Echtzeitbasierte Generierung und Verlegung von Leitungsobjekten in einem digitalen Fahrzeugmodell mit einem Virtual-Reality-System

von
Michael Symietz

Echtzeitbasierte Generierung und Verlegung von Leitungsobjekten in einem digitalen Fahrzeugmodell mit einem Virtual-Reality-System

Dissertation

zur Erlangung des akademischen Grades des
Doktors der Naturwissenschaften
an der Technischen Fakultät
der
Universität Bielefeld

vorgelegt von
Dipl.-Inf. Michael Symietz

Gutachter: Prof. Dr. Ipke Wachsmuth
Prof. Dr.-Ing. Gerhard Sagerer

Tag der Promotion: 14.6.2000

Michael Symietz

Volkswagen AG
VRLab Brieffach 1777
D-38436 Wolfsburg
GERMANY

Vorwort

Nach Beendigung der Arbeit an dieser Dissertation gilt es denjenigen zu danken, die mich im Laufe der Zeit unterstützt, aufgebaut, ertragen, mit mir zusammen diskutiert und gearbeitet haben.

Mein Dank gilt Professor Dr. Ipke Wachsmuth, der durch seine Betreuung und durch seinen fachlichen Rat diese Arbeit erst ermöglicht hat. Ohne Zögern erklärte er sich zu dieser Unterstützung bereit, als ich diesbezüglich an ihn herantrat. Mit Bedauern muß ich feststellen, daß es sich dabei nicht um eine Selbstverständlichkeit handelt. Daher gilt ihm mein besonderer Dank. Herrn Professor Dr.-Ing. Gerhard Sagerer sei an dieser Stelle ebenfalls für seine Zusage zur Übernahme des Zweitgutachtens sowie für sein Interesse an meiner Arbeit gedankt.

Die vorliegende Dissertation entstand während meiner Tätigkeit im Virtual Reality Labor (VRLab) der Volkswagen AG in Wolfsburg. Ich möchte an dieser Stelle allen Kolleginnen und Kollegen danken, die mich bei meiner Arbeit unterstützt und die mir hilfreich zur Seite gestanden haben. Mein besonderer Dank gilt Peter Zimmermann, dem Leiter des VRLab in Wolfsburg. Durch sein Engagement war es für mich überhaupt möglich, dieses Thema im VRLab zu bearbeiten. Seine Erfahrung und sein fachlicher Rat wie auch seine stete Hilfsbereitschaft waren Grundlagen, die zum Gelingen dieser Arbeit in erheblichem Maße beigetragen haben.

Bedanken möchte ich mich an dieser Stelle bei Ralf Rabätje, der einen großen Anteil am Gelingen der Arbeit hat. Die fachlichen Diskussionen vor, während und nach der Arbeitszeit wie auch seine Hilfe bei so mancher Fehlersuche im Quellcode waren eine große Unterstützung für mich. Nicht zuletzt soll auch seine direkte Zuarbeit im Rahmen einer Diplomarbeit nicht unerwähnt bleiben, die einen wichtigen Beitrag zu dieser Dissertation darstellt.

Bedanken möchte ich mich bei den Mitarbeitern des FhG-IGD, die stets ein offenes Ohr für mich hatten, wenn es um Fragen rund um das VR-System ging. Besonders erwähnen möchte ich dabei Dirk Reiners, Gabriel Zachmann und Christian Knöpfle, deren Geduld und Nerven ich im Laufe der Zeit oft strapaziert habe.

Vielen Dank an Marc Erich Latoschik für seine Hilfe und Kommentare zur Erstfassung dieser Arbeit. Ein Dankeschön auch an die mittlerweile ehemaligen Mitglieder der WG210 in Wolfsburg und an meine Freunde, die stets Verständnis für mich aufgebracht haben.

Ein besonderes Dankeschön gilt aber Christiane, Thomas und meinen Eltern. Ihr gutes Zureden sowie ihre Unterstützung erleichterten und ermöglichten mir das Durchhalten.

Inhalt

1. Einleitung

1.1. Diskussion der Aufgabenstellung

Die Entwicklung eines Automobils stellt einen komplexen Prozeß dar, der sowohl durch einen hohen Zeit- wie auch Kostenaufwand gekennzeichnet ist. Die zunehmende Globalisierung der Märkte und der damit verbundene erhöhte Konkurrenzdruck zwingt die Unternehmen der Automobilindustrie zu einer ständigen Optimierung dieses Prozesses. Durch eine Verkürzung der Entwicklungszeit ist es möglich, schneller auf die Marktanforderungen zu reagieren und somit einen Vorteil gegenüber Mitbewerbern herauszuarbeiten. Gleichzeitig ist gefordert, daß sowohl die Komplexität, die Variantenvielfalt wie auch die Qualität des Endproduktes ständig zunimmt. Um diese Forderungen erfolgreich umsetzen zu können, sind neue Prozesse und unterstützende Techniken und Technologien gefordert.

Der Einsatz der Computertechnik im Produktentstehungsprozeß und dessen Optimierung ist aus der heutigen Zeit nicht mehr wegzudenken. Dennoch ist es nach wie vor notwendig, das Produkt im Rahmen seiner Entwicklung als Prototyp aufzubauen und im Hinblick auf die verschiedensten Eigenschaften zu testen. Der Aufbau von physikalischen Prototypen (PMU-physical mock-up) ist sehr kosten- und zeitintensiv. Daher wird in zunehmendem Maße versucht, zum digitalen Prototypen (DMU-digital mock-up) überzugehen. Eine Voraussetzung dafür stellt der konsequente Einsatz der zur Verfügung stehenden CAx[1]-Systeme während der Produktentwicklung dar. Eine Grundvoraussetzung für den Aufbau eines vollständigen Entwicklungs-DMU, das den aktuellen Entwicklungsstand eines Produktes widerspiegelt, ist eine hundertprozentige, dreidimensionale Objektbeschreibung der einzelnen Produktkomponenten. Sie ist das Ergebnis des konsequenten Einsatzes von CAD-Systemen während des Konstruktionsprozesses.

In der Vergangenheit hat sich gezeigt, daß der Einsatz von CAD-Systemen für die Konstruktion und die vollständige Abbildung einzelner Teilsysteme und Objekte mit Problemen behaftet ist. Zu diesen problembehafteten Objekten gehören, speziell in der Automobil- wie auch in der Flugzeugindustrie, Leitungsobjekte. Im Rahmen einer Systemanalyse ist daher zu prüfen, welche Schwierigkeiten bei der konstruktiven Erzeugung von Leitungen unter Zuhilfenahme von CAD-Systemen auftreten. Hierbei ist zwischen den einzelnen Leitungsarten zu unterscheiden und der Zeitpunkt der Bauraumfestlegung mit zu berücksichtigen. Weiterhin besteht die Notwendigkeit, die Leitungsverlegung am physikalischen Modell zu analysieren.

Die ständige Leistungssteigerung der Computertechnik insbesondere im Grafikbereich hat zu Beginn der neunziger Jahre zu einer verstärkten Forschungs-

[1] CAx = Computer Aided Design (CAD), Computer Aided Engineering (CAE), Computer Aided Manufacturing (CAM)

tätigkeit auf dem Gebiet der virtuellen Realität (Virtual Reality, im folgenden VR genannt) geführt. Im Rahmen eines Forschungsprojektes bei der Volkswagen AG in Wolfsburg wird seit 1994 der Einsatz von VR-Techniken in der Automobilentwicklung geprüft. Im Rahmen dieses Projektes entstand die Idee, die Vor- und Nachteile zu überprüfen, die mit dem Einsatz eines VR-Systems im Bereich der Leitungsverlegung verbunden sind. Im Rahmen dieses Projektes wurde die hier vorliegende Arbeit erstellt.

Ziel der Arbeit ist es, einen Prototypen zu entwickeln, mit dessen Hilfe der Einsatz der VR-Technik auf diesem Spezialgebiet beurteilt werden kann. Zur Erreichung dieser Zielstellung gilt es mehrere Teilaufgaben zu lösen. Ausgehend von der bisherigen Arbeitsweise auf diesem Gebiet ist eine Sammlung von Anforderungen an das zu entwickelnde System zu erarbeiten. Diese Anforderungen sind mit den technischen Möglichkeiten bestehender VR-Systeme zu vergleichen. Ein vorhandenes VR-System ist so zu erweitern, daß es diese Anforderungen erfüllen kann. Es sind Aussagen darüber zu treffen, welche VR-Hardware am besten dazu geeignet ist, den Anwender in seiner Zielerreichung zu unterstützen.

Mit Hilfe des entwickelten Protoptypen soll weiterhin aufgezeigt werden, welche schnittstellentechnischen Voraussetzungen Grundlage für den erfolgreichen produktiven Einsatz eines solchen Systems sind. Die Möglichkeiten von verschiedenen VR-Systemen auf den unterschiedlichsten Anwendungsgebieten lassen sich anhand unterschiedlichster Publikationen, Veröffentlichungen und Präsentationen nachweisen. Ein dabei immer wieder auftretendes Problem ist sowohl die Datenerzeugung, wie auch die Definition der verschiedenen Interaktionen in einer solchen VR-Umgebung. Der Aufwand, der für die Vorbereitung einer VR-Anwendung notwendig ist, darf nicht unterschätzt werden und ist demzufolge ebenfalls zu untersuchen.

1.2. Strukturierung der Arbeit

In dem der Einleitung folgenden Kapitel 2 wird näher auf die Leitungsverlegung im Automobilbau eingegangen. Dazu werden unterschiedliche Leitungsarten betrachtet, die nach verschiedenen, für die Aufgabenstellung bedeutsamen Gesichtspunkten klassifiziert werden. Dabei spielen der Zeitpunkt der Bauraumfestlegung relativ zum Entwicklungsprozeß des Gesamtproduktes ebenso eine Rolle wie die bisherige Herangehensweise an diesen Prozeß. Aus diesen Analysen werden Anforderungen an ein digitales System gestellt, mit deren Hilfe bestehende Probleme besser gelöst werden sollen.

Im Kapitel 3 werden Hard- und Software-Voraussetzungen der zur Verfügung stehenden VR-Technik analysiert. Der State-of-the-Art auf diesem Gebiet wird mit den eigenen Anforderungen verglichen, um daraus das weitere Vorgehen ableiten zu können. Hierbei wird auf die Mensch-Maschine-Schnittstellen von CAD- und VR-Systemen ebenso eingegangen wie auf die Anforderungen an die Simulationsmöglichkeiten der VR-Systeme, um die gestellte Aufgabe lösen zu können. Weiterhin wird die Repräsentation der Daten und Informationen in den Systemen näher beleuchtet. Ergebnis der Analyse ist eine Sammlung von

Anforderungen an das zu entwickelnde System, auf die die weitere Arbeit aufbaut.

In Kapitel 4 wird auf die Umsetzung der einzelnen Anforderungen näher eingegangen. Schwerpunkte bilden hierbei die Anbindung des VR-Systems an die bestehende CAx-Umgebung und die damit verbundene Integration einzelner CAD-Basisfunktionalitäten in das System. Die Erweiterung des Systems um eine Datenstruktur, die die Basis für alle implementierten Verbindungsanalysen und -abläufe darstellt, ist ebenfalls Bestandteil dieses Kapitels. Weiterhin wird die Erweiterung des Systems um zusätzliche semantische Informationen beschrieben. Dazu wurde das VR-System mit einer wissensbasierten Komponente gekoppelt, in der diese semantischen Informationen basierend auf einer Objektklassenhierarchie abgebildet werden.

In Kapitel 5 wird die Umsetzung einzelner Funktionalitäten betrachtet, die auf die in Kapitel 4 beschriebenen Systemerweiterungen zurückgreifen. Dabei werden auch die einzelnen Anwendungsmöglichkeiten des Systems in Abhängigkeit vom jeweils notwendigen Anwendungsvorbereitungsaufwand diskutiert.

Die Diskussion einer geeigneten Hardwarekonfiguration und die Beschreibung einiger zusätzlicher VR-spezifischer Funktionalitäten sind Bestandteil des 6. Kapitels, das im wesentlichen die mit dem System gesammelten praktischen Erfahrungen reflektiert.

2. Leitungsverlegung im Automobilbau

Zum besseren Verständnis der Aufgabenstellung wird zunächst der Produktentwicklungsprozeß in der Automobilindustrie näher erläutert. Anschließend wird die Leitungsverlegung im Entwicklungsprozeß eingeordnet und analysiert.

2.1. Der Produktentstehungsprozeß in der Automobilindustrie

Bei der Entwicklung eines Automobils handelt es sich um einen komplexen, langwierigen, in Teilen parallel ablaufenden Prozeß. In den Gesamtprozeß sind Vertreter der verschiedensten Fachbereiche integriert, die die jeweiligen Interessen und Ziele vertreten. Bei den Fachbereichen handelt es sich um die Entwicklung, Produktion, Beschaffung, Finanz, Qualitätssicherung und den Vertrieb. Die Vertreter der einzelnen Fachbereiche nehmen an den Besprechungen der Fachgruppen, den sogenannten SETs[2], teil, auf die der vollständige zu entwickelnde Fahrzeugumfang abgebildet ist. Dieser ist in die Fachgruppen Karosserie, Ausstattung, Elektrik, Fahrwerk, Motor und Getriebe unterteilt [3]. Der Aufbau einer solchen Organisationsstruktur ist in Abbildung 1 am Beispiel eines speziellen Fahrzeugprojektes (B5-Projekt) dargestellt.

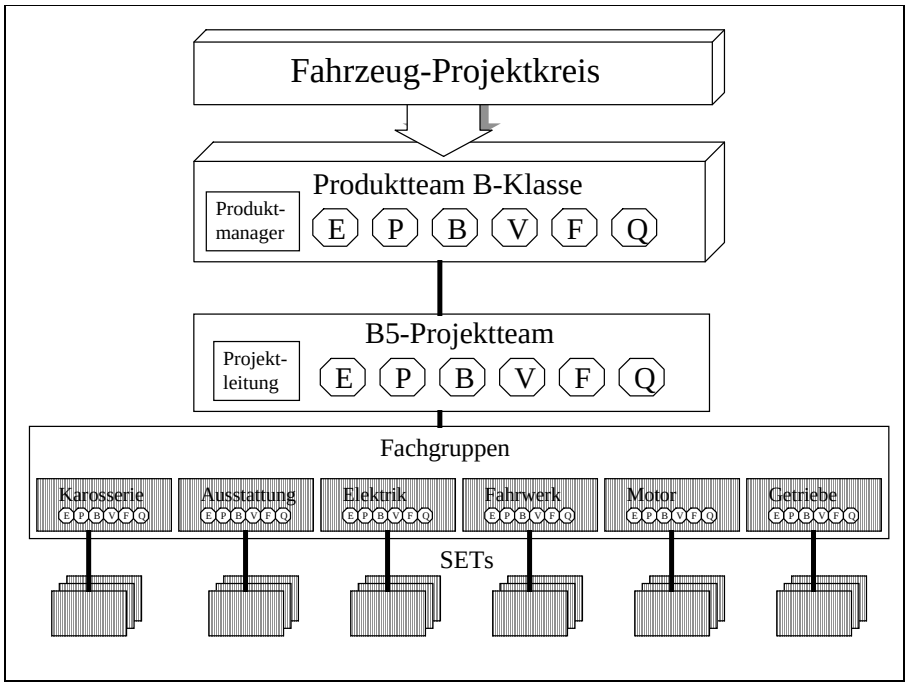


Abbildung 1: Organisationsstruktur einer Automobilentwicklung[2]

Mit Hilfe dieser Organisationsstruktur wird sichergestellt, daß die Belange einzelner Bereiche während der Produktentwicklung Berücksichtigung finden. Die SET-Treffen finden im Verlauf der gesamten Produktentwicklung statt, die sich

[2] SET-Simultaneous Engineering Team

[3] [SPIES94] Organisationsstruktur für das B5-Projekt, S.233

zum heutigen Zeitpunkt über einen Zeitrahmen von mehreren Jahren erstreckt. Die einzelnen Phasen der Produktentwicklung und ihre zeitliche Abfolge werden im folgenden Abschnitt erläutert.

2.1.1. Die Produktentstehungsphasen und ihre zeitliche Abfolge

Die Produktentwicklung in der Automobilindustrie kann prinzipiell in 6 Abschnitte unterteilt werden. Dabei handelt es sich um die Konzepterstellung, die Produktplanung, die Vorentwicklung, die Konstruktion & Entwicklung, die Produktionsvorbereitung und den Pilotanlauf. Diese Abschnitte laufen teilweise parallel ab und unterscheiden sich bei den einzelnen Herstellern lediglich in ihrer Dauer und zeitlichen Überlappung. In Abbildung 2 wird dieser Ablauf an-

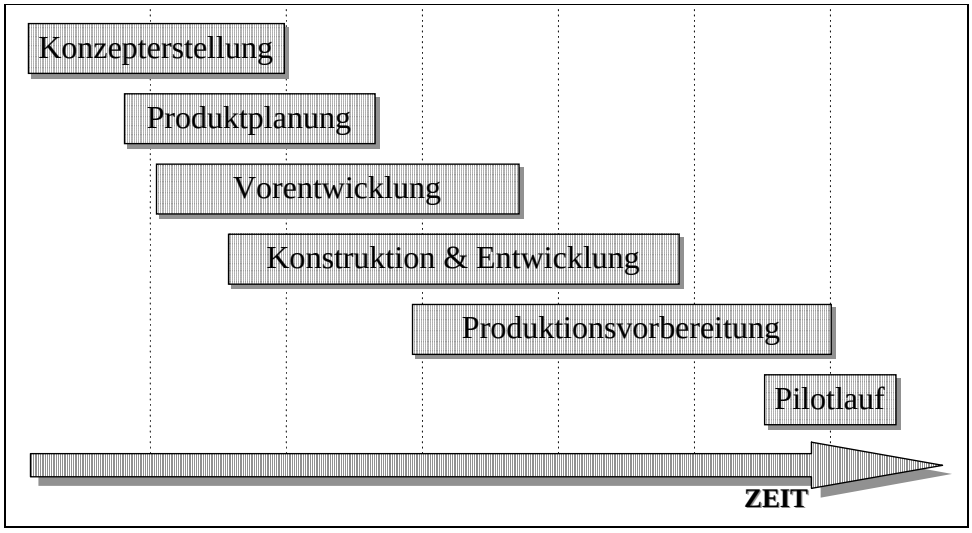


Abbildung 2: Prinzipieller Ablauf einer Automobilentwicklung in Europa[4]

hand der Entwicklung eines Automobils in Europa verdeutlicht.

Bereits in dieser sehr vereinfachten Darstellung des Produktentwicklungsprozesses wird die zeitliche Überlappung der einzelnen Abschnitte deutlich. Besonderes Augenmerk sei hierbei auf die Überlappung von Konstruktion und Entwicklung einerseits und Produktionsvorbereitung andererseits gelegt. Bereits zu einem relativ frühen Zeitpunkt wird mit der Produktionsvorbereitung begonnen. Im Rahmen der Produktionsvorbereitung werden die für die Serienfertigung notwendigen Werkzeuge in Auftrag gegeben und erprobt. Die Herstellung dieser Werkzeuge ist sehr kosten- und zeitintensiv. Nach Möglichkeit sollte ab diesem Zeitpunkt keine Änderung mehr an dem entsprechenden Fertigungsteil auftreten.

Eine detailliertere Darstellung des Projektablaufes einer Fahrzeugentwicklung ist in Abbildung 3 anhand des B5-Projektes der Volkswagen AG dargestellt.

[4] [SPIES94] Abbildung S.48

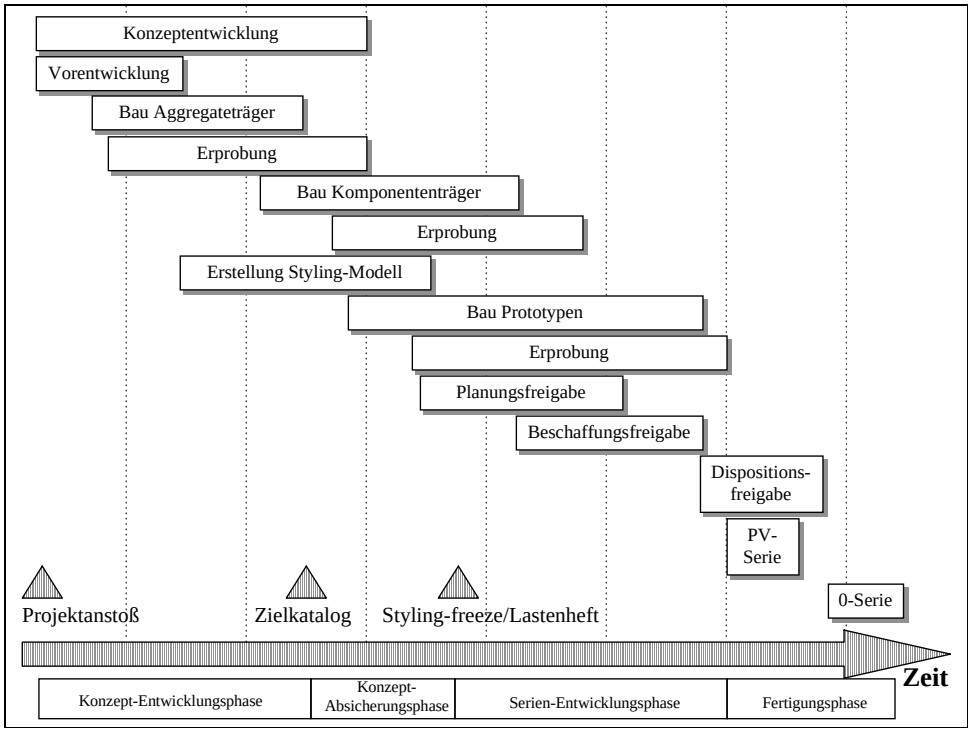


Abbildung 3: Projektablauf des B5-Projektes bei der Volkswagen AG[5]

Eine Verkürzung der Entwicklungszeit setzt eine Straffung des in Abbildung 3 dargestellten Projektablaufes voraus. Das ist möglich, indem die einzelnen Entwicklungsphasen gekürzt werden und indem sie weiter parallelisiert werden.

Die weitere Parallelisierung setzt eine Fehlerfreiheit der einzelnen Phasen voraus. Daher wird versucht, mögliche Fehlerquellen bereits zu einem sehr frühen Zeitpunkt der Entwicklung zu identifizieren und zu eliminieren. Voraussetzung dafür sind die oben dargestellten Organisationsstrukturen, mit deren Hilfe Mitarbeiter der verschiedensten Fachbereiche miteinander arbeiten und kommunizieren. Eine weitere Voraussetzung sind aber auch technische Systeme und eine ausgereifte technische Infrastruktur, die als Kommunikationshilfsmittel dienen. Warum?

In der Frühphase der Produktentwicklung sind nur wenig Informationen über das Produkt verfügbar. Oft dienen in der Konzeptphase 1:1-Schnitte mit eingezeichneten Erweiterungen als Diskussionsgrundlage für die einzelnen Fachbereichsvertreter, um die verschiedensten Probleme des entstehenden Produktes zu identifizieren. Es stehen zu diesem Zeitpunkt weder Modelle, noch fertig konstruierte CAD-Daten zur Verfügung, da deren Erzeugung sehr aufwendig ist und vor allem viel Zeit kostet[6]. Daher versucht man mit einfacheren Hilfsmitteln auszukommen. Das kann zu Fehlern führen, die erst sehr spät erkannt werden. Lediglich 10% der Probleme, die in einem Produkt-

[5] [Spies94] S. 273

[6] Quelle: Interviews mit Mitarbeitern der Volkswagen AG

entwicklungsprojekt auftreten, sind technischer Natur, 90% haben dagegen einen kommunikativen Hintergrund[7]. Einen von mehreren Gründen dafür stellen unzureichende technische Hilfsmittel dar, die kommunikationsunterstützend wirken. So kann es durchaus vorkommen, daß ein 1:1-Schnitt eines Fahrzeugvorderwagens falsch verstanden wird, daran aber entwicklungsrelevante Entscheidungen getroffen werden.

Die Verkürzung der einzelnen Entwicklungsphasen führt zu einer zwingend notwendigen Reduzierung der physikalischen Modelle und Prototypen. Hierbei stellt sich die Frage, wie bei Einhaltung der Entwicklungsziele mit einer reduzierten Anzahl von PMUs gearbeitet werden kann. Immer größere Erwartungen werden dabei an die Informationstechnik gestellt. Die Einführung des digitalen Prototypen gewinnt daher in der heutigen Zeit eine immer größere Bedeutung. Digitale Fahrzeugmodelle (DMU) beinhalten Produktinformationen, die sich aus einer einheitlichen Datenbasis (Digitaler Master) ableiten lassen. Damit können computergenerierte Präsentationen, Simulationen und Funktionalitäten aus den unterschiedlichsten Bereichen des Produktlebenszyklus nachgebildet werden. Voraussetzung dafür ist die Verknüpfung der 3D-Modelldaten mit Produktstrukturen.[8]

Zwei wesentliche Voraussetzungen muß ein digitales Modell erfüllen:

1. Die Erzeugung[9] des digitalen Modells muß wesentlich weniger Zeit in Anspruch nehmen als die Erstellung eines physikalischen Modells.
2. Die aus dem physikalischen Modell gewonnenen Erkenntnisse müssen aus dem digitalen Modell ebenfalls gewonnen werden können.

Voraussetzung für die konsequente Durchsetzung des DMU-Gedankens stellen daher eine hundertprozentige 3D-CAD-Datendurchgängigkeit wie auch eine produktorientierte Strukturierung und freie Verfügbarkeit dieser Daten in allen Fachbereichen, inklusive der externen zuliefernden Industriebereiche, dar. Ist diese Voraussetzung gegeben, so kommt man sehr schnell von den einzeln konstruierten Baugruppen und Aggregaten zum aufgebauten digitalen Modell.

Zum heutigen Zeitpunkt werden in der Automobilentwicklung gleichzeitig DMU- wie auch PMU-Methoden zur Konstruktionsabsicherung eingesetzt. Hierbei werden DMU-Methoden zum großen Teil als Hilfsmittel für den Aufbau physikalischer Prototypen genutzt[10]. Da beide Prozesse parallel zueinander ablaufen, kommt es im Laufe der Entwicklung zu einem Evaluierungsdefizit, das ebenfalls eine Fehlerquelle im Produktentstehungsprozeß darstellt und zu zeit-

[7] [SPIES94] S.293

[8] [GOMES97] Einleitung, [GEHR98], [KRAUSE98]

[9] Unter „Erzeugung eines digitalen Modells“ wird hier im wesentlichen die Ableitung des Modells aus einer umfassenden Datenbank (Digitaler Master) verstanden. Dazu zählt weiterhin die Konvertierung der das Modell beschreibenden Datenmenge in ein weiter verarbeitbares Datenformat.

[10] [GOMES97] 2.1. Geschäftsprozesse

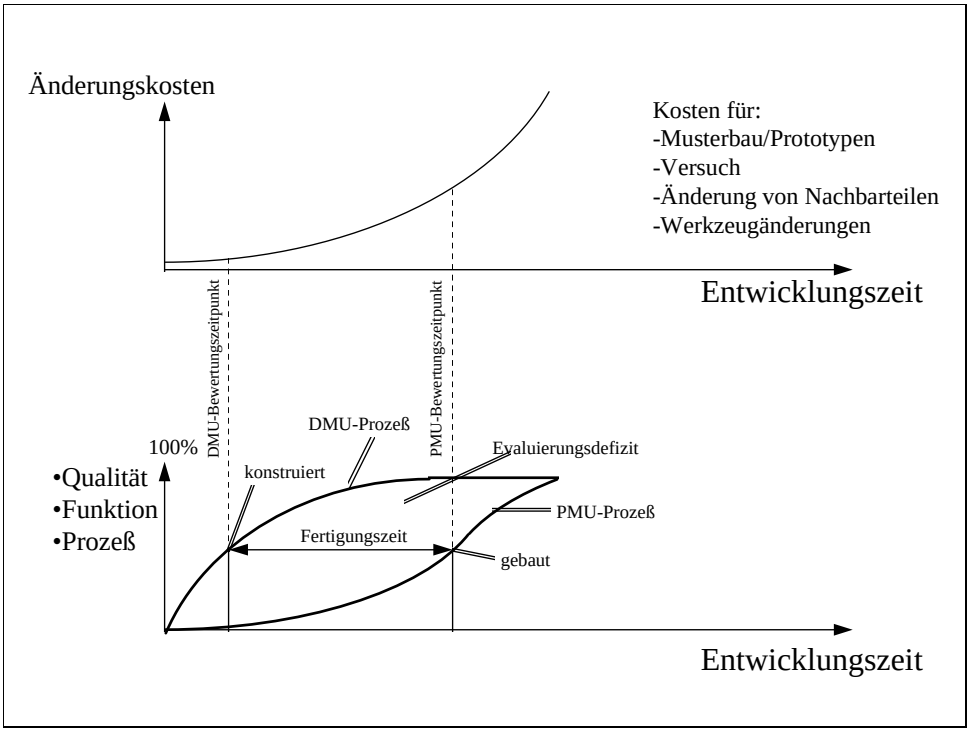


Abbildung 4: Zusammenhang zwischen DMU- und PMU-Prozeß unter Berücksichtigung der Änderungskosten

lichen Verzögerungen führen kann[11]. Das aufgebaute Produktmodell entspricht einem zu einem festen Zeitpunkt eingefrorenen Konstruktionsstand. Durch die Parallelität des Konstruktionsprozesses einerseits und des Musterbaus andererseits entsprechen daher die fertiggestellten Modelle immer einem veralteten Stand, aus dem dann Rückschlüsse gezogen werden, die wieder in den Konstruktionsprozeß Eingang finden. Durch diese zeitliche Verzögerung kommt es zu Änderungen am Produkt zu einem relativ späten Entwicklungszeitpunkt. Je später aber Änderungen auftreten, desto kostenintensiver sind sie. Diese Kosten steigen im exponentiellen Verhältnis zur Entwicklungszeit (siehe Abbildung 4).

2.1.2. Reihenfolge und Abhängigkeiten bei der Konstruktion einzelner Baugruppen

Wie im vorangegangenen Abschnitt erläutert, handelt es sich bei der Produktentwicklung eines Automobils um einen teilweise parallel, teilweise sequentiell ablaufenden Prozeß. In diesen Prozeß sind Vertreter der verschiedensten Fachbereiche wie auch Fachgruppen involviert. Ein solcher Prozeß ist zwar teilweise, aber natürlich nicht vollständig parallelisierbar. Die Konstruktion eines so komplexen Produktes besteht aus vielen Einzelaggregaten und -baugruppen, bei deren Konstruktion immer der zur Verfügung stehende Bauraum und die Umgebung mit berücksichtigt werden müssen.

[11] [GOMES97] 2.1. Geschäftsprozesse

Die konstruktive Arbeit erfolgt dabei nach der Grob-Fein Methodik. Dabei werden die entsprechenden Bauräume für die einzelnen Aggregate grob festgelegt und die Konstrukteure können mit ihrer Arbeit beginnen. Im allgemeinen wird mit dieser Arbeit nicht bei Null begonnen, sondern man baut auf Vorhandenem auf. Das gilt sowohl für die Konzeptentwicklung, wie auch für die Entwicklung einzelner Baugruppen[12].

Die Grundlage für eine Fahrzeugentwicklung stellt der Rohbau, die Plattform dar. Mit der Plattform werden die grundlegenden Eigenschaften einer Entwicklung festgesetzt. Alle in ein Fahrzeug eingebauten Teile sind am Rohbau befestigt und werden damit durch ihn beeinflußt. Daher ist die Festlegung des Rohbaus auch eine der ersten Maßnahmen im Entwicklungsprozeß.

Im Laufe der weiteren Entwicklung werden die einzelnen Aggregate und Baugruppen konstruiert. Ein Problemschwerpunkt ist dabei der Vorderwagen eines Automobils[13]. Hier befinden sich Motor und Getriebe, Kupplung, Lenkung, Lichtmaschine, Klimagerät, Wasser- und Öl-Kühler, Servogerät, Bremskraftverstärker, Luftansaugung mit Luftfilter, Scheinwerfer u.v.m.. Diese Aufzählung erhebt bei weitem nicht den Anspruch auf Vollständigkeit, soll aber die Komplexität dieses Prozesses verdeutlichen. An diesen verschiedenen Baugruppen arbeiten die verschiedensten Konstrukteure. Je weiter eine Fahrzeugentwicklung voranschreitet, desto mehr festigen sich die einzelnen Konstruktionen und um so weniger freier Bauraum steht noch zur Verfügung.

Bei der Konstruktion der einzelnen Teile ist nicht nur auf ihre räumliche Ausdehnung zu achten, sondern auch auf die unterschiedlichen funktionalen Eigenschaften und Randbedingungen. Der vollständige Vorderwagen muß sich im Fall einer Kollision sicher verhalten, eine Kraftstoffleitung darf nicht zu nahe an erhitzten Teilen vorbeigeführt werden, die einzelnen Teile dürfen nicht klappern, das Motorgeräusch soll gering ausfallen, die Einzelteile müssen montierbar sein u.v.m.. Um diese konstruktiven Eigenschaften abzusichern, werden die verschiedensten Aggregate- und Komponententräger, Teilmodelle wie auch ganze Prototypen gefertigt (siehe auch Abschnitt 2.1.1.).

Je weiter eine solche Entwicklung voranschreitet, desto mehr Aufwand ist mit einer Veränderung einer Baugruppe verbunden. Solch eine Veränderung hat meist Auswirkungen auf viele andere, in räumlicher Nähe angeordnete Aggregate. Die einzelnen Baugruppen stehen demzufolge in einer engen Beziehung zueinander und beeinflussen sich gegenseitig. Die konstruktive Reihenfolge ergibt sich aus den Beziehungen zwischen den Aggregaten und der Komplexität der Auswirkungen, die eine konstruktive Veränderung nach sich ziehen würde.

[12] Quelle: Interviews mit Mitarbeitern der Volkswagen AG

[13] Es wird hierbei davon ausgegangen, daß Motor, Getriebe, Lenkung usw. sich im Vorderwagen des Fahrzeuges befinden

2.2. Die Leitungsverlegung als Teil des Produktentstehungsprozesses

Eine Besonderheit in der Fahrzeugentwicklung stellen die Versorgungsleitungen dar. Sie verbinden die verschiedensten Baugruppen miteinander, stellen somit eine Verbindung zwischen den einzelnen Teilen her. Sie sind sehr stark von den einzelnen Baugruppen abhängig, mit denen sie verbunden sind, wie auch von den Baugruppen, an denen sie dicht vorbeigeführt werden müssen. Eine konstruktive Veränderung einer Baugruppe während der Entwicklung bedingt demnach in vielen Fällen auch eine konstruktive Veränderung von in der Nähe liegenden oder direkt verbundenen Versorgungsleitungen.

2.2.1. Systemanalyse Leitungsverlegung

2.2.1.1. Definition Leitung

Zunächst soll der in dieser Arbeit verwendete Begriff der *Leitung* näher erläutert werden. Eine Leitung im technischen Sinn ist lt. Lexikon ein:

„Anlagenelement zum Fortleiten von Signalen ... und von Energie..., oder von Stoffströmen, z.B. von Gasen und Dämpfen ... sowie Flüssigkeiten durch Rohre aus Metall, Kunststoff oder Beton“[14]

Die im Automobil vorhandenen Leitungen verbinden die einzelnen Aggregate und Baugruppen miteinander und stellen so eine Voraussetzung für die Funktionsfähigkeit des Fahrzeuges dar. Sie zeichnen sich durch ähnliche Eigenschaften aus. So haben die meisten im Fahrzeug eingesetzten Leitungen einen prismatischen Querschnitt. Weiterhin hat jede Leitung einen Anfang und ein Ende. Sie bestehen aus verschiedensten Materialien und erfüllen unterschiedlichste Aufgaben.

[14] [ROHR82]

2.2.1.2. Ziel der Systemanalyse

Aus der Menge der in einem Automobil existierenden Leitungen wurde für diese Arbeit eine Teilmenge extrahiert, die im Rahmen einer Systemanalyse näher betrachtet wurde. Es handelt sich dabei um

- Kraftstoffleitungen
- Luftleitungen
- Zündleitungen
- Hydraulikleitungen
- Unterdruck-Steuerleitungen
- Bremsleitungen
- Kühlwasserleitungen
- elektrische Leitungen
- Bowdenzüge
- Abgasrohre

Ziel der Systemanalyse war ein besseres Verständnis für die zu bearbeitende Problematik wie auch das Herausarbeiten der wesentlichen Schwachpunkte bei der Konstruktion solcher Objekte. Dazu wurden verschiedene Mitarbeiter aus unterschiedlichen Fachbereichen der technischen Entwicklung der Volkswagen AG befragt. Im Speziellen interessierten folgende Fragen:

- Wann werden die Leitungen in Form und Verlauf festgelegt?
- Welche Hilfsmittel werden dazu verwendet?
- Wer ist für die Verlegung verantwortlich?
- Welche Randbedingungen sind bei der Verlegung zu beachten?
- Wie wird die Leitung im CAD-System dargestellt, wie wird sie repräsentiert?
- Was muß man beim Festlegen des Bauraumes sehen, fühlen, testen können?

Die Ergebnisse dieser Befragung werden im folgenden Kapitel ausgewertet. Daraus werden die Anforderungen an eine VR-Anwendung abgeleitet, die den Prozeß der Leitungsverlegung unterstützen soll.

2.2.1.3. Ergebnisse der Systemanalyse

Der Ablauf der Leitungsverlegung wie auch die dabei eingesetzten Hilfsmittel sind stark von den physikalischen Eigenschaften der Leitungen abhängig. Die Leitungen werden daher in dieser Arbeit nach ihren physikalischen Eigen-

schaften klassifiziert. Dabei wird zwischen plastisch verformbaren und formstabilen Leitungen unterschieden.

Unter plastisch verformbaren Leitungen werden Objekte verstanden, die nicht vorgeformt sind. Ihre Form wird durch die auf diese Leitungen einwirkenden physikalischen Kräfte bestimmt. Dazu gehören die Arretierungskräfte an den Endbefestigungen wie auch Clipse und Befestigungsschellen zwischen diesen Punkten. Die endgültige Form ergibt sich aus der Summe aller auf sie einwirkenden Kräfte inklusive des Materialverhaltens der Leitung selbst. Leitungen mit diesen Eigenschaften können aus PVC[15], Polyamid oder Gummi bestehen, zu ihnen zählen aber auch elektrische Leitungen und Bowdenzüge.

Unter formstabilen Leitungen werden Objekte verstanden, die vorgeformt sind. Diese Schläuche werden in ihrem Verlauf festgelegt und stehen im verlegten Zustand unter einer minimalen Krafteinwirkung, d.h. unter minimaler Spannung. Es kommen hier Materialien wie Stahl (Hydraulik-Leitungen) und EPDM[16] mit einer Gewebeverstärkung zum Einsatz (siehe Abbildung 5).

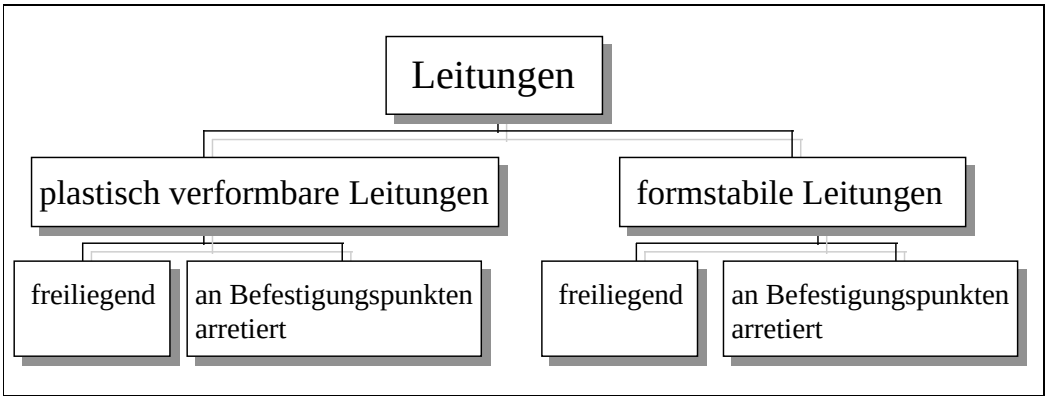


Abbildung 5: Leitungsklassifikation nach den physikalischen Eigenschaften

Eine während der Interviews immer wieder aufgeführte Schwierigkeit ist die Festlegung des Verlaufes der plastisch verformbaren Leitungen. Der Hauptgrund dafür ist bei der für diese Aufgabe unzureichenden Funktionalität der zur Verfügung stehenden CAD[17]-Programme zu finden. Die in den CAD-Programmen abgebildeten Daten basieren auf mathematischen Beschreibungen einzelner dreidimensionaler Objekte. Dabei handelt es sich um eine rein geometrische Modellierung dieser Objekte, wobei die Eigenschaften der verwendeten Materialien ebensowenig berücksichtigt werden können, wie auch die auf die Objekte wirkenden Kräfte. Daher werden diese Leitungen zwar teilweise im CAD-System modelliert, es besteht aber keine Möglichkeit für den

[15] PVC = Polyvinylchlorid

[16] EPDM = Ethylen-Propylen-Kautschuke (EPM, EPDM); Anwendungs z.B. Kfz-Kühlwasserschläuche; Quelle: [DUBBEL97]

[17] CAD = Computer Aided Design

Konstrukteur, schnell und ohne Umstände diese Konstruktion am digitalen Modell zu verifizieren. In diesen Modellierungsprozeß fließen die Erfahrungen des Konstrukteurs aus vorherigen Projekten mit ein. Aufgrund dieser Erfahrungen versucht er durch eine rein visuelle Kontrolle die Leitungen so zu modellieren, wie sie nach seiner Ansicht mit höchster Wahrscheinlichkeit verlaufen. Die Absicherung dieser Modellierung kann erst am physikalischen Prototypen erfolgen.

Eine weitere Schwierigkeit bei der Verlegung von Leitungen besteht im Finden des freien Bauraumes. Drei Hauptprobleme sind als Ursache zu nennen:

1. *Der Zeitpunkt*

   In den meisten Fällen kann der Bauraum für die Leitungen erst relativ spät festgelegt werden, da er von den in der Umgebung plazierten Aggregaten und Baugruppen abhängig ist. Deren endgültige Positionierung und beschreibende Geometrie festigt sich aber erst mit fortschreitender Entwicklungszeit.

2. *Komplexität*

   Die Leitungen erstrecken sich teilweise über größere Entfernungen innerhalb des Fahrzeuges. Um den Bauraum festlegen zu können, müssen die in der Nähe der zu verlegenden Leitung befindlichen Objekte mit in die CAD-Umgebung eingeladen werden. Sind diese vollständig verfügbar, so erzielt man sehr schnell eine große Szenenkomplexität, die teilweise die heute eingesetzten Workstations in ihrer Leistungsfähigkeit überfordern. Das führt zu wesentlich längeren Antwortzeiten des Systems, was die Arbeit wesentlich erschwert bzw. sogar teilweise unmöglich macht.

3. *Die Unterstützung durch das CAD-System*

   Die im Einsatz befindlichen CAD-Systeme können nach den Aussagen verschiedener Interviewpartner folgendermaßen für die Leitungsverlegung genutzt werden:

   Mit Hilfe von 2D-Schnitten, die durch eine CAD-Umgebung gelegt werden, kann der für die zu konstruierende Leitung benötigte Bauraum gefunden werden. Auf diesen Schnittebenen werden die Punkte festgelegt, durch die die Mittellinie der künftigen Leitung verlaufen soll. Diese wird anschließend, meist in Form einer parametrisch beschriebenen Kurve, erzeugt. Um diese Mittellinie werden später Flächen generiert, die letztendlich die Leitungsgeometrie repräsentieren. Hierbei geht man von einem groben Modell aus, das immer weiter verfeinert wird. Das Finden des Bauraumes mit Hilfe verschiedener Schnitte stellt eine umständliche, unnatürliche Vorgehensweise dar, die von den Konstrukteuren immer wieder kritisiert wurde. Die beschriebene Vorgehensweise ist sehr aufwendig und kostet daher viel Zeit. Bei auftretenden Veränderungen in der Nähe der zu konstruierenden Leitungen ist dieser Vorgang immer wieder zu wiederholen, um die Konstruktion an die neuen Randbedingungen anzupassen.

Wie bei anderen Baugruppen und Aggregaten spielt die Montierbarkeit und Austauschbarkeit einzelner Teile auch bei der Festlegung der Schlauchverläufe eine wesentliche Rolle. Diese Montageuntersuchungen werden von der eigentlichen Modellierung getrennt betrachtet, obwohl sie in einer sehr engen Wechselbeziehung zueinander stehen. Auch hier existieren Probleme bei der Nutzung von CAD-Systemen für diese Aufgabe

a) durch die eingeschränkten Simulationsmöglichkeiten dieser Systeme und

b) durch die zur Verfügung stehende Mensch-Maschine-Schnittstelle (Ein- und Ausgabehardware der CAD-Workstation) und damit verbunden durch die implementierten Interaktionsmetaphern.

Bei der Montage von Leitungsobjekten am realen Fahrzeug macht man sich intuitiv die Verformung dieser Objekte zunutze. Um einen Kühlwasserschlauch zwischen Motor und Kühler zu montieren, werden weder der Motor noch der Kühler gelockert. Der Schlauch wird vielmehr durch direkte Krafteinwirkung so verformt, daß er zunächst an dem einen Aggregat befestigt wird und danach am anderen. Dazu wird er gestaucht und gekrümmt, so daß er sich über den entsprechenden Anschlußstutzen ziehen läßt. Diese Funktionalität wird von den zur Verfügung stehenden CAx-Systemen entweder gar nicht oder nur sehr eingeschränkt unterstützt.

Selbst für den Fall, daß ein CAx-System Simulationsalgorithmen für einen solchen Vorgang bietet, ist es immer noch umständlich, mit heute gebräuchlichen Eingabegeräten eine solche Montage durchzuführen. Das zu montierende Objekt muß gegriffen werden und ein Weg muß vorgegeben werden, den es im Laufe der Simulation beschreiben soll. In der Realität wird dieser Weg direkt vom Menschen vorgegeben.

Einfache Montageuntersuchungen werden heute im Computer mit Hilfe entsprechender CAx-Werkzeuge durchgeführt. Hierbei kommen Systeme wie die Virtuelle Werkstatt (Fa. Tecoplan) oder verschiedenste CAD-Systeme zum Einsatz. Den Kern dieser Montageuntersuchungen bildet eine im Batchbetrieb laufende Kollisionsuntersuchung, die als Ergebnis sich gegenseitig durchdringende Teile markiert. Dabei können einzelne Objekte durch Transformationsmatrizen im Raum bewegt werden. Eine Berücksichtigung der Verformung der Objekte, ein natürliches Greifen und ein einfaches, natürliches Festlegen der zu kontrollierenden Bewegung stellen aber Defizite dieser Systeme dar.

Der Zeitpunkt der Leitungsverlegung relativ zur Gesamtentwicklungszeit kann nicht allgemein definiert werden. Er ist direkt vom Leitungstyp abhängig. Bei den meisten Leitungsarten konnte jedoch festgestellt werden, daß sie zu einem relativ späten Zeitpunkt in der Fahrzeugentwicklung festgelegt werden. Dieser Zustand kann darauf zurückgeführt werden, daß die Leitungen in extremer Weise von ihrer unmittelbaren Umgebung abhängig sind und fast jede geometrische Veränderung dieser Umgebung auch Auswirkungen auf die entsprechenden Leitungen hat.

2.2.2. Schlußfolgerungen aus der Systemanalyse, Anforderungen an das Zielsystem

An der Entwicklung eines komplexen Produktes sind eine Vielzahl von Spezialisten beteiligt, die letztendlich alle am gleichen Produkt arbeiten. Sie müssen untereinander kommunizieren und Informationen über den aktuellen Entwicklungsstand des Produktes miteinander austauschen, da sich die einzelnen Produktkomponenten gegenseitig beeinflussen. Die Basis für diese Kommunikation stellen dreidimensionale CAD-Modelle der einzelnen Komponenten dar, die in Form eines Digitalen Fahrzeugmodells, dem DMU, existieren.

Die Leitungen in einem Fahrzeug spielen dabei insofern eine besondere Rolle, als daß sie sich teilweise über große Bereiche des Produktes erstrecken und somit mit einer Vielzahl von Einzelkomponenten in enger Wechselwirkung stehen. Diese bestimmen direkt die Befestigungspunkte der Leitungen sowohl an den Leitungsenden, wie auch über den gesamten Leitungsverlauf hinweg. Weiterhin sind die Leitungen auch indirekt von den Einzelkomponenten abhängig, da diese den für die Leitungen zur Verfügung stehenden Bauraum bestimmen.

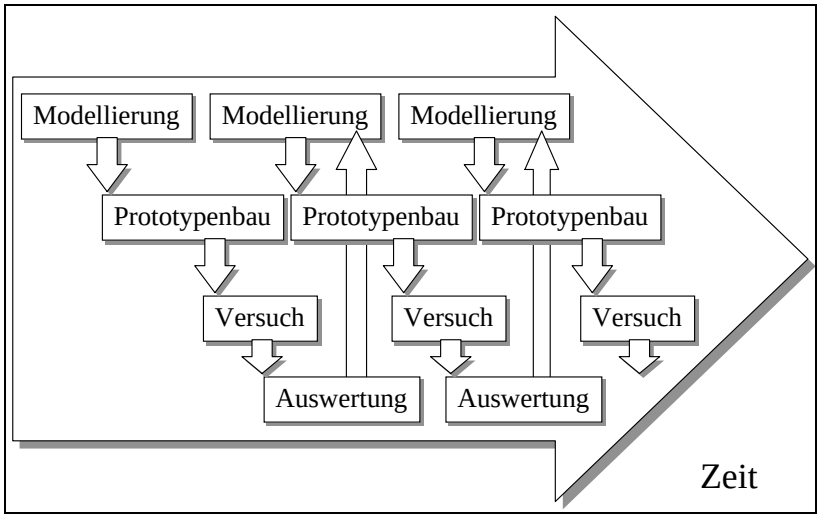


Abbildung 6: Entwicklungsschleifen

Die Festlegung des Leitungsverlaufes mit Hilfe von CAD-Systemen erweist sich aus verschiedenen Gründen als schwierig. Der Verlauf und das Aussehen der Leitungen hängt nicht nur vom zur Verfügung stehenden Bauraum ab, sondern auch von der Funktionalität und vom Material der Leitung. Die Funktionalität bestimmt das zu verwendende Material und die geometrischen Eigenschaften wie die Länge, die Querschnittsform und die Querschnittsfläche der Leitung. Das verwendete Material hat wiederum Einfluß auf den Verlauf der Leitungen aufgrund seiner physikalischen Eigenschaften. Diese physikalischen Eigenschaften werden in einem CAD-System nicht simuliert. Daher können nur Annahmen des Konstrukteurs modelliert werden, die sich auf seine Erfahrungen stützen. Um diesen Konstruktionsprozeß abzusichern, müssen die konstruierten Leitungen am physikalischen Modell getestet und gegebenenfalls geändert werden. Das kann zu mehreren Änderungsschleifen führen.

Vereinfacht dargestellt sieht ein solcher Konstruktionsprozeß also entsprechend Abbildung 6 aus.

Die Festlegung des Leitungsverlaufes, d.h. die Modellierung der Leitung, wird weiterhin durch die Bedienmetaphern der CAD-Systeme erschwert. Als CAD-Hauptnachteile werden, ebenso wie bei den durchgeführten Interviews, in der Literatur das Visualisierungs- und das Manipulationsdefizit genannt. Die Visualisierung räumlicher Objekte beruht auf einer reinen 2D-Technologie. Die 3D-Objekte werden zum größten Teil mit Hilfsmitteln wie Maus oder Digitalisiertablett modelliert und verändert. Es handelt sich hierbei um Eingabegeräte, die für den 2D-Bereich ausgelegt sind. Die Bedienmetaphern eingesetzter CAD-Systeme orientieren sich an den Datenstrukturen, die im CAD-System verwendet werden und nicht an den Objekten, die entworfen werden sollen[18]. Das führt speziell bei den Leitungsobjekten zu einem enormen Aufwand, der für die Modellierung in einer CAD-Umgebung betrieben werden muß.

Aufgrund der dargestellten Probleme werden Leitungsobjekte erst sehr spät im CAD-System modelliert, teilweise wird ihr endgültiger Verlauf erst am physikalischen Prototypen festgelegt. Die Modellierung im CAD-System ist mit einem großen Zeitaufwand verbunden, da sich der Prozeß der Bauraumfestlegung als relativ kompliziert erweist.

Wie kann der in diesem Abschnitt dargestellte Prozeß der Leitungsverlegung durch ein VR-System unterstützt werden, welche funktionalen Anforderungen werden an das System gestellt?

Das allgemeine Ziel des zu entwickelnden Systems muß es sein, den Prozeß der Leitungsverlegung hinsichtlich des Zeit- und Interaktionsaufwandes positiv zu beeinflussen. Es soll den Konstrukteur (im folgenden Anwender genannt) bei seiner Tätigkeit vor allem während der Modellierungsphase unterstützen. Hier zeigen sich Schwächen, die durch die eingesetzten CAD-Systeme nicht so einfach gelöst werden können. Der Schwerpunkt der Arbeit besteht daher in der Verbesserung der Mensch-Maschine-Kommunikation für den Prozeß der Leitungsverlegung am digitalen Prototypen.

Das System soll den Anwender bei der Bauraumfindung unterstützen. Das Finden des Bauraumes ist Voraussetzung für jede Leitungsverlegung. Dazu ist eine schnelle, dreidimensionale, stereoskopische Visualisierung der Daten notwendig, aus der einfach und damit schnell Rückschlüsse auf den zur Verfügung stehenden Bauraum gezogen werden können. Darauf aufbauend soll es möglich sein, den Verlauf der zu modellierenden Leitung direkt im dreidimensionalen Raum festlegen zu können. So kann auf die umständliche Erzeugung von Schnittebenen verzichtet werden. Der festgelegte Leitungsverlauf soll sofort visuell kontrollierbar und gegebenenfalls veränderbar sein. Eine in einem solchen VR-System erzeugte Leitung muß exportierbar sein, damit sie in einem CAD-System weiter bearbeitet und für die gesamte CAx-Umgebung transparent gemacht werden kann. Ziel ist daher eine Schnittstelle zwischen der CAD-Welt und der VR-basierten Anwendung in beiden Richtungen.

[18] [BüReiBi94]

Ein wesentlicher Unterschied zwischen den CAD-Systemen und dem VR-basierten Zielsystem soll im Definitionsbereich der möglichen Operationen liegen. Im CAD-System wird auf der mathematischen Beschreibung der modellierten Objekte mit den verschiedensten Operationen gearbeitet. So kann eine Leitung beispielsweise aus einer Aneinanderreihung einzelner Kurven bestehen. Soll der Verlauf dieser Leitung verändert werden, so müssen alle einzelnen Teilobjekte verändert werden. In der Zielanwendung sollte diese mathematische Beschreibung der Objekte dem Anwender zum größten Teil verborgen bleiben. Das System sollte Leitungen als solche erkennen und die gewünschten Operationen auch darauf anwenden. Diese Problematik stellt auch eine Voraussetzung für die Simulation der physikalischen Eigenschaften der modellierten Objekte dar. Wird bei diesem System die Physik der erstellten Leitungen noch nicht berücksichtigt, so soll doch beim Systemdesign dafür gesorgt werden, daß solch eine Simulation im Falle einer Erweiterung des Systems möglich ist. Damit wird der Weg freigehalten für die Implementierung von Montagesimulationen in der gleichen Umgebung wie auch für die Festlegung des Verlaufes von plastisch verformbaren Leitungen, deren Verlauf wesentlich durch die physikalischen Eigenschaften der Leitungen selbst beeinflußt wird.

Für die Anwendung sind verschiedenste Eingabe- und Ausgabegeräte unter dem Gesichtspunkt ihrer Eignung zu untersuchen und zu bewerten.

2.3. Leitungsverlegung und CAD-Systeme

Mit Hilfe der Systemanalyse konnte festgestellt werden, daß im Fall der Leitungskonstruktion die 3D-CAD-Durchdringung gerade zu einem frühen Zeitpunkt des Entwicklungsprozesses sehr gering ausfällt. Dabei spielt der notwendige Aufwand für die Verlegung von Leitungen in einem CAD-System eine nicht unerhebliche Rolle. Worin ist dieser Aufwand begründet?

Ein CAD-System ist ein technisches System, das der Anwender als Werkzeug für die Erfüllung der ihm gestellten Aufgaben nutzt. Im Falle der Leitungsverlegung bedeutet das, den Bauraum für eine zu konstruierende Leitung festzulegen und innerhalb dieses Bauraumes diese Leitung zu modellieren, ihre Geometrie zu beschreiben. Dabei muß der Konstrukteur alle ihm bekannten Randbedingungen beachten, die für die Konstruktion einer Leitung notwendig sind.

Diese Aufgabenbeschreibung beinhaltet eine klare Zielstellung. Diese Zielstellung wurde in einer für den Anwender verständlichen Sprache ausgedrückt. Damit hat der Anwender eine Vorstellung, ein Konzept von der Aufgabe, worauf er eine Lösungsstrategie aufbaut, sich Teilaufgaben stellt und so versucht diese Aufgabe zu lösen. Im Interaktionsmodell nach Norman[19], welches durch Abowd und Beale erweitert wurde, wird diese Sprache als die *task language* oder auch *user language* bezeichnet. Dieser steht die *system* oder *core language* gegenüber. Diese beschreibt die Datenstrukturen und Systemstati, mit

[19] [DIX93] S.92-98

deren Hilfe die systeminterne Abbildung des Anwendungsfeldes beschrieben werden kann. In CAD-Systemen beruht das Basiskonzept der *core language* auf verschiedenen mathematischen Primitiven, aus denen die darzustellenden Objekte zusammengesetzt sind. Diese mathematische Beschreibung der Objekte kann zwischen den einzelnen Fachleuten über Standardschnittstellen ausgetauscht werden und dient so als Grundlage für die Entwicklung neuer Produkte.

Der Kommunikation zwischen dem Anwender und dem technischen System dient die Mensch-Maschine-Schnittstelle. Dabei stehen im Fall des CAD-Systems mehrere Ein- und Ausgabegeräte zur Verfügung. Es handelt sich im Fall der Eingabegeräte im wesentlichen um Tastatur, Maus, Digitalisiertablett und ähnliche zweidimensionale Geräte, im Fall der Ausgabegeräte kommen Monitor, Plotter und Lautsprecher in Betracht. Mit Hilfe dieser Geräte wird der Anwender in die Lage versetzt, mit dem System zu kommunizieren[20].

Die Interaktionen mit einem System unterteilen sich nach Norman in zwei Hauptphasen, die Ausführungs- *(execution)* und die Auswertungsphase *(evaluation)*. Diese werden wiederum in sieben Teilphasen untergliedert[21]:

1. Festlegung des Interaktionszieles
2. Auswählen einer spezifischen Aktion, die zur Erreichung des Zieles notwendig ist
3. Spezifizierung einer Interaktionsabfolge
4. Durchführen dieser Aktion
5. Systemstatus feststellen
6. Systemstatus interpretieren
7. Auswerten des Systemstatus in Hinblick auf das Ziel und die im Punkt zwei definierte spezielle Aktion

Norman hat hierbei zwei Begriffe eingeführt, mit deren Hilfe er die Probleme beschrieben hat, mit denen der Anwender bei der Nutzung eines Systems konfrontiert werden kann. Es handelt sich dabei um die Lücke oder Kluft in der Ausführungsphase *(gulf of execution)* und die Kluft in der Auswertungsphase *(gulf of evaluation)*. Der *gulf of execution* beschreibt den Abstand, der zwischen den gesteckten Zielen des Anwenders und den erlaubten Aktionen des Systems bestehen kann. Der *gulf of evaluation* beschreibt den Abstand, der zwischen den Erwartungen des Anwenders und der Präsentation des internen Systemstatus besteht.

Das oben erwähnte Visualisierungs- und Manipulationsdefizit der CAD-Systeme kann mit Hilfe dieser Sichtweise ebenfalls beschrieben werden. Das

[20] [BüReiBi94]

[21] [DIX93] S.93ff.

Visualisierungsdefizit beschreibt hierbei den *gulf of evaluation*, das Manipulationsdefizit hingegen den *gulf of execution*.

Visualisierungsdefizit:

Die CAD-Systeme basieren auf einer parametrischen Beschreibung einzelner mathematischer Entitäten. Um diese zu visualisieren müssen sie gerendert[22] werden. Dieser Rendering-Prozeß unterteilt sich in zwei grundlegende Schritte. Zunächst werden die Objekte in einzelne Facetten[23] unterteilt. Diese Facetten bestehen aus ebenen Polygonen oder aus einzelnen Liniensegmenten, durch die die exakte Geometrie approximiert wird. Diese Objekte werden auf dem Ausgabegerät dargestellt. Hierbei handelt es sich um eine Projektion dieser dreidimensionalen Objekte auf eine ebenen Fläche. Dabei wird oft auf eine Drahtgitterdarstellung zurückgegriffen, da diese Darstellungsform auf den genutzten Workstations schneller visualisierbar ist als eine Flächendarstellung. Außerdem wird in dieser Drahtgitterdarstellung auch interaktiv gearbeitet, Geometrie erzeugt und verändert.

Für die Darstellung gibt es verschiedene Modi. Die Modelle können in den drei Standard-Ebenen, die durch die Koordinatenachsen des Weltkoordinatensystems gegeben sind, visualisiert werden. Dabei kann das Objekt aus der X- der Y- und der Z-Richtung betrachtet werden. Auf der Grundlage dieser einzelnen Ansichten kann sich der Anwender ein dreidimensionales Modell des dargestellten Objektes vorstellen und es bewerten. Über diese drei Projektionsebenen hinaus kann der Anwender sich auch eigene Projektionsebenen definieren und speichern, um schnell zwischen ihnen wechseln zu können.

Eine weitere Darstellungsform ist die Projektion von einem beliebigen Punkt im Raum aus. Dieser kann mit Hilfe der Maus und/oder Tastatur festgelegt werden. Auch hierbei wird das Modell auf eine ebene Fläche projiziert. Problematisch ist dabei der Tiefeneindruck, der aus dieser Darstellung gewonnen werden kann. Dieser kann durch eine Bewegung des Modells (z.B. Rotation um einen Punkt) wesentlich verbessert werden. Dazu ist eine schnelle Darstellung notwendig, damit das Modell interaktiv bewegt werden kann. Aus der wahrgenommenen Bewegungsparallaxe kann der Anwender hierbei einen Tiefeneindruck bekommen und somit Abstände und auch gegenseitige Verdeckungen einzelner Objekte bewerten.

Eine weitere Form der Darstellung ist die Nutzung von Schnitten. Durch ein Modell wird eine Schnittebene geschoben. Die Objekte, die sich mit der Schnittebene schneiden, werden dargestellt. Mit Hilfe dieser Hilfskonstruktion können Abstände bezüglich vordefinierter Ebenen exakt wahrgenommen werden. Auch hier fügt der geübte Konstrukteur die einzelnen Darstellungen in seiner Vorstellung zu einem dreidimensionalen Modell zusammen und gewinnt so Informationen über dessen räumlichen Aufbau.

[22] Rendering: Prozeß zur Visualisierung digitaler Modelle auf einem Rasterbildschirm

[23] Facetten: Linien, ebene Polygone, meist Dreiecke

Um einen gesicherten Gesamtüberblick eines CAD-Modells zu erhalten, ist der Anwender gezwungen, verschiedenste Darstellungsmodi zu nutzen und auch zwischen diesen ständig zu wechseln. Aus diesen einzelnen zweidimensionalen Darstellungen fügt er mit Hilfe seines dreidimensionalen Vorstellungsvermögens ein Modell zusammen, aus dem er seine Hypothesen und Schlußfolgerungen zieht, die die Grundlage für seine weiteren Interaktionen darstellen.

Manipulationsdefizit:

Aufgrund der zweidimensionalen Ein- und Ausgabegeräte sind auch die Manipulationen auf den dreidimensionalen Objekten im wesentlichen auf lediglich zwei Dimensionen beschränkt. Um eine Kurve im Raum festzulegen, müssen die Punkte definiert werden, durch die diese Kurve verlaufen soll. Eine Möglichkeit besteht in der direkten Eingabe der dreidimensionalen Koordinaten über die Tastatur. Der Anwender stellt sich ein Modell aber nicht in absoluten Koordinaten vor. Demzufolge extrapoliert er diese Koordinaten aus vorhandenem Wissen. Er kann beispielsweise einen vorhandenen Punkt mit der Maus selektieren und sich dessen Koordinaten anzeigen lassen. Darauf aufbauend kann er die Koordinaten des zu erzeugenden Punktes abschätzen und eingeben. Eine andere Methode für die Festlegung von Punktkoordinaten im Raum besteht in der Arbeit auf einer Arbeitsebene. Dazu wird eine Ebene im Raum definiert, die als Arbeitsebene festgelegt wird. Alle Eingaben mit einem zweidimensionalen Eingabegerät werden dann auf diese Arbeitsebene projiziert. Sollen mehrere Punkte festgelegt oder verändert werden, die nicht auf einer Ebene liegen, so ist der Anwender des Systems gezwungen, ständig zwischen verschiedenen Arbeitsebenen zu wechseln.

Eine weitere Problematik besteht in der Arbeit auf einzelnen mathematischen Strukturen. Um ein CAD-System effektiv anwenden zu können, muß ein Konstrukteur die zugrunde liegenden Datenstrukturen und ihre Eigenschaften beurteilen und bewerten können. Auf diesen einzelnen Entitäten arbeiten die verschiedensten zur Verfügung stehenden Operationen der Systeme. Mit Hilfe der einzelnen Entitäten und Operationen kann der Anwender Daten erzeugen, die das zu konstruierende Objekt repräsentieren. Dabei existieren im System aber keinerlei Informationen darüber, was die Daten eigentlich darstellen. Die Daten können zwar zu verschiedenen Mengen zusammengefaßt und auch dementsprechend abgelegt werden. Für diese Strukturierung ist ausschließlich der Anwender verantwortlich. Darüber hinaus hat sie keinen Einfluß auf die Operationen, die auf den einzelnen Entitäten ausgeführt werden. Es wird auf einer low-level-Ebene der CAD-Entitäten gearbeitet. Es bestehen keine funktionalen Abhängigkeiten zwischen einzelnen Datenmengen im System. Ein Beispiel:

Wird eine Befestigungsschelle in einer CAD-Umgebung versetzt, so muß der Konstrukteur dafür sorgen, daß die damit verbundenen direkten Folgen für das übrige Modell berücksichtigt werden. Das mit dieser Befestigungsschelle verbundene Objekt muß ebenfalls verändert werden, so daß es durch die Schelle weiterhin gehalten wird. Diese Änderung wiederum unterteilt sich in einzelne Operationen, die auf der Menge der CAD-Entitäten arbeiten und nicht auf dem

logischen Objekt, das sie repräsentieren. Dieses Fehlen funktionaler Informationen, aus denen ein solches System Schlußfolgerungen ziehen und so den Anwender eventuell entlasten könnte, ist ein wesentlicher Grund dafür, warum eine einfache, kleine Änderung an einem CAD-Modell zu einem großen Änderungsaufwand führen kann. Auch dieser Punkt führt zu einem erhöhten *gulf of execution* nach Norman.

3. VR-Systeme

Das Kapitel 3 gibt einen Überblick zum aktuellen Stand der VR-Technik. Dazu ist eine einheitliche Begriffsdefinition erforderlich. Weiterhin wird der Aufbau eines VR-Systems sowohl von der Software- wie auch von der Hardwareseite her näher erläutert. Darüber hinaus wird auf die einzelnen Ein- und Ausgabegeräte näher eingegangen. Realisierbare Interaktionen und deren Umsetzung werden anhand des VR-Systems Virtual Design II diskutiert[24]. Ausgehend vom aktuellen Stand der Technik werden die in Kapitel 2 erläuterten Anforderungen an ein VR-System zur Leitungsverlegung mit den Möglichkeiten eines VR-Systems verglichen. Ergebnis dieses Vergleiches ist ein Umfang von Funktionalitäten, der in dem System implementiert und getestet werden muß.

3.1. Begriffsbestimmung „Virtual Reality“

Virtual Reality (VR), virtuelle Realität , Cyberspace, Virtual Environment (VE), virtuelle Umgebungen (VU) und viele andere Begriffe stehen für ein Teilgebiet der Computergrafik, für eine neuartige Mensch-Maschine-Kommunikation, die seit Ende der achtziger Jahre immer mehr an Bedeutung gewinnt.

In dieser Arbeit soll der Begriff "Virtual Reality", kurz VR, angewandt werden, da er auch in der Literatur primär zu finden ist.

Erst seit Anfang der neunziger Jahre wurde das VR-Thema für breitere Anwenderkreise richtig interessant. Gründe dafür sind in dem rapiden Preisverfall für Grafikhardware mit einhergehender Performance-Steigerung zu finden. Es ist damit zu rechnen, daß dieser Trend auch weiterhin anhält und somit der Kreis der potentiellen Anwender ständig wachsen wird.

So breit gefächert wie die Forschungsrichtungen auf dem VR-Gebiet sind, so viele verschiedene Definitionen sind auch für die virtuelle Realität zu finden. Der Grund dafür liegt in der Vielfalt der Themengebiete, die in den Oberbegriff Virtual Reality einfließen.

