Für meine Eltern

## Danksagung

Ich danke Herrn Prof. Dr. J. Encarnação dafür, dass er mich dabei unterstützt hat als FH-Absolvent an der TU Darmstadt zu promovieren, für seine Ratschläge während der Erstellung der Arbeit und die Übernahme des Referates.

Weiterhin danke ich Herrn Prof.Dr. Stefan Müller für die Übernahme des Koreferates und die vielen fruchtvollen und motivierenden Diskussionen.

Ein ganz spezieller Dank geht an die aktuellen und ehemaligen Mitarbeiter der Abteilung Virtuelle und Erweiterte Realität. Ein großer Dank gilt Gerrit Voss für seine Unterstützung. Weiterhin auch meine Diplomanten Steffen Haupt und Jochen Ehnes ein Danke für ihre sehr gute Arbeit.

Ein ganz herzlicher Dank geht an meine Ex-Kollegin Mimi Lux, ohne die es sehr wahrscheinlich keine Null-Version gegeben hätte.

Da das Schreiben einer Dissertation zum größten Teil nicht während der normalen Arbeitszeit möglich ist, gilt auch der Dank meinen Freunden für ihr Verständnis der Prioritäten.

Besonderen Dank auch meinen Eltern, die mich in den letzten Jahren immer tatkräftig unterstützt und motiviert haben.

Christian Knöpfle, Frankfurt, Dezember 2003

# 1 Einleitung

## 1.1 Motivation

Virtuelle Realität – Eine Technologie, die sich gerade in den letzten Jahren sehr stark entwickelte und heute in vielen Anwendungsbereichen anzutreffen ist. Spezielle Ein- und Ausgabegeräte, Echtzeitgraphik und immersive Anwendungen prägen das heutige Bild der virtuellen Realität (VR). Aber Virtuelle Realität ist vor allem eins: Eine neuartige Mensch-Maschine Schnittstelle, die sich zum Ziel gesetzt hat, das Interagieren mit dem Virtuellen natürlich, intuitiv und interaktiv zu gestalten.

Trotz oder vor allem wegen dieses Anspruches, stellen sich bei jedem Entwurf einer neuen VR-basierten Anwendung dieselben Fragestellungen: Zum einen das Herausarbeiten und Umsetzen der Funktionalitäten, die diese neue Anwendung abdecken muss. Hier kann die Forschung vor allem dabei helfen, dem Anspruch der Echtzeitanwendung gerecht zu werden. Zum anderen das Design und die Umsetzung der Interaktionsschnittstelle, denn diese ist essenziell für die gesamte Benutzbarkeit des Systems und hat somit vor allem auch große Auswirkung auf die Akzeptanz seitens der Anwender. Hier mangelt es jedoch an zielgerichteten Vorgehensweisen und definierten Basiskonzepten, die den Schritt des Erarbeitens der Bedienoberfläche wesentlich vereinfachen würden. In den letzten Jahren wurden zwar immer wieder Ansätze und Lösungen entwickelt, die den Anspruch an VR basierte Benutzerschnittstellen teilweise erfüllten, jedoch die fehlende Möglichkeit, diese in ein vorgegebenes Schema oder einen globalen Kontext einzuordnen, schränkte deren Wiederverwendbarkeit sehr stark ein.

So ist man heute in den meisten Fällen dazu gezwungen, diese Benutzerschnittstellen vollständig neu zu entwickeln und über mehrere Iterationen das optimale Design herauszuarbeiten. Vergleicht man dies mit der Welt der Desktop-Anwendungen, in der Richtlinien (Styleguide) vorhanden und Softwaretoolkits (GUI-Bibliotheken) verfügbar sind, so ist der Aufwand bei VR wesentlich größer. Es darf dabei aber nicht verschwiegen werden, dass die Komplexität des Unterfangens intuitive, natürliche und immersive Benutzerschnittstellen zu entwickeln, wesentlich höher ist. Dies liegt vor allem in der Vielzahl zusätzlicher und neuer zu beachtender Parameter begründet. Dabei kommt hinzu, dass diese teilweise voneinander abhängen und sich gegenseitig beeinflussen. Jedoch, was sind das genau für Parameter? Welchen Einfluß und Stellenwert haben sie bei der Entwicklung? Wie sehen Softwarebasiskonzepte aus, die eine möglichst hohe Flexibilität und Anpassbarkeit bieten? Fragestellungen, die heute schwer zu beantworten sind.

Am Beispiel des VR-Design Reviews können viele dieser Fragestellungen untersucht und illustriert werden, denn die dort gestellten Anforderungen sind insbesondere in Bezug auf die Interaktion stellvertretend für eine Vielzahl von industriell geprägten Anwendungsklassen. Hervorgegangen aus dem klassischen Design Review, bei dem physikalische Prototypen (*PMU*) als Basis der Begutachtung des Entwicklungsstandes eingesetzt wurden, baut der VR-Design Review vollständig auf digitalen Prototypen (*DMU*) auf. Diese Umstellung wurde vor allem durch Forderungen nach Kostensenkung, kürzeren Innovationszyklen und der Verfügbarkeit digitaler Modelle vorangetrieben. Vor allem für die Anwender bedeutet dies einen Paradigmenwechsel: Das natürliche, intuitive Interagieren mit einem PMU wird durch ein Interagieren mit einem digitalen Modell ersetzt. Zudem arbeiten in Design Reviews immer mehrere Personen zusammen. Auch dies muss bei der Umstellung auf DMUs be-

dacht werden. Die logische Konsequenz: Der Einsatz von Virtueller Realität als Technologie, um das Erleben des Prototypen für die Anwendung ausreichend realistisch gestalten zu können und vor allem als eine Mensch-Maschine-Schnittstelle, die potentiell die Möglichkeit bietet, alleine oder in der Gruppe weiterhin natürlich und intuitiv zu interagieren.

Diese Entwicklung hin zu einer vollständigen Digitalisierung des Entwicklungsprozesses ist symptomatisch für viele Sparten, in denen in immer kürzerer Zeit neue Produkte auf den Markt gebracht werden müssen, um konkurrenzfähig zu bleiben. So schließt sich hier direkt die Frage an, wie diese entwickelten Konzepte in einen globalen Kontext eingeordnet werden können, um dadurch eine hohe Wiederverwendbarkeit auch für andere Bereiche zu erzielen.

Allgemeingültige und definierte Vorgehensweisen für den Entwurf von Benutzerschnittstellen sowie ein hohes Maß an Wiederverwendbarkeit: Der Desktopbereich hat gezeigt, dass dies kein Traum bleiben muss.

## 1.2 Problemstellung und Zielsetzung

Wie im vorherigen Kapitel dargelegt, ist die Entwicklung neuer VR-Anwendungen insbesondere dadurch geprägt, dass ein großer Teil der Entwicklungsleistung in das Design und die Umsetzung der Benutzerschnittstelle, sowie die Realisierung geeigneter Softwarebasistechnologien investiert werden muss. Dies läßt sich vor allem darauf zurückführen, dass keine definierte Vorgehensweise für die Entwicklung intuitiver und immersiver Benutzerschnittstellen existieren.

Ziel dieser Arbeit ist es, eine ganzheitliche Vorgehensweise zu entwickeln, die es erlaubt, die für das Design der Benutzerschnittstelle entscheidenden Variablen, Parameter und Randbedingungen zu identifizieren und zu klassifizieren. Auf Basis der Klassifikation soll dann ein Leitfaden für die Entwicklung der VR-Benutzerschnittstelle abgeleitet und die für die Anwendung passenden Interaktionstechniken bestimmt werden können.

Um auch eine Wiederverwendbarkeit entwickelter Komponenten zu erreichen, sowie seitens einer Anwendung flexibel auf Veränderungen der die Benutzeroberfläche betreffenden Randbedingungen reagieren zu können, sind geeignete Softwarebasiskonzepte notwendig. Um dem Anspruch des ganzheitlichen Ansatz gerecht zu werden, sind auch diese Aspekte Teil der Zieldefinition.

VR-Design Review in der Automobilindustrie wird hierbei als Stellvertreter einer Anwendungsklasse aufgefasst, die in der Industrie sehr weit verbreitet ist. Dadurch ist gewährleistet, dass die erarbeiteten Lösungsansätze auf breiter Basis eingesetzt werden können. Weiterhin adressiert VR-Design Review auch funktionale und inhaltliche Fragestellungen unter Randbedingungen, die bis heute in der Forschung wenig betrachtet wurden.

Prinzipiell sind hierbei drei Schwerpunkte zu untersuchen:

- *Benutzerschnittstelle:* Aus Sicht der Mensch-Maschine-Schnittstelle muss das Softwaresystem für den Anwender ein Werkzeug sein, welches sich auf intuitive Weise bedienen lässt und er somit effektiv seiner Arbeit nachgehen kann.
- *Funktionalität:* Aus funktionaler Sicht liegen die Schwerpunkte in der Untersuchung des digitalen Prototypen, der Dokumentation und der Integration und Evaluierung von Offline-Simulatio-

nen.
- *Kooperation:* Ein wichtiger Aspekt ist die aktive Unterstützung des kooperativen Arbeitens mehrerer Anwender, die sich an einem Ort befinden.

## 1.3 Gliederung der Arbeit

Im folgenden Kapitel wird zu Beginn das 4W-Modell in der Ausprägung für VR-Benutzerschnittstellen vorgestellt. Mit Hilfe dieses Modells können die essenziellen Randbedingungen für den Entwurf herausgearbeitet und klassifiziert werden. Ein Abhängigkeitsmodell zeigt hierbei zusätzlich, wie sich verschiedene Parameter gegenseitig beeinflussen. Im Anschluss wird das Anwendungsbeispiel Design-Review vorgestellt und mit Hilfe der Schemata untersucht. Eine Generalisierung der herausgearbeiteten Anforderungen bildet den Abschluß des Kapitels und die Basis für die Arbeiten in Kapitel 4.

In Kapitel 3 werden die Grundlagen der immersiven und intuitiven Interaktion dargelegt. Zuerst wird hierbei die Technologie „Virtuelle Realität“ kurz beleuchtet. Da Geräte in der Virtuellen Realität eine sehr große und wichtige Rolle spielen, werden hier auch relevante Aspekte der menschlichen Physiognomie vorgestellt. Weiterhin wird das menschliche (Inter)agieren allgemein betrachtet. Dies umfasst die Themenschwerpunkte menschliche Wahrnehmung, generelle Aktionsabläufe und Kommunikation. Den Abschluß bildet die Vorstellung des Konzeptes der logischen Interaktionsaufgaben.

Basierend auf den in Kapitel 2 erarbeiteten Anforderungen, sowie den Grundlagen aus Kapitel 3 wird nun in Kapitel 4 eine Benutzerschnittstelle für das Gesamtsystem entwickelt, welche auf dem Konzept der *basic interaction tasks* (logische Interaktionsaufgaben) aufsetzt. Interaktionstechniken aus der Literatur werden vorgestellt und mit Techniken, die im Rahmen dieser Arbeit entwickelt wurden, verglichen und bewertet.

In Kapitel 2 wurde insbesondere das kooperative Arbeiten und die Bedeutung der Prototypen als Mediatoren als sehr wichtige Bestandteile des Design Reviews herausgearbeitet. Auf diese Schwerpunkte wird in Kapitel 5 näher eingegangen und dabei das *Papier und Bleistift* Konzept vorgestellt, sowie die Möglichkeiten des *sketchings* präsentiert.

In Kapitel 6 wird die funktionale Seite betrachtet. Hierbei wird insbesondere auf die Untersuchung des digitalen Prototypen, sowie die Möglichkeit der Dokumentation der Ergebnisse eines Design Reviews und die Interaktion mit Simulationsdaten eingegangen. Ein weiterer Schwerpunkt dieses Kapitels sind spezielle Renderingtechniken die aufgrund von Latenzverkürzung die Interaktion verbessern. Neben der funktionalen Sicht zeigt dieses Kapitel vor allem, wie die in Kapitel 4 herausgearbeiteten Interaktionstechniken effektiv eingesetzt werden können.

Schwerpunkt von Kapitel 7 ist die softwareseitige Wiederverwendbarkeit der erarbeiteten Lösungen. Als Basis dient der Standard VRML97 und dessen Weiterentwicklung X3D, die beide kurz vorgestellt werden. Im Anschluß wird eine im Rahmen dieser Arbeit entwickelte Schnittstellendefinition für die einzelnen basic interaction tasks vorgestellt. Die Vorteile der abstrakten Schnittstellendefinition und den damit verbundenen Möglichkeiten werden ebenfalls beleuchtet.

Den Abschluß bildet Kapitel 8, welches die Ergebnisse dieser Arbeit zusammenfaßt und einen Aus-

blick für weitere Arbeiten gibt.

## 1.4 Zusammenfassung der wichtigsten Ergebnisse

Im Rahmen dieser Arbeit wird die Thematik der intuitiven und immersiven Interaktion ganzheitlich betrachtet, sowie die erarbeiteten Lösungen am Beispiel des VR-Design Reviews evaluiert. Im einzelnen wurden die folgenden Ergebnisse erzielt:

*Entwicklung einer Vorgehensweise für den Entwurf intuitiver, immersiver Benutzerschnittstellen.*

Es wurde mit dem 4W-Modell eine Vorgehensweise dargelegt, die es erlaubt, die für den zielgerichteten Entwurf einer VR-basierten Benutzschnittstelle essenzielle und entscheidende Randbedingungen und Parameter herauszuarbeiten und zu klassifizieren. Weiterhin wurde das Abhängigkeitsmodell entwickelt, welches die gegenseitigen Abhängigkeiten und Beeinflussungen zwischen Anwendung, Anwender, Hardware und Software definiert und als ein weiterer Leitfaden für den Entwurf intuitiver, immersiver Benutzerschnittstellen verwendet werden kann. Anhand des Beispiels VR-Design Review wurde die Anwendbarkeit dieser Modelle gezeigt und die daraus erwachsenden Möglichkeiten und Potenziale dargelegt.

*Evaluierung, Konzeption und Umsetzung von Interaktionstechniken für die anvisierte Anwendungsklasse auf Basis der logischen Interaktionsaufgaben, sowie Betrachtung der funktionalen Seite.*

Im Rahmen dieser Arbeit wurde das Konzept der basic interaction tasks (logische Interaktionsaufgaben) für die virtuelle Realität adaptiert und als Grundlage für die immersive und intuitive Interaktion verwendet. Unter Zugrundelegung dieses Konzeptes und den für die Anwendungsklasse VR-Design Review erarbeiteten Anforderungen wurden aus der Literatur bekannte Interaktionstechniken vorgestellt und evaluiert. Da nicht für alle Ausprägungen der logischen Interaktionsaufgaben Interaktionstechniken existieren, die den Anforderungen gerecht werden, wurden im Rahmen dieser Arbeit weitere Techniken entwickelt. Dies sind u.a. der world point grab zum Positionieren von Objekten und für das Quantifizieren der Jog-Dial und die interaktive Säule.

Dreh und Angelpunkt einer Benutzerschnittstelle ist das Menüsystem, über das der Anwender Zugriff auf die verschiedenen Funktionen des Systems erhält. In dieser Arbeit wurde ein leistungsfähiges Menüsystem entwickelt, welches selbst sehr komplexen Systemen mit vielen Funktionalitäten gerecht wird. Hierbei wurden Pie-Menüs verwendet, die sich im Vergleich zu anderen graphischen Repräsentation sehr schnell und einfach bedienen lassen.

Aus den funktionalen Anforderungen an ein System für den Design Review digitaler Prototypen können drei Kerngebiete abgeleitet werden. Die Modelluntersuchung, die Dokumentation der Ergebnisse des Reviews und die Untersuchung von Simulationsergebnissen. Für alle diese Kerngebiete wurden Funktionen entwickelt, um den Anwender optimal zu unterstützen. Hierbei wurde auch gezeigt, wie die erarbeiteten Interaktionstechniken effektiv eingesetzt werden können.

Zusätzlich wurde der Outline-Algorithmus entwickelt, welcher aus einer bestehenden Geometrie eine Pseudo-Umrißdarstellung generiert. Diese kann dann erheblich schneller als das Original-

modell dargestellt werden. Über das Konzept der adaptiven Darstellung kann dann aufgrund kürzerer Latenzen und Reaktionszeiten die Interaktion mit dem System verbessert werden.

*Entwicklung eines leistungsfähigen Konzeptes, welches die Kommunikation und Kooperation zwischen verschiedenen Teilnehmern effektiv unterstützt und auch das natürliche und intuitive Annotieren und Zeichnen auf digitalen Daten ermöglicht.*

Im Rahmen dieser Arbeit wurde das *Papier und Bleistift* Paradigma entwickelt, welches auf Basis der sog. *sketch-objects* jedem Anwender die Möglichkeit eröffnet, für sich Ideen zu skizzieren (private space) und diese dann den anderen Teilnehmern zu präsentieren. Mit Hilfe des MultiViews-Algorithmus ist es dabei möglich, dass selbst bei Verwendung eines einzigen Ausgabegerätes jeder Anwender sein Sketch-Objekt perspektivisch korrekt und stereoskopisch betrachten kann. Weiterhin wurde eine neue Methodik des Skizzierens entwickelt, welche erlaubt, bestehende 3D-Modelle zu annotieren oder z.B. auch Vorschläge zur Modelländerung darzulegen. Aufgrund dieser Konzepte erlangt der digitale Prototyp auch als Kommunikationsmedium eine neue Qualität, die sich positiv auf die Kommunikation und Kooperation auswirkt.

*Konzeption einer abstrakten Softwareschnittstelle für logische Interaktionsaufgaben auf Basis eines Standards.*

Im Rahmen dieser Arbeit wurden die logischen Interaktionsaufgaben nicht nur zur Strukturierung und Klassifizierung von Interaktionstechniken verwendet, sondern auch als abstrakte Softwareschnittstelle eingesetzt. Es wurde auf Basis der allgemeinen BIT-Anforderungen für jede logische Interaktionsaufgabe eine abstrakte Softwareschnittstelle entwickelt. Diese basiert in dieser Arbeit auf dem Standard VRML97, kann jedoch an andere Basistechnologien sehr leicht angepasst werden. Auf dieser Schnittstelle können dann seitens eines Anwendungsentwicklers beliebige Interaktionstechniken aufgesetzt werden. Weiterhin ist diese Schnittstelle so konzipiert, dass das Austauschen und Kombinieren von Interaktionstechniken sehr einfach möglich und für eine VR-Anwendung transparent ist. Diese Flexibilität wird durch eine Abstraktionsebene bzgl. der Hardwareschnittstelle zusätzlich erhöht. Das ebenfalls entwickelte Konzept des BIT-Managers illustriert das Potenzial und die Möglichkeiten, die sich aus dieser abstrakten Schnittstelle ergeben.

Die hier vorgestellten vier Hauptschwerpunkte der Arbeit und die damit verbundenen Ergebnisse decken die entscheidenden Aspekte des Entwicklungsprozesses ab. Diese Arbeit bildet somit eine solide Basis für den Entwurf und die Realisierung immersiver und intuitiver Benutzerschnittstellen für virtuelle Umgebungen – nicht nur für den VR-Design Review.

# 2 Vorgehensweise, Klassifizierung und Anwendungsklasse

Die in der Einleitung motivierte Forderung nach einer fundierten Vorgehensweise beim Entwurf von VR basierten Benutzerschnittstellen wird in diesem Kapitel aufgegriffen und behandelt.

Die hier entwickelte Vorgehensweise basiert auf zwei Modellen. Das 4W-Modell ist ein Leitfaden für das Herausarbeiten der essenziellen Randbedingungen für VR-Benutzerschnittstellen. Das Abhängigkeitsmodell zeigt die Wechselwirkungen der verschiedenen Parameter auf. Die Anwendung dieser Modelle wird anhand des VR-Design Review illustriert.

Hierzu wird in Kapitel 2.2 der Design Review näher beleuchtet. Hierbei wird insbesondere die Einordnung in die Prozesskette erläutert und stellvertretend für die verschiedenen Anwendungsbereiche drei Szenarien vorgestellt, aus denen allgemeine und anwendungsspezifische Anforderungen an ein Software-System zur Unterstützung des Design Reviews abgeleitet werden können (siehe Kapitel 2.3). Anhand dieser Szenarien werden im weiteren Verlauf der Arbeit die entwickelten Konzepte evaluiert. Es folgt ein kurzer Überblick über verschiedene Systeme, die heutzutage im Einsatz sind. Diese werden insbesondere auch im Hinblick auf die gestellten Anforderungen betrachtet.

Um der Gefahr zu entgehen, dass die hier entwickelten Konzepte nur für den Design Review sinnvoll einsetzbar gestaltet sind, werden in Kapitel 2.5 die speziellen Anforderungen an den Design Review auch auf einer abstrakten Ebene betrachtet. Dadurch können die Ansätze und Lösungen in einem globaleren Kontext eingeordnet werden und erlangen dadurch allgemeingültigeren Charakter.

## 2.1 Schemata zur Klassifizierung von VR Bedienoberflächen

Im Bereich der 2D-Benutzeroberflächen haben sich in den letzten Jahren Standards entwickelt, an denen sich Entwickler neuer Systeme orientieren können. Betrachtet man z.B. Programme für das Betriebssystem MS-Windows(tm) so fallen viele Gemeinsamkeiten in der Bedienung auf. Dies hat besonders für den Anwender den großen Vorteil, dass er aus dem Wissen über der Bedienung eines Programmes Rückschlüsse auf die Bedienung eines anderen Programms ziehen kann. So ist z.B. die Funktion „Einfügen“ im 2. Menü von links (Bearbeiten) zu finden. Für den Programmierer haben diese Standards den Vorteil, dass er auf gut funktionierende Bedienkonzepte aufsetzen kann, für die es auch Standardbibliotheken für Bedienelemente gibt, aus denen diese Oberflächen relativ einfach und schnell aufgebaut werden können.

Im Gegensatz dazu existiert heute für VR-Anwendungen noch kein „Baukasten“ aus dem eine Benutzerschnittstelle aufgebaut werden kann. Dies hat mehrere Gründe:

- Virtuelle Realität kennzeichnet neben verschiedenen anderen Ausprägungen vor allem eine neue *Mensch-Maschine Schnittstelle*, die sich insbesondere die *natürliche, intuitive Interaktion* zum Ziel gesetzt hat. Dies wird vor allem durch die Verwendung spezieller Hardware- und Softwaretechnologien erreicht. Natürlich und intuitiv bedeutet jedoch auch, dass das Wissen und die Erfahrungen des Anwenders eine sehr starke Rolle spielen und beim Design einer VR-Benutzerschnittstelle entsprechend beachtet werden müssen.
- VR stellt die unterschiedlichsten Ein- und Ausgabegeräte zur Verfügung. Welche Geräte zum Einsatz kommen, hängt unter anderem von der Anwendung und der Physiognomie der Benutzer

ab. Ist in einer Anwendung die Kommunikation zwischen verschiedenen Anwendern immanent wichtig, würde sich die Verwendung eines HMDs als sehr störend auswirken, da sich die Personen nicht mehr gegenseitig sehen könnten. In Bezug auf die Physiognomie ist es z.B. für einen Jetpiloten relativ unproblematisch, ein schweres HMD über eine lange Zeit zu tragen, wohingegen ein Ingenieur nach kurzer Zeit sicherlich Genickschmerzen bekommen und sich sehr unwohl fühlen würde.

- Unterschiedliche Geräte erlauben den Einsatz unterschiedlicher Interaktionstechniken. Wird z.B. ein Datenhandschuh verwendet, so können Gesten für die Kommunikation mit dem Rechner verwendet werden. Wird ein einfacher Stylus oder Flystick eingesetzt, so können nur Tasten gedrückt werden.
- Die zu verwendenden Interaktionstechniken werden jedoch auch durch die geforderte Funktionalität der Software beeinflusst. So müssen zur Navigation in weitläufigen Szenen andere Methoden eingesetzt werden, als in Szenen, in denen der Anwender alle Objekte in Reichweite hat.

Man mag bereits an dieser kurzen Aufstellung von Faktoren und deren Abhängigkeiten erkennen, dass es für VR-Bedienoberflächen ein „one size [concept] fits all“-Konzept nicht geben kann.

Um nun nicht ausschließlich Speziallösungen für jede VR basierte Benutzerschnittstelle zu entwikkeln, wurden im Rahmen dieser Arbeit Modelle entwickelt, die es erlauben, Problemstellungen zu klassifizieren, die für den Entwurf der Schnittstelle entscheidenden Randbedingungen herausarbeiten, sowie deren Abhängigkeiten zu illustrieren. Es handelt sich dabei um

- das 4W-Modell
- das Abhängigkeiten-Modell.

Diese zwei Modelle werden in den beiden folgenden Kapiteln näher beschrieben.

### 2.1.1 Das 4W-Modell

Mit Hilfe des 4W-Modells können aus einer konkreten Problemstellung die für diesen Anwendungsfall relevanten Parameter extrahiert und als Anforderung definiert werden. Es stellt somit ein Klassifizierungsschema für VR-Benutzeroberflächen dar.

Das 4W-Modell baut auf vier Fragestellungen auf:

- Wer nutz das System?
- Wozu wird das System verwendet?
- Was wird inhaltlich dargestellt?
- Wie wird das System verwendet?

Jede dieser Fragestellungen beinhaltet verschiedene Aspekte, unter denen sie betrachtet werden muss. Ziel ist es, über die verschiedenen Ausprägungen (Antworten) die „passenden“ Interaktionstechniken herauszuarbeiten und die Benutzeroberfläche zu realisieren. Diese Herangehensweise zur Klassifizierung wurde z.B. auch für den Zugang zu ökonomischen Daten ([LUX00]) erfolgreich angewendet.

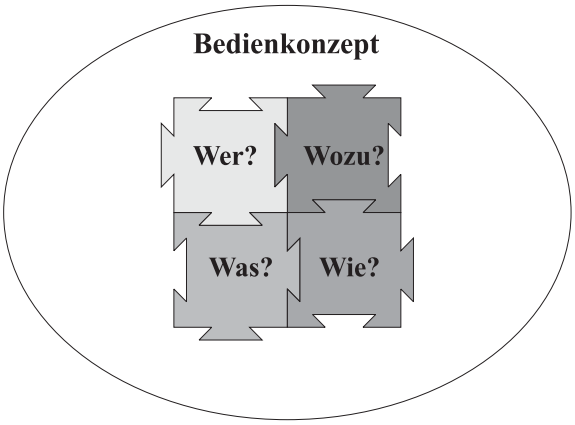


Abb. 2.1: Das 4W-Modell für VR-Benutzeroberflächen

Im folgenden werden die einzelnen Fragestellungen und die wichtigsten Faktoren näher beleuchtet. Es wird gezeigt, dass diese vier Fragestellungen auch untereinander Abhängigkeiten aufweisen.

#### 2.1.1.1 Wer? - Anwender

Diese Frage richtet sich insbesondere an den Anwendertypus des Systems, wobei verschiedene Aspekte wichtig sind.

*Erfahrungen,* die Anwender bereits mitbringen. So spielt es z.B. eine große Rolle, ob die Anwender aus einem CAD- oder VR-Umfeld kommen. In beiden Bereichen wird sehr oft das Eingabegerät Spacemouse eingesetzt. In der VR wird jedoch die Spacemouse zum Navigieren durch die Szene (Bewegen der Kamera) eingesetzt, im CAD zum positionieren der Szene (Bewegen der virtuellen Objekte). Erfahrungen und Gedankenmodelle (Verstehen einer Funktion) sind hier verschieden und es ist für beide Personengruppen sehr schwierig, eine jeweils andere Steuerung zu verstehen und intuitiv zu benutzen. Erfahrung ist somit nicht nur als eine Frage nach VR-spezifischer Erfahrung zu verstehen, sondern ganzheitlich zu betrachten.

Es hat sich auch gezeigt, dass die *organisatorische Stellung* der Anwender ebenfalls eine wichtige Rolle spielen kann. So fürchten Menschen denen die gesellschaftliche Stellung wichtig ist (z.B. Entscheidungsträger, High-Level Management), dass sie sich aufgrund des Nichtverstehens der Bedienung lächerlich machen könnten und das System deswegen nicht verwenden. Hier sollte die Bedienschnittstelle trivial sein, keine komplexen Gedankenmodelle voraussetzen und dies auch sichtbar kommunizieren (z.B. durch geeigneten Hardwareaufbau). Hierbei ist auch zu hinterfragen, wieviel Zeit für das Erlernen der Bedienung zur Verfügung steht (*Lernzeit*).

Wie bereits oben erwähnt, spielt bei der Umsetzung der Benutzerschnittstelle auch die *Physiognomie* eine wichtige Rolle, wobei hier in den meisten Fällen von „normalen“ Benutzern , ohne spezielle Fähigkeiten (z.B. Jetpilot, Hochleistungssportler) ausgegangen werden kann.

#### 2.1.1.2 Wozu? - Intention

Die Frage „Wozu“ mag im ersten Moment dazu verleiten, das anvisierte Einsatzgebiet genauer zu untersuchen. Diese wäre jedoch für das weitere Vorgehen nicht zielführend[1]. Mit dieser Fragestellung

soll in erster Linie die Intention des Anwenders hinterfragt werden, d.h. ob er mit dem System spielen, forschen, erkunden, lernen oder arbeiten möchte, wobei es natürlich auch Mischformen dieser Attribute gibt (z.B. Edutainment: spielen und lernen). Eine einfache Charakterisierung über eines dieser Attribute ist jedoch nicht ausreichend. Hier steht insbesondere auch die „Wie?“ Frage in engem Zusammenhang. So ist es z.B. beim „Arbeiten mit dem System“ wichtig zu wissen, ob die Anwender sich virtuelle Szenen nur anschauen und „aus dem Bauch“ Entscheidungen treffen oder sehr viele Funktionen zum Untersuchen der virtuellen Objekte benötigen, um Entscheidungen auch technisch abzusichern. In letzterem Fall würde dies z.B. bedeuten, dass das System über ein leistungsfähiges Menü verfügen muss, um dem Anwender alle notwendigen Funktionen anbieten zu können.

#### 2.1.1.3 Wie? - Abläufe

Insbesondere bei Arbeitsprozessen, die heute auf virtuelle Realität umgestellt werden, haben sich über Jahre hinweg Arbeitsabläufe entwickelt, die sich sehr bewährt haben. Hier sollte untersucht werden, wie diese auch in der VR-Welt umgesetzt werden können (Transfer). Weiterhin kann diese Fragestellung auch als Konkretisierung der Intention (s.o) verstanden werden.

Ein wichtiger Aspekt ist, ob das System von einem Anwender oder von mehreren Anwendern gleichzeitig verwendet wird. Gleichzeitig kann hierbei im Sinne einer verteilten (mehrere Anwender an verschiedenen Orten) oder einer kooperativen Nutzung (mehrere Anwender an einem Ort) unterschieden werden. Insbesondere bei mehreren Anwendern spielen Kommunikation und Mediatoren eine wichtige Rolle. Mediatoren sind Medien, die die Kommunikation zwischen den Anwendern effektiv unterstützen.

Die zur Verfügung stehende *Hardware* und die zu realisierenden Abläufe stehen ebenfalls in einem engen Zusammenhang. Hier müssen Anforderung an die Präsentations- und Eingabekomponente herausgearbeitet werden. Wichtig ist auch die Einschätzung des zeitlichen Rahmens, in dem das System eingesetzt wird (Häufigkeit, Dauer). Dies kann u.a. Rückschlüsse auf die körperliche Belastung der Systembenutzer zulassen.

#### 2.1.1.4 Was? - Inhalte

Für die Auswahl der Interaktionstechniken sind die Randbedingung einer typischen virtuellen Szene aus dem anvisierten Anwendungsgebiet von großer Bedeutung, u.a. Ausdehnung der Szenerie, Anzahl und Größe der einzelnen Objekte und Datentypen (Oberflächen, Volumen, Simulationsdaten).

### 2.1.2 Ausprägungen und Aspekte des 4W-Modells

Die vier W-Fragen bilden die oberste und abstrakteste Ebene, auf der die Untersuchung einer Anwendung bzgl. der Benutzerschnittstelle beginnt. Von hier aus ist es dann immanent wichtig, die richtigen Aspekte der Themenfelder Anwender, Intention, Ablauf und Inhalt zu betrachten und die damit verbundenen Auswirkungen zu erarbeiten. Die Hauptkriterien sind in der folgenden Tabelle zusammengefasst. Beispielhaft sind zusätzlich einige der Auswirkungen genannt.

1. Eine genaue Untersuchung des Einsatzgebietes ist dann wichtig, wenn es um die einzelnen Funktionen des Systems geht. Dies ist jedoch erst zu einem wesentlich späteren Zeitpunkt von Bedeutung

| | Kriterien | Auswirkungen[a] (techn., konzeptionell) |
|---|---|---|
| **Wer? Anwender** | • Erfahrung<br>• Gedankenmodelle<br>• Gesellschaftliche Position<br>• Physiognomie<br>• Lernzeit | • Systemmodell<br>• Hardware<br>• Systemkontrolle |
| **Wozu? Intention** | • Intention<br>• Spielen, Erkunden, Erforschen<br>• Lernen<br>• Arbeiten (technologisch, „aus dem Bauch“) | • Funktionsumfänge<br>• Hardware |
| **Wie? Ablauf** | • Transfer<br>• Einzelner Anwender<br>• Mehrere Anwender (kooperativ, verteilt) | • Aktionsmodell<br>• Kommunikation/Mediatoren<br>• Hardware<br>• Anzahl Zeiger<br>• Nachverfolgung<br>• Gleichzeitige Ansichten<br>• Position/Orientierung/Navigation |
| **Was? Inhalt** | • Szenenkonfiguration<br>• Datentypen (Oberfläche, Volumen) | • Bewegungsmodi |

a. Diese Tabelle umfaßt nur beispielhafte Auswirkungen und nicht alle möglichen

Tabelle 2.1: Aspekte des 4W-Modells

### 2.1.3 Das Abhängigkeiten-Modell

In den Darstellungen der W-Fragen hat sich bereits gezeigt, dass bei den verschiedenen Ausprägungen (z.B. Hardware ist abhängig von Wer, Wie, Wozu) Abhängigkeiten bestehen. Diese verschiedenen Abhängigkeiten werden im Abhängigkeitsmodell für VR-Benutzeroberflächen (siehe Abb. 2.2) dargestellt. Dieses Modell gibt vor allem vor, in welcher Reihenfolge Abhängigkeiten am günstigsten gelöst werden sollten.

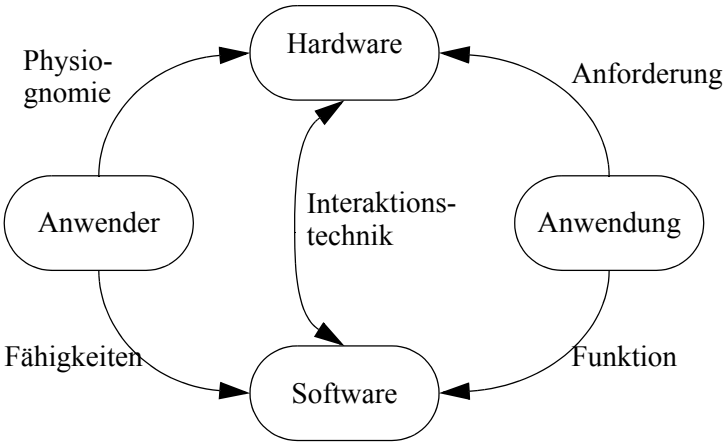


Abb. 2.2: Anforderungsmodell für VR-Benutzeroberflächen

So sollten ausgehend von der Anwendung Fragen nach den Anforderungen an die verwendete Hardware und Software gestellt werden. Dies umfaßt z.B. von seiten der Hardware die Art der Projektion und von seiten der Software den Funktionsumfang (*Intention*). Ausgehend vom Anwender muss dessen Physiognomie untersucht und als Anforderungen an die Hardware gestellt werden. Auf der anderen Seite wird die Software auch über das Wissen des Anwenders und die menschliche Aktionsweise beeinflußt.

Das Modell zeigt jedoch auch, dass die VR-Interaktionstechniken, also die Basis jeder Bedienoberfläche, sowohl von der Hardware, als auch der Software abhängen. Im Bereich der 2D-GUIs ist dies zwar prinzipiell auch der Fall, hier hat sich jedoch in den letzten Jahren mit Monitor, Maus und Tastatur ein Standard definiert und somit ist dieser Parameter fix. Aufgrund der unterschiedlichsten Hardwaremöglichkeiten, die die Virtuellen Realität bietet, ist dieser Parameter hier äußerst variabel. Es hat sich gezeigt, dass durch Festlegung (unter Berücksichtigung der Anforderungen) dieses Parameters auf bestimmte I/O-Gerätetypen die Entwicklung intuitiver Schnittstellen wesentlich vereinfacht, wenn nicht gar erst ermöglicht wird.

## 2.2 Anwendungstypus Design Review

In diesem Kapitel wird der Prozess des Design Reviews eingehender beschrieben. Zu Beginn wird das allgemeine Vorgehen und der Stellenwert im Gesamtentwicklungsprozess dargelegt. Im Anschluß werden die Begriffe *physikalischer Prototyp* und *digitaler Prototyp* definiert und ihre jeweiligen Vor- und Nachteile dargelegt. Da sie die Grundlage der Diskussionen während eines Design Reviews bilden, spielen sie eine wichtige Rolle bei der Kommunikation der Teilnehmern. Den Abschluß bildet die Vorstellung einiger typischer Anwendungsszenarien aus dem Automobilbau, in denen Design Reviews durchgeführt werden.

### 2.2.1 Allgemeines Verfahren

Der Entwicklungsprozess komplexer Produkte ist heutzutage sehr vielschichtig und durchläuft mehrere Phasen, beginnend bei der eigentlichen Idee, über die Erstellung der Anforderungen, Planung und Konstruktion bis hin zur Produktion. Im folgenden wird nun insbesondere auf die Phase der Pla-

nung und Konstruktion eingegangen, wobei hier speziell der Automobilsektor betrachtet wird. Das beschriebene Vorgehen ist jedoch im wesentlichen auch auf andere Industrien übertragbar.

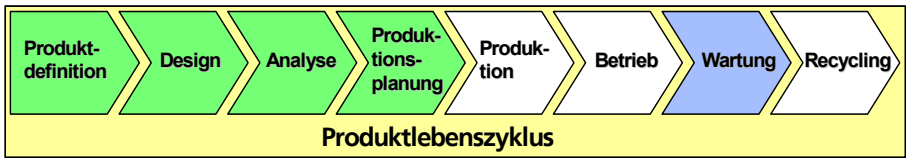


Abb. 2.3: Produktentstehungs-/lebenszyklus

Ein Gesamtfahrzeug besteht aus vielen verschiedenen Komponenten (Baugruppen), die von verschiedenen Entwicklungsteams eigenständig entwickelt werden. Hierbei werden bestimmte Anforderungen an die Komponente gestellt, die zum einen die Funktionsweise und zum anderen die Integration in das Gesamtfahrzeug sicherstellen.

Jedes Entwicklungsteam plant und konstruiert mit Hilfe eines CAD-Programms die jeweilige Komponente. Damit auch sichergestellt ist, dass die gestellten Anforderungen eingehalten werden, muss die Komponente nach bestimmten Gesichtspunkten evaluiert werden, die abhängig vom jeweiligen Anwendungsbereich sind. Diese Evaluierung, auch *Design Review* genannt, wird nicht nur am Ende einer Entwicklung durchgeführt, sondern in regelmäßigen Abständen, um zum einen den Entwicklungsstand zu überprüfen und zum anderen Fehler und Mängel in der Konstruktion so früh wie möglich zu erkennen, denn je später sie festgestellt werden, desto aufwendiger und kostenintensiver wird ihre Beseitigung.

Zu einem Design Review treffen sich mehrere Experten, um einen Planungszustand zu besprechen und zu prüfen. Wichtig hierbei ist insbesondere die Kommunikation der Experten untereinander, da viele Mängel oder Verbesserungen nur durch die Diskussion gefunden werden. Sowohl festgestellte Mängel als auch Verbesserungsvorschläge werden in einem Protokoll festgehalten, welches dem Konstrukteur als Vorlage für die notwendigen Änderungen der Konstruktion dient. Diese Fehler müssen zuerst behoben werden, bevor die Entwicklung weiter vorangetrieben werden kann. Hier ist es sehr oft üblich, dass auch nach der Beseitigung der gefundenen Mängel ein erneuter Review durchgeführt wird (*Wiedervorlage*).

### 2.2.2 Physikalische versus Digitale Prototypen

Das zu begutachtende Bauteil kann nun in zwei unterschiedlichen Ausprägungen zur Verfügung stehen: Entweder als physikalischer Prototyp oder als digitaler Prototyp:

- Unter einem *physikalischen Prototypen* (physical mock-up, PMU) versteht man den physikalische Aufbau eines Prototypen, mit dem der Anwender direkt interagieren, d.h. ihn anfassen kann. Solche Prototypen werden sehr oft aus Holz oder Ton aufgebaut. Im folgenden werden auch Papierausdrucke (2D-Plots) als physikalischer Prototyp verstanden, da auch mit diesem Medium direkt interagiert werden kann.
- Unter *digitalen Prototypen* (digital mock-up, DMU) versteht man geometrische, digitale Repräsentationen des zu begutachtenden Bauteils, die mittels einer Graphiksoftware und einem Projek-

tionssystem (z.B. Monitor, Leinwand) angezeigt werden können. Da die Bauteile heutzutage in den allermeisten Fällen mittels CAD konstruiert werden und dadurch bereits als digitale Modelle vorliegen, können diese mit entsprechender Technologie relativ einfach visuell präsentiert werden. Mit den digitalen Modellen kann jedoch nicht direkt interagiert werden, sondern nur über eine entsprechende Softwareschnittstelle.

Wurden zuerst nur physikalische Prototypen und 2D-Plots als Diskussionsgrundlage für den Review eingesetzt, so geht heute der Trend hin zu digitalen Prototypen, um zum einen Kosten und Zeit zu sparen (time-to-market) und zum anderen auch eine wesentlich größere Variantenvielfalt eines Produktes anbieten zu können, um individuellen Kundenwünschen gerecht zu werden. Diese Vielfalt ist jedoch nur dann unter vertretbarem Zeit- und Kostenaufwand möglich, wenn die eigentliche Entwicklung eines Produktes sehr schnell durchgeführt werden kann und kleinere Veränderungen auf einfache Weise in das Gesamtprojekt einbezogen und auf ihre Tauglichkeit hin evaluiert werden können ([GOME99]). Bereits aufgrund der Erstellungsdauer physikalischer Prototypen wäre ein solcher Schritt kaum zu leisten.

Jedoch hat eine Umstellung von physikalischen auf digitale Prototypen eine sehr einschneidende Wirkung für die Begutachtung, denn hier haben auf den ersten Blick die physikalischen Prototypen einige erhebliche Vorteile. So sind sie ein direktes Abbild des realen Objektes und das "Vertrauen" in ein bereits gefertigtes Objekt ist bekanntlich höher, als in eins, welches nur auf dem Bildschirm existiert. Weiterhin ist die Interaktion mit einem physikalischen Prototypen intuitiv und sehr natürlich. So können z.B. Abstandsmessungen sehr einfach mittels eines Maßstabes durchgeführt werden. Digitale Prototypen hingegen sind nur die Vorlage für das reale Produkt und somit ist nur indirekt sichergestellt, dass das reale Modell der Vorlage exakt entspricht. Auch muss ein Weg gefunden werden, um die mit dem physikalischen Prototypen möglichen Interaktionen (z.B. Messen), auch mit dem digitalen Prototypen durchführen zu können. Wichtig hierbei ist, dass die Interaktion ähnlich intuitiv, einfach und natürlich durchgeführt werden kann. Dadurch wird dieser Medienwechsel für den Anwender erleichtert und ein zügiges Arbeiten unterstützt.

### 2.2.3 Anwendungsszenarien

In den Arbeiten von [LECA00] und [BLAN98] wurden Untersuchungen im Umfeld Automobilbau durchgeführt und insbesondere auch der Design Review untersucht. Hierbei wurden u.a. die folgenden allgemeinen Anwendungsszenarien identifiziert:

- Darstellung des Standes der Entwicklungen
- Validierung von Lösungen
- Diskussionsgrundlage für Experten aus verschiedenen Bereichen, um deren Wissen zu koordinieren und in das Design zu integrieren.
- Demonstrator für Entscheidungsträger[1]

Diese allgemeinen Anwendungsszenarien lassen sich auf die folgenden konkreten Anwendungen ab-

1. Diese Anwendungsklasse zeichnet sich vor allem durch ein sehr einfaches UI aus, welches nur über sehr wenige Funktionen verfügt. Wie in der Positionierung festgelegt, ist dies kein Szenario, welches in dieser Arbeit untersucht werden wird.

bilden, die in den folgenden Kapiteln näher beschrieben werden:

- Begutachtung von Presswerkzeugen (Überprüfung von Richtlinien und Funktionen
- Lacktrocknungssimulation (Evaluierung von FE-Simulationsergebnissen)
- Begutachtung von Einbau-/Montagesimulationen

#### 2.2.3.1 Begutachtung von Preßwerkzeugen

Bei einer Gesamtfahrzeugsentwicklung wird nicht nur das Fahrzeug selbst, sondern auch die Produktion der einzelnen Bauteile geplant. Dies umfasst neben der Konzeption der gesamten Fertigungsstraßen auch die einzelnen Presswerkzeuge, die die Blech- und Metallteile des Fahrzeuges in ihre jeweilige Form pressen.

Ein Presswerkzeug besteht aus einem Stempel, einem Einsatz und einem Blechhalter. In Abb. 2.4 ist ein Presswerkzeug beispielhaft abgebildet, einmal in geöffnetem und einmal im geschlossenen Zustand.

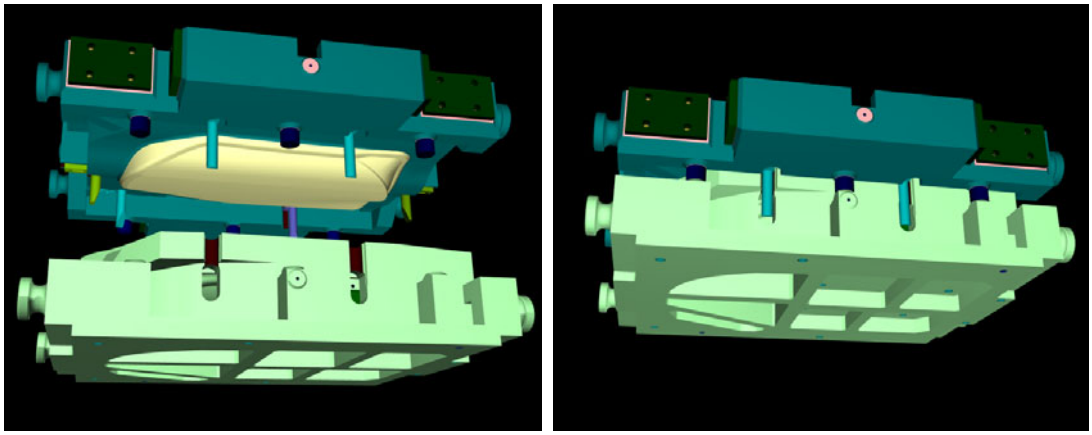

Abb. 2.4: Presswerkzeug, geöffnet und geschlossen

Für Presswerkzeuge ist es unmöglich physikalische Prototypen zu bauen, da sie extrem teuer, groß und schwer sind. So hat ein typisches Presswerkzeug eine Kantenlänge zwischen 4 und 6 Meter und wiegt zwischen 10 und 30 Tonnen. Hier ist somit besonders wichtig, dass die Anforderungen an das Werkzeug eingehalten werden, denn die Korrektur eines nicht festgestellten Designfehlers ist extrem zeit- und kostenintensiv. Sehr wichtige Werkzeuge für die Designüberprüfung sind das Messen von Abständen und Bohrlöchern, sowie die Möglichkeit achsparallele Schnitte durch das Objekt zu legen. Da Presswerkzeuge immer als „Solids“[1] modelliert werden, soll auch beim Schnitt das Innen und Außen deutlich sichtbar gemacht werden.

Für den Konstrukteur ist es weiterhin sehr interessant, die Bewegung der einzelnen Bauteile des Presswerkzeuges inklusive der Schieber (Messer, die Teile des Blechs abschneiden), als Animation ablaufen zu sehen.

Weiterhin wird die Pressung der einzelnen Blechteile vorab simuliert. Hierbei wird sowohl die Ver-

1. virt. Objekte mit einem definierten innen und außen

formung, als auch die Veränderungen der Blechdicke berechnet.

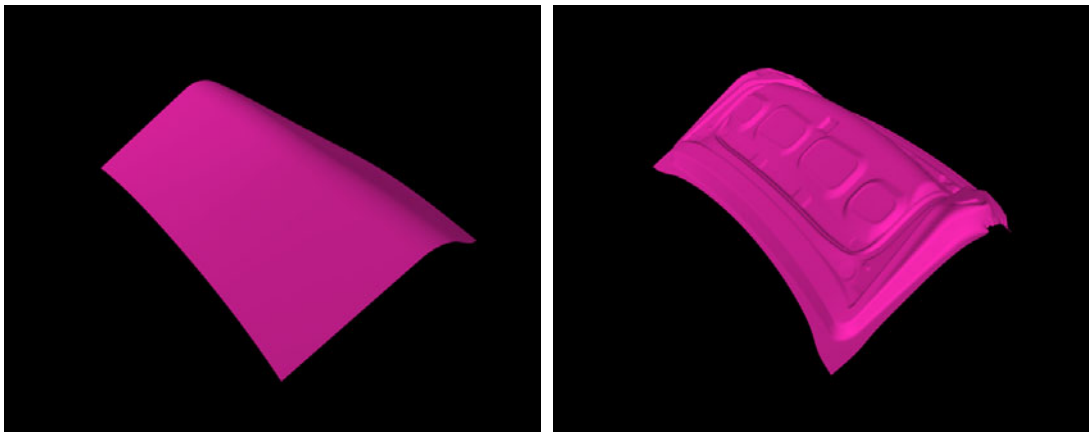

Abb. 2.5: Simulation des Preßvorgangs für t=0 und t=n bei $t \in [0;n]$

Dieses Simulationsergebnis muss visuell dargestellt werden können. Da der gesamte Verformungsprozess über mehrere Zeitschritte simuliert wird, muss es die Möglichkeit geben, zwischen einzelnen Zeitschritten diskret umschalten zu können, bzw. als Animation ablaufen zu lassen. Um kritische Blechdicken erkennen zu können, müssen diese visualisiert und interaktiv durch Plazierung von Datensonden abgefragt werden können.

#### 2.2.3.2 Lacktrocknungssimulation

Wird das Fahrzeug produziert, so wird zuerst die Karosserie zusammengebaut, verschweißt und anschließend lackiert. Hierbei wird die Karosserie eines Fahrzeuges mit Lack besprüht, auf eine bestimmte Temperatur aufgeheizt und anschließend wieder abgekühlt. Dieser Vorgang sichert eine optimale Haftung des Lacks und führt bei richtiger Anwendung zu einer gleichmäßige Färbung über die gesamte Karosserie. Hierbei ist es jedoch sehr wichtig, dass weder ein Teil der Karosserie zu heiß wird, da ansonsten der Lack verbrennen würde, noch zu kalt, da ansonsten die nötige Haftung nicht garantiert werden kann. Auch sollte der Aufheizprozess möglichst gleichförmig sein, da ansonsten die Färbung des Lacks sich verändern kann.

Dieses Szenario setzt somit vollständig auf die Begutachtung der Simulation des Aufheiz- und Trocknungsprozesses. Die Simulation berechnet für diskrete Zeitschritte, basierend auf den CAD-Daten der Karosserie, die Temperaturverteilung pro Knoten.

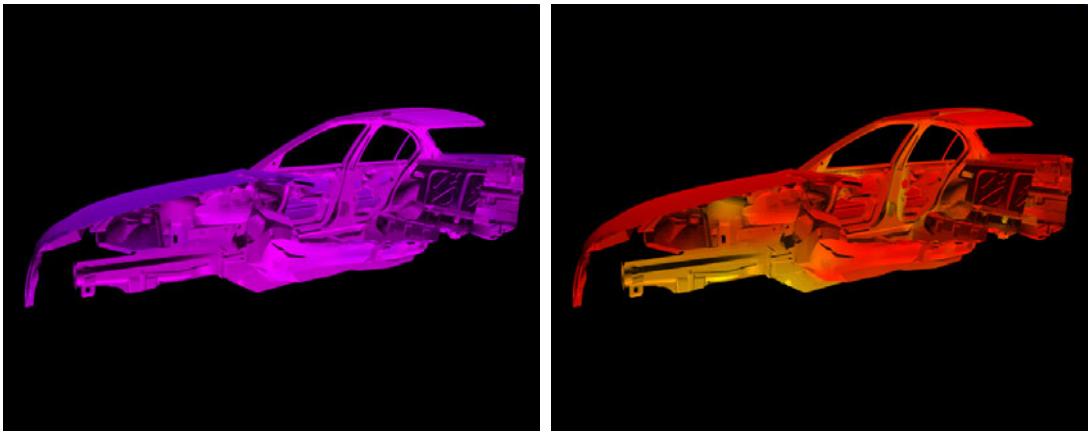

Abb. 2.6: Simulation eines Lacktrocknungsprozesses

Der Anwender muss nun die Möglichkeit besitzen, zwischen den einzelnen Zeitschritten umschalten zu können, die Temperatur an bestimmten Knoten pro Zeitschritt zu erfragen und eine Aufheizkurve, d.h. Temperaturentwicklung an einem Punkt über die Zeit, ausgeben zu lassen. Da sich bei einer Karosserie verschiedene Bleche teilweise oder ganz verdecken, muss es möglich sein, Teile der Szene auszublenden.

#### 2.2.3.3 Begutachtung von Einbau-/Montagesimulationen

Einbau-/Montagesimulationen sind bereits heute teilweise in VR möglich ([GOME99]). Hierbei versucht ein Werker ein bestimmtes Bauteil aus dem virtuellen Modell aus- oder einzubauen, ohne dabei mit dem virtuellen Modell zu kollidieren. Der zurückgelegte Weg wird aufgezeichnet und als Pfad visualisiert.

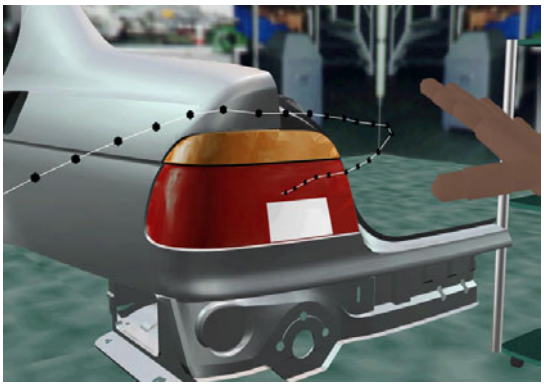

Abb. 2.7: Erzeugen eines Ausbaupfades in VR

In einem Design Review werden die verschiedenen Ausbaupfade bewertet und auch mit unterschiedlichen Modellvarianten des untersuchten Bauteils durchgespielt. Sollten hierbei Probleme auftreten, werden Lösungsansätze besprochen, inwiefern Bauteile anders angebracht werden könnten. Hierbei muss es möglich sein einzelne Bauteile zu selektieren und bauteilspezifische Informationen zu erfragen.

## 2.3 Anforderungskatalog nach dem 4W-Modell

Basierend auf den im vorherigen Kapitel vorgestellten generellen Ablauf eines Design Reviews, kann nun in einem nächsten Schritt das 4W-Modell angewendet werden, welches zu einem Anforderungskatalog führt, der in Kapitel 2.3.1 vorgestellt wird.

Wie bereits in Kapitel 2.1.1 dargelegt, umfaßt das 4W-Modell nicht das Herausarbeiten der genauen Funktionalitäten, die ein VR-Softwaresystem für den Design Review umfassen sollte. Deswegen werden diese funktionalen Anforderungen erst in Kapitel 2.3.2 beschrieben. Diese werden dann in Kapitel 6 wieder aufgegriffen und es wird exemplarisch gezeigt, wie diese in die zuvor definierte Benutzeroberfläche (in Kapitel 4 und Kapitel 5) integriert werden können.

### 2.3.1 Anforderungen an die Benutzerschnittstelle

Im folgenden werden nun die Anforderungen an die Benutzerschnittstelle vorgestellt:

**Einfache Benutzbarkeit:** Der Design Review wird zumeist von Personen ausgeführt, die zwar Experten bei der Konstruktion und Bewertung von Bauteilen sind, jedoch nicht über umfassendes Computerwissen verfügen. Somit sollte das Softwaresystem möglichst intuitiv und einfach zu benutzen sein.

**Einheitliche Benutzeroberfläche:** Teile der hier gestellten Anforderungen sind mit unterschiedlichen Softwaresystemen bereits heute umzusetzen. Diese haben jedoch unterschiedlichste und zumeist komplexe Benutzeroberflächen. Das bedeutet, dass der Anwender den Umgang mit einer Vielzahl von Softwaresystemen lernen und auch diese immer wieder anwenden muss, um mit der Bedienung vertraut zu bleiben. Dies ist jedoch in der täglichen Anwendung sehr hinderlich und deswegen stellt sich die Forderung nach einer einheitlichen Benutzeroberfläche, die auch implizit die Anwendung des Softwaresystems vereinfacht.

**Schnelles Erlernen:** Insbesondere in der Automobilindustrie ist es heutzutage üblich, sehr viele Komponenten eines Fahrzeuges von Zulieferern entwickeln zu lassen. Basierend auf den Anforderungen konstruieren sie die Komponente und besprechen dann während eines Design Reviews den Entwurf. Die Präsentation der Komponente macht es erforderlich, dass derZulieferer das System selbst bedient. Da man nicht davon ausgehen kann, dass der Zulieferer jeden Tag mit diesem System arbeitet, muss es sehr schnell erlernbar sein. Da auch interne Reviews nicht auf täglicher Basis stattfinden, gilt diese Forderung auch hierfür.

**Real-Darstellung:** Insbesondere im Bereich der Presswerkzeuge und des Stylings ist es sehr wichtig, dass die Modelle in Realgröße dargestellt werden, sodass der Anwender die Modellgröße einschätzen kann. Hierbei spielt auch die stereoskopische Darstellung eine wichtige Rolle, um eine korrekte 3D Darstellung zu erhalten. Eine wesentliche Hilfe beim Einschätzen von Abständen ist es auch, wenn der Anwender sich selbst im Vergleich zur virtuellen Szene sehen kann (*Präsenz*).

**Täglicher Einsatz:** Sowohl das System, als auch die verwendeten Ein- und Ausgabegeräte müssen auf den täglichen Einsatz ausgelegt sein, da davon auszugehen ist, dass eine Installation speziell für den Design Review von mehreren Teams oder Abteilungen geteilt wird. Das bedeutet insbesondere für das Softwaresystem, dass es jederzeit stabil läuft und nicht durch Benutzereingaben zum Absturz

gebracht werden kann. Ebenso müssen die Eingabegeräte sehr robust sein, da sie von vielen Personen benutzt werden und nicht unbedingt vorsichtig behandelt werden.

**Unterstützung von mehreren Aktoren:** In [LECA00] wurde dargelegt wie wichtig es für die Kommunikation ist, dass potenziell jeder Teilnehmer eines Design Reviews die Kontrolle über das System erhalten kann („Gleichberechtigung“). Aus diesem Grund muss das System auch die Möglichkeit bieten, dass mehrere Anwender gleichzeitig mit dem System agieren können. Da es sich als verwirrend erweisen könnte, wenn jeder Teilnehmer zu jeder Zeit alle möglichen Aktionen ausführen kann (z.B. jeder kann das sichtbare Objekt beliebig drehen), sollte zum einen die Gesamtkontrolle über das System bei einem einzigen Anwender liegen, die jedoch sehr einfach weitergegeben werden kann. Zum anderen sollte jedoch jeder Anwender über einen Satz Funktionen verfügen, die ihm eine sinnvolle Interaktion mit dem digitalen Prototypen ermöglichen, z.B. sollte jeder Teilnehmer eine Zeigemöglichkeit besitzen.

**Unterstützung von mehrere simultanen Sichten auf das Modell:** Die Kommunikation während des Reviews ist sehr wichtig, da dadurch die meisten Designfehler gefunden werden. Dies setzt aber auch voraus, dass während eine Gruppe eine bestimmte Problematik bespricht, ein Einzelner oder eine kleine Gruppe über ein anderes Problem nachdenken, dazu aber auch die entsprechende Problemstelle sehen können. An einem PMU oder Plot ist das sehr leicht möglich, da immer das ganze Modell sichtbar ist. Bei einer Darstellung des Modells am Bildschirm ist dies jedoch nicht gegeben, da das Modell immer nur aus einer Ansicht zu einer Zeit dargestellt ist und es somit nicht garantiert ist, dass die andere Problemstelle ebenfalls sichtbar ist.

### 2.3.2 Anforderungen an die Funktionalitäten

In diesem Kapitel werden die verschiedenen Anforderungen an die Funktionalität des Softwaresystems besprochen.

**Modelluntersuchung:** Die Modelluntersuchung umfasst mindestens die Möglichkeiten, die auch bei einem PMU oder Plot möglich sind. Zum einen Abstände zu messen und dadurch einzuhaltende Richtlinien zu überprüfen, zum anderen achsparallele oder arbiträre Schnitte durch ein Modell zu legen. Weitere Funktionalitäten ergeben sich aus den Anforderungen der jeweiligen Applikation.

**Skizzieren und Annotieren:** In Kapitel 3.3.3 wurde die Wichtigkeit des Skizzierens von Ideen und des Annotieren von markanten Punkten direkt auf den Plots den physikalischen Prototypen, dargelegt. Diese Möglichkeit bildet eine der wichtigsten Grundlagen einer effektiven Kommunikation und der Entwicklung neuer Ideen und Lösungen. Es müssen hier also Möglichkeiten entwickelt werden, um auch auf einem digitalen Prototypen zu skizzieren und diesen zu annotieren.

**Dokumentation:** Das Resultat einer Design Review Sitzung ist ein Ergebnisprotokoll. Dies wird bei Verwendung von PMUs und Plots meist mit Papier und Bleistift oder einer Textverarbeitung verfasst. Weiterhin wird in dem Protokoll auf Annotationen auf dem PMU oder Plot verwiesen, um eine Bemerkung besser zu illustrieren. Da diese Verweise sehr wichtig zum Verständnis sind, muss das Softwaresystem eine ähnliche Möglichkeit bieten, indem z.B. Bilder der dargestellten Szene erzeugt und mit dem Protokoll verbunden werden können. Da die meisten Konstrukteure oder externe Zulieferer dieses Softwaresystem an ihrem Arbeitsplatz nicht zur Verfügung haben, muss die vollständige Do-

kumentation auch ohne das System einzusehen sein. Hier bietet sich z.B. die Ausgabe eines HTML-Dokumentes an.

**Simulationsintegration:** Wie aus den verschiedenen Anwendungsszenarien hervorgeht, ist die Integration von Offline-Simulationen ein sehr wichtiger Bestandteil des Design Reviews. Hierbei steht insbesondere die Visualisierung von 2D Daten im 3D Raum mit skalaren Simulationswerten im Vordergrund. Der Anwender muss die Möglichkeit besitzen das Mapping interaktiv zu ändern, sowie auf verschiedene Zeitschritte zuzugreifen und Datenproben zu plazieren, sowohl für einen, als auch für mehrere Zeitschritte. In letzterem Fall sollte eine Kurve aus den einzelnen Datenproben generiert werden.

## 2.4 Vorstellung existierender Systeme

In diesem Kapitel werden nun einige Systeme vorgestellt, die den Design Review Prozeß unterstützen sollen.

Das wohl verbreitetste Tool zur Visualisierung von Planungszuständen ist der 4D Navigator ([NAVI99]), entwickelt von IBM und Dassault. Die weite Verbreitung ist insbesondere dadurch begründet, dass sehr viele Automobilfirmen CATIA ([CATI99]) als CAD-Programm einsetzen, welches ebenfalls von IBM und Dassault entwickelt wird. Der 4DN ist ein reines Desktop-Tool, welches sich als Plugin in CATIA integrieren läßt und dadurch auch den vollen Zugriff auf die gesamte Datenbank ermöglicht. So können zu beliebigen Teilen der Szene Informationen erfragt werden.

Mit 4DN können sehr komplexe Modelle durch Verwendung von *Level-of-Detail* mit akzeptabler Geschwindigkeit dargestellt werden. Weiterhin wird die stereoskopische Darstellung über Shutter-Stereo unterstützt. Um durch die Szene zu navigieren, kann der Anwender eine Spacemouse einsetzten, mit der er die virtuelle Kamera bewegen kann. Dies ist die einzige Navigationsart, die der 4DN bietet.

Für die Modelluntersuchung bietet der 4DN eine 3D Schnittfläche an, die jedoch keine spezielle Darstellung für Solids bietet. Zu Dokumentationszwecken können Kamerapositionen abgespeichert und 2D-Marker in der Szene plaziert werden. Diese Marker sind jedoch an eine bestimmte Kameraposition gebunden. Sobald die Kamera sich an einer anderen Position befindet, verschwindet der Marker.

Nach Aussagen der meisten Anwender ist das Arbeiten mit 4DN sehr kompliziert. Zum einen ist die Bedienungsoberfläche sehr schwer zu verstehen, wodurch sehr viel Zeit für die Erstellung einer Szene benötigt wird. Zum anderen ist das System sehr instabil.

Wie aus der obigen Beschreibung bereits hervorging, sind die Möglichkeiten der Modelluntersuchung und Dokumentation sehr rudimentär und es besteht auch keine Möglichkeit FE-Simulationsergebnisse zu integrieren. Somit ist dieses Tool wenig geeignet, um den Design Review effektiv zu unterstützen.

Für die Betrachtung von Simulationsergebnissen wird u.a. Animator3 [GNS00] eingesetzt. Der Animator ist ebenfalls ein reines Desktoptool, welches mit der normalen Computermaus bedient wird. Er kann verschiedenste Simulationsdatenformate einlesen, unterstützt jedoch nur 2D-Daten im 3D-Raum mit skalaren Simulationswerten pro Knoten. Er bietet die Möglichkeit das Mapping an die je-

weiligen Anforderungen anzupassen und Datensonden zu plazieren. Ebenso wie beim 4DN ist die Benutzeroberfläche wenig intuitiv zu bedienen. Der Animator ist als reines Visualisierungstool für FE-Simulationen entwickelt worden und besitzt somit keine darüber hinaus gehenden Fähigkeiten, die für den Design Review wichtig sind.

Lightning ist das VR-System des Fraunhofer-IAO ([BLAC98]), auf dessen Basis eine Applikation für den Einsatz im Bereich der Begutachtung von Presswerkzeugen entwickelt wurde ([HAEFF99]). Diese Anwendung bietet neben einer sehr einfachen Bedienung über eine Art Flystick und die Ausgabe auf einer Powerwall, Methoden, um Abstände zu messen und Schnitte durch Solids zu legen. Über ein einfach zu benutzendes Menüsystem (Interaction Ball, siehe Kapitel 4.5.3.5) können die verschiedenen Funktionen des Systems aktiviert werden. Es bietet weiterhin die Möglichkeiten zwischen Dokumentation über 3D-Markierungen und Texteingabe. Die 3D-Markierungen müssen mit der virtuellen Hand in der Szene plaziert werden. Dies ist natürlich in weitläufigen Szenen sehr aufwendig und auch eine exakte Zuordnung zu einem bestimmten Bauteil ist so nicht möglich.

Dieses System ist exakt auf die hier beschriebenen Funktionalitäten zugeschnitten und kann somit schwer erweitert werden. Dies liegt u.a. an dem hier verwendeten Menüsystem, auf welches in Kapitel 4.5.3.5 näher eingegangen wird. Weiterhin unterstützt das System nicht die Integration von FE-Simulationen und auch die Integration in den CA-Prozess ist kaum ausgeprägt.

Auf dem Markt existieren darüber hinaus verschiedene andere VR-Systeme, wie der Realizer von Opticore [OPTI00] und Realax [REAL00], um nur einige zu nennen. Diese Systeme bieten zwar eine Anbindung an die verschiedensten Ein- und Ausgabegeräte und können auch Geometriedaten einlesen und visualisieren, jedoch bieten sie keine Anwendung, die die Anforderungen des Design Review erfüllt. Keines dieser Systeme unterstützt die Visualisierung von FE-Simulationen.

## 2.5 Generalisierung der Anforderungen und Definition der Anwendungsklasse

Es würde den Rahmen dieser Arbeit sprengen, wenn hier für alle möglichen Ausprägungen des 4W-Modells Lösungen entwickelt und angeboten werden würde. Aus diesem Grund wird im Rahmen dieser Arbeit der VR-Design Review als Stellvertreter einer Anwendungsklasse genauer untersucht und die entwickelten Lösungen hieran evaluiert.

VR-Design Review ist jedoch nicht nur aufgrund der breiten Anwendung interessant, sondern auch wegen den Voraussetzungen (s.o.), unter denen ein solcher Review stattfindet. Diese fanden bis heute wenig Beachtung in der Literatur, so dass für viele Fragestellungen in diesem Bereich nur rudimentäre Ansätze beschrieben wurden.

Um die Entwicklungen dieser Arbeit jedoch etwas von der Anwendung Design Review zu lösen und dadurch eine bessere Allgemeingültigkeit zu erreichen, werden hier die auf Basis des 4W-Modells herausgearbeiteten Anforderungen generalisiert. Diese definieren nun nicht mehr die Randbedingungen für den Design Review, sondern für eine umfassender Anwendungsklasse. Diese dienen nun für die weiteren Arbeiten als Grundlage.

- **Anwender:**
  - *VR-Erfahrung:* Die meisten Anwender haben sicherlich wenig bis keine Erfahrung im

Umgang mit VR-Systemen. Hierbei ist es interessant, wie mit unterschiedlichen Gedankenmodellen umgegangen werden kann.

- **Intention:**
  - *Arbeiten auf technologischer Basis:* Hier soll insbesondere der industrielle Anwendungsfall betrachtet werden, wodurch sich die Intention „Arbeit“ von alleine erschließt. Da viele gängige VR-Anwendungen dem Anwender heute nur kleine Funktionssets zur Verfügung stellen, soll hier untersucht werden, wie dem Anwender in der virtuellen Realität eine Vielzahl an Funktionen dargeboten werden kann.
- **Ablauf:**
  - *Häufige und lange Benutzung des Systems:* Viele VR-Bedienkonzepte entstehen in einer Laborumgebung, die sich vor allem auch dadurch auszeichnet, dass solche Konzepte nicht im täglichen Einsatz unter mehrstündiger durchgehender Verwendung betrachtet werden können und die damit einhergehenden Fragestellungen nicht berücksichtigen.
  - *Kooperatives Arbeiten an einem Ort:* Viele Forschergruppen beschäftigen sich vor allem mit dem verteilten Arbeiten. Hier sind sowohl technologische Konzepte (Netzwerk etc.), als auch Interaktionsmethoden zu entwickeln und Problemstellungen zu lösen (z.B. gleichzeitiges Greifen von Objekten) und es wurden bereits interessante Lösungen entwickelt. Ein noch recht weites Feld bietet das kooperative Arbeiten in VR von mehreren Personen an einem Ort, eine Situation, die auch in der Realität sehr oft vorkommt. Auf diese Ausprägung wird in dieser Arbeit ein sehr starkes Augenmerk gelegt (siehe Kapitel 5).
- **Inhalte:**
  - Hier wird keine spezielle Ausprägung betrachtet. Auf mögliche Besonderheiten und Einflüsse auf die zu verwendenden Interaktionstechniken wird im jeweilige Kontext eingegangen.

Der verfolgte Anwendungstypus läßt sich somit umgangssprachlich mit dem folgenden Satz beschreiben:

*„Experten arbeiten gemeinsam an einer Problemlösung auf Basis virtueller Modelle“* (1)

Diesem Ausprägungsprofil kommt als Anwendungstypus der Prozess des „Design Review“ am nächsten. Dieser wird in dem nächsten Kapitel näher beschrieben und es werden weitere, spezielle Randbedingungen gemäß dem 4W-Modell herausgearbeitet.

## 2.6 Zusammenfassung

In diesem Kapitel wurden wichtige Grundlagen für die in dieser Arbeit behandelte Thematik entwickelt und vorgestellt. Die Schemata zur Klassifizierung von VR-Benutzeroberflächen geben eine fundierte und klare Vorgehensweise vor, die es erlaubt, die für das Design relevanten Anforderungen herauszuarbeiten, die dann in einem folgenden Schritt in eine konkrete Umsetzung münden. Mit Hilfe des 4W- und Abhängigkeitenmodells können Entwickler so die GUI-Problemstellung sinnvoll eingrenzen und beschreiben.

Diese Ausprägungen können auch als 4-dimensionaler Raum betrachtet werden, in dem eine Anwendungsklasse einen definierten 4D-Bereich abdeckt. Überschneiden sich der Bereich einer bestehenden Anwendung und einer neu zu entwickelnden Anwendung, können die Konzepte aus der Überlappung wiederverwendet werden.

Als Beispielanwendung wurde der VR-Design Review vorgestellt und die Vorgehensweise anhand der Modelle illustriert, sowie deren Ergebnisse dargelegt. Die Wahl fiel hierbei auf VR-Design Review aus zwei Gründen. Zum einen ist der Design Review eine repräsentative Ausprägung einer breiten Anwendungsklasse, zum anderen werden dort Fragestellungen adressiert, die in der Literatur bis heute wenig beachtet wurden.

# 3 Grundlagen der immersiven und intuitiven Interaktion

Sind die Randbedingungen und Parameter mit Hilfe der Schemata aus Kapitel 2 herausgearbeitet worden, so ist einer der nächsten Schritte, die passenden Interaktionstechniken zu identifizieren und/ oder zu entwickeln. Hierbei sind verschiedene Faktoren zu beachten, die in diesem Kapitel näher beleuchtet werden.

Zum Beginn wird in Kapitel 3.1 die Technologie Virtuelle Realität (VR) vorgestellt und die Aspekte Gerätetechnologie und Echtzeitgraphik näher betrachtet.

Nach den Modellen werden die Interaktionstechniken nicht nur durch die visuelle Darstellung und Ablauffolgen, sondern auch durch die verwendeten Geräte geprägt. Sowohl Geräteschnittstelle als auch Softwareschnittstelle werden sowohl durch den Anwender, als auch die Applikation beeinflußt. Von seiten des Anwenders sind dies die menschliche Physiognomie, Wahrnehmung und Interaktion. Von seiten der Applikation sind dies zum einen spezifische Anforderungen an die Geräteschnittstelle und zum anderen die von der Software bereitzustellenden Funktionalitäten. Letztere sind jedoch nicht grundlagenspezifisch und werden in Kapitel 6 behandelt. Im folgenden werden zu diesen Aspekten die wichtigsten Grundlagen vorgestellt.

Weiterhin wird das Konzept der logischen Interaktionsaufgaben vorgestellt (siehe Kapitel 3.3.4), welches im folgenden die Basis für die softwareseitige Entwicklung bildet und eine definierte Herangehensweise an die in Kapitel 4 vorgestellten Interaktionstechniken zulässt.

## 3.1 Virtuelle Realität - Die Basistechnologie

Die Anforderungen sowohl an die Darstellung, als auch an die Benutzeroberfläche legen den Schluss nahe, *Virtuelle Realität* (VR) als Basistechnologie für ein Softwaresystem für den Design Review einzusetzten. Im folgenden Kapitel werden zum besseren Verständnis die Grundlagen der Virtuellen Realität kurz umrissen.

Mit dem Begriff *Virtuelle Umgebungen* oder auch besser bekannt unter den Begriffen *Virtuelle Realität* oder kurz *VR*, wird eine neue Kommunikationstechnik zwischen Mensch und Maschine bezeichnet. Im Unterschied zu nicht-interaktiven, kommandogesteuerten, aber auch graphisch-interaktiven Systemen kann man mit VR-Systemen Daten auf eine Art und Weise präsentieren, die das Verstehen der präsentierten Sachverhalte erleichtert und dabei erlaubt, intuitiv mit den virtuellen Daten zu interagieren. Diese Daten werden über einen Graphikrechner in die entsprechenden Bildsequenzen umgewandelt, wobei dies nicht nur auf natürliche Umgebungen und Prozesse beschränkt ist, sondern auch künstliche (synthetische) Umgebungen geschaffen oder Informationen dargestellt werden können, die man in einer natürlichen Umgebung nicht wahrnehmen (sehen, hören, fühlen, riechen, schmecken) würde. Der Mensch kann sich in der virtuellen Umgebung intuitiv bewegen, er kann beispielsweise umherschauen oder auch Objekte greifen. Die Idee der VR ist, die reale mit der virtuellen Welt verschmelzen zu lassen und der Anwender somit als aktiver Bestandteil in die virtuelle Welt eintauchen und nicht wie ein Zuschauer durch ein Fenster (den Bildschirm) diese Welt beobachten kann. Sutherland ([SUTH65]) hat bereits 1965 diese Art der Integration des Menschen in eine computergenerierte Welt sehr treffend in seinen Vorstellungen zum ultimativen Bildschirm formuliert: *Indeed, in the ultimate display, one will not look at that world through a window, but will be immer-*

*sed in it*. Diese Immersion bedeutet aber auch, dass der Mensch den Eindruck hat, in einer realistischen Situation zu sein, auch wenn es sich um eine Simulation handelt. Die heutigen VR-Systeme, die diese Anforderungen erfüllen bzw. erfüllen müssen, können durch das in Abb. 3.1 skizzierte Modell dargestellt werden, das auf dem in [ENCA93] beschriebenen SIP-Modell basiert und auf VR-Systeme angewendet wurde.

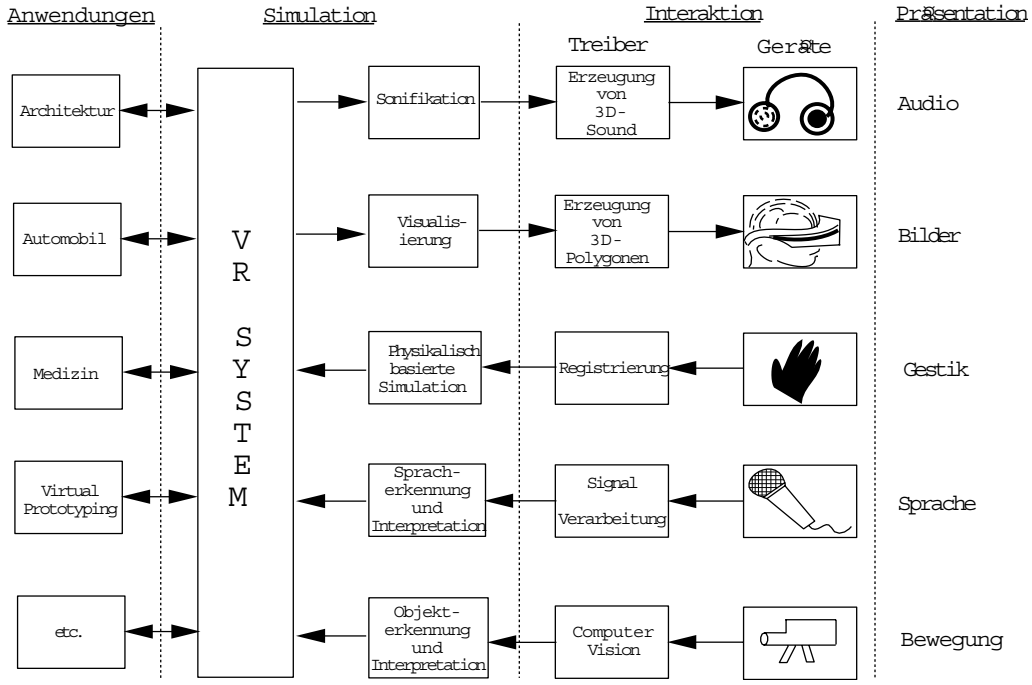


Abb. 3.1: SIP-Modell

Die Hauptbestandteile dieses Modells sind:

- **Präsentation:** Die Darstellung der virtuellen, synthetischen Umgebung unter Einbeziehung von visuellen und akustischen Wahrnehmungen.
- **Interaktion:** Die Ansteuerung von und die Kommunikation mit geeigneten Ein- und Ausgabegeräten sowie die Definition entsprechender Interaktionstechniken
- **Simulation:** Die Beschreibung der synthetischen Umgebung und damit die Simulation natürlicher und/oder synthetischer Vorgänge

Neben dem Aufbau eines VR-Systems sind die Qualität und Quantität der Darstellung, der mögliche Immersionsgrad sowie die Gerätetechnologie weitere wichtige Merkmale eines VR Systems.

