung und Bewertung einiger ausgewählter Geräte wird kurz eingegangen. Es handelt sich hier also um einen Auszug der als wichtig erachteten Geräte und erhebt nicht den Anspruch auf Vollständigkeit. (Weitere Literatur: [TJ97, GJ93])

4.2 Softwaretechnische Einbindung von 3D-Geräten in kommerziellen Umgebungen

Unabhängig vom 3D-Eingabegerät muß festgestellt werden, daß bei kommerziellen Windows-Betriebssystemen ein Fenster mit der Maus erst aktiviert werden muß, um 3D-Geräteereignisse empfangen zu können. Dies geschieht entweder dadurch, daß die Maus in ein Fenster bewegt wird (UNIX) oder aber das Fenster angeklickt werden muß (Windows). Somit muß wieder mit zwei Eingabegeräten gearbeitet werden, was für die Benutzer umständlich ist. Ferner kann ein 3D-Gerät nicht einfach die Mausfunktion übernehmen, um beispielsweise ein Menü anzuklicken. Ausnahme: In dem Sonderfall, daß es nur eine Hauptanwendung gibt, die auch nur in einem Unterfenster 3D-Ereignisse verarbeitet, können die Ereignisse über Streams (Dateidescriptoren) geleitet werden. Unabhängig davon, wo sich die Maus befindet, empfängt die Applikation die 3D-Geräteereignisse. Von einer Integration kann bisher also nicht gesprochen werden. Dies zeigt sich auch daran, daß bei Windows und SGI-IRIX die Hauptschleife (engl. Mainloop) abgeändert werden muß, indem eine Funktion hinzugefügt wird (hook in), damit überhaupt 3D-Geräteereignisse empfangen und an die Applikation verteilt werden können.

4.3 Spaceball

Mit dem Spaceball kann Translation und Rotation gleichzeitig ausgeführt werden. Er erlaubt alle 6 möglichen Freiheitsgrade (Degree-of-Freedom kurz: DOF) einer Bewegung. Wie auf Abbildung 16 zu erkennen, besteht der Spaceball aus einem kugelförmigen Ball, welcher relativ fest sitzt. Der Spaceball mißt die Druckstärke und das Drehmoment, welche auf den Ball einwirken, was eine minimale Auslenkung hervorruft. Gemessen wird durch piezokeramische Drucksensoren. Für den Benutzer scheint die Kugel während der Manipulation unbeweglich. Sobald der Spaceball losgelassen wird, kehrt er wieder in die Ruheposition zurück. Die Translationen und Rotationen werden relativ zur Ruheposition gemessen. Durch die minimalen Auslenkungen bzw. die hohe Empfindlichkeit, haben die Benutzer am Anfang große Probleme im Umgang mit dem Spaceball. Ebenso ist es schwierig, nur reine Translationen oder Rotationen auszuführen.

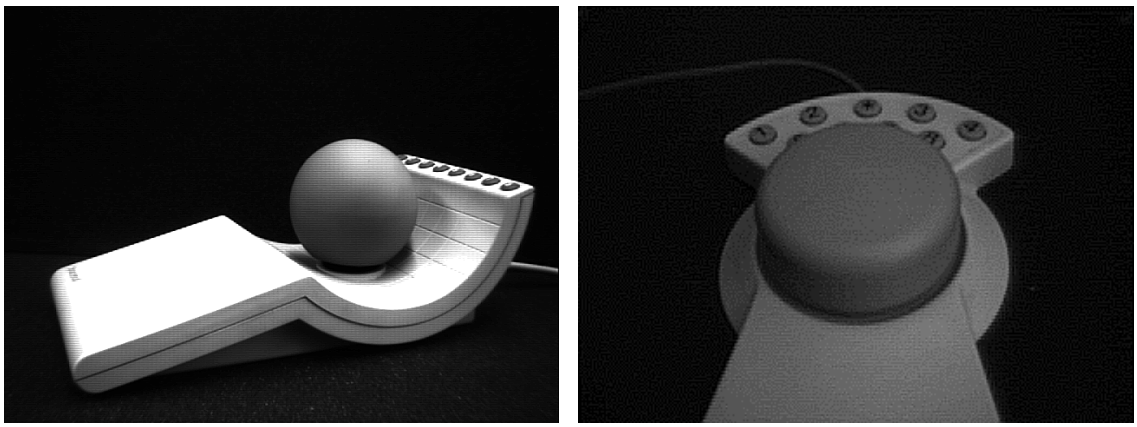


Abbildung 16: a.) Links ist ein Spaceball und rechts eine Spacemouse dargestellt

4.4 Spacemouse

Bei der Spacemouse gilt das gleiche wie beim Spaceball. Im Unterschied zu diesem fallen die Auslenkungen größer aus, das Feedback zum Benutzer ist etwas besser. Der Ball des Spaceballs wird hier durch eine Kappe ersetzt. Diese Konstruktion führt dazu, daß eine Rotation unterschiedlich gehandhabt wird. Die Rotationen um die x- und z-Achse werden durch Kippen der Kappe ausgelöst, während eine Rotation um die y-Achse einer richtigen Rotation der Kappe entspricht. Die Rotationen und Translationen werden durch 6 LEDs mit Schlitzblenden unter Zuhilfenahme einfacher Zeilenkameras (lineare Positionsdetektoren) gemessen.

4.5 Positiontrackergeräte

Positiontracker sind ein wichtiges Eingabemedium für Virtuelle Umgebungen. Ihre Leistungsfähigkeit wird mit folgenden Parametern beschrieben:

- Auflösung: Kleinste Positionsänderung, die festgestellt werden kann.
- Genauigkeit: Abweichung der gemessenen Position zur tatsächlichen.
- Updaterate: Geschwindigkeit, mit der die Meßdaten vom Tracker an den Computer übermittelt werden können.
- Latenz: Verzögerungszeit zwischen tatsächlicher Änderung und der an den Computer gemeldeten Änderung.
- Arbeitsvolumen: Raum, in welchem der Tracker mit der vorgegebenen Genauigkeit und Auflösung arbeitet.

Je nach angewandter Technik wird diese Gerätegruppe eingeteilt in magnetische, akustische, optische, mechanische und trägheitsbasierte Tracker. Der am häufigsten verwendete magnetische Tracker sowie der am WSI/GRIS verwendete akustische Tracker werden kurz vorgestellt. Trackergeräte werden in Kombination mit anderen 3D-Geräten eingesetzt. Der vom Lehrstuhl verwendete Logitech Headtracker ist mit der StereoGraphics Stereobrille verbunden. An der Brille ist an drei Positionen jeweils ein Ultraschall Empfänger angebracht. Über dem Bildschirm befindet sich ein Dreieck, an dessen Ecken jeweils ein Sender montiert ist, welche abwechselnd einen Impuls aussenden. Aus den Laufzeitunterschieden von Sender zu Empfänger wird die Position und Orientierung bestimmt. Der Tracker liefert absolute Translation und Rotationsdaten. Die Tracker müssen vor der Benutzung kalibriert werden. Der Einsatz von Ultraschall bringt eine gewisse Störanfälligkeit (Hindernis zwischen Sender und Empfänger), Einschränkung der Bewegungsfreiheit und Ungenauigkeit mit sich. Neben diesem kostengünstigen Verfahren, welches auch bei Logitechs 6D-Maus verwendet wird, gibt es noch aufwendigere und genauere Systeme. Polhemus verwendet als Sender drei senkrecht aufeinanderstehende Sendespulen, die permanent elektromagnetische Wellen unterschiedlicher Frequenz aussenden. Das hervorgerufene gemultiplexte Magnetfeld wird wiederum durch die passiven Empfangsspulen des Trackers gemessen. Diese Sender wurden auch in einen 3D-Joystick namens Spacestick des Herstellers Virtual Presence eingebaut. Die Firma Polhemus ist der Marktführer bei Headtrackerprodukten (Ultratrack Pro, FastTrack, IsoTrack), die Reichweite variiert je nach Modell von 1,5-7 m. Die Genauigkeit nimmt allerdings mit wachsender Entfernung zum Sender ab, und bei großen Abständen äußern sich diese

Ungenauigkeiten in Zitterbewegungen. Heutige Systeme wie Ultratrack Pro erlauben das Überwachen von komplizierten Bewegungsabläufen, wie Tanzen, Laufen oder Springen und noch fortschrittlichere, wie Star*Track (ebenfalls von Polhemus), die Aufzeichnung von Interaktionen zwischen zwei Teilnehmer. Diese Systeme werden neben dem VR-Bereich auch in der Filmindustrie zum Digitalisieren von Bewegungen (engl. Motion-Capture) eingesetzt. Zu den mechanischen Trackern gehören die zwei Systeme BOOM3C und FS² (Fakespace Simulation System). Der Aufbau sieht derart aus, daß sich am Ende eines mechanischen Arms, welcher mit Gegengewichten ausgestattet ist, ein Sichtgerät (HMD) befindet. Im Gegensatz zu Stereobrille und HMD, die normalerweise während der Arbeitssitzung permanent verwendet werden, greift bzw. befestigt der Benutzer das Display nur bei Bedarf vergleichbar also einem Fernglas auf einem Stativ. Der Arm hat sechs orthogonale Gelenke, welche mit Potentiometern ausgestattet sind, um die sechs Freiheitsgrade zu messen. Dieses System ist sehr genau und robust gegen Störungen im Gegensatz zu magnetischen und akustischen Trackern (Sichtbehinderung Emitter und Empfänger, Magnetische Objekte, Überlagerung von Schwingungen). Allerdings ist das Arbeitsvolumen klein und nicht alle Positionen sind zu erreichen. Überdies stört die hohe Trägheit die Benutzer beim Ausüben ihrer Arbeit.

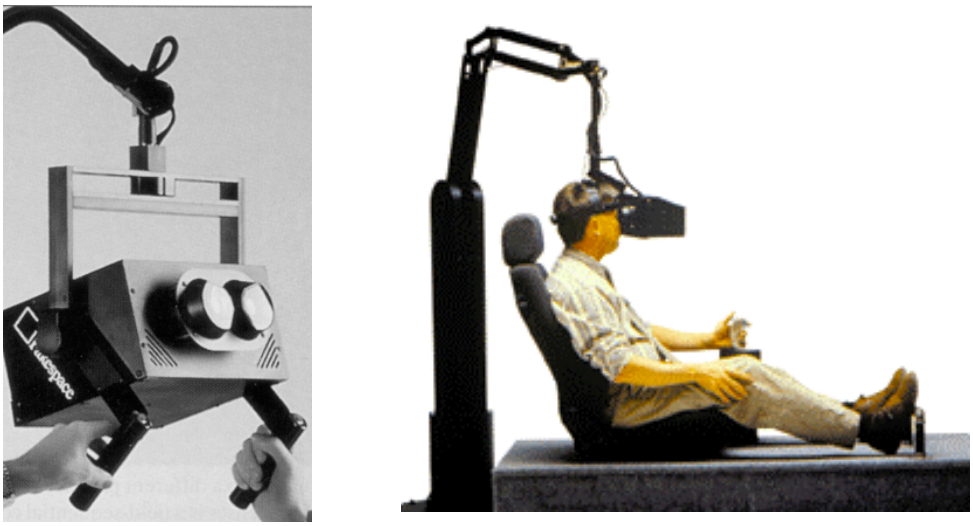


Abbildung 17: Fakespace BOOM3C mit zwei integrierten Farbbildschirmen (Auflösung 1280x1024). Dieser ist befestigt an einem Arm mit Gegengewichten. Der Arm enthält Sensoren für alle 6 Freiheitsgrade enthält.
(BOOM steht für die Abkürzung Binocular Omni-Orientation Monitor)

4.6 Datenhandschuh

Schon in den siebziger Jahren wurde der Versuch unternommen, das wohl universellste Werkzeug, die menschliche Hand, als Eingabemedium zu nutzen [SZ94]. Der inzwischen weitverbreitete DataGlove der Firma VPL Research wurde 1987 erstmals vorgestellt [ZL87]. Dieser Stoffhandschuh ist ausgestattet mit speziellen Glasfasern (Flexsensoren) welche die Messung des Beugungswinkels für jedes Fingergelenk zwischen 0° und 90° erlauben. Die Lichtintensität am einem Ende der Glasfasern wird mittels Fototransistor gemessen, am anderen Ende befindet sich eine Leuchtdiode. Über dem Gelenk sind die Glasfasern angerauht und je stärker die Beugung, desto mehr Licht tritt aus. Aus dem gemessenen Restlicht wird der Beugungswinkel errechnet. Es gibt Modelle, die auch die Spreizung der Finger erfassen können. Da keine Hand der anderen gleicht, muß der Datenhandschuh für jeden Benutzer neu kalibriert werden. Normalerweise ist auf dem Handrücken noch ein Positionstracker angebracht, damit Handposition und Orientierung im Raum ermittelt werden können. Aufgrund der Konstruktion eignet sich der DataGlove auch für die Eingabe von Gesten. Der mechanische Tracker HandMaster wurde an der Universität Utah erfundenen und von der Firma EXOS, inzwischen von Microsoft aufgekauft, weiterentwickelt. Der HandMaster wird wie ein Exoskelett[14] über die Hand gestreift. Dieses teure Produkt ist sehr genau und schnell und wird u. a. für Telemanipulationszwecke eingesetzt. Die Firma Virtual Technologies bietet den Datenhandschuh CyberGlove mit bis zu 22 Freiheitsgraden an. Zum Vergleich: Die Gelenke der menschlichen Hand besitzen 23 Freiheitsgrade, zusammen mit den 6 Freiheitsgraden der Handposition und Handorientierung ergibt dies 29 Freiheitsgrade. Wird der CyberGlove mit einem Positionstracker kombiniert, erreicht man 28 von 29 möglichen Freiheitsgraden.

Abbildung 18: Verschiedene Modelle von Datenhandschuhen: Exos Dexterous HandMaster, VPL DataGlove, Virtual Technologies CyberGlove (von links nach rechts)

[14] Ein Exoskelett ist eine äußere Stützkonstruktion. Bei Insekten, Krebsen und Spinnen ist dies der Chitinpanzer.

Es gibt auch ein System, bei dem sich der ganze Körper in einem "Datenanzug" befindet, um so alle Bewegungen des Körpers erfassen und in die Virtuelle Umgebung umsetzen zu können. Die Firma VPL bietet einen Datenanzug (engl. datasuite) an, der 68 Gelenkwinkel und somit 74 Dimensionen als Eingabe liefert.

4.7 Haptische Geräte

Alle vorgestellten Eingabegeräte haben einen entscheidenden Nachteil. Es gibt keine Rückkopplung zum Benutzer, wenn dieser ein Objekt manipuliert. Über die Plastizität, Oberflächenbeschaffenheit oder den Widerstand des Objektes erhält der Benutzer keinerlei Information, was z. B. im medizinischen Bereich bei einer OP-Simulation, wo der Tastsinn eine große Rolle spielt, äußerst wichtig ist. Erste Studien mit haptischen Systemen, welche den taktilen Feedback untersuchen, wurden gemacht [IN93, RA93, GJ93]. Der oben genannte Datenhandschuh EXOS Dexterous HandMaster kann zusätzlich um ein Kraftrückkopplungs-system (Hand Exoskeleton Haptic Display: HEHD) erweitert werden, welches das auch separat erhältliche vibrotakile Touch-Feedbacksystem (Touchmaster) enthält. Letzteres arbeitet mit Vibrationen im Bereich von 210-240 Hz an allen 4 Fingerspitzen und dem Daumen. Eine andere Methode einfacher Force-Feedback-Systeme, Druck in Handschuhen zu erzeugen, besteht im Aufblasen von kleinen Luftkammern [YJ96]. Das Exoskelett eignet sich gut für die Kraftrückkopplung, da sich die Bewegung der Hand in bestimmte Richtungen sperren läßt. Diese Methode wird ebenfalls angewandt beim Produkt PHANToM von SensAble Technologies, welches auf einer Arbeit am MIT aufbaut [MS94]. Daumen und Zeigefinger kommen in eine Art Fingerhut, der an die Rückkopplungseinheit angebunden ist. Die Auflösung liegt bei 0,03 mm und die Kraft bei maximal 8,5 N. Das Gerät wird für die realistische Simulation von chirurgischen Eingriffen durch dehnbare Oberflächen eingesetzt. Das vorgestellte BOOM3C und FSS² System kann durch Schrittmotoren in den Gelenken ebenfalls für die Kraftrückkopplung verwendet werden. Neuerdings werden spezielle Eingabegeräte auf dem medizinischen Sektor entwickelt, welche eine Kraftrückkopplung (Force-Feedback) erlauben. Beispielsweise für eine realistische Simulation von minimal-invasiven Eingriffen, zum Legen eines Katheters und zum Üben des Einführens von Nadeln bietet die Firma Immersion Corporation Force-Feedback Geräte an [IC97]. Haptische Geräte lassen sich nicht in reine Ausgabe, bzw. Eingabegeräte klassifizieren, sondern integrieren oft beide Eigenschaften.

4.8 Stereobrille

Die Stereobrille eignet sich zur Darstellung von räumlichen Szenen am Bildschirm ebenso wie auch bei Großbildschirmprojektionen. Die Technik besteht aus einem Infrarotsender, der am Computer angeschlossen wird und synchron zu den Halbbildern Signale zur Brille sendet. Die Brille besitzt einen Flüssigkristall-Verschluß. Wird am Monitor das Bild für das rechte Auge angezeigt, so ist nur das rechte Glas durchsichtig. Beim Bild für das linke Auge verhält es sich entsprechend umgekehrt. Bei der von uns verwendeten StereoGraphics Stereobrille kann maximal eine Frequenz von 144 Hz erreicht werden. Die Grafikkarte muß in der Lage sein, die Bilder non-interlaced darzustellen. Der Vorteil dieser Technik besteht in der hohen Auflösung von 1280x512. Ferner können mehrere Benutzer, ausgestattet mit einer Brille, das Stereobild betrachten. Die Stereobrille läßt sich mit einem Headtracker kombinieren. Deering stellte 1992 ein solches System vor [DM92]. Haben zwei Personen einen Headtracker, so fällt die Frequenz der Stereohalbbilder schon auf 36 Hz, und die Symptome wie Schwindel und Kopfweh werden häufiger. Für drei oder mehr Benutzer, wobei für jeden ein perspektivisch richtiges Stereobildpaar generiert wird, ist dieses System der zeitlich alternierenden Bilder ungeeignet. Ergänzend sei angemerkt, daß auch ein sehr leistungsstarker Computer für die Generierung

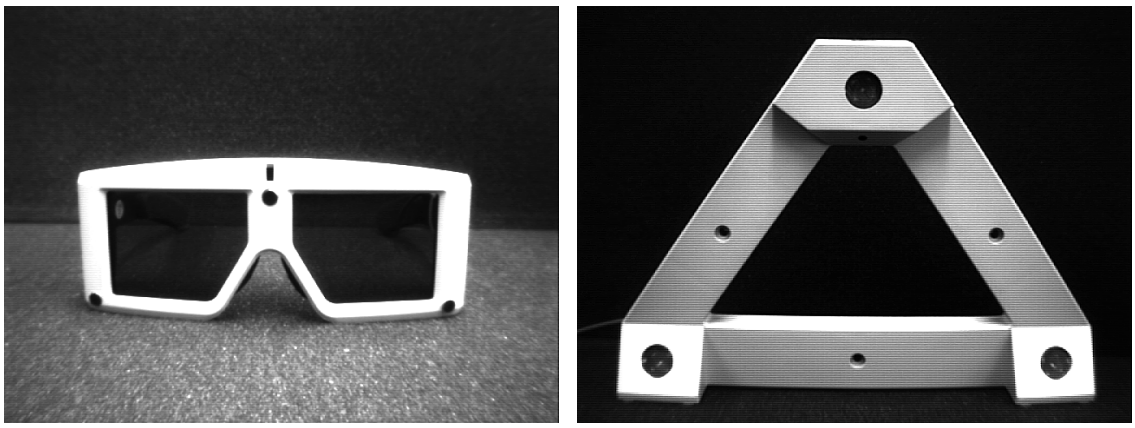
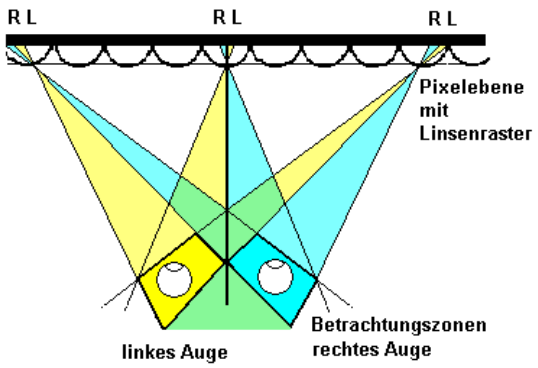


Abbildung 19: Links ist eine StereoGraphics CrystalEyes Shutterbrille und rechts ein Logitech Ultraschall Headtracker dargestellt.

der unterschiedlichen Stereoansichten nötig ist. Am Lehrstuhl verwenden wir das CrystalEyes VR System bestehend aus StereoGraphics CrystalEyes-Stereobrille und Logitech Ultraschall-Headtracker. Dieses System vermittelt zwar einen leichten "Look around" Effekt, es kommt jedoch nicht an das "Eintauchgefühl" von Headmount Display heran, da der Bewegungsraum des Kopfes sehr beschränkt ist, weil das Bildschirmbild noch im Fokus bleiben muß.

4.9 Autostereoskopische Displays

Bei den Autostereoskopischen Displays handelt es sich um Bildschirmtechniken, die die Betrachtung von 3D-Bildern ohne zusätzliche Hilfsmittel wie Farb-, Polarisations- oder Shutterbrille erlauben. In aller Regel kommen hier nur LCD-Bildschirme zum Einsatz. Durch spezielle Techniken wird erreicht, daß das linke beziehungsweise rechte Teilbild in das jeweilige



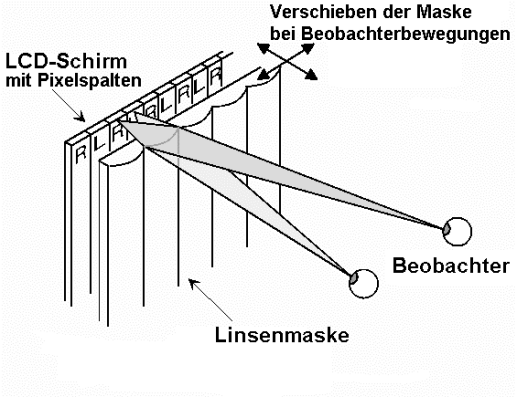
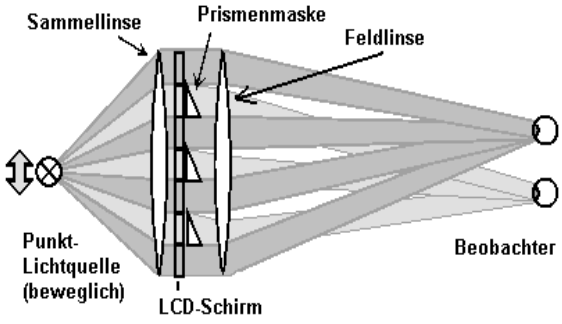
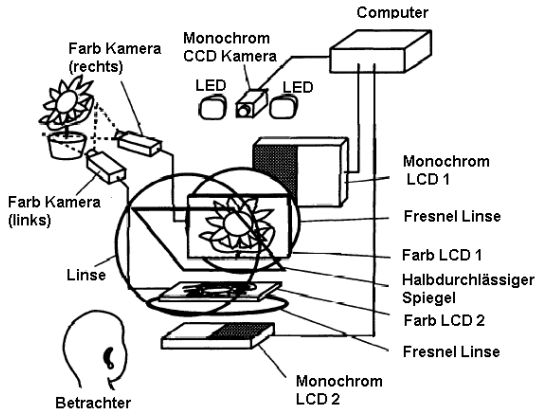


Abbildung 20: a.) Tracking-Systeme müssen bei Rasterbildschirmen die Betrachtungszonen den Kopfbewegungen nachführen. Dabei können zusätzlich die neuen korrekten Perspektiven errechnet werden.

b.) Prinzipskizze eines 3D-LCD-Displays, bei dem die Linsenraster bei Bewegung des Betrachters mechanisch verschoben werden.





c.) Schema des Dresdner Display Prototypen bei dem die Nachführung des Bildes durch eine optoelektrisch entgegengesetzte Verschiebung der Lichtquelle zum Beobachter erfolgt. Die Beleuchtung des LCD-Schirm wird durch paralleles Licht erzielt.

d.) Schema des Tekumo Displays. Die Überlagerung der Teilbilder gewährleistet die volle Pixelauflösung. Mittels monochromen LCD Displays wird die Teilbeleuchtung realisiert.

Quelle: Heinrich-Hertz-Institut a.)- c.) [PW97], d.) [OS95]

Auge geleitet wird [KJ95]. 1995 stellte Sanyo ein Verfahren unter der Bezeichnung ‚Image Splitter' vor, welches bereits seit Anbeginn dieses Jahrhunderts bekannt ist. Dabei wird ein Stereobild aus schmalen vertikalen Streifen mit einem Pixel Breite aus zwei Ansichten generiert. Das Bild besteht also abwechselnd aus den Streifen der linken Ansicht und anschließend der rechten und so weiter. Durch eine Streifenmaske, angebracht vor dem

Bildschirm, wird auf mechanische Weise erreicht, daß die Streifen des jeweiligen Halbbildes in das richtige Auge gelangen und somit der Tiefeneindruck des Stereobildpaares durch das Gehirn generiert wird. Nachteil dieses starren Verfahrens ist, daß nur ein Benutzer in einer ganz bestimmten Entfernung und einem ganz engen Positionsraum einen Stereoeindruck gewinnen kann. Auch andere Verfahrensweisen nach dem gleichen Prinzip, aber ohne Streifenmaske wurden schon versucht. Wie beispielsweise die Variante der Firma Terumo, die zwei TFT-LCDs zur Darstellung verwendete, welche durch einen Halbspiegel kombiniert wurden (bei Beibehaltung der vollen Spaltenauflösung). Zwei zusätzliche monochrome LCDs waren wiederum für die Streifenbeleuchtungen hinter den TFT-LCDs verantwortlich. Mittels eines Kopf-Trackingsystems wurde die 3D-Sehzone nachreguliert, indem die Bild-Displays und die Streifen-Displays gegeneinander verschoben wurden [OS95] (siehe Abbildung 20b.). Anstatt die Sicht auf eine Hälfte des Bildpaares mittels Maske einzugrenzen, gibt es aber noch die andere Möglichkeit, die Sicht gezielt zu fördern. Diese Technik ist seit 1908 auf alten 3D-Postkarten zu sehen, bei denen halbzylinderförmige Linsen (Lentikularsystem) die Teilbilder in das entsprechende Auge lenken. Dieses Prinzip läßt sich auch noch erweitern, indem mehrere Ansichten verwendet werden. Somit läßt sich das Bild aus mehreren Blickwinkeln betrachten (Parallax-Panoramagramm) oder man bekommt den Effekt einer Bewegung. 1992 stellte Sharp eine andere Technik vor, bei der bei Bewegung des Betrachters die Lichtquelle entgegengesetzt verschoben wurde und somit die Strahlenbündel wieder in seine Augen trafen. Dieses Trackingsystem reagierte sowohl, wenn der Betrachter sich dem Schirm näherte oder entfernte, als auch, wenn er sich zur Seite bewegte. 1996 stellte das Heinrich-Hertz-Institut auf der CeBit ein autostereoskopisches Display vor, welches von Carl Zeiss gebaut und von Silicon Graphics optimiert wurde. Eine ähnliche, neuere Entwicklung der TU Dresden im Jahre 1997 verwendet statt dem halbzylinderförmigen Lentikularsystem eine Prismenmaske und eine punktförmige Lichtquelle, welche mittels Headtracking nachgeführt werden [HS97]. Vorteil der Dresdener Lösung ist, daß ein herkömmliches Industrie LCD-Farbdisplay verwendet werden kann. Zur Beleuchtung des Displays wird paralleles Licht benötigt, welches durch die Punktlichtquelle und den Kollimator gewonnen wird. Die beiden Stereobilder werden vertikal interlaced dargestellt, wobei die ungeradzahligen Spalten das rechte Halbbild und die geradzahligen das linke Halbbild enthalten. Das Licht, das auf die ungeradzahlige Spalte trifft, wird durch die anschließende Prismenmaske so abgelenkt, daß es in das rechte Auge des Betrachters fällt. Analog wird das Licht der geraden Spalte in Richtung linkes Auge abgelenkt. Durch die Abbildungen 20a.) und 20c.) wird dieses Prinzip veranschaulicht.

Alle autostereoskopischen Displays erzeugen kein reelles dreidimensionales Bild, sondern eine Raumillusion. Dementsprechend kommt es zu einem Konflikt zwischen Akkomodation und Konvergenz, auf die noch näher eingegangen wird.

Von den Erfindern Eugene Dolgoff und Louis Tulloder von der Firma Floating Images wurde ein Vorsatz für Bildschirme entwickelt, der einen einfachen dreidimensionalen Eindruck ohne Akkomodations- und Konvergenzprobleme ermöglicht [FL97]. Dabei wird in einem horizontal zweigeteilten Bild oben das Vordergrund- und unten das Hintergrundmotiv gezeigt. Mittels des im Vorsatz eingebauten Linsensystems erscheint das untere Bild durch einen halbdurchlässigen Spiegel als virtuelles Bild hinter dem Vordergrundbild. Der Betrachter sieht somit zwei Bildebenen, auf die er fokussieren kann. Bewegt er seinen Kopf nach links oder nach rechts, so kann er hinter die Vordergrundobjekte blicken. Das Verfahren nutzt die Verdeckung, um einen dreidimensionalen Eindruck zu erzielen, vergleichbar mit der Scherenschnittechnik oder der Layertechnik bei der Trickfilmproduktion, nur daß der Abstand zwischen den Bildebenen beträchtlich größer ist. Die Erweiterung von zwei auf mehrere Bildebenen wird gegenwärtig untersucht, was zur weiteren Verringerung der vertikalen Auflösung führt. Ein Nachteil dieser Technik ist, daß hellere Objekte im Hintergrund durch die dunklen Flächen der Vordergrundobjekte scheinen. Es wird noch nach Wegen gesucht, Objekte, welche sich vom Vordergrund bis zum Hintergrund erstrecken, wie beispielsweise eine Wand, realisieren zu lassen. Ebenso ungelöst ist, wie sich bewegliche 3D-Objekte zwischen diesen beiden Bildebenen darstellen lassen, die diese Ebenen für eine realitätsnahe Darstellung auch durchdringen sollten. Primär interessant sein dürfte diese kostengünstige Technologie für den Computerspiele- und Videomarkt.

4.10 Head Mounted Display

Im Jahr 1960 baute Morton Heilig sein sogenanntes *Sensorama*; ein Gerät, das einem einzelnen Betrachter in einer halb offenen Kabine den Film einer Motorradtour durch Brooklyn zeigte und das richtige Gefühl dazu mittels Lüfter, verschiedenen Parfums und Vibrationen erzeugte. Das erste Head Mounted Display (HMD) wurde 1965 von Ivan Sutherland in einem Forschungsbericht erwähnt ("The Ultimate Display") [SI65]. Der erste Prototyp "Sword of Damocles" wurde 1968 vorgestellt und bestand aus je zwei monochromen Bildröhren [SI68]. Das Bild wurde durch Prismensystem und Vergrößerungsgläser betrachtet. Der damalige Stand der Technik erlaubte nur die Visualisierung von Drahtgittermodellen. Daß mit diesem Gerät auch räumliches Sehen möglich ist, beschreibt Sutherland mit dem Satz: "if we place two-dimensional images on the observer's retinas, we can create the illusion that he is seeing a

three-dimensional object" . Heutige HMDs haben eine farbige Flüssigkristallanzeige, was sie leichter, handlicher und somit besser tragbar macht. Es wird zwischen zwei Systemen unterschieden: die vollkommen geschlossenen, welche vorwiegend im Heimbereich anzutreffen sind, und die halbtransparenten, welche noch eine Durchsicht auf die Umgebung zulassen und im militärischen Umfeld eingesetzt werden. Ein weiteres Unterscheidungsmerkmal ist die Größe des binokularen Blickfelds. Die am häufigsten eingesetzte Weitwinkel *LEEP*-Optik [HE83] hat beispielsweise ein Sichtfeld von ca. 90°. Die Auflösung des Displays ist entscheidend für die Erlebnisqualität. Leider liegt diese im Heimbereich erst bei ca. 320x200Pixel (192000 Bildpunkte), was keine komplexen Darstellungen zuläßt. Daß HMDs heute mit Headtrackern eingesetzt werden, ebenso wie DataGloves oder Stereobrille, versteht sich aus dem vorher Beschriebenen. Der Benutzer soll in den computergenerierten dreidimensionalen Bildern umherwandern beziehungsweise darin eintauchen können.

4.11 Virtual Retinal Display

1991 erfand Thomas A. Furness III am Human Interface Technology Laboratory (HITH) der Universität Washington ein neuartiges Wiedergabegerät, das den Namen Virtual Retinal Display (VRD) erhielt [TJ95]. Wie der Name dieses Gerätes schon andeutet, wird das Bild mittels moduliertem Laserlicht direkt auf der Netzhaut abgebildet. Das bildgebende Verfahren der VRD baut sich aus vier Hauptkomponenten auf (siehe auch Abbildung 21):

- Aus der Bildquelle – wie dem VGA-Ausgangssignal eines Computers, einer Videokamera oder anderen bildgebenden Geräten - wird die Bildinformation pixelweise ausgelesen und die Pixelinformation gespeichert.
- Eine kohärente Lichtquelle wie Laserdioden, die mit Licht niedriger Intensität arbeiten, erzeugt einen einzelnen Pixel und übermittelt ihn durch die Pupille in die menschliche Netzhaut. Wenn statt eines monochromen Bildes ein Farbbild erzeugt werden soll, generieren drei Lichtquellen (rot, grün und blau) drei Bildpunkte, die zu einem Pixel der entsprechenden Farbe vereinigt werden.
- Horizontal und vertikal arbeitende Scanner bewegen die Lichtquelle zeilenweise schnell über die Retina und werfen so rasterweise das Bild auf die Netzhaut.
- Der Lichtstrahl wird durch optische Elemente so abgelenkt, daß auf der Netzhaut der Eindruck eines großen virtuellen Bildes erzeugt wird.

