Reihe Informationsmanagement im Engineering Karlsruhe

Hardy Krappe

Erweiterte virtuelle Umgebungen zur interaktiven, immersiven Verwendung von Funktionsmodellen

Band 1 – 2009

universitätsverlag karlsruhe

Hardy Krappe

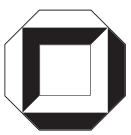
Erweiterte virtuelle Umgebungen zur interaktiven, immersiven Verwendung von Funktionsmodellen

Reihe Informationsmanagement im Engineering Karlsruhe
Band 1 – 2009

Herausgeber
Universität Karlsruhe (TH)
Institut für Informationsmanagement im Ingenieurwesen (IMI)
o. Prof. Dr. Dr.-Ing. Jivka Ovtcharova

Erweiterte virtuelle Umgebungen zur interaktiven, immersiven Verwendung von Funktionsmodellen

von
Hardy Krappe



universitätsverlag karlsruhe

Dissertation, Universität Karlsruhe (TH), Fakultät für Maschinenbau, 2009

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2009
Print on Demand

ISSN: 1860-5990
ISBN: 978-3-86644-380-8

Universität Karlsruhe (TH)
Fakultät Maschinenbau
Institut für Informationsmanagement im Ingenieurwesen

zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

Erweiterte virtuelle Umgebungen zur interaktiven, immersiven Verwendung von Funktionsmodellen

genehmigte
Dissertation
von

Dipl.-Ing. Hardy Krappe

Tag der mündlichen Prüfung:	18. Februar 2009
Hauptreferent:	Prof. Dr. Dr.-Ing. Jivka Ovtcharova
Korreferent:	Prof. Dr.-Ing. habil. Ralph Stelzer

Wenn VR für die Masse erhältlich ist, wird man es nicht nur als ein Medium ansehen, das in der physikalischen Realität benutzt werden kann, sondern vielmehr als eine zusätzliche Realität.

Jaron Lanier

Yours to you.

Danksagung

Die vorliegende Arbeit entstand neben meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Informationsmanagement im Ingenieurwesen der Universität Fridericiana Karlsruhe (TH).

Frau Prof. Dr. Dr.-Ing. Jivka Ovtcharova, der Leiterin des Instituts, gilt mein besonderer Dank für die wissenschaftliche Betreuung und die Übernahme des Hauptreferats. Herrn Prof. Dr.-Ing. habil. Ralph Stelzer danke ich für die Übernahme des Korreferats und den damit verbundenen Anregungen. Herrn Prof. Dr.-Ing. Carsten Proppe danke ich für den Vorsitz der Prüfungskommission.

Für die während meiner Zeit am Institut erlebte kollegiale Zusammenarbeit bedanke ich mich bei allen Mitarbeitern des Instituts. Besonderer Dank gilt:

- Peter Böser, ohne dem ich meine Tätigkeit am Institut nie gestartet hätte
- Thomas Paral, Oliver Hornberg und Matthias Gebauer, die mich noch während meiner Hiwi-zeit und am Anfang meines Assistentendaseins in die richtigen Bahnen gelenkt haben
- Oliver Mayer, Matthias Brenzinger, Martin Scherer, Philipp Bender, Tianfeng Zhou und Christoph Maser, die mir als studentische Hilfskräfte mit ihrem überdurchschnittlichen Engagement tatkräftig zur Seite standen
- Alexander Mahl, Milan Marinov, Robert Krikler und Berkun Culha, für die anregenden Diskussionen und das Korrekturlesen des Skripts
- Stilian Stanev, der mir den Rücken im Projekt freigehalten hat und
- Matthias Sander, der mich mit seinem Pragmatismus immer wieder zurück auf die Erde holte

Mein herzlicher Dank gilt auch meinen Eltern, die durch Ihre langjährige Unterstützung diesen Lebensabschnitt ermöglichten und Hans-Jörg Seng für seine zugetragene Weisheit.

Besonderer Dank gilt nicht zuletzt meiner besseren Hälfte Beatrix und unseren gemeinsamen Sohn Noah, die mein Fels in der Brandung waren und mich durch diese Odyssee liebevoll und geduldig begleiteten.

Hardy Krappe März 2009

Abstract

Unternehmen im Maschinenbau leiden neben dem für alle gültigen Kostendruck auch, bedingt durch Globalisierung, unter einer erheblich verschärften Wettbewerbssituation. Auf Marktanforderungen muss flexibel reagiert werden bei gleichzeitig wachsender Komplexität der Produkte. Unternehmenskonzentration und Auslagerung von Prozessen oder kompletten Unternehmensbereichen haben in den letzten Jahren den Bedarf geweckt, die Kernprozesse der Fertigungsindustrie besser miteinander zu integrieren, zu optimieren und dazu die Möglichkeiten moderner Technologien noch effektiver zu nutzen. Die wachsende Komplexität der Produkte und die immer stärkere Produktindividualisierung führen, bei immer kürzeren Produktentwicklungszeiten, zu einem außerordentlich schwer zu beherrschenden Abhängigkeitsgeflecht von Informationen in der Produktentwicklung.

Dem Erkennen und Verstehen dieser Abhängigkeiten durch Visualisierung, auch über die klassischen Domänengrenzen hinweg, kommt deshalb eine zentrale Bedeutung bei der Produktdefinition zu. Insbesondere existieren heute keine Ansätze Informationen aus frühen Konstruktionsphasen, wie z.B. der Funktionsmodellierungsphase, integriert mit der sich daraus ergebenden Bauteilgestalt und den gegenwärtigen technischen Möglichkeiten wie den Technologien der virtuellen Realität zu visualisieren.

Ziel dieser Arbeit ist dementsprechend die Entwicklung eines Prototypen, mit dessen Hilfe Funktionsmodelle in virtuelle Umgebungen übertragen und dort interaktiv nutzbar gemacht werden können. Dabei soll Wert auf einen hohen Immersionsgrad sowie auf ästhetische Interaktionselemente gelegt werden, um die Akzeptanz des Benutzers zu erhöhen. Durch die semantische Abbildung von Funktionsmodellen und die interaktive, immersive Visualisierung der Zusammenhänge der Produktfunktionen mit der Produktstruktur soll das interdisziplinäre Gesamtverständnis komplexer Produkte gefördert werden, so dass frühzeitig Probleme erkannt und Entscheidungen auf Basis verständlich aufbereiteter Informationen getroffen werden können.

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 Ausgangssituation

Die Informations- und Kommunikationstechnologien durchdringen alle Lebens- und Arbeitsbereiche in unserer Gesellschaft und bilden die technologische Basis für die Informations- und Wissensgesellschaft. Eine Spitzenposition in der globalen Informationsgesellschaft ist Voraussetzung für die Stärkung von Wettbewerbsfähigkeit und Wachstum. Mehr als die Hälfte der Industrieproduktion und über 80% der Exporte Deutschlands beispielsweise hängen heute vom Einsatz moderner Informations- und Kommunikationstechnik und elektronischer Systeme ab. In den Branchen Automobilindustrie, Maschinenbau, Medizintechnik und Logistik sind mittlerweile mehr als 80% der Innovationen durch IKT getrieben. Der Markt für IKT beträgt allein in Deutschland ca. 140 Milliarden Euro und weltweit sogar mehr als 2.100 Milliarden Euro und wächst konstant weiter, wie Bild 1.1 auf der nächsten Seite belegt. (BMBF, 2006)

Dabei führt die fortschreitende Globalisierung, verstärkt durch die rasch wachsende Kompetenz in den Niedriglohnländern, seit mehreren Dekaden dazu, dass in immer stärkerem Maße Entwicklung und Herstellung hoch technisierter Produkte und Dienstleistungen in diese Länder abwandern. Gleichzeitig werden regelmäßig wichtige technische Neuerungen nicht mehr in Produkte und Dienstleistungen umgesetzt. Das bedeutet, dass die Rendite aus Ergebnissen der Forschung und Entwicklung nur ungenügend abgeschöpft wird. Die Antwort auf diese Herausforderung kann nur sein: Ide-

en schneller in Produkte umsetzen und die vorhandenen Stärken hervorheben. (BMBF, 2006)

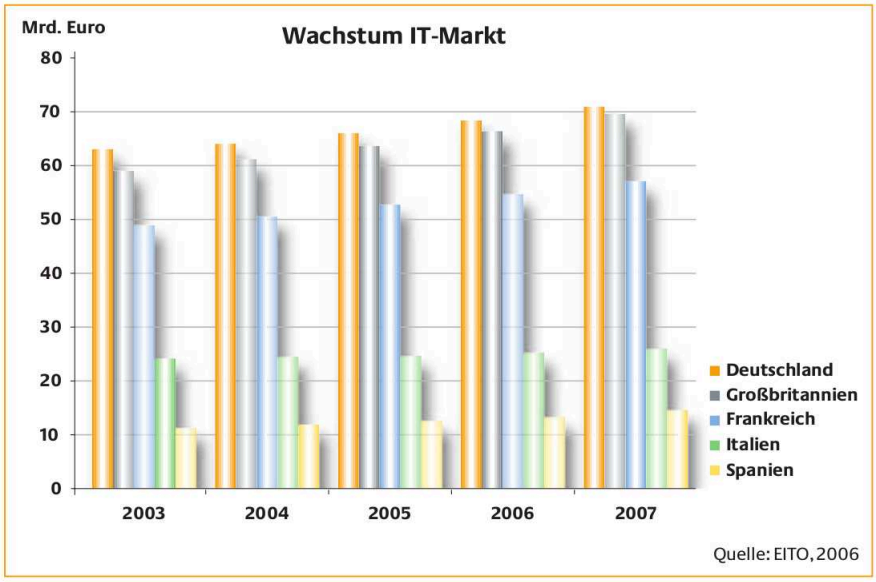


Bild 1.1: Wachstum des IKT Markts

Um Ideen schneller umsetzen zu können, müssen hohe Anforderungen an die Produktentwicklung eines Unternehmens und die zu generierenden innovativen Prozesse gestellt werden (Geiger, 1997). Der Innovationsgrad eines Unternehmens spielt dabei eine wesentliche Rolle. Ökonomisch sinnvoll ist insbesondere ein hoher Innovationsgrad bei geringen Kosten und ausgezeichneter Qualität. In diesem Kontext sind die frühen Konstruktionsphasen von entscheidender Bedeutung, denn in diesen steckt das größte Innovationspotential. Innovation bedeutet, dass

> *das neue Produkt eine neue Funktion erfüllt oder eine bekannte Funktion in neuer Qualität erreicht wird* (Langlotz, 2000).

Beide Anforderungen zeigen meist über Lösungen hinaus, die nur aus einer Ingenieursdisziplin entstammen. Daher sind innovative Produkte in der Regel von interdisziplinärer und damit komplexer Natur.

Als Bindeglied zur Entwicklung neuer innovativer Lösungen und den konstruktiven Zusammenhängen kann die Produktfunktion genutzt werden. Durch eine funktionale Produktmodellierung ist eine lösungsneutrale und

abstrakte Repräsentation der zu erstellenden konstruktiven Aufgabe darstellbar. Diese Abstraktion der konstruktiven Aufgabe und die Beschreibung von Grundkonzepten unter Nutzung der Produktfunktion wird bereits in vielen Bereichen angewendet und insbesondere in der Elektrik- und Elektronikentwicklung, in der Hydraulik und Pneumatik sowie in der Softwareentwicklung durch geeignete Entwicklungswerkzeuge unterstützt.

In der aktuellen Praxis der mechanischen Konstruktion hingegen wird eine funktionsorientierte Entwicklung nicht durch Entwicklungsumgebungen unterstützt. Ebenso wird die Produktfunktion nicht in der Produktdokumentation festgehalten. Dies ist bemerkenswert, da sowohl in der Konstruktionstheorie als auch in der industriellen Praxis die Konstruktion die Erfüllung einer Gesamtfunktion zum Ziel hat und die Zerlegung der Gesamtfunktion in Teilfunktionen sowie das Zuordnen der Produktanforderungen die Ausgangspunkte für die weitere Bearbeitung der Konstruktionsaufgabe darstellen. (Leemhuis, 2005, p.3)

1.2 Problemstellung

Durch die Möglichkeiten der gegenwärtigen Informationstechnologien hinsichtlich Visualisierung, Vernetzung, Verarbeitung und Bereitstellung von Informationen und Wissen bekommt die Konstruktionsmethodik, insbesondere in den frühen Entwicklungsphasen, eine neue Dimension. Anhand einer geeigneten Visualisierung von Produktfunktionen ist sowohl die abstrakte, als auch die abgeleitete, konkrete Abbildung der Haupt- und Teilfunktionen erforderlich, sowie die Ableitung von Informationen hinsichtlich Simulation oder Lösungsfindung. Darüber hinaus ist ein iterativer Entwurf notwendig, in dem sich die Entwicklungsschritte der Synthese und Analyse im ständigen Wechsel befinden.

Dass Computer dabei eine zentrale Rolle einnehmen, ist unbestreitbar. Die im Computer erzeugte Realität, die virtuelle Welt, erfordert für den interaktiven Umgang mit ihr eigene Fähigkeiten, neue Qualifikationen und insbesondere geeignete Interfaces. Diese sind von zentraler Bedeutung. Brenda Laurel formuliert dies folgendermaßen:

> *Ein Interface ist die Kontaktoberfläche zu einer bestimmten Sache. Unsere Welt ist voll davon. Ein Türgriff ist das Interface zwischen einer Person und einer Tür. Lenkrad, Gaspedal, Kupplung und die Instrumente sind Interfaces zwischen Fahrer und*

> *Auto. Ein Raumanzug ist das Interface zwischen Astronauten und dem All.* (Laurel, 1992, p.47)

Für jeden Bereich gibt es die geeigneten Interfaces. Die Schnittstelle zwischen Mensch und Computer wird allerdings von Bildschirm, Tastatur und Maus beherrscht. Die graphische Benutzerschnittstelle, die nach dem Leitbild eines Schreibtisches gestaltet wurde, ist ebenfalls allgegenwärtig. Jedoch ist diese Metapher* wohl für Büro- und Verwaltungsarbeiten am eingängisten. In anderen Arbeitsbereichen, wie Produktion, Medizin und allen strukturierenden, gestaltenden Gebieten (z.B. Design, Architektur etc.) erfordert der Einsatz von Computersystemen andere Metaphern und andere Ein- und Ausgabemedien als Maus, Tastatur und Bildschirm. Der Charakter und die Komplexität der dort zu bewältigenden Aufgaben macht angemessenere Alternativen notwendig (Rügge, 1999).

Die graphische Darstellung von virtuellen Welten stößt technisch jedoch immer wieder an ihre Grenzen. Die Hauptproblematik liegt im generellen Konflikt zwischen der Realität und der Darstellung dieser in Echtzeit. Aufgrund der zu hohen Anforderungen an allen beteiligten Systemen muss der graphische Detaillierungsgrad verringert werden, um bestimmte Szenarien fließend darstellen zu können. Die Bewegung durch eine korrekt beleuchtete, dynamisch adaptive, virtuelle Welt ist mit heutiger Rechenleistung nicht realisierbar, ebenso wenig die realistische Darstellung vieler komplexer Objekte zusammen in einem Bild, wie z.B. eine Menschenmenge (Bücker, 2003). Um diese und andere Aufgaben in Echtzeit zu lösen, müssen Kompromisse eingegangen werden. Diese vermindern den Realitätsgrad, erhöhen jedoch teilweise die Ausführungsgeschwindigkeit erheblich.

1.3 Zielsetzung und Vorgehensweise

Ziel dieser Arbeit ist es, Funktionsmodelle in virtuellen Umgebungen zu übertragen und dort interaktiv nutzbar zu machen. Dabei spielt die interaktive, immersive Visualisierung der Zusammenhänge der Gesamt- und Teilfunktionen mit der Produktstruktur für das interdisziplinäre Gesamtverständnis komplexer Produkte eine tragende Rolle. Ein Produkt hat keine fest definierte Funktionalität, da es immer darauf ankommt, wofür und wie es der Anwender einsetzt. Es kann prinzipiell für viele unterschiedliche Funktionen eingesetzt werden. Somit dient die virtuelle Umgebung zum

* griechisch “Übertragung”, von metà phèrein - “anderswohin tragen”

Einen dazu, die vom Konstrukteur beabsichtige Funktionsweise bildlich darzustellen, zum Anderen aber auch eine durchgängige, funktionale Sicht auf das Gesamtprodukt zu ermöglichen, und dabei den Beteiligten interaktiv Informationen zu vermitteln.

Die Arbeit betrachtet zwei Schwerpunkte. Zum Einen soll eine rechnerunterstütze, konstrukteursgerechte Funktionsmodellerstellung bereitgestellt werden, von der aus die Funktionsmodelle in die virtuelle Umgebung übertragen werden können. Zum Anderen soll die grafische Benutzerschnittstelle anwendungsbezogen erweitert werden, wobei der Fokus auf der Interaktion zwischen Mensch und Maschine liegt. Interaktion soll im Rahmen dieser Arbeit nicht nur mit dem Begriff der Kommunikation, wie es in der Informatik der Fall ist, gleichgesetzt werden. Vielmehr soll dem Verständnis Jaron Laniers gefolgt werden, dem Schöpfer des Kunstwortes "virtual reality", der im Interview mit Tom Sperlich definierte, dass

> *Interaktivität neben Kommunikation auch Überraschungen und neue Entdeckungen beinhalten muß* (Sperlich, 1995).

Ähnlich äußert sich ebenda auch Andy Lippman vom MIT Medien Labor, der Interaktivität als

> *eine gemeinsame, gleichzeitige Aktivität seitens zweier Teilnehmer, die normalerweise, aber auf keinen Fall immer, auf etwas abzielt* (Sperlich, 1995)

definiert. Beiden gemeinsam ist, dass Interaktivität als kontinuierlicher und nicht diskreter Vorgang verstanden wird.

Die Vorgehensweise ist dadurch motiviert, dass die virtuelle Realität durch das ihr innewohnende Potential die Möglichkeit birgt, mittels neuartiger Ein- und Ausgabegeräte sowie innovativen Interaktionsmetaphern dreidimensionale Welten über die Physis hinaus zu erweitern. Die Abbildung von, dem menschlichen Denken nahen, Funktionsmodellen und deren Anreicherung mit zusätzlichen Informationen ermöglicht eine ganz neue Sicht auf den Produktentwicklungsprozess. Dabei wird zum einen die Kreativität der an der Produktentwicklung beteiligten Entwickler gefördert und zum anderen die entwicklungsteamübergreifende Kommunikation unterstützt.

Um die Zielsetzung der Arbeit zu erreichen, müssen zusammenfassend folgende Lösungsschritte durchgeführt werden, deren Ergebnisse in der späte-

ren Nutzung des aus dem Konzept resultierenden Softwareprototyps dem Konstrukteur bereitgestellt werden.

- Entwicklung einer VR gerechten Notation zur Erstellung der Funktionsmodelle in einer einfach zu erlernenden Umgebung.
- Entwicklung eines Sichtenmodells basierend auf Funktionsmerkmalen für die Übertragung.
- Entwicklung geeigneter Repräsentations- und Interaktionsmetaphern für die Visualisierung und Manipulation der Funktionsmodelle in der virtuellen Umgebung.

Dabei ist die Struktur der Dissertation, wie Bild 1.2 zeigt, folgendermaßen aufgebaut.

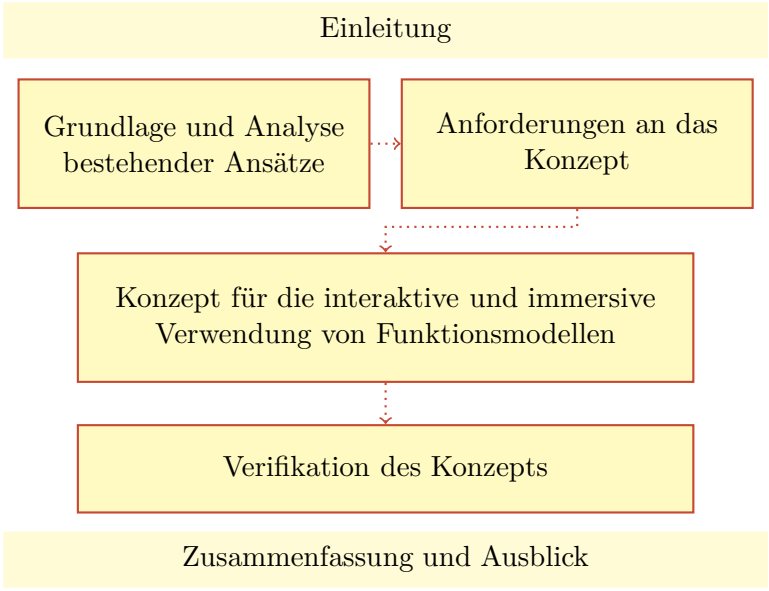


Bild 1.2: Aufbau der Dissertation

In Kapitel 1 auf Seite 1 soll die gegenwärtige Marktsituation im Bereich IKT kurz abgerissen und daraus die Problemstellung und die resultierende Aufgabenstellung abgeleitet werden. Daraufhin sollen die Zielsetzung und die benötigten Lösungschritte aufgezeigt und die Vorgehensweise erläutert werden. Außerdem wird die Struktur der Arbeit illustriert und der Nutzen kurz diskutiert.

Kapitel 2 auf Seite 9 diskutiert die zur Erreichung der Zielsetzung notwendigen Grundlagen und analysiert die bestehenden Ansätze hinsichtlich ihrer Anwendbarkeit im Kontext der Arbeit. Dabei werden die Bereiche Funktionsmodellierung, Informationsvisualisierung und virtuelle Realität genauer betrachtet.

Kapitel 3 auf Seite 79 erläutert die Anforderungen an das Konzept. Diese wurden in drei Bereiche unterteilt, in Forderungen an die Übertragbarkeit von Funktionsmodellen, Forderungen an die Semantik in virtuellen Umgebungen und in Forderungen an die Technologien in virtuellen Umgebungen.

In Kapitel 4 auf Seite 93 wird die angewandte Methodik detailliert vorgestellt, die für die interaktive, immersive Verwendung von Funktionsmodellen in virtuellen Umgebungen entwickelt wurde. Dabei werden die einzelnen, notwendigen Schritte nacheinander diskutiert, beginnend mit der Modellerstellung, der anschließend folgenden Modellabbildung, den zu erzeugenden Interaktionsmetaphern und der abschließenden 3D Visualisierung. Da der Lösungsansatz auf den unterstützenden Einsatz spezieller Interaktionsmetaphern in virtuellen Umgebungen besteht, werden diese, auf Basis des Grundgerüsts Funktionsmodell, detailliert beschrieben.

Der daraus resultierende Softwareprototyp wird in Kapitel 5 auf Seite 177 am Beispiel des mechatronischen Produktes *crashaktive Kopfstütze* eines elektronisch gesteuerten Fahrersitzes eines Automobils verifiziert.

Abschließend sollen im Kapitel 6 auf Seite 271 die Schlüsselinnovationen nochmals zusammengefasst und ein Ausblick über weitere naheliegende, nützliche Schritte gegeben werden.

KAPITEL 2

Grundlagen und Analyse bestehender Ansätze

In diesem Kapitel sollen zuerst die konstruktionsmethodischen Grundlagen dargelegt werden. Der derzeitige Stand der Technik im Bereich der Konstruktionsmethodik ist gekennzeichnet durch die bekannten Lösungsansätze der nationalen und internationalen Forschung und die darauf basierenden, am Markt existierenden Softwarewerkzeuge. Die Konstruktionsmethodik stellt einen reichen Fundus an Methoden und Regeln bereit, die das systematische Vorgehen im Konstruktionsprozess in Form von Vorgehensmodellen beschreibt (Gebauer, 2001). Charakteristisch dabei ist die Unterteilung und Strukturierung des Konstruktionsablaufs in einzelne, systematisch aufeinander aufbauende Arbeitsschritte mit definierten Arbeitsergebnissen. Da das Ziel der Arbeit die Übertragung von Funktionsmodellen in eine interaktive, immersive virtuelle Umgebungen ist, wird die funktionale Betrachtung technischer Produkte detailliert hervorgehoben. Hierbei werden die hierarchischen und flussorientierten Funktionsstrukturen insbesondere hinsichtlich ihre Realisierung in ingenieursgerechte Softwaretools analysiert. Im Anschluss werden die Grundlagen rechnerunterstützter 3D Modellerstellung kurz erläutert, da diese zur Erstellung virtueller Umgebungen notwendig sind. Daraufhin wird der Stand der Technik im Bereich der Informationsvisualisierungsmethoden diskutiert. Im Anschluss werden die bestehenden und im Rahmen dieser Arbeit relevanten Ansätze hinsichtlich der virtuellen Realität und der darin notwendigen Interaktionsmetaphern erörtert. Abschließend werden die Ansätze bezüglich der Zielsetzung bewertet.

2.1 Konstruktionsmethodik

Konstruktionsmethodik nennt man die Vorgehensweisen beim Entwickeln und Konstruieren nach Ablaufplänen mit Arbeitsschritten und Konstruktionsphasen unter Verwendung von Richtlinien und Methoden sowie technischen und organisatorischen Hilfsmitteln. Die Konstrukteure erhalten Methoden und Hilfsmittel, die es ihnen gestatten, systematisch und zielorientiert zu arbeiten, um effektiver und besser als bisher Lösungen zu finden. (Conrad, 1998)

2.1.1 Ansätze aus dem deutschsprachigen Raum

Die frühe Literatur zur Konstruktionsmethodik wie Arbeiten von Wörgebauer, Tschochner, Sobolski, Matonsek und Anderen war im wesentlichen intuitiv-orientiert (Rutz, 1985). Sie enthielt jedoch bereits erste Ansätze zu einer Systematisierung des Konstruktionsprozesses. Diese Arbeiten gelten als Ursprung des methodischen Konstruierens im deutschsprachigen Raum und bildeten den Ausgangspunkt für die, in den vergangenen Jahrzehnten erarbeiteten Ansätze von Rodenacker (Rodenacker, 1991), Roth (Roth, 2000), Koller (Koller, 1998), Pahl (Pahl, 2007), Hubka (Hubka and Eder, 1988), Ehrlenspiel (Ehrlenspiel, 2003) etc.

Ziel der deutschsprachigen Konstruktionsmethodik ist die Erstellung eines generell anwendbaren, branchenunabhängigen und allgemeingültigen Vorgehensmodells für die Beschreibung des Konstruktionsprozesses (Gebauer, 2001). Dementsprechend wurden unterschiedliche Vorgehensweisen entwickelt, die bezüglich des Ablaufs zusammenfassend in den VDI-Richtlinien 2221/2222 beschrieben sind. Diese Richtlinien stellen einen Kompromiss der unterschiedlichen Auffassungen der deutschen Konstruktionsmethodiker dar. Die VDI-Richtlinie 2221 definiert das Konstruieren als die

> *Gesamtheit aller Tätigkeiten, mit denen - ausgehend von einer Aufgabenstellung - die zur Herstellung und Nutzung eines Produkts notwendigen Informationen erarbeitet werden und in der Festlegung der Produktdokumentation enden.* (VDI 2221, 1993)

Der Konstruktionsprozess wird in mehrere Konstruktionsphasen unterteilt und durchläuft dabei generell drei Stadien: Ausgangszustand, Zustandsänderung, Endzustand. Im Ausgangszustand wird zuerst die Problembeschreibung und die daraus resultierende Aufgabenstellung aufgenommen.

Während der Zustandsänderung wird die Produktbeschreibung entlang der Lebensphasen sukzessive vervollständigt, solange bis schließlich ein Endzustand erreicht ist. Dabei nehmen sowohl die produktbeschreibenden Eigenschaften als auch der Informationsgehalt der Produktbeschreibung stetig zu. Ist der Endzustand der Produktbeschreibung erreicht, so sind alle relevanten Produkteigenschaften bekannt. Hubka bezeichnet in (Hubka and Eder, 1988) diese auch als Konstruktionsmerkmale. Sie bilden die Grundlage für die Existenz und das Verhalten des Produkts in denen der Konstruktion nachfolgenden Produktlebensphasen.

Während jeder Konstruktionsphase werden die fünf Konstruktionstätigkeiten* Problemdefinition, Lösungsfindung, Lösungsbeschreibung, Lösungsbeurteilung und Lösungsauswahl durch einen Konstrukteur ausgeführt. Bei der Problemdefinition erfolgt die Abgrenzung einer vorliegenden Problemstellung gegenüber der Umgebung, wobei Schnittstellenbedingungen beachtet werden müssen. Die Lösungsfindung ist eine konstruktionsphasenübergreifende Tätigkeit. Die Lösungsbeschreibung umfasst die Teil- oder Gesamtlösung zu einer vorliegenden Problemstellung, z.B. die Funktionsstruktur in der Funktionsmodellierung. Liegen eine oder mehrere Lösungen vor, so wird eine Lösungsbeurteilung durchgeführt, z.B. durch technisch wirtschaftliche Bewertungen. Aus den gewerteten Alternativen wird letztendlich eine Lösung ausgewählt bzw. bei nur einer vorliegenden Lösung diese bestätigt. (Michelis, 2002)

Ein weiteres wesentliches Merkmal des Konstruktionsprozesses liegt in dem iterativen Vorgehen, das durch ein nahezu beliebiges Vorwärts- und Rückwärtsschreiten sowie dem erneuten Durchlaufen von Konstruktionsphasen gekennzeichnet ist (Kunze, 2002).

Um innerhalb des Konstruktionsprozesses eine durchgängige Rechnerunterstützung zu gewährleisten, wurde dieser in vier Modellierungsphasen, nämlich der Anforderungsmodellierung, Funktionsmodellierung, Prinzipmodellierung und Gestaltmodellierung, unterteilt (VDI 2210, 1975). In diesen soll das rechnerunterstützte Erzeugen der produktbeschreibenden Informationsinhalte realisiert werden (Kläger, 1993; Rude, 1991; Seiler, 1985; Suhm, 1993). Dabei entsprechen diese in erster Näherung einer logischen Strukturierung der wichtigsten Informationszustände der Produktbeschreibung im rechnerunterstützten Konstruktionsprozess (Kunze, 2002).

* Der methodische Konstruktionsprozess bildet nicht alle Tätigkeiten ab, es gibt weitere indirekte Tätigkeiten wie Normung, Dokumentation etc. die nicht betrachtet werden.

Anforderungsmodellierung In dieser Modellierungsphase wird die rechnerinterne Abbildung der Produktinformationen ermöglicht, die in der Aufgabenklärungsphase des Konstruktionsprozesses entstehen. Die Aufgabenklärung legt das Produktverhalten durch die Produktaufgabe fest. Die Produktaufgabe, die in Substantiv-Verb-Form beschrieben wird, erläutert das Verhalten eines Produktes. Das Verb wird dabei aus einer Menge normierter Aufgabenverben entnommen, während der Begriff für das Substantiv frei wählbar ist. Eine Produktaufgabe kann in die jeweiligen Teilaufgaben zerlegt werden, die in der Produktaufgabenstruktur dargestellt werden kann (Roth, 2000). Darüber hinaus ist ein wesentlicher Aspekt der Anforderungsmodellierung die Bereitstellung der Produktanforderungen für die betroffenen Problemlösungsvorgänge auf den weiteren Konkretisierungsstufen des Konstruktionsprozesses (Kläger, 1993).

Funktionsmodellierung Im Zuge der Funktionsmodellierung erfolgt die Beschreibung der Funktionsweise eines Produkts. Die rechnerinterne Beschreibung der technischen Produktfunktionen erfordert die Möglichkeit, unterschiedliche Abstraktionsstufen zu unterscheiden, verschiedene Sichten auf die Produktfunktionen auszuprägen und im Rahmen der Lösungsfindung zu diesen Produktfunktionen Lösungsvarianten zu bestimmen (Kunze, 2002).

Prinzipmodellierung Innerhalb der Prinzipmodellierung werden die entsprechenden physikalischen, chemischen oder biologischen Effekte, die in formaler Beziehung zu den modellierten Funktionsgrößen stehen, zugeordnet. Effekte sind an bestimmte, stoffliche Strukturen gebunden, den so genannten Effektträgern. Durch die Verbindung der einzelnen Effektträger entsteht ein dem Wirkprinzip entsprechendes Gebilde, das als Wirkstruktur oder Effektkettenstruktur bezeichnet wird. (Gebauer, 2001)

Gestaltmodellierung Die Phase der Gestaltmodellierung definiert die Gestalt eines Produktes durch Form, Art, Lage und Anordnung der technischen Gebilde (Gebauer, 2001). Ausgehend von den in den vorangegangenen Modellierungsphasen festgelegten Eigenschaften und Strukturen erfolgt hier die Festlegung der geometrischen Produktbeschreibung (Kunze, 2002).

Die Verbindung dieser Modellierungsphasen, mit den darin enthaltenen Modellierungsverfahren konzeptionell zu einem dreidimensionalen Modellraum des Konstruierens beschreibt Rude (Rude, 1991).

2.1.2 Internationale Ansätze

Innerhalb dieses Kapitels sind die internationalen Aktivitäten auf dem Gebiet der Konstruktionswissenschaften beschrieben. In diesem Zusammenhang sind aus Großbritannien die Arbeiten von Archer (Archer, 1985) zu systematischen Methoden und Strukturen von Konstruktionsprozessen, die Arbeiten von Cross (Cross, 1985), zum konstruktionsmethodischen Vorgehen, von Gregory (Gregory, 1970), zur kreativitätsfördernden Methodenfindung sowie von Wallace (Wallace and Hales, 1991) mit dem Aufzeigen der Anwendbarkeit konstruktiver Vorgehensweisen bei Industrieprojekten zu nennen.

In den USA wurden von Dixon (Dixon, 1988) Grundlagen und Forschungsansätze zum methodischen Konstruieren und von Nadler (Nadler, 1981) eine allgemeine Konstruktionsmethodik erarbeitet. Ostrofsky (Ostrofsky, 1977) veröffentlichte eine Konstruktions-, Planungs- und Entwicklungsmethodik und Suh (Suh, 1990) stellt die axiomatische Konstruktionslehre vor.

Aus Japan sind insbesondere die Autoren Taguchi und Yoshikawa als Vertreter der General Design Theorie, für die Aufstellung von Methoden im Konstruktionsprozess bekannt (Yoshikawa, 1987) sowie Tomiyama (Tomiyama et al., 1992) im Bereich der Systematisierung von Konstruktionswissen.

Des Weiteren sind die Arbeiten von Andreasen (Andreasen and Hein, 1987) aus Dänemark, Pighine (Pighine, 1990) aus Italien, sowie die Arbeiten von Altshuller (Altshuller, 1984) und Odrin (Odrin, 1986) aus der ehemaligen Sowjetunion zu nennen.

Alle Ansätze folgen verschiedenen Schwerpunkten und unterscheiden sich in ihrer Herangehensweise zur Konstruktionstätigkeit. Deutlich wird jedoch, dass ein konstruktionsmethodisches Vorgehen erhebliche synergetische und ökonomische Potentiale birgt. Eine Gegenüberstellung exemplarischer Konstruktionsabläufe zur methodischen Produktentwicklung ist in Tabelle 2.1 dargestellt.

Wird im Rahmen der deutschen Konstruktionsmethodik der Konstruktionsprozess an Hand von Konstruktionsphasen beschrieben, versucht das internationale Konzert eine globale Konstruktionstheorie zu entwerfen, die im wesentlichen auf axiomatischen Aussagen beruht. So basieren sowohl die Ansätze von Suh als auch die Yoshikawas auf der mathematischen Theorie der Mengenlehre und beide versuchen eine topologische Erfassung des konstruktiven Wissens mittels eines formalen Rahmens zu erreichen.

Methoden	**Phasen**				
	Anforderungsanalyse	Funktionsstrukturen	Prinzipielle Lösung	Wirkgeometrien	Gestaltausarbeitung
VDI2221, VDI2222	Anforderungsliste	Funktionsstrukturen	Physikalische Effekte	Wirkgeometrien	Gestaltausarbeitung
Pahl/Beitz	Anforderungsliste	Funktionsstrukturen	Physikalische Effekte	Wirkgeometrien	Gestaltausarbeitung
Roth	Aufgabenformulierungsphase	Funktionelle Phase: logische, kybernetische, physikalische Funktionsstrukturen		Gestaltende Phase: geometrische Wirkstruktur, prinzipielle Lösung, Detaillierung	
Koller	Anforderungsanalyse	Funktionssynthese	Prinzipsynthese	Gestaltsynthese	Maßsynthese
Rodenacker	Festlegung logischer und physikalischer Zusammenhänge			Festlegung konstruktiver Zusammenhänge	
Hubka	Anforderungsanalyse	Funktion im Zentrum	Physikalische Effekte	Wirkgeometrie	Detaillierung
Suh	Funktionale Domäne			Physikalische Domäne	
Yoshikawa	Funktionale Anforderungsliste	Funktionsstrukturen	Prinzipielle Lösung	Wirkgeometrie	Gestaltausarbeitung

Tabelle 2.1: Gegenüberstellung von Konstruktionsmethoden nach Leemhuis (Leemhuis, 2005)

Suh entwickelt in seiner Betrachtungsweise eine Theorie, die im Wesentlichen auf Basis von zwei Axiomen[†] aufgebaut ist.

- Axiom 1: Das *Unabhängigkeitsaxiom* fordert die Unabhängigkeit der funktionalen Anforderungen.
- Axiom 2: Das *Minimalitälsaxiom* postuliert, dass diejenige Konstruktion die Beste ist, die das Unabhängigkeitsaxiom erfüllt und sich zusätzlich durch den geringsten Informationsgehalt auszeichnet.

Yoshikawa entwickelte ebenfalls einen streng axiomatisch aufgebauten Ansatz, der unter dem Namen General Design Theory (GDT) bekannt geworden ist. Dabei wird das Konstruieren als eine direkte Abbildung von einem Funktionsraum mit den gewünschten Benutzerfunktionen auf einen Attributraum mit den ausgeprägten Produkteigenschaften aufgefasst. Da sich dieses in der Realität nicht bewährt hat, wurde zusätzlich ein so genanntes Meta-Modell eingeführt, das den Abbildungsprozess zwischen Funktions- und Attributraum schrittweise in Form weiterer Modelle, wie beispielsweise Berechnungs- und Prinzipmodelle, zu beschreiben versucht (Tomiyama and Yoshikawa, 1985; Tomiyama et al., 1992; Yoshikawa, 1987).

Mit der Universal Design Theory (UDT) wurde versucht eine Konstruktionstheorie zu entwickeln, die zum einen als Kern eine allgemeine Theorie enthält, welche zur Konstruktion aller Arten von Artefakten[‡] anwendbar ist, zum anderen jedoch universalen Charakter hat und um beliebige Theorien erweiterbar sein soll. (Gebauer, 2001)

Nach der Betrachtung der methodischen Vorgehensweise bei der Produktentwicklung wird im folgenden Abschnitt detaillierter auf die Phase der funktionalen Modellierung eingegangen. Da das übergeordnete Ziel dieser Arbeit die Übertragung von Funktionsmodellen in eine virtuelle Umgebung ist, werden zunächst unterschiedliche Funktionsbegriffe voneinander abgegrenzt.

[†]Bei der Formalisierung eines Wissensgebiets ist man häufig bemüht, eine möglichst kleine Menge von Grundwissenseinheiten zu finden, die als gültig angesehen werden und aus denen alles andere Wissen mittels allgemein anerkannter Grundregeln formal abgeleitet werden kann. Solche Grundwissenseinheiten eines Gebietes werden als Axiome des Gebiets bezeichnet (Broy, 1999).

[‡]Artefakte sind vom Menschen geschaffene Gebilde, die unsere Welt in zunehmendem Maße neben den natürlichen Systemen bestimmen.

2.2 Grundlagen der Funktionsmodellierung

In der Literatur sind verschiedene Arten von Funktionen und Funktionsstrukturen bekannt (Rude, 1998, p.239). Koller kennt Elementarfunktionen (Koller, 1998, p.39), Roth beschreibt Soll-, Ist- und Trägerfunktionen (Roth, 2000, p.25ff) und arbeitet die Funktionsstruktur auf Basis der Produktaufgabenstruktur aus. Gierse nutzt Funktionen und Funktionsstrukturen, um hinsichtlich Kostenbewertungen, unter Nutzung der Wertanalyse, Aussagen treffen zu können (Gierse, 1984) und Franke fand mit der logisch-funktionalen Analogie von Elementen unterschiedliche physikalische Effekte einen umfassenden Aufbau der Getriebe und gilt deshalb auch als wesentlicher Vertreter eines funktionellen Vergleichs physikalisch unterschiedlicher Lösungselemente (Leemhuis, 2005). Die Ansätze zur Funktionsmodellierung der Universität Karlsruhe (RPK) basieren weitgehend auf den Modellen von Roth und wurden in Dissertationen erweitert (Benz, 1990; Huang, 2002; Huber, 1994). Langlotz entwirft eine Klassifikation von Funktionen bestehend aus Nutzfunktionen, Störfunktionen, Folgefunktionen sowie Kompensationsfunktionen und nutzt diese Charakterisierung zur verbesserten Lösungsfindung (Langlotz, 2000).

Im internationalen Umfeld konzipierten zum einen Tomiyama und Yoshikawa (Tomiyama and Yoshikawa, 1985) das Function/ Behavior/ State Modell (FBS-model). Kern dieses Modells ist die integrale Spezifikation von Funktion, Verhalten und Zustand eines Objektes auf Basis einer vernetzten Substantiv-Verb Struktur. Die Unterscheidung von Funktion und Verhalten wird derart beschrieben, dass die Funktion angibt, wofür das Produkt genutzt wird. Das Produktverhalten beschreibt, was das Produkt macht. Zum anderen ist die Arbeit am Georgia Institute of Technology zu erwähnen. Hier entwickelten Chandrasekaran und Goel (Chandrasekaran et al., 1993) ein Funktionsmodell, dass basierend auf der Arbeit von Tomiyama zwischen Behavior (Verhalten) und Function unterscheidet. Es wurde ein so genanntes Structure/ Behavior/ Function Modell (SBF-model) zur Funktionsmodellierung konzipiert, in dem sowohl abstrakte als auch erdachte Beschreibungen physikalischer Prinziplösungen spezifiziert werden können.

Der Verein deutsche Ingenieure definiert in der Richtlinie 2221 die Funktion aus konstruktionsmethodischer Sicht als eine

> *lösungsneutral beschriebene Beziehung zwischen Eingangs-, Ausgangs- und Zustandsgrößen eines Systems* (VDI 2221, 1993).

Nach Pahl/Beitz beschreibt eine Funktion den gewollten

> *Zusammenhang zwischen Eingang und Ausgang eines Systems mit dem Ziel, eine Aufgabe zu erfüllen* (Pahl, 2007, p.42).

Roth unterscheidet die Begriffe Aufgabe und Funktion dergestalt, dass

> *eine Aufgabe den verbal dargestellten Zweck eines Produkts beschreibt, während eine Funktion zusätzlich ein System mit Ein- und Ausgängen abgrenzt* (Roth, 2000, p.3).

Ehrlenspiel definiert eine Funktion als relativ weit gefassten Parameter, der

> *bestimmte Aufgaben oder Zwecke beschreibt* (Ehrlenspiel, 2003).

2.2.1 Funktionsmodellierungsprozess

Wie bereits in Kap. 2.1.1 auf Seite 12 beschrieben, besteht das Ziel der Funktionsmodellierung in der Definition aller Funktionen, die ein Produkt zu erfüllen hat, sowie der Modellierung ihres funktionalen Zusammenwirkens (Rude, 1998, p.237). Dabei können im Wesentlichen Funktionen in Gesamt-, Teil-, Haupt- und Nebenfunktionen unterschieden werden. Die Gesamtfunktion wird aus der Aufgabenstellung hergeleitet und beschreibt die Gesamtheit aller Funktionen, die ein Produkt verwirklicht oder verwirklichen soll. Teilfunktionen lassen sich durch die Aufteilung einer übergeordneten Funktion gewinnen. Sie sind somit hierarchisch gegliedert und können entweder eine Haupt- oder eine Nebenfunktion darstellen. Eine Hauptfunktion beschreibt den Hauptzweck eines Produkts, wohingegen alle "Nicht-Hauptfunktionen" als Nebenfunktionen bezeichnet werden. (Kunze, 2002)

Das Erzeugen einer funktionalen Produktbeschreibung im Konstruktionsablauf beginnt mit dem Formalisieren der Aufgabenstellung. Dabei soll die Kreativität des Konstrukteurs unterstützt werden, indem von so genannten Vorfixierungen Abstand genommen wird. Zur Abstraktion der Aufgabenstellung gehört zunächst das Feststellen der Gesamtfunktion des zu entwickelnden Produkts. Daran schließt sich die Festlegung der aus der Aufgabenstellung ableitbaren Teilfunktionen an. Das Ergebnis dieses Abstraktionsvorgangs sind in der Regel die Aufgabensätze, die neben der funktionalen Aufgabenbeschreibung auch die zugehörigen Anforderungen enthalten können.

Aus diesen Aufgabensätzen lassen sich Teilfunktionen des Produkts extrahieren, die durch ein Substantiv-Verb-Konstrukt beschrieben werden. Dieses Denken in Funktionen begünstigt erfahrungsgemäß das Auffinden neuer, besserer Lösungen gegenüber einer zu frühzeitigen Festlegung auf ein bestimmtes Lösungsprinzip (Roth, 2000).

Die zur Gesamtfunktion eines technischen Produkts gehörenden Teilfunktionen können in einer hierarchischen Ordnung strukturiert werden. Eine derartige Struktur wird als Produktaufgabenstruktur bezeichnet (Lossack, 1997). Zunächst enthält eine Produktaufgabenstruktur die aus der Aufgabenstellung extrahierten Teilfunktionen des zu entwickelnden Produkts, die in einem sich anschließenden Schritt bzw. während des Konstruktionsablaufs weiter unterteilt werden können. Die Unterteilung in weitere Teilfunktionen enden, wenn die erhaltenen Teilfunktionen einen dem jeweiligen Zweck angemessenen, handhabbaren Komplexitätsgrad zur Lösungssuche angenommen haben (Pahl, 2007).

Ergänzend zu der Produktaufgabenstruktur eines Produkts können die Teilfunktionen auch in einer die wesentlichen Stoff-, Signal- und Energieflüsse beschreibenden Flussanordnung, der so genannten Funktionsflussstruktur, angeordnet werden (Krappe and Marinov, 2007, p.25). Diese Vorgehensweise beinhaltet allerdings in den meisten Fällen bereits eine Lösungsvorfixierung und ist somit weniger als eine Formalisierung der Aufgabenstellung zu betrachten, als dass sie im Zuge der Konkretisierung der Produktbeschreibung die Funktionsweise des Produkts beschreibt. Aus diesem Grund können für die Funktionsflussstruktur Varianten gebildet werden, die verschiedene alternative Funktionsflüsse eines Produkts bei gleichbleibender Hauptfunktion beschreiben (Huber, 1994).

2.2.2 Funktionsstrukturen

Es werden zwei grundlegende Typen von Funktionsstrukturen unterschieden. Dies ist zum einen die Allgemeine Funktionsstruktur, die auf den drei allgemeinen Größen Materie, Energie und Information sowie einer begrenzten Anzahl an allgemeinen Operationen basiert. Sie wird im Wesentlichen für die Formalisierung der Aufgabenstellung benutzt. Aus der Allgemeinen Funktionsstruktur kann zum anderen die so genannte Spezielle Funktionsstruktur abgeleitet werden. Sie entsteht in einem ersten Schritt aus einer Konkretisierung der allgemeinen Größen in spezielle Größen. (Kunze, 2002).

Die allgemeine Funktionsstruktur repräsentiert den funktionalen Zusammenhang des Gesamtproblems und ermöglicht das funktionelle Modellieren eines technischen Systems durch allgemeine Funktionen auf abstrakter Ebene (Krappe et al., 2007). Nach Huang ist eine allgemeine Funktion eine Funktion, deren

> *Funktionsgrößen Material, Energie und Information sind und aus Kombination mit den allgemeinen Funktionsverben gebildet werden können* (Huang, 2002, p.36).

Roth hat erarbeitet, dass jede technische Evolution durch eine neue Erfindung in den Bereichen Stoff bzw. Materie, Energie und Information hervorgebracht wurde (Roth, 2000, p.17). Die drei allgemeinen Funktionsgrößen werden dabei wie folgt definiert.

1. Materie ist jedes stoffliche Vorkommen in einem beliebigem Aggregatzustand.[§]
2. Energie ist jegliches Vermögen, Materie zu bewegen. Energie als allgemeine Größe umfasst alle physikalischen Größen jeder Fachdisziplin.
3. Information als Größe beschreibt die Änderung von Werten, beispielsweise durch Signale oder Impulse und bedarf stets materieller Informationsträger (Huang, 2002).

Bei der Untersuchung der Verknüpfung dieser allgemeinen Größen mit allgemeinen Funktionsverben variieren die Autoren hinsichtlich der Verben und ihrer Anzahl (Rude, 1998, p.24). Insbesondere der Übergang zwischen den allgemeinen und speziellen Funktionsstrukturen ist dabei methodisch oftmals nicht eindeutig. Dementsprechend hat das National Institute for Standards and Technology sich unter anderem zur Aufgabe gemacht, alle wissenschaftlichen Vorgehensweisen zur Funktionsmodellierung miteinander zu vergleichen, zu klassifizieren und Taxonomien in Hinblick auf den Übergang von der allgemeinen, abstrakten Ebene zu der speziellen, konkreten Ebene zu entwickeln. Dabei wurde eine funktionale Basis von Hirtz (Hirtz et al., 2002) durch Vergleiche der Ansätze von Pahl und Beitz (Pahl, 2007), Hundal (Hundal, 1990) und Altshuller (Altshuller, 1984) geschaffen. Vor dem

[§] Als Aggregatzustand bezeichnet man die feste, die flüssige oder die gasförmige Zustandsform von Materie. (Schulz, 1996)

Hintergrund der Abbildung bestehender Systeme bzw. entwickelter Produkte mit Flussstrukturen, bietet diese Klassifizierung den umfangreichsten Ansatz für die Modellierung auf allgemeiner Ebene und den anschließenden Übergang zu speziellen Funktionen (Schubert, 2007). Eine detaillierte Beschreibung sowie der stufenweise Übergang zu speziellen Funktionsverben und damit zu speziellen Funktionen geht aus Anhang A auf Seite 279 hervor. Die Tabelle 2.2 vergleicht exemplarisch die allgemeinen Funktionsverben einiger Konstruktionsmethodiker.

Pahl/ Beitz	**Roth**			**Koller**	**Hirtz**
speichern	speichern			speichern/ entspeichern	bereitstellen
leiten	übertragen	leiten		leiten	leiten
ändern		umformen		vergößerm/ verkleinern	steuern
wandeln	wandeln			wandeln/ rückwandeln	umwandeln
verknüpfen	verknüpfen	summativ	gleiche	fügen	verknüpfen
			ungleiche	verbinden	
		distributiv	gleiche	teilen	verzweigen
			ungleiche	trennen	
				Richtung ändern	
					signalisieren
					unterstützen

Tabelle 2.2: Vergleich unterschiedlicher Funktionsverben

In der Literatur existiert weder eine konkrete Klassifizierung in Hinblick auf spezielle Funktionsverben, noch auf Funktionsobjekte. Tatsächlich werden bei den Konstruktionstheoretikern eine Vielzahl von Mischformen vorgestellt. Nach Roth entspricht die spezielle Funktionsstruktur seiner allgemei-

nen Intensitäts-/ Quantitäts Funktionsstrukur und Pahl/ Beitz vermischen jeweils allgemeine Größen und spezielle Funktionsobjekte sowie allgemeine und spezielle Funktionsverben (Schubert, 2007).

Um ein Arbeiten mit speziellen Funktionen und Funktionsstrukturen zu ermöglichen, ist es aufgrund der hohen Anzahl an Funktionsverben und Funktionsobjekten notwendig, diese zu klassifizieren. Hierbei sind die Arbeiten von Hirtz (Hirtz et al., 2002) und Langlotz (Langlotz, 2000, p.92ff) zu erwähnen, da diese, aufgrund ihres sinngemäßen, stufenweisen Aufbaus, die weiteste Grundlage zur Modellierung mit speziellen Funktionsverben bietet.

2.2.3 Entwurfswerkzeuge

Ausgehend von den Ansätzen der System- und Wertanalyse wurden Methoden zur funktionalen Systembeschreibung wie SADT (Marca and MacGowan, 1988), IDEF0 (NIST, 1993), FAST (Fowler, 1981), EPK (Scheer, 1992) oder OMT (Rumbaugh and Märtin, 1994) entwickelt.

Die Structured Analysis and Design Technique ist eine graphisch orientierte Systementwurfs- und Beschreibungsmethode, die in den USA von der Firma SofTech Inc. entwickelt wurde. Basierend auf der Vorgehensweise des strukturierten Designs (SD) und Petri-Netzen hat SADT seine Ursprünge in der Softwareentwicklung. Durch die Modularisierung des Designs sollten neben der reinen Funktionshierarchie auch die Wechselwirkungen von übergeordneten Modulen beschreibbar gemacht werden. Dies wurde durch eine Top-Down-Vorgehensweise realisiert, d.h. der schrittweisen Konkretisierung des Systems, bis ein ausreichender Detaillierungsgrad erreicht war. Dabei bedient sich SADT einer einfachen Diagrammsprache, die als graphische Elemente zur Beschreibung der Systemaktivitäten und der auszutauschenden Daten Kästen und gerichtete Pfeile enthält, welche zur Beschreibung der Systemaktivitäten und der auszutauschenden Daten dienen.

Im Rahmen des ICAM Programms der US Air Force wurden in den siebziger Jahren Möglichkeiten zur Erhöhung der Produktivität durch Rechnereinsatz erforscht. Daraus sind eine Reihe von Modellierungsmethoden - die ICAM DEFinition Methods (IDEF) - entstanden. Als funktionales Modellierungskonzept von SADT wurde IDEF0 standardisiert. Mittels IDEF0 kann ein System beliebig tief detailliert werden, indem die Funktionen in Teilfunktionen, so genannten Aktivitäten, hierarchisch zerlegt werden. Ergebnis von IDEF0 ist ein funktionales Modell, welches die Produktdatenverarbeitung in Form einer Hierarchie der verwendeten Aktivitäten und der zwischen diesen

Aktivitäten liegenden Informationsflüssen wiedergibt. Alle Funktionen und Aktivitäten werden durch ein von einem Rechteck umschlossenen Verb, das die Aktivität bezeichnet, dargestellt. Die Informationsflüsse zwischen und zu den Aktivitäten werden durch Pfeile repräsentiert. Hierbei gilt die so genannte ICOM Notation, die in Bild 2.1 dargestellt wird.

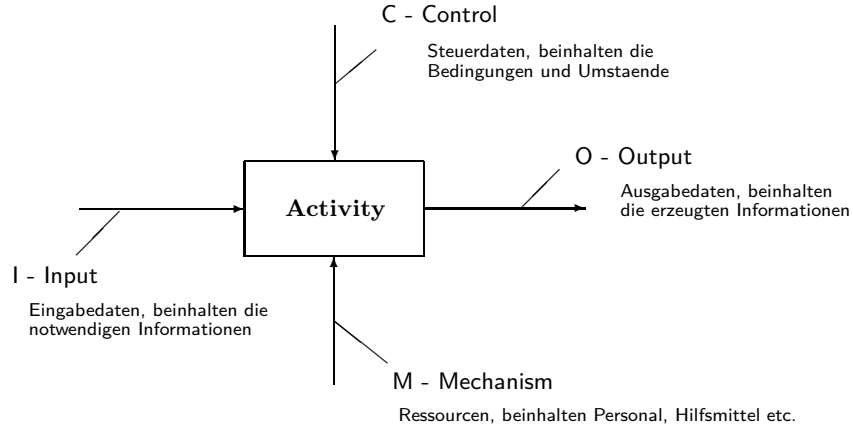


Bild 2.1: IDEF0 ICOM Notation

Die ereignisgesteuerte Prozesskette wurde an der Universität des Saarlandes entwickelt (Scheer, 1992) und dient der Darstellung von Geschäftsprozessen einer Organisation. Sie ist wesentliches Element des ARIS Konzepts. Dabei werden Prozesse in einer semiformalen Modellierungssprache grafisch als zeitlich-logische Abfolge von Funktionen dargestellt. Die beteiligten Objekte werden dazu in gerichteten Graphen mit Verknüpfungslinien und Pfeilen in einer $1/1$ Zuordnung verbunden. In einer solchen Verknüpfungskette wechseln die Objekte sich in ihrer Bedeutung zwischen Ereignis und Funktion ab, d.h. sie bilden eine alternierende Folge, die zu einem bipartiten Graphen führt.

Jede Funktion kann zusätzlich mit einem Informationsobjekt verbunden werden, aus dem Informationen geladen oder in das Informationen gespeichert werden. Aus EPK-Sicht ist ein Prozess eine Menge von Funktionen, die von einem oder mehreren Ereignissen ausgelöst werden und die für den Kunden des Prozesses ein Ergebnis von Wert erzeugen. Funktionen sind durch Ereignisse miteinander verknüpft. Ein Ereignis charakterisiert einen eingetretenen relevanten Zustand und ist auf einen Zeitpunkt bezogen. Sie können Funktionen auslösen als auch Ergebnisse von Funktionen sein. Ereignisse bilden die Ablauflogik der Funktionen und sind somit die zentralen Steuerungselemente innerhalb eines Prozesses. Eine Funktion wird durch mindestens ein Ereignis gestartet und endet in mindestens einem Ereignis. Folgerichtig wird das auslösende Ereignis als Startereignis bezeichnet.

Die FAST Methode hat ihre Ursprünge in der Wertanalyseperiode Anfang 1960. Dabei werden die Aktivitäten eines komplexen Systems in Funktionen, die das System für den Nutzer bietet, umgewandelt (Fowler, 1981). FAST kann sowohl bei Produkten als auch deren Prozessen angewandt werden. Gesamtfunktionen können dabei in Teilfunktionen sowie unerwünschte Funktionen zerlegt werden. Mit der weiteren Unterscheidung in wichtige, unwichtige und unnötige Funktionen werden Kostenschwerpunkte und somit Ansatzpunkte zur Kostensenkung ermittelt und Suchfelder für neue Lösungen zur Wertsteigerung eröffnet. Auf Basis des FAST Diagramms wird untersucht, wie durch Vereinfachung, Verbesserung und Änderung die gleiche oder nur eine unwesentlich schlechtere Qualität zu geringeren Kosten realisiert oder bei unveränderten Kosten ein höherer Nutzen geboten werden kann. Bild 2.2 zeigt exemplarisch das FAST Diagramm der Funktion *Wäsche waschen*.