### 3.1.1 VR-Gerätetechnologie

In den folgenden Abschnitten werden die verschiedenen Geräte beschrieben, die in VR-Systemen eingesetzt werden. Es werden dabei die drei Geräteklassen *Eingabegeräte*, *akustische Ausgabegeräte* und *visuelle Ausgabegeräte* behandelt.

#### 3.1.1.1 Eingabegeräte

Die typischen VR-Eingabegeräte erlauben eine multimodale und multidimensionale Interaktion innerhalb der virtuellen Szene:

**Tracking-Systeme** verfolgen die Bewegungen des Bedieners. Sie liefern eine 6D-Information über Position und Orientierung des Betrachters. Elektromagnetische Tracking-Systeme werden zur Zeit am häufigsten eingesetzt. Darüber hinaus gibt es mechanische, Ultraschall-, Video- sowie Infrarot-Systeme.

**Datenhandschuhe** werden für die Erkennung der Fingerstellung (Winkel der Fingergelenke, Fingerspreizung) des Betrachters eingesetzt. Auf diesen Daten basierend können vordefinierte Gesten erkannt und bestimmte Aktionen ausgelöst werden ([BÖHM96]. Auf dem Datenhandschuh ist ein *Tracking-Sensor* befestigt, der die räumliche Position der Hand bestimmt.
In der Entwicklung befinden sich sogenannte Datenanzüge, mit denen die Stellung von Armen und Beinen bestimmt werden kann. Der Datenanzug findet bisher jedoch nur in wenigen Forschungseinrichtungen Einsatz.

**3D-Joysticks** haben die gleiche Funktionalität wie der Datenhandschuh, die Aktionen werden im Gegensatz zum Datenhandschuh durch Schalter am Griff des Joysticks ausgelöst. Die Interaktion ist somit auf die Anzahl der Schalter begrenzt. Auch bei dem 3D-Joystick dient ein Tracking-Sensor für die räumliche Positionsbestimmung.

**Die Spacemouse** ist eine Weiterentwicklung der traditionellen Computermaus, über die anstatt nur zwei, alle sechs Freiheitsgrade (6-DOF) manipuliert werden können. Die Spacemouse ist für Desktop-Anwendungen oder kombinierte Interaktion mit dem Datenhandschuh sehr gut geeignet. Mit der Spacemouse können sehr leicht und intuitiv Begehungen {*walk-through* / *fly-through*) einer Virtuellen Umgebung gesteuert werden. Die Spacemouse besteht aus einem Puck und einem Tastenfeld. Der Puck kann sowohl entlang der drei Grundachsen transliert, als auch rotiert werden, wobei der Betrag der Translation und Rotation stark beschränkt ist. Somit bietet es sich an die Bewegungsdaten der Spacemouse relativ zu betrachten, d.h. dass die von der Spacemouse gesendete Position und Orientierung auf die aktuelle Betrachterposition addiert wird. Mit Hilfe der Tasten können weitere Aktionen ausgelöst werden, z.B. das Verändern der Navigationsgeschwindigkeit.

**Spracheingabe** ermöglicht, zusätzlich zum Datenhandschuh oder einem anderen Eingabegerät, dem System Kommandos zu übermitteln. Dies macht immer dann Sinn, wenn mit der Hand keine weiteren Interaktionen möglich sind, weil beispielsweise ein Objekt gegriffen werden und gleichzeitig navigiert werden soll. Die heute verfügbaren PC-basierten Spracheingabesysteme sind zwar bereits sehr leistungsfähig und können so trainiert werden, dass sie benutzerunabhängig arbeiten, jedoch ist die Erkennungsrate meist nicht so optimal, dass von einem produktiven Einsatz dieser Technologie eher abzuraten ist.

Neben den vorgestellten Eingabegeräten gibt es noch eine kleine Anzahl spezieller Geräte, die für besondere Anwendungen entwickelt wurden. Ein Beispiel hierfür sind *Eye-Tracking-Systeme*, die in Flugsimulatoren zum Einsatz kommen, denn Piloten bewegen ihren Kopf relativ selten, vielmehr verändern sie das Sichtfeld hauptsächlich durch Augenbewegung. Diese Systeme registrieren jede dieser

Augenbewegung und berechnen dementsprechend das aktuelle Bild. Zufriedenstellende Eye-Tracking-Systeme sind jedoch sehr kostspielig. Die sogenannten *Biosignalverarbeitungssysteme* versuchen menschliche Biosignale (z.B. EEG, Muskelkontraktionen, etc.) zu registrieren. Die ersten auf dem Markt verfügbaren Systeme lieferten jedoch sehr unbefriedigende Ergebnisse.

#### 3.1.1.2 Akustische Ausgabegeräte

Zur Präsentation akustischer Signale werden entweder die Audiofähigkeit heutiger Multimedia-Workstations genutzt oder es werden externe Geräte, beispielsweise Synthesizer, angesteuert. Die Ausgabe erfolgt über externe Lautsprecher oder Kopfhörer. Akustische Ausgabegeräte werden heute in der VR meist nur zur Ausgabe von einzelnen Signalen verwendet, z.B. für Systemmeldungen oder als Feedback beim Auftreten einer Kollision zwischen verschiedenen virtuellen Objekten. Die synthetische Sprachausgabe ist zwar bereits sehr weit entwickelt, jedoch noch immer leicht von natürlicher Sprache zu unterscheiden.

#### 3.1.1.3 Visuelle Ausgabegeräte

Die visuelle Ausgabe erfolgt entweder auf den üblichen *Desktop-Bildschirmen*, auf sogenannten *Großbildprojektionen* oder auf kopfgebundenen Darstellungssystemen. Das bekannteste Beispiel für ein kopfgebundenes System stellt der sogenannte *Datenhelm*, *head mounted display* dar. Für die Auswahl aus den verschiedenen Möglichkeiten spielt sowohl die geforderte Auflösung hinsichtlich der Bildqualität, die Physiognomie des Anwenders (Kapitel 2.1) als auch der Grad der Immersion eine wichtige Rolle. Um einen hohen Grad der Immersion zu erreichen, sollte das Gerät stereoskopische Ausgabe für die Tiefenwahrnehmung mit möglichst großem Gesichtsfeld (Weitwinkeldarstellung) bieten.

Sowohl bei den Desktop-Bildschirmen als auch bei den Großbildprojektionen kann eine stereoskopische Ausgabe über ein elektro-optisches Verschlußsystem (*Shutter-Technik*) realisiert werden. Diese Technik arbeitet mit einer Frequenz von bis zu 120Hz und schaltet alternierend zwischen dem Bild für das rechte Auge und dem Bild für das linke Auge um. Die aktiven Shutter-Brillen sind mit der Bildwiederholfrequenz des Bildschirms oder der Großbildprojektion synchron gekoppelt. Stellt der Computer das Bild für das linke Auge dar wird das rechte Auge von der Brille verdeckt und umgekehrt. Auf diese Weise erreicht man, dass das linke Auge nur die Bilder sieht, die für das linke Auge berechnet wurden und das rechte Auge nur die Bilder sieht, die für das rechte Auge berechnet wurden. Diese beiden Bilder werden vom visuellen System des Betrachters zu einem dreidimensionalen Bild zusammengesetzt. Problematisch an dieser Technik sind die hohen Kosten für die Brillen, sowie die Halbierung der effektiven Bildwiederholfrequenz, was schnell dazu führen kann, dass der Betrachter den Eindruck gewinnt, dass das Bild flimmert. Weiterhin verlieren Farben an ihrer Brillanz, sobald diese durch die Shutterbrille betrachtet werden. Somit sind sie ungeeignet für Applikationen, in denen Farbe sehr wichtig ist, z.B. im Bereich des Stylings.

Stereo-Ausgabe mit voller Bildwiederholfrequenz ist unter anderem mittels Großbildprojektion und auf Basis der Polarisationstechnik möglich (passives Stereo). Voraussetzung hierfür ist jedoch der Einsatz von Graphik-Workstations die die Möglichkeit bieten, die Bilder für das linke und das rechte Auge auf verschiedenen Ausgabekanälen gleichzeitig zu präsentieren. Die Synchronisierung der beiden Kanäle kann von der Hardware selbst erfolgen und benötigt keine zusätzliche Steuerung von au-

ßen. Jedes Bild wird dann mit der vollen Bildwiederholfrequenz über zwei Projektoren, die mit einem Polarisationsfilter bestückt sind, ausgegeben. Passive Polarisationsbrillen sorgen dafür, dass das korrekte Bild für das rechte bzw.~das linke Auge gefiltert wird. Vorteile dieser Technik sind die Gewährleistung einer hohen Bildwiederholfrequenz sowie die geringen Kosten für die passiven Polarisationsbrillen. Darüber hinaus ist der Immersionsgrad durch die größere Weitwinkeldarstellung der Großbildprojektion sehr viel stärker. Dieser Effekt wird bei der Darstellung auf mehreren Projektionsflächen, wie es beispielsweise bei der CAVE ([CRUZ93]) realisiert ist, um ein Vielfaches verstärkt. Hier werden mehrere Bilder der virtuellen Szene auf Wände, Decke und Boden projiziert. Der dadurch erzielte Eindruck ist vergleichbar mit einer holographischen Darstellung. Neben der verstärkten Immersion kommt in einer CAVE noch ein weiterer Punkt hinzu: das Gefühl der Präsenz. Der Anwender sieht seinen eigenen Körper in der virtuellen Szene und nicht nur einen vom Computer generierten Avatar, der teilweise sogar auf die Hand des Benutzers reduziert wird. Dadurch ist es dem Anwender auch wesentlich leichter, Größenverhältnisse in der virtuellen Szenerie abschätzen zu können, da er seinen Körper in direkter Relation zur Virtuellen Umgebung sehen kann.

Auch die kopfgebundenen Darstellungssysteme vermitteln dem Anwender einen starken immersiven Eindruck der virtuellen Umgebung. Die Darstellung der virtuellen Szene erfolgt über zwei kleine Monitore, je ein Monitor für ein Auge. Kopfgebundene Darstellungssysteme verwenden entweder *LCD* mit geringer Auflösung, *Kathodenstrahlröhren* (CRT) oder die weitaus teureren *Glasfaseroptiken* mit hoher Auflösung. Um ein möglichst großes Sichtfeld zu erzielen sind diese Monitore mit einer Weitwinkeloptik (ca. 100 Grad) ausgestattet. HMDs mit Kathodenstrahlröhren ermöglichen wie die Glasfaseroptiken eine hohe Auflösung (1280x1024 Pixel), besitzen jedoch ein hohes Gewicht. Sie sind daher nicht lange auf dem Kopf tragbar. Aus diesem Grund wurde auch eine Konstruktion entwickelt, die das System über ein mechanisches Gestänge vor die Augen führt, der sog. *BOOM*.

Darüber hinaus gibt es sogenannte *see-through* HMD's, die über ein semi-transparentes Display verfügen. Damit kann der Anwender die reale Umgebung weiterhin sehen. Die zusätzlichen Informationen werden über das transparente Display in den Sichtbereich des Benutzers eingeblendet. Diese Technik wird auch als *Augmented Reality* bezeichnet.

### 3.1.2 Präsentationskomponente

Die Präsentationskomponente eines VR-Systems setzt generierte Daten in visuelle und akustische Informationen um. Der Mensch nimmt über die Präsentationsebene die virtuelle Umgebung wahr und kann entsprechend interagieren. Die Leistungsfähigkeit eines VR-Systems in Bezug auf die Präsentation wird durch drei Merkmale (Qualität und Quantität der Darstellung, sowie dem Immersionsgrad) bestimmt, die in den folgenden Abschnitten erläutert werden.

#### 3.1.2.1 Immersionsgrad

Ein wichtiges Merkmal, welches die Leistungsfähigkeit eines VR-Systems bestimmt, ist der Grad der Immersion, das heißt wie realistisch die virtuelle Szene auf den Betrachter wirkt. Voraussetzung hierfür ist sowohl die räumlich-visuelle als auch die räumlich-akustische Darstellung der virtuellen Szene.

Bei der räumlich-visuellen Darstellung ist es wichtig, dass der Betrachter das Gefühl hat sich in der

virtuellen Szene zu befinden. Zum einen wird dies durch monokulare Signale wie Schatten (Lichtsimulation), perspektivische Darstellung (perspektivische Verkürzungen), parallaktische Verschiebungen von Objekten relativ zueinander bei Kopfbewegungen und Konturunschärfe bei Dunkelheit oder Nebel erreicht. Zum anderen durch das binokulare stereoskopische Sehen, das für das Tiefensehen verantwortlich ist.

Bei der räumlich-akustischen Darstellung ist wichtig, dass die Geräuschquelle, die die Audio-Information liefert, im dreidimensionalen Raum lokalisiert werden kann. Die Audio-Information wird entsprechend der Position des Betrachters in der virtuellen Szene gerendert.

#### 3.1.2.2 Qualität der Darstellung

Um eine virtuelle Umgebung sowohl visuell als auch akustisch realistisch darzustellen sind spezielle Techniken notwendig. Bei der visuellen Darstellung werden prinzipiell drei Rendering-Techniken unterschieden: *Scan-Conversion*, *Ray-Tracing* und *Radiosity*.

Bei der Scan-Conversion-Methode werden die Oberflächen von 3D-Objekten als eine Liste von Polygonen beschrieben. Jedes Polygon wird durch Eckpunkte und Kanten definiert. Entsprechend der virtuellen Position des Betrachters und seiner Blickrichtung werden alle Eckpunkte perspektivisch zum Betrachter transformiert. Ein Z-Buffer-Algorithmus findet die Oberflächen, die in der jeweiligen Perspektive gesehen werden können und stellt nur die sichtbaren Oberflächen dar. Die Farben einer Fläche werden nach einem Beleuchtungsmodell der jeweiligen Perspektive angepasst. Bei diesem Prozeß werden unter anderem Materialeigenschaften und virtuelle Punktlichtquellen ausgewertet. In Verbindung mit spezieller Graphik-Hardware ist die Scan-Conversion-Methode für Echtzeit-Anwendungen und somit auch für VR-Systeme geeignet. Spezial-Effekte, wie beispielsweise Lichtschatten und Transparenzen, Texturen auf Geometrien oder Nebel, können ohne großen Rechenaufwand integriert werden.

Bei der Strahlverfolgungsmethode Ray-Tracing können Spiegelungen, Transparenzen und Schatten berechnet werden. Im Gegensatz zu der Scan-Conversion-Methode, bei der jedes Polygon entsprechend der Betrachterposition projiziert wird, werden bei der Ray-Tracing-Methode Lichtstrahlen durch jedes Pixel des Bildschirms geschickt und jeweils der Schnittpunkt mit dem Objekt bestimmt, das als erstes von dem Strahl geschnitten wird. Bei reflektierenden oder transparenten Objekten werden zusätzliche Lichtstrahlen, die sogenannten Sekundärstrahlen, in Abhängigkeit von der virtuellen Lichtquelle rekursiv verfolgt. Die auf diese Weise entstandenen Bilder wirken sehr realistisch. Ein Nachteil dieser Methode ist jedoch der große Rechenaufwand. Aus diesem Grund wird die Ray-Tracing-Methode nur bei Standbildern eingesetzt.

Bei beiden Rendering-Methoden muss für jedes Bild die Beleuchtung aufwendig neu berechnet werden. Zur Lösung dieses Problems wird das sogenannte Radiosity-Verfahren eingesetzt, welches die Beleuchtung einer virtuellen Szene im Voraus berechnet und somit eine wiederholte Berechnung für jedes einzelne Bild erspart bleibt. Voraussetzung für den Einsatz des Radiosity-Verfahrens ist jedoch, dass eine Szene statisch ist, das heißt keine Objekte verschoben werden, die Lichtquelle nicht verändert wird und der Betrachter (sofern er in der Berechnung berücksichtigt wurde) nicht durch Positionsveränderungen die Lichtverhältnisse beeinflußt. Zur Zeit werden jedoch Verfahren entwickelt, die auch für dynamische Umgebungen die Beleuchtung berechnen können ([MUEL94]).

Die oben beschriebenen Verfahren (Scan-Conversion, Ray-Tracing, Radiosity) beziehen sich nur auf die visuelle Wahrnehmung. Für die Qualität der akustischen Darstellung sind Techniken aus dem Bereich der Sonifikation anzuwenden. Sonifikation bedeutet die rechnergestützte Transformation numerischer Daten in akustische Informationen, die für den Menschen wahrnehmbar und interpretierbar sind. Die bei der Sonifikation verwendeten Rendering-Verfahren sind in einigen Punkten dem visuellen Rendering ähnlich. Der Unterschied zwischen den Ausbreitungsgeschwindigkeiten von Schall und Licht spielt eine wichtige Rolle und erfordert entsprechende Algorithmen [ASTH95a]. Das heißt, um Akustik möglichst realistisch zu simulieren müssen physikalische Gesetze beachtet werden: Töne können gedämpft werden; die Frequenz ändert sich bei bewegter Geräuschquelle. Der Mensch kann räumlich hören, dass heißt er kann sehr genau bestimmen, wo sich die Geräuschquelle befindet. Unter Berücksichtigung dieser physikalischer Gegebenheiten werden für das Rendering von Akustik zwei Verfahren verwendet: das *Particle-Tracing-Verfahren* und das *Image-Source-Verfahren*.

- Particle-Tracing-Verfahren verfolgen die Ausbreitung von Schallpartikeln oder -strahlen innerhalb der virtuellen Szene. ähnlich dem Ray-Tracing-Verfahren beim visuellen Rendering ist das Particle-Tracing-Verfahren sehr rechenintensiv und für Echtzeitanwendungen nicht geeignet.
- Das Image-Source-Verfahren berechnet sogenannte virtuelle Schallquellen, die sich aus der Reflexion an den Wänden ergeben. Bei rechteckigen Räumen ist diese Berechnung sehr einfach und kann in Echtzeit durchgeführt werden.

Sowohl das Particle-Tracing-Verfahren als auch das Image-Source-Verfahren abstrahieren bzw. vereinfachen die physikalischen raumakustischen Eigenschaften. Die Ausbreitung eines Audio-Signals innerhalb einer virtuellen Szene wird so simuliert, dass Reflexionen, Absorptionen und Dispersionen von Objektoberflächen beachtet werden.

#### 3.1.2.3 Quantität der Darstellung

Die Quantität beschreibt in diesem Zusammenhang die dynamische Darstellung einer virtuellen Szene, sowie die dynamische Interaktion innerhalb einer solchen Szene. Die Berechnung jedes einzelnen Bildes soll dabei in einem fließenden Übergang resultieren. Dieser Echtzeitanspruch wird zum einen durch das Ausnutzen spezieller Hardware und zum anderen durch die Spezifikation einer entsprechenden Software-Architektur erreicht. Als Maßzahl kann man an dieser Stelle die Anzahl an (dargestellten) Dreiecken pro Sekunde nennen. Dreiecke können von der Grafik-Hardware auf Grund ihrer einfachen geometrischen Eigenschaften sehr schnell verarbeitet werden. Die Anzahl der in Echtzeit darstellbaren Dreiecke hängt natürlich auch von der verfügbaren Grafik-Hardware ab. Hardwarearchitekturen für Echtzeitgraphik werden u.a. in [AKEL93], [EYLE97] und [MONT97] vorgestellt.

Um die Komplexität einer virtuellen Szene zu erhöhen werden eine Reihe von Techniken angewendet, die dafür sorgen, dass Echtzeit gewährleistet werden kann ([REIN94]). Bei diesen Techniken handelt es sich um *Level-of-Detail*, *View Culling* und *Visible Volume Decomposition*. Voraussetzung für diese Techniken ist, dass die virtuelle Szene in einer hierarchischen Datenstruktur beschrieben ist, das heißt, dass Polygone zu Objekten zusammengefaßt und diese übergeordneten Teilszenen zugeordnet sind.

Bei der Level-of-Detail-Technik werden die Objekte zunächst in unterschiedlicher Komplexität mo-

delliert oder generiert ([KNOE98]). Während des Renderings wird dynamisch entschieden, wann welche Komplexitätsstufe dargestellt werden soll. Für diese Entscheidung gibt es eine Reihe Kriterien. So kann das System eine niedrigere Komplexität wählen wenn das Objekt sehr weit vom Betrachter entfernt ist und man somit kaum mehr als die Kontur erkennen kann. Entsprechend wird man eine sehr komplexe Darstellung wählen wenn sich das Objekt direkt vor dem Betrachter befindet. Damit das Umschalten zwischen den verschiedenen Komplexitätsstufen vom Anwender nicht bemerkt wird, kann entweder zwischen zwei Repräsentation geblendet oder gemorpht werden ([ASTH96]).

Das View-Culling-Verfahren beruht auf der Tatsache, dass der Betrachter normalerweise nicht die ganze Szene sehen kann. Das Verfahren bestimmt vor dem eigentlichen Rendering eine Liste von Objekten, die für die Darstellung überhaupt in Frage kommen. Dabei wird für alle Objekte überprüft ob sie sich im *Sichtkegel* des Betrachters befinden oder nicht. Die Objekte, die nicht ganz oder teilweise in diesem Kegel liegen (zum Beispiel Objekte, die sich hinter dem Betrachter befinden) haben keinen Einfluß auf das darzustellende Bild und können somit ignoriert werden.

Das Visible-Volume-Decomposition-Verfahren ([TELL92], [ZHAN98]) geht noch einen Schritt weiter. Hier werden nur die Objekte dargestellt, die sich im Sichtkegel befinden und auch tatsächlich sichtbar sind. Das heißt, dass ein Objekt, das vollständig durch ein anderes Objekt verdeckt wird, aus der Liste der Objekte entfernt wird, die gerendert werden müssen. Diese Verfahren sind auch unter dem Namen *Occlusion Culling* bekannt.

Die oben aufgeführten Verfahren beziehen sich nur auf die visuelle Präsentation. Im Bereich der akustischen Präsentation ist die Quantität der Darstellung abhängig von der Komplexität der Szene. Echtzeit-Simulation der Schallausbreitung wird nur in rechteckigen Räumen mit dem Image-Source-Verfahren erreicht. Bei komplexeren Szenen beschränkt sich die Darstellung auf das Abspielen digitalisierter Audio-Samples. Eine weiterführende Erläuterung findet sich in [ASTH95a].

Eine wichtige Forderung an die Präsentationskomponente ergibt sich aus der zeitlichen Auflösung des visuellen Systems des Menschen. Wie in Kapitel 3.3.1 erläutert, beträgt die zeitlichen Auflösung des Auges 100ms. Somit muss die Präsentationskomponente durch Anwendung der vorgestellten Techniken sicherstellen, dass zu jeder Zeit mindestens 10 Bilder/Sekunde dargestellt werden. Sollte der Fall eintreten, dass diese Framerate nicht eingehalten werden kann, wird der Anwender die Virtuelle Umgebung als unrealistisch empfinden, da das visuelle System keine kontinuierlichen Bewegungen mehr wahrnimmt, sondern eine Reihe von Einzelbildern.

### 3.1.3 Die Interaktionskomponente

Die Interaktions-Komponente bezieht sich in erster Linie auf die Interaktion im dreidimensionalen Raum. Zum einen bedeutet dies, dass die Interaktionsgeräte eine Manipulation im Raum zulassen müssen und zum anderen, dass Interaktionstechniken notwendig sind, um in der virtuellen Umgebung entsprechend den Anforderungen einer Anwendung zu interagieren. Bei sogenannten Desktop-Anwendungen, bei denen der Benutzer auf einen Bildschirm schaut, müssen 2D-Interaktionsgeräte, zum Beispiel die Maus, 3D-fähig gemacht werden. Beispielsweise kann dies durch eine virtuelle Hülle um ein Objekt realisiert werden, die mittels Maus 3D-Interaktionen möglich macht. Erst durch 3D-Interaktionsgeräte, wie zum Beispiel 6D-Tracker, Spaceball, Spacemouse, Flying Joystick oder Dataglove (siehe Kapitel 3.1.1.1), wird die intuitive Interaktion im Raum möglich.

Da die Interaktion vornehmlich mit dem Kopf und mit der Hand geschieht, wurde in VR-Systemen ein einheitliches Interaktionsparadigma festgelegt, das *flying carpet paradigm* ([ZACH96]), in dem der Anwender über die verschiedenen Eingabegeräte den *Blickpunkt* {point of view) aus dem die dargestellten Bilder berechnet werden und die virtuelle Hand steuern kann. So kann der Anwender durch die Szene navigieren, Objekte greifen etc. Welche Möglichkeiten hier zur Verfügung stehen und auf welche Art und Weise interagiert werden kann, wird in Kapitel 4 ausführlich beleuchtet.

### 3.1.4 Simulationskomponente

Die Simulations-Komponente eines VR-Systems stellt die semantische Ebene dar. Die virtuelle Umgebung wird in dieser Ebene als computer-internes Modell beschrieben. Im visuellen Bereich reicht dies von einfachen semantischen Informationen, wie beispielsweise Diagramme und Schaubilder, bis zur dreidimensionalen Geometrie. Für die Verarbeitung von komplexen Daten, wie beispielsweise Daten aus numerischen Berechnungen, die sowohl zeit- als auch raumvariant sein können, werden sogenannte Visualisierungssysteme notwendig. In einem weitergehenden Schritt der Simulationsebene soll es auch möglich sein, physikalische Gesetze und Materialeigenschaften zu simulieren. Diese Aufgabe erfüllen sogenannte physikalisch-basierte Systeme. Die natürlichen Vorgänge werden in diesen Systemen simuliert und die Simulationsergebnisse dargestellt. Bei diesen Systemen gibt es zwei verschiedene Möglichkeiten für die Anbindung der Simulation an ein VR-System. Die Simulation kann *offline* durchgeführt werden, dann wird das VR-System nur für die Visualisierung der Ergebnisse verwendet. Die Simulation kann aber auch *online* erfolgen. In diesem Fall besteht die Möglichkeit, dass die Interaktion des Benutzers direkten Einfluß auf die Simulation hat.

#### 3.1.4.1 Offline-Simulation

Für die Simulation von komplexen physikalischen Modellen beinhaltet ein VR-System die Möglichkeit der Aktivierung vorberechneter Animationen ([ASTH95]). Damit ist es möglich, auch sehr zeitaufwendige physikalische Simulationen vorzuberechnen und innerhalb einer virtuellen Umgebung als Animation darzustellen. Ein Beispiel für eine solche Animation stellt das Einfederverhalten eines Autos dar. Die einzige Interaktion besteht in diesem Beispiel in der Möglichkeit mittels virtueller Schalter die Geschwindigkeit der Animation zu regulieren. Auch wenn diese Art der Interaktion sehr eingeschränkt zu sein scheint bietet sie dem Anwender dennoch die Möglichkeit, die Ergebnisse von Simulationen direkt am Modell zu untersuchen. Weitergehende Konzepte werden u.a. in [KNOE00a] beschrieben.

Es gibt aber noch eine Zwischenstufe zur Online-Simulation. Dabei wird der größte Teil der Simulation offline durchgeführt. Ein Teil, der für die Darstellung notwendigen Berechnungen, wird dann zur Laufzeit vorgenommen. Ein Beispiel hierfür sind Applikationen, die Strömungsfelder in Virtuelle Umgebungen integrieren ([BRYS91]). Die Simulation ist dabei nicht auf Strömungsfelder, die um ein Objekt herumströmen beschränkt, sie kann auch auf Anwendungen erweitert werden, die mit einem zeitlich varianten Feld arbeiten (zum Beispiel bei der Visualisierung der Strömungsverhältnisse während der Verbrennung in einem Kolben).

#### 3.1.4.2 Online-Simulation

Die Online-Simulation natürlicher, physikalischer Vorgänge in einem VR-System steigert den reali-

stischen Eindruck der virtuellen Umgebung enorm. Die Interaktion mit der virtuellen Umgebung, beispielsweise das Verstellen der Gelenke einer Schreibtischlampe, kann in Echtzeit physikalisch korrekt simuliert werden und wirkt daher sehr realistisch. Auch das Fallen von Objekten (durch Einwirkung der Schwerkraft) zählt zu den physikalischen Simulationen, die in den Systemen integriert sind. Darüber hinaus werden kaum Simulationen integriert, da die Berechnungen zu lange dauern und damit die Echtzeitanforderung nicht mehr eingehalten werden kann. Zu den bekanntesten Beispielen für die Integration physikalisch basierter Simulation zählt das NPSNET ([MACE94]). In diesem System wurde jedoch nicht auf eine allgemeine Verwendbarkeit Wert gelegt, sondern nur auf die Lösung der spezifischen Probleme militärischer Simulationen (zum Beispiel die Flugbahn von Geschossen).

Ebenso wie im Bereich des Renderings, müssen für Online-Simulation spezielle Techniken angewendet werden, um die sehr aufwendigen Simulationen physikalischer Vorgänge in Echtzeit durchführen zu können. In [UNBE99] wird hierzu das Konzept des *Level-of-Simulation* entwickelt, mit dessen Hilfe bestehende Simulationsalgorithmen erweitert werden können, um sie in Virtuellen Umgebungen einzusetzen.

## 3.2 Geräteschnittstelle

In diesem Kapitel werden Faktoren vorgestellt, die die Wahl und das Design der Geräteschnittstelle beeinflussen.

### 3.2.1 Physiognomie

Die Physiognomie des Anwenders spielt bei der Wahl der Hardware eine sehr wichtige Rolle, denn z.B. ist es für Jetpiloten relativ unproblematisch ein schweres HMD über eine lange Zeit zu tragen, wohingegen ein Ingenieur nach kurzer Zeit sicherlich Genickschmerzen bekommen und sich sehr unwohl fühlen würde. Die Vitalität und Ausdauer des Anwenders beeinflusst neben dem oben erwähnten Ausgabegerät auch das Eingabegerät. Hier steht insbesondere das Gewicht im Vordergrund, denn muss das Gerät längere Zeit festgehalten werden, so kann dies bei längerer Benutzung zur Ermüdungen der Gliedmaßen führen.

Bei der Auswahl passender Eingabegeräte spielt neben den bereits angesprochenen ergonomischen Faktoren auch die bei der Bedienung angesprochenen Muskelgruppen eine entscheidende Rolle. Neurophysiologische Studien haben gezeigt, dass verschiedene Körperteile nicht proportional bzgl. Größe und Masse im Gehirn abgebildet werden.

Von besonderem Interesse ist hierbei, dass Hand und Finger sowohl im motorischen als auch im sensorischen Kortex im Gegensatz zu Handgelenk, Arm und Schulter überproportional repräsentiert sind. Daraus könnte geschlossen werden, dass die Einbeziehung der kleinen Muskelgruppen in einer wesentlich besseren Performance der Bedienung des Eingabegerätes resultiert.

Eine der ersten Studien, die diese Hypothese stützen, wurde von Gibbs ([GIBB62]) durchgeführt, die jedoch nur auf einem 1DOF-Test beruhte und nicht die Finger mit in die Betrachtung einbezog. In [LANG76] wurden verschiedene Muskelgruppen verglichen und festgestellt, dass die Verarbeitungsraten von Finger, Handgelenk und Arm stark differieren. Jeder Gelenk des Fingers, der Hand und des

Armes hat seine Funktion und vor allem auch seine Stärken. So sind die großen Muskelgruppen in Arm und Schulter für kraftvolle und große Bewegungen zuständig, wohingegen die kleineren Muskelgruppen in den Fingern wesentlich geschickter und genauer zu steuern sind.

In [ZHAI96] wurden auf dieser Basis Untersuchungen mit 6-DOF Eingabegeräten durchgeführt, die verschiedene Muskelgruppen in die Bedienung mit einbezogen. Zum einen ein Datenhandschuh, bei dem die Finger nicht zum Einsatz kamen und den *Fingerball*, der insbesondere mit den Fingern bedient werden konnte (siehe Abb. 3.2). Dieser ist dem 3Ball von Polhemus sehr ähnlich (POLH99b]).

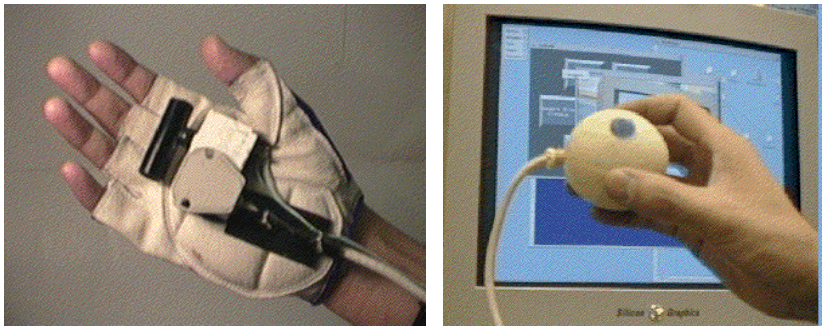

Abb. 3.2: Verwendete Geräte in der Studie [ZHAI96]

Die Untersuchungen basierten auf einer 6-DOF Positionierungsaufgabe, bei der ein Tetraeder, welches über das Eingabegerät gesteuert wurde in ein Zieltetraeder überführt werden mußte. Es zeigte sich, dass diese Aufgabe mit dem Fingerball signifikant schneller durchgeführt werden konnte, als mit dem Datenhandschuh. Somit zeigt auch diese Studie, dass das Einbeziehen der kleinen Muskelgruppen zu einer wesentlich besseren Bedienungsperformanz führt.

Weiterhin hat es sich auch gezeigt, dass die Trennung von Translation und Rotation auf zwei separate Eingabegeräte keinen entscheidenden Vorteil gegenüber einem Gerät bringt, mit dem alle sechs Freiheitsgrade gesteuert werden können ([ZHAI98]). Es ist jedoch zu beachten, dass exakte Rotationen wesentlich schwerer auszuführen sind, als Translationen. So konnte Zhai feststellen, dass die benötigte Zeit für eine exakte Rotation bis zu 580% über der Zeit für eine exakte Translation lag. Weiterhin hat Zhai in [ZHAI97] festgestellt, dass auch die Translationen entlang der drei Grundachsen zu unterschiedlichen Ergebnissen führen. So ist insbesondere die Translation entlang der Z-Achse (Tiefe) wesentlich schwieriger zu handhaben, als Translationen entlang der X- und Y-Achse. Die hier erzielten Ergebnisse sind jedoch um den Faktor 10 besser als die in [MASS89] beschriebenen. Hier wurde im Gegensatz zur Studie von Zhai keine Stereodarstellung gewählt. Auch die Rotation um die drei Hauptachsen weist unterschiedliche Ergebnisse auf, die sich jedoch mit denen der Translation dekken. Hier konnten die besten Ergebnisse bei einer Rotation um die Z-Achse erzielt werden, bei der keinerlei Verschiebung entlang der Z-Achse auftritt.

### 3.2.2 Beidhändige Interaktion

Der Mensch ist es aufgrund der täglichen Arbeit gewöhnt, mit zwei Händen zu agieren. So ist es nur natürlich mit dem Computer auch mit beiden Händen zu kommunizieren.

Untersuchungen auf diesem Gebiet (siehe [GUIA87], [HINC97]) haben gezeigt, dass durch die Einbeziehung beider Hände die Interaktionsgeschwindigkeit gesteigert werden kann, da beide Hände gemeinsam, aber auch unabhängig voneinander agieren. Aufgrund der Tatsache, dass die eine Hand eine Referenz („frame of reference“) für die andere Hand bildet, können hierbei auch Tätigkeiten beschleunigt werden, die auf den ersten Blick nur eine Hand benötigen. Als Beispiel sei hier handschriftliches Schreiben genannt, welches durch Verwendung beider Hände, die nicht stiftführende Hand fixiert das Papier, um 20% schneller von statten geht, als bei Verwendung nur einer Hand. Kommen beide Hände zum Einsatz, so kann die eine Hand das Papier entsprechend manipulieren, bei nur einer Hand ist das Papier fixiert. Schon an diesem Beispiel erkennt man, dass bei vielen Aufgaben im täglichen Leben beide Hände nicht die gleiche, sondern unterschiedliche Tätigkeiten ausführen. Man unterscheidet hierbei zwischen der *dominanten Hand* und der *nicht-dominanten Hand*. Die dominante Hand wird prinzipiell für Aufgaben, die eine höhere Präzision verlangen, verwendet oder wenn nur eine Hand bei einer Aktion zum Einsatz kommt. Die nicht-dominante Hand bildet den Referenzrahmen für die dominante Hand und wird im Zusammenspiel für Aufgaben, die nur eine grobe Positionierung benötigen, eingesetzt. Bei den meisten Menschen ist die rechte Hand die dominante Hand und die linke Hand die nicht dominante Hand.

Aufgrund der unterschiedlichen Tätigkeiten der beiden Hände ist es sinnvoll, unterschiedliche Eingabegeräte für die linke und die rechte Hand einzusetzten. Hierbei muss jedoch beachtet werden, dass diese Geräte sowohl für die linke, als auch die rechte Hand verwendet werden können, um sowohl Linkshändern, als auch Rechtshändern die Benutzung des Systems zu ermöglichen. Aus diesem Grund werden im folgenden ausschließlich die Termini „dominante Hand“ und „nicht-dominante Hand“ verwendet.

## 3.3 Softwareschnittstelle

Der Aufbau der Softwareschnittstelle wird durch verschiedene Faktoren beeinflusst. Zum einen durch menschliche Wahrnehmung, auf die in Kapitel 3.3.1 eingegangen wird. Da eine Benutzeroberfläche nicht nur Daten präsentiert, sondern auch Eingaben vom Benutzer erwartet, muss auch auf die Art und Weise der menschlichen Interaktion eingegangen werden. Diese wird in Kapitel 3.3.2 anhand des Aktionsmodels von Norman ([NORM90]) erläutert, um daraus Richtlinien für den Entwurf intuitiver Benutzerschnittstellen abzuleiten. In den letzten Jahrzehnten wurden im Bereich der Entwicklung von 2D-Benutzeroberflächen sehr viele Ergebnisse erzielt. Da diese jedoch nicht in ihrer Gesamtheit, sondern nur teilweise auf VR-Anwendungen übertragen werden können, werden im Falle von Parallelen diese Ergebnisse an entsprechender Stelle referenziert werden.

### 3.3.1 Menschliche Wahrnehmung

Der Mensch verfügt über die verschiedene Wahrnehmungssysteme. Für die Mensch-Computer-Interaktion sind dies vor allem das visuelle System (Sehsinn), das auditive System (Gehörsinn), der Tastsinn und die statischen Sinne (Muskelkraft, Lage und Bewegung) von Bedeutung ([WAND93]). Auf die statischen Sinne wurde bereits weiter oben eingegangen und im Zusammenhang mit der Interaktion in VR spielt der Tastsinn heutzutage kaum eine Rolle. Deswegen wird an dieser Stelle nur auf das visuelle und das auditive System eingegangen.

Das visuelle System wird durch das menschliche Auge repräsentiert. Auf der *Retina* werden Intensität, Wellenlänge und räumliche Verteilung des Lichts verarbeitet. Ist das Auge auf einen bestimmten Punkt fixiert, so nennt man dessen Projektion auf die Retina *Fovea*. Scharfes Sehen ist nur in einem kleinen Bereich um die Fovea möglich. Dieser Bereich hat circa einen Durchmesser von 4°. Punkte außerhalb dieses Bereichs werden nur unscharf wahrgenommen, sodas z.B. das Lesen eines Buches in diesem Bereich nicht mehr möglich ist. Hier spricht man von peripherem Sehen.

Um nun jedoch Punkte außerhalb der Fovea scharf erkennen zu können, können durch Augenbewegungen, den sog. Sakkaden, diese fixiert werden. Die Dauer einer Sakkade liegt bei etwa 30ms, wobei die Gesamtdauer einer Sakkade mit anschließender Fixation im Mittel 230ms beträgt ([CARD83]). Liegt jedoch der zu fixierende Punkte um mehr als 30° von der Fovea entfernt, so kann die Fixierung nicht mehr alleine durch eine Sakkade erreicht, sondern muss durch eine zusätzliche Kopfbewegung unterstützt werden.

Die visuell aufgenommenen Reize werden kurzzeitig noch in einem sog. *ikonischen Speicher* gehalten, wobei hier die Informationen nicht begrifflich oder symbolisch abgelegt sind, sondern direkt die Reizwerte. Die Speicherdauer beträgt im Mittel 200ms. Danach sind die Daten nicht sofort gelöscht, sondern verblassen mit der Zeit.

Die zeitliche Auflösung des visuellen Systems liegt bei 100ms, d.h. werden verschiedene Reize innerhalb dieser Zeitspanne wahrgenommen, so werden diese als eine Einheit betrachtet. Werden zwei sehr ähnliche Reize im zeitlichen Abstand von unter 100ms an verschiedenen Orten aufgenommen, dann wird eine Bewegung, die sog. Scheinbewegung, des Reizes von der einen zu der anderen Position wahrgenommen.

Das Erkennen von Zeichen oder Figuren basiert auf den Daten, die sich im ikonischen Speicher befinden. Die Auswahl der Inhalte des Speichers für die Erkennung werden automatisch oder über kognitive Prozesse des Kurzzeitgedächtnis (KZG) gesteuert, wobei hier in der Regel physikalische Merkmale, wie Helligkeit, Farbe, Intensität, Kontrast etc. ausschlaggebend sind. Welche Bereiche nun zur beachteten Figur werden, ist Teil der gestaltpsychologischen Prinzipien der Figur-Grund Unterscheidung. Diese werden z.B. in [PRINZ90] ausführlich beschrieben.

Die eigentliche Zeichenerkennung wird basierend auf dem Wissen des Langzeitgedächtnises (LZG) und den im KZG präsenten Erwartungsinformationen durchgeführt. Erwartungsinformaionen ergeben sich u.a. aus den verfügbaren Informationen über andere Objekte oder Sachverhalte. So kann der Prozess des Erkennens zweigeteilt gesehen werden. Zum einen das Erkennen auf Basis der Erwartungen (konzeptgesteuerte Verarbeitung oder auch top down processing) und zum anderen der visuellen Merkmale des Objektes (datengesteuerte Verarbeitung oder auch bottom up processing).

Die Geschwindigkeit des Erkennens hängt somit zum einen von den verfügbaren Kontextinformationen und zum anderen von der „Datenqualität“ im ikonischen Speicher und den Figurmerkmalen ab. So dauert z.B. der Prozess des visuellen Zeichenerkennens für bekannte und nicht zu komplexe Objekte, wie Buchstaben, Ziffern und einfache Wörter etwa 100ms. Bildhafte Darstellungen werden schneller erkannt, was die Verwendung von Ikonen befürworten würde. Für visuell ähnliche Objekte wird eine wesentlich längere Erkennungszeit benötigt.

Ein sehr wichtiger Prozess während der Arbeit an einem Bildschirm ist die visuelle Suche nach einem bestimmten Objekt aus einer Menge von Objekten. Dies können z.B. Wörter, Ikonen oder andere Figuren sein. Ein Beispiel für eine solche Suche ist die Menüauswahl, bei der der Anwender in einer Liste von dargebotenen Kommandos die gewünschte Option entdecken muss.

Das Entdecken kann entweder automatisch oder kontrolliert erfolgen. Bei der automatischen Suche wird die visuelle Aufmerksamkeit sofort auf das zu findende Objekt gelenkt, wobei die Suchzeit unabhängig von der Anzahl der dargebotenen Objekte ist. Automatisches Entdecken ist möglich, wenn das zu suchende Objekt sich durch ein auffälliges Merkmal von den anderen Objekten abhebt. Dies sind vor allem:

- eine besondere Farbe oder Intensität
- ein besonderes Texturmerkmal
- eine besondere Orientierung
- ein besonderes Formmerkmal, z.B. ein rundes Objekt in einer Menge von quadratischen Objekten
- Blinken oder Bewegung
- ein hervorstechendes semantisches Merkmal, z.B. ein Buchstabe unter eine Menge von Zahlen

Kontrollierte visuelle Suche wird dann erforderlich, wenn das Objekt nicht automatisch gefunden werden kann. Hierbei müssen dann die einzelnen Objekte seriell verarbeitet werden, wodurch die mittlere Suchzeit linear mit der Suchmenge ansteigt. Experimente ([PRINZ86]) habe jedoch auch gezeigt, dass die Suche nicht unbedingt immer seriell ist, sondern auch automatisches Erkennen beinhalten kann. Somit sind diese zwei Suchprozesse eher als Extrema anzusehen und werden bei der visuellen Suche eher kombiniert anzutreffen sein.

Bei der *Szenenwahrnehmung* stehen die dargebotenen Objekte in einer bestimmten räumlichen Anordnung zueinander. Die visuelle Verarbeitung einer Szene mit bekannten Objekten geschieht sehr schnell. So genügt für kleinere Szenen mit 7 Objekten oft bereits eine einzige Fixation und eine Darbietungszeit von 100 bis 200ms. Das Erfassen einer Szene mit einer Fixation bedeutet jedoch nicht, dass damit der Prozess des Erkennens abgeschlossen ist, denn die Szene kann so komplex sein, dass sie aufgrund der Gegebenheiten des visuellen Systems mit einer Fixation nicht erfasst werden kann. Jedoch werden die Informationen, die durch das Erkennen der Szene vorliegen, für die Steuerung der folgenden Sakkaden verwendet. Somit sollten komplexe Sachverhalte auf dem Bildschirm möglichst in einer räumlich-geometrischen Darstellung dargeboten werden, um die Szenenwahrnehmung ausnützen zu können.

Über das Ohr können auditive Daten aufgenommen werden, die ähnlich dem ikonischen Speicher, in einem auditiven Register, dem Echospeicher, abgelegt werden. Die sprachlichen Einheiten sind hier als phonemischer Code repräsentiert. Für einzelne Buchstaben beträgt die Speicherdauer etwa 1500ms. Diese im Vergleich zum ikonischen Speicher wesentlich längere Dauer läßt sich dadurch erklären, dass sich das gesprochene Wort über eine vergleichsweise große Zeitspanne erstreckt. Die Inhalte des Echospeichers werden symbolisch kodiert und im KZG abgelegt. Erst hier ist ein akustischer Reiz erkannt.

Die Bedeutung des Sprechens und Hörens sind für die Mensch-Computer-Interaktion vergleichswei-

se beschränkt, der Bildschirm ist das dominierende Ausgabemedium jedes Systems. Jedoch wurden bereits mit synthetischer Sprachausgabe und sprecherunabhängiger Spracherkennung einige wesentliche Erfolge erzielt, die den stärkeren Einsatz der Akustik als Teil der Benutzerschnittstelle immer weiter vorantreiben. Akustische Signale sind jedoch auf jeden Fall sehr gut geeignet, die Aufmerksamkeit des Anwenders zu erlangen.

### 3.3.2 Aktionsmodell

Ein wichtiger Aspekt beim Entwickeln einer Interaktionstechnik ist auch das Verständnis darüber, wie Menschen vorgehen, um ein Ziel zu erreichen. Denn kennt man die einzelnen Komponenten dieses Vorgangs, so kann man diese gezielt optimieren. Die Grundidee ist sehr einfach. Zu Beginn steht das Ziel, welches erreicht werden soll. Anschließend werden Aktionen ausgeführt, um das Ziel zu erreichen. Zum Schluss wird geprüft, ob das Ziel erreicht wurde. Aufbauend auf dieser Idee entwickelte [NORM90] die 7 Stufen einer Aktion:

1. Ziel festlegen (*forming the goal*)
2. Absicht (Intention) festlegen (*forming the intention*)
3. Aktion(en) festlegen (*specifing an action*)
4. Ausführen (*executing the action*)
5. Antwort aufnehmen (*perceive the state of the world*)
6. Verstehen (*interpreting the state of the world*)
7. Bewertung (*evaluating the outcome*)

Zuerst wird ein Ziel definiert, wobei diese Definition zumeist unpräzise ist, z.B. „Ich brauche mehr Licht“. Die Intention definiert, was zu tun ist, um das Ziel zu erreichen, z.B. „drücke den Lichtschalter“. Da jedoch auch die Intention nicht ausreicht, um das Ziel zu erreichen muss diese wiederum in eine Aktionssequenz zerlegt werden, also in die physikalischen Kommandos, um das Ziel zu erreichen. Sobald die Aktionssequenz definiert ist, kann sie ausgeführt werden. Anschließend wird das Ergebnis der Aktionssequenz aufgenommen, verstanden und dann mit dem eigentlichen Ziel verglichen.

Das vorgestellte Aktionsmodell ist rein konzeptioneller Natur, denn Menschen agieren nicht unbedingt immer genau nach diesem 7 Stufen Plan. Insbesondere werden meist mehrere Aktionen benötigt, um ein Ziel zu erreichen und somit das Ausführen einer Aktion und das Interpretieren des Ergebnisses dieser Aktion zyklisch wiederholt, bis das eigentliche Ziel erreicht ist. Auch kann durch Änderungen der Umstände die Intention und damit die Aktionen oder sogar das Ziel verändert werden und es muss neu geplant werden. Auch kann es passieren, dass die erwarteten Ergebnisse nicht mit den echten Ergebnissen übereinstimmen. Dann muss der Anwender auf diese „Fehler“ entsprechend reagieren können. Das bedeutet aber auch, dass er die Möglichkeit dazu hat.

Aus den 7 Stufen einer Aktion können nun einige Designrichtlinien direkt abgeleitet werden, wobei es jedoch keine feste Zuordnung zu einer bestimmten Stufe gibt und auch die einzelnen Richtlinien sehr stark miteinander verwoben sind:

**Sichtbarkeit:** Durch hinschauen kann der Benutzer erkennen, in welchem Zustand das System ist und welche Aktionen möglich sind.

**Konzeptionelles Modell:** Ein gutes konzeptionelles Modell erlaubt es uns das Resultat einer Aktion vorauszusagen. Insbesondere wenn ein System das falsche tut muss man verstehen, warum dem so ist, um es ändern zu können. Jedoch ist das Vermitteln eines guten konzeptionellen Modells nicht sehr einfach, denn es ist nicht gegeben, dass das Modell, dass der Designer als Grundlage für das erstellen des Systems verwendet hat auch das selbe ist, auf dem der Anwender seine Überlegungen basiert. Denn Designer und Anwender kommunizieren nicht direkt (wobei es hier auch zu Verständigungsschwierigkeiten kommen könnte), sondern über das Gerät, das System. Das bedeutet, dass das System das richtige Modell vermitteln sollte, so dass sich das gedankliche Modell des Anwenders mit dem des Designers deckt (siehe Abb. 3.3)

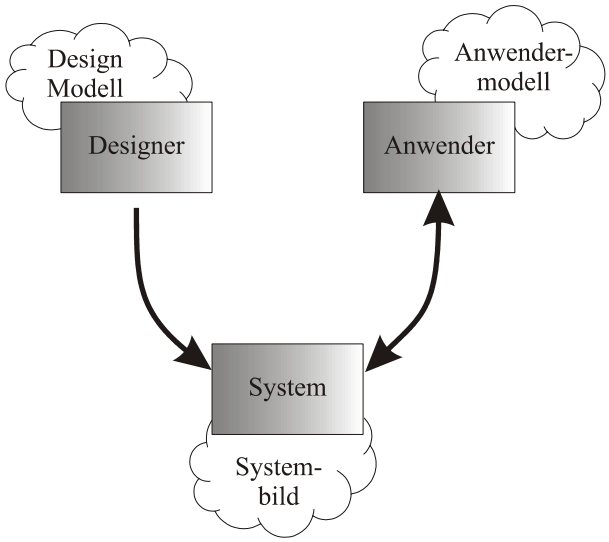


Abb. 3.3: Kommunikation zwischen Designer und Anwender (nach [NORM86])

Ein Beispiel für unterschiedliche Gedankenmodelle ist die Steuerung einer VR Szene mit Hilfe der Spacemouse. Da VR-Entwickler unter „Bewegung in einer Szene“ sehr oft „Fliegen“ verstehen, werden sie eine Spacemouse so integrieren, dass der Anwender die Kamera bewegt (*eyeball in hand*). Sollen nun CAD-Anwender das System mit der Spacemouse verwenden ergeben sich sofort Probleme, da sie es aus der CAD-Welt gewohnt sind, mit der Spacemouse die Szene zu bewegen (*scene in hand*) und nicht die Kamera. Hier hat das Systembild das Gedankenmodell der VR-Entwickler nicht transportieren können.
Ein gutes Gedankenmodell hilft dem Anwender auch, die zur Verfügung stehenden Aktionen leichter zu lernen und sich später daran zu erinnern, denn das Langzeitgedächtnis kann Informationen in einem vorgegeben Kontext leichter aufnehmen.

**Mapping:** Die Beziehung zwischen der Aktion und dem Ergebnis ist offensichtlich. Nehmen wir als Beispiel ein getracktes Objekt und einen Bildschirm, auf dem ein virtuelles Abbild dieses Objekts zu sehen ist. Wird das reale Objekt nun nach links bewegt, so erwartet der Anwender auch, dass das virtuelle Objekt nach links bewegt wird. Hier können wir auch von einem natürlichen Mapping sprechen, da hier natürliche Analogien aufgegriffen wurden. Auch die kulturellen Standards und das Hintergrundwissen von Personen spielt wie im Spacemouse-Beispiel zu den Gedankenmodellen gezeigt eine wichtige Rolle.

**Feedback:** Der Anwender erhält kontinuierlich eine Rückmeldung über das Resultat seiner Aktionen. Nehmen wir hier als Beispiel die Selektion eines Objektes mittels Mauscursor. Wird eine Taste gedrückt, so wird das Objekt unter dem Mauscursor selektiert. Drückt der Anwender nun diese Taste und das System gibt keine Rückmeldung, wird der Anwender keine Möglichkeit haben seine Aktion zu bewerten. Das Objekt kann nun entweder selektiert sein und es ist normal, dass das System keine Rückmeldung gibt oder das System hat den Tastendruck nicht mitbekommen. Ändert sich jedoch nach dem Tastendruck die Darstellung des Objektes, so weiß der Anwender sofort, dass das System den Tastendruck verstanden hat und das Objekt selektiert wurde.

**Struktur der Aktionen:** Es ist wichtig, dass der Anwender sich Aktionen leicht merken kann und das ihr Verhalten möglichst einfach oder logisch ist. Komplexe Funktionen mit vielen Nebenwirkungen sind für Benutzer nur sehr schwer handhabbar. Dann fällt dem Anwender es auch sehr schwer, Aktionen einer Intention zuzuordnen, um ein bestimmtes Ziel zu erreichen.

### 3.3.3 Kommunikation und Stellenwert der verwendeten Medien

Ein wichtiger Aspekt des Design Reviews ist die Kommunikation der Anwender untereinander, denn sie beeinflusst die Qualität eines Reviews maßgeblich. Im klassischen Design Review auf Basis physikalischer Prototypen wurde sie aufgrund der verwendeten Medien sehr gut unterstützt. Der Paradigmenwechsel hin zu digitalen Modellen wirft jedoch die Frage auf, ob dies dann auch weiterhin der Fall ist.

Um dieser Frage nachgehen zu können, werden in diesem Kapitel die Mechanismen, die die Kommunikation in der Gruppe ausmachen, herausgearbeitet. Basis bildet hierbei das klassische Arbeiten mit „physikalischen“ Prototypen. In Kapitel 5 werden diese Erkenntnisse wieder aufgegriffen und auf das Anwendungsszenario VR-Design Review übertragen.

In [PERR98] werden zwei Fallstudien beschrieben, die sich auf den Bereich der Koordination von kooperativen Entwicklungsarbeiten konzentrieren und dabei auch auf den Design Review näher eingehen. Eine besondere Aufmerksamkeit wurde dabei der Rolle der zwischenmenschlichen Kommunikation, sowie der dabei verwendeten Ausdrucksmittel ("Designartefakte"), um Ideen zu transportieren und die Kommunikation zu unterstützen, gewidmet. Lt. den Autoren werden in einer zunehmenden Anzahl von Berichten diese beiden Aspekte als immanent wichtig für den Designbereich angesehen. Werden hier neue Technologien eingesetzt, so muss sehr stark darauf geachtet werden, dass diese auch optimal unterstützt werden. Technologien, die auf diese Randbedingung nicht eingehen, können somit den Designprozess behindern und die Kreativität der Gruppe negativ beeinflussen. Es ist also von äußerster Wichtigkeit, die tägliche Praxis zu kennen und erst dann auf Basis dieser Erkenntnisse eine neue Technologie zur Unterstützung zu entwickeln.

#### 3.3.3.1 Erste Designstudie: Konstruktion einer Pumpe

Die erste Studie wurde in einem kleinen Designbüro für Präzisionspumpen durchgeführt. Die Aufgabe bestand darin eine Pumpe zu konstruieren, die effizienter und dabei aus weniger Bauteilen besteht als die bereits existierenden Pumpen. Die Hauptentwicklungsgruppe bestand aus vier Personen, die sich meistens in dem selben Raum aufhielten. Somit war jede Person der Gruppe über die Arbeiten der Anderen jederzeit informiert und während der Untersuchung wurden eine Vielzahl von spontanen

Fragen und Kommentaren beobachtet. Zusätzlich wurden jedoch auch regelmäßige Designtreffen abgehalten. Aufgrund verschiedener Fragestellungen, die sich im Rahmen der Entwicklung der Pumpe ergaben, mußten auch einige Personen der Entwicklungsgruppe mit beteiligten Firmen kommunizieren.

Großflächige Plots wurden vornehmlich als Diskussionsgrundlage verwendet, wobei die Teilnehmer der Diskussionen Anmerkungen und Änderungen sofort auf den Ausdrucken vermerkten. Die Größe der Plots (A1 Format) wurde dabei als sehr vorteilhaft angesehen, denn das am Computerbildschirm ansonsten notwendige rein- und rauszoomen der Szene entfiel dadurch und das Bauteil war in seiner Gesamtheit immer sichtbar. Um mit beteiligten Firmen zu kommunizieren wurde der Ausdruck verkleinert und via Fax versendet, wobei aufgrund des durch die Übertragung auftretenden Qualitätsverlustes Informationen verloren gingen und es weiterer Erklärungen bedurfte, bis die Informationen vollständig bei der Zulieferfirma ankamen.

#### 3.3.3.2 Zweite Designstudie: Statik und Architektur

Die zweite Studie wurde im Rahmen eines wesentlich größeren Projektes durchgeführt, an dem verschiedene Firmen direkt beteiligt waren und welches die Konstruktion eines gesamten Bürokomplexes zur Aufgabe hatte. In dieser Studie wurden die Projektteams zur Planung der Elektrifizierung (vier Ingenieure) und zur Berechnung der Statik (sieben Ingenieure) besonders beobachtet. Aufgrund der sehr hohen Komplexität des Vorhabens wurde das Gesamtproblem in kleinere Problemstellungen unterteilt. Jedes der Designteams saß in einem gemeinsamen Büro, so dass die Kommunikationswege extrem kurz waren und Treffen spontan einberufen werden konnten. Ebenso wie in der ersten Designstudie wurden Plots für die Kommunikation eingesetzt, wobei hier mehrere hundert Ausdrucke zum Einsatz kamen, die gleichzeitig aktuell waren und regelmäßig auf den neuesten Stand gebracht wurden.

Während eines Designtreffen wurden diese Zeichnungen verwendet und zusätzlich mit Anmerkungen und Änderungen versehen, die sich aus der Diskussion ergaben. Diese mussten am Ende des Design Reviews wieder in das aktuelle Design übernommen werden. Aufgrund der hohen Anzahl an Plots und Problemstellungen, wurde sehr viel Zeit darauf verwandt, das verwendete Material konsistent und auf dem neuesten Stand zu halten. Dies hat sich als das größte Problem des Projektes herausgestellt.

Kommen bei einem Review Personen mit unterschiedlichem Wissenstand und Fähigkeiten zusammen, so müssen die verwendeten Medien und Inhalte für alle Anwesenden in gleichem Maße verständlich sein. So kann man nicht davon ausgehen, dass ein Kunde ohne große Architekturerfahrung auf Basis von 2D-Ansichten sich eine realistische 3D-Sicht vorstellen kann. In diesem Projekt wurden deswegen auch 3D-Pappmodelle erstellt, an denen dann Problemstellungen besser erläutert werden konnten.

#### 3.3.3.3 Resultat der Studien

In beiden Studien war der eigentliche Designprozess sehr stark durch die Erzeugung und Modifikation der verschiedenen Designartefakte (Zeichnungen, Skizzen, Prototypen) geprägt. Sie drückten den Zustand des Designs zu einem bestimmten Zeitpunkt im Evolutionsprozess aus. Das gesamte De-

signwissen war jedoch nicht nur in diesen Artefakten "gespeichert", sondern auch in einer begleitenden Dokumentation. Die Artefakte waren weiterhin mit Metadaten versehen, z.B. Datum, Name des Erstellers, Positionierung im Gesamtprojekt etc.

Im Rahmen eines Designtreffens stellten sie die Diskussionsgrundlage dar, auf deren Basis Änderungen, Erweiterungen und Verbesserungen des Designs durchgeführt wurden. Hierzu wurden die Anmerkungen direkt auf den "Artefakten" aufgebracht, die dann ggf. in das eigentliche Design übernommen und eingearbeitet wurden. Wichtig ist jedoch hervorzuheben, dass die Designmeetings in beiden Studien die Aufgabe hatten, Fehler im bestehenden Design zu finden, Lösungen zu prototypisieren und das weitere Vorgehen festzulegen. Die eigentliche Ausarbeitung der Änderungen und Erweiterungen, sowie die komplette Erstellung neuer Komponenten war nie Bestandteil der Treffen, sondern wurde von einzelnen verantwortlichen Ingenieuren durchgeführt. Zusätzlich dienten die Artefakte auch außerhalb "offizieller" Designtreffen zur Kommunikation zwischen den Ingenieuren, die gemeinsam an einer bestimmten Problemstellung arbeiteten.

Die Studien unterstützen somit die folgenden Schlussfolgerungen:

- Die Entwicklung eines Design entsteht nicht nur aus der Intelligenz und dem Wissen von Individuen, sondern wird durch die Gesamtheit eines Teams entwickelt („Das Ganze ist mehr als die Summe seiner Einzelteile“).
- Design ist kein linearer Prozess mit festgelegten Designschritten
- Design ist ein Prozess gemeinsamen Lernens und Arbeitens, in dem Artefakte als Mediatoren und Kommunikationsmittel verwendet werden
- Verstehen von Designartefakten hängt stark von dem Wissen und der Erfahrung der jeweiligen Person ab.

Die Autoren sehen in den Ergebnissen ihrer Studien wichtige Erkenntnisse für die Entwicklung von speziellen Technologien zur Unterstützung der Designentstehung. Sie stehen jedoch dem Einsatz solcher Technologien sehr kritisch gegenüber, da sie der Meinung vertreten, dass nicht alle Szenarien und Bereiche mit Hilfe der Computertechnik verbessert werden können. Sie leiten daraus die folgende Schlussfolgerungen und Anforderungen an eine einzusetzende Technologie ab:

- Die wichtigsten Komponenten für den Designprozess sind zum einen die Interaktion zwischen verschiedenen Designern, die nicht zwangsläufig im selben Bereich arbeiten müssen und zum anderen Designartefakte, die als Mediatoren zwischen den Designern fungieren und als Diskussionsgrundlage dienen. Ein Großteil dieser Interaktionen findet in Gesprächen von Angesicht zu Angesicht statt und lt. den Autoren der Studie ist es nicht ersichtlich wie dies durch technische Hilfsmittel verbessert werden könnte. .In jedem Fall müßten die Hilfsmittel sehr ausgefeilt sein und sämtliche Kommunikationskanäle (Gesten, Stimme, Blickkontakt, Papier oder Whiteboard) übertragen können.
- In diesen Designstudien war eine große Zahl der beteiligten Personen nicht vor Ort, sondern in Zulieferfirmen. Eine Technologie muss somit den problemlosen Austausch der Daten zwischen den verschiedenen Standorten gewährleisten.
- Die Designartefakte waren durchgängig mit Metadaten versehen, um u.a. ihren Ursprung und die Einordnung in das Gesamtprojekt zu kommunizieren. Dies stellte sich für die Synchronisation

und Diskussion als absolut notwendig heraus und muss von einer Technologie ebenfalls unterstützt werden.

Bezogen auf den digitalen Design Review im Automobilbau bedeutet der Wechsel von PMU zu DMU vor allem einen Wechsel der Designartefakte oder Mediatoren. Dies bedeutet zwangsläufig auch eine Veränderung in der Interaktion und Kommunikation. Es ist somit immanent wichtig, vor allem darauf zu achten, dass die Vorteile, die reale Prototypen bieten, nicht verloren gehen und die Kommunikation/Interaktion zwischen den Teilnehmern der Begutachtung auch mit den neuen Mediatoren sinnvoll unterstützt wird (siehe Kapitel 5).

Zwar werden in der Studie die Vorteile von Zeichnungen, Skizzen und Modellen sehr stark hervorgehoben, jedoch die Nachteile kaum beleuchtet. So war die Problemstellung in der zweiten Studie wesentlich komplexer als in der ersten und resultierte in einer schieren Unmenge an Zeichnungen, die von den Mitarbeitern aktuell und konsistent gehalten werden mussten. Die Autoren gaben auch zu, dass dies das größte Problem des Projektes war. Das bedeutet, dass mit steigender Komplexität die Verwendung von Zeichnungen ein ernsthaftes Problem darstellen kann und einen erheblichen Zeitaufwand mit sich bringt. Dies deckt sich auch mit den Erfahrungen aus der Automobilindustrie und ist mit einer der Hauptgründe für den Verzicht auf physikalischer Modelle und eine Hinwendung zu einer integrierten Produktentwicklung auf Basis virtueller Modelle.

In den Arbeiten von [LECA00] und [BLAN98] wurde u.a. auch die Kommunikation und Interaktion während eines digitalen Design Review untersucht. Hierbei haben sich vier Fragestellungen ergeben, die mit den zur Zeit zur Verfügung stehenden Technologien nur ungenügend gelöst sind:

- *Wer kontrolliert das System?* Ist es eine spezielle Person oder können auch andere Teilnehmer sehr einfach die Kontrolle übernehmen, um z.B. auf bestimmte Problembereiche hinzuweisen? Die Problematik besteht hierbei, dass die Beteiligten nicht auf einer gleichen Stufe stehen und unterschiedliche Möglichkeiten besitzen, sich auszudrücken.
- *Wieviele Zeiger gibt es?* Gibt es nur einen, wie z.B. der Mauszeiger oder gibt es mehrere, sodass verschiedene Personen gleichzeitig auf Objekte deuten können? Wie sollen sie dargestellt werden?
- *Wie kann Nachverfolgung realisiert werden?* Bei einem typischen physikalischen Prototypen können die Beteiligten auch nach einem Design Review zurückkommen und sich das Bauteil ansehen und Ideen ausprobieren und überprüfen. Bei einem digitalen Prototypen ist dies nicht so einfach möglich, da dies Wissen über die Bedienung des System voraussetzt.
- *Gleichzeitiges Untersuchen unterschiedlicher Problemstellungen:* Während mehrere Personen einen physikalischen Prototypen gleichzeitig aus verschiedenen Perspektiven betrachten können, um z.B. an verschiedenen Problemstellungen zu arbeiten, können die Beteiligten einen digitalen Prototypen nur aus einer gemeinsamen Ansicht betrachten, die von einem Anwender gesteuert wird.

Diese Fragestellungen werden neben den oben dargelegten Erkenntnissen ebenfalls bei der Konzeptionierung einer intuitiven Benutzerschnittstelle für den digitalen Design Review beachtet.

### 3.3.4 Logische Interaktionsaufgaben

Ebenso wie alle physikalischen Eingabegeräte auf eine definierte Menge logischer Eingabegeräte ([FELG95]) abgebildet werden können, kann jede Form der Interaktion mit einer Benutzeroberfläche auf sog. Basisinteraktionsaufgaben oder auch *basic interaction tasks* (BIT) zurückgeführt werden ([FOLEY90]). Hierbei wird zwischen den folgenden BITs unterschieden:

- Positionierung
- Selektion
- Quantifizierung
- Text
- Rotation

Über diese BITs kann somit die gesamte Interaktion mit einer Benutzeroberfläche beschrieben werden. Die konkrete Umsetzung eines BIT wird als *Interaktionstechnik* bezeichnet. Da BITs atomar sind, können sie zur Lösung höherwertiger und komplexerer Aufgaben zusammengesetzt werden, sog. *composite interaction tasks*, kurz CIT.

Im Aktionsmodell (siehe Kapitel 3.3.2) wurde dargelegt, dass das menschliche Handeln dadurch geprägt ist, dass komplexe Aktionen schrittweise in einfachere Aktionen zerlegt werden, die dann ab einer bestimmten Komplexitätsebene direkt ausgeführt werden können. Das BIT-Modell bildet damit eine ideale Grundlage, um dieses Aktionsmodell umsetzen zu können. Das Zerlegen in atomare Aktionen kann für einen Anwender dadurch vereinfacht werden, in dem er die Optionen, die ihm das System bietet, kennt. Deswegen ist es sinnvoll, die Menge an atomaren Aktionen relativ klein zu halten, d.h. im optimalen Fall jeweils nur eine definierte Interaktionstechnik pro BIT anzubieten.

Auf technischer Ebene hat diese Abstraktion den Vorteil, dass es sehr leicht möglich ist, Interaktionstechniken auszutauschen, ohne die Funktionen des Systems daran anpassen zu müssen. Abb. 3.4 zeigt das Zusammenspiel der Komponenten Geräte, Interaktionstechnik und Funktion und der BITs als gemeinsame Schnittstelle dieser Komponenten. Auf die Konzeptionierung der BIT-Schnittstelle wird in Kapitel 7.3 näher eingegangen.

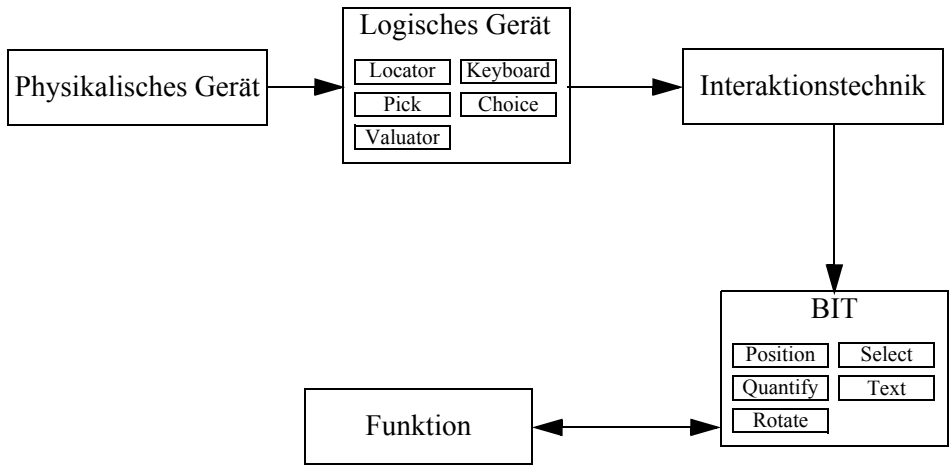

Abb. 3.4: BIT-Pipeline

Im folgenden werden nun die einzelnen BITs näher beschrieben und auf ihre Randbedingungen, sowie mögliche Beeinflussung durch andere Komponenten eingegangen.

**Positionierung:** Mittels dieser BIT werden Objekte positioniert, in dem ihnen eine neue Position zugewiesen wird. Bei der Umsetzung ist zu beachten, ob das verwendete Eingabegerät / Interaktionstechnik die notwendigen Freiheitsgrade unterstützt und auch die notwendige Genauigkeit zur Positionierung liefert. Weiterhin sollte untersucht werden, ob freie Positionierung möglich sein soll oder ein Gitter verwendet soll, um nur bestimmte Positionen zuzulassen. Insbesondere bei der Positionierung im 3D Raum mittels eines 3DOF/6DOF Gerät ist zu beachten, dass Anwender die verschiedenen Freiheitsgrade mit unterschiedlicher Priorität und Präzision behandeln (Kapitel 3.2). In der VR ist *Navigation* ein wichtiger Bestandteil der Interaktion. Da der Anwender mittels der Navigation die Kamera in der Szene bewegt und die Kamera ein Objekt der Szene ist, wird die Navigation ebenfalls über diesen BIT abgebildet.

**Selektion:** Über diese BIT werden Elemente aus einer Auswahlliste ausgesucht. Da sich Selektion sowohl auf Objektauswahl, als auch auf Menüauswahl oder allgemeiner auf Kommandoauswahl beziehen kann, unterscheidet man zwischen variabler Selektionsliste (Objekte) und relativ konstanter Selektionsliste (Kommandos).
Objektselektion kann generell auf 2 Arten umgesetzt werden: Selektion über Objektname oder durch zeigen auf das Objekt. Hierbei spielen Objektentfernung, Objektgröße und Richtung zum Objekt [POUP97] eine große Rolle. Bei der Menüauswahl sind Aspekte wie visuelle Repräsentation, Größe und Form der Darstellung, hierarchische Strukturen, Platzierung und die Anzeige der aktuellen Auswahl zu beachten.

**Quantifizierung:** Über diese BIT wird ein Wert zwischen einem gegebenen Minimum und Maximum eingegeben. Eine typische Umsetzung ist z.B. der Scrollbalken in Windows. Auch hier ist wie beim Positionieren ein wichtiger Gesichtspunkt die erforderliche Genauigkeit der Werteingabe.

**Text:** Diese BIT erlaubt es dem Anwender einen Text einzugeben, wobei die Applikation diesem Text keine spezielle Bedeutung zuordnet. Das Eingeben von Kommandos ist somit kein Text-BIT. Die Umsetzung dieser BIT wird maßgeblich vom verwendeten Eingabegerät bestimmt.

**Rotation:** In 3D benötigt man neben der Positionierung auch die Rotation, wobei in VR die Rotation als Orientierung im 3D-Raum zu verstehen ist. Aspekte der Umsetzung ist die Lage des Rotationszentrums, der Rotationswinkel, die Objektentfernung und -größe.

Diese Klassifizierung wurde von [FOLEY90] basierend auf Desktopanwendungen vorgenommen, jedoch kann sie auch direkt auf VR Anwendungen übertragen werden (s.a. [POUP97]). Der eigentliche Unterschied zwischen Desktopanwendung und VR liegt im Prinzip nur in der Umsetzung der BITs, den Interaktionstechniken.

## 3.4 Zusammenfassung

In diesem Kapitel wurden wichtige Grundlagen für die immersive und intuitive Interaktion dargelegt. Diese sind für die Umsetzung der mittels des 4W-Modells herausgearbeiteten Anforderungen in In-

teraktionstechniken und Benutzerschnittstelle entscheidend. Neben Aspekten, die die Auswahl der Gerätetechnologie beeinflussen, wurde dabei auch auf die Art und Weise der menschlichen Interaktion eingegangen.

Weiterhin wurde das für den weiteren Verlauf der Arbeit entscheidende Konzept der logischen Interaktionsaufgaben vorgestellt. Diese geben eine abstrakte Strukturierung hinsichtlich der Interaktionstechniken vor, die im folgenden Kapitel aufgegriffen wird und auch in Kapitel 7 ihre Anwendung findet.

# 4 Konzeption einer intuitiven Benutzerschnittstelle

In den vorhergegangenen Kapiteln wurden die Grundlagen gelegt, auf deren Basis nun eine intuitive Benutzerschnittstelle entwickelt werden kann. Es wurden sowohl die allg. Anforderungen an die Oberfläche beschrieben, sowie die speziellen der verschiedenen Applikationsbereiche. Weiterhin sollten nun auch die prinzipiellen Grundlagen zum Design intuitiver Benutzerschnittstellen bekannt sein (Kapitel 3).

In diesem Kapitel werden zuerst die für den Design Review auf Basis digitaler Prototypen geeigneten Ein- und Ausgabegeräte festgelegt. Anschließend wird das Konzept der *basic interaction tasks* erläutert, die eine logische Abstraktion zu den eigentlichen Interaktionstechniken darstellen. Aufbauend werden im Anschluß verschiedenste Interaktiontechniken vorgestellt, diskutiert und bzgl. der Anforderungen bewertet. Die besprochenen Techniken sind zum einen im Rahmen dieser Arbeit entworfen und zum anderen aus der Literatur übernommen worden. Hierbei werden auch die im Rahmen dieser Arbeit entwickelten Verbesserungen der bestehenden Ansätzen erwähnt.

## 4.1 Auswahl geeigneter Ein- und Ausgabegeräte

Wie im Anforderungsmodell (Kapitel 2.1.3) beschrieben, ist die Wahl der Ein- und Ausgabegeräte entscheidend für den Entwurf und die Realisierung der Softwareschnittstelle.

### 4.1.1 Eingabegeräte

Die gängigsten Geräteklassen wurden in Kapitel 3.1.2 beschrieben. Die mit Hilfe des 4W-Modells herausgearbeiteten Randbedingungen können nun zur Bestimmung der für die Auswahl des Eingabegerätes entscheidenden Anforderungen herangezogen werden. Wie im Abhängigkeitsmodell gezeigt, ist eine eindeutige Zuordnung von Anforderung zu einer Kategorie des Modells nicht immer gegeben. Es muss vielmehr darauf geachtet werden, dass die Abhängigkeiten auch beachtet werden. Die Anforderungen sind im folgenden aufgelistet, die zugehörigen Modellkategorien sind hier exemplarisch mit angegeben:

- **Gewicht:** Da ein Design Review mehrere Stunden andauern kann, darf der Arm aufgrund zu hohen Gewichtes nicht ermüden *(Anwender, Ablauf)*
- **Ergonomie:** Hierfür gilt ähnliches, wie für das Gewicht des Gerätes *(Anwender, Ablauf)*
- **Robustheit:** Installation, die speziell für den Design Review eingerichtet wurden, werden von mehreren Teams geteilt und so ist damit zu rechnen, dass die Geräte im täglichen Einsatz sind. *(Ablauf)*
- **Präzision:** Mit dem Gerät muss es möglich sein, auch feinfühlige Bewegungen zu machen. So müssen z.B. bei der Untersuchung von Simulationsdatensätzen die Datensonden exakt plaziert werden können. *(Inhalt, Intention)*
- **Hygiene:** Da die Geräte von verschiedenen Personen benutzt werden und auch während eines Reviews herumgereicht werden, muss es zum einen möglich sein, diese schnell an- und abzulegen und zum anderen muss ein gewisses Maß an Hygiene erfüllt sein *(Ablauf)*.
- **Eignung für Links- und Rechtshänder:** Eine Tatsache, die sehr oft vergessen wird, ist dass ein Softwaresystem auch von Linkshändern bedient werden können muss. Das bedeutet, dass die verwendeten Geräte sowohl für die linke, als auch rechte Hand verwendet werden können.

*(Anwender)*

- **Triggermöglichkeiten:** Über jedes Eingabegerät muss der Anwender neben Positions- und Orientierungsbestimmung auch Aktionen auslösen können. Dies kann bei einem Handschuh über Gesten und bei einem 3D-Joystick über Tasten geschehen. *(Intention)*

Wie bereits in Kapitel 3.2.2 dargelegt, ist die beidhändige Interaktion mit einem Softwaresystem in bestimmten Bereichen sehr sinnvoll und kann die Arbeitsgeschwindigkeit verbessern. Aufgrund der unterschiedlichen Fähigkeiten der linken und rechten Hand (dominant, nicht dominant), sollten jeweils angepasste Eingabegeräte eingesetzt werden (s.u.).