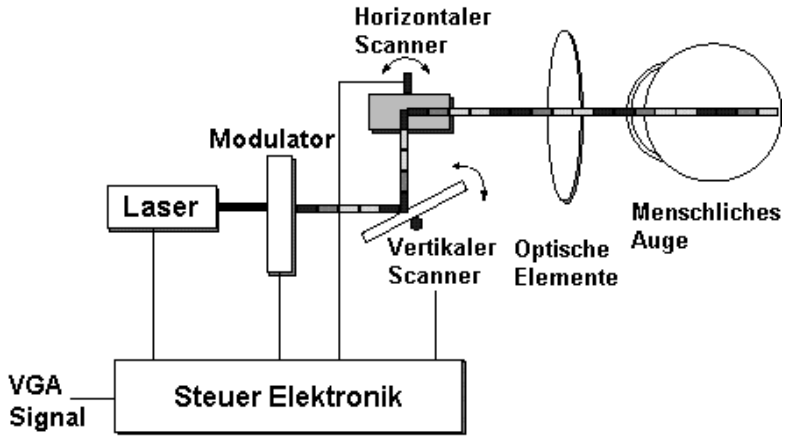


Abbildung 21: Aufbau eines Virtual Retinal Displays (nach [VP98])

Diese Technik hätte den Vorteil, daß grundsätzlich eine sehr hohe Auflösung möglich wäre. Denn die Auflösung hängt nicht von adressierbaren Pixeln, wie beim Kathoden- oder LCD-Bildschirm, ab, sondern von Beugungs- und optischen Abbildungsfehlern. Nach Aussage der Entwickler sollte auch ein sehr großes Sichtfeld (>120°) realisierbar sein. Für einen echten 3D-Eindruck wird, wie bei der Stereoprojektion, in jedes Auge ein Bild aus einer anderen Perspektive projiziert. Theoretisch wäre durch die Variation der Größe beziehungsweise Schärfe eines jeden Pixels ein unterschiedlicher Tiefeneindruck möglich. Somit sollte ein natürlicherer 3D-Eindruck entstehen. Der Konflikt von Akkomodation und Konvergenz, der bei herkömmlichen Stereobetrachtungsgeräten auftritt, würde minimiert. Eines der größten Probleme stellt die Fokussierung bei freischwebenden Bildern dar: Ist das Abbild nicht perfekt scharf oder erfolgt die Projektion nicht exakt coplanar zur Pupillenöffnung, versucht das Auge sich immer wieder auf das Objekt scharfzustellen, was nicht gelingen kann und weshalb es zu Augenschmerzen kommt. Bei Augenbewegungen treten weitere Problemen auf. Da die Austrittsöffnung des VRD bei nur 1,2 mm liegt, führen kleine Translations- und Rotationsbewegungen des Auges zu Verdeckungen des Lichtstrahls durch die Iris. Damit verschwindet das (Teil-)Bild auf der Retina. Durch die Nachführung der Austrittsöffnung des VRD gemäß der Augenbewegungen mittels Tracking soll dieser Fehler später ausgeschaltet werden. Mit dem VRD sind zwei verschiedene Modi möglich: Einmal wird nur das künstliche Bild allein, sozusagen wie in einer Dunkelkammer, betrachtet und im anderen Falle findet eine Überlagerung des Bildes mit der natürlichen Umgebung statt. Neben der Darstellung von Bildern im Rastermodus sind auch vektorgenerierte Ansichten grundsätzlich möglich. Angemerkt sei, daß bisher das VRD nur als Prototyp für industrielle und militärische Anwendungen existiert.

4.12 Großbildprojektion

Das erste System wurde vom Electronic Visualization Laboratory der Universität Illinois entwickelt und 1992 auf der ACM SIGGRAPH der Öffentlichkeit vorgestellt. 1993 beschreibt die Entwicklerin Carolina Cruz-Neira das CAVE[15]-System als eine Umgebung, die den Benutzer eintauchen läßt. Dies geschieht mit Hilfe von dreidimensionalem Tracking, Eingabegeräten zum Auswählen und Greifen (engl. picking), 3D-Audiofeedback und einer Anzahl von Projektionsschirmen, die den Benutzer umgeben und dabei einen abgeschlossenen Raum bilden [CS93]. Die Projektion erfolgt auf die drei umgebenden Wände und je nach System zusätzlich noch auf den Boden (siehe Abbildung 22a.). Auf die vierte Wand, die den Zugang bildet, und die Decke erfolgen keine Projektionen, obwohl dieses softwareseitig unterstützt würde, da der technische Aufwand für eine 6-Wände Projektion enorm hoch ist. Das CAVE™ -System wurde bisher benutzt für das Durchwandern von Räumen (engl. architectural walkthroughs), die Untersuchung von kosmischen und fraktalen Phänomenen, das Visualisieren von Algorithmen, welche auf parallelen Maschinen implementiert wurden sowie Simulationen zum besseren Verständnis von Wetter und molekularen Dynamiken [CL93]. Der Vorteil dieses Systems liegt in der hohen Auflösung. Jede Großbildschirmwand hat eine Auflösung bis zu 1024x768 Punkte in Stereo bei 120 Hz. Außerdem hat das System die Fähigkeit, den Benutzer in einem eng umschlossenen Raum zu verfolgen und ihm die Kontrolle über die Umwelt zu geben. Zusätzlich besteht die Möglichkeit, Gruppen aus mehreren Teilnehmern zu untersuchen.

Ein weiteres Projektionssystem, genannt Responsive Workbench™, wurde 1995 von dem GMD-Forschungszentrum Informationstechnik GmbH vorgestellt [KB95]. Bei diesem Ansatz wird das Bildschirmbild auf einen horizontalen, vergrößerten Arbeitstisch umgelenkt. Mehrere Personen können so mit dem projizierten virtuellen Stereobild über der Arbeitsfläche arbeiten. Zur Bewältigung der Aufgaben tragen die Benutzer Shutter-Stereobrillen und Datenhandschuhe als Hilfsmittel. Zielgruppe für dieses System sind bestimmte Applikationen aus dem Bereich Wissenschaft, Medizin und Architektur. Die Tauglichkeit dieses Systems für die Visualisierung von dynamischen Prozessen wie Herzschlag, Blutfluß oder anderen Strömungsverhalten wird gerade untersucht. Das System eignet sich für Anwendungen, die aus der Vogelperspektive zu absolvieren sind, und bei denen nicht innerhalb der Szene gearbeitet werden muß, da durch den Projektionsmechanismus sonst andere Teile verdeckt/abgeschattet

[15] Der Name "CAVE" wurde aus zweierlei Gründen gewählt. Zum einen repräsentiert er die rekursive Abkürzung für CAVE Automatic Virtual Environment und zum anderen ist er als Anlehnung an "The Smile of the Cave" aus Platos Republic, indem Philosophen die Wechselbeziehungen von Wahrnehmung, Realität und Illusion untersuchen, gedacht.

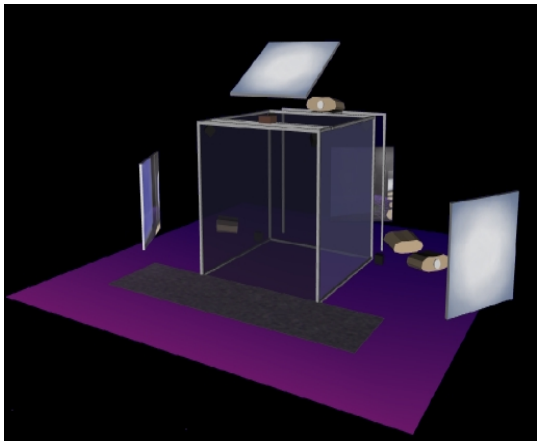
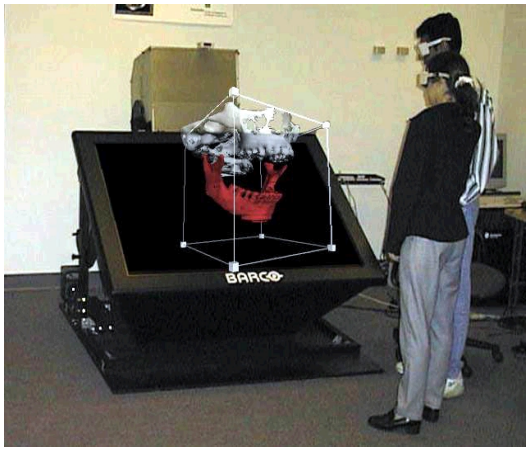


Abbildung 22: a) Aufbau eines CAVE mit 4 Projektionswänden



b.) Barcos Projektionstisch Baron

werden und nicht mehr sichtbar sind. Wird beispielsweise bei einem umgedrehten Kegel mit dem Datenhandschuh auf die Spitze gezeigt, so sind Teile der Grundfläche nicht mehr sichtbar. Die Firma Fakespace mit dem Produkt Immersive Workbench sowie Barco mit ihrer "Virtuellen Werkbank" Baron entwickelten das System weiter, jeweils mit der zusätzlichen Eigenschaft eines einstellbaren Neigungswinkels der Arbeitsfläche. Bei Barcos Projektionstisch ändert sich dabei automatisch die Perspektive (siehe Abbildung 22b.). Die Auflösung beider Geräte liegt nur bei maximal 1600x1200 Punkten. Allgemein wird bei diesem Ausgabegerät auch von einem Virtual Table gesprochen.

Cave und Virtual Table eignen sich wie angesprochen für gruppenorientiertes Arbeiten. Dies bedeutet aber auch, daß keine Eingabegeräte eingesetzt werden können, welche die Mensch-Mensch-Kommunikation stören wie beispielsweise ein Datenanzug. Ein großes Problem bei beiden Systemen liegt darin, daß für jeden Benutzer ein perspektivisch korrektes Stereobildpaar generiert werden sollte. Dazu sind einerseits Hochleistungsgraphiksysteme nötig und zum anderen sinkt die Bildrate je Benutzer beträchtlich ab, was zu Flackererscheinungen und letztlich Schwindel und Kopfweh führen kann. Zweibenutzersysteme wurden trotz dieser Einschränkung realisiert, Mehrbenutzersysteme scheitern im Augenblick noch an der technischen Begrenzung einer maximalen Frequenz von 144 Hz der Shutterbrillen.

4.13 Volumen Displays

Es wurden schon einige Prototypen von direkten Volumendisplays gebaut. Dabei wird ein wirkliches 3D-Bild in ein abgeschlossenes Volumen projiziert [CW93]. Eines dieser Displays

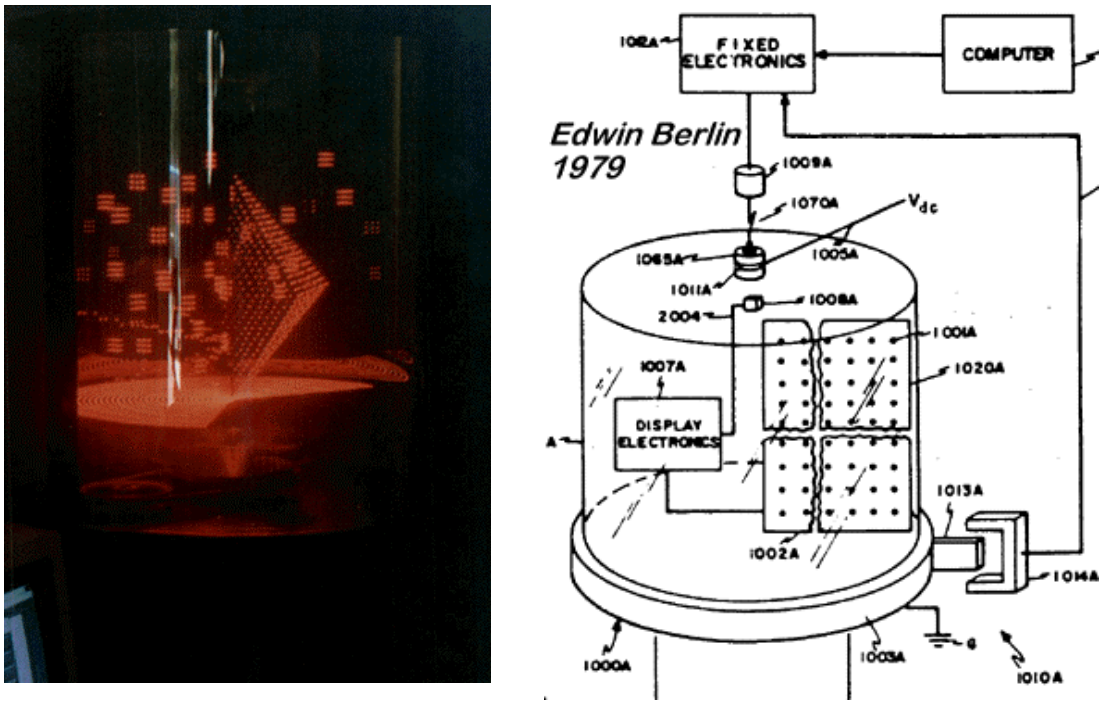


Abbildung 23: a.) Beispiele eines Volumetrischen Displays. b.) Digitales Volumetrisches System mit LEDs von E. Berlin (MIT 1979)
Die ersten Arbeiten an analogen Volumendisplays gehen schon bis in die 60er Jahre zurück (J. R. Schipper 1960)

namens Omni View von Texas Instruments hat eine rotierende Plexiglasspirale innerhalb eines Zylinders [HS93]. Auf diese Plexiglasscheibe trifft ein Laserstrahl, der absorbiert und lokal gestreut wird und somit sichtbar wird. Im Prinzip kann jedes Volumenelement durch den synchronisierten Laser auf der Plexiglasspirale angesprochen werden, bis auf die durch den Aufbau bedingte Rotationsmittelachse des Zylinders. Kreuzt der Laserstrahl das leere Volumen so bleibt er unsichtbar. Dieses Verfahren bietet nur eine geringe Auflösung und somit nur einfache geometrische Figuren. Bisher gibt es noch kein Volumendisplay, das die Darstellung der vollen Farbpalette durch mindestens drei Laserstrahlen erlaubt. Der Vorteil von Volumendisplays liegt darin, daß keine Betrachtungshilfsmittel getragen werden müssen.

4.14 Holografische Verfahren

Computergenerierte Hologramme (CGH) sind eine andere 3D-Bildtechnik, bei der ebenfalls keine Hilfsmittel für die Betrachtung gebraucht werden [DW80]. Die Berechnung der Interferenzmuster für ein monochromatisches oder Weißlicht-Hologramm ist sehr aufwendig.

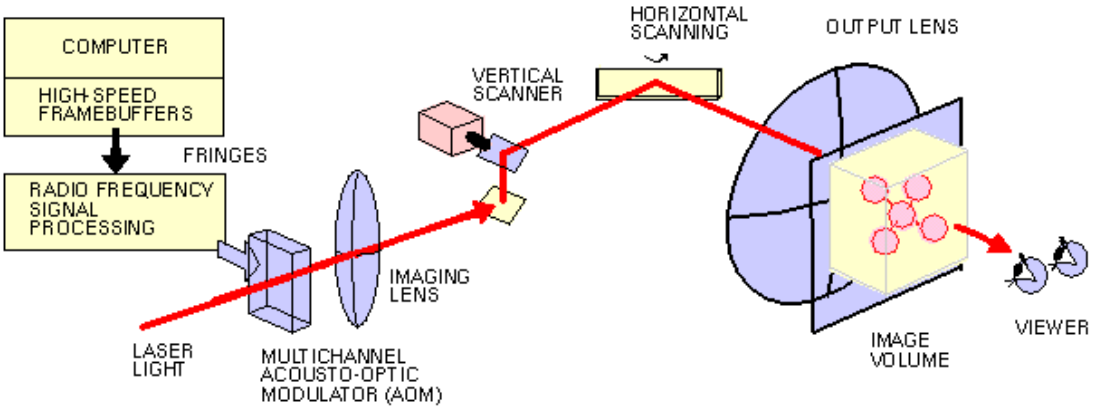


Abbildung 24: Schematische Darstellung des MIT-Holovideo Displays mit Akusto-optischem Modulator [nach LM97]

Dennoch wurde dies in Echtzeit mit einem massiv parallelen Computer (Connection Machine) erreicht. Das Hauptproblem liegt in der Wahl des richtigen Ausgabegeräts, da die Auflösung des Interferenzmusters im Bereich der Wellenlänge des Lichtes liegen muß. Drucker, Bildschirm oder Diabelichtungsgerät scheiden damit aus und es kommen modifizierte Scanner-Elektronenmikroskope zum Einsatz, die das Muster mit einem Elektronenstrahl auf einen Film schreiben. Ein anderes Verfahren verwendet als Realtime-Holographie-Display ('Holovideo') einen Kristall, indem das Interferenzbild durch Schwingungen im Ultraschall-bereich erzeugt wird. Der Kristall wird als Akusto-optischer Modulator bezeichnet, denn diese Schwingungen verändern den lokalen Brechungsindex des Kristalls, und ein Laserstrahl, der zeilenweise durch den Kristall geschickt wird, wird entsprechend dem Interferenzmuster moduliert. Die Datenmenge, die dabei entsteht, ist sehr hoch. Ein 10x10x10cm großer Gegenstand mit 30° Blickwinkel liefert ein Hologramm von 25 GByte. Bei 25 Bildern pro Sekunde und 8 Bit Auflösung betrüge die notwendige Datenübertragung 5 TByte/sec. An neuen Kompressions- und Kodierungsverfahren zum Reduzieren von Rechenzeit und Speicherbedarf wird gearbeitet, um vollfarbige interaktive Video-Holographie in Echtzeit auf Workstations möglich zu machen. Holographische Bilder vermitteln den besten Tiefenein-druck, da es keine Probleme mit dem Augenabstand, der Konvergenz und Akkomodation, wie bei stereoskopischen Bildern, oder Abdeckungsprobleme durch die transluzenten Voxel bei

Volumendisplays gibt. Volle Farbdarstellung sowie die Darstellung der Rückseite von Objekten, zum Beispiel auf Filmen in Form von Regenbogen-Hologrammen sind noch ungelöst.

4.15 Bewertung

Obwohl in diesem Feld schon einiges entwickelt worden ist und nur eine repräsentative Auswahl (weitere Informationen finden sich in den Reports [DM95, WB97, IF97]) kurz behandelt wurde, gibt es das universelle Ein- bzw. Ausgabegerät für Virtuelle Umgebungen noch nicht. Im Folgenden eine Zusammenfassung der Beschränkungen von heutigen Techniken:

- Systeme für die Verfolgung (engl. tracking) von Kopf, Hand und Körperbewegungen:
  - Beschränkung von Arbeitsvolumen, Auflösung und Genauigkeit, Mißinterpretation durch hohe Latenzzeit und mit Rauschen behafteten Daten, elektromagnetische Störungen mit anderen Geräten, hohe Anschaffungskosten.
  - Datenhandschuhe unterscheiden nur zwischen wenigen Gebärden, geringe Genauigkeit zur Bestimmung der Gelenkwinkel, hoher Aufwand beim Kalibrieren.
  - Das Tracking von Augenbewegungen ist sehr ungenau und läßt Kopfbewegungen außer acht.
- 3D-Zeigegeräte:
  - Bei bisherigen Techniken ist die Trennung von Rotation und Translation schwierig.
  - Ermüden von Armen bei frei zu tragenden 3D-Geräten.
  - Bei tischbasierten Eingabegeräten Beschränkung auf wenige VR-Felder.
- Systeme mit taktiler Rückkopplung:
  - Fehlen von Modellen und Reiz-Reaktions-Algorithmen für die Reizgenerierung.
  - Taktiles Feedback ist auf kleine Körperflächen begrenzt.
  - Die bisherigen Geräte decken keinen größeren Bereich von fühlbaren Empfindungen ab.
  - Begrenzung in der Oberflächenrepräsentation (Feinstruktur, kleine lokale Unebenheiten oder Eigenschaften wie glatt bzw. schlüpfrig).
- Systeme mit Kraftrückkopplung:
  - Beschränkung der Rückkopplung auf wenige Gelenke.
  - Fehlen von allgemeinen Modellen und Algorithmen für die kinetische Reizung.
  - Einschränkung der Bewegungsfreiheit durch die Geräte (Exoskelett).

- Keine globalen Rückkopplungen (Erdbeben, Explosion).
- Verletzungsgefahr.

- Ausgabegeräte:
  - Geringe Auflösung der Bildschirmtechniken.
  - Beschränkung der Teilnehmer durch hohen Aufwand für Standpunktberechnung.
  - Unbequem, da zusätzliche Hilfsmittel (Brille, HMD) nötig.

Der Trend bei den Aus- und Eingabegeräten geht weg von den generischen Geräten hin zu speziell auf die Aufgaben zugeschnittenen Geräten. Gerade in dem Simulationsbereich der Medizin braucht man Geräte mit der Fähigkeit der Kraft- und Gefühlrückkopplung. Diese haptischen Geräte sind aber noch in der Entwicklungsphase und erfordern eine hohe Rechenleistung im Hintergrund. Die Geräte sind zum Teil noch sehr teuer und nur durch den Einsatz in der PC-Welt wird diese Technologie erschwinglich. Beispiele hierfür sind die 3D-Zeigegeräte wie CyberMan2, welches 2D-Maus und 3D-Spacemouse integriert oder Feedbackgeräte wie Microsofts Joystick Sidewinder Forcefeedback oder die FEELit-Mouse von Immersion. Nach wie vor muß ein Benutzer umständliche Geräte anlegen (Datenhandschuh, HMD) und diese vorher noch kalibrieren, um an einer virtuellen Welt teilhaben zu können. Neben dem zusätzlichen Gewicht ist auch sein Bewegungsfeld eingeschränkt. Dies war mit ein Grund, warum das WSI/GRIS einfachere VR-Geräte wählte (CrystalEyes VR System mit Shutterbrillen und Ultraschalltracker, Spaceball und Spacemouse). Die Unterstützung von Betriebssystemseite und die Programmierung für VR-Geräte ist als proprietär zu bezeichnen. Was die 3D-Ausgabegeräte angeht, so können diese nach folgenden Kriterien unterschieden werden:

Prinzip		**Herkunft der Wellen**	**Anzahl der verschiedenen Ansichten**	**Blickpunkt-abhängige Perspektive**
Betrachten mit Hilfsmittel (stereoskopisch)	Multiplexing: Farben, Polarisation, zeitlich, räumlich.	Feste Bildebene, Blickabhängig gesteuerte Bildebene	zwei	Optional (für einen einzelnen Betrachter)
Freies Betrachten	Richtungsabhängiges Multiplexing (z. B. Beugung, Brechung, Reflektion, Verdeckung)	Feste Bildebene, Blickabhängig gesteuerte Bildebene	≥ zwei	Optional (für eine kleine Anzahl von Betrachtern)
Autostereo-skopisch	Volumetrischer Bildschirm, Elektro Holographie	Verschiedene Tiefenebenen (Schichten), ganzer Raum	Unbegrenzt	Inhärent (für eine kleine Anzahl von Betrachtern)

Tabelle 3: Einteilung von 3D-Ausgabegeräten (nach [PW97])

Abhängig von dem verwendeten System fühlt man sich mehr oder weniger in die Szene selbst eingetaucht (engl. Immersion). Unterschieden werden dabei folgende Qualitätsgruppen:

- nicht immersiv (in monitorbasierte Systeme),
- semiimmersiv (arm-mounted Systeme),
- immersiv (head-mounted Displays).

Aus der Auswahl der für diese Arbeit zur Verfügung stehenden VR-Geräte wird ersichtlich, daß der Lehrstuhl in der Kategorie "nicht immersive Systeme" tätig ist.[16]

[16] Ende 1998 wurden neue Geräte, wie der Barcos Projektionstisch Baron, PHANToM Premium und ein Magnetischer Tracker „Flocks of Birds“ von Ascension, angeschafft. Somit ist der Lehrstuhl weiterhin im nicht immersiven Bereich tätig. Die neuen Geräte wurden nicht im Rahmen dieser Arbeit eingesetzt.

5 Benutzungsschnittstelle

5.1 Allgemein

Unter einer Benutzungsschnittstelle versteht man allgemein den Teil eines Softwaresystems, welcher sich mit der Interaktion des Menschen befaßt und die gestellten Aufgaben mit der Applikation erledigen läßt. Sie ist somit der sichtbare Teil der Anwendung, die den Zugriff auf die Funktionen der Applikation erlaubt. Die Benutzungsschnittstelle spielt eine entscheidende Rolle für die Akzeptanz, Leistungsfähigkeit und letztendlich den Erfolg der Applikation.

Geschichtlich ist Ivan Sutherland als Erfinder der Graphischen Benutzungsoberflächen (engl. Abkürzung GUI) anzusehen. 1962 entwickelte er ein interaktives Zeichenprogramm, genannt Sketchpad, mit dem durch die Hilfe eines Lightpens einfache Objekte wie Punkte, Linien und Kreissegmente auf ein Vektordisplay gezeichnet werden konnten [SI63]. Eine bedeutende Benutzungsschnittstelle, die heute noch das Aussehen vieler Benutzungsoberflächen prägt, ist die graphische Oberfläche des Apple Lisa/Macintosh. Diese wurde Anfang der 80er Jahre entwickelt und basiert auf Arbeiten von Xerox PARC. Das wesentliche Kennzeichen solcher GUIs sind standardisierte graphische Elemente, wie beispielsweise Pull-Down-Menüs oder Auswahlboxen, die bei Selektion durch den Benutzer Aktionen auslösen. Der große Vorteil von GUIs resultiert daraus, daß der Benutzer den einzelnen Elementen aufgrund ihres Aussehens eine bestimmte Funktion, ein Verhalten bzw. eine Aktion zuordnen kann. Der Benutzer muß keine kryptischen Abkürzungen auswendig lernen, um die Applikation steuern zu können (z. B. frühere textbasierte Versionen von WordStar, WordPerfect, Multiplan). Ferner findet sich ein Benutzer schneller in einem neuen Programm zurecht, da sich das Muster von Aussehen und Ablauf gleicht. Die Mehrzahl der heutigen graphischen Oberflächen im Desktopbereich wird als *WIMP*-Oberfläche bezeichnet. *WIMP* steht als Abkürzung für ***W****indows* (Fenster), ***I****cons* (Symbole), ***M****enus* (Menüs) und ***P****ointers* (Zeiger) und beschreibt die vier Hauptkomponenten, die Kennzeichen einer solchen 2D-Oberfläche sind. Die Funktionen der Komponenten lassen sich kurz folgendermaßen beschreiben: Die Fenster teilen den Bildschirm in Teilbereiche auf, die Icons symbolisieren Applikationen, Vorgänge und Daten, Menüs bieten eine Liste von Auswahlmöglichkeiten und Zeiger dienen zur Selektion von GUI-Objekten. Vertreter dieses Oberflächentyps sind die gängigen Betriebssysteme der Windows-

famile (MS: Windows 3.11, Win95/98, Windows NT, Apple: MacOS, Unix: Windows X11 OSF/Motif, IBM: OS/2). Zwei technische Entwicklungen trugen entscheidend zum Siegeszug von *WIMP*-Oberflächen bei: 1.) Die Entwicklung des Rasterdisplays, was zur Ablösung des Vektordisplays führte. 2.) Die Entwickler Douglas Engelbart und William K. English entwickelten 1963 am Stanford Research Institute einen hölzernen Vorläufer der heutigen Maus. 1982 stellte Mouse Systems die erste kommerzielle Maus (3-Tasten-Maus) für den IBM PC vor. Die Apple-Maus, ursprünglich für die Lisa Computer entwickelt, und Microsofts 2-Tasten-Maus kamen ein Jahr später auf den Markt. Heute ist die Maus zusammen mit der Tastatur das wichtigste Peripheriegerät für die Interaktion mit dem Computer unter Windows-, Unix- oder Apple-Betriebssystemen. Eine Zusammenfassung über die Herkunft von elementaren Benutzungsschnittstellenobjekten ist in der Tabelle im Anhang aufgeführt. Gegenwärtig wird versucht, eine Normierung/Standardisierung für 2D-GUI-Objekte festzuschreiben [IS97] .

5.2 Benutzungsschnittstelle für Virtuelle Umgebungen

Für die Interaktion in der dritten Dimension könnte auf die bewährten und erforschten Benutzungsschnittstellenobjekte aus der 2D-Welt zurückgegriffen werden. Grundsätzlich gibt es aber Gründe, die gegen eine direkte Übernahme sprechen:

- Keine einheitliche Hardwarekonfiguration.
  Es gibt nicht das Standardsystem für VR-Umgebungen.
- Keine Unterstützung von mehrdimensionalen Eingabegeräten.
  WIMP-Oberflächen unterstützen nur 2D-Eingabegeräte. Untersuchungen zeigen, daß diese Beschränkung der Bewegungsfreiheiten von Benutzern als störend empfunden wird. [HD94]
- Maus spielt keine Rolle bei vollimmersiven Systemen.
  Die Benutzer befinden sich in einer abgeschlossenen Szene mittels HMD und sehen somit die Maus nicht. Die Rückkopplung ist erschwert. Außerdem befindet sich im Umfeld von VR-Benutzern kein Tisch.
- Geringe Auflösung der Wiedergabegeräte wie HMDs erfordern andere Widgets.
  Text bzw. Symbole können nicht erkannt werden.
- Zerstörung des Raumeindrucks durch zweidimensionale GUI-Elemente.
- Keine standardisierten GUIs.
  Richtlinien, Normierungen und Standards sowie Evaluierungen fehlen.

In der Tat werden Icons, Menüs oder Knöpfe teilweise bei der Interaktion in 3D-Szenen eingesetzt, das heißt diese künstlichen Gebilde werden in die reale Szene eingeblendet. Im folgenden soll kurz auf diese Elemente und auf die Unterschiede zum 2D-Pendant eingegangen werden.

5.2.1 3D-Buttons

3D-Buttons entsprechen in Aufgabe und Anwendung denen in der 2D-Welt. Der Schalter wird für Aktionen mit einer binären Zustandsänderung (engl. toggle) eingesetzt. Beispielsweise wird eine Eigenschaft ein- oder ausgeschaltet. Werden mehrere Buttons zu einer Gruppe zusammengefaßt, so kann eine 1 aus n Auswahl vorgenommen werden. Diese Buttons werden Radiobuttons genannt. Checkboxen erlauben eine Mehrfachauswahl m aus n. In vollimmersiven Systemen genügt es nicht, wie in der 2D-Welt, über den Button eine Aktion auszulösen um diesen zu aktivieren, sondern das 3D-Button-Objekt muß mit einem 3D-Cursor (z. B. der Hand des Benutzers mittels Datenhandschuh) kollidieren. Außer der 3D-Geometrie entspricht der 3D- dem des 2D-Buttons. Textbeschriftung, Icons (Texturen) oder Status (aktiv/inaktiv) erleichtern den Umgang.

5.2.2 3D-Menüs

Wie bei 2D-Menüs muß das 3D-Menü durch eine Benutzeraktion aktiviert werden, damit die Einzeleinträge des Popup-Menüs sichtbar werden und die entsprechende Funktion zur Erfüllung der Aufgabe ausgewählt werden kann. Es gibt Lösungen, um die Aktivierung und Auswahl des 3D-Menüs ohne Kollisionsdetektion zu erlauben, indem Sprache- oder Gestikeingabe mittels Datenhandschuh die Aktion auslösen. Damit sind wir auch bei der Hauptunterscheidung von 2D- und 3D-Menüs, und dem gleichzeitigen Problem von letzteren, welches in den Positionierungsmöglichkeiten liegt. Während 2D-Menüs eine feste Orientierung und einen festen Abstand zum Benutzer haben und deshalb immer zugänglich und lesbar sind, können 3D-Menüs durch andere Szenenelemente verdeckt sein. Sind sie zu weit weg oder ist die Orientierung im Raum ungünstig, so sind sie nicht mehr lesbar, sind sie zu nahe, ist keine stereoskopische Darstellung möglich. In diesen Fällen sind die Hauptbewertungsgrundsätze von 3D-, wie auch 2D-Menüs, Lesbarkeit und Brauchbarkeit, nicht mehr erfüllt. Eine absolute Positionierung im Raum macht also keinen Sinn, außer die Menüs sind in ein Objekt im Raum bzw. in einen Computer integriert. Nach Möglichkeit wird die Positionierung des Menüs relativ zum Beobachter/Benutzer erfolgen, wobei neben dem richtigen Abstand und der Orientierung auf die Sichtbarkeit des Menüs zu achten ist. Bei großen Menüs ergibt sich der Umstand, daß der Benutzer sich durch die Menüeinblendung gestört fühlt, da nichts von der restlichen Szene

zu sehen ist. Mit dem Einsatz von transparenten Menüs bleibt der Überblick über die Szene erhalten und der Benutzer kann weiterhin störungsfrei durch die Szene navigieren.

5.2.3 3D-Widgets und Handles

Der Begriff Widget wurde in [CS92, TJ97] wie folgt definiert: "Ein Widget ist die Kombination von Geometrie und Verhalten und wird zur Kontrolle und Informationsvisualisierung der Applikationsobjekte benutzt." Widgets wurden schon bei den 2D-Benutzungsoberflächen eingesetzt. Beispielsweise Textwidgets zur Textdarstellung, Pushbutton- und Scrollbarwidgets zur Konstruktion von Eingabemasken und Handle-Widgets zur direkten Manipulation von Graphischen Objekten wie skalieren, verschieben und drehen. 1992 wurde erstmals die Erweiterung von 2D-Widgets auf den dreidimensionalen Raum vorgestellt [CS92]. "Die gezeigten 3D-Widgets sind räumliche Objekte, die alle zusätzlichen Freiheitsgrade des dreidimensionalen Raumes nutzen können und so, im Gegensatz zu den 2D-Widgets, die meist nur eindimensional genutzt werden, die zusätzliche Flexibilität der Mehrdimensionalität nutzen können. Es wurden verschiedene Arten von implementierten Widgets präsentiert. Mit einer virtuellen Kugel können Zielobjekte intuitiv rotiert werden. 3D-Objekthandles erlauben die Skalierung des dreidimensionalen Objektes in alle drei Achsenrichtungen. Ein Farbeditor in Form eines Selektionswürfels innerhalb des RGB-Farbraumwürfels erlaubt das direkte Einstellen der diffusen Objektfarben durch die Translationskomponente des Positionstrackers." [aus TJ97]. Alle Widgets, mit Ausnahme des Farbeditors, wurden für die Mauseingabe konzipiert, nutzten also nicht die 6 DOF Eingabegeräte. Die Idee von mausbasierten 3D-Handels wurde in kommerzielle Produkte wie die Graphikbibliothek Open Inventor übernommen und bildet dort einen wesentlichen Beitrag zur Manipulation von 3D-Objekten. Eine Erweiterung des Konzepts der zweidimensionalen MagicLenses in Form von 3D-Widgets wird in [WG95, VC96] gezeigt. MagicLenses sind durchsichtige Widgetelemente, die Objekte innerhalb ihres Wirkungsbereichs vergrößern bzw. verkleinern. Die Erweiterung besteht nun darin, daß die MagicLense, in diesem Zusammenhang besser als Volumenlinse bezeichnet, im dreidimensionalen Raum ganze räumliche Bereiche modifiziert. Beispielsweise erscheinen die Objekte in anderem Zeichenstil (Drahtgitter) oder die Volumenlinse dient als Filter (Röntgengerät), es werden nur bestimmte Typen angezeigt (zeige alle Restaurants der Stadtszene) bzw. sonst unsichtbare Objekte (Kanalleitungen der Stadtszene). Es gibt auch sogenannte flache Linsen, die das aufgespannte Sichtvolumen verändern, und Implementierungsansätze, die eine Hintereinanderschaltung mehrerer unterschiedlicher Linsen erlauben.

Widgets werden oft als dreidimensionale Werkzeugobjekte dargestellt, die ein Benutzer aufnehmen kann. Bei der Kollision mit dem Objekt wird der eigentliche Arbeitsraum mit Hilfsobjekten (Manipulatoren, Dragger) angezeigt. Mit einer Klebstofftube können Objekte gruppiert werden bzw. mit der Schere wird die Gruppierung aufgehoben, eine Zange ermöglicht das Skalieren und ein Hammer ein Deformieren des Objekts.

Wie andere Interaktionstechniken soll auch der Gebrauch von Widgets möglichst einfach und natürlich für den Benutzer sein. Vorhandenes Wissen vom Umgang mit Objekten aus dem Alltag sollte in die künstliche Welt transferiert und angewendet werden. Man spricht in diesem Fall davon, daß die Interaktionstätigkeit einer Metapher folgt. Auf diesen Begriff wird in Kapitel 6 noch näher eingegangen.