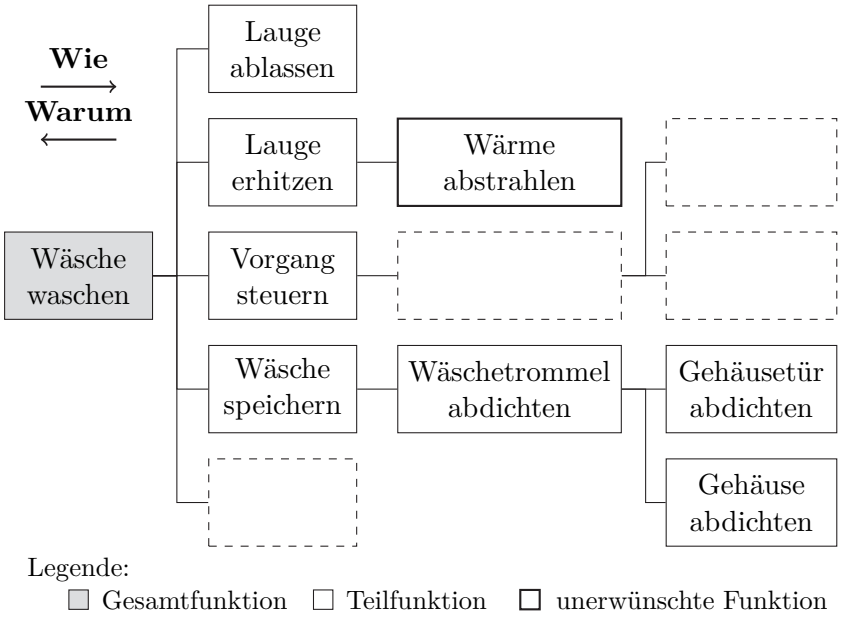


Bild 2.2: FAST Diagramm

Methoden aus der Softwareentwicklung dienen in der Regel zur Beschreibung des Verhaltens einer Funktion, und eignen sich darüberhinaus auch zur Funktionsmodellierung. Dem entsprechend stellt die OMT Methode, als Teil von UML, eine weit verbreitete Möglichkeit dar (Rumbaugh and Märtin, 1994). Das Ergebnis ist ein Datenflussdiagramm, in dem beschrieben wird, wie sich Ausgabewerte der Operationen aus den Eingabewerten

berechnen und ableiten lassen und wie Prozess, Daten und Datenspeicher zueinander in Beziehung stehen. Neben der Beschreibung des Verhaltens eines Systems werden darauf basierend Datenmodelle entwickelt. Bekannte Methoden zum Entwurf dieses Modells sind das Entity-Relationship-Datenmodell (vgl. Bild 2.3) (Chen and Knöll, 1991), die NIAM Daten-Modellierungsmethode (Wintraecken, 1990) und Express spezifiziert in der ISO 10303-21.

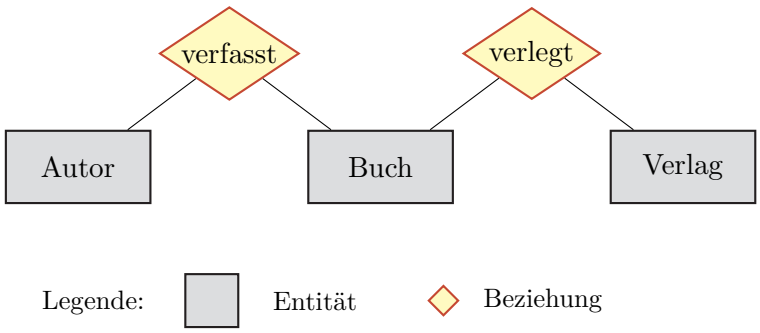


Bild 2.3: Entity-Relationship Diagramm

Obwohl eine Reihe von Funktionsmodellierungsmethoden insbesondere im Rahmen der Softwareentwicklung konzipiert wurde, ist eine ingenieursgerechte Lösung bislang nicht vorhanden. Die Integration von Funktionsstrukturen mit den Produktaufgabenstrukturen, wie sie aus der Anforderungsmodellierungsphase entstehen, oder in Produktstrukturen eines CAD Systems ist allenfalls am Rande umgesetzt worden oder gar nicht vorhanden. Funktionale Aspekte sind lediglich indirekt, durch angehängte Zusatzinformationen wie Toleranzangaben, Spaltmaße oder Spannstellenbeschreibungen als eine Information zur Erfüllung der Funktion *Verbindung schaffen* beschreibbar und dienen als Informationsbasis für spätere Prozessschritte (Leemhuis, 2005).

Im Sinne der Kopplung von Prinziplösungen mit PDM Sytemen, hat beispielsweise Dassault sein CAD System dahingehend erweitert, dass funktionale Produktbeschreibungen durch Suchmöglichkeiten, in einer, nach der TRIZ Methode (Altshuller, 1984) arbeitenden Datenbank, integriert wurden. Mit der Nutzung von TRIZ wird dem Konstrukteur Unterstützung bei der Ideengenerierung geboten. Als weiteren Beitrag zur Berücksichtigung und Dokumentation funktionaler Aspekte wurde das System CATIA um ein Zusatzmodul, den Optimizer, erweitert, in dem die Beschreibung

funktionaler Beziehungen der Einzelgeometrien untereinander, im Sinne von "welche Wirkung hat ein Bauteil oder geometrisches Objekt auf ein anderes" erweitert. In darüber hinausgehenden Arbeiten zum funktionalen Entwurf wurden CAD Systeme mit wissensbasierten Entwurfsumgebungen wie beispielsweise ICAD oder Design++ gekoppelt, die angepasst an disziplinspezifische Prozesse die Auslegung unterstützen (Leemhuis, 2005).

Im Forschungskontext haben sich zahlreiche Arbeiten mit einer integrativen Lösung zur Funktions-/ und Gestaltmodellierung innerhalb eines Systems befasst. Zu erwähnen sind hier die Systemlösungen DIICAD (Kunze, 2002) aus Karlsruhe, KALEIT (Kuttig, 1993) aus Berlin, WISKON (Eulenbach, 1990) aus Kassel, KIEF aus Tokyo (Tomiyama et al., 1989), das System IIICAD aus Amsterdam/Tokio (Tomiyama et al., 1989), das System RA/Q-SE (Irgens, 1991), sowie die Arbeiten von Chakrabarti (Chakrabarti et al., 2002) aus Cambridge und Chandrasekaran (Chandrasekaran et al., 1993) aus Ohio hervorzuheben. Jedoch konnte keines dieser Systeme sich in der Praxis etablieren.

Als Schlussfolgerung kann aus diesen konstruktionsmethodischen Grundlagen das folgende Ergebnis formuliert werden. Die funktionale Produktbeschreibung existiert auf unterschiedlichen Abstraktionsniveaus, beginnend mit der Formalisierung der Aufgabenstellung durch einen begrenzten Satz an Grundfunktionen bis hin zur Beschreibung der Produkttopologie und der Wirkzusammenhänge. In Abhängigkeit von der Abstraktionsebene der funktionalen Produktbeschreibung können jeweils verschiedene Beschreibungsformen definiert werden. Die Funktionen eines Produkts können in Hierarchien und/oder flussorientierten Netzwerken strukturiert werden. Die funktionale Produktbeschreibung wird im Konstruktionsprozess zum einen für die Beschreibung der Funktionsweise des Produkts und zum anderen für das Wiederfinden von bekannten, bereits ausgeführten Lösungen verwendet. Eine ingenieursgerechte Modellierungstechnik wurde in Softwaresystemen bislang gar nicht oder nur unzureichend umgesetzt. Integrative Ansätze, die durchgängig von der Anforderungsmodellierung bis zur Gestaltmodellierung ein Produkt abbilden können, wurden zwar im Forschungskontext erarbeitet, konnten sich aber in der Praxis nicht etablieren.

Nach der Diskussion der konstruktionsmethodischen Grundlagen, d.h. der methodischen Vorgehensweisen bei der Produktentwicklung auf unterschiedlichen Abstraktionsebenen, insbesondere der funktionalen Betrachtung von technischen Produkten, wird im folgenden Abschnitt auf die Grundlagen der 3D Modellierungsverfahren eingegangen.

2.3 3D Modellierverfahren

Modellierung zählt neben Animation und Rendering zu den Kernbereichen der Computeranimation. Ähnlich wie in den zwei anderen Bereichen stehen je nach Anwendungsgebiet und Anforderung viele unterschiedliche Zugänge zur Verfügung. Für Computeranimationen werden andere Modellierungswerkzeuge und auch andere Softwarepakete eingesetzt als z.B. im CAD-Bereich, in dem aus Daten reale Objekte erzeugt werden. Bei Modellen für Echtzeitanwendungen muss besonders auf die Topologie der Geometrie geachtet werden und es können ausschließlich Polygone eingesetzt werden (Kunstuniversity Linz).

2.3.1 Einteilung von 3D Systemen

Generell lassen sich 3D Modellierungssyteme heute meist nach den geometrischen Modellen, d.h. den Informationen zur Festlegung der Gestalt von Bauteilen unterschieden (Grabowski, 2000). Die klassifizierenden Merkmale für geometrische Modelle sind in Tabelle 2.3 dargestellt. Zur Klassifizierung von rechnerinternen Modellen wird vereinfacht meist nur die Dimensionalität des Elementraums benutzt. Man spricht dann von einem 2D oder 3D Modell.

Geometrisches Element	Klassifizierungsmerkmal	
	Dimensionalität der geometrischen Elemente	Dimensionalität des Elementraums
Ebene Linien	1D	2D
Räumliche Linien	1D	3D
Ebene Flächen	2D	2D
Gekrümmte Flächen	2D	3D
Volumen	3D	3D

Tabelle 2.3: Dimensionalität geometrischer Körper und der Elementräume

Hinsichtlich der, bei der Modellbildung verwendeten, geometrischen Elemente wird generell in drei Arten unterschieden, dem Linienmodell oder

auch Draht- bzw. Kantenmodell (wireframe model), dem Flächenmodell (surface model) und dem Volumenmodell (solid model).

Desweiteren wird bezüglich der Art der Abbildung der Informationen und ihrer Ablage im Speicher eines Rechners im Wesentlichen zwischen den generativen Modellen und akkumulativen Modellen unterschieden. Bei den generativen Modellen ist die Information in Form einer Erzeugungsvorschrift des Modells im Speicher abgelegt. Alle Modellinformationen sind hierbei implizit vorhanden. Nach außen hin werden jedoch meist nur Darstellungsinformationen sichtbar. Bei den akkumulativen Modellen ist die Menge der Modellinformationen getrennt vom Erzeugungsprogramm als Datenstruktur abgelegt. Die Konsistenz muss durch spezielle Programme überprüft werden (Grabowski, 2000). Eine Mischung beider Modellarten wird hybrides Modell genannt. Hybride Modelle vereinigen meist die Vorteile beider Modellarten, führen jedoch mitunter zu Datenredundanz.

In den kommenden Kapiteln sollen kurz die verschiedenen 3D Modellierverfahren vorgestellt werden. Diese bilden die Grundlage zur Erzeugung und späteren Darstellung aller Objekte in einem dreidimensionalen Raum. Dabei soll zuerst auf die Volumenmodellierung, und danach auf die direkte und indirekte Modellierungsverfahren eingegangen.

2.3.2 Volumenmodellierung

Die Volumenmodellierung kann als Weiterentwicklung vorhergehender Ansätze der zweidimensionalen rechnerunterstützten Modellierung (rechnerunterstützte Zeichnungserstellung) und der dreidimensionalen Flächenmodellierung (Beschreibung von Geometrien durch räumlich gekrümmte Flächen) im Hinblick auf eine möglichst umfassende und realitätsnahe Beschreibung von Produkten mit Methoden und Werkzeugen der Rechnerunterstützung angesehen werden (Kunze, 2002).

Das Ziel der Volumenmodellierung ist es, die komplette geometrische Information von Objekten im dreidimensionalen Raum zur Verfügung zu stellen.

Instanzen eines Volumenmodells werden in einem virtuellen Raum erstellt. Ein so erstellter Volumenkörper besitzt eine geschlossene Hülle, welche die eindeutige Definition von innen und außen zulässt. Dadurch ist es möglich, Kollisionen zwischen Objekten zu überprüfen. Das Volumen kann dabei direkt oder indirekt modelliert werden (Bungartz et al., 2004):

- *Direkte* Darstellungsschemata konstruieren das Volumen aus einfachen Grundkörpern. Beispiele dafür sind die Sweep Repräsentation, das Constructive Solid Geometry Modell oder das Subdivision Surface Modell.
- *Indirekt* wird das Volumen über seine Hülle beschrieben. Der typische Vertreter ist das Boundary Representation Modell. Die Beschreibung der Hülle geschieht mit ebenen oder beliebig geformten Flächen, so genannter Freiformflächen.

Direkte Modellierungsverfahren

Die grundlegenden Verfahren zur Erstellung direkter Darstellungsschemata sind die CSG Methode, das Subdivision Surface Modellierungsverfahren, die Metaballs Methode, oder die Sweep Repräsentation. Diese sollen im Folgenden kurz erläutert werden.

Constructive Solid Geometry Das CSG-Modell ist ein generatives Modell. Den Aufbau beschreibt Bild 2.4. Es handelt sich um eine Objektbeschreibung in Form einer Rechenvorschrift durch einen Binärbaum, dessen Knoten Mengenoperationen enthalten und dessen Blätter Basiselemente sind.

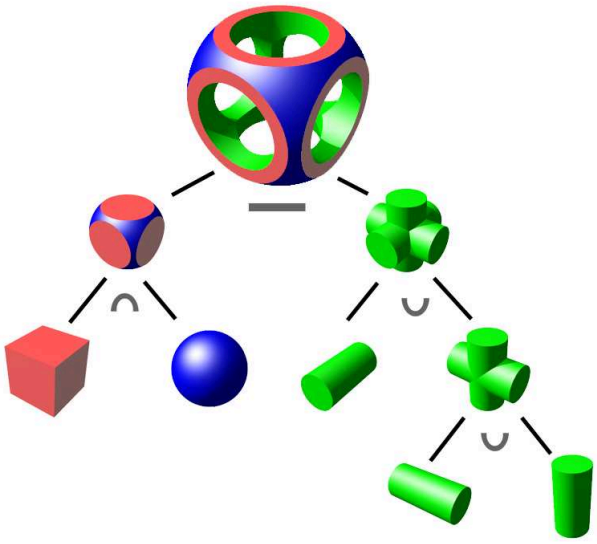
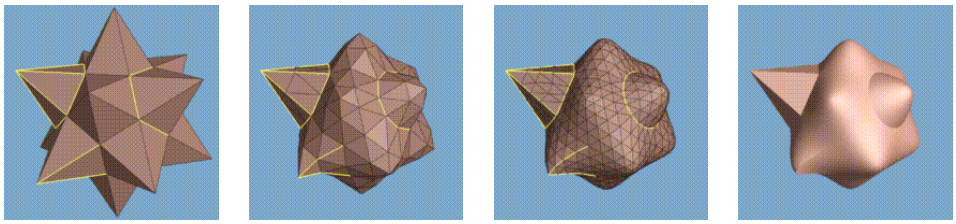


Bild 2.4: CSG Baum (ComputerBase)

Die Beschreibung des Objekts erfolgt gewissermaßen implizit, da die Rechenvorschrift für das "Sichtbarmachen" des Objekts erst ausgeführt werden muss (Grabowski, 2000). Im Gegensatz zu Polygonen und NURBS beschreiben diese Objekte nicht nur eine Fläche, sondern ein geschlossenes Volumen. Diese Volumina können nun mit Hilfe boolescher Algebra kombiniert werden. Bild 2.4 zeigt die boolsche Verknüpfung von Grundkörpern. Zur späteren Darstellung werden die sich ergebenden Oberflächen trianguliert, oder es wird direkt mit den mathematischen Beschreibungen gearbeitet. Letzteres erlaubt eine sehr detaillierte Darstellung auch bei sehr hohen Auflösungen.

Subdivision Surface Modelling Das Subdivision Surface Modelling ist eine Top-Down Modellierungsart, in der vom Abstraktem zum Konkreten anhand der Subdivision surfaces modelliert wird. Das Subdivision Surface bezeichnet die Fähigkeit, Geometrien zu festgelegten Bereichen der Oberflächen, in denen mehr Details benötigt werden, hinzuzufügen ohne Auswirkung auf andere Bereiche. Dabei stellt das Subdivision Surfaces die Verbindung von NURBS und Polygonen dar. Hierbei wird ein polygonales Grundobjekt, z.B. ein Würfel oder eine Kugel, erzeugt und dann durch die Subdivision Surfaces in einem beliebigen Detailgrad abgerundet. Mit den normalen Polygonmodellierungsfunktionen wird nun das Grundobjekt bearbeitet und so eine grobe Fassung des gewünschten Objekts erstellt. Durch die Subdivision Surfaces wird dieses grobe Polygonmodell zu einem weichen abgerundeten Modell verfeinert (Encarnação et al., 1995). Das fertige Modell besteht wiederum aus Polygonen. Die Polygone des groben Modells werden also unterteilt ("subdivided") und so an die NURBS-Fläche angenähert. Hoppe postuliert in (Hoppe et al., 1994) das Subdivision Modelling als Nachfolger der NURBS Methode. Bild 2.5 zeigt exemplarisch den Vorgang.



(a) Ausgangsnetz (b) Erster Verfeinerungsschritt (c) Zweiter Verfeinerungsschritt (d) Endzustand

Bild 2.5: Subdivision Surface Modelling (Hoppe et al., 1994)

Box Modelling Beim Box Modelling oder auch Volume Modelling wird eine komplexe Form aus einem einfachen Grundobjekt, meistens ein Quader, per Hand herausgearbeitet. Box Modelling ist wie das Poly/Edge Modelling Teil des Subdivion Surface Modellings. Beide sind Top-Down Modellierungsmethoden. Die meisten kommerziellen 3D-Pakete bieten die Möglichkeit, auf frühere Modellierschritte zurückzugreifen. Dies ermöglicht ein einfaches, schnelles und intuitives Arbeiten, gerade bei organischen Objekten und in Zusammenarbeit mit Techniken wie einem Unterteilungsflächenalgorithmus.

Polygon Modelling Das Polygon oder Edge Modelling basiert ebenfalls auf dem Subdivision Surface Modelling. Analog zum Box Modelling wird hier jedoch nicht von einem Grundkörper ausgegegangen, sondern das Modell wird Vertex für Vertex und Face für Face aufgebaut. Die Polygon Modellierung kann weiter in Hi Poly und Low Poly Modellierung unterteilt werden, bei denen die Anzahl der Details pro Geometrie differiert.

Metaballs Das Metaballs Modellierungsverfahren, auch häufig als Blobs bezeichnet, beschreibt die polygonalen Flächenannäherungen an eine isometrische Oberfläche. Ein Metaball ist das Ergebnis eines Algorithmus, der eine dehnbare Oberfläche erzeugt, die die Form einer Kugel oder eine Menge von ineinander gehenden Kugeln erzeugt (Blinn, 1982). Im Laufe der Zeit haben sich neue Grundformen entwickelt, wie zum Beispiel Zylinder oder Würfel. Darüber hinaus existieren auch Meta-Splines, die sich wie Splines verhalten, die mit einer Metaball-Oberfläche überzogen sind. Damit ist die Modellierung komplexer Körpern dem Subdivision Surfaces Modelling ähnlich. Bild 2.6 zeigt zwei sich voneinander entfernende Metaballs.

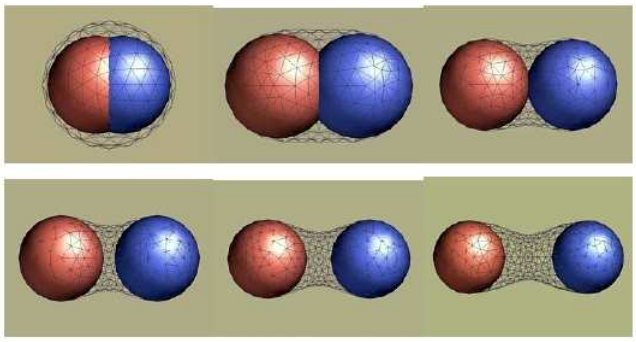


Bild 2.6: Zwei sich voneinander entfernende Metaballs (Ibanez et al., 2000)

Sweep Repräsentation Wird eine 2D Kontur oder ein Raumkörper durch den Raum bewegt, entsteht ein Sweep. Beim Sweeping, auch Lofting genannt, wird ein Objekt aus der Teilmenge des Raums erzeugt, die von einem vorhandenen Objekt während der Bewegung entlang einer bestimmten Kurve eingenommen wird. Dadurch ändert sich die Form des Objektes mit der Bewegung. Der Körper, welcher beim Sweeping entsteht, muss allerdings nicht immer ein Volumenkörper sein. Wird beispielsweise eine zweidimensionale Fläche in ihrer Ebene bewegt, so entsteht wieder ein zweidimensionaler Bereich.

Indirekte Modellierungsverfahren

Neben Modellierungstechnicken wie Boxmodellierung, CSG oder Zusammenstellen von Primitives können Flächen auch aus Kurven aufgebaut und bearbeitet werden. Diese indirekten Modellierungsverfahren sind die topologisch geometrischen Strukturmethoden (B-Reps), die splinebasierten Verfahren wie NURBS oder Patch Modelling oder das Displacement. Diese sollen im Folgenden kurz erläutert werden.

Boundary Representation Um auch komplexe Teile mit Volumenmodellierern beschreiben zu können, wurden die sogenannten B-Rep Modellierer entwickelt. In B-Rep-Modellen werden Körper durch Eckpunkte, Kanten und Begrenzungsflächen und deren topologischen Zusammenhängen beschrieben, so dass die geometrische Begrenzung des Körpers rechnerintern exakt abgebildet ist. B-Rep Modelle bilden im Gegensatz zu CSG Modellen beispielsweise die Geometrie eindeutig ab. Ihr wesentlicher Nachteil ist, dass die Entstehungsgeschichte im Modell nicht enthalten ist, so dass Änderungs- und Löschoperationen schwer möglich sind (Kunze, 2002).

Splines Splines sind Nachfolger der Bézier Kurven. Der Begriff Spline bezeichnete ursprünglich lange, flexible Metallstreifen, mit denen technische Zeichner die Oberfläche von Schiffen, Flugzeugen oder Autos konstruierten. An dem Metallstreifen wurden Gewichte angebracht, um ihm eine natürliche Krümmung zu geben (Bungartz et al., 2004). Werden Spline Kurven mit mehreren Punkten verwendet, um Flächen zu erzeugen, können generell zwei Arten differenziert werden:

- planare Kurven, bei denen alle Kontrollpunkte sich in einer Ebene befinden und

- nonplanare Kurven, bei denen die Kontrollpunkte sich nicht in einer Ebene befinden.

Planare Flächen können aus einer geschlossenen bzw. aus mehreren geschlossenen Kurven erstellt werden. So werden z.B. aus Buchstaben Flächen. Wird eine planare Kurve um eine Rotationsachse gedreht, entsteht ein Rotationskörper, der je nach Rotationswinkel und Geometrie-Output variieren kann. Die Extrusion einer planaren Kurve meint das "Herausziehen" dieser in die dritte Dimension. Dabei kann zwischen Extrusion mit Distanz, Pfadextrusion und Extrusion an mehreren Profilkurven unterschieden werden (Kunstuniversity Linz).

NURBS Non-Uniform rational B-Spline sind eine Erweiterung der Splines. Im Gegensatz zu Polygonnetzen mit vielen Dreieckflächen werden die Nurbs-Flächen und -Kurven mathematisch beschrieben. So wird ein einfaches und intuitives Modellieren möglich. Dabei erscheinen die Kanten nicht facettiert.

Splinecage/Patch Modellierung Bei der Splinecage Modellierung wird mit Hilfe von Splines ein Netzwerk konstruiert. Danach wird mit Hilfe von Oberflächenmanipulatoren eine Patchoberfläche über das Splinecage gelegt. Eine Patch-Fläche ist dabei eine aus drei bzw. vier non-planaren Kurven generierte Fläche. Mehrere Patches werden zu einer komplexen Form zusammengesetzt. Die Form der einzelnen Flächen wird durch die Kurven definiert. Meist werden zuerst Raumkurven gezeichnet, aus denen mehrere Patches generiert werden. Nurbs-Patches entstehen aus Nurbs-Kurven; Bézier-Patches sind Flächen, die auf Bézier-Kurven basieren. (Kunstuniversity Linz)

Displacement Modellierung Displacement Maps sind Graustufentexturen, die flache Nurbs oder Polygon Oberflächen geometrisch verändern. Die Maps besitzen Höheninformationen, die direkt an der Geometrie weitergegeben werden. Displacementmaps benötigen eine hohe Meshauflösung. Daraus resultieren meist sehr lange Renderzeiten.

Die im Kontext dieser Arbeit notwendigen 3D Modellierungsmethoden wurden hinsichtlich ihrer wesentlichen Charakeristika diskutiert. Zur Erstellung von dreidimensionalen Objekten, die nicht den hohen Detaillierungsgrad eines CAD Modells bedürfen, sind Verfahren wie das Box Modelling hervorragend geeignet. Speziell für das Echtzeitrendering, in dem die Polygonanzahl

möglichst gering gehalten werden muss, um einen flüssigen Ablauf zu gewährleisten, sind die direkten Modellierungsverfahren besser geeignet.

2.4 Datenschnittstellen

Eine Datenschnittstelle dient zur Kommunikation zwischen verschiedenen Kommunikationsteilnehmern, z.B. zwischen Menschen und Maschinen, oder zwischen zwei Maschinen (Halbach, 1994). Die Informationen sollen nicht nur für Menschen lesbar sein, sondern auch von Computern interpretiert und weiterverarbeitet werden können. Auch zur Kommunikation zwischen Programmkomponenten ist der Einsatz einer Datenschnittstelle notwendig. Im Rahmen dieser Arbeit wird eine Datenschnittstelle zwischen dem Modellierungswerkzeug zur Funktionsmodellierung und der virtuellen Umgebung benötigt. In diesem Kapitel werden die einsetzbaren Datenschnittstellen vorgestellt. (Zhou, 2008, p.14)

2.4.1 Softwareschnittstellen

Softwareschnittstellen oder softwareseitige Datenschnittstellen sind logische Berührungspunkte in einem Softwaresystem. Sie definieren, wie Kommandos und Daten zwischen verschiedenen Prozessen und Komponenten ausgetauscht werden. Dabei werden Schnittstellen zum Zugriff auf Systemroutinen, zur Kommunikation mit anderen Prozessen und zum Verbinden einzelner Softwarekomponenten eines Programmes bzw. programmübergreifende Schnittstellen unterschieden. (Halbach, 1994)

Interprozesskommunikation

Die Übermittlung von Daten zwischen zwei Prozessen oder Anwendungen kann über, vom Betriebssystem angebotenen, Kommunikationsmechanismen erfolgen (Tanenbaum, 2001), (Thies, 1994). Jede Anwendung wird vom Betriebssystem auf einem eigenen Speicherbereich eingeschränkt. Um jedoch einen Austausch von Daten zwischen Anwendungen zu erlauben, stellen Betriebssysteme Mechanismen wie Datenströme, Ereignisse oder Shared Memory zur Verfügung. Diese Mechanismen werden unter anderem durch gemeinsame Bereiche im Hauptspeicher implementiert, so dass die IPC die

schnellste Methode zum Austausch von Daten zwischen Anwendungen darstellt. Dafür erfordert deren Einsatz eine Synchronisation der Anwendungen, um die Konsistenz der Daten zu gewährleisten. Diese Synchronisation wird ebenfalls vom Betriebssystem durch so genannte Semaphore[¶] oder Monitore unterstützt, kann aber zu gegenseitigem Blockieren der Anwendungen, dem so genannten Deadlock führen und erfordert deswegen genaue Kenntnis der beteiligten Softwarekomponenten. Außerdem sollen die Austauschmechanismen von beiden Anwendungen bekannt sein. Die Einbettung in bereitstehende Systeme ist im Allgemeinen nicht möglich (Gray, 1997). Diese Art von Datenschnittstellen eignen sich für Datenübermittlungen, die strengen Zeitanforderungen unterliegen, wobei die beide Kommunikationsteilnehmer Eigenentwicklungen sind. (Zhou, 2008, p.14)

Datenbanken

Datenbanken bieten die Möglichkeit, Daten strukturiert auf dem Dateisystem zu speichern (Sauer, 2002), (Matthiessen and Unterstein, 2003). Datensätze werden in tabellarischen Strukturen dargestellt und von einem Datenbanken-Verwaltungssystem (DBMS) in Dateien gespeichert und verwaltet. Der Zugriff auf die Daten erfolgt über die standardisierte Abfragesprache SQL. Der Einsatz von Datenbanken für den Datenaustausch hat den Vorteil einer standardisierten Schnittstelle, die Sprache SQL, und wird bei den meisten Datenbanken-Verwaltungssystemen durch Transaktionsmechanismen unterstützt, so dass Seiteneffekte beim verteilten Lesen oder Schreiben von Daten transparent beseitigt werden. Relationale Datenbanken haben jedoch den Nachteil, eine feste, tabellarische Datenstruktur zu besitzen, und jede Änderung der Struktur der Datensätze erfordert einen neuen Entwurf des Datenbankschemas. Daher ist deren Einsatz nur für fest definierte Datenstrukturen geeignet. Zudem erfordert der Einsatz von Datenbanken die Inbetriebnahme einer dritten Software, des Datenbanken-Verwaltungssystems. (Zhou, 2008, p.15)

[¶]Ein Semaphor, von griechisch $\sigma\varepsilon\mu\alpha$ = Zeichen und $\varphi\varepsilon\rho\varepsilon\iota\nu$ = tragen, ist eine Datenstruktur mit zwei speziellen Nutzungsoperationen. Semaphore werden bei der Programmierung zur Prozesssynchronisation eingesetzt, also zur Lösung von Aufgaben, bei denen die parallele Ausführung mehrerer Prozesse/Threads eine zeitliche Abstimmung der Ausführungen erfordert (Tanenbaum, 2002).

Eigenes Dateiformat

Die zu ermittelnden Daten können auch direkt in Dateien gespeichert werden, wobei das Datenformat selbst definiert werden kann. Somit ist diese Lösung flexibler als Datenbanken für die Strukturierung der Daten und die Schwierigkeit der Synchronisation durch Zwischenspeicherung auf dem Dateisystem beseitigt. Bei komplexen Anwendungen ist die Datenbank jedoch übersichtlicher. Die Einbindung in Modellierungswerkzeuge lässt sich oftmals durch die Entwicklung eines Plugin realisieren, das den Export in dem selbstdefinierten Dateiformat unterstützt. Die Nachteile sind die Einführung eines nicht-standardisierten Formats und der daraus resultierende zusätzliche Aufwand für die Realisierung von Plugins, so dass Anwendungen dieses Dateiformat importieren und exportieren können. (Zhou, 2008, p.18)

2.4.2 Auszeichnungssprache XML

XML stellt eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien dar (Harold, 2002), (Ray, 2003). Es handelt sich um eine vom W3C herausgegebene Standardsprache für semi-strukturierte Daten (Mintert, 2004).

XML ist eine sowohl für Menschen als auch für Computerprogramme leicht lesbare und schreibbare Sprache. Außerdem hat sich XML weit in der Industrie verbreitet und viele bekannte Firmen wie Oracle, IBM, Microsoft haben bereits den XML Standard in ihren Produkten eingesetzt.