So beschäftigt man sich in Teilbereichen der VR-Forschung verstärkt mit der Simulation. Das Spektrum reicht hierbei von der Simulation relativ einfacher physikalischer Größen wie der Gravitation, über die Simulation von Bewegungsabläufen z.B. in der Robotersteuerung bis hin zu solch komplexen Zusammenhängen wie der Untersuchung aerodynamischer Eigenschaften von Objekten mit Hilfe eines virtuellen Windtunnels. Hierbei muß grundlegend zwischen zwei Arten von Simulationen unterschieden werden. Die beiden ersten Simulationen sind relativ einfache Berechnungsverfahren, die auch in einer Echtzeitumgebung gerechnet werden können. Berechnungen einer Strömungssimulation oder eines virtuellen Fahrzugcrashes hingegen sind auch

[24] Das VR-System Virtual Design II wurde am Fraunhofer-Institut für Graphische Datenverarbeitung in Darmstadt (IGD) entwickelt. Bei diesem System handelt es sich um das Basissystem, auf dem die in dieser Arbeit beschriebenen Funktionalitäten und Erweiterungen aufsetzen. Die Entscheidung für den Einsatz dieses Systems wurde unabhängig von dieser Arbeit seitens der Volkswagen AG getroffen.

heute noch nicht unter Echtzeitanforderungen berechenbar. Es handelt sich bei solchen Simulationen in einer VR-Umgebung eher um ein Postprozessing vorberechneter Daten.

In den meisten Forschungs- und Anwendungsgebieten legt man besonderen Wert auf die Dreidimensionalität, die im visuellen Bereich, aber auch in der Interaktion mit den dreidimensionalen grafischen Objekten zum Ausdruck kommt. *„VR, or VE , provide fully three-dimensional interface for both the display and control of interactive computer graphics.“*[25] Architekturanwendungen, in denen besonderer Wert auf das Betrachten und Begehen von Bauwerken gelegt wird, stellen ein Beispiel für die als „*walk through*“ bezeichneten Anwendungen dar.

Von Virtual Reality wird bei Telepresence-Anwendungen gesprochen, die mit VR-Komponenten wie Datenhelm (head-mounted display) und Datenhandschuh (data glove) gekoppelt sind. Hierbei werden nicht ausschließlich computergenerierte Bilder betrachtet. Es kommen auch Videoaufnahmen zum Einsatz, die an einem entfernt liegenden Ort aufgenommen werden, beispielsweise in verseuchtem Gelände. Diese Videoaufnahmen werden dann der interagierenden Person im Datenhelm eingeblendet, so daß sie mehr oder weniger den Eindruck bekommt, sie befände sich direkt in der dargestellten Umgebung.[26]

Es ist nicht trivial, eine allgemeingültige Definition für den Begriff Virtual Reality anzugeben. Eine in der Literatur oft angewandte Definition geht davon aus, daß es sich bei VR um eine neuartige Schnittstelle zwischen Mensch und Maschine handelt. „*VR: a combination of various interface technologies that enables a user to intuitively interact with an immersive and dynamic computergenerated environment.“*[27] „*VR is a set of computer technologies wich, when combined, provide an interface to the computer with which the user can believe he or she is actually in a computer-generated world.“*[28] Eng damit verknüpft wird der Begriff der Interaktivität „*VE is an interactive simulation of real or imaginary world.“*[29] „*VR is an interactive, not passive, experience.“*[30] Einem Anwender dieser Schnittstelle wird das Gefühl vermittelt, er befände sich innerhalb einer virtuellen Umgebung. Er kann über die VR-Schnittstelle mit Objekten der Virtuellen Umgebung direkt interagieren. Abhängig von der realisierten VR-Schnittstelle wird dem Anwender das Gefühl vermittelt, er befände sich direkt in der virtuellen Umgebung, er ist in die virtuelle Umgebung "eingetaucht". Der Fachbegriff der "Immersion" steht für diese Eigenschaft eines VR-Systems. Der Immersionsgrad ist ein Maß dafür, wie tief man in die virtuelle Umgebung eingetaucht ist. Anders ausgedrückt gibt der Immersions-

[25] [ComGr6/93] S. 679 Steven Bryson (Zitat)

[26] [ComGr6/93] S.689

[27] [Spec10/93] John A. Adam S.24

[28] [WarBu94]S.29 Pierre du Pont (Division Limited)

[29] [ComGr6/93] S.663 Christer Carlsson, Olof Hagsand

[30] [Com2/93] Randy Pausch, University of Virginia

grad an, wie weit man sich mit seinem Bewußtsein von der Realität entfernt hat.

Definition:

Als *Virtual-Reality-System* wird im Folgenden ein technisches, computerbasiertes System mit einer Virtual-Reality-Schnittstelle bezeichnet.

Unter dem Begriff *Virtual-Reality-Schnittstelle* soll im Folgenden eine Mensch-Maschine-Schnittstelle verstanden werden, mit deren Hilfe der Mensch in die Lage versetzt wird, direkt mit den in einem Computersystem existierenden virtuellen Objekten zu interagieren. Der Anwender wird dabei in einer virtuellen Umgebung durch eine eigene Datenstruktur repräsentiert und so in diese Umgebung integriert. Bei dieser Anwenderrepräsentation handelt es sich ebenfalls um ein virtuelles Objekt.

Unter *virtuellen Objekten* werden Datenstrukturen verstanden, die in einer virtuellen Umgebung existieren und diese durch ihre Existenz beschreiben. Sie können in einer virtuellen Umgebung in unterschiedlichen Repräsentationsformen vorliegen.

Eine *virtuelle Umgebung* ist ein Datenraum, der durch eine Menge virtueller Objekte beschrieben wird.

3.2. Aufbau eines VR-Systems

Im Folgenden soll der Aufbau eines Virtual-Reality-Systems näher beschrieben werden. Es gibt die verschiedensten VR-Systeme, deren Aufbau sich durch die eingesetzte Hard- und Software unterscheidet. Wie ein solches System aufgebaut ist, ist wesentlich von der Anwendung abhängig, die mit einem solchen System bearbeitet wird. Alle VR-Systeme haben jedoch einen einheitlichen schematischen Aufbau. Es handelt sich um eine Simulationsschleife, in die der Mensch als Anwender integriert ist (Abbildung 7).

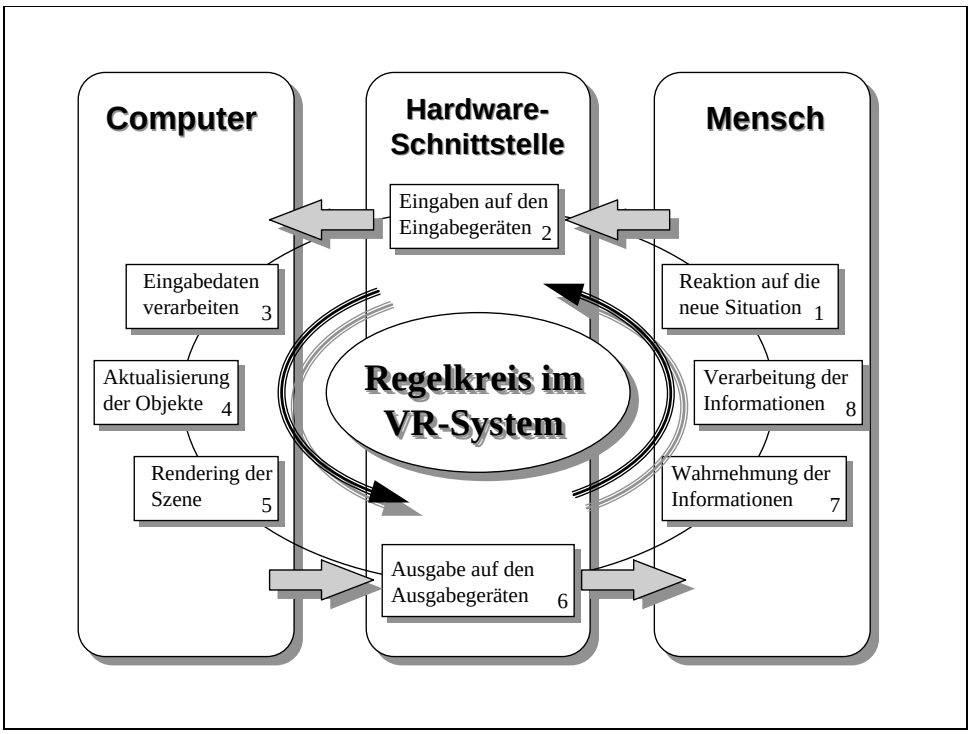


Abbildung 7: Schematischer Aufbau eines VR-Systems

Bei näherer Betrachtung fällt auf, daß sich der schematische Aufbau eines VR-Systems vom schematischen Aufbau eines allgemeinen interaktiven Systems nicht unterscheidet. Ein solches interaktives System ist nach Abowd und Beale[32] in vier Hauptkomponenten zu unterteilen (Abbildung 8):

- Anwender (U)
- System (S)
- Eingabekanal (I)
- Ausgabekanal (O)

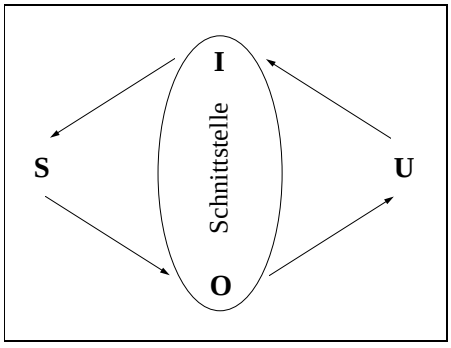


Abbildung 8: Interaktives System[31]

Diese vier Hauptkomponenten können, wie in Abbildung 9 dargestellt, den einzelnen Komponenten des VR-Systems zugeordnet werden. Der Unterschied zwischen einem herkömmlichen interaktiven System und einem VR-System besteht also nicht im grundsätzlichen Systemaufbau, sondern in den realisierten Ein- und Ausgabemetaphern, die mit Hilfe spezieller Schnittstellenhardware realisiert und umgesetzt werden.

[31] [DIX93]

[32] [DIX93] S.94ff.

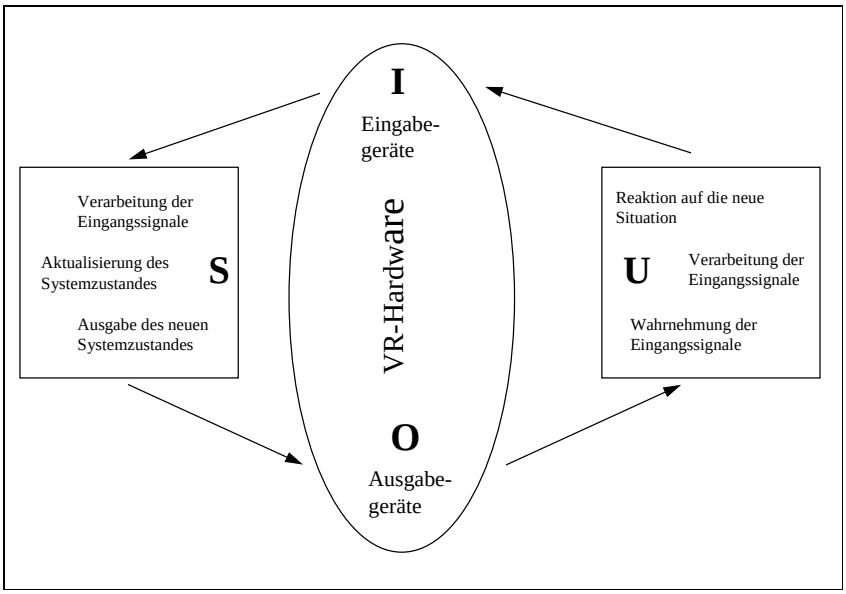


Abbildung 9: VR-System als Interaktives System nach Abowd und Beale

Die Grundidee eines VR-Systems besteht darin, einem Anwender die Möglichkeit zu geben, sich in einer computergenerierten Umgebung so zu verhalten und mit den virtuellen Objekten so zu interagieren, wie er es auch aus seiner realen Umgebung gewohnt ist. Daraus ergibt sich automatisch die Notwendigkeit der Echtzeitanforderung an ein VR-System. Die Integration des Anwenders in einem VR-System mit der Möglichkeit der freien Bewegung in dieser künstlichen Umgebung setzt voraus, daß eine Bewegung des Kopfes oder der Hand eine sofortige Aktualisierung des Systems nach sich zieht. Bei der Echtzeitanforderung handelt es sich um eine Eigenschaft, die bei allen VR-Systemen vorhanden sein muß und durch die sich diese Systeme von den restlichen interaktiven Systemen unterscheiden.

Bei der vollständigen, realitätsnahen Interaktion in einer virtuellen Umgebung handelt es sich um den Idealfall, der mit der heutigen Technik noch nicht erreicht wird. Solange dieses Ziel nicht erreicht ist, wird mit der vorhandenen Technik versucht, sich auf einen Teil der realen Welt zu beschränken, der in solch einem System in seinem Verhalten und seinen Eigenschaften nachgebildet wird.[33]

Für die Realisierung dieses Zieles kommen die verschiedensten Ein- und Ausgabegeräte zur Anwendung, die die Schnittstelle zwischen dem technischen System und dem Menschen vollkommen neu gestalten. Im Folgenden werden diese Geräte und deren Integration in ein VR-System näher betrachtet. Es wird dabei zwischen Ausgabe- und Eingabekanal des technischen Systems unterschieden (Abschnitte 3.2.1. und 3.2.2.). In den Abschnitten 3.2.3. und 3.2.4. wird anschließend auf das Geometrie- und Interaktionsmodell eingegangen. Diese Modelle werden am Beispiel des VR-Systems Virtual Design II erläutert.

[33] [KALAW93]

3.2.1. Der Ausgabekanal

Eine perfekte Illusion für den Menschen wird dann erreicht, wenn seine Sinnesorgane durch ein VR-System so perfekt stimuliert werden, daß er den Unterschied zwischen der Realität und der virtuellen Welt nicht wahrnimmt[34]. Um ihn von seiner Umgebung abzugrenzen, müssen die Sinnesorgane durch das technische System stimuliert werden. Diese Stimulation erfolgt über den Ausgabekanal der VR-Schnittstelle.

Während die Entwicklung von Geräten für die Bildausgabe, den Audio-Kanal und teilweise auch für kraftvermittelnde Geräte zum gegenwärtigen Zeitpunkt stark vorangetrieben wird, existieren kaum Veröffentlichungen zur Stimulation der übrigen Sinne des Menschen mit Hilfe von VR-Ausgabegeräten.

3.2.1.1. Der Visuelle Kanal

Das leistungsfähigste Wahrnehmungsorgan des Menschen ist das Auge[35]. Mit Hilfe des visuellen Systems nimmt der Mensch seine Umgebung räumlich wahr. Über die Ausgabegeräte für den visuellen Kanal wird ihm der Systemzustand der virtuellen Umgebung präsentiert. Dabei wird versucht, die virtuellen Objekte so realitätsnah wie möglich darzustellen. Dazu wird in den meisten Fällen eine stereoskopische Darstellung der Objekte erzeugt, wofür die Berechnung von jeweils einem Bild für jedes Auge notwendig ist. Über einen Datenhelm werden die Bilder dem entsprechenden Auge direkt zugeführt. Auf Monitoren oder Projektionsflächen werden diese Bilder auf einer Fläche dargestellt, so daß sie für jedes einzelne Auge wieder voneinander getrennt werden müssen. Für diese Trennung stehen zwei grundlegende Verfahren, das aktive[36] und das passive[37], zur Verfügung.

Um eine fließende Bewegung der Bilder zu erreichen, sind Bildraten von >= 25 Bildern/s notwendig. Damit stehen für die Berechnung eines Bildes maximal 40ms zur Verfügung. Weiterhin sind eine hohe Bildauflösung, ein hoher Kontrast wie auch ein ausreichend großes Sichtfeld notwendig.[38]

Bei den Bildausgabegeräten kann zwischen drei grundlegenden Arten, den *Monitoren*, den *Projektionssystemen* und den *Datenhelmen*, unterschieden werden.

Monitore können für eine VR-Anwendung eingesetzt werden. Hierbei wird noch zwischen den Monitoren unterschieden, die ein Stereosignal darstellen können, und denen, die dazu nicht in der Lage sind. Der Hauptnachteil der

---

[34] [KALAW93]

[35] [KALAW93]

[36] Beim *aktiven* Verfahren werden die Bilder sequentiell in schnellem Wechsel auf der Projektionsfläche dargestellt. Die Trennung dieser beiden Bilder erfolgt über eine LCD-Shutterbrille, die synchronisiert mit dem Bild ständig ein Glas abdunkelt und ein Glas öffnet.

[37] Beim *passiven* Verfahren werden beide Bilder zeitgleich dargestellt, allerdings mit Hilfe von polarisiertem Licht. Eine einfache Polarisationsbrille trennt beide Bilder wieder auf.

[38] [KALAW93] S.44ff.

Monitore ist das geringe Sichtfeld. Weiterhin kann der Anwender seinen Kopf nicht uneingeschränkt bewegen. Der Monitor wird mit seiner Kopfbewegung nicht mitgeführt.

Die *Projektionssysteme* werden in die Einseiten- und Mehrseitensysteme unterteilt. Die *Einseitensysteme* zeichnen sich durch eine hohe Bildqualität aus. Je nach Größe der Systeme kann auch ein relativ großes Sichtfeld erzielt werden. Ihr Nachteil liegt in der festen Installation. Sie werden mit der Bewegung des Anwenders ebenfalls, wie bei den Monitoren, nicht mitgeführt. Lediglich bei den *Mehrseitensystemen* kann der Anwender in verschiedene Richtungen schauen und sieht stets ein berechnetes Bild. Diese Mehrseitenprojektionen leiden allerdings unter einer schlechteren Bildqualität, deren Ursache in der geringen Helligkeit und Auflösung liegt.

Bei den *Datenhelmen*, die mechanisch mit dem Kopf des Anwenders verbunden sind, handelt es sich um die bekanntesten, im Zusammenhang mit VR-Systemen eingesetzten, visuellen Ausgabegeräte. Hierzu zählen die mit dem Kopf fest verbundenen Datenhelme (*head mounted display*, kurz HMD) und die locker mit dem Kopf gekoppelten Systeme (*head coupled displays*)[39]. Bei diesen Geräten kann sich der Anwender relativ frei im Raum bewegen und sieht immer ein dem Blickpunkt und der Blickrichtung entsprechendes Bild. Seine Bewegungsfreiheit ist lediglich durch die Verkabelung des Datenhelms bzw. durch die Befestigungsmechanik bei den gekoppelten Systemen eingeschränkt.

Für die Nutzung eines Datenhelms in einem VR-System ist das schnelle und exakte Positionstracking unentbehrlich, über das der Rechner die jeweils aktuelle Position und Orientierung des Helms im Raum erhält. Diese Positionsdaten bilden die Grundlage für die Berechnung der im Datenhelm dargestellten Bilder.

Der visuelle Renderingprozeß basiert auf Polygonmodellen. Um die Echtzeitanforderungen an diesen Prozeß erfüllen zu können, werden dafür Hochleistungsgrafikcomputer eingesetzt. Ein Teil der dafür notwendigen Berechnungsalgorithmen ist bei diesen Maschinen in Hardware realisiert. Heutige Computersysteme bieten dafür bereits enorme Rechenleistungen an. Auf einem Grafiksubsystem einer Onyx2™ InfiniteReality™ der Firma Silicon Graphics können beispielsweise bis zu 11 Millionen Polygone in einer Sekunde dargestellt werden[40].

Für die Erzeugung der Stereobilder ist es notwendig, daß jedes Auge ein separat berechnetes Bild sieht. Um eine möglichst hohe Renderingleistung zu erhalten, können zwei solcher Grafiksubsysteme so zusammengeschaltet werden, daß auf jedem System jeweils das Bild für ein Auge berechnet wird. So erhält man mit einem solchen System eine theoretisch mögliche Szenenkomplexität von 440.000 Polygonen, die in Stereo bei einer Bildrate von 25

[39] [AsDa95]

[40] [SGI98] Bei dem angegebenen Wert handelt es sich um eine Herstellerangabe, die die reine Leistungsfähigkeit der Grafikhardware beschreibt.

Bildern/s berechnet werden können. Es sei darauf hingewiesen, daß es sich bei dem für diese Berechnung zugrunde liegenden Wert von 11 Millionen Polygonen/s um eine theoretisch mögliche obere Schranke handelt, die in der Praxis nicht erreicht wird. Vielmehr werden in der Praxis Performancewerte erzielt, die bei 25-35% der theoretisch erreichbaren Leistung liegen[41].

3.2.1.2. Der Audio-Kanal

Über den Audiokanal können dem Anwender zusätzlich zu den generierten Bildern Toninformationen vermittelt werden. Eine akustische Rückkopplung ist für die verschiedensten Informationen anwendbar. Mögliche Anwendungsfelder erstrecken sich von einfachen Systemstatusinformationen, beispielsweise eine akustische Bestätigung für einen eingegebenen Befehl, bis hin zur Simulation von dreidimensionalen Geräuschquellen in einer virtuellen Umgebung. Abhängig von der Anwendung werden die Informationen über einfache feste Lautsprecher in der Nähe des Arbeitsplatzes oder über Kopfhörer dem Anwender zugeführt.

Eine interessante Installation stellt auch ein sogenannter Akustik-Boden dar, wie er in einer Mehrseiten-Projektion bei der Gesellschaft für Mathematik und Datenverarbeitung (GMD) in Sankt Augustin angewendet wird[42]. Niederfrequente Schwingungen werden über diesen Akustikboden ausgegeben und können so den Realitätseindruck der Anwendungen verstärken.

3.2.1.3. Kraftrückkopplung

Im Gegensatz zum Sehen und Hören befindet sich die Entwicklung von Kraftrückkopplungs-Hardware noch in einem relativ frühen Stadium. In Arbeiten über die Steuerung von Robotern unter Zuhilfenahme von Kraftrückkopplungsgeräten konnte gezeigt werden, daß Informationen über diesen Ausgabekanal die Steuerungsperformance in der Anwendung erhöhen kann. Dementsprechend ist auch der Einsatz dieser Geräte in virtuellen Umgebungen vielversprechend[43].

Es wird zwischen drei grundsätzlichen Geräteklassen unterschieden, die „*force-feedback*-“, die „*tactile-feedback*-“ und die „*shape-forming-devices*“[44]. Alle drei Geräteklassen erfüllen zwei wesentliche Aufgaben. Sie werden zum exakten und schnellen Positionstracking verwendet und beeinflussen die Bewegung einzelner Körperteile (meist handelt es sich um Hand und Arm des Probanden) bzw. stimulieren den Tastsinn des Anwenders. Damit gehören die

[41] Die Graphikperformance ist stark von der Struktur der Daten, der Anzahl der Lichtquellen, dem genutzten Beleuchtungsmodell, der Modelltexturierung, dem Modellumfang, von der Anzahl genutzter Environmentmappings, von der Anzahl und Größe transparenter Polygone u.v.a. abhängig. Der angegebene Wert beschreibt die praktischen Erfahrungen, die mit vorhandenen polygonalen Modellen gesammelt wurden. Siehe auch [ZIMM98], Fig. 5, S.65

[42] [GMDPM96]

[43] [AsDa95]

[44] [ZIEG96]

Kraftrückkopplungsgeräte sowohl zu den Eingabe- wie auch zu den Ausgabegeräten.

Es wird weiterhin zwischen mobilen und feststehenden Geräten unterschieden. Bei den feststehenden Geräten handelt es sich um Auf-Tisch-Geräte oder um am Boden oder an der Decke befestigte Installationen. Die mobilen Systeme sind mit dem Anwender gekoppelt und können so mit seiner Bewegung mitgehen. Bei den meisten in der Literatur beschriebenen Geräten handelt es sich um Spezialsysteme, die für ein sehr eingeschränktes Aufgabengebiet entwikkelt wurden und auch darin angewendet werden. Sie unterscheiden sich im Aktionsradius und in der Anzahl der unterstützten Freiheitsgrade sowohl bezogen auf die Trackingaufgabe, wie auch auf die Kraftrückkopplung. Weiterhin unterscheiden sie sich in der maximal möglichen Kraft, die sie ausgeben können und auch in ihrer Genauigkeit[45].

3.2.1.4. Riechen, Schmecken, Temperaturempfinden

Über die Stimulation der hier bisher noch nicht behandelten Sinne des Menschen ist sehr wenig in der Literatur zu finden. Lediglich für Spezialsysteme gibt es Entwicklungen, mit deren Hilfe auf einfache Weise diese Sinne angesprochen werden. Die Stimulation dieser Sinne hat keinen hohen Stellenwert und vor allem ist zu vermuten, daß zur heutigen Zeit ein Gerät zur Erzeugung eines beliebigen Geruchs oder Geschmacks nicht oder nur mit einem enormen Aufwand herstellbar ist.

Um in einer Anwendung diese Informationen darzustellen, kommen in heutigen Systemen Farbverläufe, Text- oder Audioausgaben ersatzweise zur Anwendung.

3.2.2. Der Eingabekanal

Für die Integration des Anwenders in eine virtuelle Umgebung und dessen möglichst natürliche Interaktionen mit den virtuellen Objekten sind spezielle Eingabegeräte notwendig, die die Aktionen des Anwenders sensieren und dem technischen System für die Weiterverarbeitung zur Verfügung stellen.

In den folgenden Abschnitten werden die dafür zur Verfügung stehenden Geräteklassen diskutiert. Dabei kann nicht auf alle existierenden Geräte eingegangen werden, da dies den Rahmen dieser Arbeit sprengen würde. Es soll jedoch ein Überblick über die aktuell technischen Möglichkeiten gegeben werden.

3.2.2.1. Positionstracking

Für die korrekte Positionierung der digitalen Repräsentation des Anwenders in der virtuellen Umgebung ist ein exaktes und schnelles Positionstracking des Anwenders in der realen Welt notwendig. Das gilt sowohl für die Position als auch für die Orientierung im Raum. Für diese Aufgabe stehen die verschie-

[45] [IWA90] [YASU96] [SUR92] [BRO90]

densten Geräte zur Verfügung. Es wird zwischen *optischen*, *magnetischen*, *mechanischen*, *inertialen* und *akustischen* Trackern unterschieden.

Die heute gebräuchlichsten Geräte sind die *magnetischen* Trackingsysteme. Sie arbeiten mit einem künstlich erzeugten Magnetfeld, das durch Sensoren, die am Meßobjekt befestigt sind, ausgewertet wird. Das erzeugte Magnetfeld ist gegenüber externen Magnetfeldern und ferromagnetischen Metallen störempfindlich, woraus eine Erhöhung der Ungenauigkeit resultiert. Dafür treten keine Störungen in Form von Verdeckungen auf, die bei den akustischen und optischen Trackingsystemen zu Totalausfällen des Meßvorgangs führen können.

Um die Genauigkeit der magnetischen Trackingsysteme zu erhöhen, kann das Magnetfeld ausgemessen und durch Interpolationsalgorithmen entzerrt werden. So können Genauigkeiten für die Positionierung im Zentimeterbereich erzielt werden[46].

Optische Trackingverfahren basieren auf Bildverarbeitungsalgorithmen. Dazu werden aus verschiedenen bekannten Positionen im Raum Bilder aufgenommen und digital weiterverarbeitet. Aus den einzelnen 2D-Bild-Koordinaten festgelegter Meßmarken können die 3D-Positionskoordinaten berechnet werden. Für die Ermittlung der Orientierung im Raum müssen mehrere bekannte Meßmarken auf den Bildern identifiziert und zugeordnet werden können. Ein Problem der optischen Trackingverfahren liegt in der Verdeckung der Meßmarken. Diese Verdeckung kann durch die Bewegung des Anwenders auftreten und zu einem Ausfall des Trackingsystems führen.[47]

Abhängig von der Auflösung und der Anzahl der eingesetzten Kameras wie auch von der Größe des aufgenommenen Raumes, variieren die Genauigkeiten dieser Systeme. Die erzielten Genauigkeiten sind aber wesentlich höher als bei den magnetischen Verfahren. Auch eine Entzerrung ist bei diesen Systemen nicht erforderlich.[48]

Die *mechanischen Trackingverfahren* arbeiten sehr schnell und sehr genau. Bei den Datenhelmen und den Kraftrückkopplungsgeräten sind diese Trackingsysteme weit verbreitet. Der Nachteil liegt in der Mechanik selbst. Durch die Mechanik unterliegen sie einer bestimmten Massenträgheit und der Bewegungsraum ist eingeschränkt. Weiterhin können die mechanischen Systeme den Blick auf die dargestellte virtuelle Szene einschränken, wenn mit Projektionsleinwänden oder Mehrseitenprojektionen gearbeitet wird. Weiterhin kann mit einem mechanischen System oft nur ein Meßobjekt aufgenommen werden. Die Nutzung eines Systems für das Tracking des Kopfes und der Hände zugleich scheidet daher aus.

Die *akustischen Trackingsysteme* arbeiten in den meisten Fällen im Ultraschallbereich. Im Raum verteilte Mikrofone nehmen die Laufzeitunterschiede

[46] [ZACH97]

[47] [KALAW93]

[48] [MadGer96]

der ausgesendeten Signale auf um daraus die Position zu ermitteln. Wie bei den optischen Trackingsystemen sind für die Ermittlung von Position und Orientierung mehrere Meßmarken (Sender) notwendig, aus denen die Orientierung des Objektes berechnet wird.

Die *inertialen Trackingsysteme* basieren auf Verfahren, bei denen die Beschleunigung gemessen wird, um daraus Position und Orientierung zu ermitteln. Problematisch ist hierbei die Akkumulation der Meßwerte, wodurch sich der Meßfehler mit der Zeit vergrößert. Daher ist eine ständige externe Aktualisierung dieser Systeme notwendig, die wiederum mit Hilfe eines anderen Verfahrens durchgeführt werden muß. Interessant sind die Inertialsysteme beispielsweise für die Verfahren, bei denen Verdeckungen auftreten können. Mit ihrer Hilfe kann die Zeit der Verdeckung überbrückt werden.

3.2.2.2. Tracking der Hand

Für die Bewegungsmessung der einzelnen Hand- und Fingergelenke stehen Datenhandschuhe zur Verfügung, die mit geeigneter Sensorik die Position von Handfläche und Fingern ermitteln. Dazu werden die Krümmungen der einzelnen Gelenke gemessen. Diese werden auf einen festen Wertebereich abgebildet, der vom VR-System während der Anwendung ausgewertet werden kann. Im Fall des in dieser Arbeit eingesetzten CyberGlove™ stehen die Werte für insgesamt 18 Gelenke zur Verfügung, die zwischen 0 und 255 liegen können. Die Gelenkkrümmung wird über Dehnmeßstreifen ermittelt. Es handelt sich bei der Messung nicht um absolute Winkelmaße sondern um eine lineare Abbildung zwischen Krümmung und Wertebereich. Daher arbeiten die Datenhandschuhe auch körperabhängig und müssen für jeden Anwender kalibriert werden[49].

Die Meßwerte des Datenhandschuhs können sowohl für eine Gestenerkennung eingesetzt werden als auch direkt das Modell der virtuellen Hand in der VR-Umgebung steuern.

Die Vorteile des Datenhandschuhs liegen in der Geschwindigkeit der Datenerfassung. Weiterhin treten durch das Meßverfahren bedingt keine Verdekkungsprobleme auf. Die Nachteile liegen in der Verkabelung heutiger Systeme, im Tragekomfort und in der erreichten Genauigkeit.

Eine weitere Methode für die Gestenerkennung stellt das optische Tracking der Hand dar. Über Kameras wird die Hand aufgenommen und mit Hilfe von Bildverarbeitungsalgorithmen wird dieses Bild ausgewertet. Für die Bewegungssteuerung der virtuellen Hand, d.h. für die einzelnen Hand- und Fingergelenke, erscheint diese Methode ungeeignet. Weiterhin ist die Anwendung des optischen Trackings auf einen begrenzten Aktionsbereich beschränkt. Auch die von den optischen Verfahren bekannten Verdeckungsprobleme sprechen gegen den Einsatz dieser Methode.

[49] [KESS95]

3.2.2.3. Spracheingabe

Ein weiterhin wichtiges Eingabemedium ist die Sprache. Es wird zwischen zwei verschiedenen Arten der Spracheingabe unterschieden. Zum einen kann sie der *einfachen Befehlseingabe* dienen. Zum anderen kann sie genutzt werden, um *natürlichsprachlich* mit einem System zu kommunizieren.

Für die *Eingabe verschiedener Befehle* ist die Spracheingabe sehr hilfreich und relativ problemlos zu realisieren. Mit Hilfe einer solchen Befehlseingabe können Knöpfe und Tasten anderer Eingabegeräte ersetzt werden. In einer virtuellen Umgebung kann es sehr störend sein, dreidimensionale Menüs zu bedienen. In immersiven Umgebungen ist auch die Tastatur nicht zu benutzen. Praktische Erfahrungen haben gezeigt, daß die unterschiedliche Funktionsbelegung der Knöpfe an der Spacemouse oder an anderen 6D-Eingabegeräten wie auch der Tasten auf der Tastatur zur Konfusion des Anwenders führen. Es fehlt dabei der Zusammenhang zwischen physikalischem Eingabegerät und logischer Funktion[50]. Es sind verschiedenste Spracheingabesysteme auf dem Markt erhältlich. Dabei werden sprecherunabhängige und -abhängige Systeme unterschieden. Es gibt kein System, das eine Erkennungsgenauigkeit von 100% bietet. Die Weiterverarbeitung der Eingaben muß daher fehlertolerant erfolgen.

Im Vergleich zur reinen Befehlseingabe über die Sprache ist die *natürlichsprachliche Kommunikation* mit einem Grafiksystem wesentlich aufwendiger zu realisieren. Dabei ist es notwendig, die teilweise vagen qualitativen Anweisungen eines Anwenders in quantitative Veränderungen der virtuellen Umgebung umzusetzen. Ein entsprechendes System muß in der Lage sein, die sprachlichen Anweisungen auszuwerten und in einzelne Befehle/Handlungen umzusetzen. Dazu muß dem System dynamisch aktualisiertes Wissen über die virtuelle Umgebung zur Verfügung stehen, das in einer rein geometrisch basierten Umgebung nicht oder nur teilweise enthalten ist.[51]

3.2.2.4. Weitere Eingabegeräte

Mit Ausnahme der Spracheingabe handelt es sich bei den bisher betrachteten Eingabegeräten um Systeme, die die Bewegungen und Aktionen des Anwenders direkt sensieren, auswerten und entsprechend dieser Daten die virtuelle Repräsentation des Anwenders steuern. Zusätzlich zu diesen Eingabegeräten gibt es Geräte, die für die Navigation durch eine virtuelle Szene eingesetzt werden. Als Beispiele seien hier die DLR-Spacemouse, der SpaceBall und der Flying Joystick genannt. Über diese Geräte kann die Bewegung eines virtuellen Objektes in allen 6 Freiheitsgraden gesteuert werden. Darüber hinaus kommen auch Geräte zum Einsatz, mit deren Hilfe bestimmte Ereignisse ausgelöst werden können. Dazu zählen Schalter und Knöpfe an den erwähnten Navigationsgeräten genauso wie auch die Tastatur des Computers.

[50] [COLL95]

[51] [JUNG97] S.19-23

3.2.3. Das Geometriemodell

Wie im Abschnitt 3.1. bereits festgestellt wurde, ist eine virtuelle Umgebung durch eine Menge virtueller Objekte definiert. Jedes einzelne virtuelle Objekt ist dabei durch seine Eigenschaften beschrieben. Diese Eigenschaften sind in einer objektspezifischen Datenstruktur enthalten und stehen so dem VR-System zur Verfügung. Zu den objektspezifischen Eigenschaften zählen die geometrische Beschreibung des Objektes, die Position und Orientierung im Raum, seine Strukturierung wie auch spezielle Simulationsattribute. Die geometrische Beschreibung der einzelnen Objekte bildet den Hauptteil der Objektdefinition. Auf der Grundlage dieser Geometrieinformation wird dem Anwender der aktuelle Systemzustand in Form der gerenderten Szene präsentiert.

Die geometrische Beschreibung der Objekte besteht aus einer Polygonmenge. Die Polygonmodelle sind strukturiert in einer Objekthierarchie abgelegt, die im Folgenden anhand des Szenengrafen des VR-Systems Virtual Design II erläutert werden soll. Die im System enthaltene Szenenbeschreibung in Form der Objekthierarchie kann als Datei abgespeichert werden. Dafür stehen Polygonformate wie der Inventor- oder der FHS-Standard zur Verfügung[52].

In der Objekthierarchie des VR-Systems Virtual Design II wird zwischen zwei grundlegenden Knotentypen unterschieden. Der *ASSEMBLY-Knoten* erlaubt es, eine beliebige Anzahl von Kind-Objekten zu besitzen. Über diesen Knotentyp wird die gesamte hierarchische Struktur des Geometriemodells definiert. Der *GEOMETRY-Knoten* enthält die eigentliche Geometriebeschreibung in Form von Polygonmengen.

Im Virtual Design II steht für jeden Knotentyp eine 4x4-Matrix zur Verfügung, über die die lokalen Koordinaten für Position und Orientierung des entsprechenden Knotens festgelegt werden. Diese akkumulierte Matrix, bezeichnet als „M_{Accu}“, berechnet sich aus sechs einzelnen lokalen Matrizen

$$M_i \qquad i = 0,...,5$$

durch eine Matrix-Multiplikation (1). Die Position und Orientierung eines Objektes „Obj“ in der Welt, abgelegt in der Matrix „$M_{ToWorld}$“, wird rekursiv aus den Weltkoordinaten seines Vater-Objektes (Parent), multipliziert mit der eigenen lokalen Matrix „M_{Accu}“ berechnet (2). Die Weltkoordinaten der objektbeschreibenden Polygone werden durch eine Multiplikation mit der „$M_{ToWorld}$“ des entsprechenden Objektes ermittelt.

$$M_{Accu}(Obj) = M_0(Obj) \times M_1(Obj) \times M_2(Obj) \times M_3(Obj) \times M_4(Obj) \times M_5(Obj) \qquad (1)$$

$$M_{ToWorld}(Obj) = M_{ToWorld}(Parent) \times M_{Accu}(Obj) \qquad (2)$$

Über diesen Mechanismus wird sichergestellt, daß sich die Transformationen eines Knotens auf diesen selbst, wie auch auf alle seine Kindobjekte in der Objekthierarchie auswirken.

[52] [FHS98], [WERNE] S. 283 ff.

Die einzelnen Knoten der Objekthierarchie können über einen Namen identifiziert werden. Außer den beiden beschriebenen Knotentypen gibt es noch verschiedene Spezialobjekte wie Billboards, Lichtquellen, Level-Of-Detail- und Kameraobjekte, auf die hier aber nicht näher eingegangen werden soll[53].

Die geometrische Beschreibung der Objekte definiert ihr Aussehen, das dem Anwender über den visuellen Renderingprozeß präsentiert wird. Die hierarchische Strukturierung der virtuellen Umgebung beeinflußt sowohl das Simulationsverhalten wie auch die Renderingperformance des Gesamtsystems.

Der Anwender ist selbst ein Teil der virtuellen Umgebung. Im einfachsten Fall ist er durch ein Kameraobjekt definiert, welches durch seine Position und Orientierung in der virtuellen Umgebung die Position und die Blickrichtung der Augen des Anwenders repräsentiert. Je nach Anwendung kann die Repräsentation des Nutzers in der virtuellen Umgebung beliebig komplex sein. Im Virtual Design II wird für alle Navigationsmodi das Flying-Carpet-Modell für den Anwender genutzt[54]. Dieser Interaktionsmetapher liegt folgende Idee zugrunde:

Das Kameraobjekt und die geometrische Repräsentation der linken wie auch der rechten Hand bewegen sich auf einem Wagen („Cart“) durch die Szene. Das Cart kann durch die verschiedensten Eingabegeräte angesteuert werden. Während der Arbeit mit diesem System hat sich die Ansteuerung über eine DLR-Spacemouse bewährt, auch die Steuerung über Handgesten und über ein Spracheingabesystem wurde erfolgreich durchgeführt. Relativ zum „Cart“ können Kamera und Hände positioniert werden. Positionstrackingsysteme und Datenhandschuhe kommen dabei zum Einsatz.

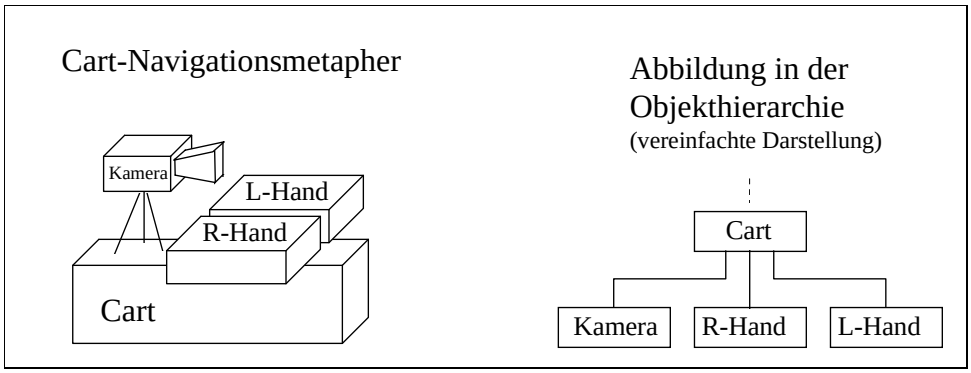


Abbildung 10: Navigationsmetapher „flying carpet“[55]

3.2.4. Das Interaktionsmodell

Für die Beschreibung der möglichen Interaktionen im Virtual Design II steht eine einfache Beschreibungssprache zur Verfügung. Sie basiert auf dem *action-event-Prinzip.* Ein bestimmtes Ereignis sorgt dafür, daß eine Aktion auf

[53] [FHS98]

[54] [ZACH96]

[55] [ZACH96]

einem oder mehreren Objekten ausgeführt wird. Mit dem Begriff „Objekt“ ist dabei ein Knoten der Objekthierarchie gemeint, der über seinen Namen referenziert wird.

Ein eine Aktion auslösendes Ereignis kann unterschiedlichen Ursprungs sein. Es wird zwischen physikalischer Eingabe (Knöpfe, Tasten, Gesten, Spracheingabe, ...), virtuellen Schaltern und geometrischen Ereignissen (Portale, Kollision zwischen zwei Objekten) unterschieden. Weiterhin können auch zeitabhängige Ereignisse definiert werden.

Eine Menge von Aktionen kann durch Ereignisse ausgelöst werden. Sie sind in [ZACH96] beschrieben und sehr vielfältig. Dazu gehören Aktionen,

- die die Visualisierungsattribute der Objekte beeinflussen,
- die eine Positionierung von Objekten erlauben („Greifen von Objekten“)
- die die Skalierung von geometrischen Objekten beeinflussen
- die Objekte ständig kopieren („sweep action“)

und viele mehr. Objekte, auf die sich spezielle Aktionen beziehen sollen, können auch zur Laufzeit über eine spezielle Selektionsaktion ausgewählt werden. Dadurch können die möglichen Interaktionen dynamisch angewendet werden.

Das gesamte Verhalten der virtuellen Szene wird über die Interaktionsbeschreibungssprache vor Anwendungsbeginn festgelegt. Dabei werden die Objekte, auf die sich die Interaktionen beziehen sollen, durch ihren Namen oder durch eine dynamische Selektion referenziert. Beide Vorgehensweisen setzen voraus, daß bei der Anwendungsvorbereitung die Objektstrukturierung im Szenengrafen bekannt ist und sich auch im Laufe der Anwendung nicht verändert. Sie ist verhaltensbestimmend, da sich die Interaktionen auf ein virtuelles Objekt und seine Kinder beziehen. Auch eine Referenzierung auf das jeweilige Vaterobjekt ist erlaubt. Es wird bei den einzelnen Interaktionen nicht zwischen verschiedenen logischen Objekten mit unterschiedlichen Funktionalitäten unterschieden, für das Interaktionsmodell sind im wesentlichen alle Objekte gleich.

Bei den Interaktionen werden physikalische Eigenschaften der Objekte lediglich in Ansätzen berücksichtigt. Ein Beispiel dafür ist die Simulation der Gravitation für einzelne Objekte. Für die Positionierung stehen einfache Aktionen zur Verfügung, die auf einer festen Verbindung zwischen der virtuellen Hand und dem gegriffenen Objekt basieren. Eine solche feste Verbindung wird dadurch erzielt, daß das zu positionierende Objekt innerhalb der Objekthierarchie die virtuelle Repräsentation des positionierenden Objektes (virtuelle Hand) als Vaterobjekt zugewiesen bekommt. Es wird so in der Objekthierarchie „umgehängt“. Eventuell auftretende Kollisionen zwischen dem gegriffenen Objekt und der Umgebung werden dabei ebensowenig berücksichtigt wie eine Durchdringung der virtuellen Hand mit dem gegriffenen Objekt. Es existiert somit keine korrigierende Einflußnahme des Systems auf die entsprechende Interaktion, wie man es eigentlich aus der Realität kennt. Zum Beispiel erfolgt

das Abstellen einer Tasse auf einem Tisch in der Realität nur durch das Zusammenspiel zwischen grober Positionierung und physikalisch basierter Wechselwirkung zwischen Tasse und Tisch. Solche Wechselwirkungen sind aber in dem beschriebenen Interaktionsmodell nicht realisiert, da es auf Matrizenoperationen basiert und keine weiterführenden Objekteigenschaften berücksichtigt.

3.3. Modellgewinnung im Entwicklungsprozeß

In diesem Abschnitt werden die technischen Abläufe erläutert, die für die Gewinnung eines VR-Modells im Entwicklungsprozeß eines komplexen Produktes notwendig sind. Die Modellgewinnung bildet die Grundlage für alle weiterführenden Anwendungen in einer virtuellen Umgebung. Da hierbei weder eine spezielle Anwendung noch eine spezielle Funktionalität betrachtet wird, liegt der Fokus auf der Gewinnung des Geometriemodells inklusive der möglichen Strukturierung der Daten.

Im Entwicklungsprozeß werden für die Konstruktion der einzelnen Baugruppen eines Produktes verschiedene CAD-Systeme eingesetzt. Voraussetzung für die Gewinnung virtueller Modelle sind flächenbeschriebene, dreidimensionale Daten, die bei der Arbeit mit den CAD-Systemen erzeugt werden. Die einzelnen Datensätze werden in einem Produktdaten-Management-System (PDM-System) verwaltet. Über das PDM-System kann auf diese Daten zugegriffen werden, sie stehen damit für die weitere Arbeit zur Verfügung.

Die CAD-Datensätze bilden die Grundlage für die Erzeugung eines Modells, das in einer VR-Anwendung verarbeitet werden kann. Die Erzeugung der Modelle erfolgt unter verschiedensten anwendungsabhängigen Randbedingungen. Sie unterscheiden sich in ihrer Auflösung, die in Relation zur Modellgenauigkeit steht. Weiterhin unterscheiden sie sich in der Strukturierung und in den Visualisierungseigenschaften.

3.3.1. Modellerzeugung

Der Austausch der CAD-Datenmodelle erfolgt über Standardschnittstellen. In der Automobilindustrie werden der IGES- und der VDA/FS-Standard häufig eingesetzt. Die einzelnen Datensätze beinhalten die mathematisch exakten Flächenbeschreibungen einzelner Objekte. Im CAD-System liegt der Anwendungsfokus auf den einzelnen CAD-Primitiven. Bestimmte Mengen dieser Primitive, zusammengefaßt in verschiedenen Datensätzen, bilden die Objekte, auf denen der Anwendungsfokus in einer VR-Anwendung liegt. Die Zusammenfassung der CAD-Primitive zu einzelnen Datensätzen bildet daher die Grundlage für die Strukturierung einer virtuellen Szene in einzelne Objekte (Abbildung 11).

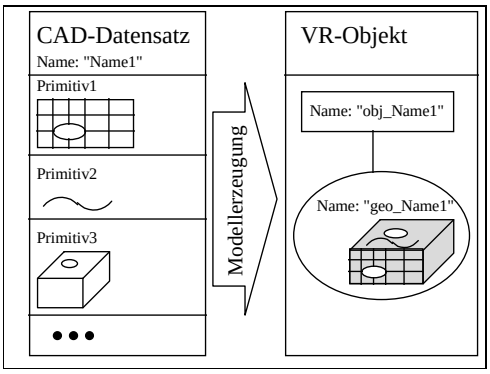


Abbildung 11: Zusammenfassung mehrerer CAD-Primitive zu einem Objekt

Aus den mathematisch exakt beschriebenen CAD-Primitiven müssen Polygonmodelle erzeugt werden (siehe Abschnitt 3.2.3.). Diese stellen eine Approximation der exakt beschriebenen CAD-Geometrie durch Dreiecksnetze dar. Für die Erzeugung dieser Modelle stehen heute verschiedenste Tesselierungswerkzeuge zur Verfügung, die teil- bis vollautomatisch diesen Prozeß unterstützen. Teilweise bieten bereits eingesetzte CAD-Systeme solche Tesselierungsoperationen an. Im einfachsten Fall kann dabei die maximale Abweichung der Tesselierungsergebnisse von der Originalgeometrie festgelegt und das Ergebnis in einem Polygonformat exportiert werden. Als Dateiformate kommen dabei Standardformate wie der Inventor- oder der VRML-Standard zum Einsatz[56].

Eine Strukturierung der so erzeugten Polygone kann auf zwei verschiedenen Methoden beruhen. Zum einen kann ein CAD-Datensatz zu einer Menge von Polygonen zusammengefaßt werden, deren Strukturierung beliebig sein kann. Eine weitere Möglichkeit besteht in der Strukturierung der Polygonmenge auf der Basis der CAD-Primitive des Ursprungsdatensatzes. Die Polygone, die aus dem gleichen CAD-Primitiv entstanden sind, werden dabei zu einer Menge zusammengefaßt. Bereits in diesem ersten Schritt wird also die später in der Anwendung zur Verfügung stehende feinste Struktureinheit festgelegt. Es handelt sich dabei um die kleinste einzeln selektierbare und für verschiedenste Interaktionen zur Verfügung stehende Geometrieeinheit. Diese ist für die Definition von einfachen Bewegungs-Constraints, für die einfache interaktive Positionierung einzelner Objekte wie auch für die Festlegung von Kollisionsuntersuchungen und vielen anderen Interaktionsdefinitionen verhaltensbestimmend.

Die so erzeugten Polygonmodelle einzelner im CAD-System modellierter Objekte werden in einem weiteren Schritt mit zusätzlichen Visualisierungsattributen versehen. Diese Visualisierungsattribute sind in sogenannten Materialdefinitionen zusammengefaßt und enthalten alle für die Visualisierung des Objektes notwendigen Attribute. Dazu gehören Farbwerte, Beleuchtungseigenschaften und Texturierungen. Es sei angemerkt, daß diese Materialdefinitionen nicht mit Materialkonstanten aus der Werkstofftechnik zu verwechseln sind.

[56] [WERNE] S. 283 ff.

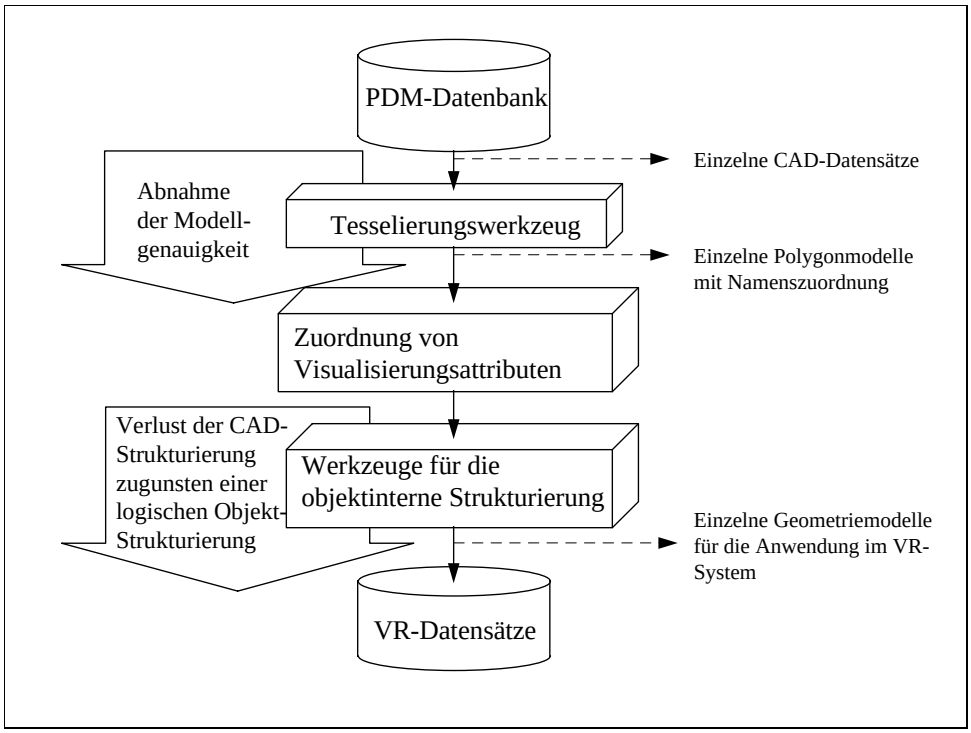


Abbildung 12: Modellerzeugung der geometrischen Objekte

Für jedes Objekt können weitere Datenvorbereitungsschritte durchgeführt werden. Dazu zählt die Erzeugung von mit unterschiedlicher Genauigkeit aufgelösten Varianten eines Objektes und deren Zusammenfassung zu einem Level-of-Detail-Objekt (LOD-Objekt). Weiterhin kann zusätzlich zur Visualisierungsgeometrie eine Kollisionsgeometrie erzeugt werden, die in der virtuellen Umgebung ausschließlich für Berechnungszwecke (Kollisionserkennung) genutzt wird. Außerdem kann die Visualisierungsgeometrie für den Renderingprozeß voroptimiert werden.

Ergebnis der Modellerzeugung sind Geometriemodelle, die einzelne logische Objekte oder Baugruppen repräsentieren, was in starkem Maße von der Strukturierung der Ursprungsdatensätze abhängig ist. Sie stehen als Polygonmodell in einer beliebigen internen Strukturierung zur Verfügung. Diese modellinterne Strukturierung beeinflußt die Leistungsfähigkeit des Gesamtsystems. Die Geometriemodelle bilden die Grundbausteine einer virtuellen Umgebung. Eine virtuelle Umgebung ist durch eine Objekthierarchie repräsentiert, wie sie in Abschnitt 3.2.3. beschrieben wurde. Diese Objekthierarchie wird in einem weiteren Anwendungsvorbereitungsschritt aus den einzelnen Geometriemodellen aufgebaut. Auf die einzelnen Geometriemodelle wie auch auf ihre Unterstrukturen kann im Rahmen der VR-Anwendung zugegriffen werden. Sie werden über ihre eindeutigen Namen referenziert. Die dabei möglichen Interaktionen werden durch die Definition des Interaktionsmodells entsprechend Abschnitt 3.2.4. vor dem Anwendungsstart festgelegt.

Bezüglich der Genauigkeit der Modellbeschreibung tritt im Rahmen des Vorbereitungsprozesses ein Informationsverlust auf, da im VR-Modell eines

Objektes die exakte mathematische Beschreibung der einzelnen Flächen und Kurven nicht mehr zur Verfügung steht. Sämtliche Berechnungen und Simulationen können ausschließlich nur auf der Polygongeometrie basieren. Bei festgelegten maximal zulässigen Fehlern während der Tesselierung kann eine Fehlerabschätzung dieser Berechnungen vorgenommen werden.

3.3.2. Informationsdifferenz zwischen technischem System und Anwender

Welche Informationen sind in der Objekthierarchie eines VR-Systems enthalten?

In der VR-Objekthierarchie sind unterschiedliche Knotentypen enthalten. Der Unterschied zwischen diesen Knotentypen besteht lediglich in der Art ihrer Verarbeitung durch den Renderingprozeß. Für das technische System handelt es sich bei den objektbeschreibenden Knoten um strukturierte Geometriemengen, die letztendlich alle auf den gleichen geometrischen Primitiven basieren.

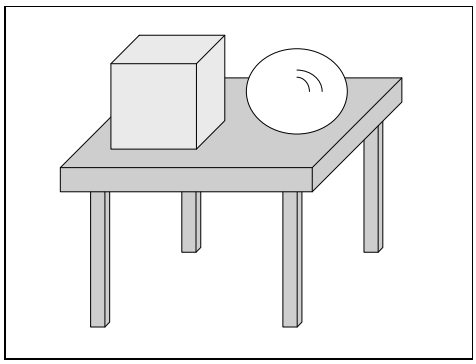


Abbildung 13: Beispielszene

Woher bezieht der Anwender seine Informationen?

In einem VR-System sieht der Anwender die virtuelle Umgebung auf dem ihm zur Verfügung stehenden visuellen Ausgabesystem. Ihm stellt sich eine vollständige Szene dar, die er interpretiert und mit seinen Erfahrungen vergleicht. Aus dieser Interpretation entwickelt er eine Erwartungshaltung. Für eine Problemlösung bildet diese Erwartungshaltung die Grundlage. Die in Abbildung 13 dargestellte Beispielszene stellt einen Tisch mit vier Beinen dar, auf dem eine Kugel und ein Quader liegen. Diese einfachen Schlußfolgerungen aus der Abbildung liegen einem VR-System nicht vor. Es hat weder eine Information über die Objekttypen Tisch, Kugel und Quader, noch weiß es etwas über deren räumliche Relationen zueinander. Diese Informationen wären für ein reines Visualisierungssystem auch nicht notwendig.

Wird in dem System eine Positionierungsoperation erlaubt, sind bereits weitere Informationen notwendig. Die einzelnen Objekte müssen selektierbar sein. Dazu muß die Geometriebeschreibung im System entsprechend der logischen Objekte strukturiert, zu einzeln selktierbaren Mengen separiert sein.

Aus einer solchen Abbildung kann weiterhin geschlossen werden, daß bei einer Positionierung des Tisches die darauf liegenden Objekte ebenfalls bewegt werden. Aufgrund der existierenden Gravitation und Reibung besteht eine Verbindung zwischen den Objekten. Im Fall der Positionierung des Quaders jedoch bleiben die anderen beiden Objekte in der Szene unberührt. Für die Simulation dieser Eigenschaft sind zusätzliche Informationen über bestehende Verbindungsrelationen zwischen den einzelnen Objekten notwendig, die das System im Fall entsprechender Interaktionen auch berücksichtigt.

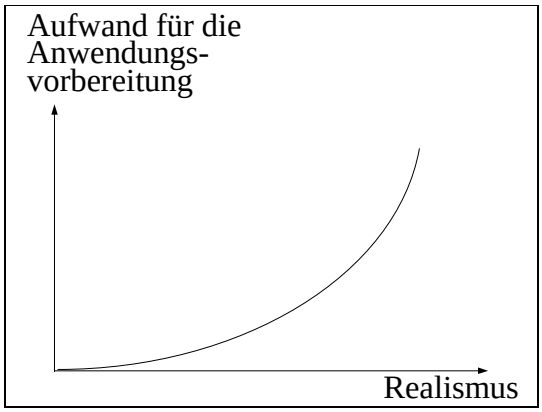


Abbildung 14: Zusammenhang zwischen Datenvorbereitungsaufwand und erzielter Realitätsnähe von Visualisierung und Simulation

Anhand dieses Beispieles soll verdeutlicht werden, daß mit der Zunahme der möglichen Interaktionen und der dabei erzielten Realitätsnähe auch der Informationsgehalt, d.h. die Abbildung der Realität im technischen System steigen muß. Das hat wiederum eine Erhöhung des Aufwandes für die Anwendungsvorbereitung zur Folge.

Eine Erhöhung der Realitätsnähe ist weiterhin eng mit einer aufwendigen Simulation gekoppelt. Die vollständige physikalisch basierte Simulation des Objektverhaltens in der virtuellen Umgebung ist jedoch nicht möglich, schon gar nicht unter Echtzeitbedingungen[57].

Um ein Systemverhalten zu erzielen, das der Erwartungshaltung des Anwenders so nahe wie möglich kommt, sind Informationen über die sich ihm darstellende Szene im VR-System notwendig, die über die geometrischen Eigenschaften der virtuellen Szene hinausgehen. Es handelt sich dabei um Strukturinformationen, räumliche und physikalische Relationen zwischen den einzelnen Objekten, die in entsprechenden Simulationen mit berücksichtigt werden. Einen großen Teil dieser Informationen kann sich der Anwender aufgrund seiner Erfahrungen aus der sich ihm präsentierenden Szene ableiten. Dazu ist das VR-System nicht in der Lage. Ohne eine entsprechend aufwendigere Daten- und Anwendungsvorbereitung stehen dem System diese Informationen nicht zur Verfügung. Das führt zu einer erhöhten Informationsdifferenz zwischen dem technischen System und dem Anwender.

3.4. Anforderungen an ein System für die Leitungsverlegung

In Kapitel 2, Abschnitt 2.2.2. wurden die funktionalen Anforderungen an ein System für die Leitungsverlegung beschrieben. In diesem Abschnitt werden diese funktionalen Anforderungen mit den Möglichkeiten verglichen, die mit der verfügbaren Technik in einem VR-System zur Verfügung stehen. Notwendige Systemerweiterungen werden diskutiert.

3.4.1. Visualisierung des Bauraumes

Für die reine Visualisierung des Bauraumes ist eine Systemerweiterung nicht erforderlich. Im Rahmen der Modellaufbereitung werden CAD-Daten in ein VR-Modell überführt. Diese VR-Modelle können in das System eingeladen und visualisiert werden. Die Modellaufbereitung sollte mit geringstem Aufwand möglich sein, damit eine Veränderung der Umgebung im CAD-System schnell im VR-System dokumentiert werden kann.

[57] [AsDa95]

3.4.2. Festlegung des Leitungsverlaufes in der virtuellen Umgebung

Der Verlauf einer Leitung ist in einem CAD-System durch die Mittellinie festgelegt, die durch parametrisch beschriebene CAD-Primitive definiert ist. Diese Mittellinie ist durch verschiedene Flächen umgeben, die die geometrischen Eigenschaften der Leitung definieren (siehe Abschnitt 2.2.1.3.). Für die Festlegung des Leitungsverlaufes in der virtuellen Umgebung soll eine entsprechende Beschreibung angewendet werden. Wie in Abschnitt 3.2.3. erläutert, sind die virtuellen Objekte durch Polygonmengen definiert. Für eine CAD-kompatible Leitungsrepräsentation ist daher eine Erweiterung des VR-Systems um CAD-Primitive notwendig. Diese müssen zur Laufzeit erzeugt und verändert werden können.

3.4.3. Import und Export von CAD-Objekten

Die Ergebnisse der Leitungsverlegung in der virtuellen Umgebung müssen dem gesamten Entwicklungsprozeß transparent gemacht werden können. Dazu ist ein Austausch der Ergebnisse zwischen den verschiedenen Systemen Voraussetzung. Aus diesem Grund muß das System eine CAD-Schnittstelle im Bereich des Datenimports wie auch -exports unterstützen. Diese Schnittstelle ist ausschließlich für die CAD-Primitive zur Verfügung zu stellen, die im VR-System erzeugt und verändert werden.

3.4.4. Situationsabhängiges Systemverhalten

Die räumlichen und funktionalen Beziehungen der virtuellen Objekte untereinander können sich im Laufe einer Anwendung ändern. Wird diese Veränderung durch das System nicht berücksichtigt, führt das zu einem Verhalten, das weder der Erwartungshaltung des Anwenders noch dem Verhalten der Objekte in der Realität entspricht.

Über ein einfaches Interaktionsmodell, wie es in Abschnitt 3.2.4. beschrieben wurde, können solche sich ändernden Situationen nur teilweise oder gar nicht berücksichtigt werden. Das VR-System ist nicht in der Lage, eine bestimmte Situation zu erfassen und dementsprechend das Objektverhalten anzupassen. Statt dessen wird eine statische Situation durch eine aufwendige, manuell durchzuführende, modellabhängige Anwendungsvorbereitung erzeugt.

Um diesen Nachteil zu umgehen, soll das VR-System durch eine Komponente erweitert werden, in der das Objektverhalten der verschiedenen virtuellen Objekte situationsabhängig definiert werden kann. Bei der Implementierung sind verschiedene Anforderungen zu berücksichtigen:

1. Der Aufwand für die Definition des Objektverhaltens ist minimal zu halten. Für die Arbeit in einer neuen virtuellen Umgebung sollte so wenig wie möglich Spezialwissen nötig sein.
2. Die Komponente soll möglichst unabhängig vom VR-System implementiert sein. Für die VR-Technik existieren zur heutigen Zeit keine Standardsysteme. Daher ist eine Umstellung auf ein neues System bei der Realisierung zu berücksichtigen.

3. Die von einem System zur Leitungsverlegung an die virtuellen Objekte gestellten Anforderungen sollten so gering wie möglich ausfallen, damit kein übermäßiger Zusatzaufwand für die Modellaufbereitung notwendig ist.

3.4.5. Grundlegende Interaktionsmöglichkeiten

Die Interaktionsmöglichkeiten sollen sich in dem System am Anwendungsziel, d.h. an den zu erzeugenden Leitungsobjekten orientieren. Für die Veränderung der Leitungsobjekte sind Interaktionen zu realisieren, die sich an Abläufen in der Realität orientieren. Mit der verfügbaren VR-Hardware existieren Ein- und Ausgabegeräte, die eine direkte Manipulation von virtuellen Objekten im freien Raum erlauben. Solche direkten Manipulationen sollen in dem System überwiegend implementiert werden.

Eine direkte Manipulation in der Realität erfolgt hauptsächlich durch die Hände. Mit den Händen können Objekte in ihrer Form und in ihrer Position im Raum verändert werden. Bei diesen Interaktionen zeigen unterschiedliche Objekte auch ein unterschiedliches Verhalten. Dieses Verhalten ist von den Objekteigenschaften selbst wie auch von der existierenden Umgebung und der damit gegebenen aktuellen Situation abhängig. Im Abschnitt 3.5. werden verschiedene grundlegende Interaktionsalternativen diskutiert, die für die Anwendung von Interesse sind. Deren Realisierung in anderen Systemen wird analysiert.

3.5. Direkte Manipulation in einer VR-Umgebung - Diskussion möglicher Alternativen

Für eine direkte Manipulation von virtuellen Objekten in einem VR-System gibt es die verschiedensten Herangehensweisen. Von besonderem Interesse für die Leitungsverlegung ist das Greifen und Positionieren von Objekten im freien Raum. Während in der Realität diese Objektmanipulationen durch die physikalischen Gegebenheiten der Objekte und der Umgebung wie auch durch die physischen Möglichkeiten des Menschen selbst beeinflußt werden, müssen in einer virtuellen Umgebung alle damit zusammenhängenden Phänomene simuliert werden. Um die verschiedenen Simulationen beurteilen zu können, wird die Positionierung eines Objektes im Raum in den folgenden Abschnitten diskutiert.

Die Positionierung von Objekten unterteilt sich in drei Teilvorgänge: Greifen, Positionieren und Lösen des Objektes.

3.5.1. Der Greifvorgang

Der Greifvorgang unterteilt sich in zwei Teile. Im ersten Teil wird das zu greifende Objekt selektiert. Danach wird eine Verbindung zwischen Greifer und gegriffenem Objekt aufgebaut.

Die *vordefinierte Objekt-Selektion* erfolgt durch ein Ereignis, das eine Greifaktion eines vor Anwendungsbeginn definierten Objektes auslöst. Bei dem Ereignis kann es sich um eine Geste, oder eine bestimmte Spracheingabe, einen Druck auf einen Schalter oder eine ähnliche Systemeingabe handeln. Berechnungsintensiver erfolgt die *dynamische Selektion* zur Laufzeit. Sie basiert auf der ständigen Abfrage einer räumlichen Relation zwischen einem Selektionsobjekt und einer Menge selektierbarer virtueller Objekte. Für die Definition und Abfrage solcher räumlichen Relationen kommt häufig eine polygonbasierte Kollisionserkennung zum Einsatz. Bei Eintreten eines entsprechenden Kollisionsereignisses ist das Objekt selektiert.

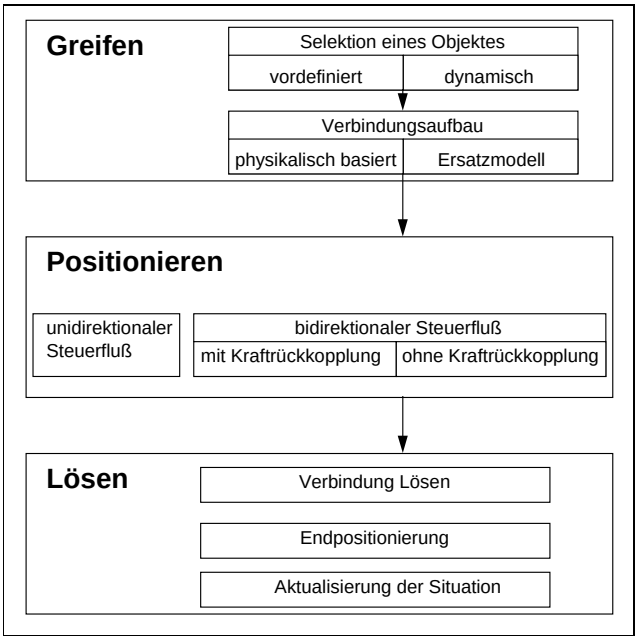


Abbildung 15: Direkte Manipulation (Positionierung) von Objekten

Ist die Selektion abgeschlossen, erfolgt der Aufbau einer Verbindung zwischen dem Greiferobjekt und dem gegriffenen Objekt. Hierbei wird zwischen dem *physikalisch basierten Verbindungsmodell* und dem *Ersatzmodell* unterschieden.