6 Interaktion

Die Interaktion selbst kann einhändig oder zweihändig (mit zwei Eingabegeräten) erfolgen. Es liegt in der Natur des Menschen, daß es eine dominante und eine nicht-dominante Hand gibt. Entsprechend unterschiedlich sind die Fähigkeiten bei der Ausübung von Tätigkeiten. Auf diesen Umstand wird jedoch im folgenden nicht näher eingegangen. Um die gestellten Aufgaben zu erfüllen, werden verschiedene Interaktionstechniken eingesetzt. Bei VR-Interaktionen hängt die Interaktionstechnik anders als bei 2D-GUI entscheidend von dem verwendeten Eingabegerät ab.

6.1 Metapher

Unter einer Metapher (engl. Metaphor) versteht man allgemein eine meist bildhafte Umschreibung, die zur besseren Veranschaulichung führt. Durch diese Bildhaftigkeit, entnommen aus den allgemeinen Lebenserfahrungsbereichen, kann ein Lernender neue Wissensbereiche schneller und einfacher erschließen. Die Bedienung von 3D-Graphik-applikationen macht naturgemäß Probleme bei neuen Anwendern. Spezielle sogenannte 3D-Metaphern sollen nun den Lernaufwand minimieren. Eine 3D-Metapher erklärt dem Benutzer die zweidimensionale Benutzungsschnittstelle einer Applikation, indem sie an das natürliche Wissen aus der dreidimensionalen Welt anknüpft. Wichtig für die Akzeptanz von Metaphern ist die Tatsache, wie vertraut man sich mit der Szene fühlt [KA93]. Für die Bewertung von 3D-Metaphern wird die Zeitdauer der motorischen Bewegung, welche durch das Fitts' Gesetz beschrieben wird, herangezogen [FP54]. Eine ausführliche Untersuchung auf der Basis einer mathematisch-informationstheoretischen Betrachtungsweise wurde in [KL93] beschrieben. Darin wird folgende Definition für 3D-Metaphern gegeben:

"Eine 3D-Metapher erklärt die Bedienung einer dreidimensionalen Operation durch ein zweidimensionales Eingabegerät. Sie besteht aus zwei Teilen: dem metaphorischen Konzept, das die Strukturabbildung vom mentalen Modell des Anwenders in die Mensch-Maschine-Schnittstelle beschreibt und der Implementierung, die die Abbildung der zweidimensionalen Eingabe in das dreidimensionale, mathematische Modell der Applikation bezeichnet. Um den Aufbau des mentalen Modells zu erleichtern, bezieht sich das metaphorische Konzept auf den dreidimensionalen Erfahrungsbericht des Anwenders."

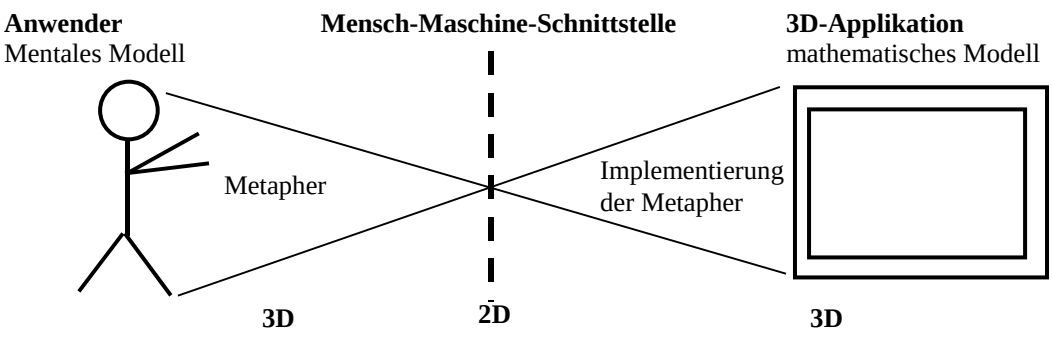


Abbildung 25: Darstellung der Mensch-Maschine-Schnittstelle

Die Abbildung 25 zeigt den Zusammenhang von 3D-Metaphern und ihre Implementierung. Die 3D-Applikation enthält das mathematische Modell einer dreidimensionalen Operation. Der Anwender hat wiederum ein dreidimensionales mentales Modell der Operation, welche er mit der Applikation ausführen möchte, im Sinn. Beide Modelle müssen nicht notwendigerweise gleich sein. In aller Regel werden sie aber aus der realen Erfahrungswelt entnommen. Ein Vermittlungsproblem resultiert daraus, daß die Mensch-Maschine-Schnittstelle bei nichtimmersiven 3D-Applikationen zweidimensional ist. Das heißt, durch zweidimensionale Eingabegeräte muß eine dreidimensionale Operation ausgeführt werden, und diese wird durch 2D-Ausgabegeräte kontrolliert. Dieses Problem tritt bei vollimmersiven Applikationen nicht auf, da dann eine 3D-Mensch-Maschine-Schnittstelle existiert.

Ein Beispiel für eine 2D-Metapher ist die bekannte Desktop-Metapher (Schreibtischoberfläche), welche in windowsbasierten GUIs vorherrschend ist. Die Erweiterung dieser 2D-Metapher in die dritte Dimension wird im nächsten Kapitel erörtert werden.

6.2 Navigation

Unter Navigation versteht man allgemein zweierlei:

- Einerseits die Fähigkeit, sich Orientierung über den momentanen Aufenthaltsort zu verschaffen.
- Andererseits die Fähigkeit, sich von einem Punkt im Raum zu einem anderen zu bewegen.

Die Art der Navigation in einem virtuellen Raum hängt von der zur Verfügung stehenden Hardware sowie von der Größe, Komplexität und Aufgabenstellung ab. In der virtuellen Welt ist es möglich, sich sowohl im Mikrokosmos extrem langsam im Nanometerbereich zu bewegen, wie auch im Makrokosmos mit (Über-)Lichtgeschwindigkeit riesige Strecken zurückzulegen. Die Geschwindigkeit der Fortbewegung des Benutzers orientiert sich an den Modellgrößen.

Beim Spaceball benutzen wir als Eingabegröße die absolute Position im Raum, die Geschwindigkeit oder die Beschleunigung. Am intuitivsten und somit am geeignetsten ist die Geschwindigkeit als Eingabegröße. Ferner wurde der Headtracker mit der Spaceballeingabe gekoppelt. Der Headtracker gibt hierbei die Richtung vor, in der die Bewegung erfolgen soll. Es zeigte sich, daß die Benutzer Schwierigkeiten hatten, sich nur in eine Richtung zu bewegen beziehungsweise sich nur um eine Rotationsachse zu drehen. Aus diesem Grund wurde ein zweiter dominanter Modus implementiert, der bewirkt, daß nur die stärkste Translations- oder Rotationsauslenkung zur Anwendung kommt (siehe dazu auch Kapitel 10.2.2.).

6.2.1 Angewandte Metapher

Die klassischen Metaphern zur Bewegung in der dritten Dimension wurden bereits 1990 von Ware und Osborne vorgestellt [WO90]:

- World/Scene in Hand (Welt-in-Hand-Metapher)
  Der Benutzer hat eine externe Sicht auf sein Objekt und manipuliert das Objekt direkt mit seinen Handbewegungen.
- Camera/Eyeball in Hand (Kamera-in-Hand-Metapher)
  Die Sicht des Benutzers wird gesteuert mittels direkter (handgeführter) Manipulation einer virtuellen Kamera.
- Point and Fly (Flug-Metapher)
  Der Benutzer navigiert fliegend durch die Szene.

6.2.1.1 World/Scene in Hand

Diese Metapher weist Parallelen zum Greifen auf. Im Gegensatz zum Greifen wird aber die ganze Szene ergriffen und transliert bzw. rotiert. Wird diese Metapher erweitert, das heißt ein Objekt in der Szene kann selektiert bzw. manipuliert werden, so spricht man auch von der Objekt-in-Hand-Metapher.

6.2.1.2 Camera/Eyeball in Hand

Bei diesem Verfahren steuert der Benutzer die Position und Orientierung der virtuellen Kamera. Diese Metapher eignet sich zum Untersuchen von Details in der nächsten Umgebung des Benutzers. Bei dieser Metapher wird die Anzahl der Freiheitsgrade oft eingeschränkt. Dies wird gerne dann verwendet, wenn sich der Navigierende durch eine Szene bewegen soll (engl. walk-through). Es wird dann nur eine Rotation um die y-Achse zugelassen aber nicht um die beiden anderen x und z. Und was die Translation angeht, so wird entweder die Translation in y-Richtung ausgeschaltet (kein Hüpfen, Schweben) oder der Navigierende bleibt durch Einbeziehen einer Gravitationskraft immer auf dem Boden (z. B. Treppen steigen).

6.2.1.3 Point and Fly

Wie der Name schon andeutet, erlaubt dieses Paradigma die Steuerung eines virtuellen Flug- oder Fahrzeugs. In kleineren Gebieten, wo es auch auf Positioniergenauigkeit ankommt, ist es der Eyeball-in-Hand-Metapher unterlegen.

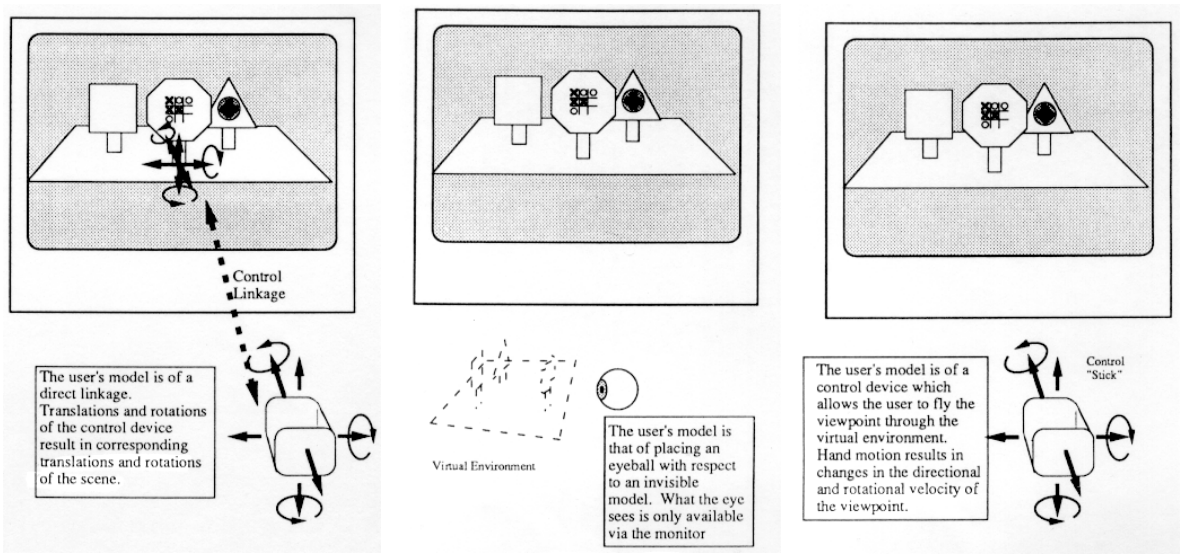


Abbildung 26: Gegenüberstellung der klassischen Metaphern Welt-in-Hand, Kamera-in-Hand, Flug (von links nach rechts) [WO90]

6.2.1.4 Unterschiede

Erfolgt die Interaktion auf Basis der Welt-in-Hand-Metapher, so folgt bei Verschieben des Objekts dieses der Bewegung. Bei der Kamera-in-Hand-Metapher, folgt die Kamera der (eigenen) Bewegung, ein stationäres Objekt wandert also in die entgegengesetzte Richtung. Zur Manipulation von selektierten Objekten in größeren Welten wird der World-in-Hand-Metapher der Vorzug gegeben, da sie dem natürlichen Ablauf entspricht. Der größte Teil der Szene

bleibt ruhig, während sich nur das selektierte Objekt bewegt. Die sogenannte Reiz-Reaktions-Verträglichkeit bleibt erhalten. Der Mensch erwartet Raumkonstanz[17], das heißt, bewegen sich große Teile seiner Umgebung, so entsteht die Illusion einer Eigenbewegung. Beim Blick von einer Brücke auf fließendes Wasser oder auf einen abfahrenden Zug auf dem Nachbargleis stellt sich zum Beispiel dieser Eindruck der Selbstbewegung ein.

6.2.2 Teleportation

Bei großen Szenen, vorwiegend bei Spielen, werden reale Objekte eingebaut, die eine Ortsverlagerung ermöglichen. Beispielsweise Aufzüge, Türen, Brücken, Tunnels, Portale oder künstliche Gebilde, die eine Teleportation oder ein "Beamen" in andere Szenenteile bzw. Welten zulassen. Dieses einfache Navigationsparadigma der Teleportation hat den Vorteil, daß keine aufwendigen Übergänge gestaltet werden müssen.

6.2.3 Navigation in virtuellen Umgebungen

Die Navigation in virtuellen Umgebungen kann abhängig vom vorhandenen Eingabegerät in folgende Gruppen unterteilt werden:

- Abbildung der direkten physikalischen Bewegung:
  Bewegt sich der Benutzer in der realen Welt einen Schritt vorwärts, so ändert sich die Position in der virtuellen Welt in gleicher Weise. Diese Art der Navigation ist natürlich und einfach, allerdings ist das Arbeitsvolumen durch die Wirkungsweise der Tracker begrenzt.
- Navigation in Blickrichtung (engl. gaze directed flying):
  Die Bewegung erfolgt in die Richtung, in welche der Benutzer schaut. Es ist ein HMD oder ähnliches Gerät für diese Navigation notwendig. Mit nichtimmersiven Displays ist diese Technik nicht anwendbar. Ein Nachteil der Navigation in Blickrichtung ist, daß der Benutzer sich nicht vorwärtsbewegen und gleichzeitig kurz zurückschauen kann.
- Handgesteuerte Navigation:
  Bei diesem Verfahren wird die Hand des Benutzers zur Steuerung eingesetzt. Dies geschieht über Gestenerkennung. Beispielsweise dient der ausgestreckte Zeigefinger dazu, die Richtung der Bewegung anzugeben. Ein Vorteil ist, daß sich der Benutzer umschauen kann, ein Nachteil, daß ungeübte Benutzer erst mühsam die Beziehung zwischen Bewegungsrichtung und Handorientierung lernen müssen.

[17] Die Bewegungen auf der Retina werden durch das sogenannte Flowfield beschrieben. Dabei wird der Verschiebungsvektor zwischen zwei Bildern in regelmäßigen Gitterpunkten berechnet.

- Navigation mit physikalischen Steuergeräten:
  In diesem Fall wird die Bewegungsrichtung durch zusätzliche Eingabegeräte wie Spacemouse oder Joystick vorgegeben. Die Probleme der Benutzung der richtigen Metapher, vor allem bei 2D-Eingabegeräten, wurde im vorigen Kapitel erörtert. Für Fahr- und Flugsimulationen sowie Walkthroughs kommt dieses Verfahren zur Anwendung, da es ein haptisches Feedback durch den Widerstand des Steuergeräts gibt.
- Navigation mit virtuellen Steuergeräten:
  Virtuelle Geräte wie Lenkräder werden zum Beispiel durch Datenhandschuhe oder Flying Mouse bedient. Vorteil: Flexibel, da alle physikalischen Steuergeräte virtuell simuliert werden können. Nachteil: Es gibt kein haptisches Feedback.
- Objektgesteuerte Navigation:
  Die Bewegung des Benutzers erfolgt durch Objekte in der Virtuellen Umgebung. Neben autonomen Bewegungsmitteln wie Fahrstuhl und Förderband gibt es sogenannte Attraktoren und Repelloren, die den Benutzer anziehen beziehungsweise abstoßen. In diese Klasse fällt auch die Navigation mittels Menüs, Icons (Viewpoints) oder Widgets.

6.3 Selektion

Die Selektion wird in zwei Kategorien eingeteilt:

- lokale Selektion
- globale Selektion

Unter der ersteren versteht man, daß der 3D-Cursor, beispielsweise das Handecho eines Datenhandschuhes, direkt mit dem Objekt kollidiert. Bei der globalen Selektion kann ein Objekt aus der Distanz ausgewählt werden. Wie bei einem Laserpointer kann ein Objekt ausgesucht werden. Durch Klicken auf den Bildschirm mittels Maus wird ein "Selektionsstrahl" losgeschickt, welcher das erste Objekt auswählt. Als Selektionsstrahl kann, als Pendant zu realen Objekten, ein Laserstrahl oder eine Taschenlampe eingesetzt werden. Letztere hat den Vorteil, daß durch Vergrößerung des Selektionsstrahls bei größerer Entfernung, weiter entfernt befindliche Objekte besser selektiert werden können. Eine Abwandlung des Selektionsverfahrens mittels Selektionsstrahl bedient sich eines Lassos. Hierbei wird durch den Benutzer ein geschlossenes Polygon gezeichnet und alle selektierbaren Objekte werden in diesem irregulären Selektionsstrahl ermittelt.

Eine *Selektion in Blickrichtung* wäre zwar denkbar und intuitiv, aber technisch noch nicht realisierbar, da die exakte Richtung der fokussierenden Augen mit den heutigen 3D-Geräten nicht exakt bestimmbar ist. Eine Selektionshilfe wie ein Fadenkreuz in halbimmersiven Geräten wie BOOM3C ermöglicht aber grundsätzlich eine Selektion in Blickrichtung.

Eine weitere Möglichkeit ist die *Selektion von vorher definierten Objekten aus einer Liste*. Entweder erfolgt die Einblendung der Liste mittels eines Menüs oder die Auswahl erfolgt mittels Spracheingabe. Der Vorteil dieser Art der Selektion ist, daß auch Objekte außerhalb des Sichtbereichs ausgewählt werden können. Ein Nachteil, daß Objekte oft schwer textuell zu beschreiben sind, zum Beispiel die Auswahl einer bestimmten Blume aus einer Blumenwiese.

Um die Selektion eines Objektes dem Benutzer sichtbar zu machen, wird das Objekt (heller) eingefärbt oder ein Rahmen um das Objekt gezogen. Im Fachjargon wird dieses Hervorheben auch *Highlighting* genannt.

6.4 Manipulation

Die Manipulation wird grob in drei Gruppen eingeteilt:

- Manipulation von Position und Orientierung
- Manipulation der Größe
- Manipulation sonstiger Parameter

Der einfachste Fall einer Manipulation ist eine Positions- bzw. Orientierungsänderung. Erleichtert wird diese Art der Manipulation durch Kollisionserkennung, künstliche Gravitation und Objektassoziationen. Unter letzteren versteht man, daß sich Objekte in der Nähe eines anderen Objektes selbständig aktiv ausrichten können (Wie ein Magnetkompaß sich nach den magnetischen Polen ausrichtet). Diese Technik wird auch *Drag&Snap* genannt. Sie wird gebraucht, wenn zwei Objekte paßgenau zusammengefügt werden sollen.

Unter diese Form der Manipulation fällt auch das Greifen von Objekten. Es wurde von den Grabbing- und Drag&Drop-Mechanismen der 2D-GUIs abgeleitet. Die oben vorgestellten Metaphern kommen hier zur Anwendung. Das Greifen wird durch die Kollision des sogenannten Handechos des Datenhandschuhs oder eines 3D-Cursors eingeleitet und führt zum Festhalten des Objekts. Der Greifvorgang kann noch realistischer implementiert werden, indem nicht die ganze Hand, sondern jeder einzelne Finger zum Greifen des Objekts beiträgt.

Etwaiges taktiles Feedback kann dann noch Aufschluß über die Elastizität des Objekts (Gummi- oder Eisenkugel) und die Stellung (Spannung) der einzelnen Finger geben.

Bei der Manipulation der Größe wird eine Verformung wie Skalieren bzw. Scherung, oder eine Zerteilung bzw. Vereinigung mit einem anderen Objekt vorgenommen. Um bei einer lokalen Deformation die Materialeigenschaften bei der Verformung bzw. Modellierung, die durch Druck- oder Zugkräfte entstehen, real, das heißt richtig, einschätzen zu können, reicht oft reines visuelles Feedback nicht aus, sondern es ist ein zusätzliches taktiles Feedback(-gerät) nötig.

Die Stärke von Manipulationen in virtuellen Welten ist die Änderung von sonstigen Parametern. Dazu zählen die Änderung von physikalischen Eigenschaften wie Steifigkeit, Reibung oder Masse beziehungsweise Materialart eines Objekts und die Einstellung der Farbe mittels Farbeditor. Gerade im graphischen Bereich ist die richtige Wahl von verschiedenen Beleuchtungsparametern, wie der ambiente, diffuse, spekulare und emissive Anteil der Farbe, sowie Glanz und Transparenz bei der Verwendung des Phong-Modells zur wirklichkeits-getreuen Darstellung entscheidend.

Wird ein Hilfsmittel wie ein Datenhandschuh zur Manipulation eingesetzt, so läßt sich die Manipulation in zwei Arten einteilen [PM91]:

- Indirekte Manipulation: Die Hand manipuliert mit dem Werkzeug das Objekt.
- Direkte Manipulation: Die Hand verwandelt sich selbst in ein Werkzeug.

6.4.1 Virtuelle Werkzeuge

Werden zur Modellierung virtuelle Werkzeuge in Form von Handles eingesetzt, so resultieren daraus zwei Vorteile:

- Punktgenaues Arbeiten ist möglich (Skalpell)
- Die Komplexität wird eingeschränkt (Bohrer).

Diese Reduzierung der Freiheitsgrade und Manipulationsmöglichkeiten ist ein wichtiges Hilfsmittel bei der Erstellung von virtuellen Welten. Ferner zeigt die gewählte Werkzeugform intuitiv, welche Interaktionstypen möglich beziehungsweise nicht möglich sind.

7 Anwendung

7.1 Radiologische Diagnose

Im Gespräch mit Radiologen zeigte sich, daß eine digitale Verarbeitung von radiologischen Daten durch die vielfältigen Bildverarbeitungsmethoden zwar äußerst flexibel und interessant ist, für die Diagnose aber trotzdem auf die Original-Röntgenbefunde zurückgegriffen wird. Der Hauptgrund dafür ist, daß eine herkömmliche Röntgenaufnahme feinere Grauabstufungen zeigt. Als ausreichend wurden lichtstarke, monochrome Monitore mit 12 Bit Grauwertauflösung genannt. Solche Monitore sind aber bisher nicht erhältlich. Zur Begegnung dieses Problems wird in radiologischen Systemen wie der MEDStation eine Zweibildschirmlösung angeboten, bei der ein selektiertes Bild auf einem zweiten monochromen und lichtstarken Monitor formatfüllend abgebildet wird. Dabei können die Grauwertschwellen beziehungsweise Grauwertbereiche beliebig von den ursprünglich 12 Bits auf 8 Bits abgebildet werden. Bezogen auf eine künstliche 3D-Welt bleibt die Frage, wie eine Zweibildschirmlösung realisiert werden kann, ohne den (semi-)immersiven Eindruck zu zerstören. Ein Zoomen auf das eigentliche Bild innerhalb der gleichen Umgebung, wie in der Virtual Lightbox realisiert, reicht aus den vorgenannten Gründen nicht aus.

7.2 VR-Techniken in Anwendungen

Im Rahmen dieser Arbeit wurde ein Prototyp (SSTViewer[18]) entwickelt, der die Erprobung von verschiedenen 3D-Eingabegeräten wie Headtracker und Spaceball mit Stereomodus sowie Metaphern ermöglichte. Der Spaceball erwies sich als äußerst empfindlich und somit schwer zu navigieren. Dies führte dazu, daß eine Filterung eingeführt wurde, die nur die stärkste Teilkraft der Translation oder die Hauptrotationsachse weitermeldet. Sinnvoll ist dieser dominante Modus zusammen mit dem Walkmetaphor, welcher nur eine Rotation um die Längsachse des Benutzers erlaubt. Die Kopplung der Spaceball- mit der Kopftrackereingabe erfolgte einmal in der Form, daß im Walkmodus der Spaceball die Bewegungsrichtung des Körpers im Raum vorgab und die Trackereingabe der Blickrichtung des "darauf aufgesetzten" Kopfes entsprach. In der zweiten Implementierung wurde diese Trennung aufgehoben und die momentane Blickrichtung wurde zum Ausgangspunkt der nächsten Bewegung. Um den vorherigen

[18] Abkürzung für **S**tereo-**S**paceball-**T**racker-Viewer

Vergleich zu gebrauchen, entspricht dies nur der Bewegung des Kopfes, hervorgerufen durch beide Geräte, die entsprechend aufeinander abgestimmt sein müssen. Es zeigte sich, daß das Trackersignal sehr störanfällig ist und der Benutzer von einem Bild zum anderen desorientiert wurde. Auch dieses Verhalten mußte softwaremäßig gefiltert werden. Neben der schon angesprochenen Halbierung der Auflösung und Bildwiederholungsfrequenz im Stereomodus des Silicon Graphics-Rechners gab es noch ein schwerwiegenderes Problem beim Selektieren. Die Ursache davon ist, daß bei diesem Rechnertyp die ungeraden Zeilen das erste und die geraden das zweite Stereohalbbild bilden. Bei dieser internen Umkonfiguration des Bildschirmspeichers werden aber die y-Positionen beeinflußt. Das erste Halbbild enthält nicht die y-Positionen 1,3,5...,383 sondern 1,2,3 und das zweite Halbbild 384, 385,..., 768 bei 1024x768 Auflösung. Laut Silicon Graphics gibt es keine Lösung für das vorgestellte Selektionsproblem im obigen Stereomodus.

Da im Stereomodus nacheinander zwei Bilder gerendert werden, geht dies ebenfalls auf Kosten des Echtzeitgefühls. Letztendlich führen die erwähnten gesundheitlichen Aspekte und diese Ergebnisse zur Erkenntnis, daß Stereo und 3D-Eingaben ein Akzeptanzproblem bei Neuanfängern oder medizinischem Personal hervorrufen und den Effekt der einfachen Bedienung in einer vertrauten Welt zunichte machen würden. Auf der anderen Seite möchte der Autor nicht ausschließen, daß in bestimmten Bereichen 3D-Eingaben und Stereoausgabe sinnvoll sein können.

Desweiteren sei nochmals erwähnt, daß die Idee, künstliche Objekte wie die Kegelbäume oder die perspektivische Wand für die Informationsvisualisierung in die Szene zu integrieren, aus Gründen der Übersichtlichkeit und Vertrautheit fallen gelassen wurde. Einerseits wurden diese Objekte als störend empfunden. Die Interaktion mit realen und künstlichen Objekten erfordert ein mentales Umschalten bei der Interaktion, da verschiedene Metaphern verwendet werden, und andererseits war kein interaktives Arbeiten möglich.

7.3 MEDStation

Das WSI/GRIS entwickelte eine modulare medizinische Applikation, welche die PACS Workstationfunktionalität mit integrierten Case Tools vereint [EG94,GF95]. Das System mit dem Namen MEDStation basiert auf Server/Client-Architektur, das heißt, die Benutzungsoberfläche wurde von der Datenhaltung und Datenverarbeitung getrennt.

Programme wie die MEDStation werden von einer breiten Schicht unterschiedlicher Benutzer angewandt. Diese reicht von den Krankenpflegern, Ärzten, Technikern bis zu Forschern,

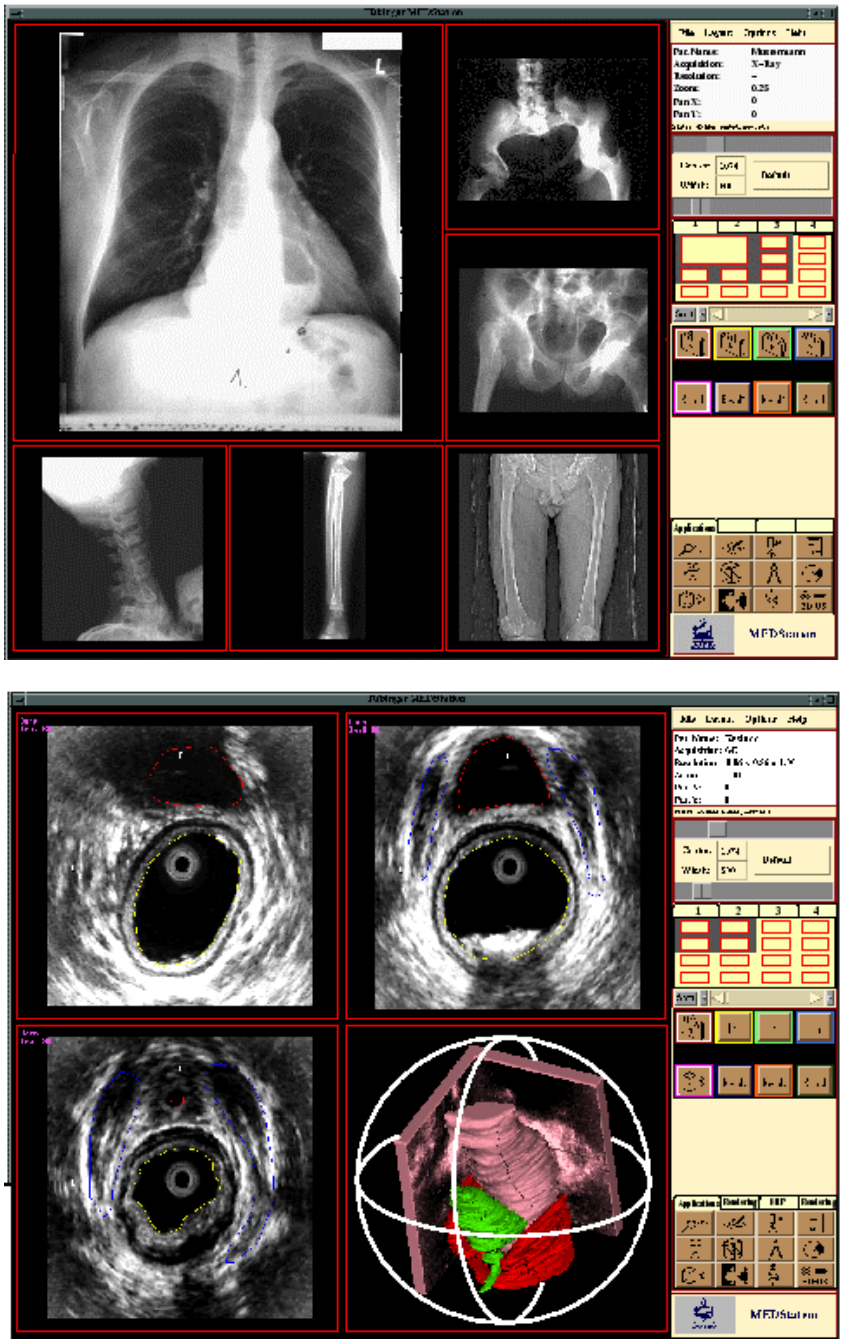


**Abbildung 27: a.) Diagnose mit konventionellen Röntgenbildern in 2D.
b.) 3D-Interaktionsunterstützung für 3D-Endosonographie.
Rechts das Menü mit dem nachgebildeten Lichtkasten Editor.**

wobei jede Berufsgruppe unterschiedliche Erfahrungen und Fertigkeiten im Umgang mit Computern hat. Ein Sammelsurium zu vieler Funktionen muß vermieden werden, während eine Adaption der Benutzungsschnittstelle am Grad der Fähigkeiten und den Aufgaben des Benutzers wünschenswert wäre. Die Benutzbarkeit des Systems ist folglich ein wesentlicher Faktor für den Erfolg solch einer Applikation. Der Benutzungsoberfläche sowie der Transparenz der möglichen Interaktionen kommt eine große Bedeutung zu. Die Mensch-

Maschine-Schnittstelle der MEDStation orientiert sich deshalb an drei Elementen, die im Alltag des Mediziners vorkommen: Der Studie, die verschiedene Daten des Patienten zu einem Fall enthält, (Bild-)Mappen, in die die Studie selbst wiederum untergliedert ist und, als drittes Element für den Umgang mit den Bildern, wurde die Lichtkasten-Metapher herangezogen.

- Dementsprechend wurde auch das Design der Benutzungsoberfläche ausgelegt. Der größte Teil des Bildschirmes wird für die Bildausgabe verwendet, das Menü selbst nimmt nur einen relativ kleinen Platz ein. In Analogie zu einem konventionellen Alternator werden die folgenden Charakteristika von Bedienung und Aussehen genutzt:
- Es gibt ein Bedienungselement, das die Segmente des Lichtkastens graphisch anzeigt. Die einzelnen Segmente nehmen je ein Bild auf und können unterschiedliche Größe haben. Die ganze Ansicht entspricht somit einem Panel. Dieses Element wurde Lightbox-Editor genannt.
- Dieser Lightbox-Editor wurde wiederum in eine Reiterliste eingebracht, wodurch verschiedene Panels angewählt werden können.
- Ein Bild oder eine Studie kann mittels einer Drag&Drop-Aktion auf einen der vorgesehenen Lichtkastensegmente gebracht werden.
- Eine Reorganisation/Umhängen der Bilder kann ebenfalls durch Drag&Drop am Lightbox-Editor erfolgen.

Die Beobachtung, daß Benutzer, die noch keine Erfahrungen mit Computern oder mit computergestützten Bildverarbeitungsprogrammen hatten, trotz der ihnen bekannten Lightbox-Metapher erhebliche Schwierigkeiten mit der Bedienung hatten, führte mit zu der Idee, eine echte virtuelle 3D-Umgebung mit Teilfunktionalität zu entwerfen.

7.4 Unsere neue Umgebung: Virtual Lightbox

Für die Umsetzung einer Virtuellen Umgebung wählte der Autor den Arbeitsplatz eines Radiologen, genauer den Befundungsraum. Nach einer Analyse vor Ort schlüsselten der Verfasser die typisch anfallenden Aufgaben folgendermaßen auf:

- Holen der Informationen aus einem Archiv (-schrank)
- Entnahme der Bilder aus der Patientenmappe
- Befestigen der Bilder an einem Alternator/Lichtkasten
- Vergrößern und Erhellen von Bildausschnitten
- Analyse und Ausmessen der Bilder mittels Lineal und Zirkel
- Anbringen von Markern und schriftlichen Notizen am Bild
- Andere Ansicht/Bilderfolge am Alternator auswählen
- Umhängen von Bildern
- Befund in Diktiergerät sprechen
- Konsultation von einem Kollegen oder Buch/Nachschlagewerk bei Unklarheiten
- Zeigen von Referenzbildern zur Weiterbildung für Kollegen

Entsprechend verwendeten wir folgende Komponenten zur Realisierung dieser vereinfachten Virtuellen Umgebung :

- Untersuchungsraum
- Alternator mit Fernbedienung
- Archivschrank des Patienten
- Mappe mit Patientendaten wie Bildern und anderen Dokumenten
- Röntgen/Kernspin-Bilder
- Werkzeuge wie Lineal, Vergrößerungsglas, Marker, Bleistift, Papierkorb
- Andere Hilfsmittel wie Mikrofon/Diktiergerät, Nachschlagewerk, Videokamera

Bei der Realisierung der Hilfsmittel durch die Einbindung externer Applikationen stellten sich besondere Probleme heraus:

Mikrofon/Diktiergerät: Die Aufzeichnungsdauer mußte vorher festgelegt werden. Ein interaktives Diktieren ist nicht möglich.

Nachschlagewerk: Das Einblenden von Text in die 3D-Umgebung, auch im halbtransparenten Modus, störte. Für interaktives Suchen sollten Kegelbäume oder ähnliches genutzt werden.