Im folgenden Abschnitt werden nun die verschiedenen in Kapitel 3.1.1.1 vorgestellten Eingabegeräte auf ihre Verwendbarkeit im Kontext Design Review bewertet.

Die *Spacemouse* verfügt zwar über eine hohe Anzahl an Triggermöglichkeiten, kann herumgereicht werden und ist robust und präzise, jedoch muss sie aufgrund ihres Gewichtes fest installiert werden und kann nicht auf Dauer in der Hand gehalten werden. Auch ist die Spacemouse nicht sofort intuitiv zu bedienen und die Vorstellung über die Bedienung differiert je nach Hintergrundwissen (Kapitel 3.3.2, konzeptionelles Modell). Steht die Spacemouse nicht auf einem Tisch, so müssen zur Bedienung beide Hände verwendet werden. Eine effektive Ausnutzung der Vorteile beidhändiger Interaktion ist somit nicht möglich.

Der *Datenhandschuh* hingegen ist sehr leicht, ergonomisch und prinzipiell intuitiv zu verwenden. Durch Greifgesten kann z.B. ein virtuelles Objekt gegriffen werden. Die entscheidenden Nachteile sind jedoch die mangelnde Hygiene und auch das Weitergeben an andere Personen gestaltet sich hierbei als sehr aufwendig.Weiterhin ist er für die gestellten Anforderungen nicht robust genug.

Die Klasse der *3D-Joysticks* kann im Prinzip alle hier aufgezählten Anforderungen erfüllen, jedoch ist hier zwischen Geräten zu unterscheiden, die wie ein Spiele-Joystick (*power grasp*) oder wie ein Stift (*precision grasp*) gehalten werden. Die Begriffe power grasp und precision grasp werden erstmals bei [MACK94] erwähnt und stehen zum einen für sichere und kraftvolle Bewegung und zum anderen für Genauigkeit und Beweglichkeit. Wie bereits in Kapitel 3.2 erläutert ist das Einbeziehen der kleinen Muskelgruppen (precision grasp) als sehr sinnvoll anzusehen, da hierdurch die Performanz bei der Bedienung des Gerätes verbessert wird.

Die wohl interessanteste kommerziell erhältliche Lösung aus der Klasse der stiftähnlichen Geräte ist der *Polhemus Stylus* ([POLH99]). Der Stylus ist eine Art Stift, der mit einem Trackingsensor und einem Taster ausgestatet ist. Über den Trackingsensor können Position und Orientierung abgefragt und über den Taster eine Aktion ausgelöst werden. Der wohl größte Nachteil dieses Gerätes ist die Anzahl der Triggermöglichkeiten. Ein Taster ist für komplexe Systeme zu wenig[1]. In der Praxis haben sich drei bis vier Tasten bewährt. Bei einer höheren Anzahl könnten zwar mehr Aktionen gleichzeitig ausgelöst werden, die Bedienung ist jedoch dann auch wesentlich komplizierter. Ein interessantes Konzept hierzu wurde in [ZELE02] vorgestellt. Hier werden Tri-State Taster verwendet, wodurch sich

1. Dies ist z.B. auch bei der Verwendung von Computern der Marke Apple zu beobachten, die über eine Ein-Tasten-Maus bedient werden. Komplexe Programme belegen die Maustaste mit verschiedenen Funktionen, die über zusätzliche Tasten auf der Tastatur dann ausgewählt werden. Dies bedeutet einen relativ hohen Lernaufwand, um das System richtig bedienen zu können.

neue Möglichkeiten in der Interaktion ergeben.

Da es neben dem Stylus kein kommerzielles Gerät gibt, welches die Anforderungen besser erfüllen könnte, wurde im Rahmen dieser Arbeit der *Flystick* entwickelt. Er ist im Gegensatz zum Stylus mit drei Tasten ausgestatet, wiegt ca. 30g ohne Kabel und kann mit einem Trackingsensor ausgerüstet werden. Die Tasten sind im vorderen Bereich in einer Reihe angeordnet, wobei der mittlere Taster sich in Form und Größe von den anderen Beiden unterscheidet. Somit kann der Anwender die drei Köpfe sehr leicht unterscheiden. Weiterhin kann er auf verschiedene Arten in der Hand gehalten werden. Siehe hierzu Abb. 4.1

Abb. 4.1: Der Flystick: power grasp (links) und precision grasp (rechts)

Eingabegeräte, die einem Stift ähneln können auch mit einem Tablett gekoppelt werden, welches dann dem Anwender ein haptisches Feedback vermitteln kann ([SHAW93]). Siehe hierzu auch Kapitel 4.4.2.3.

Wie bereits zu Beginn des Kapitels dargelegt, kann die beidhändige Interaktion eine Verbesserung der Bedienungsperformanz bedeuten und soll aus diesem Grund auch Eingang in den Entwurf der Benutzerschnittstelle finden. Bis jetzt wurden Geräte vorgestellt, die für die dominante Hand geeignet sind und auch „Einhändig“ verwendet werden können. Dabei hat sich gezeigt, dass stiftähnliche Geräte wie der Stylus und Flystick für die dominante Hand sehr gut geeignet sind. Prinzipiell könnte für die nicht-dominante Hand das selbe Gerät verwendet werden. Dies hat jedoch den entscheidenden Nachteil, dass es dadurch sehr leicht zu Verwechslungen kommen könnte und deswegen ist die Verwendung unterschiedlicher Geräte als notwendig anzusehen. Es ist zu beachten, dass diese Geräte jeweils für die linke als auch für die rechte Hand verwendet werden können, um sowohl Linkshänder, als auch Rechtshänder die Benutzung des Systems zu ermöglichen.

Werden nun zwei Geräte verwendet, so könnte über das zweite Gerät auch der Nachteil der geringen Anzahl an Knöpfen des Polhemus Stylus kompensiert werden, indem das zweite Gerät mit einer entsprechenden Anzahl an Triggermöglichkeiten, welches z.B. über die nicht-dominante Hand bedient wird, ausgestattet wird. Hierbei muss jedoch beachtet werden, dass die Aufteilung einer Aktion auf „Positionierung mit Stylus“ und „Trigger mit zweitem Gerät“ keine negativen Auswirkungen auf die Performanz der Bedienung hat. Hiervon kann jedoch prinzipiell nicht ausgegangen werden. So wurde in [OHAR87] dargelegt, dass es nur minimale Performanzunterschiede zwischen der Verwendung ei-

nes 6DOF-Gerät und zwei 3DOF-Geräten gibt, wobei hierbei jeweils Translation und Rotation getrennt sind. Siehe hierzu auch Kapitel 3.2. Bei der oben erwähnten Aufspaltung zwischen Trigger und Positionierung könnte sogar eine höhere Positionierungspräzision erreicht werden, denn durch das Drücken eines Knopfes auf dem Stylus/Flystick kann dieser etwas verwackeln.

In [ZHAI96] wurde der *Fingerball* vorgestellt, der zum einen auch die kleineren Muskelgruppen mit einbezieht und zum anderen sich wesentlich von einem Stylus oder Flystick unterscheidet. Im Rahmen dieser Arbeit wird der Puck verwendet, der mit mehreren Triggermöglichkeiten versehen ist.

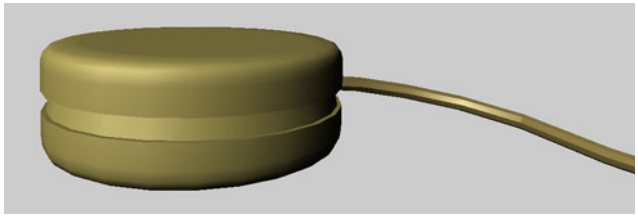

Abb. 4.2: Der Puck (nicht dominante Hand)

### 4.1.2 Ausgabegeräte

Im folgenden wird nun auf die für das Einsatzgebiet geeigneten Ausgabegeräte eingegangen. Diese müssen für den Einsatz im Design Review folgende Anforderungen erfüllen:

- **Kommunikation:** Da ein Design Review in einer größeren Gruppe durchgeführt wird, müssen alle Anwesenden die Szene sehen können und dabei kommunizieren können
- **Realgröße:** Die Darstellung der virtuellen Prototypen sollte auch in einer 1:1 Darstellung möglich sein, um den Experten die Möglichkeit zu geben, Größenverhältnisse abzuschätzen.

Kopfgebundene Ausgabegeräte (HMD) sollten hier nicht zum Einsatz kommen, da sie meistens sehr schwer und unergonomisch sind. Weiterhin beeinflussen sie die Kommunikation zwischen den beteiligten Personen äußerst negativ, denn die Kanäle der nonverbalen Kommunikation werden sehr stark eingeschränkt. So ist es mit HMDs nicht möglich die Augen des Gegenüber zu sehen, ein äußerst wichtiger Kommunikationskanal entfällt. Diese Problematik besteht selbst bei see-through HMDs, die z.B. bei dem System *Studierstube* verwendet werden ([FUHR99]), da sie zwar die Möglichkeit bieten, aus der Brille herauszusehen, jedoch nicht hineinzusehen. Obwohl Design Reviews in einer sehr technischen Umgebung durchgeführt werden, sollte der „Modefaktor“ nicht unbeachtet bleiben. Da HMD’s nicht sehr modisch sind und teilweise seltsam anmuten, ist es vielen Personen eher peinlich und unangenehm, diese Geräte zu tragen und es kann zu einer unterschwelligen Ablehnung des Gesamtsystems führen[1].

Somit können als Ausgabegeräte nur feststehende Projektionen zum Einsatz kommen bei denen der Betrachter „getracked“ wird, um auch eine realistische stereoskopische Darstellung zu ermöglichen. Hierbei können wir prinzipiell zwischen drei Kategorien unterscheiden:

1. In Gesprächen mit Endanwender hat sich auch gezeigt, dass dieser Modefaktor selbst bei Shutterbrillen zum tragen kommt

- Monitor
- Einseitenprojektionen:
  - Großbildprojektionen (Powerwall)
  - Virtual Table (Responsive Workbench)
- Mehrseitenprojektionen:
  - Holobench
  - CAVE

Um Objekte in Realgröße darstellen zu können ist der Monitor aufgrund seiner Größe nicht geeignet. Auch ist er für eine größere Gruppe von Personen nicht geeignet. Ein- und Mehrseitenprojektionen können in unterschiedlichen Größen konstruiert werden, so dass virtuelle Objekte prinzipiell in einer 1:1 Darstellung angezeigt werden können. CAVE-Installationen mit drei oder mehr Seiten sind aufgrund der würfelförmigen Anordnung der Projektionsseiten in der darzustellenden Objektgröße nicht beschränkt. Aufgrund der Größe der Projektion können auch mehrere Personen die virtuelle Szenerie betrachten.

Der wohl entscheidende Unterschied zwischen diesen Systemen ist die Anordnung der Projektionsfläche, bzw. -flächen. Bei einem Virtual Table ist die Projektionsfläche horizontal oder leicht geneigt, bei Powerwall und CAVE Systemen vertikal. Der Vorteil des Virtual Tables ist die Schreibtischmetapher, so dass hier bekannte Verhaltensweisen und Interaktionen übernommen werden können. So können Objekte auf dem Tisch plaziert (z.B.metaDESK in [ULLM97]) oder als Unterlage zum Zeichnen verwendet werden (z.B. [EHNE01]). Durch die horizontale Anordnung ist jedoch der stereoskopische Effekt einer solchen Projektion eingeschränkt: Objekte können nicht vor dem Betrachter im Raum schweben. Hier sind die Ausgabegeräte mit vertikalen Projektionswänden im Vorteil.

Wünschenswert wäre eine Kombination dieser Projektionssysteme, die entweder als ein Gerät realisiert sind, wie z.B. Holospace ([TAN98]). Dieses Gerät hat jedoch den Nachteil, dass die Projektionsflächen relativ klein und auch nicht als Unterlage zum zeichnen und schreiben geeignet sind. Eine andere Möglichkeit wären zwei Geräte, die simultan benutzt werden können (z.B. CAVE und Virtual Table).

### 4.1.3 Geräteschnittstelle - Generische Sicht

Um dem Anforderungsmodell aus Kapitel 2.1 gerecht zu werden, wurden im vorherigen Kapitel die Ein- und Ausgabegeräte evaluiert und für den Anwendungstypus Design Review festgelegt. Diese Geräte erfüllen die gestellten Anforderungen aufgrund bestimmter Attribute, die sie aufweisen. Aufgabe in diesem Kapitel ist, es diese Attribute herauszuarbeiten und dadurch eine stärkere Allgemeingültigkeit der im folgenden vorgestellten Interaktionstechniken zu erreichen, eine Konzentration auf ein bestimmtes „Hardwareprodukt“ kann dadurch vermieden werden.

Im folgenden sind dies:

- **Eingabegerät für die dominante Hand:**
  - Stiftähnliches Gerät, um auch die Finger in die Interaktion mit einzubeziehen und Skizzieren zu ermöglichen
  - Drei oder vier verschiedene Trigger, es sei denn, das Eingabegerät für die andere Hand kann

ein Fehlen dieser Möglichkeiten kompensieren

- **Eingabegerät für die nicht-dominante Hand:**
  - Das Eingabegerät muss sich hinsichtlich der Form von dem Eingabegerät für die dominanten Hand unterscheiden.- Trotzdem ist jedoch wichtig, dass die kleinen Muskelgruppen in die Bedienung mit einbezogen werden (Puck, Ball).
  - Triggermöglichkeiten

- **Ausgabegeräte:**
  - Um das Skizzieren zu ermöglichen ein Ausgabegerät mit einer horizontale Projektionsfläche (z.B. Virtual Table)
  - Für optimalen stereoskopischen Eindruck eine vertikale Projektionsfläche (z.B. Powerwall/ CAVE)

## 4.2 Interaktionstechniken als Umsetzung der basic interaction tasks

In den folgenden Kapiteln werden nun für die verschiedenen BITs unterschiedliche Interaktionstechniken vorgestellt, sowohl aus der Literatur bekannte, als auch im Rahmen dieser Arbeit entwickelte Techniken. Da in der VR zumeist nur 6DOF Geräte zum Einsatz kommen und auch die Interaktionstechniken sehr selten zwischen Positionierung und Rotation unterscheiden, werden diese beiden BITs in einem gemeinsamen Kapitel behandelt.

### 4.2.1 Eindeutige BIT-Zuordnung

Nicht alle Techniken lassen sich eindeutig einem BIT zuordnen, da sie inhärent mehrere BITs abdekken (CIT). Dies ist z.B. beim natürlichen Greifen eines Objektes der Fall. Sobald der Kontakt zwischen Hand und Objekt hergestellt und ein Event ausgelöst wurde, ist das Objekt selektiert und kann sofort bewegt werden. Bei Techniken, die aus mehreren BITs zusammengesetzt sind (wie hier im Beispiel aus Selektion und Positionierung), werden diese in dem Kapitel vorgestellt und besprochen, welches der eigentlichen Zielrichtung der Technik entspricht (im Beispiel Positionierung).

### 4.2.2 Positionierung von Hilfsobjekten

Viele Interaktionstechniken arbeiten mit geometrischen Hilfsobjekten, die auf verschiedene Arten in der virtuellen Szene positioniert oder vom Anwender gesteuert werden können. Bei der folgenden Beschreibung der verschiedenen Möglichkeiten, wird auf das Flying carpet Paradigma zurückgegriffen ([ZACH96], siehe folgende Abbildung).

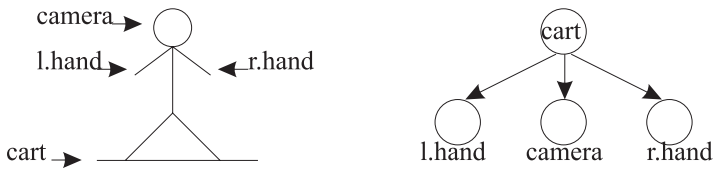

Abb. 4.3: Flying carpet Paradigma (Schema und Szenengraph)

Bei den Beschreibungen der einzelnen Interaktionstechniken wird davon ausgegangen, dass die reale Hand des Anwender die jeweilige virtuelle Repräsentation kontrolliert. Entsprechendes gilt für das Kopftracking und die virtuelle Kamera.

Folgende Positionierungsmöglichkeiten ergeben sich mittels dieses Paradigmas:

- **Fixed location:** Hier wird das Hilfsobjekt an einer festen Stelle im Weltkoordinatensystem positioniert.
- **Head relative:** Hier wird das Hilfsobjekt relativ zur Kopfposition und Blickrichtung positioniert. Bei Kopfbewegung wird die Position aktiv angepasst. Somit bleibt das Hilfsobjekt immer im Blick und kann, falls durch die Szenerie verdeckt, durch eine einfache Kopfbewegung an eine andere Position versetzt werden. Diese Positionierungmethode kann sehr leicht dadurch realisiert werden, indem das Hilfsobjekt unter den Szenengraphknoten „camera“ gehängt wird.
- **Cart relative:** In diesem Modus erscheint das Hilfsobjekt in fester Relation zur Position und Orientierung des Benutzers (Kopplung an „cart“-Knoten im Szenengraph. Damit haben Änderungen der Kameraposition/Orientierung keine Auswirkungen auf die Verortung des Hilfsobjektes. Bei Navigation durch die Szene (hier wird üblicherweise der cart bewegt), folgt das Hilfobjekt entsprechend.
- **Cursor relative:** Das Hilfsobjekt wird relativ zu einem Cursor (z.B. die virtuelle Hand) positioniert. Cursor relative kann auch mit cart relative gekoppelt werden, indem über ersteren Modus das Objekt in Relation zum Anwender positioniert wird, dann an dieser Stelle verbleibt und sich bei Navigation durch die Szene mitbewegt.

## 4.3 BIT Positionierung und Rotation

In der Virtuellen Realität wird in den meisten Fällen Positionierung und Orientierung nicht separat behandelt. Dies liegt u.a. darin begründet, dass in den meisten Fällen direkt mit 6DOF Geräten gearbeitet wird. So kann zu jedem Zeitpunkt sowohl die Position, als auch die Orientierung vom Anwender angegeben werden. Auch hat [ZHAI98] in diesem Zusammenhang untersucht, ob die Verwendung von zwei 3DOF Geräten gegenüber einem einzigen 6DOF Gerät Vorteile in der Interaktion bringt. Er kam jedoch zum Schluss, dass ein 6DOF Gerät den zwei 3DOF Geräten sogar leicht überlegen ist.

Positionierung in VR bedeutet nicht nur das Positionieren von Objekten in einer Szene, sondern auch das Positionieren der Kamera, sprich die Navigation in einer virtuellen Welt. Da die relevanten Interaktionstechniken oft sehr stark auf eines dieser zwei Einsatzgebiete ausgerichtet sind, wird im folgenden zwischen Navigation und Objektpositionierung unterschieden.

### 4.3.1 Anforderungen

Ein wichtiges zu beachtendes Kriterium bei Techniken für diese BITs ist die *Ausdehnung und Aufbau der Szene* (4W-Modell: Inhalt). Kleine, kompakte Szenerien (z.B. ein Motorinnenraum) müssen anders behandelt werden, als sehr weitläufige Szenerien (z.B. ein Flugzeug oder eine Montagestraße).

Die freie Wahl des *Rotationszentrums* ist sehr entscheidend. Dies mag zwar auf den ersten Blick als unwichtig erscheinen, ist jedoch für gut benutzbare Interaktion sehr entscheidend. Dies läßt sich am Beispiel eines großen Presswerkzeuges am ehesten erläutern. Angenommen das Werkzeug hat eine Kantenlänge von 4m und üblichweise untersuchen die Anwender ein Detail an der Außenhaut des Werkzeugs. Um das Detail von allen Seiten genauer betrachten zu können, muss das Werkzeug gedreht werden. Befindet sich der Rotationspunkt im Zentrum der Bounding Box des Objektes, so ist er mindestens 2m vom Anwender entfernt. Selbst eine kleine Drehung um 10° würde einen Versatz des betrachteten Details um ca. 35cm bedeuten. Ist der Rotationspunkt jedoch sehr nahe an dem zu betrachtenden Detail, so ist der Versatz entsprechend klein (z.B. nur 1,76 cm bei einer Distanz von 10 cm und 10° Drehwinkel).

### 4.3.2 Ansätze zur Navigation

In den folgenden Unterkapiteln werden einige Ansätze zur Navigation in Virtuellen Umgebungen vorgestellt.

#### 4.3.2.1 Pointfly mit Headtracking

Die wohl ursprünglichste und bekannteste Technik der Navigation ist der Pointfly-Modus, der insbesondere mit getracktem HMD und getracktem Datenhandschuh seine Anwendung findet. Durch Zeigen in eine bestimmte Richtung wird die Kamera in die Zeigerichtung transliert. Die Rotation der Kamera wird über den getrackten Kopf gesteuert. Dies beeinflußt jedoch nicht die Translationsrichtung, da diese nicht relativ zur Kopforientierung ausgewertet wird. Bei dieser Technik sind somit Rotation und Translation voneinander getrennt. Über zwei verschiedene Gesten mit dem Handschuh kann vorwärts, bzw. rückwärts bzgl. der Zeigerichtung geflogen werden. Die Fluggeschwindigkeit kann dabei entweder fix sein oder über eines der Fingergelenke (z.B. Daumen) variabel eingestellt werden. Diese Flugtechnik ist relativ intuitiv, setzt aber ein kopfgebundenes Displaysystem voraus, welches für den Design Review nicht gegeben ist.

#### 4.3.2.2 Pointfly ohne Headtracking

Bei nicht kopfgebundene Displaysysteme muss die Translationsrichtung der Kamera unbedingt von der Kamerarotation beeinflußt werden, da ansonsten die Zeigerichtung nicht mit der sichtbaren Bewegungsrichtung übereinstimmt. Würde man z.B. vor einer Projektionswand stehen, zum vorwärts fliegen auf die Wand deuten und anschließend die Kamera um 90° um die Hochachse drehen, so fliegt man nicht auf die Szene zu, sondern an ihr vorbei. Die Zeigerichtung stimmt somit nicht mehr mit dem visuellen Feedback überein und entspricht auch sicherlich nicht den Erwartungen des Anwenders (siehe auch Kapitel 3.3.2).

Um dieses Verhalten zu vermeiden, darf die Flugrichtung nicht nur durch die Zeigerichtung beeinflußt werden, sondern durch die Relation zwischen Zeigerichtung und Kameraorientierung. Die Relation wird hierbei nicht als absoluter Wert betrachtet, sondern als relativer Wert, der kontinuierlich auf die aktuelle Flugrichtung aufgerechnet wird. Die aktuelle Flugrichtung wird somit nur dann beibehalten, wenn Zeigerichtung mit der Kameraorientierung exakt übereinstimmt. Diese Methodik ist mit dem Lenkrad eines Autos vergleichbar: Solange das Lenkrad nicht in der Grundstellung ist, fährt das Auto eine Kurve.

Um jedoch ein solches Verfahren für VR-Anwendungen gut benutzbar zu machen, darf bei Änderungen der Zeigerichtung, die kleiner als ein Schwellwert $s_u$ sind, die Flugrichtung nicht beeinflußt werden. Damit wird das Zittern von Hand und Trackingsystem unterdrückt. Auslenkungen größer als $s_u$, werden über eine exponentielle Funktion in relative Rotationen umgerechnet und kontinuierlich auf die aktuelle Kamerarotation aufaddiert. Durch die exponentielle Funktion dreht die Kamera immer schneller, je größer die Auslenkung ist. Eine obere Schwelle $s_o$ gibt die maximale Rotationsgeschwindigkeit vor. Siehe hierzu auch Abb. 4.4.

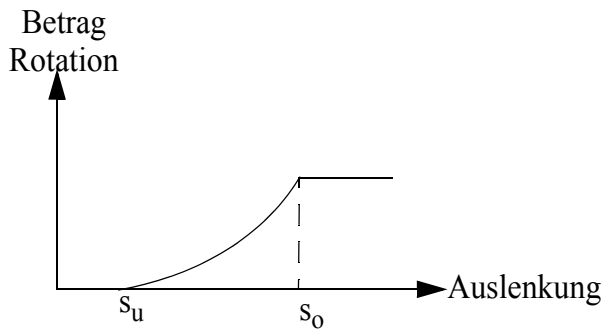


Abb. 4.4: Pointfly: Abbildung der Auslenkung auf die Rotation

Wie in Fällen anderen Anwendungen, so auch im Design Review sollte die Kamera nicht um die Sehachse rotiert sein (tilt), da ansonsten die dargestellte Szenen schief steht. Somit sollte die Rotation um die Sehachse bei der Berechnung der Kamerarotation außer Acht gelassen werden. Soll der Anwender sich auch noch auf einer festen Höhe bewegen, so darf die Kamera nur noch um die Hochachse rotieren können.

Insbesondere bei sehr weitläufigen Modellen ist eine feste Fluggeschwindigkeit eine sehr große Einschränkung in der Bedienbarkeit dieser Interaktionstechnik. Stehen eine ausreichende Anzahl an Triggermöglichkeiten zur Verfügung, so kann diese natürlich variabel gestaltet werden (beschleunigen, abbremsen). Da dies aber bekanntlich selten der Fall ist, bietet sich hier ein Automatismus an.

Es hat sich gezeigt, dass ein zum „Mapping von Auslenkung auf Rotationsbetrag“ analoges Verfahren sich hier gut bewährte. Hierbei wird dann nicht die Auslenkung , sondern die Zeit als Eingabewert verwendet.

Es hat sich weiterhin als sinnvoll erwiesen, dass beide Mappingfunktionen sich gegenseitig beeinflußen.

- Die Beschleunigung ist abhängig von dem aktuellen Rotationsbetrag
- Je höher die Geschwindigkeit desto geringer der maximale Rotationsbetrag

Dadurch wird das Navigieren in großen Anlagen (z.B. Fertigungsstraßen) wesentlich einfacher. Lange gerade Gänge können schnell durchflogen, bei verwinkelten Treppen und Gängen bleibt die Geschwindigkeit hingegen gering[1].

#### 4.3.2.3 Beidhändiges Fliegen

In verschiedenen Aufsätzen werden die Vorteile von beidhändiger Interaktion beschrieben (u.a. in [ZELE97] und [CUTL97]). Bei beidhändigem Fliegen [MINE97] wird der Richtungsvektor der durch die Position beider Hände gegeben, wird als Bewegungsvektor herangezogen. Je größer der Betrag des Vektors, desto schneller bewegt man sich die angegebene Richtung. Dadurch, dass der Vektor gerichtet ist, kann auch sehr einfach rückwärts geflogen werden. Um die Verwendung zu vereinfachen wird ein minimaler Abstand beider Hände definiert, unter dem der Betrag des Vektors immer Null ist. So ist das stoppen der Bewegung sehr einfach zu lösen.

Bei der Umsetzung der Technik wird sich jedoch die gleichzeitige Unterstützung von Links- und Rechtshändern als problematisch herausstellen: Beide Personengruppen werden annehmen, dass die Vektorspitze über ihre dominanten Hand gesteuert wird und entsprechend in unterschiedliche Richtungen „fliegen“. Es hat sich auch gezeigt, dass für das schnelle Verständnis der Technik ein mathematisch, naturwissenschaftliches Hintergrundwissen sehr hilfreich ist.

#### 4.3.2.4 Navigation mit Spacemouse

Diese Navigationstechnik ist sehr eng verbunden mit einer speziellen Hardware, der Spacemouse. Sie wird hier an dieser Stelle aufgeführt, da die Spacemouse im Gegensatz zu anderen „VR-Geräten“ auch in der Industrie recht häufig eingesetzt wird und hier sehr schön illustriert werden kann, wie wichtig es ist, die Erfahrung und das Wissen und die damit verbundenen Gedankenmodelle der Anwender zu hinterfragen.

VR-Experten verwenden die Spacemouse in den meisten Fällen zur Navigation, d.h. zur Steuerung der Kamera. Wird die Spacemouse nach vorne gedrückt, so fliegt man auf eine Szene zu oder anders ausgedrückt: Die Szene bewegt sich auf einen zu. Im CAD Umfeld hingegen wird über die Spacemouse die Lage und Orientierung der Szene im Raum gesteuert. D.h. wird hier die Spacemouse nach vorne gedrückt, entfernt sich die Szene vom Betrachter.

In Experimenten mit Personen, die aus einem CAD Umfeld stammen, konnte eine große Irritation festgestellt werden („Programmfehler!“) und erst nach Erklärungen wurde ihnen verständlich, dass sie die Kamera steuern und nicht die Szene bewegen. Trotz dieses Verstehens hatten diese Personen ein großes Problem, die Steuerung intuitiv zu bedienen.

### 4.3.3 Ansätze zur Objektpositionierung

In den folgenden Unterkapiteln werden einige der wichtigsten Ansätze zur Objektpositionierung vorgestellt.

#### 4.3.3.1 World in Miniature

Eine *World-in-Miniature* ([PAU95]) oder auch WIM ist ein verkleinertes Abbild der dargestellten Szene, welche der Benutzer in der Hand hält und manipulieren kann. Neben den Objekten der Szene ist auch die Kamera als Objekt in dem Abbild repräsentiert. Durch Umpositionieren eines Objektes

1. Auch hier kann wieder eine Analogie zur realen Welt gefunden werden: Bei einer Servolenkung im Auto wird das Mapping der Auslenkung von der Geschwindigkeit beeinflußt.

im WIM, wird das korrespondierende Objekt in der Szene analog transformiert. Durch Greifen des Kameraobjektes und Plazieren an einer anderen Stelle kann der Anwender sich somit ebenfalls durch die Szene bewegen. Nach [PAU95] ist es für den Anwender jedoch sehr störend, wenn während des Umpositionierens des Kameraobjektes die Position des Anwenders in der Welt kontinuierlich angepasst wird. Besser ist es, nach dem Plazieren des Kameraobjektes über eine Animation den Anwender zur neuen Position zu fliegen oder einen Flug in das WIM-Modell zu berechnen. Das WIM-Modell wird dadurch zur echten Szene.

Diese Technik hat den großen Nachteil, dass die gesamte Szene zweimal dargestellt werden muss, einmal in realer Größe und einmal als WIM. Bei Szenenkomplexitäten, die grundsätzlich die gesamte Graphikleistung benötigen, um noch akzeptable Bildwiederholraten zu erreichen, ist dieses Vorgehen natürlich inakzeptabel. Zwar könnte eine polygonal reduzierte Variante der Originalszene verwendet werden, jedoch wird diese bei starker Reduktion deutliche Qualitätsverluste aufweisen. Eine Vertiefung dieser Thematik findet sich u.a. in [KNOE98]. Auch andere Methoden, wie das Ersetzen der Objekte durch Würfel oder der Einsatz von Texturen führt entweder zu unansehnlichen Ergebnissen oder ist sehr aufwendig durchzuführen.

Ein weiterer Problempunkt ist das exakte Positionieren von Kamera und Objekten. Aufgrund des kleinen Maßstabes des WIM im Gegensatz zur realen Szene ist dies kaum möglich.

#### 4.3.3.2 Natürliches Greifen

Bei dieser Technik greift der Anwender mittels seiner virtuellen Hand ein Objekt der Szene, indem er das Objekt berührt und einen Event auslöst, z.B. bei einem Datenhandschuh durch die Geste „Faust“. Über eine Echtzeitkollisionserkennung ([ZACH98]) wird der Kontakt zwischen Objekt und Hand festgestellt. Das Objekt wird dann an der Hand „befestigt“ und folgt daraufhin der Handbewegung. Sobald der Anwender das Objekt losläßt bleibt es an der aktuellen Position stehen. Diese Technik ist die wohl bekannteste und auch natürlichste Art der Interaktion, da sie direkt aus der realen Welt übernommen worden ist. Der Nachteil dieser Technik ist, dass weit entfernte Objekte nicht erreicht werden können, da die Reichweite durch die Armlänge begrenzt wird.

#### 4.3.3.3 Arm Extension

Um der Limitierung der Reichweite zu begegnen, wurden sog. Arm-Extension Techniken entwickelt. Hierbei wird die Translation der Hand nicht direkt auf die Translation der virtuellen Hand abgebildet, sondern über eine spezielle Mappingfunktion skaliert. So kann mittels einer einfachen Skalierung der Aktionsbereich vergrößert werden, jedoch ist das genaue Positionieren dadurch komplizierter, da selbst bei kleinen Bewegungen der realen Hand die virtuelle Hand große Entfernungen zurücklegt. Neben einfachen Skalierungen sind auch komplexere Funktionen denkbar, z.B. [POUP96]. Hier wird die *Go-Go* Technik beschrieben, bei der bis zu einer bestimmten Armentfernung die Translation linear und dann exponentiell abgebildet wird. Siehe hierzu Abb. 4.5

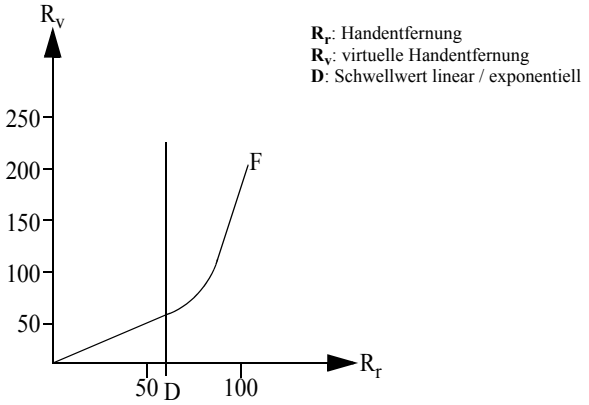


Abb. 4.5: Behandlung der Translation nach der Go-Go Technik

Die maximale Skalierung der virtuellen Hand kann hier z.B. die maximale Ausdehnung der Szene sein, sodass im Prinzip alle Objekte erreicht werden können. Hierbei muss dem System jedoch bekannt sein, wie groß die max. Reichweite der reale Hand ausgehend von einer initialen Position ist.

Eine andere Möglichkeit ist das *Stretch-Go-Go* (siehe [BOW96]), bei dem ausgehend von einer initialen Position, die Armlänge bei bestimmten Entfernungen entweder kontinuierlich verlängert oder verkürzt wird. Hierzu werden um den Benutzer drei Zonen definiert. Ist die physikalische Hand in der äußersten Zone, so wird der virtuelle Arm verlängert, ist die physikalische Hand in der mittleren Zone, so bleibt die Armlänge unverändert. Ist die reale Hand in der innersten Zone, so wird die Armlänge wieder verkürzt.

Arm-Extension Techniken haben sich insbesondere bei nicht linearer Abbildung als schwer handhabbar herausgestellt. Hierfür gibt es mehrere Gründe. Zum einen ist eine exponentielle Bewegung sehr unnatürlich und dadurch für den Anwender schwer zu verstehen. Insbesondere bei großer Entfernung tritt ein ähnliches Problem, wie bei Ray Casting auf (Kapitel 4.4.2.1). Durch kleinste Bewegungen der realen Hand legt die virtuelle Hand sehr große Entfernungen zurück und das Greifen von Objekten gestaltet sich dadurch sehr schwierig. Erschwerend kommt hinzu, dass in diesem Moment der Arm weit ausgestreckt ist, was zum einen die Auswahl nicht vereinfacht und zum anderen sehr schnell zu Ermüdungserscheinungen des Arms führt. Wird ein Schwellwert verwendet, so schränkt dies den Aktionsradius des Anwenders für „normale“ Arbeiten ein. Auch ist die Bestimmung eines optimalen Schwellwertes nicht einfach, da dieser insbesondere aufgrund unterschiedlicher Armlängen verschieden ist und somit für jeden Benutzer neu eingemessen werden muss.

Auch Stretch-Go-Go ist wenig intuitiv und schwer anzuwenden. Dies wird sogar in [BOW97] bestätigt. Weiterhin ist das Auswählen entfernter Objekte zeitaufwendig, da der Anwender erst warten muss, bis der Arm entsprechend ausgefahren worden ist. Außerdem wird der normale Arbeitsbereich durch die zwei Zonen sehr stark eingeschränkt.

#### 4.3.3.4 Homer

In [BOW97] wird das Verfahren *HOMER* beschrieben, was für *H*and-centered *O*bject *M*anipulation *E*xtending *R*ay-casting steht. HOMER ist eine hybride Technik, die sowohl die Objektselektion, als

auch Objektpositionierung und Objektrotation abdeckt. Die Objektselektion wird mittels Strahlselektion durchgeführt, woraufhin die virtuelle Hand an die Position des Objektes springt. Rotationen werde bzgl. der neuen Position der virtuellen Hand ausgeführt. Die Abbildung der Translation der physikalischen auf die virtuelle Hand geschieht derart, dass mit einer Bewegung des Arms das Objekt aus beliebiger Entfernung nahe zum Benutzer hingeführt werden kann. Bei weit entfernten Objekten ist daraufhin die Translationsgeschwindigkeit so hoch, dass exaktes Positionieren unmöglich ist.

In [PIER02] wurde die HOMER-Technik eingehender untersucht und mit der Interaktionstechnik Voodoo Dolls (siehe [PIER99]) verglichen. Es stellte sich heraus, dass das exakte Positionieren entfernter Objekte von über 80% der Versuchpersonen als sehr schwierig empfunden wurde. Zusätzlich wurde das fehlende Feedback bemängelt.

#### 4.3.3.5 Scaled World Grab

Ähnlich der Arm Extension Techniken ist der *Scaled World Grab*, entwickelt von [MINE97]. Hierbei wird die gesamte virtuelle Szene so weit skaliert, dass alle Objekte in Reichweite sind. Das gewünschte Objekt kann daraufhin gegriffen und positioniert oder betrachtet werden. Sobald das Objekt wieder losgelassen wurde, wird die Welt in ihre ursprüngliche Größe zurück skaliert, d.h. die Skalierung ist nur während der Manipulationsphase aktiv. Diese Technik kann auch für die Navigation verwendet werden, indem der Anwender sich an einem Objekt „festhalten“ und sich zu diesem Objekt hinziehen kann.

Bei diesem Ansatz treten ähnliche Probleme auf, wie bei den Arm Extension Techniken. Exakte Positionierung ist schwierig und bereits kleine Handbewegungen können zu großen Translationen naher Objekte führen.

### 4.3.4 Eigene Ansätze

Im Rahmen dieser Arbeit wurde der *world point grab* entwickelt, welcher sich als der intuitivste Ansatz zur Objektpositionierung für das gegebene Anwendungsgebiet erwies. Hierbei wird mittels eines Events der Punkt $p_1$, an dem sich die virtuelle Hand in diesem Moment befindet „gegriffen“ und alle betroffenen Objekte werden nun entlang des Vektors $\overrightarrow{p_1 p_2}$ transliert, wobei $p_2$ die aktuelle Position der virtuellen Hand ist. Das Mapping wird 1:1 abgebildet. Rotationen werden bzgl. der aktuellen Position der virtuellen Hand ausgeführt. Siehe hierzu Abb. 4.6.

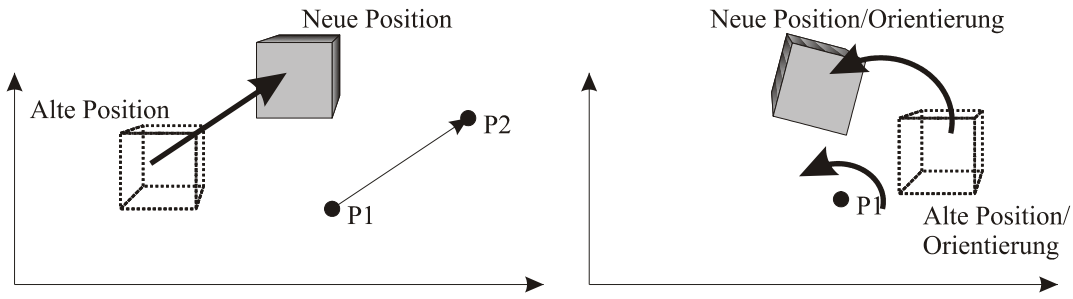


Abb. 4.6: world point grab: Position und Orientierung

Somit kann der Anwender den Rotationspunkt selber bestimmen und „weiß“ über die Proprioception, wo sich dieser in der virtuellen Szene befindet, selbst wenn die virtuelle Hand in diesem Moment für ihn nicht sichtbar ist. In bestimmten Anwendungsfällen kann es jedoch auch nützlich sein, dass das Rotationszentrum von der Applikation vorgegeben wird, um z.B. ein entferntes Objekt um dessen Zentrum drehen zu können. Siehe hierzu Kapitel 6.1.2 für die Beschreibung einer solchen Anwendung.

Diese Methodik ist dem natürlichen Greifen sehr ähnlich, jedoch muss beim world point grab das Objekt keinerlei Kontakt zur virtuellen Hand haben. Es wird somit auch keine spezielle Kollisionserkennung benötigt. Dadurch dass *world point grab* sehr stark die menschlichen Proprioception mit einbezieht, ist dieser Metapher prinzipiell auch nur im Zusammenhang mit getrackten Eingabegeräten sinnvoll zu verwenden.

world point grab läßt sich auch zur Navigation einsetzen, indem entweder die gesamte Szene oder über den invertierten Translationsvektor die Kamera transliert wird (je nach zugrundeliegendem Systemmodell, siehe Kapitel 3.3.2). Siehe hierzu auch Abb. 4.7.

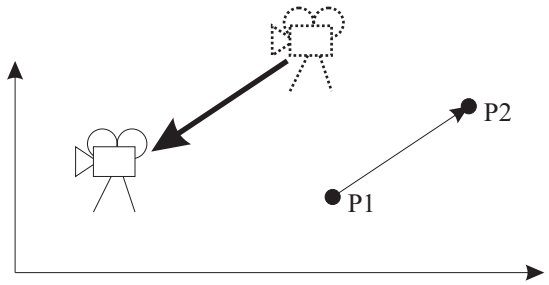


Abb. 4.7: world point grab für die Navigation

Für die Navigation in Modellen mit großer Ausdehnung ist dieser Metapher jedoch nur bedingt geeignet, hier sollte dann Pointfly verwendet werden. Hierzu wurden im Rahmen der Arbeit einige Erweiterungen entwickelt, welche bereits in Kapitel 4.3.2.1 Kapitel beschrieben worden sind. Zwar ist es auch prinzipiell denkbar diese Technik mit Arm Extension zu kombinieren, dies ist jedoch aus bereits erwähnten Gründen nicht sinnvoll.

### 4.3.5 Diskussion

Für den Bereich der Navigation wurden die Techniken Pointfly, das beidhändige Fliegen und Spacemouse-Navigation vorgestellt. Zusätzlich können die hybriden Techniken WIM und world point grab verwendet werden, die sowohl Navigation, als auch Objektpositionierung ermöglichen. Aufgrund der bereits beschriebenen Probleme bei WIM scheidet diese Methodik für den hier angestrebten Verwendungszweck aus. Insbesondere für die Navigation in großen Szenarien kämen somit entweder Pointfly oder das beidhändige Fliegen zum Einsatz. Aus der Erfahrung heraus ist jedoch der Pointfly-Modus vorzuziehen, da er intuitiver zu bedienen ist und nur ein getracktes Eingabegerät voraussetzt.

Im Bereich der Objektpositionierung unterscheiden sich die Interaktionstechniken hauptsächlich

durch die folgenden Faktoren:

- Abbildung der Bewegungen der realen Hand auf die virtuelle Hand
- Wählbarkeit des Objekteangriffspunkt (Rotationszentrum)
- Objektselektion als Bestandteil der Positionierung

Es hat sich gezeigt, dass nicht-lineare Abbildung der Bewegung der realen auf die virtuelle Hand als sehr störend empfunden wurde. Aufgrund der Immersion und der möglichst natürlichen Interaktion sind Skalierungsfaktoren, die sehr stark von 1,0 abweichen ebenfalls zu vermeiden. Bei großen Skalierungen kommt hinzu, dass die Positionierung sehr schwierig wird, da selbst kleine Bewegungen große Positionsänderungen der realen Hand nach sich ziehen. Der Angriffspunkt sollte möglichst intuitiv durch den Anwender bestimmt werden können. Feste Einstellungen, wie z.B. das Zentrum der Bounding Box des Objektes erschweren insbesondere bei größeren Objekten die Benutzbarkeit in großem Maße.

Wie in Kapitel 3.3.2 beschrieben, sollten alle Aktionen mit einem System auf einer kleinen Anzahl atomarer Aktionen aufbauen. In Bezug auf graphische Bedienoberflächen kann diese Anforderung mit Hilfe der *basic interaction tasks* auf abstrakter Ebene umgesetzt werden. Diese sollten möglich über eine jeweils eindeutige Interaktionstechnik realisiert werden. Da jedoch einige der hier vorgestellten Techniken die Objektselektion fest definieren, muss entweder diese Technikfür die „normale“ Objektselektion ebenfalls verwendet oder die Selektion dann mittels mehrerer Techniken realisiert werden. Ersteres würde den Entwurf einer intuitiven Schnittstelle stark einschränken und letzteres der oben beschriebenen Grundidee zuwider laufen.

Aufgrund dieser drei Faktoren ist der *world point grab* den anderen Positionierungstechniken vorzuziehen. Wenn im Rahmen eines Design Reviews kleinere Modelle betrachtet werden, so kann über diese Interaktionstechnik auch die Kameranavigation realisiert werden. Hierbei wird einfach das gesamte Modell gegriffen und bewegt.

## 4.4 BIT Objektselektion

Über die Objektselektion können einzelne oder mehrere Objekte der virtuellen Umgebung selektiert werden. Auf diesen Objekten können dann z.B. spezielle Funktionen angewendet werden.

Im folgenden werden zuerst die Anforderungen an den Selektionsmechanismus definiert, die sich aus den mit diesem Rahmensystem zu bearbeitenden Szenarien ergibt. Im Anschluss werden verschiedene Verfahren aus der Literatur vorgestellt, die anhand der Anforderungen bewertet und gegebenenfalls adaptiert wurden.

Wie bereits im Aktionsmodell von [NORM90] dargelegt, sollte jede Aktion ein Feedback auslösen, d.h. eine erfolgreiche Selektion sollte für den Anwender sichtbar gemacht werden. In Kapitel 4.4.5 werden einige Ansätze hierzu vorgestellt.

### 4.4.1 Anforderungen

Die wohl wichtigste Fragestellung im Zusammenhang mit Objektselektion ist die Frage nach dem Objekttyp, der selektiert werden kann. Handelt es sich z.B. um 3D-Daten eines Fahrzeuges, reicht es

zumeist aus ganze „Bauteile“ zu selektieren, d.h. ein oder mehrere Knoten in einem Szenengraph, die dieses Bauteil repräsentieren. Andererseits wird bei Anwendungen aus dem Bereich der Simulation auch die Möglichkeit benötigt, einzelne Punkte oder Flächen (Dreiecke) auf einer 2D-Oberfläche (*Data Probes*) zu selektieren. Um nun den Anwender nicht mit einer Vielzahl an Selektionswerkzeugen zu verwirren, sollte ein idealer Mechanismus sowohl die Selektion auf Objekt-, Flächen- und Punktebene unterstützen.

Ein weiteres wichtiges Kriterium ist die Anzahl, Größe und Lage der einzelnen Objekte der Szene. In den hier verwendeten Datensätzen handelt es sich um Szenen mit vielen kleinen Objekten, konzentriert auf einen begrenzten Raum, z.B. ein Motorraum.

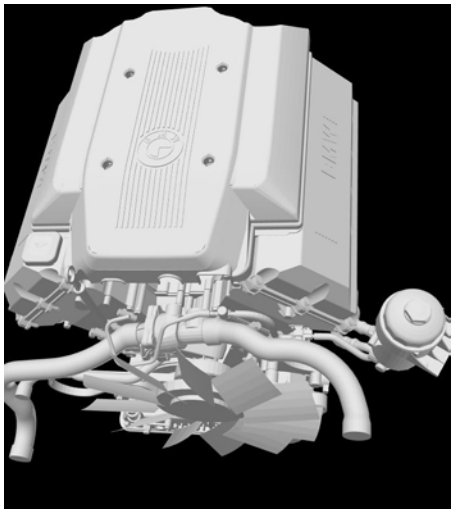

Abb. 4.8: Typisches Szenario Motorraum

### 4.4.2 Ansätze aus der Literatur

Im folgenden werden die verschiedenen aus der Literatur bekannten Ansätze kurz vorgestellt und Adaptionen und Erweiterungen, die im Rahmen dieser Abreit durchgeführt worden sind, erläutert.

#### 4.4.2.1 Ray Casting

Beim Ray Casting ([HICK94]) wird mittels eines virtuellen Laserstrahls unendlicher Länge (z.B. repräsentiert durch eine Linie) die Objektselektion durchgeführt. Dieser virtuelle Laserstrahl ist mit der virtuellen Hand des Benutzers verbunden und kann somit sehr einfach positioniert und orientiert werden. Die Anwendung dieser Metapher ist somit einem realen „Laserpointer“ sehr ähnlich. Für ein verbessertes visuelles Feedback sollte der Laserstrahl die Objekte möglichst nicht durchdringen, sondern nur von der virtuellen Hand bis zum Schnittpunkt zwischen Strahl und Objekt verlaufen.

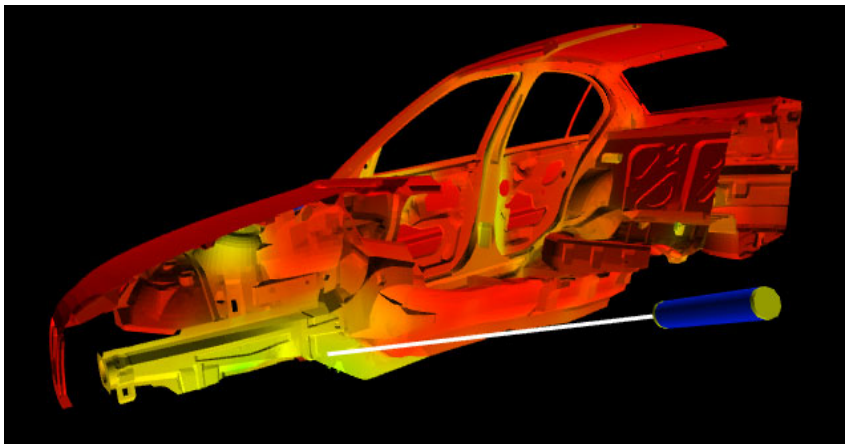

Abb. 4.9: Raycasting

Dadurch dass der Laserstrahl durch eine dünne Linie repräsentiert werden kann, ist es auch möglich einzelne Punkte auf einer Oberfläche zu selektieren. Je nach Anforderung der Anwendung kann über diesen 3D-Punkt auch die zugehörige Fläche oder das zugehörige Objekt selektiert werden.

Problematisch ist jedoch die Selektion von weit entfernten oder sehr kleinen Objekten. Bei sehr weit entfernten Objekten resultiert selbst eine kleine Winkeländerung der realen Hand in einer großen Translation des Laserstrahls. Deswegen muss der Strahl sehr exakt positioniert und orientiert werden, um das richtige Objekt zu treffen.

Um dieses Problem zu lösen, kann der virtuelle Laserstrahl durch einen Konus ersetzt werden ([LIAN94]), dessen Öffnungswinkel entweder interaktiv bestimmt werden kann oder von der Applikation vorgegeben wird. Objekte, die sich in dem Konus befinden, werden selektiert.

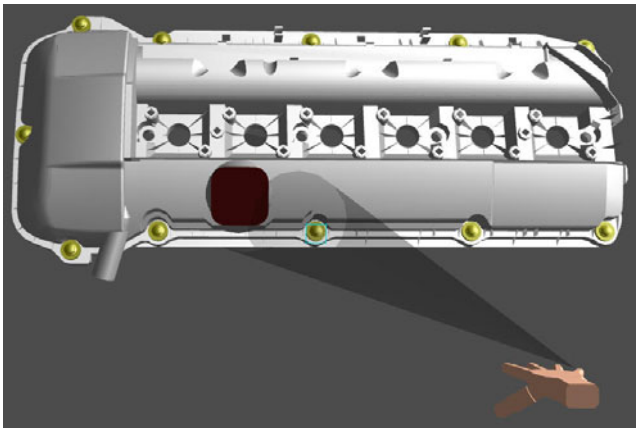

Abb. 4.10: Konusselektion

Schwierig gestaltet sich jedoch die eindeutige Objektauswahl bei vielen, kleinen sich in Strahlrichtung überdeckenden und im Konus liegenden Objekten. Um die dabei entstehenden Mehrdeutigkeiten aufzulösen, wurde im Rahmen dieser Arbeit ein entsprechender Selektionsalgorithmus entwikkelt.

**Algorithmus zur Auflösung von Mehrdeutigkeiten bei Konusselektion**

Im folgenden werden zuerst einige alternative Ansätze vorgestellt und diskutiert und anschließend der eigentliche Algorithmus besprochen.

Die wohl genaueste Methode für die Konusselektion wäre die Verwendung einer polygonal basierten Kollisionserkennung, die die Flächen der Objekte gegen den Konus schneidet. Es gibt zwar sehr viele schnelle Kollisionserkennungen (z.B. [ZACH98]), aber insbesondere bei komplexen Szenarien benötigen diese sehr viel Rechenzeit und -leistung. Insbesondere wenn das System durchgängig visuelles Feedback liefert, welches Objekt Selektionskandidat ist, muss die Kollisionserkennung in jedem Frame ein definiertes Ergebnis liefern. Diese Forderung kann die Performanz des Gesamtsystems einschränken und damit die Interaktion verschlechtern (niedrige Framerate).

Im Rahmen dieser Arbeit wurde deswegen ein zweistufiger Algorithmus entwickelt, bei dem nur Schnitte zwischen Bounding Boxen berechnet und Strahltests durchgeführt werden müssen. Beide Berechnungen sind recht schnell und einfach auszuführen.

Die Grundidee ist eine Klassifizierung der Objekte vorzunehmen, die vollständig oder teilweise im Konus liegen. Je nach Klassifikation und Entfernung von der Konusspitze wird ein sinnvoller Selektionskandidat berechnet.

Im folgenden wird der komplette Algorithmus als Pseudocode dargestellt:

```
FOR ALL (zu betrachtende Objekte o_i)
  # Bounding Box von oi: bbox_oi
  # Konusobjekt: c
  IF (bbox_oi vollständig innerhalb c) THEN
    Füge o_i zu Liste L_inner hinzu
  ELSE
    IF (bbox_oi wird von c geschnitten) THEN
      Füge o_i zu Liste L_partial hinzu
    ENDIF
  ENDIF
ENDFOR
IF (Anzahl Einträge L_inner == 0) THEN
  FOR ALL (Einträge o_p in L_partial)
    IF (Mittelstrahl s schneidet o_p) THEN
      Füge o_p zu L_inner hinzu
    ENDIF
  ENDFOR
ENDIF
IF (Anzahl Einträge L_inner == 0) THEN
  RETURN NULL
ENDIF
Bestimme Distanz dist_i aller o_i aus L_inner zur
    Konusspitze
o_result = (o_i aus L_inner mit kleinster dist_i)
RETURN o_result
```

Abb. 4.11: Pseudocode des Selektionsalgorithmus

Es hat sich gezeigt, dass dieses Vorgehen den „Erwartungen“ der Anwender am ehesten entspricht.

Bei großen Objekten mit Löchern kann dieser Algorithmus fehlerhafte Selektionsergebnisse liefern, indem der Mittelstrahl aufgrund des Lochs das Objekt nicht trifft oder das Loch größer als der Konus ist. Dies ist jedoch ein eher hypothetischer Fall, weil Anwender beim Auswählen eines Objektes mit dem Konus sicherlich nicht auf das Loch zeigen.

**Selektionsgeschwindigkeit zwischen Konus und Strahl**

Im Rahmen dieser Arbeit wurde untersucht, wie stark sich die Selektionsgeschwindigkeit bei kleinen Objekten unter Verwendung des Konus verändern. Hierzu wurden 10 kleine Objekte in der virtuellen Szene verteilt, die nacheinander angefahren und durch Knopfdruck selektiert werden sollten. Dieser Vorgang wurde zweimal wiederholt, einmal unter Verwendung des Strahls und einmal unter Verwendung des Konus. Hierbei hat sich gezeigt, dass die Selektion kleiner Objekte mittels des Konus im Mittel *doppelt* so schnell war, wie bei der Verwendung des Strahls.

### 4.4.2.2 Image Plane Interaction

Der Bereich der Image Plane Interaction wird insbesondere in [PIER97] behandelt. Hierbei interagiert der Anwender nicht direkt mit der 3D Szene, sondern mit dem 2D Abbild auf der Projektionsebene. Dies läßt sich am einfachsten an der *Sticky Finger* Technik verdeutlichen. Hierbei wird ausgehend vom Augpunkt über einen bestimmten Finger der Hand[1] ein unsichtbarer Strahl gesendet. Dann wird das Objekt selektiert, welches von diesem Strahl getroffen wird. Oder aus der Sicht des Anwenders gesprochen: Das Objekt, welches mit dem Finger verdeckt wird, wird selektiert. Der Anwender berührt somit nicht das eigentlich zu selektierende Objekt, sondern agiert nur mit dem 2D Abbild des 3D Modells. Aufbauend auf Sticky Finger wurden von Pierce noch die *Head Crusher*, *Lifting Palm* und *Framing Hands* Technik entwickelt. Siehe hierzu Abb. 4.12.

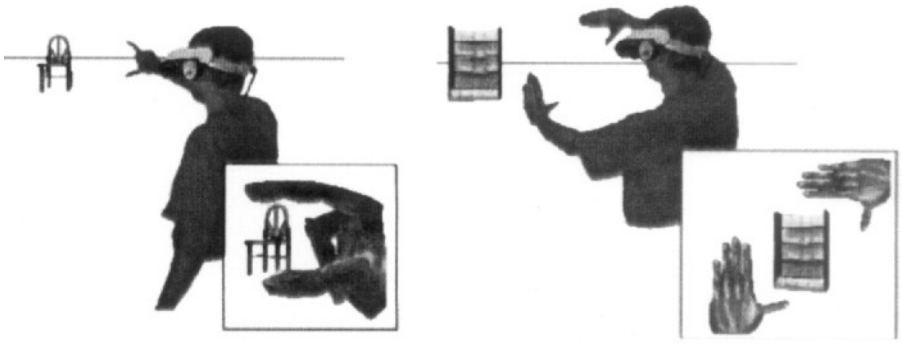

Abb. 4.12: Image Plane Selektion: Head crusher (links) und framing hands (rechts), aus [PIER97]

Bei Head Crusher werden Objekte, welche sich zwischen Daumen und Zeigefinger befinden selektiert und bei Lifting Palm die, die sich über der Handoberfläche befinden. Bei Framing Hands wird

1. Je nach Setup kann dies die reale oder virtuelle Hand sein

das Objekt, welches sich im Rahmen befindet, welcher durch die beiden Hände aufgespannt wird. Je nach Ausgabegerät kann hier entweder über die virtuelle (z.B. HMD), als auch über die reale Hand (z.B. CAVE) die Auswahl getroffen werden.

Die Verwendung der Image Plane Interaction wirft jedoch einige generelle Probleme auf. Wird Stereodarstellung verwendet, so ist die Selektion aufgrund der Disparität nicht eindeutig, da z.B. bei Sticky Finger der Finger verschiedene Objekte verdecken könnte. Diese Problematik kann leider nicht zufriedenstellend gelöst werden.

Weichen aufgrund des Trackings virtuelle und reale Augposition voneinander ab und die Selektion wird über den realen Finger durchgeführt, kann es insbesondere bei kleineren Objekten passieren, dass der Anwender über den Finger ein Objekt selektiert, der Strahltest fehlschlägt und am Objekt vorbeigeht. (siehe hierzu auch Kapitel 6.1).

Aufgrund der hier verwendeten Hardware (siehe Kapitel 4.1.3), kann nur Sticky Finger als Selektionstechnik eingesetzt werden. Wird das reale Gerät zum „Peilen“ verwendet, muss es über eine möglichst dünne Spitze verfügen, um „eindeutige“ Selektion zu ermöglichen.

Insbesondere bei Selektion über den realen Finger (reales Gerät) kommt es bei dieser Technik sehr schnell zur Ermüdung des Armes, da für jede Selektion die Hand gehoben werden muss. Wird der Finger der virtuellen Hand zum Anpeilen eines Objektes verwendet, so kann dies durch einen vertikalen Offset weitgehend verhindert werden (indirekte Interaktion).

#### 4.4.2.3 Toolglasses und Magic Lenses

Toolglasses™ und Magic Lenses™ wurden in [BIER93] vorgestellt und sind in Hinblick auf Desktop-Anwendungen entwickelt worden, können jedoch auch in der VR verwendet werden und sind hier als eine Erweiterung der Image Plane Techniken zu verstehen (siehe Kapitel 4.4.2.2). Ein Toolglass ist eine Art transparente Scheibe auf der verschiedene Figuren, die sog. Magic Lenses aufgebracht sind. Magic Lenses repräsentieren die Funktionen des Systems. Das Toolglass kann frei auf der Arbeitsfläche verschoben werden und schwebt über allen Objekten der Szene. Soll ein bestimmtes Objekt mit einer bestimmten Magic Lense bearbeitet werden, so muss nur das Toolglass entsprechend verschoben werden, bis die Magic lense über dem Objekt zum liegen kommt. Die Magic Lense wird durch Selektion über einen Cursor aktiviert. So können Magic Lenses z.B. die Teilszene, die durch sie zu sehen ist, als Drahtgitter darstellen oder die Farbe des unterliegenden Objektes verändern. Mehrdeutigkeiten können über den Cursor aufgelöst werden. Toolglasses und Magic Lenses decken somit sowohl die Objekt-, als auch die Funktionsselektion ab.

**Stift und Tablett**

In der VR lassen sich Toolglasses und Magic Lenses z.B. über *Stift und Tablett* Benutzerschnittstellen realisieren. Bei dieser Schnittstelle hält der Anwender ein durchsichtiges, getracktes Tablett in der nicht dominanten Hand und einen Stift in der dominanten Hand ([BILL97]). Das Tablett wird als virtuelles Tablett in der Szene dargestellt, wobei dessen Position und Orientierung abhängig vom realen Tablett ist. Ziel ist es dabei, dass sich reales und virtuelles Tablett visuell an der selben Position befinden.

Der reale Stift wird ebenfalls als zusätzliches virtuelles Objekt (Cursor) in der Szene dargestellt. Über das virtuelle Tablett können nun diverse Funktionen ausgelöst werden, indem der Anwender mit dem Cursor das zugehörige Funktionsobjekt auf dem virtuellen Tablett berührt. Hierbei erfährt der Benutzer auch ein haptisches Feedback, da er im selben Moment mit dem realen Stift auf das reale Tablett drückt.

In [LIND99] werden die Einflüsse dieses haptischen Feedbacks untersucht. Hierbei wurde dem Anwender die Aufgabe gestellt, mittels des Cursors ein Objekt auf dem virtuellen Tablett auszuwählen. Dieses Experiment wurde einmal mit realen Tablett und einmal nur mit dem Griff in der Hand durchgeführt. Hierbei war die Selektionsgeschwindigkeit mit haptischen Feedback im Mittel 17% kürzer.

Werden Displaysysteme verwendet, bei denen der Anwender das reale Tablett sehen kann (z.B. Virtual table, Powerwall), so kann das virtuelle Tablett so positioniert werden, dass es für den Anwender so erscheint, als ob die Objekte auf dem virtuellen Tablett sich auf dem realen Tablett befinden. Dies stellt natürlich sehr hohe Anforderungen an das Trackingsystem, um realistische Resultate zu erzielen.

Basierend auf dem Stift und Tablett-Ansatz wurden verschiedene sehr interessante Anwendungen entwickelt. Stellvertretend seien hier nur die „Virtuelle Studierstube“ ([SCHM96]), das Sketchen auf dem Tablett über Gestenerkennung ([ENCA99]) und die Verwendung von Toolglasses und Magic Lenses ([WILL99]) genannt. Diese Anwendungen zeigen sehr deutlich, welches Potential in dieser Schnittstelle liegt.

Ein nicht zu unterschätzendes Problem dieses Ansatzes ist die Ermüdung der Arme. Zieht man hier noch einmal die Untersuchung von [LIND99] heran, so stellt man fest, dass die Ermüdungserscheinung bei „Stift und Tablett in beiden Händen“ im Mittel um 83% höher ausfallen, als bei „ein Stift in einer Hand“. Da Design Reviews mehrere Stunden dauern können, ist dies ein schwerwiegendes Problem.

Zusätzlich kommt hinzu, dass während eines Reviews sowohl Tracking, als auch Stereo ausgeschaltet werden können und dann die Anwender die Stereobrillen abnehmen. Da dadurch das Kopftracking ebenfalls wegfällt, kann diese Technik nicht mehr optimal eingesetzt werden, da die Applikation keine sinnvolle Position und Orientierung für das virtuelle Tablett annehmen kann.

### 4.4.3 Der Selektionsmodus „Click and Drag to Select“

Das Fehlen von haptischem Feedback macht das exakte Positionieren und Orientieren des Selektionsobjektes (z.B. virt. Laserstrahl, „Sticky Finger“-Referenzobjekt) im 3D-Raum sehr schwierig. Hinzu kommt, dass durch das Drücken eines Triggers auf dem Eingabegerät das Gerät leicht bewegt werden könnte[1]. Dies hat zwangsläufig Auswirkungen auf die Position und Orientierung des Selektionsobjektes. Zur Kompensation kann im Falle des Raycastings der Konus (siehe Abb. 4.4.2.1) eingesetzt werden oder der übliche Selektionsmodus *click to select* durch *click and drag to select* ersetzt werden.

1. eine Tatsache, die z.B. auch beim Design der heutigen Computermaus eingeflossen ist. Nicht ohne Grund sind die Maustasten nicht seitlich angebracht, sondern müssen von oben gedrückt werden.

Bei *click to select* wird beim Ändern des internen Zustandes eines Triggers (z.B. Knopf drücken oder loslassen) die Position/Orientierung des Selektionsobjektes ausgewertet und ggf. ein Objekt selektiert. Man spricht beim Übergang von einem internen Zustand zu einem anderen auch von „Flanke“. Am Beispiel des Raycastings bedeutet dies z.B. folgendes: Wird der Selektionsknopf auf dem Eingabegerät gedrückt wird die Selektion ausgelöst. Hierzu wird der der virtuelle Laserstrahl gegen die Szene geschnitten und das vom Strahl getroffene Objekt selektiert.

Bei *click and drag to select* wird ab dem Zeitpunkt der aufsteigenden Flanke des Triggers das Selektionsobjekt kontinuierlich ausgewertet und das erste „gefundene“ Objekt selektiert. Am Beispiel des Raycastings erläutert bedeutet dies, dass nach dem Auslösen des Triggers („Knopf gedrückt“) der virt. Laserstrahl „scharf geschaltet wurde“ und dann das Objekt selektiert wird, welches als erstes vom Strahl getroffen wird. Eine hohe Framerate vorausgesetzt können somit Objekte per virt. Laserstrahl „eingefangen“ werden. Auf dieser Basis könnte auch ein Brush Select aufgebaut werden, bei dem solange Objekte selektiert werden, bis die negative Flanke des Triggers ausgelöst wird.

In der folgenden Abbildung ist der Unterschied zwischen diesen Selektionsmodi graphisch dargestellt.

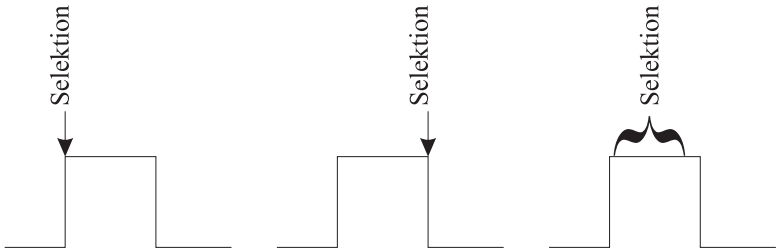


Abb. 4.13: Click to select (positive, negative Flanke) versus Click and Drag to Select

### 4.4.4 Evaluierung der Interaktionstechniken

In der Vorstellung der verschiedenen Ansätze hat sich bereits ein leichter Vorteil des Ray Castings, bzw. des Konus zur Selektion von Objekten gegenüber der Image Plane Technik herauskristallisiert. Dies liegt insbesondere an der Stereoproblematik und die zwingende Verwendung von Headtracking seitens der Image Plane Technik begründet. Offen ist jedoch die Frage, welche Methode vom Anwender intuitiver eingesetzt werden kann und wo die Ermüdungserscheinungen am geringsten sind.

Hierzu wurden im Rahmen dieser Arbeit verschiedene Untersuchungen durchgeführt, welche die Effektivität der beiden Interaktionstechniken miteinander vergliechen.

#### 4.4.4.1 Versuchsaufbau

Bei diesem Versuch muss der Proband möglichst schnell ein bestimmtes Objekt mit einer der Selektionstechniken auswählen und dann einen Knopf auf dem Eingabegerät drücken um die Auswahl zu bestätigen. Nach der Bestätigung wird die *Position* oder *Größe* des Objektes verändert und der Anwender muss es erneut selektieren. Um die bei diesem Experiment gewonnenen Daten nicht zu „verrauschen“, werden diese beiden Größen nicht gleichzeitig verändert. Da jedoch die ausschließliche

Veränderung der Größe des Objektes keinen Sinn machen würde, wird hier die Position unter der Maßgabe geändert, dass der Abstand Objekt zu Kamera unverändert bleibt.

Ist das Objekt ausgewählt, wird es in Drahtgitterdarstellung dargestellt. Dies sollte die Selektion bei beiden Techniken vereinfachen und die Probleme bei Image Plane, die aufgrund der Stereodarstellung auftreten, weitgehend kompensieren. Das zu selektierende Objekt ist ein weißer Würfel. Der Hintergrund der Szene ist schwarz, um einen möglichst hohen Kontrast zu erreichen. Der Selektionsstrahl und das Proxyobjekt für Image Plane sind rot gefärbt.

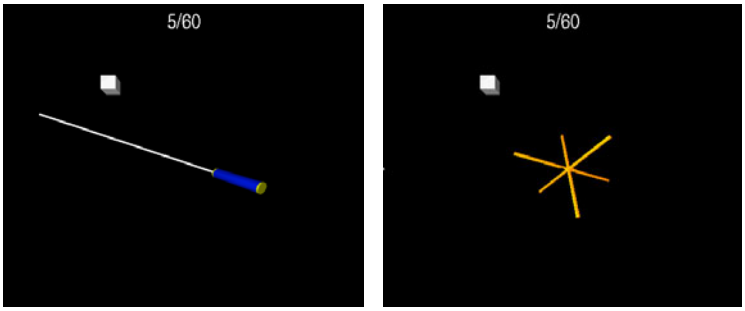


Abb. 4.14: Versuchsaufbau (Ray-Casting, Image-Plane)

Hierzu wird folgende Hypothese aufgestellt:

**H1:** Die Selektion mittels Ray-Casting ist schneller als die Image Plane Interaktion
**H0:** Die Selektion mittels Ray-Casting ist gleich schnell oder langsamer als die Image Plane Interaktion

Von grundsätzlichem Interesse bei diesem Versuch ist auch, ob der Anwender bei der Verwendung des Ray-Castings alle 6 Freiheitsgrade verwendet oder sich vielleicht in den meisten Fällen nur auf die rotatorischen Freiheitsgrade beschränkt:

**H1:** Bei Ray-Casting werden im wesentlichen nur die rotatorischen Freiheitsgrade vom Anwender benutzt.
**H0:** Bei Ray-Casting werden alle 6 Freiheitsgrade gleichberechtigt eingesetzt oder sogar die translatorischen Freiheitsgrade vorgezogen.

Träfe H1 zu, so könnte aus diesem Versuch auch eine generelle Richtlinie zur Verwendung von Rotation und Translation in Interaktionstechniken entwickelt werden.

#### 4.4.4.2 Versuchsdurchführung

Dieses Experiment wurde auf einer sgi ONYX-InfiniteReality durchgeführt. Als Projektionssystem kam eine Rückprojektion mit den Maßen 2,40m auf 2,40m zum Einsatz. Die virtuelle Szene wurde stereoskopisch mittels Shuttertechnik dargestellt. Die Testpersonen trugen CrystalEyes Stereoglasses, die mit einem Polhemus Trackingsystem versehen waren.

Für dieses Experiment wurden 10 Personen rekrutiert. Die meisten hatten bereits Erfahrung mit Vir-

tueller Realität. Den Probanden wurde mitgeteilt, dass das Ziel ist, möglichst schnell die Selektion durchzuführen.

Jeder Proband mußte zwei Tests durchlaufen, wobei beim ersten Test Ray-Casting und beim zweiten Test Image Plane zur Selektion verwendet wird. Pro Test wurden 60 Einzelselektionen durchgeführt. Bei den ersten 20 Tests wurde der Würfel ausschließlich in der Projektionsebene verschoben. Bei den folgenden 20 Tests wurde der Würfel zusätzlich in der Tiefe verschoben. Bei den letzten 20 Tests wurde wieder der Würfel nur in der Projektionsebene verschoben, jedoch zusätzlich die Größe des Objektes verändert.

#### 4.4.4.3 Versuchsergebnisse und Diskussion

Die Versuchsergebnisse haben sich auf dem 5%-Niveau als signifikant herausgestellt ($z=1,75$; $p < 0.05$). Die mittlere Selektionszeit bei *ray-casting* lag bei 1,7 Sekunden, bei *image plane* bei 2,2 Sekunden.

In Kapitel 4.4.2.2 wurde bereits angedeutet, dass es bei Stereodarstellung zu Problemen bei der Selektion kommen kann. Dies wurde zum einen von den Versuchspersonen verbal bestätigt, zum anderen zeigt sich dies auch in dem in den Versuchen gesammelten Datenmaterial. Siehe hierzu Abb. 4.15. Hier zeigt sich, dass bei größerer Entfernung von Selektionsobjekt zu dem zu selektierenden Objekt die Selektionszeiten extrem zunahmen. Siehe hierzu die mittlere Datenreihe in Abb. 4.15 bei der der Würfel zwar in seiner Größe gleich blieb, jedoch die Position auch in der Tiefe verändert wurde. Einige der Probanden versuchten dies durch das Ausstrecken des Arms zu kompensieren. Dass diese hohen Selektionszeiten nicht durch die entfernungsabhängige Objektgröße auf der Projektionswand zustande kamen zeigt sich an der dritten Datenreihe. Hier wurde die Objektgröße verändert, jedoch nicht die Entfernung des Objektes zur Kamera, respektive dem Selektionsobjekt.

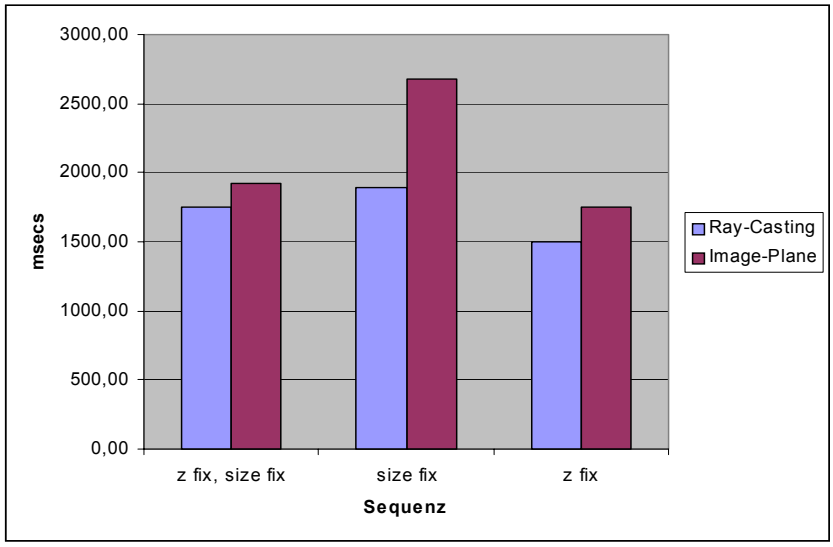


Abb. 4.15: Performanz in Abhängigkeit der Position / Objektgröße

Seitens der Probanden, die geringere Erfahrungen mit VR hatten, wurde die Image-Plane Metapher als angenehmer und direkter empfunden, jedoch auf Dauer anstrengender als das Ray-Casting. Be-

trachtet man die während der Versuche ausgeführten Translation, so stellt man fest, dass bei Image-Plane die Hand ca. 3 mal soviel transliert wurde, wie bei Ray-Casting. Siehe hierzu Abb. 4.16. Hier wurden die Mittelwerte aus den Distanzen zwischen der minimalen und der maximalen Handposition jedes der 60 Einzelversuche dargestellt. Diese Graphik gibt somit eine gute Übersicht über die unterschiedlichen Translationsaufwände zwischen den beiden Interaktionstechniken.

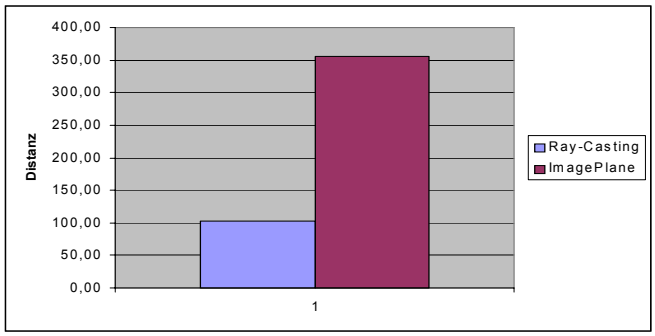


Abb. 4.16: Vergleich der Translation zwischen Ray-Casting und Image Plane (Mittelwerte)

Zusammenfassend ist somit zu sagen, dass wenn möglich die Ray-Casting-Technik der Image-Plane Technik vorgezogen werden sollte, da sie zum einen wesentlich bessere Performanz bei entfernten Objekte liefert und auf Dauer nicht so anstrengend ist. Wird das System von „Neulingen“ eingesetzt, sollte jedoch für bestimmte Situationen, bei denen die Distanz des Selektionsobjektes zum Objekt gering ist, evtl. auf Image-Plane zurückgegriffen werden. Für diese Personengruppe ist das Interagieren dann zu Beginn intuitiver.

### 4.4.5 Feedback: Anzeige der Objektselektion

Wie bereits in Normans Aktionsmodell (siehe Kapitel 3.3.2) dargelegt, sollte bei jeder Aktion des Benutzers das System Feedback geben. Diese Feststellung wird durch eine Untersuchung von Pierce [PIER02] unterstützt. Viele Funktionen eines Systems agieren direkt auf der Szene und sobald sie aufgerufen worden sind, verändern sich Objekte und Szenerie sichtbar (z.B. wireframe). Bei der Objektselektion ist es jedoch prinzipiell für die Funktionsweise nicht notwendig, dass eine Selektion ein visuelles Feedback hervorruft, denn in dem Moment, in dem eine Selektion ausgeführt wird, merkt sich das System das selektierte Objekt in einer Art Liste. Erst wenn dann eine Aktion auf den selektierten Objekten ausgeführt wird, wird sich das visuelle Erscheinungsbild der selektierten Objekte oder der Szene ändern. Deswegen ist es immanent wichtig, dass ein solches visuelles Feedback eingeführt wird, damit der Anwender weiß, ob und welche Objekte er selektiert. Hierzu gibt es verschiedene Möglichkeiten:

**Bounding Box:** Um das dargestellte Objekt wird eine Bounding Box angezeigt. Diese Anzeige ist sicherlich sehr einfach zu realisieren, ist jedoch nicht eindeutig. Insbesondere bei größeren Objekten ist die Zuordnung, welche Teile der Szene zu dem selektierten Objekt gehören nicht zu beantworten, da nicht alle Objekte, die in der Bounding Box liegen auch zum selektierten Objekt gehören.