3.5.1.1. Physikalisch basierte Verbindungsmodelle

Ein physikalisch basiertes Verbindungsmodell ist im allgemeinen sehr aufwendig zu implementieren. Es ist eng mit der Entwicklung verschiedener Kraftrückkopplungsgeräte verbunden. In der Literatur wird ein Kraftrückkopplungsgerät näher erläutert, mit dessen Hilfe in einer virtuellen Umgebung Objekte gegriffen werden können. Die Ansteuerung des in [YASU96] beschriebenen SensorGlove basiert auf den zwei wesentlichen Kraftflüssen, die bei einer solchen Simulation berücksichtigt werden müssen. Der Kraftfluß vom Anwender, von der menschlichen Hand, zum Gerät und der Kraftfluß vom virtuellen Objekt zum Gerät (siehe Abbildung 16). Der erste Kraftfluß dient der Eingabe, aus dem zweiten Kraftfluß wird die Ausgabekraft bzw. Positionierung für das Gerät berechnet. In der in [YASU96] beschriebenen Arbeit wird nicht näher erläutert, wie die virtuellen Objekte aufgebaut sind. Die folgenden objektbeschreibenden Merkmale, die über die geometrische Beschreibung der Objekte hinausgehen, sind aber Voraussetzung für die Berechnung[58]:

[58] [YASU96]

1. Kontaktmodell: exakte Kollisionspunkt-Ermittlung zwischen den Fingern und dem Objekt
2. Berechnung der Tangente zur Oberfläche in diesem Punkt
3. Objektschwerpunkt (Massenschwerpunkt) als lokaler Koordinatenursprung verfügbar
4. Objektmasse
5. Bewegungsvektoren der Finger

Um die exakte Tangente im Kollisionspunkt berechnen zu können, ist die parametrische Beschreibung des zu greifenden Objektes Voraussetzung. Bei konkaven Objekten oder bei Modellen, die zwischen den einzelnen Flächen Lücken aufweisen, kann es zu mehreren Kontaktpunkten kommen. Wie solche Probleme behandelt werden können, wird in [YASU96] nicht erläutert. Die Beschreibung des Experimentes bezieht sich ausschließlich auf einen Regelkörper, einen Zylinder.

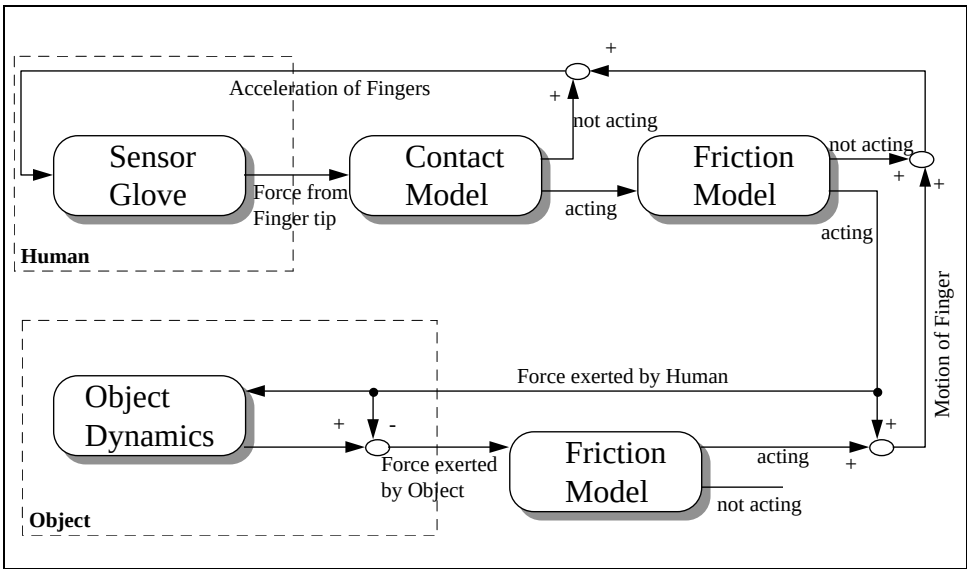


Abbildung 16: Kraftfluß zwischen Mensch (Hand) und virtuellem Objekt im Dynamic Force Simulator zur Ansteuerung des Sensor Glove (Quelle: [YASU96])

Auch die Verformung der Oberfläche einzelner Finger ist in ein physikalisch basiertes Modell integrierbar. So wurde die Fingerkuppe eines Robotergreifers mit Hilfe eines Feder-Masse-Netzes nachgebildet[59]. Die Verformung dieses Netzes auf einer virtuellen Oberfläche wurde simuliert. Bei dieser Oberfläche handelte es sich um eine Ebene. Damit wurde die Kollisionsberechnung zwischen jedem einzelnen Massepunkt und dem Objekt wesentlich vereinfacht. Die wirkenden Reibungskräfte zwischen dem Fingermodell und der Oberfläche wurden dabei nicht berücksichtigt. Auch die durch die Verformung des Fingermodells auftretenden Gegenkräfte werden bei dieser Simulation nicht quantifiziert. Die reine Deformation dieses Fingers ist hierbei das Ergebnis dieser

[59] [REZ96]

Anwendung. Aussagen über das Verhalten an nicht ebenen Flächen können nicht gemacht werden.

Weitere Möglichkeiten der Positionierung von virtuellen Objekten, die auf einem physikalischen Modell basieren, sind bei vereinzelten Spezialanwendungen zu finden. Dazu ist eine realisierte Interaktion mit Atomen und Molekülen in einer Anwendung zur Modellierung großer Moleküle zu nennen[60]. Bei diesem System steuert der Anwender einen Cursor im Raum, der über eine Feder mit einem Objekt verbunden ist. Durch die Bewegung in eine bestimmte Richtung wird über das Federelement ein Kraftvektor auf das gegriffene Objekt übertragen. Die Bewegung der Objekte wird über Energiefunktionen berechnet, die minimiert werden. Für eine ähnliche Anwendung kommt ein Kraftrückkopplungsgerät zum Einsatz, der sogenannte ARM (Argonne Remote Manipulator), über den die auftretenden Kräfte bei der Molekül-„Konstruktion“ zurückgegeben werden können[61].

Die beschriebenen Speziallösungen sind nicht auf einen allgemeinen Fall anwendbar und können daher bei der Diskussion vernachlässigt werden.

3.5.1.2. Verbindungs-Ersatzmodelle

Bei den Verbindungs-Ersatzmodellen wird das selektierte Objekt in der Objekthierarchie unter das den Greifer repräsentierende Objekt gehängt und so mit dessen Bewegung mitgeführt (siehe auch Abschnitt 3.2.3 und 3.2.4.). Es handelt sich dabei um eine rein geometrische Operation ohne eine Berücksichtigung eventuell auftretender Kräfte, so daß dabei auch auf eine Kraftrückkopplung verzichtet werden kann. Dies stellt eine relativ einfache und sehr weit verbreitete Methode zur Positionierung von virtuellen Objekten dar. Sie ist in der Literatur beschrieben und wird in dem im Rahmen dieser Arbeit eingesetzten VR-System Virtual Design II des IGD in Darmstadt ebenfalls genutzt[62]. In [IWA90] wird ebenfalls das gegriffene Objekt durch ein Verändern der Objekthierarchie mit der Hand mitbewegt. Interessant ist hierbei die Relation für den Verbindungsaufbau, die während der Simulation ständig überprüft wird. Ein Objekt gilt mit der Hand als verbunden, wenn mindestens zwei Kollisionspunkte von 16 möglichen zwischen der virtuellen Hand und dem Objekt existieren, wobei mindestens ein Kollisionspunkt davon auf dem Daumen liegen muß. Es wird mit Punktrelationen gearbeitet. Ständig wird der Abstand zwischen den Punkten und dem Objekt ermittelt. Lediglich fest vorgegebene Punkte, die sich auf der virtuellen Oberfläche der künstlichen Hand befinden, werden auf Kollision geprüft, nicht die gesamte Handgeometrie. Diese Relation muß dauernd geprüft werden, denn bei Auflösen dieser Relation wird sofort die Verbindung gelöst. Bei dem in [IWA90] beschriebenen Versuchsaufbau wird mit einem Kraftrückkopplungsgerät gearbeitet, über welches dem Anwender Informationen sowohl über die feste Oberfläche des gegriffenen Objektes, wie auch über dessen Trägheit und Gravitation vermittelt werden.

[60] [SUR92]

[61] [BRO90]

[62] [ZACH96]

3.5.2. Positionierung

Bei der Positionierung des gegriffenen Objektes in der virtuellen Umgebung gibt es die Möglichkeit, die im System abgebildete Umgebung mit zu berücksichtigen oder sie zu vernachlässigen. Wird sie vernachlässigt, kann man das gegriffene Objekt frei im Raum bewegen, es gibt keine Bewegungseinschränkungen und keine Hinweise auf realitätsfremde Zustände. Die Anwendung hat keine zusätzlichen Simulationen durchzuführen, eine Kollisionsuntersuchung ist nicht notwendig. Die Steuerung erfolgt unidirektional vom Anwender in Richtung Objekt (Abbildung 17).

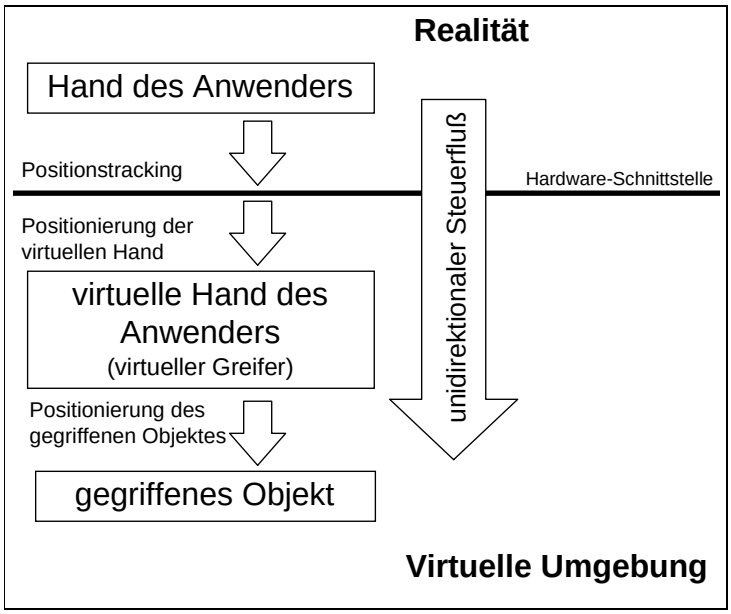


Abbildung 17: Unidirektionaler Steuerfluß Anwender → Objekt

Bei einer Berücksichtigung der virtuellen Umgebung ist der Berechnungsaufwand wesentlich größer. Es können wiederum zwei Möglichkeiten unterschieden werden.

Bei der ersten Möglichkeit werden räumliche Beziehungen, wie beispielsweise Kollisionen zwischen bewegtem Objekt und der Umgebung, akustisch und/oder visuell angezeigt[63], haben aber keinen Einfluß auf die Bewegung des gegriffenen Objektes.

Bei der zweiten Möglichkeit führt eine Wechselwirkung zwischen bewegtem Objekt und virtueller Umgebung auch zu einer Beeinflussung der Bewegungssteuerung. Dadurch wird verhindert, daß unerwünschte Zustände, wie beispielsweise die gegenseitige Durchdringung mehrerer fester Körper, vermieden werden. Das führt zu einem bidirektionalen Steuerfluß der Objektpositionierung, da bei bestimmten Zuständen eine Korrektur der vom Anwender durchgeführten Positionierung des Objektes notwendig ist. Diese Beeinflussung kann mit Hilfe von Kraftrückkopplungsgeräten direkt auf den Bewegungsapparat des Anwenders erfolgen (Alternative A), sie kann sich auf

[63] [GOMES97] S. 346

den virtuellen Greifer (Alternative B) oder auf das gegriffene Objekt (Alternative C) beziehen. Diese drei Möglichkeiten werden im Folgenden diskutiert (siehe auch Abbildung 18):

Alternative A:

Über ein Kraftrückkopplungsgerät wird der Anwender daran gehindert seine Hand so zu bewegen, daß ein unerwünschter oder in der Realität unmöglicher Positionierungszustand in der virtuellen Umgebung entsteht. Durch die Bewegungseinschränkung der realen Hand wird die Bewegung des Greiferobjektes beeinflußt. Diese Beeinflussung der Positionierungsbewegung entspricht der Wechselwirkung zwischen den Objekten in der Realität. Voraussetzung für diese Vorgehensweise ist eine aufwendige Simulation und eine damit verbundene aufwendige Modellaufbereitung. Weiterhin ist die Anwendung eines Kraftrückkopplungsgerätes notwendig.

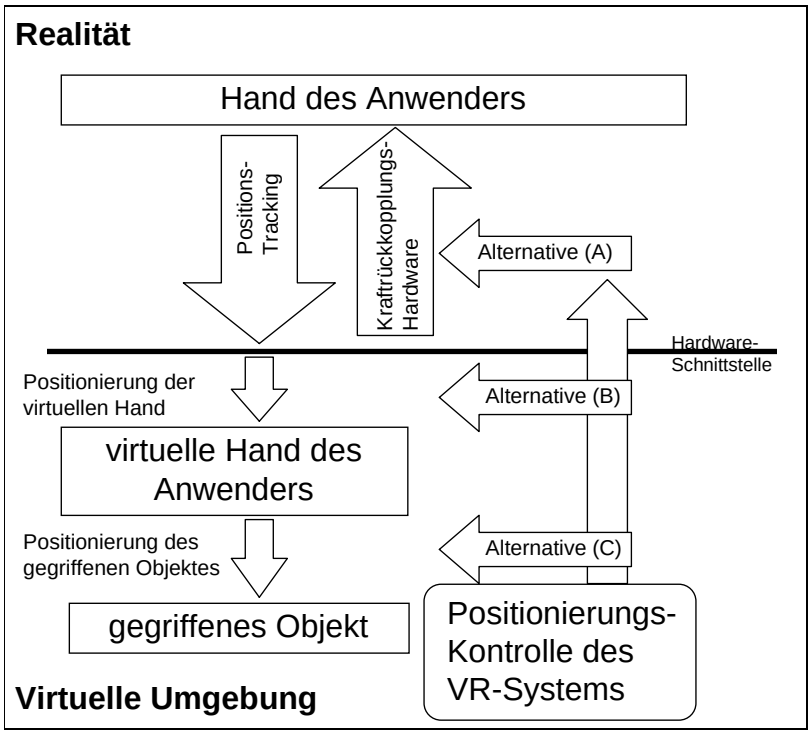


Abbildung 18: Bidirektionaler Steuerfluß bei der Objektpositionierung

Alternative B:

Es wird auf ein Kraftrückkopplungsgerät verzichtet und direkt auf das virtuelle Modell Einfluß genommen. Diese Beeinflussung bezieht sich auf das Greifobjekt. Damit wird im Fall der Einflußnahme auf die Positionierung die kinematische Kette zwischen der realen und der virtuellen Hand unterbrochen. Tests haben ergeben, daß diese Unterbrechung dazu führen kann, daß die Abbildung der Bewegung von der Realität auf die virtuellen Objekte für den Anwender nicht mehr nachvollziehbar ist.

Alternative C:

Die Beeinflussung der Positionierung erfolgt direkt am gegriffenen Objekt. Damit bleibt die kinematische Kette zwischen Realität und virtuellem Greifobjekt erhalten. Die Verbindung zwischen Greifobjekt und gegriffenem Objekt wird allerdings aufgehoben. Das kann dazu führen, daß Durchdringungen zwischen Greifer und gegriffenem Objekt auftreten bzw. daß beide Objekte sich weit voneinander entfernen.

3.5.3. Lösen

Für das Lösen einer Verbindung zwischen Greifobjekt und gegriffenem Objekt müssen zwei Fälle unterschieden werden. Im ersten Fall kann die Auflösung der Verbindung durch ein *festgelegtes Ereignis* erfolgen. Tritt dieses Ereignis ein, wird die Verbindung aufgehoben[64]. Im zweiten Fall ist es möglich, daß bei jedem Simulationsschritt eine *festgelegte räumliche Relation* zwischen dem Greifer und dem gegriffenen Objekt abgefragt wird. Ist diese nicht mehr gegeben, so wird die Verbindung ebenfalls gelöst[65].

Im Fall eines *physikalisch basierten Verbindungsmodells* ist beim Lösen einer Verbindung eine Anpassung der Objekthierarchie nicht notwendig, da sie auch beim Verbindungsaufbau nicht verändert wurde. Mit dem Lösen der Verbindung ist das Objekt wieder frei in der virtuellen Umgebung abgelegt.

In der Realität wirken nach dem Loslassen eines Objektes noch physikalische Gesetze, die die Position des Objektes noch verändern können. Ein Beispiel dafür ist das Abstellen eines Gegenstandes auf einem Tisch. Wird dieser Gegenstand nicht exakt auf der Oberfläche positioniert, so wirken auf ihn gravitative Kräfte. Diese und die Wechselwirkungen des Objektes mit der Umgebung führen zu einer endgültigen Positionierung des Objektes erst nach dem eigentlichen Lösen der Verbindung. Ist dieses Verhalten in der *virtuellen Realität* gefordert, muß es aufwendig simuliert werden.

Beim Einsatz eines *Ersatzmodells* für die Beschreibung einer Verbindung zwischen Greifobjekt und gegriffenem Objekt ist eine Veränderung der Objekthierarchie beim Lösen der Verbindung notwendig. Beim Verbindungsaufbau wurde das gegriffene Objekt in der Objekthierarchie unter dem Greifobjekt positioniert, damit es mit dessen Bewegungen mitgeführt wird. Eine Lösung der Verbindung erfordert das Umkehren dieses Vorganges, das Objekt wird wieder an die ursprüngliche Position in der Objekthierarchie positioniert. Diese Vorgehensweise kann zu Fehlern führen, die sich im Objektverhalten verdeutlichen. Ein Beispiel:

Das Ablegen eines Objektes an einer beliebigen Position im Raum kann zu einer neuen Situation in der Umgebung führen. Wird *in der Realität* beispielsweise ein Objekt von einem Tisch A auf einen anderen Tisch B gelegt und der Tisch B wird anschließend neu positioniert, dann bewegt sich das Objekt mit

[64] [ZACH96]

[65] [IWA90]

Tisch B mit. Eine Positionierung des Objektes von Tisch A nach Tisch B hat eine Änderung der Beziehungen zwischen dem Objekt und seiner Umgebung zur Folge. Um ein realitätsnahes Objektverhalten entsprechend der Erwartungshaltung des Anwenders zu simulieren, ist diese Problematik bei der Implementierung der Interaktionen mit zu berücksichtigen.

Eine Lösung dieser Problematik ist

- durch eine manuelle Anpassung des Systemverhaltens durch den Anwender,
- durch eine automatische Anpassung des Systemverhaltens mit einem erweiterten Objekt- und Interaktionsmodell oder
- durch eine vollständige Simulation der gesamten physikalisch basierten Wechselwirkungen zwischen den Objekten

möglich.

3.5.4. Fazit

Die Interaktionen während der Festlegung des Leitungsverlaufes sollen direkt auf den grafischen Objekten erfolgen. Dabei soll die direkte Manipulation mit der virtuellen Hand die grundlegende Operation darstellen. Diese Operation wird immer auf das gleiche Ablaufschema zurückgeführt. Die Objekte werden gegriffen, positioniert und wieder losgelassen.

Greifen

Das *Greifen* eines Objektes unterteilt sich in die Selektion des zu greifenden Objektes und den Verbindungsaufbau zwischen Greifobjekt und gegriffenem Objekt.

Selektion

Die Objektselektion für den Greifvorgang in einer virtuellen Umgebung kann statisch, datenabhängig vordefiniert und durch ein Ereignis ausgelöst werden. Sie kann aber auch dynamisch auf der Basis einer Kollisionserkennung durchgeführt werden. Für den Fall der Leitungsverlegung ist der dynamische Selektionsvorgang vorzuziehen, da mit Objekten interagiert wird, die erst zur Laufzeit erzeugt werden. Das schließt die statisch vordefinierte Variante aus.

Verbindungsaufbau

Beim Verbindungsaufbau zwischen Greifobjekt und gegriffenem Objekt kann wiederum zwischen einer physikalisch basierten Verbindung und einer ersatzmodellbasierten Verbindung unterschieden werden.

Die physikalisch basierte Verbindung erfordert einen hohen Simulations- und Datenvorbereitungsaufwand. Weiterhin ist der Einsatz von Kraftrückkopplungsgeräten erforderlich. In der Literatur sind dafür Speziallösungen beschrieben, die sich nicht ohne weiteres auf den Fall der Leitungsverlegung

übertragen lassen. Daher kommt die physikalisch basierte Verbindungssimulation im Rahmen dieser Arbeit nicht zur Anwendung.

Die ersatzmodellbasierte Verbindung stellt eine Alternative zur physikalisch basierten Verbindung dar. Sie beruht auf der Manipulation der Objekthierarchie, durch die das gegriffene Objekt unter das greifende Objekt „gehängt“ wird und dessen Bewegungen folgt. Es handelt sich um eine sehr einfache Methode des Verbindungsaufbaus, es ist keine zusätzliche Hardware und auch kein zusätzlicher Datenaufbereitungsaufwand erforderlich. Die ersatzmodellbasierte Verbindung wird daher als Alternative zur physikalisch basierten Verbindungssimulation für den Fall der Leitungsverlegung angewendet.

Positionieren

Das Positionieren eines virtuellen Objektes kann entweder unter Vernachlässigung oder unter Berücksichtigung der virtuellen Umgebung erfolgen. Im Hinblick auf die angestrebten Systemziele ist das vollständige Vernachlässigen der umgebenden virtuellen Objekte während einer Positionierung nicht akzeptabel.

Bei der Berücksichtigung der virtuellen Umgebung während der Positionierung eines Objektes werden zwei Fälle unterschieden. Im *ersten Fall* beeinflußt die virtuelle Umgebung das bewegte Objekt nicht. Auftretende Kollisionen zwischen Objekt und Umgebung werden beispielsweise akustisch oder visuell angezeigt (unidirektionaler Steuerfluß).

Im *zweiten Fall* wird das gegriffene Objekt durch die virtuelle Umgebung in seiner Bewegung beeinflußt (bidirektionaler Steuerfluß). Von den drei möglichen Alternativen A bis C (Abschnitt 3.5.2.) wird die letzte Alternative im Rahmen dieser Arbeit angewendet. Alternative A scheidet wegen des erhöhten Modellaufbereitungsaufwandes, des erhöhten Simulationsaufwandes und der Notwendigkeit der Kraftrückkopplung aus. Alternative B führt zu Irritationen des Anwenders und erscheint daher auch nicht als optimale Konfiguration. Alternative C kann

1. zu ungewollten Durchdringungen zwischen Greifobjekt und gegriffenem Objekt bzw. der virtuellen Umgebung führen.
2. zu einem gegenseitigen Entfernen von Greifer und gegriffenem Objekt führen.

Beide Problematiken sind für die Leitungsverlegung akzeptabel. Die erste Problematik, die gegenseitige Durchdringung, spielt im Fall der Bauraumsuche und Leitungsmodellierung keine Rolle. Solche Durchdringungen beeinflussen nicht das Anwendungsziel.

Die zweite Problematik kann durch die Umsetzung einer einfachen, effizienten Nachgreifoperation entschärft werden, durch die der Versatz zwischen Greifer und gegriffenem Objekt schnell minimiert werden kann.

Im Fall der Leitungsverlegung kommt sowohl der unidirektionale Steuerfluß wie auch der bidirektionale Steuerfluß bei der Positionierung eines Objektes zum

Einsatz. Im Fall des bidirektionalen Steuerflusses wird auf die Bewegung des gegriffenen Objektes direkt Einfluß genommen (Alternative C).

Lösen

Die Anwendung des ersatzmodellbasierten Verbindungsmodells erzwingt eine Veränderung der Objekthierarchie beim Lösen des gegriffenen Objektes. Wird dabei die aktuelle Situation in der virtuellen Umgebung, d.h. werden die räumlichen Relationen zwischen gegriffenem Objekt und Umgebung nicht mit berücksichtigt, kann das zu einem Objektverhalten führen, das der Erwartungshaltung des Anwenders nicht entspricht. Es wird eine Anpassung des Systemverhaltens erforderlich, die durch drei Alternativen realisiert werden kann. Die manuelle Anpassung des Systemverhaltens durch den Anwender scheidet hierbei aus, da sie zu immer wiederkehrenden, dem Anwender nicht verständlichen Interaktionen führt. Die vollständige, physikalisch basierte Simulation aller Wechselwirkungen zwischen den Objekten scheidet aus Gründen der Systemperformance und des Anwendungsvorbereitungsaufwandes aus. Damit bleibt die automatische Anpassung des Systemverhaltens an die jeweils aktuelle Situation mit Hilfe eines erweiterten Objekt- und Interaktionsmodells die Alternative, die im Rahmen dieser Arbeit zum Einsatz kommt.

Die im Verlauf der Anwendung zu erzeugenden Schlauchobjekte setzen sich aus mehreren Teilobjekten zusammen und können auch mit anderen virtuellen Objekten verbunden werden. Damit der Aufbau und die Trennung dieser Verbindungen wie auch die Berücksichtigung dieser Verbindungen bei den verschiedenen Interaktionen möglich ist, muß das System selbst entscheiden können, ob solche Verbindungen existieren oder nicht. Dann ist es auch möglich, daß das System den Aufbau und die Trennung einer Verbindung aktiv unterstützt.

Die Berücksichtigung der Verbindungen bei den verschiedenen Interaktionen macht auch eine Erweiterung der Interaktionsdefinition im Rahmen der Anwendungsvorbereitung notwendig. Solche Objektrelationen können mit dem bestehenden Interaktionsmodell nicht behandelt werden.

Die Erweiterung des VR-Systems um einen Verbindungsmanager, der die bestehenden räumlichen Relationen zwischen den Objekten in der virtuellen Umgebung selbständig erkennen und aktualisieren kann, ist eine Voraussetzung für effektive und realitätsnahe Interaktionen in einer virtuellen Umgebung.

3.6. Zusammenfassung

Im Kapitel 3 wurde ein Überblick zum aktuellen Stand der VR-Technik gegeben, auf dessen Basis die weitere Arbeit aufbaut. Eine Begriffsbestimmung bildet dabei die Grundlage. Der Begriff des Virtual Reality Systems wurde definiert. Der allgemeine Aufbau eines VR-Systems mit seinem Ausgabe- und Eingabekanal, seinem Geometrie und Interaktionsmodell wurde überblicksweise behandelt.

Es konnte gezeigt werden, daß sich ein VR-System in seinem allgemeinen Aufbau von einem herkömmlichen interaktiven System nicht unterscheidet.

Vielmehr sind es die Ein- und Ausgabemetaphern, die den Unterschied zu den herkömmlichen interaktiven Systemen ausmachen. Für deren Realisierung in einer Umgebung mit Echtzeitanforderungen ist die Anwendung spezieller Schnittstellenhardware erforderlich.

Die zur Zeit bestehenden Möglichkeiten und Grenzen der Ein- und Ausgabegeräte wurden in den Abschnitten 3.2.1. und 3.2.2. diskutiert. Die mit der Hardware eng verbundene Software wurde in den Abschnitten 3.2.3. und 3.2.4. behandelt. Dabei wurden anhand des VR-Systems Virtual Design II des Fraunhofer-Institutes für Graphische Datenverarbeitung in Darmstadt sowohl das Geometrie- wie auch das Interaktionsmodell eines VR-Systems beschrieben.

Bei dem Geometriemodell handelt es sich um eine Objekthierarchie bestehend aus Einzelobjekten, die hauptsächlich polygonal beschrieben sind.

Das Interaktionsmodell beruht auf dem *action-event-Prinzip* und arbeitet auf abstrakten geometrischen Objekten, die innerhalb der Objekthierarchie definiert sind.

Daraus ergibt sich, daß das Systemverhalten stark von der Objektstrukturierung der virtuellen Umgebung abhängig ist. Damit gewinnt der Prozeß der Modellgewinnung innerhab des Produktenstehungsprozesses (Abschnitt 3.3.) eine zunehmende Bedeutung. Bei der VR-Modellgewinnung handelt es sich um einen unidirektionalen Prozeß vom CAD-Datensatz zum VR-Modell. Es handelt sich um einen Übergang von der mathematisch exakten zu einer durch Polygone angenäherten Objektbeschreibung. Die Objektstrukturierung basiert hierbei auf den CAD-Datensätzen im Produktdatenmanagementsystem. Die VR-Modellaufbereitung führt zu einer Abnahme der Modellbeschreibungsgenauigkeit.

Im Abschnitt 3.3.2. wurde gezeigt, daß sich aus dem hauptsächlich geometrisch beschriebenen VR-Modell und dessen Wahrnehmung durch den Anwender eine Informationsdifferenz zwischen Anwender und VR-System ergibt. Während das System auf abstrakten geometrischen Objekten arbeitet, ordnet der Anwender den virtuellen Objekten weitere Eigenschaften zu, die diese aber nicht besitzen. Auf der Grundlage dieser falschen Annahme wird beim Anwender eine Erwartungshaltung bezogen auf die Systemfunktionalitäten und das Objektverhalten erzeugt. Diese Erwartungshaltung ergibt sich aus der Darstellung der virtuellen Objekte und basiert auf den Erfahrungen des Anwenders aus der Realität. Auf solche Erfahrungen kann das System nicht zurückgreifen. Für die Annäherung des Systemverhaltens an die Erwartungshaltung des Anwenders ist es daher erforderlich, eine aufwendigere Daten- und Anwendungsvorbereitung zu betreiben, durch die die beschriebene Informationsdifferenz verringert wird.

Im Abschnitt 3.4. wurden die funktionalen Anforderungen an ein System zur Leitungsverlegung mit den Möglichkeiten eines VR-Systems verglichen. Aus diesem Vergleich ergibt sich die Notwendigkeit der Systemerweiterung für die Bauraumfestlegung in einer virtuellen Umgebung und den Import und Export von CAD-Objekten. Weiterhin konnte gezeigt werden, daß es nicht möglich ist

mit einem einfachen Interaktionsmodell, wie es in Abschnitt 3.2.4. beschrieben wurde, ein situationsabhängiges realitätsnahes Systemverhalten entsprechend der Erwartungshaltung des Anwenders umzusetzen. Vielmehr ist eine Systemerweiterung notwendig, die ein situationsabhängiges Interagieren erlaubt und unterstützt. Dazu wurden in Abschnitt 3.5. Möglichkeiten der direkten Manipulation diskutiert und mögliche Alternativen bewertet.

Es wurde herausgearbeitet, daß sich die direkte Manipulation der Objekte immer auf das gleiche Ablaufschema Greifen - Positionieren - Lösen abbilden läßt. Diese Teiloperationen lassen sich in der unterschiedlichsten Art und Weise umsetzen. Unter der Randbedingung der Minimierung des Aufwandes für die Anwendungsvorbereitung und des Simulationsaufwandes stellt die Anwendung von Ersatzmodellen und der Verzicht auf Kraftrückkopplungsgeräte die für das System zur Leitungsverlegung günstige Konfiguration dar. Da auf eine aufwendige physikalisch basierte Simulation verzichtet wird, ist die Erweiterung des VR-Systems um einen Verbindungsmanager notwendig. Dieser muß die räumlichen Relationen zwischen den virtuellen Objekten selbständig erkennen können und aufbauend auf diesen Informationen Einfluß auf die Interaktionen nehmen. Das ist die Voraussetzung für ein situationsabhängiges Interaktionsmanagement.

4. Konzeption und Realisierung der Systemerweiterungen

Im 2. Kapitel (Abschnitt 2.2.) wurde die Leitungsverlegung als Teil des Produktentstehungsprozesses diskutiert. Dabei wurden die Nachteile der aktuellen Herangehensweise an diese Problematik herausgearbeitet. Eine Verbesserung dieses Prozesses durch die Anwendung eines VR-Systems für die Lösung dieser Aufgabe wurde dabei als Hypothese aufgestellt. Das Ergebnis dieser Diskussion ist eine Menge von Anforderungen, die solch ein System erfüllen muß. Diese Anforderungen wurden mit den technischen Möglichkeiten bestehender VR-Systeme am Beispiel des Systems Virtual Design II verglichen (3. Kapitel). Das Ergebnis dieses Vergleiches verdeutlicht die Notwendigkeit einer Erweiterung des bestehenden VR-Systems um verschiedene Funktionalitäten. Die Konzeption und Umsetzung dieser Erweiterungen werden in diesem Kapitel erläutert. Es gliedert sich in die drei Hauptabschnitte 4.1.-4.3., innerhalb derer jeweils sowohl auf die Konzeption/Motivation einer Systemerweiterung wie auch auf deren Realisierung eingegangen wird. Es sei an dieser Stelle darauf hingewiesen, daß die in diesem Kapitel beschriebenen Konzepte alle anhand eines Anwendungsprototypen umgesetzt wurden. Auf die damit gesammelten praktische Erfahrungen wird in Kapitel 6 eingegangen.

Im Abschnitt 4.1. wird die Erweiterung eines VR-Systems um zusätzliche Datenstrukturen und Funktionen diskutiert, die der Repräsentation und direkten Manipulation von CAD-Entities dienen. Die beschriebenen Konzepte und Systemerweiterungen sind im Rahmen einer Diplomarbeit[66] entwickelt worden, die im Rahmen einer Kooperation zwischen der TU Braunschweig und der Volkswagen AG betreut wurde. Die Ergebnisse dieser Diplomarbeit flossen direkt in diese Arbeit mit ein und bilden eine Grundlage, auf der weiter aufgebaut wird.

In Abschnitt 4.2. wird eine zusätzliche Datenstruktur diskutiert (Verbindungsstutzen), die sowohl für die systemunterstützte Erzeugung von Verbindungen zwischen verschiedenen Objekten wie auch für die Analyse solcher Verbindungen genutzt werden kann.

Um die Verbindungsrelationen im Systemverhalten mit berücksichtigen zu können, wird eine Objektverwaltung vorgeschlagen, die auf logischen Objekten möglichst geometrie- und damit systemunabhängig arbeitet. In dieser Objektverwaltung sind Objektrelationen beschrieben, die mit den Möglichkeiten einer rein geometriebasierten VR-Objekthierarchie nicht beschreibbar, aber für ein realitätsnahes dynamisches Objektverhalten notwendig sind. Die Konzeption und Implementierung einer solchen Objektverwaltung basierend auf einer Klassenhierarchie bildet den Schwerpunkt des Abschnittes 4.3..

[66] [RAB96]

4.1. CAD-Entitäten als Grundlage für die Leitungsrepräsentation

Im 3. Kapitel, Abschnitt 3.2.3., wurde bereits das Geometriemodell eines VR-Systems überblicksweise erläutert. Die geometrische Beschreibung der einzelnen VR-Objekte basiert auf Mengen von Polygonen. Diese Objektrepräsentation steht im Widerspruch zu verschiedensten CAD-Systemen, bei denen die geometrische Beschreibung der Objekte auf mathematischen Funktionen beruhen. Diese Repräsentationsform hat zwei *Vorteile gegenüber der polygonalen Darstellungsform:*

1. *Es handelt sich um eine exakte Beschreibung der Geometrie*
2. *Der Speicherbedarf für die einzelnen Objekte ist relativ gering im Vergleich zu einer polygonalen Repräsentation*

Der Nachteil dieser Repräsentation liegt im Berechnungsaufwand für die Visualisierung der Objekte. Die Visualisierung der parametrisch beschriebenen CAD-Objekte unterteilt sich in zwei Teilaufgaben, die Tesselierung und das eigentliche visuelle Rendering. Da es bei einem VR-System auf eine möglichst performante Visualisierung der Objekte ankommt, werden die VR-Objekte bereits in einer tesselierten Form verarbeitet, so daß lediglich der eigentliche Renderingprozeß die Performance bestimmt. Für das visuelle Rendering von Polygonen wiederum stehen effiziente Algorithmen zur Verfügung, die in speziellen Grafik-Subsystemen teilweise in Hardware gerechnet werden können. Als Beispiele dafür seien hier 3D-Grafik-Beschleunigerkarten auf dem PC-Sektor wie auch die Grafik-Supercomputer der ONYX™-Systemfamilie der Firma Silicon Graphics™ [67] genannt.

In den Abschnitten 3.4.2. und 3.4.3. wurde bereits die Notwendigkeit einer VR-systeminternen Repräsentation von CAD-Primitiven begründet. Sie bildet die Grundlage für eine erfolgreiche Rückführung der Ergebnisse einer Modellierung von virtuellen Leitungen in einem VR-System in die CAx-Welt. Für den Datenaustausch zwischen dem VR-System und der CAx-Umgebung ist weiterhin die Implementierung von Schnittstellenprozessoren notwendig, die die systeminternen CAD-Primitive importieren bzw. exportieren können. Dazu mußte eine Entscheidung für einen anzuwendenden Schnittstellenstandard getroffen werden. Dieser Standard sollte unabhängig von einem bestimmten CAD-System sein.

Unter Berücksichtigung der im Rahmen dieser Arbeit zur Verfügung stehenden heterogenen Systemumgebung wurde die Entscheidung getroffen, den IGES-Standard[68] für den direkten Datenaustausch zwischen der vorhandenen CAx-Umgebung und dem VR-System zu nutzen. Es handelt sich um ein neutrales CAD-Format, für das verschiedene Konvertierungsroutinen im Systemumfeld zur Verfügung stehen. So kann mit vorhandenen Konvertierungsroutinen aus einem IGES-Datensatz nahezu jedes CAD-Format erzeugt werden. Umgekehrt

[67] [SGI98]

[68] [IGES52]

kann nahezu jeder CAD-Datensatz in einen IGES-Datensatz konvertiert werden (Abbildung 19).

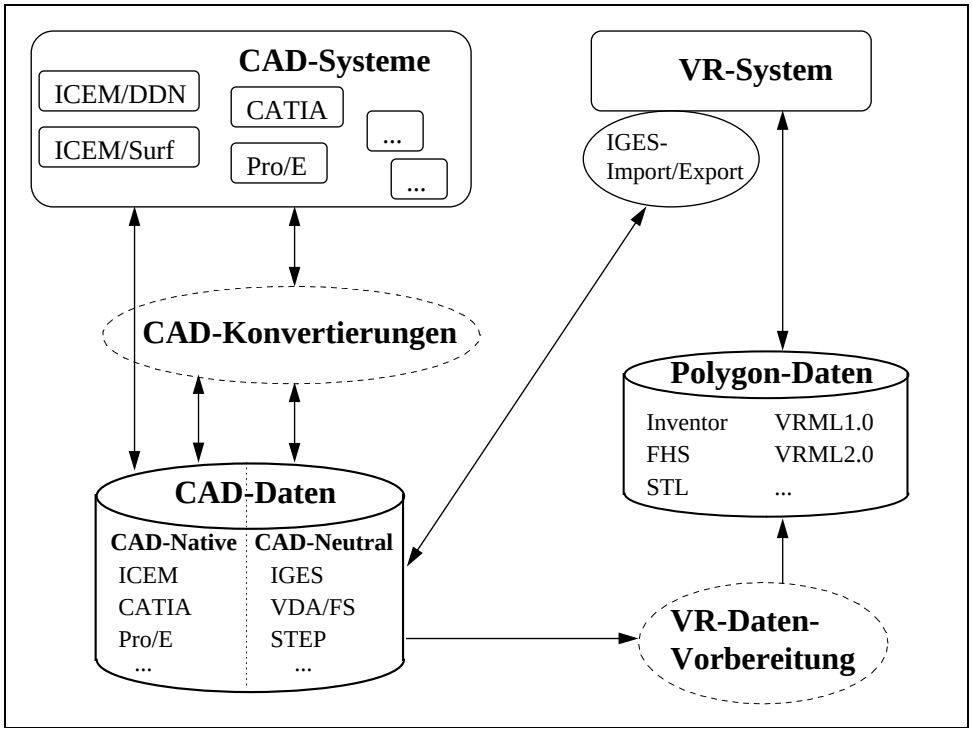


Abbildung 19: Bidirektionaler Datenaustausch zwischen einer heterogenen CAD-Umgebung und einem VR-System über den IGES-Standard

4.1.1. Der IGES-Standard als Austausch-Schnittstelle

Der IGES-Standard spezifiziert ein Produktdatenaustauschformat. Es wird zwischen drei verschiedenen Dateitypen unterschieden, die für die Datenspeicherung genutzt werden können. Dabei handelt es sich um ein IGES-ASCII-Format, ein komprimiertes ASCII-Format und ein IGES-Binärformat[69]. Die Produktdatenbeschreibung erfolgt auf der Basis der im IGES-Standard spezifizierten Informationseinheiten, die als Entities bezeichnet werden. Jeder Entitytyp ist durch eine eindeutige Entitynummer spezifiziert. Die Entities unterteilen sich in zwei Arten, die *Geometrieentities* und die *Nicht-Geometrieentities*. Die Geometrieentities repräsentieren die eigentliche geometrische Beschreibung der Objekte. Die Nicht-Geometrieentities reichern die Modelle mit Zusatzinformationen an, zu denen die Definition von Visualisierungseigenschaften (Sichtpunkt und -richtung, Linienstil, ...), Kommentaren, Dimensionierungen und Gruppierungen gehören[70].

[69] [RAB96] S.9

[70] [IGES52]

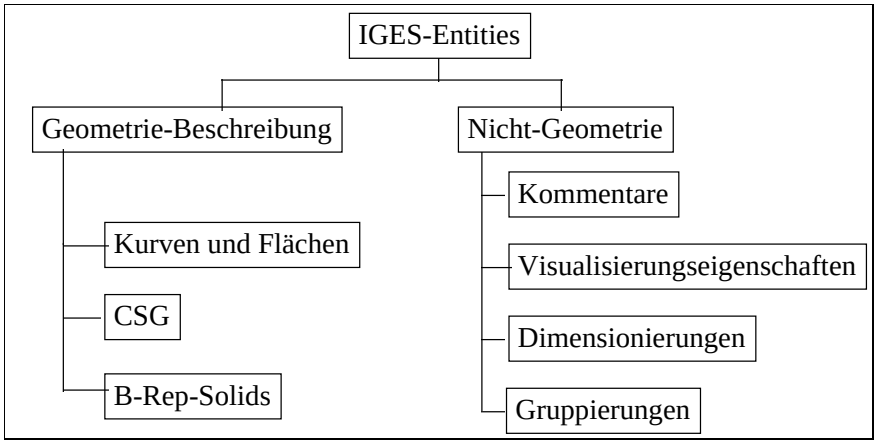


Abbildung 20: Entity-Klassen in IGES 5.2[71]

Für den Austausch der im VR-System erzeugten bzw. veränderten Geometrie kann man sich auf die Geometrieentities beschränken.

Entity-Type Number	Entity-Type	Entity-Type Number	Entity-Type
100	***Circular Arc***	122	Tabulated Cylinder
102	***Composite Curve***	***124***	***Transformation Matrix***
104	***Conic Arc***	125	Flash
106	Copious Data Linear Path Simple Closed Planar Curve	***126***	***Rational B-Spline Curve***
108	Plane	128	Rational B-Spline Surface
110	***Line***	130	Offset Curve
112	***Parametric Spline Curve***	140	Offset Surface
114	Parametric Spline Surface	141	Boundary
116	Point	142	Curve on a Parametric Surface
118	Ruled Surface	143	Bounded Surface
120	Surface of Revolution	144	Trimmed Parametric Surface

Tabelle 1: Entities der Kurven- und Oberflächengeometrie des IGES-Standards 5.2[72]

[71] [IGES52] S.3-4 Concepts of the File Structure, S.31 Classes of Entities

[72] [IGES52] S. 31

Der IGES-Standard unterscheidet wiederum zwischen drei verschiedenen Geometrieklassen:

- Primitive der Kurven- und Oberflächengeometrie
- Constructive Solids (CSG)
- B-Rep-Solids

Die Entities der Kurven- und Oberflächengeometrie enthalten die Beschreibungsprimitive, die auch für die Definition eines Leitungsobjektes von Interesse sind. Tabelle 1 enthält eine Auflistung der im Standard spezifizierten Entities mit den entsprechenden Entity-Nummern (Übernommen aus [IGES52]).

4.1.2. Linien-Entities als Definitionsbasis der Schlauchobjekte

In verschiedenen Interviews im Rahmen der Systemanalyse zur Leitungsverlegung (Abschnitt 2.2.1.3.) konnte festgestellt werden, daß der Verlauf einer Leitung in einem CAD-System durch ihre Mittellinie festgelegt wird. Diese Mittellinie kann aus verschiedensten CAD-Primitiven bestehen. Sie kann aus einem einzigen Primitiv oder aus einer Menge einzelner Primitive zusammengesetzt sein. Für die Repräsentation solcher Linienprimitive werden die in Tabelle 1 in Schrägschrift hervorgehobenen Entities durch den IGES-Standard zur Verfügung gestellt. Das Entity mit der Nummer 124 (Transformation Matrix) bildet dabei eine Ausnahme, da es sich nicht direkt um ein Linien-Entity handelt. Es ist jedoch für die freie Positionierung eines Entities im Raum notwendig und muß demzufolge durch das System unbedingt unterstützt werden.

Für die Repräsentation der Mittellinie einer Leitung wird in der vorliegenden Arbeit den IGES-Entities mit den Nummern 100 (Kreisbogen), 110 (Strecke) und 112 (parametrische Splinekurve) besondere Aufmerksamkeit geschenkt. Diese stehen für diese Aufgabe zur Verfügung und werden für die weiteren Untersuchungen näher betrachet.

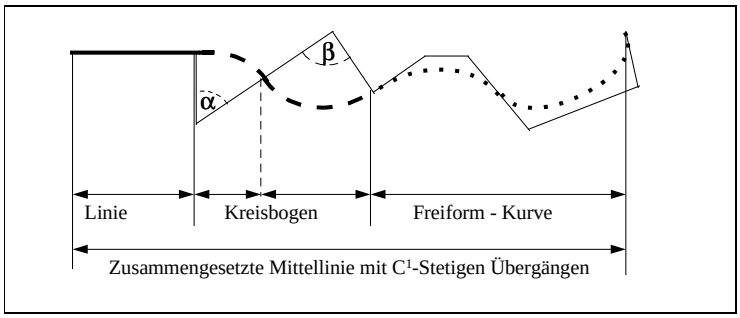


Abbildung 21: Aus einzelnen Linienentities zusammengesetzte Mittellinie

Die aus diesen Entities erzeugten Mittellinien der Leitungsobjekte sollen sich durch ihre C^1-Stetigkeit auszeichnen, was bei der Implementierung der Systemfunktionalität berücksichtigt werden muß. Es wurde versucht, diese Ei-

genschaft in Abbildung 21 zu verdeutlichen. Im Gegensatz dazu ist das Entity 102 (Composite Curve) lediglich auf eine C^0-Stetigkeit restriktioniert[73].

4.1.3. Integration der Entities in die VR-Datenstruktur

Unter Berücksichtigung der Entscheidung für die Unterstützung des IGES-Standards durch das VR-System wurden einzelne ausgewählte CAD-Primitive in das System integriert, die einer 1:1-Abbildung einzelner IGES-Entities entsprechen. Ein IGES-Entity ist dabei durch seinen Typ und die typspezifischen Parameterwerte vollständig beschrieben. Mit der Integration der CAD-Entities in das VR-System wird zwischen zwei grundlegenden logischen Objekttypen unterschieden. Dabei handelt es sich um die VR-Objekte und die CAD-Objekte, wobei die CAD-Objekte in die unterschiedlichen Entity-Typen unterteilt werden können[74]. Beide Objekttypen werden durch ihre geometrische Beschreibung im System repräsentiert, die in der Objekthierarchie des VR-Systems in Form von Polygonmengen vorliegen. Bei den VR-Objekten werden diese Polygonmengen im Rahmen der Datenaufbereitung erzeugt und während der Anwendung in ihrer Gestalt nicht weiter verändert. Diese Objekte können lediglich mit Hilfe von Transformationsmatrizen manipuliert werden, wodurch Translationen, Rotationen und Skalierungen interaktiv möglich sind.

Die CAD-Entities sind im System durch ihre Parameterwerte beschrieben, aus denen ihre polygonbasierte Approximation in der VR-Objekthierarchie eindeutig abgeleitet werden kann. Die Parameterwerte sind in der VR-Objekthierar-

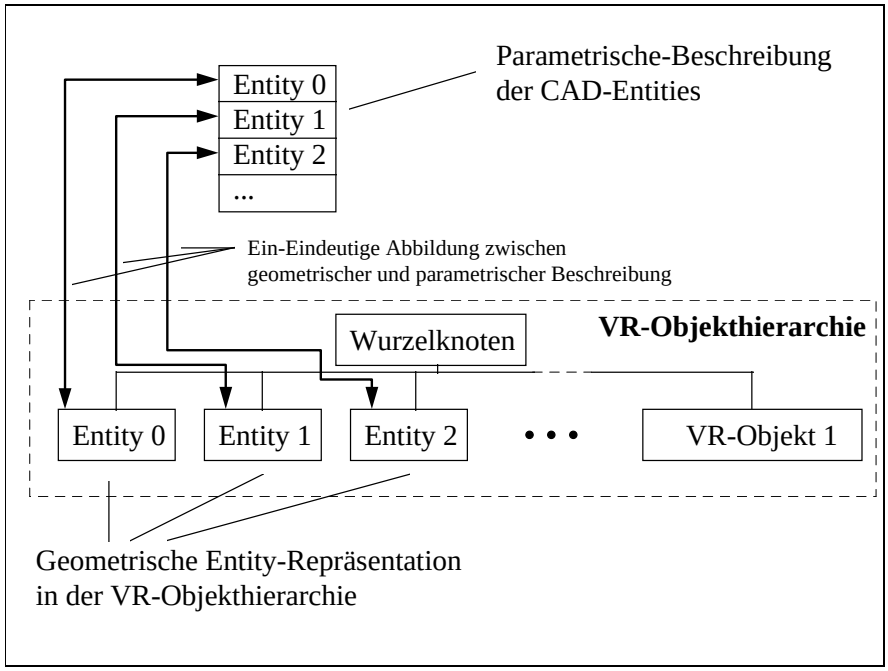


Abbildung 22: Parametrische und geometrische Beschreibung der CAD-Entities im VR-System

[73] [IGES52]

[74] [RAB96]

chie durch Punkte im Raum repräsentiert, die mit Hilfe einer einfachen Geometrie, den sogenannten Handles, visualisiert werden[75]. Diese Handles erfüllen zwei Aufgaben:

1. Sie dienen der Visualisierung und somit der Repräsentation eines Parameterpunktes in der virtuellen Umgebung
2. Über die Handles werden alle Interaktionen mit den CAD-Objekten definiert

Ebenso wie aus der parametrischen Beschreibung der CAD-Objekte, deren polygonbasierte Approximation in der VR-Objekthierarchie berechnet wird, kann aus den Handle-Objekten in der VR-Objekthierarchie auf den entsprechenden Parameterwert des CAD-Entities geschlossen werden (Abbildung 22). Daraus ergibt sich eine eindeutige Berechnungsreihenfolge während der Manipulation der CAD-Objekte. Im ersten Schritt werden aus der Handlegeometrie in der VR-Objekthierarchie die Parameterwerte des entsprechenden CAD-Objektes ermittelt und aktualisiert. Im zweiten Schritt wird die geometrische Repräsentation des CAD-Entities in der VR-Objekthierarchie neu berechnet.

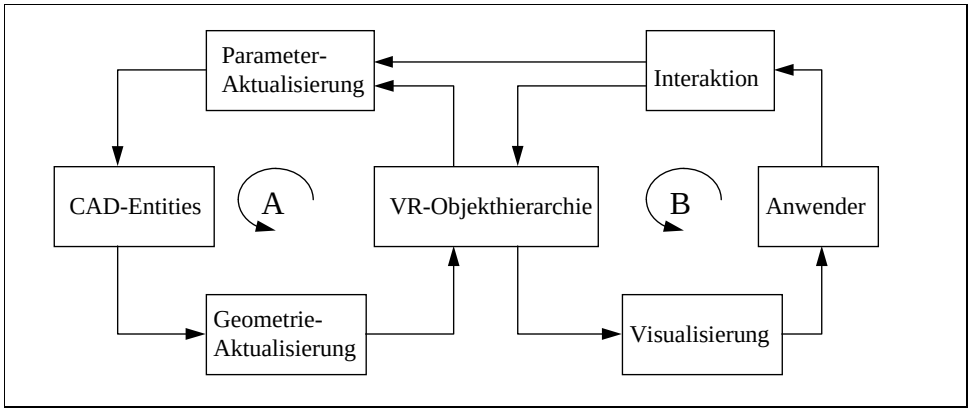


Abbildung 23: Informationsfluß in einem VR-System mit CAD-Objekten

Durch diese eindeutige Festlegung der Parameterableitung aus der Handlegeometrie ist die Veränderung eines CAD-Entities ausschließlich von der Handlegeometrie innerhalb der Objekthierarchie des VR-Systems abhängig (Abbildung 23). So ist es unerheblich, mit welcher Interaktion ein Handle neu positioniert wird.

Jedes CAD-Entity ist in der VR-Objekthierarchie durch einen Teilbaum beschrieben. Alle Objekte unterhalb des Wurzelknotens dieses Teilbaumes bilden die polygonbasierte Beschreibung des Entities, die bei jedem Schleifendurchlauf der Simulation visualisiert wird. Es sind drei verschiedene Operationen zu betrachten, die sich bezüglich des notwendigen Berechnungsaufwandes dieser CAD-Objekte unterscheiden. Diese sind eng mit den Interaktionen gekoppelt, die mit den CAD-Entities durchgeführt werden können. Dabei handelt es sich um das Erzeugen, Positionieren und Verändern der Entities.

[75] [RAB96]

Erzeugen

Das Erzeugen eines neuen CAD-Entities erfordert den größten Berechnungsaufwand. Die Entityerzeugung erfolgt bei der Neuerstellung (IGES-File einlesen oder interaktive Erzeugung) und bei der Erweiterung eines Entities. Zunächst werden die dynamischen Strukturen der parametrischen Beschreibung erstellt. Danach werden sie mit den entsprechenden Werten, die sich aus der direkten Eingabe in der VR-Umgebung oder durch die Dateidaten ergeben, berechnet und belegt.

Aus der parametrischen Beschreibung wird anschließend die Geometrie in der VR-Objekthierarchie erzeugt. Auch dieser Vorgang unterteilt sich wiederum in zwei Teilschritte. Im ersten Teilschritt werden die Datenstrukturen in der VR-Objekthierarchie erzeugt. Im zweiten Teilschritt werden die einzelnen Koordinaten entsprechend der Berechnung aktualisiert.

Die Erzeugung der Datenstrukturen in der VR-Objekthierarchie ist abhängig vom abgebildeten Entitytyp. Die Darstellung eines Punktes oder einer Linie gestaltet sich trivial, da sowohl die parametrische wie auch die geometrische Repräsentation im Dateninhalt übereinstimmen. Bei einem Kreisbogen und einer parametrischen Kurve hingegen wird der Verlauf durch einzelne Liniensegmente approximiert, sie werden tesseliert[76]. Da dieser Vorgang automatisch und schnellstmöglich ablaufen muß, wird auf eine krümmungsabhängige Tesselierung verzichtet. Eine Interpolation mit konstanter, krümmungsunabhängiger Schrittweite garantiert eine konstant bleibende Anzahl von Liniensegmenten. Damit kann bei einer Parameterveränderung auf eine neue Erzeugung der Strukturen in der VR-Objekthierarchie verzichtet werden. Der Nachteil dieser Vorgehensweise liegt in der relativ hohen Anzahl von Liniensegmenten, die für die Darstellung der Objekte notwendig sind. Nur mit dieser hohen Anzahl kann jedoch garantiert werden, daß auch bei einer starken Krümmung die Geometrie durch die Liniensegmente noch den Ansprüchen genügend exakt angenähert wird. Damit der Visualisierungsaufwand dennoch nicht zu hoch wird, werden die CAD-Entities auf eine feste Anzahl

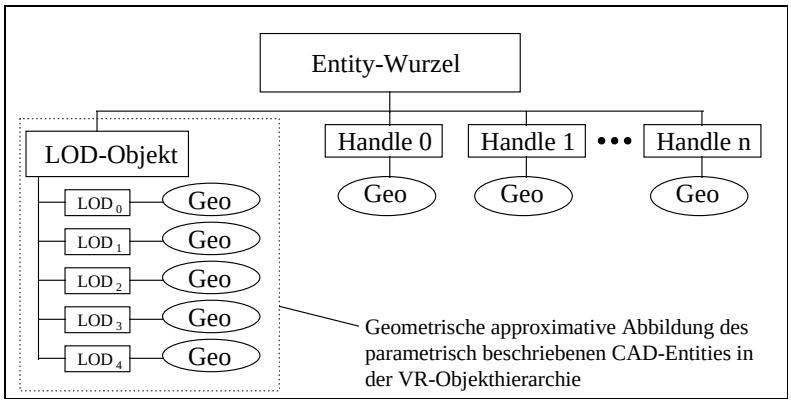


Abbildung 24: Entity-Teilbaum in der VR-Objekthierarchie

[76] [RAB96]

von Levels of Details (LOD) abgebildet.

Zusätzlich zu der geometrischen Abbildung der CAD-Entities wird bei deren Erzeugung die Handlegeometrie generiert. Auch diese Geometriegenerierung erfolgt ausschließlich bei der Erzeugung eines neuen Entities bzw. bei der Erweiterung eines solchen.

Ein Entity ist in der VR-Objekthierarchie durch einen Teilbaum repräsentiert, der den in Abbildung 24 dargestellten Aufbau besitzt.

Positionieren

Das Positionieren eines CAD-Objektes erfolgt direkt auf den geometriebeschreibenden Datenstrukturen in der VR-Objekthierarchie. Durch das Setzen einer Transformationsmatrix im Wurzelknoten des Entities wirkt sich die Positionierung direkt auf die geometrische Repräsentation in der VR-Objekthierarchie aus. Erst beim Dateiexport muß diese Transformationsmatrix für eine korrekte Abbildung des Objektes auf die parametrische Beschreibung angewendet werden. Damit sind keine Aktualisierungen der parametrischen Beschreibung während der Objektpositionierung notwendig, die die Performance des Systems zusätzlich belasten würden und die bei der Implementierung berücksichtigt werden müßten. Damit ist die Transformation des Entities ausschließlich in der VR-Objekthierarchie abgelegt. Das hat zur Folge, daß sich die Objekttransformation durch die Renderingkomponente des VR-Systems direkt auf die aus der parametrischen Beschreibung abgeleitete VR-Datenstruktur auswirkt. Die Korrektheit dieser Vorgehensweise ist durch die Eigenschaft der affinen Invarianz der verwendeten parametrischen Datenstrukturen gesichert[77].

Verändern

Während sich bei der Positionierung der Entities lediglich die Transformation des Objektes ändert, die interne parametrische Beschreibung hingegen nicht, ist im Fall der Veränderung eines CAD-Entities eine Neuberechnung der geometrischen Repräsentation in der VR-Objekthierarchie notwendig. Diese Aktualisierung erfolgt bei jedem Durchlauf der Simulationsschleife des VR-Systems. Der Informationsfluß ist in Abbildung 23 (Schleife A) abgebildet. Die Interaktionskomponente gibt der für die Geometrieaktualisierung zuständigen Komponente diejenigen Entities bekannt, die gerade verändert werden. Damit müssen nicht alle Entities einer VR-Umgebung auf eine Parameterveränderung getestet und auch nicht alle Entities bei jedem Bild neu berechnet werden. Durch die oben erläuterte (siehe Entityerzeugung) krümmungsunabhängige Tesselierung der parametrischen Geometrie bleibt der Aufbau der Datenstrukturen in der VR-Objekthierarchie unabhängig von den Parameterwerten der CAD-Entities. Damit beschränkt sich die Aktualisierung der CAD-Objekte im Fall der Veränderung auf eine reine Berechnung von Punktkoordi-

[77] [FARIN94]

naten, ohne eine Zerstörung und Neuerzeugung von Datenstrukturen in der VR-Objekthierarchie.

4.1.4. Die Anwenderschnittstelle

Die durch das System zur Verfügung gestellte Anwenderschnittstelle und deren Funktionen wurden in [RAB96] beschrieben[78,79]. Die gesamte darin erläuterte Implementierung orientiert sich an der Manipulation einzelner Entities und deren Parameter. Die Manipulation erfolgt frei im Raum über die geometrische Repräsentation eines „virtuellen Greifobjektes“, das bei der Nutzung eines Datenhandschuhs typischerweise durch die virtuelle Hand oder einen Teil der virtuellen Hand repräsentiert ist[80]. Es kann zwischen drei Funktionsklassen,

- Generierungsfunktionen
- Manipulationsfunktionen
- Systemkonfigurationsfunktionen

unterschieden werden, die sich durch die verschiedenen Einsatzweisen der virtuellen Hand voneinander abgrenzen. Bei der Generierung neuer Entities werden mit Hilfe der virtuellen Hand Positionen im Raum definiert. Sie dient als dreidimensionaler Pointer im Raum. Bei den Manipulationsfunktionen dient sie

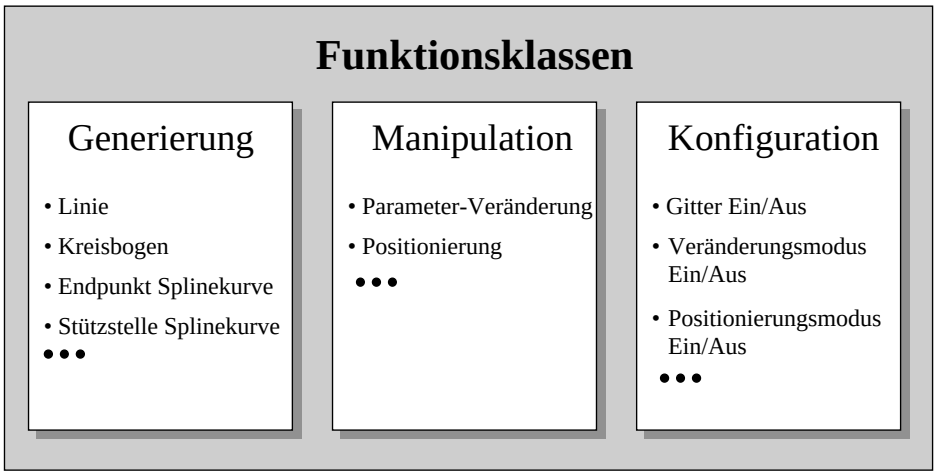


Abbildung 25: Verschiedene Funktionsklassen der Entity-Anwenderschnittstelle

sowohl als dreidimensionaler Pointer wie auch als Selektionswerkzeug, das der Auswahl des zu verändernden virtuellen Objektes dient. Während bei den

[78] [RAB96] S.35-40

[79] Aus Gründen des Arbeitsumfanges wird hier lediglich auf die für die Leitungsverlegung als wesentlich erachteten Objekttypen und Funktionalitäten eingegangen, während in [RAB96] ein größerer Funktionsumfang realisiert wurde.

[80] Im Folgenden soll für die Begriffe „virtuelles Greifobjekt“ oder „Greifer“ der Begriff „virtuelle Hand“ genutzt werden, da das „Greifobjekt“ in dem genutzten System standardmäßig durch eine Handrepräsentation dargestellt wurde. Es sei jedoch darauf hingewiesen, daß die geometrische Repräsentation dieses Objektes keinen Einfluß auf die Systemfunktionalität hat.

Generierungs- und den Manipulationsfunktionen die virtuelle Hand Voraussetzung für eine direkte Manipulation der Entities ist, werden bei den Systemkonfigurationsfunktionen die virtuellen Objekte nicht direkt verändert. Damit ist die Eingabe auch nicht zwingend mit der virtuellen Hand gekoppelt.

Die einzelnen Funktionalitäten werden über das in Abschnitt 3.2.4. beschriebene Action-Event-Prinzip ausgelöst.

Die *Generierungsfunktionen* orientieren sich an der parametrischen Beschreibung der einzelnen Entitytypen. Mit Hilfe eines Funktionsaufrufes wird genau ein Wert eines Entityparameters festgelegt. Dieser ergibt sich aus den Koordinaten der virtuellen Hand. Die Erzeugung eines Entities setzt sich aus der Erzeugung mehrerer CAD-Parameter zusammen. Beispielsweise ergibt sich damit für die Generierung einer Strecke die Notwendigkeit für zwei Funktionsaufrufe, jeweils einer für die Definition des Start- bzw. Endpunktes. Für unterschiedliche Entities werden unterschiedliche Generierungsfunktionen zur Verfügung gestellt. Während sich die Anzahl der zu definierenden Parameter

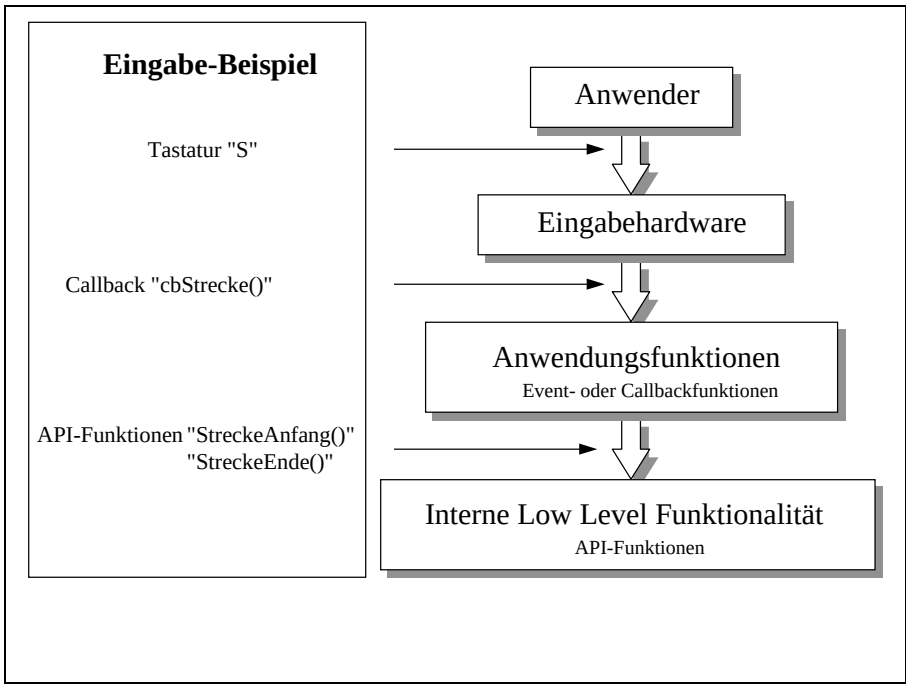


Abbildung 26: Schichtenmodell der Anwenderschnittstelle

bei einer Strecke und einem Kreisbogen nicht ändern, können sich die Splinekurven in ihrer Stützstellenanzahl unterscheiden. Daher ist eine Unterscheidung zwischen der Generierung von Anfangs- und Endpunkt und der Erzeugung der dazwischenliegenden Stützstellen notwendig. In Abbildung 26 ist die Unterscheidung der einzelnen Funktionalitäten bezüglich der unterschiedlichen Schnittstellenschichten dargestellt. Dem Anwender stellen sich die unterschiedlichen Funktionen durch die unterschiedlichen physikalischen Ereignisse (Tastendruck, Spracheingabe, ...) dar, auf die sie abgebildet sind. Darunter liegen die eigentlichen Interaktionsfunktionen (Event- oder Callback-

routinen), die wiederum interne zum API[81] gehörende Funktionen nutzen. Statusvariablen sorgen dafür, daß die internen Funktionen in der richtigen Reihenfolge abgearbeitet werden, wodurch das gleiche Eingabeereignis auf unterschiedliche interne Funktionen abgebildet werden kann. Damit reduziert sich die sich dem Anwender darstellende Menge unterschiedlicher Eingabeereignisse für die Generierung der Entities und entspricht im Minimum der Anzahl der zur Verfügung stehenden Entitytypen.