Video: Eine direkte Videostream-Integration bot die API nicht an, somit wurde der Umweg über das Zwischenspeichern von Bildern gewählt. Die Ausgabe erfolgt dann als Textur auf ein Element in der 3D-Umgebung. Dies führte zu sehr niedrigen Bildraten.

Der alternative Weg, jeweils ein eigenes Fenster außerhalb der 3D-Umgebung für diese Hilfsmittel einzusetzen, hätte den semiimmersiven Eindruck zerstört und ginge auf Kosten der Benutzbarkeit. Die gewonnenen Erkenntnisse führten zu dem Schluß, diese Hilfsmittel nicht weiter einzusetzen.

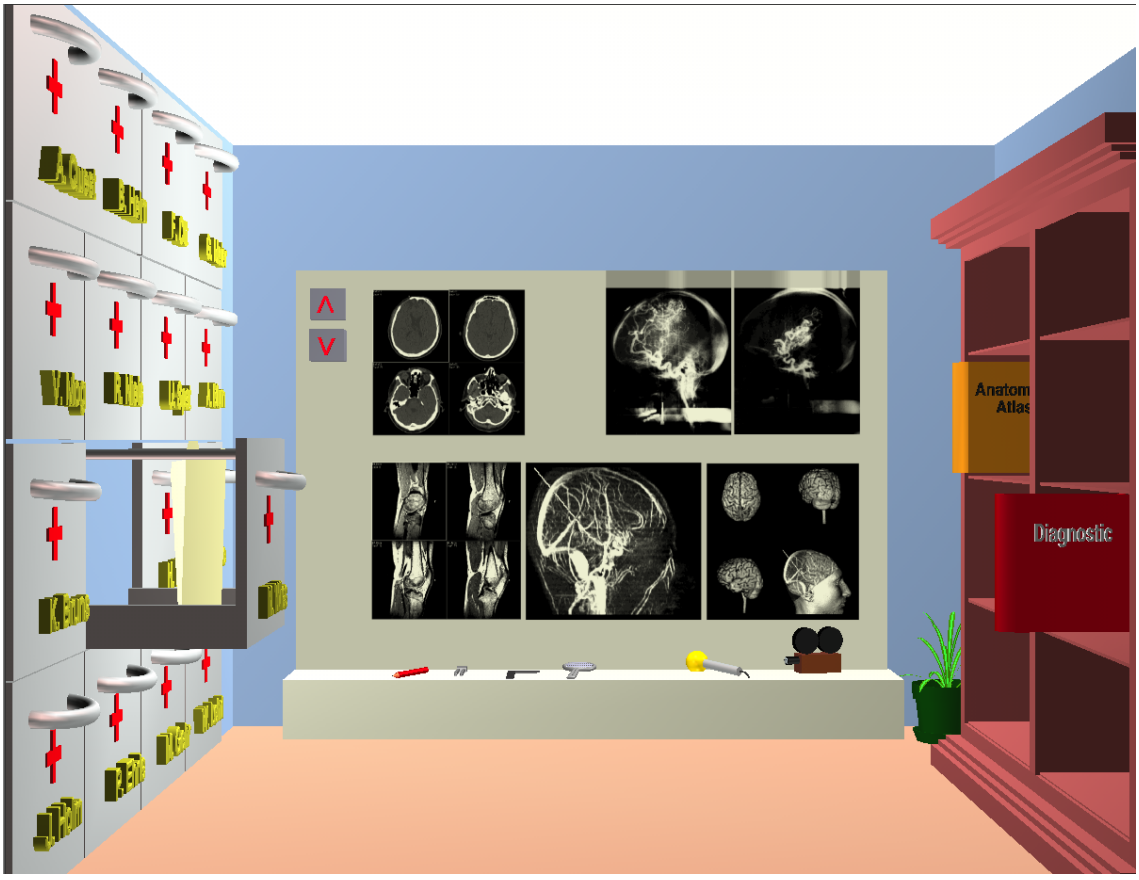


Abbildung 28: Virtual Lightbox Realere Umgebung ohne Animation

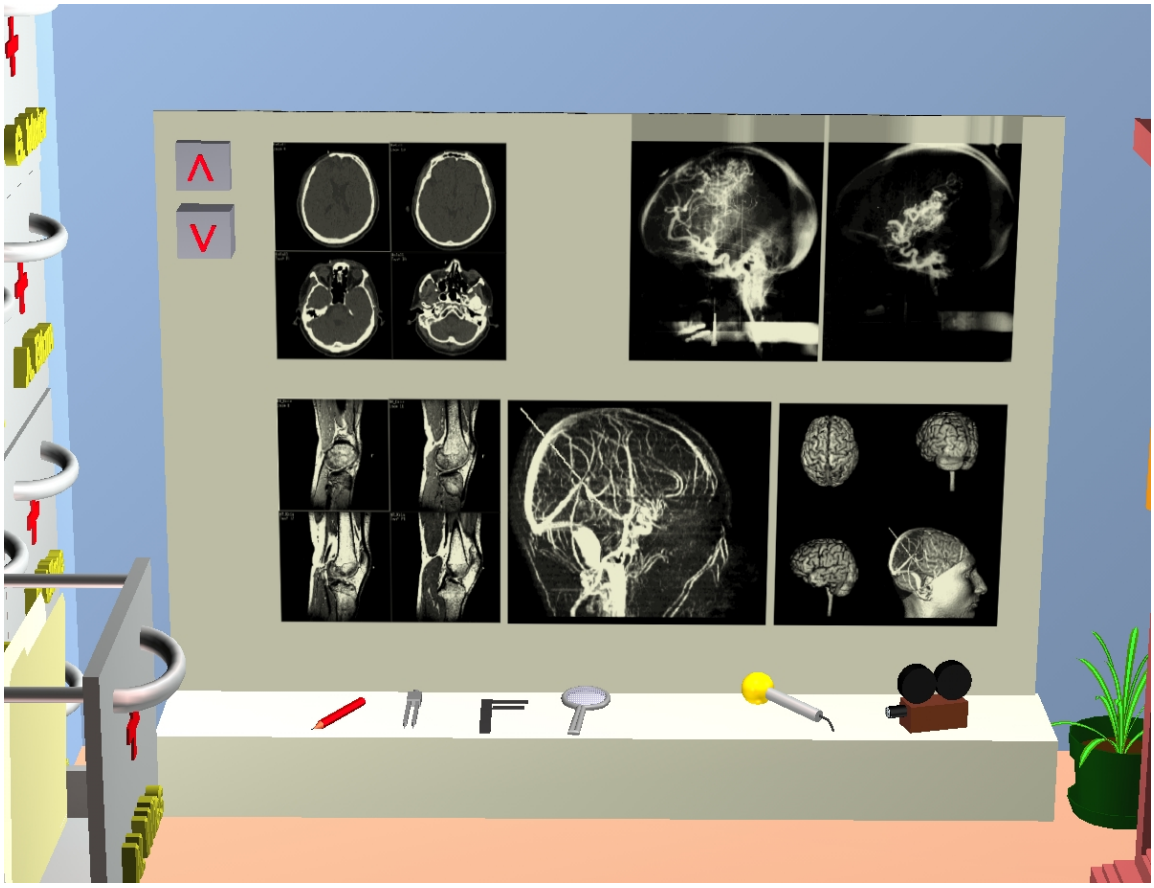


Abbildung 29: Aufsicht auf die Werkzeuge

7.5 Real-Time Rendering und automatisches, intelligentes Verhalten

Ein Benutzer wird bald ungeduldig, wenn er kein schnelles Feedback auf seine Eingaben bekommt. Auf dieses Problem stießen wir auch bei der ersten Prototypstudie. Die Umgebung war sehr aufwendig modelliert und die Antwortzeiten des Graphiksubsystems dauerten schon bei einfacher Navigation zu lange. Bei zusätzlicher Animation von Objekten verschlechterte sich das Antwortverhalten noch mehr. Um den Benutzern dennoch ein Echtzeitgefühl während einer Interaktion zu vermitteln, mußte die Szene vereinfacht werden (siehe Abbildung 30). Ein anderer Ansatz, der mit D. Schick und W. Broll verfolgt wurde, war die Unterteilung von großen Szenen in logische, relevante und beidseitig zusammengehörige Untereinheiten [BF96, BF98]. Zum Beispiel erfolgt die Partitionierung eines Hauses durch verschiedene Räume, die mittels Türen oder Aufzügen zusammenhängen.

Es zeigte sich ferner, daß Interaktionen, die der natürlichen Welt nachgebildet werden, zwar intuitiv sind, der Interaktionsvorgang selbst aber zu lange dauert. Ein Objekt zu greifen und im 3D-Raum zu bewegen ist äußerst mühsam. Da es auch keine benutzerfreundlichen 3D-Geräte

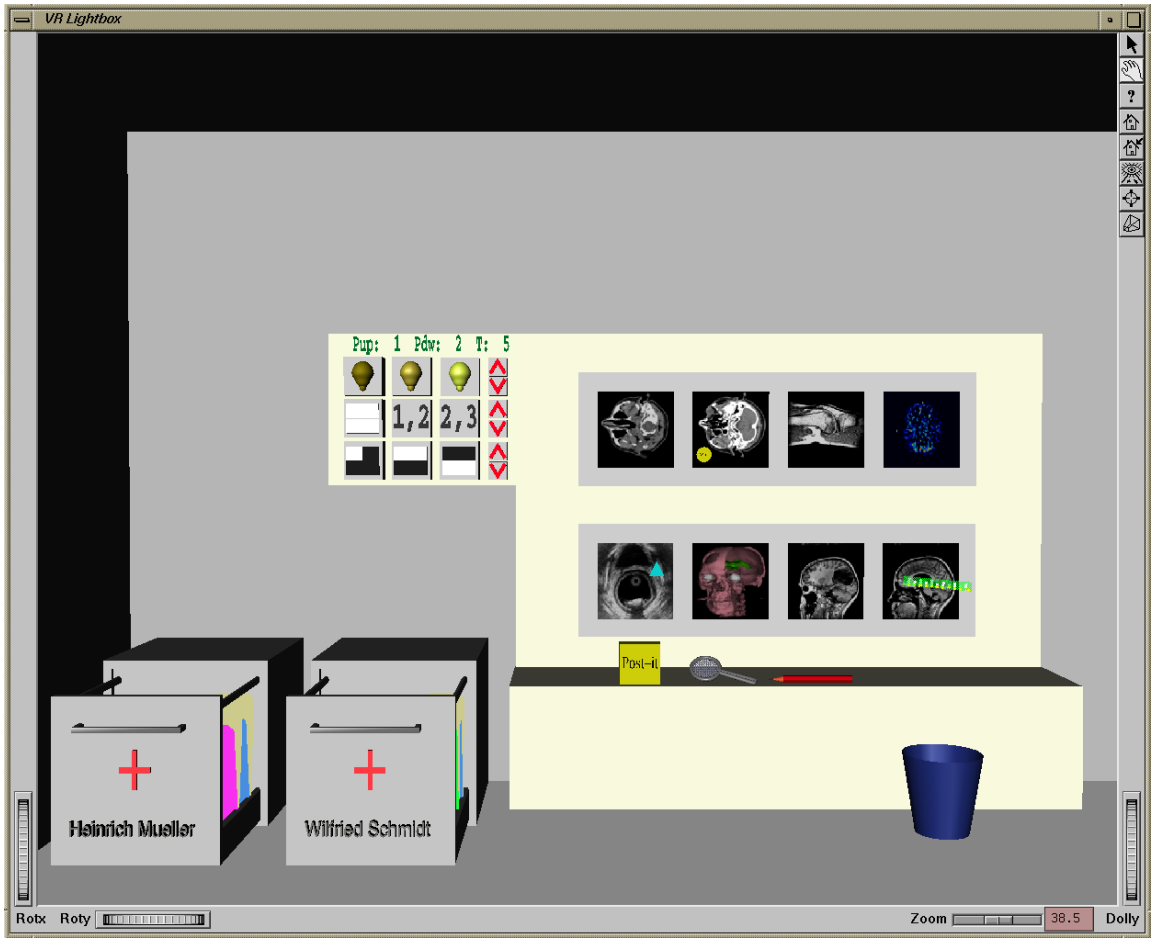


Abbildung 30: Vereinfachte animierte Szene

gibt, die diese Aufgabe einfach und schnell bewerkstelligen können, gingen wir einen anderen Weg. Die Idee besteht darin, daß die Objekte selbst automatisches (intelligentes) Verhalten erhalten, um die erforderlichen Interaktionen des Benutzers zu minimieren. Dies geschieht beispielsweise beim Öffnen einer Röntgenmappe, wobei automatisch die Bilder an die freien Alternatorpanelplätze gehängt werden, oder beim Schließen der Patientenarchivbox werden alle zugehörigen Objekte zurückgesetzt. Wird ein Werkzeug, beispielsweise der Stift, ausgewählt während noch ein Bild selektiert ist, so wird automatisch ein Label angebracht und es erfolgt eine Ausschnittvergrößerung. Ein anderer Punkt, der Probleme bereitet, betrifft die Armut der zur Verfügung stehenden Eingabemöglichkeiten. Um diese zu erhöhen, wird nacheinander in die verschiedenen Interaktionsformen gewechselt. Beim ersten Mausklick wird selektiert, beim nächsten deselektiert, beim dritten Klick in den Manipulationsmodus gegangen und schließlich

beim vierten dieser wieder verlassen. Mehr als zwei oder drei Ebenen scheinen nicht sinnvoll, da dies zu viel Zeit kostet und für den Benutzer nicht mehr einsichtig wäre. Die Benutzung von

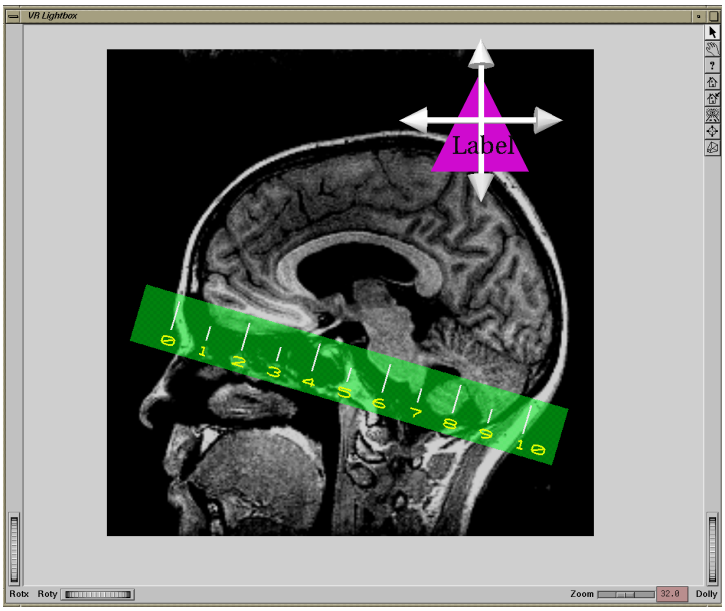


Abbildung 31: Ausschnitt des automatisch gezoomten Fensters mit teiltransparentem Lineal und Label, welches gerade verschoben wird. Um die Interaktionsmöglichkeiten zu erhöhen, wird beim ersten Klick zum Beispiel auf das Label selektiert, beim zweiten deselektiert, beim dritten erscheint der Manipulator (wie dargestellt) und beim vierten Mausklick wird dieser wieder verlassen.

Modifiziertasten, wie Steuerung und Umschalttaste, sind zwar auch machbar, die Benutzertransparenz wird aber gesenkt.

7.6 Neue Wege der Interaktion

Bei dreidimensionalen Räumen ist eine Interaktion aufwendig und zeitintensiv. Einerseits ist dies bedingt durch die auftretenden langen Wege bis ein Benutzer, zum Beispiel mittels eines 3D-Cursors, zum eigentlichen Objekt gelangt ist. Und anderseits muß noch eine komplexere Aktion erfolgen, wie die Selektion, Kollision oder Manipulation von bzw. mit Objekten. Aber gerade diese Zeitverluste, das umständliche Hantieren mit dem Objekt und das insgesamt aufwendige Handeln um schnell und erfolgreich zum erwarteten Ziel zu kommen, führt zur Frustration der Benutzer und zur Nichtakzeptanz von 3D-Applikationen. Diese eigenen Erfahrungen mit Interaktionsvorgängen im dreidimensionalen Raum führen zu folgenden Anforderungen an die Applikation:

- Minimierung der Wege
- Automatisierung (von Bewegungsabläufen)
- Reduzierung der Interaktionen

Für die oben aufgezählten Punkte führten wir den Begriff *Objekte mit intelligentem Verhalten* ein. Ferner stellte sich die Frage, wie bei einer Interaktion vorgegangen werden soll. Wir unterscheiden hier nach der Wirkungsrichtung. Das Objekt, welches die Aktion auslöst, bezeichnen wir als aktiv und jenes, welches die Wirkung erfährt, als passiv. Neben dieser Wirkungsrichtung gibt es noch die räumliche sowie die zeitliche Komponente der Interaktion. In ersterem geht es um Fragen wie beispielsweise bei Drag&Drop-Interaktionen: "Wer wird zu wem gebracht?" und die zeitliche Komponente beinhaltet Parameter wie: "In welcher Reihenfolge und wie schnell erfolgt die Interaktion?". Die reine Interaktionswirkung besteht im einfachsten Fall aus zwei Objekten und kann folgendermaßen beschrieben werden:

- Aktives Objekt $\xrightarrow{\textit{Interaktionswirkung}}$ Passives Objekt (Fall 1)

Neben diesem häufigen Fall gibt es noch folgende zwei Spezialfälle, bei denen beide Objekte gegenseitig aufeinander einwirken:

- Aktives Objekt 1 $\overset{\textit{Interaktionswirkung}}{\longleftrightarrow}$ Aktives Objekt 2 (Fall 2)

Ein Beispiel hierfür ist die Beziehung zwischen Erde und Mond, bei der auch die Rolle der räumlichen Komponente offensichtlich wird. Interagieren die beiden Objekte gleichzeitig miteinander, Objekt 1 zuerst mit Objekt 2, oder Objekt 2 zuerst mit Objekt 1, so kann dies zu unterschiedlichen Resultaten führen (Beispiel Billardspiel).

Beim Gegenstück hat die Interaktion keine Auswirkungen auf die Objekte und somit spielen weder zeitliche noch räumliche Parameter eine Rolle:

- Passives Objekt 1 $\overset{\textit{Interaktionswirkung}}{\leftarrow\|\rightarrow}$ Passives Objekt 2 (Fall 3)

Aus dem vorangegangenen stellt sich folgende Frage: Kann aufgrund der Wirkungsweise ein Interaktionsschema abgeleitet werden, das den Benutzern transparent erscheint? In einer Virtual Reality gibt es Interaktionen, die in der realen Welt nicht vorkommen und sich somit dem Benutzer nicht einfach erschließen. Betrachtet man den Fall 1 unter dem Gesichtspunkt der räumlichen Interaktionskomponente, könnte ein Benutzer ein Werkzeug zu einem anderen Gegenstand via Drag&Drop bringen (Zum Beispiel eine Lupe zum Röntgenbild in unserer medizinischen Anwendung). Ursprünglich war geplant, sämtliche Aktionen nach diesem Wirkungprinzip abzubilden, doch rasch zeigte sich, daß eine Verallgemeinerung nicht sinnvoll ist. Angenommen, wir hätten ein Dokument und wollten es vernichten, so würde der Benutzer

das Dokument intuitiv in den Papierkorb ablegen. Hier wäre also schon das Prinzip durchbrochen. Die Umgewöhnung von bekannten Interaktionsweisen fällt den Benutzern sehr schwer und birgt eine zusätzliche Fehlerquelle in sich. Die Maxime heißt also, die Vorgänge so weit wie möglich der realen Welt nachzubilden und im Zweifelsfalle, wenn beide Interaktionsformen sinnvoll sind, die der Interaktionswirkung vorzuziehen.

8 Verhalten und Kommunikation von Objekten in 3D-Benutzungsoberflächen

8.1 Ereignisse, Ereignisverarbeitung und Nachrichten

Ein Ereignis (engl. Event) geht in der Regel mit der Änderung eines Zustandes einher. Dabei können verschiedene Ebenen des Systems als Verursacher unterschieden werden. Auf Hardwareebene werden die Zeitgenerierung oder die Benutzereingaben durch Tastatur oder Maus, meist Interrupt gesteuert, an das Betriebssystem weitergegeben. Änderungen auf höherer Ebene, wie das Verschieben von Fenstern, werden auf Betriebssystemebene abgehandelt. Alle Windows-Systeme haben dazu eine Instanz, den Windows-Manager, der die Ereignisse in einer endlosen Hauptschleife aufnimmt und an die anderen Subsysteme weiterleitet. In diesem Zusammenhang wird auch von Ereignisbearbeitung gesprochen. Ein Ereignis ist also der Auslöser, der zur Generierung einer Nachricht führen kann. Durch einen Mausklick wird eine Systemnachricht durch den Windows-Manager erzeugt und dem Windows-System zur Verfügung gestellt. Jedes Windows-System wie X-11 oder Windows-32 hat aber ein eigenes Format mit unterschiedlichen Parametern für diese Systemmeldungen. Folglich braucht man eine Schnittstelle, die auf dieser betriebssystemabhängigen Schicht aufbaut und eine einheitliche, über die Plattform reichende, Kommunikationsschnittstelle schafft. Softwaretechnisch bietet dazu jedes System die Funktion, sich in die Hauptnachrichtenschleife einzuhängen und die Meldungen an die Applikation weiterzugeben.

8.1.1 Ereignisverarbeitung in Open Inventor

Open Inventor hat die Eigenschaft, daß es die Ereignisse des Betriebssystems in seine eigenen Ereignisklassen übersetzt und diese an die Inventor-Objekte, auch Szenendatenbank genannt, weiterleitet. Dies war ein Grund, Open Inventor als Programmierwerkzeug zu nehmen. Es ist unabhängig vom Windows-System, und die Anwendung ist auch auf Ereignisebene portierbar. Auf der anderen Seite wird es dadurch möglich, eine eigene Ereignisbearbeitung einzubauen,

welches um selbst entwickelte Funktionen erweitert ist. Die Abbildung 32 zeigt die einzelnen Komponenten und die Ablaufreihenfolge bis ein Ereignis zur Szene, genauer zu einem einzelnen Szenenobjekt, gelangt. Open Inventor bietet drei eigene Ereignisklassen: eine für die 2D-Position, eine für Bewegungen im 3D-Raum sowie eine allgemeine für Tasteneingaben. Letztere wird wiederum je nach Gerät in die Klassen Tastatur, Maus und Spaceball aufgeteilt. Die Eingabe dieser Geräte führt also zur Generierung eines inventoreigenen Ereignisses. Die Weiterleitung des Ereignisses erfolgt durch den Szenenmanager, der eine spezielle Aktion ausführt (engl. Handle Event Action). Dabei wird im Szenengraph nach dem Szenenelement, auch Szenenknoten genannt, gesucht, welches diesen Ereignistyp verarbeiten kann und dieser zu dessen Weiterverarbeitung übergeben. Typischerweise handelt es sich hierbei um eine Selektion oder Manipulation eines Elements.

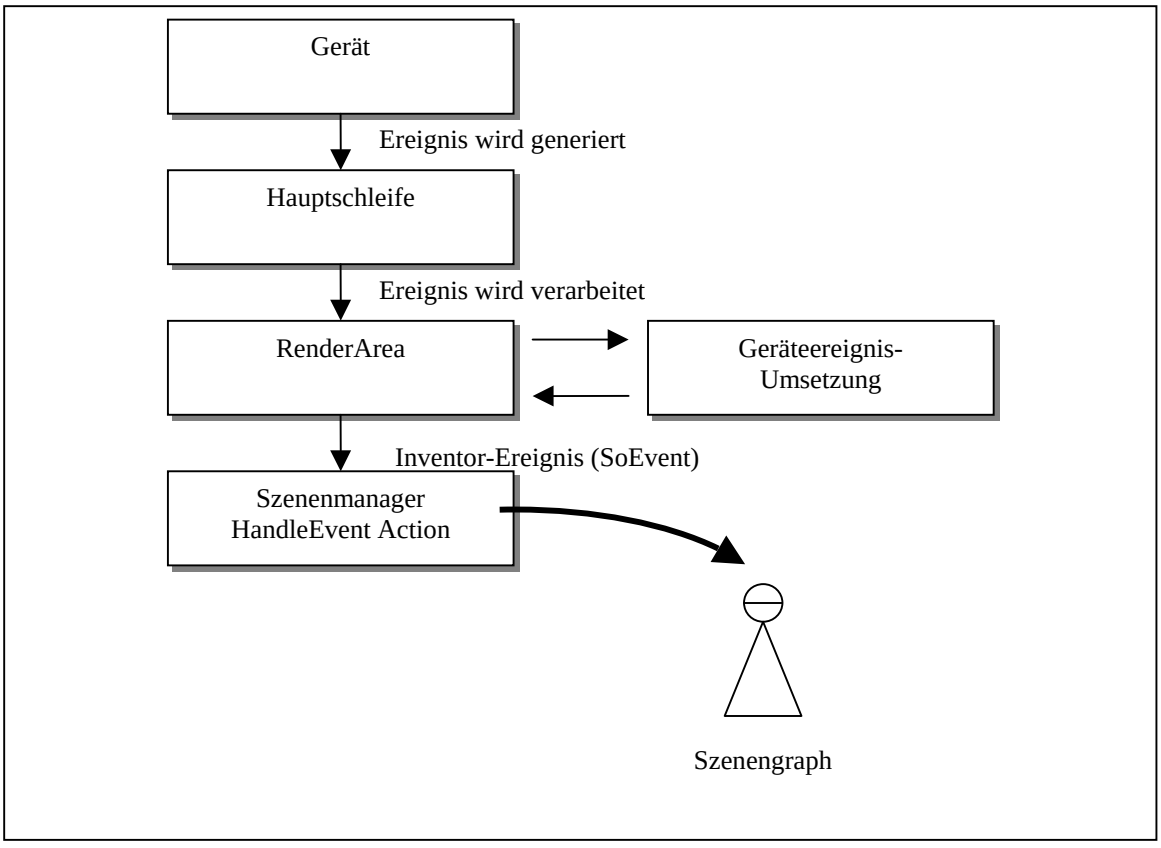


Abbildung 32: Sequenz der Umsetzung eines Ereignisses vom Gerät bis zur Szene

1.) Ein physikalisches Gerät generiert ein Ereignis.

2.) In der Hauptschleife eines Unix oder Win32 Systems werden die Ereignisse an die korrespondierenden Fenster weitergeleitet. Die OpenGL-Grafikausgabe erfolgt bei Inventor in der sogenannten Renderarea.

3.) Danach erfolgt die Umwandlung des betriebsystemabhängigen Ereignisses in ein Inventor-Ereignis (SoEvent)
4.) Dieses SoEvent wird an den SzenenManager weitergegeben, welcher durch eine sogenannte Handle Event Action die Szenendatenbank nach dem Knoten durchsucht, für den dieses Ereignis bestimmt ist und es weiterverarbeitet. Typischerweise ist dies eine Selektion oder Manipulation.

Neben den durch den Benutzer ausgelösten Interaktionen, gibt es *Sensors* und *Engines*. Bei den *Sensors* wird eine callback-Funktion aufgerufen, wenn sich entweder Daten des Szenenelements verändert haben oder zeitgesteuerte Änderungen eingetreten sind. *Engines* dienen dazu, einfache Animationen auszuführen sowie einen Teil der Szene in Abhängigkeit zu einer anderen zu setzen. Wichtig in diesem Zusammenhang ist, daß sich mit *Engines* nur starre ("fest verdrahtete") Abhängigkeiten realisieren lassen. Es sei darauf hingewiesen, daß diese Ereignisse, hervorgerufen durch Benutzereingaben, nur an ein Objekt weitergeleitet und dort verarbeitet werden, daß aber die Objekte nicht im Stande sind, miteinander zu kommunizieren.

8.1.2 Anforderungen an ein neues Modell

Die Restriktion, daß nur einzelne spezielle Elemente von Inventor-Ereignisse austauschen können und dies auch nur in vorher festgelegten rigiden Zuordnungen, soll durch ein neues dynamischeres Modell überwunden werden. Auch Elemente, die während der Laufzeit hinzukommen, sollen sofort in die Kommunikation einbezogen werden. Analog sollen Elemente, die gelöscht werden, keinen Einfluß auf die Stabilität des Systems haben. Das neue Modell muß in das Inventor-Ereignismodell integrierbar sein. Zusätzlich sollen Informationen über den Szenengraph hinaus durch das Netzwerk ausgetauscht werden.

8.1.3 Nachrichten

Unter Nachrichten werden in diesem Zusammenhang, zur Unterscheidung zu Ereignissen, die Informationen verstanden, die zwischen den Szenenobjekten beziehungsweise zwischen den Applikationen netzwerktransparent durch das neue Modell ausgetauscht werden sollen.
Die Informationen selbst lassen sich wie folgt einteilen:

- Diskrete Informationen
- Kontinuierliche Informationen

Bei einer diskreten Nachricht (Informationen) sind die Parameter wie Datentyp und Länge zum Sendebeginn bekannt. Diese sind aber nicht starr festgelegt, sondern können sich von Nachricht zu Nachricht ändern. Im Gegensatz dazu ist bei kontinuierlichen Informationen das

Ende nicht festgelegt und man spricht von einem Datenstrom (engl. Streaming). In diese Rubrik zählen Video- oder Audiodaten, die während einer Konferenz anfallen.

8.2 Anforderung an 3D-Objekte

8.2.1 Vergleich von konventionellen Benutzungsschnittstellen mit Virtuellen Umgebungen

Unsere Analyse von konventionellen 2D-Benutzungsschnittstellen mit Virtuellen Umgebungen zeigt, daß es drei wesentliche Unterschiede gibt:

- Keine standardisierten Objekte
- Kein standardisiertes Verhalten und keine komplizierten Objektbeziehungen
- Keine klare Trennung von Funktion und Repräsentation

	2D-Benutzungsoberfläche	**3D-Virtuelle Umgebungen**
Präsentation	2D-Objekte (z. B. Knöpfe, Laufleisten, Pull-Down-Menü, Cursor) Standardisierter Objektsatz	3D-Objekte als virtuelle Repräsentation von echt existierenden Objekten (z. B. Alternator) KEIN standardisierter Objektsatz
Dialog-Kontrolle	Standardisierte Objektverhalten und einfache Objektbeziehungen /Interaktionen (z.B. Setze Knopf sensitiv)	Kein standardisiertes Objektverhalten und komplizierte Objektbeziehungen (z. B. Hole nächste Alternatortafel)
Anwenderprogrammschnittstelle (engl. Abkürzung API)	Klare Trennung zwischen Applikationsprogramm und Benutzungsoberfläche möglich	Objekte enthalten Applikations-programmfunktionalität

Tabelle 4: Vergleich zwischen konventionellen 2D-Graphik-Benutzungsoberfläche und Virtuellen Umgebungen. (Übersetzt aus: [EF94])

8.3 Standardisierung

Um den obengenannten Hauptproblemen zu begegnen, besteht unser Ansatz in einer Palette von standardisierten Basisobjekten mit standardisiertem Verhalten für Virtuelle Umgebungen. Die Objekte können verschiedene individuelle Repräsentationen annehmen. Ferner haben alle ein gemeinsames Basisverhaltensrepertoire, damit die Objekte miteinander kommunizieren können, sowie die Möglichkeit, neue (spezielle) Verhaltensweisen/Funktionen hinzuzufügen.

Klasse	Repräsentation	Datentyp	Interaktions-aufgabe	Typ der Benutzer-interaktion	Beispiel
Trigger	3D-Button	Binary	Execute	Push Button	Quit Application
Switch	Switch/lever	Logical	Enable/Disable Show/Hide	Switch_to_on/ off	Turn Light on/off
Symbol	Any/3D	Logical	Execute/Call	Select	Camera: connect to collegue
Indicator	Highlighting, emphasizing	Real	State, Warning, Information	Note of ONLY	Red light: Access denied!
Selector	Multiposition lever	Discrete, exclusive	Selection 1_of_n	Set mode	Set sorting mode to patient names
Container	Box, folder, drawer	Discrete objects	Scattered organisation	Open/Close Get/Put	Arrange patientrecord
Hierarchy	Tree	Discrete objects	Organize structure	Browse, Select	Arrange stages of illness
Prospect	Wall, projections, blackboard	Discrete objects	Viewing, sorting, give overview	Multiselection, Drag, Arrage	Virtual lightbox with CT images
History	Time-tunnel	Continuous data	Back- and forthtracking	Follow, Focus_on	Traverse patient history
User	Body(Avatar), camera, invisible	Discrete objects	Change position, navigate	Go to	Move towards the lightbox

Tabelle 5: Klassifizierung von Objekten in Virtuellen Umgebungen [FG96]

8.4 Verhalten und Kommunikation (Messageconcept)

8.4.1 Allgemein

Um die Kommunikation zwischen 3D-Objekten zu gewährleisten, wurde ein Konzept entwickelt und realisiert, um speziell den Bedürfnissen der Kommunikation im 3D-Raum nachzukommen. Im folgenden wird dies unter dem eingeführten Begriff ***Messageconcept*** zusammengefaßt. Zuerst wurde eine allgemeine Fassung unter Inventor (Applikationsobjekte) realisiert und später modifiziert als Kommunikationsplattform für VRML 2.0 (Dynamic

World) eingereicht. Objekte mit vollkommen neuen Eigenschaften wie unter Open Inventor sind nicht möglich. Es können aber neue Objekte aus bereits vorhanden Teilobjekten/ Komponenten geschaffen werden. Ein wesentlicher Unterschied zum konkurrierenden VRML 2.0 (Moving World) ist, daß unser Ansatz (Dynamic World) ebenfalls wie das Messageconcept,

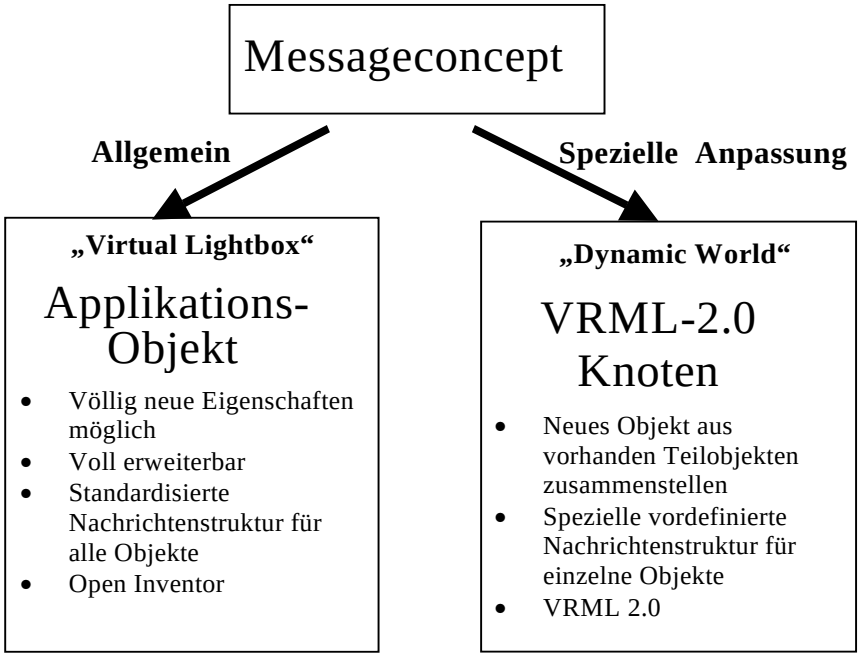


Abbildung 33: Anpassungen des Messageconcept an die unterschiedlichen Umgebungen Open Inventor und VRML 2.0 (Dynamic World)

eine flexible dynamische Adressierung ermöglicht. Damit unterstützt diese spezielle Anpassung des Messageconcept auch Objekte, die zur Laufzeit erzeugt werden, und ist zusätzlich noch netzwerktransparent (näheres siehe Kapitel 8.5).