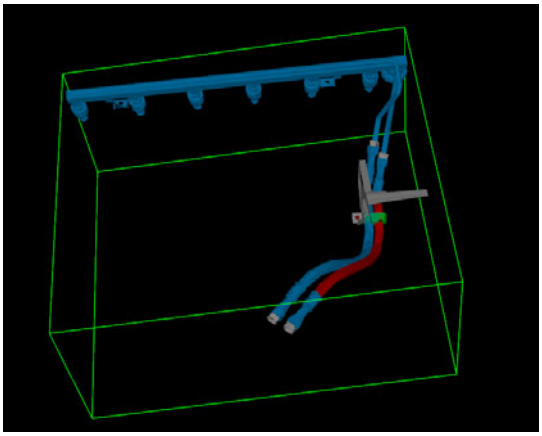

Abb. 4.17: Selektionsanzeige durch Bounding Box

**Wireframe:** Das selektierte Objekt wird als Drahtgitter dargestellt. Diese Methode ist ebenfalls sehr einfach zu realisieren, führt jedoch bei komplexen Objekten schnell dazu, dass das selektierte Objekt nicht mehr zu erkennen ist, sondern nur noch ein Geflecht von Linien.

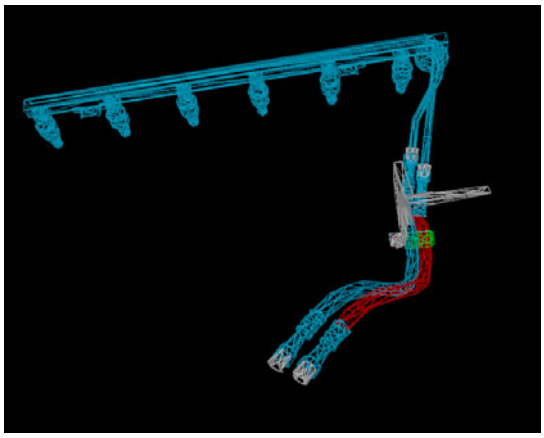

Abb. 4.18: Selektionsanzeige über Drahtgitterdarstellung

**Multi-Pass Rendering:** Die Drahtgitterdarstellung könnte durch *hidden line removal* oder *haloed lines* visuell wesentlich verbessert werden. Jedoch muss für beide Darstellungen das Objekt mehrfach gerendert werden, was bei komplexen Szenen zu einem bemerkbaren Geschwindigkeitsverlust führen kann. Ähnlich verhalten sich hybride Darstellungen, z.B. gefüllte Darstellung (z.B. gouraud-shaded), bei der zusätzlich die Dreiecksgrenzen sichtbar gemacht werden.

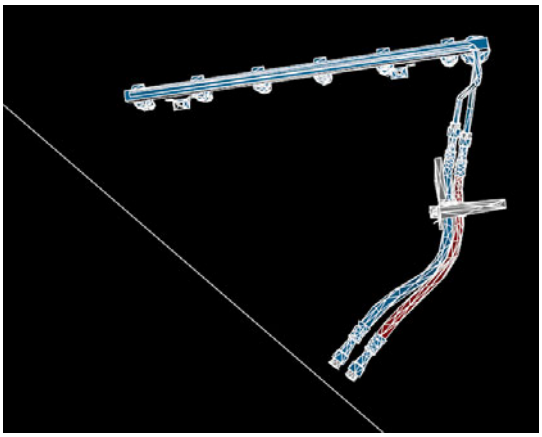

Abb. 4.19: „Scribed“ Darstellung

**Farb- und Helligkeitsänderung:** Hierbei wird die Farbe des selektierten Objektes verändert. Dies hat gegenüber den o.g. Techniken den Vorteil, dass die selektierten Objekte eindeutig identifizierbar sind und kein zusätzlicher Rendering-Pass notwendig ist. Selektierten Objekten eine bestimmte Selektionsfarbe zuzuweisen ist sicherlich in den meisten Fällen praktikabel, jedoch kann es auch hier passieren, dass ein Objekt in dieser oder einer ähnlichen Farbe eingefärbt ist. Die Selektion wäre dann nicht deutlich sichtbar. Um dieses Problem zu umgehen wird nicht die Objektfarbe verändert, sondern der ambiente Term erhöht und der diffuse Term erniedrigt. Dadurch erscheint das selektierte Objekt wesentlich heller als vor der Selektion und aufgrund der fehlenden Schattierung hebt es sich von der restlichen Szene stark ab.

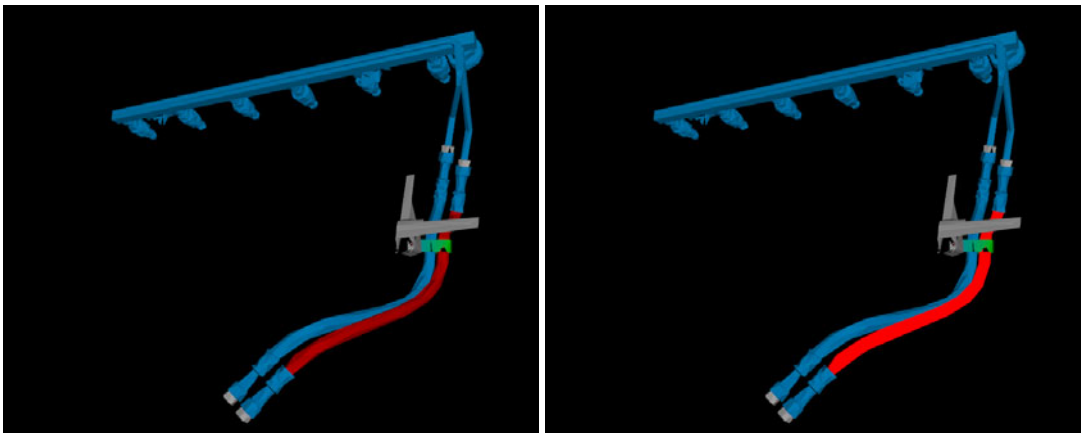

Abb. 4.20: Selektionsanzeige durch Änderung der Helligkeit. Hier wurde der rote Schlauch selektiert

## 4.5 BIT Menüselektion

Jedes Softwaresystem bietet eine Möglichkeit, um die von ihm zur Verfügung gestellten Funktionen aktivieren zu können. Dies kann von Eingabe eines Text-Kommandos (z.B. UNIX), über Auswahl aus einem hierarchischen Menüsystem (.z.B. MS-Windows) bis hin zu Spracheingabe gehen. Im folgenden wird aus Gründen der Einfachheit der Begriff *Menüselektion* anstatt *Auswahl aus einer relativ*

*festen Anzahl an Selektionsmöglichkeiten* verwendet, auch wenn strenggenommen nicht unbedingt Menüs zur Auswahl verwendet werden.

In diesem Kapitel werden unter Berücksichtigung der verschiedenen Problemstellungen Konzepte zur Umsetzung von Menüselektion in Virtuellen Umgebungen entwickelt. Hierbei wird zuerst auf Menüselektion über eine graphische Repräsentantion der Funktionsauswahl und im Anschluss daran die Auswahl mittels Sprachkommandos eingegangen.

### 4.5.1 Anforderungen

Im Zusammenhang mit einem Menüsystem gibt es verschiedenste Anforderungen, auf die im folgenden näher eingegangen wird.

**Authoring:** Für die Erstellung von Virtuellen Welten ist es natürlich sehr wichtig, auch Menüs auf einfache Art und Weise konfigurieren zu können. Insbesondere sollte es die Möglichkeit geben, auf Standardkomponenten zurückgreifen zu können, aus denen ein Menü zusammengesetzt werden kann. Gerade bei 3D-Menüs könnte sich das als schwieriger gestalten, da hier z.B. Text entweder als Polygonzug vorliegt oder eine Textur generiert werden muss. Auf der anderen Seite eröffnet diese Flexibilität auch die Möglichkeit in Handarbeit visuell sehr ansprechende Menüs zu erstellen. Solange diese jedoch nicht als Komponenten immer wieder eingesetzt werden können und nur für eine Anwendung nutzbar sind, ist der Aufwand für die meisten Applikationen zu hoch.

**Funktionsanzahl:** Wie in Kapitel 2.5 festgelegt, wird davon ausgegangen, dass das System über eine große Anzahl an Funktionen verfügt. Das Menüsystem muss also dahingehend ausgelegt sein, dass auch ein großer Funktionsumfang sinnvoll verwaltet werden und der Anwender ohne Probleme einzelne Funktionen aktivieren kann.

**Benutzbarkeit:** Wie auch bei allen anderen Interaktionstechniken steht hier die Benutzbarkeit im Vordergrund. Wichtig ist, dass der Anwender ohne große Vorkenntnisse mit dem System arbeiten kann und die Interaktion einfach und intuitiv ist.

### 4.5.2 Klassifikation

Neben den zu erfüllenden Anforderungen können bei der Menüselektion die verwendeten Techniken anhand verschiedener Parameter klassifiziert werden:

- Positionierung in der virtuellen Szene
- Menüform
- Darstellung (2D, 3D)
- Selektionsmethode

Auf diese Parameter wird in den folgenden Kapiteln eingegangen und im Anschluss eine Taxonomie erstellt.

#### 4.5.2.1 Positionierung

Menüs können auf verschiedene Arten in der Szene positioniert werden. Die unterschiedliche Positionierungsmöglichkeiten von Hilfsobjekten wurden in Kapitel 4.2.2 vorgestellt. Im folgenden wer-

den diese bewertet:

- **Fixed location:** Das Menü bleibt immer an der selben Stelle im Weltkoordinatensystem. Der wohl größte Nachteil ist, dass sobald der Anwender sich in der Szene herumdreht oder durch die Szene navigiert, das Menü für den Anwender nicht mehr sichtbar oder sogar außerhalb seiner Reichweite ist. Diese Positionierungsmethode ist dann sinnvoll, wenn der Anwender sich in einem definierten Bereich bewegt und die Blickrichtung definiert ist.
- **Head relative:** Hier bleibt das Menü immer im Blick und kann, falls durch die Szenerie verdeckt, durch eine einfache Kopfbewegung an eine andere Position versetzt werden. Dieser Modus ist jedoch nur dann gut benutzbar, wenn das Menü zentriert im Blickfeld des Anwenders erscheint. Bei Untersuchungen in einer CAVE hat sich gezeigt, dass alle Anwender größte Probleme mit dieser Methodik hatten. Da es kaum möglich ist, das Menü automatisch so zu positionieren, dass es vollkommen zentriert im Blickfeld erscheint, versuchen Anwender durch drehen des Kopfes diese Zentrierung zu erreichen. Dies hat zur Folge, dass sich das Menü ebenfalls bewegt. Diese Problematik tritt bei der Verwendung von HMD's nicht so stark auf, da hier der Augabstand manuell nachgeregelt werden kann.
  Weiterhin können Objekte, die mehr als 30° von der Fovea abweichen nur durch Kopfbewegung fixiert werden ([WAND93]), d.h. bei größeren Menüs ist es unmöglich diesen Modus überhaupt einzusetzen.
- **Cart relative:** Ein Anwendungsbeispiel für diese Positionierungsmethode sind die von [MINE97] entwickelten Pull-Down Menüs. Wenn die Pull-Down Menüs nicht verwendet werden, so befinden sie sich außerhalb des Sichtbereiches des Anwenders, z.B. über dem Kopf. Sollen die Menüs nun verwendet werden, so greift der Anwender über seinen Kopf und zieht die Menüs herunter, ähnlich einem Rolladen. Diese Technik hat zwei Vorteile. Zum einen werden Teile des Bildschirm nicht mit einem Menü verdeckt und zum anderen wird für das Öffnen des Menüs keine Triggermöglichkeit auf dem Eingabegerät belegt.
- **Cursor relative:** Das Menü wird relativ zu einem Cursor (z.B. die virtuelle Hand) positioniert und bleibt nach dem Erscheinen an dieser Stelle positioniert, da dann der Cursor für die Selektion eines Eintrages verwendet wird (an dieser Stelle wird von zweihändiger Interaktion abgesehen, siehe hierzu u.a. [WLOK95]). Ein Beispiel aus dem Bereich von 2D-Anwendungen ist das kontextsensitive Menü (u.a. unter MS-Windows 95), welches mit der rechten Maustaste geöffnet wird und an der Cursorposition erscheint.

#### 4.5.2.2 2D-Menüs versus 3D-Menüs

Menüs können in Virtuellen Umgebungen entweder als 3 dimensionales Objekt der Szene oder als 2D Bild, welches über der Szene liegt (Overlay), realisiert werden. Man spricht hierbei auch von 3D-Menüs und 2D-Menüs.

Im folgenden wird hierbei auf die wichtigsten Unterschiede und Problemstellungen näher eingegangen.

**Direkte und Indirekte Interaktion:** Der wohl größte Unterschied zwischen diesen beiden Darstellungsarten ist sicherlich die Möglichkeit der direkten Interaktion. So kann bei 3D-Menüs mittels der virtuellen Hand durch Berühren des Objektes ein Menüpunkt ausgewählt werden, wohingegen bei 2D-Menüs indirekt über die virtuelle Hand ein Cursor gesteuert wird. Die direkte Manipulation ist

nicht möglich, da die 2D-Menüs grundsätzlich nicht Teil der Szene sind und auch keinen definierten Tiefenwert besitzen. Auf den ersten Blick mag dies als sehr ungewöhnlich und als Einschränkung erscheinen, ist jedoch der Interaktion mit der Maus vor einem Bildschirm sehr ähnlich, da auch hier über die Maus indirekt ein Cursor gesteuert wird. Eine andere Möglichkeit der Steuerung wird in Kapitel 4.5.4.1 vorgestellt.

**Stereodarstellung:** Da 3D-Menüs Teil der Szene sind, werden sie bei Stereodarstellung korrekt angezeigt. 2D-Menüs haben jedoch keinen definierten Tiefenwert. Liegen z.B. die Bilder für linkes und rechtes Auge des dargestellten 2D-Menüs direkt übereinander (keine horizontale Disparität) kann es in Szenen, in denen die Objekte weit vom Betrachter entfernt sind, zu einem unangenehmen Effekten führen. Dadurch, dass die Augen des Betrachters auf einen entfernten Punkt fixiert sind und dann in sehr kurzer Distanz ein Menü erscheint, müssen die Augen erst auf diesen neuen Abstand adaptieren. Die meisten Anwender, die mit dieser Menüdarstellung gearbeitet haben, hatten durchgängig das Gefühl, beim Erscheinen des Menüs von ihm „erschlagen“ zu werden.
Um diesen Effekt zu vermeiden, sollte zur Darstellung eine positive horizontale Disparität verwendet werden. Dann erscheinen die Menüs in größerer Entfernung (hinter der Projektionsebene) und die Augen müssen sich nicht mehr so stark an die neuen Verhältnisse anpassen. Siehe hierzu auch Abb. 4.21

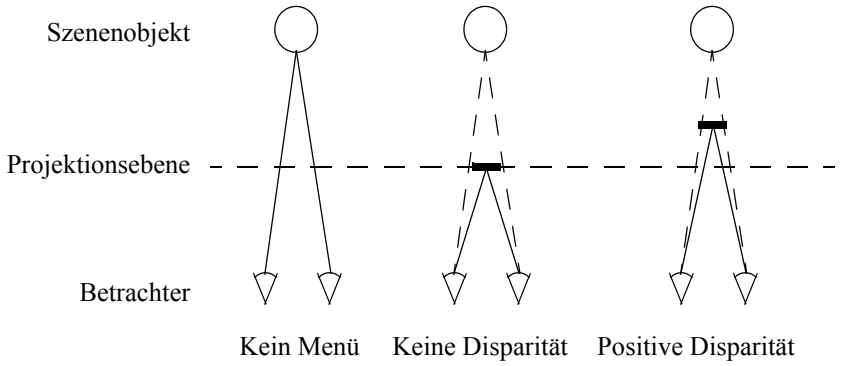


Abb. 4.21: Stereodarstellung von 2D-Menüs

**Positionierung:** Wie in Kapitel 4.2.2 beschrieben, gibt es verschiedene Möglichkeiten Objekte zu positionieren. Diese Positionierungsarten können für jegliche Art von 3D-Menü verwendet werden. Dadurch dass 2D-Menüs jedoch grundsätzlich immer in der Bildebene erscheinen, ist diese Unterscheidung nur dahingehend relevant, an welcher x,y-Position das Menü erscheint (cart relative), d.h. entweder an einer fest vorgegebenen Position oder unter Einbeziehung der 3D-Position des Cursors, aus der dann eine x,y-Position abgeleitet wird (cursor relative).

**Verdeckung:** Es gibt zwei Arten der Verdeckung: Verdeckung des Menüs durch die Szene und Verdeckung der Szene durch das Menü. Der erste Fall kann nur bei 3D-Menüs eintreten, die dynamisch in der Szene plaziert werden, z.B. in einem bestimmten Abstand vor dem Betrachter (*cart relative*). Da dynamisch plazierte Menüs erst auf „Knopfdruck“ sichtbar werden und dann an einer festen Position bleiben, kann der Anwender es bei komplexen Szenen fast nicht vermeiden, dass Teile des Menüs durch Szenenobjekte verdeckt werden. Hier muss er dann das Menü schließen und das System

dazu bringen beim nächsten Öffnen des Menüs es an einer anderen Stelle zu plazieren. Bei 2D-Menüs kann dieser Fall nie eintreten, da sie grundsätzlich über der Szene dargestellt werden.
Beide Menütypen können jedoch Teile der Szene verdecken. Da jedoch der Anwender im Moment der Menüauswahl auf das Menü konzentriert ist und weniger auf die Szenerie, stellt dies ein eher untergeordnetes Problem dar. Jedoch hat sich hierbei gezeigt, dass sehr große Menüs im Moment des Anzeigens für den Anwender sehr unangenehm sind, da innerhalb kürzester Zeit ein sehr großes Objekt in seinem Gesichtsfeld erscheint. Deswegen sollten Menüs nur so groß wie nötig sein. Weiterhin kann die Darstellung dadurch angenehmer gestaltet werden, dass Menüs langsam eingeblendet werden oder auch teiltransparent sind. Dadurch erscheint das Menü visuell „leichter“.

**Textdarstellung und Pictogramme:** Da 2D-Menüs direkt über der Szene dargestellt werden, können Schriften, als auch Pictogramme in der passenden Größe in Pixeln generiert werden. Werden Pictogramme oder Schriften auf 3D- Objekte mittels einer Textur aufgebracht, ist je nach Entfernung des Objektes vom Betrachter die Schrift / Pictogramm entweder verwaschen oder „pixelig“. Dies kann zwar durch Mip-Mapping teilweise unterdrückt werden, ist jedoch qualitativ nicht so hochwertig, wie eine direkte 1:1 Darstellung, ohne perspektivische Verzerrung oder Pixel-Zoom.

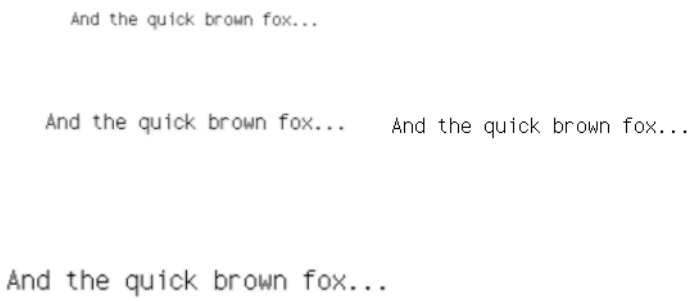


Abb. 4.22: Vergleich zwischen Text als Textur (links) und als Overlay (rechts)

Ein weiterer Nachteil des Texturansatzes ist, dass zur Benutzung des Menüsystems die Graphikhardware Texturen unterstützt muss, da ansonsten die Darstellungsgeschwindigkeit einbricht. Um die Darstellungsqualität insbesondere von Schrift zu verbessern, könnte diese auch polygonal dargestellt werden. Dies ist jedoch sehr aufwendig und verlagt u.U. sehr viel Graphikrechenleistung.
2D-Menüs hingegen können Text als auch Pictogramme sehr einfach als Bitmap darstellen und stellen somit kaum Anforderungen an die vorhandene Graphikhardware. Auch lassen sich jegliche Graphiken wie auch geometrische Figuren durch einfaches 2D Rendering erzeugen.

#### 4.5.2.3 Selektion von Menüeinträgen

Menüeinträge können prinzipiell als Objekte der Szene verstanden werden. Dies gilt jedoch nur, wenn sie in 3D-Menüs eingebettet sind. Somit können sie z.B. wie in Kapitel 4.4 beschrieben, selektiert werden. Aufgrund des speziellen Charakters von Menüs und der Möglichkeit, diese auch in 2D zu realisieren, sind hier jedoch auch noch zusätzlich andere Möglichkeiten gegeben. Im folgenden werden die verschiedenen Selektionstechniken beschrieben.

- **Ray-Casting:** Wie bereits weiter oben beschrieben, können Objekte per virtuellen Laserstrahl

selektiert werden. Dies ist jedoch prinzipiell nur bei 3D-Menüs möglich.

- **Image Plane Techniken und virtueller Cursor:** Auch diese Technik wurde weiter oben bereits beschrieben, ist jedoch nur für 3D-Menüs sinnvoll anwendbar. Jedoch kann diese Technik auch dahingehend erweitert werden, so dass sie für 2D-Menüs anwendbar wird. Hierbei „peilt“ der Anwender nicht mit Hilfe eines 3D-Objektes (bei [PIER97] die virtuelle Hand) das zu selektierende Objekt an, sondern über einen virtuellen 2D-Cursor, der in der Bildebene des 2D-Menüs erscheint. Dieser Cursor wird dabei über Translation der dominanten Hand gesteuert werden.
- **Direct Select:** Selektion durch „Berühren“ des Menüeintrages mit der virtuellen Hand. Dies kann jedoch als Sonderfall der Image Plane Technik angesehen werden und wird deswegen im folgenden nicht näher betrachtet.
- **Selektorrotation:** Im Gegensatz zum virtuellen Cursor wird bei dieser Selektionsmethode nicht die Translation betrachtet, sondern die Rotation der dominanten Hand. Über die Rotation kann dann entweder das Menü direkt oder wie beim virtuellen Cursor ein entsprechendes Proxyobjekt gesteuert werden.
- **Kopfrotation:** Durch „Ansehen“ (Auswertung der Kamerablickrichtung) eines Menüeintrages wird dieser selektiert. Kann sowohl für 2D-, als auch 3D-Menüs verwendet werden.
- **Reale Taster:** Die sicherlich einfachste Methodik ist das Abbilden von Funktion auf reale Tasten des Eingabegerätes.

#### 4.5.2.4 Taxonomie

Die hier beschriebenen Parameter können nun dazu verwendet werden, um die verschiedenen Metaphern zu klassifizieren und ihre Abhängigkeiten darzustellen. Diese Taxonomie wird in Abb. 4.23 aufgezeigt. Hierbei werden die in den folgenden Kapiteln vorgestellten Menüformen und Interaktionstechniken mit einbezogen.

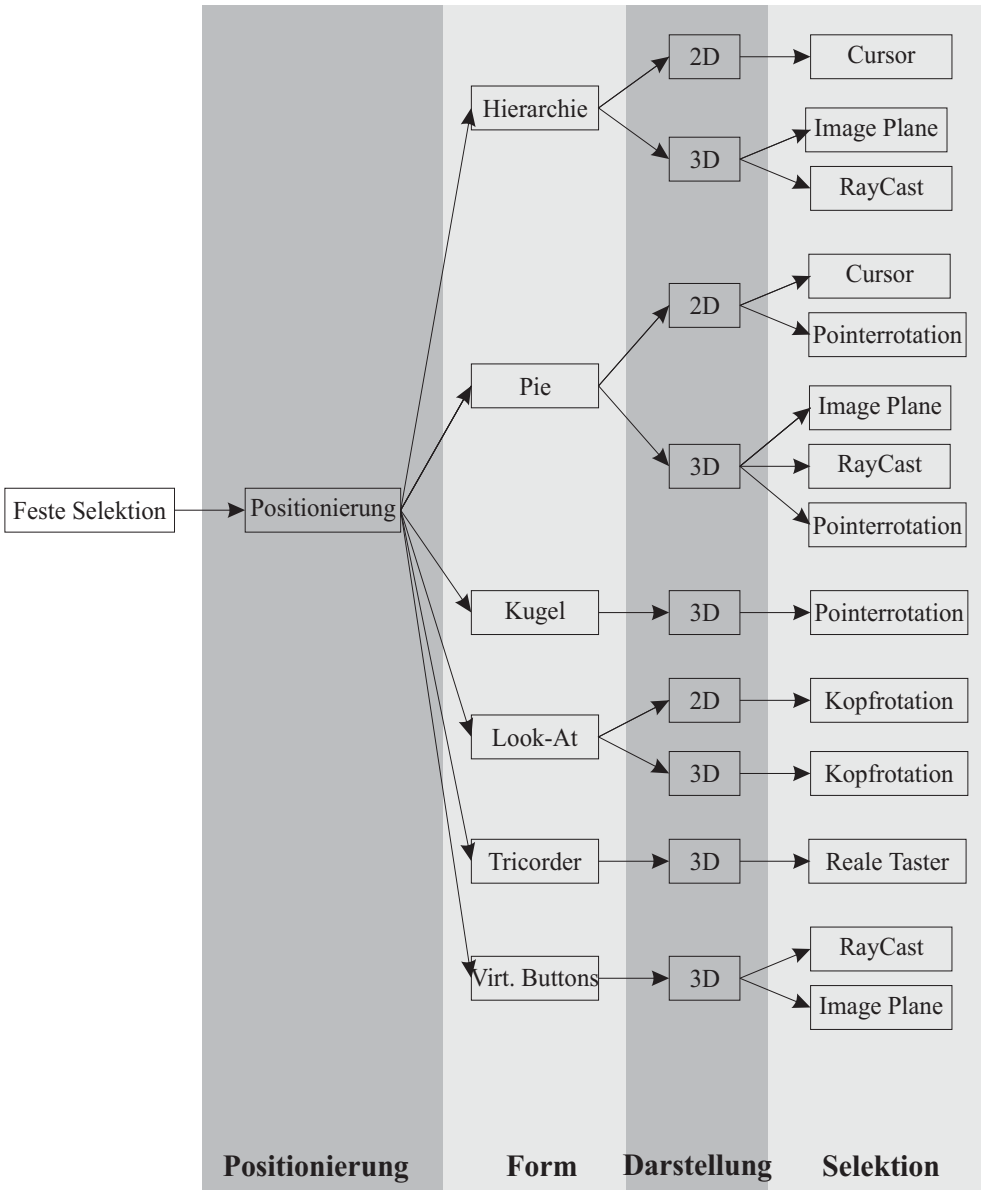


Abb. 4.23: Taxonomie der verschiedenen Menüarten und Interaktionstechniken

### 4.5.3 Ansätze aus der Literatur

Im folgenden werden nun die verschiedenen Ansätze aus der Literatur vorgestellt.

#### 4.5.3.1 Virtual Tricorder

Der Grundgedanke des von [WLOK95] entwickelten Virtual Tricorder ist sehr interessant. Hierbei ist das eigentliche Menü an der virtuellen Hand befestigt, welche über ein getracktes Eingabegerät gesteuert wird. Bei Manipulationen mit dem Cursor ist das Menü jederzeit im Sichtfeld und der Anwender kann sofort erkennen, welche Funktionen er zur Verfügung hat, ohne den Blick abwenden zu müssen. Ist das Menü im Weg kann der Anwender einfach das Eingabegerät aus dem Blickfeld nehmen. Die einzelnen Menüfunktionen werden über zugehörige Knöpfe auf dem Eingabegerät ausgelöst. Dies ist natürlich bei Eingabegeräten wie Stylus und Flystick nicht tragbar, da hier viel zu we-

nige Knöpfe zur Verfügung stehen. Selbst bei einer Spacemouse stehen nur 8 Knöpfe zur Verfügung, was für komplexe Systeme sicherlich noch zu wenig ist.

#### 4.5.3.2 Virtual Buttons

Virtual Buttons finden u.a. in [FELG95] Erwähnung und sind geometrische Objekte der Szene, die sich an einer festen Position im Raum befinden. Durch z.B. berühren mit der virtuellen Hand wird die ihnen zugehörige Funktion ausgelöst (siehe Abb. 4.24).

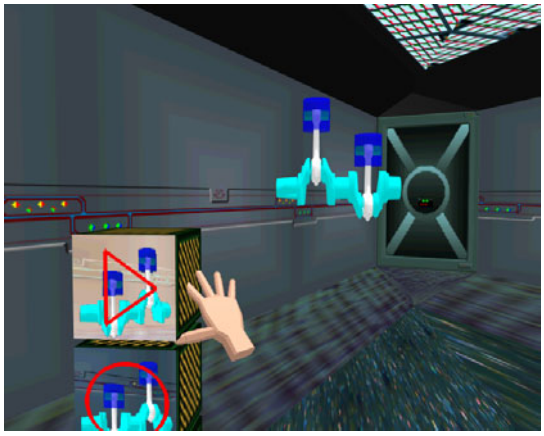

Abb. 4.24: Virtuelle Buttons

#### 4.5.3.3 Allgemeine 3D-Menüs

3D-Menüs fassen eine Gruppe von Virtual Buttons zu einem Menü zusammen. Die Menüs können entsprechend Kapitel 4.5.2.2 positioniert werden. Die einzelnen Menüpunkte können mittels der in Kapitel 4.4 beschriebenen Selektionstechniken ausgewählt werden. Diese Form der Menüs werden u.a. im VR-System Virtual Design 2 verwendet ([ASTH95b]).

#### 4.5.3.4 Look-At-Menüs

Look-At Menüs ([MINE97]) sind lineare Menüs, bei denen die Menüeinträge in einer Ebene vor dem Anwender angeordnet sind. Durch das Drehen des getrackten Kopfes und ansehen eines Menüeintrages wird dieser ausgewählt.

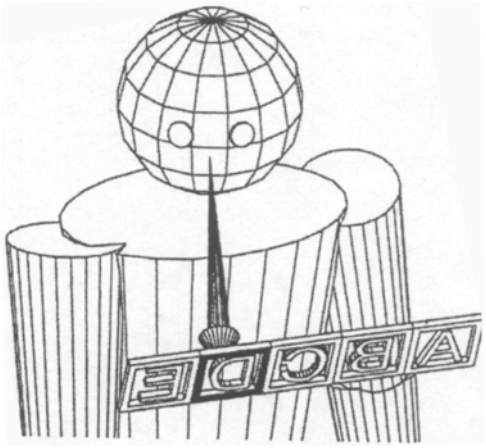

Abb. 4.25: Look-At-Menüs (aus [MINE97])

Aus Sicht eines Anwenders wird somit die Auswahl über seine „Blickrichtung“ gesteuert (Gedankenmodell). Das Systemmodell weicht hiervon jedoch ab, weil nur die Kopfrotation ausgewertet wird, jedoch nicht die Stellung der Augen (eye-tracking). Dies ist für den Anwender sehr irritierend und in der Anwendung unnatürlich. Wie entscheidend die Orientierung der Augen ist, zeigt folgendes kleines Rechenexempel:

Lt. [WAND93] kann alleine durch die Bewegung der Augen bereits ein Sehbereich von 30° abgedeckt werden. Gehen wir von einem 0,5m entfernten Menü mit 5 horizontalen Einträgen gleicher Breite und einer Menübreite von 0,5m aus, so liegen immer mehrere Einträge im direkten Blickfeld. Eine Kopfdrehung von 9° reicht aus, um den nächsten Eintrag auszuwählen. Siehe hierzu auch Abb. 4.26.

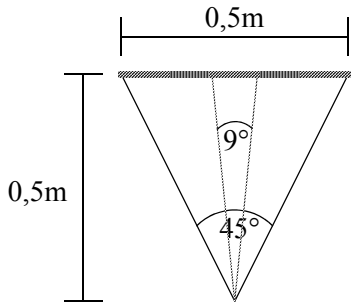


Abb. 4.26: Betrag der Kopfrotation bei Look-At Menüs

#### 4.5.3.5 Interaction Ball

Der Interaction Ball wurde von [HAEF99] entwickelt. Der Interaction Ball ist ein geometrisches Hilfsobjekt in Form einer 3D-Kugel, auf der 4 Menüpunkte aufgebracht sind (Abb. 4.27).

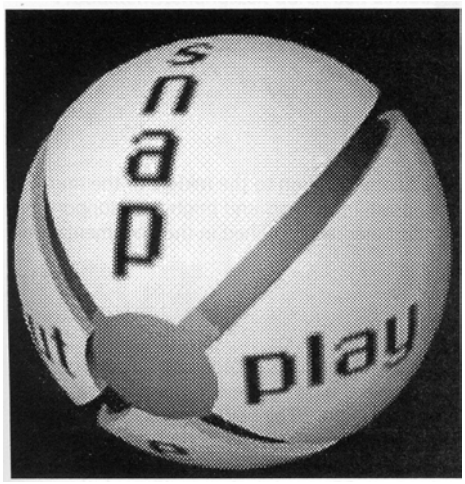


Abb. 4.27: Der Interaction Ball

Die Kugel wird über einen Event sichtbar und wird cart-relativ positioniert. Über die Rotation der Hand kann die Kugel um ihr Zentrum gedreht werden und der dann sichtbare Menüpunkt wird ausgeführt. Dadurch das die Handrotation direkt auf die Kugelrotation gemapped wird, ist die Anwendung sehr intuitiv.

Es hat sich gezeigt, dass zusammengesetzte Rotationen, um z.B. 8 Menüeinträge zu realisieren, bereits aus physischer Sicht schon sehr schwierig auszuführen sind. Somit können hier max. 4 Menüpunkte pro Ebene angeboten werden. Der Metapher scheint sehr sinnvoll für Systeme mit geringer Komplexität zu sein.

#### 4.5.3.6 TULIP-Menüs

In [BOW01] werden die sog. Tulip-Menüs vorgestellt, die jedoch auf einer speziellen Hardwaretechnologie, den Pinchgloves(tm), aufsetzen. Pinchgloves werden wie ein Datenhandschuh verwendet, sind jedoch an jeder Fingerspitze mit einem Schalter ausgerüstet. So können über Berühren der Fingerspitzen von Daumen und einem der anderen vier Finger, vier unterschiedliche Trigger ausgelöst werden.

Interessanterweise beschreibt [BOW01] das Design des 1. Prototypen und den dabei aufgetretenen Problemen, z.B. Ermüdungserscheinungen des Arms, fehlendes Feedback, arbiträre Orientierung von 3D-Text im Raum und Dissens zwischen Gedankenmodell und Systemmodell. Probleme, die auch in dieser Arbeit adressiert werden und u.a. im Grundlagen-Kapitel (Kapitel 3) behandelt wurden.

In einer zweiten Iteration wurde das Design dann entsprechend angepasst, dass diese Probleme nicht mehr so stark zu Tage traten. Für eine genaue Beschreibung der Funktionsweise sei der Leser auf die Veröffentlichung und die folgende Abbildung verwiesen.

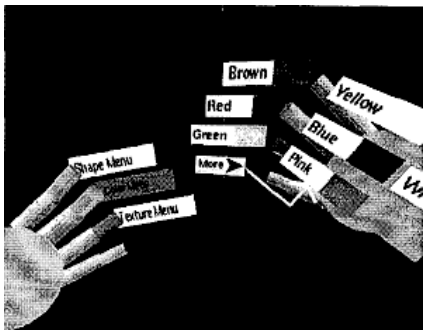


Abb. 4.28: TULIP-Menü (aus [BOW01])

Aufgrund der definierten Hardwarevoraussetzungen wird dieser Ansatz hier nicht weiter verfolgt.

### 4.5.4 Menüformen: Hierarchisch versus Pie

Wie bereits in Kapitel 4.5.1 erläutert, muss bei dem Entwurf eines Menüsystems auf einige Problempunkte geachtet werden. So haben sicherlich sowohl 2D als auch 3D-Menüs ihre Vorteile, jedoch wurde bei der Konzeption des Rahmensystems der 2D Ansatz dem 3D Ansatz vorgezogen, da der 2D Ansatz in einigen Bereichen überlegen ist und dadurch auch Nachteile kompensieren kann (siehe Kapitel 4.5.2.2).

2D Menüs können auf verschiedene Arten dargestellt werden. Die wohl bekannteste ist den Pop-Up Menüs der MS-Windows-Welt entlehnt, d.h. eine vertikale Anordnung der Menüpunkte, Untermenüs öffnen sich rechts oder links des aktiven Menüs. Der wohl größte Vorteil dieser Repräsentation ist der allgemeine Bekanntheitsgrad. Da die meisten Anwender bereits schon einmal unter MS-Windows gearbeitet haben und sich somit mit der Verhaltensweise dieser Menüs auskennen.

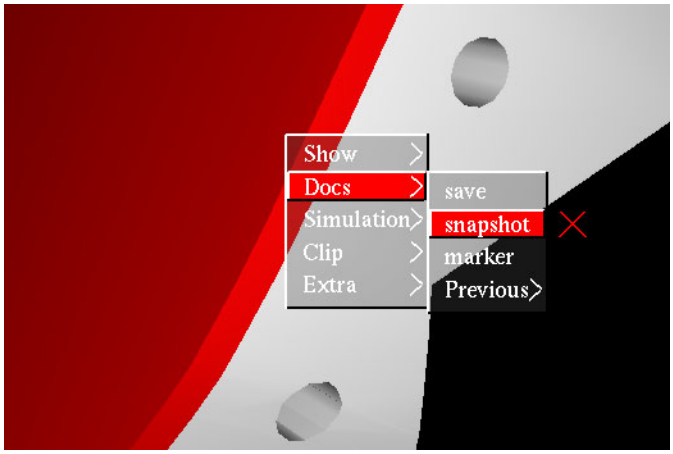


Abb. 4.29: 2D-Menü in „Windows“-Style

Eine andere Möglichkeit ist die Darstellung als sog. Pie-Menü. Hierbei sind die Menüeinträge nicht vertikal , sondern kreisförmig angeordnet. Beim öffnen des Menüs befindet sich der Cursor im Zentrum des Kreises. Durch die kreisförmige Anordnung haben alle Menüeinträge den selben Abstand

zum Kreiszentrum, dem Ausgangspunkt des Cursors. Der Kreis ist entsprechend der Anzahl der Menüoptionen in Sektoren gleicher Größe eingeteilt, wobei jeder Sektor einem Menüeintrag entspricht. Siehe hierzu die folgende Abbildung:

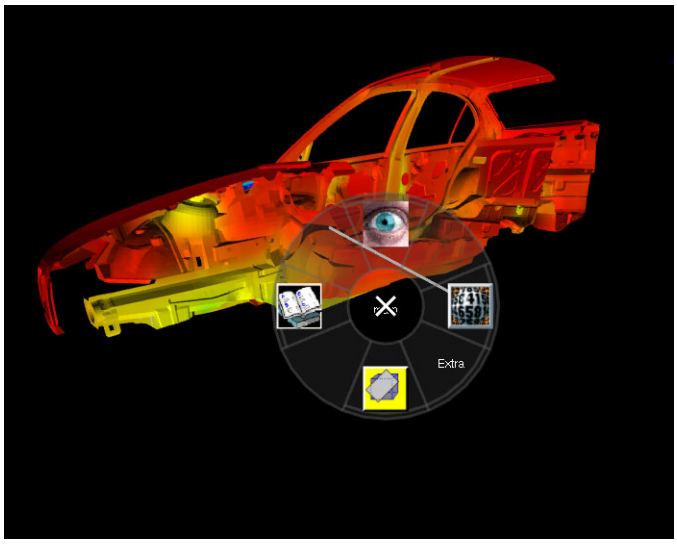


Abb. 4.30: Das Pie-Menü

Aufgrund dieser Anordnung ist die „Zieldistanz“ (Abstand Startpunkt des Cursors zu Menüeintrag) für alle Menüeinträge in dieser Ebene konstant. Die „Zielgröße“ (Bereich, in dem ein Menüeintrag ausgewählt ist) vergrößert sich mit dem Abstand des Cursors von der Ausgangssituation.

Bei linearen Menüs hingegen ist die Zielgröße pro Menüebene konstant, die Zieldistanz variabel. Die vertikale Zieldistanz skaliert dabei linear mit der Anzahl der Menüpunkte, die lineare Zieldistanz ist jedoch abhängig von der Menübreite, die je nach Inhalt stark variieren kann. Es hat sich gezeigt, dass eine Normierung (entweder Menübreite oder zurückzulegende Strecke über adaptive Skalierung) sinnvoll ist.

Beide vorgestellten Darstellungsarten wurden im Rahmen dieser Arbeit realisiert. Es zeigte sich, dass die bereits in [CALL88] und [MILLS90] beschriebenen Ergebnisse in sehr viel stärkerem Maße auch auf Virtuelle Umgebungen übertragbar sind. Dort wurde in empirischen Untersuchungen festgestellt, dass die kreisförmige Anordnung etwas schneller und vor allem fehlerfreier erfolgte, als bei linearer Anordnung in einer Menüspalte

Die linear angeordneten Menüs waren zwar von der Bedienung bekannt, das Auswählen eines Menüpunktes gestaltete sich jedoch als äußerst schwierig. War die Skalierung „Reale Hand auf Cursor“ zu niedrig, so reichte schon ein leichtes Zittern, um den falschen Menüpunkt auszuwählen, war der Schwellwert zu groß, so mußten sehr lange Wege mit der Hand zurückgelegt werden (Ermüdungserscheinungen). Das „Lesen“ eines Menüs und „Suchen“ eines Menüeintrages gestaltet sich bei der linearen Anordnung ebenfalls als sehr kompliziert. Während bei Desktop-Anwendungen in diesem Moment die Hand entspannt auf der Maus liegt, muss in VR die Hand ruhig im Raum gehalten werden. Hierbei kann es dann jedoch sehr einfach passieren, dass aus Unachtsamkeit und Ermüdung Menüpunkte aktiviert oder Untermenüs geöffnet werden. Bei Pie-Menüs wird dieses Problem über die

zentrale Aussparung umgangen (s.u.).

Im direkten Vergleich schnitten die Pie-Menüs hinsichtlich der Bedienung wesentlich besser ab und wurden auch seitens der Anwender als einfach und intuitiv benutzbar bewertet.

Im folgenden wird nun näher auf die Pie-Menüs eingegangen und die unterschiedlichen Attribute beleuchtet, die auf Basis der Grundidee diese einfache Bedienung ermöglichen.

#### 4.5.4.1 Pie-Menüs: Steuerung und Selektion

Damit nicht bereits durch kleinste Bewegungen des Cursors ein Menüeintrag oder im schnellen Wechsel verschiedene Einträge angewählt werden, muss zuerst eine untere Distanzschwelle $d_{ti}$ vom Zentrum $c$ aus überschritten werden, bevor ein Sektor ausgewählt wird. Um auch einfache Deselektion zu ermöglichen ist bei Überschreiten einer Distanz $d_{to}$ kein Eintrag selektiert. Diese beiden Schwellen werden ebenfalls graphisch dargestellt, indem die Sektoren sowohl innen, als auch außen durch ein Kreissegment beschnitten sind. Somit ist ein *undo* auf Menüebene sehr leicht möglich. Menünamen oder Pictogramme werden zentriert auf einer Kreisbahn mit dem Abstand $d_{tm}$ vom Zentrum $c$ dargestellt, wobei gilt: $d_{ti} < d_{tm} < d_{to}$. Zur Veranschaulichung siehe Abb. 4.31

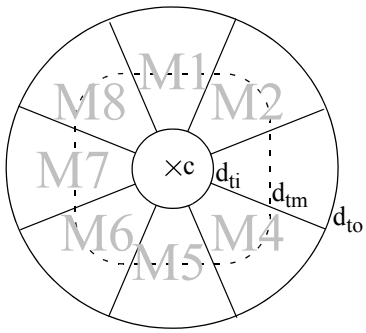


Abb. 4.31: Schematischer Aufbau der Pie-Menüs

Sobald ein Menüeintrag angewählt wurde (der Cursor befindet sich im Sektor), wird dieses farblich hervorgehoben.

Handelt es sich bei dem selektierten Menüeintrag um einen Übergang zu einem Untermenü, so wird dieser beim Überschreiten einer Distanzschwelle $d_{tm}$ automatisch geöffnet. Es ist also nicht nötig, dass der Anwender einen speziellen Event auslöst oder durch eine unachtsame Bewegung (Zittern) bereits in das Untermenü rutscht. Das Zentrum des Untermenüs ist das Zentrum des Kreissegments, welches durch den Sektor und die Distanz $d_{tm}$ gegeben ist. Das Untermenü wird daraufhin über dem alten Menü eingeblendet, wobei der selektierte Eintrag durch die innere Aussparung sichtbar bleibt. Somit verliert der Anwender nicht die Orientierung und weiß in welchem Untermenü er sich befindet. Das vorherige Menü wird ausgeblendet und im Sektor für den Übergang in das vorherige Menü wird der Name oder das Pictogramm dieses Menüs dargestellt. Dieser Sektor liegt genau entgegen der Bewegungsrichtung.

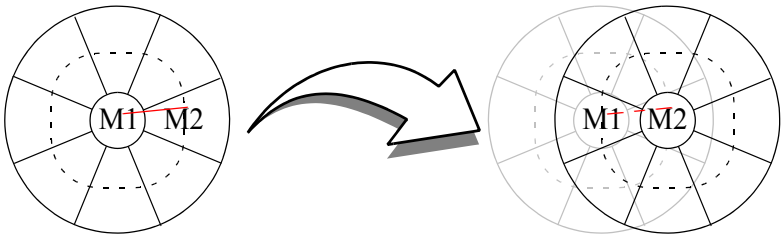


Abb. 4.32: Übergang von einem Menü zu einem Untermenü

Als vorteilhaft hat sich hier auch erwiesen, den Weg durch die verschiedenen Untermenüs dadurch zu kennzeichnen, dass diese nach dem Öffnen des nächsten Menüs nicht vollständig ausgeblendet, sondern halbtransparent dargestellt werden.

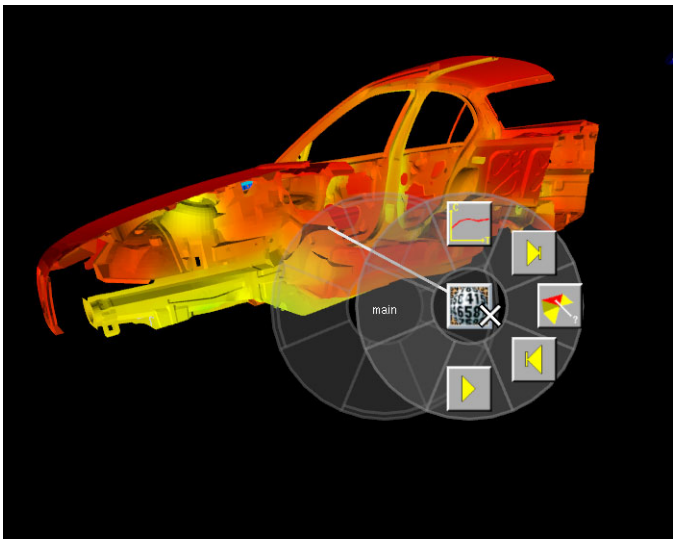


Abb. 4.33: Menüübergang bei Pie-Menüs

Repräsentiert der gewählte Menüeintrag jedoch eine Funktion, so muss der Anwender zusätzlich einen Event auslösen, um diese auszuführen. Es reicht somit nicht aus, nur die Schwelle $d_{tm}$ zu überschreiten.

Wie bereits in Kapitel 4.5.2.2 erwähnt, werden 2D-Menüs indirekt gesteuert. Die Abbildung der Translation des Eingabegerätes auf die Bewegung des Cursors ist hierbei abhängig vom verwendeten Eingabegerät:

- *Nicht-kopfgebundene Projektion (z.B: CAVE):* Wenn Anwender mit dem System interagieren, so stehen sie meistens frontal vor der Wand. Hier kann somit für die Berechnung der Cursorbewegung der *cart* verwendet werden. Dadurch wird garantiert, dass eine links-rechts Bewegung der realen Hand auch den Cursor entsprechend bewegt.
- *Kopfgebundene Projektion (z.B. HMD):* Die Orientierung des Anwenders im realen Raum wird hier nicht vorgegeben und kann beliebig sein, so dass hier besser das *camera*-Koordinatensysteme verwendet werden sollte. Es hat sich gezeigt, dass während der Menüselektion nicht das in

diesem Moment aktuelle, sondern nur das beim Öffnen des Menüs aktuelle *camera*-Koordinatensystem für die Berechnung verwendet werden sollte. Dadurch haben Kopfrotationen während der Selektionsphase keine Auswirkung auf die eigentliche Interaktion.

#### 4.5.4.2 Pie Menüs: Optimale Anzahl der Menüeinträge

Im Rahmen dieser Arbeit wurde mit einer unterschiedlichen Anzahl an Menüeinträgen pro Menüebene experimentiert. Es stellte sich sehr schnell heraus, dass die sinnvollen Werte zwei, vier und acht Einträge pro Ebene waren. Hierbei ist die Bewegungsrichtung der Hand entweder rein horizontal, rein vertikal oder schräg, wobei sich diese Richtung aus gleichen Anteilen horizontaler und vertikaler Bewegung zusammensetzt. Im allgemeinen sind dies Bewegungen, die ein Mensch intuitiv ausführen kann.

In der eigentlichen Realisierung wurde die Anzahl der Menüeinträge pro Ebene auf exakt acht festgelegt, unbenutzte Einträge werden leer dargestellt. Es hat sich auch gezeigt, dass 8 Einträgen pro Menüebene den meisten Anforderungen gerecht werden können. Dies ist auch ein Richtwert, bei dem gute Suchzeiten in Menüs zu erwarten sind. So hat u.a. [KIGER84] Untersuchungen bezüglich der Menübreite und Menütiefe durchgeführt. Hierbei zeigte sich, dass bei kleinen Menübreiten (w=2, w=4) und großer Tiefe (d=6, d=3) die längsten Suchzeiten und bei w=8 und d=2 die kürzesten Suchzeiten zu erwarten sind. Diese Untersuchungen werden u.a. von [MILL81] bestätigt.

Menüs mit einer größeren Anzahl an mögliche Einträgen werden meistens nur dann benötigt, wenn z.B. die Objekte einer Szene über Namen ausgewählt werden sollen. Hier kann auch keine maximale Obergrenze abgeschätzt werden. Hier müssen dann diese Einträge in Untermenüs geclustert oder auf eine andere Darstellungsform zurückgegriffen werden.

#### 4.5.4.3 Pie Menüs: Optimiertes Layout der Menüeinträge

Als sehr vorteilhaft hat es sich erwiesen, wichtige Funktionen auf die Hauptachsen zu legen, insbesondere die horizontale und die vertikale Achse.

In der Umsetzung des Systems ist z.B. das Simulationsmenü im rechten Sektor und in diesem Untermenü befindet sich dann ebenfalls im rechten Sektor die Funktion *Data Probe*, mit der ein Simulationsdatensatz mittels eines virtuellen Laserstrahls untersucht werden kann. Diese Funktion ist eine der wichtigsten im gesamten System. Da sie auf einer der Hauptachsen liegt kann sie in diesem Fall einfach durch eine Translation nach rechts ausgewählt werden, der Anwender muss somit bei der Auswahl nicht in bestimmten Untermenüs die Bewegungsrichtung ändern.

Neben der Möglichkeit manuell die max. 8 Einträge für das Hauptmenü und max. 7 Einträge für die Untermenüs bestimmten Sektoren zuzuweisen, gibt es auch vom System aus die Möglichkeit, dies automatisch zu tun (*automatic layout)*. Hierbei wird jeweils die Bewegungsrichtung des Cursors beim Öffnen des Untermenüs beachtet. Alle Menüeinträge des Untermenüs werden so angeordnet, dass möglichste wenig von der Bewegungsrichtung abgewichen werden muss, in dem die Optionen um den in Bewegungsrichtung liegenden Sektor angeordnet werden. Siehe hierzu Abb. 4.34.

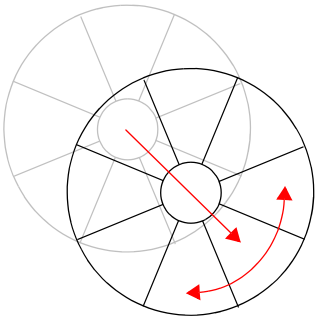

Abb. 4.34: Automatische Anordnung

#### 4.5.4.4 Selektion über Rotation

Bis jetzt wurde nur die Translation als Möglichkeit, den Cursor zu steuern, in Betracht gezogen. In VR stehen jedoch prinzipiell alle 6 Freiheitsgrade zur Verfügung. Aus diesem Grund wurde auch untersucht, ob eine Cursorsteuerung über Rotation möglich und evtl. sogar der Translation überlegen ist. Im folgenden wird nun eine Rotationssteuerung für Pie-Menüs vorgestellt.

Für die Handrotation wird die Rotation um das Handgelenk betrachtet. Mittels des Handgelenks kann um 3 Achsen rotiert werden (siehe Abb. 4.35).

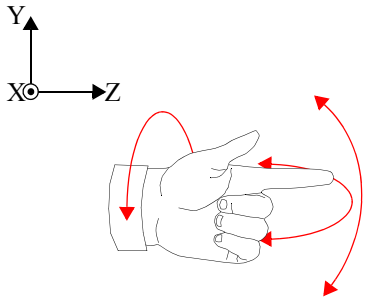


Abb. 4.35: Mögliche Rotationen der Hand

Dadurch dass die Rotationen um die 3 Achsen recht eingeschränkt sind, bietet es sich an, die Rotation als binäres Ereignis zu verstehen und nicht den Rotationswinkel für die Steuerung zu verwenden. Ein binäres Ereignis ist somit ausgehend von der Handgrundstellung (Abb. 4.35) eine Rotation in eine Richtung bis ein bestimmter Schwellwert (Rotationswinkel) überschritten wurde und eine anschließende Rotation zurück in die Grundstellung, um weitere „Ereignisse“ erzeugen zu können. Somit stehen durch Rotation 6 verschiedene Ereignisse zur Verfügung, pro Achse jeweils eine Rotation in positiver, als auch in negativer Richtung.

Damit der Anwender nicht mühevoll eine bestimmte Stellung einnehmen muss, wenn die Menüs geöffnet werden, wird die Handposition und -orientierung beim Öffnen des Menüs als Grundstellung

angenommen.

Es gibt nun verschiedene Möglichkeiten die Ereignisse auf die Auswahl von Menüpunkten abzubilden. Geht man von einem Pie-Menü mit 4 Menüpunkten aus, so kann über Rotation um die Y-Achse der linke oder rechte Sektor ausgewählt werden. Über die Rotation um die X-Achse kann der obere und untere Sektor ausgewählt werden. Sobald der Rotationswinkel über dem Schwellwert liegt, wird der Sektor ausgewählt und die dahinterstehende Funktion ausgelöst. Handelt es sich hierbei um den Übergang in ein Untermenü wird dieses zwar geöffnet, aber das Auswählen eines Sektors in diesem Untermenü ist erst möglich, wenn die Hand wieder in die Grundstellung zurückbewegt wurde. Ansonsten würde in diesem Untermenü die Funktion ausgewählt werden, die sich im selben Sektor befindet, wie der Übergang in dieses Untermenü. Dies hätte eine Kaskade zur Folge, die erst durch eine Funktion oder einen leeren Sektor beendet wäre. Eine andere Möglichkeit wäre, die Selektion des Sektors erst dann auszulösen, wenn die Hand wieder in der Grundstellung ist.

Ein großer Vorteil dieser beiden Methoden ist, dass der Anwender sich nach einiger Übung die Position eines Menüpunktes über ein Kommando in der Form „hoch-hoch-links“ merken und ohne hinsehen ausführen kann. Wird die Selektion über Translation ausgeführt, kann sich der Anwender zwar ebenfalls merken, an welcher Stelle sich eine bestimmte Funktion in der Menühierarchie befindet, kann die Selektion jedoch nur bedingt „blind“ ausführen, weil keine binären Ereignisse, sondern Wegstrecken die Basis für die Cursorbewegung bilden.

Ein Nachteil bei dem Rotationsansatz ist, dass eine getätigte Auswahl nicht rückgängig gemacht werden kann. Insbesondere während der Lernphase ist dies für einen Anwender eine spürbare Einschränkung des Nutzungskomforts. Deshalb muss die Methode um die Möglichkeit eines *undo*’s auf Menüebene erweitert werden. Hierzu wird zwischen Auswahl von Untermenüs und Auswahl von Funktionen unterschieden.

- Um ein Untermenü zu öffnen, muss der entsprechende Sektor ausgewählt werden und sobald die Hand wieder in der Grundstellung ist, wird das ausgewählte Menü aufgeklappt. Da man über den gegenüberliegenden Sektor dieses Untermenüs in das vorherige Menü gelangt, ist das *undo* hierfür bereits gelöst.
- Um eine Funktion auszulösen, muss nach der Auswahl über die Handrotation noch ein zusätzlicher Event ausgelöst werden, z.B. ein Tastendruck auf dem Interaktionsgerät. Durch diese Entkopplung von Auslösen und Ausführen ist es dann auch möglich nach der Auslösung durch eine Rotation in eine andere Richtung einen anderen Sektor anzuwählen. Ausgewählte Sektoren werden automatisch nach einer kurzen Zeitspanne wieder deselektiert. Dadurch wird auch das Schließen des Menüs ohne Funktionsausführung ermöglicht.

Im Gegensatz zum Translationsansatz stimmt beim Rotationsansatz das visuelle Feedback (Cursorbewegung) nicht mit der ausgeführten Bewegung überein. Dies erschwert insbesondere zu Beginn das Erlernen der Menübedienung.

Es hat sich jedoch auch gezeigt, dass der Rotationsansatz nur mit Pie-Menüs mit 4 Einträgen benutzbar ist. Bei Pie-Menüs mit 8 Einträgen sind die Sektoren auf den Diagonalen sehr schwer zu selektieren, da eine gleichzeitige Rotation um die X- und Y-Achse schwierig auszuführen und auf Dauer auch sehr anstrengend ist. Da bis jetzt Rotationen um die Z-Achse nicht beachtet worden sind, könnte

diese zur Auswahl dieser Sektoren herangezogen werden. Durch Rotation wird das Menü um +-45° gedreht, wodurch die Sektoren, die vorher auf der Diagonalen lagen, nun auf den Hauptachsen zum liegen kommen.

#### 4.5.4.5 Menüfunktionalitäten

Es gibt zwei verschiedene Funktionstypen, die durch das Menü ausgelöst werden können:

- Funktion, die keinen inneren Zustand haben, der das Verhalten beeinflußt. Zur Darstellung in dem Menüsystem werden hierzu sog. *Push-Buttons* verwendet.
- Funktionen, die zwischen zwei inneren Zuständen (an/aus) umschalten, je nachdem wie oft oder mit welchem Parameter sie aufgerufen werden. Hier ist es wichtig, dass der Anwender den aktuellen Zustand kennt, damit er weiß, wie die Funktion beim auslösen reagieren wird. Diese Funktionen werden in einem Menüsystem durch *Toggle-Buttons* dargestellt, die in ihrer Erscheinung den Push-Buttons gleichen, jedoch zusätzlich den inneren Zustand visuell anzeigen, z.B. durch einen kleinen Haken am Menüeintrag (Beispiel: Umschalten der Szene zwischen Drahtgitter- und gefüllter Darstellung).

Um das Konzept des *Belegungsmodus* (mehrere Funktionen pro Triggermöglichkeit, wobei immer nur eine aktiv ist) durch das Menü zu unterstützen, bietet es sich an, Toggle-Buttons Gruppen zuzuordnen. Hierbei gilt, dass pro Gruppe immer nur max. 1 Eintrag angeschaltet ist (*Radiobox*). Wird von einem Eintrag zu einem anderen umgeschaltet, so wird zuerst der aktive Eintrag ausgeschaltet (Aufruf der dahinterliegenden Funktion mit Parameter *aus*) und im Anschluß der neue Eintrag eingeschaltet (Aufruf der dahinterliegenden Funktion mit Parameter *an*). Die internen Zustände werden entsprechend angepasst.

Insbesondere im Zusammenhang mit dem *Belegungsmodus* bietet es sich an, eine Default-Funktion zu definieren, d.h. sollte die aktuell angeschaltete Funktion ausgeschaltet werden, so wird im Anschluss der Default angeschaltet.

Wie bereits eingangs erwähnt, ist das einfache Authoring eine sehr wichtige Komponente. Aus diesem Grund wurde im Zusammenhang mit dem Menüsystem eine Beschreibungssprache entwickelt, mit der Menüs und deren Funktionen sehr einfach beschrieben werden können. In der folgenden Abbildung ist der Kern dieser Grammatik als EBNF zusammengefaßt.

```
m2d         : m2d menu_def |
menu_def    : menu_head G_AUF menu_bodys G_ZU
menu_head   : MENU STRING
menu_bodys  : menu_bodys menu_body |
menu_body   : button_head button_body | LINE |
button_head: BUTTON STRING
button_body: G_AUF button_defs G_ZU |
button_defs: button_defs button_def |
button_def : ACTIVE | INACTIVE |
             TOGGLE | CHECKED |
             EVENT STRING |
             GROUP NUMBER |
             GROUPDEFAULT |
```

Abb. 4.36: Einfache EBNF-Grammatik für Menübeschreibung

### 4.5.5 Speedbar

Es hat sich sehr oft gezeigt, dass über einen längeren Zeitraum im Arbeitsprozess oft nur 2-3 Hauptfunktionen verwendet wurden, zwischen denen jedoch sehr oft umgeschaltet wurde. Hier ist eine Aktivierung über das Menü auf Dauer sehr langwierig. In konventionellen Desktopapplikationen können solche Funktionen z.B. über Tastatur-Hotkeys aktiviert werden. Um diesen schnelle Wechseln zwischen oft verwendeten Funktionen optimal zu unterstützen wurde im Rahmen dieser Arbeit die *Speedbar* entwickelt.

Die Speedbar ist ein lineares nicht hierarchisches Menü mit horizontal angeordneten Menüpunkten. Die Liste der Menüpunkte und deren Funktionen, die über die Speedbar aufgerufen werden können, setzt sich aus den zuletzt über das Hauptmenü aufgerufenen Funktionen zusammen, sortiert in der Aufrufreihenfolge, wobei die vorletzte aufgerufene Funktion an erster Stelle in der Liste steht (*LRU, last recently used*). Siehe Abb. 4.37.

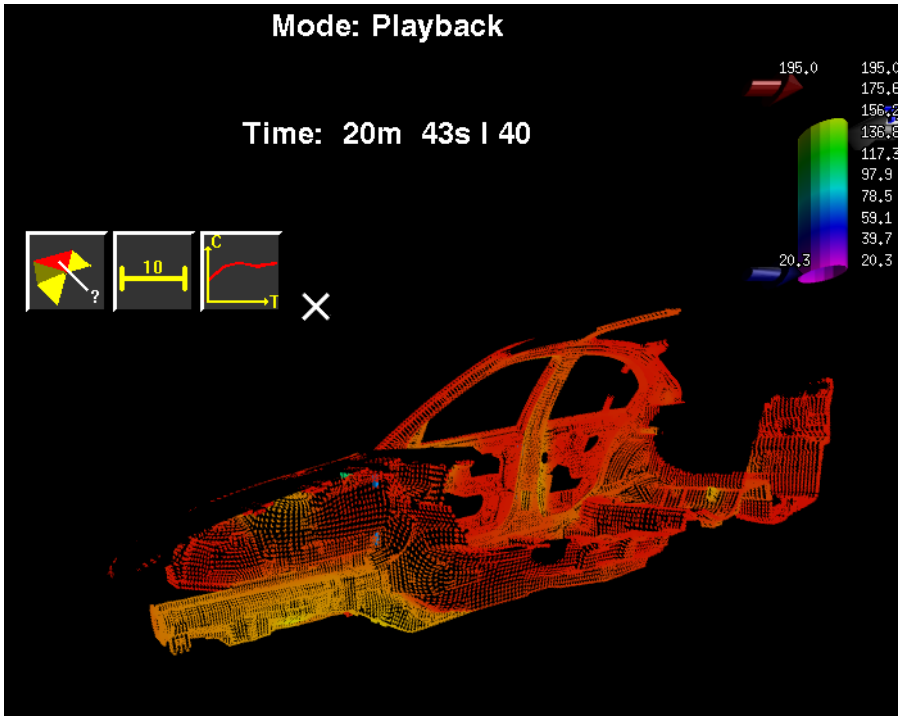


Abb. 4.37: Speedbar: Die letzten Kommandos werden als lineares Menü dargestellt

Der kritische Punkt beim Design der Speedbar ist die Definition des Aktivierungs-Triggers. Die Anzahl an Triggermöglichkeiten in Form von Knöpfen auf dem Eingabegerät ist bekanntlich beschränkt. Wie auch in Kapitel 4.6.3.1 können auf Basis der Proprioception weitere „Trigger“ realisiert werden. Um die Speedbar zu öffnen, muss der Anwender mit seinem Interaktionsgerät von der Leinwand wegzeigen und einen Event auslösen (z.B. einen Knopf drücken). Siehe hierzu auch Abb. 4.38. Da die Intention des Anwenders im Moment des aktiven Zeigens das Öffnen der Speedbar ist, kann die „normale“ Tastenbelegung geändert und eine „Speedbar-konforme“ verwendet werden. Ein versehentliches Öffnen ist dabei weitgehend ausgeschlossen.

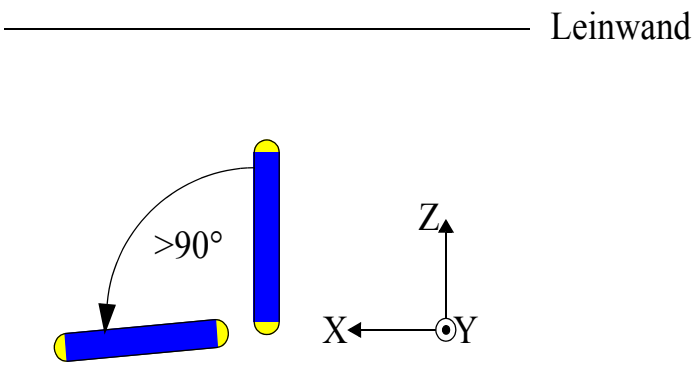


Abb. 4.38: Öffnen der Speedbar durch Drehung um Y-Achse

Sobald die Speedbar angezeigt wird, ist die vorletzte aufgerufene Funktion angewählt, denn es hat

sich gezeigt, dass Anwender sehr oft zwischen zwei Bedienmodi umschalten möchten. So kann er dann die Speedbar im Prinzip blind verwenden: Durch wegdrehen des Gerätes und zwei kurz hintereinander ausgeführten Tastendrücken (Event für Menü öffnen, Event für gewählten Eintrag ausführen) wird auf den vorher aktiven Modus umgeschaltet werden. Aus Konsistenzgründen werden Push-Buttons ebenso behandelt, obwohl hier argumentiert werden kann, dass der Anwender eine bestimmte Funktion mehrmals hintereinander auf einfache Art und Weise ausführen möchte.

Durch Translation wird ein Cursor bewegt, über den die anderen Funktionen ausgewählt werden. Hierbei wird nur die horizontale Translation ausgewertet. Mittels eines weiteren Events wird die gerade angewählte Menüfunktion ausgeführt und die Speedbar geschlossen.

### 4.5.6 Diskussion

Wie in Kapitel 2.5 dargelegt betrachten wir hier Systeme, die dem Anwender eine große Anzahl an Funktionen anbieten können sollen. Das verwendete Menüsystem muss dazu in der Lage sein.

Die *virtuellen Buttons* oder der *Tricorder* sind hierfür wenig geeignet. Auch die *Look-At Menüs* sind für einen täglichen Einsatz wenig geeignet. Dies liegt daran, dass Kopftracking eine zwingende Voraussetzung und aufgrund des fehlenden Augentrackings die Bedienung wenig intuitiv ist.

Es bleiben somit nur *hierarchische-*, *Kugel-* oder *Pie-Menüs* in der engeren Auswahl. Pie-Menüs haben den Vorteil, dass sie eine ideale Ratio zwischen Menütiefe und -breite anbieten und sowohl als 2D, als auch als 3D Variante realisiert werden können. Die Menüpositionierung kann hierbei für den jeweiligen Anwendungszweck entsprechend gewählt werden, wobei die 2D-Variante ausschließlich cart-relative realisiert werden kann. Dies sollte auch die präferierte Positionierung für 3D-Menüs sein, da es dadurch dem Anwender möglich ist, sich frei im Raum zu bewegen und auf Knopfdruck das Menü an einer definierten Stelle relativ zu seiner aktuellen Position erscheinen zu lassen. Wie in Kapitel 4.5.4 beschrieben sind die Pie-Menüs den hierarchischen Menüs in der Bedienung wesentlich überlegen.

Die in Kapitel 4.5.5 vorgestellte Speedbar ist eine zusätzliche Funktionalität mit der zwischen Systemfunktionen sehr schnell umgeschaltet werden kann und zusammen mit einem Menüsystem verwendet werden sollte.

## 4.6 BIT Quantifikation

Über die Quantifikation kann der Benutzer einen Wert zwischen einem gegebenen Minimum und Maximum eingeben.

Im folgenden werden zuerst die Anforderungen an die Quantifikation definiert und anschließend die Ansätze aus der Literatur, sowie die im Rahmen dieser Arbeit entwickelten Techniken, vorgestellt.

### 4.6.1 Anforderungen

Die größte Fragestellung im Zusammenhang mit der Quantifikation ist die Präzision der Werteeingabe. Die Präzision wird hierbei durch die zu lösende Aufgabe vorgegeben. So wird z.B. für das Konstruieren von 3D-Modellen eine sehr genaue Eingabe benötigt, die bis auf mehrere Nachkommastel-

len genau ist. Zum scrollen in einem Text über einen Rollbalken, muss dieser jedoch nur diskrete Werte eines definierten Abstands liefern.

### 4.6.2 Ansätze

In der Literatur finden sich nur sehr wenige Techniken für die reine Quantifizierung. Zumeist wird die Quantifizierung für die Objektskalierung eingesetzt. Die meisten Ansätze können zwar auch für generelle Quantifizierungsaufgaben verwendet werden, verlieren dabei jedoch stark an ihrer intuitiven Benutzung.

#### 4.6.2.1 Beidhändige Skalierung

Aufbauend auf den von [MINE97] entwickelten *hand-held widgets*, kann ähnlich dem beidhändigen Fliegen der Betrag des Vektors, der von beiden Händen aufgespannt wird, als Skalierungsfaktor für ein selektiertes Objekt verwendet werden.

#### 4.6.2.2 Virtuelle Tastatur

Die virtuelle Tastatur erscheint cart-relative vor dem Anwender in Reichweite des Arms. Die virtuelle Tastatur ist ein geometrisches Objekt mit Ziffern, Dezimalpunkt und Löschen-Taste. Durch Selektion der Tasten und auslösen eines Events kann ein beliebiger Wert eingegeben werden, wobei die Selektion z.B. über das Berühren durch der virtuellen Hand geschieht. Denkbar wäre auch das Ray Casting zu verwenden, welches insbesondere bei ausgeschaltetem Kopftracking einfacher zu benutzen sein wird.

### 4.6.3 Eigene Ansätze

Im folgenden werden nun die im Rahmen dieser Arbeit entwickelten Techniken kurz vorgestellt. Hierbei kann ähnlich wie bei Menüs zwischen verschiedenen Positionierungen unterschieden werden (siehe Kapitel 4.5.2.1). Zum einen Techniken, bei denen das Objekt, über das die Quantifizierung ausgeführt wird, *cart-relativ* oder *fixed* positioniert wird. Die *interaktive Säule* ist ein Beispiel für eine solche Technik (s.u.). Zum anderen Techniken, die wie der virtuelle Scrollbar und der Jog-Dial *cursor-relativ* sind und nur nach auslösen eines Triggers sichtbar werden.

#### 4.6.3.1 Interaktive Säule

Die Säule ist ein geometrisches Objekt, welches *cart-relativ* zum Benutzer am Bildrand positioniert wird (siehe Kapitel 4.2.2). Dadurch bleibt sie für den Anwender immer sichtbar, stört aber auch nicht im Gesichtsfeld. Sie besteht aus einem Säulenkörper und einem oder mehreren Zeigern, die den aktuell selektierten Wert angeben.

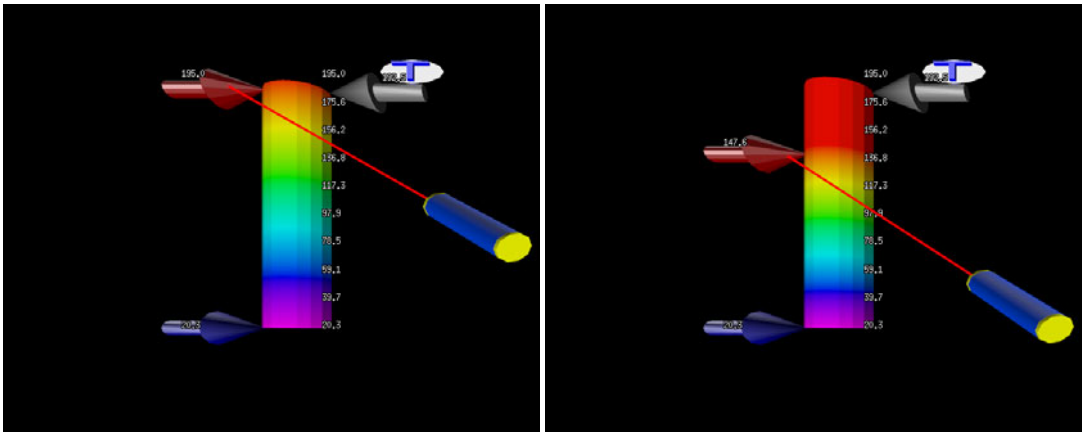

Abb. 4.39: Die interaktive Säule

Eine Möglichkeit die Zeiger zu bewegen wäre sie mit der virtuellen Hand zu greifen und nach oben oder unten zu ziehen. Dafür muss jedoch die Säule immer mit der virtuellen Hand erreichbar sein. Dies ist jedoch nicht immer gegeben, insbesondere nicht vor einer sehr großen Projektionsleinwand.

Aus diesem Grund wird ein virtueller Laserstrahl für die Selektion des Zeigers verwendet[1]. Es wäre jedoch zu umständlich, den Laserstrahl über das Menü oder einen Sprachbefehl anzuschalten und zu störend und verwirrend, wenn dieser virtuelle Laserstrahl immer sichtbar wäre. Deswegen wird der Laserstrahl automatisch dann sichtbar, wenn der Benutzer mit seinem Eingabegerät in Richtung der Säule „zeigt“ (*Proprioception*). Da auch in der realen Welt durch „Zeigen“ etwas selektiert wird, ist diese Vorgehensweise sehr natürlich! Durch „Zeigen“ wird jedoch nur eine grobe Richtung vorgegeben. Es ist somit sinnvoll, die Zielgröße entsprechend groß zu wählen. Deswegen wird hier ausgehend von der virtuellen Hand ein Strahl in Zeigerichtung gegen eine vergrößerte Bounding Box der Säule geschnitten. In der Praxis hat sich eine Skalierung der Bounding Box um das 1,5-fache bewährt.

Aufgrund des visuellen Feedbacks (Sichtbarkeit des Strahls) wird dem Anwender signalisiert, dass er jetzt mit der Säule interagieren kann. Dadurch können die Eingabetasten (Triggermöglichkeiten) z.B. mit Funktionen belegt werden, die für das Interagieren benötigt werden. Hier hat sich bewährt, die Eingabetasten, über die normalerweise die Objektpositionierung ausgelöst wird, für das Positionieren der Säulen-Zeiger zu verwenden. Damit kann dem ewigen Problem der zu geringen Zahl an verfügbaren Triggermöglichkeiten sehr effektiv begegnet werden.

Da die Säule nicht zu viel Platz auf dem Bildschirm einnehmen sollte, ist ihre maximale Größe limitiert. Auf der anderen Seite ist es jedoch sehr schwierig kleine Objekte zu selektieren. Aus diesem Grund wird hier der Selektionsmodus *click and drag to select* für die Selektion der Zeiger verwendet.

Ist ein Zeiger selektiert, so kann dieser über eine translatorische Bewegung verschoben werden. Die Selektion wird durch Auslösen eines weiteren Events wieder aufgehoben und der Zeiger bleibt an der aktuellen Position stehen. Solange die Selektion aktiv ist, muss der Strahl nicht zwingend die vergrö-

1. Die Image-Plane Technik kann hier ebenfalls verwendet werden

ßerte Bounding Box (s.o.) schneiden, da dies die Handhabung zu stark erschweren würde.

#### 4.6.3.2 Interaktive Säule für genaue Werteingaben

Die im vorherigen Kapitel entwickelte interaktive Säule eignet sich nicht unbedingt für das Eingeben von exakten Werten, insbesondere da mit zunehmender Größe des Wertebereichs es für den Anwender immer schwieriger wird den gewünschten Wert zu treffen. Aus diesem Grund wurde eine Säule mit Wertebereich entwickelt, die dieser Problematik entgegenwirkt.

Die Bedienung ist der interaktiven Säule sehr ähnlich, jedoch werden einige neue Elemente hinzugefügt und somit unterscheidet sich auch die geometrische Repräsentation der Säule. Die Säule besteht aus einem Säulenkörper, der den maximalen Wertebereich abdeckt, einen Zeiger, der den aktuellen Wert in diesem Wertebereich angibt und eine kleine Säule, die über dem Säulenkörper steckt. Weiterhin befinden sich links und rechts der Säule zwei virtuelle Knöpfe, mit denen in den Wertebereich hinein und heraus gezoomt werden kann. Siehe hierzu die folgende schematische Abbildung:

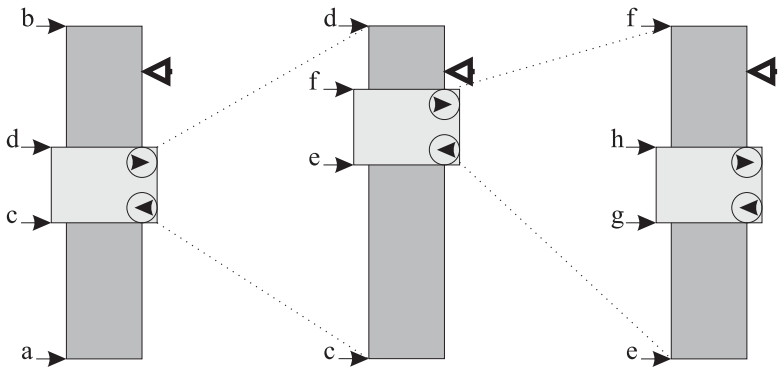


Abb. 4.40: Schematische Abbildung

Die kleine Säule gibt an, welchen Wertebereich der gesamte Säulenkörper einnehmen soll, wenn der Knopf zum hineinzoomen gedrückt wird. Der Anwender kann somit in jeder Zoomstufe den gewünschten Wertebereich einkreisen, bis er den gewünschten Wert auswählen kann. Aus dem Wertebereich herauszoomen bedeutet, dass der vorherige Wertebereich verwendet wird. Der Zeiger, die Zoomknöpfe und die kleine Säule werden wie bereits in Kapitel 4.6.3.1 beschrieben, bedient.

Der Vorteil dieser Methode ist, dass der Anwender die Genauigkeit des Wertes beliebig bestimmen kann.

#### 4.6.3.3 Jog-Dial: Rotatorische Quantifzierung

Die Säule eignet sich bereits für sehr viele Anwendungen, jedoch gibt es für einige spezielle Problemstellung wesentlich geeignetere Alternativen. Eine davon ist der *Jog-Dial*, der sich insbesondere für Anwendungen eignet, deren anzugebender Wert im Bereich [-v;+v] liegt (siehe Abb. 4.41).