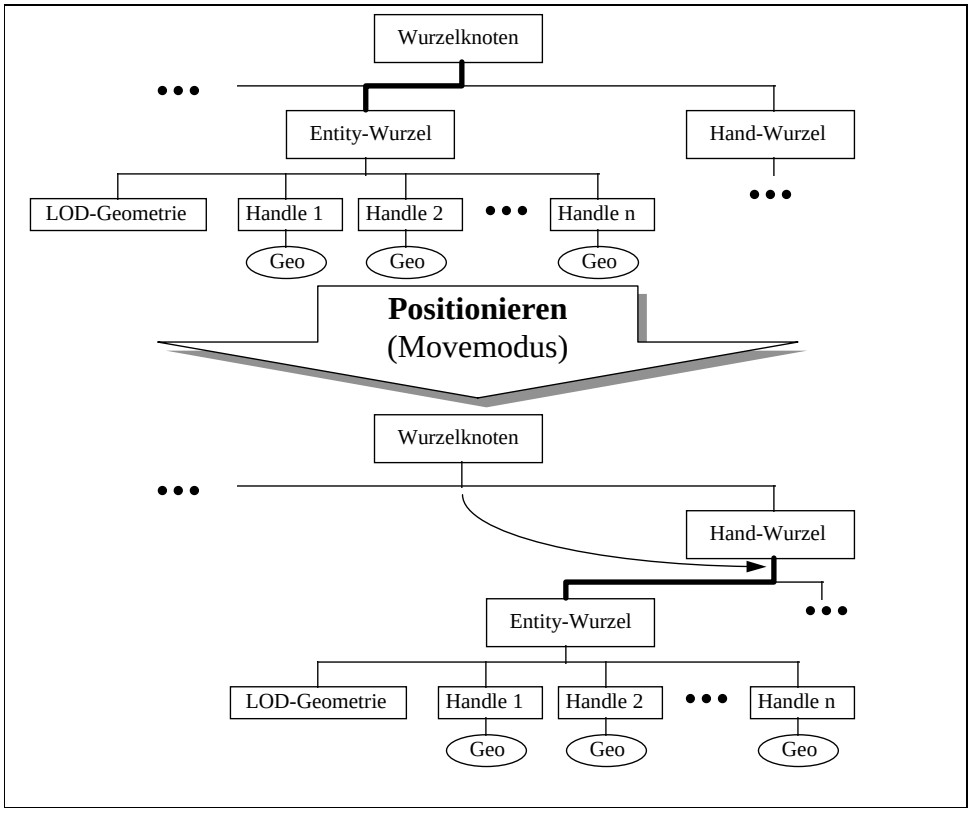


Abbildung 27: Veränderung in der Objekthierarchie beim Positionieren eines Entities

Über die *Manipulationsfunktionen* können die Entities direkt verändert werden. Diese direkte Veränderung erfolgt stets über die Handlegeometrie, die die einzelnen Parameterwerte eines Entities in der virtuellen Umgebung repräsentieren. Über die Handle kann sowohl das gesamte Entity, als auch der durch das Handle repräsentierte einzelne Parameterwert selektiert und positioniert werden. Die Selektion erfolgt durch eine Kollisionsuntersuchung zwischen der virtuellen Hand und den Handles. Dafür stehen zwei verschiedene Manipulationsmodi zur Verfügung. Die Entities können frei im Raum positioniert werden, was bereits im Abschnitt 4.1.3. unter dem Begriff „Positionieren“ erläutert wurde. Diese Manipulationsmöglichkeit wird im Folgenden als VR- oder *Movemodus* bezeichnet. Im CAD- oder *Changemodus* hingegen werden die Parameterwerte der Entities direkt verändert (siehe Abschnitt 4.1.3. „Verändern“). Beide Interaktionsmöglichkeiten werden auf eine Greifoperation zurückgeführt, über die das jeweils gegriffene Objekt positioniert wird. Bei dieser Greifoperati-

[81] API: **A**dvanced **P**rogramming **I**nterface, steht im allgemeinen in Form einer Funktionsbibliothek zur Verfügung

on kommt ein sehr einfaches Verbindungsmodell zum Einsatz, wie es bereits in Abschnitt 3.5.1.2. beschrieben wurde. Das gegriffene Objekt wird innerhalb der VR-Objekthierarchie unter das Greifobjekt positioniert, wodurch es mit dessen Bewegung mitgeführt wird. Im Movemodus handelt es sich dabei um die gesamte Entityrepräsentation in der VR-Objekthierarchie (Abbildung 27). Im Changemodus wird lediglich das selektierte Handleobjekt unter die virtuelle Hand positioniert (Abbildung 28). Die dem Anwender zur Verfügung stehende Manipulationsfunktionalität ist unabhängig vom Entitytyp implementiert und erfolgt bei jedem Entity einheitlich durch eine direkte Manipulation des Objektes.

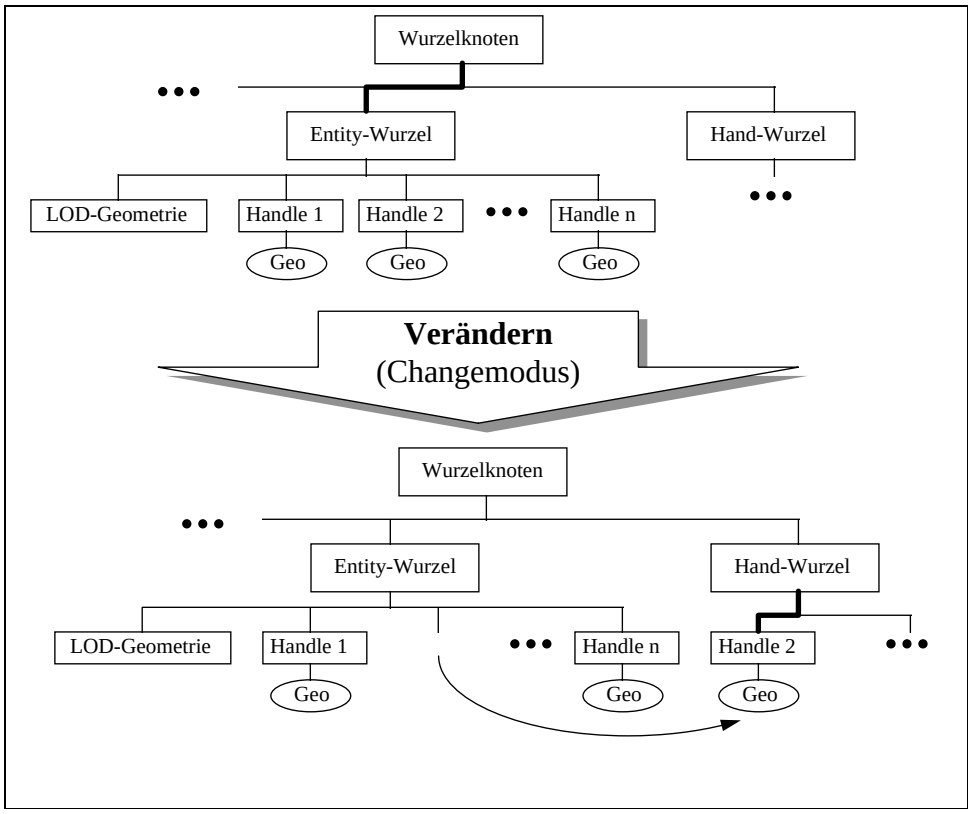


Abbildung 28: Veränderung in der Objekthierarchie beim Verändern eines Entities

Um den Move- bzw. Changemodus ein- und ausschalten zu können stehen dem Anwender Funktionen zur Verfügung, die zu den *Konfigurationsfunktionen* zuzuordnen sind. Durch sie werden die entsprechenden Modi ein- bzw. ausgeschaltet. Genau ein Modus kann zu einem Zeitpunkt aktiv sein. Demzufolge kann sich das System im Changemodus, im Movemodus oder im neutralen Modus befinden, wobei der letztgenannte Modus eintritt, wenn keiner der beiden Manipulationsmodi aktiv ist. Dieser neutrale Modus ist erforderlich, um eine unbeabsichtigte Veränderung eines Entities, zum Beispiel während einer Navigation durch die virtuelle Szene, zu vermeiden.

In der Standardeinstellung des Systems wird davon ausgegangen, daß die Entities mit der rechten Hand bearbeitet werden. Darüber hinaus ist es möglich, mit der linken Hand die Entities zu verändern bzw. mit beiden Händen

gleichzeitig zu agieren. Auch dafür stehen verschiedenste Systemkonfigurationsfunktionen zur Verfügung, mit deren Hilfe zwischen diesen Möglichkeiten gewechselt werden kann. Eine weitergehende Beschreibung dazu ist in [RAB96] zu finden. Es soll hier nicht näher darauf eingegangen werden.

Auch die Funktionen der Dateischnittstelle, die einen Import wie auch Export der Entitydatenobjekte erlauben, sollen hier zusammen mit den Konfigurationsfunktionen erwähnt werden. Diese Funktionen beziehen sich stets auf die Gesamtheit der im System existierenden Entities. Über sie können die CAD-Entities als IGES-Datensätze gelesen bzw. geschrieben werden[82].

Zusammenfassend kann festgestellt werden, daß durch die Anwenderschnittstelle Basisfunktionalitäten zur Verfügung stehen, über die die einzelnen CAD-Entities direkt erzeugt und verändert werden können. Alle Manipulationen erfolgen frei im Raum und beeinflussen die geometrische Beschreibung jeweils eines einzelnen Entities. Es bestehen keinerlei Bewegungsbeschränkungen während der Manipulation der Objekte. Weiterhin werden Funktionen zur Verfügung gestellt, über die zwischen verschiedenen Manipulationsmodi gewechselt werden kann, die einen Wechsel der jeweils aktiven Hand erlauben und die für den Datenimport und -export über die IGES-Schnittstelle sorgen. Damit stehen Basisfunktionen zur Verfügung, um Leitungsobjekte, die durch ihre Mittellinie definiert sind, im Raum zu erzeugen, diese zu verändern und die Ergebnisse dieser Modellierung in einem neutralen Schnittstellenformat zu exportieren.

Die Implementation der Anwenderschnittstelle erfolgte unabhängig von der eingesetzten Interaktionshardware. So besteht die Möglichkeit, die verschiedensten Eingabegeräte und Navigationsmodi, die mit dem eingesetzten VR-System Virtual Design II zur Verfügung stehen, im Zusammenhang mit den beschriebenen Funktionalitäten zu testen.

4.1.5. Erweiterung um Hüllgeometrie

Über die parametrisch beschriebenen Linienobjekte kann der Verlauf eines Leitungsobjektes in einer virtuellen Umgebung festgelegt werden. Beschränkt man sich jedoch auf diese Linienobjekte für die Beschreibung einer Leitung in einer virtuellen Umgebung, ist die Abschätzung des benötigten Bauraumes für eine Leitung schwierig. Eine Leitung ist in der Realität nicht nur durch den Verlauf ihrer Mittellinie, sondern auch durch ihre räumliche Ausdehnung beschrieben. Es ist zwar nicht das Ziel des hier beschriebenen Systems, die Leitungsobjekte exakt mit allen geometrischen Eigenschaften zu konstruieren, dennoch soll es die Möglichkeit bieten, ein Grobmodell einer Leitung inklusive ihrer Hüllgeometrie für die Visualisierung und Analyse zur Verfügung zu stellen. Auf diese Weise soll bereits in der virtuellen Umgebung die Möglichkeit bestehen, Kollisionen zwischen der Leitungsgeometrie und der Umgebung einfach durch Sichtkontrolle zu erkennen. Auch eine vom System durchgeführte Kollisionsuntersuchung soll mit einer solchen Geometrie ermöglicht werden. Für das Erreichen dieses Zieles wurden zusätzliche Funktionen im-

[82] [RAB96]

plementiert, die basierend auf der Entityrepräsentation in der Objekthierarchie des VR-Systems (siehe Abbildung 24) Hüllgeometrien mit einem Kreisquerschnitt erzeugen und manipulieren können. Da nicht das Ziel besteht, diese einfachen Hüllgeometrien als parametrisch beschriebene Objekte exportieren zu können, basieren diese Funktionen auf der rein polygonalen Beschreibung der Entities in der Objekthierarchie des VR-Systems.

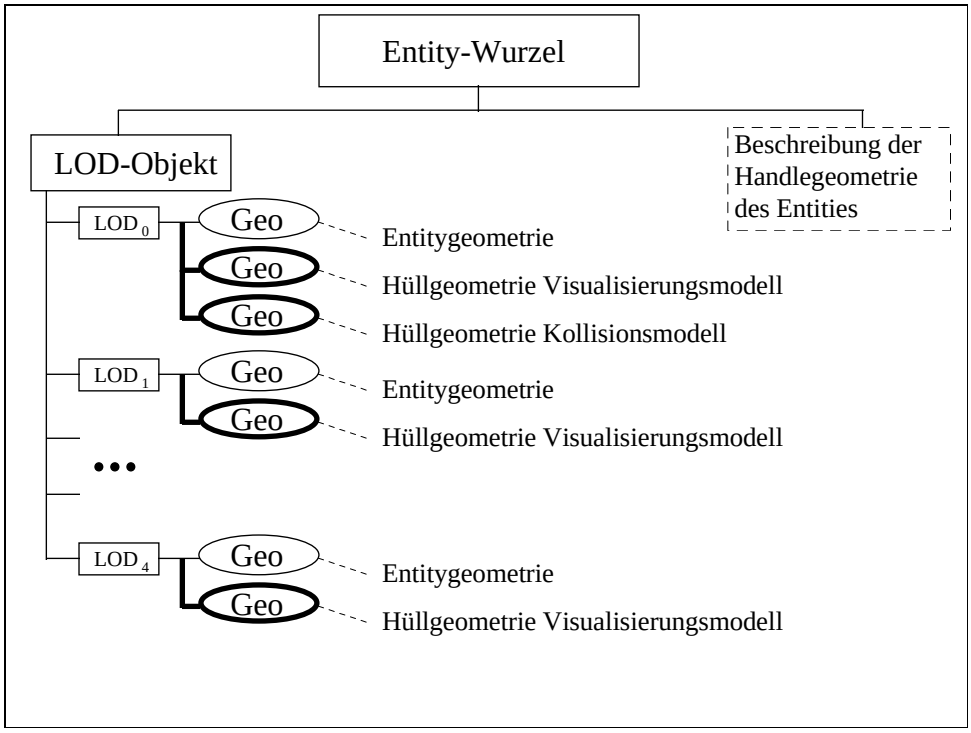


Abbildung 29: Entityteilbaum mit Hüllgeometrie in der VR-Objekthierarchie

Bei der Implementierung wurde berücksichtigt, daß die Hüllgeometrien entsprechend der Entitygeometrie selbst als LOD-Objekte erzeugt werden. Weiterhin wird zwischen einem reinen Visualisierungsmodell und einem Kollisionsmodell unterschieden. Berechnungsaufwand wie auch Modellgröße wurden minimal gehalten, damit die Repräsentation der Hüllgeometrie die Echtzeitfähigkeit des Systems nicht beeinträchtigt. Der in Abbildung 24 schematisch dargestellte Aufbau eines Entities in der Objekthierarchie des VR-Systems wurde daher entsprechend Abbildung 29 erweitert. Jede LOD-Repräsentation der Entitygeometrie enthält damit auch eine Repräsentation der Hüllgeometrie. In der höchstaufgelösten LOD-Stufe (LOD_0) ist zusätzlich ein Kollisionsmodell enthalten.

4.1.5.1. Funktionsumfang und Realisierung

Für die Erweiterung der Entityrepräsentation um die entsprechende Hüllgeometrie sind verschiedene Funktionen implementiert worden. Die Generierungsfunktionen erzeugen die für die Hüllgeometrie notwendigen Geometriebeschreibungen in der VR-Objekthierarchie. Im Fall der Veränderung eines Entities (Changemodus, siehe auch Abschnitt 4.1.4.) muß auch die Hüllgeometrie bei jedem Bild aktualisiert werden. Dabei bleiben die bei der Generierung

der Hüllgeometrie erzeugten Datenstrukturen erhalten, lediglich ihr Inhalt ändert sich, er wird neu berechnet. Darüber hinaus wurden Funktionen implementiert, mit deren Hilfe der Radius der Hüllkörper gesetzt werden kann.

Für jeden linienbasierten Entitytyp (Strecke, Kreisbogen, parametrische Splinekurve, siehe auch Abschnitt 4.1.2.) existiert jeweils eine dieser Funktionen. Aufgrund der verschiedenen Berechnungsalgorithmen ist eine entityspezifische Unterscheidung dieser Funktionen notwendig.

Als Grundlage für die Berechnung der geometrischen Beschreibung der Schlauchhülle dient die geometrische Beschreibung der Entities in der VR-Objekthierarchie. Wie in Abbildung 24 zu sehen ist, ist jedes Entity in der VR-Objekthierarchie durch eine feste Anzahl von LODs repräsentiert. Im Fall der hier betrachteten Linienentities handelt es sich bei jedem einzelnen LOD um eine sogenannte Polyline, die durch eine Menge geordneter Punkte[83] beschrieben wird. Diese Punkte sind jeweils durch eine Strecke miteinander verbunden. Die Polyline des LOD_0 stellt dabei die am höchsten aufgelöste Geometrie dar. Die Idee besteht nun darin, diese Linienobjekte (Polylines) durch Zylinder mit einem festen Radius zu umhüllen. Diese Zylinder werden aus einer Menge von Polygonen erzeugt, mit deren Hilfe ihre exakte Form approximiert wird. Die Genauigkeit dieser Approximation nimmt dabei mit ansteigendem LOD ab, so wie das auch bei den Polylines der Fall ist. Durch diese Herangehensweise ist die Hüllgeometrie eines linienbasierten Entities mit der in der VR-Umgebung existierenden Polyline und einem zugeordneten Radius eindeutig definiert.

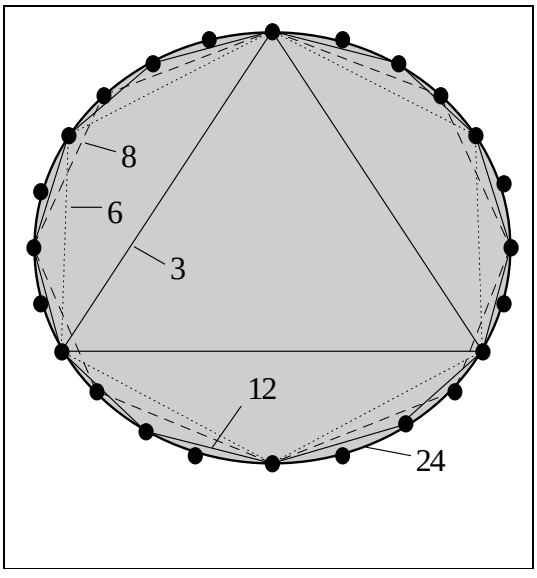


Abbildung 30: Kreisapproximation mit 3,6,8,12,24 Liniensegmenten und insgesamt 24 verschiedenen Punktkoordinaten

Für die Berechnung der Hüllgeometrie werden die drei grundlegenden Linienentities Kreisbogen, Strecke und parametrische Splinekurve unterschieden. Für alle drei Typen wurde eine vom LOD abhängige Auflösung der Außenhülle festgelegt. Der kreisförmige Querschnitt des Zylinders wird dabei in der höchsten Auflösung durch 24 miteinander verbundene Punkte angenähert. Die weiteren LODs bestehen aus 12, 8, 6 bzw. 3 Punkten. Der Vorteil dieser Auflösungen besteht darin, daß mit der Berechnung der am höchsten aufgelösten Geometrie (24 berechnete Punkte) auch die Koordinaten der

[83] Es sei hier angemerkt, daß ein Punkt dabei nicht nur durch seine Koordinaten im Raum beschrieben ist, sondern auch durch seine Normale, Farbe und Texturkoordinate. Bei diesen Betrachtungen jedoch ist zunächst lediglich seine Raumkoordinate im R^3 von Interesse.

niedriger aufgelösten Geometrien bereits berechnet sind. Die entsprechenden Punktkoordinaten werden bei der Generierung bzw. Aktualisierung der Geometrie lediglich kopiert. Das gleiche gilt für die Berechnung der Punktkoordinaten des Kollisionsmodells der Hüllgeometrie. Auch diese Punkte werden aus der geometrischen Beschreibung des höchst aufgelösten Visualisierungsmodells der Hüllgeometrie übernommen.

Auf Einzelheiten der Implementierung der Funktionen wird an dieser Stelle verzichtet.

4.1.6. Fazit

Durch die oben beschriebene Erweiterung des VR-Systems um CAD-Entities ist es möglich, in der VR-Umgebung parametrisch beschriebene Linienobjekte zu erzeugen, die den Verlauf eines Leitungsobjektes festlegen. Dieser Verlauf kann über den IGES-Standard exportiert werden und so als Grundlage für eine exakte Konstruktion in einem CAD-System dienen. Durch die Erweiterung der Linienobjekte um eine Hüllgeometrie, die durch einen Kreisquerschnitt mit definiertem Durchmesser festgelegt ist, kann in der virtuellen Umgebung ein realitätsnaher Eindruck eines Leitungsobjektes erzeugt werden. Dadurch gelingt es auf rein visuellem Weg leichter zu beurteilen, ob geometrische Durchdringungen zwischen den Leitungsobjekten und den Objekten der Umgebung existieren. Auch ein Test des Leitungsobjektes auf Kollisionen mit der Umgebung ist möglich, da die Hüllgeometrie auch für berechnete Kollisionstests genutzt werden kann.

Alle direkten Interaktionen mit den CAD-Entities basieren auf drei Grundfunktionalitäten, dem Erzeugen, Positionieren und Verändern. Die Erweiterung der Entities um die Hüllgeomterie hat den Berechnungsaufwand für diese Operationen erhöht. Diese Erhöhung der Berechnungszeit ermöglicht jedoch nach wie vor die direkte Manipulation der Objekte im Raum unter den Echtzeitbedingungen, die durch ein VR-System definiert sind.

Alle Manipulationen der CAD-Entities erfolgen frei im Raum direkt über die Handlegeometrie. Diese Interaktionen beziehen sich immer auf einzelne Entities. Diese Interaktionseigenschaften haben Vorteile wie auch Nachteile. Die *Vorteile* liegen in der Möglichkeit, den Verlauf eines solchen Objektes direkt im Raum festgelegen zu können. Es wird auf die Eingabe von exakten Koordinaten ebenso verzichtet wie auf Interaktionsbeschränkungen, die beispielsweise durch die Definition und Anwendung von Arbeitsebenen oder ähnlichen Hilfsmitteln entstehen.

Ein wesentlicher *Nachteil* liegt darin, daß die Manipulation zusammengesetzter Objekte durch die bis hier beschriebenen Interaktionsmöglichkeiten nicht unterstützt wird. Das hat zur Folge, daß es nicht möglich ist, eine Mittellinie eines Leitungsobjektes aus mehreren einzelnen CAD-Entities zusammenzusetzen und auch in ihrem zusammengesetzten Zustand zu manipulieren. Weiterhin besteht in einer solchen Umgebung auch die Notwendigkeit, ein erzeugtes Leitungsobjekt mit einem Objekt der virtuellen Umgebung zu verbinden. Soll beispielsweise der Verlauf eines Kühlwasserschlauches in einer virtuellen

Umgebung festgelegt werden, so müssen seine Schlauchenden mit dem Kühler wie auch mit der Wasserpumpe verbunden werden. Diese beiden Objekte sind in der Umgebung als virtuelle Objekte vorhanden, die durch eine Menge von Polygonen in dem System abgebildet sind. Mit dem bis hier beschriebenen System ist es lediglich möglich, den Verlauf des CAD-Entities so zu manipulieren, daß der visuelle Eindruck einer Verbindung entsteht. Die einzige Kontrollmöglichkeit dafür ist die visuelle Kontrolle des Anwenders in der Umgebung. Eine fehlende Kraftrückkopplung und die Ungenauigkeit der verschiedenen existierenden VR-Eingabegeräte führt dabei immer zu Ergebnissen, die für das exakte Festlegen eines Leitungsverlaufes ungenügend sind.

Es sei an dieser Stelle angemerkt, daß dieses Problem nicht allein dadurch zu lösen ist, indem die Eingabehardware wesentlich verbessert wird und so die Positionierung der Objekte zueinander exakter erfolgen kann. Der Aufbau einer Verbindung zwischen zwei Objekten erfolgt auch in der Realität nicht ausschließlich auf der Basis einer visuellen Kontrolle. Durch die physikalischen Eigenschaften der Objekte und den daraus resultierenden Kräften zwischen diesen wird deren direkte Manipulation wesentlich beeinflußt. Sie bilden einen Regelkreis aus Kraft und Gegenkraft, wie er im Kapitel 3.5.1.1. anhand der Realisierung des SensorGlove beschrieben wurde.

Zusammenfassend ist also festzustellen, daß das bis hier beschriebene Ergebnis der Arbeit einen unzureichenden, nicht zufriedenstellenden Stand darstellt. Die Gründe dafür sind in zwei wesentlichen Zielen zu finden, die bis hier nicht erreicht wurden:

Verbindungserzeugung

Zum einen soll es das System erlauben, verschiedene CAD-Entities so exakt zu positionieren, daß der visuelle Eindruck einer Verbindung entsteht. Dazu müssen die Endpunkte der Linienobjekte und der Anstieg in diesen Endpunkten übereinstimmen. Diese Verbindungen sollen zwischen den CAD-Entities selbst wie auch zwischen einem CAD-Entity und einem Anschlußstutzen eines Objektes der virtuellen Umgebung möglich sein.

Verbindungsberücksichtigung

Zum anderen sollen Verbindungen zwischen verschiedenen Objekten durch das System automatisch erkannt und bei den weiteren Interaktionen auch mit berücksichtigt werden können.

4.2. Verbindungsstutzen als Grundlage verbindungsorientierter Funktionen

Für die Berücksichtigung von Verbindungen zwischen verschiedenen virtuellen Objekten wird eine Datenstruktur, die Verbindungsstutzen, eingeführt. Sie bilden die geometrische Grundlage für alle verbindungsorientierten Funktionalitäten in dem System. Alle Verbindungsstutzen haben einen einheitlichen geometrischen Aufbau. Weiterhin sind sie bestimmten virtuellen Objekten zugeordnet. Es handelt sich dabei um eine sowohl logische wie auch hierarchische Zuordnung der Objekte zueinander. Mit Hilfe der Verbindungsstutzen werden Informationen in das Datenmodell integriert, die zusätzlich zu der aus CAD-Daten gewonnenen geometrischen Beschreibung der virtuellen Objekte in das System mit einfließen. Über diese Zusatzinformationen wird das technische System in die Lage versetzt, räumliche Beziehungen zwischen verschiedenen Objekten schnell und eindeutig zu analysieren. Das Ergebnis dieser Analysen wird dazu genutzt, das Simulationsverhalten des Systems zu beeinflussen.

Weiterhin können die Informationen über räumliche Beziehungen zwischen verschiedenen Verbindungsstutzen dazu genutzt werden, Bewegungsconstraints der virtuellen Objekte zu definieren und umzusetzen. Über diese Bewegungsconstraints können Nutzerinteraktionen so beeinflußt werden, daß als Ergebnis einer solchen Interaktion definierte Verbindungsrelationen entstehen. Durch den Einsatz solcher Interaktionsmodelle kann auf eine physikalisch korrekte Abbildung verschiedenster Wechselwirkungen zwischen den virtuellen Objekten verzichtet werden. Das hat weiterhin zur Folge, daß auf die für eine physikalisch basierte Umsetzung von Verbindungsoperationen notwendigen Eingangsinformationen (physikalische Konstanten, werkstoffabhängige Meßwerte, spezielle Geometrie- und Simulationsmodelle) verzichtet werden kann.

Im folgenden Abschnitt wird der geometrische Aufbau der Stutzen näher erläutert. Die Integration dieser Zusatzgeometrie in das Geometriemodell der virtuellen Objekte ist Thema des Abschnittes 4.2.2.. Die auf den Stutzen basierenden Constraints und die darauf aufbauenden Abläufe bei der Erzeugung und der Trennung von Verbindungen werden im Abschnitt 4.2.3. erläutert.

4.2.1. Geometrischer Aufbau eines Stutzens

Bei den Stutzen wird zwischen dem geometrischen und dem semantischen Modell unterschieden. Das semantische Modell wird im Abschnitt 4.3. erläutert werden. In diesem Abschnitt spielt ausschließlich die geometrische Ausprägung der Stutzen eine Rolle.

Die geometrische Beschreibung eines Stutzens ist in der VR-Objekthierarchie abgelegt. Ein Stutzen hat immer den gleichen Aufbau. Er besteht aus zwei geordneten Punkten im Raum. Aufgrund der Punktreihenfolge kann zwischen Start- und Endpunkt eindeutig unterschieden werden. Der durch Start- und Endpunkt definierte Ortsvektor wird im Folgenden als Stutzenvektor bezeichnet. Zusätzlich umgibt diese zwei Punkte eine Kollisionsgeometrie beliebiger Ausprägung. In Abbildung 31 wurde beispielhaft eine zylinderförmige Kolli-

sionsgeometrie dargestellt. Ein Stutzen hat eine eindeutige Namensgebung. Über diese Namen können der Stutzenwurzelknoten, der Stutzenvektor und die Kollisionsgeometrie referenziert werden. Durch den einheitlichen Aufbau der Stutzen kann eindeutig auf die einzelnen Komponenten eines Stutzens zugegriffen werden. Die Eindeutigkeit dieses Zugriffes ist Voraussetzung für eine autonome Analyse der Verbindungssituation in einer virtuellen Umgebung. Auf ihr bauen die verschiedensten implementierten Funktionalitäten auf.

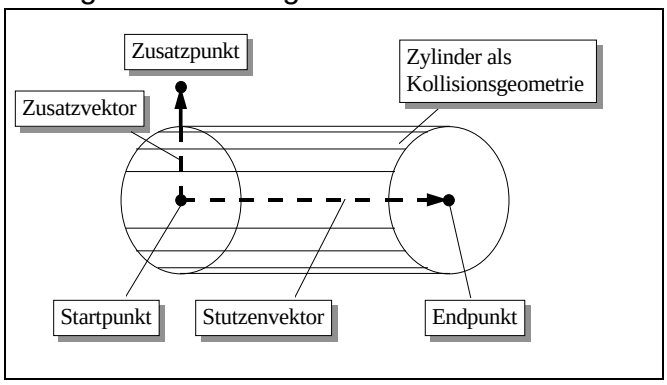


Abbildung 31: Stutzenaufbau geometrisch

Ein Stutzen kann zusätzlich zu den beschriebenen Komponenten einen dritten Punkt enthalten. Der Vektor, der durch den Startpunkt und diesen Zusatzpunkt definiert ist, bildet einen Winkel von 90 Grad zum Stutzenvektor. In Abbildung 31 ist der Aufbau eines Stutzens dargestellt, Abbildung 32 zeigt die Definition im FHS-Format und Abbildung 33 die resultierende Darstellung des Stutzens in der VR-Objekthierarchie.

```
MODEL Stutzenname

ASSEMBLY assStutzenname
{
    GEOMETRY Stutzenname-MLine
    USEMAT defaultmat
    {
        LINE
        {
            P=(0.0,0.0,0.0)
            P=(1.2,2.7,2.04)
        }
    }
    GEOMETRY Stutzenname-MLine
    USEMAT defaultmat
    {
        POLYGON
        {
            P=(0.0,0.2,0.0)
            P=...
            ...
        }
        POLYGON
        {
            ...
        }
    }
}
```

Abbildung 32: Stutzendefinition im FHS-Format

4.2.2. Integration der Stutzengeometrie in das VR-Geometriemodell

Im vorangegangenen Abschnitt wurde der allgemeine geometrische Aufbau der Stutzen dargestellt. Die Geometrie eines Stutzens ist demnach vollständig unter einem Teilbaum der VR-Objekthierarchie abgelegt. Dieser Teilbaum kann sich an beliebiger Stelle innerhalb der VR-Objekthierarchie befinden.

Unter Zuhilfenahme der Stutzen werden im Geometriemodell der virtuellen Objekte Verbindungsstellen gekennzeichnet. Damit werden diese Informationen geometrieneutral im Datenmodell gehalten. Die Verbindungsstellen sind den virtuellen Objekten zugeordnet. Diese Zuordnung spiegelt sich in der Objekthierarchie des VR-Systems wider. Ein Stutzen, der zu einem Objekt A gehört ist in der VR-

Objekthierarchie auch unterhalb dieses Objektes A abgelegt. Durch diese Festlegung wird erreicht, daß sich jede Matrixoperation auf Objekt A auch auf das zugehörige Stutzenobjekt auswirkt. Damit sind alle Positionierungsoperationen berücksichtigt, die in dem beschriebenen VR-System durchgeführt werden (siehe auch Abschnitt 3.2.3.).

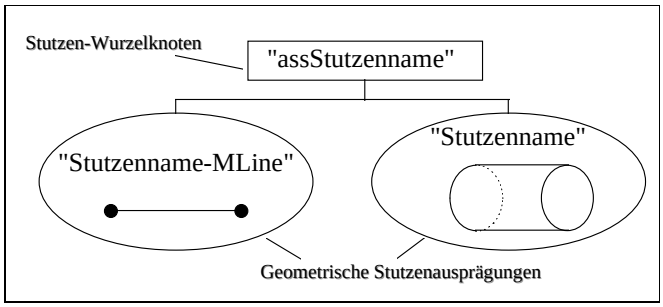


Abbildung 33: Stutzendefinition in der VR-Objekthierarchie

Die Stutzen definieren bestimmte Verbindungsstellen sowohl an den CAD-Entities als auch an den übrigen VR-Objekten, die ursprünglich aus den CAD-Daten erzeugt wurden. Damit muß die Erweiterung der Geometrie um die Stutzen für diese beiden Objektklassen unterschieden werden.

Für die *Erweiterung der CAD-Entities* wurden die Funktionen der Objekterzeugung um die Stutzenerzeugung erweitert. Bei den Linien-Entities, die für die

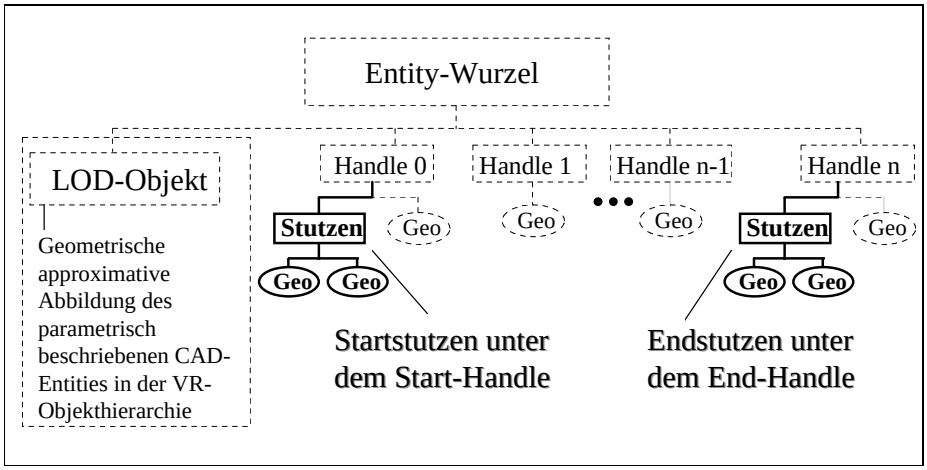


Abbildung 34: Um Start- und Endstutzen erweiterte Objekthierarchie eines Linienentities

Leitungsverlegung angewendet werden (siehe auch Abschnitt 4.1.2.), wurde der Aufbau der Entities innerhalb der VR-Objekthierarchie (Abbildung 24) um die Stutzen erweitert (Abbildung 34). Die Stutzen markieren die Punkte am Linienentity, an denen eine Verbindung zu einem weiteren Entity oder zu einem Nicht-Entity existieren kann. Der Stutzenvektor ist dabei so positioniert, daß der Stutzenstartpunkt mit dem Endpunkt der Entitylinie übereinstimmt. Der Anstieg des Stutzenvektors entspricht dabei dem Anstieg der durch das Entity definierten Kurve in diesem Punkt.

Bei den verschiedenen Linienentities (Strecke, Kreisbogen, Parametrische Splinekurve) werden die Endpunkte der Linien durch Handle repräsentiert.

Daher wurden die Stutzen in der VR-Objekthierarchie unterhalb dieser Handle positioniert, die auch durch einen eindeutigen Namen („StartHandle“ und „EndHandle“) referenzierbar sind. Damit können die Stutzen für die CAD-Entities in der VR-Umgebung automatisch erzeugt werden.

Die *Stutzenerweiterung der VR-Objekte* kann nicht automatisch wie bei den CAD-Entities erfolgen. Informationen über mögliche Verbindungsstellen werden konstruktionsseitig nicht explizit angegeben und werden auch über die genutzten CAD-Datenschnittstellen nicht übertragen. Aus der reinen flächenbeschriebenen Geometrie der Objekte können solche Informationen auch nicht automatisch extrahiert werden. Daraus ergibt sich die Notwendigkeit, die Informationen über Verbindungsstutzen im Rahmen des Datenvorbereitungsprozesses interaktiv zu erzeugen und in das VR-Datenmodell zu integrieren. Für die Erzeugung eines Stutzens sind lediglich die Informationen über den Namen des Stutzenobjektes und die Koordinaten des Start- und des Endpunktes notwendig. Aus diesen Informationen kann die hierarchische Struktur mit den entsprechenden Geometrien entsprechend Abbildung 33 generiert werden. Zusätzlich muß der erzeugte Stutzen einem Objekt zugeordnet werden.

Die Zuordnung des Stutzens zu einem VR-Objekt erfolgt innerhalb der VR-Objekthierarchie. Daraus ergibt sich eine Zusatzanforderung an die Strukturierung der virtuellen Objekte. Jedes Objekt, dem ein oder mehrere Stutzen zugeordnet sind, muß einen Wurzelknoten als Startknoten in der Objekthierarchie besitzen, dem mehrere Kinderknoten zugeordnet werden können. Nur dann kann die Zuordnung Objekt ⇔ Stutzen erfolgen. Im speziellen Fall des Virtual Design II bedeutet das, daß jedes Objekt als Startknoten einen ASSEMBLY-Knoten besitzt und daß auch dieser ASSEMBLY-Knoten im Fall einer Referenzierung dieses Objektes selektiert wird.

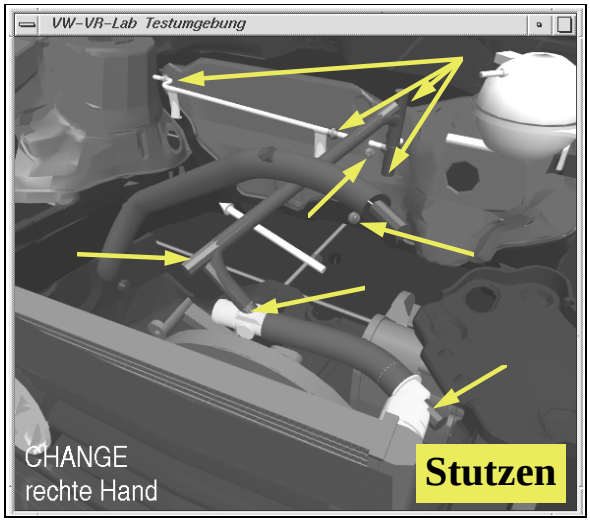


Abbildung 35: Virtuelle Umgebung mit Stutzen

4.2.3. Stutzenbasierte Verbindungssimulation

4.2.3.1. Definition und Diskussion der Stutzenverbindung

Über die Stutzengeometrie wird eine Verbindung als räumliche Relation zwischen zwei Stutzen wie folgt definiert:

Definition:

Zwei Stutzen sind miteinander verbunden, wenn sie in ihren Startpunkten übereinstimmen und die Stutzenvektoren einen Winkel von 180 Grad einschließen, also entgegengesetzt orientiert sind.

Äquivalente Verbindungsdefinitionen sind auch in der Literatur zu finden[84].

Diese räumliche Relation zwischen zwei Stutzen kann durch das System einfach getestet werden. Der Test läuft auf einen Vergleich von reellen Zahlen hinaus, der vom System sehr schnell durchgeführt werden kann.

Aus der exakten Verbindungsdefinition ergibt sich das Problem der Verbindungserzeugung. Ein Teilziel des Systems ist es, Verbindungen interaktiv erzeugen zu können. Dabei sollen sich die Interaktionsmetaphern an der Realität orientieren. Eine exakte Verbindung, bei der die Übereinstimmung der Koordinaten mit einer Genauigkeit überprüft wird, die nur durch die Rechnergenauigkeit begrenzt ist, kann ohne eine spezielle Systemunterstützung nicht erzeugt werden.

Eine spezielle Unterstützung von Verbindungsrelationen durch das System führt zu einer Erweiterung des Datenmodells, durch das die virtuellen Objekte und deren Relationen zueinander beschrieben sind. Über diese Datenmodellerweiterung kann das System objektspezifische räumliche Relationen zwischen verschiedenen Objekten abtesten und für bestimmte Operationen nutzen. Mit der Einführung der Stutzen in die virtuelle Umgebung wird versucht, eine exakte physikalisch basierte Modellierung der Objekte zu vermeiden und das aus diesen physikalisch beschreibbaren Eigenschaften resultierende Objektverhalten über nichtphysikalische Ersatzmodelle abzubilden. Diese Vorgehensweise ist mit der in [ZIEG98] beschriebenen Anwendung von wahrnehmungsbasierten haptischen Rückkopplungen vergleichbar, bei der durch den Vergleich von Realität und nachgebildeter Realität und deren bestmöglicher Annäherung aneinander auf eine physikalische Realitätssimulation verzichtet wird. Als Grund für den Einsatz eines solchen Ersatzmodells für die Simulation eines Objektverhaltens werden in [ZIEG98] folgende zwei Punkte angegeben:

1. Die exakte Beschreibung des physikalischen Modells ist nicht möglich
2. Das Ergebnis einer physikalisch korrekten Simulation vermittelt einen unrealistischen Eindruck

[84] [WITKIN87]

Als Begründung des Einsatzes eines stutzenbasierten Verbindungsmodells müssen folgende Punkte genannt werden:

1. Die exakte physikalische Beschreibung des Objektverhaltens beim Aufbau einer Verbindung oder bei der Trennung einer solchen setzt ein erweitertes Datenmodell voraus, dessen Erzeugung den Daten- und Anwendungsvorbereitungsaufwand unverhältnismäßig stark erhöht.
2. Es ist zu berücksichtigen, daß der Berechnungsaufwand für eine physikalisch korrekte Simulation als sehr hoch einzuschätzen ist, dabei aber den Echtzeitbedingungen einer VR-Umgebung genügen muß.
3. Auch der oben aufgeführte Punkt, daß eine vermeintlich physikalisch korrekte Simulation einen unrealistischen Eindruck vermitteln könnte, ist hierbei nicht auszuschließen.
4. Die Unmöglichkeit der Beschreibung des physikalischen Modells ist hierbei ebenfalls zu erwähnen. In der Realität nicht durchführbare Operationen, wie das Erzeugen einer Verbindung zwischen einer Splinekurve und einem Linienelement, erzwingt den Einsatz eines Ersatzmodells.

4.2.3.2. Verbindungszustände

Bereits im vorangegangenen Abschnitt wurde definiert, wann zwei Stutzen miteinander verbunden sind. Daraus ergeben sich die beiden Verbindungszustände „verbunden“ und „nicht verbunden“, zwischen denen im System grundsätzlich unterschieden wird. Eine räumliche Relation zwischen zwei Stutzen kann immer zwischen diesen beiden Zuständen wechseln. Eine Verbindung kann immer nur mit Unterstützung des Systems erzeugt werden. Dabei sind zwei verschiedene Möglichkeiten zu unterscheiden.

1. Möglichkeit des Verbindungsaufbaus:

Bei der ersten Möglichkeit erfolgt die Erzeugung einer Verbindung zwischen zwei Stutzen sofort. Für ein Objekt wird eine neue Transformation berechnet, in deren Ergebnis eine exakte Verbindung zwischen zwei Stutzen entsteht. Für diese Funktionalität sind die beiden Stutzen und das zu bewegende Objekt als Eingangsdaten notwendig. Die Ausführung dieser Operation erfolgt im entsprechenden VR-System von einem Durchlauf der Simulationsschleife auf den nächsten. Das transformierte Objekt „springt“ also von einem Zustand in den anderen. Es besteht keine Möglichkeit, direkt und interaktiv auf die zu erzeugende Verbindung Einfluß zu nehmen.

Es hat sich gezeigt, daß diese Umsetzung des Verbindungszustandsübergangs realitätsfern ist und zu Irritationen des Anwenders führen kann. Diese Problematik ergibt sich aus der Definition der Stutzenverbindung. Die Übereinstimmung der Startpunkte der Stutzen und die entgegengesetzte Orientierung der Stutzenvektoren legt nicht eindeutig die Orientierung der Stutzen entlang ihrer Längsachse fest. Diese Verbindungsdefinition ist unterbestimmt.

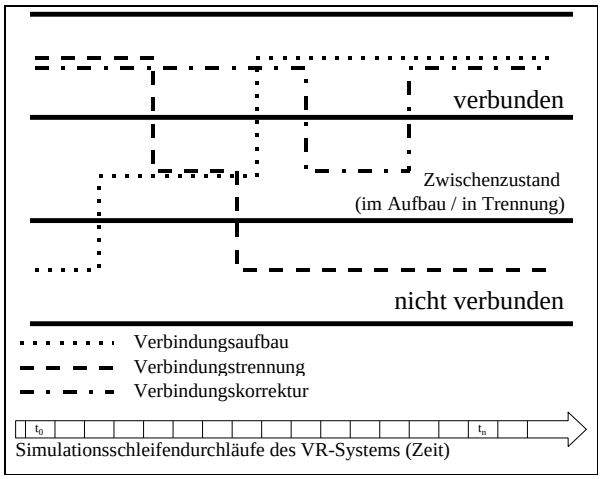


Abbildung 36: Mögliche Zustandsübergänge bei Stutzenverbindungen über einen Zwischenzustand

Die Stutzen können demzufolge zueinander beliebig verdreht sein. Das Ergebnis einer automatisch vom System erzeugten Verbindung ist damit nicht eindeutig definiert. Daher ist eine interaktive Beeinflussung der Verbindungserzeugung notwendig.

Das führt zur

2. Möglichkeit des Verbindungsaufbaus:

Um eine interaktive Beeinflussung der Verbindungserzeugung zu ermöglichen, wurde ein dritter Verbindungszustand eingeführt. Dieser Zustand beschreibt eine Verbindung, die sich im Aufbau oder in der Trennung befindet. Es handelt sich also um einen *Zwischenzustand*. Dieser Zwischenzustand kann über mehrere Durchläufe der Simulationsschleife des VR-Systems erhalten bleiben. Ausschließlich über diesen Zustand kann vom verbundenen in den nicht verbundenen und vom nicht verbundenen in den verbundenen Zustand übergegangen werden (Abbildung 36).

4.2.3.3. Stutzenbasierte Bewegungsbeschränkungen

Über die Stutzen können verschiedene Bewegungsbeschränkungen definiert werden. Diese Bewegungsbeschränkungen bilden die Grundlage für die interaktiven Verbindungsabläufe. Dabei ist ein Stutzen der aktive (Stutzen 1), auf den die Korrektur durch das System angewendet wird. Der zweite Stutzen (Stutzen 2) ist der Referenzstutzen, der zum Koordinatenvergleich herangezogen wird. Im Folgenden werden die verschiedenen möglichen stutzenbasierten Constraints näher beschrieben.

No-Constraint (MScNoConstraint[85])

Das No-Constraint bezeichnet den Zustand, bei dem keine Bewegungsbeschränkung auf den aktiven Stutzen wirkt. Der aktive Stutzen kann frei im Raum bewegt werden.

Parallelitäts-Constraint (MScGerade)

Beim Parallelitätsconstraint wird der aktive Stutzen so beeinflußt, daß die durch den Stutzenvektor definierte Gerade immer parallel zur Referenzgeraden verläuft, die durch den Stutzenvektor des Referenzstutzens definiert ist (Abbildung 37). Die Systembeeinflussung des aktiven Stutzens erfolgt durch eine Rotation um seinen Startpunkt.

Startpunkt auf der Referenzgeraden (MScFixedTranslation)

Abbildung 38 verdeutlicht die Umsetzung dieser Bewegungsbeschränkung. Der Startpunkt des aktiven Stutzens befindet sich immer auf der durch den Referenzstutzen definierten Geraden. Diese Korrektur erfolgt durch das Fällen des Lotes vom Startpunkt des aktiven Stutzens auf die Gerade und die Translation des aktiven Stutzens um diesen Vektor.

Startpunktübereinstimmung (MScFixedEndpoint)

Die in Abbildung 39 dargestellte Bewegungsbeschränkung sorgt dafür, daß die Startpunkte der beiden Stutzen übereinstimmen. Durch eine Translation des aktiven Stutzens wird dieses Constraint umgesetzt.

Die folgenden beiden Constraints sind zwar sinnvoll, werden aber für die Umsetzung der weiter unten beschriebenen Stutzenverbindungen nicht benötigt:

Startpunkt auf der Stutzenvektorebene (MScFixPlane)

Der Startpunkt des aktiven Stutzens befindet sich immer auf der Ebene, die durch den Stutzenvektor des Referenzstutzens definiert ist. Dieses Constraint wird durch eine Translation realisiert, bei der der Startpunkt des aktiven Stutzens auf die entsprechende Ebene projiziert wird (Lotpunktberechnung).

Fixe Rotation (MScNoRotation)

Eine Einschränkung der rotatorischen Freiheitsgrade um die Stutzenlängsachse kann nur mit Hilfe eines Zusatzvektors erfolgen. Dieser kann durch einen Zusatzpunkt, wie er in Abbildung 31 dargestellt ist, in das Geometriemodell integriert werden. Für die im System verwendeten Stutzenverbindungsabläufe ist diese Bewegungsbeschränkung allerdings nicht erforderlich. Der Vollständigkeit halber soll dieses Constraint an dieser Stelle erwähnt werden. Die Realisierung erfolgt über die Winkelbestimmung zwischen den Zusatzvektoren des aktiven und des Referenzstutzens.

[85] Die Kurzbezeichnungen wurden direkt aus dem C-Quellcode des beschriebenen Systems übernommen (MSconstraint.h).

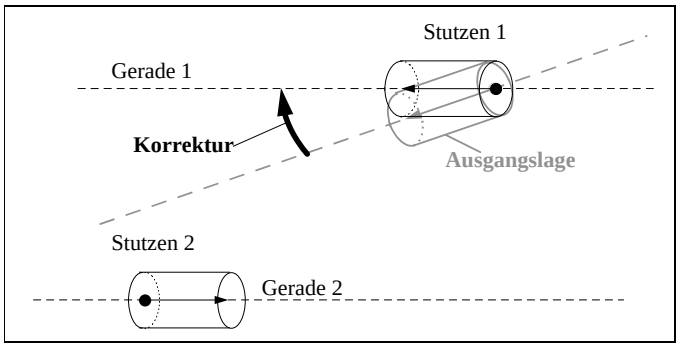


Abbildung 37: MScGerade

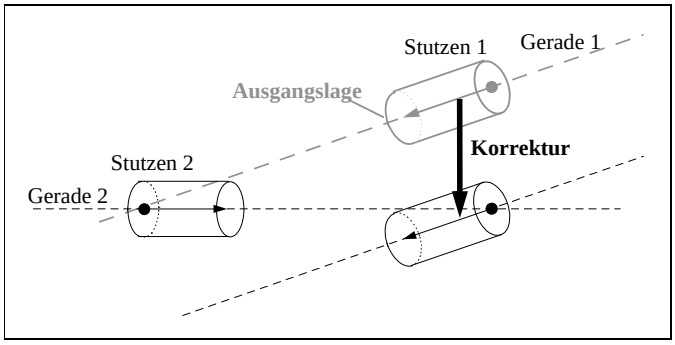


Abbildung 38: MScFixedTranslation

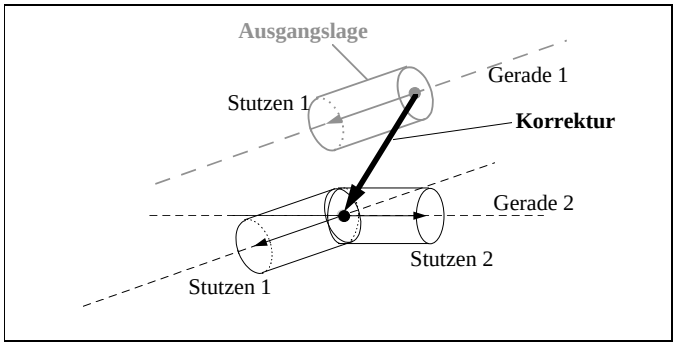


Abbildung 39: MScFixedEndpoint

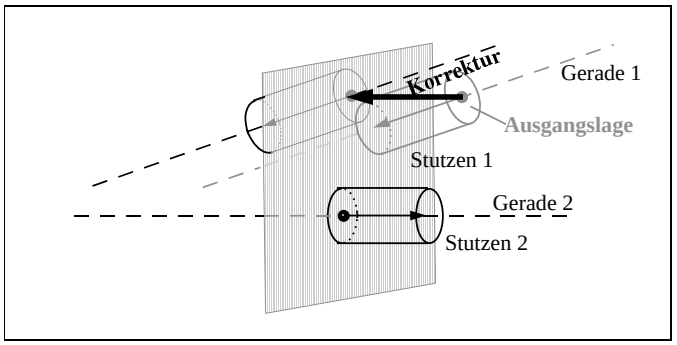


Abbildung 40: MScFixPlane

Die beschriebenen Constraints wurden so implementiert, daß sie frei miteinander kombiniert werden können. In Tabelle 2 sind einige für die Anwendung sinnvolle Kombinationsmöglichkeiten der Bewegungsconstraints aufgelistet. Diese Aufstellung verdeutlicht, daß damit einfache Eigenschaften wie die Rotation eines Schlauches auf seinem Befestigungsstutzen erzwungen werden können. Die geklammerten Kreuze in der Zeile MScFixedTranslation sollen veranschaulichen, daß diese Bewegungsbeschränkung redundant ist, wenn MScFixedEndpoint aktiv ist. MScFixedEndpoint stellt einen Sonderfall des Constraints MScFixedTranslation dar.

	Verschiebung auf einer Geraden mit erlaubter Rotation	Feste Verbindung mit ausschließlicher Rotation	Verschiebung auf einer ebenen Fläche ohne „Kippeln“	Feste Verbindung ohne Bewegungsmöglichkeit
MScGerade	X	X	X	X
MScFixed-Translation	X	(X)		(X)
MScFixed-Endpoint		X		X
MScFixPlane			X	
MScNoRotation				X

Tabelle 2: Kombinationsbeispiele unterschiedlicher Bewegungsconstraints

Der Vorteil dieser stutzenbasierten Bewegungsconstraints liegt in ihrer einfachen Realisierung. Damit ist lediglich ein geringer Berechnungsaufwand erforderlich. Das System kann mit Hilfe einfachster Vektoroperationen den Zustand zwischen zwei Stutzen analysieren und gegebenenfalls korrigieren. Die Korrektur dieser räumlichen Relation erfolgt direkt innerhalb einer Simulationsschleife. Damit wird dem Anwender wie auch dem restlichen VR-System mit seinen verschiedensten Simulationskomponenten lediglich der korrigierte Systemzustand präsentiert[86].

Ein weiterer Vorteil ist in den immer gleichen Datenstrukturen begründet, die für die Analyse, Berechnung und Korrektur genutzt werden können. Da die Möglichkeit besteht, die Stutzen jedem beliebigen virtuellen Objekt zuzuordnen, können die Bewegungsbeschränkungen auch auf jedes beliebige Objekt angewendet werden. Eine direkte Abhängigkeit von der geometrischen Ausprägung der virtuellen Objekte besteht damit nicht.

Durch die hierarchische Zuordnung der Stutzen zu den geometrischen Objekten in der VR-Objekthierarchie wirken sich alle Matrixoperationen auf diese Objekte auch auf die Stutzen aus. Durch die Abfrage der Koordinaten der Stutzen bei jedem Simulationsdurchlauf gehen diese Operationen auch direkt in die Berechnung der Bewegungsconstraints mit ein. Damit eignen sich die beschriebenen Constraints auch für Verbindungsoperationen zwischen beweg-

[86] Es sei an dieser Stelle darauf hingewiesen, daß dieser Effekt natürlich nur dann wirksam ist, wenn eine Constraintkorrektur an der zeitlich richtigen Stelle innerhalb der Simulationsschleife implementiert werden kann. Die zeitliche Abfolge der einzelnen Routinen wird noch in Kapitel 4.2.3.4. näher beleuchtet!

ten Objekten. An dieser Stelle sei vermerkt, daß diese Eigenschaft eine Voraussetzung dafür darstellt, mit beiden Händen in einer virtuellen Umgebung zu interagieren.

4.2.3.4. Constraintbasierte Verbindungsabläufe

Die im vorherigen Abschnitt beschriebenen Bewegungsbeschränkungen bilden die Grundlage für die Umsetzung verschiedener *Verbindungsabläufe*. Dabei handelt es sich um die Abläufe, bei denen sich zwei Stutzen in einem Zwischenzustand befinden wie er im Abschnitt 4.2.3.2. diskutiert wurde. Alle Verbindungsabläufe sind durch ein einheitliches Ablaufschema gekennzeichnet:

Voraussetzung:

Zu einem festen Zeitpunkt ist ausschließlich ein Stutzen aktiv und wird durch die beschriebenen Bewegungsconstraints in seiner Bewegung beeinflußt.

Ablauf:

1. Durch ein externes Ereignis ausgelöst, geht das Verbindungsattribut, durch welches eine Relation zwischen zwei Stutzen gekennzeichnet ist, vom unverbundenen Zustand in den Zwischenzustand über.
2. Durch das Setzen geeigneter Bewegungsconstraints für den aktiven Stutzen wird der Aufbau einer Verbindung zwischen den zwei Stutzen durch das System unterstützt.
3. Wiederum durch ein externes Ereignis ausgelöst, wird der Zwischenzustand verlassen und die beiden Stutzen können entweder miteinander verbunden sein oder nicht.

Das externe Ereignis für den Übergang vom unverbundenen Zustand in den Zwischenzustand kann durch eine beliebige Callbackroutine definiert werden. Dieser Callbackroutine müssen die am Verbindungsaufbau beteiligten Stutzen übergeben werden.

Das externe Ereignis, das für das *Verlassen des Zwischenzustandes* sorgt (ebenfalls eine Callbackroutine), schaltet den im folgenden Absatz beschriebenen Simulationskern ab und bringt so das System in einen von Verbindungsabläufen freien, neutralen Zustand.

Das *Setzen der Bewegungsbeschränkungen* (2. Ablaufpunkt) erfolgt durch einen Simulationskern, dem die Stutzengeometrien bekannt sind, die am Verbindungsablauf beteiligt sind. Ferner können verschiedenen Stutzen unterschiedliche Verbindungsabläufe wie auch unterschiedliche Verbindungsablaufparameter zugeordnet sein. Als Entscheidungshilfe für das Setzen oder Löschen verschiedener Bewegungsconstraints dienen die Geometriedaten der am Verbindungsablauf beteiligten Stutzen. Dabei werden der Abstand zwischen den Startpunkten der Stutzen, die jeweiligen Winkel zwischen den Stutzenvektoren und zwischen den Zusatzvektoren berücksichtigt.

In dem beschriebenen System wurden zu Testzwecken verschiedene Verbindungsabläufe implementiert, die sich jeweils durch eine unterschiedliche Pa-

rametrisierung voneinander unterscheiden lassen können. Mit den unterschiedlichen Verbindungsabläufen können verschiedene in der Realität existierende Verbindungsarten unterschieden werden. Die Parametrisierung dieser Abläufe erlaubt es, unterschiedliche geometrische Spezifika einzelner virtueller Objekte zu berücksichtigen. Die Umsetzung eines solchen Ablaufes soll hier anhand einer Steckverbindung beschrieben werden.

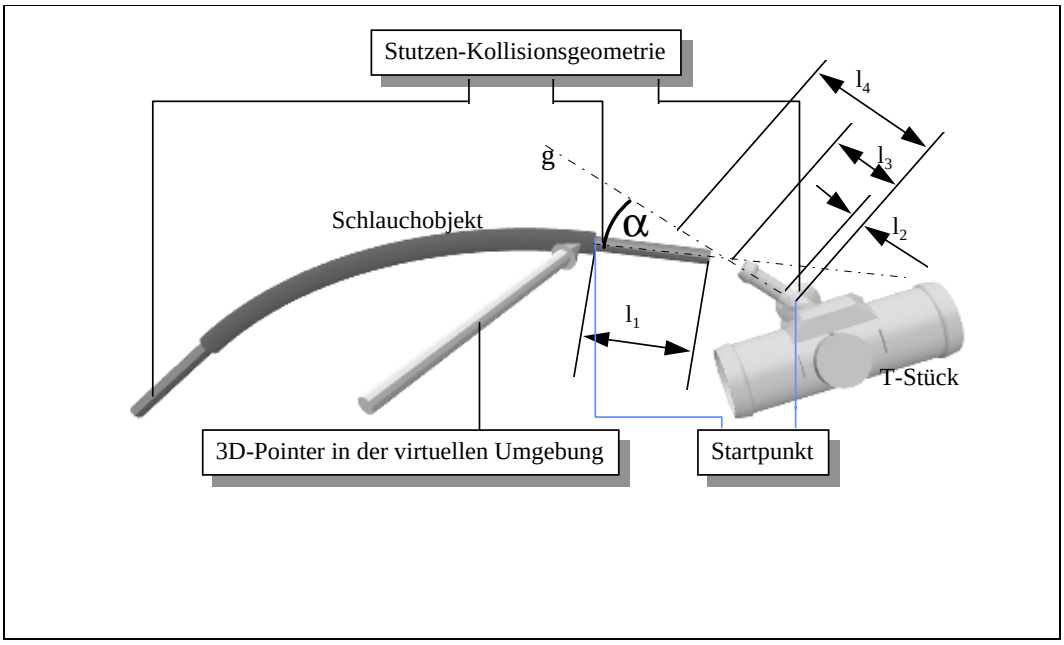


Abbildung 41: Stutzenbasierte Steckverbindung zwischen Schlauchobjekt und T-Stück

Für die *Steckverbindung* sind vier verschiedene Parameterwerte von Interesse. Diese Werte sind in Abbildung 41 mit l_2, l_3, l_4 und α bezeichnet. Der Wert α beschreibt einen Winkel, der mit dem Winkel verglichen wird, der von den beiden Stutzenvektoren eingeschlossen wird[87]. L_2 ... l_4 definieren unterschiedliche Längen, die mit dem Abstand zwischen den Startpunkten der Stutzen verglichen werden.

Der Zwischenzustand „Steckverbindung“ unterteilt sich in 3 Teilzustände (Abbildung 42):

Tritt ein externes Ereignis ein, das beide Stutzen in den ersten *Zwischenzustand* bringt, wird bei der Steckverbindung sofort das Constraint MScFixed-Translation gesetzt. Damit kann sich das Schlauchobjekt nur noch auf der Geraden *g* bewegen. Das heißt, daß sich der Startpunkt des Stutzens immer auf der durch den Referenzstutzen definierten Geraden befindet. Damit ist eine Translation ausschließlich mit dem Startpunkt auf *g* und eine Rotation um den Startpunkt des Stutzens des Schlauchobjektes möglich.

Dieser erste Zwischenzustand kann in den *zweiten Zwischenzustand* übergehen, bei dem das Constraint MScGerade dazugeschaltet wird. Dafür muß der

[87] Genau genommen handelt es sich um den Wert 180°-α, der von den Vektoren eingeschlossen wird. Für die systeminterne Berechnung wird einer der beteiligten Vektoren invertiert.

von den beiden Stutzen eingeschlossene Winkel kleiner als der Vergleichswinkel α sein. Damit ist für das Schlauchobjekt nur noch eine Translation entlang der Geraden g möglich und der Stutzen des Schlauchobjektes ist immer parallel zum Partnerstutzen angeordnet. Das hat zur Folge, daß sich das Schlauchobjekt mit seinem Stutzen ausschließlich auf der Geraden g befindet und nur noch um eine Rotationsachse rotiert werden kann. Diese Achse stimmt mit der Geraden g überein.

Dieser zweite Zwischenzustand kann in den *dritten Zwischenzustand* übergehen, bei dem zusätzlich zu den bereits wirksamen Bewegungsbeschränkungen das Constraint MScFixedEndpoint aktiv wird. Dieser Übergang wird ausgelöst, wenn der Abstand zwischen den Stutzenstartpunkten $<l_2$ wird. Damit ist auch die Translation entlang der Geraden g nicht mehr möglich. Das Schlauchobjekt befindet sich in der verbundenen Position und kann lediglich um die Stutzenachse rotiert werden.

Alle drei Zwischenzustände können durch ein externes Ereignis ausgeschaltet werden. Damit sind die wirkenden Constraints gelöst und das Objekt ist wieder frei in der virtuellen Umgebung positionierbar. Tritt dieser Fall nicht ein, kann jeweils der aktive Zwischenzustand ausschließlich in den nächst höheren oder den nächst niederen Zustand wechseln. Dafür sind bei der beschriebenen Steckverbindung die Schwellwerte l_3 und l_4 notwendig. Wird im ersten Zwischenzustand ein Abstand zwischen den Startpunkten der Stutzen ermittelt, der einen Wert größer als l_4 besitzt, dann wird der Zwischenzustand verlassen und das Schlauchobjekt ist frei in der virtuellen Umgebung positionierbar. Dieser Vorgang entspricht dem externen Ereignis, welches den Verbindungsvorgang jederzeit beenden kann.

Vom zweiten Zwischenzustand aus (MScFixedTranslation, MScGerade) kann der Verbindunsablauf in den ersten Zwischenzustand übergehen, wenn ein Abstand zwischen den Stutzenstartpunkten besteht, der größer als l_3 ist oder wenn der von den Stutzen eingeschlossene Winkel größer α ist.

Der dritte Zwischenzustand geht in den zweiten Zwischenzustand über, wenn der Abstand zwischen den beiden Startpunkten der Stutzen einen Wert größer l_2 annimmt.

Wie am Beispiel „Steckverbindung“ beschrieben, können verschiedenste Verbindungsablaufarten mit unterschiedlichen Parametrisierungen implementiert werden. Für den Test dieser unterschiedlichen Abläufe wurde im Rahmen dieser Arbeit eine Testumgebung implementiert, mit deren Hilfe auf die verschiedenen Parameter zugegriffen werden kann. Damit ist es möglich, unterschiedliche Abläufe zu testen, bevor sie innerhalb einer Anwendung zum Einsatz kommen.

Die implementierten Verbindungsabläufe erheben nicht den Anspruch, exakt mit Verbindungsabläufen übereinzustimmen, wie sie bei einer realen Montage am physikalischen Modell existieren. Diesen Anspruch soll und kann das zugrunde liegende Modell nicht erfüllen. Die Verbindungsabläufe ermöglichen aber eine Beeinflussung der Interaktionen in der Art und Weise, daß das an-

gestrebte Interaktionsziel relativ effektiv und dabei so realitätsnah wie möglich erreicht wird.

Es sei an dieser Stelle angemerkt, daß es sich bei der implementierten Anwendung um ein Modellierungswerkzeug handelt. Daher wurde bei der Umsetzung großer Wert auf eine einfache und effektive Handhabbarkeit gelegt, die Übereinstimmung des physikalischen Objektverhaltens spielte dabei eine eher untergeordnete Rolle. Bei der Arbeit mit den CAD-Entities hat es sich sogar gezeigt, daß eine zu aufwendige Verbindungsablaufsimulation die Effektivität bei der Bedienung des Systems stören kann. Daher wurde für die Entity-Verbindungen ein sehr einfaches Verbindungsablaufmodell implementiert. Dieser Entity-Verbindungsablauf springt sofort beim Einschalten der Verbindungsablaufsimulation in das bei der Steckverbindung als „3. Zwischenzustand“ bezeichnete Verhalten. Damit ist ein genaues Positionieren und Herantasten an den Verbindungsendpunkt nicht notwendig. Es handelt sich eher um eine Art „Schnappfunktion“ bei der die beiden zu verbindenden Stutzen aneinander-„schnappen“. Unabhängig davon besteht aber noch die Möglichkeit, auf die Verdrehung beider Objekte zueinander im „3. Zwischenzustand“ Einfluß zu nehmen.

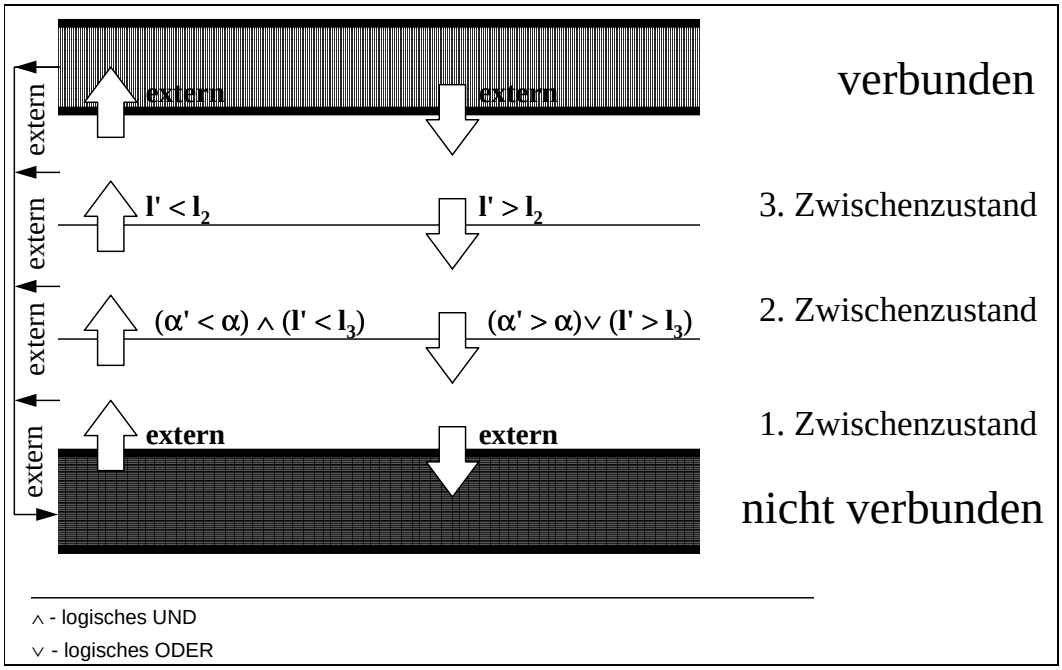


Abbildung 42: Constraintbasierter Verbindungsablauf einer Steckverbindung mit drei Zwischenzuständen

Besonders zu beachten bei der Implementierung dieser Verbindungsabläufe ist die Integration der einzelnen Komponenten im VR-System. Zu diesen Komponenten gehören sowohl die Module zum Abfragen der Eingabegeräte, zum Aktualisieren der Objekte in der Objekthierarchie, die Kollisionserkennung als Grundlage für verschiedenste Simulationen als auch der eigentliche Renderingprozeß, der den Zustand der Objekte im System dem Anwender präsentiert. Die verschiedenen Module müssen in einer festen Reihenfolge abgearbeitet werden. Diese Reihenfolge ist durch die implementierte Simula-

tionsloop definiert. Eine vereinfachte Beschreibung dieser Schleife ist in Abbildung 43 skizziert.