8.4.2 Relevanz der Information

Beim Austausch von Information kann diese verloren, verstümmelt, dupliziert oder in der zeitlichen Reihenfolge vertauscht werden. Dieser Informationsverlust wird nicht innerhalb des Nachrichtenaustausches in einer Szene auftreten, sondern beim Einsatz des Netzwerks als Übertragungsmedium. Die Wichtigkeit der Information selbst, aber auch die der zeitlichen Reihenfolge des Eintreffens, kann in folgende Fallunterscheidungen aufgeschlüsselt werden:

Unabhängige Nachrichten

Die vorauseilende/nachgelieferte Information hat keinen Einfluß auf die vorhergehende/nachfolgende Nachricht. Es handelt sich um unabhängige Nachrichten innerhalb eines Zeitfensters.

Überholte Nachrichten

Im anderen Fall muß die nachgelieferte Nachricht verworfen werden, da ihr Informationsgehalt veraltet und inzwischen durch eine neuere Nachricht aktualisiert wurde.

Abhängige Nachrichten

Eine verlorene, doppelte oder zeitlich verschobene Information führt zu einem falschen Ergebnis, weshalb eine Fehlerkorrektur zu erfolgen hat. Dies kann zur Anforderung des verlorenen Informationpaketes, zum Verwerfen der doppelten Information oder gar zur Anforderung der ganzen vorherigen Sequenz führen. Zur Erkennung solcher Fehler ist das Protokollieren der genauen Zeit unabdingbar und muß in dem Informationsobjekt mit aufgenommen werden. Ebenso kann eine fortlaufende Nummer dafür wertvolle Dienste leisten. Zusätzliche Mechanismen, wie die Prüfsumme, können bei verbindungslosen Protokollen den korrekten Inhalt absichern; oder aber Rückantworten zum Sender verifizieren die bisher eingegangenen Informationspakete. Als einfaches Beispiel für abhängige Nachrichten sei der Fall, bei dem auf eine Rotation eine anschließende Translation folgt, angeführt. Eine irrtümliches Vertauschen würde zu einem völlig anderen Ergebnis führen.

Die Unterscheidung jedes Einzelfalls wäre softwaretechnisch möglich aber aufwendig und nicht praktikabel, denn dies würde bedeuten, daß der Inhalt der Information analysiert werden müßte und je nach Informationskontext verschiedene Mechanismen angewandt würden. Zur Lösung dieser Konflikte wird deshalb eine einheitliche Strategie verwendet: Grundsätzlich müssen alle Informationspakete abgearbeitet werden. Die Verpackung der Information in das zugehörige Protokoll richtet sich nach der Anzahl der Empfänger, grundsätzlichen Wichtigkeit der Information (Informationsklasse) und Aktualisierungshäufigkeit, welcher die Information unterliegt. Abhängig vom Protokolltyp werden die Techniken zur Wiederherstellung (engl. recovery) gewählt. Wichtige Nachrichten, wie sie bei der Initialisierung anfallen, müssen sicher transportiert werden. In diesem Falle bietet sich die Verwendung des TCP/IP Protokolls an.

Die Wichtigkeit von Nachrichten kommt bei der Synchronisation zwischen verschiedenen (Teil-)Szenen oder Benutzern in verteilten Multi-User-Systemen zum Tragen. Hier erfolgt die Synchronisation je nach Typus verschieden schnell und wird mit unterschiedlichen Techniken

realisiert. Im folgenden wird die Relevanz von Nachrichten bezüglich ihrer Synchronisation mit zunehmender Wichtigkeit aufgelistet:

- Änderungen ohne Einfluß auf Gesamtsystem, deshalb ist keine Synchronisation notwendig (lokal beschränkte Änderungen).
  Schneetreiben, Wasserfall, Drehen der Räder, Art des Gehens, zufällige Änderungen (wie Funkeln von Sternen).

- Änderungen mit zeitweisem Abgleich bei Objekten mit sekundärem Einfluß.
  Sonnenaufgang, Wolkenbewegung, Verhaltensänderungen.

- Änderungen mit deterministischem, autonomen Verhalten mit primärem Einfluß
  Freier Fall, ballistische Kurve.
  Änderungen werden lokal mittels Formeln berechnet und von Zeit zu Zeit mit der tatsächlichen Position abgeglichen (siehe auch Dead Reckoning).

- Änderungen/Bewegungen mit sofortiger Synchronisationsnotwendigkeit mit primärem Einfluß.
  Benutzeraktionen wie Greifen eines Gegenstandes.
  Exklusiver Zugriff, gemeinsamer konkurrierender oder kooperativer Zugriff (siehe auch Locking).
  Schnelle und sichere Informationsverteilung mittels modifiziertem Multicasting-Protokoll.

8.4.3 Nachrichtentypen

Beim Datenfluß wird unterschieden in Steuerungsbefehle, den Nachrichtenaustausch mit kurzen Nachrichten, optional mit Parameter (Datenbanksatz) und langen Nachrichten (Multimedia-Datenstrom). Die Nachrichtentypen lassen sich auch nach ihrer Wirkung unterscheiden und führen zur folgender Klassifikation:

- Nachrichten, die zur Auslösung einer Aktion führen.
- Nachrichten, die Informationen über die Applikation, Szene oder einzelne Szenenobjekte enthalten.

Der zweite Typ von Nachrichten kann Statusinformationen enthalten. Diese können benutzt werden, um erst in Abhängigkeit von anderen Nachrichten Aktionen auszulösen oder das Verhalten der Objekte zu modifizieren. Diese nachrichtenbasierte Kommunikation zusammen mit dem Umstand, daß die Beziehungen zwischen den Objekten nicht starr festgelegt sind, liefert eine hohe Flexibilität. Beispielsweise können neue Objekte in die Szene integriert oder existierende Objekte mittels Nachricht gelöscht werden.

8.4.4 Objektnamen

Dieses Kommunikationsmodell geht von der Annahme aus, daß jedes Objekt eindeutig adressierbar ist. Dies wird durch einfache Benennung erreicht. Um auch die Gruppierung der Objekte in logische oder natürliche Einheiten zu unterstützen, werden Namensteile durch einen Punkt oder Unterstrich separiert. Speziell für den Netzwerkbereich wurde noch das @-Symbol als weiteres Separationszeichen hinzugefügt. Das Benutzen dieser hierarchischen Namensgebung ist essentiell für die Kommunikation in großen VR Szenen. Soll beispielsweise in einer Szene eine Tür geöffnet werden, so ist die Nachricht an den Adressaten der Form Ort.Straße.Hausnummer.Zimmer.Türe intuitiver und einfacher zu bewerkstelligen als mit einem Konstrukt T3012a. Zusätzlich können Wildcards wie "*" oder "?" für die Namensteile verwendet werden. Auf diese Art können mit dem Ausdruck "alternator.panels.*" alle oder "alternator.panels.1?" die Bildtafeln 10 bis 19 angesprochen werden. Ein kombinierter Einsatz dieser beiden Wildcards ist möglich. Wird nur ein einzelner Stern "*" als Namen verwendet, werden alle Objekte angesprochen. Mittels dieser Objektnamensgebung können spezielle Objekte, wie sogenannte Avatars in Multi-User-Applikationen oder Dienste zu externen Serviceprogrammen, verwirklicht werden [BE96b].

8.4.5 Nachrichtenaufbau

Ein wichtiges Element für die Kommunikation der Objekte ist der Aufbau der Nachrichten selbst. Eine Analyse der diskreten Nachrichten führte zu dem Schluß, daß sich diese nach dem selben Schema aufbauen lassen und in folgende Untereinheiten aufschlüsselbar sind:

- Absender (Von wem?)
- Sender (Zu wem?)
- Daten (Was?)
- Zeitstempel (Wann?)

Die Nachrichten wurden naturgemäß als gekapselte Objekte aufgefaßt und implementiert.

Absender der Nachricht:

Im Absender wird der eindeutige Name des Objekts festgelegt, welches die Nachricht verschickt. Diese Information kann vom Empfänger mitverwertet und zu einer veränderten Reaktion oder zur Rücksendung verwendet werden.

Empfänger der Nachricht:

Der Empfänger einer Nachricht kann ein bestimmtes Objekt oder mehrere unterschiedliche oder zu einer Gruppe gehörenden Objekte sein. Letzteres geschieht durch die Adressierung mittels Wildcards.

Zeitstempel:

Mit Hilfe des Zeitstempels wird der Zeitpunkt des Abschickens der Nachricht festgehalten. Etwaige Laufzeitunterschiede, die beim Versenden der Nachricht über das Netzwerk auftreten, können so erfaßt werden. Die zeitliche Reihenfolge der Nachrichten kann somit in einem bestimmten Zeitrahmen wiederhergestellt und korrekt abgearbeitet werden. Als Zeitbasis wird nicht die lokale Zeit, sondern die GMT/UTP Zeitzone verwendet. Somit können Laufzeit-differenzen über die Zeitgrenzen hinweg ermittelt werden.

Daten:

Die Daten selbst werden in zwei Teile aufgeteilt. Sie bestehen aus der Nachricht selbst und einer optionalen nachrichtenabhängigen Zusatzinformation (Parameter). Die Nachricht besteht aus einer Anweisung wie Open oder Up. Der Parameterteil wurde in verschiedene, festgelegte Klassen mit unterschiedlich vielen Parametern und unterschiedlichen Parametertypen eingeteilt.

8.4.6 Architektur

Bei der ersten Implementierungsarbeit zeigte sich, daß beim Empfang der Nachrichten ein reines Ausführen von Funktionen schnell an die Grenzen stößt. Denn das Verhalten konnte von außen nicht beeinflußt oder modelliert werden und lief somit immer in den gleichen Bahnen ab. Um diese Objekte flexibler agieren zu lassen, bedurfte es einer Erweiterung. Zur Abgrenzung zu den rein ausführenden Objekten führten wir für diese Art von Objekten mit erweiterten Eigenschaften den Begriff ***Objekte mit Verhalten*** ein. Dieser Name wurde auch in der Absicht gewählt, biologische Analogien zu nutzen. Der Begriff "Verhalten" impliziert komplexe eigenständige Handlungsweisen und Kommunikationsfähigkeit, im Gegensatz zu

technischen Termini wie Memberfunktionen. Grundsätzlich kann ein Objekt bezüglich des Nachrichtenaustausches vier Zustände einnehmen:

- Senden und Empfangen von Nachrichten (Standardeinstellung)
- Nur Senden
- Nur Empfangen
- Weder Senden noch Empfangen

Beim Empfang einer Nachricht werden folgende vier Schritte ausgeführt:

- Zuerst wird geprüft, ob Vorbedingungen (engl. pre-conditions) vorhanden sind. Falls dies zutrifft, wird überprüft, ob eine oder mehrere Vorbedingungen erfüllt werden.
- Im zweiten Schritt wird die Nachricht optional in einer Liste gespeichert. Dies entspricht im übertragenen Sinn sozusagen dem zeitlichen Gedächtnis des Objekts. Es stehen zwei verschiedene Modi zur Verfügung: Spielt die Häufigkeit oder zeitliche Reihenfolge eine Rolle, so wird jede Nachricht registriert (append-Mode). Im Gegensatz dazu wird im replace-Mode die letzte empfangene Nachricht gleichen Typs überschrieben. Dieser Modus ist der am meisten gebräuchliche.
- Die eigentliche Aktion/das Verhalten wird ausgeführt.
- Es wird ermittelt, ob mit diesem Verhalten noch zusätzliche Aktionen (post-action) verbunden sind. Ist dies der Fall, so werden eine oder mehrere Nachrichten gegebenenfalls noch an andere Objekte versendet. Außerdem kann das Objekt sich selbst auch Nachrichten schicken.

Aus biologischer Sichtweise besitzt jedes Objekt mit Verhalten ein Repertoire von abgeleiteten Grundverhaltensweisen, die selbstverständlich erweitert werden können. Aus diesem Grund haben alle Objekte gemeinsame Basisfunktionen, die ein Hinzufügen oder Entfernen von Verhalten zulassen, und die Fähigkeit, ihren Namen und Informationen anderen mitzuteilen. Beim Hinzufügen von Verhaltensweisen kann der Vorteil von Polymorphismus, daß dasselbe Schlüsselwort für eine ähnliche Aktion verwendet werden kann, genutzt werden. Beispielsweise gilt das Verhalten "öffnen" sowohl für die Röntgenmappe, Patientenarchivbox oder Schreibtischschublade.
Es besteht ferner die Möglichkeit, Makros zu definieren. Hierbei werden feste Beziehungen von Nachrichten und Aktionen zwischen Objekten beziehungsweise Objektgruppen festgelegt. Im übertragenen Sinne entspricht die Nutzung von Makros einem biologischen Reflex, bei

dem in festgelegten Bahnen schnell Information transportiert werden und eine vorher festgelegte Aktion/Reaktion erfolgt.

Desweiteren kann die Nachrichtenübermittlung zwischen den Objekten entweder synchron oder asynchron erfolgen. Ein Objekt mit Verhalten hat überdies die Möglichkeit, Abfragen (engl. queries) zu einem anderen Objekt zu schicken und erhält dann die dazugehörige Information zurück. Im weiteren Fall kann ein Objekt auch eine Abfrage gleichzeitig an mehrere Objekte schicken, erhält dadurch eine Liste der angeforderten Informationen und kann dann entsprechend darauf reagieren.

In Abbildung 34 ist ein einfaches Beispiel des Messageconcept dargestellt. Beim Klick auf eine spezielle Taste der Fernbedienung wird eine Nachricht an die Bildtafel geschickt, daß diese hochfahren soll. Die Bildtafel ihrerseits speichert sich diesen Vorgang mit Uhrzeit und Sender. Nicht dargestellt ist, daß die Objekte vor oder nach dem Erhalt der Nachricht weitere Informationen austauschen können. Beispielsweise kann das versendende Objekt den Zustand des Empfängers abfragen und sich gegebenenfalls die Versendung der Nachricht sparen, da der geforderte Zustand schon vorliegt. Ferner kann die Nachricht durch die neuen Informationen präzisiert und modifiziert werden. Nach Erreichen des neuen Zustands kann der Empfänger diesen an den Sender zurückübermitteln oder einfach quittieren, daß der Befehl erfolgreich angewendet wurde. Letztendlich ermöglicht dieses System des universalen Messageconcept die Freiheit, die Nachricht immer zu versenden, vorher eine Abfrage durchzuführen oder eine Rückmeldung/Quittierung zu geben. Welche Strategie zu wählen ist, entscheidet der Entwickler. Wesentlich ist, daß sich dadurch auch komplexe Kommunikationsmodelle realisieren lassen, in die mehr als zwei Objekte involviert sind.

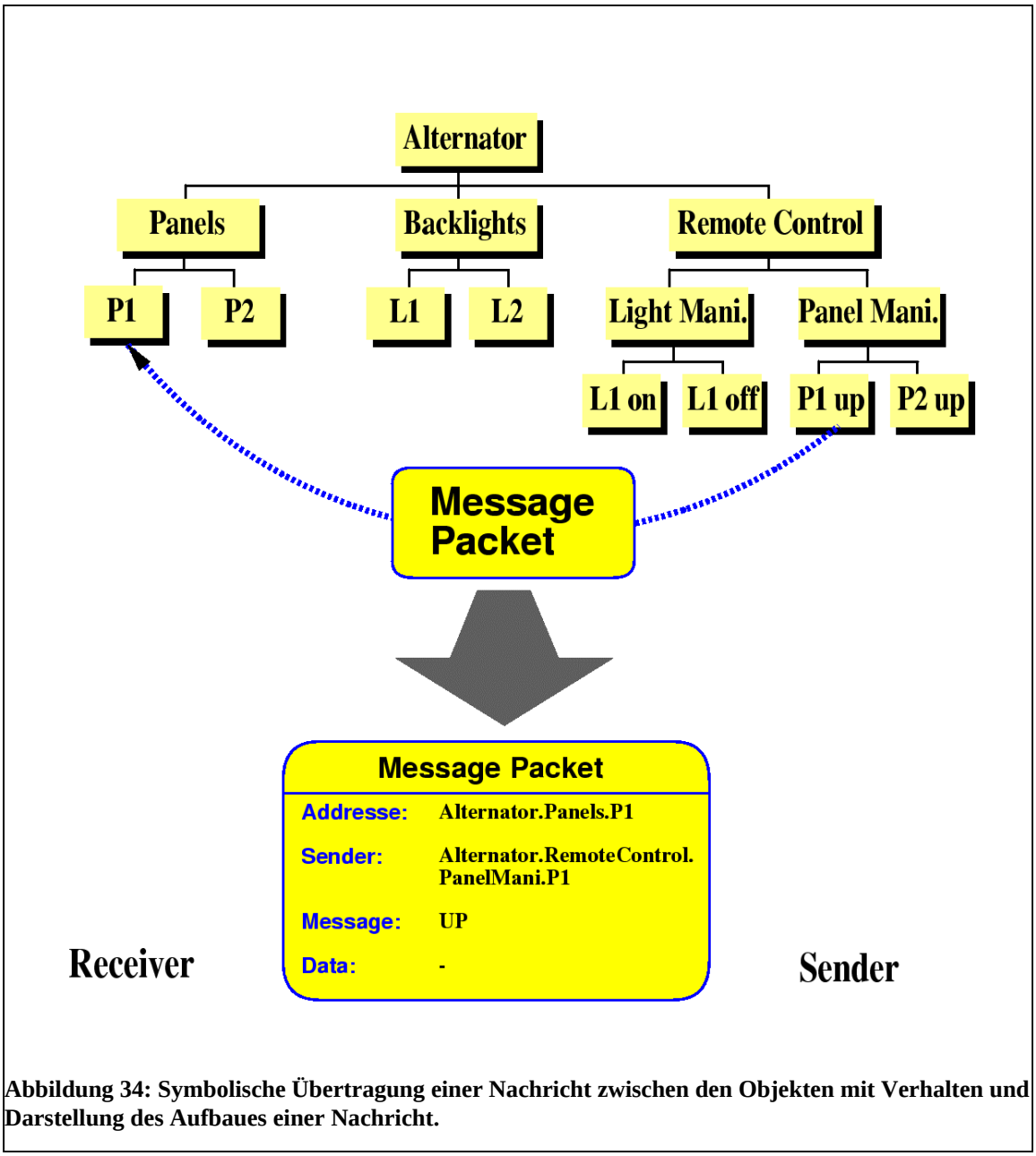


Abbildung 34: Symbolische Übertragung einer Nachricht zwischen den Objekten mit Verhalten und Darstellung des Aufbaues einer Nachricht.

8.4.7 Lokale vs. Globale Informationshaltung

Dem objektorientierten Anspruch folgend, werden konsequenterweise alle Informationen, konkreter alle Nachrichten von außen, lokal im Objekt gespeichert. Der Vorteil davon ist, daß auf diese Informationen, zum Beispiel über den Szenenstatus (Beispiel Tag/Nacht), schnell zugegriffen werden kann. Andererseits bedingt dies, daß die möglicherweise daran interessierten Objekte immer auf dem laufenden Informationsstand gehalten werden, was die Anzahl der zu versendenden Nachrichten insgesamt erhöht. Ergänzend muß erwähnt werden, daß die Objekte auch aktiv Information über Abfragen anfordern können. Das automatische Informieren ist somit nur bei wirklich wichtigen Nachrichten einzusetzen. Außerdem sollten

die Änderungshäufigkeiten nicht zu hoch sein, da sonst aus Zeitmangel der Informationsstand über die verschiedenen Objektinstanzen hinweg unterschiedlich und deshalb nicht mehr konsistent wäre. Eine andere Möglichkeit zur Beseitigung dieses Engpasses bei hoher Aktualisierungsfrequenz besteht darin, die interessanten Informationen in künstliche Objekte global abzulegen. Bei Bedarf holen sich die daran interessierten Objekte die aktuelle Information mittels einer Abfrage. Die denkbare dritte, radikalere Möglichkeit ist, auf sämtliche Registrierung von Nachrichten in Listenform im lokalen Objekt zu verzichten. Allerdings macht dieser Ansatz nur dann einen Sinn, wenn nicht auf frühere Nachrichten zugegriffen werden muß oder deren Anzahl keine Rolle spielt. Gegebenenfalls kann man auch hierfür ein (gemeinsames) Objekt für die aktuelle globale Informationshaltung einsetzen. Es kommt letztlich auf die Applikation an, ob der globalen oder lokalen Informationshaltung der Vorzug zu geben ist. Da das Messageconcept von weitgehend autonomen Objekten ausgeht, wird auf die herkömmliche Methode mittels Zeiger auf die gemeinsame Information zuzugreifen, verzichtet. Zumal diese Art des Zugriffs in verteilten Umgebungen einen erheblichen Mehraufwand erfordert.

8.4.8 Einteilung in Klassen

In der Abbildung 35 wird die Klasseneinteilung der Hauptobjekte gezeigt. Die zentrale Basisklasse heißt MessageBasic und enthält die Felder für eine einzelne Nachricht. Davon abgeleitet wurde das Element MessageNode, welches die Fähigkeit zum Senden und Empfangen von Nachrichten hat. Die Frage war, ob sich ein allgemeiner Bauplan für die Konstruktion neuer Objekte sowie deren Wiederverwendung finden läßt. Die Analyse zeigte, daß vier Elemente wesentliche Bestandteile der Objekte waren:

- Die angesprochene Kommunikationskomponente regelt den Empfang und das Senden von Nachrichten.
- Das Aussehen des Objekts wird durch eine weitere Gruppe bestimmt.
- Die (Initial-)Position des Objekts im Raum wird festgelegt.
- Optional erhielten manche Objekte noch ein viertes Element, das bestimmte Systemereignisse und Benutzereingaben (wie Mausklick) empfangen konnte, welche nicht mittels des Messageconcept übermittelt werden.

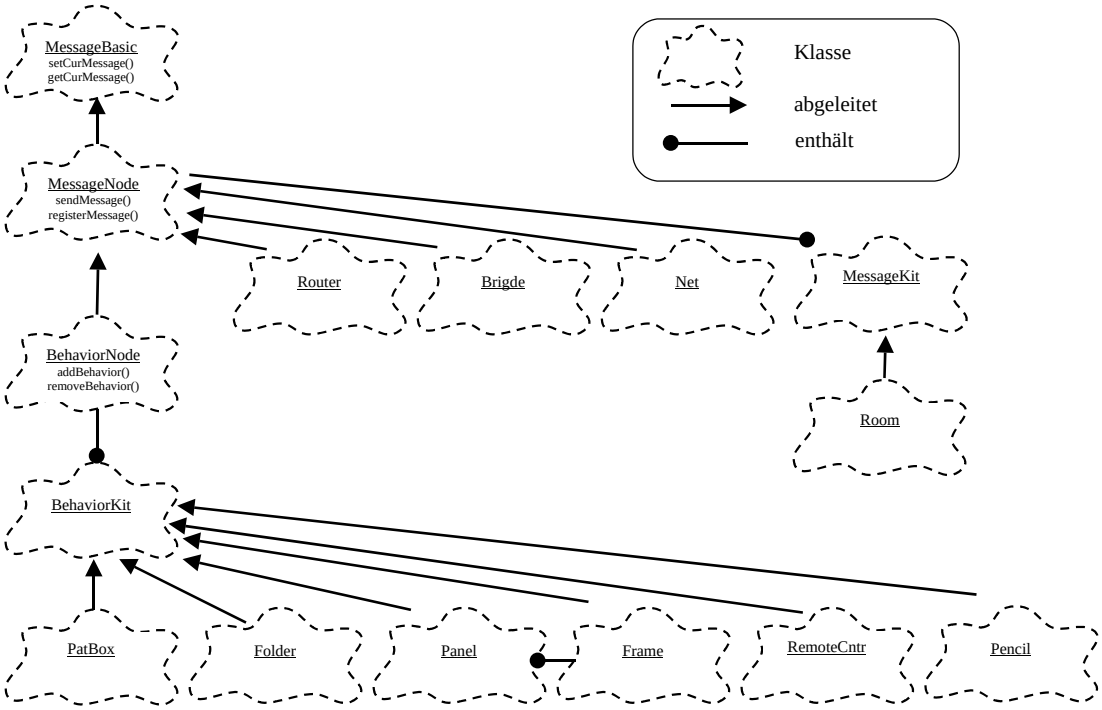


Abbildung 35: Klassenübersicht

Klassenkonzept für nachrichtenbasierte Objektkommunikation. Objekte mit Verhalten werden in einem vordefinierten Schema (MessageKit, BehaviorKit), welches mehrere unterschiedliche Teilobjekte enthalten kann, zusammengefaßt.

Kurzbeschreibung der Klassen:

MessageBasic:	Diese Klasse enthält die reine Nachricht.
MessageNode:	Hier kommt die Fähigkeit hinzu Nachrichten zu empfangen und zu senden. Empfangene Nachrichten können in Listen abgespeichert werden.
Router:	Einseitiger Nachrichtenaustausch zwischen Szenen.
Brigde:	Zweiseitiger Nachrichtenaustausch zwischen Szenen.
Net:	Zweiseiter Nachrichtenaustausch über Server/Client Struktur und unterschiedliche Protokolle (UTP,TCPIP, Multicast).
MessageKit:	Schablone für Nachrichtenaustausch. Informationen über eingehende Meldungen werden in Listenform gespeichert.
BehaviorNode:	Objekt mit Verhalten, welche in Funktionslisten abgelegt sind. Mit optionaler Vorbedingung beziehungsweise automatischer Folgenachricht oder Folgeaktion.
BehaviorKit:	Schablone „Objekt mit Verhalten“ enthält mehrere Elemente. Neues Verhalten kann vererbt oder hinzugefügt werden, hat die Fähigkeit Abfragen durchzuführen und Makrofähigkeit

Aus Übersichtsgründen wurde auf die Darstellung zahlreicher Hilfsklassen, beispielsweise zum Abspeichern der Nachricht in Listenform oder zur Auflösung der hierarchischen Namen mit und ohne Wildcards, verzichtet.

Als weiterer konsequenter Schritt kamen somit vorgefertigte gruppierte Schablonen, sogenannte Nodekits, zum Einsatz, die diese Elemente integrieren konnten. Die erste Schablone auf dieser Abstraktionsebene wurde MessageKit genannt. Diese Objekte haben zwar ein Aussehen und können Nachrichten senden oder empfangen, haben aber kein Verhalten. Dazu wurde die Klasse MessageNode weiter abgeleitet, erhielt die notwendigen

Funktionen und den Namen BehaviorNode. Analog wurde auch hier eine Schablone namens BehaviorKit kreiert. Dies ist Ausgangspunkt für alle weiteren "Objekte mit Verhalten".

Neben den erwähnten Hauptelementen in einer Schablone werden noch zusätzliche Elemente gebraucht, die jedoch von Objekt zu Objekt stark verschieden sind. Deshalb werden Objekte, abgeleitet vom BehaviorKit oder MessageKit, individuell nach ihren Erfordernissen erstellt. Der andere extreme Fall, daß alle möglichen Elemente in ein Allround-Objekt integriert werden, hat zwei entscheidende Nachteile: Die unnötigen Elemente haben einen hohen Speicherbedarf und die Systemleistung wird herabgesetzt.

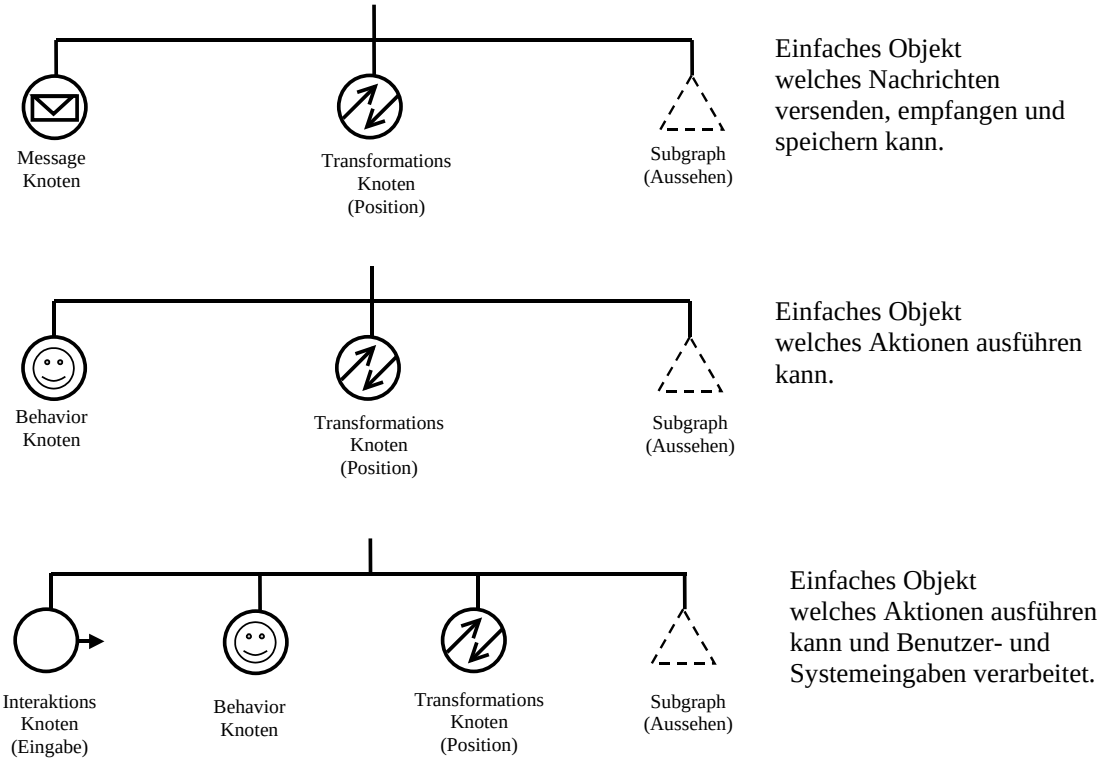


Abbildung 36: Übersicht des Zusammenspiels der neuen Elemente in den sogenannten Schablonen

8.4.9 Aufbau der wesentlichen Klassen

Ein wesentliches Element der Klassen sind Listen von Funktionen, Adressen und empfangene Nachrichten, die zum Vergleichen, Referenzieren, Bewerten und Ausführen herangezogen werden. Im folgenden wird der interne Aufbau der Klassen dargestellt. Siehe auch dazu ergänzend Abbildung 35.

MessageBasic:

Aufgabe: Das Speichern und Aufrufen einer kompletten Nachricht (Sender, Empfänger, Nachricht, optionale Daten).

```
class MessageBasic: public Node
{
    public:
    // Constructor
    MessageBasic();

    // Public Fields
    MsgItem currMsg;

    // Public Functions
    void setCurMessage(char *whichobj, char *whichmessage, char *fromobj, void* userdata);
    void getCurMessage(char **whichobj, char **whichmessage, char **fromobj, void**
userdata);

    // Destructor private though Reference count mechanism
    ~MessageBasic();
};
```

MessageNode:

Aufgabe: Das Objekt kann vier verschiedene Zustände von Senden und Empfangen einnehmen. Es kommt hier die Fähigkeit hinzu, Nachrichten zu empfangen und zu senden. Ferner wird festgelegt, ob empfangene Nachrichten abgespeichert werden und falls dies der Fall ist, ob alle oder nur die letzte aktuelle empfangene Nachricht gespeichert wird. Der Empfang wird mittels eines Sensors registriert, darauf erfolgt dann die weitere Abarbeitung zeitversetzt (asynchrone Verarbeitung). Im synchronen Falle erfolgt sie sofort.

```
class MessageNode: public MessageBasic
{
    public:
    // Constructor
    MessageNode();

    // Public Fields
    enum Messagetype {
        SEND_RECEIVE            = 0,
        SEND_NO_RECEIVE         = 1,
        NO_SEND_RECEIVE         = 2,
        NO_SEND_NO_RECEIVE      = 3,
    };
    Enum  Messagebehavior; // is from type: Messagetype

    enum Synctype {
        ASYNC        = 0,
        SYNC         = 1,
    };
    Enum  Syncbehavior; // is from type: Synctype      Important for Queries
```

```
    enum Historytype {
        SAVE_LAST    = 0,
        SAVE_ALL     = 1,
        NO_SAVE      = 2,
    };
    Enum  Historybehavior; // is from type: Historytype

    // Public Functions
    virtual void sendMessage(char * whichobj, char* whichmessage,  char *fromobj,
        void* userdata = NULL);
    virtual void registerMessage(void* data);
    virtual Bool checkMessage(char * whichobj, char* whichmessage, char *fromobj);
    static void dispatchMessage(void *data);
    void lock(void); void unlock(void); Bool isLocked(void); }; void WaitofUnlock(void);

    protected:
    FieldSensor *newMsgArrived; // Sensor to trigger if new mesage arrived
    Bool mlock;  // Locking state
    MsgList incomingMessageList; // Messages can be saved into this  Llist

    // Destructor private though Reference count mechanism
    ~MessageNode();
};
```

BehaviorNode:

Aufgabe: Objekte mit Verhalten, welche in Funktionslisten abgelegt sind. Mit optionaler Vorbedingung beziehungsweise automatischer Folgenachricht/Folgeaktion. Einzelne Funktionsaufrufe können gruppiert werden. Es wird auch für die Schablone BehaviorKit gebraucht.

```
class BehaviorNode: public MessageNode
{
    public:
    // Constructor
    BehaviorNode();

    // Public Functions
    // behavior is a single function
    void addSingleBehavior(SoFuncList *fl, void *fobj=NULL);
    void addSingleBehavior(char *whichbehavior, CallbackListCB *f, void *fobj=NULL);
    void removeSingleBehavior(char *whichbehavior, CallbackListCB *f);
    CallbackListCB* getSingleBehavior(char *whichmessage);
    void executeSingleBehavior(char *whichmessage, void *data) ;
    Bool checkSingleBehavior(char *whichmessage);
    // grouping of different functions
    void addBehaviorAction(SoMsgCtrlList *fl);
    void addBehaviorAction(char* index, char *whichobj, char *whichmessage, char *fromobj,
        void* userdata = NULL);
    void replaceBehaviorAction(char* index, char *whichobj, char *whichmessage,
        char *fromobj,
        void* userdata = NULL);
    void removeBehaviorAction(char* index, char *whichobj, char *whichmessage,
        char *fromobj);
    void executeBehaviorAction(char *whichindex);
    Bool checkBeforeCondition(char *whichindex);

    protected:
    FuncList list; // list of Functionname and Function
    MsgCtrlList CtrlMessageList; // internal list of BehaviorAction
    // functions
    CallbackList messageCB;

    // Destructor private though Reference count mechanism
    ~BehaviorNode();
};
```

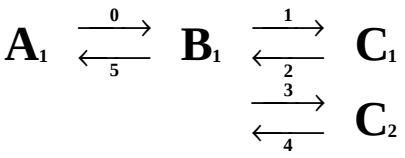
8.4.10 Spezielle Probleme und Lösungen

8.4.10.1 Synchrone/Asynchrone Nachrichtenübertragung

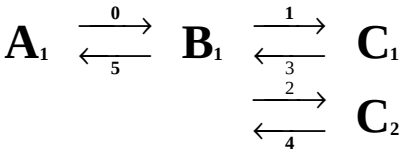
Nachrichten können synchron oder asynchron übertragen werden. Im asynchronen Modus werden die Nachrichten verschickt, ohne daß gewährleistet ist, daß sie in der zeitlichen Reihenfolge des Verschickens bei den bestimmten Objekten ankommen. Während der asynchrone Modus beim einfachen Versenden von Nachrichten der Gebräuchlichere ist, muß bei Abfragen der synchrone Modus eingesetzt werden, da hier auf die Antwort des angesprochenen Objekts gewartet wird bevor die nächste Anfrage erfolgt. Somit wird auch sichergestellt, daß ein vollständiges Ergebnis sämtlicher angefragter Objekte zum ursprünglichen Objekt zurückgemeldet wird.