Überblick

Folgend ist ein Beispiel eines XML Dokuments dargestellt:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<Fahrzeug>
   <Marke id="1">
      <Name>Volkswagen</Name>
      <Land>Deutschland</Land>
   <Marke id="2">
      <Name>Peugeot</Name>
      <Land>Frankreich</Land>
</Fahrzeug>
```

Die Daten werden in einer Baumstruktur dargestellt, wobei die Knoten dieses Baums durch folgende Syntax beschrieben werden:

$$<Bezeichner\ [Attribut]^*> Inhalt <\!/Bezeichner>$$

Bezeichner ist der Name des Knotens oder Tags und kann beliebig definiert werden. Somit ist die Sprache XML, im Gegensatz zu ähnlichen Sprachen wie HTML, frei definierbar und erweiterbar und für die Definition einer eigenen domänenspezifischen Datenstruktur geeignet. Ein Knoten kann beliebig viele selbstdefinierte Attribute enthalten, die in der Form

$$Bezeichner = Wert$$

abgebildet werden. Zwischen dem eröffnenden Tag $< \cdots >$ und dem schliessenden Tag $< / \cdots >$ befindet sich der Inhalt. Dieser kann aus Knoten, die wiederum einen XML-Baum darstellen, und/oder freiem Text bestehen (Harold, 2002).

Die einfache Struktur sowie die frei definierbaren Tags und Attribute ermöglichen eine einfache Benutzung und Lesbarkeit des Formats, ohne Vorkenntnisse von komplexen rechnerbezogenen Begriffen. XML Dokumente lassen sich mit normalen Text-Editoren erstellen und visualisieren. Darüber hinaus existieren eine Vielzahl von XML-Editoren, die Funktionalitäten zur Überprüfung der synktatischen Korrektheit anbieten. (Zhou, 2008, p.16)

Ein programmatischer Zugriff auf XML Dokumente wird in den meisten Fällen durch den Einsatz von XML-Parser unterstützt. Diese stellen die Baumstruktur des Dokuments als entsprechend verlinkte Objekte der verwendeten Programmiersprache dem Entwickler zur Verfügung. Über eine wohldefinierte API können diese Objekte erzeugt bzw. Zugriff ermöglicht und somit die XML Datei editiert werden. Die Implementierung eines Parsers und einer API für die Bearbeitung von XML Dokumenten existiert für die meisten objektorientierten Sprachen wie PHP, C++, C# und Java. Beispiel dafür ist die Bibliothek System.XML des .NET Frameworks. (Zhou, 2008, p.16)

Datenabfrage und Verarbeitung

Da XML lediglich ein Sprachkonzept zur Entwicklung von Sprachen bildet, genügt es nicht, dass zwei Systeme XML beherrschen, um gegenseitig Daten austauschen zu können. Sie müssen die gleiche XML basierte Sprache

verwenden. Für den Fall, dass die Systeme nicht die gleiche XML basierte Sprache verwenden, wird in der Regel ein MOM wie XSLT zum Zwecke des Datenaustausches eingesetzt, um, mit Hilfe von Transformationen, Übersetzer von der einen Sprache in die andere zu entwickeln (Mangano, 2006), (Bongers, 2004).

XSLT ist eine Programmiersprache zur Transformation von XML Dokumenten, die die Umwandlung eines XML Dokuments in ein anderes Dokument mit einer anderer Struktur beschreibt. Mit XML werden die notwendigen Daten einmalig angelegt, und über eine XSLT können diese gefiltert und in verschiedenen Strukturen wiedergegeben werden (Bongers, 2004).

Die XSL Transformation besteht aus einer Reihe von einzelnen Transformationsregeln und basiert auf der XML Anfragesprache XPath. XPath wurde neben dem Standard XML spezifiziert und ähnelt der Abfragesprache SQL für Datenbanken. Dabei unterstützt es im wesentlichen die Suche von Daten in einer XML Struktur (Simpson, 2002). Durch eine solche Abfrage kann eine Menge von Knoten über deren Position in der Hierarchie oder über deren Eigenschaften wie den Werten von Attribute ausgewählt werden. Dementsprechend werden die zu transformierenden Teile eines XML Dokumentes durch XPath adressiert, und durch die definierten XPath Regeln in das gegebene Zielformat umgewandelt (Mangano, 2006), (Zhou, 2008, p.17).

Die vorgestellten Methoden der Datenschnittstellen beschreiben die Grundlagen einer auf XML basierten Datenübertragung zwischen Systemen. Dabei soll der XML Standard im Verlauf der Arbeit verwendet werden, da er über einen sehr generischen Aufbau verfügt, und von einer großen Anzahl von System unterstützt wird.

Im folgenen Kapitel sollen die Verfahren zur Visualisierung von Informationen im dreidimensionalen Raum vorgestellt werden.

2.5 Informationsvisualisierung

Die computergestützte Visualisierung von Informationen gewinnt immer mehr an Bedeutung, da die Ressource Information an sich immer wichtiger wird. Information avanciert zu einem grundlegenden, strategischen Produktionsfaktor. Die entstehende internationale Informationsinfrastruktur, sprich das Internet, macht Informationen überall und zu jeder Zeit verfügbar (Englberger, 1995, p.10).

2.5.1 Visualisierung in Informationsräumen

Um den Begriff der Informationsvisualisierung zu verstehen, muss vorab der Raum, in dem die Information visualisiert wird, abgesteckt werden. Dieser enthält die abstrakten Informationen, d.h. Dokumente, Begriffe oder andere textbasierte Informationen aus Datenbanken, Retrievalsystemen oder dem Internet. In der Literatur ist nur schwer eine Definition des Informationsraumes zu finden. Der Informationsraum ist

> *durch eine Informationsmenge und eine Informationsstruktur definiert, wobei eine Informationsstruktur eine Relation ist, die die Beziehungen zwischen den Informationsobjekten beschreibt.* (Wünsche, 1997)

Eine spezifische Information kann auf diese Weise auch in unterschiedlichen Informationsräumen repräsentiert werden.

Der Begriff der Visualisierung wird anhand einer Reihe von Definitionen geprägt. Zur Zeit wird die Bezeichnung im täglichen Sprachgebrauch für nahezu jede Software verwendet, die irgendeine Form der grafischen Darstellung eines Sachverhaltes hervorbringt (Najarro, 2003). Einige Beispiele für verschiedene Definitionen seien hier in Form von Zitaten aufgeführt:

> *Visualisierung wird als eine Bezeichnung für die bildliche Formulierung und Kommunikation definiert, d.h. für die Aufbereitung von Information mit vor allem bildlichen Mitteln wie auch für die visuelle Wahrnehmung.* (Brockhaus, 2006, Band 23)

> *Visualisierung ist ein spezieller Bereich der Computergrafik, der sich vorrangig mit der Darstellung von physischen Parametern oder Objekten befasst. Visualisierung wird vornehmlich im wissenschaftlichen Bereich bei der Interpretation massenhafter und komplexer Daten eingesetzt.* (Dässler and Palm, 1998)

> *Visualisierung definiert die Umwandlung von Informationen, die ursprünglich nicht in Bildform vorliegen, in eine meist grafische Darstellung.* (Charwat, 1994)

Aus den obengenannten Definitionen wird ersichtlich, daß der Begriff der Visualisierung, in Abhängigkeit der verschiedenen Forschungsinteressen durchaus unterschiedlich interpretiert wird. In jeder Definition wird vom Begriff

der Information gesprochen, wobei aber selten der Versuch einer Definition dieses Begriffes unternommen wird.

Der Brockhaus betrachtet Information als eine

> *Auskunft, Nachricht, Mitteilung, Belehrung, eine formulierte Unterrichtung, die ursprünglich aus dem Lateinischen von Gestaltung, Bildung abgeleitet wurde* (Brockhaus, 2006).

Wesentlich für die Information ist die Wiedererkennbarkeit sowie der Neuigkeitsgehalt, so dass der bestehende Wissensschatz erweitert werden kann. Die Information ist somit eine Nachricht, die beim Empfänger eine Veränderung im Denken und Handeln auslösen kann.

Laut DIN 44300 sind Informationen

> *Kenntnisse über Sachverhalte und Vorgänge.*

Kuhlen definiert Information als die Teilmenge von Wissen,

> *die von jemandem in einer konkreten Situation zur Lösung von Problemen benötigt wird* (Kuhlen, 1995).

Einig sind sich alle Autoren darin, dass Informationen immer den Inhalt einer Nachricht, in textlicher, grafischer oder audiovisueller Form bilden. Dabei enthalten sie keine irrelevanten oder redundanten Teile.

Aus den oben genannten Definitionen leitet Dässler die Definition der Informationsvisualisierung ab. Dieser betrachtet alle Konzepte, Methoden und Werkzeuge zur visuellen Darstellung abstrakter Informationen. Die abstrakten Informationen haben bei der Visualisierung keine physikalische Datenbasis. Informationsvisualisierung beinhaltet

> *die rechnerunterstützte Aufbereitung und die visuelle Repräsentation abstrakter Informationen mit dem Ziel, den kognitiven Zugang zu elektronisch gespeicherten Daten zu erleichtern* (Dässler and Palm, 1998).

Ziel der Informationsvisualisierung ist es, die Leistungsfähigkeit des menschlichen visuellen Systems auszunutzen, um große Datenbestände zu explorieren, den kognitiven Zugang zu inhärenten Informationen zu erleichtern und

ihre relevanten Charakteristiken intuitiv zu erfassen. (Fekete and Plaisant, 2002; Schumann and Müller, 2004)

Um dieses Ziel zu erreichen, hat Shneiderman das Mantra der Informationsvisualisierung vorgeschlagen (Shneiderman, 1996)

> *Overview first, Zoom and Filter, then Detail-on-Demand.*

Hiermit wird ein prinzipielles Vorgehen festgelegt, das in nahezu allen Anwendungen zur Informationsvisualisierung verwirklicht wird. Dabei ist der Ausgangspunkt üblicherweise ein Überblicksbild (Overview), welches der Orientierung dient und allgemeine Eigenschaften einer Informationsmenge geeignet repräsentiert. Mittels Zooming kann der Fokus auf Teilbereiche des Bildes gelenkt werden. Interessante Bereiche können vergrößert und weniger relevante Bereiche vereinfacht dargestellt werden. Schließlich muss es möglich sein, auf Anforderung, Bilder mit beliebigen Details von Interesse (Detail-on-Demand) zu generieren (Schumann and Müller, 2004, p.135).

2.5.2 Klassifikation der Informationsvisualisierung

Zur Klassifikation der Informationsvisualisierung stehen verschiedene Taxonomien zur Verfügung. Schumann und Müller (Schumann and Müller, 2004, p.137) gliedern die Methoden nach der Art der Visualisierung. Dabei wird eine explizite Darstellung einer Impliziten gegenüber gestellt.

Shneiderman (Shneiderman, 1996, p.2f) klassifiziert hinsichtlich der Daten, die der Visualisierung zugrunde liegen, sowie der gewünschten Informationsvisualisierung des Benutzers. Er unterscheidet insgesamt sieben Datentypen

- 1-dimensionalen Daten wie Listen, Zeitleisten etc.
- 2-dimensionale Daten wie Landkarten, Zeitungslayouts etc.
- 3-dimensionale Daten wie reale oder virtuelle Objekte
- multidimensionale Daten wie relationale und statistische Datenbanken
- Text und Hypertext aus Webdokumenten
- Hierarchien und Graphen wie Unternehmensstrukturen oder Produktstrukturen
- Algorithmen und Software

Keim (Keim, 2002, p.2ff) erweitert die Klassifikationen von Shneiderman und Schumann um die Art der verwendeten Interaktionstechnik zu einem dreidimensionalen, orthogonalen Klassifikationsmodell. Er geht davon aus, dass für jeden Datentyp, jede Visualisierungstechnik mit jeder Interaktionstechnik kombiniert und somit jede Informationsvisualisierungstechnik in sein Modell eingeordnet werden kann.

Englberger (Englberger, 1995, p.20f) unterscheidet in seiner Klassifikation anhand der Intention. Dabei werden drei Arten voneinander abgegrenzt, die Präsentations-, die Retrieval- sowie die Explorationsvisualisierung. Bei der Präsentationsvisualisierung sind sämtliche visualisierten Informationen von Bedeutung. Es sollen Sachverhalte veranschaulicht werden, die bereits bekannt und fixiert sind. Die Retrievalvisualisierung hingegen betrachtet einen gewissen Ausschnitt der resultierenden Visualisierung. Der Benutzer weiß lediglich vage, nach welcher Information er sucht. Im Gegensatz zur vorigen Variante ist bei der Explorationsvisualisierung nur ein relativ kleiner Anteil der erreichten Visualisierung von Belang. Der Benutzer muss eine geräumige Informationsressource analysieren, denn er ist vorab über die Disposition der Daten nicht im Bilde. Vorweg wird ungerichtet nach optischen Auffälligkeiten gesucht, um Korrelationen zu extrahieren, sowie Abweichungen aufzudecken. (Scherer, 2007)

2.5.3 Methoden der Informationsvisualisierung

Noch vor Einführung des Begriffs der Informationsvisualisierung, gab es Mitte der 80er an der Universität in Singapore Überlegungen, neue VR-Technologien zu benutzen, um Informationsräume aufzubauen, in denen der Benutzer in einem Netz von abstrakten Objekten und Relationen frei navigieren konnte. Daraus entwickelte Kim Fairchild 1988 eines der ersten Visualisierungssysteme für Informationen überhaupt, genannt SemNet. (Dässler, 1999)

Einige Zeit später, zu Beginn der 90er Jahre, wurden bei XEROX PARC (Palo Alto Research Center) eine ganze Palette neuer visueller Metaphern für Informationsräume entwickelt (Card et al., 1996; Mackinlay et al., 1991; Robertson and Mackinlay, 1993; Robertson et al., 1991). Am bekanntesten sind die perspektivische Wand sowie der rotierende Kegelbaum, der in Bild 2.8 auf Seite 45 abgebildet ist. In der Folgezeit wurden weitere Visualisierungsmetaphern entwickelt und in einer Reihe von Applikation adaptiert und modifiziert.

Die entstandenen, unterschiedlichen Ansätze basieren alle auf denselben Grundlagen. Unabhängig von der Verwendung der Ausgabegeräte, wird mittels eines dreidimensionalen Raumes die vorhandene Präsentationsfläche effektiver genutzt. Die interaktive Animation erlaubt es, einen Teil der kognitiven Belastung auf das menschliche Wahrnehmungsvermögen zu übertragen. Häufig soll das Display sowohl einen detaillierten Workspace enthalten, als auch den globalen Kontext wiedergeben (Robertson and Mackinlay, 1993, p.101). Nichtsdestotrotz haben nur wenige Prototypen aus den Entwicklungslabors der Informationsvisualisierung bisher Produktreife erlangt. Zu den bekanntesten Produkten zählen im Bereich Data Mining MineSet (Silicon Graphics), im Textmining-Bereich Spire (Pacific Northwest National Laboratory), Semiomap (Semio Corporation), Perspecta (Perspecta) oder die VizControls-Technologie der XEROX-Firma InXight (Dässler, 1999). Eine Zusammenfassung mit den bedeutenden Entwicklungen und prototypischen Tools auf dem Gebiet der Informationsvisualisierung ist in Tabelle 2.4 auf der nächsten Seite dargestellt.

Um die Informationsvisualisierung so effizient wie möglich zu gestalten, sollten die Methoden möglichst auf präattentiven graphischen Features basieren, die durch unterbewusste Wahrnehmung verarbeitet und auf einen Blick erkannt werden. Hierdurch kann die Explorationszeit, die ansonsten proportional zur Anzahl der enthaltenen Objekte ist, wesentlich verkürzt werden (Fekete and Plaisant, 2002, p.118). Healey hat eine Liste dieser präattentiv erkennbaren Merkmale zusammengestellt. Dazu gehören Linienorientierung, Länge, Breite, Größe, Krümmung, Anzahl, Schattengrenze, Überschneidung, Begrenzung, Farbnuancierung, Intensität, Flimmern, Bewegungsrichtung, Glanz, stereoskopische Tiefe und Beleuchtungsrichtung (Healey et al., 1995, p.190ff).

Die im folgenden diskutierten Informationsmodellierungsmethoden werden exemplarisch und bezüglich der im Kontext dieser Arbeit nutzbringenden Methoden vorgestellt. Dabei wurde die intentative Klassifizierungsmethode nach Englberger, wie in Kapitel 2.5.2 auf Seite 40 beschrieben, verwendet, die zusätzlich in die Kategorien Informationsbäume, Informationskarten und Informationsräume unterteilt wurde.

Informationsbäume

Die wesentlichen Vertreter aus dem Bereich der Informationsbäume (engl. Information Trees) sind die hyperbolischen Bäume, die Cone/Cam Trees

Zeit	**Visualisierungsmethode**	**exemplarische Umsetzung**
1988	Fisheye View	SemNET
1990	Perspective Wall	Information Visualizer
	Cone Tree	Lyberworld
	Parallel Coordinates	ILOG Discovery
1992	Information Landscape	File System Navigator
1994	Hyperbolic Tree	WebVIZ
		Themescape
	Fly Through	HyperSpace
		HotSauce
1995		GopherVR
1996	Web Visualisierung	MineSet
		SemioNET
		WebSOM
	Web Book	WebForager
1997	Dokumentkarte	MapDisplay
1998	VRML basierte Informationsräume	Perspecta View
2000		VizControls
2002	Stardinates	Time-Tunnel
	CourseViz	WebCT
2004	InfoSky	Infosky visual explorer
	CircleView	CircleView XAMPP
2005	Dust and Magnet	Piccolo
2006	CirclePacking	Circle Pack
	Radial Hierarchical Visualisation	CircleView XAMPP

Tabelle 2.4: Chronologischer Überblick der Methoden zur Informationsvisualisierung

sowie die Perspective Walls, die im Folgenden näher beschrieben werden sollen. Darüber hinaus fallen in diesen Bereich ebenfalls die von Card vorgeschlagenen Web book und Web Forager Workspaces (Card et al., 1996), die von Heflin erarbeitete WebTOC Methode (Heflin et al., 1998), die auf der RDF Technologien basierende AuroraSitemap Methode (nDimension, 2007), die Perspective Tunnel Metapher (Mitchell and Kennedy, 1997) sowie der File System Navigator von SGI (Weal, 1996). Diese sind im Kontext dieser Arbeit jedoch zweitrangig und werden nicht näher betrachtet.

Hyperbolische Bäume Hyperbolische Bäume ermöglichen die Visualisierung großer hierarchischer Strukturen. Die Baumhierarchie wird in einer hyperbelartigen Ebene ausgebreitet und auf eine zweidimensionale, kreisförmige Fläche projiziert. Den Knoten samt ihren Nachfolgern werden keilförmige Stücke zugewiesen. Ein Überschneiden der gebogenen Kanten ist ausgeschlossen. In der Nähe des Kreisrandes wird der Baum kontinuierlich kleiner dargestellt, was eine intuitive Fisheyesicht (Furnas, 1986, p.16ff) erzeugt. Da mit steigendem Abstand vom Mittelpunkt der Kreisumfang zunimmt, ist es möglich, auch die wachsende Knotenmenge tieferer Baumschichten zu zeigen (Englberger, 1995, p.60f). Bild 2.7 zeigt einen hyperbolischen Baum.

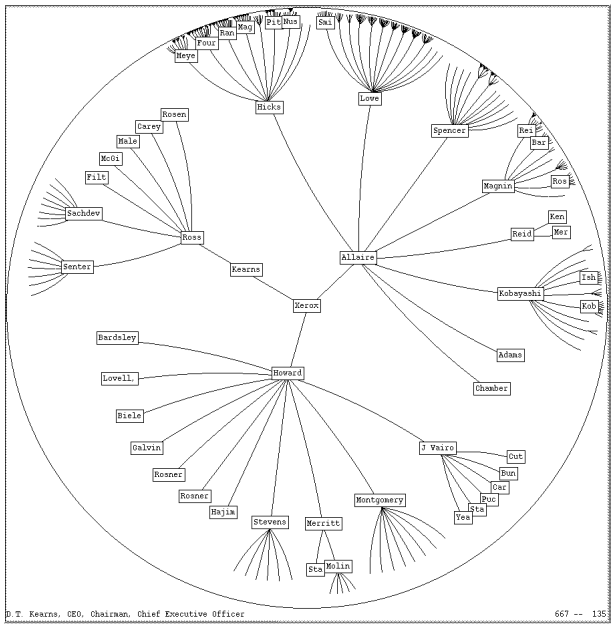


Bild 2.7: Hyperbolischer Baum mit 1004 Knoten nach (Lamping et al., 1995)

Cone/Cam Tree Cone Trees sind gerichtete dreidimensionale Graphen, die uniform in alle drei Raumrichtungen ausgelegt sind und deren Knoten wie Karteikarten dargestellt werden (siehe Bild 2.8). Im Zuge einer Vererbungshierarchie besteht zwischen den einzelnen Ebenen eine Eltern-Kind Beziehung. Die Kinder werden in gleichmäßigen Abstand entlang der Basis verteilt. Die nächste Knotenebene wird unterhalb der Ersten gezeichnet, ihre Kinder werden wiederum als Kegel dargestellt. Das Aspektverhältnis wird so festgelegt, dass der gesamte Baum auf dem Bildschirm dargestellt werden kann. Die Kegelflächen werden halbtransparent gerendert. Auf diese Weise können die Kegel leicht wahrgenommen werden, ohne die Sicht auf weiter hinten liegende Kegel zu blockieren. Wird ein Knoten selektiert, rotiert der Cone Tree diesen und alle Knoten seiner zugehörigen Hierarchielinie nach vorne und hebt sie farblich hervor. Die dreidimensionale Ansicht des Cone Trees erzeugt einen Fisheye View (Furnas, 1986, p.16ff) auf die Informationen. Durch die 3D Perspektive und das Beleuchtungsmodell wird der ausgewählte Pfad heller und größer dargestellt als die anderen Pfade.

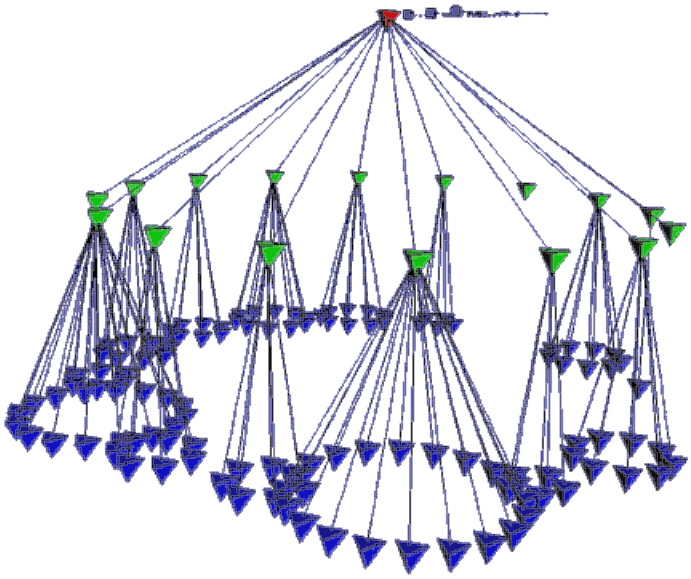


Bild 2.8: Darstellung eines Cone Trees nach (Munzner and Burchard, 1995)

Zur Objektmanipulation stehen verschiedene Interaktionstechniken zur Verfügung, die der Anwender im Stile eines virtuellen Gärtners nutzen kann. Durch Trimmen können bestimmt Knoten entfernt werden, um den Fokus auf spezielle Ausschnitte der Struktur zu legen. Ähnlich dem Veredeln kann die Struktur der Hierarchie dynamisch verändert werden, indem Knoten,

samt der in diesem Knoten verwurzelten Substruktur, an eine neue Position versetzt werden. Eine Suchfunktion ist mittels eines Texteingabe-Pop-ups implementiert. Die Visualisierung der Suchergebnisse geschieht entweder durch farbliche Hervorhebung der Knoten, oder durch Ausblenden der nicht relevanten Knoten. Cone Trees sind sehr effizient einsetzbar zur Visualisierung mittelgroßer, unregelmäßiger Hierarchien mit bis zu 1000 Knoten oder 10 Ebenen (Robertson et al., 1991, p.189ff).

Neben dem Cone Tree findet auch der Cam Tree Anwendung. Hierbei handelt es sich um einen herkömmlichen Cone Tree, der auf der Seite liegt und dessen Spitze am rechten Bildschirmrand positioniert wird. Die Funktionalität des Cam Trees entspricht in vollem Umfang der des Cone Trees. (Scherer, 2007, p.28)

Perspective Wall Resnikoff legt in seinen Studien dar, dass das menschliche Auge die enormen Informationsmengen, die in der realen Welt zur Verfügung stehen, nur dadurch aufnehmen kann, indem eine fokussierte Ansicht für die Details in eine grobe, unscharfe Ansicht für die Orientierung, integriert wird. Dem entsprechend muss eine geeignete Visualisierung das Resultat genereller Informationsverarbeitungsprinzipien, wie die selektive Ausblendung irrelevanter Informationen oder die Aggregation in abstrakte Formen, enthalten (Resnikoff, 1989).

Die Perspective Wall wurde von Mackinlay (Mackinlay et al., 1991, p.173ff) auf Basis dieses Prinzips entwickelt. Die physische Metapher des Faltens wird dazu verwendet, ein beliebiges 2D Layout in eine dreidimensionale Visualisierung zu überführen. Die Wand hat im Zentrum eine Tafel zur Darstellung von Informationen mit hohem Detaillierungsgrad. Auf jeder Seite befindet sich eine perspektivisch verkürzte Tafel, mit deren Hilfe der User den Kontext im Auge behalten kann. Zur Verstärkung der 3D Wahrnehmung, wurde die Beleuchtung so gewählt, dass diese Tafeln schattiert gerendert werden. Die vertikale Dimension der Wand dient zusätzlich zur Visualisierung verschiedener Ebenen des Informationsraumes.

Informationskarten

Die originären und wesentlichen Vertreter aus dem Bereich der Informationskarten (engl. Information Maps) sind die von Shneiderman entwickelten Tree Maps aus dem Jahr 1992, die kurz erläutert werden sollen. Neben den Tree Maps exisitieren weiterhin, die Self-Organizing Maps (Honkela

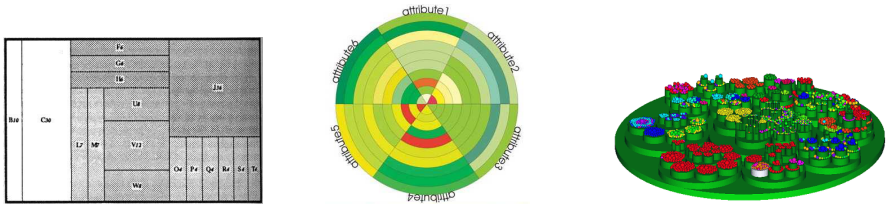
et al., 1996), die Quantum Strip Maps (Bederson et al., 2002), Visual SiteMaps (Lin, 1997), SPIRE (Havre et al., 2006), die Circle View oder Radial Hierarchical Visualization Methode (Keim et al., 2004) oder das Circle Packing (Wang et al., 2006).

Tree Maps Tree maps sind eine raumfüllende Visualisierungsmethode zur Repräsentation von großen, hierarchischen Kollektionen quantitativer Daten. Eine Treemap wird aufgebaut, indem der Bildschirmausschnitt in eine verschachtelte Sequenz von Rechtecken unterteilt wird, deren Flächeninhalt mit einem Attribut des Datensatzes korrespondiert. (Shneiderman and Wattenberg, 2001, p.73ff)

Tiefere Schichten sind jeweils innerhalb ihrer Vorgänger angeordnet. Die Blätter des Baumes sind quasi aus einer Froschperspektive zu sehen. Das ergibt eine Art Mosaik, wobei jede Kachel einen aktuellen Knoten repräsentiert. Die Lage der Rechtecke und eine variabel ausgeprägte Umrandung sollen die logische Baumstruktur widerspiegeln. Zusätzlich lässt sich durch Farbe, Ton, Beschriftung und Blinken Information kodieren. (Englberger, 1995, p.64f)

Neben den Ur-Tree maps wurden beispielsweise Cluster Treemaps (Wattenberg, 1999, p.188f) und Squarified Treemaps (Bruls et al., 2000, p.33ff) oder Ordered Treemaps (Shneiderman and Wattenberg, 2001, p.73ff) entwickelt. Diese verwenden Algorithmen, die bei dynamischen Updates relativ weiche Übergänge erzeugen, die Ordnung grob erhalten und Rechtecke mit niedrigen Aspektverhältnissen erzeugen. Auf diese Weise werden die Lesbarkeit und die Benutzerfreundlichkeit signifikant verbessert.

Bild 2.9 zeigt exemplarisch die Visualisierung der Treemaps, des Circle Views und des Circle Packings.



(a) Treemaps (b) Circle View (c) Circle Packing

Bild 2.9: Verschiedene Information Maps nach (Johnson and Shneiderman, 1991; Keim et al., 2004; Wang et al., 2006)

Informationsräume

Der Bereich der Informationsräume umfasst alle Metaphern, die im Rahmen der in Kapitel 2.5.1 auf Seite 38 vorgestellten Definition zutreffen. Dabei soll im Kontext dieser Arbeit insbesondere die Information Cubes, die Information Landscapes sowie die Document Lense hervorgehoben werden, da diese wesentlich in die Lösungsfindung einfließen. Darüberhinaus fallen in diesen Bereich die Arbeiten von Wood zur Hyperspace Visualisierung (Wood et al., 1995), Leads von Ingram (Ingram and Benford, 1995) und dass darauf basierende VR-Vibe von Benford (Benford et al., 1995), in welchem der Nutzer mehrere Schlüsselbegriffe wählen und in einem 3D-Raum plazieren kann, die Volltextdatenbank Visualisierungsmetapher Lyberworld (Hemmje et al., 1994), der Graphvisualiser, der zur Visualisierung von vernetzten Informationen, insbesondere von objektorientiertem Software-Code geschaffen wurde (Ware, 1998) sowie WWW3D von Snowdon (Snowdon, 1996). Systeme wie DIVE (Dive, 1991), in welchem internet-basiert Teilnehmer im 3D-Raum navigieren und andere Teilnehmer sehen, treffen und mit ihnen interagieren können, Mitre, Hyper-G oder Perspecta (Turetken and Sharda, 2007) sind hier ebenfalls zu nennen, finden aber im weiteren Verlauf keine weitere Beachtung.

Information Cube/Corridor Der Information Cube ist eine 3D Visualisierungstechnik für hierarchische Informationen. Er basiert auf der "nested box" Metapher (engl. geschachtelte Kisten), bei der Hierarchieebenen durch verschachtelte, halbtransparente Boxen repräsentiert werden. Die äußere Box stellt die oberste Datenebene dar, während die nächst niedrigere Datenebene mit den Würfeln innerhalb der äußeren Box korrespondiert. Jeder Würfel der zweiten Ebene enthält Würfel der dritten Ebene und so weiter. Jede Box erhält eine Bezeichnung auf der Oberfläche, Würfel der untersten Ebene präsentieren die enthaltenen Informationen als Kacheln auf ihrer Oberfläche. Bild 2.10 zeigt den Information Cube.

Die Informationsdarstellung ist jedoch keineswegs auf Textdokumente beschränkt, die Würfelmetapher kann beliebige dreidimensionale Objekte enthalten. Die Würfel werden in einer räumlichen Darstellung mittels eines konventionellen Monitors oder in einer VR Anlage angezeigt. Durch die Verwendung eines Datenhandschuhs oder einer ähnlichen Schnittstelle kann der Anwender den äußeren Würfel rotieren und verschieben. Wird ein Würfel der zweiten Ebene selektiert, zoomt das System herein und skaliert die in-

nen liegenden Würfel bildschirmfüllend.

Bild 2.10: Information Cube nach Rekimoto (Rekimoto and Green, 1993)

Ihre Semitransparenz bewirkt einerseits, dass ihre Nachfolger im Innern sichtbar sind, und andererseits, dass fern liegende, tiefe Ebenen graduell verdeckt werden. Auf diese Weise wird die Komplexität der Darstellung auf dem Bildschirm selbst bei extrem vielschichtigen Hierarchien immer auf einem angemessenen Level gehalten. (Rekimoto and Green, 1993, p.125ff)

Eine Abwandlung des Information Cubes ist der Information Corridor (vgl. Bild 2.11). Er basiert auf einer ähnlichen Metapher. Im Gegensatz zum Information Cube, bei dem der äußere Würfel den Informationsraum abschließt, ist der Informationsraum des Information Corridors jedoch unendlich.

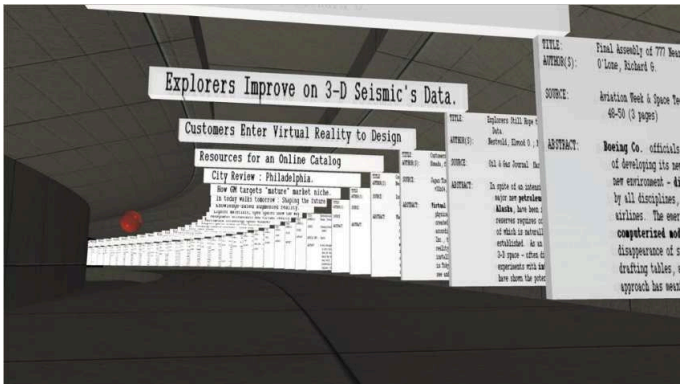


Bild 2.11: Information Corridor nach (O'Neil and Muir, 1997)

Der Anwender bewegt sich einen virtuellen Flur entlang, Zugang zu Informationen erhält der Benutzer, indem er, genau wie bei einem realen Gebäude, Türen öffnet. Diese geben Informationen entweder direkt frei, oder ermöglichen den Zugang zu weiteren Sub-Korridoren (Scherer, 2007, p.33). Bild 2.11 zeigt einen Information Corridor, der Informationstafeln anstatt Türen nutzt, um in Subräume zu gelangen oder Informationen zu erhalten.

Information Landscape Die dreidimensionale Information Landscape dient der Visualisierung von Dateiverzeichnissen. Allgemein lassen sich in dieser virtuellen Landschaft große hierarchische oder vernetzte Datenbestände auf vielfältige Weise ausbreiten. Hier werden Knoten im Raum als Sockel abgebildet, auf welchen wiederum kleine Quader positioniert sind. Dadurch wird der Informationsgehalt samt den Datenattributen dargeboten. Dabei können Fläche und Höhe der Quader, als auch Farbe und Beleuchtung genutzt werden. Zudem werden Ikons und Beschriftung verwendet. Die Interaktion erfolgt hauptsächlich durch Änderung der Perspektive. Als eine Art Drahtgeflecht erscheinen die strukturellen Beziehungen in der Ebene. Dabei ist die ganze Szene perspektivisch verzerrt in die Bildschirmebene projiziert. Dies erweckt einen impliziten Fokus auf nahe gelegene Objekte. (Scherer, 2007, p.26)

Aufgrund der ebenen Anordnung der Datenstruktur ist der Platzbedarf der Information Landscape sehr groß. Ohne eine explizite Übersichtskarte bietet dieses Modell keinen Überblick über die gesamte Struktur. Dementsprechend ist eine ungerichtete Suche nach Daten wenig ergiebig. Hohe Anforderungen werden jedoch an die Rechenleistung sowie an die Speicherkapazität der Hardware gestellt. Die immersive Visualisierung ist sinnvoll, um das virtuelle Eintauchen in den Informationsraum zu ermöglichen. (Englberger, 1995, p.40f)

Document Lens Die Document Lense basiert auf einer allgemeinen Strategie, die zum Verständnis von Dokumenten angewendet wird, deren Struktur a priori unbekannt ist. Die Seiten eines Dokuments werden in einem rechteckigen Feld ausgelegt, die Gesamtstruktur wird sichtbar und interessante Muster werden erkennbar. Um einen Teil des Dokumentes zu fokussieren und gleichzeitig den globalen Kontext zu erhalten, wird ein so genanntes “Focus und Context“ Display benötigt (Robertson and Mackinlay, 1993, p.103). Wird ein Suchalgorithmus verwendet, der die Ergebnisse farblich hervorhebt, können Muster im gesamten Dokument erkannt werden. Das

Anwendungsspektrum der Document Lens umfasst nicht nur Textdokumente, sondern alle Arten von Dokumenten, die in einer 2D Ebene ausgelegt werden können, wie Graphen, Landkarten und ähnliches.

Die im Kontext dieser Arbeit notwendigen Informationsvisualisierungsmethoden wurden hinsichtlich ihrer wesentlichen Eigenschaften diskutiert. Dabei wurde der Fokus auf diejenigen Verfahren gelegt, die im Laufe der Lösungsfindung verwendet werden sollen. Es wurde festgestellt, dass die Zahl visueller Metaphern, die in kommerziellen Programmsystemen implementiert wurden eher gering ist. Viele prototypische Systeme wurden am Benutzer vorbei entwickelt. Oftmals wurden Projekte nur unter dem Aspekt der Anwendung neuer Technologien aus dem Bereich der Computergrafik oder Datenvisualisierung durchgeführt. Eine inhaltliche Auseinandersetzung mit der Frage, ob und wie abstrakte Daten unter dem Aspekt eines Informationsmehrwertes aufbereitet und visualisiert werden sollten, wurde meist nicht geführt. Im Interfacedesign ist zudem die Benutzerakzeptanz ein entscheidendes Qualitätskriterium. Hierzu fehlen jedoch oft aussagekräftige Evaluationen (Dässler, 1999, p.13).

Laut Keim wird die Informationsvisualisierung in Zukunft noch stärker hervortreten. Er erklärt, dass

> *the ultimate goal is to bring the power of visualization technology to every desktop to allow a better, faster, and more intuitive exploration of very large data resources. This will not only be valuable in an economic sense, but will also stimulate and delight the user.* (Keim, 2002, p.6)

Als Schlüssel zum Erfolg wird das Data Mining, also die Aufbereitung der Informationen, gesehen. Visualisierung kann die Qualität der Aufbereitung und Klassifikation von Informationen nicht verbessern. Ganz im Gegenteil. Oft werden die Schwächen mangelnder Informationsaufbereitung durch den Visualisierungsprozess offenbart. Die Visualisierung kann letztlich nur so gut sein wie das Data Mining, das ihr zugrunde liegt. (Dässler, 1999; Schumann and Müller, 2004, p.140)

Im folgenden Kapitel sollen kurz die Grundlagen der virtuellen Realität diskutiert und daraufhin Metaphern aus diesem Bereich detailliert vorgestellt werden.

2.6 Virtuelle Realität

Virtual Reality gilt als Wachstumsmarkt und wird heute immer mehr als eine Schlüsseltechnologie für innovative Unternehmen angesehen. Der ehemalige US-Vizepräsident und Friedensnobelpreisträger Al Gore bezeichnete Virtual Reality als entscheidenden Wirtschaftsfaktor, der den Entwurf und die Entwicklung neuer Produkte, die Weitergabe von Wissen und unsere eigene Freizeit nachhaltig verändern wird (Hamilton et al., 1997).

Insbesondere Großkonzerne aus den Bereichen der Luft- und Raumfahrt, der Petrochemie und des Automobilbaus erkennen, dass die Technologie der virtuellen Realität Nutzenpotentiale bietet, die gewinnbringend erschlossen werden können. Im Wettlauf um kürzere Entwicklungszeiten, Fehlerreduktion in der Produktentwicklung, geringere Produktionskosten und verbesserte Produktqualität können weltweit agierende Hersteller technisch anspruchsvoller interdisziplinärer Produkte kaum noch auf diese Technologie verzichten. (Burdea and Coiffet, 1994; Durlach and Mavor, 1995; Leston et al., 1996)

2.6.1 Definition, Merkmale und Begriffsabgrenzung

Bevor es die Bezeichnung Virtual Reality gab, wurde die Idee einer sichtbaren, interaktiven und computergenerierten Scheinwelt Cyberspace genannt. Myron Krueger gab ihr den Namen Artificial Reality (Krueger, 1983). Bis zur Wortschöpfung der virtuellen Realität existierten so klangvollen Bezeichnungen wie Augmented Virtuality, Augmented Reality, Synthetic Environments, Mixed Reality, Simulated Reality, Real Reality, Natural oder Graspable User Interfaces, Physical Manipulation Interfaces, Tangible Media, immersive oder non-immersive Virtual Reality, Virtual Environments, Ubiquitous Computing oder Wearable Computing. Die Kreation des Begriffs Virtual Reality wird Jaron Lanier zugeschrieben und ins Jahr 1988 datiert (Rügge, 1999). Dabei versteht Lanier unter dem Begriff virtuell, das elektronische Abbild eines Objektes, welches keine konkrete Gegenständlichkeit besitzt. Populärwissenschaftliche Autoren wie Henning definieren VR als eine

> *neue Generation von Mensch-Maschine-Schnittstelle, die gleichzeitig ein völlig neues Medium, welches die zwischenmenschliche Kommunikation unter Einbeziehung aller Sinne in eine Scheinwelt verlagern kann, darstellt.* (Henning, 1997, p.9)

Die Vielzahl der in der Literatur zu findenden Definitionen lässt sich auf zwei unterschiedliche Schwerpunkte zurückführen. Zum Einen steht die eingesetzte Technologie zur Implementierung der Mensch-Maschine-Schnittstelle im Vordergrund (Astheimer et al., 1994, p.282f). Autoren wie Leston definieren VR dementspechend als

> *eine Menge von 3D Grafik- und Simulationswerkzeuge und Technologien, die den Benutzern erlauben, innerhalb einer computergenerierten Umgebung interaktiv und in Echtzeit zu operieren, ohne sich der Mensch-Maschine-Schnittstelle überhaupt bewußt zu sein* (Leston et al., 1996, p.25).

Zum Anderen steht der menschliche Benutzer und seine Empfindung hinsichtlich dieser neuen Schnittstelle im Vordergrund (Aukstakalnis and Blatner, 1992, p.7-10). Jonathan Steuer beispielsweise setzt den Begriff VR mit Telepresänz gleich und definiert diesen Begriff wiederum als

> *die Vorstellung des menschlichen Benutzers, sich mit Hilfe eines geeigneten Kormnunikationsmediums in einer bestimmten (realen oder virtuellen) Umgebung zu befinden* (Steuer, 1995, p.74-79).

Im Rahmen dieser Arbeit wird unter Virtueller Realität eine

> *der künftigen Mensch-Maschine-Schnittstellen verstanden, die mit dem Einsatz innovativer Ein- und Ausgabegeräte den Benutzer in eine dreidimensionale, interaktive, rechnerinterne Welt einbezieht.*

Für Bergbauer umfasst VR dabei alle Technologien, die zur Definition[‖] und echtzeitfähigen Aufbereitung[**] eines rechnerinternen dreidimensionalen Modells für die menschlichen Sinne notwendig sind (Bergbauer, 2002). Dabei

[‖] Unter der Definition rechnerinterner Modelle wird die Modellierung in Zeit und Raum, sowie die Festlegung von Randbedingungen verstanden (Encarnação and Felger, 1996)

[**] Die Aufbereitung umfasst die Erzeugung von wahrnehmbaren, im Wesentlichen seh-, hör- und fühlbaren Präsentationsformen (Encarnação and Felger, 1996; Gausemeier et al., 2001)

soll dem Benutzer durch Einbeziehung seiner Person in das Modell (Immersion) und infolge durch das Modell initiierter multimodaler Rückkopplungen (Encarnação and Felger, 1996) eine direkte Manipulation (Interaktion) ermöglicht werden.

Heim gibt einen ausführlichen Überblick über diese unterschiedlichen Merkmale von VR ab (Heim, 1993). Die maßgebenden sind die Immersion, die Interaktion sowie die Imagination. Bild 2.12 zeigt die Zusammenhänge auf.

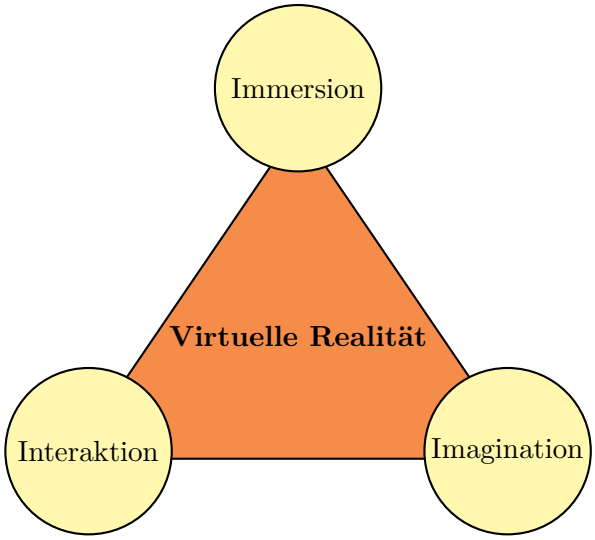


Bild 2.12: Die drei Merkmale von virtueller Realität (Burdea and Coiffet, 1994, p.6)

Immersion Den Begriff der Immersion definiert Henning als den Grad der psychophysischen Eingebundenheit in eine virtuelle Umgebung (Henning, 1997). Die Tiefe der Immersion wird als Immersionsgrad bezeichnet. Dieser hängt wiederum von der grafischen und akustischen Qualität der Simulation der virtuellen Umgebung ab. Neben dem optischen und dem akustischen Kanal können mit Hilfe geeigneter VR-Systemkomponenten zur Simulation von Form, Kraft, Geruch und sogar Geschmack weitere Empfindungskanäle des menschlichen Benutzers angesprochen und damit der Immersionsgrad weiter erhöht werden (Leinenbach, 2000, p.30). Kapitel 3.2.2 auf Seite 84 diskutiert die Anforderungen an den Immersionsgrad im Rahmen dieser Arbeit.

Eine qualitativ hochwertige Immersion ist immer eng gekoppelt mit der Visualisierung in Echtzeit. In Kapitel 3.3.1 auf Seite 86 wird dies konzeptbezogen näher betrachtet.

Interaktion Reinhold definiert Interaktion, als die Wechselwirkung, Wechselbeziehung, die gegenseitige Beeinflussung von Individuen oder Gruppen durch ihr aufeinander bezogenes Handeln (Reinhold, 1997, p.305). In der Informatik ist der Begriff der Interaktion mit dem Begriff der Kommunikation identisch. Er befasst sich damit, wie einzelne Komponenten eines Systems sich gegenseitig beeinflussen bzw. miteinander kommunizieren. Unter Interaktion im Rahmen der virtuellen Realität wird allerdings wesentlich mehr als Kommunikation verstanden. Jaron Lanier, wie bereits erwähnt, Schöpfer des Kunstwortes Virtual Reality (vgl. Kapitel 2.6.1 auf Seite 52), postuliert in einem Interview mit Tom Sperlich, dass

> *Interaktivität neben Kommunikation auch Überraschungen und neue Entdeckungen beinhalten muß.* (Sperlich, 1995)

Andy Lippman vom MIT Medien Labor definiert ebenda Interaktivität als

> *eine gemeinsame, gleichzeitige Aktivität seitens zweier Teilnehmer, die normalerweise, aber auf keinen Fall immer, auf etwas abzielt.* (Sperlich, 1995)

Beiden gemeinsam ist, dass Interaktivität im Bereich Virtual Reality als kontinuierlicher und nicht diskreter Vorgang verstanden wird.

Interaktion dient innerhalb einer virtuellen Umgebung zur Navigation und zur Interaktion mit den in dieser Umgebung befindlichen virtuellen Objekten. Sowohl Navigation als auch Interaktion müssen wiederum in Echtzeit möglich sein. Die Interaktionsmöglichkeiten innerhalb einer virtuellen Umgebung stehen in direktem Zusammenhang mit den eingesetzten VR Eingabegeräten (vgl. Kapitel 3.3.2 auf Seite 87).

Imagination Die dritte Säule bildet die Imagination. Burdea und Coiffet stellen fest, dass neben den angesprochenen technischen Voraussetzungen die menschliche Vorstellungskraft eine entscheidende Rolle spielt, wenn es darum geht, die reale Umgebung zu verlassen, um eine virtuelle Welt zu betreten und als neue Umgebung zu erleben (Burdea and Coiffet, 1994, p.4).

Brockhaus definiert Imagination im weiten, umgangssprachlichen Sinne als die Fähigkeit, sich einen Gegenstand, eine Person oder eine Szene anschaulich und bildhaft vorzustellen (Brockhaus, 2006). In den Wissenschaften ist Erkenntnis ohne Imagination oft unmöglich. Die Vorstellungskraft spielt eine bedeutende Rolle bei der Synthese von empirischen Beobachtungen und Befunden, die ohne Übersetzungsleistung und Interpretationsarbeit des Forschers keine Aussagekraft besitzen.

Zwischen den Begriffen der Virtuellen Realität und der Virtuellen Umgebung soll hier ebenfalls differenziert werden. Bei der Virtuellen Umgebung handelt es sich um eine künstliche Welt. Die Virtuelle Realität hingegen umfasst alle Elemente, die notwendig sind, um eine Virtuelle Umgebung zu generieren und zu betreiben. Somit ist die Virtuelle Umgebung nur ein Teil der Virtuellen Realität. Die zur Generierung und zum Betrieb einer Virtuellen Umgebung erforderlichen hard- und softwaretechnischen Bausteine bilden ein VR System (Astheimer et al., 1994). Dazu gehören alle Komponenten zur Modellierung der Objekte, zur Präsentation der virtuellen Welt sowie zur Navigations- und Interaktionssteuerung (Bergbauer, 2002).

Die wesentliche Abgrenzung zu herkömmlichen 3D Systemen liegt in den Merkmalen der virtuellen Realität begründet. Denn im Gegensatz zu Systemen, die durch computergestützte grafische Animation ebenfalls Reisen durch dreidimensionale virtuelle Welten ermöglichen, unterstützen VR Systeme die Navigation durch diese Welten immersiv und in Echtzeit und vermitteln dem Betrachter das Gefühl, in die virtuelle Umgebung einzutauchen (Leston et al., 1996, p.42-45). Darüberhinaus wird durch die Echtzeitinteraktion mit Objekten innerhalb der dreidimensionalen Darstellung, die direkte Kommunikation mit dem Benutzer unterstützt, was im Rahmen einer grafischen Animation nicht geleistet wird. Echtzeitnavigation sowie Echtzeitinteraktion und die daraus resultierenden Möglichkeiten zur Schaffung eines Immersionsgefühls beim Benutzer sind auch die wesentlichen Unterschiede zwischen VR-Sytemen und CAD und Multimediasystemen, auch wenn diese zunehmend über dreidimensionale grafische Benutzungsoberflächen verfügen.

2.6.2 Historische Entwicklung

Einen ausführlichen Überblick über die Entwicklungsgeschichte von Virtual Reality geben Kalawsky (Kalawsky, 1993, p.17-42), Vince (Vince, 1995, p.18-23), Bergbauer (Bergbauer, 2002) oder Hutchinson (Hutchison, 2007).

In Tabelle 2.5 werden die Meilensteine und Pioniere der Entwicklungsgeschichte von VR zusammengefasst. Es bleibt dabei festzustellen, dass sich die Bemühungen auf dem Gebiet der VR in der jüngeren Vergangenheit und in der Gegenwart auf die Leistungssteigerung und Kostenreduktion bereits existierender Systemkomponenten konzentrieren und damit der Verbreitung der Technolgie VR über Expertenkreise hinaus dienen (Leston et al., 1996, p.66ff).

Die Anfänge von VR werden gemeinhin mit dem Filmtechniker Morton Heilig verknüpft, der sich um die Entwicklung eines Erlebnistheaters bemühte, das eine Vielzahl von Besuchern mit sehr großem technischem Aufwand in eine virtuelle Welt versetzen sollte. Aus den Bemühungen um die erste virtuelle Multi-User-Umgebung wurde im Jahr 1956 schließlich die erste virtuelle Single-User-Welt in Form des Sensorama Simulators (Burdea and Coiffet, 1994, p.5-9). Obwohl Heilig bereits in seiner Simulationsmaschine 3D-Videotechnik einsetzte, gilt Ivan Sutherland als Begründer der Computergrafik im Allgemeinen und als Erfinder des HMDs im Speziellen. Im Mittelpunkt seiner Bemühungen um eine Verbesserung der Mensch-Maschine-Schnittstelle stand der Einsatz grafischer Systeme (Sutherland, 1963).

Die Forschung und Entwicklung der VR Technologie ist seitdem rasch vorangeschritten. Ursprünglich getragen durch militärische oder durch das Militär finanzierte Forschung mit dem Ziel der Entwicklung perfekter Simulations- und Trainingsbedingungen zu erschaffen, erhielt dieser Trend gerade in den USA eine deutliche Dynamik.

Das erste kommerzielle VR-Komplettsystem Reality Build For Two (RB2) der Firma VPL Research Inc. wurde ab 1989 verkauft (Blanchard et al., 1990). Insbesondere der von VPL entwickelte DataGlove gilt als Meilenstein in der Entwicklungsgeschichte von VR und erzielte eine relativ weite Verbreitung.

Die heutige VR-Technologie ist sehr breit aufgestellt, verfügt jedoch über keine Standardlösungen, die sich am Markt etabliert haben. So gibt es eine Reihe von wirtschaftlich erfolgreichen Unternehmen, die Hard- und Softwareprodukte für VR-Anwendungen anbieten. Die derzeit am Markt gängigen Systeme sind Lösungen wie Virtools von Dassault Systèmes, IDO:Software von IC:IDO, Deltagen von RTT, EON studio von EON Reality, VR Juggler von Infiscape oder das WorldToolKit von Sense8.

Daneben gibt es eine Vielzahl von VR-Systemen, welche heute immer noch in Forschungsinstituten oder einigen wenigen Entwicklungslabors großer In-

Jahr	**Meilenstein**	**Verantwortlicher**
1959	Sensorama Simulator	M. Heilig
1965	Head Mounted Display	I. Sutherland/ University of Utah
1967	GROPE-I	F. Brooks/ University of North Carolina
1975	Virtual Pushbuttons	K.C. Knowlton
	Large expanse extra perspective Optics	E. Howlett/ NASA
1978	Aspen Movie Map	A. Lippman/ MIT
1979	Magnetic Position tracking	F.H. Raab and E. Blood/ US Airforce
1981	Datenhandschuh	T. Zimmermann/ VPL
	Virtual Cockpit	T. Furness/ US Airforce
1983	Video Place	M. Krueger/ University of Wisconsin
1984	Neuromancer	W. Gibson
	Self-tracking	T.G. Bishop
1985	Data Glove	J. Lanier/ VPL
1988	Binocular Omni-Orientation Monitor	Fakespace
1989	Reality build for two	J. Lanier/ VPL
1992	CAVE	Carolina Cruz-Neira, Dan Sandin and Tom DeFanti / University of Illinois
1995	VRML	Silicon Graphics
1999	The Matrix	Wacholski Brothers
2006	C6	IOWA State University

Tabelle 2.5: Entwicklungsgeschichte VR

dustrieunternehmen entstehen. Sie sind Prototypen und aufgabenspezifische Einzellösungen, die aus mehr oder weniger ausgereiften Technologiebausteinen bestehen. Neben der Erschließung industrieller Einsatzfelder werden die in dieser Technik vorhandenen Optionen vor allem von Seiten der Unterhaltungsindustrie weiterverfolgt. Dies erscheint durchaus verständlich, da dort langfristig das größere Wachstums- und Marktpotential zu erwarten ist. (Bergbauer, 2002)

2.6.3 Aufbau von VR Systemen

Biocca formuliert als allgemeine Zielstellung bei Entwurf und Implementierung eines VR-Systems die möglichst optimale Verbindung der menschlichen Sinnesorgane mit den Ausgabegeräten (Biocca, 1997). Gleiches muß natürlich auch für die Verbindung der menschlichen (Re-)Aktionen mit den Eingabegeräten des VR-Systems gelten (Leinenbach, 2000, p.35).