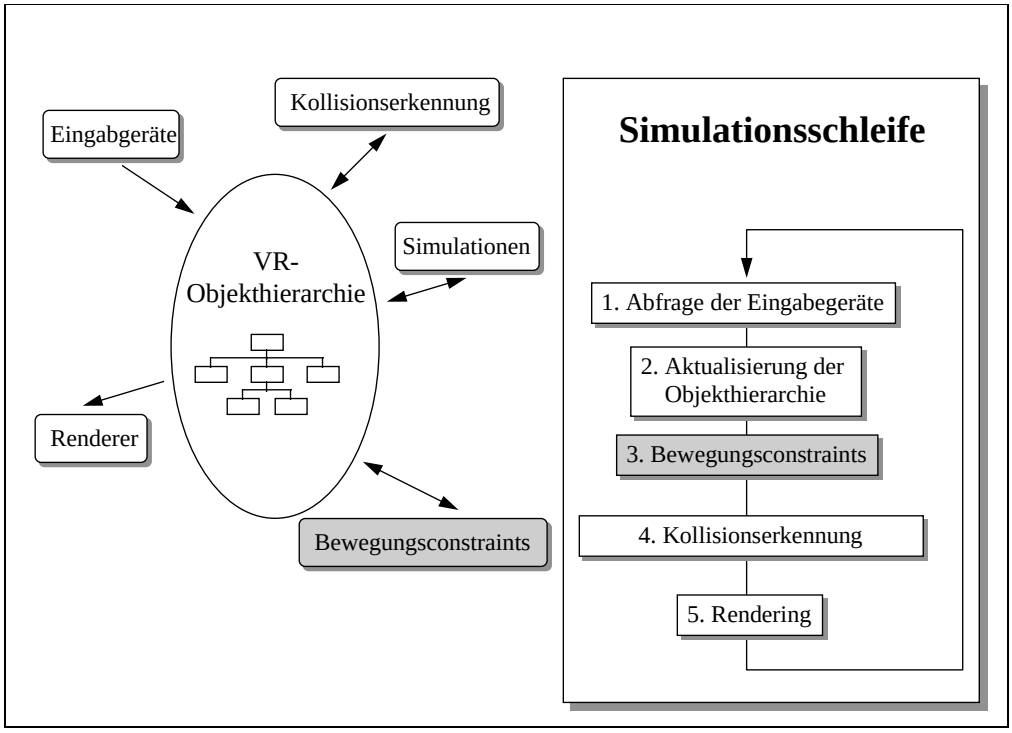


Abbildung 43: Abarbeitungsreihenfolge der verschiedenen Module im VR-System

Auf der Grundlage der abgefragten Eingabegeräte werden die Objekte in der VR-Objekthierarchie aktualisiert. Über die Kollisionserkennung werden bestimmte Ereignisse ausgelöst. Als Beispiel soll hier die Überprüfung auf gegenseitige Durchdringungen verschiedener Objekte dienen, von denen mindestens ein Objekt direkt durch den Anwender bewegt wird. Besteht eine Kollision, so wird das dem Anwender akustisch und/oder visuell über den Renderingprozeß mitgeteilt. Handelt es sich bei dem bewegten Objekt um eines, welches durch ein Constraint in seiner Bewegung eingeschränkt ist, dann besteht die Möglichkeit, daß eine Kollision eintritt bevor das Constraint wirksam ist. Nach der Korrektur der Bewegung durch das System ist es möglich, daß diese Kollision nicht mehr vorhanden ist. Der umgekehrte Fall ist ebenfalls denkbar. Durch das Abarbeiten einer Bewegungsbeschränkung kann es vorkommen, daß eine vorher nicht existierende Kollision durch das Constraint ausgelöst wird. Daher ist es notwendig, daß die Bewegungsconstraints vor der Kollisionserkennung und vor dem Rendering der virtuellen Szene abgearbeitet werden. Nur dann ist gewährleistet, daß bei jedem Simulationsdurchlauf dem Anwender ein Zustand der Gesamtszene präsentiert wird, der in sich konsistent ist und dem Simulationsergebnis entspricht.

Die dargestellte Problematik spielt eine wesentliche Rolle bei der Implementierung eines solchen Systems, bei der ein bestehendes VR-System über ein API erweitert werden soll. In einem solchen Fall ist die Implementierung nur dann möglich, wenn Einfluß auf die Abarbeitungsreihenfolge innerhalb der Simula-

tionsschleife genommen werden kann bzw. wenn das System im Quellcode zur Verfügung steht.

4.2.3.5. Eigenschaften der stutzenbasierten Verbindungsabläufe

Durch die Implementierung der stutzenbasierten Verbindungsabläufe ist es möglich, in der Realität existierende und somit bekannte Verbindungsabläufe zu simulieren, wie es oben anhand der Steckverbindung näher erläutert wurde. Ein weiterer Verbindungsablauf, der aber für das in der beschriebenen Anwendung angestrebte Anwendungsziel keine Bedeutung hat, ist die Schraubverbindung. Auch diese ist in der beschriebenen Systemumgebung implementiert worden. Sie basiert ebenfalls ausschließlich auf der Definition der Stutzengeometrie inklusive des oben beschriebenen Zusatzvektors und der darauf angewendeten Bewegungsconstraints. Über den Parameter „Schraubenlänge“ und einer Beschreibung des Schraubgewindes in Form der Gewindesteigung kann dieser Verbindungsablauf beliebig parametrisiert werden. Damit konnte gezeigt werden, daß mit Hilfe der Bewegungsconstraints eine echtzeitfähige Interaktionsbeeinflussung möglich ist, die sehr stark an die Realität angelehnt ist.

Anhand der oben beschriebenen „Schnappverbindung“ konnte weiterhin gezeigt werden, daß es diese Art der Ablaufsimulation erlaubt Bewegungsabläufe zu unterstützen, die in der Realität nicht existieren. So können Verbindungen an Objekten manipuliert werden, bei denen es sich um künstliche, in der Realität nicht existierende Objekte handelt, wie es die CAD-Entities darstellen.

Durch die ausschließliche Verwendung geometrischer Primitive als Basis für die Verbindungsablaufsimulation kann auf materialabhängige physikalische Parameter und Konstanten verzichtet werden. Es ist jedoch denkbar, für spezielle Abläufe solche Werte mit zu berücksichtigen. Ist beispielsweise die Kraft bekannt, die benötigt wird, um einen Schlauch von einem Aufsteckstutzen zu ziehen, so könnte diese Kraft als Schwellwert in einer Kraftrückkopplungssimulation genutzt werden. Die durch die Bewegungsconstraints ermittelten Korrekturvektoren und Rotationsmatrizen könnten ebenfalls in einem Regelkreis in eine Kraft umgewandelt werden, die dann direkt auf die Bewegung der Hand des Anwenders wirkt. So wäre auch eine stutzenbasierte Bewegungsbeschränkung mit Hilfe eines Kraftrückkopplungsgerätes denkbar. Es handelt sich dabei allerdings um eine Hypothese, die im Rahmen dieser Arbeit nicht weiter verfolgt wird, da diese Möglichkeit für das Erreichen des Anwendungszieles nicht entscheidend ist. Ziel des Systems ist die Möglichkeit der einfachen und effektiven Modellierung von Schläuchen. Die Untersuchung von Montagemöglichkeiten verschiedenster Schläuche würde eine solche Erweiterung allerdings interessant erscheinen lassen.

Eine wesentliche Eigenschaft der Stutzenbasierung besteht in der direkten Abfrage der Stutzen in der VR-Objekthierarchie. Zwar wurde im Abschnitt 4.2.3.4. als Voraussetzung genannt, daß zu einem dedizierten Zeitpunkt ausschließlich ein Stutzen aktiv und in seiner Bewegung beeinflußt werden kann, aufgrund der Abfrage der geometrischen Eigenschaften der Stutzen direkt in der Objekthierarchie können sich aber beide an einem Verbindungsablauf be-

teiligten Stutzen frei im Raum bewegen. Diese Möglichkeit ist Voraussetzung für das beidhändige Arbeiten in einer virtuellen Umgebung. Damit kann das Objekt mit dem aktiven Stutzen beispielsweise in der rechten Hand geführt werden, während die linke Hand das Objekt mit dem Referenzstutzen bewegt.

Die beschriebene Implementierung der stutzenbasierten Verbindungsabläufe ist auf einen aktiven Stutzen beschränkt, der in seiner Bewegung beeinflußt wird. Dadurch wird ausgeschlossen, daß sich mehrere Constraints gleichzeitig auf ein und das selbe virtuelle Objekt beziehen und dementsprechend gegenseitig negativ beeinflussen bzw. sich gegenseitig aufheben. Es ist denkbar, mehrere Objekte innerhalb einer Simulationsschleife durch die Bewegungsconstraints zu beeinflussen und so mehrere Verbindungsabläufe zum gleichen Zeitpunkt zu simulieren. Diese Möglichkeit wird im Rahmen dieser Arbeit jedoch nicht näher betrachtet. Sie ist für das Erreichen ds Anwendungszieles nicht erforderlich.

Durch die Einführung einer Verbindungsablaufsimulation unter Zuhilfenahme eines zusätzlichen Zwischenzustandes ist für den Anwender die Möglichkeit vorhanden, eine zu erzeugende Verbindung interaktiv zu beeinflussen und zu korrigieren. Durch die Anlehnung der Verbindungsabläufe an Vorgänge, die im allgemeinen aus der Realität bekannt sind, kann der Anwender einfacher erkennen und nachvollziehen, was im System vorgeht und kann Parallelen zu bekannten Vorgängen ziehen. Das ist eine Voraussetzung dafür, den Einarbeitungsaufwand in das System zu minimieren. Der durch Norman definierte und in Abschnitt 2.3. beschriebene *gulf of execution* wird damit verkleinert.

Die constraintbasierten Verbindungsabläufe arbeiten völlig unabhängig von der eingesetzten Eingabehardware. Sie konnten im Rahmen dieser Arbeit in Verbindung mit der allgemein bekannten 2D-Computermaus, mit der DLR-Spacemouse und mit einem magnetischen Trackingsystem (Polhemus Fastrak Long- und Shortranger, Ascension Flock of Birds) getestet werden. Mit allen getesteten Positionierungssystemen konnten die verschiedenen Verbindungsabläufe durchgeführt werden. Aufgrund der möglichen Parametrisierung der Verbindungsabläufe ist es darüber hinaus möglich, eine Anpassung an die verwendete Eingabehardware durchzuführen. Diese Parametrisierung ist von der Eingabegenauigkeit der verwendeten Systeme abhängig. So können die virtuellen Objekte mit Hilfe der DLR-Spacemouse und der 2D-Computermaus sehr exakt, aber relativ umständlich positioniert werden. Im Gegensatz dazu können die Interaktionen mit einem magnetischen Trackingsystem in natürlicher Art und Weise durchgeführt werden, während die Positionierungsergebnisse aber infolge der Ungenauigkeit dieser Geräte relativ unpräzise sind. Durch die Robustheit der stutzenbasierten Verbindungsabläufe gegenüber der unterschiedlichen Positionierungsgenauigkeit ist es immer möglich, das angestrebte Interaktionsziel, die Erzeugung oder Trennung einer Verbindung, zu erreichen.

Es hat sich erwiesen, daß nicht ausschließlich die Genauigkeit der Eingabegeräte entscheidend dafür ist, eine Verbindung zwischen verschiedenen Objekten einer virtuellen Umgebung interaktiv erzeugen zu können. Der Prozeß einer Verbindungserstellung in der Realität setzt sich aus mehreren unter-

schiedlichen Komponenten zusammen. Stets handelt es sich dabei um einen Regelkreis, über dessen Rückkopplung die ausführende Person ständig ihre Interaktion korrigiert. Dieser komplexe Regelkreis, der sich aus der visuellen Wahrnehmung, der akustischen und vor allem der haptischen Wahrnehmung zusammensetzt, wird in einer virtuellen Umgebung im wesentlichen auf den visuellen Kanal reduziert. Insbesondere die fehlende haptische Wahrnehmung und die damit verbundene direkte Beeinflussung der Bewegung der Hand und somit der Bewegung des gegriffenen Objektes, wie sie aus der Realität als Normalität bekannt ist, läßt sich nicht ausschließlich durch den Einsatz eines überaus exakten Eingabegerätes kompensieren. Die Positionierungsgenauigkeit der Hand im freien Raum[88] mit einer rein visuellen Rückkopplung ist durch die motorischen Möglichkeiten des Menschen beschränkt. Damit ist die Abfrage auf die Existenz einer Verbindung in einem technischen System, die im Submillimeterbereich angesetzt ist, in den weitaus meisten Fällen durch ein negatives Ergebnis gekennzeichnet. Durch den Einsatz der stutzenbasierten Bewegungsbeeinflussung und der darauf aufbauenden Verbindungsabläufe wird garantiert, daß eine erzeugte Verbindung auch vom System als solche erkannt werden kann. Damit sind sowohl eine fehlende Kraftrückkopplung wie auch die Ungenauigkeit der verschiedenen Eingabegeräte überwunden. Auf eine aufwendige exakte physikalische Simulation mit ihren Nachteilen in der Komplexität von Datenvorbereitung und Berechnung kann verzichtet werden.

Ein Nachteil der stutzenbasierten Verbindungsabläufe besteht in der Anzahl der unterschiedlichen Parameter, die während der Vorbereitung einer Anwendung bestimmt und gesetzt werden müssen. Diese Parameter können nicht aus der Geometrie der entsprechenden Objekte hergeleitet werden. Das gilt für die einzelnen Datensätze in den verschiedenen CAD-Ursprungsformaten ebenso wie für die Objekte im VR-Format. Beide Objektbeschreibungen bestehen im wesentlichen aus einer reinen Flächenbeschreibung, aus der weitergehende semantische Informationen über Verbindungseigenschaften, Simulationsverhalten oder ähnliches nicht ableitbar sind. Eine Möglichkeit der Verringerung des daraus resultierenden Anwendungsvorbereitungsaufwandes ist die Klassifizierung von Objekten nach entsprechenden Merkmalen, wie sie beispielsweise diese Verbindungsparameter darstellen. Diesen Objektklassen können die im CAD-System erzeugten geometrischen Beschreibungen zugeordnet werden. Das erfordert zwar immer noch einen erhöhten Anwendungsvorbereitungsaufwand, er reduziert sich aber wesentlich. Die einzelnen spezifischen Parameter der entsprechenden Verbindungsabläufe können so vor dem Anwender verborgen werden und er benötigt kein Expertenwissen über die Erzeugung und Zuordnung dieser Parameter zu den einzelnen virtuellen Objekten, im speziellen zu den Stutzen. Es sei an dieser Stelle auf den Abschnitt 4.3. verwiesen, in dem diese Problematik noch weiter vertieft wird.

[88] Hier wird die Position und die Orientierung betrachtet

4.2.4. Aufwandsabschätzung für die Geometriemodellerstellung der Stutzen

Die Generierung der Stutzengeometrie spielt eine wesentliche Rolle bei der Akzeptanz des beschriebenen Systems. Das leitet sich aus der Tatsache ab, daß eine Datenstruktur, die eine Stutzenverbindung beschreibt, nicht standardmäßig in einem CAD-Modell enthalten ist. Daraus ergibt sich die Notwendigkeit, ein entsprechendes Modell im Rahmen der Anwendungsvorbereitung effektiv erzeugen zu können.

Bei der Stutzengeometrie handelt es sich um eine sehr einfache geometrische Beschreibung. Diese besteht aus zwei geordneten Punkten im Raum (siehe Abschnitt 4.2.1.), aus denen sich die übrige Geometrie ableiten läßt. Der mit Hilfe der Punkte beschriebene Ortsvektor definiert eindeutig die Verbindungsrichtung des Stutzens. Sowohl die Kollisionsgeometrie wie auch der Zusatzvektor lassen sich daraus durch einfache Vektoroperationen und Generierungsfunktionen automatisch erzeugen. Damit sind im Rahmen der Stutzenerzeugung lediglich die beiden den Stutzenvektor definierenden Punkte zu ermitteln und entsprechend dem Stutzen zuzuordnen. Diese automatische Stutzenerzeugung wird standardmäßig bei den CAD-Entities angewendet (siehe Abschnitt 4.2.2.) und ist demzufolge auch auf die übrigen Stutzen übertragbar.

Im Vergleich zu einer physikalisch basierten Verbindungserzeugung handelt es sich um ein sehr einfaches und daher auch einfach zu erzeugendes Simulationsmodell. Dies ist eine wichtige Voraussetzung für die Akzeptanz des Systems. Ein physikalisch basiertes Verbindungsmodell ist direkt von Materialwerten und geometrischem Aufbau der CAD-Daten abhängig und damit weitaus aufwendiger zu erzeugen. Bei einer Veränderung von Geometrie und Material ist jedesmal ein Neuaufbau dieses Modells notwendig. Das wiederum hat einen erhöhten und nicht wünschenswerten Vorbereitungsaufwand für die Anwendung zur Folge.

4.2.5. Einordnung der stutzenbasierten Verbindungsoperationen in den Anwendungskontext

Das Ziel des beschriebenen Systems besteht darin, in einer virtuellen Umgebung den Verlauf von Leitungsobjekten modellieren zu können, um so den Anwender in einer frühen Modellierungsphase zu unterstützen (siehe auch Abschnitt 2.2.2.). Dabei kommt es auf eine einfache Benutzerschnittstelle zwischen Mensch und Maschine an, auf Bedienmetaphern, die aus der Realität bekannten Abläufen ähneln bzw. ihnen nachempfunden sind. In der Modellierungsphase ist es zunächst nicht wichtig, ob mögliche Interaktionen auch in der Realität durchführbar sind. Bei der Nutzung des Systems hat es sich gezeigt, daß eine exakte Abbildung der Realität den Anwender in seiner Arbeitseffektivität behindern kann. Die Nutzung von realitätsfremden Möglichkeiten, die eine virtuelle Umgebung bietet, kann den Bedienungskomfort und die Effektivität sogar steigern. Zu den erwähnten realitätsfremden Interaktionsmöglichkeiten gehören unter anderem folgende Funktionalitäten:

- Drahtgitterdarstellung und teiltransparente Visualisierung von Objekten

- Gegenseitige Durchdringung von Objekten
- Schnittgenerierung in der virtuellen Umgebung
- Aus- und Einblenden von virtuellen Objekten zur Laufzeit
- Blickpunkte und -richtungen, die in der Realität nicht eingenommen werden können
- Vergrößerungen und Verkleinerungen der virtuellen Objekte einschließlich der Repräsentation des Anwenders in der Umgebung

Bei diesen Interaktionen, die auch aus anderen Modellierungssystemen bekannt sind, handelt es sich um Möglichkeiten, die sich auf die Bedienungseffektivität positiv auswirken, in der Realität aber nicht oder nur mit großem Aufwand möglich sind.

Die stutzenbasierten Verbindungsoperationen stellen eine Interaktionsmöglichkeit dar, die zwar an Abläufe angelehnt ist, die aus der Realität bekannt sind und mit Erfahrungen aus der Realität verglichen werden können. Der dabei entstehende direkte Bewegungsablauf der Objekte kann jedoch von einem Bewegungsablauf in der Realität stark abweichen. Ein Beispiel dafür sind die realisierten Abläufe der Verbindungsoperationen zwischen den CAD-Entities.

Bei der Leitungsverlegung handelt es sich um ein Modellierungssystem, nicht aber um eine Montage-, Freigangs- oder Demontageuntersuchung. Im Gegensatz zu einem solchen System ist bei dem Modellierungssystem das Ergebnis eines Verbindungsablaufes entscheidend und dessen einfache und effektive Bedienbarkeit, nicht aber der Ablauf, der zu diesem Ergebnis führt. Damit genügen die hier beschriebenen stutzenbasierten Verbindungsabläufe den Anforderungen, die an dieses System gestellt werden.

Durch die Anwendung der stutzenbasierten Verbindungsabläufe ist es möglich, mit unterschiedlichster Ein- und Ausgabehardware zu arbeiten. Es hat sich im Laufe der Arbeit gezeigt, daß dabei die Genauigkeit dieser Geräte eine untergeordnete Rolle spielt. Die stutzenbasierten Verbindungsabläufe verhalten sich gegenüber diesen geräteabhängigen Spezifika sehr robust.

Durch den Verzicht auf eine physikalisch exakte Verbindungssimulation ist das Vorhandensein der gesamten am Verbindungsprozeß beteiligten Objekte keine Voraussetzung für die Funktionsfähigkeit des Systems. Die Verbindungsabläufe können damit ohne ein Geometriemodell in der virtuellen Umgebung ablaufen. Die einzige Voraussetzung ist das Vorhandensein der Stutzengeometrie. Damit eröffnet sich die Möglichkeit, auf Geometrieobjekte in einer solchen Umgebung zu verzichten. Dieser Punkt ist sehr wichtig, solange die zur Verfügung stehende Systemhardware in ihrer Performance nur einen eingeschränkten Modellumfang zuläßt. Ein vollständiges Vorderwagenmodell eines frontgetriebenen Automobils ist mit der heute zur Verfügung stehenden Rechnerleistung nicht mit der erforderlichen Performance und gleichzeitig benötigten Genauigkeit darstellbar. Es ergibt sich die Notwendigkeit, das Modell zu vereinfachen, Teile interaktiv auszublenden und die Objekte für den Rende-

ringprozeß zu optimieren. Durch die stutzenbasierten Verbindungsabläufe können diese Funktionalitäten durchgeführt werden, ohne auf die Verbindungsabläufe verzichten zu müssen.

4.3. Objektklassenhierarchie als Grundlage für die Verarbeitung nichtgeometrischer Objektinformationen

4.3.1. Motivation

Im 3. Kapitel (Abschnitt 3.2.3.) wurde ausgeführt, wie die Objekte in einem VR-System beschrieben sind. Dabei handelt es sich weitestgehend um eine hierarchisch strukturierte geometrische Repräsentation der einzelnen Objekte. Diese Repräsentation wurde um eine geometrische Datenstruktur, die Stutzen, erweitert, um Verbindungsstellen zu kennzeichnen. Im vorangegangenen Abschnitt wurden Verbindungsoperationen beschrieben, die auf der Stutzengeometrie ablaufen können. Dabei können verschiedene Ablaufarten unterschieden werden, die sich wiederum durch eine unterschiedliche Parametrisierung verschieden verhalten können. In den Datenstrukturen eines VR-Systems sind nicht notwendigerweise Strukturen enthalten, in denen solche Zusatzinformationen abgelegt werden können.

Durch die Einführung der CAD-Entities und der Stutzengeometrie wurde im beschriebenen VR-System bereits eine neue Qualität der Objektklassifizierung eingeführt. Diese Klassifizierung orientiert sich nicht mehr ausschließlich an den geometrischen Primitiven, die für das Rendering entscheidend sind (LODs, Stripes, Linien, Punkte, ...), sondern an den Simulationseigenschaften, die diese Objekte besitzen bzw. an den Eigenschaften und Attributen, die diejenigen realen Objekte besitzen, die mit Hilfe der geometrischen Objektbeschreibung im VR-System abgebildet und simuliert werden.

Die angestrebte hohe Interaktivität des Systems erzwingt es, daß die im System abgelegten Daten so abgebildet sind, wie es ein Anwender auch aufgrund seiner Erfahrungen aus der Realität erwartet. Aus den sich ihm präsentierenden virtuellen Objekten schließt er auf deren Eigenschaften insbesondere im Falle einer Interaktion. Das gilt vor allem für Interaktionen mit zusammengesetzten Objektstrukturen. Stimmt das Verhalten der Objekte mit seinen Erfahrungen und den daraus gefolgerten Hypothesen für das Verhalten überein, erhöht das wesentlich den Bedienkomfort und somit auch die Akzeptanz des Systems. Für dieses angestrebte Ziel sind zusätzliche Datenstrukturen und darauf ablaufende Analysen bzw. Simulationen notwendig, die im System durchgeführt werden.

Für die Implementierung einer entsprechenden Objektsemantik werden Datenstrukturen entwickelt, mit deren Hilfe die unterschiedlichsten für die Leitungsverlegung notwendigen Zusatzinformationen verwaltet werden können (siehe auch Abbildung 44). Damit wird eine Annäherung zwischen dem kognitiven, auf Erfahrungen basierenden Weltbild des Anwenders und dem im Sy-

stem abgelegten Datenmodell angestrebt. Das führt zu einer Verringerung des von Norman beschriebenen *gulf of execution* (Abschnitt 2.3.).

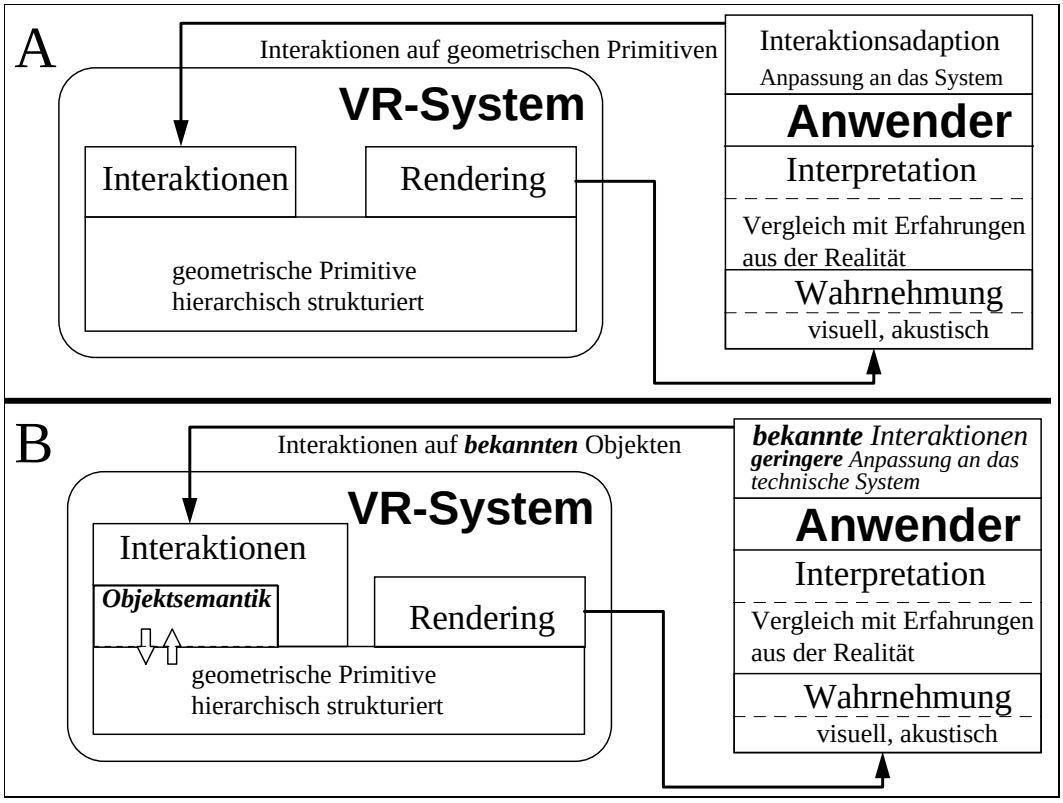


Abbildung 44: Integration einer Objektsemantik in ein VR-System zur realitätsnahen Abbildung eines Weltausschnittes
A:Standardaufbau in einem VR-System
B:erweitertes VR-System

4.3.2. Anforderungen an eine semantische Klassenhierarchie

Im Folgenden werden die Vorüberlegungen näher beschrieben, die zu der Systemerweiterung um eine Objektsemantik und deren Design geführt haben. Es wird zunächst die softwaretechnische Umsetzung dieser Idee diskutiert und begründet und anschließend auf die funktionalen Ziele eingegangen.

4.3.2.1. Softwaretechnische Anforderungen

Zunächst ist als notwendige Voraussetzung zu nennen, daß ein VR-System um zusätzliche semantische Objektinformationen zu erweitern ist. Daraus ergeben sich mehrere wesentliche Anforderungen:

1. Systemperformance

Wie bereits im 3. Kapitel beschrieben handelt es sich bei einem VR-System um eine Simulationsschleife, die unter Echtzeitbedingungen arbeitet. Sie präsentiert dem Anwender unmittelbar Veränderungen der Datenbasis über den Renderingprozeß. Dabei treten Verzögerungen zwischen Aktion (Eingabe des Anwenders) und Reaktion (Beenden des Rendering) im Millisekundenbereich auf. Die Systemperformance muß erhalten bleiben. Sie

stellt eine Voraussetzung für die Nutzung der unterschiedlichsten VR-spezifischen Hardware dar.

2. Direkte Manipulation

Die Interaktionen in einem VR-System erfolgen durch direkte Manipulation der Objekte in der Umgebung. Diese an die Realität angelehnte Interaktionsmetapher stellt die Hauptkomponente der Mensch-Maschine-Kommunikation dar. Wie bereits in Abschnitt 3.1. erläutert, handelt es sich bei einem VR-System im wesentlichen um eine Mensch-Maschine-Schnittstelle. Diese soll dem Anwender möglichst realitätsnahe Interaktionen anbieten. Daher soll dem Anwender als Hauptkommunikationsmöglichkeit die direkte Manipulation der geometrischen Objekte zur Verfügung stehen.

3. Datentrennung

Die zusätzlichen Datenstrukturen werden getrennt vom VR-System abgelegt. Das betrifft sowohl die Datenhaltung zur Laufzeit wie auch die Datenhaltung im Rahmen der Anwendungsvorbereitung und Anwendungssicherung.

Voraussetzung für die Integration zusätzlicher Datenstrukturen in das Geometriemodell eines VR-Systems wäre ein sehr tiefer Eingriff in das VR-System selbst. Diese enge Kopplung wäre demzufolge auch stark von den internen Strukturen eines solchen Sytems abhängig. Mehrere Gründe sprechen gegen diesen Ansatz[89]:

A) Offenlegung der inneren Datenstrukturen durch den Systemanbieter

Für die Erweiterung der internen Modellstrukturen eines VR-Systems müßten die vorhandenen Strukturen frei erweiterbar sein. Das setzt eine Offenlegung des entsprechenden Datenmodells und dessen Erweiterbarkeit durch den Systemanbieter voraus. Diese Möglichkeit besteht bei den wenigsten auf dem Markt erhältlichen Systemen. Im Falle eines System-Redesign würde dies zu erheblichen und tiefgreifenden Anpassungen an ein entsprechendes neues System führen.

B) Fehlende Standards für die Datenhaltung in VR-Systemen

Ein entsprechender Ansatz stellt eine Speziallösung für ein gegebenes System dar. Die internen Datenstrukturen und deren Verwaltung in verschiedenen VR-Systemen unterscheiden sich grundlegend. Diese Tatsache impliziert den nächsten Nachteil solch eines Ansatzes.

C) Portierungsmöglichkeit zwischen verschiedenen Systemen

Die Entwicklung kommerziell verfügbarer VR-Systeme befindet sich noch in einem sehr frühen Stadium. Die Entwicklung immer schnellerer Systemhardware und fehlende, vom Anbieter unabhängige Standards führen dazu, daß sich diese Systeme in einem ständigen Fluß befinden, daß

[89] Es sei an dieser Stelle angemerkt, daß die Praxisnähe dieser Arbeit auch einen nicht zu unterschätzenden Einfluß auf die Gestaltung des Systems genommen hat. Das äußert sich explizit in den folgenden 3 Punkten!

sie immer wieder auf neue Systemsoftware angepaßt werden müssen. Ein weiterer Grund besteht in der zwangsläufig notwendigen Performance eines solchen unter Echtzeitbedingungen laufenden Systems. Daraus ergibt sich eine zwangsläufige Nähe der Software zur genutzten Hardware, die sich in einer ständigen Weiterentwicklung befindet. Daher ist auch in der näheren Zukunft damit zu rechnen, daß bestehende VR-Systeme immer weiterentwickelt werden und daß ständig neue Systeme angeboten werden. Der Portierungsaufwand des Systems muß bei der Entwicklung daher ebenfalls berücksichtigt werden.

Aus den genannten drei Gründen wurde für die Datenerweiterung im System eine lose Kopplung favorisiert. Das führt zu einer Trennung zwischen den geometrischen und den semantischen Informationen im System. Es wird daher auch im Folgenden zwischen dem „VR-System“ und der „Objektsemantik“ als getrennte Systeme unterschieden.

Die Integration zusätzlicher Datenstrukturen in ein vorhandenes System impliziert auch die Möglichkeit der Abspeicherung dieser Informationen. Zwei Möglichkeiten sind dabei zu diskutieren.

A) Erweiterung eines vorhandenen Geometrieformates

Eine Erweiterung eines vorhandenen Beschreibungsformates für virtuelle Objekte hat den Vorteil, daß alle ein Objekt beschreibenden Primitive und Attribute in einem Datensatz enthalten sind. Dieser Ansatz ist bei einer engen Kopplung zwischen vorhandenen Systemdaten und den zusätzlichen Strukturen direkt im Systemkern von Vorteil.

Von Nachteil ist die Tatsache, daß eine entsprechende Formaterweiterung eine Erweiterung der Datenimport- und Datenexportfunktionalitäten des VR-Systems erforderlich macht. Dabei handelt es sich um Routinen, die zum Standardumfang eines typischen VR-Systems gehören und im allgemeinen nicht für eine entsprechende Erweiterung und Anpassung vorgesehen sind. Weiterhin ist ein erweitertes Datenformat nicht unbedingt kompatibel zu anderen Systemen, die das gleiche Format verarbeiten können.

B) Einführung eines zusätzlichen Datenformates

Die Unabhängigkeit von Import- und Exportfunktionalitäten eines VR-Systems wie auch die interne Trennung zwischen Geometriebeschreibung und Objektsemantik führen zu einer Favorisierung eines zusätzlichen Datenformates, getrennt von existierenden VR-Datenbeschreibungsformaten.

4. Systemtrennung

Entsprechend der Datentrennung zwischen Objektsemantik und VR-Objekthierarchie wird auch eine Systemtrennung zwischen VR-System und Objektsemantik angestrebt. Eine lose Kopplung beider Teilsysteme soll die

Parallelisierbarkeit verschiedener Verarbeitungsschritte ermöglichen. Weiterhin wird dadurch eine Asynchronität von der Simulationsschleife des VR-Systems angestrebt. Damit wird eine weitestgehende Erhaltung der ohne die Systemerweiterung möglichen Renderingperformance erzielt.

Wie beim dritten Punkt, der Datentrennung, spielen auch hier praktische Aspekte wie die Unabhängigkeit vom System und die Portierbarkeit des Systems eine Rolle.

Durch die System- und Datentrennung der Objektsemantik wird eine lose Kopplung des Systems erreicht. Es entsteht eine strikte Trennung zwischen geometrischer und semantischer Datenbeschreibung. Darüber hinaus ist aber ein gegenseitiger Datenzugriff der beiden Teilsysteme und deren Synchronisation zu gewährleisten.

5. *Objektorientierung*

   Wie im eigentlichen VR-System ist auch in der Objektsemantik ein objektorientierter Implementationsansatz anzustreben. Dieser unterstützt softwaretechnisch die notwendige Klassifizierung der einzelnen Objekte nach gemeinsamen wie auch unterschiedlichen Eigenschaften. Weiterhin wird eine Erweiterung eines Systems um neue Objekte auf der Grundlage der Vererbungsmechanismen unterstützt. Durch die Nähe einer entsprechenden Objektsemantik zu den eigentlichen Daten, den virtuellen Objekten, ist eine entsprechende Anpassungsmöglichkeit eines solchen Systems an neu zu erzeugende Anwendungsumgebungen unausweichlich.

4.3.2.2. Funktionale Anforderungen und Ziele

Die semantischen Informationen über die Objekte erweitern das geometrieorientierte Datenmodell des VR-Systems um Strukturen und Informationen, die über die reine geometrische Repräsentation der Daten hinaus gehen. Es handelt sich um anwendungsorientierte Zusatzinformationen, die speziell auf die Anwendung „Leitungsverlegung“ zugeschnitten sind. Mit der Implementierung dieser Objektsemantik werden verschiedene Anforderungen umgesetzt und Ziele verfolgt, die im Kapitel 5 weiter diskutiert werden. Hier sollen einige Grundkonzepte erläutert werden, die bei der Umsetzung beachtet werden müssen.

Der erste Schwerpunkt liegt auf einer relativ einfachen Möglichkeit der Anwendungsaufbereitung. Zusätzliche Attribute und Objektinformationen sind datenabhängig und bedürfen eines zusätzlichen Aufwandes bei der Vorbereitung einer Anwendung. Dieser Aufwand muß aus Akzeptanzgründen minimiert werden. Daher ist es notwendig, die im Rahmen der Anwendungsvorbereitung durchzuführenden Schritte so redundanzfrei wie möglich zu halten. Anpassungen des Systems an die einzuladende virtuelle Umgebung müssen einfach realisierbar sein. Eine Basis dafür kann eine skriptartige Beschreibungssprache darstellen, die durch eine grafische Bedienoberfläche erzeugt und bearbeitet werden kann.

Bei der Anwendungsvorbereitung ist auf internes Systemwissen so weit wie möglich zu verzichten. Dieses Systemwissen sollte vor dem Anwender gekapselt werden. Als Beispiel sei hier die interne Strukturierung der VR-Objekthierarchie genannt, über die das Objektverhalten bei Interaktionen definiert werden kann. Dieses Spezialwissen über geometrische Primitive und deren Strukturierung in einer Baumhierarchie ist bei der letztendlichen Anwendung für den Nutzer nicht von Interesse.

Mit Hilfe der Objektsemantik wird ein spezieller kleiner Weltausschnitt, zugeschnitten auf eine spezifische Anwendung, in einem objektorientierten Datenmodell abgebildet. Mit diesen Zusatzinformationen soll das System in die Lage versetzt werden, im Falle der Interaktionen des Anwenders mit den geometrischen Objekten des Systems ein Objektverhalten umzusetzen, das der Erwartungshaltung des Anwenders entspricht. Dazu müssen auf den Objekten der Semantik wie auch der VR-Objekthierarchie Analysen ablaufen, auf deren Basis bestimmte Entscheidungen durch das System getroffen werden. Das setzt zum einen ein festgelegtes Regelwerk voraus und zum anderen eine Kommunikationsmöglichkeit zwischen VR-System und Objektsemantik. Im Rahmen dieser Kommunikation muß die Möglichkeit bestehen, auf die geometrischen Daten in der VR-Objekthierarchie zu Analysezwecken zuzugreifen wie auch die ablaufenden Interaktionen direkt zu beeinflussen.

Mit Hilfe der zusätzlichen Objektinformationen und den darauf ablaufenden Algorithmen wird das Ziel verfolgt, daß sich das System immer wieder an neue Situationen anpaßt und sich bei den Interaktionen entsprechend verhält. Dabei soll der Anwender stets direkt mit der grafischen Repräsentation der Objekte interagieren und auch stets die treibende Kraft sein. Ziel ist es dabei nicht, daß das System voll- oder semiautomatisch Leitungen und Schläuche im freien Raum verlegt. Vielmehr soll das der Anwender selbst tun und dementsprechend sein umfangreiches Expertenwissen schöpferisch umsetzen. Dabei stellt das System lediglich ein Werkzeug dar, mit dessen Hilfe er diese Tätigkeit effektiv erledigen können soll. Ausschließlich der das System nutzende Mensch bestimmt den Verlauf der einzelnen Leitungsobjekte. Die möglichen Interaktionen sollen dabei am Anwendungsziel orientiert sein und nicht an den internen Strukturen der geometrischen Objekte.

Die Erweiterung des Systems um die Objektsemantik und den darauf ablaufenden Prozessen stellt eine Erweiterung des in Abbildung 7 dargestellten 4. Punktes „Aktualisierung der Objekte“ dar. Der Ablauf dieser Objektaktualisierung ist dabei an die beim Menschen ablaufenden Verarbeitungsschritte angelehnt, die bei einer Kommunikation/Interaktion mit seiner Umwelt ablaufen. Bereits in Abschnitt 2.3. wurde dieser Ablauf, wie er von Norman[90] beschrieben wurde, erläutert. Dabei werden die beiden Hauptphasen „Evaluierung“ und „Execution“ in insgesamt sieben Teilphasen untergliedert.

[90] [DIX93] S.93ff.

<table>
<tr><th>Nr</th><th>Mensch</th><th>System</th></tr>
<tr><td>1.</td><td>Systemstatus feststellen</td><td rowspan="3">Evaluierungsphase:
- Analyse der gerade ablaufenden Interaktion
- Analyse der Objektsemantik
- Analyse der VR-Objekthierarchie</td></tr>
<tr><td>2.</td><td>Systemstatus interpretieren</td></tr>
<tr><td>3.</td><td>Auswerten des Systemstatus in Hinblick auf das angestrebte Interaktionsziel und die zugehörigen Teilziele</td></tr>
<tr><td>4.</td><td>Festlegung des Interaktionsteilzieles</td><td rowspan="4">Execution-Phase:
- Aktualisierung der Objektsemantik
- Beeinflussung der VR-Objekthierarchie
- Beeinflussung der VR-Simulationsprozesse
- Fertigmeldung bei Datenkonsistenz an die VR-Simulationsschleife</td></tr>
<tr><td>5.</td><td>Auswählen einer spezifischen Aktion, die zur Erreichung des Zieles notwendig ist</td></tr>
<tr><td>6.</td><td>Spezifizierung einer Interaktionsabfolge</td></tr>
<tr><td>7.</td><td>Durchführen der daraus resultierenden Teilaktion</td></tr>
</table>

Tabelle 3: Vergleich der Verarbeitungsschritte beim Menschen mit den angestrebten Interaktionsabläufen im System

In Tabelle 3 werden die Abläufe beim Menschen den Abläufen in der Objektsemantik gegenübergestellt, die bei der Objektaktualisierung durchgeführt werden. Auch während der Objektaktualisierung wird zwischen der Evaluierungs- und der Executionphase, wie beim Interaktionsmodell nach Norman, unterschieden. Während der Evaluierungsphase analysiert das System die bestehende Situation. Dabei greift die Objektsemantik direkt auf die Datenstrukturen des VR-Systems zu, um die eigenen Attribute zu aktualisieren. Im Gegensatz zur Mensch-Maschine-Kommunikation, bei der ein Renderingprozeß den aktuellen Systemstatus in eine für den Menschen interpretierbare Form bringen muß, wird hierbei direkt auf die internen Datenstrukturen des VR-Systems zugegriffen. Auf der Grundlage der aktualisierten Attribute interpretiert das System die bestehende Situation. Das Ergebnis dieser Interpretation ist eine direkte Beeinflussung der Datenstrukturen und Simulationsabläufe im VR-System. Diese Beeinflussung ist mit der Executionphase des Interaktionsmodells nach Norman zu vergleichen.

Die beschriebene Idee zur Systemerweiterung führt zu dem in Abbildung 45 dargestellten Systemaufbau. Das VR-System wird durch einen Interaktionsmanager erweitert.

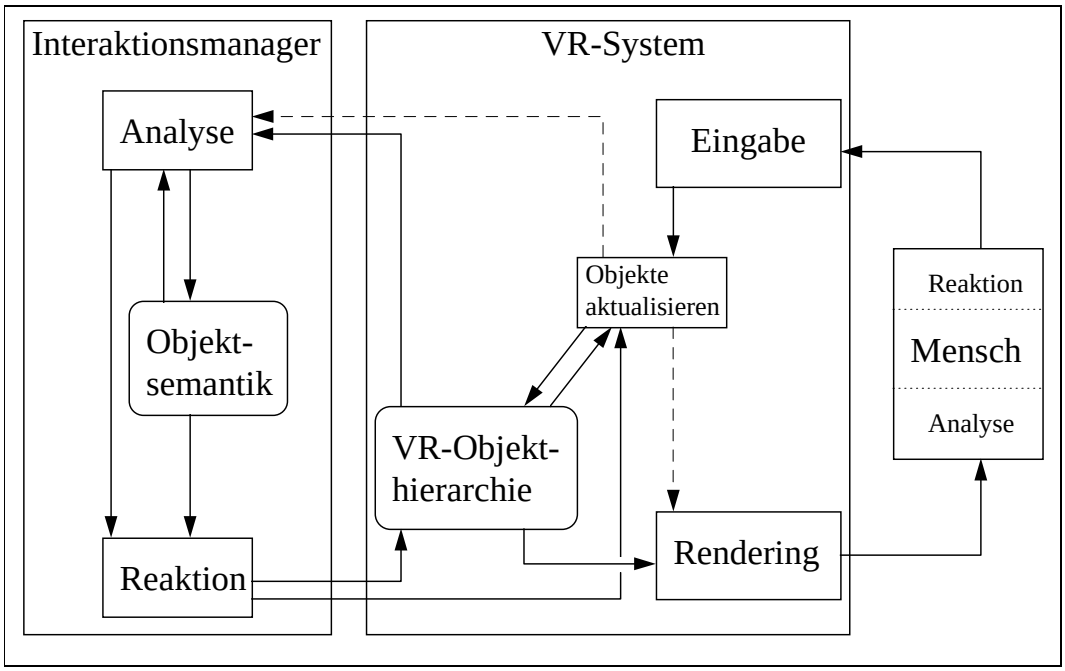


Abbildung 45: VR-Systemerweiterung um einen auf einer Objektsemantik aufbauenden Interaktionsmanager

Der Interaktionsmanager ist lose mit dem VR-System gekoppelt. Er wird durch die Objektaktualisierung des VR-Systems getriggert. Daraufhin erfolgt eine Analyse der aktuellen Situation, bei der direkt auf die VR-Objekthierarchie zugegriffen werden kann. Diese hat eine Aktualisierung der Objektsemantik zur Folge. Anschließend erfolgt eine Reaktion, die sich auf die VR-Objekthierarchie wie auch auf die im VR-System ablaufenden Simulationen auswirkt. Dieser Ablauf erfolgt nicht bei jedem Durchlauf der VR-Simulationsschleife. Es wird zwischen Interaktionen unterschieden, bei denen der Interaktionsmanager getriggert werden muß und bei denen das nicht notwendig ist.

4.3.3. Diskussion des Implementierungssystems

Die Implementierung des Interaktionsmanagers, der aus der eigentlichen Objektsemantik und den darauf ablaufendenden Verarbeitungsprozessen besteht, erfolgte im System CLIPS. Clips steht als Abkürzung für **C** **L**anguange **I**ntegrated **P**roduction **S**ystem. CLIPS beinhaltet die Basiskomponenten eines regelbasierten Systems:

- Faktenwissen: Daten, auf deren Basis Schlußfolgerungen getroffen werden
- Wissensbasis: enthält alle Regeln
- Inferenzmaschine: steuert den gesamten Ablauf der Regelabarbeitung

Darüber hinaus bietet das System einen objektorientierten Programmieransatz, der in der COOL, der **C**lips **O**bject **O**riented **L**anguage, verankert ist.

Als drittes wesentliches Merkmal sind die Eigenschaften einer prozeduralen Sprache zu nennen, die ebenfalls von CLIPS unterstützt werden[91].

Allein diese breitgefächerten programmierspachlichen Merkmale gaben noch nicht den Ausschlag für die Entscheidung, dieses System zu nutzen. Das System wurde am NASA/Johnson Space Center entwickelt. Dabei wurden drei wesentliche Ziele verfolgt:

1. Einfache Portierbarkeit und damit Unabhängigkeit von Hardware und Betriebssystem
2. Geringe Systemkosten
3. Einfache Integrationsmöglichkeit mit externen Systemen

Der erste wie auch der dritte Punkt haben die Entscheidung für dieses System wesentlich beeinflußt. Darüber hinaus ist die ausführliche Systemdokumentation zu erwähnen.

Der in dem System entwickelte Code, die objektorientierten Strukturen, die Fakten und Regeln können in jedem beliebigen Texteditor verändert und erweitert werden. Die Compilierung des Quellcodes erfolgt erst zur Ladezeit. Durch diese Vorcompilierung erfolgt die Ausführung des Codes sehr schnell. Die Datenbasis kann jedoch ständig erweitert und angepaßt werden.

4.3.4. Die Klassenhierarchie der Objektsemantik

Wie bereits in den vorangegangenen Abschnitten dargestellt, dient die Objektsemantik vorrangig dem Ziel, zusätzliche Informationen über die virtuellen Objekte abzulegen und zu verwalten.

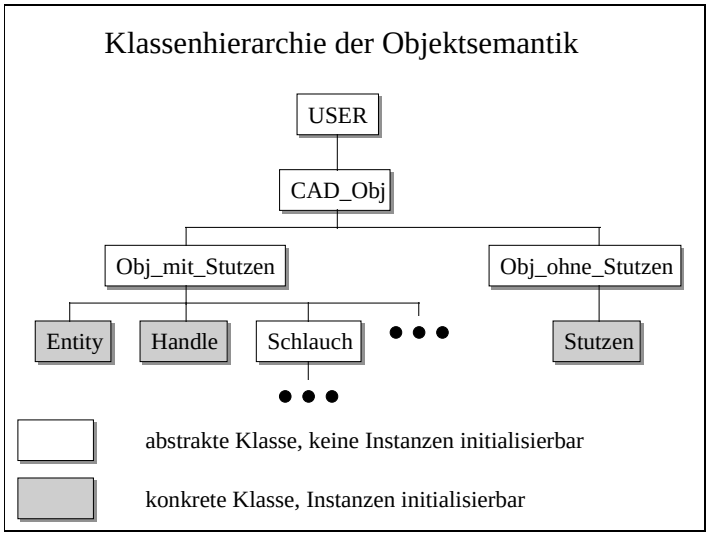


Abbildung 46: Basis-Klassenhierarchie der Objektsemantik

[91] [GiarRile] S.363ff.

Der Kern der Objektsemantik besteht aus einer Klassenhierarchie, in deren Instanzen zur Laufzeit diese Zusatzinformationen initialisiert, abgelegt und aktualisiert werden. Die Basis für die Klassenhierarchie hat immer den in Abbildung 46 dargestellten Aufbau. Dieser Aufbau ist datenunabhängig und bleibt somit immer gleich.

Die Klasse USER ist die im System CLIPS zur Verfügung stehende Oberklasse, aus der sich alle selbstdefinierten Unterklassen ableiten. Grundsätzlich wird zwischen abstrakten und konkreten Klassen unterschieden. Aus abstrakten Klassen können keine Instanzen der entsprechenden Klasse initialisiert werden. Sie dienen lediglich dem strukturierten Aufbau der Klassenhierarchie. Lediglich aus den konkreten Klassen lassen sich Objekte (Instanzen) initialisieren. In der Objektsemantik gibt es eine gemeinsame Oberklasse, die Klasse *CAD_Obj*. Eine Klasse beinhaltet Klassenvariablen, die sogenannten *Slots* und Klassenmethoden, die mit *Messagehandler* bezeichnet werden. Werden aus einer Oberklasse Unterklassen abgeleitet, so werden alle Slots und Messagehandler von der Oberklasse an die Unterklasse „vererbt“. Es können neue hinzugefügt beziehungsweise existierende „überladen“ also neu definiert werden[92].

Slot	Beschreibung
KinderAnz	Anzahl der Kinder des Objektes
Kinder	Geordnetes Feld mit Instanznamen von Kindern
ParentInstance	Instanznamen der Vaterobjekt(e)
linkGrabStat	Statusvariable (TRUE/FALSE)
CAD_VR	Objektstatus CAD oder VR
K-Fest	Karosseriefest (unbeweglich) oder nicht (beweglich) (TRUE/FALSE)

Tabelle 4: Wichtigste Slots der Klasse CAD_Obj

In der beschriebenen Objektsemantik wird grundlegend zwischen zwei Objektklassen unterschieden, die sich von der Oberklasse *CAD_Obj* ableiten, die Klassen *Obj_mit_Stutzen* und *Obj_ohne_Stutzen*.

Die Klasse der *Stutzen* selbst leitet sich von der Oberklasse *Obj_ohne_Stutzen* ab. Dabei handelt es sich um eine konkrete Klasse, aus der zur Laufzeit Instanzen initialisiert werden. Für jeden in der VR-Objekthierarchie eingeladenen Stutzen existiert eine Instanz der Klasse *Stutzen* in der Objektsemantik. Um die in Abschnitt 4.2.2. erläuterte Zuordnung der Stutzen zu den entsprechenden virtuellen Objekten auch in der Objektsemantik abzubilden, existieren in jedem Objekt der Klassenhierarchie die Slots *ParentInstance*, *Kinder* und *KinderAnz*. Diese sind in der Oberklasse CAD_Obj definiert. Über diese Slots kann eine dem VR-System ähnliche Objekthierarchie aufgebaut werden.

[92] [GIARR]

Slot	Beschreibung
Stutzen_MLine_Pointer	Adresse der geometrischen Beschreibung des Stutzenvektors
Connect_to	Feld der Stutzeninstanzen, mit denen der Stutzen eine Verbindung eingehen kann
CB-with	Feld der Stutzeninstanzen, mit denen eine Kollisionserkennung aktiv ist
Connected	Enthält Informationen über den Verbindungszustand des Stutzens (TRUE/FALSE)
Connected-Is-False	Trennungszwang für den Stutzen (TRUE/FALSE)
EntityStutzen	Ist es ein Stutzen eines Entities? (TRUE/FALSE)

Tabelle 5: Wichtigste Slots der Klasse Stutzen

Alle in das VR-System eingeladenen Objekte, die einen oder mehrere Stutzen besitzen, besitzen auch eine Repräsentation in der Objektsemantik. Diese Klassen leiten sich von der Oberklasse *Objekt_mit_Stutzen* ab. In Abbildung 46 sind die Klassen *Handle* und *Entity* als konkrete Klassen dargestellt. Diese sind für die Repräsentation der CAD-Entities notwendig und standardmäßig in der Klassenhierarchie enthalten. Weiterhin existiert eine abstrakte Klasse *Schlauch*, von der sich alle möglichen Schlauchklassen ableiten. Zusätzlich zu den beschriebenen Unterklassen von *Objekt_mit_Stutzen* können, abhängig von der geladenen VR-Szene, weitere Klassen abgeleitet werden, die nicht zu den Entities, Handles und Schläuchen gehören, denen aber Stutzen zugeordnet werden.

Slot	Beschreibung
StutzenAnz	Anzahl der zugehörigen Stutzen
StutzenFeld	Geordnetes Feld von Stutzen-Instanznamen
ConnectClass	Geordnetes Feld von Klassennamen
ConnectInstanceNr	Geordnetes Index-Feld
Folgend Parameterwerte für die Verbindungsabläufe:	
ConstraintAType	Geordnetes Feld von Verbindungsablauftypen
ConstraintWinkel	Geordnetes Feld von Floatwerten
ConstraintFixEnd	Geordnetes Feld von Floatwerten
ConstraintRelFixed	Geordnetes Feld von Floatwerten
ConstraintRelTotal	Geordnetes Feld von Floatwerten

Tabelle 6: Wichtigste Slots der Klasse Obj_mit_Stutzen

Mit Hilfe der beschriebenen Klassenhierarchie lassen sich verschiedenste im VR-System eingeladene Grafikobjekte unterschiedlichen Objektklassen zuordnen. Durch diese Klassenzuordnung und die jeweils in den entsprechenden

Instanzen enthaltenen Werte der Slots unterscheiden sich die unterschiedlichen Objekte voneinander.

Die Hauptinformation der Instanzen der Klasse *Obj_mit_Stutzen* ist die Verbindungsmöglichkeit dieser Objekte mit anderen Objekten. Dazu gehören die geometrische Repräsentation des Objektes selbst, der geometrische Ort einer Verbindung, die Art des Verbindungsablaufes und das oder die konkreten Partnerobjekte, mit denen eine Verbindung eingegangen werden kann. Diese Informationen sind in den Slots der Instanzen der Klasse *Obj_mit_Stutzen* enthalten. Sie entsprechen im wesentlichen den Informationen, die ein Anwender aus der geometrischen Repräsentation eines Objektes aufgrund seiner Erfahrungen ableiten kann, die in der CAD-Repräsentation eines Objektes aber nicht enthalten sind.

4.3.5. Verarbeitung allgemeingültiger und situationsabhängiger Objektinformationen anhand eines Beispieles

Die Strukturierung der semantischen Informationen in der Klassenhierarchie wird am Beispiel eines Luftschlauches und eines Schlauchverbindungsstückes näher erläutert.

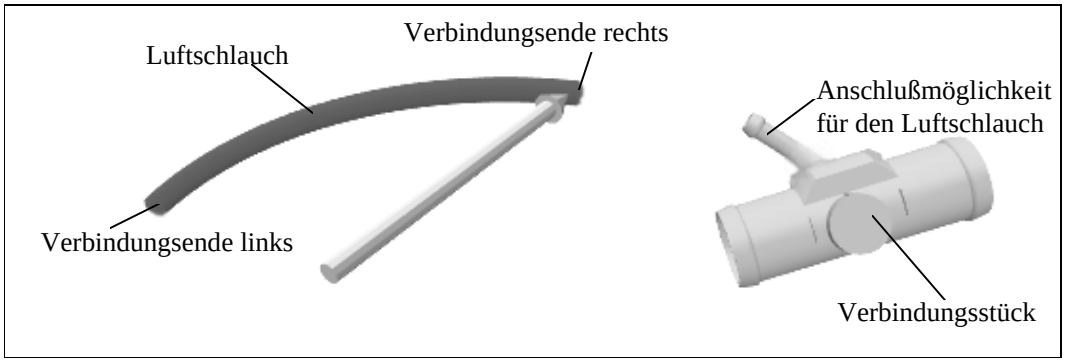


Abbildung 47: Schlauch und Schlauchverbindungsstück mit Informationen, die aus der geometrischen Repräsentation geschlußfolgert werden können

In Abbildung 47 sind die Informationen als Text dargestellt, die der Anwender beim Analysieren der dargestellten Szene sofort erhält. Dabei wird zwischen Informationen unterschieden, die konkret eine Besonderheit der aktuellen Situation darstellen und Informationen, die einen allgemeingültigen Charakter haben. Der Anwender erkennt einen Luftschlauch und ein Verbindungsstück. Dabei handelt es sich um eine Besonderheit der aktuellen, ihm sich darstellenden Situation. Aus dieser Erkenntnis kann er Schlußfolgerungen ziehen, die auf der Allgemeingültigkeit bestimmter Merkmale der erkannten Objekte beruhen. Dazu gehören folgende Erkenntnisse:

A) Der Luftschlauch kann an seinen beiden Enden eine Verbindung zu einem anderen Objekt eingehen.

B) Das Verbindungsstück kann an drei verschiedenen Stellen eine Verbindung mit einem Schlauchobjekt eingehen.

C) Handelt es sich um einen bestimmten, ihm bekannten Luftschlauch, dann weiß er, daß genau ein definiertes Ende des Schlauches auf das Verbindungsstück aufgesteckt werden kann.

D) Am Verbindungsstück ist lediglich eine Anschlußmöglichkeit für ein beliebiges Ende des abgebildeten Schlauches verfügbar.

E) Bei dem Verbindungsstück und dem Schlauch handelt es sich um bewegliche Objekte. Die Position des Verbindungsstückes im Raum wird durch die mit ihm verbundenen Schläuche festgelegt.

Seine Erwartungshaltung besteht also in der Hypothese, daß der Schlauch mit der in Abbildung 47 gekennzeichneten Anschlußmöglichkeit des Verbindungsstückes zusammengefügt werden kann.

Der Informationsgehalt in der Objektsemantik ist den Schlußfolgerungen des Anwenders sehr ähnlich. Auch hier wird zwischen allgemeingültigen und konkret von der aktuellen Situation abhängigen Informationen unterschieden. Diese Informationen sind unterschiedlich im System abgelegt. Der Vorteil der *allgemeingültigen Informationen* besteht darin, daß sie völlig unabhängig von den geometrischen Daten und somit von den eingeladenen VR-Objekten existieren. Diese Informationen sind fest in der Klassenhierarchie abgelegt.

Die *konkret von der aktuellen Situation abhängigen Daten* werden in den Instanzen der Klassen gespeichert, die beim Start der Anwendung initialisiert werden.

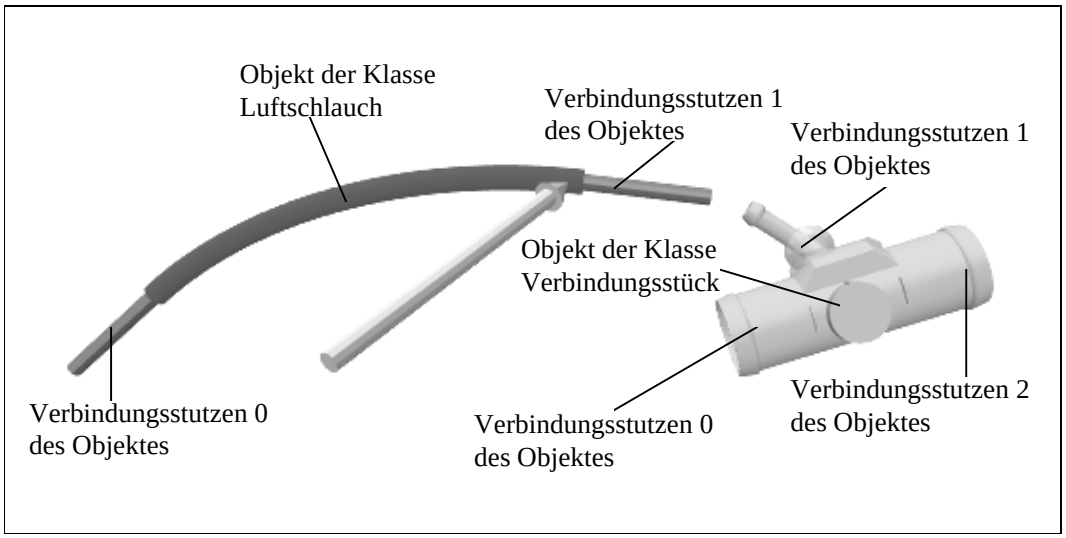


Abbildung 48: Schlauch und Schlauchverbindungsstück mit Informationen, die in der Objektsemantik abgelegt sind

In dem beschriebenen Beispiel werden in der Objektsemantik 7 verschiedene Objekte unterschieden. Dabei handelt es sich um 5 Verbindungsstutzen (Instanzen) der Klasse *Stutzen*, um ein Objekt (Instanz) der Klasse *Luftschlauch* und um ein Objekt (Instanz) der Klasse *Verbindungsstück*. Die Existenz dieser geometrischen Objekte in der virtuellen Umgebung und deren Repräsentation als Instanzen verschiedener Klassen der Objektsemantik ist

von der aktuellen Situation abhängig. Zwischen den beiden Objektrepräsentationen (geometrisch, semantisch) gibt es eine eineindeutige Abbildung.

In der Klassenhierarchie der Objektsemantik sind u.a. folgende Informationen allgemeingültig abgelegt:

A) Ein Objekt (Instanz) der Klasse Luftschlauch kann über genau zwei Stutzen mit anderen Objekten eine Verbindung eingehen.

B) Ein Objekt (Instanz) der Klasse Verbindungsstück kann über genau drei Stutzen mit anderen Objekten eine Verbindung eingehen.

C) Der Stutzen mit der Nummer 1 eines Luftschlauch-Objektes kann eine Verbindung mit dem Stutzen Nummer 1 eines Verbindungsstück-Objektes eingehen.

D) Der Stutzen mit der Nummer 1 eines Verbindungsstück-Objektes kann eine Verbindung mit dem Stutzen Nummer 1 eines Luftschlauch-Objektes eingehen.

E) Der Schlauch wie auch das Verbindungsstück sind bewegliche, nicht befestigte Objekte.

In der Klassenhierarchie sind so die möglichen Verbindungen zwischen verschiedenen Objekten in Form eines Verbindungsgrafen abgebildet. Dieser Verbindungsgraf ist unabhängig von den konkret eingeladenen VR-Objekten definiert und damit auf verschiedene Anwendungen übertragbar (Abbildung 49). Er wird in den folgenden Abschnitten als „nicht konkretisierter Verbindungsgraf“ bezeichnet.

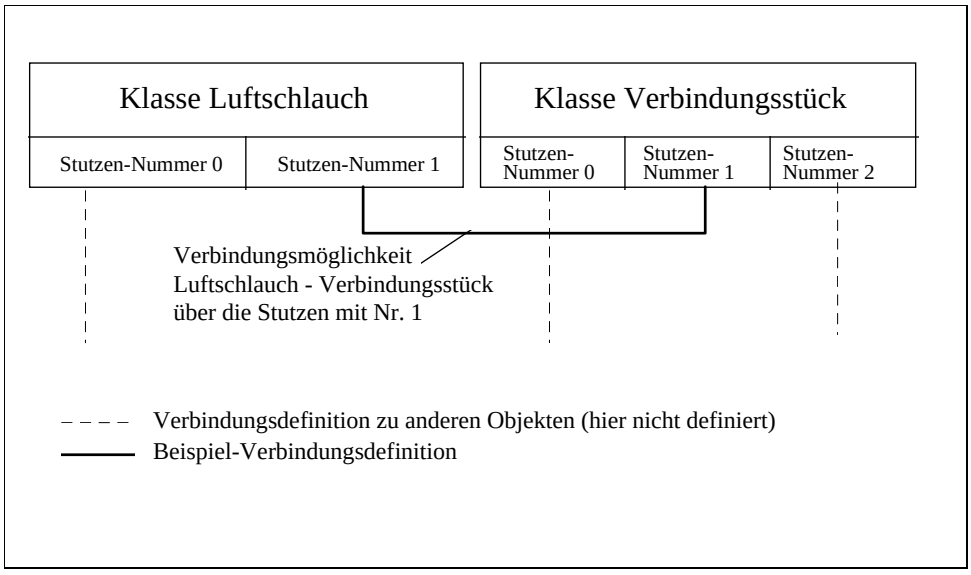


Abbildung 49: Einfacher Beispielverbindungsgraf für die Verbindungsmöglichkeiten von zwei Objekten

4.3.6. Der Klassenverbindungsgraf als Grundlage für die Interaktionssteuerung des VR-Systems

Auf der Basis des Klassenverbindungsgrafen können die in der virtuellen Umgebung eingeladenen geometrischen Objekte in der Objektsemantik korrekt initialisiert werden. Das System kann selbst entscheiden, welche Stutzeninstanz mit welcher anderen Stutzeninstanz eine Verbindung eingehen kann. Nur für diese im Verbindungsgraf abgelegten möglichen Verbindungen werden die entsprechenden Variablen und Routinen initialisiert.

Wie bereits in Abschnitt 4.2.3.4. erläutert, wird eine Verbindungsoperation durch ein externes Ereignis ausgelöst. Dieses externe Ereignis wird durch die Kollisionserkennungs-Simulation im VR-System realisiert. Dieser Kollisionserkennung werden Geometriepaare übergeben und eine zugehörige Kollisions-Callbackroutine, die bei Eintreten der Kollision abgearbeitet wird[93]. Mit dem Wissen über die möglichen Verbindungen zwischen den eingeladenen Objekten kann das System diese Initialisierung selbständig durchführen. Lediglich zwischen jenen Stutzenpaaren wird eine Kollisionsabfrage initialisiert, zwischen denen auch eine Verbindung sinnvoll ist.

Diese Einschränkung der Kollisionspaare auf die Anzahl der aus Anwendungssicht notwendigen Paare bringt mehrere Vorteile mit sich. Zum einen vermindert es wesentlich die Anzahl der Kollisionspaare, die auf gegenseitige Kollision geprüft werden müssen. Das wirkt sich auf die Performance der Kollisionserkennung günstig aus. Weiterhin wird die Erzeugung von fehlerhaften, unsinnigen Verbindungen gemindert. Die folgenden zwei Punkte begründen die Erzeugung solcher Fehlverbindungen.

1. Die Unwissenheit bzw. die Unsicherheit des Anwenders kann dazu führen, daß er nicht mit Sicherheit weiß, welches Objekt mit welchem Objekt eine Verbindung eingehen kann. Dadurch kann es geschehen, daß beispielsweise ein Schlauchobjekt mit einem falschen, dafür nicht vorgesehenen Verbindungsstutzen verbunden wird. Die Integration des Verbindungswissen kann somit den Anwender bei der Interaktion mit der virtuellen Umgebung unterstützen, indem er testen kann, welche Verbindungen wofür vorgesehen sind.

2. Die ungenauen Eingabegeräte des VR-Systems wie auch die hohe Anzahl und damit hohe räumliche Dichte von Verbindungsstutzen kann zu unbeabsichtigten Kollisionen führen. Damit wird der Ablauf zur Verbindungserzeugung automatisch initialisiert, ohne daß der Anwender diese Verbindung aufbauen wollte. Das in der Klassenhierarchie integrierte Verbindungswissen kann demzufolge die Fehlertoleranz des Gesamtsystems erhöhen. Nicht beabsichtigte Verbindungen werden vermieden.