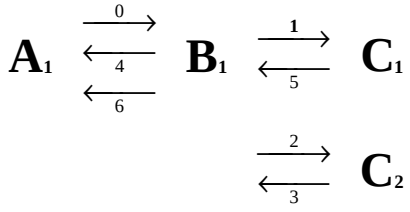
Zeitliche Reihenfolge von Abfragen:

Synchron



Asynchron





B2 : Objekttype mit Instanznummer

$\xrightarrow{3}$: Abfrage mit zeitlicher Reihenfolge

$\xleftarrow[4]{}$: Antwort mit zeitlicher Reihenfolge

8.4.10.2 Überlagerung von Präsentation und Funktion

Die Überlagerung von Präsentation und Funktion liegt, wie angesprochen, in der Natur von 3D-Objekten begründet. Im Zusammenhang mit unserem standardisierten Objekt und Verhaltensset bedeutet dies, daß alle Containerobjekte den gleichen Basisfunktionsumfang enthalten wie alle anderen Objekte auch, sowie spezielle Funktionen, die zur Ausübung der Containerfähigkeit notwendig sind. Beispielsweise haben alle Containerobjekte wie Schublade, Mappe, Schachtel oder Schrank Funktionen zum Öffnen und Schließen. Die Art, wie das Öffnen oder Schließen erfolgt, wird entscheidend vom Aussehen bestimmt. Und die internen Mechanismen, die dies verwirklichen, sind uneinheitlich. Zum einen handelt es sich um unterschiedliche Teile, die sich öffnen lassen, zum anderen erfolgt die Öffnung selbst verschiedenartig, durch Translation, Rotation oder Kombination von beidem. Zwei Lösungsmöglichkeiten sind hierzu denkbar: entweder man unterteilt die Containerobjekte weiter in Unterklassen nach Art der Öffnung, oder die Funktion des Objekts muß individuell angepaßt werden. Letzterer Ansatz wurde für die Virtual Lightbox Anwendung gewählt.

8.4.10.3 Kontextbetrachtung

Im Gegensatz zu 2D-Objekten ist das Verhalten von 3D-Objekten stärker kontextsensitiv. Das bedeutet, daß das gleiche 3D-Objekt verschiedene Verhaltensweisen, abhängig von der Umgebung, ausführen muß. Erschwerend kommt hinzu, daß es nur eingeschränkte Interaktionsmöglichkeiten mit dem Objekt beziehungsweise mit den Objekten (verglichen mit der realen Welt) gibt. Um dennoch eine größere Anzahl von Interaktionen in der künstlichen Welt durchführen zu können als nur Selektion und Verschiebung im Raum, bleibt als Lösung nur, die Interaktionen zwischen verschiedenen Objekten festzulegen. Ein einfaches Beispiel soll diesen Sachverhalt verdeutlichen:

- Wird auf den Stift geklickt und es wurde vorher ein Papierblatt selektiert, so wird in den Schreibmodus gegangen.
- Wird auf einen Stift geklickt, ohne daß vorher ein anderes Objekt selektiert wurde, kehrt der Stift auf seinen Standardplatz zurück.
- Wurde aber statt des Papiers vorher der Papierkorb ausgewählt, dann wird der Stift aus der Virtuellen Umgebung entfernt.

Aus den Kontextüberlegungen leiten sich folgende Fragen ab:

- Muß das Objekt wissen, mit welchen anderen Objekten es eine Interobjektbeziehung hat und wie diese es beeinflussen?

- Wie erfährt das Objekt, wie und wann es einen anderen Modus anwenden muß?
- Wird das Objekt automatisch von den korrespondierenden Objekten benachrichtigt oder holt es aktiv den neuesten Status von diesen (aktive/passive Benachrichtigung)?
- Was geschieht bei Mehrdeutigkeit?
  Was soll ausgeführt werden?
  Beispiel: Wenn beide, das Papier und der Papierkorb, beim Anklicken des Stiftes selektiert wurden.
- Spielt die zeitliche Reihenfolge bei der Interaktion eine Rolle?
  Soll dann eine andere Aktion ausgeführt werden?
  Beispiel: Wurde vorher ein Papierblatt und dann ein Radiergummi selektiert, so wird in den Löschmodus gegangen.
  Wurde aber zuerst der Radiergummi und danach das Papierblatt selektiert, wird der bisherige Inhalt komplett gelöscht.

Für das eigentliche Hauptproblem "Was hat der Benutzer als nächstes vor?", gibt es naturgemäß keine Lösung. Verstärkt wird dies in Virtuellen Umgebungen noch dadurch, daß es nur wenige Interaktionen wie Selektion oder Greifen gibt und sich somit keine konkreteren Vorhersagen machen lassen. Die kontextabhängige Betrachtung erweitert jedoch den Interaktionsraum. Unser Lösungsansatz des Messageconcept mit Kontextbetrachtung soll kurz die oben angesprochenen Fragen beantworten.

Eine Möglichkeit besteht darin, daß das Objekt selbst aktiv Meldung zu den korrespondierenden Objekten verschickt. Unser allgemeines Nachrichtenformat enthält alle Felder, damit mitgeteilt werden kann: Objekt x wurde zum Zeitpunkt t selektiert. Die Objekte sind somit immer auf dem neuesten Informationsstand. Grundsätzlich kann diese Nachricht an alle Objekte gesendet werden (Statusnachricht), aber da nicht alle Objekte an diesen Informationen interessiert sind und der Nachrichtenverkehr in Grenzen gehalten werden soll, ist es sinnvoll, daß sich die Objekte bei dem korrespondierenden Objekt anmelden und mitteilen, daß sie an den entsprechenden Meldungen interessiert sind.

Ein anderer Ansatz besteht darin, daß ein Zielobjekt die korrespondierenden Objekte nach einem bestimmten Zustand abfragt. Hier kommt die vorgestellte Query-Komponente zum Einsatz, die dem anfragenden Objekt Auskunft über den Status und die Zeit der Selektion gibt.

Die oben genannten Probleme der Mehrdeutigkeit und Reihenfolge von Interaktionen sind somit zwar unterscheidbar, aber letztlich liegt es an der Applikation, ob es auch einen Sinn

macht. Dies wurde bereits im Messageconcept berücksichtigt und die Ausführung von ganz verschiedenen Verhaltensweisen, die kontextabhängig sind (ausgelöst durch die gleiche Interaktion), können realisiert werden.

8.4.10.4 Wiederverwendung des Codes mittels Makros

Makros bestehen aus einer Menge von starr festgelegten Aktionen zu vorher bestimmten und bekannten Objektadressen oder Objektgruppen mit den dazugehörigen Nachrichten. Neben der Gruppierung von häufig benutzten Verhaltensweisen bieten Makros den Vorteil, daß sie nur einmal definiert werden müssen und auf verschiedenen Instanzen, unabhängig vom tatsächlichen Namen des sendenden Objekts, ausgeführt werden, da dieser zur Echtzeit ermittelt und mitgesendet wird. Ferner können diese Makros von Objekten mit anderem Aussehen wiederverwendet werden.

Beispiel:

Definition:

```
// Abfrage eines Drittobjekts welches Panel gerade angezeigt wird
ownAction.appendMsgToAutoRunList("?PanelShow","alternator*.panel*", "?ShowPanel",
this->getName())

// Beim Drücken eines Fernbedienungsknopfs soll ein Panel am Alternator hochgefahren
werden
ownAction.appendMsgToAutoRunList("PanelUp","alternator*.panel*","UpPanel", this-
>getName())

spätere Ausführung
this->executeAction("?PanelShow");
this->executeAction("PanelUp");
```

8.4.10.5 Locking

Innerhalb von verteilten virtuellen Applikationen können ein, zwei oder mehrere Benutzer oder Objekte auf ein anderes Objekt zugreifen. Dieser Zugriff kann in folgende Kategorien aufgeschlüsselt werden:

- Gemeinsamer kooperativer Zugriff, wie das Wegtragen eines Tisches.
- Gemeinsamer konkurrierender Zugriff, wie an einem Seil ziehen.
- Exklusiver Zugriff auf ein Objekt, wie ein Werkzeug.

Zur Lösung des letzteren Falls werden einfache abgewandelte Lockingmechanismen aus der Datenbankwelt verwendet. Hierbei wird durch eine Variable der augenblickliche Zustand des Objekts dargestellt, wobei es zu Zugriffskonflikten kommen kann, wenn zum Beispiel gleichzeitig Objekt A das Objekt B und Objekt B das Objekt A ändern möchte. In SQL-

Datenbanken gibt es einen zentralen Zugriffsmanager, der nach einer variablen Zeit einen Rollback auslöst. Bei anderen traditionellen Datenbanken, wie dBASE, muß das Programm selbst durch geeignete Mechanismen Sorge für die Auflösung tragen, da es keine übergeordnete Instanz gibt. Analog kann bei zentralen Netzwerkmodellen auf Server/Client-Basis ein Manager die Zugriffe steuern, während bei dezentralen Netzwerkmodellen auf Peer-to-Peer Basis die Auflösung erheblich aufwendiger ist. Die automatische Auflösung von Konflikten ist vor allem in Multi-User-Applikationen wichtig.

Es gibt verschiedene Arten des Lockings, hier die wichtigsten:

- Exklusives Locking: Ein Client hat den alleinigen Zugriff auf die Datenbank, die anderen können weder Lesen, Schreiben, Ändern, Löschen oder Hinzufügen.
- Shared Locking: Nur das Lesen der Werte von einem oder mehreren Clients ist möglich, nicht aber das Ändern oder Löschen von Werten.
- Soft Locking: Die vorherige Sperrung wird aufgehoben oder durch eine Aktion mit höherer Priorität überschrieben.
- Opportunistisches Locking (Oplock): Hierbei gibt es verschiedene Arten, die eine Kombination der obigen Lockingmechanismen darstellen. Fordert der erste Client einen Oplock an und kein anderer Client hat Zugriff, so erhält dieser vom Server einen exklusiven Oplock. Wenn ein zweiter Client auch auf die Daten zugreifen möchte, die bereits der erste Client geöffnet hat, so fragt der Server den ersten Client an, ob er die Lockart auf Shared Locking (in diesem Zusammenhang auch als Level II Oplock bezeichnet) ändern darf. Ist dies der Fall, werden die lokal zwischengespeicherten Daten in die Datenbank geschrieben, der Client informiert den Server, daß er nun in den Shared Locking Modus wechselt. Der Server wiederum informiert den zweiten Client, daß dieser ebenfalls in den Shared Locking Modus wechseln darf.

Neben den Lockingtypen können verschiedene Ebenen des Locking einer klassischen Datenbank auf eine grafische Szenendatenbank übertragen werden:

- Sperrung der kompletten Szene
- Sperrung eines einzelnen Objektes oder einer Objektgruppe

Angemerkt sei, daß aus Komplexitätsgründen beim Locking oft das gesamte Objekt gesperrt wird, obwohl unabhängige Teiländerungen vorgenommen werden könnten.

Auch das Messageconcept nutzt einfaches Locking für die Zugriffssteuerung. Dadurch wird die Annahmefähigkeit von Nachrichten geregelt. Wird die Lockingvariable gesetzt, so bedeutet dies, daß das Objekt mit dem Ausführen einer Funktion/einem Verhalten beschäftigt ist und somit keine andere Nachricht, beispielsweise von einer zweiten Instanz, über das Netzwerk annehmen kann. Allerdings muß noch ein Spezialfall, der bei Datenbanken nicht vorkommt, gesondert behandelt werden. Hierbei sendet ein Objekt sich selbst eine Nachricht in der Form: $\mathbf{A}_1 \xrightarrow{0} \mathbf{A}_1$. Damit keine Endloswarteschleife bis zur Freigabe des gesperrten Objekts entsteht, müssen Nachrichten mit der gleichen Kennung von Sendernamen und Objektnamen durchgelassen werden. Ein Nachteil des Locking ist, daß das gesamte Objekt gesperrt ist, obwohl unabhängige Teiländerungen gemacht werden könnten.

Locking ist auch das Mittel, welches den Zugriff in Multi-User-Applikationen steuert [BE96a]. Ferner eignet sich das Lockingverfahren wie oben aufgeführt auch, wenn die Objekte selbst in eigenständigen Prozessen, wie Threads in einer Applikation, implementiert wären. Dies läßt aber die gegenwärtige Szenengraph-Implementierung von Silicon Graphics nicht zu.

8.4.10.6 Besondere Ereignisse und Aktionen

Bei der Implementierung des Messageconcept zeigte sich, daß drei Aspekte besonders berücksichtigt werden müssen:

- Die Objekte müssen ein a-priori Wissen zum Zeitpunkt ihrer Erzeugung haben (wie die Position im Raum oder ihren Namen, welcher temporär sein kann).
- Die Änderung der Objekthierarchie zur Laufzeit muß erkannt werden (beispielsweise durch eine Drag&Drop-Aktion).
- Spezielle Management-Objekte sind für die Erzeugung, das Kopieren, Verschieben oder Löschen von Objekten zuständig. Ferner sind sie zuständig, um Mehrdeutigkeiten wie die Namen und Gruppenzugehörigkeit auszuschließen.

8.4.10.7 Verteilung von Informationen

Um Informationen zwischen zwei Applikationen direkt austauschen zu können, gibt es zwei spezielle Objekte, die wir Router und Bridge nannten. Ersteres erlaubt die Weitersendung von Nachrichten in nur eine Richtung, während die Bridge einen zweiseitigen Austausch zuläßt. Überdies gibt es einen dritten Typ, der es erlaubt, Nachrichten über ein Netzwerk

auszutauschen. Beim Netzwerkaustausch gilt die Einschränkung, daß es keinen Sinn macht, die Adresse eines Objekts oder von Daten zu übermitteln, sondern gegebenenfalls das Objekt selbst. Alle drei Objekte haben die Eigenschaft, daß sie Nachrichten filtern können, das heißt, sie leiten nur Nachrichten von einem bestimmten Sender oder einer Sendergruppe oder für einen bestimmten Empfänger oder eine Empfängergruppe weiter. Dies wird auch als Source- oder Destinationrouting bezeichnet. Neben der Entlastung des Netzwerkverkehrs führt dies auch zur Entlastung der Applikation, da diese nur die relevanten Nachrichten erhält, die weiterverarbeitet werden müssen. Andere Netzwerkfunktionalität, wie Kollisionserkennung, verteiltes Verhalten und Unterstützung von Avataren, ist erst im speziellen Messageconcept (Dynamic World) enthalten.

In großen Virtuellen Umgebungen mit verteiltem Objektverhalten und verteilten Benutzerinteraktionen muß durch zusätzliche Mechanismen die Konsistenz der Information gewährleistet werden. Dies kann durch weitere eigenständige Programme erreicht werden. Folgende unterschiedliche Aufgaben können sie zusätzlich unterstützen, kontrollieren oder managen: Die An- und Abmeldung von Benutzern, Gewährleistung des Synchronismus von Ereignissen und Nachrichten auch über langsame Verbindungen, Unterstützung von Kollisionserkennung und Kooperationen sowie im Fehlerfalle die Einleitung einer Wiederherstellung von Nachrichtenabläufen oder der Szene für einen kleinen Ausschnitt der Gesamtwelt (Partitionierung ohne/mit Replikation), für den sie verantwortlich sind [BF97].

8.5 Anpassung des Messageconcept für VRML 2.0 (Dynamic World)

In Zusammenarbeit mit GMD und ZGDV entstand die Ausarbeitung für die Beschreibungssprache der nächsten VRML-Generation [BE96a, BE96b]. Teile, wie die Netzwerkkomponenten, wurden am WSI/GRIS [SD97] und die Interaktionskomponenten am GMD [BW98] in modifizierter Form implementiert. Die Basis für die Kommunikation zwischen den Objekten basiert auf dem Messageconcept. In der Literatur wurde zur Unterscheidung zwischen einem "Objekt mit Verhalten" und diesem auf VRML 2.0 basierendem, der Begriff "Kunstobjekt" eingeführt. Wie schon erwähnt, ist VRML 2.0 nur eine Dateibeschreibung, die geometrische Daten des Objektes, sein Verhalten und ihre Beziehungen untereinander enthält. Es gibt somit kein eigenständiges Programm für die Objekte selbst. Um dennoch eine hohe

Variabilität der unterschiedlichen Objektanforderungen zu gewährleisten, wird ein Objekt aus mehreren, frei wählbaren Subkomponenten im Baukastenprinzip aufgebaut.

Die Zerlegung einer Interaktion/eines Verhaltens in Subkomponenten und Multi-User-Objekte, wie die Avatare, beruhen auf den Arbeiten von Broll [BW98] und werden hier nicht weiter diskutiert. In den nächsten Abschnitten sollen nur die relevanten Teile, die durch diese Arbeit entstanden sind, wie spezielle Komponenten und das abgewandelte Messageconcept, vorgestellt werden [BE96a].

8.5.1 Nachrichtenaufbau

Die wesentliche Änderung zum allgemeinen Messageconcept besteht in der Festlegung der mitgelieferten Daten, das heißt, es gibt verschiedene vordefinierte Nachrichtentypen. Wir unterscheiden folgende sechs Arten:

- Knotennachrichten
- Feldnachrichten
- Abfragenachrichten
- Systemnachrichten
- Szenengraphnachrichten
- Benutzerdefinierte Nachrichten

Der Sender und Empfänger sowie die Zeitmarke sind weiterhin in jeder Nachricht enthalten. Der Sendername und die Zeitmarke werden automatisch beim Versand gesetzt und können auch nicht geändert werden. Der Empfänger nutzt diese Informationen zur Auswertung oder Rücksendung, um eine andere Aktion/ein anderes Verhalten abhängig vom Sender zu tätigen. Im Unterschied zum allgemeinen Messageconcept gibt es einen Standardempfänger einer Nachricht. Dieser ist, sofern nichts anderes angegeben wird oder durch bestimmte Objekte beim Versand aktualisiert wurde, das lokale Objekt oder der Gruppenknoten selbst.

Die allgemeine Syntax lautet:

```
messageName {
  <messageFields   New Value>
  ...
  sender          # SFAddress (Sender der Nachricht - nur lesbar)
  recipients [.] # MFAddress (Ein oder mehrere Nachrichtenempfänger)
  timeStamp       # SFTime (Zeit wann die Nachricht versendet wurde)
}
```

Man beachte, daß diese Felder aus dem universellen Messageconcept stammen.

Hinweis: Die Bedeutung der Feldnamen wie SFAddress wird im Anhang erläutert.

8.5.1.1 Knotennachrichten

Die Nachrichten für Knotenobjekte sind mit identischen Feldern, wie die dazugehörigen Knotenobjekte selbst, aufgebaut.

Beispiel:

```
messageTransform {
    translation             # SFVec3f
    rotation
       # SFRotation
    scaleFactor             # SFVec3f
    scaleOrientation        # SFRotation
    center                  # SFVec3f
}

messageCube {
    width                   # SFFloat
    height                  # SFFloat
    depth                   # SFFloat
}
```

8.5.1.2 Feldnachrichten

Feldnachrichten übermitteln einen einzelnen Wert oder mehrere Werte desselben Typs. Der Hauptnutzen von Feldnachrichten besteht darin, daß die Nachrichtengröße gegenüber den Knotennachrichten sehr viel kleiner sind. Deshalb eignen sie sich für einen Transfer über Netz oder unter Echtzeitbedingungen.

Die allgemeine Syntax lautet:

```
messageSFfield {
    value defaultValue        # field of type SFfield
}

messageMFfield {
    values [ defaultValue1,   # field of type MFfield
             defaultValue2,
             ...           ]
}
```

Beispiele hierfür wären:

```
messageSFFloat {
    value 3.141592
}

messageMFString {
    values [ "Alpha", "Beta", "Gamma"]
}
```

8.5.1.3 Abfragenachrichten

Abfragenachrichten haben den gleichen syntaktischen Aufbau wie die Knoten- und Feldnachrichten. Eine Abfragenachricht führt zum automatischen Rücksenden der angeforderten Werte zum Sender. Wird eine Anfrage an mehrere Empfänger verschickt, so sendet jeder von ihnen eine Antwort an den Sender zurück. Im Unterschied zum Messageconcept wird nur der asynchrone Modus unterstützt.

```
message?SFFloat {
    value                   #SFFloat
}
message?Transform {
    translation             # SFVec3f
    rotation                # SFRotation
    scaleFactor             # SFVec3f
    scaleOrientation        # SFRotation
    center                  # SFVec3f
}
```

8.5.1.4 Systemnachrichten

Systemnachrichten werden vom Browser/Viewer zum Szenengraph geschickt. Dies können Maus-, Tastatur- oder 3D-Eingaben sein.

```
message3DInput {
    translation 0.0 0.0 0.0          # SFVec3f
    orientation 0.0 0.0 1.0 0.0      # SFRotation
}

messageMouse {
    button LEFT             # SFBitMask [ NONE, LEFT, MIDDLE, RIGHT ]
    action PRESS            # SFEnum [ NONE, PRESS, RELEASE, ACTION ]
    multiple  1             # SFInt
    posX                    # SFInt
    posY                    # SFInt
    modifier NONE           # SFBitMask [ NONE, SHIFT, CTRL, ALT, META]
    coordinates 0.0 0.0 0.0 # SFVec3f
}

messageKey {
    posX                    # SFInt
    posY                    # SFInt
    modifier NONE           # SFBitMask [NONE, SHIFT, CTRL, ALT, META ]
    key '\0'                # SFChar
    coordinates 0.0 0.0 0.0 # SFVec3f
}
```

8.5.1.5 Szenengraphnachrichten

Mit Szenengraphnachrichten ist es möglich, einzelne Knoten oder Szenengruppen hinzuzufügen, zu verschieben oder zu löschen. Diese Aktionen kann der angesprochene Knoten aber nicht selbst ausführen, das bewerkstelligt der Szenenmanager. Es besteht die Möglichkeit, einen neuen Namen des Knotens mitzugeben, welcher dann wiederum in den

kompletten hierarchischen Namen eingeht. Empfänger ist ein Gruppenknoten oder ein Kunstobjekt.

```
messageAddNode {
    node              # SFNode   (Neue Elemente/Knoten)
    label             # SFString (Neuer Name)
}
```

Beispiel für das Hinzufügen eines Objekts:

```
messageAddNode {
    node Separator {
        Transformation { translation 0.0 5.0 0.0 }
        Sphere { radius 2.0 }
    }
    label "lamp"
    recipients myWorld.mainHall
}
```

```
messageRemoveNode {

}
```

Diese Nachricht führt zum Löschen des entsprechenden Knotens. Ist der Empfänger ein Objekt mit Verhalten oder ein Gruppenknoten, so werden alle abhängigen Kindesknoten mitentfernt.

```
messageMoveNode {
    destination       # SFAddress
    label             # SFString
}
```

Durch das Versenden dieser Nachricht kommt es zum Umpositionieren von Elementen. Die neue Position für die Elemente wird im Feld *destination* spezifiziert. Anzumerken wäre noch, daß die alte ursprüngliche Position im Feld *recipient* der Nachricht spezifiziert wurde. Handelt es sich beim Empfänger um ein Objekt mit Verhalten oder Gruppenknoten, so werden die abhängigen Knoten, beziehungsweise Teilbäume des Szenengraphen, mitumgesetzt. Wird nur ein Name für das Feld *label* angegeben, so führt dies zur Umbenennung des Objekts.

```
messageCopyNode {
    destinations      # MFAddress
    label             # MFString
}
```

Beispiel für das Kopieren:

```
CopyNode {
    destination yourWorld.beach.Table
    recipients myWorld.mainHall.Table.Material
```

}

Diese Nachricht ermöglicht das Duplizieren einzelner Elemente oder Teile des Szenengraphs. Da im Feld *destination* mehrere Adressen angegeben werden können, können durch das Verschicken einer Nachricht mehrere Instanzen auf einmal erzeugt werden. Analog zu den vorherigen Beispielen kann ein neuer Name vergeben werden.

```
messageLinkNode {
    destinations     # MFAddress
    label            # MFString
}
```

Im Gegensatz zur vorigen Nachricht werden keine neuen Knoten oder Szenengraphteile erstellt, sondern nur eine gemeinsame, geteilte Instanz erzeugt.

Begrifflich ist bei den Szenennachrichten zu beachten, daß der Empfänger unterschiedlich ausgelegt wird. Dies liegt in der Besonderheit begründet, daß mehr als die zwei Beteiligten (Sender und Empfänger) in eine Aktion involviert sind. Einerseits gibt es hier die neue Position (beziehungsweise Positionen) *destinations,* an der die Aktion stattfinden soll*,* auf der anderen Seite die Objekte, die kopiert oder umgesetzt werden sollen. Letztere wurden aus Konsistenzgründen *recipient* genannt und nicht in Analogie zu *destination* als *source* bezeichnet.

8.5.1.6 Benutzerdefinierte Nachrichten

Die bisherigen Nachrichten erlauben keine anpaßbaren Nachrichteninhalte. Durch die Verwendung von benutzerdefinierten Nachrichten wird dieses Manko behoben. Durch das Schlüsselwort MESSAGE wird eine neue Deklaration eingeleitet. Die einzelnen Felder können, müssen aber keine Vorgabewerte enthalten.

MESSAGE messageName { <fields> }

```
MESSAGE messageAnimation {
  SFVec3f fromposition 0.0 0.0 0.0
  SFVec3f toposition   0.0 0.0 10.0
  SFLong  step         10
}
```

Die internen Felder für Sender und Empfänger sowie für den Versendezeitpunkt werden automatisch an jede benutzerspezifische Nachricht angefügt und brauchen nicht extra deklariert zu werden.

8.5.2 Objektnamen

Die Namensgebung entspricht weitgehend der des Messageconcept. Die hierarchischen Namen werden durch Punkte getrennt. Der Einsatz von "?" und "*" als Wildcards ist ebenso möglich. Zusätzlich wird die Unterstützung von relativen Adressen wie "." für das aktuelle Objekt oder ".." für das Elternobjekt integriert.

Erlaubte Zusammensetzungen:

```
*                   Alle Objekte in der Szene
?                   Das Objekt auf höchster Ebene der Szene
*myObject           Alle Objekte die als Namensteil "myObject" enthalten
*myObject*          Wie oben jedoch mit allen abhängigen Kindsobjekten
*myObject.*         Nur das Kindobjekt von "myObject"
*myObject.?         Alle direkten Kindsobjekte von "myObject"
.                   Gegenwärtiges Objekt
..                  Elternobjekt
..*                 Elternobjekt mit allen ihren Kindern
*branch..           Elternobjekt vom Objekt "branch"
```

Nicht erlaubte Konstrukte sind:

```
*..
**
*Building**Chair
```

Es kann auch auf einen bestimmten Knoten oder ein bestimmtes Feld zugegriffen werden. Für die Lesbarkeit ist weiteres Klammern erlaubt:

```
KnotenAdresse: Objektadresse.Knotenname
               (Objektadresse).Knotenname

Feldadresse:   Objektadresse.Knotenname.Feldname
               (Objektadresse).Knotenname.Feldname
               Knotenadresse.Feldname
               (Knotenadresse).Feldname
               Objektadresse.Feldname
               (Objektadresse).Feldname
```

8.5.3 Neue Objekttypen

Der Knoten Connector entspricht dem Wesen nach dem vorgestellten Router und den Bridge-Objekten des allgemeinen Messageconcept. Dieser führt ebenfalls zur einer Erweiterung des Empfängerkreises einer Nachricht. Im Feld *messageMask* wird der Nachrichtentyp spezifiziert, der behandelt werden soll. In den Feldern *sourceRoutes* oder *destinationRoutes* wird der (oder die) ursprüngliche(n) Sender oder Empfänger eingetragen, deren Meldungen weitergeleitet werden. Somit kann eine weitere Filterung stattfinden. Zusätzlich können weitere neue Empfänger im Feld *destinations* eingetragen werden.

```
Connector {
 messageMask          # MFInput
 sourceRoutes *       # MFAddress
```

```
 destinationRoutes *  # MFAddress
 destinations .       # MFAddress
}
```

Der Knoten Interface hingegen bietet eine Verbindung zwischen einem Szenengraphen und einer externen Applikation. Diese Applikation kann sich auf dem lokalen Rechner oder auf einem entfernten Server befinden. Je nach Eintrag des Felds *direction,* kann dieser Knoten für Eingaben, Ausgaben oder bidirektionalen Austausch eingesetzt werden. Eingehende Nachrichten (außerhalb des Szenengraphs) ohne weitere Spezifikation des Empfängers werden an die Adressaten, die im *forward* Feld aufgeführt sind, weitergeleitet. Für ausgehende Nachrichten wurde das Feld *service* reserviert, um noch Zusatzinformation, wie den Name der externen Applikation oder sonstiges, mitzuschicken. Ein externer Server wird im Feld *servername* festgelegt, wobei ein Port mitangegeben werden kann. (janosch.gris.uni-tuebingen.de:1234 oder 134.2.176.91). Die Verbindung erfolgt über das Netzwerkprotokoll (TCP/IP).

```
Interface {
 direction IN         # SFBitMask [IN, OUT, INOUT]
 forward              # MFAddress
 service ""           # SFString
 servername ""        # SFString
}
```

8.5.4 Komplexes Objektverhalten

Ein komplexeres Objekt besteht aus mehreren Verhaltensweisen, die sich wiederum aus mehreren Komponenten zusammensetzen (siehe auch Kapitel 8.5).

Der allgemeine Aufbau eines Verhaltens:

```
Behavior {
    <fields>
    <triggers>       # different components
    <engines>
    <activators>
    <deactivators>
    <sensors>
    <queries>
    <actions>
    <scripts>
}
```

Allgemeines Beispiel:

```
Behavior {
  Trigger {
    inputs []        # MFInput (no message specified by default)
```

```
    condition        # SFCondition (no event - no condition)
    active TRUE      # SFBool
  }
  Action {
    condition        # SFCondition (no condition - send always)
    outputs []       # MFOutput (send message)
  }
}
```

Die Verhaltenselemente eines Objektes können also in die drei klassischen Bereiche von Eingabe, Verarbeitung und Ausgabe zerlegt werden. Das folgende einfache Beispiel zeigt, wie die Komponenten modelliert werden, um bei einem doppelten rechten Mausklick das Würfelobjekt blau einzufärben.

```
DEF myCube {                                  # objectdefinition
 Behavior makeBlue{                           # behavior
    MouseTrigger {
      input messageMouse mouse                # receiving mouse messages
      condition mouse.button == RIGHT &&      # right double click
                mouse.action == DOUBLE
    }
    Action {
       outputs messageMaterial material {     # sending material blue
         diffuseColor 0.0 0.0 1.0             # to local object by default
       }                                      # because no recipients
    }                                         # specified
  }                                           # end of behavior

  Material {                                  # by defaut grey
  }
  Cube {                                      # shape
  }
}
```

Auch ein Verhalten mit Statusabfragen von dritten Objekten ist möglich. Die Abfragekomponente hat folgende Struktur: Eine optionale Vorbedingung kann im Feld *condition* angegeben werden. Wird diese nicht erfüllt, wird auch keine Abfrage gestartet. Bei *inputs* wird eine Abfragenachricht mit den zugehörigen Feldern spezifiziert. Nach dem Erhalten der Antwort werden die korrespondierenden Werte übertragen und stehen für die weitere Verarbeitung zur Verfügung.

```
Query {
     condition          # SFCondition precondition
     inputs []          # send querymessage
                        # and wait of answer
}


DEF LightSwitchBehavior {
    SFAddress light MyWorld.House7.Light
    MouseTrigger { }
    Query {
        inputs message?SFBool isOn {       # is light on
               recipients (light).on       # field of Light
        }
    }
```

```
    Action {
        condition (isOn.value == TRUE)      # check answer
        outputs messageSFBool switchOff {   # do something
                 value FALSE
                 recipients (light)
        }
    }

}
```

Wie beschrieben, kann ein "Objekt mit Verhalten" des allgemeinen Messageconcept Nachrichten empfangen, filtern, auswerten und neue Nachrichten versenden. Alle diese Fähigkeiten sind auf diese Objekte übertragen und weiterentwickelt worden. Die empfangenen Nachrichten können gespeichert und ausgewertet werden. Dies wird durch die Benutzung der Felder SFRegister/MFRegister möglich (näheres siehe Anhang 10.3). Auch die Kernidee von Abfragen der Zustände dritter Objekte kann, wie gezeigt, durch die abgewandelte Query-Objekt initiiert werden.

8.5.5 Vergleich zum VRML 2.0/97 Ansatz

Beim verabschiedeten VRML 2.0/VRML97 Standard, der weitgehend aus dem konkurrierenden Ansatz Moving World hervorging, gibt es folgendes Schlüsselkonzept für die Kommunikation.

Drei Ereignisklassen:

eventIn	Nur Empfang
eventOut	Nur Senden
exposedField	Empfangen und Senden

Ereignistypen:

Senden:	*_changed	(position_changed, color_changed)
Empfangen:	set_*	(set_on, set_color, set_transform)

Routes:

Durch sogenannte Routes werden die Verbindungen zwischen den Feldern von Knoten definiert. Die allgemeine Beschreibung lautet:

```
ROUTE <name>.<feld/eventName> TO <name>.<feld/eventName>
```

Beispiel:

Vorgang: Durch Mausklick geht das Licht an.

```
DEF CLICK TouchSensor {enable TRUE}
DEF LIGHT DirectionalLight {on FALSE}
ROUTE CLICK.enabled_changed TO LIGHT.set_on
```

Beim VRML 2.0/97 Ansatz handelt es sich nicht um ein echtes Ereignismodell. Die Nachrichten "wandern" nicht entlang des Szenengraphs, vielmehr handelt es sich um miteinander geteilte Datenfelder (engl. "shared field"). Es gibt somit auch keinerlei Informationen über Sender oder Empfänger. Neben dem Vorteil der einfachen Implementierung mittels Zeiger gibt es folgende Nachteile gegenüber unserem Modell:

- Nur statische, keine dynamischen Verbindungen. Keine Unterstützung von neu generierten Objekten zur Laufzeit.
- Kein Ansprechen von Gruppen von Objekten, da keine hierarchische Namensgebung.
- Keine Rückmeldung zum Sender möglich (Sender-Information fehlt).
- Keine Weiterleitung von Ereignissen/Nachrichten über Dritte (Empfänger-Information fehlt).
- Kein Abfragen von Attributen der Objekte möglich.
- Keine Verteilung von Ereignissen/Nachrichten über das Netzwerk oder mehrere Applikationen möglich.
- Ereignisse mit mehreren Parametern werden nicht unterstützt.
- Kein Sicherheitsmechanismus vorhanden, der sicher stellt, daß das Ereignis auch ankommt. Mehrere Objekte teilen sich ein Feld und beim Eintreten von gleichzeitigen Änderungen gehen manche verloren oder es kommt zu unvorhersagbaren Überlagerungen.
- Erstellte neue Objekte oder gelöschte Objekte müssen speziell behandelt werden (kein automatisches Empfangen, kein einfaches Löschen).
- Keinerlei Netzwerksupport. Durch eine komplizierte und aufwendige Schnittstelle, genannt Living-Worlds, soll nachträglich Multi-User-Fähigkeit verliehen werden. Der Kooperationsaspekt fehlt völlig.