Dabei spielen die technischen Anforderungen an die Leistungsfähigkeit eines VR-Systems eine entscheidende Rolle. Durlach beschrieb im Jahr 1995 eine minimale Bildgenerierungsrate von 10 Bildern pro Sekunde und eine maximale Verzögerungszeit zwischen Benutzereingabe und Systemreaktion von 0,1 Sekunden als Minimalanforderung. Ausgehend von der Annahme, dass ein photorealistisches Bild einer realen Umgebung aus ungefähr 80 Millionen Polygonen besteht, folgert er, dass dementsprechend ein leistungsfähiges VR-System 800 Millionen Polygone pro Sekunde berechnen können muss, um zu einer Bildgenerierungsrate von 10 Bildern pro Sekunde zu gelangen (Durlach and Mavor, 1995, p.250-252). Gegenwärtige Verfahren wie das Echtzeit Raytracing von Slusallek sind in der Lage bis zu 2 Milliarden Polygone dazustellen (Schmittler et al., 2004). Dennoch bewegen sich die Computer im Hochleistungsbereich weiterhin im zwei bis dreistelligen Millionenbereich.

In den folgenden Abschnitten werden zunächst unterschiedliche Ein- und Ausgabegeräte von VR vorgestellt. Dabei werden die im Rahmen dieser Arbeit genutzten Geräte hervorgehoben. Daraufhin werden die zentralen Hard- und Softwarekomponenten eines VR-Systems betrachtet.

Eingabegeräte

Die Eingabegeräte eines VR-Systems dienen der Navigation durch die virtuelle Umgebung und der Interaktion mit den darin befindlichen virtuellen

Objekten. Sie werden in der Regel durch Körperbewegungen oder Sprache des menschlichen Benutzers angesprochen. Eine zentrale Rolle spielt das Tracking System, dass die jeweilige Position des Benutzers im Raum ermittelt. Darüber hinaus sind Datenhandschuh, Spracheingabe, Spacedevices, Wands und Joysticks weitere Eingabemittel.

Tracking Anstatt die ganzheitlichen Körperbewegungen des Benutzers aufzuzeichnen, werten Tracking Systeme die Position und Orientierung einzelner Körperteile im Zeitverlauf aus, um daraus den Bewegungsablauf innerhalb der virtuellen Welt ableiten zu können. Am häufigsten werden Position und Orientierung des Kopfes und einer Hand des Benutzers ausgewertet. Zu den wichtigsten technischen Verfahren des Position Tracking zählen elektromagnetische, mechanische, akustische und optische Verfahren, die in Tabelle 2.6 mit ihren Vor- und Nachteilen dargestellt sind.

System-prinzip	**Vorteile**	**Nachteile**
Mechanisch	Hohe Genauigkeit	Geringe Bewegungsfreiheit
	Geringe Latenzzeit	Kleiner Aktionsraum
Elektromagnetisch	Große Bewegungsfreiheit	Störanfälligkeit durch magnetische Felder
	Hohe Genauigkeit	Hohe Latenzzeit
	Kleiner Empfänger	
Akustisch	Große Bewegungsfreiheit	Störanfälligkeit durch akustische Interferenzen
	Kleiner Empfänger	Kleiner Aktionsradius
Optisch	Großer Aktionsradius bei aktiven Verfahren	Sichtverbindung zur Kamera
	Große Bewegungsfreiheit	Hohe Verzögerungszeiten

Tabelle 2.6: Vergleich der Systemprinzipien beim Tracking nach Ovtcharova (Ovtcharova, 2006a)

Die entsprechenden Systeme sind im Fall des Kopf- und Handtracking in HMDs und Datenhandschuhe größtenteils bereits integriert. Tracking der Augen wird mittels der Messung der Bewegungen der Augen ermittelt,

um daraus die grafische Repräsentation der virtuellen Umgebung für diejenige Blickrichtung zu berechnen, die der gemessenen Augenposition entspricht. (Leinenbach, 2000)

Datenhandschuh Neben dem Tracking zählen Datenhandschuh und Datenanzug zu den wichtigsten Eingabegeräten eines immersiven VR Systems. Datenhandschuhe messen die Krümmung der Finger des Trägers und können dadurch Gesten zur Fortbewegung innerhalb der virtuellen Umgebung oder zur Interaktion mit virtuellen Objekten erkennen. Datenanzüge ermöglichen dies für den ganzen Körper. Beide Systeme integrieren Trackingkomponenten, um neben Bewegungsinformationen auch Daten über Position und Orientierung der Hand oder anderer Körperteile des Benutzers an den zentralen Steuerrechner (vgl. Kapitel 2.6.3) des VR-Systems übermitteln zu können. Einen Überblick über die wichtigsten Systeme geben (Burdea and Coiffet, 1994, p.32-45), (Bormann, 1994, p.51-57) und (Aukstakalnis and Blatner, 1992, 159-168).

Spracheingabe Ein weiteres, intuitives Mittel der Eingabe besteht in der Weiterentwicklung existierender Spracherkennungssysteme. Zu den wichtigsten Schwierigkeiten auf diesem Entwicklungsweg zählen neben den Erkennen ganzer Sätze statt einzelner Wörter, eine akzeptable Mindestgröße des verfügbaren Vokabulars sowie die Unabhängigkeit des Systems vom Sprecher (Biocca, 1997, p.56ff).

Wiimote Die Wiimote ist ein Eingabegerät der Firma Nintendo. Entwickelt wurde das Gerät als Controller für die Spielkonsole Wii. Über Bluetooth und einem zusätzlichen Gerätetreiber kann die Wiimote aber auch mit einen handelsüblichen Computer bedient werden. Die Wiimote verfügt über drei Bewegungssensoren, die in entsprechenden Richtungen die Beschleunigung messen (siehe Bild 2.13). Durch geschickte Nutzung der Werte dieser Sensoren, ist es möglich, die erzeugten Bewegungen zu interpretieren. Darüber hinaus können mittels der Schwerkraftsensoren weitere Orientierungswerte errechnet werden. Diese können auch bei langsamer Bewegung und im Stillstand ausgelesen werden. Genauso ist es möglich die Orientierung des Gerätes innerhalb bestimmter Grenzen zu bestimmen.

Weitere Eingabemöglichkeiten bieten ein Steuerkreuz und sieben Tasten. Über eine Infrarot Leuchtdiode kann, mit Hilfe eines Zusatzgerätes, ein

Zeigen auf den Bildschirm mit dem Gerät aufgenommen und verarbeitet werden. Ein Nachteil der Wiimote ist, dass die translatorischen Beschleunigungssensoren bei langsamen Bewegungen nicht sensibel genug reagieren. Es werden nur sehr schnelle Bewegungen, wie sie z.B. beim zügigen Schwingen der Hand mit dem Gerät auftreten, ausgewertet. Positionen können dadurch nicht bestimmt werden. (Brenzinger, 2007, p.7)

Bild 2.13: Wiimote Controller

Weitere Eingabegeräte Neben den Tracking, Datenhandschuh und Wiimote stehen insbesondere in den Desktop-VR oder Fishtanksystemen, weitere Eingabegeräte zur Verfügung. Prinzipiell besteht bei diesen die Möglichkeit der Dateneingabe über Tastatur, auch wenn dadurch die Immersion zerstört wird. Darüber hinaus können Kugeln, Wands, Joysticks oder herkömmliche PC-Mäuse, die mit Position Tracking-Komponenten aufgerüstet werden, eingesetzt werden. Daneben reagieren die Spacedevices auf Druck-, Zug- und Drehbewegungen und ermöglichen so eine sehr präzise Navigation durch virtuelle Räume.

Ausgabegeräte

In VR Umgebungen wird versucht die Sinne des Menschen so anzusprechen, dass ein möglichst hoher Grad der Immersion entsteht. Dabei spielen visuelle Reize eine zentraler Bedeutung. Das Sehvermögen ist ohne Zweifel der wichtigste menschliche Sinn. Durch die Augen empfängt das Zentralnervensystem ungefähr 75% aller wahrgenommen Informationen, während lediglich

13% mit dem Gehör und die restlichen 12% mit Hilfe der anderen Sinnesorgane aufgenommen werden (Scherer, 2007, p.17). Zur Beeinflussung der Immersion gibt Henning einige Faktoren an, die den Grad der Immersion fördern oder hemmen können (vgl. Kapitel 3.1 auf Seite 87), allerdings ist ein unverzichtbares Merkmal immersiver Konzepte, dass der Betrachter das Gefühl haben muss, die Szenerie aus seiner Sicht zu erleben (Brenzinger, 2007, p.8). Die Vielzahl der verfügbaren Ausgabegeräte eines VR-Systems lassen sich dementsprechend in fünf Kategorien visuelle, auditive, haptische, olfaktorische und gustatorische Systeme einordnen.

Visuelle Systeme Einen ausführlichen Überblick über visuelle Ausgabegeräte geben Bormann (Bormann, 1994, p.73-84), Mania/Ellis/Steed und Billinghurst (Mania et al., 2002) sowie Falter/Rötting und Springer (Falter et al., 1993). In immersiven Systemen kommen in der Regel HMDs zum Einsatz. Diese stellen pro Auge einen Bildschirm und eine spezielle Weitwinkeloptik zur Verfügung. Alternativ dazu wurde der BOOM entwickelt. Dieser besteht im Wesentlichen aus einem HMD vergleichbaren Sichtsystem, das sehr leistungsfähig und daher in der Regel sehr schwer ist. Deshalb wird es an einer mechanischen Vorrichtung aufgehängt, was eine freie Bewegung unmöglich macht.

Die Visualisierung muss nicht direkt vor dem Auge stattfinden. VRDs speisen mittels Laserstrahl die Informationen direkt auf die Netzhaut und ermöglichen so ein völlig hilfsmittelfreies, intuitives Sehen (Doniec, 2003). Weit verbreitet ist die entfernte Darstellung vom Auge mittels eines Computermonitors oder großflächiger Videowände. Dies wird entweder durch verschlussgesteuerter Shutter-Brillen erreicht, die abwechselnd das, für das linke und rechte Auge dargestellte, Computerbild verschließen, oder mittels Polarisationsbrillen, bei denen das übereinandergelegte Bild durch Polfilter getrennt wird. In einer CAVE bilden vier bis sechs zu einem Würfel angeordnete Videowände, einen realen Raum, der das Erlebnis einer virtuellen Umgebung ermöglicht. Zukünftig versprechen holographische Systeme, Heliodisplays und 3D Monitore eine größere interaktive, natürliche Visualisierung (Bimber and Raskar, 2006).

Auditive Systeme Ziel auditiver Systeme ist die Schaffung einer 3D-Sound-Umgebung, in der insbesondere Töne, die mit virtuellen Objekten verknüpft sind, räumlich wahrgenommen werden können und ihren Ursprung auch bei Bewegung durch den virtuellen Raum beibehalten. Die meisten kommerziell

verfügbaren Systeme basieren auf Kopfhörern, die häufig in HMDs integriert sind, oder im Falle räumlicher Darstellung, auf einer Menge im realen Raum angeordneter Lautsprecher (Leinenbach, 2000).

Haptische Systeme Haptische Systeme geben taktiles Feedback an den Benutzer der VR Umgebung aus. Dadurch soll das Gefühl vermittelt werden, einen Gegenstand der virtuellen Welt tatsächlich zu berühren und seine Oberflächeneigenschaften sowie die von ihm ausgehenden Kräfte zu spüren. Taktile Systeme sprechen die in der menschlichen Haut befindlichen Rezeptoren an, die Reize wie konstanter Druck, Druckveränderung, Vibration oder Temperatur wahrnehmen. Kraftrückkopplungssysteme dagegen sprechen die in den Muskeln und Gelenken liegenden kinästhetischen Sensoren an und dienen zur Ermittlung der Lage, Bewegung und Drehung einzelner Körperpartien und der auf sie einwirkenden Kräfte. (Leinenbach, 2000)

Olfaktorisch und Gustatorische Systeme Geruchs- und Geschmackssinn benötigen Ausgabegeräte, welche die chemischen Sinne des Menschen reizen. Während über die Realisierbarkeit und Sinnhaftigkeit olfaktorischer Systeme zur Stimulation des Geruchsinns diskutiert werden kann, erscheint die Realisierbarkeit gustatorischer Systeme zum Ansprechen des Geschmacksinns als fragwürdig.

VR Hardware

Zentraler Bestandteil eines VR Systems ist das Rechnersystem, welches die virtuelle Umgebung in Echtzeit erzeugt. Es besteht aus drei Hauptkomponenten, den Grafikprozessoren, den Renderern und einem Netzwerk/ Bussystem, das den Datenaustausch zwischen den einzelnen Komponenten übernimmt (vgl. Bild 4.10 auf Seite 104) (Leinenbach, 2000). Die zentrale Aufgabe des Rechnersystems ist die Verarbeitung der über die angeschlossenen Eingabegeräte eingehenden Daten, die Steuerung der Ausgabegeräte in Abhängigkeit der Eingabedaten sowie das Ablegen der Modelldaten der virtuellen Umgebung (Bormann, 1994, p.93). Dabei sollte die Hardware in der Lage sein, für eine Stereoprojektion wenigstens zwei getrennte Rasterbilder gleichzeitig zu generieren. Da eine wesentliche Anforderung an das Rendering von VR Systemen in der Echtzeitfähigkeit liegt, muss hier ein Rasterbild mindestens in einer 1/8 Sekunde geliefert werden, damit eine Animation noch einen flüssigen Eindruck vermittelt. Bei High-End VR

wird der Anspruch auf wenigstens 1/25 Sekunde pro Rasterbild gehoben. Die Regeln für Echtzeitrendering gelten für alle VR Systeme, die auf Echtzeit-Visualisierungssoftware wie Direct3D, Java3D, OpenGL oder seine Ableger Inventor, Iris Performer etc. basieren. (Scheurich, 2007)

Waren früher zur Berechnung der anspruchsvollen Grafik noch sehr teure spezielle Workstations nötig, werden heute Standard PC Systeme der höchsten Leistungsklasse verwendet. Diese Systeme haben die traditionellen Workstations zunehmend verdrängt, da ihre Leistungsfähigkeit in den letzten Jahren stark gestiegen ist. PC Systeme können darüber hinaus zur weiteren Leistungssteigerung zu so genannten Clustern verbunden werden. Im VR Bereich sind Cluster üblich, die Rechenoperationen zur Erzeugung der VR Szene und der Interaktion von Rechenoperationen zum Rendern von Bildern trennen. Besonders im Bereich großer Displays wird jedem Projektor ein PC zugeordnet, der das entsprechende Bild für diesen Projektor rendert. (Brenzinger, 2007, p.11)

VR Software

Um eine VR Szene zu erstellen, sind verschiedene Softwarewerkzeuge notwendig. Leston differenziert dabei zwischen Softwaresystemen zur Gestaltung der virtuellen Umgebung, zwischen unterschiedlichen Run-Time Umgebungen sowie zwischen integrierten VR Entwicklungsumgebungen (Leston et al., 1996, p.10f).

Unter der Gestaltung der virtuellen Umgebung wird die Modellierung der im virtuellen Raum befindlichen dreidimensionalen Objekte verstanden. Hierzu gehören neben der Festlegung der Geometrie der Objekte, die entweder aus Objektbibliotheken stammen oder mit Hilfe von 3D Modellierungssystemen generiert werden, auch die Bestimmung der Oberflächengestalt sowie die Definition weiterer statischer und dynamischer Objekteigenschaften. Von besonderer Bedeutung für die Performance, d.h. für eine fließende Echtzeitinteraktion innerhalb der virtuellen Umgebungen und damit auch für den erreichten Immersionsgrad, sind eine sinnvolle Segmentierung der virtuellen Welt und der Einsatz von LoD Verfahren. Im Rahmen dieser Arbeit wird dies in Kapitel 4.1.2 auf Seite 103 genauer diskutiert. Neben dem Einsatz von 3D Modellierungssystemen erlauben auch Grafikbibliotheken wie die Direct3D von Microsoft, Java3D von Sun oder OpenGL von SGI die manuelle Programmierung dreidimensionaler Objekte und deren Darstellung mittels Szenengraphen. Diese sind speziell zur Darstellung von Graphik in

Echtzeit gedacht. Szenengraphen werden dabei hierarchisch als Baumstruktur dargestellt, so dass jeder Knoten genau einen Elternknoten besitzt und dem beliebig viele Kinderknoten zugeordnet werden können. Es ist damit klar, dass jeder Kindknoten auch wieder einen eigenen Unterbaum bilden kann, wodurch ein hierarchischer Aufbau vorliegt. Bild 2.14 beschreibt den Aufbau eines Szenengraphen.

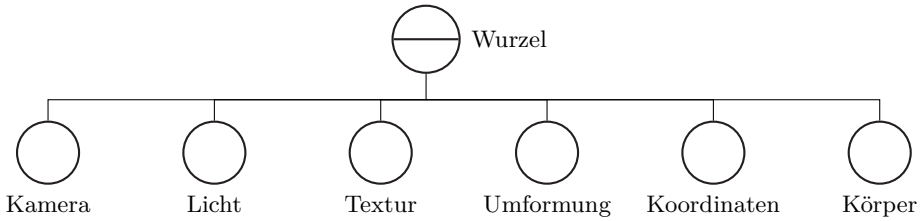


Bild 2.14: Ein einfacher Szenengraph

Zusätzlich verfügt jeder Knoten des Szenengraphen über eine eigene Transformationsmatrix. Sollte diese Matrix manipuliert werden, wird nicht nur der dazugehörige Knoten transformiert, sondern auch alle untergeordneten Knoten. Kapitel 4.1.3 auf Seite 109 beschreibt die Verwendung der Szenengraphen im Rahmen dieser Arbeit. Daneben ermöglichen aber auch existierende VR Systeme die Erzeugung neuer virtueller Objekte, ohne dass es einer Programmierung bedarf (Deering, 1996).

Die Run-Time (Laufzeit) Umgebung stellt den Kern des Gesamtsystems dar. Sie steuert die Simulation innerhalb der virtuellen Umgebung in Abhängigkeit von den zur Laufzeit durchgeführten Interaktionen des Benutzers mit den virtuellen Objekten. Die Steuerung der Simulation bedeutet hierbei insbesondere die Berechnung der Einzelgrafiken in Abhängigkeit vom aktuellen Standpunkt und Blickwinkel des Betrachters in Echtzeit und das Übermitteln dieser Bilder an die entsprechenden Ausgabegeräte (Leston et al., 1996, p.101). Wichtige Verfahren hierzu werden im Rahmen der Renderpipeline eingesetzt (vgl. Kapitel 4.1.2 auf Seite 103). Dazu gehören die Shading Verfahren, wie Flat, Gouraud oder Phong, sowie raytracing oder radiosity Verfahren.

Bei den integrierten VR Entwicklungsumgebungen muss zwischen grafischen Entwicklungsumgebungen und solchen Systemen unterschieden werden, die im Wesentlichen auf der Bereitstellung von Funktions- oder Klassenbibliotheken zur Verwendung innerhalb herkömmlicher Software Entwicklungsumgebungen basieren. Die primäre Aufgabe der Entwicklungsumgebung

liegt darin, eine Programmierschnittstelle zur Abstraktion der Hardware anzubieten. Hauptsächliche Kriterien für solche Systeme sind Performanz und Flexibilität, wobei diese oftmals in einem direkten Konflikt miteinander stehen (Richter, 2005, p.46). Die wichtigsten Hersteller bieten neben der Programmierschnittstelle auch grundlegende grafische Funktionalität zur Realisierung einfacher VR Anwendungen an. Einen ausführlichen Überblick über integrierte VR Entwicklungsumgebungen geben (Richter, 2005, p.46ff), (Bauer, 1996, p.34f) sowie (Leston et al., 1996, p.193-293). In Kapitel 5 auf Seite 177 werden die im Kontext dieser Arbeit betrachteten VR Softwarepakete, tabellarisch zusammengefasst.

2.6.4 Interaktionsmetaphern

Metaphern werden im Allgemeinen verwendet, um abstrakte Teilaspekte in einem komplexen Zusammenhang zu erklären und so das Verständnis zu fördern. Interaktionsmetaphern stellen eine Interaktionsform dar, die komplexe Zusammenhänge auf einfache, leicht Verständlichere überträgt. Sie werden im Rahmen der von VR Entwicklungsumgebungen, wie sie im Kapitel 2.6.3 auf Seite 65 vorgestellt wurden, angewendet. Dabei beschreibt die Interaktionsmetapher eine thematisch verbundene Gruppe von Elementen mit symbolischen Bedeutungen oder Repräsentationen mit Assoziationen zu anderen Kontexten oder unterstelltem Fachwissen. Es kann sich hierbei um visuelle Motive, geläufige ikonische Darstellungen oder parallele Situationen aus der physischen Realität handeln (Beilharz and Reffat, 2004, p.371). Dabei soll die Interaktion möglichst natürlich und selbsterklärend erfolgen. Dies geschieht, indem semantische Verbindungen zwischen den Funktionen der Software und dem bestehenden Wissen des Benutzers hervorgerufen werden (Heckel, 1991, p.498f).

Interaktionsmetaphern können in exozentrisch und egozentrisch unterschieden werden. Hat der Benutzer die gesamte virtuelle Umgebung im Blickfeld und kann er diese von mehreren Seiten betrachten, befindet er sich außerhalb dieser Welt und es kommen exozentrische Metaphern zum Einsatz. Von egozentrischen Metaphern wird dagegen gesprochen, wenn der Anwender sich während der Benutzung innerhalb der Welt befindet und die Dinge um sich herum betrachtet und verändert (Schwall, 2004, p.8). Im Folgenden sollen die im Rahmen dieser Arbeit betrachteten Interaktionsmetaphern kurz diskutiert werden.

World-in-Miniature

Die wohl bekannteste exozentrische Metapher ist die so genannte WIM Metapher (Stoakley et al., 1995, p.265-272). Dabei arbeitet der Benutzer mit einer verkleinerten Darstellung der virtuellen Umgebung, die ikonische Repräsentationen aller Objekte enthält und vor allem den Vorteil bietet, viel einfacher manipulierbar zu sein als eine real skalierte Welt. Alle Objekte in dieser Miniaturwelt können in sechs Freiheitsgraden auf jede beliebige Distanz verändert werden.

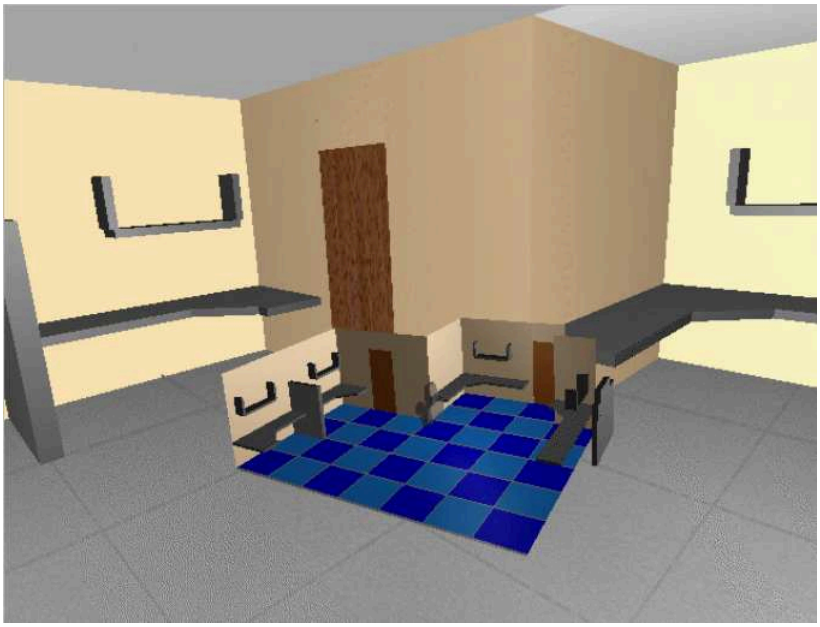


Bild 2.15: World in Miniature nach (Stoakley et al., 1995)

Scaled-world Grab

Die scaled-world Grab Metapher wird in erster Linie zur Skalierung der virtuellen Umgebung des Benutzers eingesetzt. Darüber hinaus besteht die Möglichkeit, das der Benutzer selbst skaliert wird. Ziel ist es, mittels einer egozentrischen Interaktionsmetapher, wie z.B. der virtuellen Hand, das ausgewählte Objekt berühren zu können. Das Skalieren hat den Effekt, dass der Benutzer, solange er sich nicht bewegt, die Skalierung nicht wahrnimmt. Dreht er sich jedoch oder bewegt sich von seiner bisherigen Position weg,

stellt er fest, dass er zum Riesen oder zum Zwerg geworden ist. (Schwall, 2004, p.9)

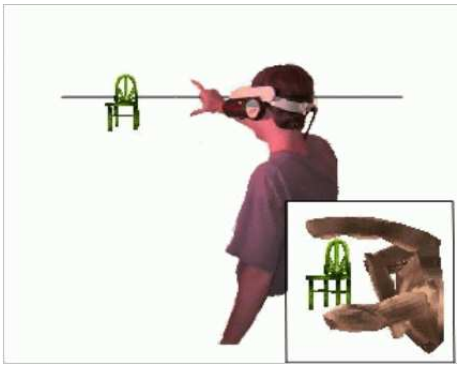


Bild 2.16: Scaled-world Grab nach (Pierce et al., 1997)

Virtuelle Hand

Die Selektion von Objekten in virtuellen Umgebungen ist eine der grundlegenden Techniken in nahezu jeder Applikation. Die Metapher der virtuellen Hand ist die am weitesten verbreitete egozentrische Manipulationstechnik. Eine virtuelle Repräsentation der Hand des Benutzers wird in der virtuellen Umgebung dargestellt. Durch bewegen einer getrackten Hand oder eines anderen Eingabegeräts wird die virtuelle Hand entsprechend mitbewegt. Die Auswahl der 3D Objekte erfolgt durch die Kollision. Der große Vorteil dieser Metapher besteht in ihrer Einfachheit und Intuitivität, da sie der Berührung eines Objektes in der realen Welt sehr ähnlich ist. Demgegenüber stehen jedoch zwei bedeutende Nachteile. Die Tätigkeit kann, abhängig vom Eingabegerät, sehr ermüdend sein. Des Weiteren ist die Reichweite stark limitiert. Konzepte wie die Go-Go Metapher (vgl. Kapitel 2.6.4 auf Seite 71) lösen das Problem auf Kosten von abnehmender Genauigkeit. Alternativ kann der Selektion eine Navigation durch die virtuelle Umgebung vorausgehen, was jedoch nicht für jede Anwendung wünschenswert ist. (Boeck et al., 2006, p.54)

Ray Casting

Ray Casting (dt. Strahl aussenden) ist eine weitere egozentrische Metapher zur Selektierung von weit entfernten Objekten innerhalb einer virtuellen

Umgebung. Imitiert wird ein Laser Pointer oder der Lichtkegel einer Taschenlampe. Der Strahl geht von der Hand des Benutzers oder von einem virtuellen Zeiger aus, wie Bild 2.17 darstellt. Auf diese Weise können alle unverdeckten Objekte einer virtuellen Szene ausgewählt werden. Objekte, die mit der Virtuellen Hand nicht erreichbar sind, werden mittels Ray Casting selektierbar. (Mine, 1995, p.4) (Boeck et al., 2006, p.55)

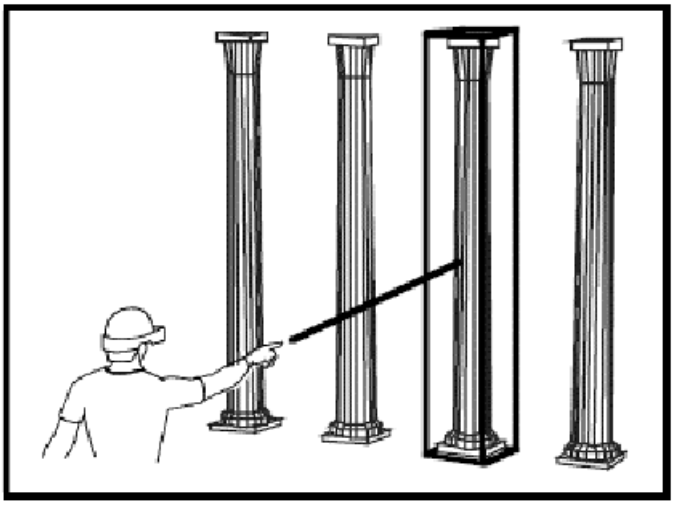


Bild 2.17: Selektion mittels Ray Casting nach (Mine, 1995, p.5)

Aperture Based Selection

Diese egozentrische Metapher nutzt einen Kreis, der parallel zur Projektionsebene in der virtuellen Umgebung schwebt, in Kombination mit einem Kegel, der vom Augpunkt des Betrachters ausgeht und dabei durch eine Blende (engl. Aperture) verläuft. Wird die Blende in X- oder Y-Richtung bewegt, kann auf Objekte gezielt werden, um diese zu selektieren. Da der Durchmesser des Kreises konstant bleibt, bewirkt eine Translation in Z-Richtung, dass der Kegel enger oder weiter wird (Forsberg et al., 1996, p.95f). Die eigentliche Selektion funktioniert analog zur Ray Casting Metapher. Da das Auswählen jedoch nicht mehr auf Rotationen sondern auf Translationen beruht, wird die Kontrolle signifikant erleichtert und der Benutzer hat einen Geschwindigkeitsvorteil bei der Ausführung von Aufgaben. Dabei werden die Vorteile der Virtuellen Hand und des Ray Casting miteinander vereint, ohne weitere Nachteile zu erzeugen (Boeck et al., 2006, p.55).

Go-Go Metapher

Die Go-Go Metapher verlängert den virtuellen Arm des Benutzers. Dadurch können auch weiter entfernte Objekte selektieren und manipuliert werden (Schwall, 2004, p.11f.). Die von Poupyrev entwickelte Technik beruht auf der nichtlinearen Übertragung der getrackten Bewegung der realen Hand des Benutzers auf die bewirkte Bewegung der virtuellen Hand. Durch die Verwendung zweier Trackingsensoren kann die genau Position der menschlichen Hand in Polarkoordinaten (R_r, φ, θ) in einem benutzerzentrierten Koordinatensystem ermittelt werden. R_r stellt den Betrag des Vektors vom Koordinatenursprung in der Brust des Benutzers zu dessen Hand dar, also die Länge seines Armes. Die virtuelle Hand wird an der Stelle (R_v, φ, θ) innerhalb der virtuellen Szene dargestellt, wobei R_v die Länge des virtuellen Armes ist, die nach der Übertragungsfunktion F

$$F(R_r) = R_v = \begin{cases} R_r & \text{für } R_r < D \\ R_r + k(R_r - D)^2 & \text{für } R_r > D \end{cases}$$

bestimmt wird. Hierbei ist $k > 0 \wedge k < 1$ ein konstanter Wert, und bezeichnet den Dehnungskoeffizienten des Arms. Die Übertragungsfunktion F teilt den Raum um den Benutzer in zwei Teile. Solange der Benutzer mit nahen Objekten für $R_r < D$ interagiert, ist die Übertragung linear. Für $R_r > D$ steigt die übertragende Entfernung quadratisch an. Der Wert für D wurde auf 2/3 der Armlänge des Benutzers festgelegt, wobei die Standardarmlänge eines mittelgroßen Mannes nach DIN-33402 740 mm beträgt (vgl. Anhang B auf Seite 283). Bild 2.18 zeigt den Verlauf der Go-Go Übertragungsfunktion. (Scherer, 2007, p.46)

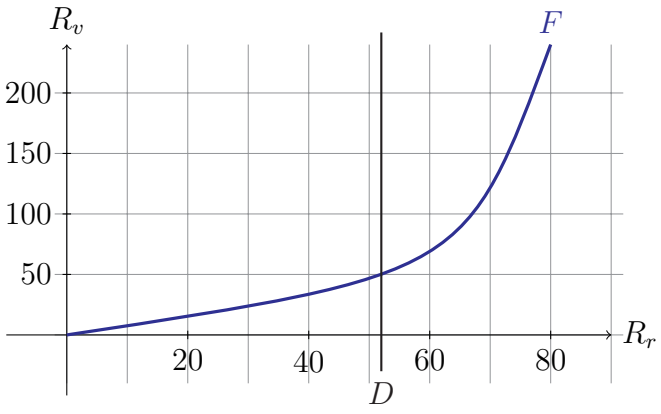


Bild 2.18: Go-Go Übertragungsfunktion nach (Poupyrev et al., 1996, p.79f)

Virtual X-Ray

Die Virtual X-Ray Metapher erlaubt es, Objekte dynamisch, halbtransparent darzustellen, und so Zugang zu verdeckt liegenden Objekten zu erhalten (vgl. Bild 2.19).

(a) X-Ray Metapher solid (b) X-Ray Metapher transparent

Bild 2.19: Virtual X-Ray Metapher (Elmqvist et al., 2007)

Sowohl die zeitliche Performance als auch die eindeutige Verwendung von Interaktionstechniken, wie Entdeckung, Selektion und Manipulation, wird signifikant verbessert (Elmqvist et al., 2007). Nachteile der Metapher sind die erhöhte visuelle Komplexität und die leicht verminderte Tiefenwirkung der virtuellen Umgebung, wodurch die Immersion beeinträchtigt wird. Durch gezielte Auswahl der 3D Objekte kontrolliert der Benutzer, in welchen Teilen der virtuellen Welt die dynamische Transparenz aktiv sein soll. Um den Effekt der verminderten Tiefenwirkung so gering wie möglich zu halten, werden nicht komplette Bauteile ausgeblendet, sondern lediglich diejenigen Oberflächen halbtransparent dargestellt, die auch wirklich Zielobjekte verdecken. Alle anderen Merkmale der stereoskopischen Tiefenwahrnehmung (Stereopsis, Bewegungsparalaxe, relative Größe, atmosphärische Perspektive und Texturgradienten) werden von der Metapher nicht beeinflusst und leisten weiterhin ihren Beitrag zur Entstehung einer räumlichen Darstellung (Scherer, 2007, p.55f).

Virtual Tricorder

Der virtuelle Tricorder ist das virtuelle Duplikat eines realen Eingabegeräts. Dabei wird das reale Gerät getrackt. Durch Berührung kann der Benutzer alle sechs Freiheitsgrade steuern und erhält jeweils taktiles Feedback. Er sieht jedoch nur den virtuellen Tricorder (Wloka and Greenfield, 1995, p.39f).

Auf diese Weise identifiziert er die taktilen Informationen, die er vom realen Gerät erhält mit dem Virtuellen, so dass die Immersion verstärkt wird.

Grundlage ist die Werkzeugmetapher, die, anders als beispielsweise der Datenhandschuh, darauf beruht, dass der Mensch, evolutionsgeschichtlich betrachtet, schon immer mit Werkzeugen arbeitet, also gar keine direkten Mensch-Objekt Schnittstellen zur totalen Immersion benötigt werden (Scherer, 2007, p.48). Eine indirekte Interaktionsmetapher bringt unnatürliche Interaktionstechniken wie skalieren, fliegen oder selektieren den menschlichen Gewohnheiten näher. Durch bewusste Verwendung von 2D Menüs soll die Verständlichkeit und Lesbarkeit unterstützt werden. Die Verwendung eines einzigen Multifunktionswerkzeugs ermöglicht eine einheitliche Umgebung für eine Vielzahl von Interaktionstechniken.

Pen & Tablet

Metaphern zur Systemkontrolle sind bei komplexen virtuellen Umgebungen bedeutend, um Aufgaben wie Systemmodus einstellen, Befehle auswählen oder Lautstärke regeln einfach auszuführen. Die egozentrische Kreide & Tafel Metapher wurde am MIT zur zweidimensionalen Ein- und Ausgabe für virtuelle Umgebungen entwickelt (Bowman and Hodges, 1997, p.13ff). Durch die Verbindung der physischen Geräte mit ihrer virtuellen Repräsentation entsteht eine 2D Schnittstelle, die der Benutzer mit sich durch die virtuelle Welt tragen kann. Analog zur Werkzeugmetapher, werden auch hier die Vorteile dieser ausgenutzt. Mittels der 2D Benutzerschnittstelle werden Objekte, wie Menüs, Schaltflächen und Schieberegler auf der Oberfläche des virtuellen Tablets dargestellt. Dabei ist die ständige Verfügbarkeit, neben dem taktilen Feedback der Stiftbewegung einer der wesentlichen Vorzüge der Metapher.

Sprach- und Gestensteuerung

Thórisson (Thórisson et al., 1992, p.139f.) entwickelte eine weitere egozentrische Interaktionsmetapher, die Spracheingabe mit Gestensteuerung kombiniert. Da die Sprache nicht an eine räumliche Metapher gebunden ist, eignet sie sich besonders für beschreibende Tätigkeiten, also zur Interaktion mit Objekten, unabhängig vom Grad deren visueller Exposition. Gesten hingegen werden zur direkten Objektmanipulation eingesetzt (Billinghurst,

1998, p.60ff). Durch die Kombination von Sprache und Gesten wird die Erkennungsgenauigkeit verbessert und die Länge der Spracheingabe verkürzt sich signifikant. Beides führt zu einem schnelleren Abschluss von Tätigkeiten und einer Reduzierung von Eingabefehlern (Scherer, 2007, p.50f).

Auditive Kommentare

Die Darstellung von Texten ist in virtuellen Umgebungen, im Gegensatz zu zweidimensionalen Umgebungen, ungeeignet. Dementsprechend schlagen Bowman und Hodges Audiokommentare als weitere Modalität vor, um abstrakte Informationen in virtuellen Umgebungen zu präsentieren (Bowman and Hodges, 1997, p.16f). Vorzüge liegen vor allem in den kognitiv unterschiedlichen Tätigkeiten des Gehirns, das dadurch ein paralleles Verarbeiten von visuellen als auch auditiven Informationen ermöglicht. Dabei kann durch unterschiedliche Betonung oder Variation der Sprachgeschwindigkeit auf die Informationsaufnahme Einfluss genommen werden. Ein wesentlicher Nachteil liegt in der zeitlichen Linearität von Audiokommentaren.

2.6.5 Anwendungsgebiete

Die dreidimensionale und intuitive Visualisierung von Informationen ist der wichtigste Einsatzschwerpunkt von VR Technologien und steht im Mittelpunkt fast jedes Anwendungsgebiets. Tabelle 2.7 vermittelt einen Überblick über die zentralen Anwendungsgebiete von VR. Die darzustellenden Informationen können sowohl konkrete als auch abstrakte Sachverhalte betreffen. Dreidimensionale und interaktiv begehbare Gebäudevisualisierungen sind ein Beispiel für die Darstellung konkreter Sachverhalte. Die intuitive Visualisierung komplexer numerischer Repräsentationen wissenschaftlicher Konzepte ist ein Beispiel für die Darstellung abstrakter Sachverhalte. (Bryson, 1996)

Die Tatsache, dass mehrere Benutzer gleichzeitig innerhalb einer immersiven Informationsvisualisierung navigieren, interagieren und miteinander kommunizieren können, macht virtuelle Umgebungen zu einem geeigneten Informations- und Kommunikationsmedium für Anwendungen im Umfeld der CSCW. Viele VR Anwendungen nutzen die generelle Möglichkeit, virtuelle Welten als verteilte Arbeitsumgebungen zu gestalten und einzusetzen. Das hohe Maß an Intuitivität in der Informationsvisualisierung und die vielfältigen Interaktionsmöglichkeiten innerhalb dieser Darstellungen machen

VR darüber hinaus zu einem geeigneten Medium für Trainings- und Ausbildungssysteme, die in unterschiedlichen Branchen zum Einsatz gelangen können. (Leinenbach, 2000)

branchenspezifische Anwendungsgebiete	**branchenneutrale Anwendungsgebiete**			
	Informationsvisualisierung	Computer supported cooperative work	Computer based training & education	Teleoperation
militärische Anwendungen				
Kampfsimulation	gering	mittel	groß	
Flug-/ Fahrsimulation	mittel		groß	
Virtual Walkthrough	groß			
Virtual Human	groß		groß	
Operationstraining	groß		groß	
Dienstleistungen				
Telediagnose/-eingriff	groß	mittel		groß
Computerspiele	groß	mittel	gering	
Edutainment	groß	mittel	groß	
Virtual Gallery	groß	gering	gering	
Industrie				
EIS/DSS	groß			
Fernbedienung/-service	groß	mittel		groß
Electronic Commerce	groß			
Prozessmanagement	groß	groß		
Informationsmanagement	groß	groß		
Visual Engineering	groß	groß		

Tabelle 2.7: Anwendungsgebiete von VR (Leinenbach, 2000)

2.7 Bewertung des gegenwärtigen Stands der Technik

Aus den konstruktionsmethodischen Grundlagen heraus kann formuliert werden, dass die funktionale Produktbeschreibung auf unterschiedlichen Abstraktionsniveaus existiert, beginnend mit der Formalisierung der Aufgabenstellung durch einen begrenzten Satz an Grundfunktionen bis hin zur Beschreibung der Produkttopologie und der Wirkzusammenhänge. In Abhängigkeit von der Abstraktionsebene der funktionalen Produktbeschreibung können jeweils verschiedene Beschreibungsformen definiert werden. Die Funktionen eines Produkts können in Hierarchien und/oder flussorientierten Netzwerken strukturiert werden. Die funktionale Produktbeschreibung wird im Konstruktionsprozess für die Beschreibung der Aufgaben des Produkts, der Funktionsweise des Produkts sowie für das Wiederfinden von bekannten, bereits ausgeführten Lösungen verwendet. Darüber hinaus wurde festgestellt, dass weder eine konkrete Klassifizierung in Hinblick auf spezielle Funktionsverben, noch auf Funktionsobjekte existiert. Die Literatur beschreibt hier nur Mischformen derselben. Eine ingenieursgerechte Modellierungstechnik wurde in Softwaresystemen bislang gar nicht oder nur unzureichend umgesetzt. Integrative Ansätze, die durchgängig von der Anforderungsmodellierung bis zur Gestaltmodellierung ein Produkt abbilden können, wurden zwar im Forschungskontext erarbeitet, konnten sich aber in der Praxis nicht etablieren.

Aus dem Bereich der 3D Modellierung wurden die im Kontext dieser Arbeit notwendigen 3D Modellierungsmethoden hinsichtlich ihrer wesentlichen Charakeristika diskutiert. Zur Erstellung von dreidimensionalen Objekten, die nicht den hohen Detaillierungsgrad eines CAD Modells bedürfen, sind Verfahren wie das Box Modelling hervorragend geeignet. Speziell für das Echtzeitrendering, in dem die Polygonanzahl möglichst gering gehalten werden muss, um einen flüssigen Ablauf zu gewährleisten, sind die direkten Modellierungsverfahren geeignet.

Mittels der vorgestellten Softwareschnittstellen wurden die Grundlagen einer auf XML basierten Datenübertragung zwischen Systemen analysiert. Dabei wurde der XML Standard, der im Verlauf der Arbeit verwendet werden soll, eingehend betrachtet, da er über einen sehr generischen Aufbau verfügt, und von einer großen Anzahl von Systemen unterstützt wird. Ebenso wurde die Interprozesskommunikation betrachtet, sowie eine datenbankbasierte und dateibasierte Übertragung analysiert. Zudem wurden die Abfra-

gesprache XPath und die XSL Transformation zwischen unterschiedlichen XML Schemas untersucht.

Die im Umfeld dieser Arbeit notwendigen Informationsvisualisierungsmethoden wurden hinsichtlich ihrer wesentlichen Eigenschaften analysiert. Dabei wurde der Fokus auf diejenigen Verfahren gelegt, die im Laufe der Lösungsfindung verwendet werden sollen. Es wurde festgestellt, dass die Zahl visueller Metaphern, die in kommerziellen Programmsystemen implementiert wurden eher gering ist. Viele prototypische Systeme wurden am Benutzer vorbei entwickelt. Oftmals wurden Projekte nur unter dem Aspekt der Anwendung neuer Technologien aus dem Bereich der Computergrafik oder Datenvisualisierung durchgeführt. Eine inhaltliche Auseinandersetzung mit der Frage, ob und wie abstrakte Daten unter dem Aspekt eines Informationsmehrwertes aufbereitet und visualisiert werden sollten, wurde meist nicht geführt. Im Interfacedesign ist zudem die Benutzerakzeptanz ein entscheidendes Qualitätskriterium. Hierzu fehlen jedoch oft aussagekräftige Evaluationen.

Im Bereich der virtuellen Realität bleibt festzustellen, dass sich die Bemühungen in der jüngeren Vergangenheit und in der Gegenwart auf die Leistungssteigerung und Kostenreduktion bereits existierender Systemkomponenten konzentrieren und damit der Verbreitung der Technologie selbst. Insbesondere die Entwicklung in der Hardware macht die VR Technologien erschwinglich, so dass diese mittlerweile sehr breit aufgestellt ist. Sie verfügt jedoch über keine Standardlösungen, die sich am Markt etabliert haben. So gibt es eine Reihe von wirtschaftlich erfolgreichen Unternehmen, die Hard- und Softwareprodukte für VR Anwendungen anbieten. Dabei haben sich die einsetzbaren Ein- und Ausgabegeräte in den letzten Jahren kaum verändert. Ebenso sind die Interaktionsmetaphern wenig weiterentwickelt worden, so dass ein Hauptmanko von VR der gewinnbringende Einsatz in bestehenden Wertschöpfungsketten bestehen bleibt.

Aus den hier diskutierten Lösungsansätzen wird ersichtlich, dass die Verwendung von virtuellen Umgebungen zur Darstellung von Funktionsstrukturen nicht existiert. Eine ingenieursgerechte, rechnerunterstützte Funktionsmodellierung ist ebenso nicht vorhanden. Des Weiteren wurde auch im Forschungskontext keine rechnerunterstützte Übertragung von Funktionsstrukturen sowie deren interaktive Visualisierung in virtuellen Umgebungen entwickelt. Als Konsequenz aus der Analyse ergibt sich die Forderung nach einem Konzept zur rechnerunterstützten Erstellung von Funktionsstrukturen, die direkt in eine virtuelle Entwicklungsumgebung übertragen,

und in einer entsprechenden Umgebung visualisiert werden können, so dass letztendlich das Potential von VR genutzt werden kann,

> *to allow a better, faster, and more intuitive exploration of very large data resources. This will not only be valuable in an economic sense, but will also stimulate and delight the user.* (Keim, 2002, p.6)

KAPITEL 3

Anforderungen an das Konzept

Die zielgerichtete virtuelle Darstellung von Informationen, die in den frühen Phasen der Produktentwicklung entsteht, erfordert eine möglichst vollständige Produktbeschreibung. Die im Kontext dieser Arbeit betrachteten Informationen werden hauptsächlich aus den Erfahrungen vorangegangener Produktvarianten, aus aktuellen Marktanalysen und Benchmark-Studien und den daraus resultierenden Anforderungen gewonnen. Dabei spielt die Informationsvollständigkeit im Rahmen einer frühzeitig beginnenden Qualitätssicherung eine wesentliche Rolle. Um diese Vollständigkeit zu garantieren, muss schlussendlich der gesamte Produktlebenszyklus betrachtet und die relevanten Daten aggregiert werden. Bild 3.1 zeigt die einzelnen Phasen des Produktlebenszyklus am Beispiel der Automobilindustrie nach Ovtcharova (Ovtcharova, 2006b).

Die Forderungen an eine frühstmögliche, geeignete virtuelle Darstellung dieser Informationen anhand von Funktionsmodellen in einer dementsprechend angereicherten, virtuellen Umgebung sollen in diesem Kapitel betrachtet werden. Dazu werden diese an den Abbildungsprozess der Funktionsmodelle erläutert, daraufhin die Forderungen an die Semantik sowie der Technologie innerhalb der virtuellen Umgebung formuliert. Im darauf folgenden Kapitel 4 auf Seite 93 wird auf Basis der Forderungen der zugrundeliegende Lösungsansatz vorgestellt.

Wichtige Aspekte im Gesamtkonzept bestehen unter anderem in der Nutzung geeigneter Funktionsmodellierungsmethoden, sowie in der Generierung zweckmäßiger visueller Interaktionsmetaphern im Rahmen einer skalierbaren VR Lösung.

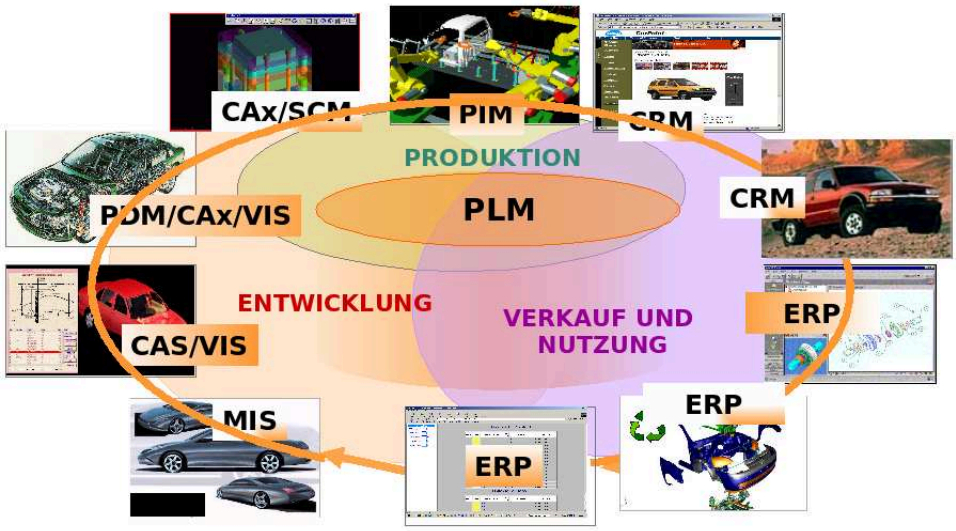


Bild 3.1: Produktlebenszyklus in der Automobilbranche (Ovtcharova, 2006b)

Dabei liefert der Aspekt der Funktionsmodellierung das Grundgerüst für die technische Spezifikation des zu entwickelnden Produktes (vgl. Kapitel 2.2 auf Seite 16). Aufbauend auf den Funktionshierarchien sowie den Funktionsstrukturen sollen diese genutzt werden, um Funktionsmodelle normalisiert in eine virtuelle Umgebung zu überführen. Der Prozess der Überführung von Funktionsblockdiagrammen auf Szenengraphen stellt einen wesentlichen Schwerpunkt der vorliegenden Arbeit dar.

Der zweite Aspekt besteht in dem kontextorientierten Gebrauch von Interaktionsmetaphern in VR (vgl. Kapitel 2.6.4 auf Seite 67). Die Metapher ist ursprünglich eine rhetorische Figur, bei der ein Wort nicht in seiner wörtlichen, sondern in einer übertragenden Bedeutung gebraucht wird, und zwar so, dass zwischen der wörtlich bezeichneten Sache und der übertragen Gemeinten eine Beziehung der Ähnlichkeit besteht. Der Duden definiert den Begriff Metapher als Wort mit übertragener Bedeutung oder als bildliche Wendung (Dudenredaktion, 2006). In der virtuellen Realität dienen diese Metaphern als Ausdruck einer Beziehung zwischen geometrischen Objekten und dem analogen, menschlichen Denken. Die geometrische Repräsentation eines Stuhles beispielsweise kann als Sitzmetapher begriffen werden. Jedoch sind Metaphern grundsätzlich nicht kohärent. Im Beispiel des Stuhles ist es ebenso denkbar, diesen als Leiter oder als Befestigungsgegenstand entfremded zu verwenden.

Die klare Unterscheidung zwischen Interaktionsmetaphern, die als Kommunikationsschnittstelle zwischen Mensch und Maschine dienen, und Funktionsmodellen, die technische Spezifikationen anwendergerecht repräsentieren, stellt sicher, konstruktive, technische Tätigkeiten und computergrafische, anwendungsorientierte Tätigkeiten strikt voneinander zu trennen.

3.1 Forderungen an die Übertragbarkeit von Funktionsmodellen

Die Verarbeitung der Anforderungen in den frühen Phasen der Produktentstehung zeichnet sich durch große Unsicherheiten in den Vorgaben für die Entwickler aus. Die vorhandenen Informationen sind in der Regel unscharf und werden durch iterative Prozesse erst im Laufe der Produktentwicklung konkretisiert.

3.1.1 Erstellung der Funktionsmodells

Anhand der Erstellung von Funktionshierarchien und Funktionsstrukturen wird das zukünftige Produkt erstmals funktional vollständig beschrieben und damit ein Grundstein für die weiteren Konstruktionsphasen gelegt. Mittels der Übertragung der Funktionsmodelle in eine virtuelle Umgebung soll das menschlich, kreative Denken stärker unterstützt werden (Krappe and Marinov, 2007, p.25f). Die sich daraus ergebenden, wesentlichen Forderungen an die Funktionsmodelle liegen in der Kohärenz von Funktionsverben und Funktionsobjekte bezogen auf die betrachteten Produktkomponenten, sowie in der Kohärenz der verwendeten Notation.

Um den Abbildungsprozess zweckmäßig zu methodisieren, besteht eine weitere Forderung in der Verwendung geeigneter Standards, was in Kapitel 3.3.3 näher diskutiert werden soll. Da die Erstellung von Funktionsmodellen im Maschinenbau durch kommerzielle Werkzeuge momentan nicht ausreichend unterstützt wird (vgl. Kapitel 2.2.2 auf Seite 18), muss eine Möglichkeit konzipiert werden, Funktionsmodelle ingenieursgerecht zu erstellen. Dabei sollen Methoden aus anderen Domänen auf ihre Übertragbarkeit analysiert und entsprechend in Betracht gezogen werden.

3.1.2 Erstellung der 3D Szenen

Eine zentrale Rolle zur Erstellung von 3D Szenen in virtuellen Umgebungen spielt der Szenengraph (vgl. Kapitel 2.6.3 auf Seite 65). Die Anzeigesoftware traversiert diesen Graphen und akkumuliert entlang eines jeden Pfades, die in den Knoten enthaltenen Informationen, etwa Transformationen, Farbinformationen und geometrische Daten. Ein Knoten kann über verschiedene Pfade erreicht werden, d.h. er kann in unterschiedlichen Kontexten ausgewertet werden. In klassischer 3D Software wurde der Szenengraph im Programmcode des 3D Systems selbst aufgebaut (Diehl, 2002). Dabei war er in dem 3D System fest integriert. Um jedoch die Abbildung von Funktionsmodellen auf Szenengraphen zu ermöglichen, muss dieser aus der Anwendung auslesbar sein. Darüber hinaus sollen die Spezifikationen der Knoten und Felder des verwendeten Szenengraphen eindeutig spezifiziert werden können.

3.1.3 Schnittstellen

Neben der Verwendung eines ingenieursgerechten Werkzeugs zur Funktionsmodellerstellung sowie der klar spezifizierten Struktur einer 3D Szene besteht eine weitere Forderung in der Nutzung transparenter Datenschnittstellen (vgl. Kapitel 2.4 auf Seite 33). Wulf (Halbach, 1994) definiert Schnittstellen als den Teil eines Systems, der der Kommunikation dient. Schnittstellen können unterschieden werden in Datenschnittstellen, Maschinenschnittstellen, Hardwareschnittstellen, Softwareschnittstellen und Benutzerschnittstellen. Im Rahmen des Abbildungsprozesses von Funktionsmodellen auf Szenengraphen sind die Softwareschnittstellen von besonderem Interesse (vgl. Kapitel 2.4.1 auf Seite 33). Basierend auf offenen Standards wie SGML oder XML sollen Austauschformate benutzt werden, die weit verbreitet und eindeutig definiert sind.

3.2 Forderungen an die Semantik in virtuellen Umgebungen

Die Kombination von virtueller Realität und Informationsvisualisierung ermöglicht ein besseres Verständnis der Relationen zwischen Wahrnehmung

und abstrakten Informationen (Bowman et al., 2003, p. 81ff). Die effektivsten Informationsarten sind jene, die sehr eng mit der Umgebung verknüpft sind.

3.2.1 Visualisierung in angereicherten virtuellen Umgebungen

Eine der wesentlichen Anforderungen an angereicherte virtuelle Umgebungen leitet sich aus dem Credo Shneidermans (Shneiderman, 1996, p. 336ff) ab. Dadurch soll eine bedarfsgerechte Visualisierung abstrakter Detailinformationen zu einem bestimmten Objekt der virtuellen Szene ermöglicht werden. Um eine solche Funktionalität zu gewährleisten, muss der Benutzer zunächst erkennen können, dass mit einem Objekt weitere Informationen assoziiert sind. Weiterhin muss er das Objekt selektieren können. Auf diese Weise wird die Visualisierung der Zusatzinformationen ermöglicht. (Scherer, 2007)

Desweiteren muss während der Nutzung von virtuellen Umgebungen die Visualisierung der Informationen für den Anwender möglichst effizient gestaltet werden. Zur Auswahl des Anzeigeortes soll dabei zwischen weltfixierten, displayfixierten, objektfixierten und anwenderfixierten Bezugssystemen unterschieden werden. Die weltfixierten Informationen sollen im Weltkoordinatensystem der virtuellen Umgebung platziert werden. Displayfixierte Informationen sollen sich ähnlich wie eine Blende vor der Kamera verhalten und immer im Vordergrund angezeigt werden. Objektfixierte Informationen sollen mit einem bestimmten Objekt verknüpft werden und diesem stets folgen. Anwenderfixierte Informationen sollen mit der Blickrichtung verknüpft werden und immer orthogonal zu dieser stehen. Die dabei entstehenden sogenannten Billboards sollen eine durchgängige Lesbarkeit der dargestellten Informationen garantieren. (Scherer, 2007)

Eine weitere Forderung besteht in der impliziten Darstellbarkeit abstrakter Informationen. So sollte es beispielsweise ermöglicht werden, verschiedene Farben von Objekten als Indikatoren von Eigenschaften dieser zu verwenden. Ebenso muss es möglich sein, textuelle Informationen in die virtuelle Umgebung zu integrieren. Dabei steht die Lesbarkeit stets im Vordergrund. Um die Übersichtlichkeit zu erhalten und gleichzeitig gute Lesbarkeit sicherzustellen, müssen Konzepte erarbeitet werden, die beispielsweise die Abhängigkeit von Objektentfernung berücksichtigen.

3.2.2 Level of Immersion

Weiterhin ist das Abtauchen in eine virtuelle Umgebung eng mit dem so genannten Immersionsgrad (Level of Immersion) verknüpft. Teilweise kann dieser durch photorealistische Darstellung (Merkmal der Imagination, vgl. Kapitel 2.6.1 auf Seite 55) erhöht werden, aber nach Bartle (Bartle, 2006) ist dies nicht allein ausschlaggebend. Der höchste Immersionsgrad wäre dann erreicht, wenn der Benutzer bereit ist, sich als Teil der virtuellen Umgebung zu sehen. Dazu werden insbesondere im Diskurs des Game Designs vier unterschiedliche Levels of Immersion unterschieden (Bartle, 2006), nämlich Player, Avatar*, Character und Persona. Diese unterscheiden sich wie folgt:

- Player - Der Player ist ein Mittel zur Beeinflussung der virtuellen Umgebung.
- Avatar - Der Avatar ist ein Repräsentant des Anwenders in der virtuellen Umgebung.
- Character - Der Character ist ein vom Computer generierter Player.
- Persona - Die Persona ist Teil der Identität des Anwenders. Der Anwender ist selbst mit der virtuellen Welt verbunden.

Im Rahmen dieser Arbeit besteht die Forderung, dass für die Minimalrepräsentation des Level of Immersion, wenigstens die Stufe eines Avatar erreicht werden soll.

3.2.3 Befinden des Anwenders

Neben den Forderungen nach einer geeigneten Visualiserung und dem Level of Immersion stellt die Forderung nach dem Wohlbefinden der Anwenders einen weiteren wesentlichen Aspekt dar. Eines der möglichen Probleme, das in virtuellen Umgebungen auftreten kann und welches bei der Entwicklung von VR Inhalten zu beachten ist, ist die Simulatorkrankheit†. Bei einigen Personen können während oder nach längerem Aufenthalt in einer virtuellen Umgebung Probleme auftreten, welche sich meist in Schwindelgefhlen, Kopfschmerzen und Desorientierung zeigen. Die Symptome ähneln denen

* von Sanskrit “avatar” = grafische Darstellung, Animation, Karikatur o.ä. als Verkörperung des Benutzers im Cyberspace

† engl. simulator sickness

der Kinetose[‡], welche auch als Reise- oder Bewegungskrankheit bekannt ist. Die Kinetose wird durch starke Reizung des Gleichgewichtsorgans (Vestibularapparat) infolge von Einwirkung von Progressiv-, Zentrifugal-, Winkel-, Coriolis Beschleunigungen, aber auch durch fehlende Übereinstimmung (Diskordanz) der Erregung des Gesichts- u. Gleichgewichtssinnes und über eine Reizung vegetativer Stammhirnzellen hervorgerufen (LaRoche, 1999). Im Gegensatz zur Kinetose lässt sich bei der Simulatorkrankheit kein einzelner Faktor für das Auslösen der Probleme isolieren (LaViola, Jr., 2000). In „Simulator Sickness in Virtual Environments“ (Kolasinski, 1995) werden verschiedene Faktoren aufgezeigt, welche für das Auftreten der Simulatorkrankheit verantwortlich sein können. Dabei wird zwischen Faktoren unterschieden, die vom Anwender, von den eingesetzten Technologien oder vom Inhalt der VR Umgebung abhängen. Im Kontext der vorliegenden Arbeit werde solche Faktoren betrachtet, die von der eingesetzten Technologie und den dargestellten Inhalten beeinflusst werden.

3.3 Forderungen an die Technologie in virtuellen Umgebungen

Die technologischen Grundlagen für virtuelle Umgebungen hängen von den Merkmalen der virtuellen Realität, Immersion, Interaktion und Imagination ab (vgl. Kapitel 2.6.1 auf Seite 52). Nach Ovtcharova (Ovtcharova, 2006a) wird die Imagination durch Intelligenz (Semantik) ersetzt. Diese Änderung hebt die zugrundeliegende, kontextbezogene Verknüpfung der Produktdaten stärker hervor. Bild 3.2 ordnet die drei Merkmale in einem drei dimensionalen Raum an.

Dementsprechend leiten sich die wesentlichen Anforderungen aus der Fähigkeit zur Immersion, der Möglichkeit zur Interaktion sowie aus dem Integrationspotential in bereits bestehende Entwicklungsumgebungen ab. Die im Rahmen dieser Arbeit relevanten Integrationsaspekte fokussieren auf die Prozessintegration, die Erweiterbarkeit, die Skalierbarkeit sowie die Nutzung bestehender Standards.

Mit der Darstellung komplexer dreidimensionaler Welten in Echtzeit werden die Grundvoraussetzungen für virtuelle Umgebungen geschaffen. Dazu muss die graphische Berechnung einer Szene, das Rendering, von der Hardware

[‡] engl. motion sickness

unterstützt werden. Aus dieser Hardwareunterstützung sowie der Architektur eines Renderingsystems ergeben sich verschiedene Anforderungen, welche beim Aufbau einer virtuellen Umgebung beachtet werden müssen, wie z.B. die Darstellung von Objekten durch Polygone oder die Strukturierung der Daten (Lutz, 2004).

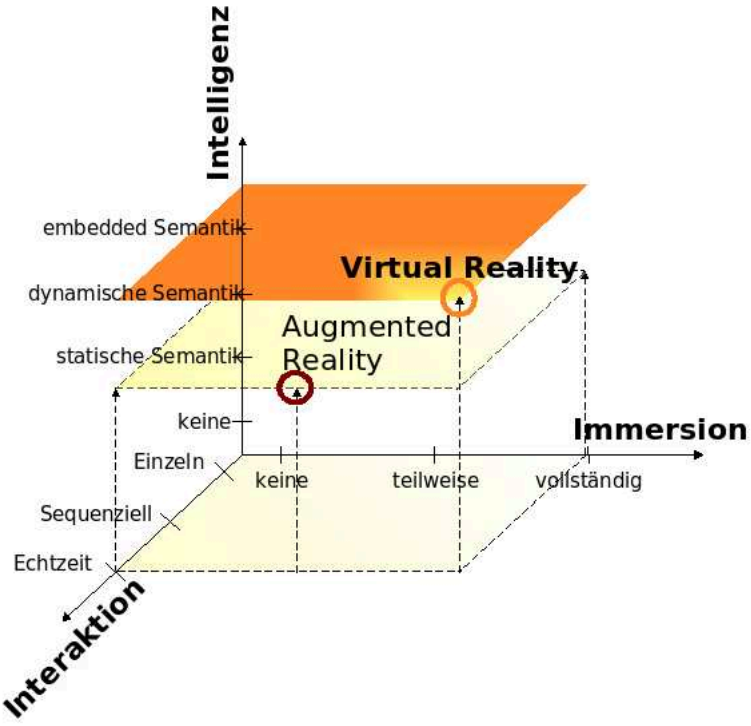


Bild 3.2: Achsen virtueller Umgebungen (Ovtcharova, 2006a)

3.3.1 Sicht der Immersion

Als entscheidenes Merkmal einer virtuellen Umgebung gilt die Immersion (vgl. Kapitel 2.6.1 auf Seite 54). Sie ist neben der Interaktion und der Semantik (vgl. Kapitel 3.3) mitverantwortlich für die unterschiedliche Erlebnistiefe, mit der virtuelle Realität erlebt werden kann.

Um beim Anwender ein Gefühl der Immersion zu erzeugen, müssen verschiedene Aspekte berücksichtigt werden. Wesentliche Bestandteile einer immersiven virtuellen Umgebung sind geeignete Ausgabegeräte, welche das Blickfeld des Benutzers möglichst ausfüllen, wie z.B. HMDs, Powerwalls[§] oder

[§] Grobildleinwand

CAVEs und mit denen eine stereoskopische Darstellung möglich ist. Darüber hinaus sollte der visuelle Eindruck noch durch Einbindung weiterer Sinne, wie beispielsweise durch Töne, unterstützt werden. Henning (Henning, 1997) fasst die immersionshemmenden bzw. immersionsfördernden Einflussfaktoren tabellarisch zusammen. Tabelle 3.1 gibt diesen Zusammenhang wieder.

Immersionshemmend	**Immersionsfördernd**
geringer Interaktivitätsgrad	hoher Interaktivitätsgrad
geringe Bildrate	hohe Bildrate
geringe Bildkomplexität	hohe Bildkomplexität
geringe Bildauflösung	hohe Bildauflösung
wenige Benutzeraktionen	viele Benutzeraktionen
Darstellung auf Bildschirm	Darstellung im HMD
monoskopisches Sehen	stereoskopisches Sehen
eingeschränktes Sichtfeld	uneingeschränktes Sichtfeld
kein Tracking	Tracking
kein Ton	3D Ton

Tabelle 3.1: Einflussfaktoren auf den Immersionsgrad

Die Qualität der Darstellung muss an die Anforderungen des Inhalts und der aus dem Konzept abgeleiteten Methodik angepasst sein. Dies muss nicht zwingend in photorealistische Darstellung münden. Nichtsdestotrotz muss die virtuelle Umgebung aber soweit mit den Erwartungen des Benutzers übereinstimmen, dass er bereit ist, sich als Teil dieser zu sehen.

3.3.2 Sicht der Interaktion

Neben der Immersion ist die Interaktion ein weiterer wesentlicher Bestandteil einer virtuellen Umgebung (vgl. Kapitel 2.6.1 auf Seite 55). Während in den letzten Jahren der Fokus der Forschung stark auf dem Merkmal der Imagination (vgl. Kapitel 2.6.1 auf Seite 55), also den Rendering- und Visualisierungsprozessen lag, ist heute eine Neuausrichtung hin zur sozialen und objektorientierten Interaktion zu beobachten. Laut Frederick P. Brooks (Brooks, Jr., 2003) besteht momentan die größte Herausforderung darin, virtuelle Realitäten für soziale Interaktion nutzbar zu machen.

Die Möglichkeit zur Interaktion resultiert aus der Echtzeitfähigkeit der vorhandenen Hardware, da nur so der Rechner auf jede Eingabe des Benutzers

sofort reagieren kann. Um den Ablauf der Präsentation interaktiv verändern zu können, benötigt der Benutzer spezielle Interaktionsgeräte. Diese Geräte müssen entsprechend den Voraussetzungen, die der Benutzer mit sich bringt, entworfen sein. Ebenso müssen sie geeignet sein, den semantischen Forderungen an eine virtuelle Umgebung, wie sie in Kap. 3.2 auf Seite 82 erläutert wurden, gerecht zu werden.

Generell gilt, dass eine VR Szene immer nach Interaktion verlangt. Wenn eine virtuelle Szene den Anspruch hat hoch immersiv zu sein, dann bedingt dies gleichzeitig einen hohen Grad an Interaktion (Brenzinger, 2007). Das immersive Erlebnis wird umso stärker, je mehr die virtuelle Umgebung auf Aktionen des Anwenders reagieren kann und somit die Interaktion zwischen Realität und Virtualität vertieft wird.

Daher besteht eine wesentliche Forderung an Interaktion darin, auf möglichst alle Aktionen des Nutzers kontinuierlich und durchaus überraschend zu reagieren (vgl. Kapitel 2.6.1 auf Seite 55). Aus dieser Forderung entspringt automatisch die Folgerung, dass ein zu nutzendes Visualisierungswerkzeug in der Lage sein muss, dem Entwickler die Möglichkeit zu geben, adaptiv kontinuierliche Interaktivität in eine Szene an jedem Interaktionsobjekt einzubringen.

Alle Interaktionen stellen weiterhin die Anforderung intuitiv bedienbar zu sein, und in Echtzeit abzulaufen, da sonst keine immersive, virtuelle Umgebung entstehen kann. Die Forderung nach Echtzeit korreliert mit den vorhandenen Ein- und Ausgabegeräten. Die hieraus resultierenden Probleme, wie Latenz, Präzision, Verdeckung etc. müssen vorab bedacht werden. Dabei sind Information wie Bewegungsrichtung oder aktuelle Entfernung zum Interaktionsobjekt maßgeblich, um dem Anwender zu helfen, sich in einer virtuellen Umgebung zu Recht zu finden.

Die Interaktion stellt aber nicht nur Forderungen an die Hardware, sondern auch an die rechnerinterne Repräsentation der Szene. So müssen beispielsweise die 3D Modelle so strukturiert sein, dass diese getrennt voneinander zu manipulieren sind.

3.3.3 Sicht der Integration

Die Betrachtung der Integration der Ergebnisse aus virtuellen Umgebungen in bereits bestehende Entwicklungsumgebungen ist für die erfolgreiche Verwendung dieser unabdingbar. Dabei stehen die Aspekte Prozessintegration,

Erweiterbarkeit, Skalierbarkeit und Standardisierung im Vordergrund, da diese direkten Bezug zum Kontext dieser Arbeit haben.

Prozessintegration

Generell ist die Akzeptanz, ein Softwaresystem einzuführen, sehr davon abhängig, inwieweit bestehende Daten weiterverwendet bzw. ohne Verlust migriert werden können (Scheer, 1992). Vorhandene Daten aus beispielsweise CAD oder PDM-Systeme sind "de facto" Säulen einer jeden Produktentwicklung. Die Einführung von angereicherten, virtuellen Umgebungen kann in der industriellen Praxis nur so geschehen, dass existierende Prozessketten nicht ersetzt, sondern in einen semantischen Bezug gebracht werden (Krappe, 2005, p.47f). Die Forderung lautet dementsprechend, die Wiederverwendung bestehender Daten und Weiterverwendung erzeugter Daten in einer durchgängigen Prozesskette zu gewährleisten. Dazu müssen die notwendigen, offenen Schnittstellen geschaffen werden, die eine einfache Migration der Daten über Systemgrenzen hinweg ermöglicht.

Erweiterbarkeit

Die Forderung nach Erweiterbarkeit des Visualisierungswerkzeugs bezieht sich auf die Fähigkeit der Software, mittel- und langfristig einsatzfähig zu sein. Diese Forderung muss vor dem Hintergrund des Mooreschen Gesetzes betrachtet werden, das besagt, dass sich die Anzahl an Transistoren auf einem handelsüblichen Prozessor alle achtzehn Monate verdoppelt. Raymond Kurzweil bezieht dieses Gesetz auf die Rechenleistung pro 1000 Dollar und stellt fest, dass diese sich in den Jahren 1910 bis 1950 im Abstand von drei Jahren (mechanische Rechenmaschinen), von 1950 bis 1966 etwa alle zwei Jahre und jetzt etwa jährlich verdoppelt (Kurzweil, 2001). Im Zusammenhang mit virtuellen Umgebungen folgt, dass die Eingliederung dieser in bestehende Prozessketten nur dann sinnvoll ist, wenn die Erweiterbarkeit hinsichtlich der Leistungsfähigkeit der gesamten Wertschöpfungskette des Unternehmens berücksichtigt wird.

Skalierbarkeit

Die Leistungssteigerung wird gegenwärtig als priorisiertes Ziel angesehen, jedoch unter Berücksichtigung der anfallenden Kosten. Um einen hohen Grad

an Immersion zu gewährleisten, ist die Darstellung der Szenen in Echtzeit[¶] unabdingbar. In der industriellen Praxis sind zwei Strategien möglich:

- Hochleistungsrechner wie z.B. SGI Onyx oder Origin mit mehreren Grafikpipelines
- Zusammenschalten (Clustering) von Standardrechner

Momentan lässt sich der Trend beobachten, dass monolithische Hochleistungsrechner mehr und mehr an Bedeutung verlieren und mit vertretbaren Qualitätseinbußen die erforderliche Leistung auch von einem Verbund (Cluster) von Standardrechnern erbracht werden kann (vgl. Kapitel 2.6.3 auf Seite 64). Daraus ergibt sich die Forderung, durch Clustering skalieren zu können. Allgemein lässt sich sagen, dass ein Softwareprodukt gut skaliert, wenn es bei der zehnfachen Leistung proportional mit den zehnfachen Ressourcen auskommt (Richter, 2005).

Eine virtuelle Umgebung muss in mehreren Hinsichten skalierbar sein. Eine einzige große Powerwall muss in mehrere Bereiche aufteilbar sein, wobei die einzelnen Prozessoren jeweils einen Teil des zu visualisierenden Universums[‖] erzeugen. Auch andere Konfigurationen, wie z.B. CAVEs mit bis zu zwei Projektoren pro Wand, müssen unterstützt werden. Allerdings sollte das System letztlich so aufgebaut sein, dass die Kosten für Hard- und Software skalierbar sind. Die gleiche Anwendung soll, wenn auch in geringerer Qualität, auch auf einfacherer Hardware funktionieren.

Verwendung von Standards

Wird über Vereinheitlichung, gleich welcher Ausprägung gesprochen oder geschrieben, so wird häufig auf dem Begriff Standard bzw. Standardisierung zurückgegriffen. Der Brockhaus definiert Standard allgemein als

> 1) Richtschnur, Maßstab, Norm; 2) Normalmaß, Normalausführung einer Ware; 3) die im allgemeinen Qualitäts- und Leistungsniveau erreichte Höhe.(Brockhaus, 2006, p.79)

[¶] Echtzeit bedeutet die Zeit, die Abläufe in der realen Welt verbrauchen und die mit wenigsten 24 frames per second (fps) angezeigt werden

[‖] Bei auf Szenengraphen basierten Systemen wird unter einem Universum die Gesamtheit aller Objekte die zu Visualisieren sind verstanden. Dazu können Lichtquellen, Kameras, geometrische Objekte etc. gehören

Die Verwendung von Standards ist im Sinne der Prozessintegration eine Notwendigkeit. Insbesondere die Softwareschnittstellen müssen auf normierte Standards oder Industriestandards aufsetzen, um überhaupt eine industriegerechte Prozessdurchgängigkeit zu ermöglichen. Offene Datenaustauschformate wie XML, VRML oder STEP und Industriestandards wie CSV oder 3DS sollen in diesem Rahmen angewendet werden.

Weitere Anforderungen

Zusätzlich zu den oben genannten Forderungen existieren noch weitere, allgemeine Forderungen, die ebenfalls berücksichtigt werden müssen.

Stabilität Als erstes ist hier die Stabilität zu nennen. Unterbrechungen, die durch abgestürzte Hard- oder Software entstehen, können nicht geduldet werden.

Robustheit Auch müssen alle eingesetzten Geräte robust genug sein, um die Benutzung durch meist ungeübte Benutzer unbeschadet zu überstehen.

Bedienbarkeit Ein weiterer wichtiger Punkt ist die Bedienbarkeit. Sowohl das Bedienen des Systems, d.h. das Starten und Stoppen der virtuellen Umgebung, als auch die Bedienung müssen ohne größere Vorkenntnisse und Übung möglich sein.

Sicherheit Weiterhin sind Sicherheitsaspekte zu betrachten. Es darf zu keiner Zeit eine Gefährdung der beteiligten Personen bestehen und Effekte wie die Simulatorkrankheit (vgl. Kap. 3.2.3 auf Seite 84) müssen so weit wie möglich reduziert werden.

3.4 Zusammenfassung der Anforderungen

Tabelle 3.2 fasst die Forderungen an den Lösungsweg, der im Rahmen dieser Arbeit entwickelt wird, zusammen.

Bereich	**Anforderungen**
Funktionsmodellierung	Ingenieursgerechte, softwareunterstützte Erstellung von Funktionsmodellen
	Verwendung von Modellierungsstandards
	Verwendung kohärenter Funktionsverben und Funktionsobjekte
	(Semi-)automatische, rechnerunterstützte Übertragbarkeit der Funktionsmodelle
3D Modellierung	Auslesbarer, eindeutig spezifizierter Szenengraph
	Echtzeitfähige rechnerinterne 3D Modelle
	Eindeutige, übertragbare Koordinatensysteme
Datenübertragung	Transparente Datenschnittstellen
	Standardisierte Softwareschnittstellen und Austauschformate
	Mögliche Anbindbarkeit an bereits bestehende Softwarewerkzeuge
Informationsvisualisierung	Overview first, Zoom and Filter, Detail-on-Demand
	Bedarfs- und kontextgerechte Visualisierung
	Effiziente Informationsaufbereitung
	Implizite Darstellung abstrakter Informationen
Virtuelle Realität	Fähigkeit zur Immersion
	Geeignete Ausgabegeräte
	Einbindung weiterer Sinne
	Möglichkeit der Interaktion
	Kontextgerechte Interaktionsmetaphern
	Geeignete Interaktions-/ Eingabegeräte
	Echtzeitfähige, manipulierbare Szenengraphen
	Sozial, Objektorientiert, Intuitiv
	Integrationsaspekte
	Integration in bestehende Prozesse
	Erweiterbarkeit und Skalierbarkeit der Systeme
Anwenderakzeptanz	Hoher Immersionsgrad
	Wohlbefinden des Anwenders
Allgemein	Stabilität, Robustheit, Sicherheit, Bedienbarkeit

Tabelle 3.2: Zusammenstellung der Anforderungen

KAPITEL 4

Konzept für die immersive Verwendung von Funktionsmodellen

Ziel dieser Arbeit ist es, Funktionsmodelle in virtuellen Umgebungen nutzbar zu machen. Dabei spielt die interaktive, immersive Visualisierung der Zusammenhänge der Funktionen von Produktkomponenten für das Gesamtverständnis komplexer Produkte eine tragende Rolle. Ein Produkt hat keine fest definierte Funktionalität, da es immer darauf ankommt, wofür und wie es der Anwender einsetzt. Es kann prinzipiell für viele unterschiedliche Funktionen eingesetzt werden. Somit dient die Visualisierung dazu, die vom Konstrukteur beabsichtige Funktionsweise bildlich darzustellen.

Die Vorgehensweise ist dadurch motiviert, dass die virtuelle Realität durch das ihr innewohnende Potential die Möglichkeit birgt, mittels neuartiger Ein- und Ausgabegeräte sowie innovativen Interaktionsmetaphern dreidimensionale Welten über die Physis hinaus zu erweitern. Die Abbildung von, dem menschlichen Denken nahen, Funktionmodellen und deren Anreicherung mit zusätzlichen Informationen ermöglicht eine ganz neue Sicht auf den Produktentwicklungsprozess. Dabei wird zum einen die Kreativität der an der Produktentwicklung beteiligten Entwickler gefördert und zum anderen die entwicklungsteamübergreifende Kommunikation unterstützt.

Der Lösungsansatz des in dieser Arbeit vorgestellten Konzepts zur interaktiven, immersiven Verwendung von Funktionsmodellen basiert auf den

unterstützenden Einsatz spezieller Interaktionsmetaphern in virtuellen Umgebungen, die audible und visuelle Zusatzinformationen an die im Produktentwicklungsprozess beteiligten Personen auf Basis des Grundgerüsts Funktionsmodell zur Verfügung stellen. Bild 4.1 zeigt in einer Übersicht die nach diesen Ansatz durchzuführenden Schritte zur Abbildung von Funktionsmodellen in einer virtuellen Umgebung. Diese Schritte werden zunächst kurz vorgestellt und dann in den folgenden Abschnitten ausführlich beschrieben.

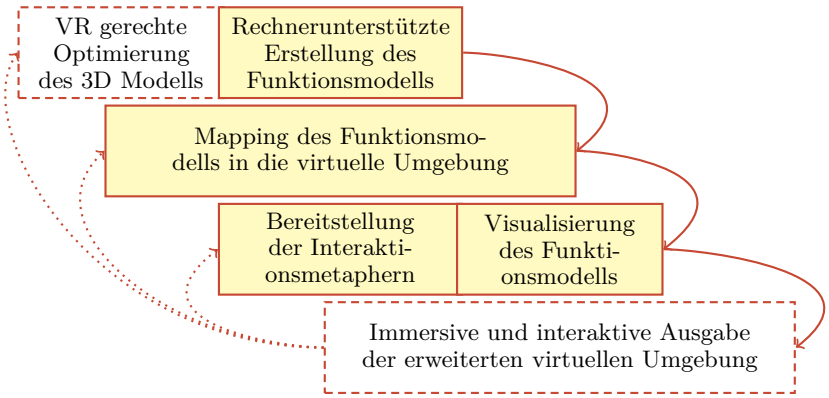


Bild 4.1: Übersicht über das methodische Vorgehen

Ausgangspunkt für die Konzeptentwicklung kann entweder die Erstellung der 3D Szene oder die Erstellung des Funktionsmodells sein. Da je nach Konstruktionsart* unterschiedliche Datenstände vorliegen, wird der Konstrukteur beispielsweise bestehende 3D Modelle bei einer Anpassungs- oder Variantenkonstruktion wiederverwenden und dementsprechend lediglich die Funktionsstruktur neu erstellen. Bei Neukonstruktionen hingegen ist die Erstellung sowohl des 3D Modells als auch der Funktionshierarchie und der Funktionsstruktur erforderlich. Wird mit der Entwicklung des 3D Modells begonnen, so muss dieses alle relevanten Informationen wie Gestalt, Beleuchtung, Detaillierung etc. enthalten. Die Echtzeitdarstellung der 3D Szene ist für die Erweiterung virtueller Umgebungen unabdingbar und wird neben der Erstellung derselben in Kapitel 4.1 beschrieben.

*Pahl/ Beitz unterscheiden drei Konstruktionsarten (Pahl, 2007). Während der Neukonstruktion wird ein neues Lösungsprinzip für ein System bei gleicher, geänderter oder neuer Aufgabenstellung erarbeitet. Bei der Anpassungskonstruktion wird ein bereits bekanntes System bei gleichgebliebenem Lösungsprinzip an eine veränderte Aufgabenstellung mit teilweise unterschiedlicher Funktion angepasst. Die Variantenkonstruktion variiert von Größe und/oder Anordnung innerhalb vorausgedachter Grenzen, wobei Funktion und Lösungsprinzip gleich bleiben, wie bei Baureihen und Baukästen.

Basierend auf diesen Informationen wird im nächsten Schritt das Funktionsmodell respektive die Funktionshierarchie und die Funktionsstruktur erstellt. Die Funktionshierarchie ist an die Produktaufgabensätze von Roth angelehnt (vgl. Kapitel 2.2.1 auf Seite 17) und ähnelt in der Darstellung der FAST Methode (vgl. Kapitel 2.2.3 auf Seite 21). Sie bildet unter anderem die Verbindung zwischen der Phase der Anforderungsmodellierung und der Phase der Funktionsmodellierung und wird in Kapitel 4.2.1 erläutert. Die Funktionsstruktur ist an die Notation von Langlotz angelehnt (vgl. Kapitel 2.2.2 auf Seite 18) und löst sich von der in der Konstruktionsmethodik gängigen, unpräzisen Bedeutung. Zur Erstellung der Funktionsstruktur werden die Funktionsflüsse analysiert und mittels Funktionsobjekte und Funktionsverben, die physikalische Ein- und Ausgangsgrößen miteinander verbinden, abgebildet. Kapitel 4.2.2 beschreibt den Erstellungsprozess detailliert.

Im dritten Schritt wird das Funktionsmodell und der Szenengraph mittels geeigneter Datenschnittstellen (vgl. Kapitel 2.4 auf Seite 33) verknüpft. Dieses Mapping bildet den ersten wesentlichen Schwerpunkt des Konzepts und wird in Kapitel 4.2.4 und Kapitel 4.2.5 ausführlich beschrieben.

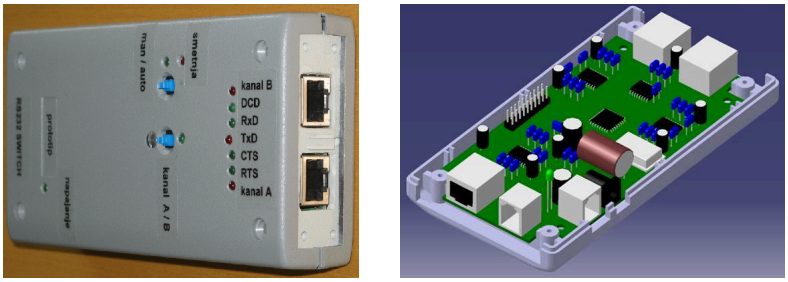
Im vierten Schritt werden die Interaktionen definiert. Dazu werden auf Basis der vorhandenen Ein- und Ausgabemöglichkeiten die für eine Erweiterung der virtuellen Umgebung zweckmäßigen Interaktionsmetaphern (vgl. Kapitel 2.6.4 auf Seite 67) entwickelt. Diese dienen der gezielten Interaktion zwischen Benutzer und System und stellen den zweiten Schwerpunkt des Konzeptes dar. Kapitel 4.3.2 widmet sich ausführlich der Beschreibung des Vorgehens.

Während des fünften Schritts werden die Funktionsmodelle in der virtuellen Umgebung adäquat dargestellt (vgl. Kapitel 2.5 auf Seite 37). Dazu werden einerseits die Funktionshierarchien als Taxonomie für die virtuelle Szene verwendet und andererseits die Funktionsstrukturen zur Sichtenbildung herangezogen. Kapitel 4.3.1 und Kapitel 4.3.3 diskutieren die Zusammenhänge.

Durch die immersive, interaktive, stereoskopische Darstellung soll das Verständnis komplexer Produkte erleichtert werden und so zu einem raschen Informationsrückfluss führen. Dieser wird durch die Beteiligten in den entsprechenden Prozessschritten und Legacy-Systemen zurückgeführt.

Der dargestellte Prozess gibt eine Übersicht über die Vorgehensweise von der Modellerstellung, von Funktion und Gestalt, der Kopplung von Funktionsmodell und 3D Szene hin zur Visualisierung dieser in virtuellen Umge-

bungen. In den folgenden Abschnitten werden die einzelnen Prozessschritte ausführlich vorgestellt und in Kapitel 4.1 und Kapitel 4.2 anhand des Beispielprodukts *Switch* erklärt (Bild 4.2).



(a) Reales Produkt (b) 3D-CAD Modell

Bild 4.2: Beispielprodukt: Switch

4.1 Erstellen der 3D Umgebung

Die gegenwärtige Leistung von Computern ist nicht in der Lage, die Realität in ihrer Gesamtkomplexität in Echtzeit zu reproduzieren. Um die Realität aber möglichst realitätsgetreu abzubilden, muss in Bereichen, in denen dies zu komplex ist, die Realität vereinfacht dargestellt werden. Dies kann zum einen durch bewusstes Ausnutzen der Schwächen des Systems Mensch getan werden, beispielsweise indem der Macula/Fovea† Effekt ausgenutzt wird, zum anderen können die im Bereich der Computergraphik entwickelten Algorithmen zur Simplifizierung von 3D Grafiken genutzt werden. Das Bild 4.3 stellt die beteiligten Systeme und die systemtechnischen Abläufe der Visualisierung aus Softwaresicht dar. Tabelle 3.2 auf Seite 92 fasst die Anforderungen an das 3D Modell präzise zusammen.

Die Anwendungssysteme, die eine Entwicklungsumgebung zur Erstellung der Funktionsmodelle oder zur Modellierung der virtuellen Umgebung zur Verfügung stellen, verwenden die vom Betriebssystem über die API bereitgestellten Grafiksysteme. Die Render Pipeline ist dabei ein Teil des Grafiksystems. Die Vorgänge, die innerhalb der Render Pipeline ablaufen, werden

† Als Macula lutea (der gelbe Fleck) wird der Bereich der menschlichen Netzhaut mit der größten Dichte von Sehzellen bezeichnet. Das schärfste Sehen findet in der Fovea centralis statt, einem Bestandteil des Macula lutea, der ausschließlich Zapfen enthält (LaRoche, 1999, p.1163)

in Kapitel 4.1.2 genauer beschrieben. Teile des Grafiksystems sind darüber hinaus die Ausgabegeräte wie Monitore, Drucker, HMDs, Powerwalls etc. sowie die Eingabegeräte wie Tastatur, Maus, Joysticks, Wands, Tracker. Über das GUI kann der Benutzer (das System Mensch) via Grafiksystem der Software interagieren.

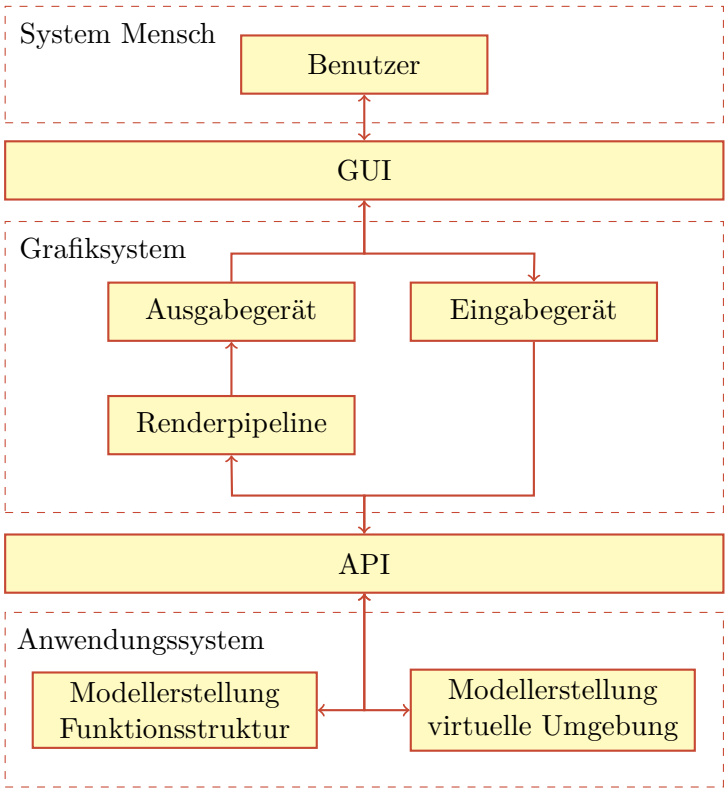


Bild 4.3: Schematische 3D Software Architektur zur Visualisierung

Die Berechnung der 3D Modelle, die in der virtuellen Umgebung angezeigt werden, wird von Grafiksystemen geleistet. Neben den dazu notwendigen Renderverfahren, spielt auch der Aufbau der Datenstruktur eine wichtige Rolle. Dementsprechend müssen für diese Graphen, wirkungsvolle und schnelle mathematische Algorithmen gefunden werden. Der im Kontext dieser Arbeit konzipierte Ablauf zur effizienten Erstellung der 3D Umgebung wird in Bild 4.4 erläutert. Ausgangspunkt ist die Modellierung der 3D Szene. Hier wird auf den Aufbau des 3D Modells sowie den Modellierungsverfahren eingegangen. Im Rahmen dieser Arbeit ist eine effiziente Modellierung vor dem Hintergrund eines Echtzeit Renderings zu sehen. Aus diesem Grund wird im nächsten Schritt das erstellte 3D Modell durch Optimierungsverfah-

ren verbessert. Dazu werden spezielle Techniken wie das effiziente Rendern von Gestaltprimitiva, der optimierte Einsatz von Licht, Material und Textur, sowie Simplifizierungstechniken wie Culling und LoD erläutert. Mittels einer effizenten Datenstruktur des Szenengraphen kann die Performanz der 3D Szene weiter verbessert werden. Abschließend wird der Export in das VR System vorbereitet.

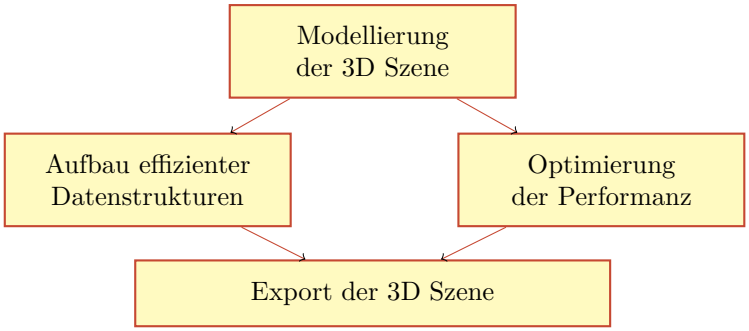


Bild 4.4: Ablauf der Bestimmung von 3D Daten

4.1.1 Modellierung der 3D Szene

Um in einer virtuellen Szene einen realen Gegenstand abzubilden, sollte das grafische Objekt, welches diesen Gegenstand repräsentiert, geometrisch möglichst korrekt sein. Daher sollte die geometrische Struktur der beiden Objekte, des Originals sowie dessen Repräsentation, idealerweise übereinstimmen. Meistens ist dies jedoch nicht möglich, da komplexe oder kurvenreiche Objekte aus der realen Welt zu aufwendig für die Berechnung gegenwärtiger Computer sind (Slater et al., 2001). Da sich 3D Objekte in der Computergrafik aus Punkten[‡] zusammensetzen, die zu Dreiecken verbunden werden, erklärt sich die Problematik der exakten Reproduktion einer Kurve von selbst.

Aus diesem Grund muss die Modellierung der 3D Szene möglichst ressourceschonend aber dennoch geometrisch realistisch aufgebaut sein. Der Ablauf, der im Rahmen dieser Arbeit gewählt wurde, stellt Bild 4.5 dar. Dabei werden zuerst der Aufbau einer 3D Szene erläutert und daraufhin die Erstellung des 3D Modells diskutiert. Sowohl die Grundlagen zum Aufbau als auch

[‡] Punkte werden in der Fachsprache auch Vertices genannt, in der Einzahl Vertex

die Erstellung des Modells werden unter dem Gesichtspunkt der Erstellung einer virtuellen Umgebung betrachtet, bei der Echtzeitanforderungen eine wesentliche Rolle spielen.

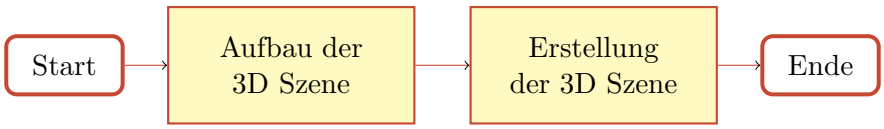


Bild 4.5: Prozess 3D Modellierung

Aufbau von 3D Szenen

Grundlage einer jeden 3D Szene bildet das Koordinatensystem. Dabei hat jedes 3D Modell ein eigenes lokales Koordinatensystem und alle Angaben über dieses Modell beziehen sich auf dieses System. Auf diese Weise können mehrere Modelle des gleichen Typs in der virtuellen Welt koexistieren und bei Transformationen auf ein und dasselbe Grundmodell zurückgreifen. Anhand des über den Achsen aufgespannte Koordinatensystem werden Länge, Breite und Höhe der Szene definiert. In der Regel dient der Ursprung des Koordinatensystems gleichzeitig auch als Referenzpunkt für das 3D Modell mit den Koordinaten $(0, 0, 0)$.

Zur Anzeige auf dem Bildschirm und für die verschiedensten Berechnungen ist es darüberhinaus notwendig, einem Modell einen eindeutigen Platz in der virtuellen Umgebung zuzuordnen. Für diesen Zweck gibt es ein weiteres Koordinatensystem, das Weltkoordinatensystem, in dem sich das 3D Modell nach der Modelltransformation befindet (vgl. Kapitel 3.2.1 auf Seite 83). In diesem werden letztlich die 3D Modelle der Szene zur Gesamtszene angeordnet.

Darüberhinaus beinhaltet jede 3D Szene mindestens eine Lichtquelle, die unterschiedliche Formen oder auch Farben haben kann und eine Kameraposition, die die perspektivische Sicht definiert. Eine vollständige 3D Szene ist in Bild 4.6 auf der nächsten Seite dargestellt.

Erstellung der 3D Szene

Zur Modellierung eines 3D Objektes gibt es verschiedene Modellierungstechniken. Im Bereich der 3D Modellierungsverfahren muss zwischen Splineba-

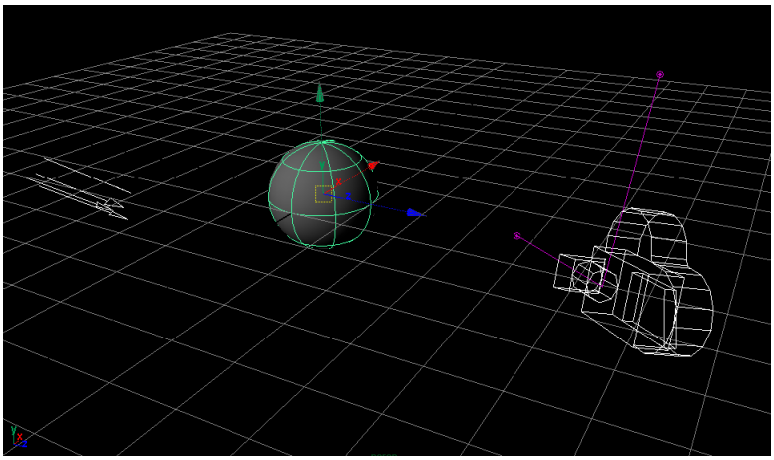


Bild 4.6: Beispiel einer vollständig beschriebenen 3D Szene (Huber, 2002)

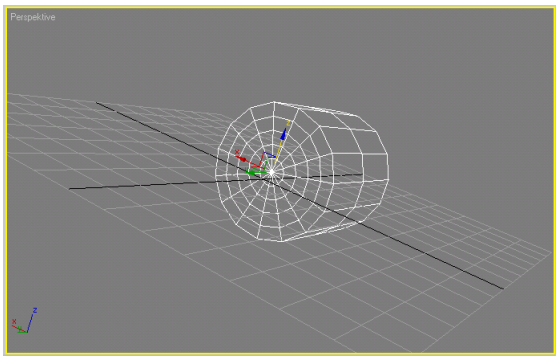
sierten und Polygonbasierten Verfahren unterschieden werden. Wie bereits in Kapitel 2.3 auf Seite 26 beschrieben, besteht der wesentliche Unterschied darin, dass die Beschreibung der 3D Modelle entweder durch Kurven (Splines) oder durch Polygone erfolgt. Im Rahmen dieser Arbeit soll ein Modellierungsverfahren gewählt werden, dass sowohl schnell und intuitiv anwendbar ist, als auch ein professionelles Modellieren zulässt. Darüberhinaus soll es von den gängigen 3D Modellierungswerkzeugen ausreichend unterstützt werden.

Die in Kapitel 2.3.2 auf Seite 28 vorgestellte *Box Modelling* Technik wird als geeignetes Modellierungsverfahren ausgewählt. Dieses basiert auf der Polygonbeschreibung von Oberflächen. Ausgangspunkt bei der Modellierung ist ein Grundkörper, der schrittweise durch Extrusion (Aufteilung einer Quaderseite in mehrere Polygone) und Abschrägung (Extrusion mit sich verjüngenden oder breiter werdenden Polygonen) in die entsprechende Form gebracht wird. Dabei wird auf die Polygone eine automatische Glättung angewandt, die aus den kantigen Polygonübergängen natürliche Kurven macht. (Knebel, 2003)

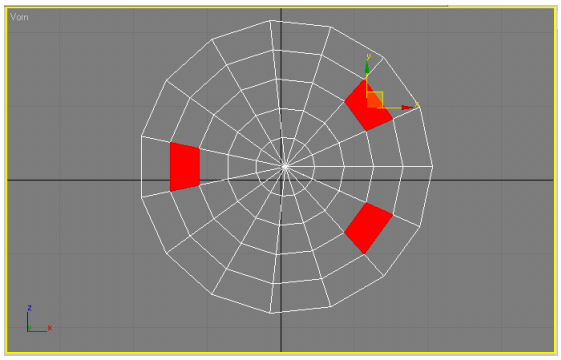
Der Modellierungsvorgang ist von der 3D Modellierungssoftware unabhängig und kann zumindest in ähnlicher Form, auch in anderen Modellierungsprogrammen Verwendung finden. Anhand der Teilkomponente Bipolartransistors des Produktbeispiels *Switch* soll die Vorgehensweise der Methode erläutert werden.

Als Grundform des späteren Modells wird beim Box Modelling Verfahren

als erstes ein Geometrieprimitiv erstellt. Bild 4.7 (a) veranschaulicht den erstellten Grundkörper, dessen Oberfläche bereits mit einem Polygonnetz überzogen wurde.



(a) Erstellen eines Bipolartransistors



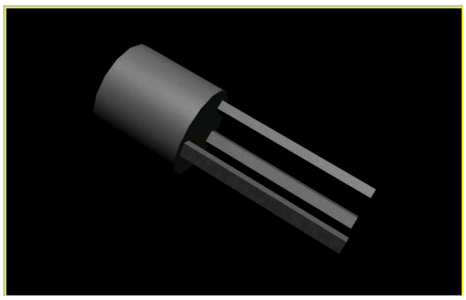
(b) Auswahl der Polygone der Füße

Bild 4.7: Vorgehensweise bei dem Box Modelling Verfahren (a)

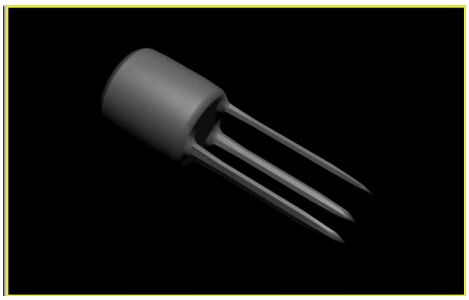
Da anschließend einzelne Polygone des Grundkörpers bearbeitet werden müssen, wird die Drahtgitteransicht als bevorzugte Arbeitsumgebung eingestellt. Somit sind alle Linien und Punkte des Körpers sichtbar. Bei komplexen Körpern muss vorab die Nutzung der Transformationsfeatures bedacht werden, um Modellierungszeit und -aufwand zu sparen. Damit die einzelnen Polygone, Punkte oder Linien des Quaders bearbeitet werden können, muss der Grundkörper in ein bearbeitbares Polygon umgewandelt werden. Zur eigentlichen Modellierung verrichten die Polygonbearbeitungswerkzeuge Extrudieren und Abschrägung einen Großteil der Modellierungsarbeit. Durch Extrudieren eines Polygons kann dieses vom ursprünglichen Grund-

körper weg bewegt werden. Es entstehen an allen Kanten des Polygons neue Polygone, die dieses mit dem Grundkörper verbindet.

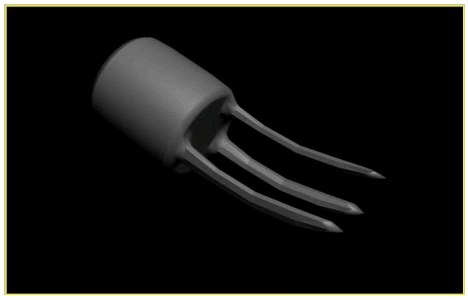
In Bild 4.7 (b) sind drei rot markierte Polygone sichtbar, die im nächsten Schritt extrudiert und geglättet werden. In einer Variante des Bipolartransistors wurden die Füße abgeschrägt. Dabei entspricht das Abschrägen dem Extrudieren, im Zusammenspiel mit einer Verjüngung oder einer Verlängerung der Polygonseiten. Das Ergebnis nach der Glättung ist in Bild 4.8 zu sehen.



(a) Bipolartransistor extrudiert



(b) Bipolartransistor geglättet



(c) Bipolartransistor abgeschrägt

Bild 4.8: Vorgehensweise bei dem Box Modelling Verfahren (b)

Bild 4.8 (b) zeigt deutlich, dass die Kanten des Ausgangspolygons durch die automatische Glättung verschwunden sind. Um diesen Effekt deutlicher zu erkennen, wurde die Ansicht mittels des Beleuchtungsverfahren Gouraud Shading[§] erzeugt.

Auf Basis dieser Modellierungsmethode werden alle im Verlauf der Arbeit erzeugten 3D Objekte der 3D Szene modelliert.

[§] Das Gouraud Shading ist ein Verfahren, um Polygon-Flächen zu schattieren.

4.1.2 Optimierung der Performanz

Die in einer virtuellen Umgebung dargestellten Szenen müssen schnell zu berechnen sein, und gleichzeitig möglichst realistisch aussehen, um die Illusion des Betrachters zu festigen. Um dem gerecht zu werden, wurde in der Vergangenheit der Fokus der Forschung und Entwicklung auf das Merkmal der Imagination (vgl. Kapitel 2.6.1 auf Seite 55), sprich der Erzeugung von fotorealistischen Grafiken in Echtzeit gelegt. Trotz des Aufwandes ist es dennoch nicht möglich die Komplexität der Realtiät in einer virtuellen Umgebung vollständig abzubilden. Daher werden spezielle Techniken benötigt, die in der Lage sind, die gewünschten Grafiken in Echtzeit zu erzeugen. Hierbei muss immer die Balance zwischen Detailierungsgrad der virtuellen Welt und der Bildwiederholrate erhalten werden. Viele Techniken benutzen vereinfachte physikalische oder chemische Modelle, andere wiederum entbehren jeglicher realistischen Grundlage. Die Verknüpfung dieser Techniken wird im Rahmen dieser Arbeit verfolgt. Bild 4.9 zeigt den zu begehenden Weg.

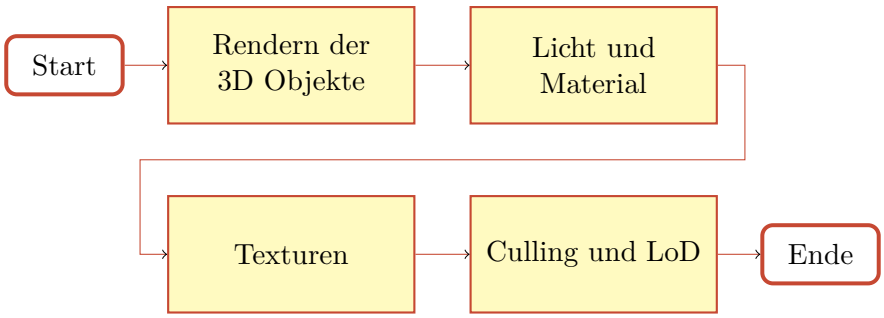


Bild 4.9: Ablauf der Optimierung der Geometrie

Rendern der 3D Objekte

Eine im Computer erzeugte, virtuelle Umgebung setzt sich aus einer Vielzahl von grafischen Primitiven, wie Punkte, Linien, Polygone zusammen. Diese wurden in der ISO 19107 zusammengefasst. Die Verarbeitung dieser Primitiven effizient zu implementieren, ist Ziel der Optimierung. Um dies zu erreichen, bedarf es einiger Grundkenntnisse der aktuellen Hardware, da der Aufbau der Hardware die Herangehensweise zur Effizienssteigerung bestimmt. Bild 4.10 auf der nächsten Seite zeigt den statischen Aufbau und

die Verknüpfung der Elemente, die an der Rendering Pipeline beteiligt sind.

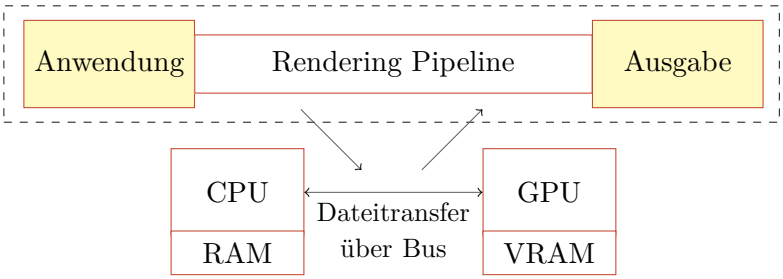


Bild 4.10: Schematische Darstellung der Rendering Pipeline

Lag in der Vergangenheit das Hauptproblem der 3D Visualisierung in der begrenzten 3D Hardware, die Rendern von Pixeln[¶] und das Ausführen von Transformationen stark einschränkte, sind die gegenwärtigen Grafikkarten fähig, mehr als 140 Millionen Vertices pro Sekunde zu transformieren und zu rendern. Um die Effizienz beim Rendern zu steigern, müssen dementsprechend die Engpässe in der Kommunikation zwischen den Hardwarekomponenten optimiert werden. Dies setzt voraus, dass die effiziente Kommunikation innerhalb der Hardwareebenen CPU und RAM sowie GPU und VRAM auch auf die Kommunikation zwischen diesen ausgeweitet wird. Denn im Gegensatz zu den Transferraten innerhalb der beiden Ebenen läuft der Transfer über den Bus nur sehr langsam ab und genau hier liegt Optimierungspotential.

Vor allem Texturen, Vertex- und Indexdaten beanspruchen den Transferbus sehr. Um den Transfer effizienter zu gestalten, sollten die Polygone nach der verwendeten Textur sortiert werden. Texturen sind dabei immer quadratisch anzulegen und die Größe sollte immer in 2^n Potenzen festgelegt werden. Dieses Verfahren wird von den handelsüblichen Grafikkarten bis maximal 2^{12} Pixel unterstützt. Um den VRAM nicht zu verschwenden, sollte die Texturgröße auf $2^6 * 2^6$ Pixel eingestellt werden, da die wenigsten Texturen in größere Raster passen. Die Texturgröße sollte auch nicht zu klein gewählt werden, da sonst zu viele Dreiecke entstehen (vgl. Kapitel 4.1.2 auf Seite 107).

Darüberhinaus sollen Vertex- und Indexdaten gegebenenfalls redundant im RAM und im VRAM gehalten werden. Wie dieses Problem genau gelöst

[¶] Pixel, im Deutschen als Bildpunkt bezeichnet, ist die kleinste Einheit, die ein Bildschirm oder ein Drucker darstellen kann.

wird, ist Aufgabe einer 3D Render Engine (vgl. Kapitel 2.6.3 auf Seite 64). Das grobe Management übernimmt meistens schon die verwendete API, wie in Bild 4.3 auf Seite 97 veranschaulicht. So entscheidet beispielsweise Direct3D wann es welche Texturen in den VRAM lädt und wann sie wieder zurück in den RAM geschrieben werden (Bücker, 2003).

Licht und Material

Licht ist ein wichtiger Bestandteil einer virtuellen Umgebung, jedoch sehr komplex in seiner Struktur, so dass beim Rendering vereinfachte Lichtmodelle benutzt werden. Diese vereinfachten Modelle sind trotzallem sehr umfangreich, so dass die Benutzung von Lichtquellen vorab überlegt sein muss. Werden außerhalb der Echtzeitdarstellung Verfahren wie Ray-Tracing oder Radiosity eingesetzt, die für jeden Punkt genau berechnen können, wie viel Licht und wie viel Schatten auf diesen fallen, und darüber hinaus sogar die Lichtreflexion mit einbeziehen, ist dies in der virtuellen Realität nicht möglich, da der hohe Realitätsgrad weit über das Vermögen der gegenwärtigen Echtzeit 3D Render Engines hinaus geht.

Dementsprechend werden für die Echtzeitdarstellung andere Verfahren eingesetzt. So wird ein bestimmtes Material, in dem chemische und physikalische Materialbeschaffenheit stark vereinfacht werden, für die 3D Objekte definiert und direktes oder ambientes Licht statisch festgelegt. Die Farbe eines Pixels errechnet sich dann aus der Farbe der Vertices, aus dem Material der Polygone, aus allen Lichtquellen sowie der Textur des Objektes. Reflektierendes Licht wird entsprechend dem verwendeten Material betrachtet. Handelt es sich um eine raue Oberfläche wird das diffuse Reflektionsmodell zur Berechnung verwendet, bei einer glatten Oberfläche dagegen kommt das spekuläre Reflektionsmodell zum tragen (Jenny, 2007). Die Zusammenhänge von Material und Lichtquelle verdeutlich Bild 4.11 auf der nächsten Seite. Auf Schatteneffekte soll dabei vorerst gänzlich verzichtet werden.

Wie komplex die Berechnung des Lichts ist, soll die folgende Formel verdeutlichen, welche lediglich für die Berechnung einer Lichtquelle verantwortlich ist.

$$i_{tot} = a_{glob} \cdot \; m_{amb} \; + \; m_{emi} \; + \; cSpot \, (i_{amb} \; + \; d \, (i_{diff} \; + \; i_{spec}))$$

Die Berechnungskomplexität für n Lichtquellen lautet

$$i_{tot} = a_{glob} \cdot \; m_{amb} + m_{emi} + \sum_{k=1}^{n} cSpot^{k} \; \left(i_{amb}^{k} + d^{k} \left(i_{diff}^{k} + i_{spec}^{k} \right) \right)$$

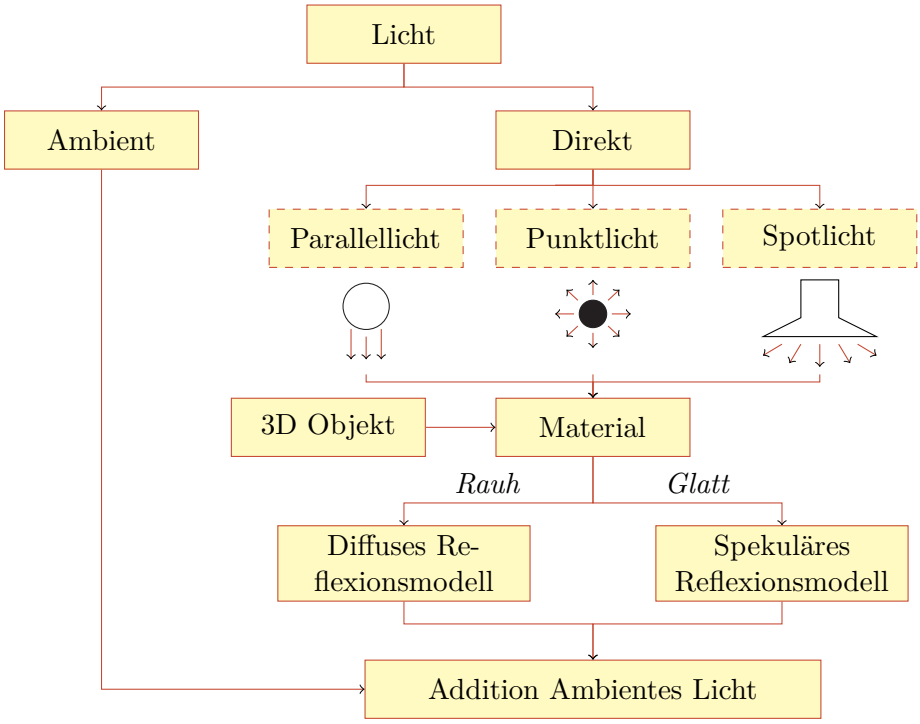


Bild 4.11: Übersicht über die Lichtquellen beim Rendering (Zerbst, 2000)

Die Gesamtlichtintensität berechnet sich dabei aus der Interaktion zwischen Lichtquellen, Objektgeometrie und deren Materialien (m_{amb}, m_{diff}, m_{spec}, m_{emi}) zuzüglich einem globalen Beleuchtungsfaktors a_{glob}.

Die für den Lichtintensitätsabfall benötigte Formel

$$d = \frac{1}{s_c} + s_l \mid s_{pos} - p| \; + s_q \, |s_{pos} \; - \; p|^2$$

veranschaulicht den Zusammenhang zwischen Lichtintensität und Entfernung der Lichtquelle. Dabei bezeichnet s_c die konstante Abnahme, s_l die lineare Abnahme und s_q die quadratische Abnahme der Lichtintensität. $|s_{pos} - p|$ beschreibt die Entfernung der Lichtquelle zu dem betrachteten Punkt p.

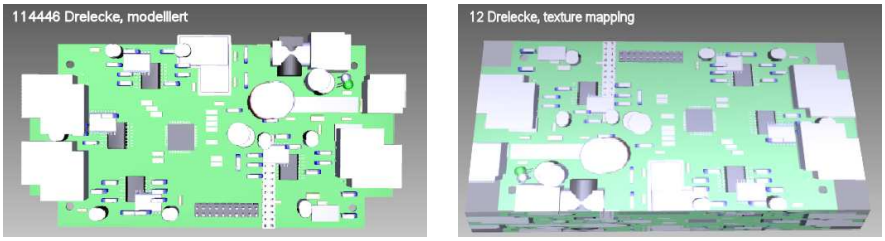
Um die Geschwindigkeit zu steigern, kann für statische Objekte und Lichtquellen die Beleuchtung bereits im Voraus berechnet und in einer so genannten LightMap[‖] gespeichert werden. Die Werte aus dieser Lightmap können

[‖] Eine Lightmap ist eine Bitmap, in der Licht- und Schatteninformationen für Source-Maps gespeichert sind. Diese werden durch den 3D Renderer additiv mit den Texturinformationen verrechnet.

genutzt werden, um einen Pixel aufzuhellen oder abzudunkeln, ohne weitere aufwendige Berechnungen.

Texturen

Texturen erlauben es einen höheren Detailgrad einer 3D Szene vorzutäuschen, ohne weitere grafische Primitve zu verwenden. Dazu projizieren sie ein Bild, das die gewünschten Details beschreibt auf die Oberfläche eines einfachen Polygons. Das *Texture Mapping* repliziert dabei mosaikartig Texturen und benötigt keine zusätzlichen Polygone. Ein Beispiel für das Texture Mapping zeigt Bild 4.12.



(a) Modellierte Platine (b) Texture Mapping

Bild 4.12: Platine des Switches

Der Vorteil dieser Technik ist jedoch auch dessen Nachteil, denn durch die nicht vorhandene geometrische Information kann keine korrekte Beleuchtung für das Objekt berechnet werden. Verfahren wie das “Bump Mapping” versuchen diesen Nachteil zu umgehen.

Bei Echtzeitanwendungen ist es wichtig, dass das Arbeiten mit Texturen effizient abläuft. Neben der in Kapitel 4.1.2 auf Seite 103 erwähnten wichtigen Sortierung der Polygone nach Texturen und der damit verbundenen Entlastung des Transferbusses existieren noch weitere Techniken, um die Effizienz zu steigern, wie beispielsweise die Komprimierung von Texturen. Mittels dieser Technik kann sowohl Speicherverbrauch als auch Bustransferrate minimiert werden. Für diese Möglichkeit stehen etliche Formate, wie 3Dc, DXTC oder S3TC zur Verfügung, die speziell für dieses Anwendungsgebiet erstellt wurden.

3D Objekte einer virtuellen Umgebung haben nicht immer die gleiche Größe. Je nach Position der Kamera können sie sehr klein, aber auch sehr groß sein. Wird unabhängig von der Größe eines Objektes immer die gleiche Textur

verwendet, muss die GPU diese zur Laufzeit an das Objekt anpassen. Das bedeutet, das Rechenleistung zur ständigen Skalierung der Textur verwendet werden muss. Das MipMapping** ist ein Verfahren, welches hier ansetzt. Es stellt dem System verschiedene Größen einer Textur zur Verfügung und wählt je nach Skalierung des Objektes im aktuellen Frame die entsprechende Textur aus.

Darüber hinaus bieten aktuelle 3D Schnittstellen die Möglichkeit an, das Verwalten von Texturen zu übernehmen. Sie übernehmen damit die Entscheidung, wann sich eine Textur im RAM und wann sie sich im VRAM befinden soll. Dennoch bleibt beispielsweise für sehr zeitkritische und spezielle Anwendungen die Notwendigkeit offen, das Verwalten der Texturen manuell zu steuern.

Culling und LoD

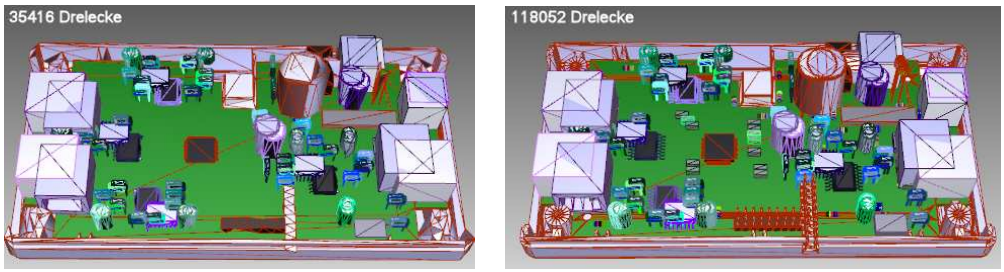
Das Ziel von Culling Algorithmen ist es, die Menge der am Ende wirklich sichtbaren grafischen Primitiven zu filtern, und nur diese weiter durch die Render Pipeline zu geben. Diese Art des Cullings wird als Visibility Culling bezeichnet. Daneben können durch weitere Culling Techniken, wie das Backface Culling, das View Frustrum Culling oder Occulsion Culling 3D Objekte mittels spezieller Algorithmen zusätzlich gefiltert werden.

Mittels des Detail Cullings kann die Polygonzahl eines Objektes unter bestimmten Bedingungen minimiert werden. Eine spezielle Form des Detail Culling ist der Level of Detail. Bei diesem werden verschiedene Detailstufen eines Modells entweder programmiertechnisch oder direkt in der Modellierungssoftware erstellt. Das 3D System entscheidet, welches dieser Modelle zum Rendern verwendet wird. Oft wird allein die Entfernung eines Objektes zur Kamera als Auswahlkriterium für eine bestimmte Detailstufe des Modells verwendet. Ein wesentliches Problem von der LOD Technik ist das Popping. Dieses wird durch den Übergang zwischen einer Detailstufe zur nächsten erzeugt, wenn die Unterschiede in den Detailstufen zu gravierend sind. Mittels verschiedener Algorithmen soll das Popping umgangen werden.

Im Rahmen dieser Arbiet wurde auf den C-LOD Algorithmus (Continuous Level of Detail) zurückgegriffen (Roettger et al., 1998, p.316f). Dieser hat die Besonderheit, dass die darzustellende Geometrie nicht in mehreren diskreten

** MIP ist die Abkürzung für multum in parvo, was soviel wie “viel auf kleinem Platz” bedeutet. Eine MIP-Map ist eine Folge von Rasterbildern des selben Motivs, jedoch mit abnehmender Auflösung.

Detailstufen gespeichert wird. Stattdessen wird sie kontinuierlich angepasst, wobei in einem Objekt fließende Übergänge zwischen mehreren Detailstufen existieren. Objekte werden dabei während der Laufzeit simplifiziert. Dazu werden benachbarte Vertices kombiniert und das Objekt somit sukzessive vereinfacht. Wird der Weg der Simplifizierung gespeichert, ist dieser Prozess, auch wieder umkehrbar. Bild 4.13 zeigt den Einsatz von LOD am Beispiel des Switch.



(a) Switch mit 35416 Polygone (b) Switch mit 118052 Polygone

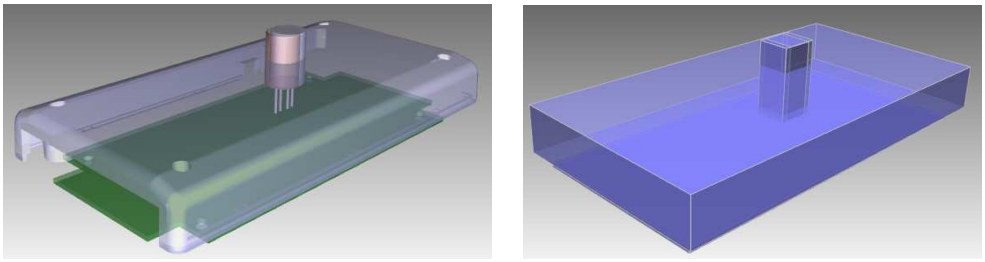
Bild 4.13: Detail Culling (LOD) des Switches

4.1.3 Aufbau effizienter Datenstrukturen

Grundlage virtueller Umgebungen sind die geometrischen Daten. Da die Geschwindigkeit der Visualisierung den wichtigsten Aspekt darstellt, ist die Verwendung effizienter Datenstrukturen unabdingbar. Häufig werden hierarchische top-down Datenstrukturen verwendet, so dass eine Ebene alle Elemente der darunter liegenden Ebene beinhaltet. Auf diese Weise lässt sich schnell eine große Menge nicht sichtbarer Daten aussortieren. Die zwei wesentlichen Ansätze werden durch die Bounding Volume Hierarchie, die die Aufteilung nach geometrischen Objekten vornimmt und das Spatial Partitioning, bei der eine Unterteilung des Raumes vorgenommen wird, repräsentiert.

Im Rahmen dieser Arbeit wird das Konzept der Bounding Volume Hierarchie verwendet. Diese ist von der Datenstruktur her ein Baum, so dass die Wurzel des Baumes die komplette Szene beinhaltet kann. Sinnvoll ist dies beispielsweie beim View Frustrum Culling (vgl. Kapitel 4.1.2). Dabei wird eine Datenstruktur erstellt und mittels einfachen Schnittpunktberechnungen analysiert, ob ein Knoten im View Frustrum liegt oder nicht. Ist dies

nicht der Fall, können diese Knoten mit allen Subknoten aus der Berechnung herausgenommem werden, da alle Kinder Teil des Elternknotens sind. Auf diese Weise lassen sich sehr oft mehrere Tausend Vertices mit nur ein oder zwei Berechnungen sparen. Dieses Verfahren wird auch beim Ray-Tracing oder bei Kollisionserkennung zwischen Objekten angewandt. Bild 4.14 zeigt die Kollision von Bipolartransistor und Gehäuseabdeckung des Switch.



(a) Kollision Transistor/Abdeckung (b) Bounding Boxes

Bild 4.14: Kollisionserkennung anhand der BVH Methode

Im Pseudocode sieht die oben gezeigte Kollisionserkennung wie folgt aus:

```
Bool Kollision (Bipolartransisitor BPT, Abdeckung A)
{if (!TestKollisionBV(BPT.BV, A.BV))
          return false;
  if (!BPT.Leaf && !A.Leaf)
     {  if (Volume(BPT) > Volume(A))
          {  foreach (Objekt child in BPT.Children)
                    Kollision(child, A);
          }
        else
          {  foreach (Objekt child in BPT.Children)
                    Kollision(child, BPT);
          }
     }
  else if (BPT.Leaf && !A.Leaf)
     {  foreach (Objekt child in BPT.Children)
              Kollision(child, BPT);
     }
  else if (!BPT.Leaf && A.Leaf)
     {  foreach (Objekt child in BPT.Children)
```

```
                Kollision(child, A);
      }
   else if (BPT.Leaf && A.Leaf)
      {  return DreiecksTest(BPT.Triangle, A.Triangle)
      }
   return false;
}
```

Um die Szenen effizienter zu rendern und die Berechnungen der Kollisionsabfragen zu beschleunigen, wird im Rahmen dieser Arbeit ein Szenengraph verwendet. Die BVH wird zusammen mit dem Szenengraphen mitgeführt. Jedem Knoten ist also zusätzlich ein Bounding-Volumen (Bounding Box) zugeordnet, das die räumliche Ausdehnung des Knotens samt Kindknoten anzeigt.

Der Szenengraph ist eine objektorientierte Datenstruktur durch welche der Aufbau einer zwei- oder dreidimensionalen Szene dargestellt wird (vgl. Kapitel 2.6.3 auf Seite 65 sowie Kapitel 3.1.2 auf Seite 82). Die Erstellung des Szenengraph ist mit entscheidend für die Geschwindigkeit der grafischen Visualsierung, denn je nachdem wie die Anzeigesoftware den Szenengraph traversiert und entlang eines jeden Pfades akkumuliert, verändert sich die Performanz des Sytems. Sollen im Szenengraph beispielsweise Transformationen ausgeführt werden, so spielt deren Position eine entscheidende Rolle (Knebel, 2003). Die Veränderung eines Geometrieknotens durch eine Rotation um einen definierten Winkel, bedeutet, dass der entsprechende Transformationsknoten links von dem Geometrieknoten im Szenengraphen angeordnet sein muss. Denn beim Traversieren wird zuerst die Rotationsmatrix aus dem Transformationsknoten ausgelesen. Die berechnete Rotation wird anschließend auf die im Knoten gespeicherte Geometrie angewandt.

Allgemein repräsentiert der Wurzelknoten die Gesamtszene und die Kinderknoten jeweils nur einen Ausschnitt der Szene. Die Blattknoten stellen dann ein geometrisches Objekt dar. Als Beispiel wird der Szenengraph eines Bipolartransistor erzeugt, der Teil des Produktbeispiels *Switch* ist. Dieser besteht aus dem Gehäuse, dem Collector, der Basis und dem Emitter. Damit wird ein Baum generiert, der einen Wurzelknoten und vier Kindknoten besitzt. Der Wurzelknoten stellt den Bipolartransistor dar. Jeder der Kindknoten repräsentiert jedoch nur noch eine Komponente. Bild 4.15 zeigt den Szenengraph des Bipolartransistors.

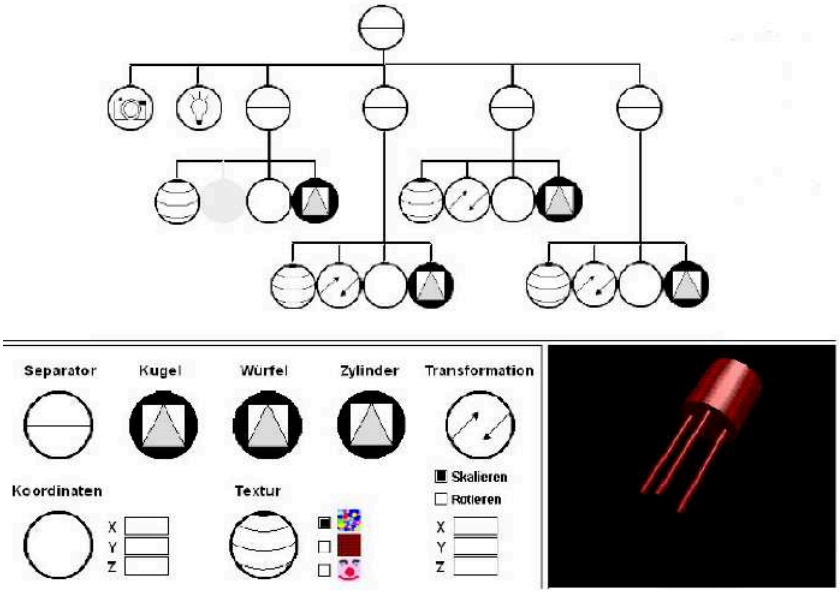


Bild 4.15: Beispielszenengraph eines Bipolartransistors

Szenengraphen sind so genannte Middleware, welche auf den Low-Level APIs aufbauen, um unter anderem eine räumliche Organisation der vorliegenden Daten zur Verfügung zu stellen, die normalerweise von einem 3D System benötigt wird.

4.1.4 Export der 3D Szene

Der Export von 3D Daten aus Modellierungsprogrammen in ein VR Datenformat ist nicht immer ohne weiteres möglich. Neben einer Reihe von freien Konvertierungsprogrammen wie beispielsweise des offenen ISO/IEC 14772 Standards, ist ein Großteil der gängigen 3D und VR Formate proprietär, d.h. diese können nur von den jeweiligen Anwendungen gelesen werden. Es gibt aber auch Datenformate, die von mehreren Programmen verarbeitet werden können und sich somit für den Datenaustausch eignen. Jedoch gilt bei diesen Formaten oft, dass sie - im Gegensatz zu den proprietären Formaten - nur einen Teil der Informationen transportieren können. So kann das weit verbreitete Wavefront OBJ Format, beispielsweise nur Geometrieinformationen transportieren, während das BioVision BVH Format nur skelettbasierte Animationsdaten speichern kann.

Der charakteristische Ablauf der Produktentwicklung erfordert die Verwendung von unterschiedlichen Expertensystemen. Dabei spielt das Datenüber-

tragen seit jeher eine wichtige Rolle. Die im Rahmen dieser Arbeit betrachteten 3D Daten stammen aus den typischen Vertretern gängiger CAD und Visualisierungssysteme. Bild 4.16 zeigt diese sowie die verwendeten Datenaustauschformate. Bei der Auswahl der Datenformate ist zu beachten, dass sie innerhalb eines Bereichs durch Systeme anderer Anbieter ersetzt werden können. Die Systeme der in Bild 4.16 dargestellten Prozesskette sind dementsprechend austauschbar. So kann anstelle von ProEngineer auch CATIA oder UG NX verwendet werden. Die Systeme sind alle in der Lage, mit den abgebildeten Datenschnittstellen zu arbeiten.

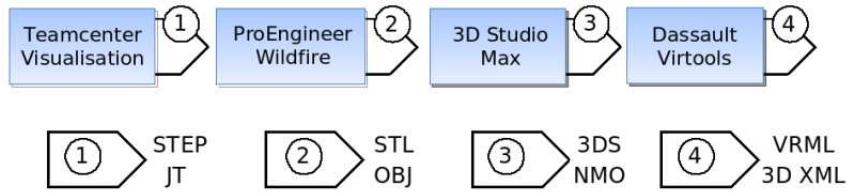


Bild 4.16: Exportprozess der 3D Daten

Die in Betracht gezogenen Datenaustauschformate zeichnen sich insbesondere durch ihre Versatilität, ihre Funktionsvielfalt und ihrem weitverbreiteten Einsatzspektrum aus. Die im Kontext der Arbeit zu nutzenden Formate sollen hier kurz dargestellt werden.

- Das STEP Datenaustauschformat (ProSTEP/iViP) ist ein Standard (ISO 10303) zur Beschreibung von Produktdaten. Aufgrund der Standardisierung eignet sich STEP besonders gut für den Datenaustausch zwischen verschiedenen Systemen. Dabei umfasst es neben den physischen auch funktionale Aspekte eines Produktes.

- Die STL Schnittstelle (Burns, 1989) beschreibt eine (Quasi-) Standardschnittstelle vieler CAD Systeme. Es dient der Bereitstellung geometrischer Informationen aus dreidimensionalen Datenmodellen und beruht dabei auf der Beschreibung der Oberfläche von 3D Körpern mit Hilfe von Dreiecksfacetten.

- Das 3DS Format (FileFormat) ist sehr komplex und kann neben der Geometrie von Objekten auch Transformationen und hierarchische Beziehungen der Objekte speichern. Die Komplexität von 3DS ist ein

Grund dafür, dass der Datenaustausch nicht immer problemlos verläuft. Typische Defekte sind falsch rekonstruierte Hierarchien, sowie gekürzte Texturnamen, da viele Programme die Dateinamen der Texturen auf MS-DOS Konventionen zurechtstutzen.

- Das NMO Dateiformat ist ein von Virtools entwickeltes proprietäres Format, das Objekte oder Szenen enthält. Diese sind in der Regel in anderen 3D Systemen entstanden.

- Das 3D XML Datenaustauschformat (Dassault Systèmes) beschreibt Informationen von der grafischen Darstellung bis hin zu technischen Daten inklusive der Produktstruktur, Geometrie und Applikationen auf Basis von XML. Diese von Dassault Systèmes propagierte Schnittstelle soll über den gesamten Produktlebenslauf hinweg nutzbar sein.

- VRML ist eine Beschreibungssprache für 3D Szenen, deren Geometrien, Ausleuchtungen, Animationen und Interaktionsmöglichkeiten. Ursprünglich als 3D-Standard für das Internet entwickelt, wird das Format auch von den meisten 3D-Modellierungsprogrammen unterstützt.

Darüber hinaus bieten die meisten Softwarehersteller so genannte Exportplugins für die gängigsten Modellierungsprogramme an. Bild 4.17 auf der nächsten Seite zeigt die grafische Oberfläche am Beispiel des Virtools Exportplugin von 3D Studio Max. Obwohl Dassault Systèmes den Export von 3DS oder 3D XML Dateien in das proprietäre Datenformat von Virtools ermöglicht, sind zu dessen Nutzung entweder spezielle Registrierungsprozesse oder die Entgeltung der Software notwendig. Ähnliches gilt auch für die Dateiformate von EON Reality oder Quest3D (vgl. Tabelle C.1 auf Seite 289). Das freie Virtools Exportplugin ermöglicht die zweckmäßige Übertragung von Elementen wie Materialien, Texturen und Animationen von 3D Studio Max nach Virtools verlustfrei. Neben den 3D Objekten lassen sich auch Animationen, die bereits in dem 3D Modellierungsprogramm erstellt wurden, oder auch Charaktere übertragen werden. Hierzu existieren eine Fülle von Einstellmöglichkeiten, die insbesondere die Optimierung der Geometrie und die Verbesserung der Performanz nach den bereits vorgestellten Konzepten ermöglichen.

Zudem ist es notwendig jedem 3D Objekt eine eindeutige Objektbezeichnung zuzuweisen. Die kohärente Benennung gleicher Objekte in den un-

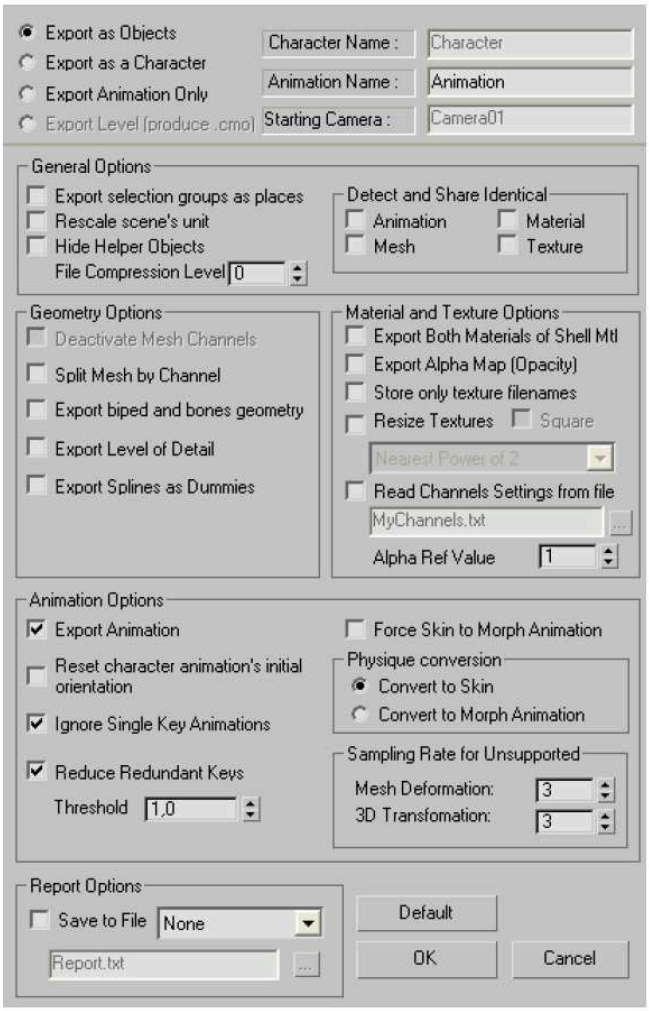


Bild 4.17: Frontend des Virtools Exporter für 3DS Max

terschiedlichen Produktentwicklungsphasen ist notwendige Grundlage zur erfolgreichen Abbildung der Daten über die Phasengrenzen hinweg.

4.2 Erstellung und Mapping des Funktionsmodells

Funktionsmodelle finden im industriellen Umfeld zunehmend mehr Beachtung. Jedoch ist das Verständnis des Begriffs Funktionsmodell nicht eindeutig. Obwohl der Begriff der Funktion aus konstruktionsmethodischer Sicht definiert wurde (vgl. Kapitel 2.2 auf Seite 16), erlauben die naheliegenden Begriffe keine eindeutige Definition. Dem entsprechend verschmelzen die Begriffe Produktfunktionsstruktur, Funktionsstruktur, Funktionshierarchie oder Funktionsflussstruktur miteinander, so dass keine klare Trennung und kein kohärentes Verständnis möglich ist. Ähnliches gilt für Substantiv-Verb-Strukturen, Produktaufgabenstrukturen, verbalen Konstrukten und so weiter.

Im Rahmen der vorliegenden Arbeit finden die Begriffe der Funktionsstruktur und der Funktionshierarchie Verwendung und sollen deshalb eindeutig voneinander getrennt werden. Die Funktionsstrukturen wurden in Kapitel 2.2.2 auf Seite 18 erläutert und werden im Kontext dieser Arbeit als flussorientierte Strukturen verstanden, die von allgemeine in spezielle Funktionsdarstellungen überführt werden können. Unter Funktionshierarchien werden, ähnlich zu den Produktaufgabenstrukturen oder den Wertanalysemethoden (vgl. Kapitel 2.2 auf Seite 16), hierarchisch gegliederte Graphen verstanden, die im Unterschied zu den Funktionsstrukturen mittels Substantiv-Verb Form wie z.B. Werkstück greifen, Tür öffnen etc. Produktfunktionen beschreiben. Diese können wie bei Roth auf den Aufgabensätzen basieren (vgl. Kapitel 2.2.1 auf Seite 17), oder aber funktional beschriebene Aktivitäten eines komplexen Systems darstellen. Funktionen einer Hierarchieebene werden durch Zerlegung dieser an die nächsthöhere Hierachieebene weitergegeben. Durch diese Hierarchisierung lassen sich komplexe Produkte von einer Gesamtfunktion bis zu Elementarfunktionen zurückführen. Unter dem Begriff des Funktionsmodells soll die Verbindung von Funktionshierarchie und Funktionsstruktur verstanden werden. Tabelle 3.2 auf Seite 92 fasst die Anforderungen an die Funktionsmodellierung präzise zusammen.

Durch die klare Trennung von Funktionsstruktur und Funktionshierarchie wird das Verständnis der funktionalen Vorgänge sowie der funktionalen Verknüpfungen der Komponenten innerhalb des Produktes erhöht. Zur Darstellung der hierarchischen Zusammenhänge der Produktfunktionen sind die Funktionshierarchien besser geeignet. Sie sind nahe dem menschlichen Den-

ken, lassen sich schnell abstrahieren und konkretisieren und sind durch den hierarchischen Aufbau klassifizierbar (Krappe and Marinov, 2007, p.26). Die Funktionsstruktur hingegen vermittelt die konkreten physikalischen Ein- und Ausgangsgrößen eines jeden Funktionsobjektes, sowie den Zusammenhang der Funktionsobjekte mit den eigentlichen Funktionen.

Da das Ziel dieser Arbeit in der interaktiven und immersiven Nutzung von Funktionsmodellen in virtuellen Umgebungen liegt, ist es notwendig, die Funktionsmodelle respektive Funktionshierarchien und Funktionsstrukturen derart zu erzeugen, dass die inhärenten Informationen in einer virtuellen Umgebung nutzbar gemacht werden können. Die bestehenden Ansätze zur Funktionsmodellierung wurden in Kapitel 2.2 auf Seite 16 ausführlich diskutiert. Dabei wurde unter anderem festgestellt, dass eine ingenieursgerechte Vorgehensweise zur Funktionsmodellerstellung mittels Softwaretools, in Ermangelung dieser, nicht ausreichend unterstützt wird. Einen Überblick über die notwendigen Schritte zur Erstellung und Abbildung des Funktionsmodells in einer virtuellen Umgebung zeigt Bild 4.18.

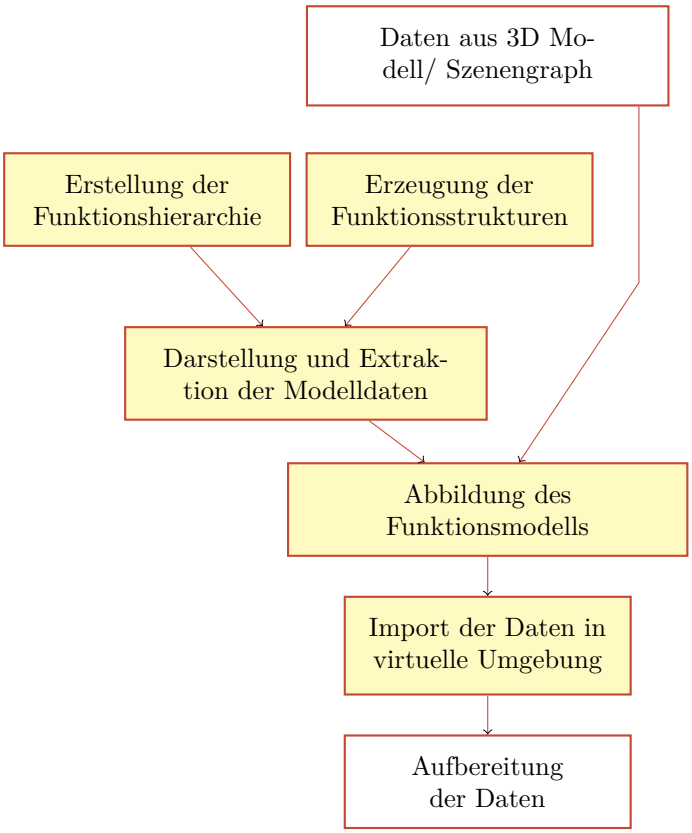


Bild 4.18: Prozeß der semantischen Abbildung von Funktionsmodellen

Dabei beschreiben die gelb gefüllten Rechtecke, die in diesem Kapitel relevanten Schritte, während die weiss gefüllten Rechtecke, Prozesse, die in anderen Kapiteln diskutiert wurden, darstellen. Die erzeugten Daten aus dem Szenengraph, wie die 3D Objekte, deren geometrische oder topologische Ausprägungen etc. wurden in Kapitel 4.1 auf Seite 96 erläutert. Die Erstellung der Interaktionsmetaphern, sowie die Visualisierung der Funktionsmodelle in der virtuellen Umgebung wird in Kapitel 4.3 diskutiert.

Die Abbildung der Funktionsmodelle bedarf im ersten Schrtt einer genormten Erstellung der Funktionshierarchie. Dazu sollen die Produktfunktionen in einer Substantiv-Verb Form benannt und als Graph hierarchisch angeordnet werden. Darauffolgend wird die Funktionsstruktur erstellt. Dies geschieht basierend auf einer Funktionsobjekt-Funktionsverb Beziehung. Dabei sollen die Funktionsobjekte und Funktionverben mittels klassifizierten Bezeichnungen versehen werden. Den Funktionsobjekten sollen ebenso die physikalisch korrespondierenden Ein- und Ausgangsgrößen, die Führungsgrößen, zugewiesen werden. Ist das Funktionsmodell vollständig erarbeitet, folgt die Extraktion der relevanten Daten aus dem Funktionsmodell. Dabei werden die Modelldaten aus den 2D Graphen in eine neutrale Form umgewandelt (normalisiert), um weitere Verarbeitungen homogen und unabhängig von dem verwendeten Modellierungswerkzeug vornehmen zu können. Da die Funktionsmodellierung und deren Visualisierung in verschiedenen Systemen stattfinden und verschiedener Interpretationen der Daten unterliegen, ist eine Abbildung von 2D Graphen auf die bestehende 3D Szenengraphen notwendig. Dies wird in dem Schritt Abbildung des Funktionsmodells auf den Szenengraph erläutert. Abschließend wird der Import des Funktionsmodells in die VR Entwicklungsumgebung diskutiert. Hier werden die extrahierten Informationen aus dem Funktionsmodell, mittels der vorher durchgeführten Abbildung, in die VR Entwicklungsumgebung überführt, so dass die in der VR Entwicklungsumgebung spezifizierten Interaktionen auf diese Daten zurückgreifen können. Anhand des Beispielproduktes *Switch* sollen die Schritte verdeutlicht werden.

4.2.1 Erstellung der Funktionshierarchie

Wichtige Aspekte für die rechnerunterstützte Darstellung der Funktionsmodelle sind die Klassifizierung und Strukturierung, Hierarchisierung sowie die Modularisierung der Funktionsmodelle (Schmidt, 2002). Diese sollen im Rahmen einer semantischen Abbildung bei der Funktionsmodellierung und

den dafür verwendeten Abbildungsvorschriften berücksichtigt werden. Um darüber hinaus die Semantik eines Funktionsmodells rechnerunterstützt extrahieren und darstellen zu können, müssen die in den Funktionsmodellen vorkommenden Bestandteile und deren Gestaltung analysiert werden. Im Rahmen dieser Arbeit werden Funktionsmodelle in Funktionshierarchien und Funktionsstrukturen unterteilt (vgl. Bild 4.18 auf Seite 117). Die Funktionshierarchien werden hierarchisiert, die Funktionsstrukturen klassifiziert, strukturiert und modularisiert (vgl. Kapitel 4.2.2 auf Seite 126). Der erste Schritt im Prozess der rechnerunterstützen Funktionsmodellerstellung besteht in der Erzeugung der Funktionshierarchie.

Die Funktionshierarchie dient der Darstellung der hierarchischen Zusammenhänge der Produktfunktionen. Insbesondere durch die Substantiv-Verb Form lässt sich die Funktionshierarchie schnell abstrahieren und konkretisieren. Dadurch wird eine Skalierbarkeit des Funktionsmodells ermöglicht. Durch die Definition einer Funktionshierarchie wird eine übersichtliche Visualisierung des Funktionsmodells in der virtuellen Umgebung entwickelt, und die Abbildung von Informationen für eine hierarchisch eingerichtete Darstellung ermöglicht. Dies wird anhand des hierarchischen Aufbaus erzielt, der eine eindeutige Klassifizierung darstellt, durch welche eine Überführung in die virtuelle Umgebung, wie sie in Kapitel 4.3.1 beschrieben wird, möglich wird.

Um die Funktionshierarchie bestimmen zu können, sind wie in Bild 4.19 ersichtlich, mehrere Prozessschritte notwendig.

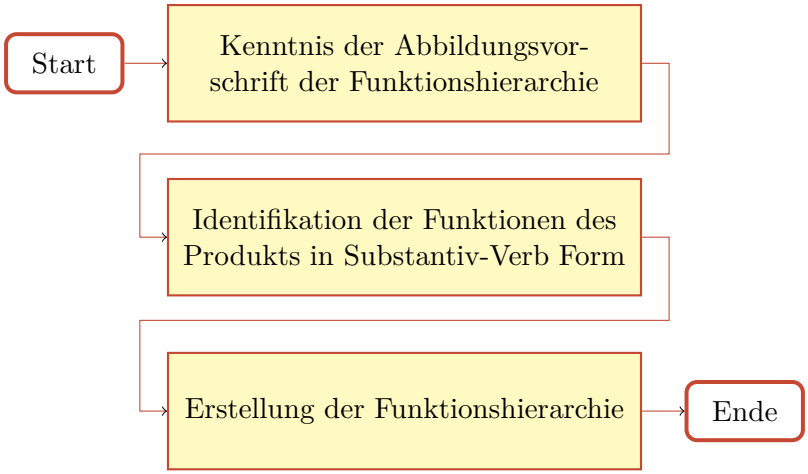


Bild 4.19: Ablauf zur Erstellung der Funktionshierarchie

Die Darstellung einer allgemeingültige Abbildungsvorschrift der Funktionshierarchien initiert den Prozess der Erstellung der Funktionshierarchie. Daraufhin sollen auf Basis der identifizierten Aktivitäten des Produktes, die Beschreibung der Funktionen in Substantiv-Verb Form erfolgen. Diese werden im letzten Schritt zu einem hierarchisch gegliederten Graphen angeordnet.

Abbildungsvorschrift zur Erstellung der Funktionshierarchie

Zur eindeutigen, replizierbaren und rechnerverarbeitbaren Darstellung der Funktionshierarchien in immer gleicher Weise wird die folgende Abbildungsvorschrift im Rahmen dieser Arbeit vorgeschlagen (vgl. Bild 4.20). Dabei wurde den Arbeiten von Roth (Roth, 2000) und Fowler (Fowler, 1981) Rechnung getragen.

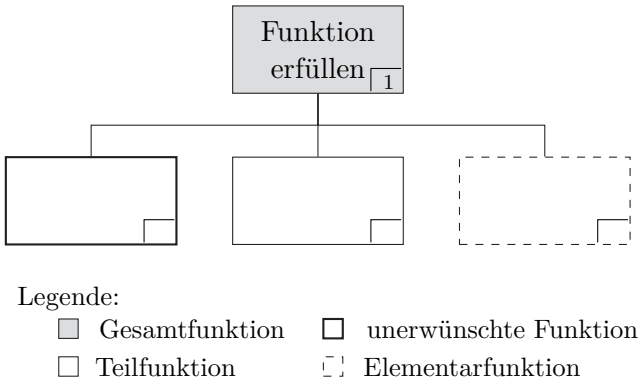


Bild 4.20: Abbildungsvorschrift zur Erstellung der Funktionshierarchie

Die Gesamtfunktion des Produktes soll in Form eines eingegrauten Rechtecks dargestellt werden. Alle daraus ableitbaren Teilfunktionen werden in einem leeren Rechteck abgebildet. Lassen sich Teilfunktionen nicht weiter zerlegen, so sind dies Elementarfunktionen, die als Rechteck mit gebrochener Berandung dargestellt werden. Da neben erwünschten Funktionen auch unerwünschte oder störende Funktionen auftreten können, sollen diese als Rechteck mit fetter Berandung gekennzeicnet werden.

Die Verbindung zwischen den Elementen erfolgt hierarchisch. Eine Gesamtfunktion, kann aus Teilfunktionen, Elementarfunktionen und unerwünschten Funktionen bestehen, während eine Teilfunktion ebenfalls aus weiteren Teilfunktionen, Elementarfunktionen und unerwünschten Funktionen beste-

hen kann. Elementarfunktionen und Unerwünschte Funktionen sind nicht weiter zerlegbar.

Jedes Element der Funktionshierarchie muss eindeutig beschrieben werden. Zur Beschreibung der Produktfunktion soll die Substantiv-Verb Form verwendet werden. Diese besteht aus einem Substantiv, das die Sache umschreibt, und einem Verb, das die Aktivität der Sache umschreibt. Dabei ist die Wahl des Substantivs oder Verbs unbeschränkt, solange die Schilderung der Aktivität eines Produktes oder einer Produktkomponente eindeutig und verständlich erfolgt. Darüberhinaus wird jedes Element der Hierarchie, im rechten unteren Kästchen, bezüglich seiner Position in der Hierarchie klassifiziert. Die Gesamtfunktion trägt die Klassifizierungsnummer 1. Alle darunterliegenden Ebenen werden von links nach rechts gezählt. Das Element der Teilfunktion wäre somit als 1.2 zu klassifizieren, da es auf der zweiten Ebene, als zweites Element positioniert ist. Entsprechend wäre das Element der Elementarfunktion als 1.3 zu klassifizieren. Bild 4.20 zeigt die Notationsregeln der Funktionshierarchie.

Eine zusätzliche Eigenschaft der Funktionshierarchie stellt die hierarchische Gliederung in funktionale Aspekte des Produkts dar. Wird nämlich die Funktionshierarchie von der Gesamtfunktion zu den Elementarfunktionen gelesen, so muss immer die Frage nach dem Wie geklärt werden. In der anderen Lesrichtung ergibt sich eine Fragestellung, die das Warum aufgreift. Diese Charakteristik bezieht sich lediglich auf die Verbindung der Elemente, Gesamtfunktion, Teilfunktion und Elementarfunktion, die ein Produkt funktional beschrieben.

Produktfunktionen in Substantiv-Verb Form

Um die Funktionen des Produkts sowie der beteiligten Produktkomponenten beschreiben zu können, müssen vorab die Anforderungen und somit die Produktaufgaben eindeutig geklärt sein. Die Funktionen sollen für die Erstellung der Funktionshierarchie nur allgemein und möglichst abstrakt und neutral formuliert werden. Eine solche Formulierung lässt sich am besten in einer Substantiv-Verb Form erreichen. Die Substantiv-Verb Form die hier verwendet werden soll, ist in ihrer Darstellung und in ihrem Sinn nicht mit der von Tomiyama oder Roth zu verwechseln. Während Tomiyama eine netzartige Verknüpfung von Verben mit beliebigen Substantiven beschreibt (vgl. Kapitel 2.2 auf Seite 16), nutzt Roth ein festgelegtes Substantiv-Verb Konstrukt zur Erstellung der Produktaufgabenstruktur (vgl. Kapitel 2.2.1

auf Seite 17). Roth versucht durch die Festlegung der Verben die rechnerunterstützte Durchgängigkeit der Modellierungsphasen zu verbessern.

Im Rahmen der vorliegenden Arbeit dient die Funktionshierarchie jedoch der klassifizierbaren Visualisierung der Funktionen in einer virtuellen Umgebung, wobei je nach Klassifizierungstiefe detailliertere Sichten auf das Zusammenspiel der Produktkomponenten fließend dargestellt werden können. Um dies zu erreichen, wird die Funktionshierarchie mit der Funktionsstruktur rechnerintern gekoppelt, was in Kapitel 4.2.4 und Kapitel 4.3 ausführlich diskutiert wird.

Am Produktbeispiel des RS232 Switches soll die Identifikation der Produktfunktionen erläutert werden. Dieser wurde bereits vorab in Bild 4.2 auf Seite 96 kurz eingeführt und in Kapitel 4.1 auf Seite 96 modelliert. Um die Produktfunktionen des RS 232 Switches genau identifizieren zu können, muss dessen Funktionsweise hinlänglich bekannt sein. Zum besseren Verständnis zeigt Bild 4.21 das CAD Modell des Switches und benennt die Hauptkomponenten. Die wesentlichen Elemente sind, das Gehäuse, das Display, der Schalter, die Ein-/und Ausgänge (Ports), der RJ45 Transceiver (Sende- und Empfangseinheit), der RS232 Datenswitch sowie der Prozessor (8051).

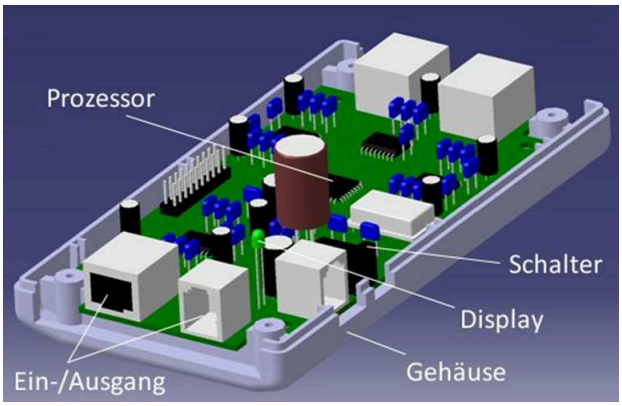


Bild 4.21: Benanntes CAD Modell des RS232 Switch

Die primäre Funktion des RS232 Switches besteht in der Schaltung zwischen RS232 Kanälen durch das Erkennen festdefinierter Eingangsgrößen. Der RS232 Switch ermöglicht die Kommunikation zwischen drei RS232 tauglichen Geräten in der Weise, dass ein Gerät die Kommunikation einleitet und die zwei anderen in Warteposition geschaltet werden. Dabei benutzt

das RS232 Protokoll zur Datenübertragung ein einfaches asynchrones serielles Verfahren. Seriell bedeutet, dass die einzelnen Bits des zu übertragenden Bytes nacheinander über eine einzige Datenleitung versendet werden. Asynchron bedeutet, dass es keine Taktleitung gibt, die dem Datenempfänger genau sagt, wann das nächste Bit auf der Datenleitung liegt. Alle RS232-Leitungen, mit Ausnahme der Masseleitung, arbeiten mit den Spannungspegeln $+12V$ (für eine logische '0') und $-12V$ (für eine logische '1'). Um Daten minimalistisch von einer Datenquelle zu einem Datenempfänger zu übertragen, werden eigentlich nur zwei Drähte benötigt, eine Masseleitung und eine Datenleitung. (Sprut)

Die Verschaltung verschiedener RS232 Geräte durch den Switch kann manuell oder automatisch erfolgen. Dabei verhält sich der RS232 Switch im Sinne der Kommunikation neutral. Es existiert keine Fehlerkorrektur hinsichtlich "false bit/byte". Ebenso werden die übertragenden Daten in keiner Weise verändert. Im automatischen Modus wird nach empfangenem Befehl der Haupteinheit die Kommunikation zwischen RS232 Switch und den anhängenden Geräten abgeschaltet. Der Prozessor übernimmt dann die Kommunikation mit dem Schalter. Nach durchgeführter Schaltung des aktiven Kanals, stellt wiederum der RS232 Switch die Kommunikation zwischen der Haupteinheit und der untergeordneten Einheit auf einem neuen Kanal innerhalb weniger Mikrosekunden her.

Aus der zweckmäßigen, funktionalen Beschreibung des Gesamtprodukts und der wesentlichen, beteiligten Produktkomponenten des RS232 Switches lassen sich die jeweiligen Produktfunktionen ableiten.

Komponente	**Produktfunktion**
Gehäuse	Komponenten sichern
Display	Status signalisieren
Schalter	Signal umwandeln
Ein/Ausgang	Komponenten verbinden
RJ45 Transceiver	Signal importieren
RS232 Switch	Signal leiten
Prozessor 8051	Daten verarbeiten

Tabelle 4.1: Ableitung der Produktfunktionen des *RS232 Switch*

Für das Gesamtprodukt RS232 Switch kann die Funktion der *Schaltung zwischen RS232 Kanälen* festgehalten werden. In Substantiv-Verb Form könn-

te dementsprechend auf abstrakter Ebene die Produktfunktion *Signal leiten* formuliert werden. Dies lässt sich methodisch für alle beteiligten Komponenten entsprechend bewerkstelligen. Tabelle 4.1 listet die wesentlichen Komponenten mit den zugehörigen, in der Substantiv-Verb Form formulierten Produktfunktionen auf.

Erstellung der Funktionshierarchie

Sind die Produktfunktionen für alle relevanten Produktkomponenten in Substantiv-Verb Form hinreichend beschrieben, müssen diese in einer hierarchischen Form angeordnet werden. Der Aufbau der Funktionshierarchie strukturiert sich dabei von der Gesamtfunktion, als Wurzel des Graphen, über die Teilfunktionen hin zu den Elementarfunktionen, als dessen Blätter. Diese Anordnung entspricht einem gestaltorientertem, konstrukteursgerechtem Denken. Durch die Gruppierung von Elementarfunktionen in Teilfunktionen, sowie dieser in der Gesamtfunktion entspricht dies der Vorgehensweise zur Erstellung einer Produktstruktur in einem CAD System. In Kapitel 4.2.2 auf Seite 133 wird die Klassifizierung von Produktstruktur und Funktionsstruktur näher betrachtet, Kapitel 4.3 beschreibt die semantische Verwendung der Informationen.

Im Rahmen einer methodisch korrekten Erstellung der Funktionshierarchie muss jedes Element einer übergeordnete Ebene als eine Gruppierung von Elementen der darunterliegenden Ebene, entsprechend der geometrischen Repräsentation, dargestellt werden. Der Hierarchiebaum wird dabei von links nach rechts traversiert, entsprechend der Produktstruktur. Auf diese Weise kann von der primären Funktionsbeschreibung eines Produkts bis zur elementaren Teilfunktionsbeschreibung ein komplexes Produkt detailliert und dabei disziplinübergreifend modelliert werden. Die klassifizierten 3D Objekte des Szenengraphs können somit im Funktionsmodell aus ihren topologischen Zusammenhängen herausgelöst, neu angeordnet und wieder zurück in ihre Ursprungspositionen überführt werden. Neue topologische Anordnungen realisieren neue Sichten auf das Produkt, die mit Zusatzinformationen wie Metadaten und Verknüpfungselementen erweitert werden. Folglich ermöglicht eine geometriegebundene Hierarchisierung der Funktionen, neben dem nachvollziehbaren Produktaufbau, vor allem neue topologische Anordnungen der 3D Objekte, und somit unterschiedliche Sichten in der virtuellen Umgebung.

In Bild 4.22 ist die Funktionshierarchie der in Tabelle 4.1 identifizierten Komponenten des RS232 Switches, entsprechend der in Kapitel 4.2.1 auf Seite 120 vorgestellten Notation, modelliert worden.

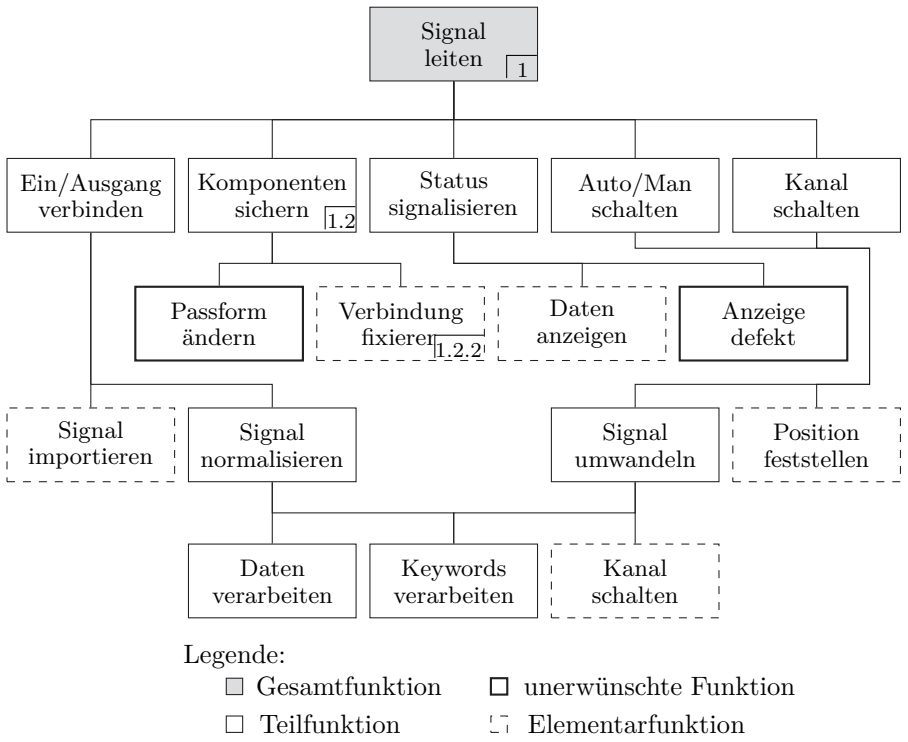


Bild 4.22: Funktionshierarchie des RS232 Switch

Die Gesamtfunktion des Switches, nämlich ein RS232 *Signal* einer an dem Switch angeschlossenen Einheit zu einer Zweiten weiterzu*leiten*, steht an der Wurzel des Baumes. Von diesem leiten sich die Teilfunktionen der geometrischen Gruppen der darunterliegenden Ebene ab. Dies sind *Ein-/Ausgang verbinden*, *Komponenten sichern*, *Status signalisieren*, *Auto/Man schalten* sowie *Kanal schalten*. Analog dazu lässt sich jede Funktion wiederum als Wurzelfunktion betrachten, zu der eine Reihe weiterer Elemente funktional zugeordnet ist. So ist die Funktion *Status signalisieren* beispielsweise Wurzelfunktion der Elementarfunktion *Daten anzeigen* und der unerwünschten Funktion *Anzeige defekt*. Wird Bild 4.22 von oben nach unten gelesen, beschreibt jedes übergeordnete Element wie das untergeordnete Element umgesetzt wird. Wie wird der *Status signalisiert*? Indem die *Daten angezeigt* werden. Durch die umgekehrte Lesart wird die Frage des Warums geklärt. Warum werden die *Daten angezeigt*? Um den *Status zu signalisieren*. Die

Position der Elemente in der Hierarchie ist exemplarisch bis zur Elementarfunktion *Verbindung fixieren* erfolgt. Diese ist in der dritten Ebene an zweiter Stelle von links positioniert, was der Position 1.2.2 in der Funktionshierarchie entspricht. Die Hierarchisierung kann bis zu dem Punkt fortgeführt werden, an dem die Komponenten des Produkts nicht weiter zerlegbar sind oder eine weitere Unterteilung nicht mehr im Sinne der Produktentwicklung ist. Eine derart erstellte Funktionshierarchie ist die Basis zum eindeutigen Verständnis des Produkts mit seinen Funktionen.

4.2.2 Erzeugung der Funktionsstruktur

Die Erzeugung der Funktionsstruktur setzt die Kenntnis der hierarchischen Zusammenhänge der jeweiligen Produktfunktionen, wie sie in Kapitel 4.2.1 auf Seite 118 behandelt wurde, voraus und verbindet die anfallenden physikalischen Größen der zugehörigen Funktionselemente miteinander. Funktionsstrukturen wurden in Kapitel 2.2.2 auf Seite 18 erläutert und werden im Rahmen dieser Arbeit als flussorientierte Strukturen verstanden, die von allgemeine in spezielle Darstellungen überführt werden können.

Darüber hinaus sollen die Funktionsstrukturen klassifiziert, strukturiert und modularisiert werden, um sie rechnerverarbeitbar zu machen. Dabei sollen gleiche Typen von Objekten in einer Klasse abstrahiert werden. Diese werden durch das Funktionsobjekt, das Funktionsverb und der Transition (den Funktionsfluss) beschrieben. Funktionsobjekte sind gekennzeichnet durch die Führungsgröße, die mittels Attributen die physikalischen Ein- und Ausgangsgrößen quantitativ beschreibt. Für die Abstraktion der Funktionsobjekte stehen verschiedene Klassifizierungssysteme wie eCl@ss (eClass), ETIM (Etim), classmate (classmate), Proficl@ss (proficlass) etc. zur Verfügung. Funktionsverben beschreiben die Funktion, wie die Eingangsgrößen in die Ausgangsgrößen umgesetzt werden und können ebenfalls verschiedenartig klassifiziert werden (vgl. Kapitel 2.2.2 auf Seite 18). Transitionen bezeichnen die Verbindungen zwischen Funktionsobjekten und Funktionsverben. Bild 4.23 veranschaulicht die Klassifizierungs- und Strukturierungsmöglichkeiten.

Je nach Anwendungszweck wird eine der Taxonomien ausgewählt. Bei der Klassifizierung wird jedes Element des Funktionsmodells einer Klasse zugeordnet und durch von der Klasse abhängige unterschiedliche objektivierbare Eigenschaften beschrieben. Die Klassifizierung erleichtert die semantische

Abbildung zwischen verschiedenen Systemen durch die Spezifikation der vorhandenen Eigenschaften jeder Klasse.

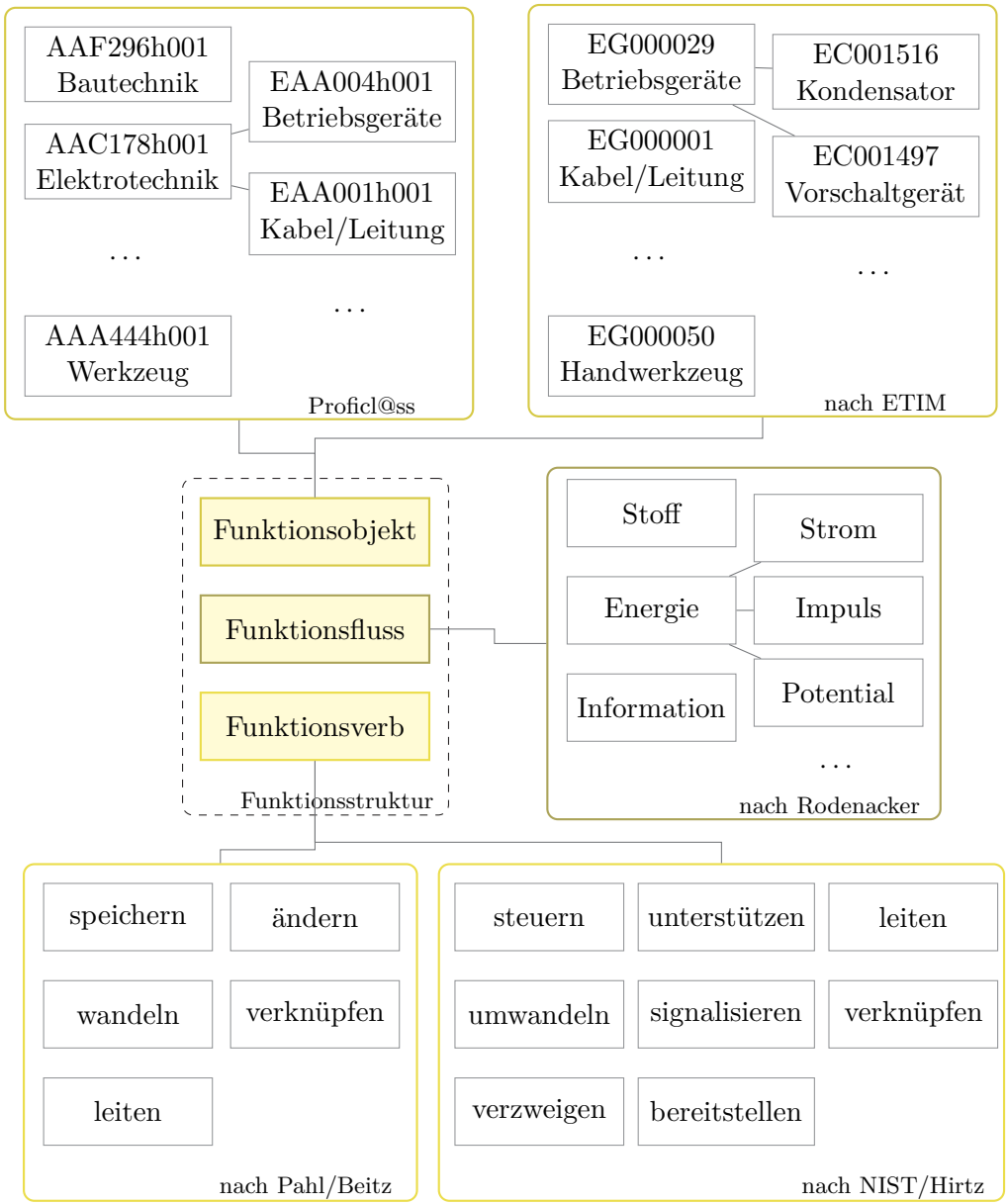


Bild 4.23: Klassifizierung und Sturkturierung der Funktionsstruktur

Die Funktionsstruktur ergänzt sich mit der Funktionshierarchie in der Form, dass hinter jeder identifizierten Produktfunktion in der Funktionshierarchie, eine auf physikalischen Größen basierende Funktionsstruktur steht. Bei Elementarfunktionen, also dem minimalen Fall, besteht diese lediglich aus drei

Elementen, nämlich zwei Funktionsobjekten, verbunden durch ein Funktionsverb, während Teilfunktionen aus vielen Elementen bestehen können. Um die Funktionsstruktur eindeutig und rechnerverarbeitbar zu halten, ist die Menge der Funktionsverben klein zu halten. Konstruktionsmethodiker wie Roth (Roth, 2000), Rodenacker (Rodenacker, 1991), Krumhauer (Krumhauer, 1974), Koller (Koller, 1998), Langlotz (Langlotz, 2000), Ullman (Ullman, 1992) oder Hirtz (Hirtz et al., 2002) definieren alle eine ähnliche aber dennoch unterschiedliche Menge an Funktionsverben (vgl. Tabelle 2.2 auf Seite 20). Prinzipiell lässt sich der im Folgenden beschriebene Ansatz zur Bestimmung der Funktionsstruktur für jede beliebige jedoch eindeutige und feste Menge von Funktionsverben nutzen. So können die Funktionsverben nach einem der oben genannten Konstruktionsmethodiker, aber auch ein firmenspezifischer Satz an Funktionsverben als Grundlage der Funktionsstruktur genutzt werden.

Zur Bestimmung der Funktionsstruktur sind die in Bild 4.24 dargestellten Prozessschritte zu durchlaufen.

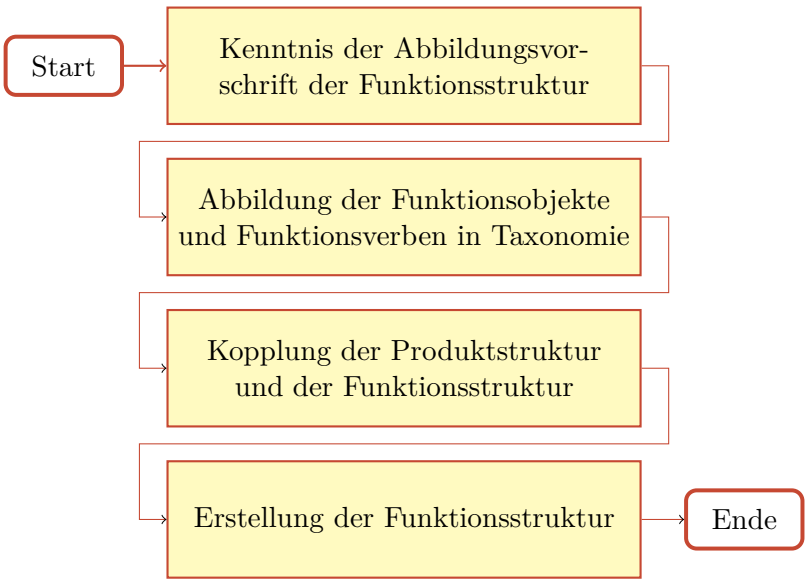


Bild 4.24: Ablauf der Funktionsstrukturerstellung

Um die Funktionsstruktur eindeutig und wiederholbar abbilden zu können, ist die Kenntnis einer exakten Abbildungsvorschrift notwendig. Auf Basis dieser sollen im zweiten Schritt die Funktionsobjekte ihren Führungsgrößen mit den entsprechenden Attributen zugewiesen, sowie die Funktionsverben

bestimmt werden. Als Grundlage soll die Taxonomie von Hirtz dienen (vgl. Anhang A auf Seite 279). Die Kopplung der Produktstruktur mit der Funktionsstruktur anhand einer kohärenten Klassifizierung wird darauffolgend diskutiert. Abschließend werden die Funktionselemente zu einer Funktionsstruktur flussartig miteinander verbunden.

Abbildungsvorschrift zur Erstellung der Funktionsstruktur

Zur eindeutigen, replizierbaren und rechnerverarbeitbaren Darstellung der Funktionsstrukturen in immer gleicher Weise wird die folgende Abbildungsvorschrift im Rahmen dieser Arbeit vorgeschlagen (vgl. Bild 4.25). Diese orientiert sich an den Arbeiten von Langlotz (Langlotz, 2000, p.133ff) wurde aber hinsichtlich der Zielstellung der Arbeit erweitert.

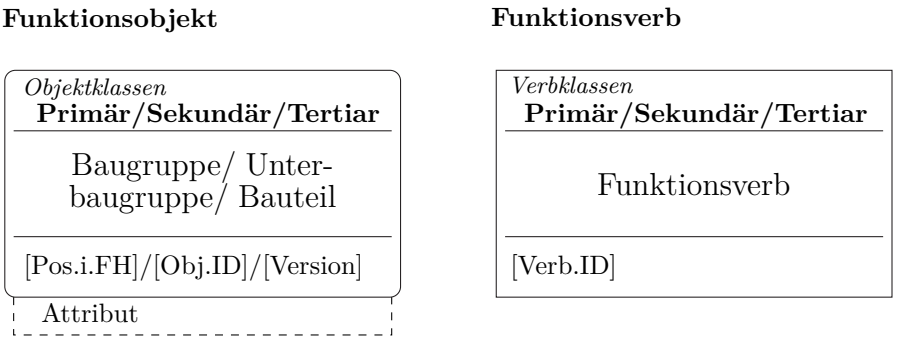


Bild 4.25: Abbildungsvorschrift zur Erstellung der Funktionsstruktur

Funktionsobjekte werden in einem Rechteck mit abgerundeten Ecken dargestellt. Dieses ist in drei Teile unterteilt. In der Kopfzeile werden die Objektklassen des Funktionsobjektes festgelegt, im Textfeld die Gestaltelemente und in der Fußzeile die Ordnungselemente. Die Objektklassen basieren auf der Taxonomie von Hirtz und sind im Anhang A auf Seite 279 aufgelistet. In der Kopfzeile des Funktionsobjekts werden die primären, sekundären und tertiären Objektklassen beschrieben. Ist eine tertiäre Objektklasse nicht vorhanden, so wird die Sekundäre doppelt verwendet. Das mittlere Textfeld kennzeichnet das, dem Funktionsobjekt zugehörige, Gestaltelement. Dieses kann ebenfalls dreifach beschrieben werden, als Baugruppe, Unterbaugruppe oder Bauteil. Existieren lediglich ein oder zwei Gestaltelemente, werden diese doppelt oder dreifach aufgeführt. Die Unterteilung der Objektklassen und der Gestaltelemente korreliert dabei. In der Fußzeile wer-

den die Ordnungselemente Position in Funktionshierarchie, Objekt-ID und Versionsnummer festgehalten. Dabei beschreibt die Position der Funktionshierarchie die Klassifizierungsummer der Produktfunktion, die während der Erstellung der Funktionshierarchie im vorhergehenden Schritt (vgl. Kapitel 4.2.1) generiert wurde. Die Objekt-ID identifiziert das Funktionsobjekt kohärent im Gesamtzusammenhang, während die Versionsnummer die Wiederverwendbarkeit der Funktionsstrukturen bei der Varianten- und Anpassungskonstruktion sicherstellt. Darüberhinaus wird unterhalb des Rechtecks eine gestrichelte Box angehangen, in der die Attribute der Führungsgröße respektive anderer Metadaten hinzugefügt werden können.

Funktionsverben werden in einem einfachen Rechteck modelliert, das aus drei Teilen besteht. Der obere Teil beschreibt die Verbklassen der Funktion, das Textfeld die Funktion selbst, der untere Teil enthält die Verb-ID. Die Verbklassen entsprechen denen bei Hirtz klassifizierten und sind analog zu den Objektklassen in Anhang A auf Seite 279 gelistet. Diese werden in drei Gruppen unterteilt, wobei jede Verbklasse benannt werden muss. Ist keine tertiäre Verbklasse vorhanden, wird die Sekundäre doppelt verwendet. Das Textfeld beschreibt die eigentliche Funktion in Form des Funktionsverbs.

Abbildung der Funktionsobjekte und Funktionsverben

In dieser Arbeit wird die Taxonomie der Funktionsverben und Funktionsobjekte aus dem NIST nach Hirtz (Hirtz et al., 2002) genutzt, da diese die kleinste Menge an Verb- und Objektklassen für interdisziplinäre Produkte definiert. Dabei wurde eine dreistufige Unterteilung von Verb- und Objektklassen vorgeschlagen. Für die Funktionsverben wurden in der abstrakten, primären Verbklasse acht Verben identifiziert, in der sekundären Verbklasse 21 Verben und in der konkreten, tertiären Verbklasse mehrere Dutzend Verben (vgl. Tabelle 2.2 auf Seite 20). Im Anhang A auf Seite 279 wird der vollständige Satz an Funktionsverben aller Verbklassen dargestellt. Die Funktionsverben aus der primären Verbklasse werden in Tabelle 4.2 hinsichtlich ihrer Funktion beschrieben.

Die Funktionsobjekte wurden ebenfalls dreifach unterteilt. Dabei wird in der primären Objektklasse zwischen Materie, Signal und Energie unterschieden. In der sekundären Objektklasse wird diese Unterteilung weiter verfeinert (vgl. Tabelle 4.3). Die tertiäre Objektklasse umfasst, wo möglich, weitere Konkretisierungen und listet synonym verwendbare Nomen auf (vgl. Anhang A auf Seite 279). Eine Unterteilung von primärer zu sekundärer Ob-

jektklasse ist immer möglich.

Verbklasse	
primär	*Beschreibung*
verzweigen (branch)	Änderung der Anzahl, z.B. eine Eingangskraft in zwei Ausgangskräfte verzweigen
leiten (channel)	Änderung des Ortes, z.B. ein Signal von einem Ort A zu einem Ort B leiten
verknüpfen (connect)	Änderung der Anzahl, z.B. zwei Eingangskräfte zu einer Ausgangskraft verknüpfen
steuern (control magnitude)	Änderung des Zustands, z.B. das Starten und Stoppen eines Motors
umwandeln (convert)	Änderung der Art, z.B. Druck in Kraft umwandeln
bereitstellen (provision)	Änderung der Zeit, z.B. eine Kraft (in einer Feder) bereitstellen
signalisieren (signal)	Änderung des Zustands, z.B. signalisieren einer Änderung
unterstützen (support)	Änderung der Zeit, z.B. eine Kraft halten

Tabelle 4.2: Funktionsverben nach Hirtz (Hirtz et al., 2002)

Objektklasse	
primär	*sekundär*
Materie	Menschlich, Gasförmig, Flüssig, Fest, Plasma, Gemisch
Signal	Status, Steuerung
Energie	Mensch, Akustisch, Biologisch, Chemisch, Elektrisch, Elektromagnetisch, Hydraulisch, Magnetisch, Mechanisch, Pneumatisch, Radioaktiv, Thermisch

Tabelle 4.3: Objektklassen nach Hirtz (Hirtz et al., 2002)

Jedoch ist eine weitere Unterteilung in die tertiäre Objektklasse nicht immer zweckmäßig. Beispielsweise lässt sich Energie und menschlich ebenso

wenig sinnvoll weiterunterteilen, wie Energie und chemisch, oder Energie und thermisch.

Eine solche übergangslose Differenzierung von primären, abstrakten Objektklassen, über sekundären, zu tertiären, konkreten Objektklassen bildet die Grundlage für die nahtlose Kopplung der Gestaltelemente mit den Funktionsobjekten, denn auf Basis der Objektklassen findet die gegliederte Einteilung der Funktionsobjekte statt. In den meisten industriellen Anwendungsfällen existiert bereits eine Produktgestalt, da es sich in der Regel um Anpassungs- oder Variantenkonstruktionen handelt. Dem entsprechend konnte in Kapitel 4.1 auf Seite 96 bereits eine 3D Umgebung modelliert werden. Somit sind die in der virtuellen Umgebung vorhandenen 3D Objekte bekannt.

Anhand dieser Voraussetzungen sollen am Beispiel des RS232 Switches die Funktionsobjekte und Funktionsverben, die an der Produktfunktion *Kanal schalten* beteiligt sind, exemplarisch abgebildet werden (vgl. Tabelle 4.4). Der Hebel, der zum Schalten dient, ist Bauteil der Unterbaugruppe Schalter, die wiederum der Baugruppe Switch untergeordnet ist. Wird eine Kraft von der Hand auf den Hebel ausgeübt, ist dies laut Taxonomie von Hirtz zuallererst eine Form der Energie. Diese wird vom Mensch geleistet und als Kraft aufgebracht. Dadurch verändert sich die Position des Hebels. Durch die Positionsänderung wird die mechanisch aufgebrachte Kraft in ein elektronisches Signal gewandelt, das anschließend weiterverarbeitet werden kann. Diese Produktfunktion ist im RS232 Switch zweimal vorhanden.

Funktionsobjekt	**Objektklasse**
Arm/ Hand/ Finger	Energie/ Mensch/ Kraft
Switch/ Schalter/ Hebel	Energie/ Mensch/ Kraft
Swicth/ Schalter/ IC	Signal/ Steuerung/ Getrennt
Funktionsverb	**Verbklasse**
Kraft importieren	leiten/ importieren
Kraft in Signal wandeln	umwandeln/ umwandeln

Tabelle 4.4: Zugeordnete Objekt-/Verbklassen der Funktion *Kanal schalten* des RS232 Switches

Zum Einen wird dadurch, wie eben beschrieben, die Produktfunktion *Kanal schalten* umgesetzt, zum Anderen wird so ebenfalls die Produktfunktion

Auto/Man schalten realisiert. Bild 4.2 auf Seite 96 zeigte die Produktgestalt, an der zwei Hebel zu erkennen sind, und Bild 4.22 auf Seite 125 erläuterte diesen Zusammenhang im Rahmen der Erstellung der Funktionshierarchie. Tabelle 4.4 zeigt die beteiligten Funktionsobjekte und deren Zuordnung zu den Objektklassen, sowie die Funktionsverben und die Zuordnung zu den Verbklassen.

Kopplung von Produktstruktur und Funktionsstruktur

Die Produktstruktur gliedert die Produktgeometrie in modulare Strukturen. Die einzelnen geometrischen Module werden in Baugruppen und Einzelteile unterteilt. Dabei wird das Gesamtprodukt in einer ersten Abstraktionsstufe in Baugruppen aufgeteilt, welche die Produktgliederung darstellen. Durch das Auflösen der Baugruppen in Komponenten wird die Produktgliederung in die Produktstruktur überführt (DIN-199-1, 2002). Bild 4.26 zeigt eine typische Produktstruktur in einer hohen Abstraktionsebene.

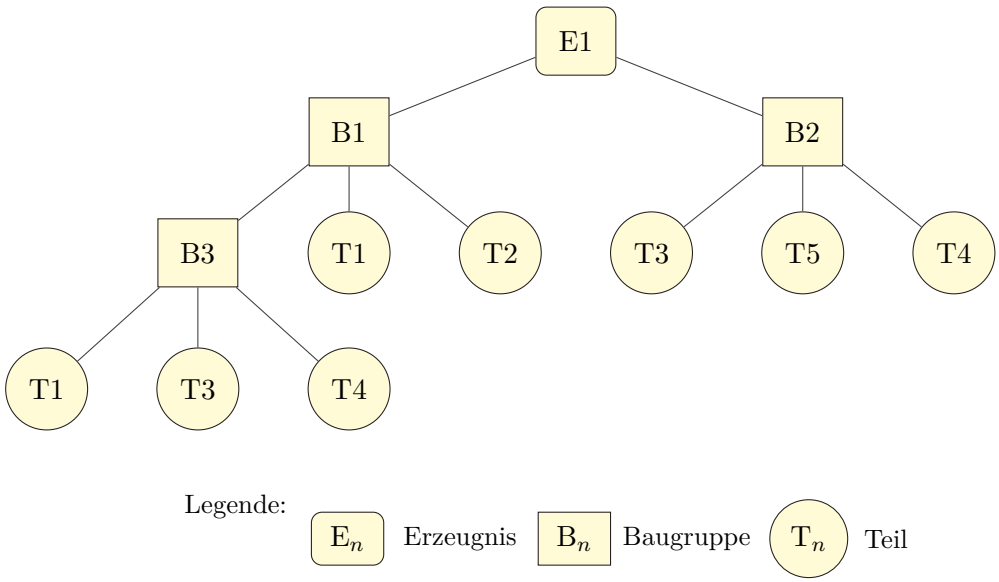


Bild 4.26: Produktstrukturbaum nach DIN-199