[93] [ZachColl]

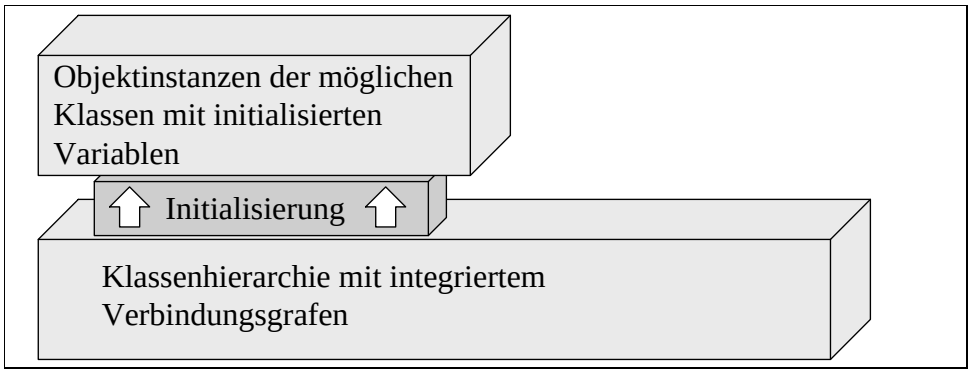


Abbildung 50: Initialisierung der konkreten Objektinstanzen basierend auf den in der Klassenhierarchie abgespeicherten allgemeinen Objektinformationen

4.3.7. Die Integration der CAD-Entities in die Klassenhierarchie

In den bisherigen Betrachtungen wurde auf die besonderen Eigenschaften der CAD-Entities und ihrer Handle in der Objektsemantik noch nicht eingegangen. Die sie repräsentierenden Klassen in der Hierarchie leiten sich von der Oberklasse *Obj_mit_Stutzen* ab. Zwei Besonderheiten ergeben sich aus den realisierten und in Abschnitt 4.1.4. beschriebenen Funktionalitäten. Diese Besonderheiten werden in den folgenden Abschnitten näher erläutert.

4.3.7.1. Stutzenzuordnung

Wie bereits in Abschnitt 4.2.2. erläutert, werden den CAD-Entities automatisch bei ihrer Erzeugung in der VR-Objekthierarchie die Stutzengeometrien zugeordnet. Diese sind in der VR-Objekthierarchie direkt unter den Start- und Endhandles angeordnet (Abbildung 34). Über diese Handle laufen die gesamten Interaktionen mit den Entities ab. Dabei wird in dem VR-System zwischen dem Move- und dem Changemodus unterschieden. Im Movemodus beziehen sich die Interaktionen direkt auf das gesamte Entity, lediglich Transformationen wirken sich auf die Geometrie aus. Im Changemodus hingegen wirken die Interaktionen auf ausschließlich ein Handle, die CAD-Strukturen im CAD-Entity werden bei jedem Schleifendurchlauf der VR-Simulationsschleife aktualisiert (Abschnitt 4.1.4.). Diese Besonderheit der CAD-Entities wurde auch in der Klassenhierarchie der Semantik mit berücksichtigt. Bei den Objekten wird zwischen den beiden Modi Changemodus und Movemodus unterschieden. Je nach aktuellem Modus für ein entsprechendes Entity sind die zugehörigen Stutzenobjekte entweder dem Entity-Objekt oder dem Handleobjekt zugeordnet. Diese wechselnde Zuordnung drückt sich in der Objektsemantik bei den Entity-Stutzen in einem wechselnden Vaterobjekt aus. Befindet sich ein Entity im Movemodus, so besitzen die zugehörigen Stutzen als Verweis auf das Vaterobjekt in ihren Slots das Entity selbst. Im Falle des Changemodus existiert dort ein Verweis auf das entsprechende Handle-Objekt. Die Modi Movemodus und Changemodus können für jedes einzelne Entity separat gesetzt werden.

Durch diese Art der Abbildung der Entities und Handles in der Objektsemantik wird die sich im VR-System darstellende Situation direkt abgebildet. Zwar wird zwischen den Handle und den Entities unterschieden, ihnen ist jedoch sowohl

auf der Grafikseite wie auch auf der Semantikseite jeweils nur ein und der selbe Stutzen zugeordnet. Dadurch wird erreicht, daß die in den Stutzeninstanzen der Objektsemantik enthaltenen Informationen nicht redundant abgelegt sind.

4.3.7.2. Verbindungsgraf

Eine weitere Besonderheit der Abbildung der Entities in der Objektsemantik besteht in der Definition des Verbindungsgrafen in der Klassenhierarchie. Im Gegensatz zu den bisher betrachteten Objektklassen, die sich aus der Oberklasse *Obj_mit_Stutzen* ableiten, können die in der virtuellen Umgebung möglichen Verbindungen der Entities nicht wesentlich eingegrenzt und restriktioniert werden. Da die CAD-Entities die Basisprimitive für eine Modellierung in der virtuellen Umgebung darstellen, müssen sie auch mit jedem beliebigen anderen Entity-Stutzen wie auch mit jedem beliebigen anderen Verbindungsstutzen einer anderen Objektklasse in der virtuellen Umgebung eine Verbindung eingehen können. Daraus ergibt sich, daß die sich aus der Einführung des Verbindungsgrafen ergebenden Vorteile für die Bedienung des Systems (siehe Abschnitt 4.3.6.) nicht auf die Objektklassen *Entity* und *Handle* übertragbar sind.

4.3.8. Die Komponenten des Interaktionsmanagers

Im Folgenden wird die Umsetzung des Interaktionsmanagers anhand seiner einzelnen Komponenten erläutert. Auf den in Abbildung 45 dargestellten funktionalen Aufbau des Interaktionsmanagers wird dabei wesentlich detaillierter eingegangen. Die einzelnen Komponenten werden zunächst kurz vorgestellt. Anschließend werden die grundlegenden Konzepte bezüglich der Anwendungssteuerung und der Informationsspeicherung detailliert beschrieben.

Der Interaktionsmanager unterteilt sich in 4 unterschiedliche Komponenten. Die Trennung des Interaktionsmanagers in diese Komponenten ergibt sich aus deren Funktionen und den ihnen zur Verfügung stehenden Informationen.

Das Modul „Kopplung“ stellt die Kommunikationskomponente dar, die Informationen von der VR-Simulationsschleife erhält und auch Informationen an die VR-Simulationsschleife zurückgibt. Diese Informationen werden in Form von Fakten den einzelnen Komponenten zur Verfügung gestellt. Aufgrund dieser Fakten werden bestimmte regelbasierte Abläufe initiiert, die durch die unterschiedlichen Komponenten abgearbeitet werden. Während dieser Abarbeitung können die verschiedenen Komponenten direkt auf die Datenstrukturen des VR-Systems wie auch auf die Datenstrukturen des Interaktionsmanagers zugreifen. Nach Abschluß dieser Abarbeitung wird über das Kopplungsmodul dem VR-System signalisiert, daß die Verarbeitung der Informationen abgeschlossen ist.

Die Komponente „Laden, Initialisieren“ ist für die Erzeugung der eigentlichen semantischen Objekte verantwortlich. Dabei greift das Modul auf zwei Informationsquellen zurück. In einer Datei (externe Informationen) sind die Informationen über verschiedene Objekte aufgelistet, die für die Anwendungsinteraktionen mit den im VR-System abgebildeten speziellen virtuellen Objekten relevant sind. Diese Informationen wurden in Abschnitt 4.3.5. als „*konkrete,*

von der aktuellen Situation abhängige Daten“ bezeichnet. Sie sind direkt von den im VR-System eingeladenen virtuellen Objekten abhängig. Die zweite Informationsquelle für die Komponente „Laden, Initialisieren“ ist das Kopplungsmodul. Im Falle der Erzeugung eines neuen CAD-Entities im VR-System muß auch dieses Objekt eine Repräsentation in der Objektsemantik besitzen. Die Komponente „Laden, Initialisieren“ erzeugt die entsprechenden Instanzen und initialisiert diese. Weiterhin sorgt es für alle sich aus der Initialisierung eines neuen Objektes ergebenden Aktualisierungen der übrigen semantischen Objekte.

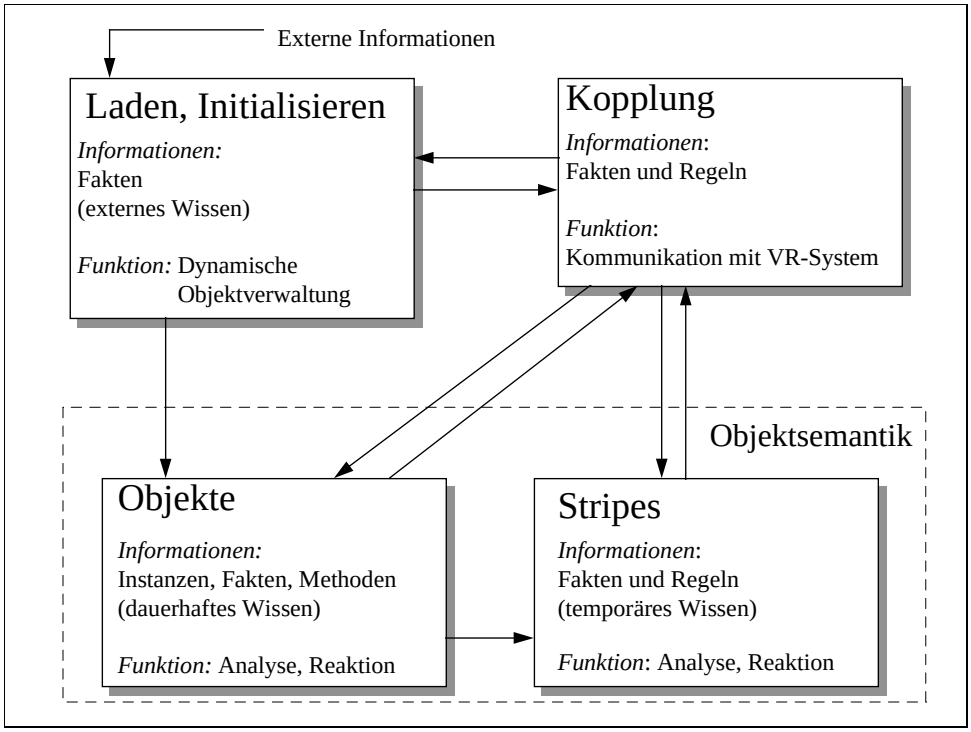


Abbildung 51: Die Komponenten des Interaktionsmanagers

Die Komponente „Objekte“ umfaßt die eigentlichen semantischen Objekte, also die konkreten Instanzen der verschiedenen Objektklassen. Innerhalb dieser Objektklassen sind spezifische Klassenmethoden und allgemeine Funktionen definiert, über die die aktuelle Situation im VR-System analysiert wird, über die diese Situation in der Objektsemantik abgebildet wird und über die auf das VR-System direkt Einfluß genommen wird. Weiterhin wird in dieser Komponente temporäres Wissen in Form von Fakten erzeugt, das von der Komponente „Stripes“ weiterverarbeitet wird.

Die Komponente „Stripes“ bietet eine anwendungsspezifische Sicht auf die verschiedenen semantischen Objekte. Diese Komponente erhält ihre Informationen ausschließlich von der Komponente „Objekte“, die diese in Form von Fakten generiert. Daraus entsteht ein temporäres Faktenwissen, über das auf spezielle Eigenschaften und Zusammenhänge sehr schnell zugegriffen werden kann.

4.3.8.1. Kopplung

Für die Kopplung der VR-Simulationsschleife mit dem Interaktionsmanager stehen zwei grundlegende Realisierungsmöglichkeiten zur Verfügung. Die *erste Möglichkeit* besteht in der direkten Erzeugung von Fakten, durch die im Interaktionsmanager die Verarbeitungsschritte in Form von abzuarbeitenden Regeln aktiviert werden. Dafür ist es notwendig, daß an unterschiedlichsten Stellen im VR-System die unterschiedlichsten, eng an das System CLIPS angegliederten Funktionen integriert werden müssen. Über diese Funktionen müssen Fakten erzeugt und die Inferenzmaschine des Systems angestoßen werden. Daraus ergibt sich eine sehr enge Verzahnung des regelbasierten Interaktionsmanagers mit dem VR-System. Da die Kopplung beider Systeme möglichst lose erfolgen soll, wurde ein *zweiter Weg* eingeschlagen, um die beiden Systeme miteinander zu verbinden. Dabei kommunizieren beide Systeme über ein bidirektionales Messagepassing. Dieses wurde über einen gemeinsamen Speicherbereich implementiert. Über diesen gemeinsamen Speicherbereich können beide Systeme Informationen in Form von Textstrings austauschen. Diese Informationen dienen ausschließlich der Ablaufsteuerung und Synchronisation beider getrennter Systeme.

Der Kern des Kopplungsmodules wird durch eine Regel repräsentiert, die ständig die Schnittstelle zum VR-System nach neuen Informationen abfragt. Aus diesen Informationen werden durch diese Regel Fakten generiert, die den Ablauf des Interaktionsmanagers steuern. Sind keine neuen Informationen verfügbar, wird die Schnittstelle erneut abgefragt. Durch diese Implementierung einer eigenen Loop im Interaktionsmanager („Kopplungsllop“) sind die beiden Systeme voneinander entkoppelt.

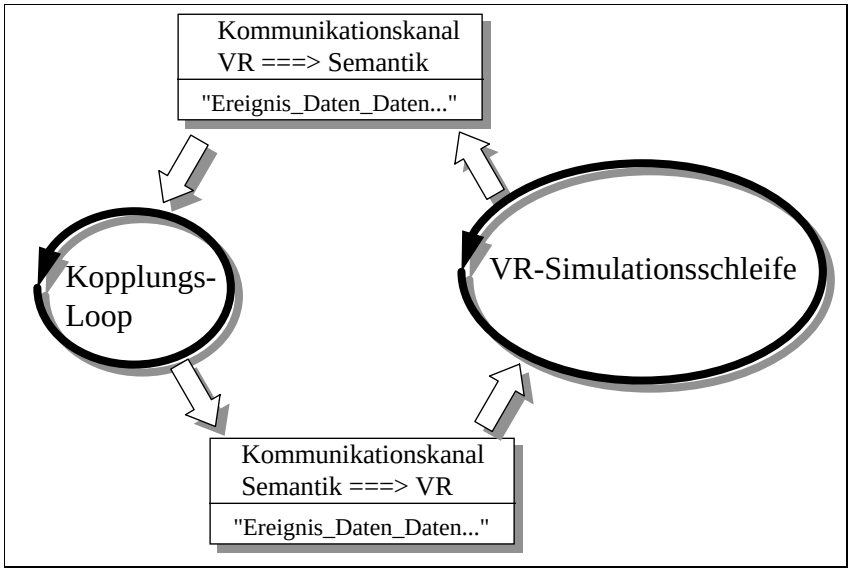


Abbildung 52: Ereignisorientierte Kommunikation zwischen VR-Simulationsschleife und Kopplungsloop über Messagepassing

Für die Aktualisierungsfrequenz der Objekte in der Objektsemantik stehen ebenfalls zwei Alternativen zur Verfügung. *Zum einen* besteht die Möglichkeit der ständigen Aktualisierung bei jedem Schleifendurchlauf des VR-Systems.

Die *zweite Möglichkeit* besteht in einer ereignisorientierten Kopplung beider Systeme. Eine kontinuierliche Aktualisierung der Objekte synchron zur VR-Simulationsschleife erweist sich bei genauer Analyse als unnötig. Das ergibt sich daraus, daß eine Änderung der Informationen in der Objektsemantik nur im Fall einer direkten Manipulation oder einer Systemeingabe notwendig wird, die sich direkt auf ein oder mehrere Objekte im VR-System bezieht. Ein Großteil der Interaktionen des Anwenders innerhalb einer VR-Anwendung hat aber reinen Navigationscharakter. In diesem Fall ist eine Aktualisierung der semantischen Objekte nicht notwendig, da sie keine geometrischen Informationen enthalten. Um die Aktualisierung der Objektsemantik mit maximaler Performance umsetzen zu können, wäre auch bei einer solchen Umsetzung eine Einschränkung des Suchraumes notwendig. Diese Einschränkung ist durch eine gezielte Information darüber möglich, welche Interaktion mit welchen beteiligten Objekten erfolgt ist. Daraus ergibt sich, daß an unterschiedlichen Stellen im VR-System entsprechende Informationen generiert werden müssen.

Diese Überlegungen haben dazu geführt, daß der Interaktionsmanager ereignisorientiert getriggert wird. Bei bestimmten Interaktionsereignissen des VR-Systems werden bestimmte Informationen an die Objektsemantik gesendet (Kommunikationskanal „VR ==> Semantik“ Abbildung 52). Diese Informationen unterteilen sich in zwei Teile. Der erste Teil enthält das Interaktionsereignis, der zweite Teil spezifische Daten, beispielsweise über virtuelle Objekte, die an der Interaktion beteiligt sind oder Werte von Statusvariablen. Durch die ereignisorientierte Steuerung des Interaktionsmanagers und die lose Kopplung beider Systeme wird erreicht, daß die Performance der VR-Simulationsschleife im Falle einer reinen Navigation durch die Systemerweiterung nicht beeinflußt wird.

Die in Tabelle 7 aufgezählten Interaktionsereignisse im VR-System werden dem Interaktionsmanager über das Kopplungsmodul mitgeteilt.

Die aus diesen Ereignissen resultierenden notwendigen Verarbeitungsschritte können in unterschiedlicher Form durch den Interaktionsmanager abgearbeitet werden. Die Abarbeitung kann parallel zum VR-System erfolgen oder mit der Simulationsschleife des VR-Systems synchronisiert werden. Grundsätzlich wird ein Interaktionsereignis, welches dem Interaktionsmanager mitgeteilt wird, durch diesen bestätigt. Die Bestätigung erfolgt über den Kommunikationskanal „Semantik ==> VR“ (Abbildung 52). Sendet das VR-System dem Interaktionsmanager eine Information über ein Interaktionsereignis, wartet es bis zu der entsprechenden Bestätigung. Eine nebenläufige Abarbeitung wird erreicht, wenn die Loop des Moduls Kopplung sofort nach Erhalt einer Ereignisinformation die entsprechende Bestätigung sendet und erst anschließend die notwendigen Verarbeitungsschritte initialisiert.

Die Synchronisation zwischen dem Interaktionsmanager und der VR-Simulationsschleife ist bei bestimmten Interaktionsereignissen unabdingbar. Die Notwendigkeit der Synchronisation ergibt sich aus der Tatsache, daß der Interaktionsmanager im Verlauf seiner Verarbeitungsschritte auch auf die geometrischen Informationen des VR-Systems zugreift und auch verschiedenste Simulationsmodule mit Daten und Informationen versorgt. Während dieser Zu-

griffe können temporäre Inkonsistenzen der VR-Datenbasis entstehen, die zu undefinierten Zuständen führen. In diesen Fällen ist eine Synchronisation beider Prozesse Voraussetzung für einen korrekten Ablauf. Ausschließlich eine konsistente Datenbasis darf dem Anwender durch den Renderingprozeß des VR-Systems präsentiert werden.

Ereignis	Daten	Beschreibung
GRABBED	*Objektname*	Objekt *Objektname* wurde gegriffen
RELEASE	*Objektname*	Objekt *Objektname* wurde losgelassen
COLLISION	*Objektname1 Objektname2*	Eine Kollision zwischen *Objektname1* und *Objektname2* ist eingetreten
SPLIT		Verbindungstrennung wird eingeleitet
SPLITFROM	*Objektname*	Verbindungstrennung von *Objektname* wird eingeleitet
ENTITYNEW	*EntityName StartStutzenName EndStutzenName EntityTyp*	Erzeugung eines neuen Entities mit dem Namen *EntityName* mit den zugehörigen Stutzen *StartStutzenName* und *EndStutzenName* vom Typ *EntityTyp*
REMOVE_ENTITY	*EntityName*	Das Entity *EntityName* wird gelöscht
SETMOVEALL		Das System wird in den Movemodus gesetzt
SETCHANGEALL		Das System wird in den Changemodus gesetzt
SET_ALL_ENTITIES_ RADIUS_CB		Eine Kollisioncallback wird gesetzt um den Radius der Hülle eines Entities setzen zu können
UNSET_ALL_ ENTITIES_RADIUS_CB		Obige Kollisionscallback wird wieder gelöscht
COLHULL	*EntityName*	Der Radius der Hülle des Entities *EntityName* wird gesetzt
COLHULL_END	*Radius EntityNamenFeld*	Der Radius der Entities im *EntityNamenFeld* wird auf den Wert *Radius* gesetzt
SCHRUMPF_ COLLHULL	*EntityName Radius*	Das Entity *EntityName* hat einen Radius für die Hülle vom Wert *Radius*
DebugMode	*Mode (ON OFF)*	Der Debugmodus wird ein- bzw. ausgeschaltet, je nach Wert *Mode*
STOP		Die Kopplungsloop des Interaktionsmanagers wird abgeschaltet

Tabelle 7: Interaktionsereignisse die dem Interaktionsmanager mitgeteilt werden

Eine Parallelisierung der Abarbeitung innerhalb des Interaktionsmanagers wurde nicht realisiert. Das heißt, die Verarbeitungsschritte des Interaktionsmanagers erfolgen alle sequentiell innerhalb der Kopplungsloop. Daraus resultiert eine ungleichmäßige Abarbeitungszeit dieser Loop. Durch die lose Kopplung zwischen Interaktionsmanager und VR-Simulationsschleife und die geringe Häufigkeit von Interaktionsereignissen bezüglich der Anzahl der VR-Simulationsschleifendurchläufe wirkt sich dies aber nicht nachteilig auf die Performance des Gesamtsystems aus.

4.3.8.2. Laden, Initialisieren

Zu Beginn einer Anwendung ist die Initialisierung der semantischen Objekte notwendig. Die verfügbaren Objektklassen werden zuvor in das System eingeladen. Über eine Initialisierungsdatei wird dem System bekanntgegeben, für welche VR-Objekte ein semantisches Objekt erzeugt werden soll. Weiterhin ist die Zuordnung der Stutzen zu den Vaterobjekten darin verschlüsselt. Diese Informationen werden in Form einer Liste vorher erstellt. Diese Liste wird eingeladen und durch die Komponente „Laden, Initialisieren“ ausgewertet.

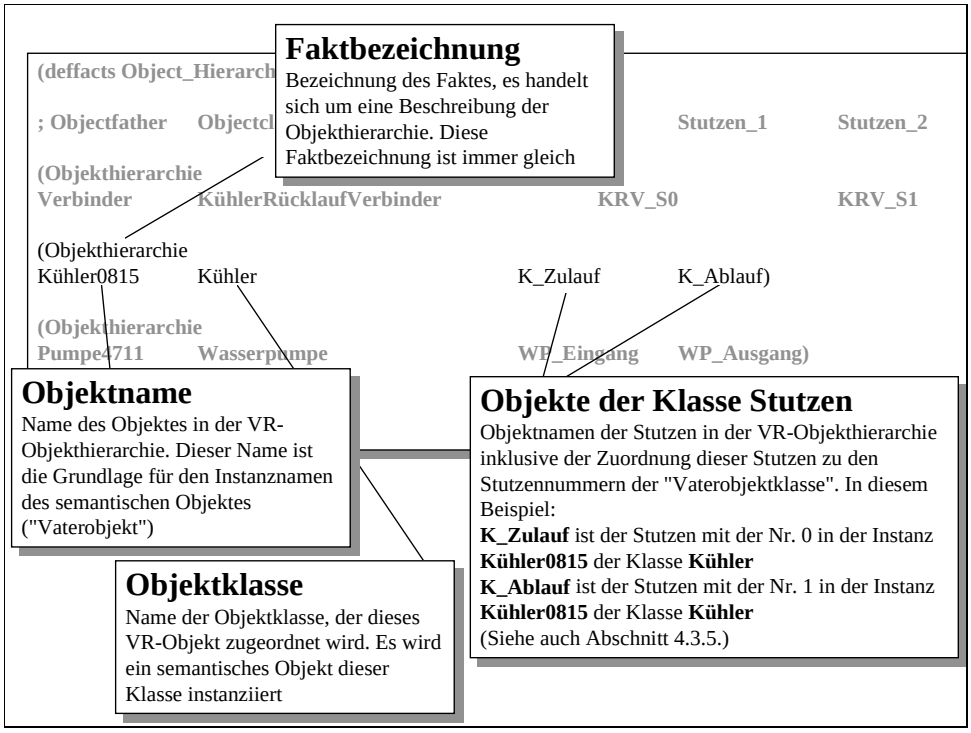


Abbildung 53: Objektliste als Basis für die Initialisierungskomponente

In Abbildung 53 ist eine solche Liste abgebildet und erläutert. Mit dieser Liste wird der in der Objekthierarchie allgemeingültig definierte Verbindungsgraf für einen speziellen Anwendungsfall konkretisiert (siehe auch Abschnitt 4.3.5.). Es erfolgt die Zuordnung der VR-Objekte zu den semantischen Objektklassen. Das gilt sowohl für das „Vaterobjekt“ wie auch für die dazugehörigen Stutzen. Weiterhin werden die Stutzenobjekte ihrem Vaterobjekt zugeordnet. Damit wird die Objekthierarchie eines „Objektes mit Stutzen“ definiert. Diese Objekthierarchie definiert die hierarchische Anordnung der VR-Objekte in der VR-Objekthierarchie. Diese Zuordnung hat einen statischen Charakter, das heißt, sie verändert sich im Laufe einer Anwendung nicht. Über die einfache hierarchische Anordnung der Stutzen zu den Vaterobjekten hinaus werden die Stutzenobjekte den Stutzennummern des Verbindungsgrafen zugeordnet. Wie in Abschnitt 4.3.5. beschrieben, wird der Verbindungsgraf über geordnete Stutzenfelder definiert. Die Zuordnung der konkretisierten Stutzeninstanzen zu diesen nummerierten logischen Stutzen erfolgt über die Reihenfolge der Stutzennamen in der Liste. Dabei entspricht dem ersten Stutzen der logische Stutzen 0 und die folgenden Stutzen werden jeweils um den Wert 1 hochgezählt.

Die Initialisierung der semantischen Objekte zur Startzeit erfolgt in zwei Teilschritten. *Im ersten Schritt* werden die Objektinstanzen der jeweiligen in der Liste angegebenen Klasse erzeugt. Dafür wird die im VR-System eingeladene Szene nach den in der Liste angegebenen Objektnamen durchsucht. Nur diejenigen semantischen Objekte werden erzeugt, für die auch eine entsprechende grafische Repräsentation im VR-System existiert. Das Ergebnis dieses ersten Schrittes ist eine Menge semantischer Objekte, deren Slots noch nicht initialisiert sind.

Im *zweiten Teilschritt* werden die Slots der semantischen Objekte mit sinnvollen Werten belegt. Die hierarchische Zuordnung der Stutzen zu ihren Vaterobjekten erfolgt ebenso wie die konkrete Initialisierung des Verbindungsgrafen zwischen den Stutzen. Weiterhin wird der Verbindungszustand der einzelnen Stutzenverbindungen getestet und gesetzt. Für die genannten Initialisierungen greift der Interaktionsmanager auf die grafischen Informationen des VR-Systems zurück. Die statische Zuordnung der Stutzen zu ihren Vaterobjekten wird in der VR-Objekthierarchhie ebenso erzeugt wie in der Objektsemantik. Um den Verbindungszustand zu setzen, greift der Interaktionsmanager auf die grafische Beschreibung der Stutzen in der VR-Objekthierarchie zurück. Das Ergebnis eines Koordinatenvergleiches stellt die Basis für die Initialisierung dieser Slots dar (siehe auch Abschnitt 4.2.3.1.).

Die Initialisierung der CAD-Entities unterscheidet sich in geringem Maße von der Initialisierung der übrigen Objekte. Das ist in deren automatischer Generierung zur Laufzeit begründet. Die hierarchische Zuordnung der Stutzen zum Vaterobjekt wird durch das VR-System selbst erledigt. Weiterhin können die Entity-Stutzen mit jedem beliebigen anderen Stutzen in der virtuellen Szene eine Verbindung eingehen. Die Verbindungsmöglichkeiten werden also nicht durch einen Verbindungsgrafen eingeschränkt.

Aus der automatischen hierarchischen Zuordnung der Stutzen zu den „Vaterobjekten“ sowohl in der Objektsemantik wie auch in der VR-Objekthierarchie auf der Basis der in Abbildung 53 dargestellten Objektliste ergeben sich mehrere Vorteile. Lediglich eine Hierarchiedefinition durch den Anwender ist erforderlich. Er benötigt keine speziellen Kenntnisse über die hierarchische Abbildung der Objekte im VR-System und deren Manipulation. Ein zusätzlicher Hierarchieeditor für die grafischen Objekte ist nicht erforderlich. Inkonsistenzen zwischen VR-Datenbasis und Objektsemantik werden ausgeschlossen.

Durch die Abfrage der in das VR-System eingeladenen grafischen Objekte werden nur die semantischen Objekte erzeugt, die auch im VR-System existieren. Eine Veränderung der VR-Datenbasis (beispielsweise eine Darstellung einer Teilmenge einer Szene) erzwingt damit keine Anpassung der Objektliste der Objektsemantik. Das System initialisiert nur im VR-System eingeladene und damit existierende VR-Objekte.

Die automatische Generierung des Verbindungsgrafen in der Objektsemantik und die damit verbundene automatische Analyse und Initialisierung der Verbindungszustände zwischen den Stutzen ermöglicht den Verzicht auf eine Abspeicherung dieser Informationen. Damit müssen diese semantischen

Informationen auch nicht gepflegt und verwaltet werden. Weiterhin ist die Initialisierung verschiedener Kollisions-Callbackroutinen im Rahmen der Anwendungsvorbereitung nicht notwendig, da sich deren Initialisierung aus den Verbindungszuständen der Stutzenverbindungen ergeben.

Damit ist der Anwendungsvorbereitungsaufwand für das System auf ein Minimum reduziert.

4.3.8.3. Objekte

Die Objekt-Komponente des Interaktionsmanagers ist durch die Menge der initialisierten Objektinstanzen, deren Slots, Methoden und die darauf abarbeitbaren Funktionen definiert. Die semantischen Objektinstanzen sind den grafischen Objekten im VR-System eineindeutig zugeordnet. Diese Zuordnung erfolgt über die Objektnamen. Die semantische Repräsentation eines virtuellen Objektes hat den gleichen Namen wie die grafische Repräsentation in der

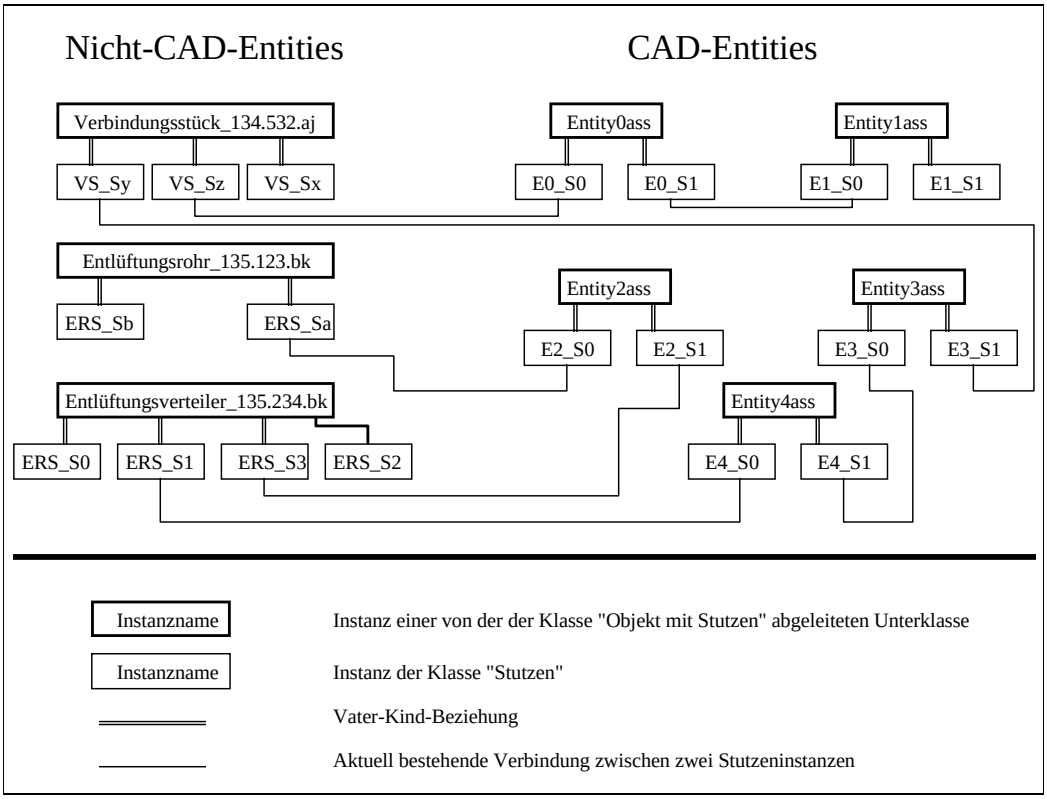


Abbildung 54: Schema eines konkretisierten Verbindungsgrafen in der Objektsemantik

VR-Objekthierarchie. Die zusätzliche Abspeicherung der Objektadresse (Pointer) eines Objektes in der VR-Objekthierarchie als Slot in der Objektsemantik hat sich als vorteilhaft erwiesen. Wird vom Interaktionsmanager relativ oft auf ein grafisches Objekt zugegriffen, dann muß dieses Objekt nicht erst in der VR-Objekthierarchie gesucht werden (Durchsuchen der Baumstruktur der VR-Objekthierarchie und Vergleich der Objektnamen), sondern kann direkt über die Adresse angesprochen werden. Diese Möglichkeit wurde speziell für die Stutzen implementiert (siehe Tabelle 5: Slot „Stutzen_MLine_Pointer“), da diese relativ häufig auf ihren aktuellen Verbindungsstatus abgefragt werden.

Dazu wird auf die geometrischen Informationen des Stutzens (Punktkoordinaten des Stutzenvektors) in der VR-Objekthierarchie zugegriffen.

Die Objekt-Komponente stellt alle Methoden und Funktionen zur Verfügung, die durch die Regeln der Komponente Kopplung aufgerufen und abgearbeitet werden. Diese Methoden und Funktionen dienen der Analyse und Aktualisierung der einzelnen Slots der semantischen Objekte. Durch diese semantischen Objektdaten werden zwei Hauptinformationen über die virtuelle Szene abgebildet. Dabei handelt es sich zum einen um die Zuordnung der Stutzen zu den entsprechenden Vaterobjekten. Weiterhin wird der Verbindungsgraf zwischen den Objekten über die Slots der Stutzenobjekte abgebildet. Es handelt sich hierbei um den konkretisierten, von der jeweiligen aktuellen Situation abhängigen Verbindungsgrafen. In Abbildung 54 ist ein kleiner Ausschnitt eines möglichen Verbindungsgrafen schematisch dargestellt. Auf die Darstellung der einzelnen Slots der Instanzen und deren Inhalte wurde aus Gründen der Übersichtlichkeit verzichtet. Für die Bearbeitung der in den Slots der Objektinstanzen enthaltenen Informationen wie auch für die daraus resultierenden Konsequenzen bezüglich des Gesamtverhaltens des Systems ist es eine Voraussetzung, daß die Methoden und Funktionen der Objekt-Komponente des Interaktionsmanagers auf die Komponenten des VR-Systems zugreifen. Sie können die VR-Objekthierarchie wie auch die Simulationskomponenten des VR-Systems manipulieren und somit beeinflussen. Weiterhin greifen sie zu Analysezwecken auf die geometrischen Informationen der Objekte in der VR-Objekthierarchie zu. Auch die Generierung temporären Faktenwissens wird durch die Objekt-Komponente durchgeführt.

4.3.8.4. Stripes

Mit Hilfe der Komponente „Stripes“ wurde eine anwendungsspezifische Sicht auf die Instanzen der Objekt-Komponente realisiert. Es handelt sich um Faktenwissen, das die Verbindungen repräsentiert, die die einzelnen CAD-Entities über ihre Stutzen eingegangen sind. Die hier beschriebenen Stripes sind nicht mit den geometrischen, auf Dreiecksdaten basierenden Tristripes zu verwechseln, wie sie beispielsweise im OpenGL-Standard definiert sind[94]. Die in diesem Abschnitt beschriebenen Stripes haben mit dieser speziellen Dreiecksrepräsentation lediglich den Namen gemein.

Die im Interaktionsmanager eingeführten Stripe-Fakten haben alle den gleichen Aufbau. Sie beginnen mit einem Typ-Token, der den Stripetyp spezifiziert. Nach diesem Typ-Token folgen verschiedene Namen von Instanzen, die in der Objekt-Komponente enthalten sind.

Es wird zwischen drei verschiedenen Stripetypen unterschieden. Dabei handelt es sich um den Freestripe, den Leftstripe und den Left-Right-Stripe (kurz LRStripe).

[94] [WooNeiDa] S. 42 ff.

(Stripe-Typ OMS SI SI OMS [SI SI OMS [...]])

OMS	Instanzname eines Objektes einer Klasse, die sich von der Klasse "Objekt mit Stutzen" ableitet
SI	Instanzname eines Objektes der Klasse Stutzen

Abbildung 55: Aufbau eines Stripefakts

Der *Freestripe* repräsentiert diejenigen miteinander verbundenen CAD-Entities, die keine Verbindungen zu Objektinstanzen einer Nicht-Entity-Klasse eingegangen sind. Es handelt sich also um zusammengesetzte Objekte, die ausschließlich aus CAD-Entities bestehen.

Der *LeftStripe* repräsentiert diejenigen miteinander verbundenen Entities, die an genau einem Stripeende eine Verbindung zu einer Objektinstanz einer Nicht-Entity-Klasse eingegangen sind. Es handelt sich also um zusammengesetzte Objekte, die aus CAD-Entities und genau einer Nicht-CAD-Entity-Objektinstanz bestehen. Diese Nicht-CAD-Entity-Objektinstanz bildet ein Ende des Stripes.

Der *LRStripe* repräsentiert diejenigen miteinander verbundenen Entities, die an beiden Stripeenden eine Verbindung zu einer Objektinstanz einer Nicht-Entity-Klasse eingegangen sind. Die miteinander verbundenen CAD-Entities sind also von zwei Nicht-CAD-Entity-Objektinstanzen eingeschlossen.

Die Bildung der Stripes erfolgt zur Laufzeit. Diese Fakten werden durch ein Regelwerk erzeugt, das wiederum durch temporäre Fakten gesteuert wird. Diese temporären Fakten werden in der Objekt-Komponente erzeugt. Dabei handelt es sich um Fakten, die den Aufbau bzw. die Trennung einer Verbindung dokumentieren. Sie enthalten die Namen des an der Verbindung/Trennung beteiligten Stutzenpaares. Aus diesen temporären Fakten und den Stripefakten werden neue Stripes erzeugt, nicht mehr vorhandene Stripes gelöscht beziehungsweise bestehende Stripes aufgetrennt. Durch diesen Mechanismus dokumentieren die Stripes die jeweils aktuell aus einzelnen CAD-Entities modellierten Leitungen und Schläuche.

Die Stripe-Fakten enthalten zu der Gesamtheit der semantischen Objekte und deren Slots redundante Informationen über die bestehenden Entityverbindungen. Sie stellen eine anwendungsspezifische Sicht auf diese umfangreichen Informationen dar. Bei den hier beschriebenen Fakten handelt es sich um die Abbildung der Schlauchfragmente und Schläuche, die sich zum jeweiligen Zeitpunkt in der virtuellen Umgebung befinden und sich aus verschiedenen Einzelentities zusammensetzen. Durch die Abbildung dieser Informationen in Form von einfach strukturiertem Faktenwissen können darauf sehr effektiv Anfragen an das regelbasierte System verarbeitet werden. Diese Anfragen müssen somit nicht die Objektinstanzen und den darin abgebildeten Verbindungsgrafen analysieren, sondern können direkt die Inferenzmaschine des regelbasierten Systems CLIPS nutzen. Ein weiterer Vorteil besteht in der Übersichtlichkeit und der einfachen Lesbarkeit dieser Fakten. Dadurch ist eine

programmiertechnische Erweiterung des Systems um leitungsspezifische Funktionalitäten relativ leicht umsetzbar.

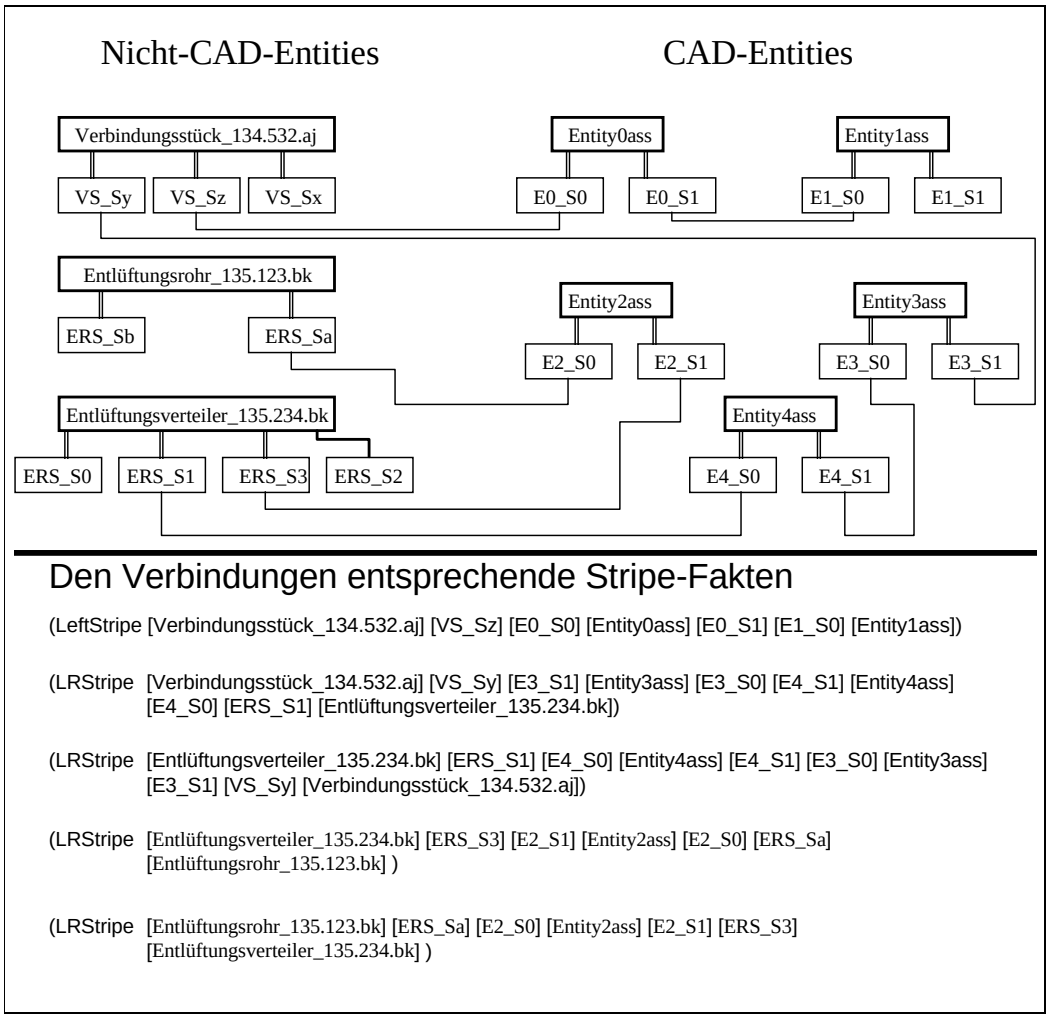


Abbildung 56: Verbindungsgraf aus Abbildung 54 mit entsprechenden Stripe-Fakten

4.3.9. Ergebnisse

Es wurde eine Objektklassenhierarchie mit dem Ziel eingeführt, im System enthaltene *nicht*geometrische (semantische) Objektinformationen zu verwalten. Sie bilden die Grundlage für ein realitätsnahes Interaktionsverhalten der Objekte im System, insbesondere im Falle der Manipulation zusammengesetzter Objektstrukturen. Dazu wurden zunächst sowohl die softwaretechnischen wie auch die funktionalen Anforderungen an diese Klassenhierarchie diskutiert.

Diese Vorüberlegungen führten zu einem Interaktionsmanager, der

- lose mit dem VR-System gekoppelt ist,
- durch das VR-System getriggert (aktiviert) wird,

- auf die semantischen Daten der Objektklassenhierarchie wie auch auf die Objekthierarchie des VR-Systems zugreifen kann.

Die Implementierung erfolgte im System CLIPS, einem regelbasierten System mit objektorientierter Spracherweiterung, welches sich durch seine Systemunabhängigkeit, seine einfache Portierbarkeit und seine einfache Integrationsmöglichkeit mit anderen Systemen auszeichnet.

In CLIPS wurde die Objektklassenhierarchie implementiert, die die Grundlage für die Verwaltung der semantischen Informationen bildet. Mit Hilfe dieser Objektklassen wird ein kleiner Weltausschnitt mit allen Informationen beschrieben, die für die Leitungsverlegung notwendig sind.

Es konnte beim Systemdesign zwischen *allgemeingültigen* und *konkreten, von der aktuellen Situation abhängigen* Daten unterschieden werden. Diese Unterscheidung hat eine wesentliche Bedeutung für den Aufwand bei der Anwendungsvorbereitung und spiegelt sich in den Klassen und Instanzen der Objekthierarchie unterschiedlich wieder. Es wurde dementsprechend ein Klassenverbindungsgraf eingeführt, über den mögliche Verbindungen zwischen Objektklassen allgemeingültig und unabhängig von der geometrischen Repräsentation definiert werden können. Im Ergebnis führt das zu einer Vereinfachung der Anwendungsvorbereitung, woraus sich wiederum eine Fehlervermeidung und eine Performancesteigerung während der Anwendung ergibt.

Die besonderen Eigenschaften der CAD-Entities wurden in einer speziellen Klasse berücksichtigt, so daß ihr Verhalten im System auch korrekt in der Objektklassenhierarchie abgebildet ist.

Der systemtechnische Aufbau des Interaktionsmanagers wurde beschrieben. Er unterteilt sich in die vier verschiedenen Komponenten *Kopplung*, *Laden/Initialisieren*, *Objekte* und *Stripes*, auf deren Aufbau und Funktion näher eingegangen wurde.

4.4. Zusammenfassung

Im 4. Kapitel wurden Konzepte für eine Erweiterung eines bestehenden VR-Systems und deren Umsetzung diskutiert und erläutert, die sich aus den Überlegungen und Diskussionen der vorangegangenen Kapitel ergeben haben.

Ein vorhandenes VR-System wurde um ein neues Datenobjekt, das CAD-Entity, erweitert. Dieses unterscheidet sich von den übrigen virtuellen Objekten in seiner systeminternen Beschreibung. Es ist sowohl mathematisch exakt (CAD-Repräsentation) wie auch in polygonaler Form (VR-Repräsentation) im System abgelegt. Dabei ergibt sich die VR-Repräsentation eindeutig durch eine Tesselierung aus der CAD-Repräsentation.

Die Entscheidung für die enge Anlehnung der Umsetzung an den IGES-Standard ist dadurch begründet, daß es sich um eine systemunabhängige Datenschnittstelle handelt, die im Bereich der Automobilindustrie weit verbreitet ist und durch die meisten CAx-Systeme unterstützt wird. Es wurden im

Rahmen einer Diplomarbeit[95] Generierungs-, Manipulations- und Systemkonfigurationsfunktionen entwickelt und in das eingesetzte VR-System Virtual Design II integriert. Weiterhin wurden Prozessoren zum Datenaustausch (IGES-Import und -Export) implementiert, über die eine Ergebnisrückführung zu einer vorhandenen CAx-Umgebung ermöglicht wird.

Die CAD-Entities, insbesondere die Linienobjekte, bilden die Grundlage für die datentechnische Repräsentation der Leitungsobjekte in der virtuellen Umgebung. Sie werden für die Darstellung der Mittellinie eines Leitungsobjektes genutzt. Sie können über die direkte Manipulation in ihrer Position und in ihrem Verlauf in der virtuellen Umgebung bearbeitet werden. Dazu wurde eine bidirektionale Konvertierung zwischen CAD- und VR-Repräsentation implementiert.

Die Arbeit mit den beschriebenen Linienobjekten hat gezeigt, daß die notwendigen Tesselierungsoperationen den Echtzeitanforderungen des Gesamtsystems genügen.

Die ausschließliche Beschreibung der Schlauchobjekte durch ihre Mittellinie hat sich als unzureichend erwiesen. Daher wurden die CAD-Entities um eine einfache, polygonbasierte Hüllgeometrie erweitert. Der sich daraus ergebende erhöhte Berechnungsaufwand genügt ebenfalls den Echtzeitanforderungen des Gesamtsystems. Die erhöhten Anforderungen an die Renderingperformance konnten durch den Einsatz von Levels of Detail minimal gehalten werden. Durch den Einsatz der Hüllgeometrie stellen sich die Leitungsobjekte realitätsnäher dar. Weiterhin können die Hüllgeometrien zur Kollisionserkennung genutzt werden. Eine Durchdringung einer Leitung eines bestimmten Durchmessers mit einem beliebigen anderen Objekt der virtuellen Umgebung kann sehr schnell durch Sichtkontrolle erkannt werden. Die Ermittlung des freien Bauraumes konnte durch die Hüllgeometrien abgesichert werden.

Das Arbeiten auf einzelnen CAD-Entities hat gezeigt, daß die Berücksichtigung von Verbindungen zwischen den einzelnen Entities wie auch zwischen den Entities und den Objekten der virtuellen Umgebung notwendig ist. Es konnte weiterhin gezeigt werden, daß eine Berücksichtigung von Verbindungen nur dann möglich ist, wenn das System sowohl diese Verbindungen selbst erkennen kann als auch deren Aufbau und Trennung aktiv unterstützt.

Die in Kapitel 3 Abschnitt 3.5. durchgeführten Analysen und Alternativendiskussionen führten zu einer Entscheidung für den Einsatz von Ersatzmodellen bei der direkten Manipulation der Objekte. Die gleichen Überlegungen waren maßgebend für die Entscheidung, eine Hilfsstruktur in das System zu integrieren, die die Grundlage für die Verbindungsoperationen bildet.

Bei den daraus resultierend eingeführten Stutzen wird zwischen einem geometrischen und einem semantischen Modell unterschieden. Die Stutzen haben einen sehr einfachen Aufbau, der in den Abschnitten 4.2.1 und 4.2.2. näher erläutert wurde.

[95] [RAB96]

Es konnte gezeigt werden, daß mit Hilfe dieser einfachen Stutzen Verbindungsabläufe simuliert werden können, die der Realität nachempfunden sind (Abschnitt 4.2.3.). Dazu wurde ein Verbindungsmodell mit 3 verschiedenen Zuständen entwickelt. Der *Verbundene Zustand*, *nicht verbundene Zustand* und ein *Zwischenzustand* wurden eingeführt. Über den Zwischenzustand ist es dem Anwender möglich, direkt auf die zu erstellende oder die zu trennende Verbindung Einfluß zu nehmen. Diese Möglichkeit der Einflußnahme ist zwingend notwendig, da die in Abschnitt 4.2.3.1. eingeführte Definition einer Verbindung zwischen Leitungsobjekten unterbestimmt ist (siehe auch Abschnitt 4.2.3.2.).

Basierend auf den Stutzengeometrien wurden sechs verschiedene Bewegungsbeschränkungen eingeführt die beliebig miteinander kombiniert werden können. Sie bilden die Grundlage für die verschiedenen Verbindungsabläufe, die über Parameterwerte im System definiert werden. Mit Hilfe einer extra zu diesem Zweck entwickelten Testumgebung können die Parameterwerte für verschiedene Verbindungsabläufe ermittelt werden. Damit konnte gezeigt werden, daß basierend auf den Stutzen über die Bewegungsbeschränkungen Verbindungsabläufe wie eine Steckverbindung oder eine Schraubverbindung realisiert werden können.

Das eingeführte stutzenbasierte Verbindungsmodell hat mehrere Vorzüge:

1. Das System kann jederzeit selbständig und eindeutig entscheiden, ob eine Verbindung zwischen zwei Stutzen besteht oder nicht.
2. Durch die eingeführten frei kombinierbaren Bewegungsbeschränkungen unterstützt das System den Anwender beim Aufbau und der Trennung einer Verbindung. Damit arbeitet diese Verbindungssimulation weitgehend unabhängig von der Genauigkeit der Eingabegeräte und damit unabhängig vom eingesetzten Eingabegerät.
3. Auf den Einsatz von Kraftrückkopplungsgeräten kann verzichtet werden.
4. Durch die direkte Abfrage der Stutzen in der virtuellen Umgebung können beide an einem Verbindungsaufbau beteiligten Stutzen sich bewegen. Das beidhändige Arbeiten wird dadurch ermöglicht.
5. Der Verzicht auf eine physikalische Simulation ermöglicht das Arbeiten mit unvollständigen geometrischen und physikalischen Modellen und vereinfacht so die Datenaufbereitung.
6. Der Berechnungsaufwand für die Bewegungsbeschränkungen ist minimal. Er wirkt sich auf die Performance des Gesamtsystem nicht negativ aus.

Der Einsatz der Stutzen und der damit verbundenen parametrisierten Verbindungsabläufe erfordert zusätzliche, teilweise geometrieunabhängige Informationen, die erzeugt und im System verwaltet werden müssen. Es handelt sich um einen Übergang von rein geometrischen Objekten zu Objekten mit unterschiedlichen semantischen Eigenschaften. Eine Verwaltung und Berücksichtigung dieser Eigenschaften, insbesondere der Verbindungsrelationen zwischen den Objekten während der Interaktionen, ist in dem eingesetzten VR-System

Virtual Design II nicht vorgesehen. Daraus ergab sich die Notwendigkeit der Erweiterung dieses Systems um eine Komponente, die diesen Anforderungen genügt.

Es wurde ein Interaktionsmanager implementiert, dem die notwendigen Zusatzinformationen über die Objekte in Form einer Objektklassenhierarchie zur Verfügung stehen (siehe Abschnitt 4.3.). Die Implementierung erfolgte im System CLIPS, einem regelbasierten System mit einer objektorientierten Erweiterung.

Der Interaktionsmanager besteht aus vier verschiedenen Komponenten. Dabei handelt es sich um die Module *Kopplung*, *Laden/Initialisieren*, *Objekte* und *Stripes*. Der Interaktionsmanager ist mit dem VR-System lose gekoppelt, wird durch das VR-System getriggert und hat sowohl Zugriff auf die semantischen Zusatzinformationen einer Objektklassenhierarchie wie auch auf die geometrischen Informationen des VR-Systems.

Die Informationsverarbeitung innerhalb des Interaktionsmanagers ist vergleichbar mit dem Ablauf einer Informationsverarbeitung des Anwenders. Es wird zwischen der Evaluierungsphase und der Executionphase unterschieden (Abschnitt 4.3.2.2.).

Die Arbeit mit dem System hat gezeigt, daß bei den verwalteten Informationen zwischen *allgemeingültigen* und *konkreten, von der aktuellen Situation abhängigen* Daten unterschieden werden kann (Abschnitt 4.3.5.). Diese Unterscheidung macht es möglich, unter Ausnutzung der Objektklassenhierarchie den Aufwand für die Anwendungsaufbereitung minimal zu halten.

5. Direkte Manipulation unter Berücksichtigung bestehender Verbindungen

Im 4. Kapitel wurden die Basiserweiterungen für ein VR-System diskutiert, die für die Leitungsverlegung in einer virtuellen Umgebung notwendig sind. Es handelt sich um die Einführung von CAD-Entities für die Repräsentation der Mittellinie eines Leitungsobjektes. Weiterhin wurden diese Objekte um eine Hüllgeometrie erweitert. Die Einführung von Stutzen stellt die datentechnische Basis für die Erzeugung, Manipulation und Analyse von Verbindungen zwischen verschiedenen Objekten dar. Dafür sind zusätzliche semantische Informationen notwendig, die in einem Interaktionsmanager mit einer Objektklassenhierarchie verwaltet werden können. Auf der Grundlage dieser Erweiterungen wurden in dem VR-System Virtual Design II Interaktionsabläufe dahingehend erweitert, daß sie bestehende Verbindungen mit berücksichtigen. Die Umsetzung dieser Erweiterungen wird im 5. Kapitel diskutiert.

Es wird auf die Interaktionsabläufe eingegangen, bei denen die semantischen Informationen inklusive des Objekt-Verbindungsgrafen eine Rolle spielen. Dabei handelt es sich sowohl um die Operationen „Greifen“ und „Loslassen“ eines Objektes (Abschnitt 5.1.), wie auch um die Operationen zur Veränderung des Hüllgeometrieradius der Entities (Abschnitt 5.2.). In beiden Fällen sollen bestehende Verbindungen zwischen den virtuellen Objekten Berücksichtigung finden um so die Anzahl der notwendigen Befehlseingaben minimieren zu können. Eine große Rolle spielt hierbei der Klassenverbindungsgraf, wie er in den Abschnitten 4.3.5. und 4.3.6. beschrieben wurde. Die Definition dieses Verbindungsgrafen stellt einen Anwendungsvorbereitungsaufwand dar, der zu Akzeptanzproblemen führen kann. Daher werden die Anwendungsmöglichkeiten des Systems in Abhängigkeit von der Vollständigkeit des definierten Verbindungsgrafen im Abschnitt 5.3. diskutiert.

5.1. Das Greifen und Loslassen eines virtuellen Objektes

Wie bereits in Abschnitt 3.2.4. beschrieben wurde, können in einer virtuellen Umgebung verschiedene ereignisgesteuerte Interaktionen definiert und durchgeführt werden. Die für die virtuelle Leitungsverlegung wichtigste Interaktion ist der in Abschnitt 3.5.1. näher erläuterte Greifvorgang. Er stellt die Basisinteraktion für die direkte Manipulation der virtuellen Objekte dar. Bereits in Abschnitt 3.5.4. wurde auf die Notwendigkeit hingewiesen, daß das System den Anwender bei der direkten Manipulation unterstützt. Der dabei erwähnte Verbindungsmanager muß demnach in der Lage sein, selbständig Verbindungen zu erkennen, deren Aufbau und Trennung zu unterstützen und bestehende Verbindungen im Falle der verschiedenen Interaktionen zu berücksichtigen.

In diesem Abschnitt wird zunächst das angestrebte Systemverhalten beschrieben. Anschließend werden die möglichen sich unterscheidenden Lösungsansätze diskutiert. Der realisierte Interaktionsablauf wird mit seinen Vor- und Nachteilen beschrieben.

5.1.1. Angestrebtes Systemverhalten

Das angestrebte Systemverhalten läßt sich sehr kurz zusammenfassen:

Bei den Greif- und Positionierungsinteraktionen werden die zwischen den einzelnen virtuellen Objekten bestehenden Verbindungen mit berücksichtigt. Damit wird verhindert, daß eine bestehende Verbindung zwischen zwei Objekten ungewollt zerstört wird, wenn eines der verbundenen Objekte neu gegriffen und positioniert wird.

Dieses Verhalten des Systems führt dazu, daß die Anzahl der notwendigen einzelnen Interaktionen auf ein Minimum reduziert wird. Weiterhin nähert sich das Systemverhalten der Erwartungshaltung des Anwenders an, wodurch eine Akzeptanzsteigerung erreicht wird.

Mit dem angestrebten Systemverhalten sind mehrere Spezialfälle verbunden, deren Verhalten festgelegt und umgesetzt wird.

5.1.2. Diskussion möglicher Lösungsansätze

Für die Realisierung einer Verbindungsberücksichtigung besteht die Möglichkeit der Zusammenfassung einzelner miteinander verbundener virtueller Objekte zu einem Aggregat, wie das im Rahmen des CODY-Projektes[96] realisiert

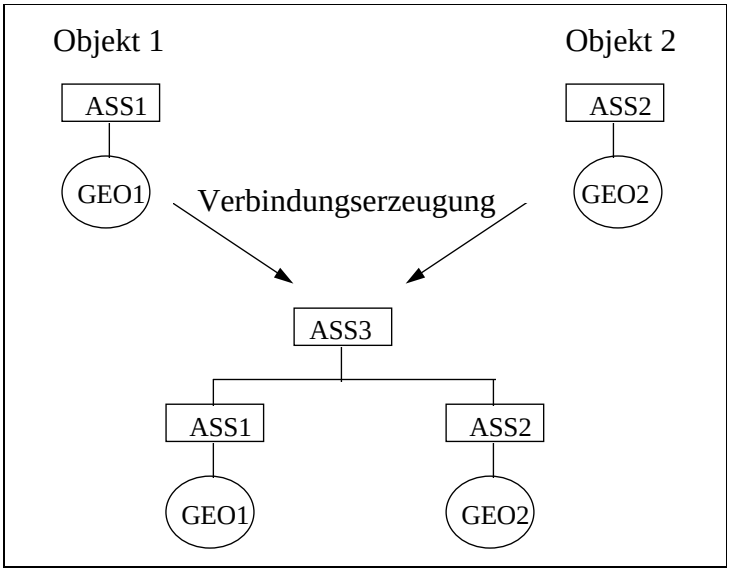


Abbildung 57: Zusammenfassung einer Verbindung in der VR-Objekthierarchie

wurde. Bei der Umsetzung dieses *Aggregatekonzeptes* werden die zusammengefügten Einzelobjekte in Form von Listen verwaltet. Diese Aggregate können im Rahmen einer Interaktion gemeinsam selektiert, positioniert, gefügt und aufgetrennt werden. Eine direkte geometrische Abbildung solcher zusammengesetzter Objekte in einem VR-System ist über den Szenengrafen realisierbar. Alle miteinander verbundenen Objekte können beispielsweise über einen ASSEMBLY-Knoten (siehe auch Abschnitt 3.2.3.) zusammengefaßt und

[96] [JUNG96]

so gemeinsam positioniert werden. Da sich der Status des Verbindungszustandes zwischen den einzelnen Objekten im Verlauf einer Anwendung ändern kann, muß auch die Zusammenfassung der Objekte über die ASSEMBLY-Knoten dynamisch angepaßt werden. In Abbildung 57 ist die Änderung der Objekthierarchie für zwei einzelne Objekte dargestellt, die miteinander verbunden werden. In diesem Beispiel entsteht ein neues ASSEMBLY-Objekt (ASS3), das beide Einzelobjekte (ASS1, ASS2) zusammenfaßt. Dieses neue Objekt muß im Fall einer Positionierung eines der beiden Teilobjekte Ergebnis der Selektion sein, wodurch die Verbindung der beiden Einzelobjekte auch während dieser Interaktion beibehalten bleibt. Ihre relative Position zueinander bleibt somit unverändert.

Die beschriebene Abbildung der Verbindungen durch Zusammenfassung der Objekte zu Aggregaten bzw. die Umsetzung dieses Konzeptes direkt in der Objekthierarchie des VR-Systems hat den Vorteil, daß die Verbindungen statisch in der Objekthierarchie abgebildet sind und erst dann wieder aktualisiert werden müssen, wenn eine Verbindung aufgetrennt wird oder neu entsteht. Ein Greifen eines Objektes und ein anschließendes Greifen eines anderen Objektes führt nicht zwangsläufig zu einer Umstrukturierung der VR-Objekthierarchie. Lediglich das gegriffene Objekt wird unter die virtuelle Hand positioniert und wieder zurückgesetzt. Die Selektion eines Teilobjektes eines Aggregates führt automatisch zur Selektion des Gesamtaggregates. Dabei spielt es keine Rolle, welches Teilobjekt selektiert wird.

Die beschriebene Herangehensweise an die Verbindungsberücksichtigung über das Aggregatekonzept kann aus zwei Gründen auf den speziellen Fall der Leitungsverlegung nicht portiert werden.

Der *erste Grund* ist die Integration der CAD-Entities in das System. Die Unterscheidung von Move- und Changemodus (Abschnitt 4.1.4.) bei der Interaktion mit den CAD-Entities führt zu einer sich ständig verändernden VR-Objekthierarchie in Abhängigkeit vom Interaktionsmodus und in Abhängigkeit vom jeweils gegriffenen Objekt. Damit ist der Vorteil der stabilen VR-Objekthierarchie für die Dauer einer bestehenden Verbindung nicht mehr gegeben. Die VR-Objekthierarchie ist damit abhängig vom gegriffenen Objekt anzupassen.

Der *zweite Grund* für eine andere Herangehensweise an die Verbindungsberücksichtigung ist die Umsetzung einfacher logischer und physikalischer Zusammenhänge. Die Selektion eines Teilobjektes führt in der Realität nicht zwangsläufig zu einer Veränderung des zusammengesetzten Gesamtobjektes. Als Beispiel sei hier ein am Motorblock befestigter Schlauch genannt. Dieser ist systemintern über eine Stutzenverbindung mit dem Motorobjekt verbunden. Über das Aggregate-Konzept würde ein aus zwei Teilobjekten bestehendes Aggregat in der Objekthierarchie abgebildet werden, das durch die Selektion eines der beiden Objekte neu positioniert werden könnte. Im Falle der Selektion des Motorblockes führt diese Herangehensweise zu einem sinnvollen Ergebnis. Wird aber der Schlauch selektiert, würde der Anwender erwarten, daß entweder keine Positionierung erfolgt oder daß die Verbindung zwischen Schlauch und Motorblock aufgetrennt und der Schlauch allein positioniert wird. Daraus ergibt sich, daß die Greifoperation eines zusammengesetzten Objektes

abhängig vom gegriffenen Objekt ablaufen muß. Es handelt sich um einen gerichteten Greifvorgang.

Die Herangehensweise an die direkte Manipulation der Objekte unter der Berücksichtigung bestehender Verbindungen impliziert eine dynamische Anpassung der VR-Objekthierarchie an die jeweilige Situation. Dabei ist die Situation vom CAD-Manipulationsmodus, von den in der virtuellen Szene bestehenden Verbindungen und vom jeweils für die Interaktion selektierten virtuellen Objekt abhängig.

5.1.3. Ablauf der gerichteten Greif- und Positionierungsinteraktion

Für die Umsetzung der gerichteten Greifoperation steht in der Objektsemantik der konkretisierte Verbindungsgraf zur Verfügung. Das zu greifende Objekt wird in der virtuellen Umgebung durch eine Kollisionserkennung selektiert. Das selektierte Objekt wird an das Modul „Kopplung“ des Interaktionsmanagers propagiert. Existiert in der Objekt-Komponente des Interaktionsmanagers ein entsprechendes semantisches Objekt, so wird von diesem Objekt aus der Verbindungsgraf durchtraversiert. Diese Traversierung erfolgt mit Hilfe eines rekursiven Algorithmus, der alle Verbindungen dieses Objektes sucht, die zu anderen Objekten bestehen. Mit den gefundenen Objekten erfolgt der gleiche Ablauf. Die gefundenen Objekte, zu denen eine feste Verbindung besteht, werden in der Objekthierarchie des VR-Systems gesucht und unter das jeweils aufrufende Objekt „gehängt“. Damit werden alle fest verbundenen Objekte mit der Bewegung des selektierten Objektes mitgeführt. Diese Vorgehensweise erlaubt es, zusätzliche Attribute in der Objektsemantik in Abhängigkeit von der Traversierungsrichtung im Verbindungsgrafen mit zu berücksichtigen. Für den im vorangegangenen Abschnitt erläuterten Motorblock, der beim Greifen des angeschlossenen Schlauches nicht mitbewegt wird, existiert ein Attribut K-Fest mit dem Wert „TRUE“ (siehe auch Tabelle 4). Dieses wird bei der Traversierung des Verbindungsgrafen abgefragt. Nur wenn dieses Attribut den Wert „FALSE“ besitzt, wird auch das entsprechende Objekt als fest verbunden interpretiert. Ist das nicht der Fall, so endet an dieser Stelle der Algorithmus. Das Objekt mit dem Attribut K-Fest = „TRUE“ wird nicht unter das aufrufende Objekt „gehängt“ und damit ist die Verbindung zwischen beiden Objekten gelöst.

Im Verlauf der Traversierung können einzelne Objekte den Simulationskomponenten des VR-Systems bekanntgegeben werden, wenn für sie im Fall einer Neupositionierung neue Simulationsrechnungen durchgeführt werden müssen. Dieser Fall ist besonders für die CAD-Entities wichtig, bei denen im Changemodus bei jedem Bild die Splineinterpolation neu durchgeführt werden muß. Für diese Berechnung muß der CAD-Entity-Kernel die Objekte kennen, die sich bei jedem einzelnen Bild verändern.

Die situationsabhängige Umstrukturierung der VR-Objekthierarchie im Fall einer Greifoperation führt zu einer VR-Objekthierarchie-Struktur, die der Struktur im Verbindungsgrafen des Interaktionsmanagers entspricht. In Abbildung 58 ist schematisch ein möglicher Verbindungsgraf dargestellt. Er besteht aus neun Objekten, die über ihre Stutzen unterschiedlich miteinander verbunden sind. Zur Vereinfachung sind die Stutzen nicht mit abgebildet. Das Objekt mit der

Nummer 9 ist ein Objekt, dessen Attribut (Slot) K-Fest den Wert „TRUE“ besitzt. Anhand von drei unterschiedlichen Fällen wird die in Abhängigkeit vom gegriffenen Objekt resultierende hierarchische Struktur des VR-Szenengrafen dargestellt. In den Fällen 1 und 2 wird jeweils ein Objekt gegriffen, dessen Attribut (Slot) K-Fest den Wert „FALSE“ besitzt.