8.6 Erweiterungen/Diversifikation

8.6.1 Hilfesystem

Wie soll Hilfe in einer virtuellen Welt dargestellt werden und wie weit soll diese Hilfe gehen? Kann dabei die dritte Dimension sinnvoll genutzt werden? Ein möglicher Ansatz ist die Personifizierung des Hilfesystems durch eine künstliche humanoide Gestalt [EF95] (siehe Abbildung 37). Durch verschiedene Gesichtsausdrücke wird das Vertrauen des Systems in die gerade getätigte Benutzeraktion dargestellt [EM97].
Dieses Verfahren wird als anthropomorphisches Hilfe-Feedback bezeichnet. Verschiedene andere Objekte können dem Benutzer zusätzlich helfen.

Beispiele:

Der Fußboden dient als ortsunabhängiges Hilfeobjekt.

Ein eingeschaltetes rot/grünes Spotlight lenkt die Benutzeraufmerksamkeit auf sich, um anzuzeigen, daß neue Hilfevorschläge vorliegen oder eine Aktion nicht ausgeführt werden kann.

Zur Verwirklichung solch eines Hilfesystems sind drei unterschiedliche Objektklassen notwendig:

- Bereichsspezifische (Szenen-) Objekte
- Agenten
- Autonome Hilfeobjekte

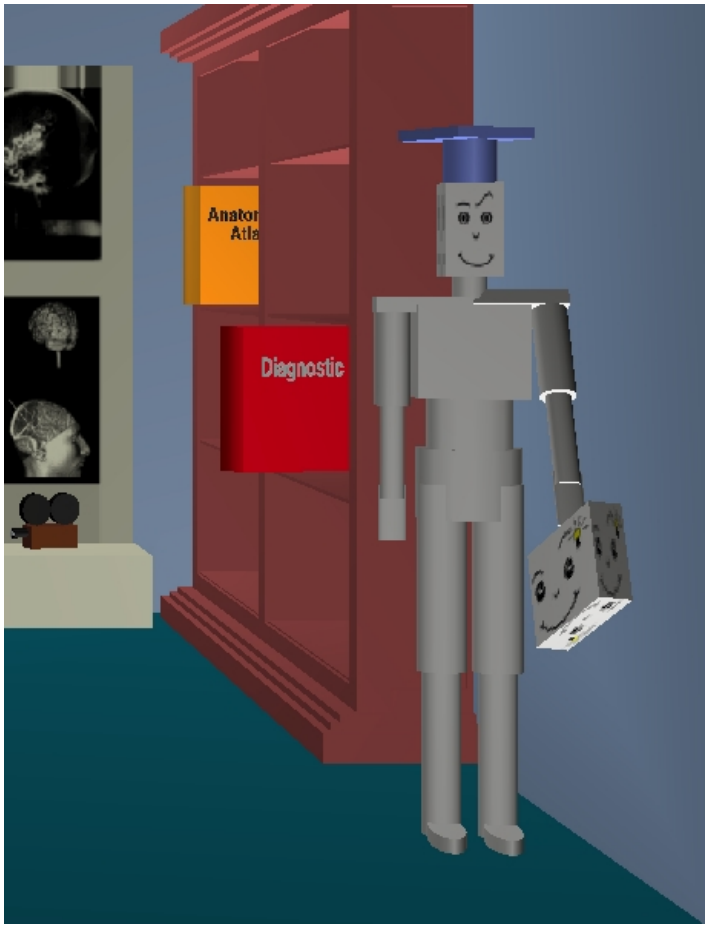


Abbildung 37: Künstlicher adaptiver Agent in einer Virtuellen Umgebung

Dem Agenten fällt dabei eine wesentliche Rolle zu. Er analysiert sämtliche Nachrichten und leitet die entsprechenden Aktionen ein. Es gibt verschiedene Arten von Hilfestellungen, die

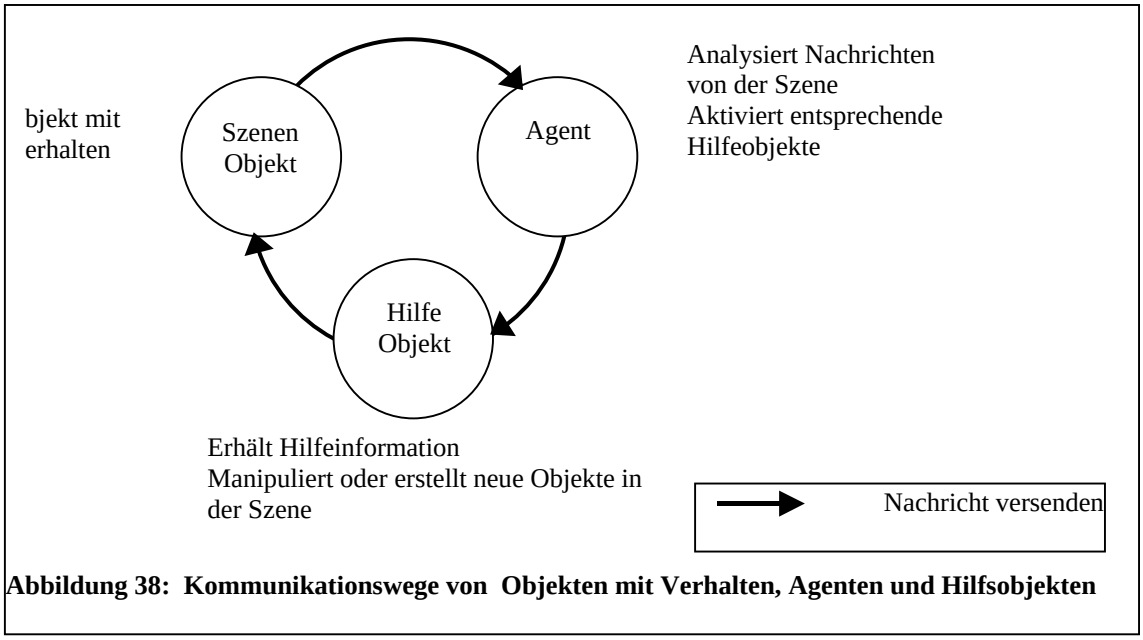


Abbildung 38: Kommunikationswege von Objekten mit Verhalten, Agenten und Hilfsobjekten

wiederum je nach Situation und Benutzertypus angewandt werden. Die Entscheidung darüber, welche eingesetzt werden soll, fällt dem Agenten zu. Und da die einzelnen Hilfen unterschiedlichen Informationsgehalt, Präsentations- und Mediencharakter haben, sollte jedes dieser Hilfeobjekte durch ein einzelnes autonomes Objekt realisiert werden. Durch den universellen Ansatz bildet das Messageconcept den Kern der Kommunikation zwischen diesen drei Objektklassen und nicht nur mit den (3D-)Objekten der Szene. Der Aufbau des einzelnen Nachrichtenpakets entspricht identisch dem des Vorgestellten.

8.6.2 Integrativer Ansatz

Hier soll eine allgemeinere Lösung für die Kommunikation der Objekte untereinander vorgestellt werden. Während im Hauptteil der Arbeit die Objekte in einem azyklischen hierarchischen Graph angeordnet waren, können hier die einzelnen Objekte der Applikation ungeordnet sein. Beim ersten Aufruf melden sich die Objekte an einer zentralen Instanz an, um ihren Namen mitzuteilen, damit die spätere Namensauflösung erfolgen kann. Die Objektklassifikation, die Objektnamen und das Standardnachrichtenformat entsprechen dem in Kapitel 8.4.6 beschriebenen. Mittels des darunterliegenden Messageconcept besteht nun die Möglichkeit, verschiedene Techniken aus 2D-, 3D- oder VR-Benutzungsoberflächen in einer Applikation zu kombinieren.

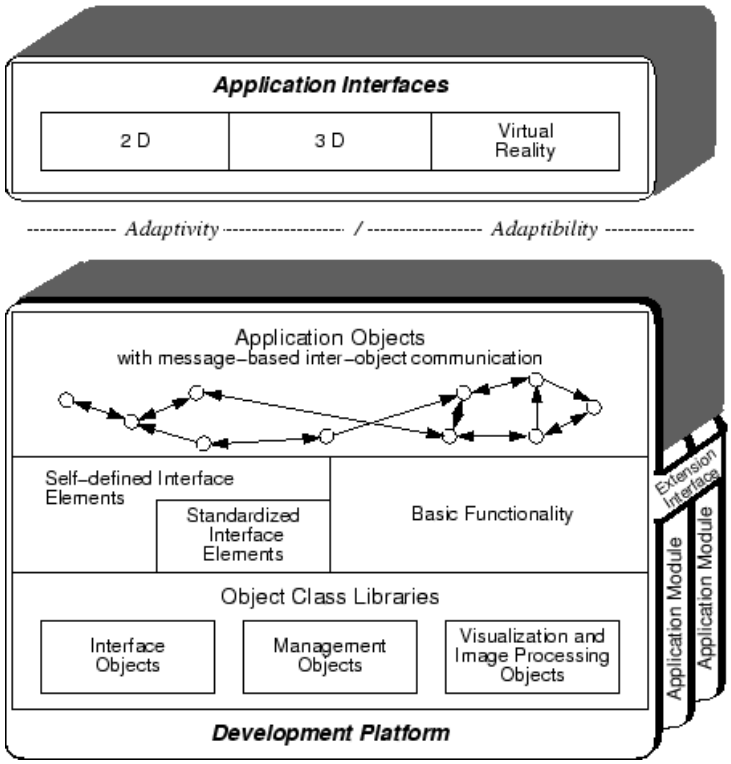


Abbildung 39: Modulare Architektur mit Integration der Teilobjekte durch Nachrichtenaustausch und klare Trennung von der Benutzungsfäche.

8.6.3 Konferenzsystem

In einer Vorstudie wurde ein Prototyp (genannt: sharedViewer) implementiert, der es zwei Benutzern ermöglicht, dieselbe Szene auf zwei verschiedenen Bildschirmen zu betrachten. Hierbei konnte zu einem Zeitpunkt nur ein Benutzer die Szene manipulieren, und es bestand die Möglichkeit, den anderen Kollegen durch einen gemeinsamen Zeiger auf interessante Bereiche aufmerksam zu machen. Ferner kann ein Benutzer auch die Blickposition seines Kollegen einnehmen. Sinn war es, die Anforderungen einer verteilten Anwendung als Vorstufe zur Teleradiologie kennenzulernen und im Messageconcept zu berücksichtigen. Diese proprietäre Lösung baute auf dem X11-Windows-System auf. Für ein professionelles telemedizinisches System, wie beispielsweise in der MEDStation integriert, bedarf es neben dem Austausch von Bilddaten über das Netz auch anderer Techniken, wie der Integration eines Videosystems, damit Fachgespräche, wie die Beurteilung von Befunden, über große Distanzen zwischen verschiedenen Spezialisten geführt werden können. Da die aufkommenden immensen Datenmengen eventuell gleichzeitig an viele entfernte Teilnehmer gesendet werden sollen, bietet sich ein Protokoll auf Multicast-Basis an. Offen bleibt, wie solch eine Konferenzszene in eine realistische 3D-Szene, wie zum Beispiel die Virtual Lightbox, zu integrieren wäre. Denkbar wäre ein separater virtueller Konferenzraum.

9 Zusammenfassung/Ausblick

Trotz intensiver Forschung im grafischen, dreidimensionalen Bereich, gibt es noch interessante und offene Fragen. Diese Arbeit beantwortet folgende Fragen:

- Wie kann eine medizinische 3D-Applikation für Radiologen aussehen?
- Welche Anforderungen werden an Interaktion und Benutzungsoberfläche gestellt?
- Wie müssen die internen 3D-Graphikobjekte beschaffen sein? Welche Anforderungen müssen diese erfüllen und wie können diese klassifiziert werden?
- Wie sieht eine solche Kommunikation zwischen den Objekten aus?
- Was für Möglichkeiten bietet das Messageconcept bezüglich Benutzerinteraktion und Objektverhalten?
- Welche besonderen Anforderungen leiten sich aus der Dreidimensionalität ab?

An einem Prototypsystem wurde die Tauglichkeit von 3D-Eingabegeräten und Stereoausgabe, sowie die Integrationsmöglichkeit von künstlichen Objekten für die Informationsvisualisierung, wie Kegelbäume, geprüft, für die radiologische Applikation als nicht geeignet eingestuft und somit verworfen. Ferner wurden verschiedene Metaphern untersucht. Durch den Einsatz moderner Software, basierend auf Open Inventor, wurde ein weitgehend realer Prototyp der radiologischen Umgebung, ohne Interaktionen und Animationen, realisiert. Es zeigte sich, daß eine realistische Umgebung die Grafikhardware deutlich überforderte und ein interaktives Arbeiten nicht möglich war. Aus diesem Grund wurde eine vereinfachte Umgebung geschaffen.

Die 3D-Objekte wurden hinsichtlich ihrer enthaltenen Informationen, Aufgaben und ihrer Funktionalität klassifiziert. Innerhalb einer Gruppe haben die Objekte aber verschiedenes Aussehen. Es wurde deshalb nach einem Weg gesucht, 3D-Objekte mit ähnlicher Funktionalität schnell zu realisieren. Ein anderer wesentlicher Aspekt der Analyse war, daß die Objekte miteinander kommunizieren müssen, um ihre Informationen auszutauschen, Aktionen steuern/koordinieren oder auslösen zu können. Der Autor entwickelte als Erster ein generelles Kommunikationskonzept, abgestimmt auf 3D-Objekte. Im Gegensatz zu später realisierten Modellen wie VRML 2.0/97 findet ein echter Nachrichtenaustausch statt. Ferner wurde für jedes dieser sogenannten "Objekte mit Verhalten" ein eindeutiger hierarchischer Name

vergeben. Hierdurch kann sehr einfach eine Gruppe von Objekten angesprochen werden. Beim allgemeinen Messageconcept liegt ein einheitliches Nachrichtenformat vor. Dieses besteht aus den vier Feldern Sender, Empfänger, der Nachricht selbst und optional den Daten. Da auch ein Nachrichtenaustausch zwischen verschiedenen Szenen, Applikationen oder über Netz mit verschiedenen Protokollen möglich ist, kann ein breites Spektrum der Kommunkationsmöglichkeiten abgedeckt werden. Multi-User-Anwendungen mit künstlichen Avataren können ebenso realisiert werden. Ferner können die Zustände von anderen Objekten abgefragt und auf ältere Nachrichten ausgewertet werden. Das allgemeine Messageconcept bietet einen universellen Ansatz für ein breit gefächertes Einsatzgebiet. Wie gezeigt, kann es in unterschiedlichen Bereichen, wie Hilfestellungen mit Agenten oder Applikationsobjekten, angewendet werden. Es wurde ferner auf die speziellen Anforderungen von VRML 2.0 (Dynamic World) übertragen. Das Nachrichtenformat wurde in diesem Fall vorher festgelegt oder kann auch benutzerspezifisch aus vordefinierten Datentypen zusammengestellt werden.

Um nochmals auf das Problem Geschwindigkeit zurückzukommen: Eine Parallelisierung, so daß jedes Objekt als eigener Thread implementiert wird, läßt zum gegenwärtigen Zeitpunkt die Szenengraphimplementation von Silicon Graphics nicht zu. Es gibt keinen Lockingmechanismus, der sicherstellt, daß mehrere unabhängige Prozesse gleichzeitig auf den Szenengraphen zugreifen oder ihn ändern können. Unklar ist trotz mehrfacher Nachfrage, ob dies mit der kommenden Version namens *Fahrenheit* machbar ist. Die Parallelisierung mittels Symmetrischem Multiprozessor System unter Windows NT soll in dieser Version zwar ausgenutzt werden, nähere Details, wie und was parallelisiert werden kann, fehlen. Eine andere Variante wäre die strikte Trennung der Nachrichten-Verhaltenskomponente von der Geometrie, welche eine Beziehung (engl. constraints) untereinander haben. Jeder der beiden Graphen könnte somit in einem eigenen Prozeß ablaufen. Vor dem Rendering müssen die Graphen aber abgeglichen werden. Der Renderingprozeß selbst sollte dann schneller ablaufen, das Problem der langsamen Darstellung von großen komplexen Szenen bliebe jedoch. Aus objektorientierter Sicht müßte die Beschreibung von Geometrie und Verhalten für ein Objekt zusammen erfolgen. Erst dann generiert ein Parser die zwei unterschiedlichen Graphen.

Erst durch neuere Graphikkartenarchitekturen[19] kann der Stereomodus mittels zweier Graphikpipelines preiswerter und schneller realisiert werden. Je eine Pipeline besteht aus einer eigenen Geometrie- und Rasterisierungseinheit und entspricht der Sicht eines Auges. Auch die Einbindung von einer oder mehreren CPUs je Pipeline ist möglich. Die automatische

[19] Intergraphs Wildcard 4100 Karte mit ParaScale™ sowie 3DLabs PowerThread™ Technologie, beide unter Windows NT, seien hier namentlich als Stellvertreter aufgeführt.

Lastverteilung auf zwei oder mehrere Pipelines im normalen (nicht stereo) Modus führt selbstverständlich auch zu einer Leistungssteigerung einer Applikation. Somit sind in den nächsten Jahren aufwendigere und wirklichkeitsgetreuere Virtuelle Umgebungen möglich, die einen vertrauteren Umgang zulassen. Die Entwicklung von neuartigen 3D-Eingabegeräten, Sprachsteuerung oder 3D-Ausgabegeräten nehmen ebenfalls eine zentrale Schlüsselstellung für die Akzeptanz von Anwendungen ein.

Was verteilte Applikationen angeht, so gibt es hier noch ein breites Forschungsfeld [BF96, BF97]. Sowohl für die Unterstützung von kooperativen oder konkurrenten Multi-User-Systemen (Gruppenentscheidungen, Mehrbenutzerinteraktion inklusive Kollisionserkennung) als auch der dazu geeigneten Infrastruktur (Netzwerkserver mit Fehlerkorrekturfunktionalität, Multicast-Protokollaufsatz) gibt es keinen Standard. Ebenso gibt es kein einheitliches standardisiertes Verfahren, wie eine Reduktion von Netzwerknachrichten in großen Welten zu bewerkstelligen sein soll. Dies kann zwar durch einzelne Verfahren, wie durch die Partitionierung der Welt in kleinere Bereiche, beispielsweise in sechseckige Felder wie bei NPSNET [MZ94], erfolgen, aber ebenso sind hierarchische Partitionierungen auf Objektbasis denkbar. Das heißt, die Nachrichten bleiben innerhalb eines Objektes, wie ein Zimmer in einem Haus oder ein Haus in einer Stadt, beschränkt.

Eine andere Herausforderung wäre ferner, daß die 3D-Objekte autonomer und intelligenter werden und somit eigene Schlüsse ziehen und selbsttätig handeln könnten. Hier ist nicht nur an einen Hilfeagenten zur Unterstützung in einer fremden Umgebung gedacht, sondern an einen Kommunikationspartner.

Das hier vorgestellte Messageconcept läßt aber noch Raum für zukünftige Arbeiten. Bisher werden die Objekte mit Verhalten programmiert, ebenso wäre eine Bibliothek von verschiedenen Verhaltensmustern denkbar. Ein neues Objekt entsteht dann durch einfaches Zusammenstellen dieser vordefinierten Verhaltenskomponenten, vielleicht mittels Grafikeditor. Auf dieselbe Weise können die Beziehungen zwischen den Objekten und die Interaktionen modelliert werden. Verschiedene offene Fragen stehen noch im Raum: Inwieweit sollen die Entwickler Objekte aus den Subkomponenten entwickeln und wann sollen sie auf vorgefertigte spezialisierte Objekte zurückgreifen? Wie weit soll die Spezialisierung der Objekte und auch ihr vordefiniertes Verhalten gehen? Der Vorteil des Messageconcept ist, daß es sich um ein allgemeines Konzept handelt. Es muß jedoch auch die kritische Frage gestellt werden, wann die Einheitlichkeit durchbrochen werden sollte. Vielleicht brauchen zwei Objekte eine wirklich schnelle Verbindung ("Firewire") miteinander und der vorgestellte

Nachrichtenaustausch dauert zu lange. Nicht abschließend beantwortbar ist auch die Frage: Wie weit soll die Autonomie der *Objekte mit Verhalten* gehen? Soll jedes Objekt Netzwerkkomponenten, eine Art Gedächtnis oder Manager für VR-Geräte haben und wann eignet sich ein separates globales Objekt hierfür?

Das Hauptproblem dieser Art der Virtuellen Umgebungen besteht nach wie vor darin, daß man für das interaktive Arbeiten wirkliche Hochleistungsgraphiken braucht (mindestens 5 Bilder/sec, um wenigstens den Eindruck von interaktiven Arbeiten zu haben und 15 Bilder/sec für wirkliches interaktives Arbeiten), um bewegte Bilder mit konstanten Bildraten anbieten zu können. Je realer die Anwendung, desto höher sind die Anforderungen an die zwei Faktoren polygonaler Geometriedurchsatz und Texture-Mapping-Qualität. Multiprozessorsysteme verhelfen zwar zu einer Leistungssteigerung, aber von einer ausreichenden Graphikleistung im Forschungs-, Industrie- oder Privatbereich sind wir mindestens ein bis zwei Hardwaregenerationen entfernt. Ferner ist die Entwicklung von neuartigen 3D-Eingabegeräten und autostereoskopischen Ausgabegeräten anzustreben. Die bisherigen Geräte können zwar in manchen Spezialfällen ganz vernünftig eingesetzt werden, sind aber nicht alltagstauglich. Auf der softwaremäßigen Seite ist man hier schon weiter, und es ist zu hoffen, daß bald die ersten allgemeinen Anwendungen erscheinen, die der virtuellen Welt auf jedermanns Schreibtisch zum Durchbruch verhelfen.

10 Anhang

10.1 Protokoll

Mitschnitt der Kommunikation zwischen den Objekten mit Erklärungen:

```
Vorgang: Hänge Bild an den ersten freien Platz eines aktiven Panels

Ereignis: Klick on Folder
Send Message: AddPicture from patbox1.folder to alternator data: ./ct1.rgb
Receiver  Alternator do:
   Send Query-Message: ?ShowedPanels from alternator to alternator.panel*
   Send Query-Message: ?FirstFreePictures from alternator to
alternator.panel1.frame* [Result of first Query]
   Send Message: SetPicture from alternator to alternator.panel1.frame1 [Result
of second Query] data ./ct1.rgb

Vorgang: Helligkeit mittels Fernbedienung für ein selektiertes Bild senken
Ereignis: Klick auf Taste Helligkeitvermindern der Fernbedienung

Send Message: SetDownUserBrightness from alternator.rmtcntr to
alternator.panel*
?SelectPicture
alternator.panel1.frame2 SetDownUserBrightness

Vorgang: Ausschnitt eines Bildes betrachten
Ereignis Klick auf das Vergrößerungsglas welches über ein Bild gebracht wurde.
?GetBoundingBox
  To alternator_panel1_frame1 From magglass

Vorgang: Ein Label an einem Bild befestigen
Ereignis Klick auf einen Bleistift welcher über ein Bild gebracht wurde.
alternator_panel1_frame2 Message: AttachLabel

Object: * Message: Delete From: Trash
Object : Trash Message: DeleteMe To Trash From None
```

10.2 Integration Ausgabegeräte und Open Inventor:

Spaceballeingabe:
(relative Position, Knöpfe)

Abgefrage bei Änderungen
Callback: Motion3Event, SpaceballButtonEvent

SCENE_IN_HAND
CAMERA_IN_HAND
WALKING
FLYING
MOVING_OBJECT
CONTINUE_WALKING
CONTINUE_FLYING
Periodischer Aufruf für kontinuierliche Bewegung (1. Ableitung) des letzten Wertes von Geschwindigkeit u. Rotation bei den Metaphoren:
MOVING_OBJECT
CONTINUE_WALKING
CONTINUE_FLYING

Hauptteil

```
// Case: Spaceball and Methaphor
  // Important to setup callbacks
  SoTransform *xform = new SoTransform;
  SoEventCallback *cbNode = new SoEventCallback;
  // Setup the transform nodes
  sceneGraph->addChild(xform);
  sceneGraph->addChild(cbNode);

//Case : Spaceball with Metaphor
  cbNode->addEventCallback(SoMotion3Event::
getClassTypeId(),eventSpacewithMetaphorCB, xform);
cbNode->addEventCallback(SoSpaceballButtonEvent::
getClassTypeId(), eventSpacewithMetaphorCB, xform);

  // automatic animation for contiuous
  //working,  flying
  ani = new SoTimerSensor(aniSensor, xform);
  ani->setInterval(SbTime(0.08));
  ani->schedule();

  sceneGraph->addChild(scene);
}
```

Berechnen der neuen Position und Orientierung.
Berücksichtigung von optionaler dominanter Filterung.

```
Void
EventSpacewithMetaphorCB(void *userData, SoEventCallback *cb)
{
  const SoEvent *event = cb->getEvent();
  SoCamera *camera = crtglbViewer->getCamera();
  SoTransform *xform = (SoTransform *) userData;
  SbVec3f trans, axis, newAxis;
  SbRotation rot;
  SbMatrix m;
  Float angle;

  if (event->isOfType(SoMotion3Event::getClassTypeId()))
  {
    SoMotion3Event *motion = (SoMotion3Event *) event;
    // get original value
    trans = motion->getTranslation();
    motion->getRotation().getValue(axis,angle);

    // Important, first look after dominate behavior !!!
    if (Spacebutton.getDominate()){ ... }
    // decide which translation can be used
    if (Spacebutton.getTranslation()){ ...  }
    // decide which rotation can be used
    if (Spacebutton.getRotation()) { ... }
    // calculate new angel
    // get old orientation
    m.setRotate(camera->orientation.getValue());
    // importen to get the right direction
    m.multVecMatrix(trans,trans);
    axis*=angle; angle = axis.length(); axis.normalize();
    //orient spaceball rotation to our camera orientation
    m.multVecMatrix(axis,newAxis);
    axis = newAxis;

    // set new Values
    motion->setTranslation(trans); rot.setValue(axis,angle); motion->setRotation(rot);

    // save last velocity values for fly
    if (angle!=0)  { svangle=angle; svaxis=axis; }
    if ((trans[0] != 0) && (trans[1] != 0) || (trans[2] != 0)) svtrans = trans;
    svstart=1;

    switch (Spacebutton.getMetaphor())
    {
    case SpaceballStatus::SCENE_IN_HAND:
```

```
      xform->translation.setValue(
      xform->translation.getValue() + motion->getTranslation().getValue());
      xform->rotation.setValue(
      xform->rotation.getValue() * motion->getRotation().getValue());
    break;
    case SpaceballStatus::CAMERA_IN_HAND:
    case SpaceballStatus::WALKING:
    case SpaceballStatus::FLYING:
      camera->position.setValue(camera->position.getValue() + trans);
      camera->orientation.setValue(camera->orientation.getValue() * rot);
    break;
    case SpaceballStatus::MOVING_OBJECT:
    case SpaceballStatus::CONTINUE_WALKING:
    case SpaceballStatus::CONTINUE_FLYING:
      // nothing because svtrans and rot already set
    break;
   }
  }
 ...
}


//
void
SoSceneViewer::aniSensor(void *data, SoSensor *)
{
 SbRotation rot;
 SoTransform *xform = (SoTransform*)  data;
 SoCamera *camera = crtglbViewer->getCamera();
  // set new Values
  if (svstart==0) return;
  rot.setValue(svaxis,svangle);

  switch (Spacebutton.getMetaphor())
  {
  case SpaceballStatus::MOVING_OBJECT:
    xform->translation.setValue(xform->translation.getValue() + svtrans);
    xform->rotation.setValue(xform->rotation.getValue() * rot);
    break;
  case SpaceballStatus::CONTINUE_WALKING:
  case SpaceballStatus::CONTINUE_FLYING:
    camera->position.setValue(camera->position.getValue()
    + svtrans);
    camera->orientation.setValue(camera->orientation.getValue()
    * rot);
        break;
  }
}
```

Trackereingabe
(absolute Position im Raum)

Periodische Abfrage der Trackerposition über Timerknoten.
Kopplung mit Spaceball nicht mit allen Metaphoren sinnvoll. (Freiheitsgrade einschränken)
Trennung von Spaceball und Trackerbewegung (analog: Kopf und Köperbewegung) oder Synergie (beide Geräte wirken auf das selbe Auge)
wie hier gezeigt.

Hauptteil

```
// Case: Spaceball and Tracker
  // Important to setup callbacks
  SoTransform *xform = new SoTransform;
  SoEventCallback *cbNode = new SoEventCallback;
  // Setup the transform nodes
  sceneGraph->addChild(xform);
  sceneGraph->addChild(cbNode);
  cbNode->addEventCallback(SoMotion3Event::
getClassTypeId(),eventSpacewithTrackerCB, xform);
  cbNode->addEventCallback(SoSpaceballButtonEvent::
getClassTypeId(), eventSpacewithTrackerCB, xform);

  SoTimerSensor *tracktimer = new SoTimerSensor(
trackerCB, xform);
  tracktimer->setInterval( SbTime(0.1) );
  tracktimer->schedule();
  sceneGraph->addChild(scene);
}
```

Berechnung der neuen Position und Orientierung
Korrektur von Trackerfehlern

```
void
eventSpacewithTrackerCB(void *userData, SoEventCallback *cb)
{
  const SoEvent *event = cb->getEvent();
  SoTransform *xform = (SoTransform*) userData;
  SoCamera *camera = crtglbViewer->getCamera();

  SbVec3f trans, axis;
  Float angle;

  If (event->isOfType(SoMotion3Event::getClassTypeId()))
  {
    SoMotion3Event *motion = (SoMotion3Event *) event;
    if (first==TRUE)
    {
      crtglbViewer->viewAll();
      camera = crtglbViewer->getCamera();
      c1.setValue(1., 0., 0.); c2.setValue(0., 1., 0.); c3.setValue(0., 0., 1.);
      cc1.setValue(1., 0., 0.); cc2.setValue(0., 1., 0.); cc3.setValue(0., 0., 1.);
     first =FALSE;
    }

    // Original value
    trans = motion->getTranslation();
    motion->getRotation().getValue(axis,angle); // get orig. Values
    camera->position.setValue(camera->position.getValue() + trans[0]*cc1 + trans[1]*cc2
+ trans[2]*cc3 );
    // Rotation
    if (axis[1]<0) angle=-angle;
    SbRotation RM(cc2,angle);
    RM.multVec(cc1,cc1); RM.multVec(cc2,cc2); RM.multVec(cc3,cc3);
    RM.multVec(c1,c1); RM.multVec(c2,c2); RM.multVec(c3,c3);
    camera->orientation.setValue(camera->orientation.getValue()*RM);
  }
 }

void trackerCB(void *userData, SoSensor *data)
{
 float px,py,pz,ax,ay,az;
 SbVec3f trans ;
 Float angle;
 Float dpx,dpy,dpz;
 Static float tdpx,tdpy,tdpz;
 Static float oldax, olday, oldaz;
 float dax, day, daz;

 SoCamera *camera = crtglbViewer->getCamera();
```

```
 If ( logiRead(&px,&py,&pz,&ax,&ay,&az,&but) == 0)
   {
   if (first==TRUE)
    {
     // Init
     crtglbViewer->viewAll();
     camera = crtglbViewer->getCamera();
     c1.setValue(1., 0., 0.); c2.setValue(0., 1., 0.); c3.setValue(0., 0., 1.);
     cc1.setValue(1., 0., 0.); cc2.setValue(0., 1., 0.); cc3.setValue(0., 0., 1.);
     // save Values
     tdpx = px; tdpy = py; tdpz =pz;
     oldax = ax; olday = ay; oldaz = 0;
     first =FALSE;
  }
  trans = SbVec3f(px,py,pz);
  dpx = (px - tdpx); dpy = (py - tdpy); dpz = (pz - tdpz;
  if ( (fabs(dpx)>10.0) || (fabs(dpy)>10.0) || (fabs(dpz)>10.0))return;   // mybe the
tracker track false

  tdpx = px; tdpy = py; tdpz =pz;
  trans.setValue(dpx, dpy,dpz);
  camera->position.setValue(camera->position.getValue() + trans[0]*c1 + trans[1]*c2 +
trans[2]*c3 );
  dax = (ax-oldax); day = (ay-olday); daz = 0 ;
  // look if the boundary has crossed
  ...
  oldax = ax; olday = ay; oldaz = 0;
  // Rotation grad to radians
  SbRotation R1(c1,dax * PI / 180.0);  SbRotation R2(c2,day * PI / 180.0);
  SbRotation RM=R1*R2;
  RM.multVec(c1,c1); RM.multVec(c2,c2); RM.multVec(c3,c3);
  camera->orientation.setValue(camera->orientation.getValue()*RM);
 }
}
```

10.3 Anhang Feldnamen (Dynamic Worlds)

SFAddress/MFAddress

Dieses Feld enthält eine (SFAddress), keine oder mehrere (MFAddress) Objekt- bzw. Knotenadressen.

```
Beispiel: World.Tuebingen.University.Floor9th.RoomNoP16

SFAddress identifier objectAddress

MFAddress [identifier objectAddress,
           identifier objectAddress,
           ... ]
```

SFCondition

Dieses Feld kann eine Bedingung enthalten. Als Ausdruck ist jeder Feldtyp erlaubt. Eine Kombination ist mit den folgenden Operatoren erlaubt. <op>: ==, !=, <, >,<=, >=, && (and) , || (or). Mehrere Bedingungen können durch Klammern geschachtelt werden.

```
FieldName [<expression>, <expression>]
expression: identifier <op> identifier
            (identifier <op> identifier)
            fieldName
            value
```

Beispiel für SFCondition:

```
Trigger {
    inputs [ SFVec3f pos, SFColor col]
    condition [((pos[0] >= 0.0) && (pos[0] <= 10.0)) || (col == 0.0)]
}
```

SFField/MFField

Dieses Feld enthält einen (SFField), keinen oder mehrere (MFField) Einträge eines Knotenfeldes. Jeder Eintrag besteht aus dem Feldtyp gefolgt von einem eindeutigen Bezeichner. Wahlfrei ist eine Vorinitialisierung erlaubt.

```
SFFloat angle 45.0
SFString myname "Otto"]

SFField fieldType identifier <defaultValue>

MFField [ fieldType identifier <defaultValue>,
          fieldType identifier <defaultValue>,
          ... ]
```

SFInput/MFInput

Dieses Feld enthält einen (SFInput), keinen oder mehrere (MFInput) Einträge von Nachrichtentypen die empfangen werden. Der Bezeichner muß eindeutig sein.

```
SFInput:
messageTransform transform

MFInput:
[ messageRotation rot, messageSFLong number, messageCube shape]

SFInput messageType identifier
```

```
MFInput [ messageType identifier,
          messageType identifier,
          ... ]
```

SFOutput/MFOutput

Dieses Feld enthält einen (SFOutput), keinen oder mehrere (MFOutput) Einträge von Nachrichtentypen die versendet werden. Der Bezeichner muß eindeutig sein. Eine Initalisierung mit einem Vorgabewert ist wahlfrei möglich.