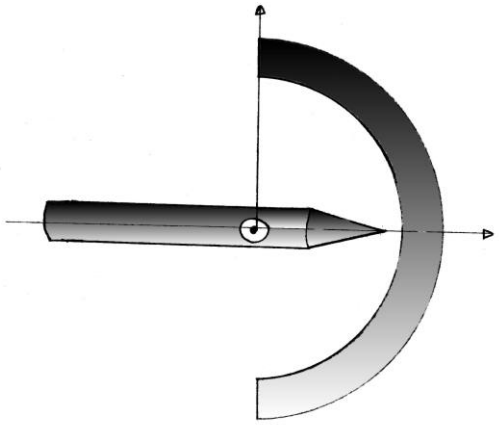

Abb. 4.41: Schematische Darstellung eines Jog-Dials

Der hier für Anwendungen in Virtuellen Umgebungen vorgestellte Jog-Dial ist in der Bedienung an den Jog-Dial, wie man ihn an vielen Videorecordern findet, angelehnt. Der Jog-Dial ist ein Knopf der in einem Bereich von typischerweise +/- 90 Grad gedreht werden kann und über den Betrag des Winkels die Abspielgeschwindigkeit steuert. Das Vorzeichen des Winkels beeinflußt dabei die Abspielrichtung.

Der Jog-Dial besteht aus einem Halbkreis und einem Zeiger (s.o.), die beide an der virtuellen Hand des Anwenders befestigt sind. Der Halbkreis repräsentiert hierbei den Wertebereich und der Zeiger zeigt auf den aktuell selektierten Wert. Der Zeiger rotiert um das Zentrum des Halbkreises und wird durch Drehung des Unterarms gesteuert. Im Gegensatz zur interaktiven Säule wird der Jog-Dial über einen Event aktiviert und solange er aktiv ist wird die Drehung des Unterarms ausgewertet und der Zeiger bewegt. Wird der Zeiger bei jeder Aktivierung in die Nullstellung zurückversetzt, weiß der Anwender ohne hinsehen in welcher Position der Zeiger sich befindet und kann solange nur grobe Werteangaben benötigt werden, „blind“ einen neuen Wert eingeben (Proprioception). Dadurch muss der Anwender auch nicht den Arm heben, um den Jog-Dial in das Blickfeld zu bringen, denn dies kann auf Dauer zu Ermüdungserscheinungen des Unterarms führen.

**Zoneneinteilung**

Weiterhin besteht die Möglichkeit den Wertebereich [-v;+v] nicht linear auf den Halbkreis des Jog-Dials zu mappen, sondern diesen in Zonen einzuteilen, die symmetrisch zur Nullstellung sind. Siehe hierzu Abb. 4.42.

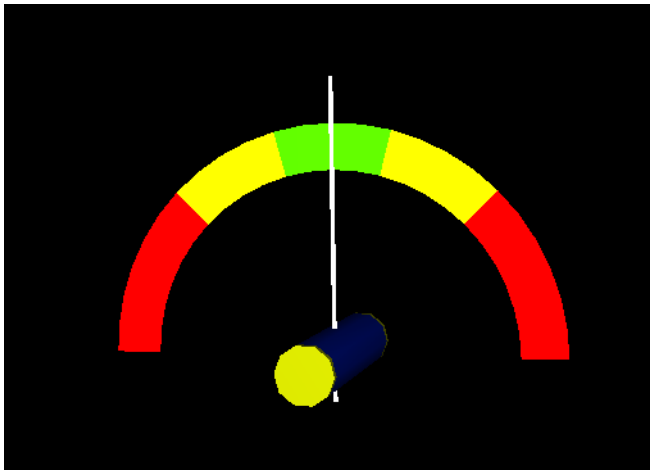

Abb. 4.42: Der Jog-Dial mit Zoneneinteilung

Jeder Zone kann nun entweder ein fester Wert oder ein Wertebereich zugeordnet werden. Zur besseren Verständlichkeit sollten diese Werte jedoch wie bei einem linearen Mapping von links nach rechts stetig zunehmen.

Auch über den Jog-Dial können exakte Werte eingeben werden. Hierzu muss nur der Jog-Dial in 12 Zonen eingeteilt werden, wobei 10 Zonen die Ziffern 0 bis 9 zugeordnet sind, Zone 11 der Dezimalpunkt und Zone 12 ein „Löschen der letzten Ziffer“. Über anfahren der einzelnen Zonen und auslösen eines Events (z.B. Knopfdruck auf dem Eingabegerät) können somit beliebige Ziffernfolgen eingegeben werden, die beim Beenden der Eingabe über den Jog-Dial als Wert zurückgegeben wird.

#### 4.6.3.4 Virtueller Scroll-Bar: Translatorische Quantifzierung

Der virtuelle Scroll-Bar ist dem aus 2D-GUI’s bekannten Rollbalken entlehnt und wird über die Translation des Cursors gesteuert. Entsprechend wird der virtuelle Scroll-Bar bei Auslösen eines Events *cursor-relativ* positioniert und angezeigt. Der Selektionsbalken wird hierbei auf einen vorher definierten Wert gesetzt. Dann kann über die Translation des Cursors der Selektionsbalken nach links und rechts bewegt werden, wobei der aktuelle Zahlenwert über diesem ausgegeben wird (siehe Abb. 4.43). Nach Auslösen eines weiteren Events ist der aktuelle Wert gewählt und der Scroll-Bar wird wieder unsichtbar geschaltet.

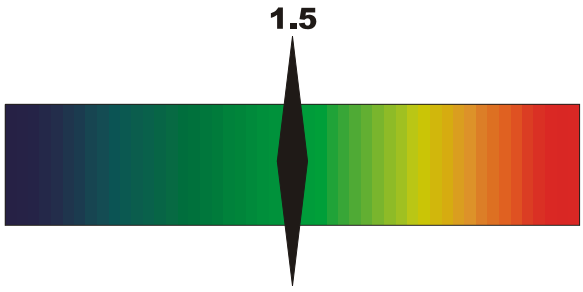


Abb. 4.43: Virtueller Scroll-Bar

Die in Kapitel 4.6.3.3 beschriebene Zoneneinteilung kann beim Scroll-Bar ebenfalls verwendet werden. Auch können hier die Vorteile der Proprioception ebenso ausgenutzt werden.

### 4.6.4 Diskussion

Die von Mine vorgestellte Technik „interaktive Skalierung“ (s.o.) eignet sich aufgrund der Interaktion vor allem für das Skalieren von Objekten der Szene. Sie kann jedoch auch für generelle Quantifizierung eingesetzt werden, jedoch müssen dann Wege gefunden werden, wie der Wert der Quantifizierung angezeigt und auch nach der der Interaktion sichtbar ist. Im Falle der Objektskalierung ist dies natürlich sehr einfach, da der Wert direkt aus der Objektgröße abgeleitet werden kann.

Für generelle Quantifizierungsaufgaben bietet sich die interaktive Säule an. Die Werte sind während und nach der Interaktion sichtbar und es sind auch multidimensionale Qunatifizierungsaufgaben umsetzbar. Dies geschieht entweder indem mehrere Zeiger pro Säule (bei gleichem Wertebereich) oder mehrere Säulen eingesetzt werden. Weiterhin kann sie auch zur Eingabe exakter Werte eingesetzt werden. In Praxistests hat sich die Säule sehr bewährt.

Der Jog-Dial ist ähnlich wie die interaktive Skalierung eine Interaktionstechnik, die sich eher für spezielle Anwendungen der Quantifizierung eignet, z.B. Animationssteuerung. Ein großer Vorteil der Technik ist, dass die Anwendung weitesgehend ermüdungsfrei für den Anwender ist. Er muss nur um sein Handgelenk rotieren, aufwändige Translationen der Hand und des Arms sind nicht notwendig.

Im Gegensatz dazu muss der Anwender bei dem virtuellen Scroll-Bar seinen Arm sehr viel bewegen. In informellen Tests konnte hinsichtlich der Geschwindigkeit und Genauigkeit der Werteangabe kein Unterschied festgestellt werden. Es hat sich jedoch gezeigt, dass die Translationen beim Scroll-Bar auf Dauer anstrengender als die Bedienung des Jog-Dials war.

## 4.7 BIT Texteingabe

Diese BIT erlaubt es dem Anwender einen Text einzugeben, wobei die Applikation diesem Text keine spezielle Bedeutung zuordnet.

Im folgenden werden nun zuerst die Anforderung an diesen BIT definiert und im Anschluss daran die Umsetzung dargelegt.

### 4.7.1 Anforderungen

Die Texteingabe wird bei der Evaluierung von Planungszuständen insbesondere für die Protokollfunktionalität benötigt. Das bedeutet, dass zusammenhängende Texte eingegeben werden müssen. Spracheingabe hat sich hier als zu fehleranfällig erwiesen, denn um eine hohe Akzeptanz beim Anwender zu erreichen, muss das Systems nahezu alle gesprochenen Worte perfekt und fehlerlos erkennen können. Ein weiterer Nachteil von Spracheingabe ist, dass der Anwender ein zusätzliches Gerät tragen muss.

### 4.7.2 Ansätze aus der Literatur

Durch die strengen Anforderungen an den Text-BIT bzgl. Leistungsfähigkeit und zur Verfügung stehender Eingabegeräte, gibt es eigentlich nur eine Lösung, die im folgenden Kapitel beschrieben wird.

### 4.7.3 Realisierter Ansatz

Für die Eingabe langer Texte ist es am sinnvollsten einen separaten PC o.ä. zu verwenden, auf dem über die Tastatur der Text eingegeben werden kann. Die Texteingabe sollte jedoch mit dem VR-System verbunden sein, um den eingegebenen Text sofort weiterverarbeiten zu können oder ihn in der Szene darzustellen. Die Darstellung in der Szene ist jedoch nur bei sehr kurzen Texten sinnvoll, da ansonsten zu viel von der Szene verdeckt werden würde.

Eine sicherlich sehr interessante Möglichkeit insbesondere für Virtual Tables oder Stift und Tablett Interface stellt die Verwendung eines Schrifterkennungssystem dar, welches die direkt auf den Table, bzw. das Tablett geschriebenen Worte erkennt und zu einem Text zusammensetzt.

## 4.8 Zusammenfassung

In diesem Kapitel wurden aus der Literatur bekannte, aber auch neu entwickelte Interaktionstechniken nach dem Konzept der logischen Interaktionsaufgaben (*basic interaction task*, BIT) klassifiziert und gemäß den in Kapitel 2.5 formulierten Anforderungen evaluiert. Interaktionstechniken sind als Umsetzung oder Realisierung der logischen Interaktionsaufgaben (basic interaction task) zu verstehen. Da für diese Evaluierung nach dem Abhängigkeitsmodell eine Festlegung der Gerätetechnologie notwendig ist, wurde diese auf generischer Ebene zu Beginn durchgeführt.

Im Rahmen der Evaluierung zeigte sich, dass für einige BITs keine Interaktionstechniken existierten, die auf Basis der definierten Voraussetzungen die an sie gestellten Anforderungen erfüllten. So wurden im Rahmen dieser Arbeit neue Interaktionstechniken entwickelt: Der world point grab zum Positionieren und für die Quantifizierung die interaktive Säule, der virtuelle Scrollbar und der Jog-Dial.

Ein weiterer offener Punkt, der in der Literatur selten angegangen worden ist, ist das Menüsystem. So zeichnen sich auch heute noch sehr viele VR-Anwendungen dadurch aus, dass sie nur über wenige Funktionen verfügen, die der Anwender direkt verwenden kann. Hier wurde deswegen ein leistungsfähiges Menüsystem entwickelt, welches sehr leicht zu konfigurieren ist und in der Bedienung Alternativkonzepten weit überlegen ist. Die graphische Repräsentation dieses Menüsystems basiert auf den sog. Pie-Menüs. Über eine auf der Proprioception basierende weitere Interaktionstechnik (Speedbar) konnte die Verwendung des Menüsystems noch weiter verbessert werden.

Neben dem Herausfiltern der für die Anwendungsklasse geeigneten Interaktionstechniken, konnten auch Regeln und Muster herausgearbeitet werden, die sich bei Anwendung positiv auf die einfache und intuitive Bedienbarkeit auswirken. Dies sind u.a.:

- *Proprioception:* Das Wissen des Menschen über Körperrelationen kann in den meisten Fällen für die Interaktion sinnvoll eingesetzt werden. Hierbei muss jedoch darauf geachtet werden, dass Position und Orientierung alle beteiligten Körperteile dem System direkt oder indirekt bekannt sind (z.B. kann das Auslösen einer Aktion durch Berühren der Hüfte nicht über ausschließliches Tracking von Kopf und Hand sinnvoll realisiert werden).
- *Minimale Bewegung:* Wird ein System über einen langen Zeitraum eingesetzt, so sollten die Bewegungen, die für das Interagieren notwendig sind, minimiert werden, um Ermüdungserscheinungen vorzubeugen. Indirekte Interaktion oder Rotationen sind hierbei hilfreiche Ansätze.

- *Continious Feedback:* Es ist immanent wichtig, dass eine Interaktionstechnik jederzeit Änderungen im Systemstatus visuell oder akustisch signalisiert. Niedrige Frameraten erhöhen zusätzlich die Notwendigkeit von Feedback.
- *Rotation:* Es hat sich gezeigt, dass auch die Rotation für das Interagieren sinnvoll eingesetzt werden kann (z.B. Jog-Dial) und bei der Entwicklung von Interaktionstechniken auch berücksichtigt werden sollte.
- *Mappings:* Nicht-lineare Bewegungsmappings sind für den Anwender unnatürlich und schwer anwendbar
- *Schwellwerte:* Tracking und auch ungewollte, kleine Bewegungen seitens des Anwenders führen dazu, dass die Eingabewerte immer leicht verrauscht sind. Der Einsatz von Schwellwerten bietet sich hier an.
- *Genauigkeit der Interaktion:* Bei der Interaktion in 3D ist immer zu bedenken, dass ein Anwender seine Hand (als Wertegeber für die virtuelle Hand) nur sehr schwer exakt im Raum positionieren und orientieren oder über einen längeren Zeitraum still halten kann. „Magnetfelder“, wie z.B. in Kapitel 6.1.1 beschrieben können hier helfen.
- *Direkte versus indirekte Interaktion:* Mit natürlicher und somit intuitiver Interaktion verbindet man eigentlich die direkte Interaktion. Jedoch ist das indirekte Interagieren nicht unintuitiver, sondern bietet Möglichkeiten, die das Arbeiten noch vereinfachen (z.B. Minimierung der Bewegungen aufgrund eines vertikalen Offsets zwischen virtueller und realer Hand, Umgehen der Verdeckungsproblematik)

# 5 Unterstützung der Kommunikation und Kooperation

In diesem Kapitel wird nun näher auf die Unterstützung der Kooperation und Kommunikation eingegangen. Siehe hierzu vor allem das Kapitel 3.3.3, in dem die Studien von [PERR98] vorgestellt wurden. In diesen Studien wurden diese Aspekte als Grundpfeiler für einen effektiven Design Review herausgestellt. Hierbei hat sich auch gezeigt, welchen Stellenwert Mediatoren in der zwischenmenschlichen Kommunikation einnehmen. In den erwähnten Studien wurden hierzu Plots eingesetzt, die von den Teilnehmern eines Design Reviews mit verschiedenfarbigen Stiften annotiert wurden. So konnten Ideen skizziert werden, um den anderen Personen einen Sachverhalt besser zu erläutern. Weiterhin konnten Designänderungen und Anmerkungen aufgebracht und zum Beispiel im Anschluß von den verantwortlichen CAD-Designern in das virtuelle Modell eingearbeitet werden.

Weitere wichtige Aspekte ([LECA00]) sind die Gleichberechtigung der Teilnehmer und die Möglichkeit Ideen auch „privat“ entwickeln zu können (*private spaces*, [BLAN98]). Die Gleichberechtigung setzt voraus, dass jede Person nahezu ähnliche Möglichkeiten[1] hat, mit dem System zu arbeiten. Bestimmte Funktionalitäten, bei denen es sinnvoll ist, dass sie nur eine Person (gleichzeitig) ausführen kann (z.B. Bewegen des virtuellen Prototypens), sollten jedoch auf einfache Art und Weise abgegeben werden können. Die „private“ Entwicklung von Ideen bedeutet, dass prinzipiell jeder Anwender seine Sicht auf ein virtuelles Modell haben kann.

So ergeben sich für die Unterstützung der Kommunikation und Kooperation die folgenden Schwerpunkte, auf die in den nachfolgenden Kapiteln näher eingegangen wird:

- Gemeinsames und gleichberechtigtes Arbeiten
- Möglichkeit des privaten Arbeitens, um eigene Ideen entwickeln zu können
- Unterstützung eigener 3D-Ansicht, um effektiv an privaten Skizzen arbeiten zu können
- Skizzieren auf den virtuellen Prototypen

In Kapitel 4.1.3 wurden die generischen Ein- und Ausgabegeräte beschrieben, die als Grundlage für die Entwicklung der hier beschriebenen Benutzerschnittstelle dienen. Im Rahmen der Umsetzung der o.g. Schwerpunkte hat sich ergeben, dass auf Eingabeseite auch die Stereobrillen als Eingabegerät betrachtet werden können. Siehe hierzu Kapitel 5.1.2.2. Als Ausgabegerät wurden nicht kopfgebundene Projektionen festgelegt. Bereits an anderen Stellen wurde auf die Problematik des exakten Positionierens der Hand im 3D-Raum eingegangen. Aus diesem Grund ist für das *Skizzieren* eine feste Unterlage entscheidend. Deswegen wird im folgenden davon ausgegangen, dass das Ausgabegerät eine möglichst horizontal angeordnete Projektionsscheibe offeriert, auf die die Anwender direkt mit einem Stift skizzieren kann. Der vorgestellte Ansatz bietet auch die Möglichkeit mit vertikal angeordneten Projektionsscheiben zu arbeiten, verliert aber dadurch etwas an seiner Natürlichkeit.

In den folgenden Kapiteln wird zuerst auf das Design der eigentlichen Benutzerschnittstelle eingegangen und im Anschluß eine Methodik vorgestellt, um *private spaces* auch softwareseitig zu unter-

1. Optimal wäre sicherlich, wenn jedem Anwender alle Möglichkeiten des Systems zur Verfügung stehen. Eine solche Anforderung würde jedoch eine enorme Anzahl von zu lösenden Problemstellungen nach sich ziehen, die in keiner Relation zum eigentlichen Nutzen stehen würde. Es ist hierbei auch zu bedenken, dass es auch in der Praxis sehr selten vorkommt, dass immer alle Funktionen allen Teilnehmern zur Verfügung stehen.

stützen. In Kapitel 5.3 wird das Skizzieren näher beschrieben.

## 5.1 Benutzerschnittstelle: Gemeinsam und doch Unabhängig

In diesem Kapitel wird insbesondere auf die Benutzerschnittstelle eingegangen, um den in Kapitel 3.3.3.3 dargelegten Anforderungen gerecht zu werden. Um eine möglichst gute Kommunikation zwischen den beteiligten Personen erreichen zu können, müssen Möglichkeiten geschaffen werden, um sowohl miteinander zu kooperieren, als auch in einem *private space* eigene Ideen zu entwickeln.

Im folgenden werden nun einige Ansätze aus der Literatur dargestellt. Im Anschluß wird das *Papier und Bleistift Paradigma* vorgestellt, welches im Rahmen dieser Arbeit entwickelt wurde. Es bildet die Grundlage für die weitere Umsetzung der Benutzerschnittstelle. Dann werden einige Basisfunktionalitäten vorgestellt, die die Benutzerschnittstelle bieten muss.

### 5.1.1 Ansätze aus der Literatur

Der Bereich des kooperativen Arbeitens ist bereits seit längerem ein beliebtes Forschungsgebiet, wobei hier jedoch in den meisten Fällen Kooperation im Sinne eines verteilten Arbeitens an verschiedenen Standorten verstanden wird. Die Unterstützung des gemeinsamen Arbeitens an einem Ort wurde jedoch bis jetzt nur sehr wenig erforscht.

Erwähnenswert sind hier die Arbeiten der TU Wien ([SCHM96]). Hier wird seit einigen Jahren eine Mehrbenutzer-Augmented Reality-Architektur für Anwendungen aus dem Bereich Visualisierung, Präsentation und Ausbildung, die sog. „Studierstube“, entwickelt. Mit Hilfe dieses Systems kann einer Gruppe von Personen 3D-Graphik stereoskopisch präsentiert werden, wobei jeder Teilnehmer ein *see-through HMD* trägt (siehe Abb. 5.1).

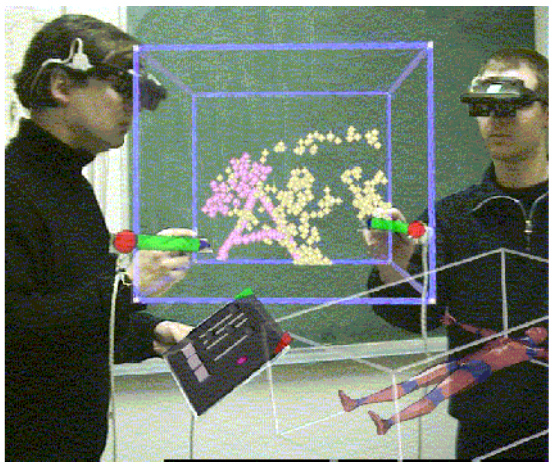

Abb. 5.1: Studierstube (aus [FUHR99])

Da jeder Anwender sein eigenes Ausgabegerät besitzt, sind individuelle Sichten auf ein und das selbe virtuelle Modell sehr leicht zu realisieren. Sind diese mit einem Trackingsystem ausgestattet, können sie auch durch den Anwender intuitiv verändert werden. Wie bereits in Kapitel 3.1.1.3 dargelegt, wirkt sich das Tragen eines HMDs negativ auf die Kommunikation und Interaktion zwischen den An-

wendern aus. Dadurch ist der Ansatz der Studierstube für die hier fokussierte Anwendungsklasse nicht gangbar. Ein weiterer Fokus der Arbeiten rund um die Studierstube liegt in der Entwicklung einer geeigneten Client-Server Architektur und speziellen Erweiterungen, um hochqualitative Augmented Reality Anwendungen zu betreiben.

In [FUHR99] wird ein ebenfalls auf der Studierstube basierendes System vorgestellt, welches neben der gleichzeitigen Unterstützung mehrerer Anwender auch den Multitaskingbetrieb multipler Augmented Reality Anwendungen ermöglicht. Mit dieser Umgebung sollten die Entwicklung neuer Anwendungen auf Basis von 6DOF-Interaktionstechniken, sowie 2D-Techniken mit dem PIP (*personal interaction panel*, [SZAL96]) wesentlich erleichtert werden.

Das System Flatland ([MYNA99]) basiert auf herkömmlichen Whiteboards und erweitert diese um einige spezielle Fähigkeiten, die für die Teamarbeit und Kommunikation wichtig sind. Bei dem Entwurf des Systems stand eine nahtlose Integration in die Büroumgebung im Vordergrund. Weiterhin wurden die folgenden drei Themenschwerpunkte adressiert:

- Techniken um den Platz auf dem Board zu optimieren
- Die Möglichkeit „Verhalten“ flexibel zu definieren, um die Semantik unterschiedlicher Anwendungen möglichst einfach abbilden zu können.
- Mechanismen, um die Historie der erzeugten Skizzen zu verwalten

Eingaben seitens des Anwenders werden automatisch in sog. Segmente gruppiert, die frei auf dem Board verschoben werden können. Steht nicht mehr ausreichend Platz auf dem Board zur Verfügung werden die Segmente, die am längsten nicht mehr verwendet wurden, stark verkleinert. Die Segmente können als ToDo-Liste, Skizzen, Rechenaufgaben etc. verstanden werden und je nach Typus wird ein unterschiedliches Verhalten auf die Segmente angewendet. So wird z.B. bei Rechenaufgaben über eine Handschrifterkennung diese gelöst und vom System das Ergebnis automatisch ausgegeben. Die Speicherfunktionalität wurde durch die Realisierung einer Historie und einem Zeitrollbalken (*time slider*) umgesetzt.

Wie bereits eingangs erwähnt, sind die Arbeiten im Bereich der Kooperation ohne Verteilungsaspekt sehr rar gesät und die hier vorgestellten bieten keine richtige Lösung für die im Rahmen dieser Arbeit vorgestellten Problemstellung. Sie zeigen jedoch einige interessanter Konzepte auf (z.B. die Integration einer Historie und eines Zeitrollbalken), die bei der Entwicklung eines neuartigen Konzeptes eingearbeitet werden sollten.

### 5.1.2 Papier und Bleistift Paradigma: Konzept einer Bedienoberfläche

Im folgenden wird das im Rahmen dieser Arbeit entwickelte Basiskonzept der Bedienoberfläche, das *Papier und Bleistift Paradigma* (veröffentlicht in [EHNE01]), vorgestellt und im Anschluß näher auf die einzelnen Elemente und Funktionalitäten eingegangen.

Das *Papier und Bleistift Paradigma* wurde zum Zweck des gemeinsamen, aber auch privaten Arbeitens an einem virtuellen Prototypen entwickelt und beschreibt die digitale Umsetzung von Papier und Bleistift in einer virtuellen Welt. Das „Papier“ wird hierbei über die zwei folgenden Elemente beschrieben:

- Sketch-Objekte
- Prototype-Objekte

Diese werden in Kapitel 5.1.2.1 eingehender beschrieben. Der „Bleistift“ wird zum einen über das physikalische Eingabegerät und zum anderen über die damit verbundenen Interaktionen definiert (siehe Kapitel 5.1.2.2).

Bei der Entwicklung des Basiskonzeptes wurde insbesondere der Schwerpunkt des privaten Arbeitens beachtet, denn es ist prinzipiell möglich mehrere Instanzen „Papier“ und „Bleistift“ zu erzeugen und somit den gleichzeitigen Zugriff auf diese Ressourcen für jeden Anwender zu realisieren. Die unterstützung des private Arbeiten wird durch die Integration des in Kapitel 5.2.2 beschriebenen MultiViews-Algorithmus noch verbessert. Hiermit werden erstmals auch personalisierte 3D-Ansichten für mehrere Benutzer ohne Einsatz spezieller Hardware ermöglicht.

#### 5.1.2.1 Das Papier: Sketch-Objekte und Prototype-Objekte

Die Grundidee zur Umsetzung beruht auf der Tatsache, dass beim klassischen Arbeiten mit Papier und Bleistift grundsätzlich eine beliebige Anzahl an Papierblättern zur Verfügung steht. So können eigene Ideen auf diesen Blättern skizziert werden. Sieht der Ersteller diese als gelungen an, so kann er sie den anderen Teilnehmern präsentieren oder ansonsten verwerfen. Nachteile hierbei sind:

- Der virtuelle Prototyp steht nur auf dem Hauptplot zur Verfügung und nicht auf dem separaten Blatt
- Aufwand des Übertragens der Idee auf den Hauptplot.

Zwar können Ideen auch sofort auf dem Hauptplot entwickelt werden, verlieren dadurch aber ihren privaten Charakter. Im Rahmen dieser Arbeit wurden aus diesem Grund die *Sketch-Objekte* entwikkelt, die in der Analogie einem Stück Papier entsprechen auf dem der Benutzer zeichnen kann. Neben den Sketch-Objekten existiert noch ein spezielles Sketch-Objekt, das *Haupt-Sketch-Objekt*. Dieses wird als Diskussionsgrundlage für die gesamte Gruppe verwendet und nimmt auf der Projektionsfläche den größten Raum ein. Siehe hierzu auch die schematische Darstellung in Abb. 5.2.

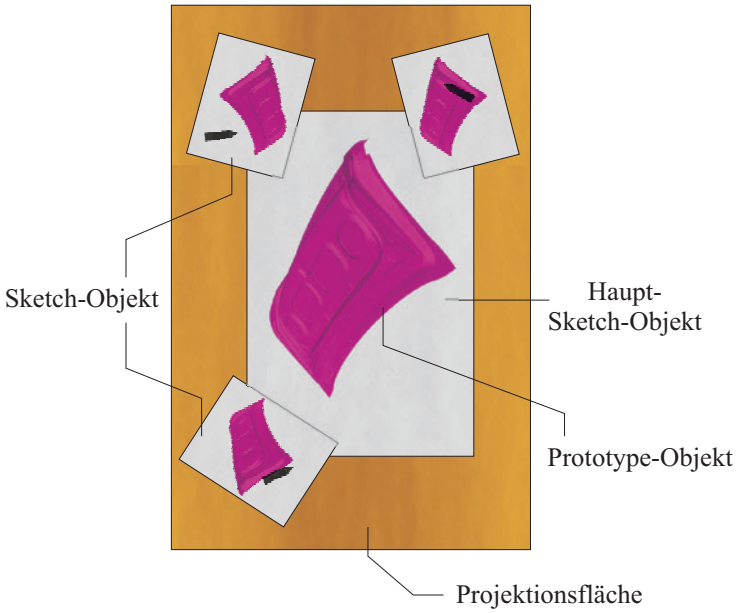


Abb. 5.2: Papier und Bleistift Paradigma

Das System unterstützt eine beliebige Anzahl von Sketch-Objekten. Weiterhin kann jedes Sketch-Objekt von jedem Anwender bearbeitet werden, wobei jeder Anwender seinen eigenen Stift zur Verfügung hat.

Natürlich wäre es keine wirklich große Verbesserung zu konventionellen Design Reviews, wenn die Sketch-Objekte nur leeres virtuelles Papier wären. Aus diesem Grund sind sie nicht leer, sondern können auch CAD-Modelle enthalten. So steht auch für private Skizzen der virtuelle Prototyp zur Verfügung und man kann auf ihm skizzieren. Diese Prototypen sind jedoch nicht wie bei Plots auf reine 2D-Modelle beschränkt, sondern aufgrund des verwendeten Software- und Hardwaresystems können auch 3D-Modelle (stereoskopisch) dargestellt werden[1]. Um die Sketch-Objekte aus der richtigen Perspektive darstellen zu können, ist jedem eine *Kamera* zugeordnet.

Aufgrund der Unterstützung von 3D-Objekten sind somit Sketch-Objekte nicht als plane Fläche zu verstehen, sondern eher mit einem Fenster in eine eigene 3D-Welt vergleichen, in der sich der virtuelle Prototyp, das *Prototype-Objekt*, befindet. Das Sketch-Objekt liegt hierbei in der Projektionsebene und kann auch nur in dieser bewegt werden. Das Lage und Größe dieses in der Projektionsebene liegenden Fenster in die virtuelle Welt wird im folgenden als *Paper-Objekt* bezeichnet. Alle Sketch-Objekte werden in einer Liste verwaltet. In Abb. 5.3 sind die Komponenten eines Sketch-Objektes und die Verwaltungsliste noch einmal schematisch dargestellt.

1. Dies ist noch eher vergleichbar mit 3D-PMU's, die aus Holz, Ton, o.ä. gefertigt werden. Dies sind jedoch immer nur Einzelstücke und in einem DR kann somit auch nur auf eine einzige „Instanz" zurückgegriffen werden

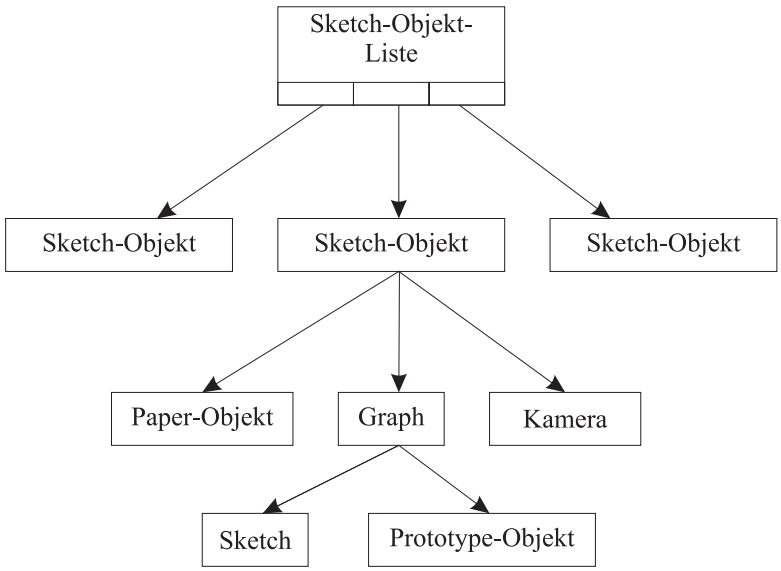


Abb. 5.3: Schematischer Aufbau eines Sketch-Objektes und der Sketch-Objekt-Liste

Die Sketch-Objekt-Liste wird beim Rendering von vorne nach hinten traversiert, wodurch eine feste Reihenfolge gegeben ist und das „stapeln“ von Sketch-Objekten ermöglicht wird. Dies wird in Kapitel 5.2.2 näher beschrieben. Eine Erweiterung wird in Kapitel 6.2 aufgezeigt.

#### 5.1.2.2 Der Stift: Interaktion auf der Projektionsfläche

In den Anforderung wurde definiert, dass das System mehrere Stunden von den selben Anwendern verwendet werden können soll, ohne dass hierbei übermäßige Ermüdungserscheinungen aufgrund der verwendeten Hardware und Softwareschnittstelle auftreten. Die Hardwareseite, als auch die vorgestellten Interaktionstechniken wurden bereits dahingehend optimiert. Die Verwendung der Tischprojektion bietet jedoch eine weitere Möglichkeit das Arbeiten mit dem System zu erleichtern, denn sie bietet den Anwendern die Möglichkeit die Arme auf der Projektionsfläche abzustützen. Bei der Umsetzung der Interaktionen muss somit darauf geachtet werden, dass der Anwender möglichst oft den Tisch als Ablage für die Arme verwenden und direkt mit den Interaktionsgeräten auf ihm arbeiten kann (z.B. skizzieren). Zusätzlich gibt ihm das haptische Feedback eine wesentlich bessere Kontrolle über seine Interaktion (siehe Kapitel 4.4.2.3). In dem Aufsatz von Rekimoto [REKI02] wird eine für diese Anwendung sehr interessante Technologie beschrieben, die sog. Smartskins. Dies sind spezielle Berührungssensoren, die eine gute Auflösung besitzen und neben den Positionen, an denen der Anwender die Oberfläche berührt auch die Entfernung von der Tischoberfläche berechnen können.

Aus diesem Grund werden nun zwei verschiedene Aktionsmodi eingeführt:

- **2D-Modus:** Das Interaktionsgerät befindet sich sehr dicht oder direkt auf der Projektionsfläche ( $Distanz < \varepsilon$ )
- **3D-Modus:** Interaktionsgerät befindet sich oberhalb der Projektionsfläche ( $Distanz \geq \varepsilon$ )

So können in Abhängigkeit des Modus unterschiedliche Funktionalitäten zur Verfügung stehen, die mit den Interaktionsgeräten direkt ausgeführt werden können. Im folgenden wird eine beispielhafte

Zuordnung und Umsetzung aufgezeigt, die sich in der Praxis als sehr effektiv erwiesen hat.

**Der Stift**

Im 2D-Modus wird der Stift entweder zum Skizzieren oder zum Löschen von Teilen der Skizze verwendet. Hier wird die Analogie zu Bleistiften mit Radiergummi am anderen Ende aufgegriffen. Je nachdem welche Seite des Stiftes die Tischoberfläche berührt, wird die entsprechende Aktion ausgeführt. Im 2D.Modus können auch Menüeinträge aus dem Toolglass-Menü gewählt werden (s.u.).

**Der Puck**

Im 2D-Modus kann über den Puck ein Toolglass-Menü gesteuert werden. Dieses ist direkt an die Position und Orientierung des Pucks gekoppelt, wodurch es sich immer in der selben relativen Lage zum Puck befindet. Somit kann es sehr einfach auch einen Referenzrahmen für die dominante Hand bilden, um z.B. verschiedene Funktionalitäten auswählen zu können.

Weiterhin kann über den Puck das Sketch-Objekt gegriffen werden, indem er auf dessen Rand gelegt und der Puck-Knopf gedrückt wird. Solange der Knopf gehalten wird, kann das Sketch-Objekt verschoben und gedreht werden.

Ist der Puck im 3D-Modus, so kann hierüber das Prototype-Objekt positioniert und orientiert werden. Dies kann über den *world point grab* realisiert werden.

**Die Stereobrille**

Über die Stereobrille kann die Zuordnung zwischen Sketch-Objekt und den Geräten hergestellt werden. Hierzu nimmt der Anwender seine Brille in die Hand und tippt kurz mit ihr auf das zu selektierende Sketch-Objekt, indem er mit der Brille die Projektionsfläche an der entsprechenden Stelle berührt. Dann wird die 3D-Ansicht des nun selektierten Sketch-Objektes auf diese Brille eingestellt. Zusätzlich könnte dadurch auch eine Bindung der Eingabegeräten an dieses Sketch-Objekt erreicht werden.

### 5.1.3 Basisfunktionalitäten

Nachdem das Grundkonzept für das kooperative Arbeiten erläutert wurde, wird in diesem Kapitel näher auf das minimale Set an zur Verfügung stehenden Funktionen eingegangen, die für das intuitive Arbeiten benötigt werden. Gleichzeitig wird beschrieben, wie im Rahmen dieser Arbeit die Bedienung dieser Funktionalitäten gelöst wurde.

- **Erzeugen eines neuen Sketch-Objektes:** Hierbei wird ein neues Sketch-Objekt mit zugehörigem Prototypen-Objekt erzeugt. Diese Funktionalität wird über das Toolglass-Menü erreicht und erstellt ein neues Sketch-Objekt auf Basis des Sketch-Objektes über dem der Puck sich befindet.
- **Bewegen eines Sketch-Objektes:** Hierzu wird der Puck auf die Umrandung des Sketch-Objektes (genauer: das Paper-Objekt) geschoben, und ein Knopf auf dem Puck gedrückt. Solange dieser nun gedrückt ist, bewegt sich das Sketch-Objekt mit dem Puck mit (*Drag-Mode*).
- **Skalieren und Ikonifizeren eines Sketch-Objektes:** Die Größe eines Sketch-Objektes wird durch das Paper-Objekt vorgegeben und bei der Erzeugung mit einer Default-Größe initialisiert. Um nun das Paper-Objekt zu vergrößern oder zu verkleinern müssen beide Interaktionsgeräte

eingesetzt werden. Der Puck wird auf die Umrandung des Sketch-Objektes gesetzt und der Puck-Knopf gedrückt (*Drag-Modus*, s.o.). Gleichzeitig wird die Stiftspitze auf der Umrandung positioniert und mittels Drücken des Stiftknopfes gegriffen. Durch Veränderung der relativen Position der beiden Greifpunkte (Puck, Stift) zueinander, wird die Größe des Paper-Objektes entsprechend angepasst. Durch das Greifen der Eckpunkte ist eine Skalierung sowohl in x-, als auch in y-Richtung möglich. Beim Greifen gegenüberliegender Seiten kann jedoch nur eine Richtung manipuliert werden.

Diese Skalierung hat nun in erster Linie nur Auswirkungen auf das Paper-Objekt, die Größe des Prototype-Objektes bleibt unverändert. Soll auch diese verändert werden, so kann entweder die Entfernung Objekt zu Kamera verändert werden oder der Darstellungsmaßstab angepasst werden (s.u.).

- **Zusammenführen von Sketch-Objekten:** Eine sehr wichtige Funktion, um eine private Skizze in das Haupt-Sketch-Objekt zu integrieren. Hierzu wird der Puck über dem Quell-Sketch-Objekt positioniert und der Puck-Knopf gedrückt und gehalten. Dann wird der Puck über dem Ziel-Sketch-Objekt positioniert und der Knopf losgelassen (*Drag&Drop*). Der Inhalt des Quell-Sketch-Objektes wird daraufhin in das Ziel-Sketch-Objekt kopiert. Aus Sicht der Bedienung liegt der Unterschied zum einfachen Drag-Modus also darin, dass hier der Puck über dem Inhalt positioniert wird.
- **Skizzieren:** Um Ideen zu prototypisieren oder Anmerkungen an ein Sketch-Objekt anzubringen, muss der Anwender die Möglichkeit des Skizzierens haben. Aufgrund der Komplexität ist diesem Punkt ein eigenes Kapitel gewidmet. Siehe hierzu Kapitel 5.3.
- **Undo und Historie:** Über Undo können Anmerkungen und Skizzen, die an einem Sketch-Objekt angebracht wurden, schrittweise zurückgenommen werden. Über die Historie können diskrete Zustände eines Sketch-Objekte gespeichert, sowie auch verschiedene Sketch-Objekte miteinander verbunden werden. So ist es zum Beispiel möglich, die verschiedenen Entwicklungsstufen einer Idee aufzuzeigen.
- **Maßstab (Zoom):** Da die Anwender dieses Systems meist Ingenieure, Architekten oder Designer sind, ist die Darstellung eines Objektes in einem bestimmten Maßstab sehr vertraut, d.h. dass eine freie Wählbarkeit der Größenverhältnisse nicht notwendig oder sogar unerwünscht ist. Über das Toolglass-Menü kann der Maßstab des Prototype-Objektes entsprechend angepasst werden und somit auch auf die Größe des Paper-Objektes angepasst werden.
- **Skizzen „stapeln“:** Ebenso wie es auch mit „normalem“ Papier möglich ist, dieses übereinander zu stapeln, um dadurch Platz auf dem Schreibtisch zu schaffen, ist es auch hier möglich, virtuelles Papier zu stapeln. Hierzu müssen nur die einzelnen Sketch-Objekte übereinander gelegt werden. Durch einfaches Positionieren des Pucks auf dem Rand des Sketch-Objektes und kurzen Druck auf den Puck-Knopf, wird das selektierte Sketch-Objekt als oberstes auf den Stapel gelegt. Aus technischer Sicht wird hierzu das Sketch-Objekt an den Anfang der Liste der Sketch-Objekte eingefügt.

  Eine Erweiterung des Stapelns wäre die Möglichkeit nicht nur eine 2D, sondern eine 3D-Position einer Skizze bestimmen zu können. Das bedeutet, dass sich die Projektionsfläche (Tisch-Metapher) sich dem Anwender nicht als 2D-Fläche, sondern als echter 3D-Raum präsentieren würde. Diese Möglichkeit wird aus mehreren Gründen nicht angeboten. Es würde einen Bruch im Interaktionsmetapher bedeuten und zusätzliche Probleme in der Interaktion aufwerfen. Auch existieren Veröffentlichungen ([COCK02]), in denen dargelegt wird, dass die Verwendung der 3.

Dimension für eine Bedienoberfläche nicht immer effektiv ist und sogar kontraproduktiv sein kann. Ein 2D oder 2,5 D Ansatz ist in bestimmten Fällen vorzuziehen (wie das hier realisierte Stapeln).

## 5.2 MultiViews: Individuelle 3D-Sichten für mehrere Benutzer

In diesem Kapitel wird der im Rahmen dieser Arbeit entwickelte MultiViews-Algorithmus vorgestellt. Er ermöglicht individuelle 3D-Ansichten für mehrere Benutzer unter Verwendung einer einzelnen Projektionsfläche. Im folgenden wird nun kurz die Motivation für die Entwicklung dargelegt und anschließend der Algorithmus skizziert.

### 5.2.1 Motivation und Stand der Technik

Die Notwendigkeit dieses Algorithmus ergibt sich zwingend aus der Unterstützung des privaten Arbeitens. Hierbei muss der Anwender die Möglichkeit besitzen effektiv und intuitiv an einer Problemstellung arbeiten zu können, ohne die anderen Teilnehmer des Design Review zu stören, darf aber dabei nicht den Anschluß an eine stattfindende Diskussion zu verlieren. Wichtig für das effektive Arbeiten ist neben der Verwendung dedizierter Eingabegeräte auch die korrekte 3D-Ansicht des virtuellen Modells. Ersteres ist bei entsprechender Infrastruktur unter Verwendung der weiter oben beschriebenen Ansätze relativ einfach zu realisieren. Für letzteres werden zwar in der Literatur zwei konzeptionelle Ansätze beschrieben, die jedoch für diese Problemstellung weniger geeignet sind:

- **Mehrere Ausgabegeräte:** In [SCHM96] wird die Umsetzung verschiedener 3D-Sichten auf ein virtuelles Modell durch die Verwendung mehrerer See-Through HMD's realisiert. So kann jeder Anwender in seinem HMD die gewünschten Informationen visualisieren, ohne die anderen Beteiligten bei der Arbeit zu stören. HMD's sind jedoch wie in Kapitel 4.1 ausgeführt für den Einsatz im Bereich des Design Reviews nicht geeignet. Eine Möglichkeit wäre jedoch mehrerer Projektionswände/-tische. Hierbei sind zwei Szenarien vorstellbar:
  - Jeder Anwender arbeitet an einem dedizierten Ausgabegerät.
  - Für das private Arbeiten werden dedizierte Ausgabegerät eingesetzt. Zusätzlich gibt es ein „Hauptausgabegerät", an dem die Hautpdiskussion stattfindet

  Bei beiden Szenarien ist aufgrund der räumlichen Distanz die Kommunikation zwischen den Beteiligten beeinträchtigt ([LECA00]). Weiterhin sind auch die finanziellen Aufwendungen bei dieser Variante nicht zu unterschätzen.
- **Spezielle Hardware:** In [AGRA97] wird eine hardwarebasierte Modifikation eines Virtual Tables beschrieben, die es erlaubt zwei Anwendern eine persönliche 3D-Ansicht zu präsentieren. Hierzu wird die Projektion mit 120Hz-Shutterstereo betrieben und das dabei übliche abwechselnde Darstellen der Bilder für das linke und das rechte Auge so erweitert, dass zuerst die Stereobilder für den ersten und dann die Stereobilder für den zweiten Betrachter angezeigt werden. Das bedeutet jedoch, dass zum einen die Shutterbrillen modifiziert werden müssen und dabei die effektive Wiederholrate von 60Hz auf 30Hz pro Auge fällt, sodas die Bilder sehr stark flimmern. Leider sind heutige Projektionssysteme nicht in der Lage, Bilder mit wesentlich höherer Taktfrequenz darzustellen, sodas diese Lösung zum einen nicht beliebig skalierbar und zum anderen aus Gründen der Ergonomie für den anvisierten Verwendungszweck nicht geeignet ist. Aus technischer Sicht ist es auch in Zukunft nicht absehbar, dass sich die maximale Taktfrequenz stark

erhöhen wird.

Im folgenden Kapitel wird nun der Ansatz beschrieben, der im Rahmen dieser Arbeit entwickelt wurde.

### 5.2.2 Der MultiViews - Algorithmus

Der dem *MultiViews-Algorithmus* zugrundeliegende Ansatz unterscheidet sich sehr stark von den im vorherigen Kapitel beschriebenen Ansätzen. Es handelt sich hierbei um eine rein softwarebasierte Lösung, die insbesondere auf das private Arbeiten zugeschnitten ist. Hierzu wird auf das oben entwickelte Konzept der Sketch-Objekte zurückgegriffen. Mit MultiViews kann jedem Sketch-Objekt eine eigene Kamera (Betrachterstandpunkt) zugeordnet werden, aus deren Position dann das Sketch-Objekt dargestellt wird. Hierbei können prinzipiell beliebig viele Sketch-Objekte über ein Projektionsgerät dargestellt werden.

Der Einfachheit halber wird der Algorithmus auf Basis des Graphik-API OpenGL beschrieben, kann aber auch mit anderen hardwarenahen Graphikschnittstellen (z.B. Direct3D) realisiert werden

```
Lösche Z-, Back-Puffer
Setze gesamten Stencil-Puffer auf 0
Schalte Stencil-Puffer an
FOR all (Sketch-Objekte soi in Sketch-Objektliste)
  Setze OGL-Kamera auf soi(Kamera)
  Setze Z-, Back-Puffer auf read-only
  Konfiguriere Stencil-Puffer:
    PASS auf always
    Jedes passierende Pixel Pxy setzt Stencil-Pixel SPxy auf 1
  Zeichne Paper-Objekt soi(Paper)
  Konfiguriere Stencil-Puffer:
    PASS, wenn Stencil-Pixel !=0
    Setze Stencil-Puffer auf read-only
  Setze Back-, und Z-Puffer auf read-write
  Traversiere und rendere Szenegraph von soi(Graph)
  Konfiguriere Stencil-Puffer:
    PASS auf always
    Jedes passierende Pixel Pxy setzt Stencil-Pixel SPxy auf 0
  Zeichne Paper-Objekt soi(Paper)
ENDFOR
Schalte Stencil-Puffer aus
Setze Back-Puffer auf read-only
Setze OGL-Kamera auf Default-Viewpoint
FOR all (Sketch-Objekte soi in Sketch-Objektliste)
  Zeichne Paper-Objekt soi(Paper)
ENDFOR
Setze Back-Puffer auf read-write
Zeichne restliche Szene (z.B. toolglass-menu)
swap von Front-, und Back-Puffer
```

Abb. 5.4: MultiView-Algorithmus als Pseudocode

Der wohl entscheidende Nachteil dieser Methode ist sicherlich, dass ein bestimmtes Sketch-Objekt nicht gleichzeitig verschiedenen Betrachtern „korrekt“ dargestellt werden kann, wie es z.B. über die Methode aus [AGRA97] möglich wäre. Eine Lösung wäre, dieses Sketch-Objekt zu replizieren und jedem Betrachter eines zuzuweisen[1].

### 5.2.3 Performanzsteigerung durch Clustering

Multiple Ansichten eines komplexen virtuellen Modells bedeuten zwangsläufig, dass bei Verwendung eines einzigen Graphikrechners die Renderingperformanz mit zunehmender Anzahl immer weiter abfällt. Bereits bei zwei Ansichten fällt die Framerate im Schnitt auf 50%. Um diese Problematik zu entschärfen, können hier Ansätze, wie in Kapitel 6.2 entwickelt, eingesetzt werden.

Eine andere Möglichkeit ist, Clustering zu verwenden. Das Prinzip ist einfach: Mehrere Graphikrechner werden für das Rendering eines Bildes eingesetzt. Hierbei kann die Unterteilung des Bildes auf verschiedene Arten und unter Berücksichtigung unterschiedlicher Kriterien erfolgen. Abbildung 5.5 illustriert das Prinzip. Das Zusammenfügen (compositing) der Bilder kann entweder über einen weiteren Computer, der mit den Graphikrechnern über ein Netzwerk verbunden ist oder einer dedizierten Hardware (image composer) erfolgen. Siehe [CHEN01] für eine Übersicht.

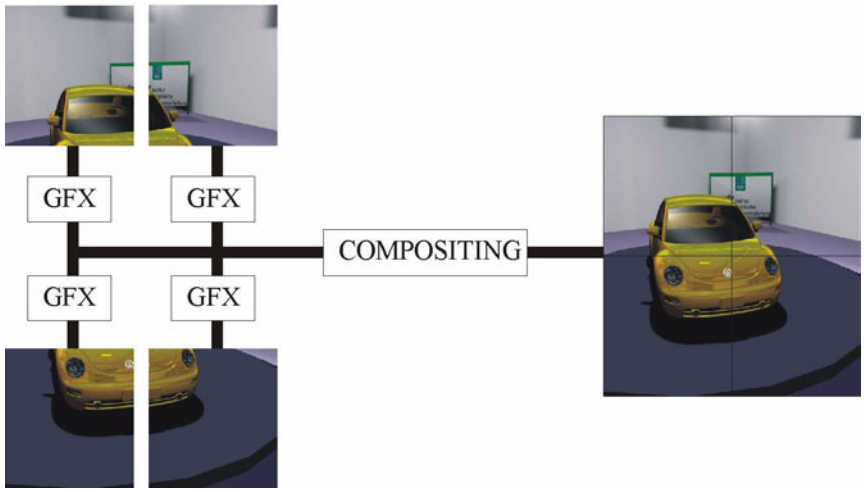


Abb. 5.5: Prinzip des Clusterings

Im Rahmen des Papier und Bleistift-Paradigmas könnte die Aufteilung der pro Computer berechneten Bildinhalte auf Basis der Sketch-Objekte erfolgen, z.B. ein Graphikrechner pro Sketch-Objekt. Dadurch könnte die Leistung des Gesamtsystems auf hohem Niveau erhalten bleiben und die Performanz jedes einzelnen Sketch-Objektes wäre unabhängig von allen anderen Sketch-Objekten. Dies setzt jedoch voraus, dass der compositing-Vorgang nicht auf die Fertigstellung aller Bildinhalte wartet und dann erst ein neues Gesamtbild ausgibt. Aufgrund der Eigenschaften von Sketch-Objekten können hier jedoch Lösungen gefunden werden.

1. An dieser Stelle sei auch auf die Arbeiten von Kitamura ([KITA01] verwiesen, der ebendieses Vorgehen über seine Illusionhole-Technik realisiert. Wie der Leser jedoch schnell feststellen können wird, ist diese Technik für den hier verfolgten Anwendungszweck wenig geeignet.

## 5.3 Skizzieren

Wie bereits in Kapitel 3.3.3 dargelegt ist das Skizzieren (*sketching*) von Ideen eine sehr wichtige Funktionalität, um insbesondere die Kommunikation der Teilnehmer an einem Design Review effektiv zu unterstützen. Es wurde auch gezeigt, welche Vorteile die Verwendung von Papier und Plots hat: Skizzieren kann einfach und natürlich durchgeführt werden.

Ziel dieses Teilaspektes der vorliegenden Arbeit ist es, das Skizzieren auf dem digitalen Prototypen ähnlich intuitiv zu gestalten, wie das Skizzieren auf einem Papier unter Verwendung eines normalen Stifts. Ziel hierbei ist nicht das Modellieren von Objekten, wie es bereits von einigen Systemen umgesetzt wird (siehe Kapitel 5.3.1), sondern in erster Linie das Annotieren, also ohne eine vom System initiierte Interpretation der Semantik der Zeichnungen. Weiterhin wird das Skizzieren in diesem Kontext auf Basis bereits existierender Modelle angewendet.

In den folgenden Kapiteln wird zuerst auf einige Ansätze aus der Literatur eingegangen. Im Anschluss wird das Prinzip des Skizzierens auf der Tischoberfläche dargelegt und dann mögliche Umsetzungen entwickelt. Den Abschluss bildet eine Diskussion der Ergebnisse und Erweiterungsmöglichkeiten.

### 5.3.1 Ansätze aus der Literatur

In der Literatur finden sich mehrere Ansätze, die das Modellieren und Konstruieren von Modellen auf Basis von Handzeichnungen umsetzen, jedoch keine Arbeiten, die sich mit dem Skizzieren als Annotation für bestehende Modelle auseinandersetzen. Im folgenden werden zwei Modellierungsansätze vorgestellt und es wird gezeigt, dass sie für den hier angestrebten Ansatz nicht geeignet sind.

SKETCH ([ZELE96]) wurde mit dem Gedanken entwickelt, die Lücke zwischen Handskizzen und computerbasiertem Modellieren zu schließen, in dem Funktionen beider Ansätze in einem System kombiniert werden. SKETCH basiert auf einem „Gestenerkennungssystem“ über das alle Interaktionen mit dem System durchgeführt werden können. Als Eingabemedium wird hierbei eine 3-Tasten Maus verwendet. Um nun ein geometrisches Objekt zu erzeugen, muss der Anwender bestimmte Gesten (*sequence of strokes*) mit der Maus ausführen. Hierbei steht ein fest definierter Satz an Gesten zur Verfügung, um einfache Grundprimitive, wie Würfel, Kugel, Quader, Extrusionskörper etc. zu erzeugen. So kann ein Quader über drei Linien, die sich in einem Punkt treffen konstruiert werden. Um das System bedienen zu können, muss der Anwender nur die vorgegebenen Gesten auswendig lernen. So ist das Konstruieren einfacher Grundprimitive mit diesem System sehr schnell und leicht möglich, komplexere Objekte können jedoch hiermit nicht erstellt werden.

Teddy ([IGAR99]) ist SKETCH sehr ähnlich, da es ebenfalls zum schnellen Modellieren von Objekten eingesetzt werden kann. Hier stehen jedoch nicht streng geometrische Figuren im Vordergrund, sondern Objekte auf Basis von Freiformflächen. So erzeugt das System aus einer gezeichneten 2D-Freiformfläche ein dreidimensionales, polygonales Modell dieser Freiformfläche. Über verschiedene Zeichengesten können Bereiche aus diesem Körper herausgeschnitten oder auch angefügt werden. Dabei ist es jedoch nicht möglich Löcher zu erzeugen. Alle so erzeugten Objekte sehen Spielzeugpuppen sehr ähnlich. Einige Beispiele, die mit Teddy erzeugt wurden, sind in Abb. 5.6 zu sehen

Abb. 5.6: Modelle, die mit Teddy erstellt wurden (aus [IGAR99])

Beide hier vorgestellten Systeme sind auf Ihre Art extrem leistungsfähig und zeigen neue Wege in der Mensch-Maschine Interaktion auf. Interessant hierbei ist, dass aufgrund von einfachen Gesten komplexere Aktion seitens des unterliegenden Softwaresystems ausgeführt werden. Jedoch sind weder SKETCH noch Teddy für das Sketching im Rahmen des Design Reviews geeignet, da sie das Zeichnen auf einem bestehenden Objekt nicht unterstützen. Auch sind die mit diesen beiden Ansätzen realisierbaren Objekte alle von einem sehr ähnlichen Typus.

Aus diesem Grund wird im folgenden Kapitel das Grundprinzip des Skizzierens für den Einsatz im Design Review erläutert und im Anschluss werden verschiedene Ansätze zur Umsetzung entwickelt und diskutiert.

### 5.3.2 Grundprinzip des Skizzierens

Das freihändige Zeichnen und Modellieren im 3D-Raum wurde von verschiedenen Forschergruppen eingehend untersucht (siehe [DEISS00]), ohne dabei wirklich ein überzeugendes Resultat zu erzielen. Konsens ist, dass solange keine Force-Feedback Geräte wie z.B. das PHANToM ([MASS94] eingesetzt werden und die Hand frei im Raum bewegt wird, die für diese Aufgabe benötigte Präzision nicht erzielt werden kann und sehr schnell Ermüdungserscheinungen eintreten. Beim Einsatz von unterstützenden Geräten sollte jedoch beachtet werden, dass diese die Sicht auf das Projektionssystem behindern können.

Aus diesem Grund erscheint es sinnvoll, die Projektionsfläche als Stütze für die Hand und Referenzpunkt zu verwenden und das Skizzieren auf der Oberfläche des Projektionssystems zu ermöglichen. Dadurch erhält der Benutzer die notwendige Kraftrückkopplung, um effektiv arbeiten zu können. Weiterhin ist diese Interaktion dem Skizzieren mit Stift und Papier sehr ähnlich. So kann der Anwender entweder eine 2D-Skizze erstellen oder auf einem dargestellten 3D-Objekt, dem virtuellen Prototypen, skizzieren. Das Zeichnen auf dem 3D-Objekt könnte jedoch für den Anwender verwirrend sein, wenn aufgrund der stereoskopischen Darstellung das Objekt aus der Projektionsfläche herausragt. Dann würde der Stift auf der Projektionsoberfläche sich innerhalb des Objektes bewegen. Dieser

Problematik kann jedoch sehr leicht begegnet werden, indem der Augenabstand auf Null gesetzt wird, sobald der Stift sich nahe der Projektionsfläche befindet. Dadurch wird der stereoskopische Effekt „ausgeschaltet“.

Um nun auf 3D-Objekten zeichnen zu können, müssen die 2D-Positionsdaten des Stiftes in 3D-Daten im Objektraum überführt werden. Hierzu wird ein Strahl vom Augpunkt (Kameraposition!) in Richtung der Spitze des Stiftes gesendet. Der dem Augpunkt am nächsten gelegene Schnittpunkt mit dem 3D-Objekt ist die gesuchte 3D-Position. Siehe hierzu Abb. 5.7.

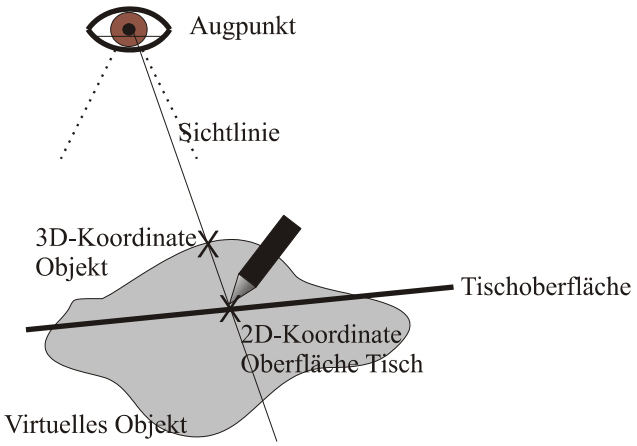


Abb. 5.7: Transfer der 2D-Koordinate der Stiftspitze in den Objektraum des 3D-Objektes

Dieser Punkt kann nun dazu verwendet werden einen sichtbaren 3D-Punkt zu erzeugen oder kann als Stützpunkt für einen Linienzug dienen.

### 5.3.3 Methodik für das Skizzieren auf 3D Objekten

In diesem Kapitel werden nun verschiedene Möglichkeiten zur Umsetzung des oben erläuterten Grundprinzips entwickelt und diskutiert. Hierbei werden die folgenden Methoden untersucht:

- Darstellung über Eckpunktfarben
- Darstellung über Texturen
- Darstellung als 3D-Liniensegmente
- Darstellung als 2D-Linien auf Texturen

#### 5.3.3.1 Darstellung über Eckpunktfarben

In diesem Fall wird wie in Abbildung 5.7 angegeben ein 3D-Punkt im Raum berechnet. Der diesem Punkt am nächsten liegende Eckpunkt des 3D-Objektes wird in der aktuellen Stiftfarbe eingefärbt. Sie hierzu auch Abb. 5.8. Dies setzt jedoch voraus, dass das 3D-Objekt entsprechend fein tesseliert ist, d.h. aus sehr vielen kleinen Polygonen besteht. Da dies bei typische CAD-Objekten, wie sie im Design Review verwendet werden, nicht unbedingt der Fall ist und auch ein nachträgliches retesselieren sich aus Geschwindigkeitsgründen verbietet, wird dieser Lösungsansatz nicht weiter verfolgt.

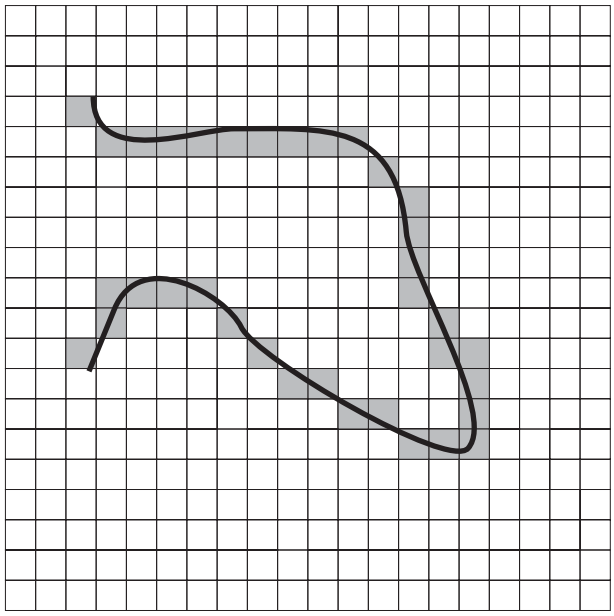

Abb. 5.8: Skizzieren durch Einfärben des Modells (hier schematisch auf Basis von Flächenfarben gezeigt)

#### 5.3.3.2 Darstellung über Texturen (Pixmaps)

Eine der ersten Dinge, die ein Modellierer für Echtzeitgraphiksysteme lernt, ist die Tatsache, so wenige Polygone zu verwenden und Details, die zur Darstellung eine hohe Anzahl an Dreiecken benötigen würden, mit Texturen darzustellen. Somit scheint es eine der besten Lösungen „leere“ Texturen zu verwenden und das virtuelle Objekte mit diesen zu texturieren. Anschließend kann direkt auf dem Objekt gezeichnet werden indem die 3D-Koordinate in den 2D-Texturraum der entsprechenden Textur transferiert und dort diese Textur entsprechend eingefärbt wird. Die Auflösung der Zeichnung ist hierbei direkt abhängig von der Größe der Textur und der Größe der Objektoberfläche, auf die sie abgebildet wird. Das löschen von Zeichnungen läßt sich über diesen Ansatz ebenfalls sehr leicht realisieren.

Die Problematik dieser Lösung liegt in der Abbildung der Textur(en) auf das virtuelle Objekt, die verständlicherweise automatisch erfolgen sollte. Hierbei muss gesichert sein, dass jedes Dreieck des Objektes texturiert ist und jeder Texturbereich nur genau einem Dreieck zugeordnet wird (u.a. [LEEB01]). Sinnvollerweise sollten dabei auch benachbarte Dreiecke mit benachbarten Texturbereichen texturiert sein, denn ansonsten kann es aufgrund der Texturfilterung zur Erstellung von Mipmap-Texturen (siehe z.B. [WOO99]) zu ungewollten Artefakten kommen, wenn diese Mipmap-Texturen angezeigt werden. Weiterhin sollte das Verhältnis Anzahl Texel (Texturpixel) zu Objektoberfläche konstant sein, um eine möglichst gleichförmige Zeichenauflösung zu garantieren.

Diese allgemeine Problemstellung ist auch unter dem Begriff des sog. *unfolding* bekannt, bei dem arbiträre 3D-Objekte auf eine 2D-Oberfläche abgewickelt werden. Diese Technik ist insbesondere bei der Verarbeitung von Blechen von großem Interesse ([GUPT98]). Um die Abwicklung zu erreichen, kann entweder das Objekt nur an den Kanten aufgeschnitten werden (*edge unfolding*) oder es werden zusätzlich auch Schnitte durch Flächen erlaubt (*general unfolding*). Es wird vermutet, dass jedes konvexe Polyeder (Solids) mit edge unfolding abgewickelt werden kann, mathematisch bewiesen ist je-

doch nur das general unfolding ([OROU98]). Typische CAD-Modelle bestehen jedoch meist auch aus konkaven Polyeder. Es wird zwar in [BERN99] gezeigt, dass es konkave Polyeder gibt, die nicht per edge unfolding abgerollt werden können, z.B. ein Tetraeder mit zusätzlichen Tetraedern auf jeder Seite. Siehe hierzu Abb. 5.9.

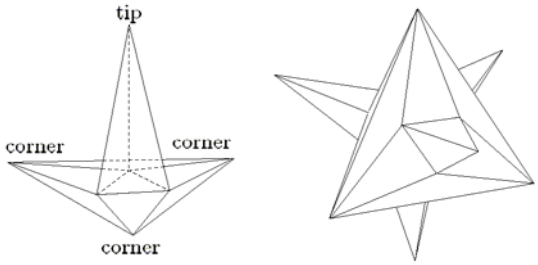


Abb. 5.9: Objekt, welches nicht mit edge folding abgewickelt werden kann (aus [BERN99])

Dieses Objekt kann jedoch per general unfolding abgewickelt werden (siehe Abb. 5.10) und steht somit nicht im Widerspruch zu dem o.g. Beweis.

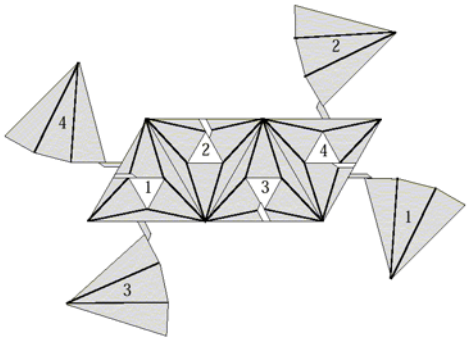


Abb. 5.10: General unfolding des Objektes aus Abb. 5.9 (aus [BERN99])

In dem selben Aufsatz wird jedoch auch gezeigt, dass es offene Polyhedra gibt, die nicht abgewickelt werden können. Offene Polyhedra sind hierbei non-solid Objekte, wie sie sehr häufig im CAD verwendet werden (z.B. zur Darstellung von Blechen).

Eine Lösung, um dieser Problematik generell zu begegnen, ist das Aufteilen des nicht abwickelbaren Objektes in abwickelbare Einzelobjekte (siehe Abb. 5.11). Lösungsansätze hierzu werden in [WANG97] beschrieben. Geht man von den dort genannten Berechnungszeiten aus, so würde die Berechnung der Abwicklung eines typischen CAD-Bauteils mit mehreren 10.000 Flächen im Stundenbereich liegen.

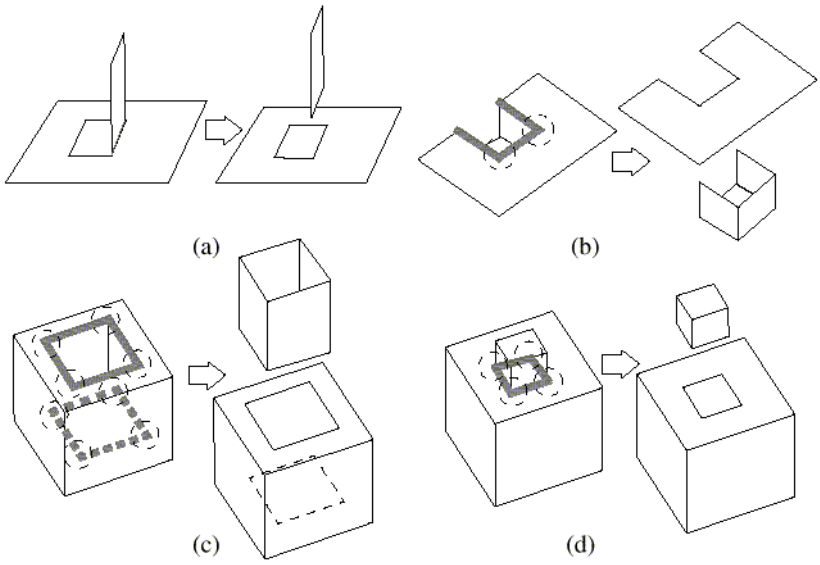


Abb. 5.11: Unfolding durch Separation (aus [WANG97])

#### 5.3.3.3 Darstellung über Linienzüge

Eine weitere Möglichkeit, um das Skizzieren zu realisieren ist die Darstellung mittels 3D-Linienzügen. So müssen keinerlei Texturen erzeugt und auf die Objektgeometrie aufgebracht werden. Auch müssen die 3D-Koordinaten auf der Objektoberfläche nicht in 2D-Texturkoordinaten umgewandelt werden, sondern können im Prinzip direkt verwendet werden, um die Stützpunkte der Linienzüge zu bilden.

Bei der Erzeugung der Linienzüge ist darauf zu achten, dass nur zu diskreten Zeitpunkten die 2D-Stiftkoordinaten vom System verarbeitet werden können. So kann es bei schnellen Bewegungen oder geringen Samplingfrequenz passieren, dass der Abstand zwischen zwei Samples relativ groß ist. Die dabei entstehende Problematik ist in Abb. 5.12 illustriert. Teile der skizzierten Linie liegen innerhalb des Objektes, wohingegen andere Abschnitte über dem virtuellen Prototypen schweben.

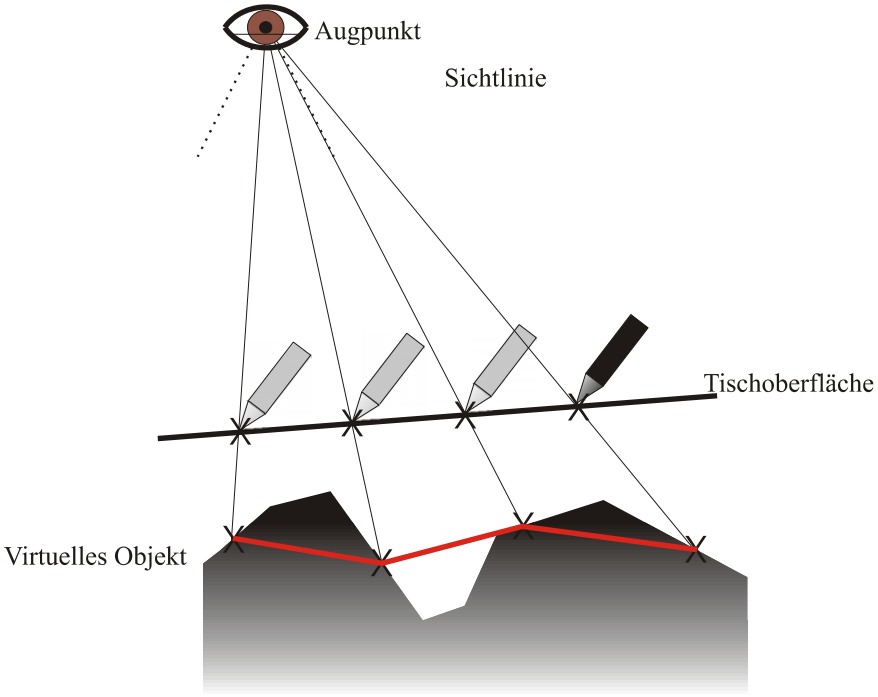


Abb. 5.12: Naiver Ansatz: Verbindung der gesampelten Punkte

Eine einfache Lösung wäre sicherlich die Samplingfrequenz soweit zu erhöhen, dass die dabei auftretenden Fehler minimiert werden. Dies hat jedoch zur Folge, dass sehr viele Linienstücke erzeugt werden müssen, wodurch die Darstellungsgeschwindigkeit unnötig beeinträchtigt wird. Auch ist die maximale Samplingfrequenz durch externe Faktoren (z.B. Tracking) begrenzt und somit kann diese Lösung nur zu einem unbefriedigenden Ergebnis führen.

Aus diesem Grund wird im folgenden Abschnitt ein Algorithmus entwickelt, der zum einen die Anzahl der erzeugten Liniensegmente minimiert und dabei ein optimales Ergebnis liefert.

**Der Skizzierungsalgorithmus**

Um den Linienzug aus Abb. 5.12 korrekt darzustellen, müssen zusätzliche Stützpunkte eingefügt werden, so dass weder das Objekt durchdrungen wird, noch die Linie über dem Objekt schwebt. Die prinzipielle Vorgehensweise dieses Algorithmus ist anhand von Abb. 5.13 illustriert. Hier sind die neu einzufügenden (virtuellen) Stützpunkte als „+“ dargestellt und ergeben sich prinzipiell aus dem Schnitt aller Polygonkanten mit der Projektion des Linienzuges zwischen den echten Stützpunkten auf die Oberfläche des Objektes.

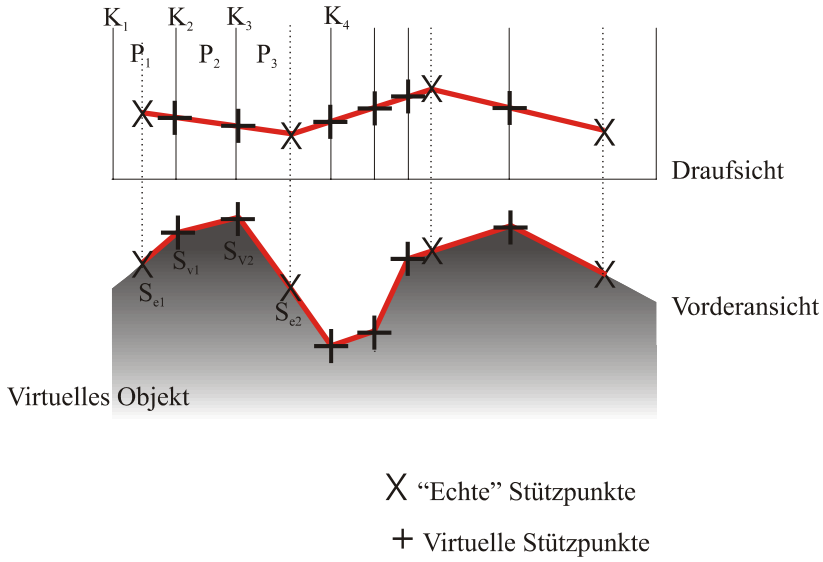


Abb. 5.13: Generierung virtueller Stützpunkte

Betrachten wir dazu den Linienzug $S_{e1}$=>$S_{e2}$. Der projizierte Linienzug auf die Oberfläche des virtuellen Objektes schneidet die Kante $K_2$ und $K_3$. Diese Schnittpunkte sind die neuen Stützpunkte $S_{V1}$ und $S_{V2}$. Daraus ergibt sich dann der Linienzug $S_{e1}$=>$S_{V1}$=>$S_{V2}$=>$S_{e2}$, welcher sich der Oberfläche des Objektes optimal anpaßt. Die Projektion des Linienzuges $S_{e1}$=>$S_{e2}$ auf die Oberfläche des virtuellen Objektes wird durch einen Schnitt der Ebene E, die über die drei Punkte $S_{e1}$, $S_{e2}$ und dem Augpunkt definiert wird und der Objektoberfläche erreicht. Zur weiteren Erläuterung des Algorithmus gehen wir davon aus, dass das Objekt trianguliert ist. Die Erweiterung auf nicht triangulierte Objekte ist größtenteils ein Implementationsproblem[1] und würde nur der Verständlichkeit der folgenden Beschreibung abträglich sein. Auch werden aus diesem Grund in der folgenden Darstellung Sonderfälle nicht betrachtet. Diese werden im Kapitel Behandlung von Sonderfällen auf Seite 123 eingehender erläutert und Lösungsstrategien entwickelt, sowie die Integration in den Algorithmus beschrieben.

Zu Beginn wird der Adjazenzgraph G(V,E) für das virtuelle Objekt berechnet, wobei hier die Polygone des Objektes durch V und die einzelnen Kanten der Polygone durch E repräsentiert werden. Hierbei wird davon ausgegangen, dass die virtuellen Objekte 2-mannigfaltig sind. Objekte, die keine Solids sind und nicht der Eulerformel

$$N_{vertices} - N_{edges} + N_{faces} - N_{holes} = 2(N_{components} - N_{genus}) \qquad (2)$$

entsprechen kommen jedoch im CAD recht häufig vor (z.B. Bleche). Hierbei treten Polygonkanten auf, die nur eine Nachbarfläche besitzen (Randkanten). Diese können z.B. im Graphen G durch Schleifen repräsentiert werden. Mehrfache Kanten sollten bei der Erzeugung des Graphen entfernt werden.

1. Um auch nicht triangulierte Objekte zu unterstützen, muss bei die Implementierung eine temporäre lokale Triangulierung der Flächen durchgeführt werden. Dies ist immanent wichtig, da nur bei Dreiecken garantiert werden kann, dass diese planar sind und es bei Nichtbeachtung dieser Tatsache zu Durchdringung der gezeichneten Linie mit dem Objekt kommen kann.