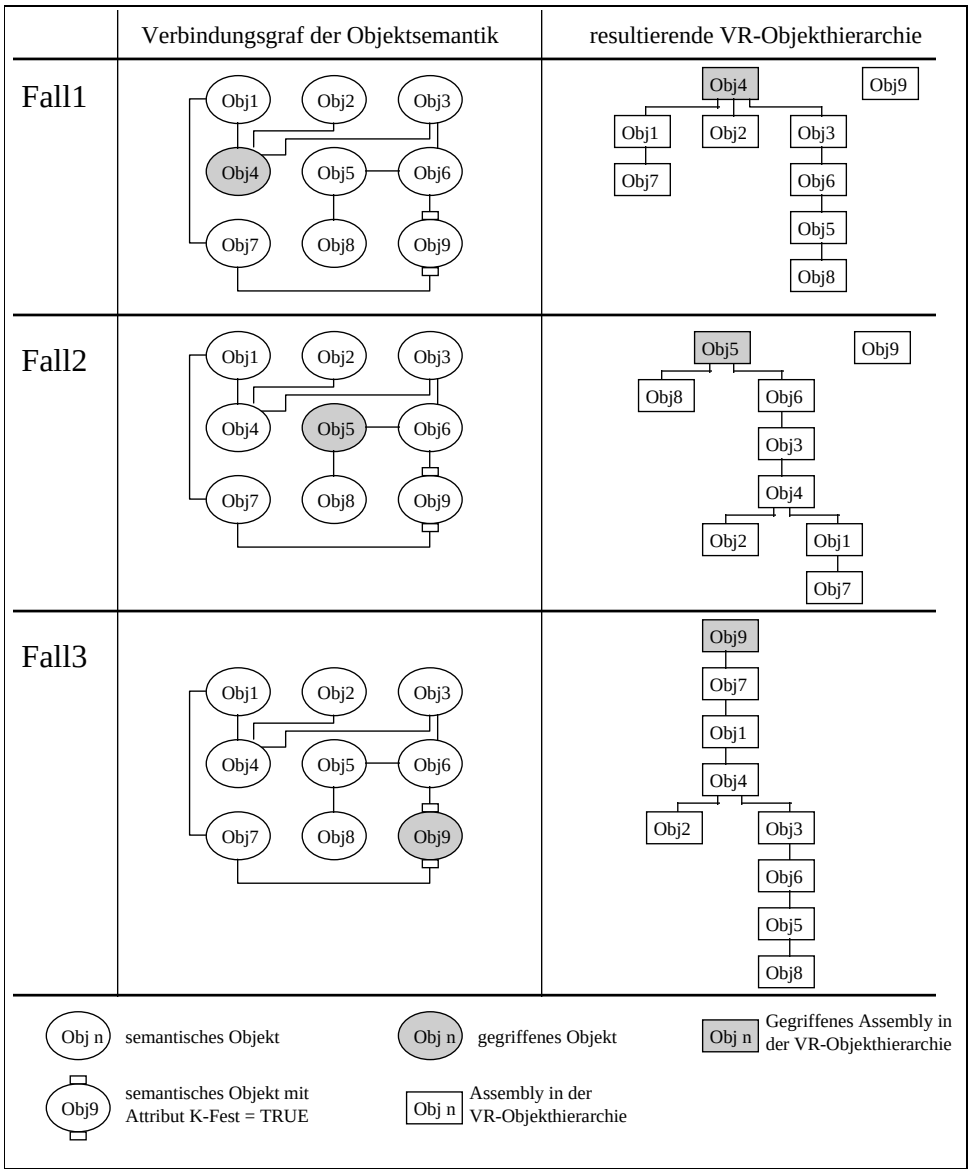


Abbildung 58: Verbindungsgraf der Objektsemantik und aus dem gerichteten Greifvorgang resultierende VR-Objekthierarchie bei unterschiedlichen gegriffenen Objekten

Die Verbindungen der Objekte 7 und 6 zum Objekt 9 werden dabei jeweils aufgetrennt. Damit wird das Objekt 9 bei dieser Greifoperation nicht mit positioniert. Es bleibt an seiner festgelegten Position. Im Fall 3 wird das als nicht beweglich gekennzeichnete Objekt 9 gegriffen. In diesem Fall bleiben die Verbindungen zu den Objekten 6 und 7 erhalten und alle verbundenen Objekte werden durch den Greifvorgang positioniert.

Die im Verlauf der Greifoperation aufgebaute VR-Objekthierarchie ist abhängig von der Durchlaufrichtung des zugrunde liegenden rekursiven Algorithmus durch die Stutzenfelder der einzelnen Instanzen der Klasse „Objekt mit Stutzen“. Eine Invertierung dieser Durchlaufrichtung führt zu einer invertierten VR-Objekthierarchie.

Das Loslassen des gegriffenen Objektes führt zu einer Umkehrung des Prozesses. Alle gegriffenen Objekte werden wieder innerhalb der VR-Objekthierarchie unter ihr ursprüngliches Vaterobjekt positioniert. Die Objektsemantik analysiert die einzelnen Verbindungszustände der Stutzen und aktualisiert gegebenenfalls die Werte der entsprechenden Attribute. Das ist notwendig, da das Ergebnis der Greifoperation eine neu erzeugte Verbindung sein kann. Diese wird dann bei der nächsten Greifoperation mit berücksichtigt.

Der bisher beschriebene Ablauf der Greifoperation gilt sowohl für die CAD-Entities als auch für die übrigen Objekte. Die Berücksichtigung der beiden Manipulationsmodi (Movemodus, Changemodus[97]) der CAD-Entities stellt eine Erweiterung des beschriebenen Ablaufes dar. Diese Erweiterung ist für den Fall des Changemodus notwendig. Für dessen Berücksichtigung wird die Struktur der Objektsemantik der Struktur im VR-System angepaßt. Die Stutzen der CAD-Entities befinden sich in der VR-Objekthierarchie direkt unter den Handles (siehe Abschnitt 4.2.2.). Das direkte Vaterobjekt des Entitystutzens ist also das entsprechende Handle. Dieses Objekt besitzt genau einen Stutzen.

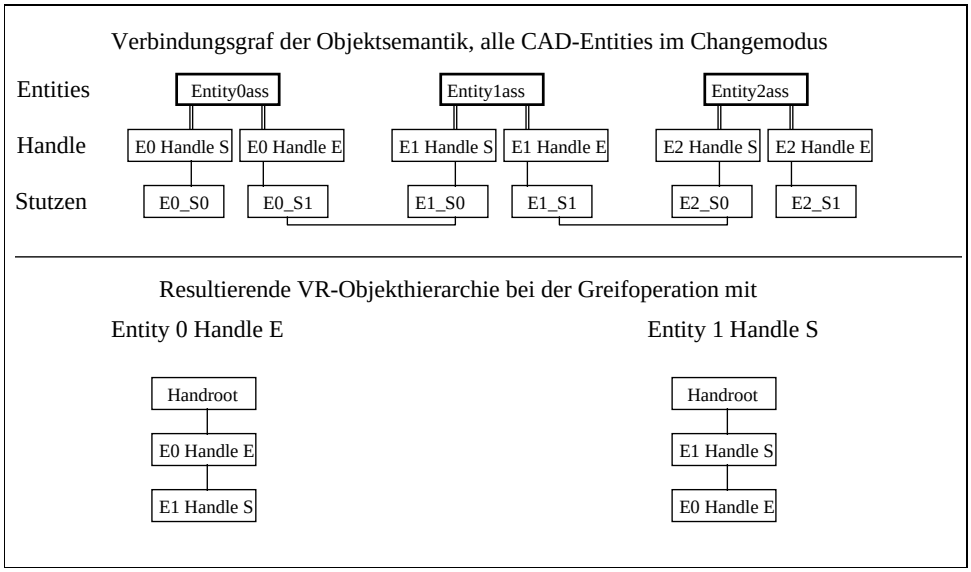


Abbildung 59: Greifoperation im Changemodus

Befindet sich ein CAD-Entity im *Movemodus* und eine Greifoperation erfolgt, dann wird als Vaterobjekt für die Stutzen das Entity selbst und nicht das Handle definiert. Damit kann beim Traversieren des Verbindungsgrafen der jeweils andere Entitystutzen gefunden und auf eine eventuell bestehende Verbindung untersucht werden.

[97] siehe auch Abschnitt 4.1.4.

Befindet sich das Entity hingegen im *Changemodus*, so wird nicht das CAD-Entity sondern das entsprechende Handle des CAD-Entities als Vaterobjekt definiert. Das Handle besitzt keinen weiteren Verbindungsstutzen. Damit ist die Traversierung des Verbindungsgrafen an dieser Stelle beendet. Über diesen Mechanismus der unterschiedlichen Vaterobjekte bei den CAD-Entities wird also der Verbindungsgraf der Objektsemantik direkt verändert. Dies führt zu unterschiedlichen Ergebnissen im Fall einer Greifoperation in Abhängigkeit davon, ob sich ein CAD-Entity im Move- oder Changemodus befindet. Dieser Sachverhalt wird in Abbildung 59 bis Abbildung 61 dargestellt. Jeweils drei im System enthaltene CAD-Entities sind abgebildet, von denen Entity0 mit Entity1 und Entity1 mit Entity2 über ihre Stutzen verbunden sind. In allen drei Fällen werden zwei Situationen dokumentiert. Es wird jeweils der aus einer Greifoperation resultierende Szenengraf der VR-Objekthierarchie dargestellt, wenn das Endhandle des Entitiy0 gegriffen wird und wenn das Starthandle des Entity1 gegriffen wird.

In Abbildung 59 befinden sich alle abgebildeten CAD-Entities im Changemodus. Die Vaterobjekte der Stutzen sind im Verbindungsgrafen der Objektsemantik die Handle. Die Traversierung des Verbindungsgrafen endet daher beim jeweils verbundenen Handle. In der VR-Objekthierarchie wird daher unter das gegriffene Handle jeweils das damit verbundene Handle „gehängt“. Für jeweils beide Handle wird die Operation „Verändern“ durchgeführt, wie sie bereits in Abschnitt 4.1.3. beschrieben wurde.

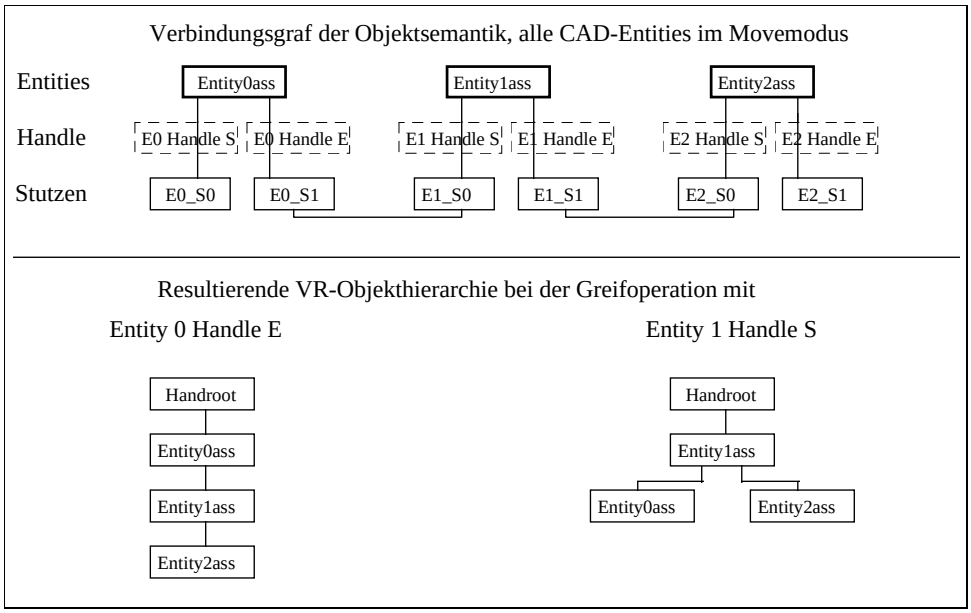


Abbildung 60: Greifoperation im Movemodus

In Abbildung 60 befinden sich alle abgebildeten CAD-Entities im Movemodus. Die Vaterobjekte der Stutzen sind im Verbindungsgrafen der Objektsemantik direkt die Entities. Die Traversierung des Verbindungsgrafen endet daher nicht beim jeweils verbundenen Handle, sondern erfolgt weiter über die Entitystutzen. Dem Interaktionsmanager wird nicht das gegriffene Handle, sondern das entsprechende Entity als gegriffenes Objekt übergeben. In der VR-Objekthier-

archie wird unter das gegriffene Entity jeweils das damit verbundene Entity „gehängt“. Der resultierende Szenengraf in der VR-Objekthierarchie ist abhängig vom jeweils gegriffenen Objekt. Für die gegriffenen Entities wird die Operation „Positionieren“ durchgeführt, wie sie bereits in Abschnitt 4.1.3. beschrieben wurde.

In Abbildung 61 befinden sich zwei CAD-Entities im Movemodus (Entity0 und Entity2) und ein CAD-Entity im Changemodus (Entity1). Die aus dieser Situation resultierenden Szenengrafen im Falle der entsprechenden Greifoperation sind ebenfalls abgebildet und verdeutlichen den Unterschied zu den beiden vorher beschriebenen Situationen.

In allen drei abgebildeten Situationen werden die bestehenden Verbindungen zwischen den CAD-Entities berücksichtigt und sie bleiben im Verlauf der Interaktion erhalten. Das Verhalten des Systems unterscheidet sich jedoch in den drei Situationen, was durch die unterschiedliche Strukturierung der VR-Objekthierarchie deutlich wird.

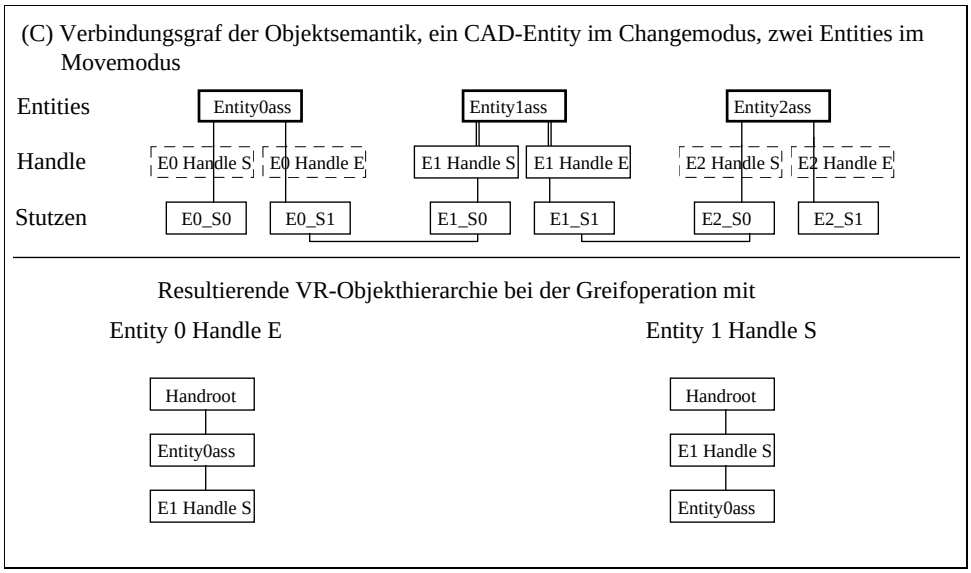


Abbildung 61: Greifoperation im gemischten Modus

5.1.4. Eigenschaften der gerichteten Greif- und Positionierungsinteraktion

Die Implementierung der gerichteten Greif- und Positionierungsinteraktion ermöglicht die Erhaltung bestehender Verbindungen zwischen den einzelnen Objekten im VR-System. Insbesondere die Anpassung der VR-Objekthierarchie an die jeweilige Situation, die durch das gegriffene Objekt und die unterschiedlichen Objektattribute gekennzeichnet ist, ermöglicht die Berücksichtigung der gestellten Anforderungen an ein System zur Modellierung von Schlauchobjekten.

Durch die Abbildung des Verbindungsgrafen in der Objektsemantik werden die bestehenden Verbindungen zwischen den Objekten sehr schnell gefunden. Es muß nicht jeder Stutzen mit jedem im System befindlichen Stutzen auf eine

Verbindung überprüft werden, da in den Slots der Objektklasse „Stutzen“ die möglichen Verbindungen wie auch die aktuell bestehenden Verbindungen abgelegt sind. Daraus resultiert eine Suchraumbeschränkung für das System. Das Anpassen der VR-Objekthierarchie erfolgt durch sehr schnelle Pointeroperationen. Daraus ergibt sich ein sehr schneller Ablauf der beschriebenen Anpassungen im Fall einer Greif- und Positionierungsinteraktion.

Im Verlauf der Traversierung des Verbindungsgrafen und der Aktualisierung der einzelnen semantischen Objekte werden auch die jeweils gesetzten Kollisionscallbackroutinen für die Stutzen aktualisiert. Für die Stutzen, die eine Verbindung mit einem anderen Stutzen eingegangen sind, werden die entsprechenden Callbackroutinen gelöscht. Für die Stutzen, bei denen eine Auftrennung der Verbindung erfolgt, werden diese Routinen wieder gesetzt. Über diesen Mechanismus wird verhindert, daß ein Objekt, welches durch eine virtuelle Szene bewegt wird und mit einem bereits verbundenen Stutzen kollidiert, durch einen constraintbasierten Verbindungsablauf in seiner Bewegung beeinflußt wird.

Über den gerichteten Greifvorgang können bewegliche und unbewegliche Objekte berücksichtigt werden, ohne dabei auf die grundsätzliche Bewegungsmöglichkeit der virtuellen Objekte verzichten zu müssen. Das erhöht die Praktikabilität des Systems. Durch das Greifen eines Schlauchobjektes, das mit einem relativ unbeweglichen Teil (Beispiel Motor) verbunden ist, wird nicht sofort der Motor neu positioniert. Diese Eigenschaft stimmt in ihren Grundzügen mit dem Verhalten in der Realität überein. Soll aber eine Motorpositionierung in der virtuellen Umgebung getestet werden, so kann das unter Beibehaltung der bestehenden Verbindungen erfolgen, indem der Motor selbst das gegriffene Objekt ist. Mit dieser Herangehensweise schließt das System die in der Realität nicht oder nur eingeschränkt möglichen Tests und Interaktionen nicht aus.

Die in der VR-Objekthierarchie abgebildete Verbindungskette von Objekten ermöglicht eine schnelle Auftrennung einer Verbindung. Die Trennoperation kann in den meisten Fällen auf das „Umhängen“ eines Teilbaumes zurückgeführt werden. Lediglich für den Fall einer bestehenden Schleife im Verbindungsgrafen trifft das nicht zu.

5.2. Anpassung des Hüllgeometrieradius der CAD-Entities

Mit Hilfe der in Abschnitt 4.3.8.4. beschriebenen Stripefakten werden Funktionalitäten umgesetzt die dafür sorgen, daß der Radius der Hüllgeometrie miteinander verbundener Entities immer konstant ist. Im Verlauf der Bearbeitung der CAD-Entities in der virtuellen Umgebung können auch die Hüllgeometrieradien der einzelnen CAD-Entities verändert werden. Diese Funktionalität wirkt sich stets auf ein einzelnes CAD-Entity aus. Unter der gegebenen Voraussetzung, daß ein aus mehreren CAD-Entities bestehendes Schlauchobjekt (Stripe) immer einen konstanten Radius der Hüllgeometrie besitzen soll, ergibt sich die Notwendigkeit für den Anwender, selbst für diesen Radiusabgleich zu sorgen. Über die Stripefakten steht dem Interaktionsmanager eine Sicht auf die virtuelle Szene zur Verfügung, die der Sicht des Anwenders auf diese Sze-

ne sehr ähnlich ist. Dies ist die Grundlage für die Implementierung eines regelbasierten Systemverhaltens, das der Erwartungshaltung des Anwenders sehr nahe kommt. Es bildet die Voraussetzung dafür, daß der in Abschnitt 2.3. beschriebene *gulf of execution* nach Norman für das beschriebene System verringert wird.

Drei Fälle müssen bei der dynamischen Anpassung des Hüllgeometrieradius unterschieden werden. Der *erste Fall* trifft zu, wenn der Anwender einem Schlauchstück explizit einen neuen Radius zuweist. Der *zweite Fall* trifft zu, wenn ein bestehendes Schlauchstück um ein oder mehrere CAD-Entities erweitert wird. Im *dritten Fall* erkennt das System selbständig die Modellierung eines Schlauches einer bestimmten Schlauchklasse und weist diesem Schlauch einen in der Objektklasse abgelegten Initialisierungswert für den Schlauchradius zu.

1. Fall: Explizites Zuweisen eines Wertes für den Radius

Ein Schlauch oder Teil eines Schlauches wird im System durch eine Menge miteinander verbundener CAD-Entities abgebildet. Ohne die Berücksichtigung der bestehenden Verbindungen muß der Anwender jedem einzelnen CAD-Entity den gewünschten Radius zuweisen. Mit Hilfe der Stripefakten werden für diese Interaktion Regeln definiert die dafür sorgen, daß eine solche Zuweisung sich immer auf den gesamten Stripe und damit auf das gesamte Schlauchstück bezieht.

2. Fall: Erweiterung eines Schlauchabschnitts

Bei der Modellierung eines Schlauches werden immer wieder ein oder mehrere zusammengesetzte CAD-Entities mit einem weiteren Schlauchstück verbunden. Unterscheiden sich die Hüllgeometrieradien dieser Einzelstücke, so werden sie durch das System automatisch aufeinander abgeglichen. Dazu muß das System entscheiden, welcher der zwei möglichen Radienwerte für die Zuweisung genutzt werden soll. Für einen entstehenden FreeStripe oder LRStripe gilt dabei die gleiche Regel. Der Radius des längeren Teilstückes[98] wird auf den resultierenden Stripe angewendet. Im Fall des LeftStripes ist eine Striperichtung definiert, über die der Radius selektiert wird. Der Wert des Radius am Stripeanfang[99] wird für alle Entities dieses Stripes gesetzt.

3. Fall: Zuweisen eines Klassenradius

Die als 3. Fall bezeichnete Situation trifft zu, wenn ein LeftStripe erzeugt wird. In diesem Fall wird ein CAD-Entity mit einem Nicht-CAD-Entity über die entsprechenden Stutzen verbunden. Aus dem Stutzen dieses Objektes und der Objektklasse des zugehörigen Vaterobjektes kann das System die Schlauchklasse ermitteln, mit der dieser Stutzen allgemein verbunden werden kann. Dabei greift das System auf die Informationen des nicht konkretisierten Verbindungsgrafen zurück, der im Abschnitt 4.3.5. kurz erläutert wurde. Für diese

[98] Die Länge eines Teilstückes ist hierbei durch die Anzahl der CAD-Entities definiert, aus denen das Teilstück zusammengesetzt ist.

[99] Als Stripeanfang wird das CAD-Entity definiert, das mit dem Nicht-Entity verbunden ist

Analyse ist die Existenz der entsprechenden Schlauchklasse, aber nicht notwendigerweise eine initialisierte Instanz dieser Klasse Voraussetzung. Wenn diese Objektklasse existiert, dann kann aus deren vorinitialisiertem Slot „Radius“ der Wert für den Radius des LeftStripe ermittelt werden. Das System nutzt damit den nicht konkretisierten Verbindungsgrafen, der in der Objektklassenhierarchie definiert ist, um den aktuell modellierten Schlauchtyp (die Schlauchklasse) automatisch zu ermitteln. Es sei an dieser Stelle angemerkt, daß diese Möglichkeit nur bei einer entsprechend vollständigen Definition der Klassenhierarchie inklusive der Radiuswerte der Schlauchobjekte besteht. Für die Modellierung der Schläuche in dem beschriebenen System ist es jedoch nicht zwingend erforderlich, daß die Klassenhierarchie dementsprechend vollständig definiert ist.

5.3. Unterschiedliche Anwendungsmöglichkeiten des Systems

Abhängig von der Komplexität der semantischen Informationen kann das System für unterschiedliche Aufgabenstellungen und Untersuchungen eingesetzt werden. Je vollständiger die virtuelle Umgebung mit Hilfe der semantischen Informationen im System beschrieben wird, desto größer ist der notwendige Anwendungsvorbereitungsaufwand. Es werden drei Komplexitätsstufen für die Objektsemantik unterschieden. Diese unterteilen sich nach der Art der definierten Objektklassen. Die Klasse der CAD-Entities ist im System stets enthalten. Damit stehen alle Funktionalitäten für die Bearbeitung dieser Objekte zur Verfügung. Weiterhin wird zwischen den Umgebungsobjekten ohne Stutzen und den Umgebungsobjekten mit Stutzen unterschieden. Die Umgebungsobjekte mit Stutzen unterteilen sich nochmals in die Schlauchobjekte und die Nicht-Schlauchobjekte. Damit können drei wesentliche Anwendungsarten des Systems unterschieden werden.

5.3.1. Umgebungsgeometrie ohne Stutzen, mit CAD-Entities

Bei der Anwendung mit der geringsten Komplexität bezüglich der semantischen Beschreibung der abgebildeten Objekte handelt es sich um eine virtuelle Umgebung, in der geometrische Objekte enthalten sind, diese aber dem Interaktionsmanager nicht bekanntgegeben werden. Der Anwender sieht diese Objekte, da sie in das VR-System eingeladen werden. Sie besitzen keine Stutzen. Der Aufwand zur Vorbereitung dieser Anwendung beschränkt sich auf die Definition der Daten, die in diese Umgebung eingeladen werden sollen. Dem Anwender stehen alle Funktionalitäten zur Erzeugung und Bearbeitung der CAD-Entities zur Verfügung. Die virtuelle Umgebung kann er zum Finden des Bauraumes für die entsprechenden Schläuche und Leitungen nutzen. Die Leitungen können jedoch nicht exakt mit den Anschlüssen verbunden werden, die die eingeladenen virtuellen Objekte möglicherweise zur Verfügung stellen. Die Genauigkeit der Positionierung der Anschlüsse ist von der Genauigkeit des genutzten Trackingsystems und der visuellen Kontrolle des Anwenders abhängig.

Die Anwendungsmöglichkeiten dieser Konfiguration bestehen vorzugsweise in der Konzeptphase einer Konstruktion. Die Anwendung des Systems hat ge-

zeigt, daß sich mit einer solchen Konfiguration der notwendige Bauraum für Versorgungsleitungen effektiv abschätzen läßt. Weiterhin können damit die Lage (Position und Orientierung) von Anschlußstutzen konzeptionell festgelegt werden. Es werden also die Daten für die Anschlußstutzen der zu konstruierenden Bauteile aus den verlegten Leitungen ermittelt.

5.3.2. Umgebungsgeometrie mit Stutzen und CAD-Entities

Bei dieser Konfiguration des Systems stehen neben den CAD-Entities Umgebungsobjekte mit Stutzen zur Verfügung. Es handelt sich also um Objekte, deren konstruktiver Aufbau festgelegt ist und die über Schläuche und Leitungen miteinander verbunden werden sollen. Der Aufwand für die Anwendungsvorbereitung ist komplexer als in der im vorangegangenen Abschnitt beschriebenen Konfiguration. Es sind sowohl die entsprechenden Objektklassen zu erzeugen und die Objekthierarchie, also die Zuordnung der Stutzen zu den Objekten, und die Zuordnung der geometrischen Objekte zu den Objektklassen festzulegen. Durch den Verzicht auf die Definition der Schlauchklassen kann aber auf den nicht konkretisierten Verbindungsgrafen, wie er in Abschnitt 4.3.5. beschrieben wurde, verzichtet werden.

Mit dieser Konfiguration kann der Bauraum von Schläuchen und Leitungen inklusive ihrer exakten Anschlüsse zu den Umgebungsobjekten untersucht und festgelegt werden. Eventuell notwendige Verschiebungen der Umgebungsobjekte unter Beibehaltung bestehender Verbindungen zu den CAD-Entities sind möglich.

Diese Systemkonfiguration stellt ein Optimum für die Modellierung von Schläuchen und Leitungen in einer virtuellen Umgebung dar. Die Nutzung von Norm- und Übernahmeteilen bei einer Konstruktion reduziert den notwendigen Vorbereitungsaufwand für die Anwendung des Systems, da bereits definierte Objektklassen immer wieder angewendet werden können. Es steht fast der gesamte Funktionsumfang des Systems zur Verfügung. Lediglich die Funktionen, die auf semantische Informationen der Schlauchklassen zurückgreifen, wie beispielsweise die automatische Zuweisung eines Hüllgeometrieradius für einen bestimmten Schlauchtyp (siehe auch Abschnitt 5.2. - 3. Fall), sind nicht verfügbar.

5.3.3. Umgebungsgeometrie mit Stutzen, Schläuchen und CAD-Entities

Eine vollständige Konfiguration des Systems wird erzielt, wenn alle Umgebungsobjekte mit ihren Verbindungsstutzen in der Objektklassenhierarchie definiert sind, inklusive der die einzelnen Umgebungsobjekte verbindenden Schlauchklassen. Damit stehen alle implementierten Funktionalitäten zur Verfügung. Der Vorbereitungsaufwand für eine entsprechende Anwendung ist im Vergleich zu den bereits beschriebenen Konfigurationen der höchste. Für diesen Fall ist die vollständige Definition des nicht konkretisierten Verbindungsgrafen notwendig. Die Praxis hat gezeigt, daß für die Festlegung des Bauraumes verschiedener Schläuche und Leitungen, d.h. also die reine Modellierung, diese Komplexität nicht erforderlich ist.

Für Montage- und Demontageuntersuchungen oder für die Anwendung des Systems zu Schulungszwecken kann diese komplexe Abbildung der Objektsemantik jedoch interessant werden. Es kann mit Hilfe dieser Informationen entschieden werden, ob ein Schlauchobjekt die korrekten Anschlüsse miteinander verbindet. Dieser Anwendungsfall soll jedoch hier nicht betrachtet werden, er ist nicht Gegenstand dieser Arbeit.

5.3.4. Fazit

In Abhängigkeit von der Vollständigkeit der semantischen Beschreibung der virtuellen Umgebung können unterschiedliche Anwendungszenarien mit unterschiedlichen Zielstellungen definiert werden. Die Komplexität der semantischen Beschreibung der virtuellen Szene steht dabei in direkter Proportionalität zum Vorbereitungsaufwand für eine entsprechende Anwendung.

Semantisch beschriebene Objekte	Vorbereitungs-aufwand	Anwendungsziel
CAD-Entites	niedrig	Konzeptionelle Bauraumfestlegung von Schläuchen und Leitungen
CAD-Entities Umgebungsobjekte (keine Schläuche und Leitungen)	mittel	Bauraumfestlegung von Leitungen mit exakten Verbindungen zur Umgebung; Möglichkeit der Verschiebung von Umgebungsteilen inklusive Verbindungserhaltung
CAD-Entities Umgebungsobjekte, Schläuche und Leitungen	hoch	Schulung, Montage-/ Demontageuntersuchungen

Tabelle 8: Anwendungszenarien abhängig von der Komplexität der semantischen Beschreibung der virtuellen Umgebung

Die semantische Beschreibung der virtuellen Umgebung ohne die zugehörigen Schlauchklassen hat sich als die effektivste Konfiguration für die angestrebten Systemziele herausgestellt. Diese Einschätzung beruht auf den Erfahrungen, die mit dem System in der Praxisanwendung gemacht wurden. Sie basiert auf der Tatsache, daß die für das System notwendigen semantischen Informationen im Rahmen einer Anwendungsvorbereitung in Handarbeit erzeugt werden müssen. Diese Vorgehensweise wird sich ändern, wenn in Zukunft semantische Informationen ein fester Bestandteil von Konstruktionsmodellen sind und an einer in der Prozeßkette Konstruktion vorgelagerten Stelle erzeugt und über entsprechende Schnittstellen auch mit übertragen werden.

5.4. Zusammenfassung

Mit den im 4. Kapitel beschriebenen Erweiterungen des VR-Systems stehen Objekte, Informationen und Simulationsabläufe zur Verfügung die es erlauben, realitätsnahe Basisoperationen mit virtuellen Objekten durchzuführen. Bei diesen Basisoperationen wurden die in der virtuellen Umgebung bestehenden

Verbindungen noch nicht berücksichtigt. Die Erweiterung der Interaktionen um diese Verbindungsberücksichtigung wurde im 5. Kapitel beschrieben.

Das Greifen, die Manipulation (Positionieren und Verändern) und das Loslassen virtueller Objekte wurde um die Berücksichtigung vorhandener und erzeugter Verbindungen erweitert. Ziel war es dabei, bestehende Verbindungen während einer Interaktion zu erhalten. Zwei mögliche Lösungsansätze wurden diskutiert.

Das aus der Literatur bekannte *Aggregatekonzept* wurde analysiert. Es konnte gezeigt werden, daß dieses Konzept nicht geeignet ist, CAD-Manipulationen, wie sie mit den CAD-Entities möglich sind, zu unterstützen. Weiterhin wurde gezeigt, daß ein gerichteter Greifvorgang, der in seinem Verhalten vom gegriffenen Objekt abhängt, für die Leitungsverlegung in einer virtuellen Umgebung notwendig ist. Auch dieses gerichtete Greifen wird durch das Aggregatekonzept nicht unterstützt.

Es wurde daher die *gerichtete Greif- und Positionierungsinteraktion* eingeführt und beschrieben. Diese ist in ihrem Verhalten abhängig

- von den bestehenden Verbindungen zwischen den Objekten
- vom Manipulationsmodus der CAD-Entities
- vom selektierten Objekt.

Mit der gerichteten Greif- und Positionierungsinteraktion wurde die Möglichkeit geschaffen, mehrere in der virtuellen Umgebung befindliche CAD-Entities gleichzeitig in unterschiedlichen Manipulationsmodi zu halten und diese auch zu berücksichtigen. Damit können bestehende Verbindungen während der Interaktion erhalten werden.

Weiterhin können bestehende Verbindungen abhängig vom gegriffenen Objekt berücksichtigt werden. Das erlaubt die Definition eines realitätsnäheren Verhaltens der Objekte. Beispielsweise können mit der Positionierung eines Kühlers automatisch alle daran befestigten Schläuche mitpositioniert werden. Im Falle der Positionierung eines Kühlwasserschlauches aber wird die Verbindung zum Kühler automatisch unterbrochen, damit dieser an seiner festgelegten Position verharrt.

Die Umsetzung der gerichteten Greif- und Positionierungsinteraktion ist durch verschiedene Eigenschaften gekennzeichnet:

- Bestehende Verbindungen zwischen den Objekten werden durch das System automatisch analysiert und konfiguriert. Das erfolgt sowohl beim Anwendungsstart für alle Objekte als auch bei jeder Interaktion partiell für die beteiligten Objekte.
- Ein schnelles Finden von Verbindungen durch eine auf dem Verbindungsgrafen basierende Suchraumbeschränkung wird erreicht.

- Eine Fehlervermeidung bei der Verbindungserzeugung durch Zu- und Abschalten von Systemfunktionen für verbundene und nichtverbundene Stutzen ist möglich.
- Schnelle Trennoperationen sind für verbundene Objekte möglich durch die logische Übereinstimmung zwischen Verbindungsgraf und VR-Objekthierarchie.

Einen weiteren Schwerpunkt stellt im 5. Kapitel die Festlegung der Hüllgeometrieradien dar. Die auf einzelnen CAD-Entities arbeitenden Basisoperationen wurden dahingehend erweitert, daß bestehende Verbindungen zwischen den CAD-Entities mit berücksichtigt werden. Dabei wird davon ausgegangen, daß der Radius eines Schlauches über seine gesamte Länge konstant bleibt.

Es wird zwischen drei verschiedenen Fällen unterschieden.

1. Beim *expliziten Zuweisen* eines Wertes für den Hüllgeometrieradius werden alle mit dem aktiven CAD-Entitiy verbundenen Objekte mit berücksichtigt.
2. Bei der *Erweiterung eines Schlauchabschnittes* um ein oder mehrere CAD-Entities wird der Hüllgeometrieradius über das gesamte entstehende zusammengesetzte Objekt automatisch, nach festgelegten Regeln egalisiert.
3. Es wird ein *Schlauchklassenradius* basierend auf der Definition des nicht konkretisierten Verbindungsgrafen der Objektklassenhierarchie zugewiesen. Hierbei erkennt das System aufgrund einer Verbindung zwischen CAD-Entitiy und Nicht-CAD-Entity die Schlauchklasse, die die CAD-Entities repräsentieren. Der in der Schlauchklasse definierte Radius wird den Entities automatisch zugewiesen.

Die Berücksichtigung der Verbindungen zwischen den Objekten bei der Manipulation der Hüllgeometrieradien vereinfacht die Benutzung des Systems duch eine Minimierung der explizit einzugebenden Interaktionsbefehle. Es konnte gezeigt werden, daß der Anwender dadurch von sich wiederholenden Interaktionen entlastet wird.

Im 5. Kapitel wurde gezeigt, daß die Berücksichtigung bestehender Verbindungen bei den verschiedenen Interaktionen zu einer Interaktionsvereinfachung führt. Sie setzt aber eine vordefinierte Objektsemantik voraus, die wiederum den Anwendungsvorbereitungsaufwand erhöht. Abhängig von der Komplexität der vorhandenen semantischen Informationen wurde daher die Anwendbarkeit des Systems hinsichtlich der Anwendungsziele untersucht.

Es wird zwischen 3 verschiedenen Komplexitätsstufen unterschieden.

Bei einem *niedrigen* Anwendungsvorbereitungsaufwand sind lediglich die CAD-Entities semantisch beschrieben. Diese Konfiguration eignet sich für eine konzeptionelle Bauraumfestlegung von Schläuchen und Leitungen.

Bei einem *mittleren* Aufwand sind die CAD-Entities und die Umgebungsobjekte semantisch definiert. Diese Konfiguration eignet sich für eine exakte Festle-

gung des Leitungsverlaufes mit exakten Verbindungen zur Umgebungsgeometrie.

Mit einem *hohen* Aufwand sind sowohl die CAD-Entities, die Umgebungsobjekte wie auch die Objektklassen für die Schläuche und Leitungen definiert. Mit dieser Konfiguration ist das System nicht nur für die Leitungsverlegung, sondern auch für Montage- und Demontageuntersuchungen wie auch für Schulungszwecke einsetzbar.

6. Praktische Erfahrungen

In den bisherigen Abschnitten wurden die im Rahmen dieser Arbeit realisierten Systemerweiterungen beschrieben. Insbesondere wurden die speziell für die Leitungsverlegung integrierten Systemkomponenten, die CAD-Entities, Verbindungsstutzen und semantischen Zusatzinformationen inklusive der auf diesen Systemkomponenten basierenden Funktionalitäten beschrieben. Eine Erweiterung eines VR-Systems um diese zusätzlichen Komponenten führt nicht zwingend zu einem allgemein akzeptierten Werkzeug, mit dem effektiv das Anwendungsziel erreicht werden kann. Vielmehr wird die Definition eines optimalen Systems durch die Fülle der Konfigurationsmöglichkeiten eines VR-Systems, der heute zur Verfügung stehenden Ein- und Ausgabehardware wie auch der Gestaltungsmöglichkeiten einer Anwenderschnittstelle erschwert. Aus diesem Grund werden in diesem Kapitel die mit dem System gesammelten Erfahrungen zusammengefaßt und die sich daraus ergebenden Systemkonfigurationen erläutert.

Es wird auf die Gestaltung der Anwenderschnittstelle eingegangen. Diese unterteilt sich in die Hardwarekonfiguration wie auch in Softwarefunktionalitäten, die sich für die Anwendung der Leitungsverlegung als besonders nützlich erweisen. Dabei handelt es sich um Funktionalitäten, die nicht ausschließlich für die Leitungsverlegung angewandt werden können, sondern auch auf andere Anwendungen in einer virtuellen Umgebung übertragbar sind. Abschließend werden die einzelnen Schritte erläutert, die für die Vorbereitung einer Anwendung notwendig sind.

6.1. Anwenderschnittstelle

Aus der Vielzahl der möglichen Ein- und Ausgabegeräte ist es schwierig Gerätekombination zu finden, die sich für die Anwendung der virtuellen Leitungsverlegung als vorteilhaft erweisen. Lediglich der visuelle Ausgabekanal wird in diesem Abschnitt betrachtet. Andere mögliche Ausgabekanäle spielen für das beschriebene System keine beziehungsweise nur eine unbedeutende Rolle. Die Wahl eines geeigneten Ausgabegerätes hat Auswirkungen auf die Wahl der angewendeten Eingabegeräte. Daher wird hier zunächst das angewendete Ausgabesystem diskutiert. Anschließend wird die Eingabehardware unter der speziellen Berücksichtigung der Ausgabegeräte bewertet.

6.1.1. Das visuelle Ausgabesystem

Bei der Leitungsverlegung in einer virtuellen Umgebung handelt es sich um eine hochinteraktive Anwendung. Sie ist durch eine hohe Datenkomplexität gekennzeichnet. Ein effektives Arbeiten ist nur dann möglich, wenn der Anwender sich gut in der virtuellen Umgebung orientieren kann und in die Lage versetzt wird, die virtuellen Objekte zielsicher und schnell für seine Interaktionen zu selektieren. Daraus ergeben sich verschiedene Eigenschaften der visuellen Ausgabesysteme, anhand derer die Eignung für die beschriebene Anwendung beurteilt werden muß.

Die hohe Datenkomplexität ist nur mit einem Ausgabesystem zu bewältigen, das eine hohe Renderingperformance erlaubt. Eine gute Orientierung in einem VR-System ist durch ein großes Sichtfeld gegeben. Eine hohe Immersion verbunden mit einer hohen Eigenpräsenz des Anwenders in der virtuellen Umgebung ist die Voraussetzung für zielsichere und schnelle Interaktionen. Weitere wichtige Kriterien sind der Nutzerkomfort und die Kommunikationsmöglichkeit mit anderen Anwendern, d.h. die Mehrbenutzerfähigkeit des Systems.

	CAVE™-like Mehrseiten-projektion	Stereoprojektions-wand Powerwall	Benches	HMD[100] BOOM™	Monitor
Redering-performance	0	+	0	+	-
Sichtfeld (field of view)	+	0	0	-	-
Immersion des Anwenders	+	0	0	+	-
Eigenpräsenz des Anwenders	+	+	+	-	+
Nutzerkomfort	0	0	0	-	+
Kommunikations-möglichkeit	+	+	+	-	+
Raum-, Proportions- und Größen-beurteilung	+	0	-	0	-
Bewegungsfreiheit	+	0	0	+	-
- niedrig 0 mittel + hoch					

Tabelle 9: Beurteilung verschiedener visueller Ausgabesysteme

Die Beurteilunsmöglichkeiten von Größe, Raum und Proportionen der abgebildeten Geometrien spielen eine wichtige Rolle bei der schnellen Beurteilung vorhandener Freiräume. Die Bewegungsfreiheit des Anwenders stellt eine Voraussetzung für die schnelle und realitätsnahe Navigation innerhalb eines eingeschränkten Bewegungsraumes dar. Dabei ist insbesondere die Möglichkeit der Betrachtung eines Objektes aus verschiedenen Blickpositionen gemeint. In [RIEDEL98] werden verschiedene Ausgabesysteme bezüglich des Immersionsgrades bewertet. Dabei spielt die Immersion des Anwenders in der virtuellen Umgebung und im abgebildeten Produkt eine Rolle. Weiterhin wird die Möglichkeit der Simulation menschlicher Aktivitäten in einer virtuellen Umgebung in Abhängigkeit vom Ausgabesystem beurteilt. Für die Beurteilung

[100] Als HMD stand im Rahmen der Arbeit ein hochauflösender Helm (Auflösung: 1280x1024 Pixel) der Firma n-Vision zur Verfügung.

werden die möglichen Ausgabesysteme in fünf verschiedene Klassen unterteilt. Es wird dabei zwischen Mehrseitenprojektionssystemen (CAVE™-like, CUBEs), Powerwalls, Benches, HMD und BOOM™ unterschieden. Diese Bewertung und eigene Erfahrungen mit den unterschiedlichen visuellen Ausgabesystemen führen zu der in Tabelle 9 dargestellten Beurteilung der möglichen Ausgabesysteme für den speziellen Fall der Leitungsverlegung. Die Übereinstimmung der Beurteilungsergebnisse von HMD und BOOM™ in [RIEDEL98] führen zu einer Zusammenfassung dieser beiden Systeme. Die Menge der Ausgabesysteme wird weiterhin durch den gewöhnlichen stereofähigen Computermonitor ergänzt.

Die Renderingperformance des Ausgabesystems wird in Tabelle 9 nach dem genutzten Stereoverfahren beurteilt. Das aktive Verfahren (siehe auch Abschnitt 3.2.1.1.) erzwingt das Rendern von zwei Bildern auf einer Geometriepipe des Grafikrechners[101]. Das passive Verfahren ermöglicht eine Verteilung der Bilder für das rechte und das linke Auge auf zwei unabhängige Geometriepipes. Damit ist das passive Verfahren bezüglich der Renderingperformance leistungsfähiger. Diese Sichtweise erlaubt im Grunde nur eine Unterscheidung von zwei Qualitätsstufen. Die Unterscheidung von drei Qualitätsstufen bei der Beurteilung in Tabelle 9 ist damit zu begründen, daß bei der Nutzung von mehreren Projektionsflächen auch eine Aufteilung der zu verarbeitenden Geometrie auf die unterschiedlichen Grafikpipes erfolgt. Damit ist die für jede einzelne Grafikpipe zu verarbeitende Datenmenge reduziert und erhöht zum Teil die Renderingperformance.

Die Darstellungsqualität der einzelnen Systeme wird in Tabelle 9 nicht mit berücksichtigt. Sie spielt bei der virtuellen Leitungsverlegung keine so große Rolle wie beispielsweise bei einer Designbeurteilung.

Schlußfolgerung:

Die Auswertung der Tabelle 9 zeigt, daß aus heutiger Sicht das für die Leitungsverlegung am besten geeignete visuelle Ausgabesystem die Mehrseitenprojektion ist.

Die Nutzung des gewöhnlichen Computermonitors für die Leitungsverlegung ist zwar möglich, die Handhabung der Anwendung verliert jedoch an Intuitivität. Das ist auf die nicht vorhandene 1:1-Darstellung der virtuellen Umgebung und die geringe Immersivität des Systems zurückzuführen. Die für einen Monitorarbeitsplatz am besten geeigneten Eingabegeräte sind Tastatur, Maus und Spacemouse. Die Steuerung der Anwendung über diese Geräte erschwert die Navigation und direkte Manipulation der Objekte im Raum. Die einfache Navigation und Objektmanipulation stellt aber einen wesentlichen Vorteil der Anwendung der virtuellen Realität für die Leitungsverlegung dar. Dieser Vorteil geht daher bei der Nutzung des Monitores verloren.

[101] Heutige Hochleistungsgrafikrechner können eine oder mehrere Geometriepipes besitzen, auf die die Berechnung der Stereobilder verteilt werden kann. Diese Möglichkeit der Verteilung der Berechnung besteht jedoch nur beim passiven Stereoverfahren.

Der geringe Nutzungskomfort und die eingeschränkten Kommunikationsmöglichkeiten, die der HMD wie auch der BOOM™ bieten, lassen diese Ausgabesysteme für die praktische Anwendung ausscheiden. Eine längere Arbeit unter einer völligen visuellen Abgrenzung zur Außenwelt ist dem regelmäßigen Anwender eines VR-Systems nicht zuzumuten. Darüber hinaus hat es sich gezeigt, daß das geringe Sichtfeld dieser Systeme ein natürliches Arbeiten nicht zuläßt. Die einzige bei der Arbeit zur Verfügung stehende Rückkopplung ist die Visualisierung der virtuellen Hand, da die reale Hand bei diesem Ausgabesystem nicht zu sehen ist. Um das Bild der realen Hand aber im Sichtfeld zu haben, muß man als Anwender entweder ständig nach unten auf die eigenen Hände schauen oder mit erhobenen Händen arbeiten. Beide Varianten führen schnell zu Ermüdungserscheinungen.

Die Benches wie auch die Wandprojektionen (Powerwall, Stereoprojektionswand) sind für die Leitungsverlegung bedingt geeignet. Der hauptsächliche Nachteil der Benches liegt im geringen Sichtfeld und der eingeschränkt möglichen 1:1-Darstellung der virtuellen Umgebung. Speziell die unter dem Begriff Holobench bekannten Benches mit zwei Projektionsebenen erlauben eine effektive direkte Interaktion mit den virtuellen Objekten. Das Arbeiten vor der eigentlichen Projektionswand ist dabei sehr gut möglich, da die reale Hand als Referenzobjekt bei der Interaktion genutzt werden kann. Das gilt nicht für die Stereoprojektionswand. Bei den direkten Manipulationen der virtuellen Objekte steht dem Anwender lediglich das virtuelle Bild der Hand hinter der Projektionswand als Referenzobjekt zur Verfügung. Das erzwingt eine relativ hohe Bildrate während der Interaktion. Diese wiederum ist bei den passiven Stereoverfahren erreichbar. Damit ist der Anwender aber gezwungen seinen Kopf nicht nach rechts oder links zu neigen, da sonst die Polarisation und damit der dreidimensionale Effekt verloren geht.

Die Anwendung der Mehrseitenprojektion für die Leitungsverlegung ist die beste Alternative, die zum heutigen Zeitpunkt zur Verfügung steht. Zwei Nachteile sind dennoch zu nennen. Dabei handelt es sich um die relativ geringe Darstellungsqualität, die sich in einer relativ geringen Helligkeit und in einem geringen Kontrast ausdrückt. Die durch das aktive Stereoverfahren bedingte geringe Bildrate wird durch die Aufteilung der geometrischen Objekte auf die verschiedenen Wände teilweise, jedoch bei weitem nicht ganz kompensiert. Daraus ergibt sich eine geringere Bildrate als bei den passiven Verfahren. Die Bildrate ist aber entscheidend für eine effektive Interaktion mit den virtuellen Objekten. Der sich aus diesen Tatsachen ergebende Nachteil relativiert sich bei der Mehrseitenprojektion durch die hohe Eigenpräsenz des Benutzers in der virtuellen Umgebung bei einer gleichzeitig hohen Immersion und einem sehr großen zur Verfügung stehenden Sichtfeld. Dadurch erhält er eine sehr gute Beziehung zu den dargestellten Objekten. Bei einer Selektion eines virtuellen Objektes steht ihm damit seine reale Hand als Referenzobjekt zur Verfügung, wodurch er sehr schnell eine gewünschte Position anfahren kann. Die durch eine geringe Bildrate ausgelöste Verzögerung der Bewegung der virtuellen Hand spielt damit eine wesentlich geringere Rolle als bei den alternativen Ausgabesystemen. Die direkten Manipulationen können trotz geringer Bildrate noch effizient ausgeführt werden.

6.1.2. Eingabesystem

Die Gerätekonfiguration für das VR-System wird hier unter dem Blickwinkel der Nutzung einer Mehrseitenprojektion betrachtet. Es werden zunächst die genutzten Geräte und die damit gesammelten Erfahrungen diskutiert. Anschließend wird überblicksweise auf Softwarefunktionalitäten eingegangen, die für die virtuelle Leitungsverlegung entwickelt wurden, aber auch auf andere Anwendungen und Hardwarekonfigurationen übertragbar sind.

6.1.2.1. Hardware

6.1.2.1.1. Bewegungstracking

Der Einsatz eines Trackingsystems ist allein bei Verwendung der Mehrseitenprojektion unumgänglich. Die Aufnahme von Position und Orientierung des Anwenderkopfes zur Steuerung der virtuellen Kamera stellt die Voraussetzung für eine geometrisch korrekte Abbildung der Geometrie innerhalb einer Mehrseitenprojektion dar. Aber auch die direkte Manipulation der virtuellen Objekte macht ein Bewegungstracking für mindestens einen Unterarm unumgänglich. Im genutzten CUBE mit 3 quadratischen Projektionsseiten einer Kantenlänge von 2,5 Metern[102] wird ein magnetisches Trackingsystem vom Typ Fastrak™ der Firma Polhemus eingesetzt. Dabei kommt das sogenannte Longrange-System zum Einsatz, das einen für die Maße des CUBEs ausreichendes Arbeitsfeld abdeckt.

Die Nichtlinearität und Ungenauigkeit des eingesetzten Trackingsystems macht eine Magnetfeldentzerrung notwendig, die softwareseitig gelöst wird[103]. Der Einsatz des Systems hat gezeigt, daß die Praktikabilität durch die genutzten Kabel eingeschränkt ist. Die Verwendung des Polhemus Stylus™[104] hat sich für die Leitungsverlegung als günstig erwiesen. Er ist im Gegensatz zum Datenhandschuh einfacher zu benutzen. Es kann auf eine Gestenerkennung und -kalibrierung verzichtet werden und eine einfache Weitergabe des Gerätes an weitere Personen kann ohne besondere Umstände erfolgen.

6.1.2.1.2. Sprache

Die Eingabe von Instruktionen ist für jede Computeranwendung unumgänglich. Die dafür gängigen Eingabegeräte Maus und Tastatur stehen in einer immersiven VR-Umgebung nicht zur Verfügung. 3D-Menüs können in der virtuellen Umgebung nur dann genutzt werden, wenn dem Anwender ein Eingabegerät zur Verfügung steht, mit dem er das Menü bedienen kann. Der Stylus™ oder ein anderes mit einem Bewegungstrackingsystem versehenes Eingabegerät läßt sich dafür nutzen. Die 3D-Menüs führen aber zu Problemen bei der direkten Manipulation von virtuellen Objekten. Als Beispiel sei die Positionierung eines CAD-Entities genannt. Bei der Nutzung eines 3D-Menüs bestünde nicht die Möglichkeit, während der Positionierung des Objektes in eine andere Dar-

[102] Zwei Seitenflächen und der Boden werden als Projektionsflächen genutzt

[103] [ZACH97]

[104] Beim Stylus handelt es sich um einen in allen 6 Freiheitsgraden getrackten Stift mit einem Eingabeknopf. Er wird zusammen mit dem Fastrak-System von der Fa. Polhemus angeboten.

stellungsart umzuschalten. Daher wird auf die Eingabe von Sprachbefehlen zurückgegriffen. Diese Eingabe erfolgt unabhängig von den anderen Eingabegeräten und kann kabellos über ein Funkmikrofon betrieben werden. Die für die Leitungsverlegung entwickelte und angewendete Sprachsyntax ist im Anhang grafisch abgebildet.

6.1.2.1.3. Datenhandschuh, Stylusknopf

Wie bereits oben erläutert, erfolgen die direkten Manipulationen der virtuellen Objekte über ein getracktes Eingabegerät. Der Einsatz des Polhemus Stylus™ hat sich gegenüber dem Datenhandschuh als vorteilhafter erwiesen. Er läßt sich problemloser weiterreichen und benötigt ein Kabel weniger für seinen Betrieb. Das System zur virtuellen Leitungsverlegung unterstützt zwar die Arbeit mit beiden Händen, dennoch hat die Praxis gezeigt, daß der Einsatz nur eines Manipulationsgerätes genügt. Der Nutzen von Handgesten für die Kommandoeingabe muß als kritisch bewertet werden. Als Gründe hierfür sind die Kalibrierungsnotwendigkeit und damit die Nutzerabhängigkeit und die Unsicherheit der Gestenerkennung zu nennen.

6.1.2.1.4. Spacemouse

Die Spacemouse wird in der Anwendung als reines Navigationsinterface genutzt. Sie dient der Positionierung des Anwenders in der virtuellen Umgebung. Diese Navigation beschränkt sich allerdings hauptsächlich auf den Start der Anwendung, um eine geeignete Position für die Arbeit in der virtuellen Umgebung festzulegen. Obwohl die Spacemouse für diese Zwecke vom System unterstützt wird, sei an dieser Stelle angemerkt, daß sie für die Leitungsverlegung eine geringe Rolle spielt. Ihre Bedienung beansprucht beide Hände des Anwenders und auch dieses Gerät arbeitet nicht kabellos. Die Abbildung der Navigation auf den Polhemus Stylus™ macht die Spacemouse in dem System überflüssig.

6.1.2.2. Software

Im Folgenden wird auf eine kleine Anzahl nützlicher Interaktionsmöglichkeiten eingegangen, ohne diese jedoch ausführlich zu dokumentieren. Diese Funktionalitäten vereinfachen die Bedienung des Systems und sind teilweise auch auf andere virtuelle Umgebungen übertragbar. Eine kurze funktionale Beschreibung soll genügen, um die Möglichkeiten des Systems zu umreißen.

6.1.2.2.1. Anpassung der Größe des virtuellen Anwendermodells

Über das in Abschnitt 3.2.3. beschriebene Flying-Carpet-Modell wird der Anwender in der virtuellen Umgebung repräsentiert. Über die Skalierung des Gesamtmodells wie auch der einzelnen Teilobjekte und deren Bewegungsgeschwindigkeiten[105] kann auf die Bewegungssteuerung Einfluß genommen werden. Durch die Veränderung dieser Faktoren kann der Ein-

[105] Die geometrische Repräsentation der Handmodelle kann durch einen Skalierungsfaktor verändert werden. Weiterhin kann auf den Augenabstand Einfluß genommen werden. Die Bewegungsgeschwindigkeit des Kameramodells und der Hände wie auch des gesamten Carts können einzeln angepaßt werden.

druck vermittelt werden, als würde der Anwender in der virtuellen Umgebung verkleinert und vergrößert werden. Diese Anpassung ist dann sinnvoll, wenn an relativ kleinen Einzelheiten der virtuellen Szene gearbeitet werden soll. Durch die Verkleinerung des Anwenders können die virtuellen Objekte aus einem geringeren Abstand betrachtet werden als in einer 1:1-Darstellung. Weiterhin werden die durch die Eingabegeräte bedingten absoluten Fehler bei der Eingabe ebenfalls geringer. Problematisch und für den Anwender ungewohnt ist jedoch die Wahl geeigneter Skalierungsfaktoren. Dazu wurde ein Modul entwickelt, das die Anpassung dieser Faktoren automatisiert. Als Eingabewert wird der Abstand des Kameraobjektes zum nächsten virtuellen Objekt in Blickrichtung genutzt. Stellt der Anwender fest, daß der Augenabstand nicht mehr optimal ist, kann über ein festgelegtes Ereignis die Neuberechnung der Skalierung des Anwendermodells ausgelöst werden. Anschließend skaliert das System über kleine Zwischenschritte auf den neu berechneten Skalierungsfaktor. Dieser „weiche“ Übergang über eine lineare Interpolation macht die Veränderung (Vergrößerung oder Verkleinerung) für den Anwender nachvollziehbar. Diese Herangehensweise stellt sicher, daß lediglich eine Eingabe notwendig ist, um eine geeignete Skalierung des Anwendermodells zu erreichen. Dabei ist es unwichtig, ob es sich um eine Vergrößerung oder um eine Verkleinerung des Modells handelt.

6.1.2.2.2. Handbewegungsgeschwindigkeit

Bei einer 1:1-Abbildung des Anwenders auf das virtuelle Modell des Anwenders müssen für den Betrachter Position und Orientierung von virtueller und realer Hand übereinstimmen. Das hat den Vorteil, daß die reale Hand als Referenzobjekt bei den Interaktionen genutzt werden kann (siehe auch Abschnitt 6.1.1.). Sowohl durch die Kabellängen der Eingabegeräte wie auch durch die Projektionsleinwände der Mehrseitenprojektion ist aber eine erhebliche Bewegungsbeschränkung für den Systemnutzer gegeben. Eine Überwindung dieser Problematik wird durch die Navigation des Anwenders durch die virtuelle Szene erreicht. Eine solche Navigation ist aber nicht immer erwünscht, beispielsweise wenn gerade eine geeignete Position für die eigentlichen Interaktionen gefunden wurde. Mit der Beeinflussung der Bewegungsgeschwindigkeit der virtuellen Hand kann der Interaktionsraum ohne eine Navigation vergrößert werden. Eine geeignete Skalierung sorgt dafür, daß auch weiter entfernt liegende Objekte mit Hilfe der Hand erreicht werden können, ohne sich durch die virtuelle Umgebung navigieren zu müssen. Eine Verkleinerung der Handgeschwindigkeit ist ebenso möglich. Sie sorgt für eine genauere Positionierbarkeit gegriffener Objekte und erhöht so die Bedienungseffizienz des Systems.

6.1.2.2.3. Werkbank für Schlauchstücke

Die Verläufe der CAD-Entities und damit der zu verlegenden Schläuche werden bei der Erzeugung der Objekte festgelegt. Die Vorgehensweise wurde ausführlich in Abschnitt 4.1.4. beschrieben. Unterschiedliche Eingabemöglichkeiten stehen dem Systemanwender zur Verfügung. Alle Erzeugungsfunktionen werden auf die Eingabe einzelner Punktkoordinaten zurückgeführt. Diese Herangehensweise hat sich in der Praxis als verhältnismäßig umständlich herausgestellt. Eine virtuelle Werkbank als Interaktionsmetapher vereinfacht diese Interaktion. Auf der Werkbank liegen verschiedene Beispielschläuche (CAD-Entities). Um ein entsprechendes Objekt zu erzeugen, muß es lediglich durch Berühren mit der virtuellen Hand ausgewählt werden. Es befindet sich danach automatisch im Movemodus und wird mit der virtuellen Hand positio-

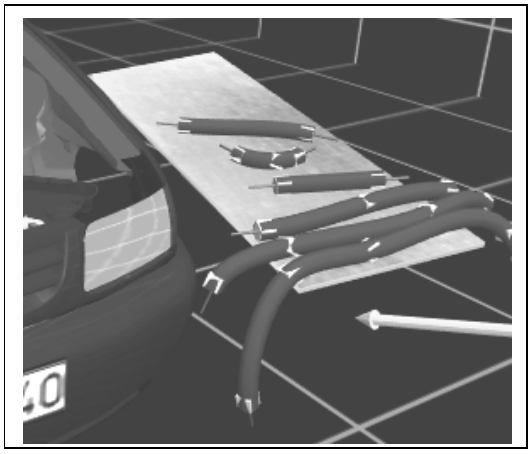


Abbildung 62: Virtuelle Werkbank

niert. Das reduziert die unterschiedlichen Befehle, die der Anwender sonst kennen muß und verbirgt den CAD-Entity-Typ vor dem Anwender. Für ihn handelt es sich bei allen CAD-Entities um Schlauchstücke, die in der gleichen Art und Weise erzeugt werden können.

6.1.2.2.4. Navigation

Wie bereits oben erläutert, ist die Navigation durch die virtuelle Umgebung eine grundlegende Interaktionsmöglichkeit, die über die verschiedensten Eingabegeräte und Bedienmetaphern durchgeführt werden kann. Eine relativ einfache Navigation über die Spacemouse hat sich in der Mehrseitenprojektion als umständlich erwiesen, da sie den Einsatz beider Hände des Anwenders erfordert. Als Alternativen stehen die Navigation über Sprache, mit dem Handschuh und dem Stylus^{TM} zur Verfügung. Die dabei effektivste Methode stellt die Navigation mit Hilfe des Stylus^{TM} dar. Da sich dieser sowieso in der Hand des Nutzers befindet, wurde eine entsprechende Navigationsmöglichkeit implementiert. Dabei „schiebt“ der Anwender die virtuelle Szene unter seinen Füßen durch. Die Bewegung wird aus der Drehung des Stylus^{TM} im Raum berechnet. Sie erfolgt ausschließlich in der X-Y-Ebene, die Höhe in der virtuellen

Umgebung (Z-Achse), in der sich der Betrachter befindet, bleibt konstant[106]. Diese Navigationsmöglichkeit ist ausschließlich für eine Mehrseitenprojektion geeignet, da ein Nicken und Wanken der virtuellen Repräsentation des Anwenders bei diesem Ausgabesystem nicht notwendig ist. Die Betrachtung der virtuellen Umgebung aus verschiedenen Blickrichtungen erfolgt durch die reale Bewegung des Nutzers.

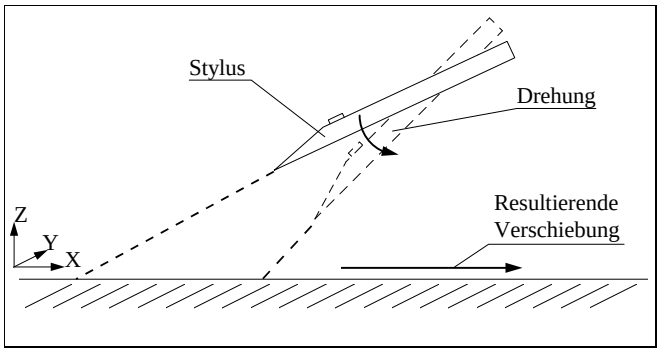


Abbildung 63: Navigation mit dem Stylus™

6.1.2.2.5. Nachgreifen

Die im System umgesetzte Vorgehensweise bei der Positionierung der Objekte (Abschnitt 3.5.2. Alternative C) läßt eine Unterbrechung der Verbindung zwischen virtueller Handrepräsentation und gegriffenem Objekt zu. Das kann im Laufe der Arbeit dazu führen, daß für die weitere effektive Arbeit ein Nachgreifen des Objektes erforderlich ist. Auch ohne eine systemseitige Bewegungsbeeinflussung des gegriffenen virtuellen Objektes kann eine Korrektur der Positionierung des gegriffenen Objektes relativ zur virtuellen Hand notwendig sein. Damit diese Korrektur schnell und einfach ablaufen kann, wurde ein einfaches Loslassen und wieder Greifen eines zu positionierenden Objektes implementiert. Diese Funktionalität wurde sowohl für den Stylus™ wie auch für den Datenhandschuh implementiert. Bei der Nutzung des Stylus™ kann das gegriffene Objekt nur dann mit der virtuellen Hand bewegt werden, wenn der Bedienknopf des Stylus™ gedrückt wird. Wird dieser losgelassen, bleibt das gegriffene Objekt an seiner Position. Für den Datenhandschuh wird ständig eine Greifgeste abgefragt. Dafür werden nur wenige der Handschuhsensoren, die die Bewegung des Daumens und des Zeigefingers registrieren, abgefragt. Die Abfrage erfolgt unabhängig vom Nutzer und braucht daher nicht vor jeder Anwendung kalibriert werden.

Wird für ein gegriffenes Objekt ein Nachgreifen notwendig, befindet sich das Objekt trotzdem im gegriffenen Zustand. Die dynamische Umstrukturierung des VR-Szenengrafen, die beim Start des Greifvorganges durch den Interaktionsmanager durchgeführt wurde, bleibt erhalten. Der Interaktionsmanager

[106] In der Automobilindustrie wird oft ein rechtshändiges Fahrzeugkoordinatensystem genutzt, bei dem die Z-Achse nach oben zeigt, die X-Achse in Richtung Fahrzeugheck und die Y-Achse in Richtung rechte Fahrzeugseite. Dieses Koordinatensystem wird auch in der VR im Automobilbereich, speziell bei der Volkswagen AG, als Weltkoordinatensystem eingesetzt.

bekommt über das Nachgreifen keine Information vom VR-System. Dies führt schließlich zu einer Performancesteigerung.

6.2. Die Anwendungsvorbereitung

Für die Leitungsverlegung in einer virtuellen Umgebung und deren Akzeptanz ist der für die Anwendungsvorbereitung notwendige Aufwand von erheblicher Bedeutung. Dabei gilt: Je geringer der Aufwand, je größer die Praktikabilität und Akzeptanz der Anwendung. Die notwendigen Vorbereitungsschritte wurden bereits in Abschnitt 5.3. in Abhängigkeit vom jeweiligen Anwendungsziel beschrieben. Dennoch soll an dieser Stelle ein Überblick über die einzelnen Konfigurationsschritte und -dateien gegeben werden.

In Abbildung 64 wird das System in die zwei Hauptkomponenten Interaktionsmanager und VR-System unterteilt. Für beide Komponenten sind unterschiedliche Vorbereitungsschritte und Konfigurationsdateien zu pflegen.

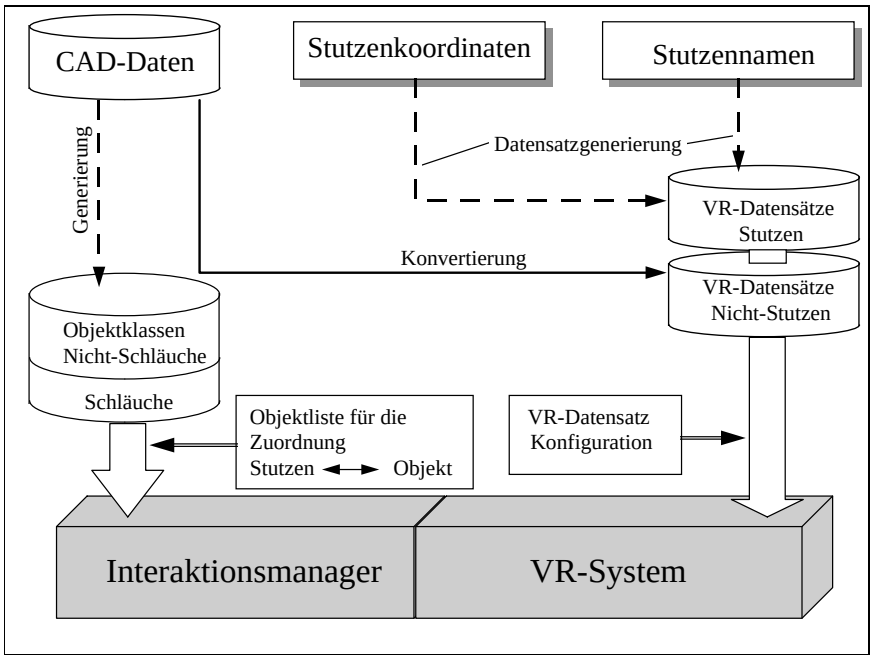


Abbildung 64: Anwendungskonfiguration für die Leitungsverlegung

Für das VR-System werden die VR-Datensätze als Eingabedaten genutzt. Es wird zwischen den Stutzen und den Nicht-Stutzen unterschieden. Die Erzeugung der Datensätze der Nicht-Stutzen erfolgt über Konvertierungswerkzeuge aus den CAD-Daten, wie es in Abschnitt 3.3.1. beschrieben wurde. Für die Stutzendatensätze müssen die Koordinaten des Stutzenvektors bekannt sein. Dann können sie über einen Texteditor direkt im genutzten VR-Datenformat (FHS-Format) definiert werden. Eine Alternative ist die Erzeugung dieser Geometrien im CAD-System und deren Konvertierung. Aber auch in diesem Fall müssen Anpassungen direkt im Datenformat vorgenommen werden. Die Namensgebung und Struktur der Objekte muß definiert werden. Dafür kann ein

Hierarchieeditor angewendet werden, der das Inventor-Datenformat unterstützt.