Hierbei ist erkennbar, dass das Gesamtprodukt die Wurzel der Produktstruktur darstellt. Diese ist als Oberbaugruppe eines Produktes zu verstehen und beinhaltet alle Unterbaugruppen in einer hierarchischen Gliederung. An der Wurzel sind die Baugruppen des Erzeugnisses angefügt, welche die

Produktgliederung darstellen. An diese sind die zur jeweiligen Gruppe gehörenden Teile angehangen.

Die Produktstruktur wird als Basis für die Klassifizierung verwendet, in der das Erzeugnis bzw. Gesamtprodukt die Oberklasse, die Baugruppen die Unterklassen und die Einzelteile die Elemente innerhalb der Klassifizierung bilden. Dabei stellt der in Bild 4.26 dargestellte Produktstrukturbaum bereits die Visualisierung der verwendeten hierarchischen Klassifikationsstruktur als Baumstruktur dar. In dieser Struktur besitzt jedes Kind-Objekt nur ein Eltern-Objekt, womit die hierarchische Abbildung bis zur obersten Baugruppe, dem Erzeugnis, gewährleistet ist. Die hierarchische Klassifikation erfolgt durch einen Klassifizierungsschlüssel auf Nummernbasis, wobei die einzelnen Stellen dieses Schlüssels voneinander abhängig sind (Ovtcharova, 2006a).

Bild 4.27 zeigt die Überführung des Produktstrukturbaumes in einen Klassifizierungsbaum, dessen Bezeichnungen der Wurzel und Blätter auf Nummernbasis erstellt sind. Die hierarchische Klassifizierung erlaubt das Erkennen der räumlichen Beziehungen zwischen den Einzelteilen untereinander und zwischen den Einzelteilen und ihren übergeordneten Baugruppen.

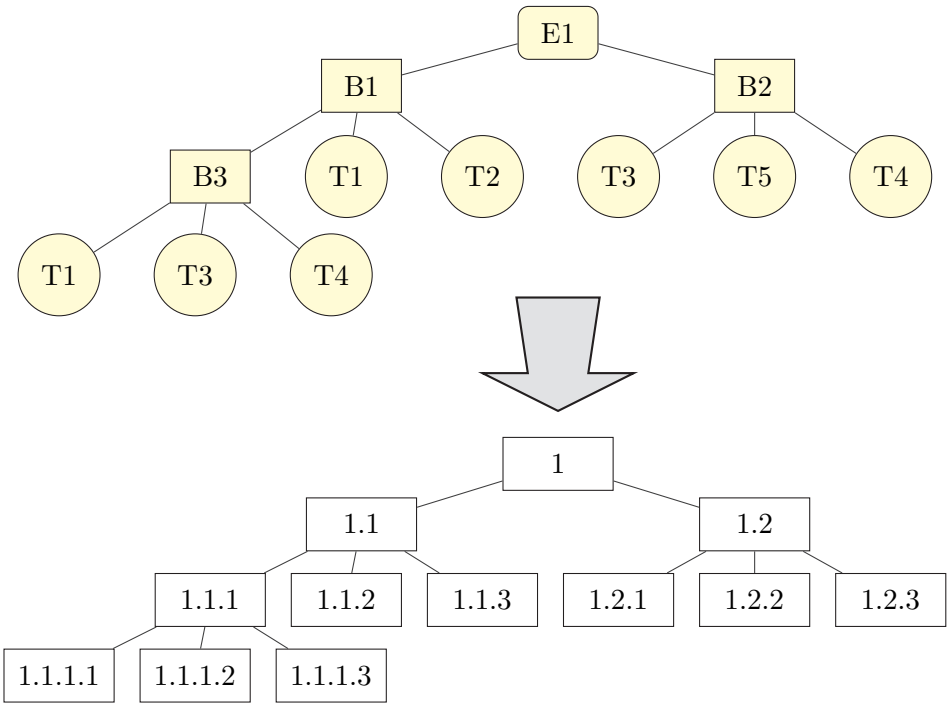


Bild 4.27: Durchführung der Produktklassifizierung anhand des Produktstrukturbaums

Nutzen davon ist die eindeutige und widerspruchsfreie Identifizierung aller Komponenten und Baugruppen. Die Ordnungsnummer der Klassifizierung wird direkt in die einzelnen Objektnamen überführt, damit die geometrische und topologische Zuordnung jederzeit durch die Bezeichnung gewährleistet ist. Die dafür verwendete Notation besteht aus der Ordnungsnummer, einem Trennelement sowie dem Bezeichner. Die Ordnungsnummer leitet sich dabei direkt aus den Ebenen des Produktstrukturbaums ab, während der Bezeichner aus dem Namen des Objektes abgeleitet wird. Die Notationsregel sieht wie folgt aus:

$$< Ordnungsnummer >< Trennstrich >< Bezeichner >$$

Die so erhaltene Bezeichnung findet sich in der Abbildungsvorschrift der Funktionsstruktur (vgl. Bild 4.25 auf Seite 129) in der Objekt-ID und dem Texteld wieder, wobei die Ordnungsnummer in der Objekt-ID und der Bezeichner im Textfeld hinterlegt wird.

Bild 4.28 stellt einen Ausschnitt der übertragenden Klassifizierung aus dem Produktstrukturbaums des RS232 Switches dar.

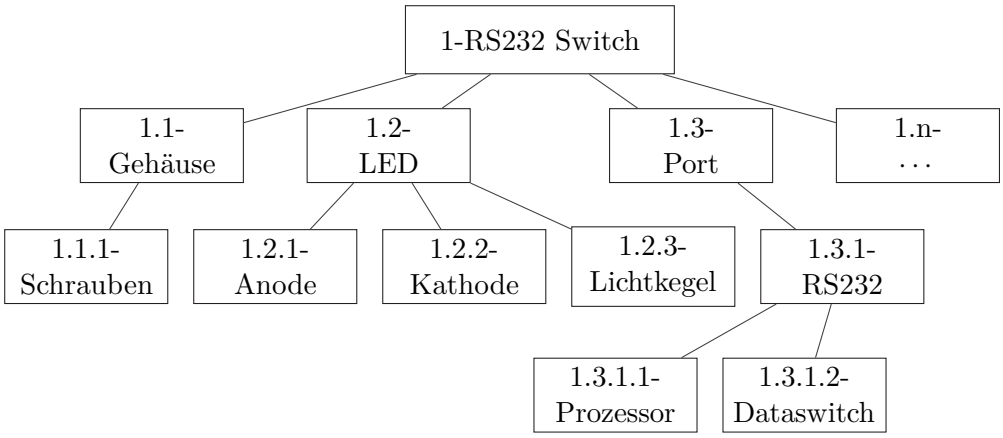


Bild 4.28: Produktklassifizierung des RS232 Switches

Für das Beispiel des Switches steht das Erzeugnis *RS232 Switch* an der Wurzel des Baumes. Unterbaugruppen dieses sind die *Ports*, das *Gehäuse*, die *LEDs* sowie die beiden *Schalter A und B*. Unterhalb der Baugruppe *Ports* sind desweiteren die Komponenten *RJ45* unf *RS232* angeordnet, die sich noch weiter unterteilen lassen. Die Funktion *Signal leiten* beschreibt somit die erste Ebene. Gemäß der konzipierten Notation wird diese im Szenengraph auf das 3D Objekt RS232 Switch abgebildet, wobei dem 3D Objekt die

Bezeichnung $1 - RS232Switch$ zugeordnet wird. Bei der Nummerierung der Klassifizierung wird der Baum der Funktionsstruktur von links nach rechts gelesen. Ein Element der 2. Ebene beispielsweise würde eine Bezeichnung $1.x - Objektname$ zugeordnet werden. Dementsprechend erhält die Funktion *Status signalisieren*, welche der Komponente *LED* zugeordnet wird, im Szenengraph die Bezeichnung $1.2 - LED$. Ein Element der 3. Ebene würde analog eine Bezeichnung $1.1.x - Objektname$ erhalten. Dieses Vorgehen lässt sich bis zur vollständigen Beschreibung des Modells fortsetzen. In der Funktionsstruktur wird das Funktionsobjekt *LED* wie Bild 4.29 darstellt modelliert.

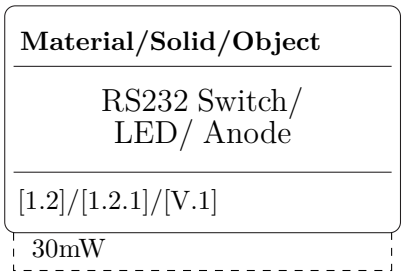


Bild 4.29: Funktionsobjekt *LED*

Erstellung der Funktionsstruktur

Sind die Funktionsobjekte und Funktionsverben eindeutig identifiziert, klassifiziert und auf die Objekt- und Verbklassen abgebildet, kann im finalen Schritt die Funktionsstruktur erstellt werden. Diese ordnet die Funktionen sequentiell, ihrem Ablauf entsprechend, in einer flussartigen Struktur an. Eine Funktion wird immer durch den Zusammenhang Funktionsobjekt, Transition, Funktionsverb, Transition, Funktionsobjekt beschrieben. Auf abstrakter Ebene wird die Führungsgröße eines Funktionsobjektes mittels des Funktionsverbes in die Führungsgröße eines anderen Funktionsobjektes überführt. Das bedeutet das an einer korrekt beschriebenen Funktion in der Funktionsstruktur immer genau drei Elemente beteiligt sein müssen.

Darüberhinaus werden zu Funktionsobjekten respektive Funktionsverben die zusätzlichen Informationen, wie die Position in der Funktionshierarchie, die Objekt-ID oder Verb-ID, die Versionsnummer des Funktionsobjekts, sowie die Attribute hinzugefügt.

Bild 4.30 beschreibt die Funktionsstruktur des Automatisch/Manuell Schalters des RS232 Switch.

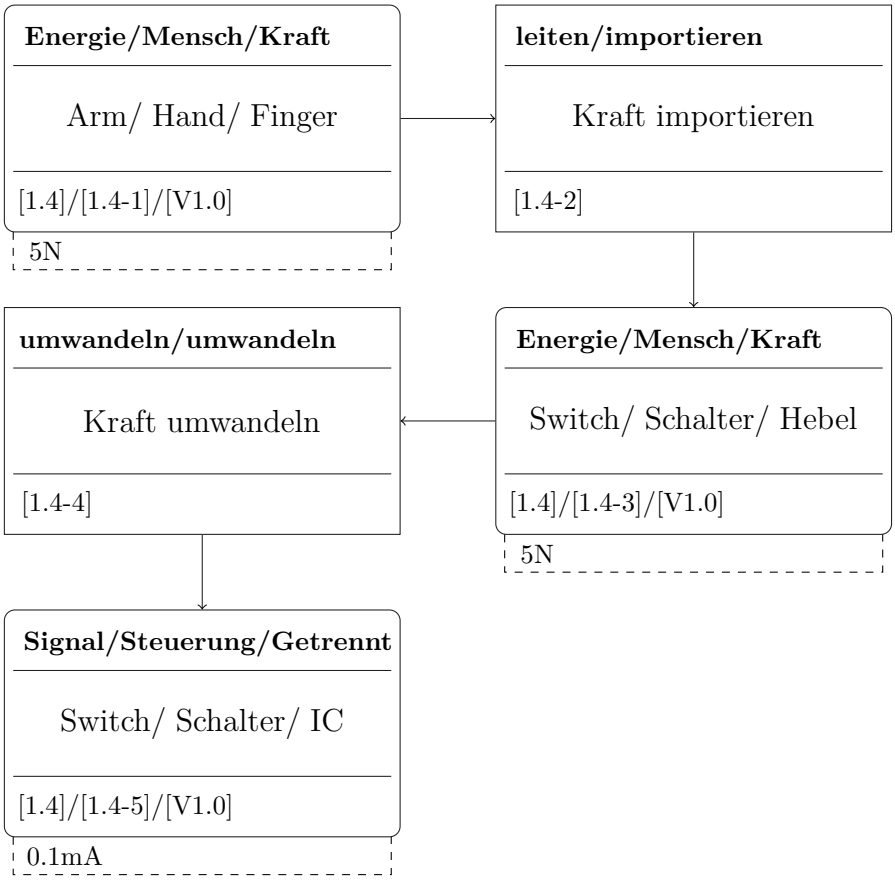


Bild 4.30: Funktionsstruktur des Schalters am RS232 Switch

Um den Schalter zu kippen, wird anfangs eine Kraft vom Menschen auf das Objekt übertragen. In der Funktionsstruktur wird dieser Zusammenhang in den ersten drei Elementen beschrieben. Die Baugruppe Arm, Unterbaugruppe Hand, Bauteil Finger importiert eine Kraft von $5N$ auf die Baugruppe Switch, Unterbaugruppe Schalter, Bauteil Hebel, die den Hebel neu positioniert. Um die übertragene Kraft im Switch weiternutzen zu können, muss diese im nächsten Schritt in ein elektronisches Signal gewandelt werden. Dementsprechend wird die durch die Kraft $5N$ neu positionierte Baugruppe Switch, Unterbaugruppe Schalter, Bauteil Hebel mittels der Baugruppe Switch, Unterbaugruppe Schalter, Bauteil IC in eine Stromstärke von $0.1mA$ umgewandelt. D.h. der durch den Schaltvorgang geschlossene Steuerstromkreis bewirkt einen Strom im Laststromkreis. Dadurch wird das so erzeugte

elektronische Signal vom Transceiver aufgenommen und an den Prozessor geleitet, der die Information weiter verarbeitet.

Analog zu der Produktfunktion *Auto/Man schalten* kann die gleiche Funktionsstruktur für die Produktfunktion *Kanal schalten* wiederverwendet werden. Die Funktionsstruktur wäre identisch aufgebaut, würde jedoch die Funktionsobjekte an die Position 1.5 der Funktionshierarchie anordnen und die Versionsnummer der wiederverwendeten Funktionsobjekte von V1.0 auf V1.1 im Rahmen desselben Produktes erhöhen.

4.2.3 Darstellung und Extraktion der Modelldaten

Nach dem das Funktionsmodell gemäß der Vorgehensweise aus Kapitel 4.2.1 und Kapitel 4.2.2 erstellt wurde, muss es am Rechner dargestellt werden. Dabei wird die Semantik zunächst in 2D Graphen abgebildet bzw. hinterlegt. Die rechnerunterstützte Abbildung der Funktionsmodelle ist notwendig, um die Modelle durchgängig von unterschiedlichen Entwicklungsphasen aus zu bearbeiten, zu speichern und später in der VR Szene, sprich auf dem Szenengraph, zu importieren (vgl. Bild 4.18 auf Seite 117). Naturgemäß erzeugt die rechnerverarbeitbare Darstellung zusätzliche Daten, die für die Semantik der Funktionsmodelle keine Rolle spielen. Diese müssen später entsprechend extrahiert werden.

Zur Darstellung der Funktionsmodelle stehen verschiedene Entwurfswerkzeuge zur Verfügung (vgl. Kapitel 2.2.3 auf Seite 21). Je nach gewählter Modellierungsmethode können die Datenstrukturen der erstellten Funktionsmodelle unterschiedlich sein. Jedoch müssen die Modellierungswerkzeuge alle relevanten Informationen aus den Modellen aufnehmen und diese als 2D Graphen veranschaulichen können. Zusätzlich müssen von den Modellierungswerkzeugen Softwareschnittstellen angeboten werden (vgl. 2.4 auf Seite 33), so dass auf die Informationen aus dem Funktionsmodell von anderen Anwendungen wie der VR Entwicklungsumgebung, zugegriffen werden kann.

Die Funktionsstrukturen stellen, wie in Bild 4.23 gezeigt, bereits ein Klassenmodell dar, das aus Funktionsverb, Funktionsobjekt und Funktionsfluss (Transition) besteht. Darüberhinaus kann anhand des Funktionsmodells ebenfalls ein Objektmodell beschrieben werden, das unterschiedliche Sichten auf das Gesamtprodukt herstellt (vgl. Kapitel 4.3.1). Beide Modelle stellen eine einfache Datenstruktur dar.

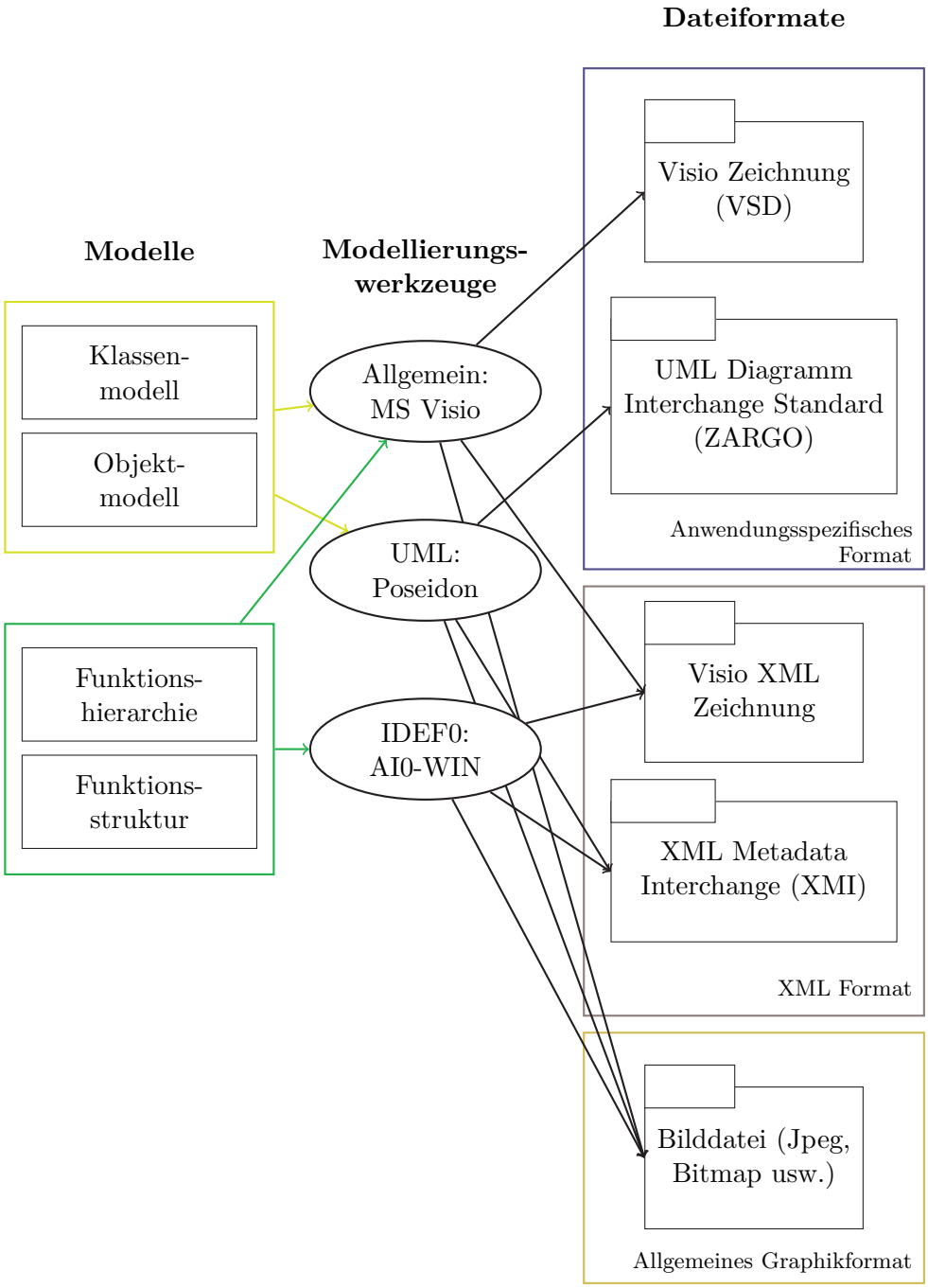


Bild 4.31: Funktionsmodelle sowie die zugehörigen Datenformate

Dementsprechend können diese mit beliebigen Modellierungswerkzeugen, die beispielsweise Standards wie UML oder ERD beherrschen, erstellt werden. Aufgrund der Vielfalt der Notationen von Funktionshierarchien und Funktionsstrukturen sind diese nur anhand solcher Werkzeuge zu erfassen, die in der Lage sind, die gewünschte Notation zu implementieren. So können Funktionsstrukturen basierend auf der IDEF0 Notation beispielsweise mit AI0 WIN einfach gezeichnet, aber nur eingeschränkt in eine UML Notation überführt werden. Um diversen Anwendungsbereichen gerecht zu werden, sind oft Notationen in Mischformen bestehend aus mehreren Notationen in Verwendung. Für solche speziellen Fälle soll ein Modellierungswerkzeug die Möglichkeit bieten, benutzerdefinierte Notationen beispielsweise in Form von Schablonen zu implementieren. Dies wird in der Regel von allgemeinen Modellierungswerkzeugen unterstützt.

Zur Veranschaulichung und Einbindung in der Produktdokumentation können die gezeichneten Modelle normalerweise in gängige Bildformate gespeichert werden. Für die Interoperabilität mit anderen Anwendungen bietet ein Großteil der Modellierungswerkzeuge die Möglichkeit an, neben eigener auch standardisierte oder spezifizierte und leicht lesbare Dateiformate für die Speicherung der Zeichnung zu verwenden (vgl. Bild 4.31).

Wie im Kapitel 2.4.2 auf Seite 35 beschrieben, ist XML eine sehr verbreitete Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten und wird oft für den Austausch von Daten zwischen unterschiedlichen Systemen eingesetzt. Außerdem unterstützen die meisten Modellierungswerkzeuge das Importieren/Exportieren in ein auf XML basierendes Dateiformat. Deshalb wird im Rahmen dieser Arbeit davon ausgegangen, dass die erstellten Funktionsmodelle in einem auf XML basierenden Dateiformat zur Verfügung gestellt werden. Ist dies nicht den Fall, d.h. die eingesetzten Modellierungswerkzeuge unterstützen nur eigene spezifische Dateiformate, muss für diese Werkzeuge ein Programm zur Umsetzung des Eigenformats in einem XML-Format implementiert werden.

Die aus der Darstellung am Rechner erzeugten Daten liegen nun in XML vor. Die mit XML definierbaren Schemata sind spezifisch auf Programme und Anwendungsbereiche abgestimmt. Das Visio XML Schema (VXD Format) besteht beispielweise aus den übergeordneten Knoten Style, Masters, Pages, Shapes, Connection. Dieses Schema ist darstellungsorientiert und speichert Informationen über die Gestaltung des Dokuments wie bespielsweise die Form der verwendeten Grafiken, deren Positionen, Formatvorlagen und so weiter. Im Allgemeinen besitzen alle Visualisierungswerkzeuge

ein solches darstellungsorientiertes Format, da diese für die Gestaltung von Dokumenten entworfen wurden. Die mit Grafiken definierten Daten und somit Semantik sind zwar auch in dem Dateiformat enthalten, der Zugriff kann aber nur über die übergeordneten Darstellungsinformationen erfolgen. Im Gegensatz dazu speichern notationsspezifische Werkzeuge meistens nur die Semantik des Diagramms wie Objekte, Funktionen, deren Relationen und so weiter. Die Informationen zur Darstellung werden entweder verworfen, oder den notationsspezifischen Elementen untergeordnet, so dass ein semantikorientiertes Speicherformat entsteht.

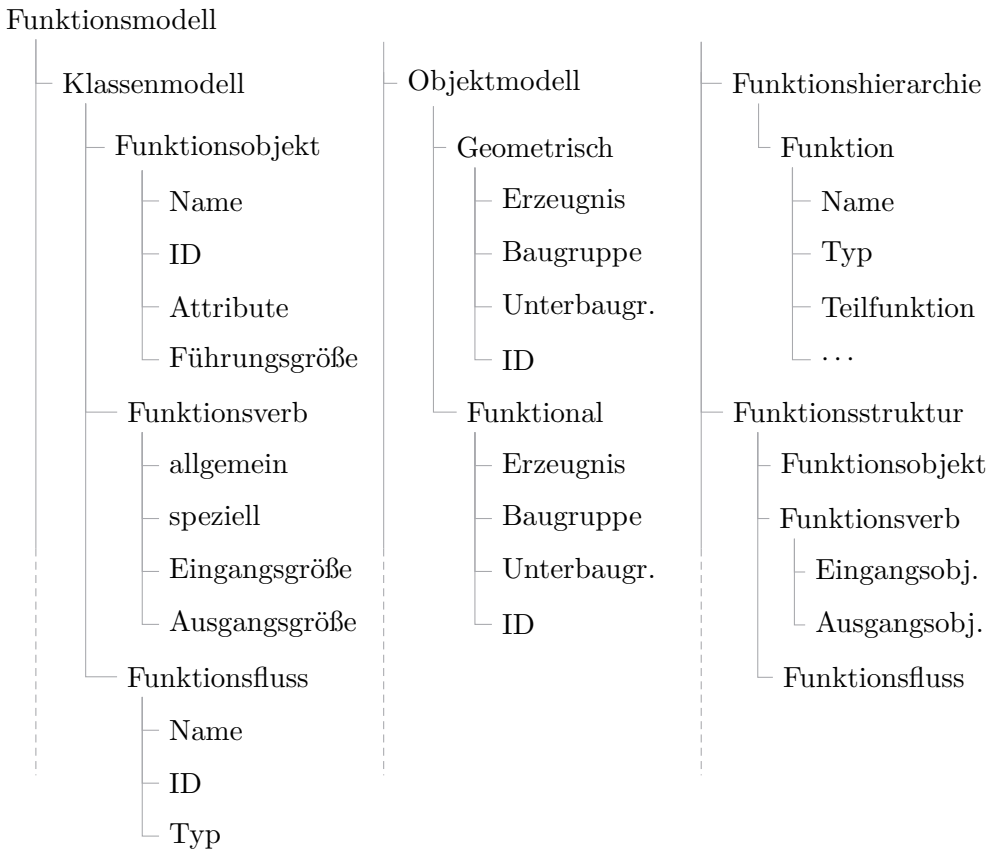


Bild 4.32: XML Struktur nach Filterung und Extraktion

Für die Einführung der Semantik eines Funktionsmodells in einen Szenengraph ist ein semantikorientiertes Speicherformat erforderlich. Ferner ist aufgrund der Vielfalt der in den Modellierungswerkzeugen verwendeten XML Derivaten eine einheitliche Schnittstelle für die Übermittlung der relevanten Modelldaten zwischen Funktionsmodell und virtueller Umgebung

notwendig. Eine einheitliche XML Struktur muss daher definiert werden und soll als Zwischensprache dienen. Bild 4.32 illustriert die Struktur die für eine solche Schnittstelle im Rahmen dieser Arbeit vorgeschlagen wird. Die Struktur ist semantikorientiert und semistrukturiert, d.h. dass sie sowohl für den Menschen verständlich, als auch für die maschinelle Verarbeitung geeignet ist. Sie enthält alle in Kapitel 4.2.1 und Kapitel 4.2.2 definierten Grundelemente die benötigt werden, um die Semantik eines Funktionsmodells vollständig darzustellen.

Je nach verwendeter Notationsregel oder Modellierungswerkzeug folgt eine Transformation des Ausgangsformats, so dass die Funktionsmodelle in dem beschriebenen, normalisierten Format gespeichert werden können. Unter der Annahme, dass das Ausgangsformat des Modellierungswerkzeugs eine XML basierte Sprache definiert, wird diese Transformation mit Hilfe von XSLT durchgeführt (vgl. Kapitel 2.4.2 auf Seite 36).

Nachdem die Daten in einem normalisierten, auf XML basierenden Format zur Verfügung stehen, können diese von andere Anwendungen, wie z.B. einer VR Entwicklungsumgebung, genutzt werden. Der Zugriff erfolgt dabei über der in den objektorientierten Sprachen implementierten API für XML Dateien (vgl. Kapitel 2.4.2 auf Seite 35). Somit erlaubt die Implementierung eines einzigen Plugins innerhalb der Zielumgebung die Unterstützung des Imports von Funktionsmodellen aus unterschiedlichen Modellierungswerkzeugen. Ohne Transformation des Ausgangsformats in dem festgelegten Zwischenformat würde die Unterstützung eines neuen Modellierungswerkzeugs die Implementierung eines neuen Plugins für jede unterstützte Zielumgebung bedeuten.

Darüber hinaus lässt sich definierte Struktur ebenfalls in Form einer relationalen Datenbank implementieren. Die einheitliche Schnittstelle zwischen Modellierungswerkzeuge und virtueller Umgebung wäre dann durch das entsprechende Datenbankschema realisiert. Allerdings würde dies die Inbetriebnahme eines Datenbankenverwaltungssystems erfordern. Somit müsste die eingesetzte Transformationsmethode XSLT durch ein komplexes, spezifisches Programm für den Import von Daten in der Datenbank ersetzt werden (vgl. Kapitel 2.4.1). Im Falle von großen Datenbeständen ist der Einsatz von Datenbanken jedoch geeigneter als der von XML Dateien, insbesondere vor dem Hintergrund der schlechten Skalierbarkeit von XML Parser und der ineffizienten Abfrage und Transformation umfangreicher Datenbestände.

Am Beispiel des RS232 Switches sind die Modelle der Funktionshierarchie (vgl. Bild 4.22) und der Funktionsstruktur (vgl. Bild 4.30 bereits erstellt

worden. Ein Klassenmodell beeinhaltet die Klassen Funktionsobjekt, Funktionsverb und Funktionsfluss jeweils in Abhängigkeit zueinander. Bild 4.33 stellt auszugsweise ein abstraktes Klassenmodell der Funktionsobjekte dar.

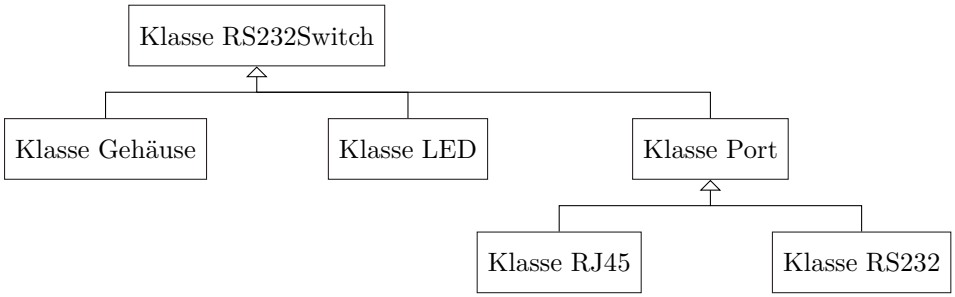


Bild 4.33: Klassenmodell der Funktionsobjekte des RS232 Switches

Das Objektmodell beinhaltet die verschiedenen Sichten auf Basis unterschiedlicher Kriterien, die in Kapitel 4.3.1 genauer erklärt werden.

Objekte	Geometrische Sicht (Produktstruktur)	Funktionale Sicht (Komponenten sichern)	Funktionale Sicht (Status signalisieren)
1-RS232Switch	X		
1.1-Gehäuse	X	X	
1.1.1-Schrauben	X	X	
1.2-LED	X		X
1.3-Port	X		
1.3.1-RS232	X		
1.3.2-RJ45	X		
1.3.1.1-Prozessor	X		X
1.3.1.2-Datenswitch	X		
1.4-SchalterA	X		
1.4.1-Transceiver	X		X
1.5-SchalterB	X		

Tabelle 4.5: Filterung von funktionalen und geometrischen Sichten

Dem entsprechend illustriert Tabelle 4.5 neben der Produktsicht eine exemplarische Auswahl der Objekte, die für die funktionale Sicht *Status signalisieren* und *Komponenenten Sichern* notwendig sind.

4.2.4 Mapping des Funktionsmodells

Nachdem das Funktionsmodell erstellt (vgl. Kapitel 4.2.1 auf Seite 118 und 4.2.2 auf Seite 126), am Rechner dargestellt und die notwendigen Daten aus dem Ausgangsformat des Modellierungswerkzeugs extrahiert und in einem normalisierten Format zusammengefasst wurden (vgl. Kapitel 4.2.3 auf Seite 138), erfolgt in diesem Schritt, die Abbildung zwischen den Elementen des Funktionsmodells und den Elementen des Szenengraphs, so dass eine Zuordnung der Daten auf den Szenengraph stattfinden kann. Hierfür ist eine Ermittlung der Daten aus dem Szenengraph erforderlich, so dass die abzubildenden Elemente ausgewählt werden können.

Bevor die Daten ermittelt werden können, muss festgelegt werden, welche Daten abzubilden sind und in welcher Weise dies zu erfolgen hat. Abbildung 4.34 illustriert die Datenstrukturen aus dem Funktionsmodell und dem Szenengraph, sowie deren Zusammenhänge. Informationen, die sowohl im Funktionsmodell als auch im Szenengraph enthalten sind, können bidirektional abgebildet werden. In diesem Fall sind das die 3D Objekte, deren Eigenschaften und deren Hierarchie. Die Funktionsflüsse sowie die Funktionsverben und die Funktionshierarchie werden zusätzlich aus dem Funktionsmodell in den Szenengraph importiert. Hier können Informationen wie die der Funktionsverben im Rahmen physikalischer Simulationsverfahren zur Spezifikation des Verhaltens der Objekte verwendet werden. Da dies nicht im Rahmen der hier vorliegenden Arbeit betrachtet werden soll, ein Bezug durch die 3D Objekte jedoch exisitiert, werden diese Informationen nur unidirektional abgebildet.

Die Abbildung der Objekte aus dem Funktionsmodell auf Objekte der Szene kann im wesentlichen mittels drei verschiedenen Methoden erfolgen.

Erstens kann durch die Abbildung der Objekthierarchie des Funktionsmodells auf die in der Form von Gruppen in der Szene existierende Hierarchie eine Abbildung der Objekte automatisch abgeleitet werden. Der Wurzelknoten der Objekthierarchie wird auf die größte umfassende Gruppe abgebildet. Daraus werden alle Kinderknoten der Wurzel automatisch auf die Untergruppen der umfassenden Gruppe abgebildet, basierend auf Informationen wie beispielsweise die Anzahl der Kinderknoten oder die Tiefe des unteren Teilbaums. Rekursiv werden mit dem gleichen Verfahren alle Kinderknoten und Untergruppen abgebildet. Dies erfordert aber das Vorhandensein einer vollständigen Objekthierarchie in dem Funktionsmodell und eines kohärenten Zusammenhangs (1-zu-1 Beziehung) zwischen der Objekthierarchie des

Funktionsmodell und der Objekthierarchie der Szene.

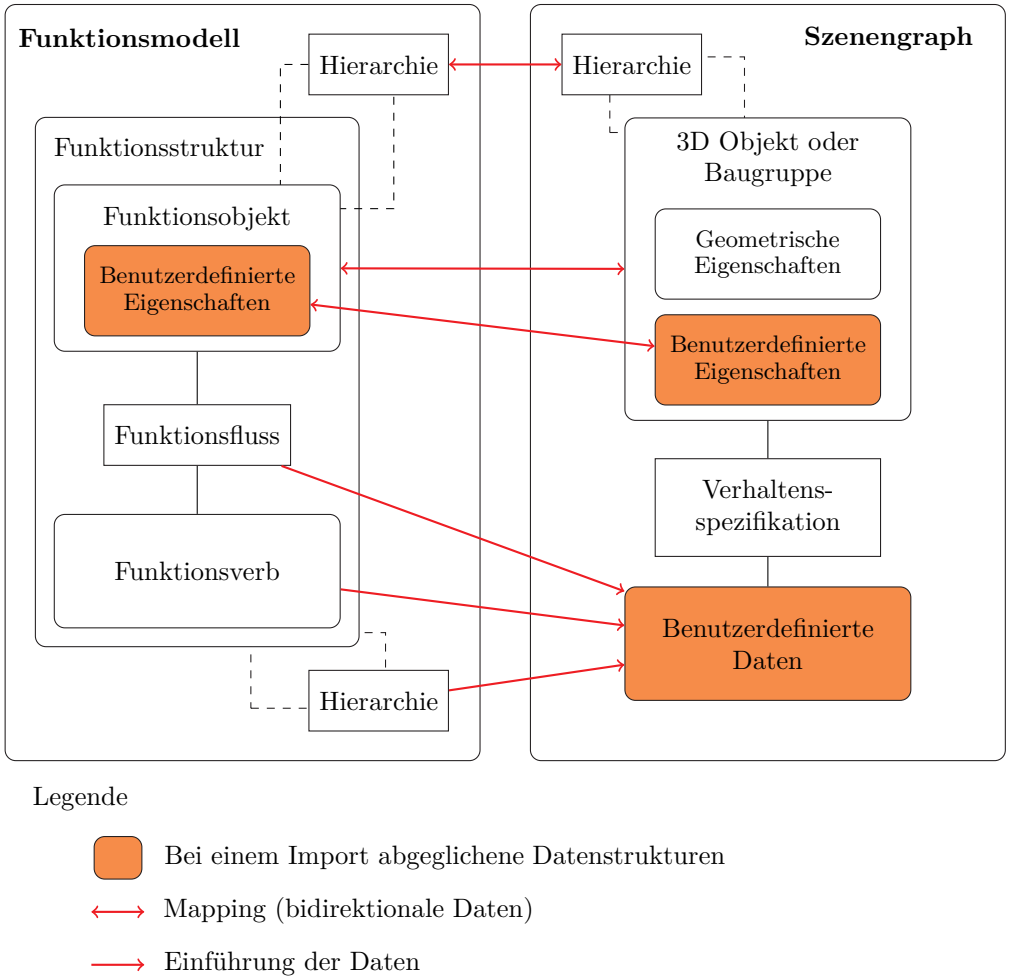


Bild 4.34: Konzeptionelles Mapping von Funktionsmodell auf Szenengraph

Zweitens kann die Abbildung automatisch aufgrund der Namen der Objekte vorgenommen werden. Objekte aus der Objekthierarchie des Funktionsmodells werden auf Objekte oder Gruppen der Szene abgebildet, die gleiche Namen besitzen. Dafür sollen die gegebenen Objektnamen eindeutig sein und konsistent zwischen Funktionsmodell und Szene gehalten werden. Um die strenge Gleichheit der Namen zu umgehen, können adaptivere Lösungen eingesetzt werden, wie die Betrachtung der Namenskomposition (zwei Objekte werden abgebildet, wenn der Name des Ersten in dem Namen des Zweiten enthalten ist) oder die Berechnung von Editierdistanzen (zwei Objekte werden abgebildet, wenn die Namen von beiden die minimale Anzahl

der Operationen Einfügen, Löschen und Ersetzen von Buchstaben minimieren (Levenshtein, 1966)). Dieses Verfahren erfordert eine Beziehung zwischen der Namensgebung in dem Funktionsmodell und in der Szene, und widerspricht dem in der Einführung postuliertem Prinzip der parallelen und unabhängigen Entwicklung von Funktionsmodellen und Szenengraph.

Drittens kann eine manuelle Abbildung durch Eingabe einer Abbildungsmatrix über eine grafische Benutzeroberfläche realisiert werden. Somit bleiben die Benennung der Objekte sowie die Vollständigkeit und Konsistenz der Objekthierarchie ohne Einfluss auf die Abbildung von Funktionsmodelle und Szene und können unabhängig voneinander bearbeitet werden.

Das erste und zweite Verfahren zur Automatisierung kann genau dann eingesetzt werden, wenn der Benutzer Unterstützung in Form von zusätzlichen Angaben zu Abbildungen bei der Vervollständigung der Abbildungsmatrix braucht. Durch die beiden ersten Verfahren ergibt sich eine semiautomatisierte und flexible Methode für die Abbildung von Objekten. Bei der Abbildung zwischen den Funktionsobjekten des Funktionsmodells und den 3D Objekten oder der Gruppierung dieser im Szenengraph muss beachtet werden, dass das gleiche Objekt mehrmals in einem Funktionsmodell dargestellt werden kann, um die Lesbarkeit und Eindeutigkeit der Flüsse zu gewährleisten. Somit ist die Beziehung zwischen Objekten des Funktionsmodells und Objekten des Szenengraphs keine eindeutige 1-zu-1 Beziehung, sondern eine mehrdeutige n-zu-1 Beziehung.

Neben der Abbildung der Objekte sollen ebenfalls die Eigenschaften dieser gemappt werden. Da die sich in der Szene bereits befindenden Eigenschaften im Allgemeinen nur geometrische Eigenschaften des Objekts beschreiben, existieren keine Konflikte zwischen Parameter des Funktionsmodells und Parameter der Szene. Außerdem können die aus dem Funktionsmodell importierten Eigenschaften durch Typisierung unterschieden werden, so dass eine Trennung zwischen geometrische Eigenschaften und funktionalen Eigenschaften besteht. Aufgrund dieser Beseitigung eventueller Konflikte kann die Abbildung der Eigenschaften von Objekten automatisch erfolgen und erfordert keine manuelle Eingabe.

Inkonsistenzen zwischen verschiedenen Darstellungen eines Objekts in dem Funktionsmodell, wie z.B. unterschiedliche Werte desselben Parameters, führen zu Konflikten bei der Abbildung der Parameter. Diese Konflikte können nicht rechnerisch gelöst werden und führen zu einem Datenverlust. Daher ist eine Konsistenzprüfung bei der Funktionsmodellierung oder spä-

testens bei der Extraktion der Daten aus dem Funktionsmodell (vgl. Kapitel 4.2.3) notwendig.

Das Bild 4.35 beschreibt in Form eines Sequenzdiagramms das gesamte Vorgehen bei der Abbildung zwischen Funktionsmodell und Szenengraph. Zunächst holt die Benutzeroberfäche die Liste der Objekte aus der Funktionsmodellierung und die Liste der Objekte aus der Szene. Basierend auf diese beiden Listen kann der Benutzer eine Zuordnung der Objekte durch das Ausfüllen der Abbildungsmatrix durchführen. Sobald die Objekte abgebildet sind, werden deren Eigenschaften aus dem Funktionsmodell und aus der Szene entnommen und automatisch auf Grund der Namensgebung und Typen abgebildet.

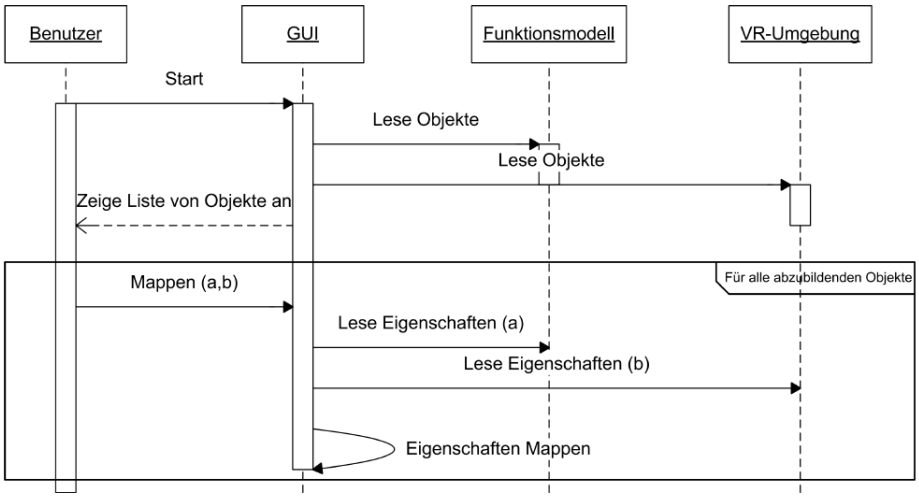


Bild 4.35: Sequenzdiagramm der Abbildung zwischen Funktionsmodell und Szenegraph

4.2.5 Import des Funktionsmodells

Nach dem Erstellen der Funktionshierarchie und der Funktionsstruktur, wurden diese mit Hilfe der Modellierungswerkzeuge in ein XML Format gespeichert und durch XSLT in eine festgelegte und einheitliche XML Struktur transformiert. Basierend auf der normalisierten XML-Struktur-Schnittstelle wurde eine Abbildung zwischen Elementen des Funktionsmodells und den entsprechenden Elementen der 3D Szene ermöglicht. Dafür wurden die abbildbaren Elemente aufgelistet und entsprechende Vorgehensweisen für das

Mapping vorgeschlagen. Bild 4.36 fasst das gesamte Vorgehen zum Mapping der Funktionsmodelle auf einen Szenengraphen zusammen und illustriert die für jeden beschriebenen Schritt einsetzbaren Werkzeuge und Artefakte.

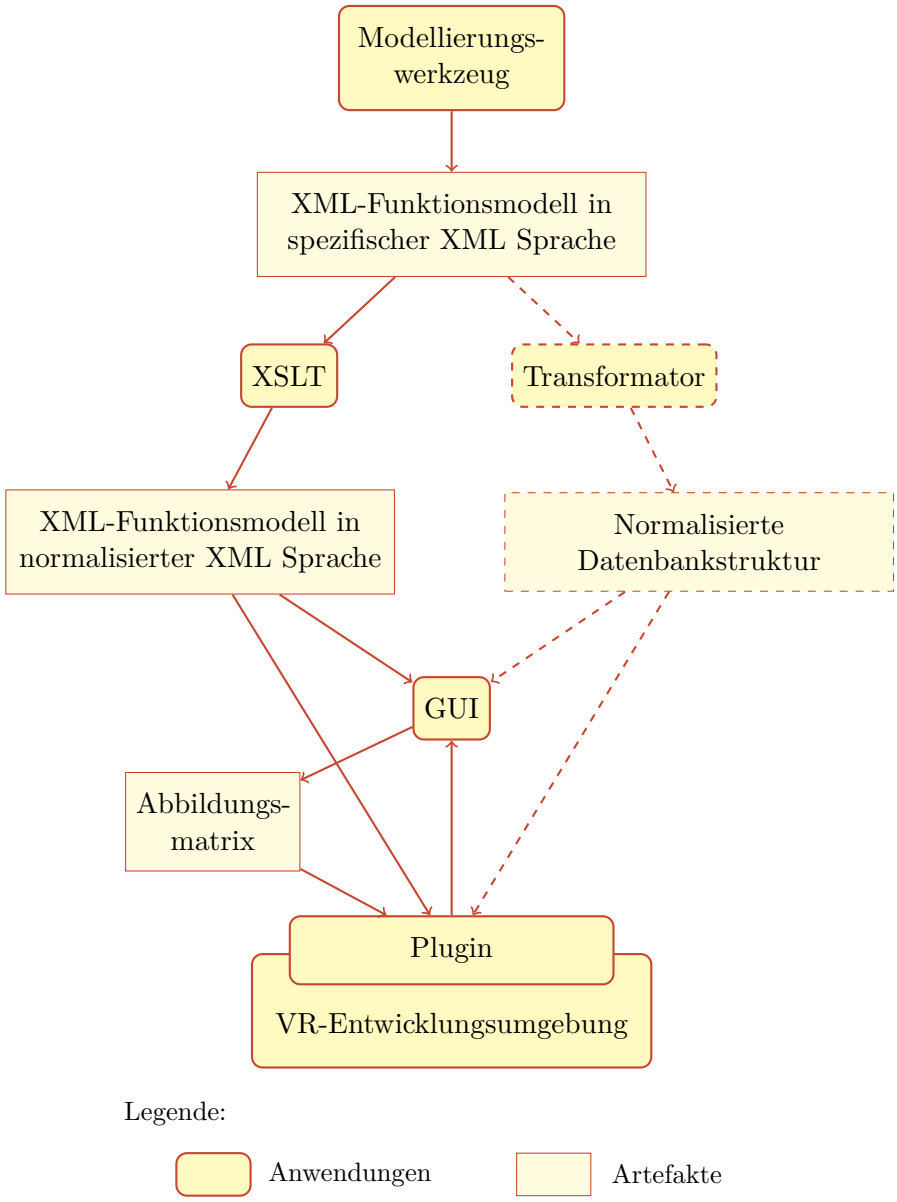


Bild 4.36: Mapping von Funktionsmodellen in virtuelle Umgebungen

Das Importieren von Daten in die VR Entwicklungsumgebung ist von den in der Umgebung angebotenen Mechanismen zur Verwaltung von Daten in der 3D Szene abhängig und benötigt daher jeweils eine spezifische Imple-

mentierung. Dabei sind im Rahmen der Übertragung der Funktionsmodelle die benutzerdefinierten Eigenschaften für Objekte, die Funktionshierarchie und die Funktionsstrukturen zu berücksichtigen. Diese wurden bereits in Bild 4.34 farblich hervorgehoben dargestellt.

Im Allgemeinen bietet eine VR Entwicklungsumgebung die Möglichkeit, eigene Eigenschaften an 3D Objekte einzubinden (vgl. Kapitel 2.6.3 auf Seite 65). Diese zusätzlichen Daten müssen meistens in generischen Datenbeständen oder in von der VR Entwicklungsumgebung vorgegebenen, spezifischen Strukturen abgelegt werden.

Da VR Entwicklungsumgebungen im Standardfall für das Zusammenstellen der 3D Szene nur den Import von Multimediadaten wie 3D Modelle, Töne, Bilder und Videos unterstützen, muss die Einbindung zusätzlicher Daten direkt über die Render Engine (vgl. Bild 4.10) oder einem SDK erfolgen, um die Funktionalitäten zum Import eigendefinierter Daten und Formate zu implementieren. Das SDK stellt dabei eine API zur Verfügung, die einen Zugriff auf die Geometrie der Szene, auf die Verhaltensspezifikationen und auf die spezifische Datenstruktur zur Speicherung eigener Daten erlaubt. Somit können neue, eigene Stukturen in dem Datenbereich der VR Entwicklungsumgebung erzeugt und mit den Daten aus dem Funktionsmodell befüllt werden. Wegen der Vielfalt und der Spezifität der Datenstrukturen, kann hierfür kein allgemeiner Ansatz eruiert werden. Daher muss jede zu unterstützende, virtuelle Umgebung bezüglich ihrer Mechanismen zur Datenlagerung und zur Verhaltenspezifikation analysiert werden, um eine optimale Strukturierung der importierten Daten zu erzeugen.

Für den Fall, dass kein Datenbereich in der VR Entwicklungsumgebung vorgesehen ist, können mit Hilfe des SDK neue, eigene Verhaltensspezifikationsbausteine entwickelt werden, die lediglich auf die im Schnittstellenformat (entweder in der XML Datei oder in einer Datenbank, siehe Bild 4.36) erfassten Daten zugreifen, und diese den anderen Bausteinen zur Verfügung stellen. Somit können die Verhaltensspezifikationen zur Interaktion in der Szene auf die Daten aus dem Funktionsmodell zugreifen, ohne dass ein Importieren der Daten erforderlich ist. Dieser Ansatz hat den Vorteil, die Schwierigkeiten eines Imports zu umgehen, aber erfordert die Implementierung von komplexen Bausteinen für den Zugriff auf die Daten. Außerdem sind die Funktionsmodelldaten in diesem Fall kein Bestandteil der Szene, sondern als separate Datei vorhanden, was ein Austausch der Daten komplexer macht und eine Kopplung von Funktionsmodell und Szenengraph nur teilweise abdeckt.

4.3 Interaktive Visualisierung des Funktionsmodells

Die Darstellung der Funktionsmodelle in einer virtuellen Umgebung soll sich nicht auf eine pure Informationsvisualisierung beschränken, sondern das Potential der VR Technologien darüber hinaus ausschöpfen. Dazu soll eine möglichst realitätsnahe, virtuelle Umgebung, entsprechend den Merkmalen der virtuellen Realität, wie sie in Bild 2.12 auf Seite 54 diskutiert wurden, erzeugt werden (vgl. Kapitel 4.1), die mittels Methoden der Informationsvisualisierung (vgl. Kapitel 2.5.3 auf Seite 41), um verwandte, abstrakte Informationen angereichert wird. Tabelle 3.2 auf Seite 92 fasst die Anforderungen präzise zusammen.

Die virtuelle Umgebung wird als vollständig synthetisch und somit sehr flexibel betrachtet, so dass die dargestellten Objekte über die Physis hinaus reichen können. Die nahtlose Einbettung der Informationsobjekte spielt dabei eine entscheidene Rolle, ebenso wie die Übernahme aller vorhandenen Eigenschaften wie Selektierbarkeit, Verschiebbarkeit, Skalierbarkeit und Manipulation durch den Anwender.

Dabei kann generell die Anwendung von VR Technologien in zwei verschiedene Kategorien, die sich nach ihrer Nutzung klassifizieren lassen, unterschieden werden. Zum Einen kann abstrakte Information dazu verwendet werden, die virtuelle Umgebung zu kontrollieren. Beispielsweise können Objekte, durch Selektion in einem separaten, abstrakten Display, in der virtuellen Umgebung hervorgehoben werden, oder uninteressante Objekte ausgeblendet werden. Zum Anderen kann die virtuelle Umgebung als Zugang zu abstrakten Informationen dienen.

Um dies zu erreichen, werden im Rahmen dieser Arbeit auf Basis von Funktionsmerkmalen Sichten definiert, Interaktionsmetaphern verwendet, die einen intuitiven Informationsaustausch anhand der Informationsvisualisierungmethoden zwischen virtueller Umgebung und Anwender ermöglichen und eine reversible, sichtenorientierte topologische Neuanordnung der 3D Objekte in der virtuellen Umgebung, entsprechend den Produktfunktionen, entwickelt.

Zur Visualierung der Funktionsmodelle werden folgende Schritte durchgeführt. Erstens müssen die erhaltenen Daten aufbereitet werden. Dazu werden anhand der Funktionsmerkmale Sichten generiert, die dem Anwender fließende Übergänge zwischen den funktionalen und anderen Produktzusam-

menhängen visualisieren. Zweitens werden die Interaktionsmetaphern beschrieben, die die Interaktion zwischen Anwender und virtueller Umgebung ermöglichen. Drittens wird die geometrische und topologische Darstellung der Produktfunktionen in der virtuellen Umgebung diskutiert. Bild 4.37 illustriert die Schritte, wobei die gelb markierten Rechtecke die in diesem Kapitel beschriebenen Bausteine repräsentieren, während die weiss gefüllten Rechtecke, Prozesse, die in anderen Kapiteln diskutiert wurden, darstellen.

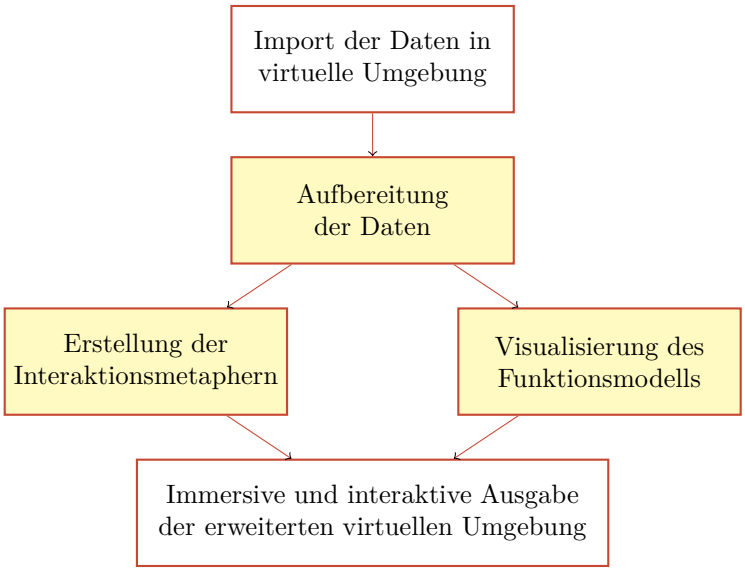


Bild 4.37: Schritte zur Visualisierung des Funktionsmodells

4.3.1 Aufbereitung der erzeugten Daten

Die Zielstellung der Arbeit ist die Abbildung von Funktionsmodellen in virtuellen Umgebungen. Da die Funktionsmodelle, respektive die Funktionshierarchien und Funktionsstrukturen interdisziplinärer Produkte sehr komplex werden können, wird im Rahmen dieser Arbeit eine Gruppierung von Funktionen vorgeschlagen. Dabei soll die Gruppierung derart erfolgen, das die Funktionen in Form von funktionalen Sichten abgebildet werden. Diese gestalten die Navigation innerhalb der virtuellen Umgebung verständlicher und intuitiver, da die Menge der dargestellten Objekte immer überschaubar bleibt. Um die Sichten klassifizierbar zu machen, werden so genannte Funktionsmerkmale eingeführt, anhand derer der Fokus jeder einzelnenen Sicht festgelegt werden soll. Die hierbei notwendige Vorgehensweise wird

in Bild 4.38 schematisch dargestellt und soll anschließend näher erläutert werden.

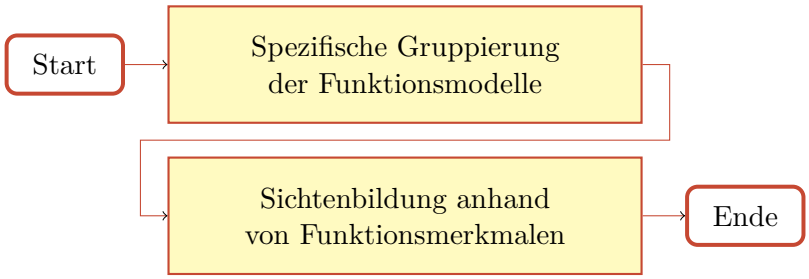


Bild 4.38: Aufbereitung des Funktionsmodells

Zuerst werden die möglichen Arten der Grupierungen untersucht. Dabei werden zwei unterschiedliche Vorgehensweisen in Betracht gezogen, die für diese Arbeit relevant sind. Anschließend sollen die Funktionsmerkmale als klassifizierende Elemente eingeführt werden, auf deren Basis die funktionalen Sichten erzeugt werden, die für eine übersichtliche, intuitive Interaktion in der virtuelle Umgebung notwendig sind.

Spezifische Gruppierung der Funktionsmodelle

Für eine flexible Sichtenbildung müssen die Funktionsmodelle adaptiv gruppierbar sein. Dafür sind zwei Ansätze denkbar.

Ein möglicher Ansatz besteht in einer Top-Down Vorgehensweise, bei der die Funktionen ausgehend von einer Gesamtfunktion in Teilfunktionen, Elementarfunktion und unerwünschte Funktionen gruppiert werden. Diese Vorgehensweise ist überaus intuitiv, da sie auch bei der Erstellung der Produktstruktur angewandt wird und damit dem Konstrukteur vertraut ist. Um die Sichten festzulegen, werden über die Positions-ID die an einer Produktfunktion beteiligten Elemente der Funktionsstruktur entsprechend der Klassifizierung aus Kapitel 4.2.2 auf Seite 133 eindeutig zugewiesen. Allerdings entsteht daraus nur eine bauteilorientierte Sicht. Bild 4.39 zeigt die allgemeingültige Umsetzung dieses Ansatzes. Dabei liegt die Produktstruktur der Funktionshierarchie zugrunde. Die Funktionshierarchie wiederum ist mit der Funktionsstruktur über die Ordnungselemente in der Fußzeile der Funktionsobjekte verbunden. Jedes Funktionsobjekt existiert in einer Funktionsstruktur genau einmal und ist durch seine Kenngrößen eindeutig

identifizierbar. Die Zuordnung von Baugruppen-, Unterbaugruppen- oder Bauteilbezeichnungen wurde bereits in Kapitel 4.2.2 auf Seite 133 diskutiert.

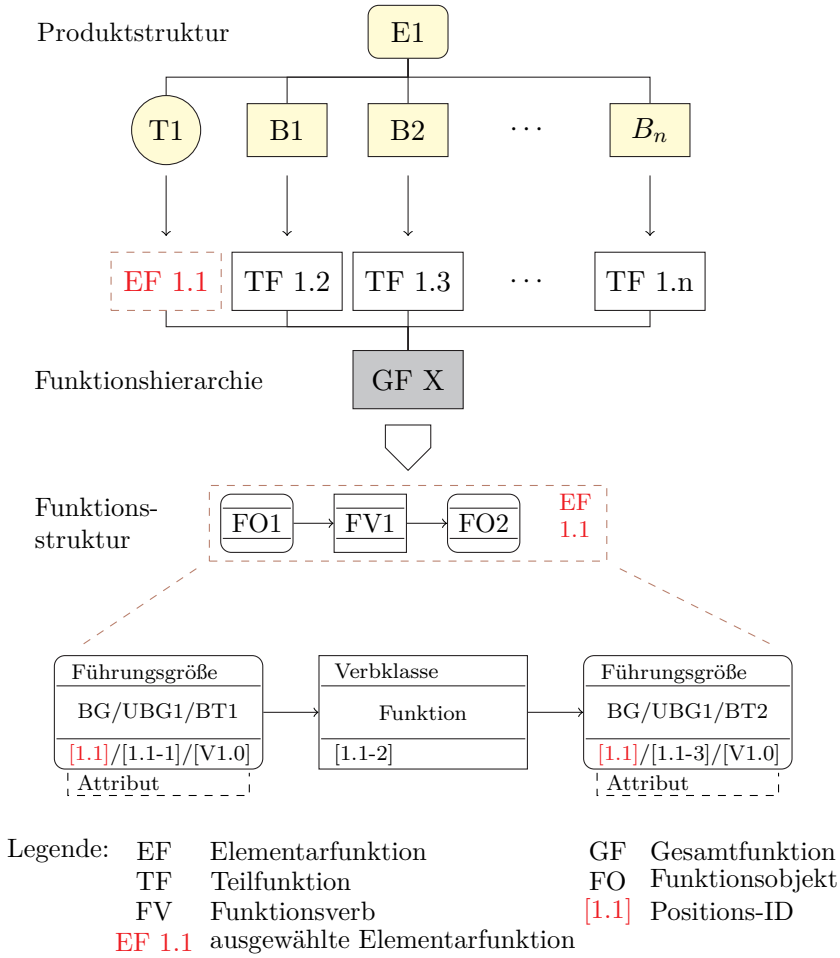


Bild 4.39: Sichtenbildung anhand der Top-Down Methode

Ein zweiter Ansatz stellt die Bottom-Up Vorgehensweise dar, die die Funktionen auf Basis beliebiger Merkmale gruppiert, und somit eine freie Sichtengenerierung ermöglicht. Die beliebigen Funktionsgruppierungen dienen dazu die Komplexität durch vom Anwender gewünschte Einschränkungen zu verringern und somit den Gesamtüberblick zu erhalten. Die Gruppierungskriterien können beliebig gewählt werden und sollen im Rahmen dieser Arbeit als Funktionsmerkmale bezeichnet werden. Kapitel 4.3.1 auf Seite 155 diskutiert diese detailliert. Bild 4.40 beschreibt die Vorgehensweise bei der

Bottom-Up Methode. Auch hier können die Funktionen aus der Produktstruktur abgeleitet werden. Jedoch wird die Anordnung der Teilfunktionen in der Funktionshierarchie nach der Auswahl entsprechend der Funktionsstruktur getroffen.

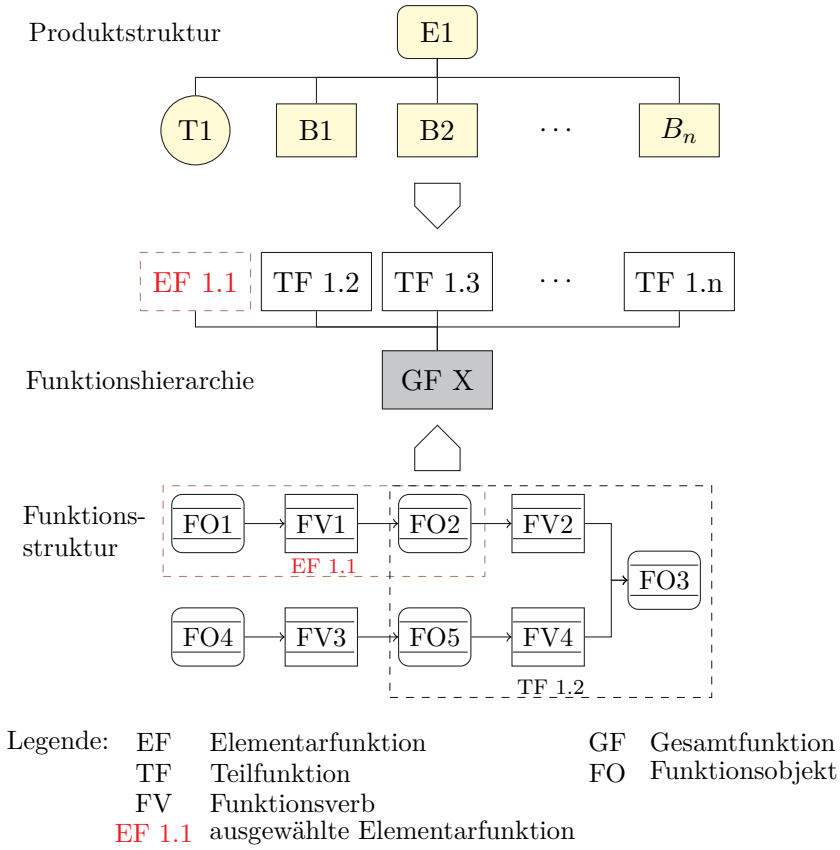


Bild 4.40: Sichtenbildung anhand der Bottom-Up Methode

Zur dynamischen Generierung von Sichten sollen Funktionsmerkmale verwendet werden. Der Vorteil dieser Klassifizierung liegt im Wesentlichen in der Verringerung der Komplexität der Produktvisualisierung als auch in der Adaptivität des Gezeigten, passend zum Präsentationszweck. Darüber hinaus kann mittels der dynamischen Sichtenbildung der Abstraktionsgrad der Darstellung skaliert werden. Je nach Detaillierungsgrad des Funktionsmodells sowie des ausgewählten Funktionsmerkmals können so beispielsweise firmenübergreifende Spezifikationen weitergegeben bzw. erfüllt werden, ohne dabei das inhärente Wissen eines Wirkprinzips einer Funktion offenzulegen.

Sichtenbildung anhand von Funktionsmerkmalen

Funktionsmerkmale spielen hierbei eine entscheidende Rolle, indem sie den Fokus der Sicht festlegen. Bei der Darstellung der Funktionshierarchie, wie sie in Kapitel 4.2.1 und Bild 4.39 erfolgte, wurde als Funktionsmerkmal eine bauteilorientierte Sicht zugrunde gelegt. Neben dem Funktionsmerkmal Bauteilorientierung ist die Klassifizierung auch nach Kosten von Funktionen, Qualität der Funktion, Wiederverwendbarkeit, Herstellbarkeit, Produktaufgaben etc. vorstellbar.

Die Top-Down Vorgehensweise zur Erzeugung der Funktionshierarchie, in der Funktionen in Teilfunktionen aufgegliedert werden, ist nicht geeignet unterschiedliche Funktionsmerkmale zu berücksichtigen, sondern betrachtet lediglich das bauteilorientierte Funktionsmerkmal. Die Bottom-Up Vorgehensweise bietet hierfür größere Flexibilität. In dieser werden die Produktfunktionen in Funktionsgruppen zusammengefasst, die dazu dienen, den Überblick über die Funktionsstruktur zu erleichtern, um die Komplexität beherrschen zu können. Mit jeder Funktionsgruppe können die verschiedenen Sichten der Funktionsstruktur, entsprechend des fokussierten Funktionsmerkmals, erstellt werden. Beispielsweise lassen sich Funktionen nach den Aufgaben, die sie erfüllen, zusammenfassen. Dem entsprechend könnten bei einem PKW Funktionen zusammengefaßt werden, die der Kraftstoffversorgung, dem Antrieb, der Heizung, dem Innenraum, der Sicherheit etc. dienen.

Bild 4.41 zeigt den Aufbau verschiedener Sichten auf Basis einer Funktionsstruktur desselben Gesamtprodukts sortiert nach unterschiedlichen Funktionsmerkmalen. Im Falle des Funktionsmerkmals A wird die Funktionshierarchie aus denjenigen Funktionen erstellt, die in der jeweiligen Sicht notwendig sind. Dabei werden die Funktionen aus den Funktionsobjekten und Funktionsverben der Funktionsstruktur gruppiert. Die Produktstruktur kann weiterhin der Identifikation der Produktfunktionen dienen, wird aber bei der Bottom-Up Methode zur kohärenten Abbildung der Bauteile auf die Funktionsobjekte genutzt, da die Hierarchisierung bereits aus der Funktionsstruktur heraus geschieht. Der untere Teil von Bild 4.41 zeigt die Abbildung der Produktstruktur auf die Funktionsobjekte in der Funktionsstruktur, wie dies auch in Kapitel 4.2.2 auf Seite 133 methodisch erarbeitet wurde. Der Ausschnitt A1.1 findet sich in der Funktionshierarchie der Sicht A wieder.

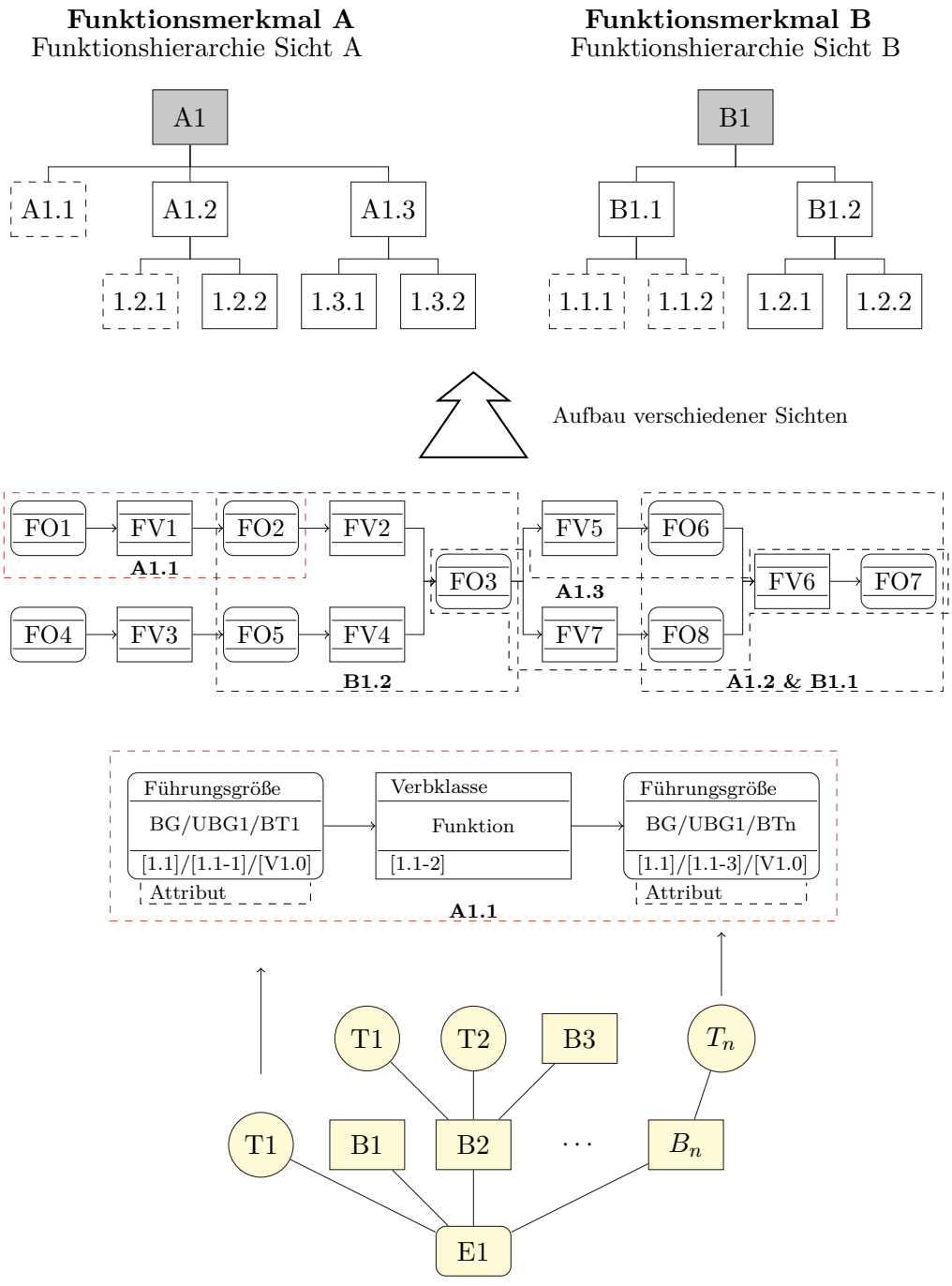


Bild 4.41: Aufbau verschiedener Sichten

4.3.2 Metaphern für die Interaktion

Ein wesentlicher Aspekt der vorliegenden Arbeit behandelt neue Methoden zur Verbesserung der Anwendungsmöglichkeiten der vorhandenen VR Technologien auf Basis von Interaktionsmetaphern.

Die Aussagekraft der Interaktionsmetaphern hängt letztendlich von der zur Verfügung stehenden Hardware ab. Je präziser ein Eingabegerät dreidimensionale Objekte in einer virtuellen Umgebung manipulieren kann, desto verständlicher wird die verwendete Metapher. Generell gilt, dass eine virtuelle Szene immer nach Interaktion verlangt. Wenn eine virtuelle Szene den Anspruch hat hoch immersiv zu sein, dann bedingt dies gleichzeitig einen hohen Grad an Interaktion. Das immersive Erlebnis wird umso stärker, je mehr die virtuelle Umgebung auf Aktionen des Anwenders reagieren kann und somit die Interaktion zwischen Realität und Virtualität vertieft wird. Dabei stellen Interaktionsmetaphern eine Interaktionsform dar, die komplexe Zusammenhänge in einfache, leicht Verständlichere überträgt. Sie werden bei der Erstellung von virtuellen Umgebungen in VR Entwicklungsumgebungen (vgl. Kapitel 2.6.3 auf Seite 65) auf Objekte einer Szene implementiert, die später, während der Präsentation an einer stereoskopischen Wand, die Kommunikation zwischen Mensch und Maschine ermöglichen.

Die Interaktionsmetapher beschreibt eine thematisch verbundene Gruppe von Elementen mit symbolischen Bedeutungen oder Repräsentationen mit Assoziationen zu anderen Kontexten oder unterstelltem Fachwissen. Dabei soll die Interaktion möglichst natürlich und selbsterklärend erfolgen. Die meistverbreiteste Interaktionsmetapher ist die Desktopmetapher, die alle gegenwärtigen Betriebssysteme nutzen. Hier werden als Eingabegeräte Tastatur und Maus verwendet. Zur Ausgabe dient ein Bildschirm. Die Interaktion zwischen Mensch und Computer findet mittels der Ein- und Ausgabegeräte auf dem virtuellen Schreibtisch, dem Desktop, statt (Svoboda and Ravasio, 2003).

Da der Anspruch dieser Arbeit darin liegt, die Potentiale der VR Technologien durch neuartige Interaktionsmetaphern weiter auszuschöpfen, mussten die konventionellen, gewohnten Denkwege verlassen werden. Dementsprechend wurden zwei Arten von Interaktionsmetaphern betrachtet. Zum einen egozentrische Metaphern, bei denen der Anwender sich während der Benutzung innerhalb der Welt befindet und die Dinge um sich herum betrachtet und verändert und zum anderen exozentrische Metaphern. Hier hat der Benutzer die gesamte virtuelle Umgebung im Blickfeld und kann diese von

mehreren Seiten betrachten, d.h. dass er sich außerhalb dieser Welt befindet. Durch die Kopplung der Interaktionsmetaphern mit den Methoden der Informationsvisualisierung entstanden die hier entwickelten Interaktionsmetaphern.

Interaktionsmetapher Avarworm

Vor dem Hintergrund desktopgebundener Ansätze, denen konventionelle Eingabegeräte zugrunde liegen, benötigt die Entwicklung andersartiger Interaktionsmetaphern, auf Basis neuer Ein- und Ausgabegeräte, neuartige Denkansätze, die die Vorteile von 3D Eingabegeräten wie beispielsweise Tracking oder Wiimote entsprechend ausnutzen. Gerade im Zusammenspiel mit stereoskopischen Projektionswänden sind solche Eingabemöglichkeiten anzustreben, da sie ein freies Bewegen im Raum ermöglichen und somit die Immersion fördern.

3D Eingabegeräte benötigen in der virtuellen Welt stets eine Repräsentation auf die sie wirken. In der Regel sind dies entweder Kameras, die so eine egozentrische Perspektive ermöglichen, oder Avatare, die eine dreidimensionale Repäsentation des Anwenders darstellen und dementsprechend eine exozentrische Ansicht erzeugen (vgl. Kapitel 3.2.2 auf Seite 84).

Im Rahmen dieser Arbeit wurden Versuche mit primitiven Körpern, wie Bällen oder Würfeln durchgeführt, um das Bedienen mit einem getrackten Objekt zu testen. Dabei wurde festgestellt, dass bei getrackten Gegenständen meist Probleme beim Erkennen der aktuellen Bewegungsrichtung und der aktuellen Entfernung von anderen Körpern auftreten, so dass es schwierig war, gezielt Kollisionen hervorzurufen. Sowohl Bewegungsrichtung als auch Entfernung sind allerdings wichtige Daten, die dem Nutzer helfen, sich in der virtuellen Umgebung zurechtzufinden. So ist es als Benutzer eines Trackingsystems wichtig zu wissen, in welche Richtung (relativ zum angestrebten Ziel) sich der Avatar in der virtuellen Umgebung bewegt, um beispielsweise eine Berührung mit vorhandenen, anderen virtuellen Objekten gezielt zu gewährleisten. Genauso wichtig ist die Information, wie weit entfernt sich der Avatar vom Ziel befindet. Insbesondere bei der Verwendung stereoskopischer Projektionen, wie Powerwalls, oder CAVEs ist es notwendig, die Entfernung zwischen Avatar und Objekt einschätzen zu können, da eine Bewegung durchaus auch physisch stattfinden kann. (Brenzinger, 2007)

Aus diesen Überlegungen und Vorversuchen resultierte die Idee zu einer wurmartigen Gestalt, der im Rahmen dieser Arbeit als Avarworm bezeich-