Nun wird für jeweils zwei aufeinanderfolgender 3D-Stiftpositionen $SP_1$ und $SP_2$ der folgende Algorithmus durchlaufen[1]:

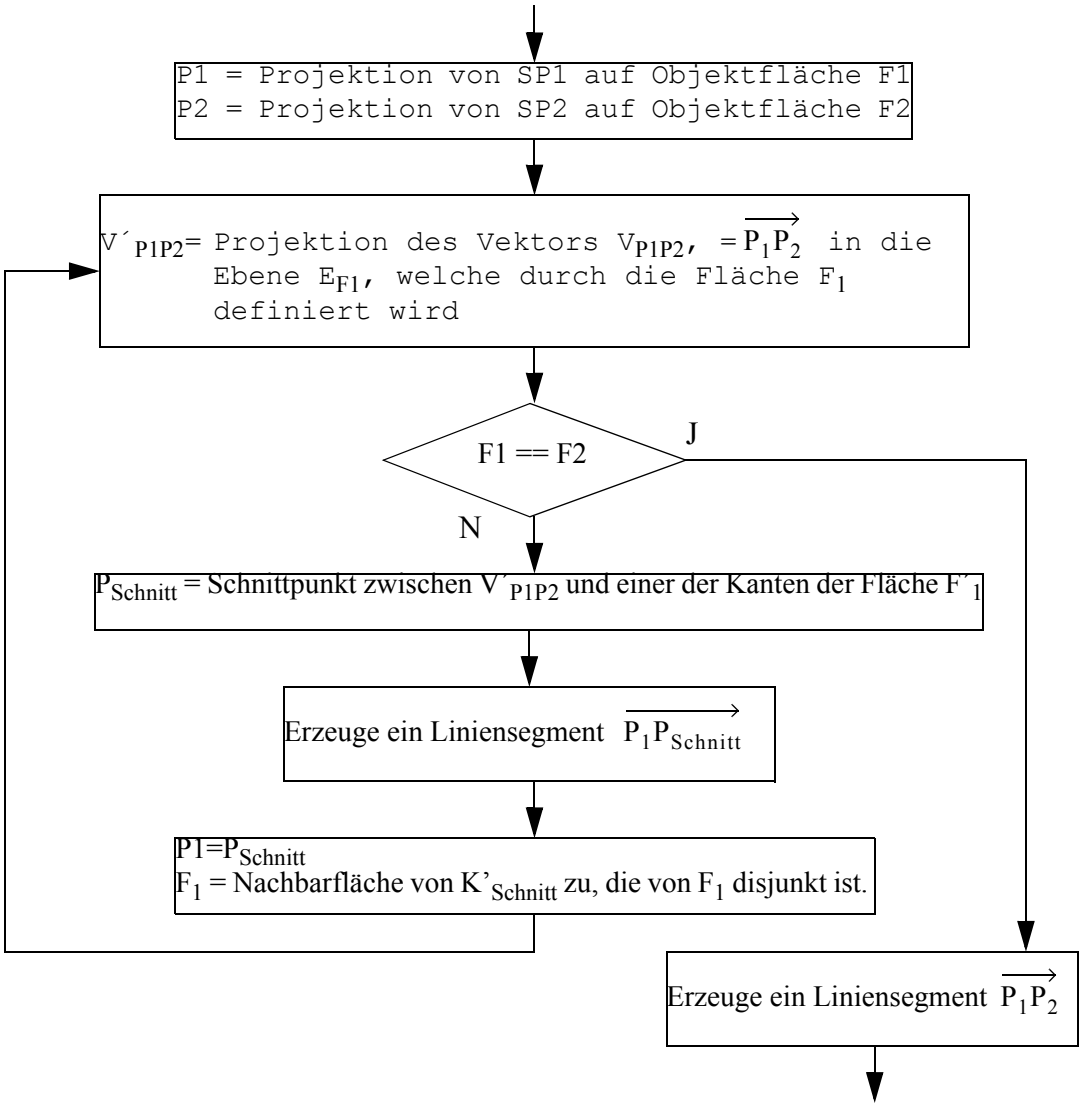


Abb. 5.14: Berechnung eines Linienzuges als Teil des Sketching-Algorithmus

Hierbei wird die Projektion von $SP_1$ und $SP_2$ auf die Objektoberfläche realisiert, indem die Vektoren $\overrightarrow{P_{Kamera}SP_1}$ und $\overrightarrow{P_{Kamera}SP_2}$ gegen alle Flächen des virtuellen Objektes geschnitten und die $P_{Kamera}$ am nächsten liegenden Schnittpunkte $P_1$ und $P_2$ sowie die zugehörigen Flächen $F_1$ und $F_2$ berechnet werden.

Die Projektion des Vektors $V_{P1P2}$ wird als Schnitt zweier Ebenen realisiert, wobei die eine Ebene durch $E_{F1}$ vorgegeben und die andere Ebene $E_{proj}$ durch $V_{P1P2}$, sowie $\overrightarrow{P_{Auge}P_1}$ und $\overrightarrow{P_{Auge}P_2}$ definiert wird. Dabei erhält man die Schnittgerade $V'_{P1P2}$.

1. Hier wird davon ausgegangen, die Geräte zur Positionserfassung (z.B. magnetisches Tracking) des Stiftes 3D-Werte liefert, die bereits im Koordinatensystem der virtuellen Welt vorliegen. Dies wird im allg. von allen VR-Systemen geleistet.

Der Schnittpunkt $P_{Schnitt}$ zwischen $V'_{P1P2}$ und einer der drei Kanten $K_1$, $K_2$, $K_3$ der Fläche $F'_1$ wird folgendermaßen realisiert: Da sowohl $V'_{P1P2}$, als auch die Kanten in einer Ebene liegen handelt es sich hier prinzipiell nur noch um eine 2D-Problemstellung. Der Einfachheit halber werden deswegen $V'_{P1P2}$ und die Kanten in eine der Grundebenen über einen Koordinatensystemwechsel $M_{Trafo}$ transformiert und nun kann mittels „Schnitt zweier Geraden in der Ebene“ der Schnittpunkt $P'_{Schnitt}$ berechnet werden. Die geschnittene Kante ist hierbei $K'_{Schnitt}$. Hierbei ist natürlich darauf zu achten, dass die Kanten nicht unendlich lang sind, sondern durch ihre Stützpunkte begrenzt sind. Über die Inverse von $M_{trafo}$ wird $P'_{Schnitt}$ in das Objektkoordinatensystem transformiert und man erhält den Schnittpunkt $P_{Schnitt}$.

Nach dem vollständigen Durchlaufen des Algorithmus kann nun ein Linienzug generiert werden, der die Punkte P1 und P2 verbindet und das virtuelle Objekt aus mathematischer Sicht nicht durchdringt, sondern direkt auf dessen Oberfläche aufliegt. Bei den heute verwendeten Graphik-API’s wie OpenGL und DirectX können jedoch numerische Rundungsfehler dazu führen, dass bei koplanaren Objekten die Darstellung zu flackern beginnt. Dies kann aber mit den von diese API’s zur Verfügung gestellten Funktionen weitgehend vermieden werden (*polygon offset*).

Eine zusätzliche Problematik entsteht bei der Handhabung von Linienprimitiven durch die Graphik-API. Hier wird die Liniendicke in Pixeln angegeben und somit ist die Darstellung unabhängig von der Entfernung zum Betrachters. Insbesondere die Tiefenwirkung (depth cue) wird dadurch negativ beeinflußt. Eine für OpenGL spezifische Lösung könnte hierbei die Darstellung der Linien unter Verwendung von Nebel (GL_FOG) sein, der jedoch nur auf den Alphakanal der Linienfarbe wirkt. Hierdurch verblassen die Linien, je weiter sie vom Betrachter entfernt sind, bis sie dann endgültig unsichtbar werden.

Ein großer Vorteil dieser Methode ist jedoch, dass die erzeugten Linieninformationen als Linienobjekte im System vorhanden sind und somit auch sehr einfach wieder vom Anwender gelöscht werden können. Weiterhin besteht hier die Möglichkeit auch im freien Raum zu skizzieren und nicht nur auf dem Objekt selbst (siehe Kapitel Behandlung von Sonderfällen auf Seite 123). Dies ist dann sehr hilfreich, wenn der Anwender z.B. eine Erweiterung des Objektes anzeichnen möchte.

**Behandlung von Sonderfällen**

Es kann der Fall eintreten, dass der Anwender eine Linie zeichnet, die nur teilweise auf dem Objekt zum liegen kommt. Eine mögliche Intention des Anwenders könnte sein, dass an einem virtuellen Objekt eine Erweiterung angebracht und dieses skizziert werden soll. Hier können wir zwischen verschiedenen Sonderfällen unterscheiden. Siehe hierzu auch Abb. 5.15:

- Der Linienzug beginnt auf dem Modell und endet außerhalb des Objektes (a)
- Der Linienzug beginnt außerhalb des Modell und endet auf dem Objekt (b)
- Der Linienzug beginnt und endet auf dem Modell, liegt jedoch teilweise außerhalb des Objektes (c)
- Der Linienzug liegt komplett außerhalb des Objektes (d)

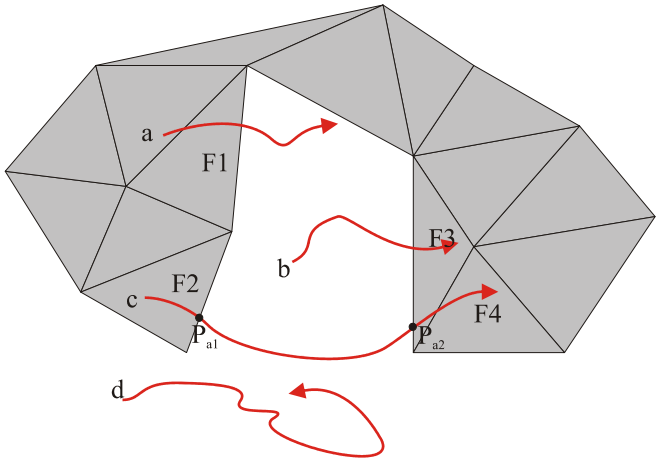


Abb. 5.15: Mögliche Sonderfälle beim Skizzieren außerhalb des Objektes

Die Grundproblematik dieser Sonderfälle ist die fehlende Höhenangabe, wenn außerhalb des Objektes gezeichnet wird. Im folgenden werden nun die verschiedenen Fälle betrachtet und Lösungen entwickelt:

Im Fall (a) ist der Beginn des Linienzuges definiert. Hier bietet es sich an die Ebene $E_{F1}$, die durch das Randpolygon $F_1$ definiert wird, zur Berechnung der Höhenwerte zu verwenden. Es wird hier also sozusagen ein unsichtbares Polygon eingefügt, auf dem die Linie zum Liegen kommt. Ähnlich ist Fall (c) gelagert, bei dem die Linie am Ende das Objekt wieder erreicht. Hier wird zu Beginn wie bei Fall (a) verfahren. Sobald jedoch die Linie wieder das Objekt trifft (Fläche $F_3$), werden die Höhenwerte des „freischwebenden“ Liniensegmentes korrigiert. Hierzu kann entweder linear oder mittels Hermite zwischen $P_{a1}$ und $P_{a2}$ die Höhenwerte interpoliert werden.

Fall (b) und Fall (d) ähneln sich soweit, dass der Startpunkt nicht auf dem Objekt liegt. Hier bietet es sich an den Höhenwert zu verwenden, der vom Stift geliefert wird, sodas der Linienzug genau in der Projektionsebene dargestellt wird. Sobald der Stift auf das Objekt trifft (Fall b), werden die Höhenwerte des freischwebenden Abschnittes wie in Fall (a) gehandhabt.

Ein weiterer Sonderfall kann bei konkaven Objekten oder bei großen Höhenunterschieden auftreten, da hier Stützpunkte auf unterschiedlichen Bereichen des digitalen Prototypen zum Liegen kommen können. Siehe hierzu Abb. 5.16.

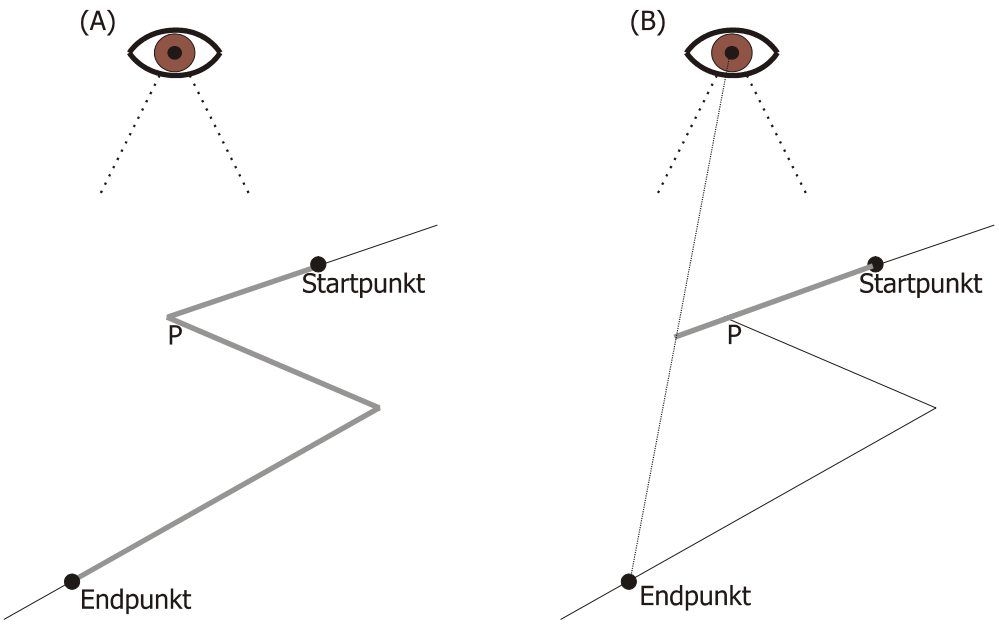


Abb. 5.16: Problematik bei konkaven Objekten oder großen Höhenunterschieden: 2 Lösungen

Hier gibt es zwei Lösungen, die jedoch von der eigentlichen Intention des Anwenders abhängen. Entweder wollte der Benutzer eine Linie auf dem Objekt zeichnen (Fall A) oder eine Erweiterung skizzieren (Fall B). Um zwischen diesen Fällen unterscheiden zu können wird ein Grenzwert eingeführt, über den zwischen Fall A und Fall B unterschieden werden kann. Hierzu wird der Ausdruck in der folgenden Formel berechnet und gegen diesen Grenzwert getestet. Aufgrund des besseren Verständnis wurden die Bezeichnungen aus der Beschreibung des Skizzierungsalgorithmus übernommen.

$$\left| \frac{\left|\overrightarrow{P_{Kamera}P_1}\right| - \left|\overrightarrow{P_{Kamera}P_2}\right|}{\overrightarrow{SP_1SP_2}} \right| = d \quad \begin{cases} d < e & \text{Fall A} \\ d \geq e & \text{Fall B} \end{cases} \tag{3}$$

So ist es dann z.B. für den Anwender sehr leicht möglich das Verkleinern einer Einkerbung zu annotieren. Siehe hierzu Abb. 5.17.

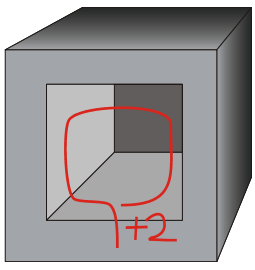


Abb. 5.17: Beispielhafte Annotation: Einkerbung verkleinern

**Optimierungsstrategien**

Eines der größten Problemstellungen bei der Verwendung dieses Ansatzes ist die Anzahl der zu ge-

nierenden Liniensegmente. Eine hohe Anzahl kann die Darstellungsgeschwindigkeit empfindlich senken.

Bei sehr hoher Samplingrate der Stiftbewegung kann es vorkommen, dass mehrere Stiftpositionen ($SP_x$) auf einem Dreieck liegen und dabei zum Größtenteil „redundant“ sind, da sie eine nahezu gerade Linie bilden (siehe Abb. 5.18). Hier ist es dann sinnvoll unter Berücksichtigung einer durch den Benutzer zu definierenden maximalen Abweichung von der ursprünglichen Linie diese Stützstellen zu entfernen.

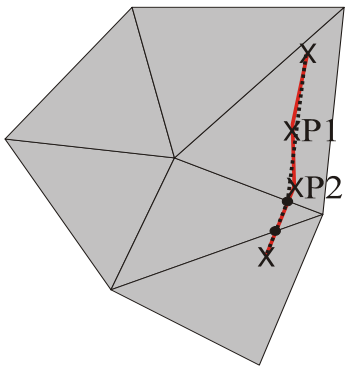


Abb. 5.18: Entfernung redundanter Punkte (hier $P_1$ und $P_2$)

Im folgenden wird ein Algorithmus zur Vermeidung redundanter Stützpunkte vorgestellt. Die maximale Abweichung wird als $d_{max}$ vorgegeben:

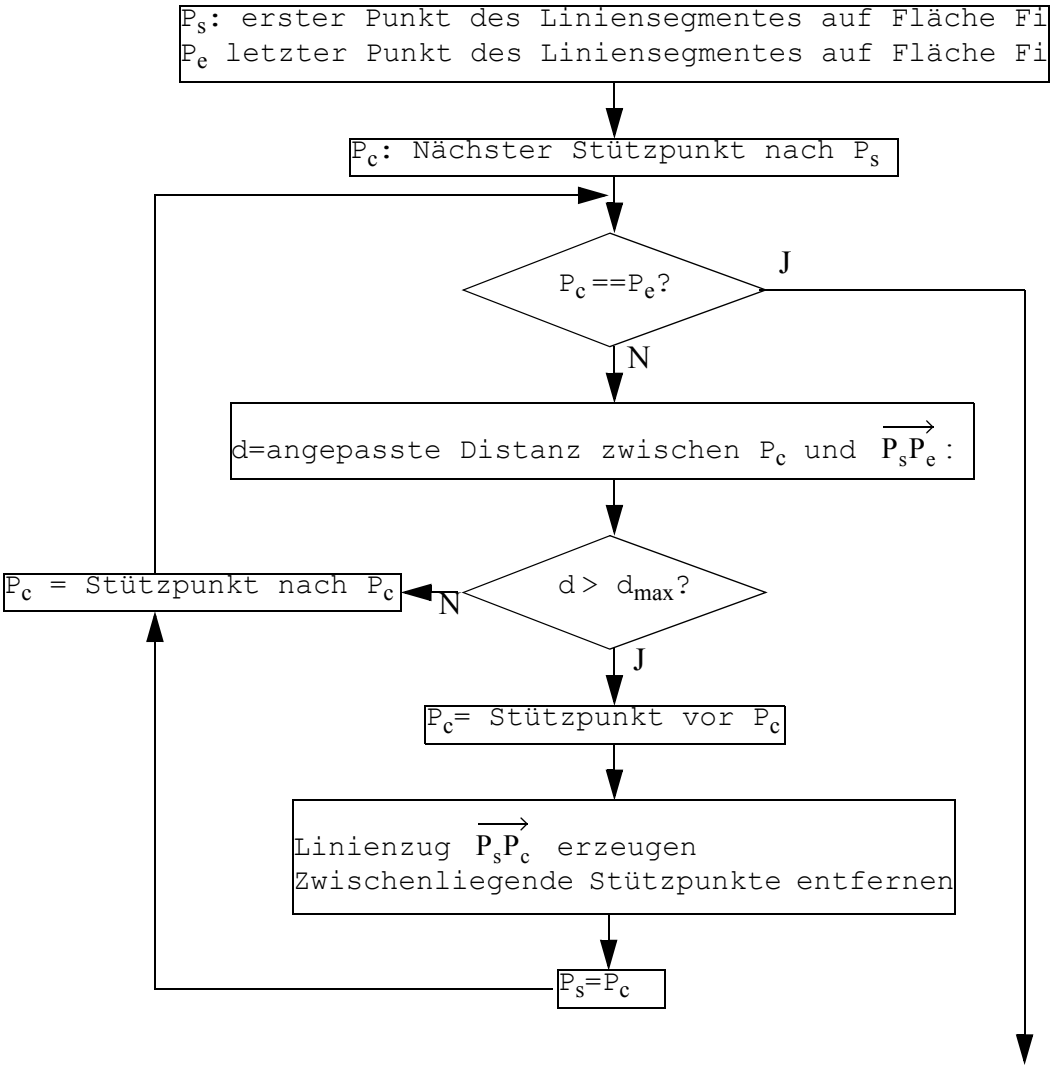


Abb. 5.19: Optimierungsalgorithmus

Das Berechnen der angepassten Distanz d zwischen $P_c$ und $\overrightarrow{P_sP_e}$ kann nicht über eine einfache Abstandsrechnung zwischen Punkt und Gerade gelöst werden, da hierbei nicht die Länge des Liniensegmentes beachtet wird und der in Abb. 5.20 illustrierte Fall nicht korrekt behandelt werden würde.

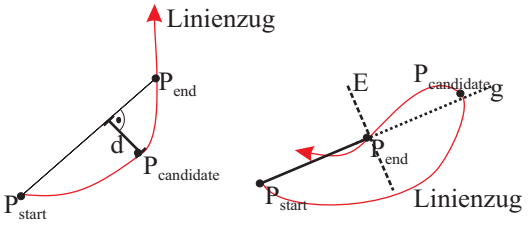


Abb. 5.20: Optimierung: Prinzip und Sonderfall

Um dies zu lösen bietet sich folgendes Vorgehen an:

Gegeben sei

$$E_c: \ \overrightarrow{P_sP_e} \cdot \zeta - \overrightarrow{P_sP_e} \cdot P_c = 0 \tag{4}$$

$$g: \ P_s + \lambda_s \overrightarrow{P_sP_e} = \zeta \tag{5}$$

Dann kann über ein Einsetzungsverfahren $\lambda_s$ und der Schnittpunkt $P_r$ zwischen $E_c$ und g berechnet werden:

$$\lambda_s = \frac{\overrightarrow{P_sP_e} \cdot P_c - \overrightarrow{P_sP_e} \cdot P_s}{\left|\overrightarrow{P_sP_e}\right|} \tag{6}$$

$$P_r = P_s + \lambda_s \overrightarrow{P_sP_e} \tag{7}$$

Um den in Abb. 5.20 illustrierten Sonderfall abzufangen muss nun auf Basis von $\lambda_s$ die entsprechende Abstandberechnung durchgeführt werden.

$$-\infty < \lambda_s < 0 \ : \ d = \left|\overrightarrow{P_sP_c}\right| \tag{8}$$

$$0 \le \lambda_s \le 1 \ : d = \left|\overrightarrow{P_rP_c}\right| \tag{9}$$

$$1 < \lambda_s < \infty \ : d = \left|\overrightarrow{P_eP_c}\right| \tag{10}$$

Dieses Verfahren kann nicht nur auf eine Fläche beschränkt sein, sondern prinzipiell auch flächenübergreifend angewandt werden. Dies zieht jedoch die o.g. Probleme nach sich. Solange jedoch der absolute Abstand zwischen Linie und Objektoberfläche nicht zu groß wird, wird der Anwender nicht bemerken, dass die Linie über dem Objekt schwebt und über einen entsprechenden *polygon offset* die Objektdurchdringung vermieden werden.

#### 5.3.3.4 Darstellung über Linienzüge und Texturen

Auch können die beiden zuletzt beschriebenen Methodiken kombiniert werden. Hierzu müssen die 3D-Linienzüge im Objektraum in 2D-Linienzüge im Texturraum umgewandelt und in die Textur gerendert werden. Da die Linienzüge weiterhin im System zur Verfügung stehen, kann das Löschen einzelner Linien sehr einfach realisiert werden, indem die restlichen 3D-Linienzüge wieder in die Texturen gerendert werden. Auch können bei der Transformation in 2D-Linienzüge im Texturraum noch Linienglättungsfunktionen eingefügt werden, um die Skizze „schöner“ darstellen zu können. Weitere Möglichkeiten wäre die Einbindung von Schrifterkennungssystemen, die auf Basis der Textur versuchen Zahlen und Buchstaben zu erkennen.

### 5.3.4 Diskussion

Wie bereits zu Beginn dargelegt sollte der Anwender nicht nur darauf beschränkt sein, auf dem vir-

tuellen Prototypen zu skizzieren, sondern auch daneben, um z.B. zu annotieren oder Erweiterungen zu skizzieren, die noch nicht Teil des Objektes sind. Dies kann prinzipiell mit jeder Methode realisiert werden, bedeutet jedoch bei den texturbasierten Ansätzen die zusätzliche Generierung einer Trägergeometrie, auf die die Textur gelegt werden kann. Deren Position und Lage kann hierbei analog zum Vorgehen für 3D-Linienzüge aus Kapitel 5.3.3.3 hergeleitet werden.

Problematisch hierbei ist jedoch die Festlegung der Größe dieser Fläche, da sie groß genug für die Annotation sein muss. Bei 3D-Linienzügen ist dies trivial, da hier die 3D-Koordinaten direkt vorliegen. Bei dem ausschließlichen Einsatz von Texturen wie in Kapitel 5.3.3.2 geschildert, liegen diese Daten jedoch nicht vor, sondern sind in der Textur gespeichert und müssen deswegen zusätzlich gespeichert werden.

Obwohl sowohl der Texturansatz als auch die 3D-Linienzüge für das Skizzieren eingesetzt werden könnten, wurde in dieser Arbeit der Ansatz auf Basis von 3D-Linienzügen gewählt. Dies liegt insbesondere darin begründet, dass der Prozess des unfoldings für die Komplexität der hier zu betrachtenden Objekte sehr hoch und die Lösung zeitaufwändig ist. Sollte an anderer Stelle ein befriedigender Ansatz entwickelt werden, so könnte dieser Weg erneut evaluiert werden.

In diesem Kapitel wurde in großem Maße auf die Basistechnologie des Sketchings eingegangen. Um die eigentliche Bedienung und Verwendung komfortabler zu gestalten, könnten bestimmten Linienzügen Aktionen zugeordnet werden. So könnte ein „X“ über einer Annotation deren Löschung bedeuten oder es könnten Funktionen, wie sie z.B. Teddy bietet, integriert werden.

Eine sicherlich interessante und wichtige Funktionalität, die hier unterstützt werden muss, ist die der Gruppierung. Über die Gruppierung können Teile der Skizze zu einem Gesamtobjekt zusammengefasst werden. Betrachte man noch einmal das Beispiel aus Abb. 5.17. Hier ist zum einen die kreisförmige Annotation und zum anderen der Text „+2“ zu erkennen. Aufgrund der in diesem Beispiel geschickt gewählten Positionierung, kommen sowohl die kreisförmige Annotation, als auch der Text auf der Oberfläche des Würfels zum liegen. Soll der Text nun nicht über, sondern neben dem Objekt zum liegen kommen, kann das System nur eine feste Positionierungshöhe annehmen. Über eine Gruppierung kann jedoch eine Höhe zugeordnet werden. Hierbei könnte folgendermaßen verfahren werden: Wenn einem Objekt $o_i$ die Default-Höhe zugewiesen wurde, so wird das zuerst selektierte Objekt $o_1$ als Referenz verwendet. Der $o_i$ am nächsten liegende Punkt p von $o_1$ wird zur Berechnung der korrekten Höhe verwendet.

## 5.4 Zusammenfassung

Kommunikation und Kooperation sind wichtige Elemente des Design Reviews. Wie in Kapitel 3.3.3 gezeigt, steht die Qualität des Reviews in direkter Abhängigkeit zu einer effektiven Kommunikation zwischen den Teilnehmern und den verwendeten Medien. In diesem Kapitel wurden Lösungen vorgestellt, um auf Basis digitaler Modelle eine effektive Kooperation und Kommunikation zu ermöglichen.

Das Basiskonzept ist dabei das im Rahmen dieser Arbeit entwickelte *Papier und Bleistift Paradigma*. Es erlaubt mehreren Personen, die sich an einem gemeinsamen Ort befinden, gleichzeitig an einem digitalen Prototypen zu arbeiten. Hierbei kann die Gruppe zusammenarbeiten, aber es besteht auch

für jeden Einzelnen die Möglichkeit, einen persönlichen Entwurf anzufertigen oder einer Idee nachzugehen (private spaces). Im Gegensatz zu herkömmlichen Methoden, bei denen nur eine singuläre Ansicht des digitalen Prototypen möglich ist, bietet das Papier und Bleistift Paradigma multiple und personalisierte Ansichten auf den digitalen Prototypen. Dies wird über den hier entwickelten *Multi-Views-Algorithmus* realisiert. Zusätzlich wurden sinnvolle Basisfunktionalitäten entwickelt und vorgestellt, die das effektive Arbeiten unterstützen.

Weiterhin wurde das Skizzierens auf dem digitalen Prototypen konzipiert und in das Papier und Bleistift-Paradigma nahtlos integriert. Diese Funktionalität ermöglicht es jedem Teilnehmer des Reviews direkt auf dem digitalen Prototypen seine Anmerkungen zu „zeichnen“. Dies können textuelle Annotationen sein, aber auch Strichzeichnungen, die Veränderungen des Designs andeuten sollen. Dieses Zeichnen wird nicht wie bei vielen Systemen üblich, frei im 3D-Raum ausgeführt, sondern direkt auf der Projektionsscheibe. Dies ist für den Anwender eine sehr natürliche Form der Interaktion, ähnlich des Zeichnens auf Papier und gibt ihm zusätzlich das notwendige Force-Feedback, um den Stift exakt führen zu können. Über speziell entwickelte Methodiken werden diese 2D-Informationen sinnvoll auf das digitale 3D-Modell übertragen.

Über das Papier und Bleistift-Paradigma und dem Skizzieren greifen nun die wichtigsten Mechanismen für eine gute Kooperation und Kommunikation nun auch bei „digitalen Prototypen“.

# 6 Funktionale Unterstützung des Design Review Prozess

In den vorherigen Kapiteln wurde auf die allgemeinen Anforderungen des Design Review Prozess an ein Softwaresystem eingegangen und verschiedene Applikationsbereiche vorgestellt. Weiterhin wurden verschiedene Interaktionstechniken vorgestellt, über die der Anwender mit dem System kommunizieren kann.

In diesem Kapitel wird nun die funktionale Unterstützung für den Design Review Prozess dargelegt. Wie in jedem größeren Softwaresystem ist die Zahl der Funktionalitäten immens und mit jedem neuen Entwicklungszyklus werden neue Funktionen hinzugefügt. Es wäre sicherlich ein unmögliches Unterfangen alle Funktionen, die für den Design Review Prozess im Hinblick auf die Applikationsbereiche wichtig sind, hier zu beschreiben, insbesondere da sich viele neue Funktionen erst aus der längeren Anwendung mit dem System ergeben. Deswegen wird hier „nur“ ein Set von Basisfunktionen vorgestellt, das absolut notwendig für den Einsatz eines solchen Systems ist und das sich auch basierend auf den gemachten Erfahrungen und Diskussionen mit dem Endanwender ergeben hat. Da die eigentliche Realisierung der Funktionen in den meisten Fällen offensichtlich ist, werden die Funktionen nur allgemein beschrieben und auf ihre notwendigen Ausprägungen eingegangen.

Die Basisfunktionalitäten lassen sich in drei Kategorien einordnen:

- Funktionen, die das Untersuchen des Modells unterstützen
- Funktionen, die die Dokumentation unterstützen
- Funktionen, die das Manipulieren von Offline-Simulationen erlauben.

Hohe Darstellungsraten, insbesondere in der Phase des Interagierens, sind für eine gute Anwendbarkeit des Systems von entscheidender Bedeutung. Hierbei hat man sich jedoch den aktuellen Hardwareentwicklungen zu unterwerfen. Diese werden zwar rasant leistungsfähiger, jedoch werden auch die zu untersuchenden 3D-Modelle immer komplexer, so dass sich die in der Praxis erzielten Frameraten in den letzten Jahren kaum geändert haben. Die in Kapitel 5.2 vorgestellten Konzepte erhöhen die darzustellende Komplexität zusätzlich. Ein Ausweg bildet hier eine *adaptive Darstellung* des virtuellen Prototypen, die in Kapitel 6.2.2 vorgestellt wird. Durch diese adaptiven Darstellungen kann die Renderingrate und die Interaktion verbessert werden (siehe auch [WARE94]).

## 6.1 Untersuchung des virtuellen Prototyps

In diesem Kapitel werden Funktionen entwickelt, die das Untersuchen eines virtuellen Prototyps auf Schwachstellen oder die Einhaltung von Richtlinien unterstützen. Folgende Funktionen wurden als wichtig erachtet und werden im Anschluss näher beschrieben:

- Sichtbarkeit von Objekten beeinflussen
- Stereodarstellung an- und ausschalten
- Kopftracking an- und ausschalten
- Drahtgitterdarstellung, Kamera zurücksetzen
- Distanzen messen
- Schnitte durch das Modell legen
- Orthogonale Ansicht

- Zeigefunktionalität für mehrere Benutzer

**Sichtbarkeiten**

Das Schalten der Sichtbarkeit sollte sowohl über eine Namensliste, als auch über Objektselektion mittels Strahl und Konus durchgeführt werden können. Wird die Sichtbarkeit über Namen gesteuert, so bieten sich die traditionellen hierarchischen Menüs an, da diese auch für sehr viele Objekte geeignet sind. Bei sehr vielen Objekten sollte die Liste durch verschiedene Hierarchiestufen, die sich z.B. aus den geladenen Modellen ergeben, verkürzt werden. Darüberhinaus sind Einträge, die sowohl alle Objekte sichtbar, alle Objekte unsichtbar und die Sichtbarkeit aller Objekte invertieren, als sehr sinnvoll anzusehen.

Wird das Schalten der Sichtbarkeit einzelner Objekte über den virtuellen Strahl oder Konus durchgeführt, so werden die Objekte durch Selektion auf unsichtbar geschaltet. Da die inverse Operation nicht über die Selektion möglich ist, muss ein undo-Buffer eingeführt werden, der in umgekehrter Reihenfolge Objekte wieder sichtbar schaltet (z.B. durch längeres Drücken der Taste, mit der normalerweise die Selektion bestätigt wird).

**Stereoskopische Darstellung und Tracking**

Durch die Stereodarstellung erhalten die Anwender sehr oft neue Einblicke und ein besseres Verständnis über die dargestellte Szene. So ist es u.a. für den Bau von Presswerkzeugen interessant einen Eindruck der Größe der Werkzeuge zu bekommen. Jedoch ist langes Stereo-sehen körperlich sehr anstrengend. Auch ist das Tragen der Stereobrillen auf Dauer unangenehm. Zusätzlich beeinträchtigen diese Brillen die Kommunikation zwischen den beteiligten Personen, da die Augen durch diese Brillen nicht sichtbar sind (ähnlich einer Sonnenbrille). Die Augen sind jedoch ein wichtiger Informationsträger bei der zwischenmenschlichen Kommunikation. Aus diesem Grund muss es möglich sein, die Stereodarstellung auszuschalten. Dies hat jedoch Implikationen auf die Benutzerschnittstelle.

Zum einen sind Interaktionstechniken, bei denen der Tiefeneindruck, der durch das Stereo-sehen entsteht, unterstützend wirkt, nicht mehr so gut zu verwenden.

Zum anderen wollen die Anwender nach dem Ausschalten der Stereodarstellung auch die Brille absetzen. Da das Trackingsystem in den meisten Fällen an der Brille befestigt ist, können Interaktionstechniken, die die Kopfposition und -orientierung kennen müssen, nicht verwendet werden (z.B. Image Plane Techniken, siehe Kapitel 4.4.2.2).

Weiterhin muss es die Möglichkeit geben, das Kopftracking explizit an- und ausschalten zu können. Da insbesondere bei längeren Diskussionen über eine bestimmte Situation die Blickrichtung nicht mehr geändert wird, ist es sehr störend, wenn sich das Bild bewegt. Sei es, weil die getrackte Person sich bewegt oder durch das Rauschen des Trackers. Auch wenn die getrackte Brille weggelegt wird muss das Tracking ausgeschaltet werden, da ansonsten je nach Lage der Brille das Bild in einer verzerrten Perspektive dargestellt wird.

In der Praxis hat sich gezeigt, dass die Anwender am Anfang sowohl Stereo, als auch Tracking verwenden, diese aber spätestens nach 30min ausschalten, da sie dann weder Stereo, noch Tracking für die weitere Diskussion benötigen.

In den folgenden Kapiteln wird nun auf die restlichen Funktionen aus obiger Liste eingegangen.

### 6.1.1 Distanzen messen

Eine sehr wichtige Funktionalität ist das Messen von Abständen, insbesondere zum Überprüfen der Einhaltung von Richtlinien. Hierbei kann der Benutzer zwei Marker auf der Oberfläche der Modelle setzen und anschließend wird die Entfernung zwischen den Markern ausgegeben. Siehe hierzu Abb. 6.1.

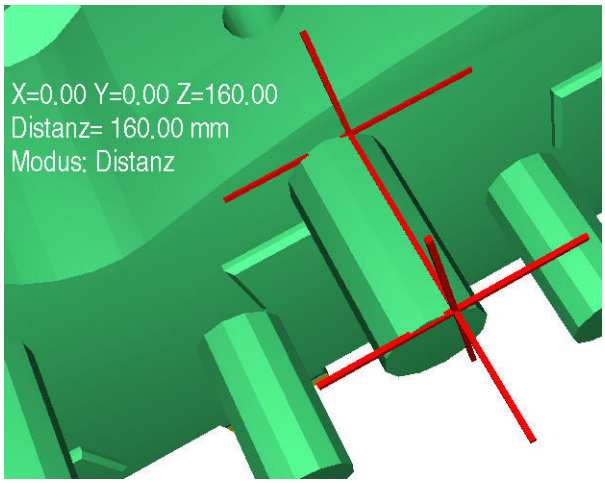


Abb. 6.1: Messen von Distanzen

Neben dem Abstand sollten auch die Distanzen der Marker entlang der Achsen des Koordinatensystems ausgegeben werden. Somit können Fälle, wie in Abb. 6.2 illustriert, ebenfalls sinnvoll behandelt werden.

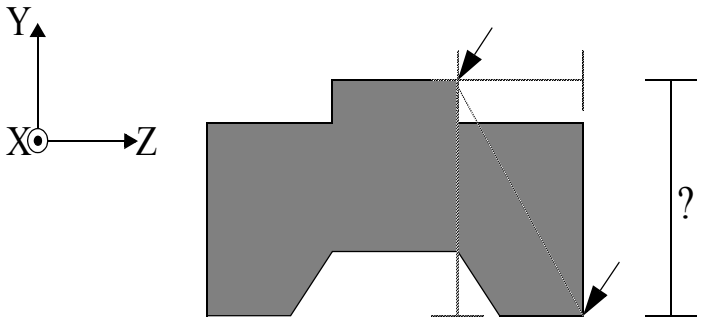


Abb. 6.2: Messen der maximalen Objektausdehnung in Y-Richtung

Hier soll die maximale Objektausdehnung in Y-Richtung berechnet werden und ohne Angabe der Distanzen entlang der Einheitsachsen müsste mehrfach gemessen werden.

Zum Positionieren eines Markers wird eine Selektionstechnik benötigt, die 3D-Punkte auf einer Oberfläche im 3D-Raum liefert. Dies leistet der virtuelle Laserstrahl (siehe Kapitel 4.4.2.1). Die Po-

sition des Markers wird durch den Schnitt zwischen Strahl und Modell definiert. Dadurch kann der Plazierungspunkt sehr genau angegeben werden und es ist sichergestellt, dass der Marker sich exakt auf der Oberfläche des Modells befindet. Das exakte Positionieren des Markers ist jedoch sehr schwierig, insbesondere, wenn der Marker auf eine Kante oder an den Rand einer Bohrung gesetzt werden soll. Aus diesem Grund sollte der Marker beim Plazieren zum nächstliegenden Eckpunkt des unterliegenden polygonalen Modells springen (Gravitationsfeld). Hierbei sollten jedoch nur Eckpunkte in Betracht gezogen werden, deren Distanz zum Schnittpunkt zwischen Strahl und Modell kleiner als ein gegebener Schwellwert ist. Ist kein Eckpunkt nah genug, so wird der Marker wie gehabt an den Schnittpunkt des Strahls mit dem Modell gesetzt. Der virtuelle Strahl bleibt hiervon unbeeinflußt.

Da der Anwender nicht genau wissen kann, wohin der Marker „springen“ wird, ist ein kontinuierliches Feedback sehr wichtig. Das bedeutet, dass der Marker zu einem Zeitpunkt t0 (durch auslösen eines Triggers) auf die Oberfläche gesetzt wird und dann der Auftreffpunkt des virtuellen Laserstrahls auf die Oberfläche kontinuierlich evaluiert und die Position des Markers entsprechend adaptiert wird. Wird ein weiterer Trigger ausgelöst (t1), ist die Punktselektion abgeschlossen[1]. Diese Vorgehensweise ist ähnlich dem „click and drag to select“-Selektionsmodus aus Kapitel 4.4.3.

**Algorithmus zum vereinfachten Messen eines Bohrlochdurchmessers**

Da insbesondere Preßwerkzeuge etliche Bohrlöcher aufweisen, deren Durchmesser überprüft werden muss, ist das Messen mit o.g. Methode aufwendig. Denn hierbei müssen die Marker immer an zwei exakt gegenüberliegende Eckpunkte des Bohrlochs gesetzt werden.

Es bietet sich also an, mittels eines Algorithmus herauszufinden, ob ein Marker an den Rand eines Bohrlochs plaziert wurde. Hierzu muss zwischen Bohrlöchern in Non-Solids und Solids unterschieden werden.

Bei Non-Solids kann ein Bohrloch über eine Randkantenverfolgung und die sog. „rechte Hand“-Regel gefunden werden.

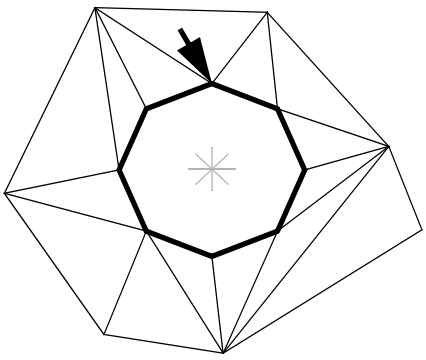

Abb. 6.3: Bestimmung eines Bohrlochs

1. Dadurch wird das Plazieren insbesondere an Bohrlöchern wesentlich einfacher, da der Anwender sofort sieht auf welchen Punkt des Bohrlochs der Marker springt.

Bei Solids kann aus offensichtlichen Gründen nicht mit dem Randkriterium gearbeitet werden. Hier wird eine Kante, deren anstoßenden Flächen in einem Winkel von 90° zueinander stehen, als Randkante betrachtet. Ansonsten wird analog verfahren.

### 6.1.2 Schnitte setzen

Eine sehr wichtige Funktion ist die Möglichkeit Schnitte durch Modelle legen zu können, um den inneren Aufbau des Modells sehen zu können oder um einen Querschnitt zu erhalten.

Um Schnitte setzen zu können, kann der Anwender eine Schnittebene an beliebige oder vordefinierte Positionen und Orientierung setzen. Die Schnittebene wird durch ein Objekt im Raum repräsentiert. Hierbei sollte es neben der Möglichkeit der freien Bewegung auch eingeschränkte Bewegung geben. Wichtig hierbei ist insbesondere das Festsetzen der Orientierung, um z.B. einen Schnitt entlang der Grundachse durchzuführen.

Weiterhin muss es möglich sein die Schnittebene auch aus größerer Entfernung manipulieren zu können, wobei das Rotationszentrum immer im Mittelpunkt der Schnittebene liegen sollte. Somit ist es dann möglich, die Schnittebene in das Modell hineinzuschieben und dann die Schnittebene zu rotieren, ohne dass sie ihre Position verändert. Hierfür bietet sich die *world point grab* Metapher mit definiertem Rotationszentrum aus Kapitel 4.3.4 an.

Die geschnittenen Objekte können entweder Solids oder Non-Solids sein. Dies sollte auch bei der Visualisierung des Schnittes beachtet werden, denn bei Schnitten durch Solids ist innen und außen definiert und somit ist die Schnittfläche definiert und kann z.B. mittels einer Schraffur sichtbar gemacht werden. Siehe hierzu Abb. 6.4.

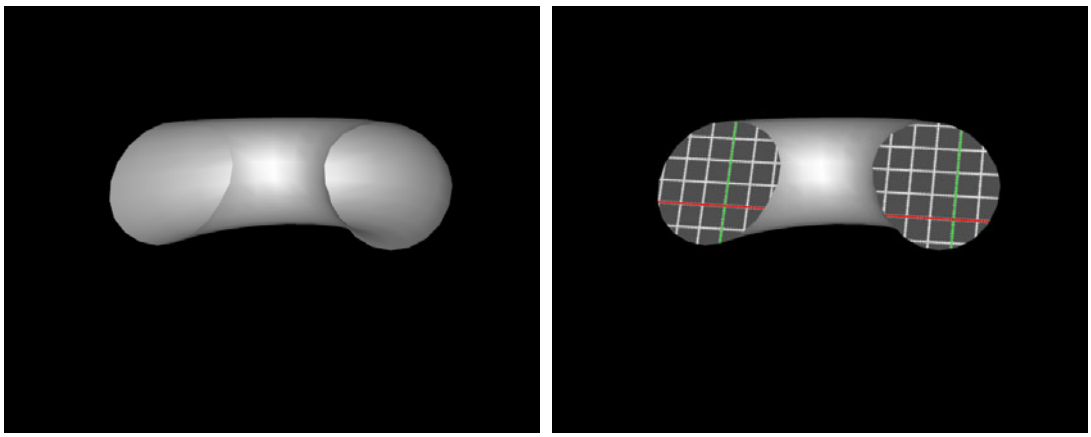

Abb. 6.4: Schnitt durch einen Torus: Einfach (links) und Solid (rechts)

Die Schnittfläche kann auch in der Objektfarbe eingefärbt und mit einem Raster versehen werden. Dadurch können Entfernungen und Größenverhältnisse besser abgeschätzt werden. Um verschiedenste Darstellungsmöglichkeiten der Schnittfläche zu unterstützen, wurde im Rahmen dieser Arbeit auch ein spezieller Schnittebenen-Algorithmus entwickelt. Dieser wird im folgenden Kapitel näher beschrieben.

#### 6.1.2.1 Schnittebenen-Algorithmus

In [BLYT99] wurde bereits ein Algorithmus zum Rendern von „Solid"-Schnitten vorgestellt. Bei diesem Algorithmus muss jedoch die Szene zweimal gerendert werden (2-Pass), wodurch die Darstellungsrate stark gesenkt wird.

In Rahmen dieser Arbeit wurde ein 1-Pass Algorithmus entwickelt, um nicht unnötig Darstellungsleistung zu verschwenden. Er basiert auf der folgenden mathematischen Eigenheit von Solids zur Bestimmung von innen und außen:

Man sendet ausgehend von einem beliebigen Punkt A einen Strahl in eine wiederum beliebige Richtung. Über die Anzahl der Schnitte zwischen Strahl und Solidoberfläche kann bestimmt werden, ob der Punkt A innerhalb oder außerhalb eines Solids liegt. Ist die Anzahl der Schnitte gerade (Anzahl modulo 2 ergibt 0), so liegt der Punkt A außerhalb, ist sie ungerade befindet sich der Punkt A innerhalb eines Objektes (Anzahl modulo 2 ergibt 1). Siehe hierzu Abb. 6.5.

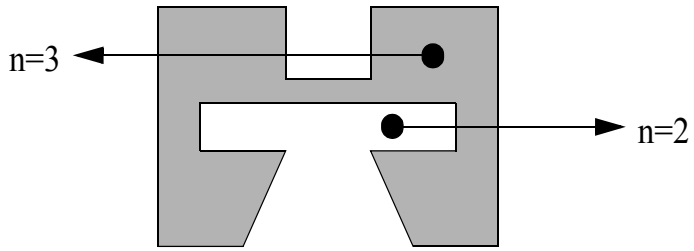


Abb. 6.5: Definition von innen und außen bei Solids

Die Ausrichtung der einzelnen Flächen des Solids ist für die korrekte Ausführung des Algorithmus von keiner Bedeutung.

Der Algorithmus wurde ursprünglich für OpenGL entwickelt, kann aber auch mit anderen hardwarenahen Graphikschnittstellen (z.B. Direct3D) realisiert werden. Diese Schnittstellen müssen jedoch frei definierbare Schnittebenen (im folgenden GL-Schnittebene genannt) und Stencil-Puffer unterstützen. Folgende Schritte umfaßt der Algorithmus:

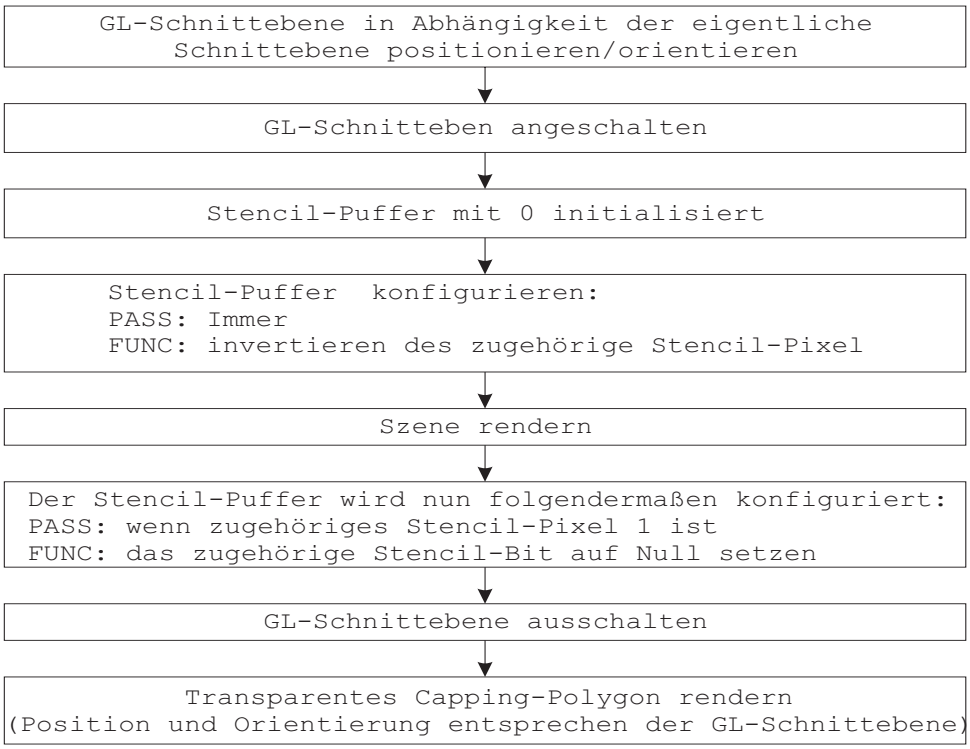


Abb. 6.6: SolidClip-Algorithmus (single pass, single color)

Mit dem oben beschriebenen Algorithmus hat jeder Schnitt durch die Szene das selbe Aussehen, welches durch die gerenderte Fläche im letzten Schritt vorgegeben wird. Um die Initialisierung des Stencil-Puffers zu sparen, kann beim Rendern der Fläche im letzten Schritt der Stencil-Puffer so konfiguriert werden, dass jedes gerenderte Pixel den Pufferinhalt auf 0 zurücksetzt.

Um verschiedene Modelle auch unterschiedlich einfärben zu können, kann der obige Algorithmus auch für jedes einzelne Objekt durchlaufen werden. Hierbei muss sichergestellt sein, dass jedes Einzelobjekt ein Solid ist. Das mehrfache Rendern des *capping polygons* kann jedoch zu einem Engpass in der Füllleistung der Graphikhardware führen, wodurch sich die Performanz verschlechtert.

Die obige Erweiterung funktioniert jedoch nur, wenn das gesamte Modell in einzelnen Solid-Objekten vorliegt. Ist dies nicht der Fall („polygon soup“) kann über einen erweiterten Algorithmus die spezielle Färbung erreicht werden. Hierbei handelt es sich um einen im Rahmen dieser Arbeit entwikkelten 2-Pass Algorithmus, der eine korrekte Ausrichtung der Flächennormalen voraussetzt. Weiterhin muss die Graphikschnittstelle einen *polygon offset* ([WOO99]) und Transparenzen unterstützen. Der Algorithmus umfasst die folgenden Schritte:

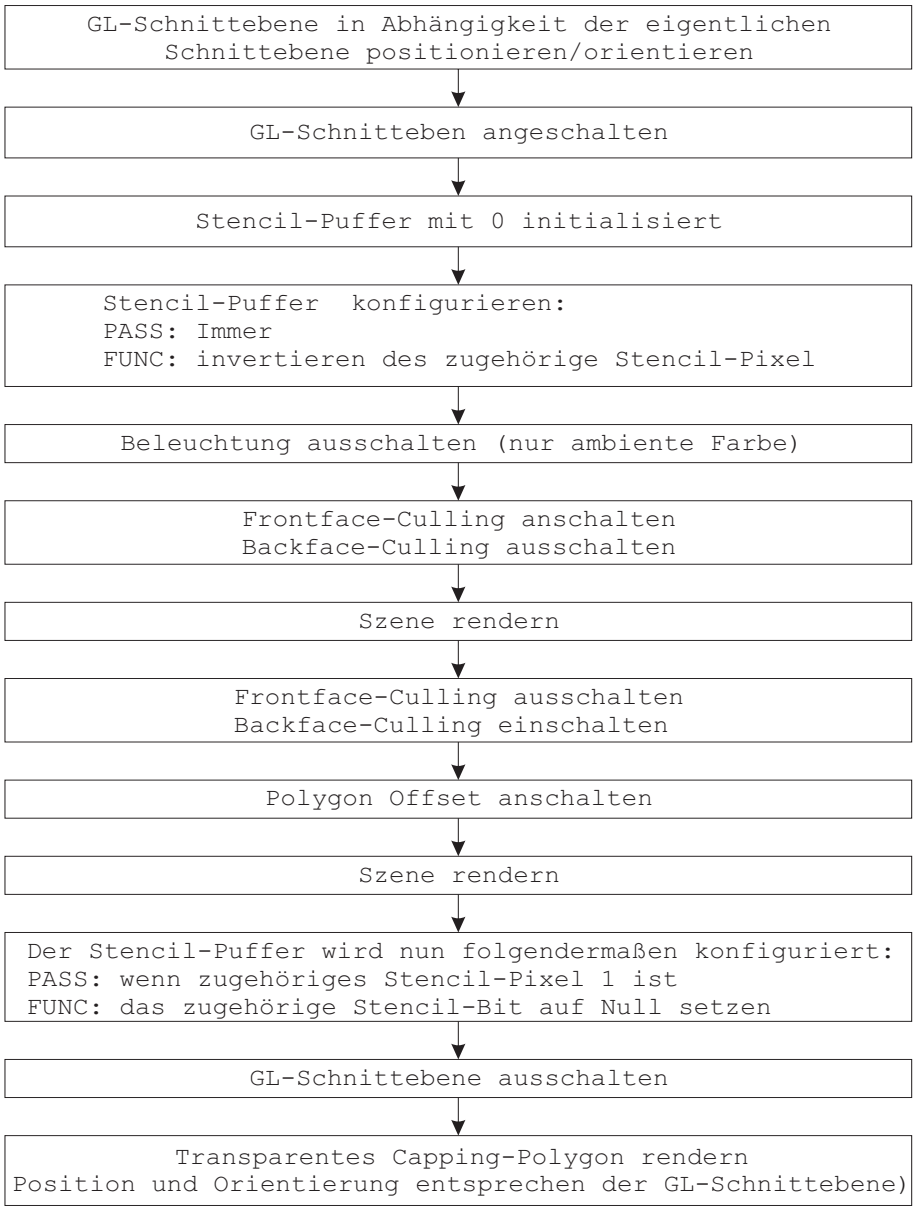


Abb. 6.7: SolidClip-Algorithmus (multi-pass, multi-color)

Das Ergebnis dieses Algorithmus ist in Abb. 6.8 zu sehen.

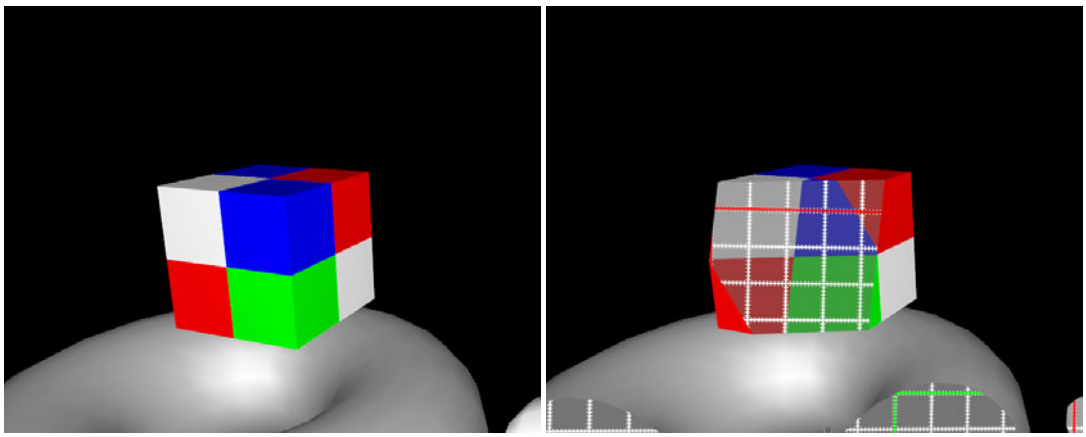

Abb. 6.8: Schnitt durch mehrfarbiges Objekt

### 6.1.3 Verschiedene Projektion

Für Anwender, die sehr lange am Zeichenbrett gearbeiten haben, ist die Interaktion in Virtuellen Umgebungen mit digitalen Prototypen sicherlich sehr ungewohnt. Insbesondere die andere Art der Darstellung wird eine größere Umgewöhnung bedeuten. Deswegen sollte ein solches System auch die Möglichkeit bieten, die Szene in einer achsparallelen orthogonale Ansicht darzustellen. Hierbei sollte es möglich sein, zwischen den drei verschiedenen Projektionsachsen umzuschalten und eine Schnittebene entlang der Projektionsachse präzise zu verschieben. Somit lassen sich ähnliche Darstellungen, wie vom Zeichenbrett gewohnt, erreichen. Zusätzlich bietet die VR-Lösung dem Vorteil, dass beliebige Schnitte durch das Modell gelegt werden können.

Neben der achsparallelen orthogonalen Ansicht sollte das VR-System auch die Möglichkeit einer Parallelprojektion bieten, da diese Form der Darstellung üblicherweise in CAD-Programmen verwendet wird.

### 6.1.4 Zeigefunktionalität für mehrere Benutzer

In Kapitel 3.3.3 wird u.a. die Frage nach der Anzahl von Zeigern behandelt. In [VINC99] wurde dazu festgestellt, dass der Einsatz von Laserpointern aus zwei Gründen nicht optimal ist:

- Aufgrund der nicht sichtbaren Zuordnung Lichtpunktes zu Gerät es für die Anwender schwierig ihren Marker zu identifizieren
- Es ist sehr aufwendig den anderen Teilnehmern zu kommunizieren, welcher Laserpunkt in diesem Moment interessant ist.

Um die visuelle Zuordnung Gerät zu Selektionspunkt zu erreichen, können virtuelle Laserstrahlen (Ray Casting, siehe Kapitel 4.4.2.1) eingesetzt werden. So ist auch die Zuordnung „Strahl zu einer Person“ für alle Beteiligten offensichtlich. Jeder Strahl wird über ein separates getracktes Eingabegerät gesteuert. Über Knopfdruck kann der Strahl an- und ausgeschaltet werden.

Auch hier ergibt sich wieder das Problem des exakten Positionierens und Orientierens im 3D-Raum, gepaart mit den auftretenden Ermüdungserscheinungen. Insbesondere das längere Fixieren eines 3D-

Punktes mit dem virtuellen Laserstrahl ist für einen Anwender sehr schwierig. Wird dann noch die Szene von einem anderen Anwender bewegt, um z.B. den fixierten Punkt besser ins Blickfeld zu bringen, ist es gar unmöglich, diesen Punkt weiter zu fixieren.

Eine Möglichkeit diesem Problem zu entgehen ist das Konzept der *Locked Beams*. Mit Locked Beams kann der Anwender über einen Event (z.B. Taste gedrückt) den Strahl frei bewegen. Schneidet der Strahl ein Objekt der Szene und wird in diesem Moment ein weitere Event ausgelöst (z.B. Taste loslassen), so bleibt das Strahlende an diesem Schnittpunkt „hängen“. Ansonsten wird der Strahl ausgeschaltet. Um den Strahl wieder zu lösen, muss der Anwender erneut einen Event auslösen (s.o). Somit ist es sehr einfach einen Problempunkt mit dem Strahl zu markieren und die Person, die die Navigation in der Szene übernimmt, kann durch diesen visuellen Hinweis sehr einfach diesen Punkt ins Blickfeld bringen.

## 6.2 Adaptive Darstellungen

Wie bereits in [WARE94] beschrieben, ist die Qualität der Interaktion sehr stark von der Framerate abhängig. Somit ist es für ein gut bedienbares und hochinteraktives System von großer Wichtigkeit, eine möglichst hohe Framerate zu garantieren. Diese ist in erster Linie von der Komplexität des virtuellen Prototypen abhängig. Wird das Papier und Bleistift Paradigma verwendet, kommen zusätzlich noch die in Kapitel 5 beschriebenen Sketch-Objekte, die diesen virtuellen Prototypen replizieren und somit die Gesamtkomplexität mit jedem Sketch-Objekt vervielfachen.

Auf diese Problematik wird nun in den folgenden Kapiteln eingehender eingegangen und eine Lösung entwickelt. Zuerst wird in Kapitel 6.2.1 der allgemeine Aufbau des Lösungsansatzes beschrieben. Im Anschluß werden zwei spezielle Kernpunkte dieser Methodik näher beleuchtet, der Outline Algorithmus (Kapitel 6.2.2) und der Threshold für Kopfbewegung (Kapitel 6.2.1.1).

### 6.2.1 Adaptive LOD’s und Texture Replacement

Um hohe Frameraten zu erzielen steht es außer Frage, dass die Komplexität der zu rendernden Szene reduziert werden muss. Wichtig dabei ist aber auch, dass keine visuellen Beeinträchtigungen dabei auftreten. Hierzu wird in der Literatur insbesondere der Ansatz der Polygonreduktion verfolgt. Hierbei werden Polygone aus dem virtuellen Objekt entfernt, ohne dass sich das Aussehen dadurch stark ändert. Siehe hierzu [KNOE98] für eine Übersicht.

Eine andere Möglichkeit besteht darin, dass das virtuelle Objekt in eine andere Darstellung überführt wird, z.B. in eine linienbasierte Umrissdarstellung (siehe Kapitel 6.2.2). Hierbei sind sehr hohe Reduktionsraten möglich. Zusätzlich ist das Darstellen von Linien auf vielen Hardwareplattformen schneller, als das Darstellen von Polygonen.

Die eigentliche Lösung liegt in der Beantwortung der Frage „*Zu welchem Zeitpunkt muss ein virtuelles Objekt gerendert werden?*“. Hierauf gibt es zwei Antworten:

1. Wenn der Anwender das virtuelle Objekt „greift“ und bewegt (*Interaktionsphase*).
2. Wenn sich die relative Positionierung von Kamera und Objekt ändert sich (z.B. wenn der Anwender den Kopf bewegt und das Objekt aus einer neuen Perspektive gerendert werden muss oder wenn das Sketch-Objekt bewegt wird)

Sobald also keine der oben genannten Bedingungen eintritt, kann auf das rendern des virtuellen Objektes prinzipiell verzichtet werden und es steht die gesamte Graphikperformanz für andere Aufgaben zur Verfügung.

Im Falle der *Interaktionsphase* ist jedoch ein ständiges Update notwendig, wobei aufgrund des Interagierens auch eine sehr hohe Framerate notwendig ist. Jedoch steht bei der Neupositionierung des virtuellen Modells die Erfüllung dieser Aufgabe im Vordergrund, d.h. visuelle Details des Objektes sind von geringerem Interesse. Es kann also in dieser Phase auf eine niedrig aufgelöste Version des Prototypen zurückgegriffen werden. Da eine Umrissdarstellung das Potenzial für eine hohe Performanz hat, wird im folgenden Kapitel der Outline-Algorithmus vorgestellt.

Da die Kameraposition und -orientierung mittels eines Trackingsystems ermittelt wird, welches leicht verrauschte Daten liefert und auch der Anwender sich immer leicht bewegt, wird es den Anschein haben, dass interagiert wird. Es muss hier also ein Schwellwert gefunden werden, über den das Renderingupdate kontrolliert wird, ohne die visuelle Wahrnehmung des Anwenders zu stark zu beeinflussen. Die Ermittlung dieses Parameters wird in Kapitel 6.2.1.1 aufgezeigt.

Man mag an dieser Stelle nun einwenden, dass aufgrund der heute verwendeten Graphik-APIs im Prinzip bei jedem Umschalten zwischen Front- und Back-Puffer die gesamte Szene neu gerendert werden muss. Weiterhin erzeugen Funktionalitäten, wie z.B. das Sketching oder das Plazieren von Markern neue Objekte in der virtuellen Szene. Dies kann jedoch über Textur- und Z-Puffer Ersetzung gelöst werden. Hierbei wird das virtuelle Objekt einmal dargestellt, in eine einzelne Textur umgewandelt und gesichert. Zusätzlich wird noch der Z-Puffer gespeichert. angepasst auf das Konzept der Sketch-Objekte bedeutet dies, dass das Prototype-Objekt in eine Textur gerendert wird und der Z-Puffer nach dem Rendern des virtuellen Prototype-Objektes, jedoch vor dem Darstellen von Zusatzobjekten, wie Sketches oder Markern, gesichert wird.

Da das System mehrere Sketch-Objekte unterstützt, kann bei einer sehr schlechten Ratio zwischen Graphikperformanz und Szenenkomplexität noch ein *Load-Balancing* mit eingeführt werden, um die Ressource Graphikleistung möglichst effektiv einzusetzen und eine hohe Interaktionsrate selbst bei mehreren Anwendern zu garantieren. Hierzu werden die Renderupdates priorisiert, wobei generell gelten sollte, dass die Interaktionsphase immer eine höhere Priorität als die Kamerabewegung hat.

In Abb. 6.9 wird der Vorgang der Texturgenerierung und eine Variante des Load-Balancing vorgestellt. Hierzu wurde zum einen das Sketch-Objekt dahingehend erweitert, dass zum einen festgelegt werden kann, in welcher Variante (Prototyp, Textur, Reduzierte Geometrie) es dargestellt wird und zum anderen, wie lange jede dieser Darstellungen auf der aktuell verwendeten Graphikhardware benötigt. Weiterhin wird ein Grenzwert $d_{threshold}$ eingeführt, der das Umschalten zwischen Texturvariante und Prototypengeometrie und damit die Generierung der Textur steuert. Zusätzlich wird ein Grenzwert $t_{max}$ für die maximale Zeit pro Frame eingeführt.

```
t = 0
t2= 0
FOR (alle Sketch-Objekte soi in Sketch-Objektliste)
  IF (soi(Prototype) ist gegriffen) THEN
    Schalte Rendergeometrie von soi auf reduzierte Variante
    t = t + soi(RenderTime_ReduzierteVariante)
  ELSE
    IF (CalcThresh(soi(Kamera),soi(Kamera_Last) < d_threshold) THEN
      IF (soi(TextureDirty) == TRUE) THEN
        erzeuge neue Textur/Z von soi aus soi(Prototype)
        Schalte Rendergeometrie von soi auf Texturvariante
        t = t + soi(RenderTime_Textur) + soi(RenderTime_Texturgen)
        soi(Kamera_Last) = soi(Kamera)
        soi(TextureDirty) = FALSE
      ELSE
        t = t + soi(RenderTime_Textur)
      ENDIF
    ELSE
      soi(TextureDirty) = TRUE
      soi(Kamera_Last) = soi(Kamera)
      Schalte Rendergeometrie von soi auf soi(Prototype)
      t2 = t2 + soi(RenderTime_Textur)
    ENDIF
  ENDIF
ENDFOR
FOR (alle Sketch-Objekte soi in Sketch-Objektliste)
  IF (soi(TextureDirty) == TRUE) THEN
    t = t + soi(RenderTime_Prototype)
    t2 = t2 - soi(RenderTime_Textur)
    IF (t + t2 > t_max) THEN
      Schalte Rendergeometrie von soi auf Texturvariante
    ENDIF
  ENDIF
ENDFOR
# rendere nun die Sketch-Objektliste von vorne nach hinten
```

Abb. 6.9: Pseudocode: Texturgenerierung und Load Balancing

#### 6.2.1.1 Schwellwert der Neuberechnung

Das Verwenden der Texturersetzung setzt jedoch voraus, dass die dabei auftretenden perspektivischen Verzerrungen bei Kopfbewegungen nicht zu groß werden und damit der visuelle Eindruck gestört wird. Insbesondere bei Mehrseitenprojektionen führt bereits ein kleiner Fehler in der Perspektive dazu, dass die Übergänge an den Kanten der Projektionswände nicht mehr stetig erscheinen. Weiterhin kann aufgrund der falschen Perspektive auch die Interaktion und z.B. das Abschätzen von Größenverhältnissen negativ beeinflußt werden.

Eine gute Wahl des Kriteriums $d_{threshold}$ ist somit sehr wichtig. Diese Fragestellung zu untersuchen ist jedoch äußerst komplex, da hier sehr viele Faktoren eine Rolle spielen, die im realen Einsatz äußerst schwer zu beeinflussen sind. Dies sind u.a.:

- Entfernung des virtuellen Objektes zur Kamera
- Beschaffenheit des Objektes (Umfang der Tiefeninformationen, Grad des Realismus)
- Anforderungen seitens der Anwendung
- Menschliche Wahrnehmung

Auch ist die Bewertung der einzelnen Effekte sehr kompliziert. Insbesondere da z.B. das korrekte Abschätzen von Größenverhältnissen auch bei normaler Darstellung problematisch ist[1]. Aufgrund dieser Faktoren ist ein Versuchsaufbau äußerst schwierig.

Im praktischen Einsatz zeigt sich jedoch, dass Anwender z.B. in der CAVE oder vor einer Powerwall die einmal eingenommene Position selten verlassen und auch sonst kaum den Kopf translieren. Es hat sich gezeigt, dass sich mit einem Wert für $d_{Threshold}$ von +/- 5cm recht gute Ergebnisse erzielen lassen.

### 6.2.2 Outline-Algorithmus

In diesem Kapitel wird der Outline-Algorithmus vorgestellt, der aus einer gegebenen Ausgangsgeometrie eine optimierte Umrissdarstellung erzeugt. Diese kann dann anstatt des eigentlichen Objektes dargestellt und dadurch die Graphikausgabe beschleunigt werden. Auch bietet dieser Algorithmus eine neue Möglichkeit, Selektion darzustellen, indem die Umrißdarstellung über das eigentliche Objekt eingeblendet wird. Dies ähnelt etwas der *scribed* Darstellung ist jedoch für die Graphikhardware wesentlich „günstiger“ darzustellen.

Die Umrissdarstellung wird hierbei aus den Flächenkanten des zu betrachtenden Objektes erzeugt. Hierzu wird zuerst eine Kantenliste erstellt, dann die Kanten klassifiziert und anschließend aufgrund der Klassifizierung die Umrisskanten ausgewählt. In einem nachfolgenden Schritt können diese noch zusätzlich optimiert werden. Siehe hierzu Abb. 6.10.

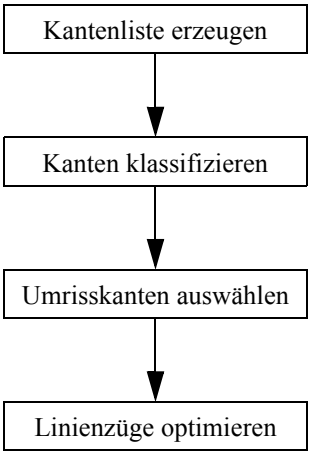


Abb. 6.10: Prinzipieller Ablauf des Algorithmus

Im folgenden wird nun auf die einzelnen Schritte des Algorithmus näher eingegangen.

1. Dies läßt sich höchstwahrscheinlich mit der Diskrepanz zwischen Akkommodation und Vergenz erklären

#### 6.2.2.1 Kantenliste erzeugen

In der Kantenliste müssen prinzipiell alle Flächenkanten gespeichert werden, um sie im Anschluß klassifizieren zu können. Voraussetzung ist jedoch, dass jede Kante nur ein einziges mal in dieser Liste abgelegt wird. Das bedeutet zum einen, dass Eckpunkte, die auf der gleichen Position zum Liegen kommen, zusammengefasst werden. Dadurch können Kanten zweier aneinanderstoßender Flächen als eine Kante erkannt werden. Zum anderen wird die Kantenrichtung ignoriert, so dass eine Kante $\overrightarrow{AB}$ der Kante $\overrightarrow{BA}$ entspricht.

Ein einfacher Ansatz wäre die Kantenliste in ihrer Gesamtheit aufzubauen. Dies bedeutet jedoch, dass bei einer Realisierung auf Basis einer Liste der Aufwand zum Einsortieren neuer Kantenelemente mit der Anzahl der Kanten linear steigt.

Bezieht man jedoch zwei einfache Randbedingungen mit ein, so wird sogar nur eine temporäre Kantenliste benötigt, in der nur die Kanten gespeichert sind, die von einem Eckpunkt abgehen. Hierzu gehen wir davon aus, dass die Liste der Eckpunkte des betrachteten Objektes von vorne nach hinten durchlaufen wird. Es sei

$i_{current}$: Aktueller Punktindex
$E(i_1, i_2)$: Betrachtete Kante mit den Stützpunkten $i_1$ und $i_2$, wobei gilt: $i_1 < i_2$

Kanten werden nur dann in die temporäre Kantenliste K eingetragen, wenn die folgenden beiden Bedingungen zutreffen:

$$i_1 = i_{current} \quad \text{oder} \quad i_2 = i_{current} \tag{11}$$

$$i_1 \geq i_{current} \quad \text{und} \quad i_2 \geq i_{current} \tag{12}$$

In der temporären Kantenliste K werden für jede Kante neben den zwei Stützpunkten auch die Nachbarflächen abgelegt. Der Sachverhalt sei kurz am in Abb. 6.11 illustrierten Beispiel erläutert. Es sei $i_{current} = 1$. Die folgenden Kanten werden in die temporäre Kantenliste eingetragen: (1,2), (1,6), (1,4), (1,10). Da im Algorithmus alle an den Punkt $P(i_{current})$ anstoßenden Flächen betrachtet werden, wird z.B. die Kante (1,2) sowohl bei F1, als auch bei F2 gefunden. Dadurch wird zum einen die Kante in die Liste K eingetragen und zum anderen die zweite Nachbarfläche gefunden.

Es sei nun $i_{current} = 2$. Die Kante (1,2) wurde bereits im vorherigen Schritt betrachtet und darf deswegen nicht in die temporäre Kantenliste aufgenommen werden. Dies wird durch die oben genannte Bedingung (12) verhindert, da hier $i_1 < i_{current}$ ist.

.

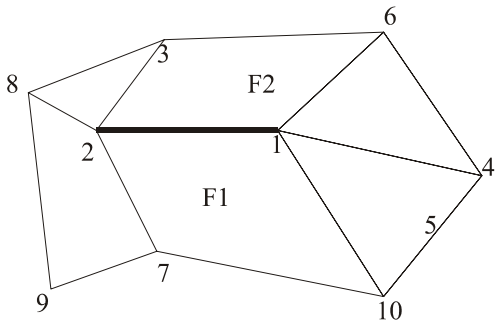


Abb. 6.11: Temporäre Kantenliste

Im folgenden wird nun der Algorithmus im Detail dargelegt.

```
i=0
WHILE (i < Anzahl aller Punkte)
  FOR j (über alle Flächen, die an P(i) anstoßen)
    FOR E (über alle Kanten E(p1,p2) der Fläche F(j) )
      IF (E(p1) == i || E(p2) == i)
        IF (E(p1) >=i && E(p2) >=i)
          /* in Kantenliste K eintragen */
          IF (E(p1,p2) in K enthalten)
            F(j) zu K(E(p1,p2)) hinzufügen
          ELSE
            Neuen Eintrag in K mit E(p1,p2) und F(j) erzeugen
          ENDIF
        ENDIF
      ENDIF
    ENDFOR
  ENDFOR
  /* Kanten klassifizieren (s.u.) */
  /* Kanten zum Umriß hinzufügen (s.u.) */
  i=i+1
  Lösche temporäre Kantenliste K
ENDWHILE
```

Abb. 6.12: Algorithmus zur Kantenerzeugung

Ist die temporäre Kantenliste aufgebaut, können die darin enthaltenen Kanten klassifiziert werden. Dies wird im folgenden Kapitel beschrieben.

#### 6.2.2.2 Klassifikation

Bei der Klassifizierung der Kanten wird zwischen drei Kantentypen unterschieden

- *Randkanten* haben nur eine Nachbarfläche

- *Innere Kanten* haben zwei Nachbarflächen, die in einem Winkel $\alpha > \varepsilon_{max}$ zueinander stehen.
- *Normale Kanten* sind weder Randkanten, noch innere Katen

Zur Umrißdarstellung werden dann Randkanten und innere Kanten verwendet. Dieser Sachverhalt ist beispielhaft in Abb. 6.13 illustriert. Hierbei wird $\varepsilon_{max}$ mit 45° angenommen. Dadurch ergeben sich drei innere Kanten (gestrichelt) und 14 Randkanten (punktiert).

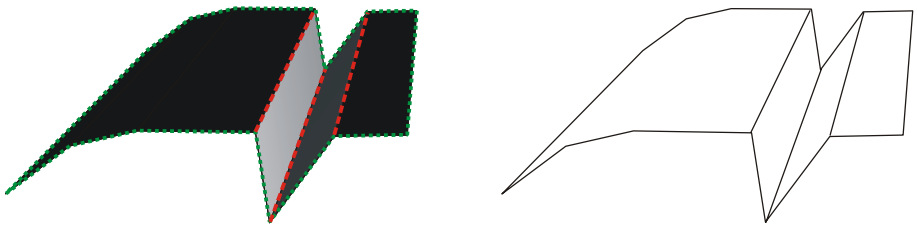

Abb. 6.13: Kantenklassifizierungen: Randkanten (grün/punktiert), Innenkanten (rot/gestrichelt)

#### 6.2.2.3 Optimierung

Als abschließender Schritt werden die gefundenen Umrißkanten optimiert. Dieser Prozeß ist zweistufig:

1. *Generierung von Linienzügen* (linestrips), um die Anzahl der zu transformierenden Eckpunkte zu verringern.
2. *Linienreduktion:* Weichen zwei aufeinanderfolgende Linienabschnitte in ihrer Orientierung voneinander nur geringfügig ab, so werden sie zu einer Linie zusammengefaßt. Hierzu kann der in Kapitel Optimierungsstrategien auf Seite 125 vorgestellte Ansatz verwendet werden.

Um möglichst lange Linienzüge zu generieren, kann ein rekursiver Ansatz gewählt werden. Hierbei wird mit der ersten noch nicht bearbeiteten Kante begonnen und in beide Richtungen anhängende Kanten gesucht. Wurden mehrere weiterführende Kanten gefunden, so wird über Rekursion der längstmögliche Linienzug bestimmt oder bis eine max. Länge erreicht wurde.

#### 6.2.2.4 Beispiele

Bei Verwendung dieses Algorithmus wird die Anzahl der zu transformierenden Punkte auf typischerweise ca. 25% der Ausgangspunktemenge reduziert. Die Anzahl der Linien liegt bei ca. 35% der Anzahl der Dreiecke des Ausgangsobjektes. In den folgenden Abbildungen sind zwei Beispiele des Algorithmus dargestellt.