Die Definition der gesamten virtuellen Szene erfolgt in einem Konfigurationsfile. In diesem File sind alle VR-Datensätze aufgelistet, die beim Start der Anwendung eingeladen werden müssen. Das Konfigurationsfile wird mit einem Texteditor bearbeitet.

Für den Interaktionsmanager müssen die Objektklassen definiert werden, die den in das VR-System eingeladenen Objekten entsprechen. In den Objektklassen muß die Anzahl der zum Objekt zugeordneten Stutzen definiert werden. Diese Definition erfolgt im CLIPS-Code. Ein Grundgerüst steht bei der Definition zur Verfügung, in das die entsprechenden Eintragungen getätigt werden müssen. So wird verhindert, daß die Definition der Klassen die Kenntnis der CLIPS-Syntax voraussetzt. Die Anzahl der notwendigen Definitionen ist von der Anzahl der in der Umgebung abgebildeten Stutzen abhängig. Es wird zwischen den Schläuchen und den Nicht-Schläuchen unterschieden. Sollen auch die Schlauchklassen definiert werden, dann ist der in Abschnitt 4.3.5. erläuterte Verbindungsgraf zu definieren. Es sei an dieser Stelle nochmals angemerkt, daß für die Modellierung von Leitungen diese Definition nicht erforderlich ist.

Die Anwendungsvorbereitung wird durch die Zuordnung der geometrischen Objekte zu ihren Objektklassen und der Stutzen zu ihren Vaterobjekten abgeschlossen. Diese Zuordnung erfolgt in der in Abschnitt 4.3.8.2. beschriebenen Objektliste, die ebenfalls als Textdatei gepflegt wird.

Im Anhang sind die einzelnen Konfigurationsdateien anhand eines Beispieles dargestellt.

Wie oben beschrieben, erfolgen die Erzeugung der Stutzen wie auch der Objektklassen des Interaktionsmanagers größtenteils durch einfache Texteingaben in einem Editor. Das gleiche gilt für die in Abschnitt 4.3.8.2. beschriebene Objektliste, über die die Initialisierungskomponente des Systems die notwendigen Informationen über die eingeladenen Objekte erhält. Die Pflege solcher Textdateien und Listen birgt die Gefahr einer hohen Fehleranfälligkeit und verlangt vom Anwender ein Spezialwissen über das System. Abhilfe dafür kann eine grafische Bedienungsoberfläche schaffen. Diese muß einfach zu bedienen sein und den Benutzer bei der Vorbereitung einer Anwendung aktiv unterstützen. Durch die Arbeit konnte gezeigt werden, welche zusätzlichen Informationen für die Anwendung des Systems notwendig sind. Die Entwicklung eines solchen Anwendungsvorbereitungswerkzeuges ist jedoch nicht Gegenstand dieser Arbeit und wird daher hier auch nicht weiter diskutiert.

6.3. Zusammenfassung

In diesem Kapitel wurden praktische Erfahrungen diskutiert, die mit dem beschriebenen System gesammelt wurden. Dabei wurde zunächst auf die Anwenderschnittstelle hardwareseitig eingegangen. Es konnte festgestellt werden, daß die Wahl der eingesetzten Eingabegeräte stark vom visuellen

Ausgabegerät abhängig ist. Die Diskussion möglicher visueller Ausgabegeräte hat zu einer Empfehlung der Mehrseitenprojektion geführt. Dabei spielen die Eigenschaften der in dieser Arbeit diskutierten Anwendung eine erhebliche Rolle. Zu diesen Eigenschaften zählen die hohe Interaktivität, der zu visualisierende große Datenumfang und die umzusetzende direkte Manipulation der Objekte.

Es wurden die Eingabegeräte unter der Randbedingung der eingesetzten Mehrseitenprojektion betrachtet. Dabei konnte festgestellt werden, daß der Einsatz von *magnetischem Tracking*, *Spracheingabe mit fester Syntax* für die Befehlseingabe, *Stylus*^{TM} zur direkten Manipulation der Objekte und *Spacemouse* für eine Anfangspositionierung eine gute Kombination für die Leitungsverlegung darstellt. Es hat sich weiterhin gezeigt, daß ein einhändiges Arbeiten in der virtuellen Umgebung für diese Anwendung völlig ausreichend ist.

Im Rahmen der beschriebenen Arbeit wurden Funktionalitäten entwickelt, die nicht ausschließlich für die Leitungsverlegung in einer virtuellen Umgebung geeignet sind, sich aber als äußerst nützlich erwiesen haben. Auf diese Funktionalitäten wurde mit einer kurzen Beschreibung eingegangen. Dabei handelt es sich um

- die automatische und interpolierte Anpassung der virtuellen Anwendergröße
- die Manipulation der Handgeschwindigkeit, zum einen zur Vergrößerung des Interaktionsraumes (Vergrößerung) und zum anderen zur Verfeinerung der Bewegung (Verkleinerung)
- die Implementierung einer virtuellen Werkbank, für die Vereinfachung der Schlaucherzeugung und Verringerung der Anzahl der Sprachbefehle
- die Navigation in einer Ebene mit Hilfe des Sylus^{TM}
- die Integration eines einfachen „Nachgreifens“ für jegliche gegriffenen virtuellen Objekte unter Nutzung des Stylus^{TM} aber auch des Datenhandschuhs ohne nutzerabhängige Kalibrierung.

Einen weiteren Schwerpunkt bildete die Beschreibung der einzelnen Schritte, die für die Vorbereitung einer Anwendung zur Generierung und Verlegung von Leitungsobjekten notwendig sind. Es wurden die dafür notwendigen Informationen beschrieben und die damit in engem Zusammenhang stehenden Dateien. Diese Informationen bilden die Grundlage für ein Anwendungsvorbereitungswerkzeug, das die speziellen Erfordernisse der virtuellen Leitungsverlegung berücksichtigt.

7. Zusammenfassung und Ausblick

Durch die Analyse des Produktentstehungsprozesses eines Automobils konnten Schwierigkeiten aufgezeigt werden, die bei der konstruktiven Auslegung der Verschlauchung von Baugruppen und Aggregaten auftreten. Es konnte gezeigt werden, daß die Ursache dieser Schwierigkeiten hauptsächlich bei den eingesetzten CAx-Systemen zu finden ist. Das Finden des freien Bauraumes, die Generierung und Manipulation der Schläuche und Leitungen gestaltet sich sehr aufwendig und kompliziert. Die Gründe hierfür liegen in den zweidimensionalen Bedienmetaphern der Systeme und dem daraus resultierenden Visualisierungs- und Manipulationsdefizit. Zusammen mit den notwendigen Absicherungsmaßnahmen am physikalischen Prototypen führen diese Schwierigkeiten heute zu einer Verlagerung der Verschlauchung auf einen späten Entwicklungszeitpunkt. Das birgt die Gefahr von Qualitätseinbußen, Zeitverlusten und hohen Änderungskosten.

Anhand einer Analyse der aktuellen Entwicklungen auf dem Gebiet der virtuellen Realität konnte gezeigt werden, daß diese neue Form der Mensch-Maschine-Kommunikation Vorteile mit sich bringt, die die oben beschriebenen Systemnachteile aufheben. Allerdings wurden die folgenden Probleme identifiziert:

Die zur Verfügung stehende Ein- und Ausgabehardware funktioniert hinsichtlich Genauigkeit und Bedienungsergonomie nicht in der Perfektion, wie es für realitätsnahe Interaktionen erforderlich wäre. Weiterhin sind die verfügbaren Funktionalitäten in einer virtuellen Umgebung noch weit von der Realität entfernt. Ein hochinteraktives VR-System, das über die reine Datenvisualisierung hinausgeht und ein realitätsnahes Objektverhalten ermöglicht, bedingt ein wesentlich höheres Maß an Simulationsberechnungen und Informationen über die im System abgebildeten virtuellen Objekte. Damit ist zwangsläufig eine Erhöhung des Datenvorbereitungsaufwandes verbunden. Eine vollständige Simulation der in der Realität existierenden Zusammenhänge und Prozesse ist mit einem solchen System jedoch aus heutiger Sicht noch nicht realisierbar. Die übliche Abbildung der Daten in einem VR-System erlaubt nicht oder nur sehr eingeschränkt die direkte Rückführung von Ergebnissen in ein CAD-System.

Ein vorhandenes VR-Basissystem wurde daher um die folgenden Punkte erweitert:

CAD-Entities

Das VR-System wurde um CAD-Basisprimitive erweitert, die generiert und manipuliert werden können. Mit Hilfe dieser Basisprimitive können Leitungsverläufe in dem VR-System abgebildet werden. Sie bilden die Grundlage für den Ergebnisdatenaustausch zwischen dem VR-System und der CAx-Umgebung.

Stutzenbasierte Verbindungssimulation

Zusätzliche Datenstrukturen (Stutzen), die mögliche Verbindungsstellen der Objekte geometrisch kennzeichnen, bilden die Grundlage für realitätsnahe Interaktionen. Mit den Stutzen wurde bewußt auf die physikalische Simulation und Bewertung von Verbindungen verzichtet. Dadurch wird der notwendige Anwendungsvorbereitungsaufwand minimiert und die Systemperformance optimiert. Das Systemverhalten ist damit auch nicht von der Vollständigkeit der eingeladenen virtuellen Objekte abhängig. Die Umsetzung der stutzenbasierten Verbindungssimulation erfordert weder ein beidhändiges Arbeiten, noch den Einsatz einer Kraftrückkopplung. Sie schließt beides aber auch nicht aus. Sie verhält sich robust gegenüber den verschiedensten VR-Eingabegeräten.

Situationsabhängige Interaktionen

Durch die Implementierung von situationsabhängigen Interaktionen, deren Verhalten von den bestehenden Objektverbindungen ebenso abhängt wie vom jeweils selektierten Objekt, konnte ein realitätsnahes Verhalten erzielt werden. Dazu wurde ein Interaktionsmanager implementiert, der in der Lage ist eine aktuelle Situation zu analysieren und daraus resultierend das Systemverhalten zu beeinflussen. Die Vorgehensweise bei der Abarbeitung der Informationen ist dabei vergleichbar mit der Vorgehensweise des Anwenders im Falle einer Interaktion. Es werden zwei Phasen unterschieden: die Evaluierungsphase und die Execution-Phase. Das führt zu einer Verkleinerung der Differenz zwischen der Erwartungshaltung des Anwenders und dem tatsächlichen Systemverhalten und ist somit Voraussetzung für eine effektive Bedienung des Systems.

Semantische Objektinformationen

Ein realitätsnahes Verhalten bedingt Zusatzinformationen, die über eine rein geometrische Objektbeschreibung hinausgehen. Diese für das umgesetzte Systemverhalten notwendigen semantischen Informationen sind in einer zusätzlichen Objektrepräsentation abgelegt. In dieser Objektklassenhierarchie sind die für die Leitungsverlegung notwendigen Parameter und Attribute abgebildet, auf die der Interaktionsmanager zurückgreifen kann. Ihm steht damit eine auf die Leitungsverlegung abgestimmte und klar abgegrenzte Beschreibung eines Weltausschnittes zur Verfügung.

Durch die Systemerweiterungen wurde ein Modellierungswerkzeug entwickelt, das sich an dem zu modellierenden Objekt und an den aus der Realität bekannten Abläufen orientiert. Die Stutzen bilden die Basis für die Erzeugung und Analyse von Objektverbindungen und für die Umsetzung unterschiedlicher Verbindungsabläufe. Auf dieser Grundlage ist das System in der Lage, schnell und selbständig bestehende Verbindungen zu erkennen und in seinem Verhalten zu berücksichtigen. Anhand der Implementierung der gerichteten Greif- und Positionierungsinteraktion wie auch der Anpassung des Hüllgeometrieradius zusammengesetzter Schlauchobjekte wurden die Vorteile eines solchen Systemverhaltens gezeigt. Bei den verbindungsorientierten Interaktionen entsprechen die Abläufe innerhalb des Interaktionsmanagers den Abläufen der Informationsverarbeitung beim Anwender. Das System analysiert die sich dar-

stellende aktuelle Situation und reagiert abhängig von ihr. Weiterhin unterstützt das System den Anwender bei der Erzeugung und Trennung von Verbindungen zwischen verschiedenen Objekten weitestgehend unabhängig von der Präzision der genutzten VR-Eingabehardware.

Mit dem System konnte gezeigt werden, daß in einer VR-Umgebung mit heute zur Verfügung stehender Hardware Leitungsobjekte effektiv modelliert werden können. Die Ergebnisse einer solchen Modellierung in einer virtuellen Umgebung sind über neutrale Geometrieschnittstellen in ein CAx-Umfeld rückführbar. Das ermöglicht heute die Endkonstruktion im CAD-System mit der Sicherheit, daß keine geometrische Kollision zwischen den Leitungsobjekten und der Umgebung existieren. Dennoch ist für die Absicherung der Konstruktion der Test am physikalischen Prototyp notwendig. Insbesondere kann das beschriebene System eine Montagesimulation der meist flexiblen Schläuche nicht leisten.

Mittelfristig ist zu erwarten, daß die Simulation flexibler Bauteile und damit auch die Simulation von Montagevorgängen am digitalen Prototypen möglich sein wird. Dafür wird eine spezielle Modellaufbereitung notwendig sein, die sowohl geometrische und materialspezifische Parameter als auch Informationen über mögliche Objektverbindungen beinhaltet. Die in dem in dieser Arbeit beschriebenen System für die virtuelle Leitungsverlegung enthaltenen semantischen Informationen über die Leitungsobjekte, ihre Klassifizierung wie auch die während der Anwendung gewonnenen Informationen über ihre Verbindungen zu den Umgebungsobjekten können somit die Grundlage für weitergehende Untersuchungen am digitalen Prototypen sein. Damit handelt es sich bei dem System um ein Modellierungswerkzeug, das nicht ausschließlich geometrische Objektbeschreibungen, sondern auch semantische Informationen über die Objekte und ihre Umgebung generiert. Seine Funktionalität reicht damit weit über die Möglichkeiten heute eingesetzter Modellierungswerkzeuge hinaus. Die Arbeit zeigt einen Weg auf, wie die in absehbarer Zukunft benötigten zusätzlichen Produktinformationen bereits im Modellierungsprozeß implizit erzeugt werden können. Diese Eigenschaft wird eine immer größere Bedeutung erlangen, da sie zur Verminderung der aufwendigen Anreicherung von Produktdatenbeschreibungen um semantische Informationen führt.

Werden die semantischen Informationen direkt im Modellierungssystem für die Simulationsmodellaufbereitung genutzt, können Modellierung und Absicherung sogar in einer Umgebung erfolgen und es kann zumindest teilweise auf eine Hardwareabsicherung verzichtet werden. Die Erweiterung des beschriebenen Systems um eine Simulationskomponente für flexible Schläuche wird daher Gegenstand zukünftiger Arbeiten sein.

Die mit dem System gesammelten Erfahrungen und die hier beschriebenen Ergebnisse der Arbeit werden in ein System einfließen, das im Bereich der Leitungsgenerierung und -manipulation am digitalen Prototypen kommerziell eingesetzt werden wird.

Literatur

[AsDa95] Astheimer, Peter; Dai, Fan; Felger, Wolfgang; Göbel, Martin; Haase, Helmut; Müller, Stefan; Ziegler, Rolf: „Virtual Design II - an Advanced VR System for Industrial Applications“, in „Virtual Reality World´95“, S.337-363, Conference Documentation, IGD confernces & seminars, a devision of Computerwoche Verlag, Rheinstraße 28, D-80803 Munich/ Germany

[BRO90] Brooks, Frederick P.; Ouh-Young, Ming; Batter, James J.; Kilpatrick, P.Jerome: „Project GROPE-Haptic Displays for Scientific Visualization“, in „Computer Graphics, SIGGRAPH'90 Conference Proceedings“, ISSN 0097-8930, S.177-185

[BUCK95] Buck, Mathias; Grebner, Klaus: „Werkzeuge zur Interaktion in virtuellen Welten“, in „Modelling Virtual Worlds Distributed Graphics: Beiträge zum internationalen Workshop MVD'95“, 27.-28. Nov 1995 in Bad Honnef Bonn, S.57-68, Infix, 1995, ISBN 3-929037-98-X

[BüReiBi94] Büttner, K.; Reinemuth, J.; Birkhofer, H.: „Effizienzsteigerung bei der rechnerunterstützen Konstruktionstätigkeit durch die Synthese aus CAD und Virtueller Realität“ in [WarBu94] S. 323-337

[COLL95] Dave Collins: „Designing Object-Oriented User Interfaces“, The Benjamin/Cummings Publishing Company, Inc., 390 Bridge Parkway, Redwood City, CA 94065, ISBN 0-8053-5350-X, S. 212-217

[Com2/93] Zeitschrift „Computer“ Februar-Ausgabe 1993

[ComGr6/93] Zeitschrift „Computer & Graphics“ Ausgabe November/Dezember 1993

[DaFelFr] Fan Dai; W. Felger; Th. Frühauf; M. Göbel; D. Reiners; G. Zachmann: „Virtual Prototyping Examples for Automotive Industries“; http://www.igd.fhg.de/www/igd-a4

[DAI98] Fan Dai (Editor): ”Virtual Reality for Industrial Applications”, Springer Verlag Berlin Heidelberg 1998, ISBN 3-540-63348-0

[DEU97] Deussen, O. Lorenz, P:„Simulation & Animation ´97”, Proceedings der Tagung vom 6.-7.März 1997 der Otto-von-Guericke-Universität Magdeburg, ISBN 1-56555-111-7

[DIX93] Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale: „Human-Computer Interaction“, Prentice Hall International (UK) Limited 1993, ISBN 0-13-458266-7 (hbk.) ISBN 0-13-437211-5 (pbk.)

[DUBBEL97] Herausgeber: Beitz, W.; Grote, K.-H.: „Dubbel - Taschenbuch für den Maschinenbau“, 19. Auflage; Springer Verlag 1997

[EUGR96] The International Journal of the Eurographics Association: „Computer Graphics forum“, Volume 15 Nr. 3 Conference Issue Futuroscope-Poitiers, France; August 26-30, 1996; ISNN 0167-7055

[FARIN94] Farin, Gerald: „Curves and Surfaces for CAGD. A practical Guide“, 3rd Ed., dt. Übersetzung von Hans J. Wolters; Friedrich Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/ Wiesbaden, 1994; ISBN 3-528-16542-1

[FHS98] Fraunhofer Institut für Graphische Datenverarbeitung (FhG IGD) in Darmstadt: „Syntax description of the Fraunhofer VR-data-exchange format“ (FHS 7/1/98); Im Lieferumfang des Systems Virtual Design II enthalten

[GEHR98] Gehrke, Dr.-Ing. U.; Scheibler, Dipl.-Ing. M.: „Ein effektives Produktdatenmanagement - Rückgrat für die virtuelle Produktentwicklung“ in [VDI1435] S.13-30

[GIARR] Giarratano, Joseph C.: „CLIPS - C Language Integrated Production System Version 6.03 User´s Guide, Reference Manual Volume I, II, III“; NASA Lyndon B. Johnson Space Center Information Systems Directorate; Software Technology Branch

[GiarRile] Giarratano, Joseph C; Riley, Gary: „Expert Systems: principles and programming“; PWS Publishing Company, Boston; ISBN 0-534-93744-6

[GMDPM96] Gesellschaft für Mathematik und Datenverarbeitung (GMD): Pressemitteilung September 1996; http://www.gmd.de/de/presse-archiv/PrMitt9609.html

[GOMES97] Gomes de Sá, A.; Kress, H.; Müller, St.: „Digital Mock-Up in der Einbau und Montagesimulation“, Tagungsband Systems ´97, VDI Berichte Nr. 1357; VDI-Tagung „Neue Generation von CAD/CAM-Systemen - Erfüllte und Enttäuschte Erwartungen“; VDI-Verlag, S. 337-356; München, 28.-29.10. 1997

[GREEN] Green, Mark: „Virtual Reality User Interface: Tools and Techniques“; in „Computer Graphics Around the World: Computer Graphics International '90, Proceedings of the Conference“; London, Blenheim Online, 6-8 Nov 1990, S. 51-67

[IGES52] IGES Initial Graphics Exchange Specification, An American Standard; Digital Representation for Communication of Product Definition Data US PRO/IPO-100, IGES5.2, U.S. Product Data Association, Copyright © November 1993 by the IGES/PDES Organization

[IWA90] Iwata, Hiroo: „Artificial Reality with Force-Feedback: Development of Desktop Virtual Space with Compact Master Manipulator“, in „Computer Graphics, SIGGRAPH'90 Conference Proceedings“, ISSN 0097-8930, S.165-170

[JUNG96] Jung, Bernhard; Wachsmuth, Ipke: „Ein wissensbasiertes System für die 3D-computergraphische Montage-Simulation“, in Ruland, D. (ed.): „Verteilte und intelligente CAD-Systeme: Tagungsband CAD´96“ (S. 107-119), Bonn: Gesellschaft für Informatik; Kaiserslautern/Saarbrücken: DFKI, 1996

[JUNG97] Jung, Bernhard: „Wissensverarbeitung für Montageaufgaben in virtuellen und realen Umgebungen“, Sankt Augustin: Infix, Hundt Druck GmbH, Köln 1997, ISBN 3-89601-157-X

[JUNG99] Jung, B.; Milde, J.-T.; Uthmann, Th.: „Intelligente Virtuelle Umgebungen“; Workshop: 23. Deutsche Jahrestagung für Künstliche Intelligenz, 13.-15.9.1999 Bonn; „Report - Situierte Künstliche Kommunikatoren 1999/04“; Bielefeld: SFB 360; ISSN 0946-7572

[KALAW93] Kalawsky,Roy S.: „The Science of Virtual Reality and Virtual Environments“, Addison-Wesley Publishing Company Inc., ISBN 0-201-63171-7

[KESS95] Kesslew, G. Drew; Hodges, Larry F.;Walker, Neff: „Evaluation of the CyberGlove as a Whole-Hand Input Device“, in „ACM Transactions on Computer-Human Interaction“, Vol.2, No. 4, December 1995, S. 263-283, ISSN 1073-0516

[KRAUSE98] Krause, Prof. Dr.-Ing. F.-L.; Tang, Dr.-Ing. T.; Ahle, Dipl.-Ing. U.: „Virtuelle Produktentstehung“ in [VDI1435] S.1-11

[KUHN95] Kuhn, Kühnapfel, Deussen: „Echtzeitsimulation deformierbarer Objekte zur Ausbildungsunterstützung in der Minimal-Invasiven Chirurgie“, in „Modelling Virtual Worlds Distributed Graphics: Beiträge zum internationalen Workshop MVD'95“, 27.-28. Nov 1995 in Bad Honnef Bonn, S.169-177, Infix, 1995, ISBN 3-929037-98-X

[LENZ98] Lenzmann, Britta: „Benutzeradaptive und multimodale Interface-Agenten“; Infix 1998, Dissertationen zur künstlichen Intelligenz, Band184; ISBN 3-89601-184-7

[MadGer96] Madritsch, F.; Gervautz, M.: „CCD-Camera Based Optical Beacon Tracking for Virtual and Augmented Reality“ in [EUGR96] S. C-207-C-216

[RAB96] Rabätje, Ralf: „Integration von CAD-Basisfunktionen in einer Virtual-Reality-Umgebung“; Diplomarbeit an der Technischen Universität Braunschweig, Institut für Nachrichtentechnik, Betreuer: Prof. Dr.-Ing. E. Paulus; Abgabe 1996

[REZ96] Reznik, Dan; Laugier, Christian: „Dynamic Simulation And Virtual Control of a Deformable Fingertip“, in „Proceedings of the IEEE International Conference on Robotics and Automation“, Minneapolis, Minnesota April 1996, S.1669-1674

[RIEDEL98] Riedel, Oliver; Breining, Ralf; Häfner, Ulrich; Blach, Roland: „Use of Immersive Projection Environments for Engineering Tasks“;in „Von der virtuellen Realität zum Rapid Prototyping“, Proceedings der 3. Euroforum-Fachtagung zur schnellen Produktentwicklung, 10./11. November 1998, Lindner Hotel Rheinstern, Düsseldorf; Herausgeber: EUROFORUM Deutschland GmbH Düsseldorf

[ROHR82] Rohr, B.; Wiele, H. (Hg.): „Lexikon der Technik“, VEB Bibliographisches Institut Leipzig, 1982, 1. Auflage, Lizenz-Nr. 433-130/69/82, LSV 3007, Best.-Nr. 577 209 8

[SGI98] SiliconGraphics Computer Systems: „Onyx2 Technical Specifications“; http://www.sgi.com

[SHNEI92] Shneiderman, Ben: „Designing the user interface: Strategies for effective HCI“, Addison Wesley 1992 (2nd Edition), ISBN 0-201-57286-9

[SNYD95] Snyder, John M.: „An Interactive Tool for Placing Curved Surfaces without Interpenetration“, in „Computer Graphics, SIGGRAPH’90 Conference Proceedings“, Addison-Wesley ISBN 0-201-84776-0, S.209-218

[Spec10/93] Zeitschrift IEEE Spectrum Ausgabe 10 - 1993

[Spies94] Spies, S.: „Management von Automobilentwicklungen - Umsetzung bereichsübergreifender Integration“; Dissertation der Hochschule St. Gallen für Wirtschafts-, Rechts- und Sozialwissenschaften zur Erlangung der Würde eines Doktors der Wirtschaftswissenschaften; Dissertation Nr. 1561; DIFO Druck GmbH; Bamberg, 1994

[SUR92] Mark C. Surles: „An Algorithm With Linear Complexity For Interactive Physically-Based Modelling Of Large Proteins“, in „Computer Graphics SIGGRAPH’92 Conference Proceedings“, ISSN 0097-8930, S.221-230

[SYM97] Symietz, M.: „Grundlegende Interaktionen mit virtuellen Prototypen”; in [DEU97] S.24-35

[SYM98] Symietz, M.: „Virtuelle Leitungsverlegung an einem digitalen Fahrzeugmodell“; Praxis-Forum-Tagung Bad Nauheim 14./15.5.98: „Synthese von Berechnung, Simulation und Visualisierung“

[SYM99] Symietz, M.; Rabätje, R.: „Einsatz wissensbasierter Methoden bei der VR-Modellgenerierung“ in [JUNG99]

[VDI1435] Herausgeber: Verein Deutscher Ingenieure: „Prozeßketten für die virtuelle Produktentwicklung in verteilter Umgebung“; VDI-Berichte 1435; ISBN 3-18-091435-1; VDI-Verlag 1998

[VDI98] Herausgeber VDI: „Plastics in Automotive Engineering - New Applications Shaing the Future“; VDI Düsseldorf 1998

[WarBu93] Herausgeber: Warnecke, H.J.; Bullinger, H.-J.: „IPA-/IAO-Forum Virtual Reality '93, Anwendungen und Trends“; Springer-Verlag 1993

[WarBu94] Herausgeber: Warnecke, H.J.; Bullinger, H.-J.: „IPA-/IAO-Forum Virtual Reality '94, Anwendungen und Trends“; Springer-Verlag 1994

[WA95] Wachsmuth, Ipke; Lenzmann, Britta; Jung, Bernhard: „Communicating with Virtual Environments, A Survey of Recent Work at the University of Bielefeld“, in „Modelling Virtual Worlds Distributed Graphics: Beiträge zum internationalen Workshop MVD’95“, 27.-28. Nov 1995 in Bad Honnef Bonn, S.93-97, Infix, 1995, ISBN 3-929037-98-X

[WERNE] Wernecke, Josie: „The Inventor Mentor“; Addison - Wesley 1995; ISBN 0-201-62495-8

[WITKIN87] Witkin, A.; Fleischer, K.; Barr, A.: „Energy Constraints On Parametrized Models“; in „Computer Graphics, Volume 21, SIGGRAPH’87 Conference Proceedings“, ACM-0-89791-227-6/87/007/0225, S.225-232

[WITTIG] Wittig, A.: „Vektoren in der analytischen Geometrie“; Friedr. Vieweg & Sohn GmbH, Verlag, Braunschweig 1968; Best.-Nr. 0812

[WooNeiDa] Woo, M.; Neider, J.; Davis, T.: „OpenGL Programming Guide“; second Ed.; ISBN 0-201-46138-2; Addison Wesley Developers Press

[YASU96] Yasuharu, Kunii; Hashimoto, Hideki: „Virtual Environment For Haptic Interfaces In Robotic Network System“, in „Proceedings of the IEEE International Conference on Robotics and Automation“, Minneapolis, Minnesota April 1996, S.545-550

[ZachColl] Zachmann, G.: „Precise and high-speed collision detection in interactive realtime visualization systems“, Diplomarbeit, Technische Hochschule Darmstadt, Fachbereich Informatik, 1994; ftp://ftp.igd.fhg.de/outgoing/zach/collision.ps.Z

[ZACH96] Zachmann, G.: „A Language for Describing Behavior of and Interaction with Virtual Worlds“, in „Selected Readings in Computer Graphics 1996“, Fraunhofer IRB Verlag 1997, ISBN 3-8167-5142-3, ISSN 0948-3950

[ZACH97] Zachmann, G.: „Distortion Correction of Magnetic Fields for Position Tracking“ in Proc. Computer Graphics International (CGI´97), June 23-27 1997, Hasselt and Diepenbeek, Belgium

[ZIEG96] Ziegler, Rolf: „Haptic Displays-How Can We Feel Virtual Environments?“; in „Selected Readings in Computer Graphics 1996“, Fraunhofer IRB-Verlag 1997, ISSN 0948-3950, ISBN 3-8167-5142-3, S. 155-179

[ZIEG98] Ziegler, Rolf: „System zum integrierten Einsatz von haptischen Displays in virtuellen Umgebungen“; Dissertation zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.) an der Technischen Hochschule Darmstadt, Fachbereich Informatik, ISBN 3-8265-3743-3, Shaker-Verlag 1998

[ZIMM97] Zimmermann, P.; Symietz, M.; Starke, A.; Schulze, M.; Rabätje, R; P Purschke, F.: „Virtual Reality (VR) - New Methods of Improving and accelerating Vehicle Development“; Proceedings of the 5th International Conference Florence ATA 1997

[ZIMM98] Zimmermann, Peter; Purschke, F.; Rabätje, R.; Schulze, M.; Symietz, M., Tegel, O.: „Virtual Reality (VR) - Research and Application at Volkswagen“; in [VDI98] S. 43 - 68

[ZIMM1/98] Zimmermann, P.; Symietz, M.; Starke, A.; Schulze, M.; Rabätje, R; P Purschke, F.:„Virtual Reality (VR) - New Methods of Improving and accelerating Vehicle Development“; in [DAI98] S. 105-122

[ZIMM2/98] Zimmermann, P.; Purschke, F.; Rabätje, R.; Schulze, M.; Symietz, M.; Tegel, O.: „Virtual Reality (VR) – Forschung und Anwendung bei Volkswagen“; Proceedings der Tagung „Digitale Prototypen: Neue Werkzeuge für die innovative Produktentwicklung“ 24./25.6.1998 am Fraunhofer Institut für Graphische Datenverarbeitung, Darmstadt

Anhang

A) Sprach-Syntaxgraf

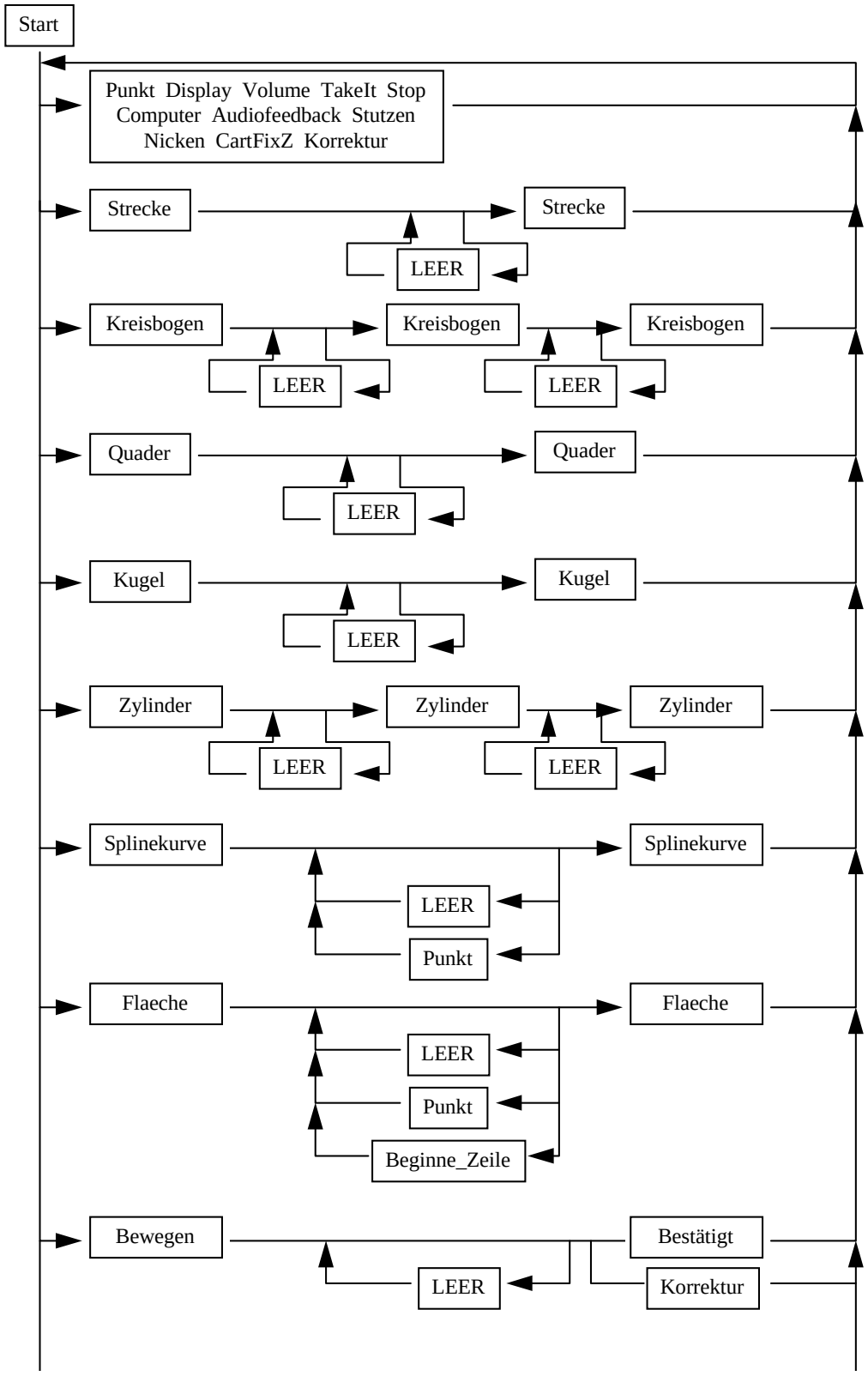
Die in der Leitungsverlegung eingesetzte Spracherkennung nutzt eine Syntaxbeschreibung zur Eingrenzung der jeweils zu erkennenden Wortmenge. Die genutzte Syntax wird hier im Anhang A in Form eines Grafen dargestellt.

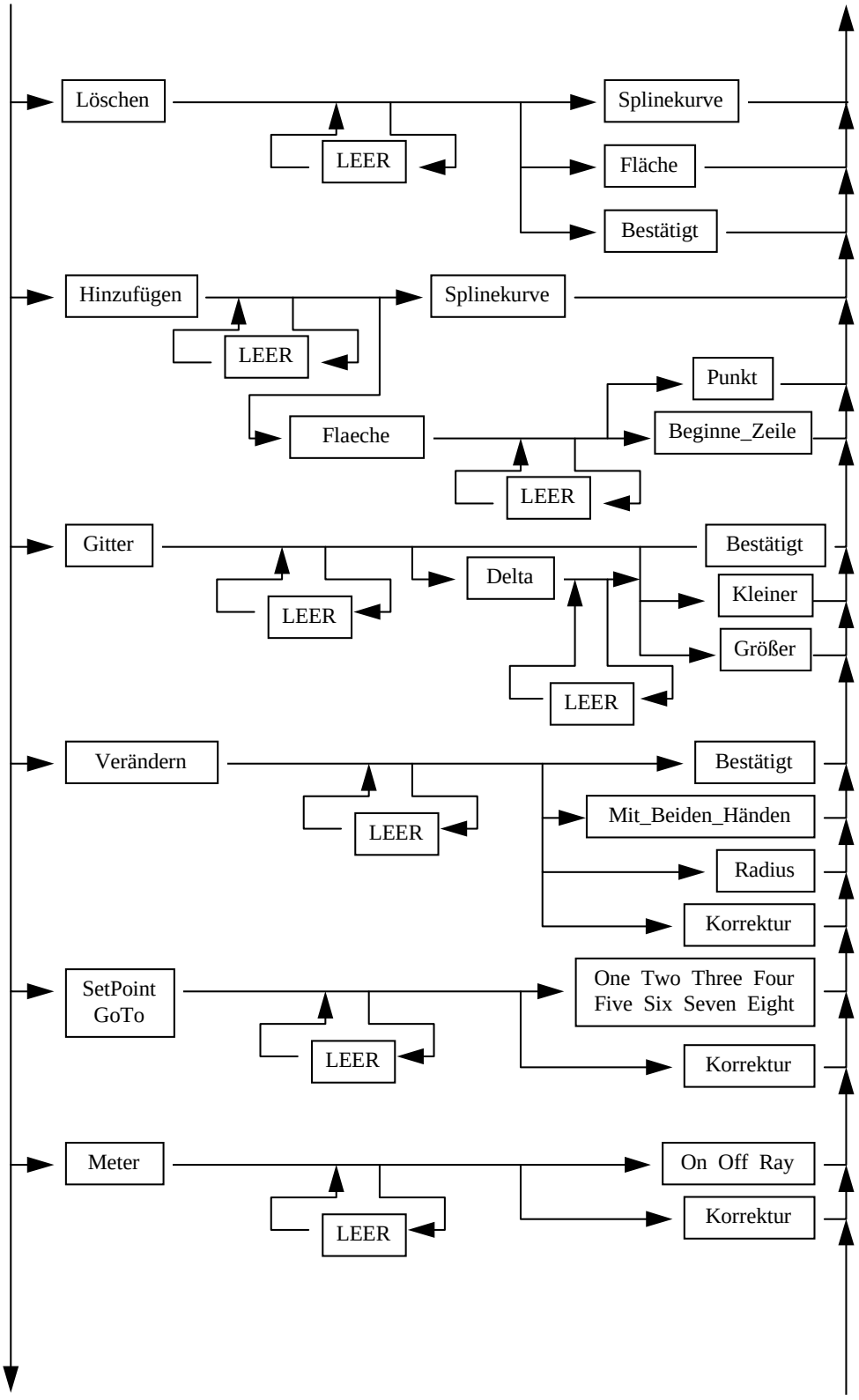
Erläuterung der folgenden Abbildungen:

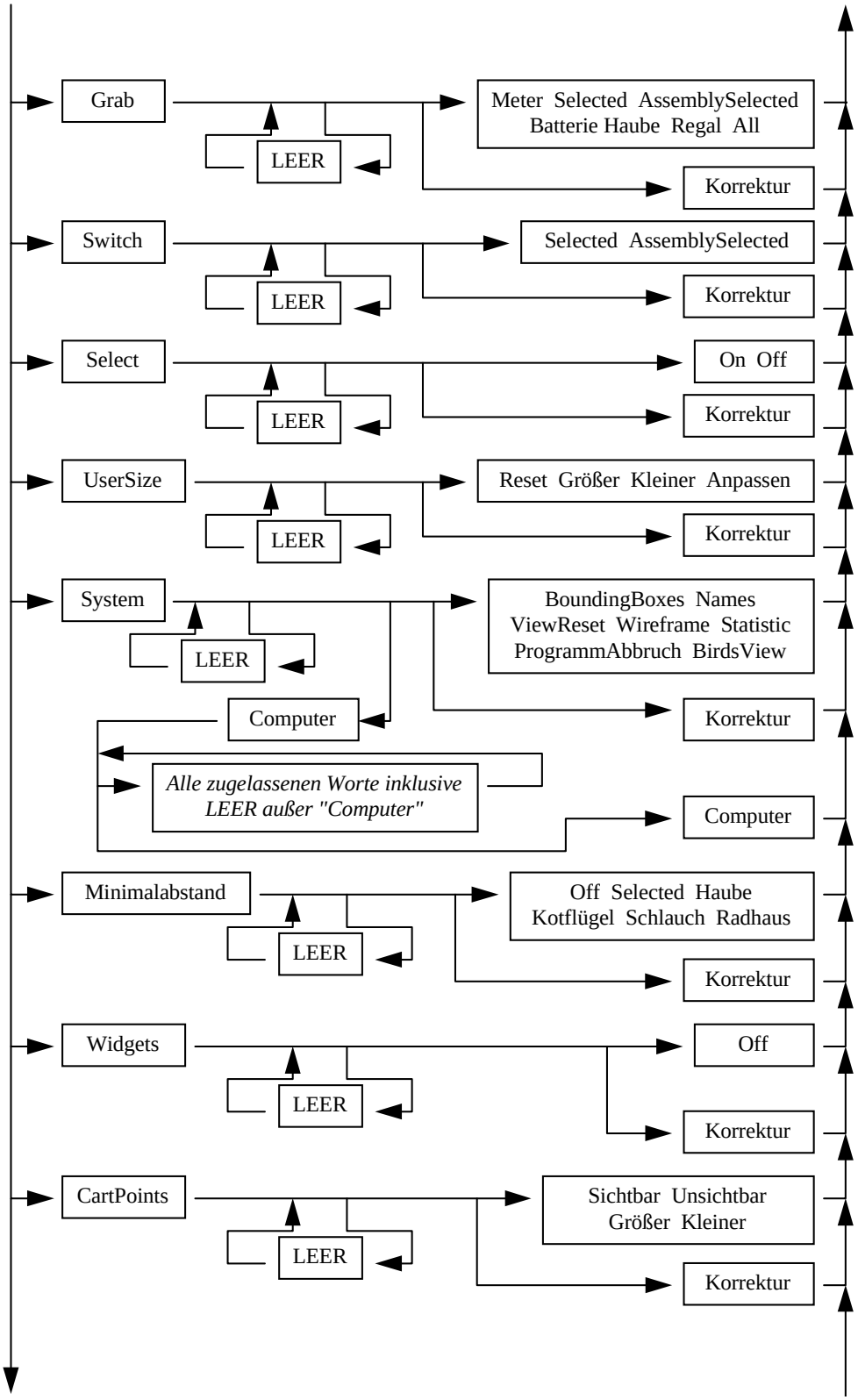
Beim Knoten „LEER“ handelt es sich um ein Spezialwort, welches durch die Spracherkennungshardware erkannt wird, wenn eine Pause eintritt.

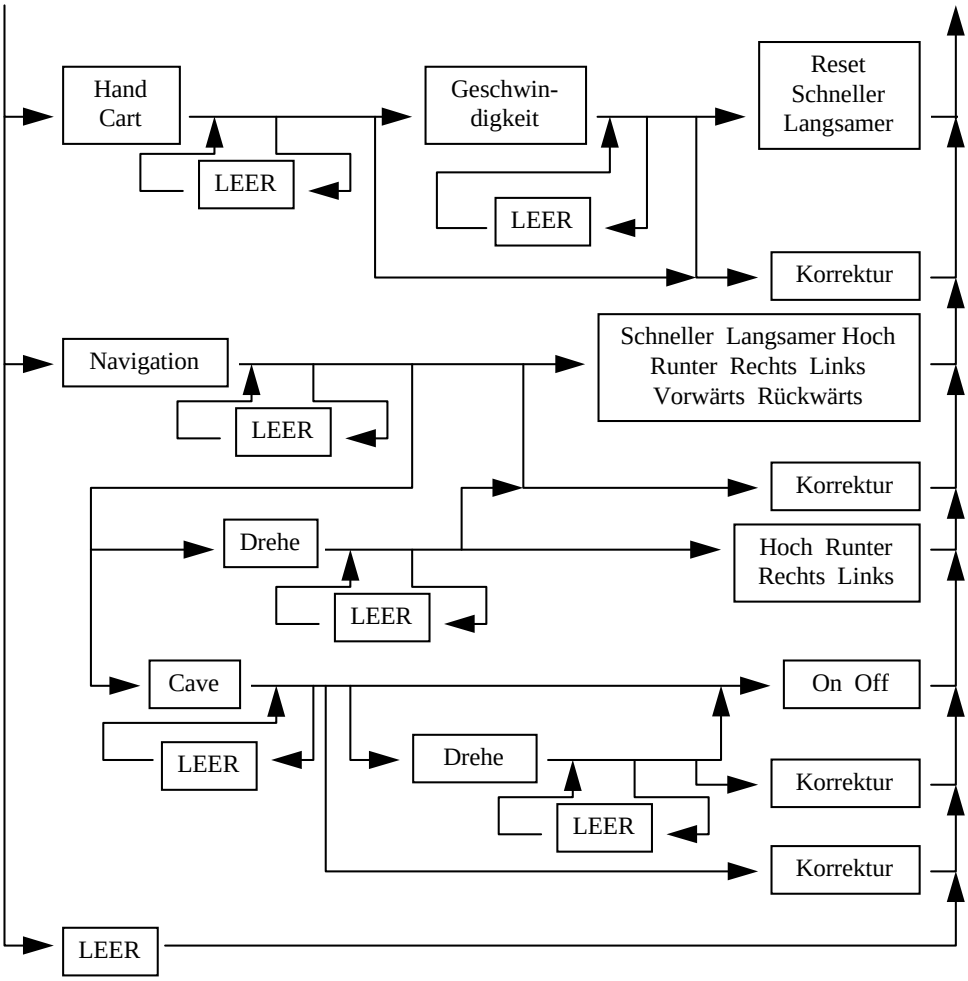
Mehrere Wörter in einem Knoten sind als Alternative zu verstehen. Wird ein Wort aus der Menge erkannt, wird zum nächsten Knoten übergegangen.

Über die Wortfolge „System Computer“ wird die Spracherkennung in eine Schleife gesetzt, in der sie versucht alle möglichen Worte in einer flachen Hierarchie zu erkennen. Erst bei Erkennen eines speziellen Begriffs, in diesem Fall „Computer“, wird die Schleife wieder verlassen. Damit wird auf Anwendungsseite die Spracheingabe über einen Sprachbefehl abgeschaltet und wieder eingeschaltet.


Start
Punkt Display Volume TakeIt Stop
Computer Audiofeedback Stutzen
Nicken CartFixZ Korrektur
Strecke
Strecke
LEER
Kreisbogen
Kreisbogen
Kreisbogen
LEER
LEER
Quader
Quader
LEER
Kugel
Kugel
LEER
Zylinder
Zylinder
Zylinder
LEER
LEER
Splinekurve
Splinekurve
LEER
Punkt
Flaeche
Flaeche
LEER
Punkt
Beginne_Zeile
Bewegen
Bestätigt
LEER
Korrektur


Löschen
LEER
Splinekurve
Fläche
Bestätigt
Hinzufügen
LEER
Splinekurve
Punkt
Flaeche
Beginne_Zeile
LEER
Gitter
LEER
Delta
Bestätigt
Kleiner
Größer
LEER
Verändern
LEER
Bestätigt
Mit_Beiden_Händen
Radius
Korrektur
SetPoint
GoTo
LEER
One Two Three Four
Five Six Seven Eight
Korrektur
Meter
LEER
On Off Ray
Korrektur


Grab
LEER
Meter Selected AssemblySelected
Batterie Haube Regal All
Korrektur
Switch
LEER
Selected AssemblySelected
Korrektur
Select
LEER
On Off
Korrektur
UserSize
LEER
Reset Größer Kleiner Anpassen
Korrektur
System
LEER
BoundingBoxes Names
ViewReset Wireframe Statistic
ProgrammAbbruch BirdsView
Computer
Korrektur
Alle zugelassenen Worte inklusive
LEER außer "Computer"
Computer
Minimalabstand
LEER
Off Selected Haube
Kotflügel Schlauch Radhaus
Korrektur
Widgets
LEER
Off
Korrektur
CartPoints
LEER
Sichtbar Unsichtbar
Größer Kleiner
Korrektur


Hand
Cart
LEER
Geschwin-
digkeit
LEER
Reset
Schneller
Langsamer
Korrektur
Navigation
LEER
Schneller Langsamer Hoch
Runter Rechts Links
Vorwärts Rückwärts
Korrektur
Drehe
LEER
Hoch Runter
Rechts Links
Cave
LEER
Drehe
LEER
On Off
Korrektur
Korrektur
LEER

B) Konfigurationsdateien

LoadObjects:

```
;*********************************************************************
; Laden von Semantikobjekten gesteuert durch eine VR-Umgebung
; Version 0.2
; Author: Michael Symietz
;*********************************************************************
; Zum Aufbau der Semantikobjekte ist es erforderlich, den interessierenden
; Objektnamen, wie er im VR-System existiert, in die Liste
; "Interessierende_Objekte" einzutragen. Die zugehoerige Objektklasse
; wird ebenfalls in dieser Liste gefuehrt. Mit Hilfe dieser Information
; werden die Instanzen im System entspreched dem VR-System generiert.
; Der Aufbau der richtigen Hierarchie erfolgt im Modul JoinSplit.clp
;
;*********************************************************************
;#####################################################################
; Eintrag der relevanten Objekte mit den dazugehoerigen Klassen, die durch ein
; assert-Kommando dem System bekannt gemacht werden sollen

(deffacts Objects_of_interest        ;Objekt                              Objekt-Klasse
;_____________________________________________________________________________
(Interessierende_Objekte
../verbinder/abzw_1j0_122_291_a_02.iv.fhb.fhs                Kuehler_Ruecklauf_Abzw_Klasse
assKuehlerruecklauf_6Q0_122_051_G-S1                         Stutzen
KR_Endobjekt                                                 KR_Endobjekt_Klasse
assKuehlerruecklauf_6Q0_122_051_G-S0                         Stutzen
../verbinder/1c0_121_438_verteilerrohr_02.iv.fhb.fhs KV_T_Stueck_Klasse
../verbinder/1c0_122_291_02.iv.fhb.fhs                       KV_V_Klasse
../verbinder/6q0_121_619-t-stueck_02.iv.fhb.fhs              Entl_TS_Klasse
../verbinder/6q0_121_447_c-entl-rohr_02.iv.fhb.fhs           EntlR_Klasse
KV_T2_Endobjekt                                              KV_T2_Endobjekt_Klasse
ZK_Endobjekt                                                 ZK_Endobjekt_Klasse
HZR_Endobjekt                                                HZR_Endobjekt_Klasse
AusgleichB_Endobjekt                                         AusgleichB_Endobjekt_Klasse
assKuehlervorlauf_6Q0_121_101_K_TEIL2-S1                     Stutzen
assKuehlervorlauf_6Q0_121_101_K_TEIL2-S0                     Stutzen
assKuehlervorlauf_6Q0_121_101_K_TEIL1-S1                     Stutzen
assKuehlervorlauf_6Q0_121_101_K_TEIL1-S0                     Stutzen
assAM_K_VORLAUF_6Q0_121_447_C-S0                             Stutzen
assAM_K_VORLAUF_6Q0_121_447_C-S1                             Stutzen
assAM_ZYL_KOPF_6Q0_121_447_C-S0                              Stutzen
assAM_ZYL_KOPF_6Q0_121_447_C-S1                              Stutzen
assAM_HZ_ROHR_6Q0_121_447_C-S0                               Stutzen
assAM_HZ_ROHR_6Q0_121_447_C-S1                               Stutzen
assAM_AUSGLEICHBEH_6Q0_121_447_C-S1                          Stutzen
assAM_AUSGLEICHBEH_6Q0_121_447_C-S0                          Stutzen
assAM_ENTLUEFT-ROHR_6Q0_121_447_C-S0                         Stutzen
assAM_ENTLUEFT-ROHR_6Q0_121_447_C-S1                         Stutzen
)
)
```

```
;#############################################################################
; Objekthierarchie im Semantikbaum erstellen
; Ein Gesamtobjekt besteht aus mehreren Teilobjekten. Der Aufbau der
; Gesamtobjekte aus den entsprechenden Teilobjekten wird hier durchgefuehrt.
; Die Information darueber, welche Instanz aus welchen Teilinstanzen aufgebaut
; wird, ist in der folgenden Liste „Object_Hierarchy“  als "Objekthierarchie"-Faktum einzutragen
;
;
(deffacts Object_Hierarchy          ;Objekt-Father              Objekt-Children
;Name   Klasse                      Stutzen0          Stutzen1          Stutzen2          ...)
;_____________________________________________________________________________________
;
   (Objekthierarchie
../verbinder/abzw_1j0_122_291_a_02.iv.fhb.fhs                  Kuehler_Ruecklauf_Abzw_Klasse
assKuehlerruecklauf_6Q0_122_051_G-S1 )
   (Objekthierarchie
KR_Endobjekt                                                  KR_Endobjekt_Klasse
assKuehlerruecklauf_6Q0_122_051_G-S0 )
   (Objekthierarchie
KV_T2_Endobjekt                                               KV_T2_Endobjekt_Klasse          ass-
Kuehlervorlauf_6Q0_121_101_K_TEIL2-S1)
   (Objekthierarchie
ZK_Endobjekt                                                  ZK_Endobjekt_Klasse
assAM_ZYL_KOPF_6Q0_121_447_C-S0)
   (Objekthierarchie
HZR_Endobjekt                                                 HZR_Endobjekt_Klasse
assAM_HZ_ROHR_6Q0_121_447_C-S0)
   (Objekthierarchie
AusgleichB_Endobjekt                                          AusgleichB_Endobjekt_Klasse
assAM_AUSGLEICHBEH_6Q0_121_447_C-S1)
   (Objekthierarchie
../verbinder/1c0_121_438_verteilerrohr_02.iv.fhb.fhs KV_T_Stueck_Klasse                 assKueh-
lervorlauf_6Q0_121_101_K_TEIL2-S0       assKuehlervorlauf_6Q0_121_101_K_TEIL1-S1 as-
sAM_K_VORLAUF_6Q0_121_447_C-S0 )
   (Objekthierarchie
../verbinder/1c0_122_291_02.iv.fhb.fhs                         KV_V_Klasse
assKuehlervorlauf_6Q0_121_101_K_TEIL1-S0)
   (Objekthierarchie
../verbinder/6q0_121_619-t-stueck_02.iv.fhb.fhs                Entl_TS_Klasse
assAM_K_VORLAUF_6Q0_121_447_C-S1          assAM_ZYL_KOPF_6Q0_121_447_C-S1    as-
sAM_HZ_ROHR_6Q0_121_447_C-S1          assAM_ENTLUEFT-ROHR_6Q0_121_447_C-S0)
   (Objekthierarchie
../verbinder/6q0_121_447_c-entl-rohr_02.iv.fhb.fhs             EntlR_Klasse
assAM_ENTLUEFT-ROHR_6Q0_121_447_C-S1    assAM_AUSGLEICHBEH_6Q0_121_447_C-S0)
)
```

Klassendefinition der Objekte der Oberklasse Nicht-Schläuche:

```
;*********************************************************************
; Nicht-Schlaeuche
;*********************************************************************

;******************** Beispielklasse*************************************
;(defclass Klassenname (is-a Obj_mit_Stutzen)
;                       (role concrete)
;                       (pattern-match reactive)
;
;;------------------- S0-CW-Inst ---------------------------------------
;   (slot S0-CW-Inst (create-accessor read-write)
;           (type INSTANCE-NAME)
;           (default [NIL])
;   )
;;------------------- S0-CW-Class --------------------------------------
;   (slot S0-CW-Class (create-accessor read-write)
;           (type SYMBOL)
;           (default NOCLASS)
;   )
;)
;
;
;(defmessage-handler Klassenname initinstance ()
;
;   (send ?self put-StutzenAnz 1)
;   (send ?self put-ConnectClass (create$ Klassenname(n) Verbindungsklasse))
;   (send ?self put-ConnectInstanceNr (create$ 1))
;   (printout t "Objekt " ": ...")
;   (send ?self print)
;   (printout t "...wurde initialisiert" crlf crlf)
;
;)
;
;;
; Kuehler-Ruecklauf
;

;******************** Kuehler_Ruecklauf_Abzw_Klasse ***********************************
(defclass Kuehler_Ruecklauf_Abzw_Klasse (is-a Obj_mit_Stutzen)
                       (role concrete)
                       (pattern-match reactive)

;------------------- S0-CW-Inst ---------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class --------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
)
```

```
(defmessage-handler Kuehler_Ruecklauf_Abzw_Klasse initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ KR-Schlauch_Klasse))
   (send ?self put-ConnectInstanceNr (create$ 1))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)

;****************** KR_Endobjekt_Klasse ************************************
(defclass KR_Endobjekt_Klasse (is-a Obj_mit_Stutzen)
                        (role concrete)
                        (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
)


(defmessage-handler KR_Endobjekt_Klasse initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ KR-Schlauch_Klasse))
   (send ?self put-ConnectInstanceNr (create$ 0))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)
;
;
; Kuehler-Vorlauf
;
;

;****************** KV_T2_Endobjekt_Klasse ************************************
(defclass KV_T2_Endobjekt_Klasse (is-a Obj_mit_Stutzen)
                        (role concrete)
                        (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
```

```
   (slot S0-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
)

(defmessage-handler KV_T2_Endobjekt_Klasse initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ KV_T2_Schlauch_Klasse))
   (send ?self put-ConnectInstanceNr (create$ 1))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)


;******************* KV_T_Stueck_Klasse ************************************
(defclass KV_T_Stueck_Klasse (is-a Obj_mit_Stutzen)
                  (role concrete)
                  (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
          (type INSTANCE-NAME)
          (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
;------------------- S1-CW-Inst ----------------------------------------
   (slot S1-CW-Inst (create-accessor read-write)
          (type INSTANCE-NAME)
          (default [NIL])
   )
;------------------- S1-CW-Class ----------------------------------------
   (slot S1-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
;------------------- S2-CW-Inst ----------------------------------------
   (slot S2-CW-Inst (create-accessor read-write)
          (type INSTANCE-NAME)
          (default [NIL])
   )
;------------------- S2-CW-Class ----------------------------------------
   (slot S2-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
)
(defmessage-handler KV_T_Stueck_Klasse initinstance ()

   (send ?self put-StutzenAnz 3)
```

```
   (send ?self put-ConnectClass (create$ KV_T2_Schlauch_Klasse xx xx))
   (send ?self put-ConnectInstanceNr (create$ 0 x x))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)


;******************** KV_V_Klasse ************************************
(defclass KV_V_Klasse (is-a Obj_mit_Stutzen)
                      (role concrete)
                      (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
)


(defmessage-handler KV_V_Klasse initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ xx))
   (send ?self put-ConnectInstanceNr (create$ x))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)

;******************** Entl_TS_Klasse ************************************
(defclass Entl_TS_Klasse (is-a Obj_mit_Stutzen)
                      (role concrete)
                      (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
;------------------- S1-CW-Inst ----------------------------------------
   (slot S1-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
```

```
   )
;------------------- S1-CW-Class ----------------------------------------
   (slot S1-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
;------------------- S2-CW-Inst -----------------------------------------
   (slot S2-CW-Inst (create-accessor read-write)
          (type INSTANCE-NAME)
          (default [NIL])
   )
;------------------- S2-CW-Class ----------------------------------------
   (slot S2-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
;------------------- S3-CW-Inst -----------------------------------------
   (slot S3-CW-Inst (create-accessor read-write)
          (type INSTANCE-NAME)
          (default [NIL])
   )
;------------------- S3-CW-Class ----------------------------------------
   (slot S3-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
)

(defmessage-handler Entl_TS_Klasse initinstance ()

   (send ?self put-StutzenAnz 4)
   (send ?self put-ConnectClass (create$ xx xx xx xx))
   (send ?self put-ConnectInstanceNr (create$ x x x x))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)
)
;******************** ZK_Endobjekt_Klasse ************************************
(defclass ZK_Endobjekt_Klasse (is-a Obj_mit_Stutzen)
                  (role concrete)
                  (pattern-match reactive)
;------------------- S0-CW-Inst -----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
          (type INSTANCE-NAME)
          (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
          (type SYMBOL)
          (default NOCLASS)
   )
)
(defmessage-handler ZK_Endobjekt_Klasse initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ xx))
   (send ?self put-ConnectInstanceNr (create$ x))
```

```
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)

;****************** HZR_Endobjekt_Klasse ************************************
(defclass HZR_Endobjekt_Klasse (is-a Obj_mit_Stutzen)
                        (role concrete)
                        (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
)


(defmessage-handler HZR_Endobjekt_Klasse initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ xx))
   (send ?self put-ConnectInstanceNr (create$ x))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)

;****************** AusgleichB_Endobjekt_Klasse ************************************
(defclass AusgleichB_Endobjekt_Klasse (is-a Obj_mit_Stutzen)
                        (role concrete)
                        (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
)
(defmessage-handler AusgleichB_Endobjekt_Klasse initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ xx))
   (send ?self put-ConnectInstanceNr (create$ x))
   (printout t "Objekt " ": ...")
```

```
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)


;******************** EntlR_Klasse ************************************
(defclass EntlR_Klasse (is-a Obj_mit_Stutzen)
                       (role concrete)
                       (pattern-match reactive)

;------------------- S0-CW-Inst ---------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
            (type INSTANCE-NAME)
            (default [NIL])
   )
;------------------- S0-CW-Class --------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
            (type SYMBOL)
            (default NOCLASS)
   )
;------------------- S1-CW-Inst ---------------------------------------
   (slot S1-CW-Inst (create-accessor read-write)
            (type INSTANCE-NAME)
            (default [NIL])
   )
;------------------- S1-CW-Class --------------------------------------
   (slot S1-CW-Class (create-accessor read-write)
            (type SYMBOL)
            (default NOCLASS)
   )
)


(defmessage-handler EntlR_Klasse initinstance ()

   (send ?self put-StutzenAnz 2)
   (send ?self put-ConnectClass (create$ xx xx))
   (send ?self put-ConnectInstanceNr (create$ x x))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)
```

Klassendefinition der Objekte der Oberklasse Schläuche:

```
;*****************************************************************
; Beispielklasse
;*****************************************************************
;******************** KR-Schlauch_Klasse ************************************
;(defclass Klassenname (is-a Schlauch)
;                       (role concrete)
;                       (pattern-match reactive)
;)
;
;(defmessage-handler Klassenname initinstance ()
;
;   (send ?self put-StutzenAnz 2)
;   (send ?self put-ConnectClass (create$ Klassennamen))
;   (send ?self put-ConnectInstanceNr (create$ Stutzen-Nummern))
;   (printout t "Objekt " ": ...")
;   (send ?self print)
;   (printout t "...wurde initialisiert" crlf crlf)
;)
;
;*****************************************************************
; SCHLAEUCHE
;*****************************************************************
;******************** KR-Schlauch_Klasse ************************************
(defclass KR-Schlauch_Klasse (is-a Schlauch)
                       (role concrete)
                       (pattern-match reactive)
)
(defmessage-handler KR-Schlauch_Klasse initinstance ()

   (send ?self put-StutzenAnz 2)
   (send ?self put-ConnectClass (create$ KR_Endobjekt_Klasse Kuehler_Ruecklauf_Abzw_Klasse))
   (send ?self put-ConnectInstanceNr (create$ 0 0))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)
)
;******************** KV_T2_Schlauch_Klasse ************************************
(defclass KV_T2_Schlauch_Klasse (is-a Schlauch)
                       (role concrete)
                       (pattern-match reactive)
)
(defmessage-handler KV_T2_Schlauch_Klasse initinstance ()

   (send ?self put-StutzenAnz 2)
   (send ?self put-ConnectClass (create$ KV_T_Stueck_Klasse KV_T2_Endobjekt_Klasse))
   (send ?self put-ConnectInstanceNr (create$ 0 0))
   (send ?self setRadius (symbol-to-instance-name [20.0]))
   (printout t "Objekt " ": ...")
   (send ?self print)
   (printout t "...wurde initialisiert" crlf crlf)

)
```

Klassendefinition der Objekte der Oberklasse Schrauben:

```
;*****************************************************************************
; SCHRAUBEN
;*****************************************************************************

;******************** Schraube ************************************
(defclass Schraube (is-a Obj_mit_Stutzen)
                (role concrete)
                (pattern-match reactive)

;------------------- S0-CW-Inst ---------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class --------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
; folgend zusaetzliche Eigenschaften fuer MSconstraint.x (Schraubenspezifika)
;------------------- ConstraintSLaenge ---------------------------------
   (multislot ConstraintSLaenge (create-accessor read-write)
           (type FLOAT)
           (default (create$ ))
   )
;------------------- ConstraintProRad ----------------------------------
   (multislot ConstraintProRad (create-accessor read-write)
           (type FLOAT)
           (default (create$ ))
   )
;------------------- ConstraintNewCycle --------------------------------
   (multislot ConstraintNewCycle (create-accessor read-write)
           (type INTEGER)
           (default (create$ ))
   )
)

(defmessage-handler Schraube initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ Mutter))
   (send ?self put-ConnectInstanceNr (create$ 1))
   (send ?self put-K-Fest FALSE)  ; beweglich
   ;(printout t "Objekt " ": ...")
   ;(send ?self print)
   ;(printout t "...wurde initialisiert" crlf crlf)

)
```

```
(defmessage-handler Schraube  relink ()
; hebt alten Verweis auf automatisch erstellte Instanz auf
; linkt neues Objekt dran
; Unterscheidung: Ein Stutzen wird sofort an das Mutterobjekt angelinkt: in der Semantik
;als auch in der VR-Grafik
;Bei Anlinken eines Stutzens wird auch dessen Variable Connect_to gesetzt
;
; automatisch erstellten Instanzen loeschen
;
;
(printout t "relink: " (instance-name ?self) " Kinder: " (send ?self get-Kinder) crlf)

   (bind ?idx 0)
   (while (neq ?idx (send ?self get-StutzenAnz))
          (bind ?idx (+ ?idx 1))
          (bind ?Feld (send ?self get-StutzenFeld))
          (if (neq Null (nth$ ?idx (send ?self get-Kinder)))
           then
             ; neue Instanz anlinken
             (if (instance-existp (symbol-to-instance-name (nth$ ?idx (send ?self get-Kinder))))
              then
                    (send ?self put-StutzenFeld (insert$ ?Feld ?idx (symbol-to-instance-name (nth$ ?idx
                                                                  (send ?self get-Kinder))) ) )
                    (bind ?Stutzen (nth$ ?idx (send ?self get-StutzenFeld)))
                    (send ?Stutzen set-ParentInstance  (create$ (instance-name ?self) (instance-name
                                                                  ?self)) )
                    (bind ?InstanceName (getConnectInstanceName (nth$ ?idx (send ?self get-
                                                                  ConnectClass))
                                              (nth$ ?idx (send ?self get-ConnectInstanceNr)) ))

                    (if (eq ?InstanceName Null)
                     then
                       (printout t "relink: Null kam zurueck von getConnectInstanceName!!" crlf crlf)
                     else
                       (send ?Stutzen put-Connect_to ?InstanceName)

                       ; Folgend:Constraint-Werte setzen für die Schraubverbindung
                       (slot-insert$ ?self ConstraintAType ?idx 3)
                       (slot-insert$ ?self ConstraintSLaenge ?idx 30)
                       (slot-insert$ ?self ConstraintRelTotal ?idx 100)
                       (slot-insert$ ?self ConstraintWinkel ?idx 0.21816616)
                       (slot-insert$ ?self ConstraintProRad ?idx 3.183)
                       (slot-insert$ ?self ConstraintNewCycle ?idx 1)
                       ; Ende

                    ) ;if
                    (linkObjects  (nth$ ?idx (send ?self get-Kinder))  (instance-name-to-symbol
                                  (instance-name ?self)))
              else
                    (send ?self put-StutzenFeld (insert$ ?Feld ?idx Null))
             ) ;if
            else
             (send ?self put-StutzenFeld (insert$ ?Feld ?idx Null))
           ) ;if
   ) ;while
)
;****************** Mutter ************************************
```

```
(defclass Mutter (is-a Obj_mit_Stutzen)
                (role concrete)
                (pattern-match reactive)

;------------------- S0-CW-Inst ----------------------------------------
   (slot S0-CW-Inst (create-accessor read-write)
           (type INSTANCE-NAME)
           (default [NIL])
   )
;------------------- S0-CW-Class ----------------------------------------
   (slot S0-CW-Class (create-accessor read-write)
           (type SYMBOL)
           (default NOCLASS)
   )
)

(defmessage-handler Mutter initinstance ()

   (send ?self put-StutzenAnz 1)
   (send ?self put-ConnectClass (create$ Schraube))
   (send ?self put-ConnectInstanceNr (create$ 1))
   (send ?self put-K-Fest TRUE)  ; beweglich
   ;(printout t "Objekt " ": ...")
   ;(send ?self print)
   ;(printout t "...wurde initialisiert" crlf crlf)

)
```