Beispiel:

```
SFOutput:
messageTransform transform { translation 1.0 2.0 0.0
                             rotation 0.0 1.0 0.0 0.3 }

MFOutput:
[ messageRotation rot { rotation 1.0 1.0 0.0 1.5 },
  messageSFLong number { value 5},
  messageCube shape { width 4.0 recipients [*myObject.shape] ]
```

```
SFOutput messageType identifier initialization

MFOutput [ messageType identifier initialization,
           messageType identifier initialization,
           ... ]
```

SFRegister/MFRegister

Dieses Feld enthält einen (SFRegister), keinen oder mehrere (MFRegister) Einträge von Nachrichtentypen die registriert werden sollen. Im Gegensatz zum SFInput-Feld, werden die empfangenen Nachrichten in Tabellenform abgespeichert. Auf eine einzelne Nachricht kann mittels des Indexbezeichners [] zugegriffen werden. Die Nachrichten werden in der Reihenfolge der Ankunft abgespeichert, das heißt der erste erhält Index n=1 der folgende n+1. Die Anzahl der gespeicherten Nachrichten kann mittels des Index mit n=0 abgefragt werden.

```
SFRegister:
messageTransform transform

MFRegister:
[ messageRotation rot, messageSFLong number, messageCube shape]

rot[0]    // Number of registered messages
shape[10] // Single message number 10 in the array
```

```
SFRegister messageType identifier

MFRegister [ messageType identifier,
             messageType identifier,
             ... ]
```

10.4 VR-Historisches

Chronologisch aufgeführt sind die wichtigsten Ereignisse im Überblick, welche die VR-Entwicklung beeinflußten.

Jahr	Ereignis
1960	Morton Heling stellt den *Sensorama* Prototypen vor, eine multisensorische Umgebung, bei der ein Teilnehmer einige vorher aufgenommenen Erfahrungen sehen, hören, riechen und fühlen kann.
1965	Ivan Sutherland schlägt das *Ultimate Display* vor: "A display connected to a digital computer gives us a chance to gain familiarity with concepts not realizable in the physical world... The screen is a window through which one sees a virtual world. The challenge is to make that world to look real, act real, sound real, feel real."
1968	Ivan Sutherland baut das *Sword of Damocles*, das erste Head Mounted Display bestehend aus zwei Kathodenröhren.
1971	Fredericke Brooks entwickelt das GROPE-II Prototypsystem, welches Kraft-Rückkopplungen (engl. force-feedback) erlaubt.
1975	Myron Krueger zeigt *VIDEOPLACE* "a conceptual environment with no physical existence"
1982	Thomas Furnes III demonstriert *VCASS (Visually Coupled Airborne Systems Simulator)* ein Head Mounted Display inkl. Tracking mit 6 Freiheitsgraden von Position und Orientierung. Es erlaubte die erste komplette Isolierung des Benutzers von der umgebenden Welt.
1985	Scott Fischer entwickelt im Rahmen des VIVED Projekt die AMES Virtual Interface Environment Workstation. " a head-mounted, wideangle, stereoscopic display system controlled by operator position, voice and gesture for use as multipurpose interface environment"
1985	VPL Research wurde gegründet. Der *DataGlove,* entwickelt von Thomas Zimmermann, wird als erstes Produkt vorgestellt. Das Gerät ermöglicht die Ermittlung der Position/Orientierung jedes einzelnen Fingers.
1988	VPL Research stellt das Produkt *Eyephone* vor, ein kommerziell erhältliches Head Mounted Display.
1989	Autodesk präsentiert das erste PC-basierte VR-System
1989	Die Firma Fake Space Labs stellt mit *BOOM* ein neues VR-Gerät vor. An einem mechanischen Arm befindet sich eine Box, welche zwei Bildschirme enthält. Diese Konstruktion ermöglicht das Messen der Position und Orientierung der Box. Ein Benutzer kann die Box greifen und mittels Bewegungen durch die virtuelle Welt navigieren.
1992	Electronic Visualization Laboratory entwickelt *CAVE,* ein Umgebungs-Stereo-Projektionssystem.
1995	*Responsive Workbench,* entwickelt von der GMD, wird vorgestellt. Über einen Arbeitstisch werden stereoskopische Bilder projiziert.

Abgeändert nach [CC93]

Übersicht über 2D-und 3D-Interaktionen [PA94]

Applikation	Textverarbeitung		Kalkulation		Graphik	
UI-Toolkits	Windows GUI		OSF/Motiv		OpenLook	
Metaphor	Desktop Metaphor			Direkte Manipulation		
Interaktionsaufgabe	Selektion		Picken			
Interaktionstechnik	Menü	Icon	Knopf	Drücken (Click)	Ziehen (Drag)	
Nachrichten/ Ereignisse	X/Y Position	Taste gedrückt	Knopf gedrückt	Bewegung		
Eingabegeräte	Tastatur	Maus	Joystick	Trackball		

Referenzmodell für 2D-Interaktion

Applikation	Virtuelle Realität		CAD		Simulation	
3D-API/Toolkit	OpenGL		Inventor		Direct3D	
Metaphor	VR-Metaphor		Direkte Manipulation		Architektur-Metaphor	
Interaktionsaufgabe	Selektion	Rotation	Skalieren	Bewegen	Modi-fizieren	Navi-gieren
Interaktionstechnik	3D-Menü	Zeigen	Greifen	Loslassen		
Nachrichten/ Ereignisse	2D-Bewegung	Taste gedrückt	3D-Bewegung (6DOF)	Dreh-moment	Druck-kraft	
Eingabegeräte	Spaceball	3D-Maus	Joystick	Eye-Tracker	Daten-Handschuh	

Referenzmodell für 3D-Interaktion

Erfinder von Graphischen Elementen von Benutzungsoberflächen (nach [SA96])

Element	**Erfinder**
Tastaturbasierte Menüs	Unbekannt, vor 1977
Hierarchische Textmenüs	Unbekannt, wahrscheinlich UCSD Pascal 1978 oder früher
Bitmap Graphikbildschirme	Xerox Parc, 1978 PERQ war das erste kommerielle Produkt
BitBLT Raster Operationen	Xerox Parc (Dan Ingalls)
Lichtgewehr	SAGE Luftverteidigungssystem und Vorläufer. Cape Cod System MIT, ca. 1995.
Lichtgriffel	Unbekannt, 1960 oder früher
Joystick	Analoger Luftverkehrskontrollschirm in den 50er Jahren, z. B. RBDE-5 (Radar Bright Display Equipment) von Raytheon. Ebenso Weltraumspiele 1962 oder früher
Trackball	Digitale Flugkontrollschirme, Mitte der 60er Jahre
Zeigegeräte mit Bildschirmanzeige, Maus	Douglas Engelbart, Stanford Research Institute, 1963 erste Erwähnung, Weiterentwicklung Mitte der 70er Jahre
Änderung des Cursors um Systemstatus anzuzeigen	Xerox PARC, William Newmann
Änderung des Cursors um Kontext anzuzeigen	David Tilbrook, 1975
Menüs	Xerox PARC
Popup Menüs	Xerox Parc (Dan Ingalls)
Pulldown Menüs	Apple (LISA-System)
Menüleiste	Apple (LISA-System)
Hierarchische Menüs	Xerox PARC Paeth bei Smalltalk
Menüeinträge löschen	Apple (LISA-System) oder Ed Anson 1980 oder früher oder Xerox PARC 1982 oder früher
Tastenabkürzungen für Menüeinträge	Apple (LISA-System) oder Ed Anson 1980 oder früher
Checkmarken für Menüeinträge	Apple (LISA-System)
Überlappende Fenster	Xerox Parc (Dan Ingalls)
Geteilte Fenster	Xerox Parc
Nachrichtenschlangen (Event Queues)	Simula, dann Apple (LISA-System) oder Ed Anson, GKS 1975
Icons	Xerox Parc David Smith (XeroxStar), dann LISA, Macinthosh
Laufleisten Scroll Bars	Xerox Parc
Push Button	Xerox Parc
Radio Button	Xerox Parc Kaehler
Check Box	Xerox Parc
Grauschattierung von inaktiven Schaltern	David Tilbrook 1995
Dialogboxen	Xerox Parc XeroxStar
Konzept von Ressourcen	Apple Horn
Mehrere Schriftarten und Stile im Text	Xerox Parc oder Wang Textverarbeitung (1978 oder früher)
Modeless Interaction	Xerox Parc Tesler
Kopieren, Einfügen, Ausschneiden mit Maus	Xerox Parc
Listboxen	Xerox Parc

11 Anhang: Publikationen, die im Zusammenhang mit dieser Arbeit bereits erschienen sind

[1] J. Fechter, T. Grunert, L. M. Encarnação, and W. Straßer, User-Centered Development of medical Visualization Applications: Flexible Interaction through Communicating Application Objects., Computer & Graphics, Special Issue on medical Imaging, Vol. 20, No. 6, 763-774, 1996.

[2] L. M. Encarnação, J. Fechter, T. Grunert, and W. Straßer, User-Tailored Interaction Development in 2D, 3D, and VR. In Computer Graphics Forum, Proc. Of the EUROGRAPHICS '96 Conference, Vol. 15, No. 3, 433-442, 1996.

[3] H.-H. Ehricke, J. Fechter, W. Straßer, and R. Niemeyer, A Virtual Environment Approach to the Design of Man-Machine Interface in Medicine. In P. Barahona, M. Veloso, and J. Bryant, editors, Medical Informatics in Europe ´94, 551-555, 1994.

[4] T. Grunert, J. Fechter, G. Stuhldreier, H.-H. Ehricke, M. Skalej, R. Kolb, and P. E. Huppert, A PACS Workstation with integrated CASE tool and 3D-Endosonography application. In Computer Assisted Radiology CAR95, 293-298, 1995.

[5] T. Grunert, H. H. Ehricke, J. Fechter, R. Kolb, and M. Skalej, PACS Man-Machine Communication via Virtual Environments. In EuroPACS'94, 1994.

[6] H.-H. Ehricke, T. Grunert, T. Buck, J. Fechter, U. Kloos, W. Straßer, and R. Kolb, Imaging and Graphics in medicine: Concept of an Object-Oriented Platform for Clinical Research. In P. Barahona, M. Veloso, and J. Bryant, editors, Medical Informatics in Europa '94, 567-572, 1994.

[7] W. Broll and J. Fechter, Interaction and Behavior in web-based Shared Virtual Environments, In IEEE Global Internet'96, 1996.

[8] W. Broll, D. England, J. Fechter, and T. Koop., The Power of Dynamic Worlds, VRML 2.0 Proposal, http://www.gris.uni-tuebingen.de/gris/proj/vr/proposal/dynamic-Worlds.html, 1996.

[9] W. Broll, J. Fechter und D. Schick, Avatare und Assistenten in Multiuser-VRML-Umgebungen, AAA '97 Agenten, Assistenten, Avatare, 1997.

[10] T. de A. Buck, J. Fechter, and Wolfgang Straßer, Rule-based image processing environment applied to medical imaging. In H. U. Lemke, K. Inamura, C.C. Jaffe, and R. Felix, Computer Assisted Radiology (CAR) 93, Springer Verlag, 1993.

Technical Reports:

[1*] J. Fechter, Mensch-Maschine Interaktion mit Einbindung von Techniken der Virtuellen Realität an einem Beispiel der Medizin, in Jahresbericht 1993 Arbeitsbereich GRIS, Hrsg. Prof. Dr.-Ing. W. Straßer, Bericht WSI-GRIS 3-94, ISSN 0946-3852, 1993.

[2*] J. Fechter, Mensch-Maschine Interaktion mittels nachrichtenbasierter Objekt-kommunikation in Virtuellen Umgebungen, in Jahresbericht 1994 Arbeitsbereich GRIS, Hrsg. Prof. Dr.-Ing. W. Straßer, Bericht WSI-GRIS 3-95, ISSN 0946-3852, 1994.

[3] W. Broll, D. England, J. Fechter, and T. Koop, Towards Interactive Virtual Environments: Interaction and Behavior Extensions to VRML, VRML 2.0 Proposal, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, Technical Report, No. WSI--96--11, ISSN 0946-3852, 1996.

[4] L. M. Encarnação, J. Fechter, and T. Grunert, User-Tailored Interaction Development in 2D, 3D, and VR with UC-AID med. Technical Report WSI-96-5, Wilhelm-Schickard Institut für Informatik, University of Tübingen, Germany, 1996.

[5] L. M. Encarnação and J. Fechter, Applying Adaptive Interface reasoning to Virtual Environments, Technical Report WSI-94-17, Wilhelm-Schickard Institut für Informatik, University of Tübingen, Germany, Oktober 1994. Erschien auch modifiziert in: Abridged Proc. (Poster Session) of the 6th International Conf. On Human-Computer Interactions. (HCI International '95), Elsevier, 3, 1995.

* Es sollen nicht alle Jahresberichte aufgeführt werden, sondern nur diejenigen welche wesentliche Schlüsselelemente enthalten und zu diesem Zeitpunkt noch nicht publiziert waren.

LITERATURVERZEICHNIS

[AR92] R. Akka, Automatic software control of display parameters for stereoskopic graphics images, StereoGraphics Corporation, 2171-H East Franciso Blvd., San Rafael, CA 94901, 1992.

[BB90] C. Blanchard, S. Burgess, Y. Harvill, J. Lanier, A. Lasko, M. Obermann, and M. Teitel, Reality Built for Two: A Virtual Reality Tool, AMC SIGGRAPH Computer Graphics, 1990 Symposium on Interactive 3D Graphics, Vol. 24, No. 2, 1990.

[BE96a] W. Broll, D. England, J. Fechter, and T. Koop, Towards Interactive Virtual Environments: Interaction and Behavior Extensions to VRML, VRML 2.0 Proposal, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, Technical Report, No. WSI--96--11, ISSN 0946-3852, 1996.

[BE96b] W. Broll, D. England, J. Fechter, and T. Koop, The Power of Dynamic Worlds, VRML 2.0 Proposal, http://www.gris.uni-tuebingen.de/gris/proj/vr/proposal/dynamic-Worlds.html, 1996.

[BF93] S. Bryson and S. Feiner. Virtual Reality for Visualization. In IEEE Visualization '93 Tutorial 1, 1993.

[BF96] W. Broll and J. Fechter, Interaction and Behavior in web-based Shared Virtual Environments, In IEEE Global Internet'96, 1996.

[BF97] W. Broll, J. Fechter und D. Schick, Avatare und Assistenten in Multiuser-VRML-Umgebungen, AAA '97 Agenten, Assistenten, Avatare, 1997.

[BG95] J. Boyle and P. M. D. Gray, The Design of 3D Metaphors for Database Visualization, IFIP 3rd Visual database Conference, 1995.

[BP94] G. Bell, A. Parisi, and M. Pesce. The Virtual Reality Modeling Language. Version 1.0, http://www.eit.com/vrml/vrmlspec.html, 1994.

[BS91] S. Bryson, Interaction of Objects in a virtual environment: a two-point paradigm, RNT Technical Report RNR-91-009, 1991.

[BW97] W. Buxton, A directory of sources for input technologies, http://www.dgp.toronto.edu/~people/BillBuxton/InputSources.html, 1997.

[BW98] W. Broll, Objekt-Orientiertes Interaktionsmodell zur Unterstützung verteilter virtuellen Umgebungen, Dissertation Eberhard-Karls-Universität Tübingen, im Druck, 1998.

[CC93] C. Cruz-Neira, Virtual Reality Overview, In: Applied Virtual Reality, ACM SIGGRAPH 20th International Conference on Computer Graphic and Interactive Techniques, Cource Notes 23, 1993.

[CJ87] J. Coutaz, PAC, an Object Oriented Model for Dialog Design, In H.-J. Bullinger und B. Shackel (Hrsg.), Human-Computer Interaction Interact'87, Elsevier Science Publishers, 431-436, 1987.

[CH97] P. E. Chung, Y. Huang, S. Yajnik, D. Liang, J. C. Shih, C.-Y. Wang, and Y.-M. Wang, DCOM and CORBA Side by Side, Step by Step, and Layer by Layer. http://www.bell-labs.com/~emerald/dcom_corba/Paper.html, 1997.

[CL93] C. Cruz-Neira, J. Leigh, M. Papka, C. Barnes, S. Cohen, S. Das, R. Engelmann, R. Hudson, T. Roy, L. Siegel, C. Vasilakis, T. DeFanti, and D. Sandin, Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment, Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality, 59-66, 1993.

[CS92] D. Conner, S. Snibbe, K. Herndon, D. Robbins, R. Zeleznik, and A. van Dam, Three-dimensional widgets, Computer Graphics 1992 Symposium on Interactive 3D Graphics, Vol. 25, No. 2, 183-188, 1992.

[CS93] C. Cruz-Neira, J. Sandin, and T. DeFanti, Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE, Computer Graphics, Proceedings of Siggraph 1993, 135-142, 1993.

[CW93] T. L. Clifton and F. L. Wefer, IEEE Computer Graphics and Application, Vol 13, No. 4/57, 1993. Siehe auch: http://www.elec.canterbury.ac.nz/labs/graphics/av_vol.html

[DH93] D. J. Duke and M.D. Harrision, Abstract Interaction Objects, Computer Graphics Forum,Vol. 12, No. 3, 25-36, 1993.

[DM92] M. Deering, High Resolution Virtual Reality, SUN Microsystems Corporation, ACM Computer Graphics, Siggraph ´92, 1992.

[DM95] N. I. Durlach and A. S. Mavor, Virtual Reality Scientific and Technological Challenges, National Academy Press: Washington DC, 1995.

[DS88] S. Deering, Host Extensions for IP Multicasting, RFC 1112, Stanford University, 1988.

[DW80] W. J. Dallas, Computer Generated Holograms, The Computer in Optical Research, Topics in applied Physics, Springer: New York, Vol. 41, 291, 1980.

[EF94] H.-H. Ehricke, J. Fechter, W. Strasser, and R. Niemeyer. A Virtual Environment Approach to the Design of Man-Machine Interface in Medicine. In P. Barahona, M. Veloso, and J. Bryant, editors, Medical Informatics in Europe '94, 551-555, 1994.

[EF95] L. M. Encarnação and J. Fechter, Applying Adaptive Interface reasoning to Virtual Environments, Abridged Proc. (Poster Session) of the 6th International Conf. On Human-Computer Interactions. (HCI International '95) ,Elsevier, 3, 1995.

[EG94] H.-H. Ehricke, T. Grunert, T. Buck, J. Fechter, U. Kloos, W. Straßer, and R. Kolb, Imaging and Graphics in medicine: Concept of an Object-Oriented Platform for Clinical Research. In P. Barahona, M. Veloso, and J. Bryant, editors, Medical Informatics in Europe '94, 567-572, 1994.

[EM97] L. M. Encarnação, Concept and realization of intelligent user support in interactive graphics applications, Disseration, Eberhard-Karls-Universität Tübingen, 1997.

[ES94] S. R. Ellis, What are virtual environments?, IEEE Computer Graphics and Applikations, Vol. 14, No. 1, 17-22, 1994.

[ES97] S. R. Ellis, Virtual Environments and Environental Instruments, http://duchamp.arc.nasa.gov/papers/veei/veei.html, 1997.

[FG96] J. Fechter, T. Grunert, L. M. Encarnação, and W. Straßer, User-centered Development of Medical Visualization Applications: Flexible Interaction through Communicating Application Objects, Computer & Graphics, Special Issue on Medical Imaging, Vol. 20, No. 6, 763-774, 1996.

[FI97] I. Feldberg, Lan's VR buying guide, http://www.cs.jhu.edu/~feldberg/vr/vrbg.html, 1997.

[FL97] Floating Images, Real-Depth 3-D Imaging, White Paper, http://www.floatingimages.com/whiteppr.html, 1997.

[FP54] P. M. Fitts, The information capacity of the human motor system in controlling the amplitude of movement, Journal of Experimental Psychology, Vol. 47, 381-391, 1954.

[FP90] G. Faconti and F. Paterno, An approach to the formal specification of the components of an interaction. In C. Vandoni and D. Duce,editors, Eurographics 90, 481-494, 1990.

[FT92] T. Fröhlich, Interaktion in stereoskopischen Bildern, Technische Hochschule Darmstadt Fachbereich Informatik Fachgebiet Graphisch-Interaktive Systeme, 1992.

[GC97] C. Graham, Database Visualization and VRML: cyber23: Virtual Architecture: Database Visualization, http://www.best.com/~cyber/virarch/article.html,1997.

[GF95] T. Grunert, J. Fechter, G. Stuhldreier, H.-H. Ehricke, M. Skalej, R. Kolb, and P. E. Huppert, A PACS Workstation with integrated CASE tool and 3D-Endosonography application. In Computer Assisted Radiology CAR95, 293-298, 1995.

[GH98] R. Grzeszczuk, C. Henn, and R. Yagel, Advanced Geometric Techniques for Ray Casting Volumes, SIGGRAPH98, Cource Notes, 44-104, 1998.

[GJ93] J. Gallenbacher, Grafische Eingaben: Geräte, Methoden, Perspektiven, Evaluierung, Diplomarbeit am Fachbereich Graphische Interaktive Systeme TH Darmstadt, 1993.

[HD94] K. Herndon, A. d. Dam, and M. Gleicher, Workshop on the challenges of 3d interaction. CHI Bulletin, Vol. 26, No. 4, 1994.

[HE83] E. M. Howlett: Wide angle color photography method and system, U.S. patent Nr. 4406532, 1983.

[HG98] S. Hopwood and A. Gordon, Fahrenheit Defining the Future of Graphics, http://www.sgi.com/developers/technology/graphics/fahrenheit.html, 1998.

[HK94] M. Hemmje, C. Kunkel, and A. Willett, LyberWorld - A Visualization User Interface Supporting Fulltext Retrieval in Croft, W.B. and van Rijsbergen, C.J. (eds) Proc. of the 17th Annual Int. Conference on Research and Development in Information, Springer Verlag, 249-257, 1994.

[HS93] B. A. Hobbs and M. R. Stytz, A User Interface to a True 3-D Display Device, Proceedings of the Fifth International Conference on Human-Computer Interaction, Vol. 2, 579-584, 1993.

[HS97] S. Hölldobler, 3D LCD-Display, TU-Dresden Projektgruppe 3D-Display, http://ddd001.inf.tu-dresden.de/~3ddisp/, 1997.

[IC97] Immersion Corporation, Medical Products,WWW: http://www.immerse.com/WWWpages/medical.html, 1997.

[IE93] IEEE Institute of Electrical and electronics Engineers, International Standard, ANSI/IEEE Standard 1278-1993, Standard for Information Technology, Protocols for Distributed Interactive Simulation, 1993.

[IN93] H. Iwata and H. Noma, Volume Haptization, Proceedings of the IEEE 1993 Symposium on Research Frontiers in Virtual Reality, 16-23, 1993.

[IS97] International Organization for Standardization (ISO), ISO/IEC JTC1/SC18/WG9 - User/Systems Interfaces and Symbols, http://www.ds.dk/ds/it/newlists/011809.htm, 1997.

[KA93] A. E. Kaufman, Volume synthesis principles, In H. Hagen, H. Müller, and G. M. Nielson, editors, Focus on Scientific Visualization, Springer, 123-137, 1993.

[KA93] A. J. Klinger, Mensch-Maschine-Dialog 2 (Eine Einführung in graphische Fenstersysteme.), Skript, Universität Karlsruhe, 1993.

[KB95] W. Krüger, A. Bohn, B. Fröhlich, H. Schüth, W. Strauss, and G. Wesche, The responsive workbench: A virtual work environment, IEEE Computer Dociety, Vol. 28, No. 7, 42-48, 1995.

[KG95] G. Knittel, A PCI-based Volume Rendering Accelerator, Proceedings of the 10th Eurographics Workshop on Graphics Hardware'95, 73-82, 1995.

[KJ95] J. Krätzschmar, Brille vergessen 3D-Displays für Raumillusion ohne Sehhilfe, c't, No. 11, 210-214, 1995.

[KL93] L. Kettner, Mathematisch-Informationstheoretische Untersuchung von 3D-Metaphern, Diplomarbeit, Universität Karlsruhe Institut für Betriebs- und Dialogsysteme Abteilung Dialogsysteme und graphische Datenverarbeitung, 1993.

[KM97] M. J. Kilgard, Realizing OpenGL: Two Implementations of one Architecture, Proceedings 1997 Siggraph/Eurographics Workshop on Graphics Hardware, ACM Siggraph, 45-55, 1997.

[KP88] G. E. Krasner und S. T. Pope, A Cookbook for using the Model-View-Controler User Interface Paradigm in Smalltalk-80, Journal of Object-Oriented Programming, 1988.

[KU94] U. Kloos, Graphisch-Interaktive Simulation unter Berücksichtigung medizinischer Fragestellungen, Dissertation, Fakultät für Informatik der Eberhard-Karls-Universität Tübingen, 1994.

[KV96] V. Kumar, MBone Interactive Multimedia on the Internet, New Riders Publishing, ISBN 1-56205-397-3, 1996.

[LL91] L. Lipton: The CrystalEyes Handbook, StereoGraphics Corporation, 2171-H East Franciso Blvd., San Rafael, CA 94901, ISBN 0-9629566-0-0, 1991.

[LM90] M. Levoy, A Hybrid Ray Tracer for Rendering Polygon und Volume Data, IEEE Computer Computer Graphics & Applications, Vol. 10, No. 2, 33-40, 1990.

[LM97] M. Lucente, Interactive three-dimensional holographic displays: seeing the future in depth, Computer Graphics (ACM SIGGRAPH) Vol. 31, No. 2, 1997.

[MH95] B. Mitra, Y. Honda, K. Matsuda, G. Bell, C. Yu, and C. Marrin, Moving worlds: behaviors for VRML, Presented at VRML'95 Symposium, http://webspace.sgi.com/moving-worlds, 1995.

[MP94] P. Maes, Social interface agents: Acquiring competence by learning from users and other agents. In O. Etzioni (Ed.), Software Agents - Papers from the 1994 Spring Symposium (Technical Report SS-94-03), AAAI Press, 71-78, 1994.

[MR91] J. D. Mackinlay, G. G. Robertson, and S. K. Card, The Perspective Wall: Detail and Context Smoothly Integrated. In Proceedings of ACM CHI'91 Conference of Human Factors in Computing Systems and Graphics Interface, ACM SIGCHI, ACM-Press, 173-179, 1991.

[MS94] T. H. Massie and J. K. Salisbury, The PHANToM Haptic Interface: A Device for Probing Virtual Objects, ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 1994.

[MS95] Microsoft Bob™, Microsoft, 1995.

[MZ94] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, NPSNET: A Network Software Architecture for Large Scale Virtual Environments, Presence, Teleoperators and virtual Environments, Vol. 3, No. 4, 265-287, 1994.

[OP97] R. Osborne, H. Pflister, H. Lauer, N. McKenzie, S. Gibsion, W. Hiatt, and T. Ohkami, EM-Cube: An Architecture for Low-cost Real-Time Volume Rendering, Proceedings 1997 Siggraph/Eurographics Workshop on Graphics Hardware, ACM Siggraph, 131-146, 1997.

[OS95] S. Omori, J. Suzuki, and S. Sakuma (Terumo Corp.), Stereoscopic Display System Using Backlight Distribution, SID'95 The Society for Information Display, ISSN 0097-0966X/95/2601, Vol. 26, 855-858, 1995.

[PA89] A. Pope, The SIMNET Network and Protocols, BBN Systems and Technologies, Cambridge, Massachusetts, BBN Report No. 7102, 1989.

[PA94] P. Astheimer et al, Die Virtuelle Umgebung -- Eine neue Epoche in der Mensch-Maschine-Kommunikation, Informatik-Spektrum, Vol. 17, No. 6, 1994.

[PB88] B. Page, R. Bölckow, A. Heymann, R. Kadler, and H. Liebert, Simulationen und moderne Programmiersprachen (Modula-2, C, Ada), Springer-Verlag, Fachbericht Simulation Band 8, 1988.

[PB98] B. Preim, Interaktive Illustrationen und Animationen zur Erklärung komplexer räumlicher Zusammenhänge, Dissertation, Informatik, Fakultät für Informatik Otto-von-Guericke-Universität Magdeburg, 1998.

[PJ80] J. Postel, User Datagram Protocol, RFC 768, 1980.

[PJ81] J. Postel, Transmission Control Protocol, RFC 793, 1981.

[PK94] M. D. Pesce, P. Kennard, and A. S. Parisi, Cyberspace, In Proceedings of The First International Conference on The World-Wide Web, 1994.

[PM91] M. J. Prime, Human Factors Assessment of Input Devices for EWS Rutherford Appleton Laboratory, RAL-91-033, 1991.

[PW97] P. Siegmund and W. Matthias, 3-D Displays: A review of current technologies, DISPLAYS, Vol. 17, 100-110, 1997.

[RA93] L. Rosenberg and B. Adelstein, Perceptual Decomposition of Virtual Haptic Surfaces, Proceedings of the IEEE 1993 Symposium on Research Frontiers in Virtual Reality, 46-53, 1993.

[RD96] D. Rogerson, Inside COM, Microsoft Press Redmond Washington, 1996.

[RM91] G. G. Robertson, J. D. Mackinlay, and S. K. Card, Cone Trees: Animated 3D Visualizations of Hierarchical Information, In Proceedings of ACM CHI'91 Conference of Human Factors in Computing Systems and Graphics Interface, ACM SIGCHI, ACM-Press, 189-194, 1991.

[RS95] S. Ressel, Aufbau einer Klassenbibliothek von dreidimensionalen Interaktions-elementen und Validierung räumlicher Anordnungskonzepte zur Point-of-Interest Betonung, Diplomarbeit, Institut für Simulation und Graphik, Otto-von-Guericke-Universität Magdeburg, 1995.

[SA93] M. Segal and K. Akeley, The OpenGL™ Graphics System: A Specification, Ver. 1.1, Silicon Graphics Inc, 1993.

[SA96] A. E. Siegman, An Unofficial History of Graphical User Interfaces, http://www-ee.stanford.edu/~siegman/GUI_history.html, 1996.

[SA97] A. K. Sinha, Netzwerkprogrammierung unter Windows NT 4.0 Systemarchitektur und Methoden der Interprozeß-Kommunikation, Addison-Wesley, ISBN 3-8273-1103-9, 1997.

[SC92] P. S. Strauss and R. Carey, An object-oriented 3D graphic toolkit, Computer Graphics, Vol. 26, No. 2, 341-349, 1992.

[SD97] D. Schick, Netzwerkunterstützung für verteilte VR-Systeme, Diplomarbeit, Wilhelm-Schickard-Institut für Informatik Lehrstuhl für Graphische Interaktive Systeme, Eberhard-Karls-Universität Tübingen, 1997.

[SE97] Sense 8 & Co, World ToolKit (WTK) http://www.sense8.com/products/wkt.htm, 1997.

[SI63] I. E. Sutherland, Sketchpad: A Man-machine Graphical Communication System, Proceedings of the Spring Joint Computer Conference, MD: Spartan Books, 328-345, 1963.

[SI65] I. E. Sutherland, The Ultimate Display, Proceedings of IFIPS Congress 1965, New York, New York, Vol. 2, 506-508, 1965.

[SI68] I. E. Sutherland, A Head-Mounted Three-Dimensional Display, AFIPS Conference Proceedings, Vol. 33, Part I, 757-764, 1968.

[SJ92] J. M. Spivey, The Z Notation: A Reference Manual, 2nd Edition, University of Oxford, 1992.

[SM97] T. Selker, I. May, and S. Zhai, Spatial Interface Can Facilitate Target Acquisition, IBM Almaden Research Center, http://www.almaden.ibm.com/cs/user/papers/spatial/spatial.htm, 1997.

[ST87] R. F. Schmidt und G. Thews, Physiologie des Menschen, 23 Auflage, Springer Verlag, ISBN 3-540-16685-8, 1987.

[SU97] SUN, Java3D API, http://java.sun.com/products/java-media/3D/index.html, 1997.

[SZ94] D. J. Sturman and D. Zeltzer, A survey of glove-based input. IEEE Computer Graphics and Applications, Vol. 14, No. 1, 1994.

[TA88] A. S. Tanenbaum, Computer Networks, 3rd Edition, Englewood Cliffs, NJ: Prentice-Hall, 1988.

[TG96] T. Tonnenhofer and E. Gröller, Autostereogramms- Classification and Experimental Investigations in Proceedings of 12. Spring Conference on Computer Graphics, 1996.

[TJ95] M. Tidwell, R. S. Johnston, D. Melville, and T. A. Furness III, The Virtual Retinal Display - A Retinal Scanning Imaging System. In Proceedings of Virtual Reality World '95, 325-333, 1995.

[TJ97] J. Tesch, Eine dreidimensionale Benutzerschnittstelle für Virtuelle Umgebungen, Diplomarbeit Eberhard-Karls-Universität Tübingen, Fachbereich Informatik Fachgebiet Graphisch-Interaktive Systeme, 1997.

[VP98] E. Viirre, H. Pryor, S. Nagata, and T. A. Furness III, The Virtual Retinal Display, A New Technology for Virtual Reality and Augmented Vision in Medicine. In D. Stredney, S.J. Weghorst (Ed.) Proceedings of Medicine Meets Virtual Reality, 252-257, 1998.

[TN96] N. Thompson, 3D Graphics Programming for Windows 95, Microsoft Press, ISBN 1-57231-345-5, 1996.

[VC96] J. Viega, M. J. Conway, G. Williams, and R. Pausch, 3D magic lenses. In ACM UIST, 1996.

[VC97] VRML Consortium, Virtual Reality Modeling Language (VRML), ISO/IEC DIS 14772, http://www.vrml.org/Specifications/VRML97/DIS/part1/foreword.html, 1997.

[WG95] M. Wloka and E. Greenfield, The virtual tricorder: A uniform interface for virtual reality. In UIST95 Proceedings, 39-40,1995.

[WJ94] J. Wernecke, The Inventor Mentor Programming Object-oriented 3D Graphics with Open Inventor, Release 2, ISBN 0-201-62495-8, 1994.

[WO90] C. Ware and S. Osborne, Exploration and Virtual Camera Control in Virtual Three Dimensional Environments, In Proceedings of the 1990 Symposium on Interactive 3D Graphics, Special Issue of Computer Graphics, Vol. 24, No.2, 175-183, 1990.

[YJ96] C. Youngblut, R. E. Johnson, S. H. Nash, R. A. Wienclaw, and C. A. Will, Review of Virtual Environment Interface Technology, Institute for Defense Analyses –IDA, IDA Paper P-3186, WWW: http://www.hitl.washington.edu/scivw/IDA/, 1996.

[ZL87] T. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. ACM Conference on Human Factors in Computing Systems and Graphics Interfaces, 189-192, 1987.

Lebenslauf

von

Jürgen Fechter, geboren am 21. Februar 1962 in Sigmaringen

07.07.1999	Dissertation/Rigorosum
22.06.1999	Geburt des Sohnes Merlin
20.02.1999	Heirat mit Brigitte Abrell
April 1993-1999	Firma Ott&Adis, Rottenburg/Neckar
	Angestellter: Projektleiter und Systemmanager
Oktober 1990- März 1993	Firma Iltis, Rottenburg/Neckar
	Angestellter: Leiter Basistechnologie
August 1990	Diplom
1983 – 1990	Biologiestudium an der Eberhard-Karls-Universität Tübingen
Oktober 1982 – 1983	Zivildienst Ergotherapie Kurklinik Aulendorf
Juli – September 1982	Grundwehrdienst Murnau
1979 – 1982	Wirtschaftsgymnasium Saulgau
	Fachgebundene Hochschulreife
1974 – 1979	Realschule Ostrach
1972 – 1974	Hauptschule Ostrach
1968 – 1972	Grundschule Ostrach

Tübingen, den 7. Juli 1999