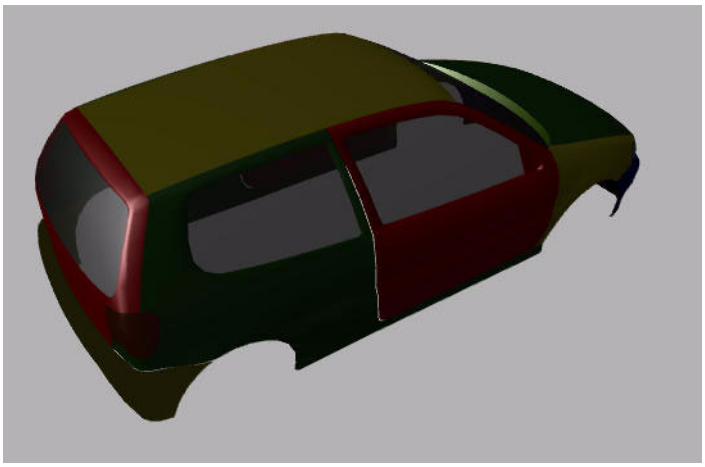

Abb. 6.14: Ausgangsmodell VW Polo (4384 Dreiecke, 6822 Punkte)

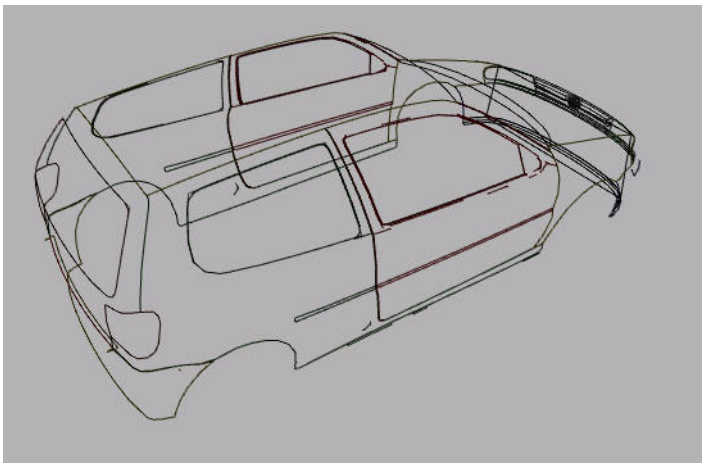

Abb. 6.15: VW Polo als Kantenmodell (1669 Linien, 1786 Stützpunkte)

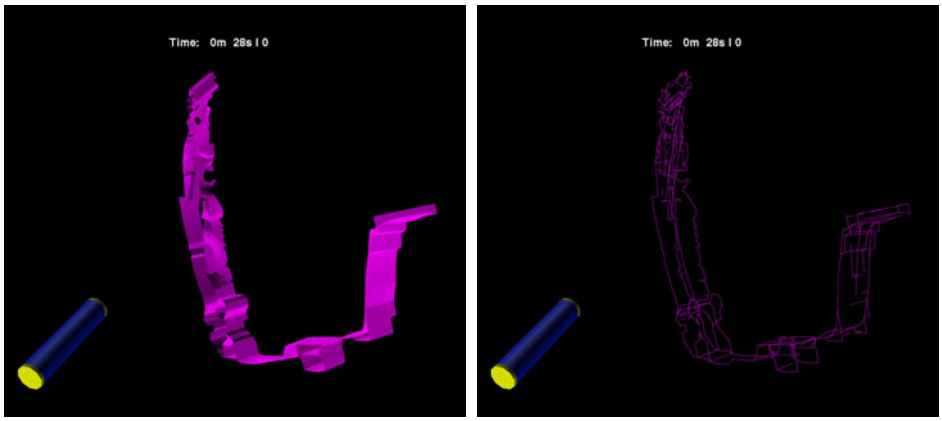


Abb. 6.16: Der Outline-Algorithmus am Beispiel von FE-Daten
(2911 Dreiecke und 4107 Eckpunkte zu 874 Linien und 958 Eckpunkte)

## 6.3 Dokumentation

Die Ergebnisse eines Design Reviews werden während der eigentlichen Sitzung in einem Protokoll festgehalten. Um Problemestellen besser beschreiben zu können, wird z.B. bei der Verwendung von Zeichnungen im Protokoll auf Markierungen in der Zeichnung verwiesen. Diese Markierungen wurden während des Reviews zur Zeichnung hinzugefügt. Für die Überarbeitung des Modells steht dem Konstrukteur / Designer dann das Protokoll und die Zeichnung zur Verfügung.

Für die Dokumentation auf Basis digitaler Prototypen muss es ebenfalls eine Möglichkeit geben, um Markierungen in die Szene zu setzen und anschließend diese dann zusammen mit dem Protokoll dem Konstrukteur zur Verfügung zu stellen. Die Markierungen können auf verschiedene Arten zur Verfügung gestellt werden:

- Die Darstellung der Problemzone als statisches Bild mit sichtbaren Markern
- Modell und / oder Marker liegen als 3D Modelle in einer standardisierten Szenenbeschreibungssprache (z.B. VRML-2) vor.
- Die Marker werden direkt in den CAD-Daten gespeichert.

In allen Fällen muss der eigentliche Text des Protokolls mit der Markerdarstellung verknüpfbar sein. Hierfür bietet sich HTML ([HTML99]) und als zukunftsträchtige Technik XML ([XML98]) an. Zur Darstellung von 3D-Modellen bietet sich VRML/X3D an [VRML97] Der Vorteil dieser Beschreibungssprachen ist die enge Verknüpfung mit dem Internet und den zugehörigen Diensten und Softwarelösungen. Da heutzutage für alle bekannten Hardwareplattformen Web-Browser zur Verfügung stehen, kann jeder Konstrukteur das Protokoll an seinem Arbeitsplatzrechner anschauen. Auch die Einarbeitungszeit in die Bedienung eines Web-Browsers kann als minimal angesehen werden.

Neben der Verfügbarkeit geeigneter Softwarewerkzeuge ist auch die einfache Integration in das Intranet eines Unternehmens als positiver Aspekt der Verwendung dieser Technologien zu sehen.

Sobald in einem Design Review ein Modell erneut vorgelegt wird, wird zuerst das Protokoll des letzten Reviews durchgegangen und die dort festgehaltenen Problemstellen anhand des aktuellen Mo-

dells erneut überprüft. In Kapitel 6.3.3 wird hierzu eine Methode vorgestellt, mit der dieser Prozess auf sehr einfache und elegante Art und Weise durchgeführt werden kann.

### 6.3.1 HTML/XML basierte Dokumentation

Bei HTML/XML basierter Dokumentation werden keine Informationen im eigentlichen CAD-Modell abgelegt, d.h. das Ergebnis der Sitzung liegt in einer oder mehrerer externer Dateien vor. Eine wichtige Fragestellung ist hierbei, in welcher graphischen Form die Markierungen abgelegt und dem Anwender präsentiert werden. Wie bereits oben dargestellt kann dies entweder über statische Bilder oder VRML97/X3D-Geometriemodelle realisiert werden.

Eine VRML97-Szene kann mit den meisten heute erhältlichen Web-Browsern betrachtet werden. VRML97 bietet neben der freien Navigation durch die Szene auch die Möglichkeit verschiedene Kamerapositionen abzuspeichern und diese auf Knopfdruck anzufahren. Der Protokolltext liegt dann ebenfalls als XML/HTML vor, damit dieser auch im Web-Browser dargestellt werden kann.

Die Erstellung des Protokolltextes sollte jedoch innerhalb des Design Review Softwaresystems durchgeführt werden und nicht z.B. mit einem externen Editor. Dadurch besteht dann die Möglichkeit, automatisch wichtige Zusatzinformationen und Hyperlinks in den Text einzufügen. Zusatzinformationen können u.a. Name und Version der geladenen Modelle sowie Position der generierten Marker sein. Über Hyperlinks können Anmerkungen im Text direkt mit einer bestimmten Kameraposition in der VRML97 basierten Szene oder statischen Bildern verknüpft werden.

Der VRML97 basierte Ansatz weisst jedoch 3 Probleme auf:

- Der Arbeitsplatzrechner des Konstrukteurs muss über ausreichende Graphikleistung verfügen
- Der benötigte Speicherplatz ist relativ hoch, insbesondere da es kein binäres VRML97 Format gibt.
- Die Geheimhaltung der Daten ist wesentlich kritischer, insbesondere, wenn das Protokoll mit Zulieferern ausgetauscht wird, die nur Teile der gesamten Szene entwickelt haben. Auf Bildern hingegen sind nur die jeweiligen Problemzonen und für das Protokoll relevanten Teile sichtbar.

Bei beiden Ansätzen muss mit dem VR-System die Möglichkeit bestehen, Marker in der Szene zu plazieren und Text einzugegeben. Weiterhin muss es für den „Text und Bild“-Ansatz auch die Möglichkeit geben, Bilder der aktuellen Szene abzuspeichern. Auf diese Funktionen wird in den folgenden Kapiteln eingegangen.

#### 6.3.1.1 Markierungen

Marker können an jeden beliebigen Punkt der Oberfläche der geladenen Modelle plaziert werden. Jeder Marker wird mit einer eindeutigen Zahl versehen, damit im Protokolltext sehr leicht ein bestimmter Marker referenziert werden kann.

Die Positionierung der Marker erfolgt sinnvollerweise über Ray Casting (siehe Kapitel 4.4.2.1). Damit kann garantiert werden, dass die Marker nicht in der Luft schweben, sondern direkt auf der Oberfläche der Modelle aufgebracht sind. So können zu jedem Marker die folgenden Informationen automatisch generiert und im Protokoll abgespeichert werden:

- Zuordnung Marker zu Teilmodell
- Position jedes Markers.

Über diese beiden Informationen ist es für den Konstrukteur wesentlich einfacher das passende CAD-Modell und die markierte Position im Modell zu finden.

Damit ein plazierter Marker nicht das Blickfeld versperrt, muss es die Möglichkeit geben, seine Orientierung festzulegen. Dies ist jedoch beim Ray Casting nicht direkt möglich. Deswegen wird der Plazierungsvorgang mehrstufig realisiert.

1. Zuerst wird über den Strahl die gewünschte Position ausgewählt und über einen Event als Position für die Markerspitze festgelegt.
2. Die Orientierung des Markers wird dann über den Vektor zwischen Position der Hand, als der Event ausgelöst wurde und aktueller Position der Hand bestimmt. Gedanklich kann der Anwender sich vorstellen, dass er in diesem Moment den Markerkopf in der Hand hält und bewegt, währen die Spitze fixiert ist.

Somit kann über einfache Translation der Hand die Ausrichtung des Markers angepasst werden. Über einen weiteren Event wird die aktuelle Orientierung des Markers beibehalten und der Plazierungsvorgang ist beendet. Werden die beiden Events auf „Taste drücken“ und „Taste loslassen“ abgebildet, so wird bei einem kurzen Druck auf die Taste der Marker an der gewünschten Stelle positioniert und eine Standardorientierung angenommen, die für die meisten Anwendungen nicht verändert werden muss.

Das System muss auch über ein Löschkommando verfügen, um falsch plazierte oder überflüssige Marker wieder aus der Szene zu entfernen. Dies kann entweder über Selektion eines Markers geschehen oder über einen Event, der die Marker nach dem LIFO (last in, first out) Verfahren löscht.

#### 6.3.1.2 Erzeugte Bilder

Ein wichtiger Bestandteil eines aussagekräftigen Protokolls sind Bilder. Da es nicht sehr sinnvoll ist, immer den gesamten Bildschirm abzuspeichern, muss der Anwender die Möglichkeit haben eine Region zu definieren, die dann als Bild gespeichert wird. Das Definieren dieser Region kann entweder in 2D oder in 3D geschehen.

**Definition der Region in 2D (Overlay)**
Hierbei wird die aktuelle 3D-Position der virtuellen Hand auf eine 2D-Koordinate im Bildschirmkoordinatensystem abgebildet. Ist die virtuelle Hand außerhalb des sichtbaren Bereiches, so liefert diese Transformation ungültige Ergebnisse, die entsprechend auf sinnvolle Werte abgebildet werden müssen. Über zwei Events werden zwei 2D-Koordinaten definiert, die die Eckpunkte eines Rechtecks bilden. Der Bildschirminhalt des Rechtecks kann dann gespeichert werden. Im Prinzip wird mit diesem Verfahren die Größe des gespeicherten Bildes festgelegt und nicht der Bildinhalt, da bei jeder Bewegung der Szene sich der Bildschirminhalt ändert, der Rahmen jedoch in konstanter Relation zum Betrachter steht.

**Definition der Region in 3D**

Bei diesem Verfahren wird die zu speichernde Region über ein 3D-Objekt der Szene definiert. Dieses Objekt besteht z.B. aus einem Rahmen und einer Kugel, die über 4 Linien mit den Eckpunkten des Rahmens verbunden ist.

Die Kugel repräsentiert hierbei die Kameraposition und die Relation zwischen Rahmen und Kugel definiert die weiteren Kameraparameter, wie den Öffnungswinkel und das Höhen-/Breitenverhältnis. Um diese Parameter auch interaktiv verändern zu können, muss der Anwender die relative Position zwischen Kugel und Rahmen ändern können. Dieses „Kameraobjekt“ kann nun beliebig in der Szene positioniert werden und über einen Event wird daraufhin ein Bild basierend auf den durch dieses Kameraobjekt definierten Parametern erzeugt. Das Objekt sollte cart-relativ positioniert werden.

#### 6.3.1.3 Textbeschreibung

Die Alternativen der Eingabe von Text sind, wie bereits in Kapitel 4.7 aufgezeigt, für dieses Anwendungsgebiet sehr beschränkt. Aufgrund der praktischen Erfahrungen sollte der Text über eine Tastatur direkt in ein Editorfenster des Design Review Systems eingegeben werden. Da der Bildschirm des Graphikrechners meist vollständig durch das VR-System belegt ist, bietet sich hier ein über ein Netzwerk an den Graphikrechner angebundener Zweitrechner an.

### 6.3.2 CAD basierte Dokumentation

Die CAD basierte Dokumentation unterscheidet sich von der XML basierten Dokumentation dadurch, dass die Markierungen nicht getrennt von den CAD-Daten vorliegen, sondern das diese in den CAD Daten gespeichert werden.

Der Text muss jedoch weiterhin getrennt vorliegen, da heutige CAD-Systeme (z.B. CATIA) keine adäquate Möglichkeit bieten, in einem CAD-Datensatz den Protokolltext zu speichern. Es gibt zwar alternative Dateiformate, wie z.B. STEP, diese müssten jedoch die „alten“ Formate flächendeckend ersetzen und vollständig in den CAD-Prozess integriert sein. Da es den Rahmen dieser Arbeit sprengen würde, wird diese Thematik hier nicht weiterverfolgt.

### 6.3.3 Unterstützung bei Wiedervorlage

Das erstellte Protokoll ist sicherlich in erster Linie für den Konstrukteur interessant, da er das CAD-Modell entsprechend den Anmerkungen anpasst und erweitert. Darüberhinaus wird es aber auch bei der Wiedervorlage herangezogen, um zu prüfen, ob alle festgestellten Mängel und Problempunkte abgearbeitet worden sind. Mit einer XML basierten Dokumentation muss nun jeder Problempunkt zuerst im Modell gesucht und anschließend die Szene so positioniert werden, dass dieser gut zu erkennen ist. Evtl. muss auch die Schnittebene entsprechend konfiguriert werden oder diverse Objekte abgeschaltet werden.

Da dies natürlich sehr aufwändig ist und bereits bei der Erstellung des Protokolls die Szene entsprechend positioniert und konfiguriert war, muss es die Möglichkeit geben den aktuellen Zustand für jeden Problempunkt mit abspeichern zu können. Am Ende eines Reviews wird dann nicht nur ein Protokoll erzeugt, sondern auch eine Metadatei, die für jeden Problempunkt die Szenenkonfiguration mit abspeichert. Bei Wiedervorlage wird nun das neue Modell und diese Metadatei eingeladen und die einzelnen Konfigurationen müssen anschließend nur noch entsprechend aktiviert werden. Da es sich

hierbei in den allermeisten Fällen um die selben Bauteile handelt, ist die Konfiguration auch für die neue Szene gültig. Siehe hierzu auch Abb. 6.17.

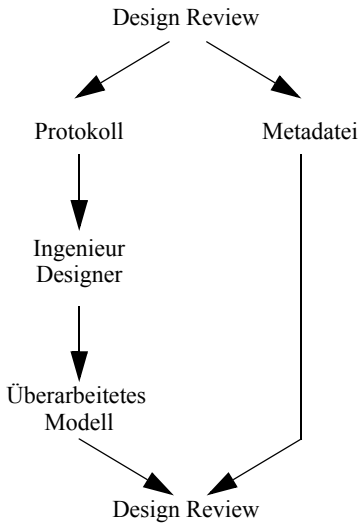


Abb. 6.17: Konzept für den vereinfachten Design Review einer Wiedervorlage

Zu jedem Problempunkt werden die folgenden Informationen in der Metadatei gespeichert:

- Bauteilreferenzen
- Sichtbarkeit der Objekte
- Position und Lage der Objekte
- Position und Lage der Kamera
- Status, Position und Lage der Schnittebene
- Markierungen
- Textbeschreibung
- Erzeugte Bilder

In der Metadatei werden nur Referenzen und bewusst keine Bauteilgeometrien abgespeichert, da diese zu einem späteren Zeitpunkt dazugeladen werden. In welcher Form diese Referenzen abgespeichert werden, hängt davon ab, wie im CAD-Prozess Bauteile identifiziert werden.

## 6.4 Integration von Offline-Simulationen

Es hat sich in der Praxis gezeigt, dass nicht nur die rein geometrische Betrachtung wichtig ist, sondern auch die Einbindung zusätzlicher Daten, die aus Offline-Simulationen gewonnen werden, eine immer zentralere Rolle spielt. Als Beispiel sei hier die Lacktrocknungssimulation angeführt. Nach dem Aufbringen des Lacks wird die Karosserie möglichst gleichmäßig auf eine bestimmte Temperatur erhitzt und im Anschluß abgekühlt, sodass der Lack trocknen kann. Bei diesem Prozess muss insbesondere auf die gleichmäßige Trocknung, wie auch auf die maximal erreichte Temperatur geachtet werden. Entsprechend den Ergebnissen der Simulation muss die Positionierung der Heizlüfter entsprechend angepasst werden. Andere Anwendungsfälle neben der Lacktrocknung sind u.a. Crash-, Blechpress,- und Strömungssimulationen. Da die Berechnungszeiten dieser Simulationen im Stundenbereich lie-

gen, spricht man auch von Offline-Simulationen. Das bedeutet auch, dass bei Veränderung der Randbedingungen, die gesamte Simulation neu berechnet werden muss.

Da die Simulationsergebnisse die Konstruktion wesentlich beeinflussen, können sie als ein weiteres Werkzeug gesehen werden, das während eines Design Reviews eingesetzt wird. Traditionell ist es jedoch so, dass jedes Simulationsprogramm seine Ergebnisse in einem proprietären Format ablegt und eine eigene Visualisierungskomponente besitzt, um die simulierten Daten zu betrachten. Die Bedienung dieser Visualisierungskomponenten sind je nach System unterschiedlich und die eigentlichen Funktionalitäten eher rudimentär. Es existieren zwar auch interaktive Visualisierungssysteme, die mehrere Simulationsdatenformate lesen und visualisieren können (z.B. Animator3 [GNS00]), jedoch ist deren Bedienung meist sehr umständlich. In Hinblick auf die Anforderungen, die für den Design Review bzgl. der Bedienung gestellt wurden (siehe Kapitel 2.3.2), ist es sinnvoll, für alle Aufgaben im Rahmen des Design Reviews das selbe Softwaresystem einzusetzen. Dies hat insbesondere auch den Vorteil, dass die Anwender nur ein System kennen müssen.

Eine sehr wichtige Funktion, die ein solches System unterstützen muss, ist die Möglichkeit das Mapping der Simulationsdaten auf die geometrischen Objekte interaktiv anzupassen, um optimale Ergebnisse bei der Visualisierung zu erzielen. Das Mapping ist Teil der Visualisierungspipeline (siehe Abb. 6.18), die sich grob aus 3 Transformationen zusammensetzt ([UPSO89]):

- Filtering: Hier werden die Simulationsdaten in eine Form umgewandelt, die für die weitere Verarbeitung in der Pipeline zweckmäßiger ist.
- Mapping: In diesem Prozess werden die aufbereiteten Daten auf graphische Objekte und Attribute abgebildet.
- Rendering: Hier werden aus den im vorherigen Schritt aufbereiteten graphischen Objekten und Attributen Bilder generiert, die dem Benutzer präsentiert werden.

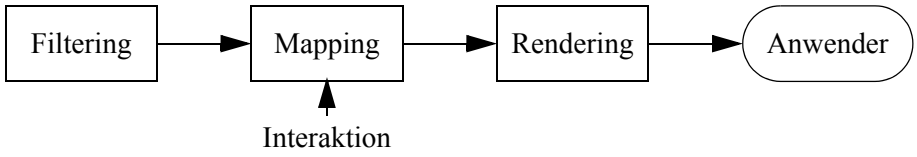


Abb. 6.18: Die Visualisierungspipeline

Auf das Filtering und Rendering wird hier nicht explizit eingegangen, da dies Schritte sind, die nicht direkt mit der Benutzerschnittstelle zusammenhängen.

Neben dem Mapping muss es auch möglich sein, die Simulationsdaten auch quantitativ zu untersuchen. Dies kann durch das Plazieren von Datensonden erreicht werden, die in Kapitel 6.4.3 beschrieben werden.

Da die Simulationsdaten in den beschriebenen Szenarien sich über die Zeit ändern, muss es weiterhin die Möglichkeit geben, zwischen den verschiedenen Zeitpunkten umzuschalten und diese zu visualisieren. Dies wird in Kapitel 6.4.4 näher erläutert.

### 6.4.1 Gruppierung und Sichtbarkeit

Da die simulierten Datensätze in den meisten Fällen sehr komplex sind, muss das System auch die Möglichkeit bieten, Teile der visualisierten Szene auszublenden. Dies setzt jedoch voraus, dass die Simulationsdaten eine logische Gruppierung aufweisen.

Die Selektion auszublendender Objekte sollte über das Ray Casting realisiert werden, da diese Selektionstechnik bereits für die meisten anderen Funktionalitäten des Systems ebenfalls verwendet wird. Hierbei wird im Moment der Selektion das Objekt sofort ausgeblendet und nicht erst in eine Selektionsliste aufgenommen, auf die dann die Funktion „Ausblenden“ angewendet wird. Da in einem Kraftfahrzeug viele Objekte sich verdecken, wäre ein solches Vorgehen sehr umständlich.

Das wieder sichtbar Schalten der Objekte kann natürlich nicht mehr über Ray Casting geschehen. Hier gibt es verschiedene Methoden, die angewendet werden können:

- Die unsichtbaren Objekte werden in einer Liste gehalten, die über das Menüsystem dargestellt werden kann. Wird ein Objekt aus dieser Liste ausgewählt, so wird es wieder sichtbar geschaltet. Bei sehr vielen Objekten wird diese Methode jedoch sehr schnell unübersichtlich und ist auch nur bei entsprechender Benennung der Objekte effektiv. Diese ist jedoch in den seltensten Fällen gegeben.
- Die ausgeblendeten Objekte werden über ein Kommando in umgekehrter Reihenfolge wieder eingeblendet. Dies kann u.a. folgendermaßen realisiert werden: Wird das unsichtbar Schalten über Ray Casting realisiert, so wird das vom virtuellen Strahl getroffene Objekt über einen kurzen Druck auf eine Taste des Interaktionsgerätes unsichtbar geschaltet. Über einen lagen Druck auf die Taste werden die Objekte wieder sichtbar geschaltet. Damit ist das Schalten der Sichtbarkeit für den Anwender sehr einfach, da er insbesondere Falschauswahlen sofort korrigieren kann. Möchte er jedoch ein bestimmtes Objekt wieder sichtbar schalten, ist diese Methode weniger geeignet.
- Sollen bestimmte Objekte wieder sichtbar geschaltet werden und die Selektion über eine Auswahlliste erweist sich als unpraktikabel, so sollte die Möglichkeit bestehen, zwischen der Menge der sichtbaren Objekte und der Menge der unsichtbaren Objekte umzuschalten. So ist es dann sehr leicht möglich auch Objekte wieder der Menge der sichtbaren Objekte hinzuzufügen.

### 6.4.2 Mapping

Um die simulierten Daten zu visualisieren, müssen diese zuerst über einen Mappingprozess auf geometrische Objekte und Attribute abgebildet werden. Der Mappingprozess ist der zentrale Verarbeitungsschritt in der Visualisierungspipeline. Die Abbildung erfolgt mit einer Transferfunktion $f_T$.

$$f_T(\text{Daten}) = (\text{Objekt},\text{Attribut})$$

Die Komplexität der Transferfunktion variiert sehr. Meistens werden Geometrie und Zeit bis auf eine Skalierung unverändert übernommen. Lineare Funktionen werden eingesetzt, um quantitative Informationen beizubehalten, während nicht-lineare Funktionen die Feinheiten in den Daten hervorheben können. Es gibt keine Standardverfahren für die Auswahl der richtigen Transferfunktion. Es ist daher von höchster Wichtigkeit, dass der Ingenieur die Transferfunkton interaktiv ändern kann, um die für ihn wichtigen Informationen optimal präsentiert zu bekommen.

In den hier vorgestellten Szenarien werden für jeden Knoten des Simulationsdatensatzes skalare Werte berechnet (z.B. Temperatur). Farbe ist die meistverwendete und oft die beste Repräsentation ([TUFT83]) von skalaren Variablen. Das ist dadurch zu erklären, dass ein Teil des Farbenspektrums für das Auge eine kontinuierliche, geordnete Skala bildet. Farbe eignet sich jedoch nicht zur Repräsentation höherwertiger Simulationsdaten [KARL94].

Für die angesprochenen Szenarien muss es somit möglich sein, die Transferfunktion, d.h. die Abbildung der Simulationswerte auf Farbwerte interaktiv zu ändern. In den beschriebenen Szenarien ist es insbesondere sehr wichtig, dass das gesamte Farbenspektrum auch auf sehr kleine Wertebereiche abgebildet werden kann, um Unterschiede in den Simulationswerten, die in diesem Bereich liegen, gut erkennen zu können.

Das Ändern des Farbmappings wird über eine interaktive Säule (siehe Kapitel 4.6.3.1) realisiert. Über zwei Zeigern können zwei skalare Werte $s_1$ und $s_2$ in einem Bereich $[s_{min};s_{max}]$ definiert werden. Es gilt $s_1 < s_2$. Das Farbenspektrum mit dem Wertebereich $[c_{min};c_{max}]$ wird dabei linear auf den Wertebereich $[s_1;s_2]$ abgebildet. Werte zwischen $[s_{min};s_1[$ werden auf das untere Ende des Farbenspektrums und Werte zwischen $]s_2;s_{max}]$ auf das obere Ende des Farbenspektrums abgebildet. Die Abbildungsfunktion sieht wiefolgt aus:

$$c = f(s) = \begin{cases} c_{min} & s < s_1 \\ c_{min} + \dfrac{s - s_1}{s_2 - s_1}(c_{max} - c_{min}) & s_1 \le s \le s_2 \\ c_{max} & s > s_2 \end{cases} \qquad (13)$$

Die aktuelle Abbildung der Simulationswerte auf die Farbwerte wird direkt auf dem Säulenkörper angezeigt. Durch Bewegen der beiden Zeiger kann nun das Farbmapping angepasst werden. In Abb. 6.19 ist dies sehr gut zu erkennen.

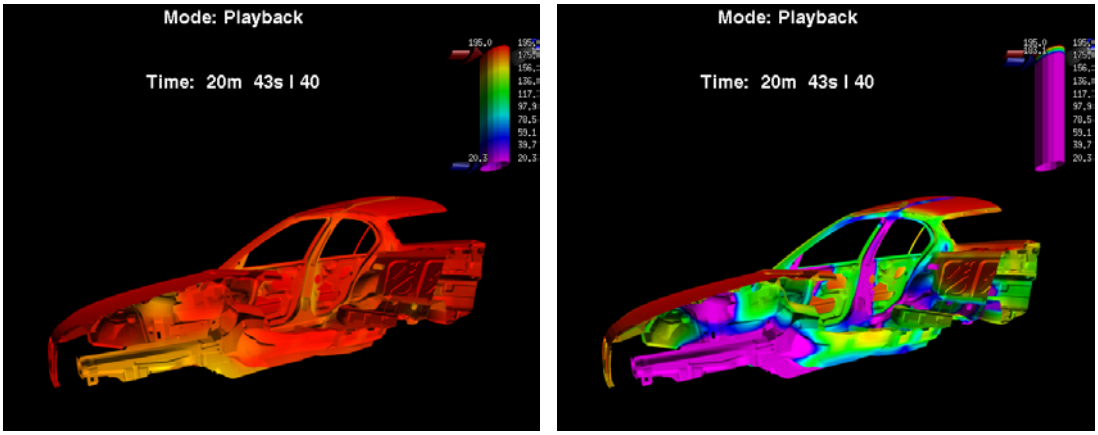


Abb. 6.19: Darstellung der Simulationsdaten mit unterschiedlichem Farbmapping

Es muss hierbei gewährleistet sein, dass jegliche Veränderung des Farbmappings sich sofort auf den visualisierten Simulationsdatensatz auswirkt. Wird z.B. OpenGL auf einer Graphikworkstation mit Hardwaretexturen verwendet, so kann dies durch Anpassung der Texturmatrix und Verwendung von Texturkoordinaten, die eine 1D-Farbtextur referenzieren, erreicht werden.

In der Praxis hat sich gezeigt, dass es sehr sinnvoll sein kann, Bereiche mit „uninteressanten“ Simulationswerten auszublenden. Hierzu wird die interaktive Säule durch einen dritten Zeiger und einen Wahlschalter erweitert. Je nach Stellung des Wahlschalters werden alle Simulationswerte über oder unter dem Wert, den der Zeiger angibt, nicht dargestellt. Eine Alternative wäre auch die Darstellung als Punktewolke, um die sichtbaren Simulationsdaten im Gesamtkontext zu sehen (Abb. 6.20).

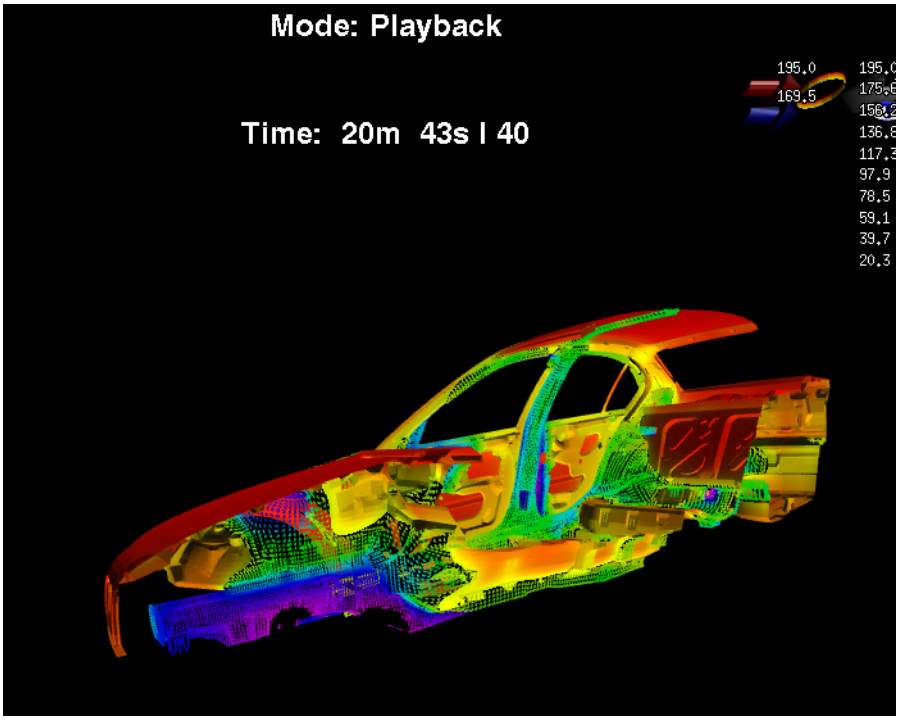


Abb. 6.20: Kombinierte Darstellung mit Punktewolke

### 6.4.3 Datensonde

Eine der wichtigsten Funktionalitäten für die Evaluierung der Simulationsdaten ist die Möglichkeit, eine Datensonde zu plazieren, d.h. die Simulationswerte an frei definierbaren Position abzufragen. Für 2D-Daten im 3D-Raum sind hier nur Punkte auf der Oberfläche des Datensatzes zulässig, bei 3D-Daten im 3D-Raum kann die Probe an jeden beliebigen Punkt im Raum gesetzt werden.

Für das Plazieren der Datensonde auf einer Oberfläche bietet sich das Ray Casting an, da hiermit sowohl die Position relativ genau angegeben werden kann, als auch sichergestellt ist, dass nur Punkte auf der Oberfläche des Datensatzes selektiert werden. Bei der Ausgabe des Datenwertes muss es möglich sein, diesen kontinuierlich ausgeben zu lassen, d.h solange der Anwender den Strahl über das Modell bewegt, wird der ausgegebene Wert aktualisiert. Der Wert sollte sehr nahe dem Schnittpunkt des Strahls mit der Oberfläche ausgegeben werden und nicht an einer definierten Position auf dem Bildschirm. Das erleichtert dem Anwender die Bedienung, da er zum Lesen des Wertes den

Blick nicht vom virtuellen Laserstrahl abwenden muss. Wie bereits in Kapitel 3.3.1 erwähnt, kann ein um 30° von der Fovea abweichendes Objekt nicht mehr allein mit einer Sakkade, sondern nur mit einer zusätzliche Kopfbewegung fixiert werden.

Da die Datensonde bei 3D Daten im 3D Raum an jeder beliebige Position plaziert werden kann, ist hier die Verwendung von Ray Casting nicht sinnvoll. Hier bietet es sich an, die Datensonde durch ein spezielles Objekt zu repräsentieren und diese dann analog der in Kapitel 6.1.2 beschriebenen Schnittebene zu positionieren.

In beiden Fällen ist jedoch auch die Wertänderung an einer bestimmten Position über die Zeit ein wichtiges Kriterium. Dieser Verlauf der Wertänderung sollte als Kurve auf dem Bildschirm ausgegeben werden. Damit kann z.B. in der Lacktrocknung sehr leicht untersucht werden, ob die Karosserie linear aufgeheizt wurde.

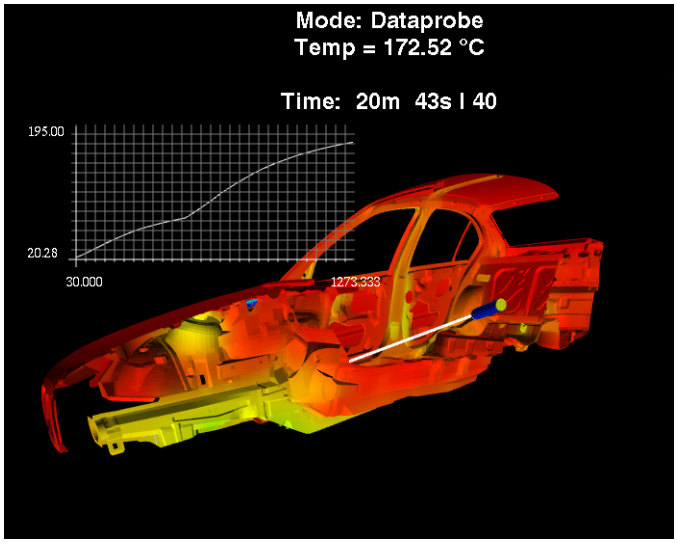


Abb. 6.21: Datenprobe mit Darstellung der zeitlichen Veränderung

### 6.4.4 Animationssteuerung

Die meisten Simulationen werden nicht nur für einen festen Zeitpunkt berechnet, sondern für eine ganze Folge von Zeitpunkten. So werden z.B. für Crash-Simulationen typischerweise 100 Einzelschritte simuliert. Die Zeitspanne, über die sich diese Simulationszeitpunkte erstrecken, ist abhängig von der jeweiligen Applikation.

Das System muss somit eine Möglichkeit bieten, zwischen den einzelnen Simulationszeitpunkten umschalten zu können, sowohl diskret, als auch als Animation, d.h. zwischen den einzelnen Zeitschritten wird kontinuierlich umgeschaltet, wobei die Abspielgeschwindigkeit auch interaktiv geändert werden können soll.

Zur Steuerung der Animation bietet sich der in Kapitel 4.6.3.3 vorgestellte Jog-Dial an, denn auch in der realen Welt wird er zu Steuerung von Animationen eingesetzt.

Für die Animationssteuerung bietet es sich an, den Wertebereich von [-v;+v] nicht linear auf den

Halbkreis des Jog-Dials zu mappen, sondern diesen in Zonen einzuteilen, die symmetrisch zur Nullstellung sind. In der Praxis hat sich eine Einteilung des Jog-Dial in sieben Sektoren als sinnvoll erwiesen (siehe hierzu Abb. 4.42).

Befindet sich der Zeiger in der mittleren Zone, so liefert der Jog-Dial [0] zurück, d.h. die Animation bleibt stehen. Dadurch das der Zeiger bei jeder Aktivierung in die Ursprungsposition zurückspringt, kann eine laufende Animation sehr einfach gestoppt werden, indem der Jog-Dial nur sehr kurz aktiviert wird und somit auch der Zeiger nicht mehr aus der Grundstellung gedreht werden kann. Die nächsten Zonen (im Bild gelb) liefern einen festen Wert $-v_{fix}$, bzw $+v_{fix}$ und die äußeren Bereiche einen linear interpolierten Wert in einem definierten Wertebereich $[-v_{max};-v_{fix}[$, bzw. $]+v_{fix};+v_{max}]$. So kann ein für die Anwendung sinnvoller Abspielwert $v_{fix}$ sehr einfach eingestellt werden, als auch auf einfache Art und Weise die Abspielgeschwindigkeit wesentlich erhöht werden.

## 6.5 Zusammenfassung

In diesem Kapitel wurde der VR-Design Review vor allem von seiner funktionalen Seite betrachtet, wobei drei Funktionsgruppen identifiziert wurden:

- Funktionen, die das Untersuchen des Modells unterstützen
- Funktionen, die die Dokumentation unterstützen
- Funktionen, die das Manipulieren von Offline-Simulationen erlauben.

Die für jede Gruppe wichtigsten Funktionalitäten wurden beschrieben. Hierbei lag der Fokus vor allem auch auf der Art und Weise, wie diese Funktionen über die Benutzerschnittstelle abgebildet und welche Interaktionstechniken eingesetzt werden sollten. Sie können somit auch als „testbed“ für die in Kapitel 4 entwickelten Techniken angesehen werden. Dabei wurde gezeigt, dass alle Funktionen über das Konzept der logischen Interaktionsaufgaben und der Interaktionstechniken aus Kapitel 4 realisiert werden können.

Da die Qualität der Interaktion stark mit der Latenz korreliert, wurde hier ein Ansatz entwickelt, der es während der Phase des Interagierens ermöglicht, die Framerate zu erhöhen und damit die Latenz zu senken (adaptive Darstellung). Ein spezieller Algorithmus erzeugt hierzu eine einfachere Repräsentation der Objekte, mit denen interagiert wird (Pseudo-Umrissdarstellung). Während der Interaktion wird dann diese Repräsentation verwendet. Zusätzlich wird durch Texture- und Z-Replacement das Rendering der Restszene erheblich beschleunigt. Dieser Ansatz kann auch beim Papier und Bleistift-Paradigma sinnvoll eingesetzt werden, um auch bei vielen Sketch-Objekten eine akzeptable Framerate zu erreichen.

# 7 Basisarchitektur für logische Interaktionsaufgaben

In den vorherigen Kapiteln wurde der Design Review Prozess allgemein, als auch die verschiedenen Applikationsszenarien vorgestellt. Es wurden mit Hilfe des 4W-Modells unterschiedliche Anforderungen herausgearbeitet und von der eigentlichen Anwendung abstrahiert, um einen allgemeingültigeren Lösungsansatz verfolgen zu können (siehe u.a. Kapitel 2.5).

Ein zentraler Punkt im Design Review ist die Interaktion. In dieser Arbeit wurde das Konzept der *basic interaction tasks* (BIT) als Basis für den Entwurf einer intuitiven Benutzerschnittstelle gewählt und verschiedene Interaktionstechniken vorgestellt und evaluiert (siehe Kapitel 4). Weiterhin wurden Konzepte vorgestellt, die die Kommunikation zwischen den Teilnehmern unterstützen (siehe Kapitel 5) und Funktionen beschrieben, die für das Arbeiten immanent wichtig sind (Kapitel 6). Dort wurde insbesondere auf den Grundlagen aus Kapitel 4 aufgebaut und gezeigt, wie die vorgestellten Interaktionstechniken hier eingesetzt werden können.

Es stellt sich nun die Aufgabe, diese Konzepte, Interaktionstechniken und Funktionen in Einklang zu bringen. Zwar kann mit den bis hier vorgestellten Ansätzen bereits eine intuitive Benutzerschnittstelle realisiert werden, eine Strukturierung dieses Vorgehens wäre jedoch insbesondere hinsichtlich der Flexibilität des Systems sehr sinnvoll.

In diesem Kapitel wird nun eine Architektur vorgestellt, die ein solches strukturiertes Vorgehen ermöglicht. Ziel hierbei ist es jedoch nicht, eine Architektur für ein komplett neues VR-System zu entwerfen, sondern auf Basis standardisierter VR-Technologien Erweiterungen zu spezifizieren, die den Entwurf interaktiver und intuitiver Benutzerschnittstellen ermöglicht. Im Rahmen dieser Arbeit wird hierbei auf X3D/VRML97 aufgesetzt. Die dort beschriebenen Basiskonzepte (fields and routes) finden sich nicht nur in VRML-Browsern, sondern auch in den verschiedensten, heute eingesetzten VR-Systemen wieder. Zwar sind die Konzepte dort teilweise leicht adaptiert, für einen Experten sollte aber das anpassen der hier vorgestellten Architektur ein Leichtes sein.

Die weiteren Ziele, die mit dieser Architektur verfolgt werden sollen, sind vielfältig:

- Darlegung einer Vorgehensweise zur Umsetzung des abstrakten BIT-Konzeptes auf Basis der bekannten Softwaretechnologie X3D/VRML97.
- Die Architektur soll einen möglichst einfache Realisierung von Interaktionstechniken erlauben, die auf mehreren BITs aufbauen (z.B. die *interaktive Säule* aus Kapitel 4.6.3.1).
- Die Abhängigkeit der Interaktionstechniken von der verwendeten Hardware soll für Anwendungen transparent sein. Das bedeutet, dass bei einem Wechsel von Ein-/Ausgabegeräten und der damit verbundenen Anpassung der Interaktionstechniken seitens der Anwendung keine Änderungen vorgenommen werden müssen.
- Es werden Komponenten beschrieben, die essenziell für die Umsetzung eines Bedienkonzeptes sind.

Die folgenden Kapitel gliedern sich nun wie folgt: In Kapitel 7.1 wird kurz auf X3D/VRML97 eingegangen und die Basiskonzepte vorgestellt. Im Anschluss werden weitere Strukturen vorgestellt, die für die Umsetzung der Konzepte notwendig sind. In Kapitel 7.3 wird auf Basis des *fields and routes* Konzept, eine abstrakte Schnittstelle für die verschiedenen BITs definiert. In Kapitel 7.4 wird dann

gezeigt, wie diese abstrakten BIT-Knoten in konkrete Interaktionstechniken überführt werden können. Weiterhin wird das Konzept des BIT-Managers vorgestellt und die Initialisierung/Instanziierung von komplexen CITs beschrieben. An einem Beispiel werden die Vorteile der Abstraktion aufgezeigt.

## 7.1 Basistechnologie VRML97/X3D

VRML steht für Virtual Reality Modelling Language, wobei diese Bezeichnung etwas verwirrend sein könnte, denn VRML ist weder Virtuelle Realität (VR) noch eine Sprache zur Modellierung. Wie bereits dargestellt ist eine der wichtigen Komponenten von VR die Immersion, die durch Verwendung spezieller Ein- und Ausgabegeräte erreicht wird. VRML schließt Immersion weder ein noch aus, jedoch ist die aktuelle Spezifikation vollständig auf Monitor, Tastatur und Maus ausgerichtet; Geräte die nicht VR-typisch sind. Um eine Modellierungssprache zu sein, müsste VRML wesentlich mehr geometrische Primitive und Modellierungsfunktionalitäten besitzen. Diese sind in VRML auf das absolut notwendigste reduziert.

Was ist VRML nun wirklich? Auf diese Frage gibt es verschiedene Antworten. Im Kern ist VRML ein einfaches 3D Austauschformat. Es definiert die heute üblichen Semantiken, wie hierarchische Transformationen, Lichtquellen, Viewpoints, Geometrie, Animation, Nebel, Materialeigenschaften und Texturmapping. Auf diesem Kern basierte auch das eigentliche Ziel beim Entwurf von VRML: Ein effektives Dateiformat für 3D-Szenen zu schaffen.

Die zweite Antwort ist, dass VRML die 3D-Analogie zu HTML ist. Das bedeutet, dass VRML als eine einfache, multiplattformfähige Sprache zum Publizieren von 3D Webseiten gedacht ist. Die Verwendung von 3D auch auf Webseiten wird insbesondere dadurch motiviert, dass bestimmte Informationen in 3D am Besten wahrgenommen werden können, z.B. Spiele, wissenschaftlich-technische Visualisierung, Lernwelten und Architektur. Diese Applikationstypen verlangen typischerweise intensive Interaktion mit dem System, Animation und die Möglichkeit für den Anwender Inhalte zu erforschen. Diese Anforderungen können mit herkömmlichen seiten-, text- oder bildbasierten Formaten (z.B. HTML) nur sehr schlecht erfüllt werden.

Die dritte Antwort ist, dass VRML eine Technologie darstellt, die sowohl 3D, 2D, als auch Text und Multimedia in einem stimmigen Konzept integriert. Werden diese Medien mit Skriptsprachen und Internet kombiniert, so stehen Tür und Tor für ein neues Genre interaktiver Anwendungen offen.

Die vierte Antwort ist die wohl am meisten publizierteste und am heftigsten debattierte: VRML ist die Basis für Cyberspace und on-line virtual communities, wie sie z.B. in dem Buch Neuromancer des Science-fiction Autors William Gibson beschrieben wurden. Kritiker haben jedoch zu Recht darauf hingewiesen, dass VRML über keinerlei Datenbank- und Netzwerk-Fähigkeit verfügt, die für Mehrbenutzeranwendungen notwendig sind.

Die Antwort auf "Was ist VRML?" umfaßt somit eigentlich alle vier Antworten. Jedoch gibt es auch einige falsche Antworten: VRML ist keine Programmierbibliothek für Anwendungsentwickler, auch wenn die Wurzeln von VRML bei Open Inventor liegen, welches ein umfangreiches API zur 3D-Programmierung zur Verfügung stellt. Faktisch ist VRML eine erweiterte Untermenge von Open Inventor.

### 7.1.1 Basiskonzepte

Das Basiskonzept von VRML 97 baut im Prinzip auf den folgenden Komponenten auf:

- Knoten
- Felder
- Events
- Routes

Knoten sind die eigentlichen Objekte einer virtuellen Szene. Die Daten der Objekte sind in Feldern abgelegt. Um Daten zwischen Knoten austauschen zu können, (z.B. eine virtuelle Taste drücken und eine Lichtquelle wird geschaltet) können Felder über Routes verbunden werden. Siehe die folgende Abbildung.

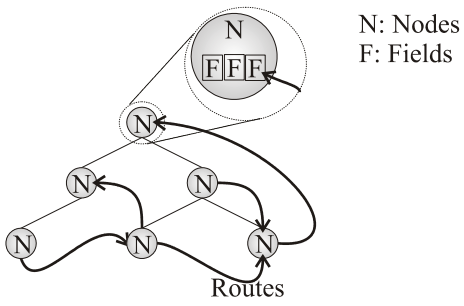


Abb. 7.1: Die einzelnen Basiskomponenten von VRML97

Eine weitere Kommunikationsmöglichkeit stellen Events dar. Im folgenden werden diese Komponenten näher vorgestellt.

#### 7.1.1.1 Knoten

Knoten (Nodes) repräsentieren in VRML die Objekte und das Verhalten einer virtuellen Umgebung, entsprechend gibt es für verschiedene Objekte auch unterschiedliche Typen von Knoten. Diese Knoten sind in einem hierarchischen Szenengraph angeordnet. In der Spezifikation werden 55 unterschiedliche Knoten beschrieben, die sich in verschiedene Kategorien gliedern lassen, u.a. Geometrie, Aussehen und Sensoren. Die Gruppe der Sensoren definiert die Interaktionen des Anwenders mit dem System und wird deswegen in Kapitel 7.3 eingehender beleuchtet.

#### 7.1.1.2 Felder

In Feldern werden Daten gleichen Typs gespeichert, wobei im Standard definiert ist, welche Datentypen unterstützt werden. In einem Feld können entweder ein einziger Wert (*Ein-Wert-Feld*, *single value field*) oder mehrere Werte (*Mehr-Wert-Feld, multiple value field*) des gleichen Typs abgelegt werden. Felder sind immer einem Objekt (*Knoten, Node*) zugeordnet.

Felder unterscheiden sich jedoch nicht nur durch die Anzahl der zu speichernden Werte und den Wertetypen, sondern auch, ob in diese Felder von außen geschrieben werden kann oder nicht (ähnlich public/private bei C++). Die VRML97 Spezifikation stellt folgende Mechanismen zur Verfügung:

- *field*: In diese Felder kann nur der Knoten selber schreiben und lesen
- *exposedField*: In diese Felder kann sowohl von außen gelesen, als auch geschrieben werden und funktionieren somit ähnlich wie eventIn und eventOut (s.u.)

#### 7.1.1.3 Events

Events sind Methoden eines Knotens, die als Parameter ein Feld (s.o) verlangen. Es kann zwischen zwei Typen von Events unterschieden werden: *eventIn* und *eventOut*. *eventIn* ist dazu geeigneten Daten von außen zu empfangen. Diese können dann in der Methode verarbeitet und dann z.B. in einem *field* abgelegt werden. Somit sind *eventIns* den *exposedField*s recht ähnlich: Bei exposedFields werden die Daten direkt in einem Feld des Knotens abgelegt, bei eventIns wird dies über die Programmlogik der eventIn-Methode gesteuert. Über *eventOuts* können über Routes Daten an andere Knoten versendet werden.

#### 7.1.1.4 Routes

Die Verbindung zwischen einem Node, der einen Event generiert und einem Node, der ihn empfängt, wird als *route* bezeichnet. Diese Verbindungen sind unidirektional und es können auch nur Felder/ Events gleichen Typs miteinander verbunden werden.

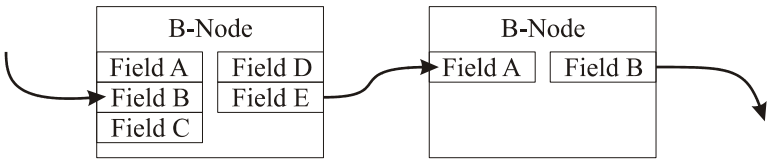


Abb. 7.2: Routes verbinden Felder / Events

Schickt ein Node einen eventOut aufgrund eines empfangenen eventIn ab, so können Eventkaskaden entstehen. Diese Events enthalten alle den gleichen Zeitstempel, um möglicherweise entstehende Schleifen (*loops*) abbrechen zu können.

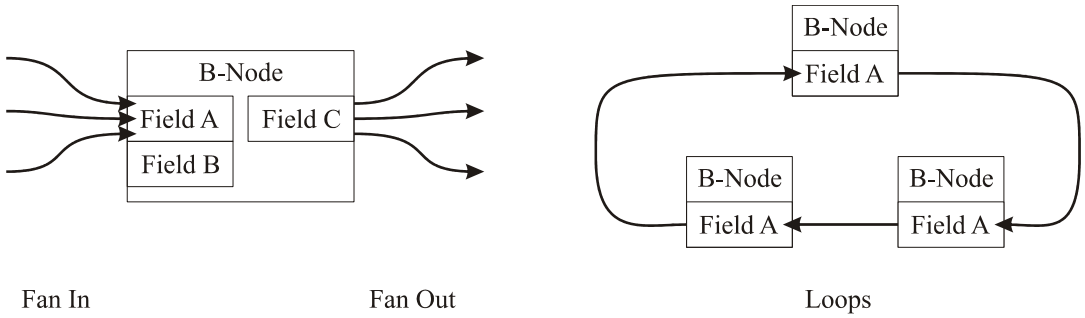


Abb. 7.3: Unterstützung von Fans und Loops

### 7.1.2 Erweiterbarkeit

Die Erweiterbarkeit von VRML wird durch das sogenannte Prototyping ermöglicht. Es erlaubt eine

einfache Definition von neuen Knoten. Diese Definition kann einen einzelnen Knoten mit speziellen Feldern enthalten oder sogar mehrere Knoten, Prototypen und Routes umfassen. Die eigentliche Logik eines solchen neuen Knotens kann z.B. durch ein Script realisiert werden.

### 7.1.3 X3D

In den letzten Jahren hat sich insbesondere im Internet und WWW eine Vielzahl von unterschiedlichsten Formaten etabliert, so dass es heute schwer ist, Daten austauschen und interpretieren zu können. Aus diesem Grund wird nun sehr stark die Verwendung XML propagiert, welches Mechanismen zur Verfügung stellt, um den Wildwuchs an Formaten in den Griff zu bekommen. Dies bedeutet jedoch für viele bestehende Standards, so auch HTML und VRML, dass Änderungen an der Spezifikation durchgeführt werden müssen. Das XML-kompatible Proposal zu VRML97 ist X3D.

Im Prinzip ist X3D VRML97 sehr ähnlich und von funktionaler Seite wurden hier nur sehr wenige Änderungen eingeführt. Neben dem Streamlining für XML hat man sich jedoch einer Problematik von VRML angenommen, die sich erst bei der Entwicklung der VRML-Browser gezeigt hat: Das gesamte Funktionsset von VRML zu implementieren ist nicht ganz einfach. Insbesondere gibt es Funktionen wie Volume-Rendering und 3D-Audio, die sehr komplex und aufwendig sind.

Um Entwicklern nun die Implementierung zu vereinfachen, aber nicht Funktionalitäten aus dem Standard zu entfernen, wurden sog. Profiles entworfen. Profiles fassen bestimmte Funktionalitäten zusammen, beginnend bei einigen Basisfunktionalitäten bis hin zum kompletten Funktionsset. So muss ein Entwicklerteam nicht von Anfang an alles implementieren, sondern kann sich auf ein Profile konzentrieren.

## 7.2 Weitere Basisstrukturen

Wie im vorherigen Kapitel dargestellt, definiert VRML97/X3D ausschließlich die Basiselemente, mit denen interaktive, virtuelle 3D-Welten aufgebaut werden können. Zusätzlich werden einige Grundfunktionalitäten in Form von Knoten bereitgestellt.

Im Verlauf dieser Arbeit hat sich jedoch herausgestellt, dass es gerade für ein VR-Rahmensystem sinnvoll ist, einige weitere Strukturen fest vorzugeben, die zum einen dem Entwickler „Vorgaben“ machen, denen er sich unterordnen muss (stringentere Entwicklung), ihm aber dadurch auch die Entwicklung erleichtern können. Aufgrund des „domain-independant“-Anspruchs von VRML97/X3D sind diese Erweiterungen für den Standard sicherlich nicht geeignet, in der hier verfolgten Ausrichtung jedoch sehr hilfreich.

So hat sich als sinnvoll herausgestellt, bereits beim Startup des Systems einen Basisszenengraphen vorzugeben, dessen Knoten definierte Funktionen übernehmen. Dieser Graph ermöglicht es dann z.B. sehr einfach die Hilfsobjekte der Interaktionstechniken (s.u.) entsprechend zu positionieren (siehe Kapitel 4.2.2). Da einige der Interaktionstechniken auf Interaktionen definierter „Körperteile“ des Anwenders aufbauen, bietet es sich auch an, eine entsprechende Abbildung relevanter „Körperteile“ auf Knoten im Szenengraph anzubieten.

Im folgenden Schaubild ist der Basisszenengraph illustriert:

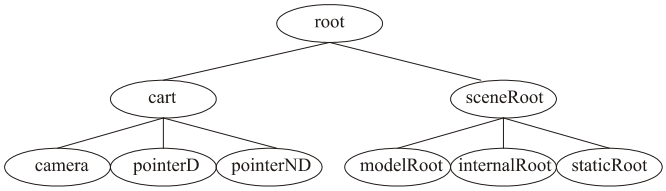


Abb. 7.4: Basis-Szenengraph

Die Bedeutung der einzelnen Knoten ist wie folgt:

- *cart:* Der „fliegende Teppich“ (nach *flying carpet paradigma* aus [ZACH96])
- *camera:* Die Hauptkamera (Auge des Betrachters)
- *pointerD:* Eingabegerät der dominanten Hand
- *pointerND:* Eingabegerät der nicht-dominanten Hand
- *sceneRoot*: Gesamte virtuelle Szene
- *staticRoot*: Statischer Teil der virtuellen Szene (d.h.
- *modelRoot*: Objekte, die zur Laufzeit / beim Startup eingeladen werden und mit denen dann interagiert wird
- *internalRoot*: Objekte, die von Komponenten angelegt wurden und in der Szene positioniert sein sollen (z.B. Marker)

Die Wurzel des Basisszenengraph kann jeder Knoten beim System „erfragen“.

## 7.3 BITs auf Basis von Fields and Routes

In diesem Kapitel werden die Anforderungen an die Umsetzung der BITs auf Systemebene formuliert und eine Realisierung auf Basis des Fields and Routes Konzeptes von VRML entwickelt. Das hier vorgestellte Konzept ist jedoch generisch genug, um auch auf anderen Basistechnologien aufgebaut zu werden.

### 7.3.1 Allgemeine Anforderungen und Basisfunktionalitäten

Im folgenden werden zuerst die allgemeinen Funktionalitäten dargelegt, die jeder BIT-Knoten bieten muss (Basisklasse). Weiterhin werden Richtlinien dargelegt, die für die Umsetzung der Interaktionstechniken beachtet werden sollten.

Jede Interaktion mit einem System hat ihren definierten Beginn und ihr definiertes Ende (Abschluss der Interaktion). In den meisten Fällen müssen zu beiden Zeitpunkten seitens der Interaktionstechnik eine oder mehrere Aktionen durchgeführt werden. Somit muss der Interaktionstechnik signalisiert werden, wann diese Zeitpunkte stattfinden. Dies kann z.B. über ein bool-Feld realisiert werden (*enabled*), bei dem *true* den Beginn und *false* das Ende der Interaktionsphase kennzeichnet.

Untersucht man die verschiedenen Interaktionstechniken, so stellt man fest, dass einige mit geometrischen Hilfsobjekten arbeiten, die in der virtuellen Szene plaziert werden. Mit diesen kann oder muss der Anwender interagieren. Beispielhaft seien hier die virtuellen Buttons (Kapitel 4.5.3.2) oder

die interaktive Säule (Kapitel 4.6.3.1) genannt. Diese Hilfsobjekte sind entweder vorab definiert und werden dann in das System geladen oder werden online generiert (z.B. WIM). In beiden Fällen müssen diese geometrischen Objekte an der „richtigen“ Stelle in den Szenengraph eingehängt werden. Hierbei entscheidet sich dann auch, wie das Objekt in der Szene positioniert wird (siehe Kapitel 4.2.2). Diese Positionierung ist jedoch von verschiedenen Faktoren abhängig, u.a. verwendete Hardware, Typ und Konzeptionierung der Anwendung. Somit kann in den meisten Fällen die Interaktionstechnik nicht über die Positionierung des Hilfsobjektes entscheiden und muss dies der Anwendung überlassen (Feld *helper_object*). In den Fällen, in denen die Positionierung jedoch seitens der Interaktionstechnik vorgegeben ist (z.B. Jog-Dial, siehe Kapitel 4.6.3.3), sollte das einhängen in den Szenengraph automatisch erfolgen. Dies setzt jedoch voraus, dass es einen definierten Basis-Szenengraphen gibt, dessen Knoten definierte Funktionen übernehmen. Siehe hierzu Kapitel 7.2.

Eine Zusammenfassung findet sich in der folgenden Tabelle.

| **Feldtyp** | **Datentyp** | **Bezeichnung** | **Erläuterung** |
|---|---|---|---|
| InField | SFBool | enabled | An- und Ausschalten des Knoten |
| exposedField | SFRef | helper_object | Hilfsobjekt für die I.technik |

Tabelle 7.1: Gemeinsame Basisklasse der BIT-Knoten

Seitens der Umsetzung der BIT-Knoten gilt es insbesondere darauf zu achten, dass der Szenengraph durch Anwenden der Technik nicht verändert wird[1]. So wurde z.B. in älteren VR-Systemen das „greifen“ von Objekten mit der virtuellen Hand (siehe Kapitel 4.3.3.2) in der Art realisiert, dass während des Greifens das Objekt im Szenengraph direkt unter die virtuelle Hand gehängt wurde. Nach dem loslassen wurde es dann wieder unter den alten Knoten zurückgehängt und die Transformation entsprechend angepasst.

1. Es sei denn dies ist der Sinn und Zweck

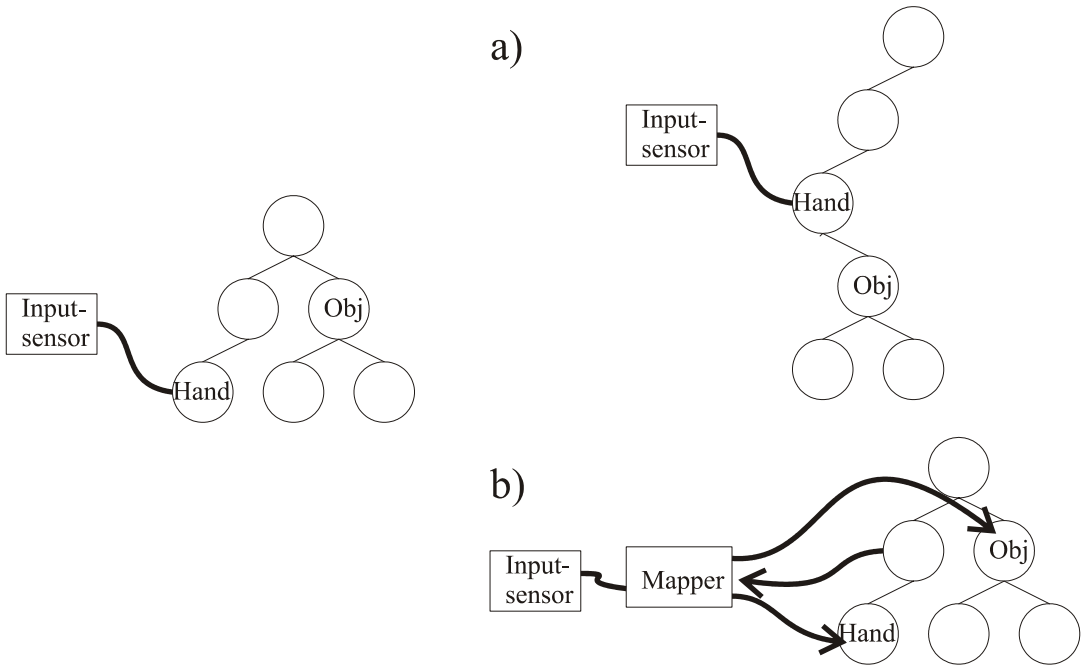


Abb. 7.5: Greifen von Objekten: Umhängen (a) versus Proxy (b)

Der große Vorteil der Methode: Während der Phase des Greifens folgt das Objekt der virtuellen Hand automatisch. Diese Vorgehensweise zieht jedoch die folgenden Nachteile nach sich:

- Zu jedem gegriffenen Objekt muss der Vaterknoten im Szenengraph gemerkt werden, um ein korrektes Zurückhängen zu gewährleisten.
- Beim Zurückhängen des Knotens kann nicht gewährleistet werden, dass die Reihenfolge der Knoten, die am Vaterknoten hängen die selbe ist, wie vor dem greifen. Diese Reihenfolge kann z.B. wichtig sein, wenn einige der Objekte transparent sind und die Objekte vor dem rendern nicht sortiert werden (können). Dann kann dies zu Darstellungsfehlern führen
- Spezielle Bewegungsmodi, bei denen die Handbewegung nicht direkt auf das Objekt gemapped werden soll, sind mit o.g. Methode nicht möglich (z.B. eingeschränkte Bewegung feste Bewegungsachsen, entlang eines Rasters, Offsets)

Weitere Beispiele, die gegen temporäre Veränderungen im Szenengraph sprechen, lassen sich auch im Zusammenhang der anderen BITs finden.

### 7.3.2 Position und Orientierung

Im Gegensatz zu Kapitel 4, in dem bei diesem BIT zwischen Navigation und Objektposition unterschieden wurde, muss dies auf Systemebene nicht gemacht werden. Somit können Interaktionstechniken sowohl für die Navigation, als auch die Objektposition über dieses BIT realisiert werden.

Wie bereits oben dargelegt ist eine Realisierung über das Umhängen von Objekten im Szenengraph nicht möglich. Somit wird Position und Orientierung als *Mapping-Objekt* realisiert, welches ausgehend von 6D-Eingabedaten diese nach vorgegebenen Schema umwandelt und auf das zu bewegende Objekt anwendet.

Ein BIT für Position und Orientierung benötigt die folgenden Informationen:

- *6D-Eingangswerte:* Untersucht man hier die verschiedenen Techniken, so stellt man fest, dass diese 6D-Werte entweder direkt von einem Gerät (z.B. Spacemouse-Navigation) oder indirekt über ein Hilfsobjekt (z.B. natürliches Greifen) abgegriffen werden. Beide Möglichkeiten sollten dem Anwendungsentwickler offen stehen.
- *Objekte:* Das BIT benötigt eine Referenz auf das zu bewegende Objekt oder Teilszenengraph. Im Zusammenhang mit dem BIT Objektselektion wird man feststellen, dass es sinnvoll ist auch ganze Objektlisten verarbeiten zu können
- *Koordinatensystem:* Das aktuelle Koordinatensystem beeinflußt maßgeblich die Abbildung der Eingabewerte auf die Ausgabewerte und erlaubt es z.B. das Rotationszentrum des gegriffenen Objektes entweder vom Proxy-Objekt oder vom gegriffenen Objekt abhängig zu machen (siehe Kapitel 4.3.1).
- *Frei definierbare Mappings: Die* Einschränkung von Bewegungen oder Bewegungen entlang eines vorgegebenen Rasters sind nur zwei Möglichkeiten, das Mapping der Ein- auf die Ausgabewerte maßgeblich zu beeinflussen. Diese sollten entsprechend den Anforderungen seitens der Anwendung eingesetzt werden können.

In VRML97/X3D kann für das Rotieren eines Objektes der *SphereSensor* verwendet werden. Hierbei werden die zu rotierenden Objekte über den Vaterknoten des SphereSensors bestimmt. Ähnlich arbeitet auch der *TouchSensor*. Würde der Positionierungs-BIT auf dem selben Konzept aufsetzen, so könnten die zu bewegenden Objekte $O_m$ dadurch bestimmt werden, indem dieser Knoten an den Vaterknoten von $O_m$ gehängt werden würde. Dies ist einfach und elegant zu lösen, wird jedoch bei Verwendung von Selektionslisten unhandlich, was an dem folgenden Szenario kurz verdeutlicht wird:

Der Anwender möchte beliebige Objekte in einer Gruppe zusammenfassen und diese dann bewegen. In einem ersten Schritt würde er sie über das Selektions-BIT selektieren. Diese selektierten Objekte lägen dann in einer Liste als Referenzen vor (*MFRef*). Siehe hierzu Kapitel 7.3.3. Nun sollen die Objekte bewegt werden. Nach dem VRML-Ansatz müßte nun das System automatisch für jede Referenz ein Positionierungs-BIT instanziieren und unter das entsprechende Objekt hängen, sowie die evtl. notwendigen Routes erzeugen (zu Proxy-Objekt, Knoten mit frei definierbarem Mapping, etc.). Dies ist jedoch nur sehr schwer zu realisieren, da hierbei das System exakt wissen muss, welche Aktionen genau auszuführen sind.

Aus diesem Grund wird hier der Ansatz verfolgt, dass die Objekte, auf die sich eine Aktion auswirkt als Liste übergeben werden, die dann vom Knoten abgearbeitet wird. Dies hat jedoch auch zur Konsequenz, dass das Berechnungsergebnis entweder über eine temporäre Route propagiert oder die Transformation direkt über die entsprechenden Methoden des Objektes gesetzt werden. Der Vaterknoten des BITs gibt das zu verwendende *Koordinatensystem* vor. Ist der Vaterknoten nicht gesetzt, werden jeweils die Koordinatensysteme der selektierten Objekte verwendet.

Die folgende Tabelle fast die Schnittstelle zusammen:

| Feldtyp | Datentyp | Bezeichnung | Erläuterung |
|---|---|---|---|
| InField | SFMatrix | 6dof | Eingangswert von Gerät |
| InField | SFRef | proxy | Eingangswert von Proxy-Objekt |
| InField | MFRef | objects | Zu bewegende Objekte |
| OutField | SFRef | cur_object | Aktuelles Objekt |
| OutField | SFMatrix | loop_out | loop für freie Mappings |
| InField | SFMatrix | loop_in | loop für freie Mappings |

Tabelle 7.2: Schnittstelle des Positionierungs-BIT

Um freie Mappings realisieren zu können, stehen im Prinzip zwei Vorgehensweisen offen:

- Es werden neue Knoten vom Positionierungs-BIT abgeleitet, die zusätzlich spezielle Mappings anbieten (*fat interface*).
- Es werden Knoten entwickelt, die nur ein spezielles Mapping anbieten (*small interface*)

Der Nachteil des fat interfaces ist, dass Kombinationen aus verschiedenen Mappings nur über Ableitung realisiert werden können. Im zweiten Fall können auf VRML-Ebene durch geschicktes verrouten der Knoten Mappings kombiniert werden. Deswegen wird hier der zweite Ansatz verfolgt.

Mappings können über die Schnittstelle *loop_out* und *loop_in* an das Positionierungs-BIT „angeschlossen“ werden. An *loop_out* liegt jeweils die aktuelle Positionierungsmatrix an, die dann vom Mappingknoten entsprechend behandelt und über *loop_in* zurückgeschrieben werden. Gehen von *loop_out* keine Routen ab, dann entfällt dieser Schritt und der Knoten führt sein eigenes Mapping durch. Als zusätzliche Information wird in *cur_object* das gerade behandelte Objekt zur Verfügung gestellt. Dies könnte für die freien Mappings wichtig sein.

Diese Vorgehensweise wird am Beispiel „world point grab“ und translatorische Bewegung entlang eines gegebenen Rasters illustriert. Der Rasterknoten wird hierbei der Einfachheit halber so realisiert, dass über ein 3D-Vektor die Rasterweiten entlang der X, Y und Z-Achse angegeben wird:

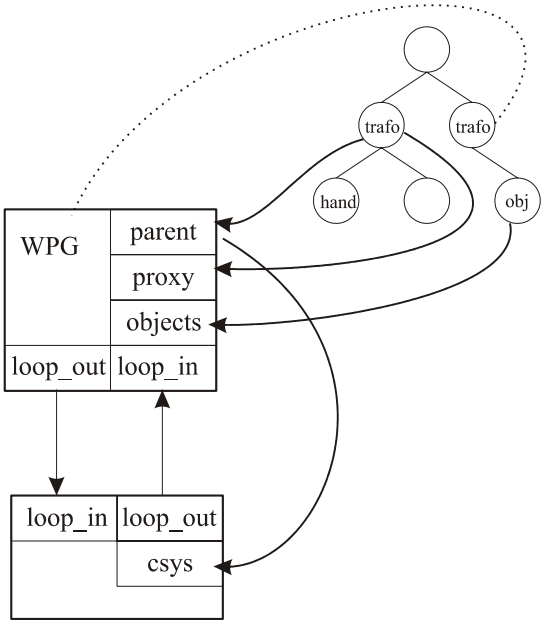


Abb. 7.6: Beispiel zu Positionierungs-BIT

### 7.3.3 Objektselektion

Objektselektion ist eine sehr wichtige Funktionalität für ein VR-System. Sie wird insbesondere dann benötigt, wenn eine Aktion auf eine nicht vorher definierbare Menge von Objekten wirken soll, z.B. die Sichtbarkeit von Objekten ändern. Diese Objektmenge wird meist durch den Anwender festgelegt. Der Selektions-BIT muss die folgenden Attribute anbieten:

- *Selektierbare Objekte:* Wie bereits im Positionierungs-BIT, wird hier mit einer Liste von Referenzen auf Objekte gearbeitet, die über den Selektions-BIT potentiell selektiert werden können.
- *Selektionstyp:* Der Selektionstyp gibt an, ob Oberflächenpunkte, Flächen oder Geometrieobjekte selektiert werden sollen. Alle drei Varianten werden jedoch nicht von allen Techniken unterstützt.
- *Selektionsart:* Ähnlich wie in „normalen“ Desktopprogrammen muss die Möglichkeit bestehen, Einfachselektion (single selection), als auch Mehrfachselektion (multi selection) anzuwenden. Bei multi selection muss der Anwender zu jeder Zeit auch wieder Objekte aus der Selektion entfernen können.

Als „Rückgabewert“ liefert ein Selektions-BIT eine Liste von selektierten „Einheiten“ des vorgegebenen Selektionstyp. Selektionslisten sind prinzipiell immer multi-fields. Bei single select wird nur der erste Eintrag dieser Liste verwendet.

Im folgenden wird zwecks der besseren Verständlichkeit als Selektionstyp Geometrieobjekt angenommen. Die Erweiterungen, die für den BIT notwendig sind, um auch die anderen Selektionstypen zu unterstützen, werden im Anschluss beschrieben.

Über das *enabled*-field wird der Selektionsprozess angeschaltet. Je nach Zustand des immediateCopy-Wertes wird ein selektiertes Objekt sofort oder erst wenn *enabled* wieder auf false geht in die Aus-

gangsliste selected kopiert. Mit Hilfe dieser Funktionalität wird gerade bei multi-selection vermieden, dass Selektionslisten propagiert werden, die noch unvollständig sind (*immediate* ist false). Andererseits ist es für die Umsetzung eines direkten Selektionsfeedbacks (z.B. Umschalten auf Wireframe) sinnvoll, jederzeit das gerade aktuell selektierte Objekt zu kennen (*immediate*==true). Brush Select ist eine weitere Anwendung für diesen Fall.

| Feldtyp | Datentyp | Bezeichnung | Erläuterung |
|---|---|---|---|
| InField | MFRef | selectable | Selektierbare Objekte |
| InField | SFBool | immediate | Sofort nach selected kopieren |
| OutField | MFRef | selected | Selektierte Objekte |

Tabelle 7.3: Interface für Selektions-BIT (single selection)

Um auch Mehrfachselektion zu unterstützen, muss dass in der vorherigen Tabelle dargestellte Interface um einige Felder erweitert werden, die in der folgenden Tabelle dargestellt sind.

| Feldtyp | Datentyp | Bezeichnung | Erläuterung |
|---|---|---|---|
| InField | SFBool | multiselect | Mehrfachselektion |
| InField | SFBool | selected_twice | Verhalten bei erneuter Selektion des gleichen Objektes |
| InField | SFBool | add | Objekt selektieren |
| InField | SFBool | remove | Objekt deselektieren |

Tabelle 7.4: Zusätzliches Interface für multi-selection

Mehrfachselektion ist im Prinzip eine Selektion in der Selektion, d.h. der Prozeß des Selektierens kann nicht nur ausschließlich über *enabled* geschaltet werden. Über *enabled* wird hier nur der Selektionsmodus an oder ausgeschaltet und entsprechend das *selected*-Feld aktualisiert. Über das Feld *add*, bzw. *remove* wird das Ergebnis der Selektionsmethode abgefragt und das entsprechende Objekt in einer internen Liste abgelegt oder daraus entfernt. Das Verhalten ist hierbei ähnlich dem *enabled* bei Single-Select. Es hat sich auch gezeigt, dass es sinnvoll ist, einen Selektionsmodus zu unterstützen, bei dem Objekte bei zweifacher Selektion wieder aus der internen Liste entfernt werden.

Um das Selektieren auch auf andere Objekttypen zu erweitern, müssen weitere Felder zum Interface hinzugefügt werden:

- Liste von Flächenindizes (MFInt)
- Liste von 3D-Punkten (MFVec)
- Bool-Werte, die angeben, welche Objekttyplisten sinnvolle Werte enthalten

Da grundsätzlich aus einem 3D-Punkt auf einer Oberfläche auf die Fläche (Dreieck) und das eigentlich Geometrieobjekt geschlossen werden kann, ist es sinnvoll bei Selektionsmethoden, die nur 3D-Punkte liefern auch die anderen Felder entsprechend anzupassen. Hierbei gilt dann, dass Einträge mit gleichem Index zusammengehören.

Am folgenden Beispiel wird die Single-Selektion anhand von Raycasting mit kontinuierlichem Feedback gezeigt. Der Raycasting-spezifische Teil wurde hier außen vor gelassen, um die Darstellung nicht zu unübersichtlich zu gestalten. Für die Feedback-Anzeige werden zwei zusätzliche, sehr einfache Knoten benötigt:

- *Wire/solid*: Dieser Knoten schaltet zwischen den Darstellungen Wireframe und Solid hin und zurück. Art der Darstellung ist abhängig vom aktuellen Darstellungszustand des Objektes.
- *Doubler*: Dieser Knoten schickt im Prinzip den Feldinhalt zweimal weiter. Dadurch kann das Umschalten beim wire/solid-Knoten sehr einfach realisiert werden. Nach dem *enable* schickt der Knoten das erste Objekt, dass in *in* anliegt direkt weiter und wird in einer internen Liste abgelegt. Bei jeder weiteren Feldänderung von *in*, wird das Objekt der internen Liste über *out* versendet und dann das neue Objekt (*in*-Feld) in der internen Liste gespeichert.

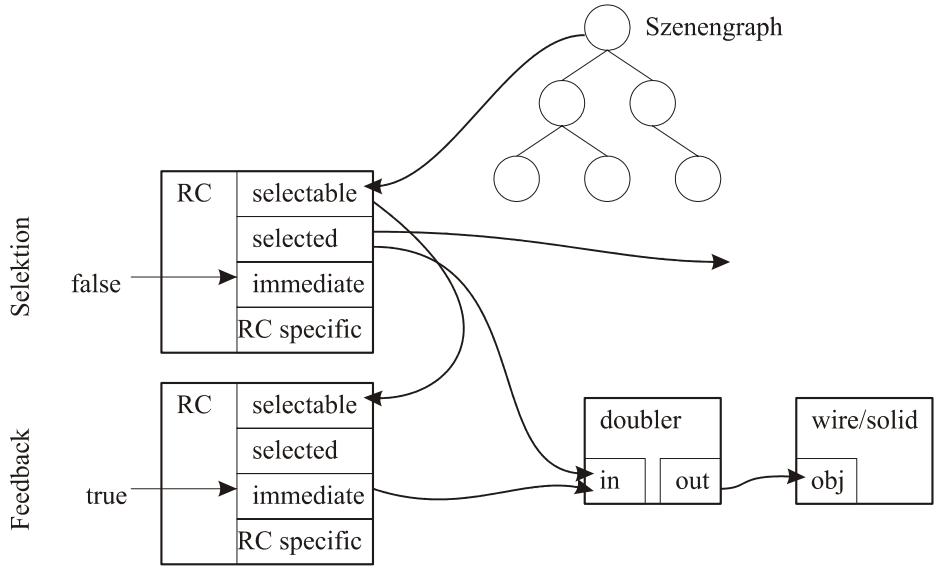


Abb. 7.7: Beispiel Selektions-BIT auf Basis von Raycasting

Um die Selektion durchzuführen, wird dem Raycasting der potentiell zu selektierende Szenengraph übergeben. Nach der Durchführung der Selektion wird das Ergebnis in *selected* bereitgestellt und kann weiter verarbeitet werden. Um das Feedback zurückzusetzen, wird *selected* zusätzlich in den Doubler geroutet. Über den zweiten Raycasting-Knoten wird das Feedback realisiert. Da *immediate* auf true steht, kann ein kontinuierliches visuelles Feedback bei der Selektion erreicht werden.

### 7.3.4 Menüselektion

Sobald ein Programm über mehrere Funktionen verfügt, müssen diese über ein Menüsystem aktiviert werden können. Im Prinzip handelt es sich hierbei zwar auch um einen Selektions-BIT, jedoch differieren die Anforderungen an Menüselektion und an Objektselektion sehr stark, so dass eine Trennung hier sehr sinnvoll ist.

Wie bereits in Kapitel 4.5.4.5 beschrieben, besteht ein graphisches Menü aus verschiedenen Komponenten:

- *Menü*: Bilden den Rahmen eines Menüs und fassen eine beliebige Anzahl an Menüknöpfen zusammen. Um Untermenüs zu realisieren und eine Hierarchie zu ermöglichen, können Menüs „geschachtelt“ werden.
- *PushButton*: Push Buttons führen auf Knopfdruck eine Funktion aus (z.B. „Speichern“)
- *ToggleButton*: Diese Art von Knöpfen führt einen internen Status (*bool*), der durch Auswahl umgeschaltet wird.
- *RadioButton*: RadioButtons sind Toggle-Buttons, die einer Gruppe zugeordnet sind. Für dieser Gruppe gilt, dass max. ein Toggle-Button den Status *true* führt. Ist ein Button A true und ein Button B der selben Gruppe wird auf true geschaltet, dann wird Button A automatisch auf false zurückgesetzt.

Diese Komponenten sind eng miteinander verzahnt und können über die folgende Ableitungshierarchie umgesetzt werden.

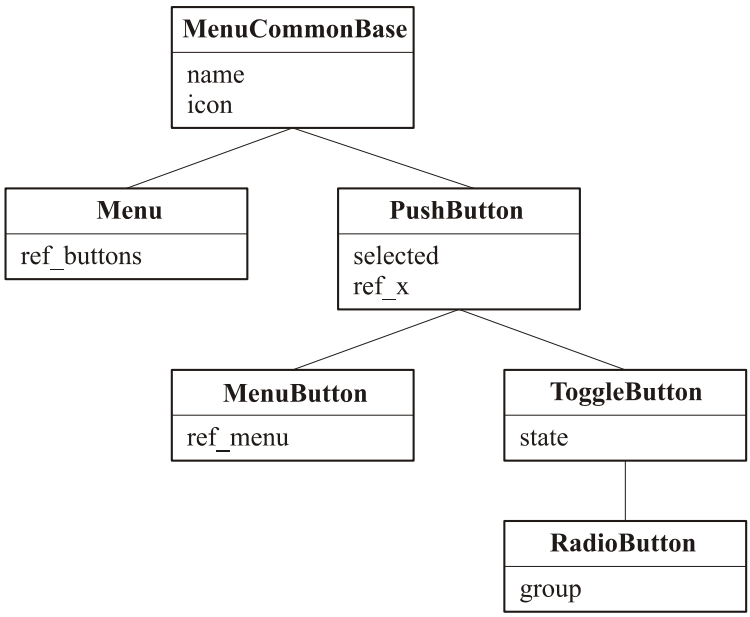


Abb. 7.8: Ableitungshierarchie der Menükomponenten

Die Erläuterung der einzelnen Attribute sind in der folgenden Tabelle zusammengefaßt.

| Feldtyp | Datentyp | Bezeichnung | Erläuterung |
|---|---|---|---|
| inField | SFString | name | Benennung des Eintrages |
| InField | SFImage | icon | Icon |
| InField | MFRef | ref_buttons | Referenz auf die Buttons im Menü |
| InField | SFBool | selected | Signal, dass Menüeintrag selektiert wurde |
| OutField | SFBool | trigger | In Abhängigkeit von Buttontyp und state wird der aktuelle Status versendet |
| exposedField | SFRef | ref_x | Zusätzliche Informationen für spezielle Menürepräsentationen |
| InField | SFRef | ref_menu | Referenz auf Untermenü |
| exposedField | SFBool | state | Status des Togglebuttons |
| inField | SFInt | group | Zugehörigkeit zu einer Radio-Gruppe |

Tabelle 7.5: Zusammenfassung der Menüschnittstelle

Über eine generische Beschreibungssprache (siehe Kapitel 4.5.4.5) oder einer direkten Definition via VRML97, kann nun über diese Knoten eine abstrakte Menühierarchie erzeugt werden, die losgelöst ist von der graphischen Repräsentation und der zu verwendenden Interaktionstechnik.

Mit Hilfe sog. *Menü-Engines* lassen sich diese abstrakten Definitionen in eine graphische Repräsentation überführen, mit der der Benutzer dann auch interagieren und Menüeinträge auswählen kann. Hierzu wird der Menü-Engine der Wurzelknoten der abstrakten Definition übergeben. Die Engine traversiert das Menü und erzeugt je nach Typus der Engine automatisch eine graphische Repräsentation. Wir können hierbei zwischen zwei Darstellungsformen unterscheiden:

- Darstellung der Menüstruktur mittels 3D-Geometrie, die in der virtuellen Szene plaziert wird. Hierbei kann dann z.B. über die vorgestellten Techniken zur Objektselektion ein Menüeintrag ausgewählt werden.
- Darstellung der Menüstruktur als 2D-Overlay oder in Form eines 2D-GUI.

Im folgenden wird zuerst auf die 3D-Darstellungsform eingegangen. Hier hat das abstrakte Interface der Engine die folgende Struktur:

| Feldtyp | Datentyp | Bezeichnung | Erläuterung |
|---|---|---|---|
| InField | SFRef | root_menu | Wurzelknoten des abstrakten Menüs |
| OutField | SFRef | root_node | Wurzelknoten der graphischen Repräsentation |
| InField | SFRef | selected_obj | Aktuell selektiertes graphisches Objekt |

Tabelle 7.6: Engine-Interface

Je nach verwendetem Menüsystem, wird von dieser Basisklasse eine entsprechende Engine abgeleitet.

Das folgende Schaubild zeigt, wie sich dieses Konzept in das Gesamtsystem integriert. In diesem Beispiel erzeugt die Engine eine graphische Repräsentationen der Menüknöpfe, die dann in den Szenengraph gehängt werden kann. Methoden, wie dieses „einhängen“ realisiert werden kann, sind in Kapitel 7.2 beschrieben. Über *Raycasting* können die graphischen Objekte selektiert werden. Das Auslösen der hinterliegenden Funktion wird jedoch auch hier über das enable-Feld gesteuert. Erst wenn es wieder auf *false* wechselt, wird das *selected*-Feld des Knoten in *selected_obj* entsprechend getriggert.

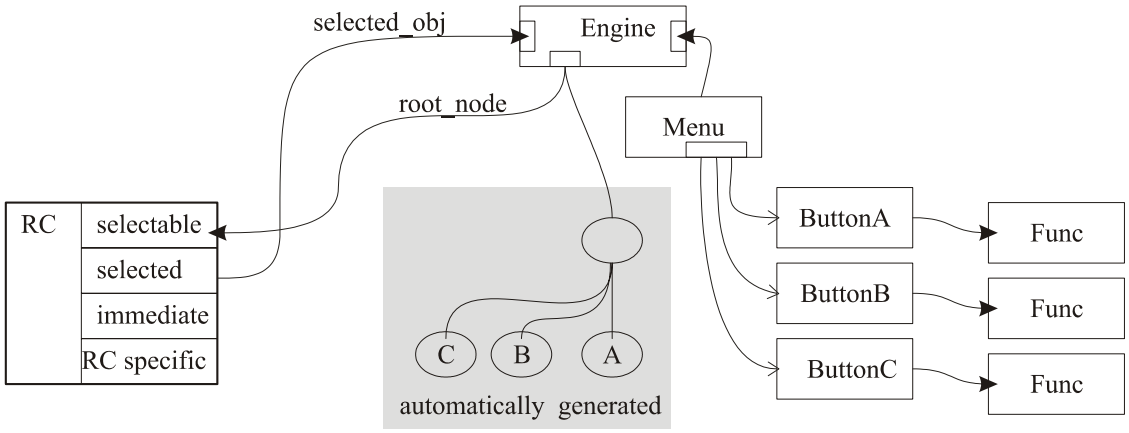


Abb. 7.9: Beispiel einfache Menüselektion: 3D Repräsentation und Raycasting

So können auch sehr einfach Menüübergänge realisiert werden. Hierzu wird Raycasting in den *immediate*-Mode gesetzt, so dass der Engine dauernd das selektierte Objekt bekannt ist. Wird ein Menu-Button selektiert, kann die Engine die graphische Repräsentation entsprechend austauschen oder erweitern.

Im Falle von 2D-Menüs (d.h. Overlay oder 2D-GUI) ist natürlich eine Selektion wie oben dargestellt nicht möglich. Im Falle des Overlays, bei dem ein Cursor in der Bildebene gesteuert wird, kann dies über ein Positionierungs-BIT realisiert werden. Im Falle der 2D-GUI greifen die Mechanismen, die das entsprechende GUI-Toolkit anbietet. Für den Anwendungsentwickler bleibt, dies jedoch transparent, da er nur die Engine entsprechend konfigurieren muss.

### 7.3.5 Quantifizierung

Über den BIT „Quantifizierung“ ist es dem Anwender möglich, Zahlenwerte in das System einzugeben. Wie bereits in Kapitel 4.6.1 beschrieben, unterscheiden sich diese Zahlenwerte durch die folgenden Attribute:

- *Typ des Zahlenwertes*, z.B. Integer, Float, Double
- *Randbedingungen*, z.B. Wertebereiche, Schrittweite und Präzision

Ein Interface für Quantifizierung könnte nun dadurch geprägt sein, dass die verschiedenen Typen von Zahlenwerten über unterschiedliche Felder abgebildet werden, wobei dann die Interaktionstechnik entscheiden muss, welche(s) Feld(er) sie mit Werten „beliefern“ kann. Ein Anwendungsentwickler müsste zusätzlich prüfen, welche Interaktionstechnik ihm die richtigen Werte liefern kann. Da jedoch

im Prinzip die verschiedenen Typen aufeinander abgebildet werden können (ggf. unter Verlust der Präzision), wird hier ein Ein-Feld-Ansatz verfolgt, der auch bei 2D-GUI Systemen zu finden ist.

Integraler Datentyp für die Quantifizierung sollte eine Fließkommazahl mit hoher Präzision (*double*). sein, wobei VRML97 nur Fließkommazahlen mit einfacher Präzision (*float*) unterstützt. Dieser Wert kann dann über Konvertierungsknoten in andere Zahlenformate umgewandelt werden.

Zusätzlich können die Randbedingungen über Felder bestimmt werden. Die Schnittstelle orientiert sich hierbei wiederum an 2D-GUI Systemen.

| **Feldtyp** | **Datentyp** | **Bezeichnung** | **Erläuterung** |
|---|---|---|---|
| InField | SFFloat | min | Untere Werteschranke |
| InField | SFFloat | max | Obere Werteschranke |
| InField | SFInt | precision | Anzahl Nachkommastellen |
| InField | SFFloat | stepsize | Schrittweite |
| OutField | SFFloat | value | Ausgabewert |

Tabelle 7.7: Interface des Quantifizierungs-BIT

In der folgenden Abbildung ist ein Beispiel für die Umsetzung des Quantify-BIT mittels eines virtuellen Scrollbars, zu sehen.

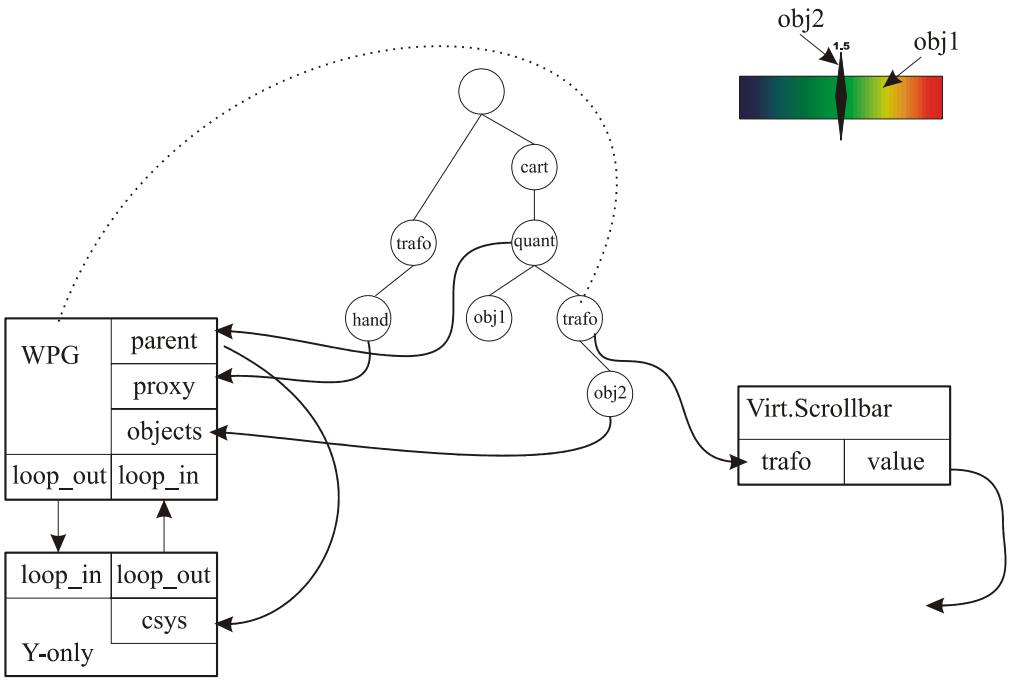


Abb. 7.10: Beispiel: Quantify

### 7.3.6 Text

Der Text-BIT ist prinzipiell sehr einfach umzusetzen, insbesondere da der Inhalt des Fließtextes ausschließlich für die Anwendung Relevanz besitzt, jedoch nicht für die Bedienoberfläche. Die Definition des Interface ist somit entsprechend trivial:

| Feldtyp | Datentyp | Bezeichnung | Erläuterung |
|---|---|---|---|
| OutField | MFString | text | Text |

Tabelle 7.8: Interface des Text-BIT

Aufgrund des sehr einfachen Interfaces wird hier von einem Beispiel abgesehen.

## 7.4 Abstraktion und CIT

Wie in den vorherigen Beispielen zu den einzelnen BITs bereits erkannt werden konnte, bauen viele der Interaktionstechniken für einen speziellen BIT auf anderen BITs auf: Der virtuelle Scrollbar benötigt ebenso wie das Raycasting einen Positionierung-BIT, die interaktive Säule benötigt sogar zusätzlich noch einen Objektselektions-BIT. Man bezeichnet solche Interaktionen auch als CITs, d.h. *composite interaction tasks*.

Diese enge Verflechtung zwischen Interaktionstechnik und BIT hat für den Entwickler einer VR-Anwendung verschiedene Konsequenzen:

- Er muss wissen, welche BITs eine bestimmte Interaktionstechnik benötigt. Diese sind wiederum über Interaktionstechniken realisiert und müssen ebenfalls im System zur Verfügung stehen.
- Der Entwickler muss die einzelnen Felder der Schnittstelle der Interaktionstechnik entsprechend verrouten. Dies umfaßt nicht nur die Felder, die ihn für seine Anwendung interessieren (z.B. bei Quantify das *value*-Feld), sondern auch alle anderen Felder, die das korrekte Funktionieren der Interaktionstechnik gewährleisten. Wie bereits an den einfachen Beispielen gesehen, kann dies recht aufwendig sein.
- Benötigt eine Interaktionstechnik Hilfsobjekte, so müssen diese natürlich auch entsprechend im System vorhanden sein, sowie „richtig“ verortet (einhängen an speziellen Knoten des Szenengraphs) und verroutet werden.
- Unabhängig von den BITs, muss er für die gestellte Aufgabe die „richtige“ Interaktionstechnik finden (Abhängigkeiten-Modell, 4W-Modell).

Ausgehend von der eigentlichen Intention des Entwicklers (z.B. Quantify für immersive Anwendung) sind dies natürlich sehr viele Aufgaben, die zur Umsetzung dieser Intention notwendig sind. Ziel ist es deswegen, das Integrieren einer Interaktionstechnik in eine VR-Anwendung und damit das Bereitstellen eines BIT stark zu vereinfachen.

### 7.4.1 Abstrakte Knoten und konkrete Interaktionstechniken

Ein ideales Vorgehen wäre, dass der Anwender seine zu verwendende Hardware, sowie einige andere globale Randbedingungen festlegt und dann die Interaktionstechniken automatisch in das System

eingebunden werden und die einzelnen BIT zur Verfügung stehen. Zwecks Optimierung sollte der Entwickler aber weiterhin die Möglichkeit erhalten, in diesen Zuordnungsprozess einzugreifen. In der Konsequenz bedeutet dies, dass eine Anwendung nur abstrakte BIT-Knoten verwendet (siehe Kapitel 7.3) und die konkreten Interaktionstechniken dann aus einem Komponentenpool / Datenbank hinzugeladen werden. Siehe hierzu die folgende Abbildung.

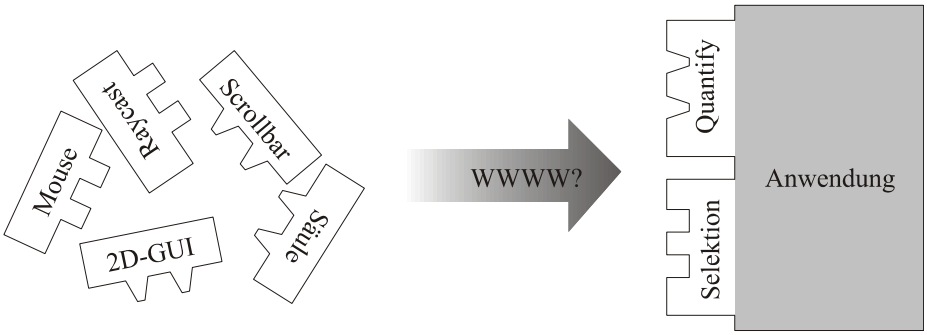


Abb. 7.11: Vorteil der Abstraktion: Größere Unabhängigkeit von Randbedingungen

Um nun die abstrakte BIT-Schnittstelle durch eine konkrete Interaktionstechnik zu ersetzen, die als Subset seiner Schnittstelle diese abstrakte BIT-Schnittstelle enthält, gibt es verschiedene Möglichkeiten (siehe Abb. 7.12)

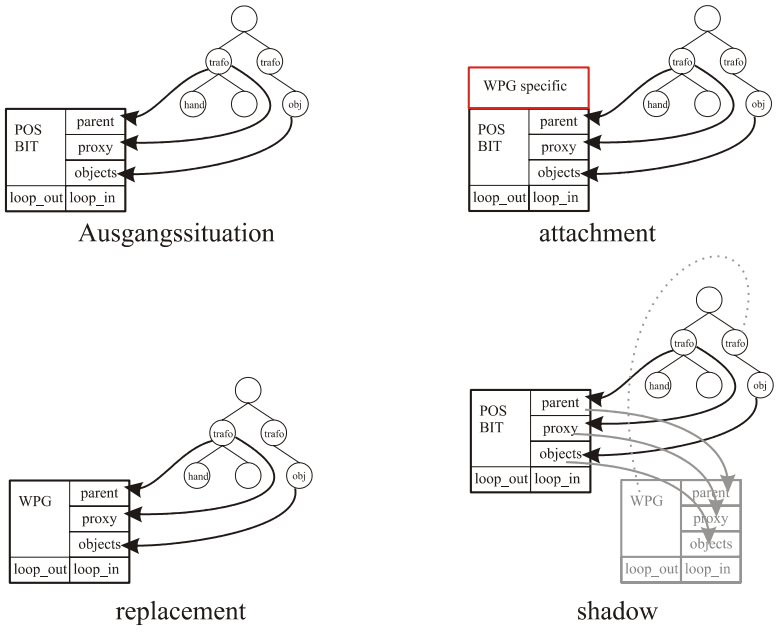


Abb. 7.12: 3 Möglichkeiten zum Ersetzen der abstrakten Schnittstelle

- *attachment*: Hier wird der abstrakte Knoten durch zusätzliche Funktionalität erweitert (z.B. zusätzliche Felder), wobei sich hierbei auch die Implementierung des Verhaltens bei Feldänderungen des abstrakten Interfaces ändert. So muss die Berechnung der neuen Positionierungswerte entsprechend angepasst werden (vgl. *Decorator-Pattern*).

- *replacement*: Hierbei werden die Routen in die Felder des abstrakte Interface auf die entsprechenden Felder des neuen Knoten umgesetzt. Dann wird der abstrakte Knoten gelöscht. Falls bereits in einem Feld eine Referenz abgelegt ist, die auf diesen Knoten verweist, könnte dieses Vorgehen zu Problemen führen.
- *shadow:* in diesem Fall bleibt der abstrakte Knoten erhalten. Der hinzugeladene Knoten wird durch *Feldsharing* oder *Weiterleitungs*-Routen an diesen abstrakten Knoten angeschlossen.

Welche Vorgehensweise endgültig gewählt wird, ist sicherlich zu einem großen Maße davon abhängig, wie das VRML-Basissystem designed und entwickelt wurde, bzw. wird. Im folgenden wird davon ausgegangen, dass die *attachment*-Variante verwendet wird.

### 7.4.2 Identifikation und Ausprägungen

Um einen abstrakten durch eine konkreten Knoten ersetzten zu können, muss es möglich sein, diese konkreten Knoten zu identifizieren, d.h. den BIT, den dieser Knoten abdeckt, zu ermitteln. Dies kann z.B. über einen flexiblen *typing-Mechanismus* geschehen, der vom Basissystem zur Verfügung gestellt wird. Wird die attachment-Variante verfolgt (*Decorator*-Pattern), dann ist die konkrete Interaktionstechnik vom abstrakten BIT-Interface abgeleitet. Unterstützt der typing-Mechanismus ein *inheritedFrom()*, so ist die Identifikation (BIT-Zugehörigkeit) sehr leicht möglich.

Hinzu kommt jedoch, dass wie bereits mit dem 4W- und Abhängigkeitsmodell gezeigt, die verschiedenen Interaktionstechniken nicht für alle Einsatzzwecke optimal sind und ggf. auch bestimmte Voraussetzungen seitens der Anwendung zu erfüllen sind. Diese Informationen sind für die Auswahl der „richtigen“ Interaktionstechnik ebenfalls sehr wichtig.

Untersucht man die in Kapitel 4 vorgestellten Interaktionstechniken, so stellt man fest, dass sich die Interaktionstechniken bzgl. der Voraussetzungen vor allem in drei Punkten unterscheiden:

1. Anzahl der unabhängigen DOF-Inputs mit meist definierter Zuordnung zu einem „Körperteil“ des Anwenders und einem geometrischen Element in der virtuellen Szene (linke/rechte Hand, Kopf/Kamera).
2. Notwendigkeit eines Selektions-BIT. Dieser kann in den meisten Fällen über eine beliebige Interaktionstechnik realisiert werden, jedoch gibt es Ausnahmen, bei denen eine spezielle Selektions-Interaktionstechnik gefordert ist. So z.B. die Look-At Menüs in Kapitel 4.5.3.4, bei denen die Selektion über die Kopfrotation gesteuert wird. Hier muss dann Raycasting als Selektions-BIT verwendet werden, bei der die Positionierung/Orientierung des Strahls über die Kamera festgelegt wird.
3. Hilfsobjekt und dessen Positionierung in der Szene: Das Hilfsobjekt kann vorab fest definiert sein und wird beim Startup in das System geladen oder es wird zur Laufzeit online generiert (z.B. WIM). Die Positionierung des Hilfsobjektes kann meist nicht direkt aus der verwendeten Interaktionstechnik abgeleitet werden, sondern ist abhängig vom Typ der Anwendung, der Hardware und dem Grundkonzept Applikation, die seitens des Entwicklers entworfen wurde. Automatismen können jedoch für eine grobe Positionierung helfen.

Eine Beschreibung einer Interaktionstechnik läßt sich somit in *offers* und *requirements* unterteilen:

1. *Offers:* Diese beschreiben, welche BITs eine Interaktionstechnik anbietet

2. *Requirements:* Diese beschreiben, welche Anforderungen eine Interaktionstechnik an eine bestehende Anwendung hat (s.o.).

Am Beispiel der interaktiven Säule illustriert, bedeutet dies folgendes:

| Offers | Requirements |
|---|---|
| • Quantifizierungs-BIT mit mehreren Float-Werten | • Eingabedaten für Quantifizierung (min, max)<br>• Zusätzliche BITs:<br>  • Selektions-BIT (für Zeiger)<br>  • Positionierungs-BIT (für Zeiger)<br>• Hilfsobjekt: Säule<br>• Logische Geräteklasse |

Tabelle 7.9: Offers und Requirements der interaktiven Säule

### 7.4.3 Initialisierungsprozess eines BIT und CIT

Am Beispiel der interaktiven Säule wird nun im folgenden die Funktionsweise und damit das Potential des hier vorgestellten Konzeptes vorgestellt:

Beim Starten des Systems durchsucht der BIT-Manager alle Knoten, auf denen die Anwendung aufbaut. Findet er einen abstrakten BIT-Knoten, so muss dieser durch einen konkreten Knoten ersetzt werden. Im Beispiel verwendet die Anwendung einen Quantifizierungs-BIT (a). Der BIT-Manager sucht nun im Knotenpool nach einem Interaktionsknoten, der zu diesem abstrakten Knoten paßt (b). Aufgrund der Randbedingungen wird die interaktive Säule (pillar) in das System geladen (c) und initialisiert (d). Nach der Initialisierung kann nun der Pillar-Knoten den Quantify-BIT-Knoten ersetzen / erweitern (e). Siehe hierzu auch die folgende Abbildung.

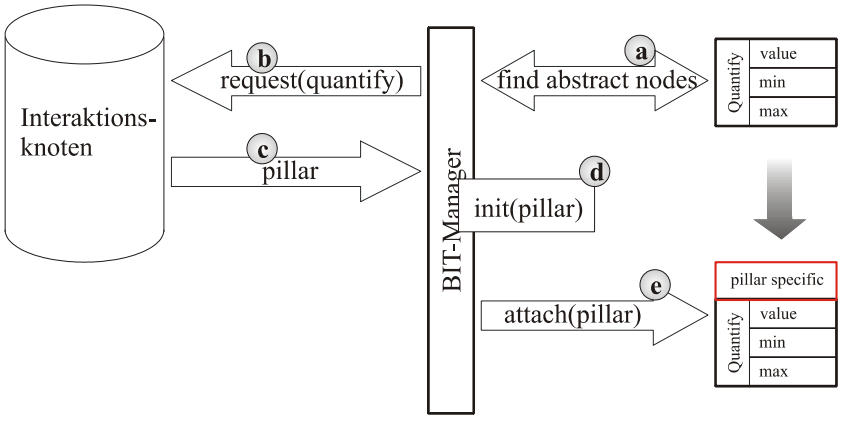


Abb. 7.13: Ablaufschema

Der eigentliche Initialisierungsvorgang der interaktiven Säule ist etwas komplexer und wird nun im folgenden beschrieben. Wie bereits oben dargelegt, benötigt die interaktive Säule einen Selektions-

BIT, um einen Zeiger selektieren zu können. Zusätzlich wird ein Positionierungs-BIT verwendet, um den selektierten Zeiger zu bewegen. Im Prinzip wird nun das in Abb. 7.13 dargestellte Schema wieder aufgegriffen, mit dem Unterschied, dass die konkreten BIT-Knoten nicht im Pool, sondern im System „gesucht“ werden. Dadurch ist es dann ohne großen Aufwand möglich, für Selektionsaufgaben ausschließlich die selbe Interaktionsmetapher zu verwenden.

In der Initialisierungsphase sucht auch hier wieder der BIT-Manager nach abstrakten Knoten. Er findet dabei den Selektions-BIT (1) und prüft nun im System, ob es hier bereits einen konkreten Knoten gibt. Ist dieser vorhanden, so wird er gecloned (2). Dies umfaßt neben dem Duplizieren des Knotens auch das duplizieren der Routes, die von und zu diesem Knoten führen (siehe Abb. 7.14).

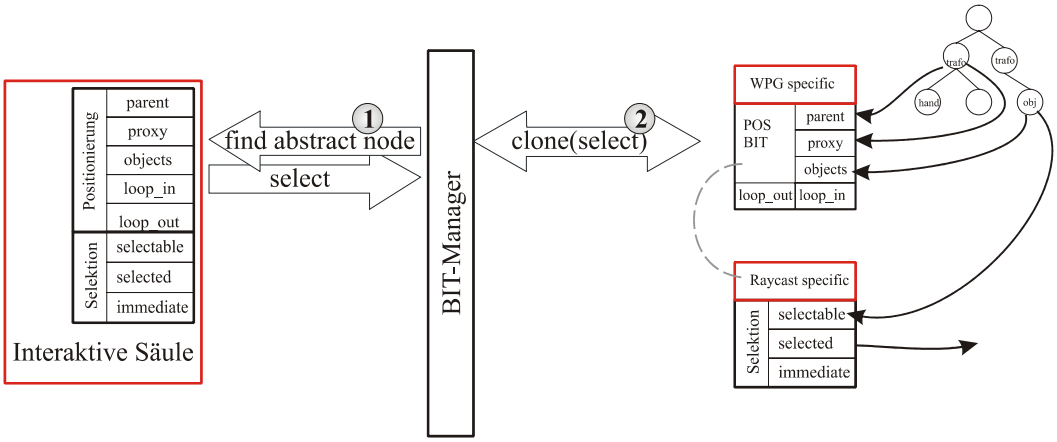


Abb. 7.14: Initialisierungsphase: Suchen abstrakter Knoten und passender Realisierung

In Abbildung 7.15 und Abbildung 7.16 ist der clone-Vorgang illustriert. Der neue Knoten wird an die interaktive Säule „übergeben“ und der dortige abstrakte Knoten konkretisiert.

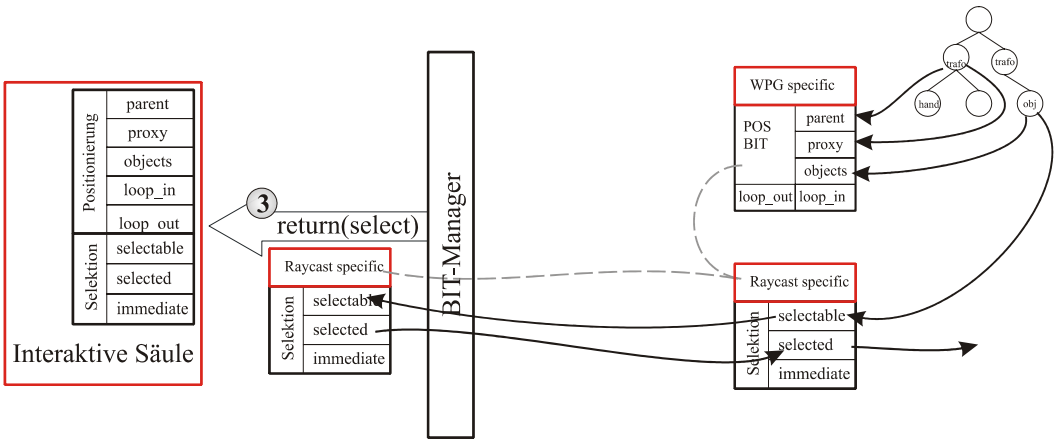


Abb. 7.15: Initialisierungsphase: clonen der Realisierung

Da die interaktive Säule von sich aus weiß, welche Objekte selektiert werden sollen, werden die zugehörigen Routes (Feld *selectable*) entsprechend auf das geometrische Objekt der interaktiven Säule

umgelegt (4). Siehe hierzu Abbildung 7.16.

Für den Entwickler einer Interaktionstechnik, wie der interaktiven Säule ist es hierbei transparent, welche Interaktionstechnik für die Selektion verwendet wird, da durch das Clonen die konkrete Realisierung entsprechend „mitkopiert“ wird. Hier im Beispiel ist Raycasting angedeutet (welches ebenfalls auf mehreren BITs aufbaut). Ebensogut könnte die Selektion auch über 2D-Maus und Mauscursor erfolgen.

Im folgenden findet der BIT-Manager einen weiteren abstrakten BIT-Knoten (Positionierung), sucht und cloned den entsprechenden konkreten Knoten, liefert diesen an die interaktive Säule zurück und erweitert/ersetzt den dortigen abstrakten Knoten (Schritte 5, 6, 7 in Abbildung 7.16).

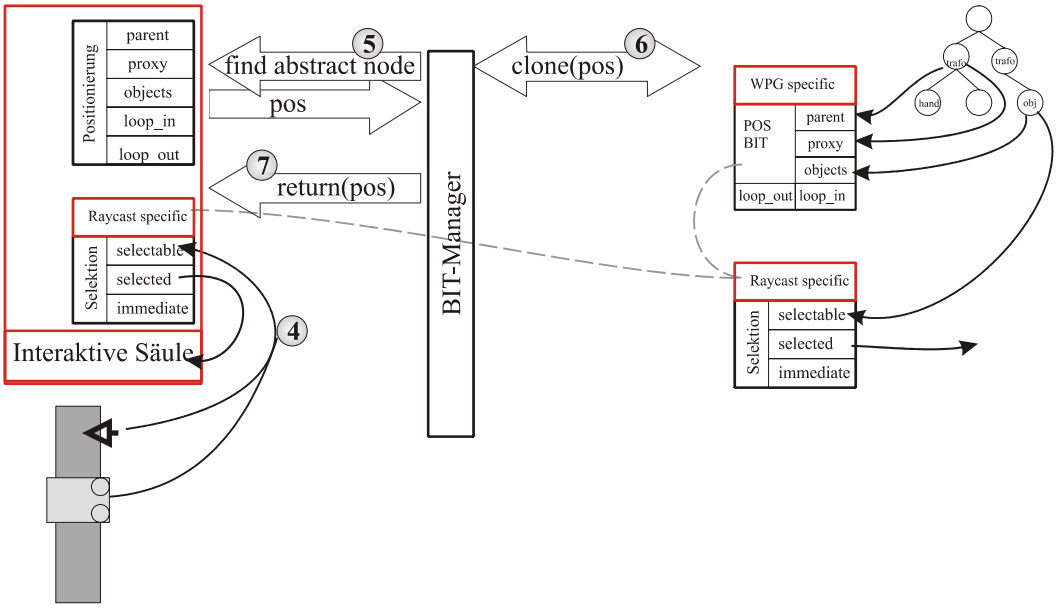


Abb. 7.16: Konkretisierung und re-routing

Auch hier wird im Anschluß dass notwendige Rerouting vorgenommen (Schritt 8 in Abbildung 7.17). Das zu bewegende Objekt ist hier immer das Selektierte, welches im eigenen Koordinatensystem bewegt wird. Der Übersichtlichkeit wegen wurde der Einschränkungsknoten für die ausschließliche Bewegung entlang der Y-Achse in dieser Darstellung weggelassen.

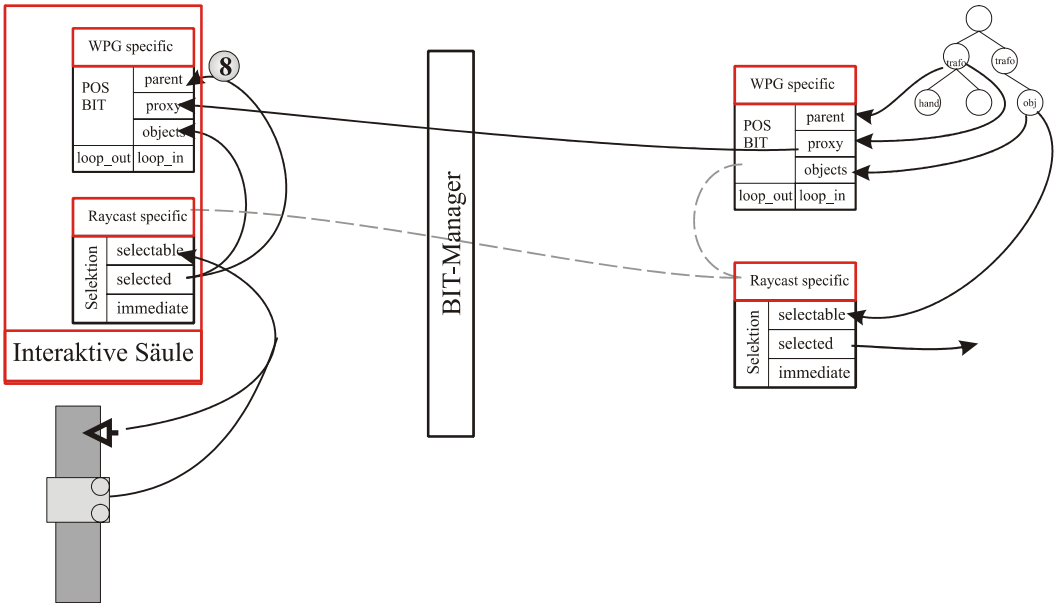


Abb. 7.17: Initialisierungsphase: Finales Rerouting

Nun ist die Initalisierungsphase für die interaktive Säule abgeschlossen und der in Abbildung 7.13 illustrierte Prozeß der Konkretisierung eines Knotens vollzogen.

## 7.5 Zusammenfassung

Das abstrakte Konzept der logischen Interaktionsaufgaben (*basic interaction task*) ist einer der zentralen Punkte in der Entwicklung VR-basierter Benutzerschnittstellen. In den vorherigen Kapiteln hat es vor allem dabei geholfen, die verschiedenen Interaktionstechniken zu klassifizieren und Funktionalitäten hinsichtlich deren Anforderungen bzgl. der Benutzerschnittstelle strukturiert zu untersuchen.

In diesem Kapitel wurden nun auch die softwareseitigen Aspekte dieses Konzeptes betrachtet. Auf Basis des Standards VRML97 wurden abstrakte Schnittstellen entworfen, die die Anforderungen der logischen Interaktionsaufgaben abdecken. Interaktionstechniken, die eine dieser BITs repräsentieren, können dann auf dieser Schnittstelle aufgebaut werden. Aufgrund der Standardisierung ist somit das Austauschen einer Interaktionstechnik für die eigentliche Anwendung transparent. Der große Vorteil: Die Anwendung fordert nur noch BITs an und ist somit u.a. unabhängig von der verwendeten Gerätetechnologie. Es ist dann ein leichtes, die Anwendung an die zu verwendenden Geräten und GUI relevanten Randbedingungen anzupassen. Dies geschieht dann durch einfaches Austauschen der Interaktionstechniken.

Dieses Vorgehen ist nicht nur für die Anwendung sinnvoll einzusetzen, auch die Interaktionstechniken selber können hiervon profitieren. Diese sind in vielen Fällen CITs (composite interaction tasks), d.h. sie bauen auf verschiedenen BITs auf (z.B. die interaktive Säule). Sind diese abstrakt gehalten, so können hier die selben Mechanismen zum Zuge kommen wie bei der eigentlichen Anwendung.

Dieses Ersetzen des abstrakten BITs durch eine konkrete Interaktionstechnik wird über den in dieser

Arbeit konzipierten BIT-Manager realisiert.

# 8 Zusammenfassung und Ausblick

## 8.1 Zusammenfassung

Ziel dieser Arbeit war es, eine ganzheitliche Vorgehensweise für die Entwicklung intuitiver und immersiver Benutzerschnittstellen für virtuelle Welten zu erarbeiten und am Beispiel des VR-Design Reviews zu demonstrieren und zu evaluieren.

Hierzu wurde in einem ersten Schritt das 4W-Modell für VR-Benutzeroberflächen entwickelt, welches dem Entwickler einen Leitfaden zur Identifikation und Klassifikation der für die Benutzerschnittstelle entscheidende Variablen, Parameter und Randbedingungen an die Hand gibt. Über das 4W-Modell werden die vier Bereiche Anwender des Systems (*Wer*), Intention bzgl. des Einsatzes des Systems (*Wozu*), Abläufe (*Wie*) und Inhalte (*Was*) adressiert. Zu diesen globalen Fragestellungen wurden im Rahmen der Arbeit die jeweils wichtigsten Teilaspekte herausgearbeitet, um eine zielgerichtete Vorgehensweise und Untersuchung zu ermöglichen. Zusätzlich wurde das Abhängigkeitenmodell entwickelt, welches aufzeigt, wie sich Anwendung, Anwender, Hardware und Software gegenseitig beeinflussen. So können die Wechselwirkungen der herausgearbeiteten Anforderungen erkannt und zur Vermeidung geeignete Lösungsstrategien entwickelt werden. Dies wurde anhand des VR-Design Reviews anschaulich demonstriert. Mit diesen Modellen hat ein Anwendungsentwickler somit mächtige Werkzeuge an der Hand, die ihm bei der Definition der Anforderungen an die zu realisierende VR-Benutzeroberfläche wesentlich unterstützen.

Für das weitere Vorgehen wurde das Konzept der logischen Interaktionsaufgaben verwendet. Dieses Konzept postuliert, dass jegliche Interaktion mit einer Benutzeroberfläche auf vier Basisinteraktionsaufgaben (basic interaction tasks) heruntergebrochen werden kann. Der Vorteile sind offensichtlich: Zum einen wird die Forderung des Normanschen Aktionsmodells nach atomaren Aktionen erfüllt. Zum anderen erlaubt es, jegliche Interaktionstechnik zu klassifizieren und dann ähnlich eines Baukastens beliebig komplexe Interaktionen auf diesen vier Interaktionsaufgaben aufzubauen. Interaktionstechniken sind somit auch als Realisierung der logischen Interaktionsaufgaben anzusehen.

Mit Hilfe dieses Konzeptes wurden dann basierend auf den Anforderungen verschiedene Interaktionstechniken untersucht und evaluiert. Da nicht für alle Ausprägungen der logischen Interaktionsaufgaben Interaktionstechniken existieren, die den Anforderungen gerecht wurden, wurden im Rahmen dieser Arbeit weitere Techniken entwickelt. Dies sind u.a. der world point grab zum Positionieren von Objekten und für das Quantifizieren der Jog-Dial und die interaktive Säule.

Zusätzlich wurde ein Menüsystem entwickelt, welches selbst hochkomplexen Systemen mit einer Vielzahl an Funktionen gerecht wird. Verschiedene graphische Ansätze, als auch Interaktionsmöglichkeiten wurden dabei untersucht. Dabei stellte sich heraus, dass Pie-Menüs um ein Vielfaches leichter zu bedienen sind, als herkömmliche, hierarchische Menürepräsentationen. Zusätzlich wurden Mechanismen integriert, die sich auch bereits im Umfeld der Desktop-Systeme als sinnvoll erwiesen haben (verschiedene Typen von Menübuttons).

Aus der Evaluierung der verschiedenen Techniken konnten im Rahmen dieser Arbeit einige informelle „goldene Regeln“ abgeleitet werden, die bei der Umsetzung, aber auch Neuentwicklung von Interaktionstechniken beachtet werden sollten. U.a. ist die menschliche Proprioception ein wertvolles

Hilfsmittel für intuitiv zu verwendende Interaktionstechniken, aber auch die Verwendung der oft „vergessenen“ Rotation kann vorteilhaft eingesetzt werden.

Anwendung finden die erarbeiteten Interaktionstechniken insbesondere bei den für eine Applikation typischen Funktionalitäten. Da die Evaluierungsgrundlage dieser Arbeit der VR-Design Review ist, wurden aus den drei Kerngebieten Modelluntersuchung, Dokumentation der Ergebnisse des Reviews und Untersuchung von Simulationsergebnissen die jeweils wichtigsten Funktionen identifiziert und mit Hilfe der verschiedenen Interaktionstechniken umgesetzt. Dabei wurden auch einige funktionale Algorithmen (z.B. solid-clipping), sowie Konzepte zur Latenzverkürzung (adaptive Darstellung) entwickelt.

Ein weiteres Schlüsselelement der adressierten Anwendungsklasse ist die Kommunikation und Kooperation zwischen verschiedenen Personen. Hierzu wurde im Rahmen dieser Arbeit ein Konzept zur Unterstützung des kooperativen Arbeitens entwickelt, das sog. *Papier und Bleistift Paradigma*. Es erlaubt zum einen, dass mehrere Anwender an ein und dem selben Objekt gemeinsam arbeiten. Zum anderen unterstützt es das Konzept des *private space*, d.h. die Entwicklung von Ideen, ohne die anderen Beteiligten des Reviews zu stören oder mit einzubeziehen. Diese „Privatssphäre“ ist für das schnelle Entwickeln von Ideen und Konzepten immanent wichtig. Um dem Anwender eine möglichst expressive Ausdrucksform zu ermöglichen, wurde im Rahmen dieser Arbeit das Skizzieren auf 3D-Modellen entwickelt. So können sehr einfach und intuitiv z.B. Anmerkungen an einem Modell angebracht werden. Damit mehrere Anwender auf einem Ausgabegerät ihre Skizze perspektivisch korrekt sehen können, wurde der MultiViews-Algorithmus entwickelt und in das System integriert.

Die softwareseitige Umsetzung der auf Basis der Anforderungen herausgearbeiteten Interaktionstechniken wäre aus Sicht eines Anwendungsentwicklers nun der nächste Schritt. Auch hierfür wurde im Rahmen dieser Arbeit eine strukturierte Vorgehensweise entwickelt. Diese setzt auf den bereits oben erwähnten logischen Interaktionsaufgaben auf, die bis hier nur zur Strukturierung und Klassifizierung von Interaktionstechniken verwendet wurden. Auf dieser Basis und der allgemeinen BIT-Anforderungen für jede logische Interaktionsaufgabe wurde eine abstrakte Softwareschnittstelle entwickelt. Als Beschreibung dieser Schnittstelle wurde auf dem Standard VRML97 aufgesetzt, diese kann jedoch leicht auch an andere Basistechnologien adaptiert werden. Auf dieser Schnittstelle können dann seitens eines Anwendungsentwicklers wiederum beliebige Interaktionstechniken aufgesetzt werden. Das entwickelte Konzept erlaubt es nun, Interaktionstechniken relativ beliebig auszutauschen und zu kombinieren, ohne Änderungen an der eigentlichen VR-Anwendung vornehmen zu müssen. So kann z.B. eine Applikation sehr leicht an unterschiedliche Gerätetechnologien angepasst werden.

Beginnend beim Herausarbeiten der Anforderungen an eine immersive, intuitive VR-Benutzerschnittstelle, über die Auswahl der „richtigen“ Interaktionstechniken, bis hin zu einer flexiblen Softwarebasis, adressiert die vorliegende Arbeit die wichtigsten Stationen der Entwicklung von VR-basierten Benutzerschnittstellen. Für einen Softwareentwickler eine entscheidende Hilfe für seine Arbeit, aber auch ein Ansatz für eine allgemeingültige und definierte Vorgehensweise für den Entwurf von VR-Benutzerschnittstellen, gepaart mit einem hohen Maß an Wiederverwendbarkeit.

## 8.2 Ausblick

Ob eine neu entworfene Interaktionstechniken intuitiv und immersiv zu verwenden ist, hängt von vielen Faktoren ab. Neben den bereits angesprochenen Aspekten des 4W-Modells gibt es jedoch auch einige „goldene Regeln“ und „Tipps und Tricks“, die bei einer Umsetzung beachtet werden sollten. In Kapitel 4.8 wurde ein solches Regelwerk exemplarisch begonnen und sollte im Rahmen zukünftiger Arbeiten aufgegriffen und zu einer Art *Best-Practice-Guide* erweitert werden.

In der letzten Zeit hat sich, vor allem aufgrund fallender Preise für die zu verwendende Gerätetechnologie, Virtuelle Realität auch in anderen Anwendungsbereichen immer stärker durchgesetzt. Hierbei sind vor allem der Edu-/Entertainment-Sektor, aber auch Anwendungen aus dem Bereich Kunst und Kultur zu nennen (z.B. [BEHR01]).

Mit Hilfe des 4W-Modells wird man sehr schnell feststellen, dass sich die Voraussetzungen und Anforderungen im Vergleich zu der in dieser Arbeit behandelten Anwendungsklasse sehr stark unterscheiden (Laienpublikum, spielerisches Lernen). Diese Anwendungsklasse mit dem hier beschriebenen Ansatz zu untersuchen würde zum einen die Übertragbarkeit der erzielten Ergebnisse unter Beweis stellen, aber vor allem auch neue Erkenntnisse bzgl. der hier zu verwendenden Interaktionstechniken liefern und Lösungsansätze für dieses Gebiet aufzeigen.

Versteht man zusätzlich das 4W-Modell als einen 4D-Raum, in dem die Ergebnismenge einer Anwendungsklasse einen Teilraum abdeckt, so wird durch Evaluierung weiterer Anwendungsklassen (s.o.) der gesamte 4D-Raum immer weiter erfaßt. Das langfristige Ziel sollte dabei sein, auf Basis einer großflächigen Abdeckung des 4D-Raums einen „Style-Guide“ für intuitive und immersive Interaktion zu entwickeln. Dieser könnte ähnlich eines Flussdiagrammes organisiert sein, welches an definierten Punkten nach Entscheidungen („*if*-Abfragen“) verlangt, die unter Zugrundelegung des 4W-Modell getroffen werden können. Hierzu müßte jedoch auch der Kriterienkatalog zu Anwender, Intention, Ablauf und Inhalt entsprechend komplettiert werden. Nach Durchlaufen des Flussdiagramms wären dann die Interaktionstechniken bestimmt und könnten über die in Kapitel 7.3 entwickelte Schnittstelle in eine Anwendung integriert werden. Aufgrund dieser definierten Schnittstelle ist es auch möglich, VR-GUI-Bibliotheken zu entwickeln, die den Prozess der Entwicklung zusätzlich vereinfachen würden.

# 9 Literaturverzeichnis


[AGRA97] Agrawala, M., Beers, A. C., Fröhlich, B., Bolas, M., Hanrahan, P.: „The two-user Responsive Workbench: support for collaboration through individual views of a shared space“, in Proceedings of the 24th annual conference on Computer graphics & interactive techniques, pp. 327-332, August 1997

[AKEL93] Akeley, K.: „Reality Engine Graphics“, in Proceedings of the Annual Conference on Computer Graphics, ACM Press, pp. 109-116, 1993

[ASTH95a] Astheimer, P.: „Sonifikation numerischer Daten für Visualisierung Virtuelle Realität“, Dissertation, Technische Hochschule Darmstadt, 1995

[ASTH95b] Astheimer, P., Dai, F., Felger, W., Göbel, M., Haase, H., Müller, S., Ziegler, R.: „Virtual Design II - An Advanced VR System for Industrial Applications“, in Proceedings of Virtual Reality World ’95, 1995

[ASTH96] Astheimer, P., Knöpfle, C.: "3D-Morphing and its Application to Virtual Reality", M. Göbel, J. David, P. Slavik, J. J. van Wijk (Eds.), "Virtual Environments and Scientific Visualization '96", pp. 85-93, Springer, 1996

[BERN99] Bern, M., Demaine, E.D., Eppstein D., Heng-Shiang Kuo, E., Mantler, A., Snoeyink, J.: „Unfoldable Polyhedra with Triangular Faces“, in Proceedings of the 4th CGC Workshop on Computational Geometry, October, 1999

[BEHR01] Behr, J., Fröhlich, T., Knöpfle, C., Kresse, W., Lutz, B., Reiners, D., Schöffel, F.: „The Digital Cathedral of Siena - Innovative Concepts for Interactive and Immersive Presentation of Cultural Heritage Sites“, in Proceedings of ICCHIM 2001, Milan, 2001

[BIER93] Bier, A., Stone M., Pier, K. Buxton, W., DeRose, T.: „Toolglass and Magic Lenses: The See-Through Interface“, Proceedings of SIGGRAPH 1993

[BILL97] Billinghurst, M., Baldis, S. Matheson, L., Philips, M.: „3D Palette: A Virtual Reality Content Creation Tool“, in Proceedings of VRST 1997, pp. 155-156

[BLAC98] Blach, R., Landauer, J., Rösch A., Simon, A.: „A Highly Flexible Virtual Reality System“, in Future Generation Computer Systems, Special Issue on Virtual Environments; Elsevier Amsterdam, 1998

[BLAN98] Blanco, E.: „L’emergence du produit dans la conception distribree“, PhD thesis, Institut National Polytechnique de Grenoble, 1998

[BLYT99] Blythe, D.: „Lighting and Shading Techniques for Interactive Apllications“, in Course notes #12 of SIGGRAPH 1999, 1999

[BÖHM96] Böhm, K.: „Ein generisches 3D-User Interface Toolkit mit Verfahren zur Gebärdenerkennung“, Dissertation, Technische Hochschule Darmstadt, 1996

[BOFF86] Boff, K. R., Kaufman, L., Thomas, J.P. (Eds.): „Handbook of Perception and Human Performance“, New York, John Wiley and Sons

[BOW97] Bowman, D. A.: „An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments", in Symposium on Interactive 3D Graphics, 1997

[BOW01] Bowman, D. A., Wingrave, C. A.: „Design and Evaluation of Menu Systems for Immersive Virtual Environments“, in Proceedings of IEEE-VR 2001, pp. 149-156, Yokohama, Japan

[BRYS91] Bryson, S., Levit, C.: „The Virtual Wind Tunnel: An Environment for the Exploration of Three Dimensional Unsteady Flows“, in Proceedings of Visualization 1991, 1991“

[CALL88] Callahan, J., Hopkins, D., Weiser, M., Shneiderman, B.: „An empirical comparison of pie vs. linear menus", in Proceedings of SIGCHI 1988, pp. 95-100

[CARD83] Card, S. K., Moran, T. P., Newell, A.: „The psychology of human-computer interaction“, Hillsdale, NJ, Lawrence Erlbaum Associates, 1983

[CATI99] CATIA V4, Produktbroschüre, http://www.catia.ibm.com/prodinfo/ccdspecs2.html, 1999

[CHEN01] Chen, H., Chen, Y., Finkelstein, A., Funkhouser, T., Li, K., Liu, Z., Samanta, R., Wallace, G.: „Data Distribution Strategies for High-Resolution Displays“, Computers & Graphics , 25(5), October, 2001.

[COCK02] Cockburn, A., McKenzie, B.: „Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments“, in Proceedings of SIGCHI 2002, pp. 203-210

[CRUZ93] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A.: „Surround-Screen Projection Based Virtual Reality: The Deisgn and Implementation of the CAVE“, in Proceedings of SIGGRAPH 1993, pp. 135-142, 1993

[CUTL97] Cuttler, L., Fröhlich, B., Hanrahan, P.: „Two-handed Direct Manipulation on the Responsive Workbench“, in Proceedings of the Symposium on Interactive 3D Graphics, 1997

[DEIS00] Deisinger, J., Blach, R., Wesche, G., Breining, R., Simon, A.: „Towards Immersive Modeling - Challenges and Recommendations: A Workshop Analyzing the Needs of Designers“, in Proceeding sof Eurographics Workshop on Virtual Environments, Amsterdam, 2000

[EHNE01] Ehnes, J., Knöpfle, C., Unbescheiden, M.: „The Pen and Paper Paradigm - Supporting Multiple Users on the Virtual Table“, in Proceedings of IEEE-VR 2001, Yokohama, Japan

[ENCA93] Encarnação J.L., Astheimer, P., Felger, W., Frühauf, T., Göbel, M., Müller, S.: „Graphics and visualization: The Essential Features for the Classification of Systems“, in Proceedings of ICCG 1993, 1993

[ENCA98a] Encarnação J.L., Knöpfle, C., Müller, S., Unbescheiden, M.: "Einsatz innovativer Technologien der virtuellen Realität - Experimente und Anwendungen mit 3-, 4- und 5seitigen CAVEs", in Internationales Wissenschaftliches Kolloquium 1998", 1998, pp. 3-16

[ENCA98b] Encarnação, M.: „Transparent Distributed Team Spaces“, in topics 3/1998, INI-GraphicsNet, pp. 26. 1998

[ENCA99] Encarnação, L. M., Bimber, O., Schmalstieg, D., Chandler, S.D.: „A Translucent Sketchpad for the Virtual Table Exploring Motion-based Gesture Recognition“, in Proceedings of EUROGRAPHICS 1999

[EYLE97] Eyles, J., Molnar, S., Poulton, J., Greer, T., Lastra, A., England, N., Westover, L.: „PixelFlow: The Realization“, in Proceedings of SIGGRAPH 1997, 1997

[FELG95] Felger, W: “Innovative Interaktionstechniken in der Visualisierung”, Dissertation Technische Hochschule Darmstadt, 1995

[FITT54] Fitts, P. M.: „The information capacity of the human motor system in controlling the amplitude of movement", in Journal of Experimental Psychology, 47/1954, 381-391.

[FOLEY90] Foley, J., van Dam, A., Feiner, S., Hughes, J.: „Computer Graphics: Principles and Practice", Addison Welsey, 1990, Second Edition

[FROE00] Froehlich T.: „IDEAL“, in Proceeding of EUROGRAPHICS 2000, 2000

[FUHR99] Fuhrmann, A., Schmalstieg, D.: „Concept and Implementation of a Collaborative Workspace for Augmented Reality“, in GRAPHICS ’99, Volume 18 (1999), 3, 1999

[GAMM95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: „Design Patterns - Elements of Reusable Object-Oriented Software“, Addison-Wesley Professional Computing Series, Addison-Wesley, 1995

[GELB90] Gelberg, L., Kamins, D., Parker, D., Sacks, J.: „Visualization techniques for structured and unstructered scientific data“, in Course notes of SIGGRAPH 1990 no. 27, „State of the art in data visualization“, 1990

[GIBB62] Gibbs, C.B.: „Controller design: Interactions of controlling limbs, time-lags and gains in positional and velocity systems“, in Ergonomics, 5, 2 (1962), pp. 385-402

[GNS00] Animator3, Produktbeschreibung unter http://www.gns-mbh.com

[GOME99] Gomes de Sa, A., Zachmann, G.: “Virtual Reality as a Tool for Verification of Assembly and Maintenance Processes”, in Computer & Graphics, vol 23, no 3, June 1999, pp. 389 - 403

[GUIA87] Guiard, Y.: „Asymetric Division of Labor in Human Skilled Bimanual Action: The last Kinematik Chain as a Model“, in Journal of Motor Behavior 19(4), pp. 486-517, 1987

[GUPT98] Gupta, S.K., Bourne, D.A., Kim, K.H., Krishnan, S.S.: „Automated process planing for sheet metal bending operations, in Journal of Manufacturing Systems, 17(5), pp. 338-360, 1998

[HAAS99] Haase, H., Bock, M., Hergenroether, E., Knöpfle, C., Koppert, H-J., Schröder, F., Trembilski, A., Weidenhausen, J.:„Where Weather Meets the Eye - A Case Study on a Wide Range of Meterological Visualizations for Diverse Audience“, Data Visualization '99, pp 261-266, Springer, 1999

[HAEF99] Häfner, U., Doulis, M., Simon, A.: „Evaluation of complex CAD data in a virtual environment", in Proceedings of the 3rd international Immersive Projection Technology Workshop, 1999, pp. 157-165

[HICK94] Hickley, K., Pausch R., Goble, J., Kassel, N.: „A Survey of Design Issues in Spatial Input“, in Proceedings of UIST ’94, 1994, pp. 213-222

[HINC94a] Hinckley, K., Pausch, R., Gobe, J.C., Kassell, N.F.: „A Survey of Design Issues in Spatial Input“, in Proceedings of ACM UIST’94 Symposium on User Interface Software & Technology, pp. 213-222, 1994

[HINC94b] Hinckley, K., Pausch, R., Gobe, J.C., Kassell, N.F.: „Passive Real-World Interface Props for Neurosurgical Visualization“, in Proceedings of CHI ’94 Conference on Human Factors in Computing Systems, pp. 452-458, 1994

[HINC97] Hinckley, K., Pausch, R., Proffitt, D.: „Attention and Visual Feedback: The Bimanual Frame of Reference“, in Proceedings of 1997 Symposium on Interactive 3D Graphics, Providence RI, USA, April 1997, ACM

[HTML99] HTML 4.01 Specification, Raggett (Ed.), http://www.w3.org/TR/1999/REC-html401-19991224, W3C Recommendation 24 December 1999

[IGAR99] Igarashi, T., Matsuoka, S., Tanaka, H.: „Teddy: A Sketching Interface for 3D Freeform Design“, Proceedings of ACM SIGGRAPH 1999, Los Angeles CA USA, pp. 409-416, 1999

[KARL94] Karlsson, K.: „Ein Interaktives System zur Visuellen Analyse von Simulationsergebnissen“, Dissertation TH Darmstadt, Shaker, 1995

[KIGER84] Kiger, J.I.: „The depth/breadth trade-off in the design of menu-driven user interfaces“, in International Journal of Man-Machine Studies, 20, 1984, pp. 201-213, 19

[KITA01] Kitamura, Y., Konishi, T., Yamamoto, S., Kishino, F.: „Interactive Stereoscopic Display for Three or More Users“, in Proceedings of ACM SIGGRAPH, 2001

[KNOE98] Knöpfle, C., Schiefele, J.: „Data Preparation on Polygonal Basis“, in F. Dai (Ed.): „Virtual Reality for Industrial Applications“, Springer-Verlag, 1998

[KNOE00a] Knöpfle, C..: „Interaction with Simulation Data“, to be published in Proceedings of Eurographics Workshop on Virtual Reality 2000

[KNOE00b] Knöpfle, C., Voß, G.: „An intuitive VR user interface for Design Review“, to be published in Proceedings of Advanced Visual Interfaces 2000

[KNOE02] Knöpfle, C.: „Working together - A VR based approach for cooperative digital design review “, in Proceedings of Advanced Visual Interfaces 2002

[KURT9x] Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, B.: „The Design and Evaluation of a GUI Paradigm based on Tablets, Two-hands, and Transparency“, Technical report, Alias | Wavefront, http://reality.sgi.com/gordo_tor/papers/t3/t3.fm.html

[LANG76] Langolf, G. D., Chaffin, D. B., Foulke, J.A.: „An investigation of Fitt’s law using a wide range of movement amplitudes“, in Journal of Motor Behaviour, 8, 2, 1976, pp. 113-128

[LEEB01] Leeb, V., Radetzky, A., Auer, L. M.: „Interactive Texturing by Polyhedron Decomposition“, in Proceedings of IEEE-VR 2001, pp. 165-171, Yokohama, Japan, 2001

[LIAN94] Liang, J., Green, M.: „JDCAD: A Highly Interactive 3D Modeling System“, in Computers and

Graphics, volume 18 (4), Jly/August 1994, pp. 499-506

[LIND99] Lindeman, R., Sibert, J., Hahn, J.: „Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments“, in Proceedings of SIGCHI 1999, pp. 64-71

[LUX00] Lux, M.: „Ein offenes Rahmensystem zur Wissenskristallisierung ökonomischer Daten“, Dissertation TUD, Fraunhofer IRB Verlag, 2000

[MACE94] Macedonia, M.R., Zyda, M.J., Pratt, D.R., Barham, P.T., Zeswitz, S.: „NPSNET: A Network Software Architecture for Large-Scale Virtual Environments“, MIT Presence, 3(4), pp. 265-287, 1994

[MACK94] MacKenzie, C.L., Iberall, T.: „The Grasping Hand“, North-Holland, Amsterdam, 1994

[MASS89] Massimo, M.J., Sheridan, T.B., Roseborough, J.B.: „One hand tracking in six degrees of freedom", in Proceedings IEEE International Conference on Systems, Man and Cybernetic, 1989, pp. 498-503

[MASS94] Massie, T. H., Salisbury, J. K.: „The PHANToM Haptic Interface: A Device for Probing Virtual Objects“, in Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems, ASME Winter Annual Meeting, Chicago, November 1994, 1994

[MILL81] Miller, D.P.: „The depth/breadth tradeoff in hierarchical computer menus“, Proceedings of the 25th Annual Meeting of the Human Factors Society, pp. 296-300

[MILL90] Mills, Z., Prime M.: „Are all menus the same? - An emprical study“, in D. Diaper, D. Gilmore, G. Cockton and B. Shacked (Ed.): „Human-computer interaction“, Interact 1990, Proceedings of the IFIP TC 13 third international conference on human-computer interaction, 1990 (pp. 423-427)

[MINE97] Mine, M.: „Moving objects in space: Exploiting proprioception in virtual environment interaction", in Proceedings of SIGGRAPH 1997

[MONT97] Montrym, J.S., Baum, D., Dignam, D.L., Migdal, C.J.: „InfiniteReality: A Real-Time Graphics System“, in Proceedings of SIGGRAPH 1997, 1997

[MUEL94] Müller, S., Schöffel, F.: „Fast Radiosity Repropagation for Interactive Virtual Environments Using a Shadow-Form-Factor-List“, in Proceedings of the 5th Eurographics Workshop on Rendering, pp. 325-342, 1994

[MYNA99] Mynatt, E. D., Igarashi, T., Edwards, W. K., LaMarca, A.: „Flatland: New Dimensions in Office Whiteboards“, in Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems, ACM CHI'99, Pittsburgh PA USA, pp. 346-353, 1999

[NAVI99] 4D Navigator, Produktbroschüre, http://www.catia.ibm.com/prodinfo/n4d.html, 1999

[NICH96] Nichols, B. Buttlar, D., Proulx Farrell, J., Farrell, J.: „Pthreads Programming: A POSIX Standard for Better Multiprocessing“, O’Reilly & Associates; 1996

[NORM86] Norman, D.A.: „Cognitive engineering“, in D.A. Norman & S.W. Draper (Eds.), „User centered system design: New perspectives on human-computer interaction“, Hillsdale, NJ: Erlbaum Associates, 1986

[NORM90] Norman, D.: „The design of everyday things", Currency Doubleday, 1990

[OHAR87] O’Hara, J. „Telerobotic control of a dexterous manipulator using master and six-DOF hand controllers for space assembly and servicing tasks“, in Proceedings of Human Factors Society 31st, Annual Meeting, 1987

[OMG98] Object Management Group: „The Common Object Request Broker - Architecture and Specification“, Revision 2.2, Februar 1998

[OPTI00] Opticore Opus Realizer, Produktbroschüre, http://www.opticore.se/products, 2000

[OROU98] O’Rourke, J.: „Folding and unfolding in computational geometry“, in Proceedings of Japan Conference Discrete and Computational Geometry, LNCS, Tokyo, December 1998]

[PAU95] Pausch, R., Burnette, T.: „Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures“, Proceedings of SIGGRAPH 1995

[PERR98] Perry, M., Sanderson, D.: „Coordinating joint design work: the role of communication and artefacts“, in „Design Studies“, pp. 273-288, Elsevier Science Ltd, 1998

[PIER97] Pierce, J.: „Image Plane Interaction Techniques in a 3D immersive environment", in Symposium on Interactive 3D Graphics, 1997

[PIER99] Pierce, J., Stearns, B., Pausch, R.: „Voodoo Dolls: Seamless Interaction at Multiple Scales in Virtual Environments“, in Proceedings of the 1999 Symposium on Interactive 3D Graphics, pp. 141-145

[PIER02] Pierce, J., Pausch, R.: „Comparing Voodoo Dolls and HOMER: Exploring the Importance of Feedback in Virtual Environments“, in Proceedings of SIGCHI 2002, pp105-112

[PIME93] Pimentel, K, Teixeira, K.: „Virtual Reality. Through the new looking glass“, New York: Windcrest, 1993

[POLH99] Polhemus Stylus, Produktbeschreibung unter http://www.polhemus.com/stylusds.htm

[POLH99b] Polhemus 3Ball, Produktbeschreibung unter http://www.polhemus.com/

[POUP96] Poupyrev, I., Billinghurst, M., Weghorst, S., Ichikawa, T.: „The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR“, in Proceedings of the ACM Symposium on User Interface Software and Technology (UIST), 1996

[POUP97] Poupyrev, I., Weghorst, S., Billighurst, M., Ichikawa, T.: „A Framework and Testbed for Studying Manipulation Techniques for Immersive VR", in ACM VRST 1997, pp. 21-28

[PRINZ86] Prinz, W., Nattkemper, D.: „Effects of secondary tasks on search performance“, in Psychological Research, 48, pp. 47-51, 1986

[PRINZ90] Prinz, W.: „Wahrnehmung“, in H. Spada (Ed.), „Lehrbuch Allgemeine Psychologie“, Verlag Hans Huber, Bern, pp. 25-114, 1990

[REAL00] Realax Realtime (RXrealtime), Produktbroschüre, http://www.realax.de/docs/html/products/urxre.htm, 2000

[REKI02] Rekimoto, J.: „SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces“, in Proceedings of SIGCHI 2002,

[REIN94] Reiners, D.: „Hochqualitatives Real-Time Rendering für Virtuelle Umgebungen“, Diplomarbeit, Technische Hochschule Darmstadt, 1994

[ROHL94] Rohlf, J., Helman, J.: „IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics“, in Proceedings of SIGGRAPH 1994, pp. 381-394, 1994

[SAGE77] Sage, G.H.: „Introduction to Motor Behaviour“, Addison-Welsey Publishing Company, Inc, 1977

[SACHS91] Sachs, E., Roberts, A., Stoops, D.: „3-Draw: a tool for designing 3D shapes“, in IEEE Computer Graphics and Applications 11(6), pp. 18-26

[SCHM96] Schmalstieg, D., Encarnação, Fuhrmann A., Szalavari, Z, Gervautz, M.: „Studierstube - An Environment for Collaboration in Augmented Reality“, in Proceedings of Collaborative Virtual Environments 1996, Nottingham UK

[SHAW93] Shaw, A., Green, M., Liang, Y., Sun, Y.: „Decoupled simulation in virtual reality with the MR toolkit“, in ACM Transactions on Information Systems, Vol. 11, No. 3, pp. 254-258, April 1993

[STOA95] Stoakley, R., Conway, R.M., Pausch, R.: „Virtual Reality on a WIM: Interactive Worlds in Miniature“, in Proceedings of SIGCHI 1995, pp. 265-272

[SUTH65] Sutherland, I.E.: „The Ultimate Display“, in Proceedings of the IFIP Congress, pp. 506-508, 1965

[SZAL96] Szalavari, Z., Gervautz, M.: „The Personal Interaction Panel - A Two-Handed Interface for Augmented Reality“, in Technical report TR-186-2-96-20, Institute of Computer Graphics, Vienna University of Technology, Austria, September 1996

[TAN98] Projektionsfirma T.A.N.: Holospace-Produktbeschreibung, http://www.tan.de

[TELL92] Teller, S.J.: „Visibility Computations in Densely Occluded Polyhedral Environments“, PhD thesis, University of California, Berkely, CA 94720, 1992, also available as Technical Report UCB/CSD-92-708, http://cstr.cs.berkeley. edu/TR/UCB:CSD-92-708

[TUFT83] Tufte, E. R.: „The visual display of quantitative information“, Cheshire, Conneticut: Graphics Press, 1983

[ULLM97] Ullmer, B., Ishii, H.: „The metaDESK: Models and Prototypes for Tangible User Interfaces“, in Proceedings of UIST 97 Banff, pp. 223-232, Alberta, Canada, 1997, ACM

[UNBE99] Unbescheiden, M.: „Physikalisch basierte Simulation in Virtuellen Umgebungen“, Dissertation, Technische Universität Darmstadt, 1999

[UPSO89] Upson, C., Kerlick, D.: „Volumetric visualization techniques“, in SIGGRAPH 1989 course notes no. 13: „2D and 3D visual workshop“, 1989

[USOH99] Usoh, M., Arthur, K., Whitton, M.C., Bastos, R., Steed, A., Slater M., Brooks, F.P.: „Walking > Walking-in-Place > Flying, in Virtual Environments“, Proceedings of SIGGRAPH 1999

[VINC99] Vinck, D.: „Ergonomics or Human Factors“, in AIT DMU-VI deliverable of Task 1.5, 1999

[VRML97] The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997, 1997

[WAND93] Wandmacher, Jens,: „Software-Ergonomie“, Walter de Gruyter-Verlag, 1993

[WANG97] Wang, C.H., Manufacturability-driven decomposition of sheet metal products“, PhD-Thesis, Carnegie Mellon University, The Robotics Institute, 1997

[WATS99] Watsen, K., Darken, R., Capps, M.: „A Handheld Computer as an Interaction Device to a Virtual Environment“, in Proceedings of the 3rd International Immersive Projection Technology Workshop (IPT), 1999, pp. 51-57

[WARE94] Ware, C., Balakrishnan „Reacing for Objects in VR Displays: Lag and Frame rate“, in ACM Transactions on Computer-Human Interaction, Vol. 1, No. 4, December 1994, pp. 331-356

[WILL99] Williams, G., Faste, H., McDowall, I., Bolas, M.: „Physical Presence - Palettes in Virtual Spaces“, in Proceedings of the 3rd International Immersive Projection Technology Workshop (IPT), 1999, pp. 65-74

[WLOK95] Wloka, M., Greenfield, E.: „The Virtual Tricorder: A unified Interface for Virtual Reality“, in Proceedings of the ACM Symposium on User Interface Software and Technology (UIST), 1995, pp. 39-40

[WOO99] Woo, M., Neider, J., Davis, T., Shreiner, D.: „The OpenGL Programming Guide - 3rd Edition“, OpenGL Architectural Review Board, 1999

[XML98] Extensible Markup Language (XML) 1.0 , http://www.w3.org/TR/1998/REC-xml-19980210, W3C Recommendation 10-February-1998

[ZACH96] Zachmann, G.: „A Language for Describing Behaviour of and Interaction with Virtual Worlds", Proceedings of ACM Conference VRST, July, 1996.

[ZACH98] Zachmann, G.: „Rapid Collision Detecion by Dynamically Aligned DOP-Trees“, in Proceedings of VRAIS ’98, 1998

[ZELE96] Zeleznik, R. C., Herndon, K. P., Hughes, J. F.: „SKETCH: An Interface for Sketching 3D Scenes“, in Proceedings of ACM SIGGRAPH '96. pp. 163-170, August 1996

[ZELE97] Zeleznik, R., Forsberg, A., Strauss, P.: „Two Pointer Input For 3D Interaction“, in Proceedings of the Symposium on Interactive 3D Graphics, 1997

[ZELE02] Zeleznik, R. C., LaViola, J. J., Feliz, D. A., Keefe, D. F.: „Pop Through Button Devices for VE Navigation and Interaction“, in Proceedings of IEEE Virtual Reality 2002

[ZHAI96] Zhai, S., Milgram, P., Buxton, W.: „The Influence of Muscle Groups on Performance of Multiple Degree-of-Freedom Input“, in Proceedings of SIGCHI 1996, 1996

[ZHAI97] Zhai, S., Milgram, P., Rastogi, A.: „Anisotropic Human Performance in Six Degree-of-Freedom Tracking: An Evaluation of 3D Display and Control Interfaces", in IEEE Transactions on

Systems, Man and Cybernetics – part A:Systems and Humans. Vol. 27, No. 4, July 1997, pp 518-528

[ZHAI98] Zhai, S., Milgram, P.: „Quantifying Coordination in Multiple DOF Movement and Its Application to Evaluating 6 DOF Input Devices“, in Proceedings of SIGCHI 1998, pp. 320-326, 1998

[ZHAN98] Zhang, H., Manocha, D., Hudson, T., Hoff, K.E.: „Visibility Culling using Hierarchical Occlusion Maps“, in Proceedings of SIGGRAPH 1998, 1998

# 10 Lebenslauf

**Persönliche Daten**

- Vorname, Nachname: Christian Knöpfle
- Geburtsdatum- und Ort: 15.1.1972 in Wattwil / CH
- Familienstand: ledig
- Staatsangehörigkeit: Deutsch
- Adresse: Rohrbachstraße 35, 60389 Frankfurt

**Schulausbildung und Studium**

- 1978-1982 Grundschule in Niedernhausen (Theißtalschule)
- 1982-1991 Gymnasium in Wiesbaden (Gutenberggymnasium)
  Abschluss: Abitur
- 1991-1995: Informatikstudium an der FH Darmstadt
  - Sudienschwerpunkte: Graphische Datenverarbeitung und Betriebssysteme
  - Thema der Diplomarbeit: 3D-Morphing in Virtuellen Umgebungen (durchgeführt am Fraunhofer-IGD)
  - Abschluss: Diplom Informatiker (Note: 1.06, bester Abschluss des Jahrgangs)·
  - 1 Semester Auslandspraktikum an der Universität in Sunderland / GB

**Berufserfahrung**

- 1.10.1995-31.3.1996 Fraunhofer-IGD in Darmstadt
  Hilfswissenschaftlicher Mitarbeiter in der Abteilung Visualisierung und Virtuelle Realität des Fraunhofer-IGD
- 1.4.1996-29.2.2000 Zentrum für Graphische Datenverarbeitung e.V. in Darmstadt
  Wissenschaftlicher Mitarbeiter in der Abteilung Visualisierung und Virtuelle Realität des Fraunhofer-IGD
  Forschungsschwerpunkte: Virtuelle Realität, Interaktion in virtuellen Welten, Augmented Reality
- ab 1.3.2000 Zentrum für Graphische Datenverarbeitung e.V. in Darmstadt
  Stellvertretender Abteilungsleiter und Leiter des Projektbereiches Virtuelle Realität der Abteilung Visualisierung und Virtuelle Realität des Fraunhofer-IGD

# 11 Eigene Veröffentlichungen

- Astheimer, P., Knöpfle, C.: "3D-Morphing and its Application to Virtual Reality", M. Göbel, J. David, P. Slavik, J. J. van Wijk (Eds.), "Virtual Environments and Scientific Visualization '96", pp. 85-93, Springer, 1996
- Knöpfle, C., Schiefele, J.: "Data Preparation on Polygonal Basis", in F. Dai (Ed.): "Virtual Reality for Industrial Applications", Springer-Verlag, 1998
- Encarnação J.L., Knöpfle, C., Müller, S., Unbescheiden, M.: "Einsatz innovativer Technologien der virtuellen Realität - Experimente und Anwendungen mit 3-, 4- und 5 seitigen CAVEs", in Internationales Wissenschaftliches Kolloquium 1998", 1998, pp. 3-16
- Haase, H., Bock, M., Hergenroether, E., Knöpfle, C., Koppert, H-J., Schröder, F., Trembilski, A., Weidenhausen, J.:"Where Weather Meets the Eye - A Case Study on a Wide Range of Meterological Visualizations for Diverse Audience", Data Visualization '99, pp 261-266, Springer, 1999
- Knöpfle, C.: "Interaction with Simulation Data", in Proceedings of Eurographics Workshop on Virtual Reality 2000
- Knöpfle, C., Voß, G.: "An intuitive VR user interface for Design Review", in Proceedings of Advanced Visual Interfaces 2000
- Knöpfle, C., Müller, M.: "Investigating FE-Datasets in Virtual Environments", in Proceedings of 16th IMACS World Congress (IMACS 2000), 2000
- Hergenröther, E., Knöpfle, C.: „Cable Installation in Virtual Environments“, in Proceedings of the International Conference on Modelling and Simulation (MS2001), 2001
- Hergenröther, E., Knöpfle, C.: „Installation and Manipulation of a Cable Harness in Virtual Environments“, in Proceedings of IASTED International Conference on Robotics and Manufacturing (RM2001), May 2001
- Ehnes, J., Knöpfle, C., Unbescheiden, M.: „The Pen and Paper Paradigm - Supporting Multiple Users on the Virtual Table“, in Proceedings of IEEE-VR 2001, Yokohama, Japan
- Behr, J., Fröhlich, T., Knöpfle, C., Kresse, W., Lutz, B., Reiners, D., Schöffel, F.: „The Digital Cathedral of Siena - Innovative Concepts for Interactive and Immersive Presentation of Cultural Heritage Sites“, in Proceedings of ICCHIM 2001, Milan, 2001
- Knöpfle, C.: „Working together - A VR based approach for cooperative digital design review“, in Proceedings of Advanced Visual Interfaces 2002, Trento, 2002
- Behr, J., Eschler, P., Fröhlich T., Knöpfle, C., Lutz, B., Müller, S., Roth, M.: „Cybernarium Days 2002 - A public experience of Virtual and Augmented Worlds“, in Proceedings of Cyberworld 2002, Tokio, 2002
- Kresse, W., Reiners, D., Knöpfle, C.: „Color Consistency for Digital Multi-Projector Stereo Display Systems : The HEyeWall and The Digital CAVE“, in Proceedings of IPT 2003, Zurich, 2003
- Knöpfle, C.: „No Silver Bullet but Basic Rules - User Interface Design for Virtual Environments“, in Proceedings of HCI International 2003, 2003
- Weidenhausen, J., Knöpfle, C., Stricker, D.: „Lessons learned on the way to industrial augmented reality applications, a retrospective on ARVIKA“, will appear in Computers and Graphics, 2003