net wird. Dieser besteht aus einer Reihe von Bällen, die nach hinten immer kleinere Durchmesser besitzen. Gesteuert wird dabei nur der erste Ball, der als Avarball bezeichnet wird, aus dem der Avarworm entstand und der zudem für alle Aktionen das wesentliche Interaktionselement darstellt. Der nächst kleinere Ball folgt dem Avarball mit einem definierten Abstand a_i. Die darauf folgenden Bälle wiederum ihrem Vorgänger. Sinn dieser aufeinanderfolgenden Ballsequenz besteht in der ständigen Sichtbarkeit der Bewegungsrichtung des Avarworms. Auch bei stillstehendem Avarworm ist die zuvor erfolgte Bewegungsrichtung zu erkennen, da die nachfolgenden Bälle auf der Spur des Avarworm liegen bleiben. Bild 4.42 zeigt den Avarworm in unterschiedlichen Bewegungen. Aus dem Richtungsvektor $\vec{R}$ lässt sich die Bewegungsrichtung ableiten.

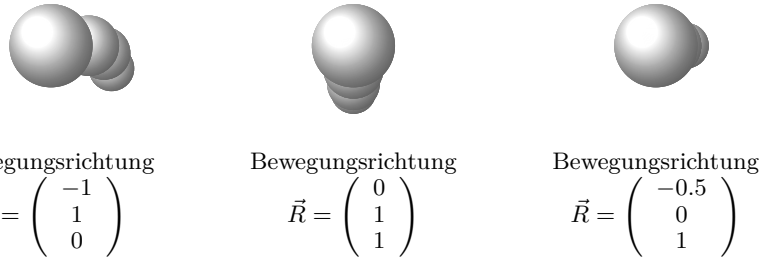


Bild 4.42: Bewegungsrichtung des Avarworm

Der wesentliche Unterschied zwischen einer herkömmlichen 3D CAD Anwendung an einem 2D Ausgabemedium und einer VR Anwendung besteht in deren tatsächlichen Nutzung der dritten Dimension des Raumes. Die Abschätzung der Größe des Avarworms in der Tiefe und die damit verbundene Abschätzung der Distanz zu anderen Objekten muss jedoch bedacht werden. Um dem gerecht zu werden, wurden vier virtuelle Lichtkugeln um den Avarball platziert, die in einem bestimmten Radiusbereich bei Annäherrung an ein Ojekt leuchten. Dies ermöglicht beim Annähern an ein potentielles Kollisionsobjekt, dass dieses beleuchtet wird. An einer Projektionsfläche entsteht bei dieser Annäherung zuerst ein Lichtpunkt, der sich vergrößert, bis das Innere der Lichthohlkugel erreicht wird. Sodann werden Kreise erzeugt. Eine Kombination mehrerer Lichtquellen, mit verschiedenen Farben und Radiusbereichen, kann so die Entfernung des Avarworm von anderen Objekten anzeigen. Das Konzept zur Annährung an ein Objekt ist in Bild 4.43 zu sehen. Die Vektoren $\vec{r_1}$ bis $\vec{r_4}$ zeigen die unterschiedlichen Radien der vier Lichtkugeln, denen jeweils unterschiedliche Farben konzentrisch zugeordnet

werden. Die Farbgebung imitiert dabei die Kälte- und Wärmefarben der Finite Element Berechnungswerkzeuge.

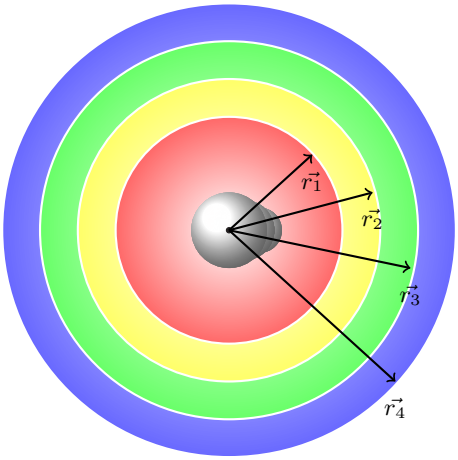


Bild 4.43: Annäherung des Avarworms an ein Objekt

Eine weitere im Avarworm integrierte Darstellungsmetapher bezieht sich auf die Geschwindigkeit des Avatars. Die Bälle des Avarworm folgen einander nicht sofort, sondern mit steigender Verzögerung. So zieht sich der Avarworm bei steigender Geschwindigkeit immer mehr auseinander. Dabei vergößert sich der Abstand $|a|$ des Avarworms. Da Strecke zu Geschwindigkeit und Beschleunigung im Relation steht, kann vom Benutzer intuitiv die Geschwindigkeit erkannt werden. Das Konzept dieses Effekts ist in Bild 4.44 zu sehen.

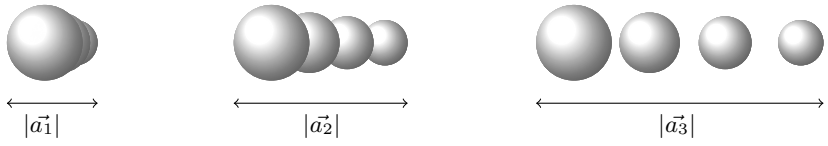


Bild 4.44: Geschindigkeitsbestimmung anhand des Avarworms

Der Avarworm dient als Mensch Maschine Schnittstelle. Er versucht dabei den Benutzer, als Teil der virtuellen Umgebung zu repräsentieren. Wie in Kapitel 3 auf Seite 79 und Tabelle 3.2 auf Seite 92 zusammenfassend gefordert, wird der Avarworm in der Szene dazu benutzt, die Interaktion des Nutzers mit den Objekten in der Szene zu ermöglichen. Die im Avarworm

integrierten Interaktionsmetapher werden auch von ungeübten und unsicheren Nutzern intuitiv richtig erkannt und benutzt. Ein weiterer Vorteil liegt in der ästhetischen Gestalt des Avarworms, der die Akzeptanz des Nutzers erhöht.

Audiovisueller Hilfe-Avatar ViMa

In Kapitel 2.6.4 auf Seite 67 und Kapitel 4.3.3 auf Seite 170 wurde bereits erwähnt, dass keine einheitlichen Standards oder Richtlinien für die Visualisierung von Metaphern für Funktionsverben im dreidimensionalen Raum vorhanden sind. Somit ist keine Notation für die Erkennung dieser Funktionselemente gegeben. Eine Visualisierung von Führungsgrößen hemmt ebenfalls das intuitive Verständnis dieser Strukturen (Maser, 2008). Daher werden in dieser Arbeit audiovisuelle Hilfsmittel verwendet, die dem Nutzer eine Hilfestellung innerhalb der virtuellen Umgebung sind.

Das Hinzufügen von Audiokommentaren zu 3D Objekten stellt eine wesentliche Modalität dar, um abstrakte Informationen in virtuellen Umgebungen zu präsentieren. Dabei verdecken Audiokommentare nicht die virtuelle Szene und erlauben das parallele Sammeln von sowohl visuellen Informationen aus der Umgebung und als auch abstrakten Audioinformationen, da es sich um unterschiedliche kognitive Tätigkeiten handelt, die einander nur marginal beeinflussen. Durch unterschiedliche Betonung oder Variation der Sprachgeschwindigkeit kann ebenfalls auf die Informationsaufnahme Einfluss genommen werden. Ein Nachteil liegt in der zeitlichen Linearität von Audioinformationen.

Es gibt verschiedene Möglichkeiten das Abspielen eines Audiokommentars zu triggern. Annäherung an bestimmte 3D Objekte mit hinterlegtem Audiokommentar wäre die Naheliegenste. Dazu muss dem 3D Objekt eine entsprechende Bounding Box, eine es umgebende sensitive Zone, zugewiesen werden. Eine andere Möglichkeit besteht darin, Kommentare an die Blickrichtung des Betrachters zu koppeln. Weiterhin können Kommentare verwendet werden, um Zustandsänderungen zu signalisieren, oder objektbezogene Informationen können genau dann abgespielt werden, wenn der Benutzer ein spezifisches Objekt in der virtuellen Welt selektiert. Je nach Trigger kann das Auffinden von Audiokommentaren in virtuellen Welten recht kompliziert werden, so dass es für manche Anwendungen sinnvoll ist, standardisierte Icons einzublenden, deren Anklicken das Abspielen des betreffenden Kommentars auslöst.

Impulsgeber für den im Rahmen dieser Arbeit entwickelten, audiovisuellen Hilfe-Avatar ViMa war Karl Klammer, der Avatar für die Hilfefunktionen innerhalb der MS-Office Anwendungen (Maser, 2008). Der in dieser Arbeit darzustellende Avatar soll eine graphische Erscheinung innerhalb der virtuellen Umgebung darstellen und dem Nutzer durch Sprachausgabe eine Hilfestellung für ungewohnte, komplexe Abläufe oder Darstellungen sein.

ViMa steht für "Virtuelle IMI Mitarbeiterin" und soll dem Nutzer eine Hilfestellung für die Bewegung in der virtuellen Umgebung sein. Die Metapher für ViMa ist der Buchstabe *i*, welcher allgemein für *Information* steht. ViMa ist in Bild 4.45 dargestellt.

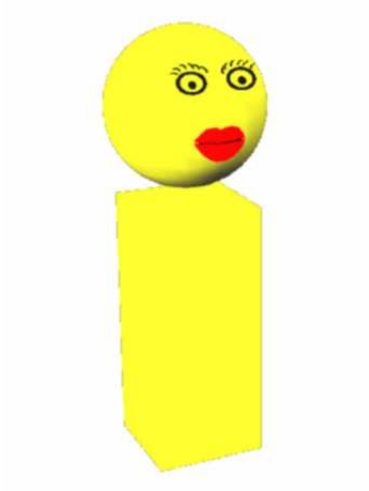


Bild 4.45: Der audiovisuelle Hilfe-Avatar ViMa

Die Aufgabe des Hilfe-Avatars ViMa besteht darin, innerhalb der Funktionsvisualisierung Hilfestellung zu leisten. Dabei spricht sie nochmals die nicht sofort intuitiv visuell verständlichen Verbnamen aus und nennt dabei deren Führungsgrößen, die entsprechend den Anforderungen aus Tabelle 3.2 auf Seite 92 möglichst immersiv dargestellt werden sollen. Somit gibt sie dem Anwender der virtuellen Umgebung nicht nur Hilfestellung, sondern ebenso Metainformationen weiter.

Ballgesteuerte Systemsteuerung mittels der Go-Go Metapher

Wie bereits bei der Interaktionsmetapher Avarworm erwähnt, liegt ein wesentlicher Unterschied bei der Arbeit in einer virtuellen Umgebung in der

räumlichen Dimension, durch welche die Tiefe des Raumes tatsächlich in die Arbeitsumgebung eingeschlossen und nicht nur simuliert werden kann. Sie markiert das sichtbare Unterscheidungsmerkmal zu herkömmlichen Desktopsystemen. Um diese besser zu nutzen, wurde ein Konzept entwickelt, welches Steuerungsfunktionalitäten der Szene in die Tiefe des Raumes verlagert, von wo sie bei Bedarf jederzeit in den Vordergrund geblendet werden können. Als Bedienelemente wurden farblich getrennte Kugeln verwendet, da diese sich von dem Interaktionsavatar Avarworm besser selektieren lassen, als vergleichbare Elemente. Bild 4.46 zeigt die Visualisierung der Bedienelemente.

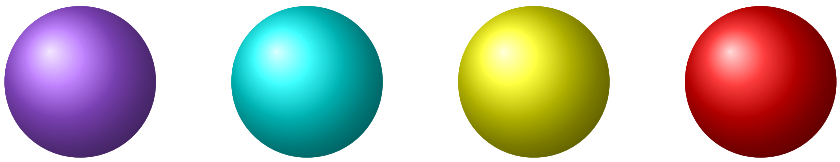


Bild 4.46: Bedienelemente in der virtuellen Umgebung

Um die Bedienelemente aus der Tiefe des Raumes erreichen zu können, wurde als Hilfsmittel die Go-Go Metapher verwendet. Diese verlängert den virtuellen Arm des Benutzers, um Objekte, die in der Tiefe des Raumes liegen, zu erreichen, um diese zu selektieren und manipulieren zu können. Die Go-Go Metapher beruht auf der nichtlinearen Übertragung der getrackten Bewegung der realen Hand des Benutzers auf die bewirkte Bewegung der virtuellen Hand (vgl. Kapitel 2.6.4 auf Seite 71). Darüberhinaus wurde ein System entwickelt, das die Bedienelemente aus allen Kamerapositionen verfügbar macht. Damit wird gewährleistet, das diese nicht plötzlich verschwinden. Bild 4.47 beschreibt das entwickelte System. Die Repräsentation des Benutzers in der virtuellen Umgebung, der Avarworm (vgl. Kapitel 4.3.2), befindet sich in einem definierten Abstand $|\vec{r_1}|$ von einem virtuellen Bezugspunkt B entfernt. In doppelter Entfernung $|\vec{r_2}| = 2 \cdot |\vec{r_1}|$ zum Bezugspunkt B befinden sich die Bedienelemente. Die Distanz d zwischen dem Avarworm und den Bedienelementen ergibt sich somit auf $d = |\vec{r_1}| + |\vec{r_2}|$. Ist der Betrag $|\vec{r_2}|$ größer als $^2/_3$ der durchschittlichen Standardarmlände (vgl. Anhang B auf Seite 283) wird der virtuelle Arm mittels der Go-Go Metapher um den Betrag $|R_r + k(R_r - D)^2|$ nicht linear verlängert (vgl. Kapitel 2.6.4 auf Seite 71). Zudem sind die Systembälle opaque gehalten. Das heißt, das diese im Hintergrund vollständig transparent erscheinen, während sie in der Annäherung zu der Kamera die Transparenz nach und nach verlieren, so dass

eine bequeme Auswahl vor der Kamera erfolgen kann.

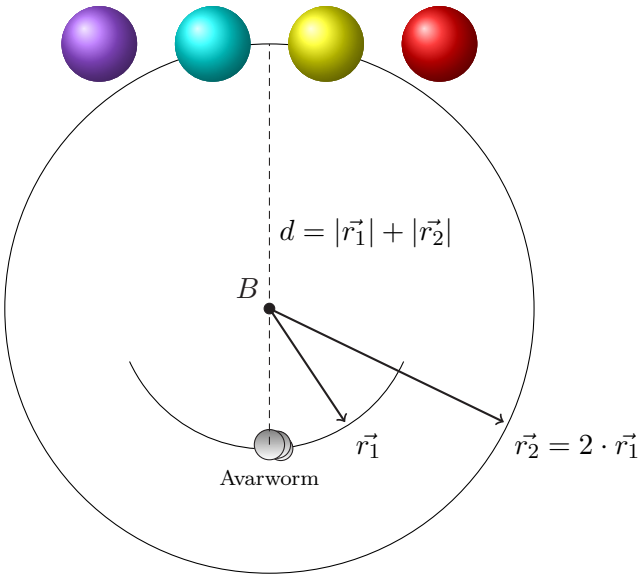


Bild 4.47: Positionsbestimmung der Bedienelemente

Dabei bezeichnet jede Farbe eine dem Bedienelement hinterlegte Funktion. Funktionen wechseln sich dabei je nach Bedarf in der Szene ab. Generell soll der Benutzer mittels der Bedienelemente in die Lage versetzt werden, durch die Auswahl einzelner Komponenten in beliebiger Reihenfolge einzelne Funktionen auszuwählen und so entweder unterschiedliche Sichten oder eine detailliertere Visualisierung auszulösen. Dabei hat er die Möglichkeiten entweder ein Objekt auszuwählen oder ein Objekt zu verdecken. Eine Funktion kann genau dann visualisiert werden, wenn alle Objekte, die an der Funktion beteiligt sind, noch in der Szene vorhanden sind. Wurde beispielsweise der Detaillierungsgrad der Visualisierung bereits auf wenige Objekte reduziert, so sind nicht mehr alle Funktionen selektierbar. Wählt der Bediener nacheinander Objekte aus oder schließt sie aus, wird der Kreis visualisierbarer Funktionen sehr schnell eingegrenzt, da in der Regel nur wenige Objekte an sehr vielen Funktionen beteiligt sind. Das Eingrenzen durch gezieltes Auswählen funktioniert, indem alle Funktionen, an welchen das Objekt nicht beteiligt ist, aus dem Kreis möglicher Funktionen ausgeschlossen werden. Das Eingrenzen durch gezieltes Verdecken funktioniert genau umgekehrt, es werden alle Funktionen, an denen das Objekt beteiligt ist, aus dem Kreis ausgeschlossen.

Interaktive räumliche Sphären

Durch die räumliche Dimension, kommt es bei den meisten ungeübten Benutzern zu einer Orientierungslosigkeit in der virtuellen Umgebung, die die Arbeit mit den Objekten, und damit die Akzeptanz des Benutzers, deutlich negativ beeinflusst. Die Desktopmetapher bietet in solchen Situationen die Escapetaste, um wieder in einen ursprünglichen Zustand zurückzufinden. In den meisten VR Entwicklungsumgebungen und den dazugehörigen VR Eingabegeräten sind ebenfalls Buttons vorhanden, die solche Aktionen übernehmen können. Im Rahmen dieser Arbeit wird jedoch versucht von den herkömmlichen Denkwegen abzukommen und die Interaktion zwischen virtueller Welt und Anwender deutlich intuitiver zu gestalten.

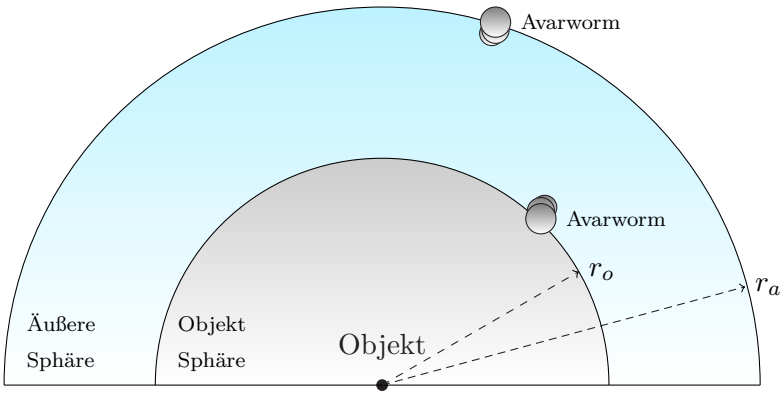


Bild 4.48: Begrenzungen der Interaktionsradien

Um ein Verirren in der Szene zu vermeiden, wurden transparente Sphären um die virtuelle Umgebung gelegt, die bei Berührung durch den Avatar bestimmte definierbare Aktionen triggern können. Das ist in zweierlei Hinsicht notwendig. Zum Einen wird mit einer halb geschlossenen Sphäre des Radius r_a das Verlassen der Szene verhindert (vgl. Bild 4.48). So ist es nicht möglich unter den Boden zu navigieren, oder in den Horizont zu fliegen. Bei Berührung der äußeren Sphäre, wird die Kamera zum Mittelpunkt der Sphäre hin neu ausgerichtet. Zum Zweiten können bestimmte Objekte durchflogen werden, was bei größeren Obkjekten manchmal zum Verlust der Orientierung führt. Obwohl das Hindurchfliegen durch Objekte nicht realistisch ist, vereinfacht es jedoch die Navigation deutlich. Generell ist es möglich, hinter jedem Objekt einen Kollisionsmechanismus zu legen. Dies ist aber nur in physikalisch relevanten Szenen notwendig, und ist im Sinne

der Forderung nach Skalierbarkeit (vgl. Kapitel 3.3.3 auf Seite 89) nicht näher betrachtet worden. Bild 4.48 zeigt konzeptionell die Funktionaliät der Begrenzungssphären.

Neben der äußeren Sphäre, sind diese auch als Steuerungselement um Objekte herum denkbar. Dazu werden die Objekt-Sphären wiederum in einen bestimmten Abstand r_o festgelegt und können beliebige Aktionen auslösen. Dabei ist zu beachten dass bei jeder Berührung der Objekt-Sphäre die damit verbundene Aktion wieder und wieder ausgelöst wird. Aus diesem Grund ist der Einsatz der interaktiven räumlichen Sphären nur begrenzt sinnvoll.

World in Window

Der Nutzer der virtuellen Umgebung kann sich dreidimensional im Raum bewegen, doch ist seine visuelle Wahrnehmung zweidimensional. Dadurch ist es dem Nutzer nicht möglich, eine Bewegung linear zu seiner Blickrichtung eindeutig zu verfolgen. Diesen Nachteil kann die virtuelle Umgebung, im Gegensatz zur realen, durch ein integriertes zweidimensionales Ansichtsfenster ausgleichen. Der Effekt soll im Rahmen dieser Arbeit als "World in Window" Metapher bezeichnet werden, und verwendet zum einen die picture-in-picture Metapher und stellt zum anderen eine zweidimensionale Abwandlung der Interaktionsmetapher "World in Miniature" dar.

Bei der exozentrischen WiM arbeitet der Benutzer mit einer verkleinerten Darstellung der virtuellen Umgebung, die ikonische Repräsentationen aller Objekte enthält und vor allem den Vorteil bietet, viel einfacher manipulierbar zu sein als eine real skalierte Welt. Alle Objekte in dieser Miniaturwelt können in sechs Freiheitsgraden auf jede beliebige Distanz verändert werden (vgl. Kapitel 2.6.4 auf Seite 68). Im Unterschied zu der picture-in-picture Metapher bleibt bei der WiW Metapher die Immersion weiterhin erhalten. Außerdem ist die Interaktion mit der virtuellen Welt trotz gleichzeitiger Nutzung der Metapher möglich.

Durch Verwendung der WiW Metapher wird die Interaktion im Sinne von Jaron Lanier zusätzlich um überraschende Stilmittel ergänzt (vgl. Kapitel 1.3 auf Seite 4). Diese vermitteln weitere Informationen, die im betrachteten Zusammenhang nützlich erscheinen, jedoch dabei nicht sofort ins Auge fallen. Ähnlich wie die interaktiven räumlichen Sphären ist auch die WiW Interaktionsmetapher mit Bedacht einzusetzen, um eine Überfrachtung der Szene zu vermeiden.

Kollisionserkennung

Wesentlicher Bestandteil im realen Leben ist der Tastsinn, der taktiles und haptisches Feedback überträgt. Durch die taktile Wahrnehmung wird das Erkennen von Druck, Berührung und Vibrationen auf der Haut ermöglicht , was oft auch als Oberflächensensibilität bezeichnet wird. Dabei ist es dem Menschen möglich Objekte zu berühren, oder diese miteinander zu verknüpfen. Diese physische Verknüpfung wird stark durch die Gravitationskraft beeinflusst, die alle Objekte gleichermaßen betrifft. In der virtuellen Umgebung muss die Gravitationskraft für jedes Objekt simuliert werden. Dabei können insbesondere Kollisionen zwischen Objekten erkannt werden, die den realitätsnahen Eindruck verstärken, und in bestimmten technischen Anwendungsfällen genau untersucht werden.

Kollisionen sind somit ein inhärenter Teil der Produktstruktur, die gesonderte Betrachtung erfordern. Dementsprechend wurde im Rahmen dieser Arbeit hinter allen notwendigen Objekten eine Physik implementiert, die Kollisionen berücksichtigt. So ist beispielsweise die Bewegung eines Sitzes nicht nur konstruktionsbedingt auf bestimmte Freiheitsgrade beschränkt, sondern auch durch den in der jeweiligen Situation zur Verfügung stehenden Bauraum. Wenn z.B. der Verfahrschlitten den hinteren Anschlag berührt, kann die Rückenlehne nicht komplett nach hinten geneigt werden, weil vorher eine Kollision mit der Rücksitzbank erfolgt. Schematisch ist dies in Bild 4.49 dargestellt.

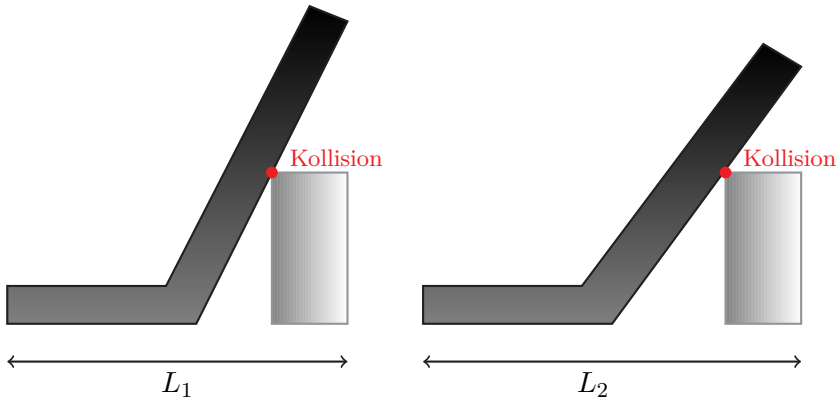
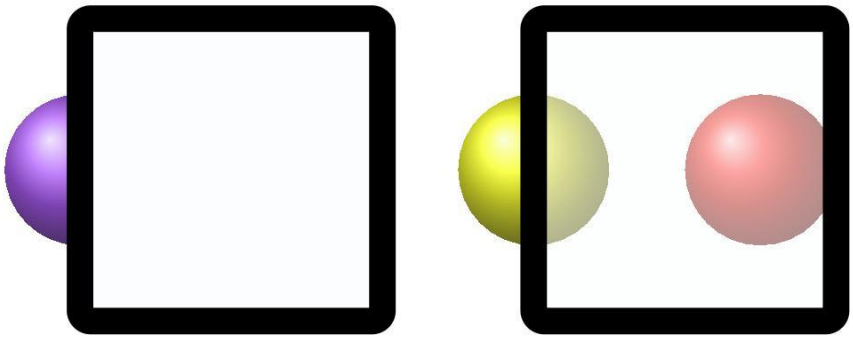


Bild 4.49: Beweglichkeit einer Rückenlehne in Abhängigkeit von der Schlittenstellung

Halbtransparente Objekte

In der virtuellen Welt werden in erster Linie Oberflächen von Objekten dargestellt und gerendert. Der Inhalt der meisten Objekte ist für den Benutzer nicht von Interesse. Dieser Fakt wird für den Entwurf von 3D Modellen bereits bedacht, um die Anzahl der Polygone in einer Szene, und damit die Rechenzeit des Computers so gering wie möglich zu halten. Dennoch kann es erwünscht sein, hinter die Oberfläche mancher Objekte zu schauen.

Im Rahmen dieser Arbeit wurde dazu die Virtual X-Ray Metapher in abgewandelter Form genutzt, die es erlaubt, Objekte dynamisch, halbtransparent darzustellen, und so Zugang zu den verdeckt liegenden Objekten zu erhalten (vgl. Kapitel 2.6.4 auf Seite 72). Bild 4.50 zeigt die Verwendung transparenter Objekte, um dahinterliegende Objekte nutzbar zu machen.



Verdeckte Objekte hinter undurchsichtiger Wand

Verdeckte Objekte hinter semitransparenter Wand

Bild 4.50: Halbtransparente dargestellte Objekte

Dabei hat sich gezeigt, dass sowohl die zeitliche Performanz als auch die eindeutige Verwendung von Interaktionstechniken, wie Entdeckung, Selektion und Manipulation, sich signifikant verbessert hat. Allerdings wird durch die erhöhte visuelle Komplexität eine stärkere Rechenlast und eine leicht verminderte Tiefenwirkung der virtuellen Umgebung in Kauf genommen, wodurch die Immersion beeinträchtigt wird.

Durch gezielte Auswahl der 3D Objekte kontrolliert der Benutzer, in welchen Teilen der virtuellen Welt die dynamische Transparenz aktiv sein soll. Um den Effekt der verminderten Tiefenwirkung so gering wie möglich zu halten,

werden nicht komplette Bauteile ausgeblendet, sondern lediglich diejenigen Oberflächen halbtransparent dargestellt, die auch wirklich Zielobjekte verdecken. Alle anderen Merkmale der stereoskopischen Tiefenwahrnehmung (Stereopsis, Bewegungsparalaxe, relative Größe, atmosphärische Perspektive und Texturgradienten) werden von der Metapher nicht beeinflusst und leisten weiterhin ihren Beitrag zur Entstehung einer räumlichen Darstellung.

4.3.3 Visualisierung der Produktfunktionen

Zur Visualsierung des Funktionsmodells in der virtuellen Umgebung, sind insbesondere die Funktionsstrukturen mit den Funktionsobjekten, Transitionen und Funktionsverben von Bedeutung. Dabei ist zu beachten das es momentan keinen Standard zur Visualisierung von Funktionsstrukturen in virtuellen Umgebungen gibt. Dies erfordert eine sensible Erarbeitung der Visualisierung von Zusammenhängen zwischen Funktionsobjekten und Funktionsverben, damit diese vom Nutzer intuitiv in ihren Funktionen zugeordnet werden können.

Die aus der Funktionsmodellierung vorhandenen Daten, auf die zur Informationsvisualisierung zurückgegriffen werden kann sind:

- Funktionsobjekte als Gegenstand
- Funktionsverben als Name
- Funktionsflüsse (Transitionen)
- Gruppierung in Teilfunktionen
- Gruppierung in Elementarfunktionen

Im Folgenden wird die Visualisierung dieser Informationen in einer virtuellen Umgebung sequentiell diskutiert und Vorschläge einer intuitiven, wiederverwendbaren Darstellung derselben gemacht. Dabei werden die Forderungen aus Tabelle 3.2 auf Seite 92 berücksichtigt.

Visualisierung der Funktionsobjekte

Die Visualisierung der Funktionsobjekte erfolgt durch ihre geometrische Darstellung. Dabei werden die geometrischen Objekte entweder aus bereits

vorhandenen Daten einer Anpassungs- oder Variantenkonstruktion gewonnen und aus dem CAD Tool in die VR Entwicklungsumgebung importiert, oder sie müssen im Rahmen einer Neukonstruktion vorab modelliert werden. In beiden Fällen ist für die Darstellung in Echtzeit die in Kapitel 4.1 erläuterte Vorgehensweise zu nutzen.

Visualisierung der Funktionsverben

Aus Kapitel 2.5 auf Seite 37 ist bereits deutlich geworden, dass gegenwärtig keine wissenschaftlichen Forschungen oder Ergebnisse existieren, die sich mit der Erarbeitung visueller Interaktionsmetaphern für Funktionsverben beschäftigen. Arbeiten der Konstruktionsmethodiker wie Roth (Roth, 2000), Pahl und Beitz (Pahl, 2007), Koller (Koller, 1998) oder Ehrlenspiel (Ehrlenspiel, 2003) visualisieren Funktionsverben im zweidimensionalen Raum. Dabei bezeichnet Ehrlenspiel beispielsweise Funktionsverben als elementare Operationen. Die Operationen sind aber selbst im zweidimensionalen Raum für einen Betrachter nicht intuitiv erkennbar. Tabelle 4.6 verdeutlicht dies.

elementare Operation	**Darstellung**	**technische Operation**
leiten		leiten, zuführen, abführen, tragen, transportieren, lagern (im Sinne von “Kraftleiten”), übertragen, dichten, schalten, isolieren, unterbrechen etc.
ändern		ändern, vergrößern, verkleiner, umlenken, übersetzen, umformen, verlängern, verdichten, zerspanen, schmelzen, verdampfen, reflektieren
wandeln		wandeln, umsetzen, erzeugen, absorbieren, verbrennen, ersetzen, messen etc.
vereinigen		vereinigen, verzweigen, überlagern, summieren, aufteilen, zusammenführen, verbinden, montieren, entmischen, vermischen, trennen etc.
speichern		speichern, dämpfen, glätten, lagern (im Sinne von “Stofflagern”), aufstauen, sammeln etc.

Tabelle 4.6: Darstellung von Funktionsverben nach (Ehrlenspiel, 2003)

Zur Ermittlung geeigneter Metaphern wird in dieser Arbeit anhand eines methodischen Vorgehens die auf Basis der NIST Notation (vgl. Anhang A auf Seite 279) vorgeschlaganen Funktionsverben betrachtet. Dabei wurde

die Brainstorming Methode gewählt, da diese als Problemlösungsstrategie die Synergien einer Gruppe ausnutzt (Clark, 1973).

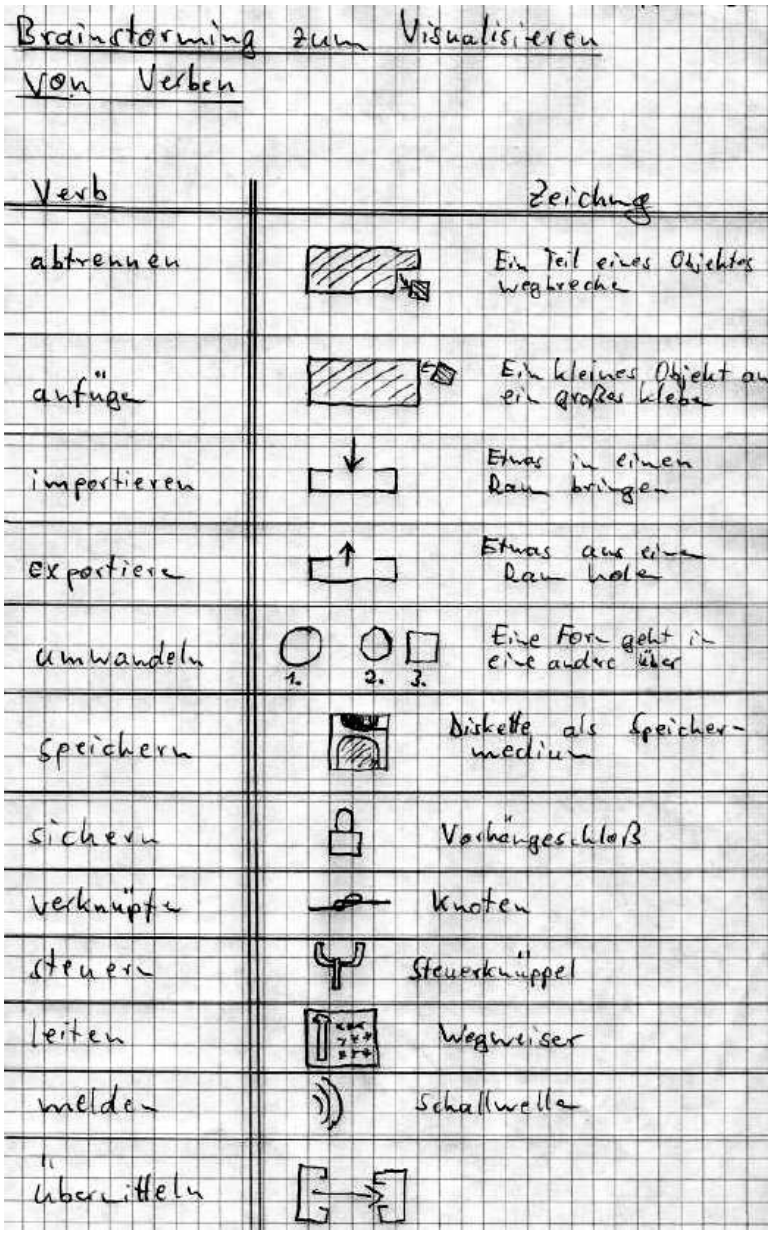


Bild 4.51: Brainstorming zur Ermittlung visueller Metaphern für Funktionsverben

Durch diese Evaluierung in einer Gruppe wird das subjektive Verständnis mehrerer Personen zusammengefasst und liefert einen, innerhalb der Grup-

pe, objektiven Konsens für die Erstellung der Metaphern. Das Ergebnis stellt eine visualisierte Metapher für das jeweilige Funktionsverb dar.

Das Brainstorming wurde mit einer Gruppe von zehn Personen durchgeführt, in der die Personen unterschiedliche Altersklassen und Ausbildungsniveaus repräsentierten. Die Geschlechter waren ebenfalls zu je 50% vertreten. Die visualisierten Metaphern sollten in der Validierung von einer anderen Personengruppe (zwölf Personen) den Verbnamen zugeordnet werden. Ergebnis der Validierung war, dass neun Personen die Verben zu 100% den Metaphern zuordnen konnten. Dies kann als Indikator dafür angesehen werden, dass die Metaphern intuitiv von Nutzern zugeordnet werden können, wenn diese die zu verwendeten Verbnamen kennen. (Maser, 2008)

Die Informationsweitergabe von Führungsgrößen durch Visualisierung hat ergeben, dass dies für den Nutzer eine Steigerung der Komplexität zur Folge hat, da der Nutzer für diese Metaphern ebenfalls eine Notation benötigt. Deswegen ist hier auf audiovisuelle Elemente zurückgegriffen worden (vgl. Kapitel 4.3.2 auf Seite 161). Bild 4.51 auf der vorherigen Seite zeigt das Ergebnis des durchgeführten Brainstormings.

Anhand von Bild 4.51 ist jedoch nicht sofort ersichtlich, wie die Metaphern der Funktionsverben zu modellieren sind. Dementsprechend ist die Darstellung der Funktionsverben zum verbesserten Verständnis iconografisch in Bild 4.52 auf der nächsten Seite visualisiert. Dabei ist lediglich eine Auswahl der möglichen Funktionsverben exemplarisch untersucht worden.

Visualisierung der Richtung des Funktionsflusses

Im Rahmen einer intuitiven Nutzung virtueller Umgebungen besteht eine große Herausforderung darin, die gängigen Desktopmetaphern von den 2D Computerbildschirmen zu überwinden. Da in einer virtuellen Umgebung die dritte, räumliche Dimension in die Darstellung integriert werden soll, müssen die Konzepte entsprechend angepasst werden.

Dementsprechend sollen zur Darstellung der Funktionsflussrichtung explizit keine Pfeile verwendet werden. Die Richtung ist folglich durch die Orientierung der Objekte innerhalb des dreidimensionalen Raumes zu modellieren. Dies bedeutet auch, dass ein Großteil der an der Funktion beteiligten Elemente keine symmetrischen Körper darstellen dürfen, sondern eine erkennbare Vorder- und Rückseite haben müssen, damit der Nutzer intuitiv die Flussrichtung erkennen kann. Bild 4.53 auf Seite 174 zeigt beispielhaft, wie

ein Körper (hier ein Funktionsverb) mit definierter Vorder- und Rückseite den Funktionsfluss beeinflusst.

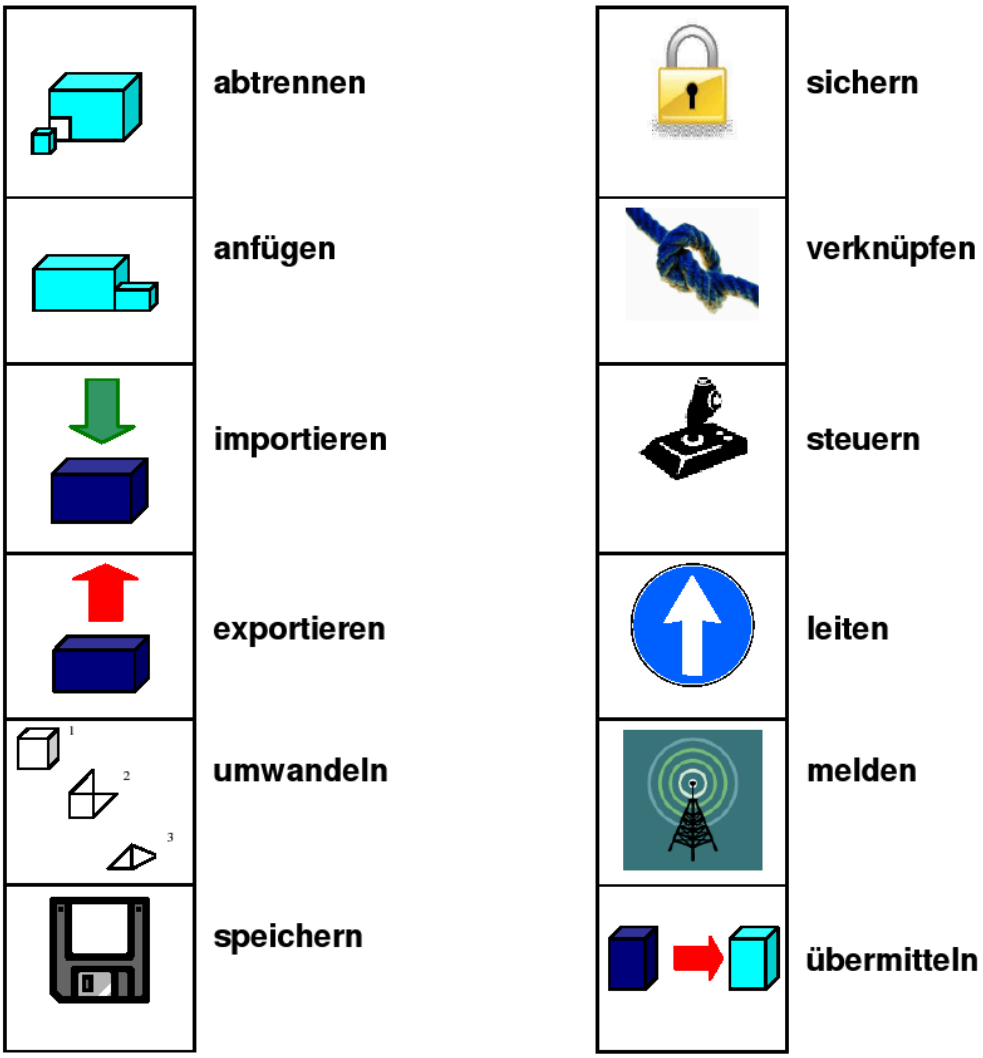


Bild 4.52: Visualisierte Funktionsverben

Visualisierung der Gesamtfunktion

Die Gesamtfunktion eines Produktes setzt sich aus Teilfunktionen, Elementarfunktionen und unerwünschten Funktionen zusammen (vgl. Kapitel 4.2.1 auf Seite 118). Eine Elementarfunktion setzt sich aus Funktionsobjekt, Funktionsverb und Transition zusammen (vgl. Kapitel 4.2.2 auf Sei-

te 126). Die Funktionsobjekte und Funktionsverben sind anhand der Vorgehensweisen aus den vorangegangenen Kapiteln eindeutig erkennbar und zuzuordnen. Hierbei kann das Funktionsobjekt als Eingangsgröße der Funktion betrachtet werden.

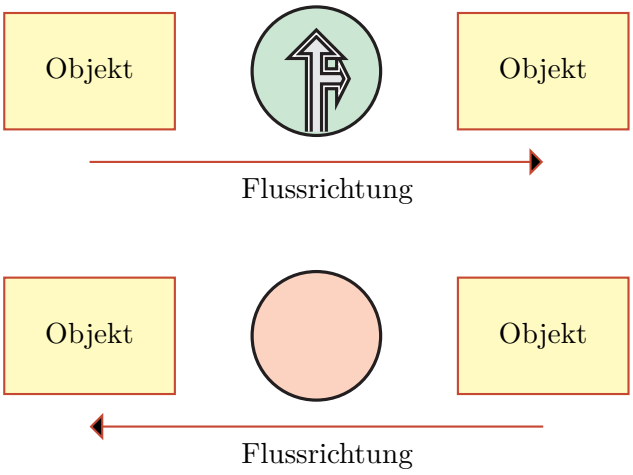
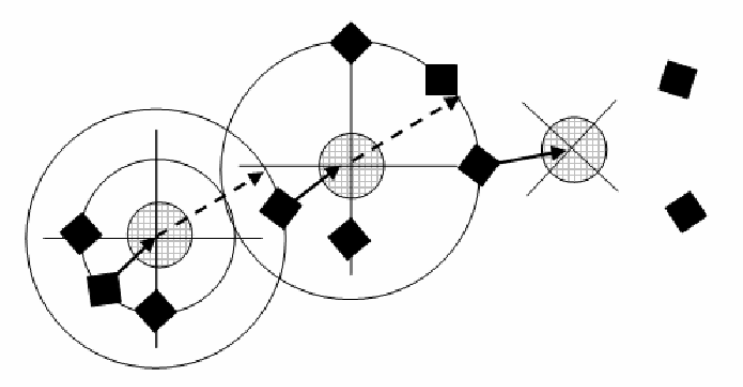


Bild 4.53: Modellierung des Funktionsflusses durch Optimierung der Objekte

Zur Identifizierung dieser Eingangsgröße wird eine spezifische Relativposition des Objektes zu seinem Verb festgelegt. Da mit der Erweiterung von Eingangs- und Ausgangsgrößen die Elementarfunktion zur Teilfunktion bis hin zur Gesamtfunktion erweiterbar ist, bietet sich eine Kreisanordnung der Objekte um das Verb an. Die Objekte, die die Ausgangsgrößen darstellen, sind ebenfalls im Kreis um das Verb anzuordnen, aber in entgegen gesetzter Orientierungsrichtung zum Verb. Diese entgegengesetzte Orientierungsrichtung impliziert den größtmöglichen Umfang der Kreisanordnung für die jeweilige Objektgruppe (Eingang-/ Ausgangsobjekte) zu $\pi * r$.

Zur weiteren Unterscheidung zwischen Eingangs- und Ausgangsobjekten sind die Radien der jeweiligen Kreisanordnung unterschiedlich groß gewählt. Bild 4.54 verdeutlicht die Vorgehensweise zur Visualisierung der Gesamtfunktion. Dabei werden bewusst keine Verbindungslinien zwischen Objekten und Verben zur Identifikation der Beziehungen genutzt, da diese bereits im zweidimensionalen Raum in Form von Skizzen Verwendung finden. In der virtuellen Umgebung erscheint somit das funktionsverarbeitende Funktionsverb nahe an dem zugehörigen Funktionsobjekt, das die Eingangsführungsgröße hinzufügt. Die Weitergabe der resultierenden Ausgangsgröße an

das folgende Funktionsobjekt ist visuell immer mit einen größerem Abstand verbunden. Somit lässt sich auch im Fall von mehreren Funktionen eines Funktionsverbes eindeutig der Überblick erhalten.



Legende:

- Funktionsobjekt Eingang/Ausgang
- Funktionsverb
- → Radius Eingangsobjekt – Verb
- -→ Radius Verb – Ausgangsobjekt

Bild 4.54: Visualisierung der Funktionsstruktur in der virtuellen Umgebung

KAPITEL 5

Verifikation des Konzepts

Im Folgenden wird die Verifikation des Konzepts anhand der prototypischen Realisierung mittels dem Modellierungswerkzeug Visio und der VR Entwicklungsumgebung Virtools erläutert. Ziel der durchgeführten Realisierung war es, auf Basis moderner Technologien ein Szenario zu entwickeln, in dem die wesentlichen, in Kapitel 4 auf Seite 93 vorgestellten Verarbeitungsmechanismen, implementiert wurden.

Anhand eines komplexen, interdisziplinären Produktbeispiels aus dem Bereich der Automobilindustrie wird das entwickelte Konzept verifiziert, so dass zukünftig die Potentiale der Funktionsmodellierung verknüpft mit den Technologien der virtuellen Realität von vielen Unternehmen profitabel genutzt werden können. Bei dem ausgewählten Produkt handelt es sich um eine crashaktive Kopfstütze eines elektronischen Fahrzeugsitzes eines Automobilherstellers.

Zuerst werden die für die Erstellung des Forschungsprototypen eingesetzten Entwicklungsumgebungen erläutert. Die entwickelte Funktionalität wird daraufhin in einem Anwendungsszenario vorgestellt.

5.1 Entwicklungsumgebungen

5.1.1 Modellierungswerkzeug Visio

Die erstellten Funktionsmodelle werden gegenwärtig zum größten Teil nur zu Dokumentationszwecken verwendet und unterliegen keinen weiteren rech-

nerunterstützten Verarbeitungen (vgl. Kapitel 2.2 auf Seite 16). Daher sind modellspezifische Werkzeuge geeignet und weit verbreitet. Diese haben aber den Nachteil, auf eine fest definierte Notation beschränkt zu sein. In dem Fall der Funktionsmodellierung werden trotz der Einführung von standardisierten Notationen (vgl. Kapitel 2.2.3 auf Seite 21) weitere Notationen erforscht (vgl. Kapitel 4.2 auf Seite 116, um unter anderem Erweiterungen durch computergestützte Verarbeitungen anzubieten.

Für solche nicht standardisierten Notationen ist der Einsatz von Modellierungssoftware der Entwicklung von neuen modellspezifischen Werkzeugen vorzuziehen. Modellierungsstools bieten Bibliotheken von 2D Diagrammobjekten (auch Shapes genannt) an, mit denen Diagramme erstellt werden können. Die Einführung neuer Bibliotheken erlaubt somit die Unterstützung von neuen Notationen. Dia, Kivio und Microsoft Visio sind drei verbreitete Visualisierungstools, wobei Dia und Kivio, trotz deren Verfügbarkeit als Open-Source Projekte, beschränkt bezüglich der Erweiterbarkeit und der Herstellung von neuen Bibliotheken sind. Aufgund dieser Ausschlußkriterien, sowie des Quasistandards der Microsoft Office Suite, in der Visio integriert ist, wird das Modellierungswerkzeug Visio gewählt.

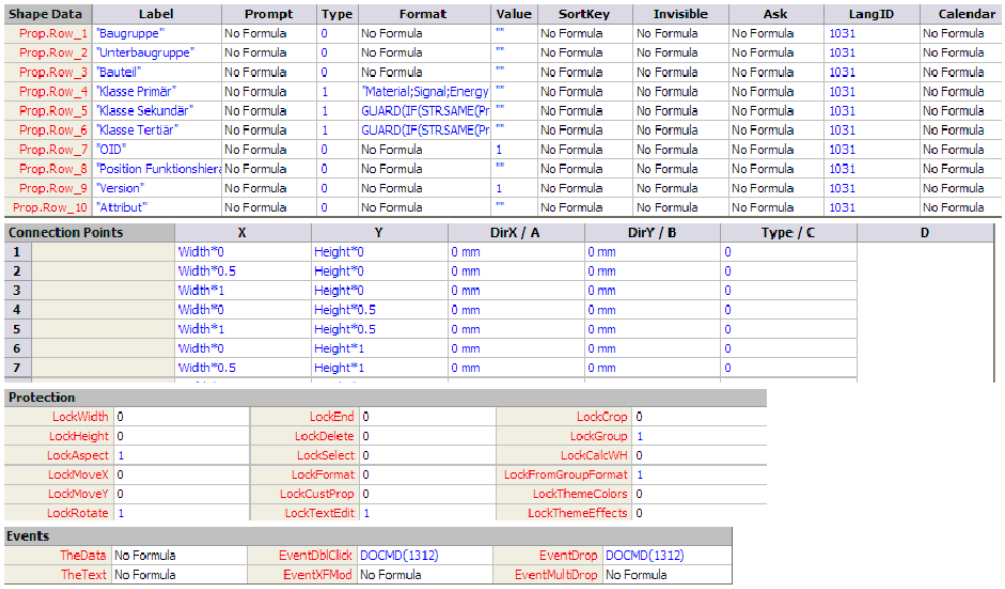
Visio ist eine weit verbreitete, proprietäre Modellierungssoftware von Microsoft für Windows und Bestandteil des Microsoft Office Systems. Sie wird bereits in der Industrie zur Modellierung von Flussdiagrammen und Geschäftsprozesse verwendet und ist somit ein de-facto Standard. Visio ist für die Darstellung von Diagrammen zur Visualisierung und Kommunikation von Prozessen und anderen Informationen entwickelt worden. Daher enthalten die mitgelieferten Bibliotheken (Templates) graphische Elemente mit wenigen Benutzungsregeln und Verhaltensbeschreibungen, so dass die Elemente flexibel verwendet werden können, ohne spezifische Notationseinschränkungen mitzubringen. Dies hat zur Folge, dass die mit Visio erstellten Modelle meistens nur zur Dokumentation dienen und keiner automatisierten, synktatischen Überprüfung unterliegen (Zhou, 2008).

Das Microsoft Visio SDK steht als kostenloser Download zur Verfügung und beinhaltet das Werkzeug ShapeStudio, das den Benutzern die Möglichkeit bietet, neue Schablonen für Diagrammobjekte zu definieren (Microsoft, 2008a). Damit können neue Diagrammobjekte definiert werden, wobei sowohl die grafische Darstellung als auch Objektdaten und Verhalten beschrieben werden können. Die grafische Beschreibung erfolgt durch die Zusammensetzung von primitiven Vektorgrafikelementen (Shapes). Diese können dann zu einem zusammenfassenden Diagrammobjekt (Master Sha-

pe) gruppiert werden. Jedes Shape wird einem Shapesheet zugeordnet, dass deren Eigenschaften (Daten, Verhalten, Größe, Position, und so weiter) als Tabelle von Funktionen und Werte darstellt (Zhou, 2008).

Bild 5.1 zeigt den Auszug eines Shapesheets. Es beinhaltet die Abschnitte ”Shape Data”, zur Definition von Objektdaten, ”Connection Points”, zur Definition von Anhangspunkte für Konnektoren, ”Protection”, für die Einschränkung von Benutzeraktionen auf das Shape, und ”Events” zur Beschreibung des Verhaltens des Shapes bei bestimmten Ereignissen wie z.B. einem Doppelklick. Ausführliche Beschreibungen der verschiedenen Abschnitte, deren Inhalte und verwendbare Funktionen in Shapesheet sind in Kapitel 5.2, bei (Microsoft, 2008b) und (Martin, 2007) zu finden.

Durch die Verfügbarkeit eines Werkzeugs zur Spezifikation von selbsdefinierten Diagrammobjekten, und die Möglichkeit sowohl die grafische Darstellung als auch das Verhalten zu beschreiben, erfüllt Visio alle Anforderungen als Modellierungswerkzeug für die Funktionsmodellierung. Dafür ist aber die Entwicklung eines Templates zur Funktionsmodellierung notwendig. Der resultierende Aufwand ist jedoch geringer verglichen mit der Entwicklung eines kompletten Modellierungswerkzeugs für die Unterstützung einer neuen Notation.



Shape Data	Label	Prompt	Type	Format	Value	SortKey	Invisible	Ask	LangID	Calendar
Prop.Row_1	"Baugruppe"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_2	"Unterbaugruppe"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_3	"Bauteil"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_4	"Klasse Primär"	No Formula	1	"Material;Signal;Energy	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_5	"Klasse Sekundär"	No Formula	1	GUARD(IF(STR.SAME(Pr	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_6	"Klasse Tertiär"	No Formula	1	GUARD(IF(STR.SAME(Pr	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_7	"OID"	No Formula	0	No Formula	1	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_8	"Position Funktionshier	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_9	"Version"	No Formula	0	No Formula	1	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_10	"Attribut"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula

Connection Points	X	Y	DirX / A	DirY / B	Type / C	D
1	Width*0	Height*0	0 mm	0 mm	0	
2	Width*0.5	Height*0	0 mm	0 mm	0	
3	Width*1	Height*0	0 mm	0 mm	0	
4	Width*0	Height*0.5	0 mm	0 mm	0	
5	Width*1	Height*0.5	0 mm	0 mm	0	
6	Width*0	Height*1	0 mm	0 mm	0	
7	Width*0.5	Height*1	0 mm	0 mm	0	

Protection					
LockWidth	0	LockEnd	0	LockCrop	0
LockHeight	0	LockDelete	0	LockGroup	1
LockAspect	1	LockSelect	0	LockCalcWH	0
LockMoveX	0	LockFormat	0	LockFromGroupFormat	1
LockMoveY	0	LockCustProp	0	LockThemeColors	0
LockRotate	1	LockTextEdit	1	LockThemeEffects	0

Events					
TheData	No Formula	EventDblClick	DOCMD(1312)	EventDrop	DOCMD(1312)
TheText	No Formula	EventXFMod	No Formula	EventMultiDrop	No Formula

Bild 5.1: Auszug eines Shapesheets vom Visio SDK

5.1.2 Visualisierungswerkzeug Virtools

Die in der Industrie vielfach eingesetzten CAD/CAM Werkzeuge stellen Funktionen zur Modellierung und Visualisierung von 3D Modellen zur Verfügung, bieten aber nur wenige Möglichkeiten zur Interaktion mit den 3D Objekten im Modellraum. Im Allgemeinen kann das betrachtete 3D Modell nur in Position und Orientierung verändert werden, um die verschiedenen Blickwinkel zu betrachten. Demgegenüber bieten VR Entwicklungsumgebungen erweiterte Interaktionsmöglichkeiten mit einer 3D Szene, die bei dem Entwurf interdisziplinärer Systeme sinnvoll eingesetzt werden können, z.B. um eine Simulation des Verhaltens des zu entwickelnden Produktes durchzuführen.

Die Technologie der virtuellen Realität ist in vielen Bereichen einsetzbar (vgl. Kapitel 2.6.5 auf Seite 74). Gegenwärtige Ansätze erstellen im Allgemeinen die 3D Modelle anhand von Standardmodellierungswerkzeugen wie CAD/CAM oder 3D Modeller. Diese Modelle werden dann für die Nutzung in VR optimiert und in die VR Entwicklungsumgebungen importiert (vgl. Kapitel 4.1.1 auf Seite 98). VR Entwicklungsumgebungen, auch Autorensysteme genannt, bieten Möglichkeiten zur Beschreibung des Verhaltens der 3D Objekte, und somit der Spezifikation von Interaktionen. Die mit dem Autorensystem erstellte 3D Szene inklusive der Interaktionen wird daraufhin wiedergegeben, wobei die Szene in Echtzeit berechnet werden muss (vgl. Kapitel 3.3.3 auf Seite 89). Bild 5.2 illustriert den üblichen Ablauf bei dem Einsatz von VR Autorensystemen.

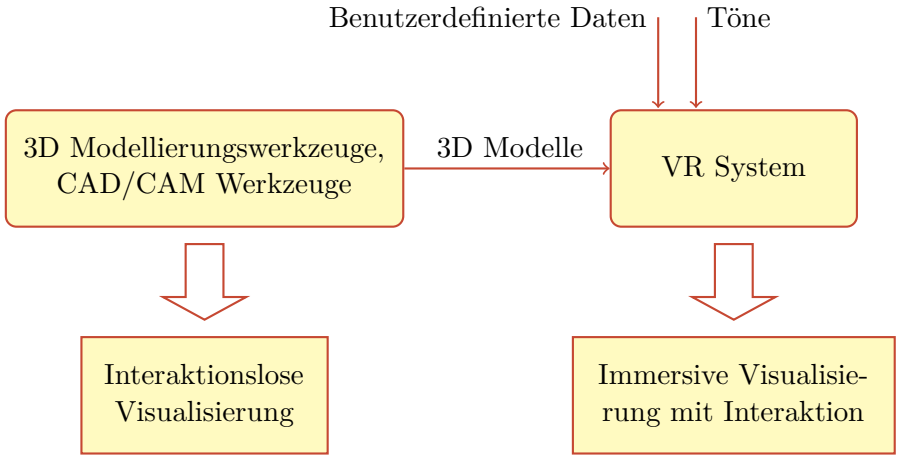


Bild 5.2: Gängiger Ansatz von VR Autorensystemen

Wichtig bei der Auswahl einer VR Entwicklungsumgebung sind die Flexibilität zur Definition der Interaktionen (eingeschränkte oder erweiterbare Aktionen), die Interoperabilität mit den Modellierungswerkzeugen (Unterstützung der jeweiligen Dateiformate) und die Möglichkeit, den Objekten benutzerdefinierte Daten für den Zweck einer Simulation zuordnen zu können. Tabelle 5.1 gibt ein Überblick über diese Eigenschaften bei den VR Autorensystemen Cult3D, Vizard, Vision4D und Virtools.

Kriterien	**Autorensysteme**			
	Cult3D von Cycore Systems	Vizard von World Viz	Vision 4D von Vision4D	Virtools von Dassault
Einbindung von Daten	Frei definierbare Properties für Objekte	Feste Properties	Feste Properties	Frei Definierbare Properties (Attribute) und Typen
Interaktionen	Gegebene, dynamische Sätze von Aktionen und Events	Action Library mit Skripte	Statische Beschreibung mit Timeline	Skripte (Building Blocks), erweiterbar durch SDK
Erweiterbarkeit	Java API für Entwicklung von Plugins	Python API für Plugins, nur für Rendering und Datenimport	keine API, nur kommerzielle Plugins	SDK und C++ API für Plugins zur Einbindung eigener Module
Modellformate und Interoperabilität	3DSMAX, Maya und Plasma	3DSMAX, Maya und 3DS (über Importer)	VMRL97, 3DS (über Importer)	3DSMAX, Maya, Lightwave, DirectX

Tabelle 5.1: Übersicht und Vergleich der Funktionalitäten verschiedener VR Autorensystem nach Zhou (Zhou, 2008)

In diesem Kontext stellt die VR Entwicklungsumgebung Virtools eine Software zur VR-, Präsentations- und Spieleentwicklung dar. Verallgemeinert

gesagt dient diese der Entwicklung von interaktiven 3D Visualisierungen. Sie wurde von Dassault Systèmes 2005 aufgekauft und deren Produktportfolio via 3D XML hinzugefügt. Neben Quest3D ist Virtools eines der meist verbreiteten Autorensysteme (vgl. Anhang C auf Seite 285).

Die Virtoolsumgebung besteht aus folgenden Teilen (Zhou, 2008):

- Die Grafische Benutzeroberfläche, um visuell die 3D Objekte und deren Verhalten zu bearbeiten.
- Die Behavior Engine, welche die interaktiven Handlungen verwaltet und diese bei der Visualisierungsphase ausführt.
- Die Render Engine, welche die Szene in Echtzeit berechnet und wiedergibt.
- Die VSL, um spezifische verhaltensbeschreibende Funktionen basierend auf vordefinierte Funktionen (Blöcke) grafisch zu entwickeln, ohne Vorkenntnisse einer Programmiersprache.
- Das Virtools SDK, das eine Entwicklung eigener verhaltensbeschreibender Funktionen in der Programmiersprache C++ ermöglicht, sowie die Entwicklung und das Einbinden von Plugins, Datenimport/-export, Renderer oder spezifische Befehle und Anwendungen unterstützt.

Darüberhinaus bietet Virtools 400 vordefinierte, wiederverwendbare Verhaltensbausteine, sogenannte Building Blocks, die dem Nutzer zur Verfügung stehen und welche in einer grafischen Skriptoberfläche miteinander verbunden werden können. Zudem ist ein Kollisionsmanagement, die physikalische Simulationen sowie eine integrierte Virtools Skriptsprache enthalten, die die Realisierung eigener Building Blocks ermöglichen.

Die 3D Objekte werden in gängigen 3D Modellformaten importiert und können in der Szene angeordnet und beleuchtet werden. Die visuelle Bearbeitung von Verhalten und Interaktivität erfolgt dann im Werkzeug Schematic mit Hilfe einer grafischen Darstellung von Verhaltensbausteinen und ihren Verknüpfungen. Mit Hilfe des Werkzeugs “Hierarchy Manger” lassen sich Beziehung von Objekten, beispielsweise für Unterbaugruppen, einfach realisieren. Den Objekten innerhalb der virtuellen Umgebung können Attribute zugewiesen werden, was die Kommunikation mit den Verhaltensbausteinen

durch die dadurch erhaltenen spezifischen Eigenschaften der Objekte erleichtert. Dies erfolgt mit Hilfe des Werkzeugs “Level Manager”. Ebenso können Objekte zu Gruppen zusammengefasst werden, was ebenfalls einer Vereinfachung der Verhaltensmodellierung bewirkt. Der Signalfluss wird durch Verbindungslinien zwischen den Building Blocks realisiert. Durch das Plugin Physicks Pack ist es möglich, den Objekten und der virtuellen Umgebung reale Eigenschaften, wie Schwerkraft, Gewicht, Elastizität und so weiter zuzuordnen (Maser, 2008).

Durch die Zugehörigkeit zur Firma Dassault Systems ist eine Integration in deren Produktportfolio über 3D XML sowie der kommenden Produktlebenszyklusplattform PLM 2.0 gegeben. Weiterhin sind Plugins für den Import von Modellen aus verschiedenen 3D Modellern enthalten. Bild 5.3 zeigt die Benutzeroberfläche von Virtools mit der Bibliothek der vordefinierten Verhaltensbausteine (Data Ressources und Building Blocks) auf der rechten oberen Seite, dem visualisierten dreidimensionalen Raum auf der linken oberen Seite, und einem leeren Schematic Werkzeug zur Modellierung von Verhalten mittels Behavior Blocks auf der Unterseite. Die Icons für das Umschalten der Unterseite auf die Werkzeuge Level Manager und Entity Setup zur Attributzuweisung sind ebenfalls dargestellt (Maser, 2008).

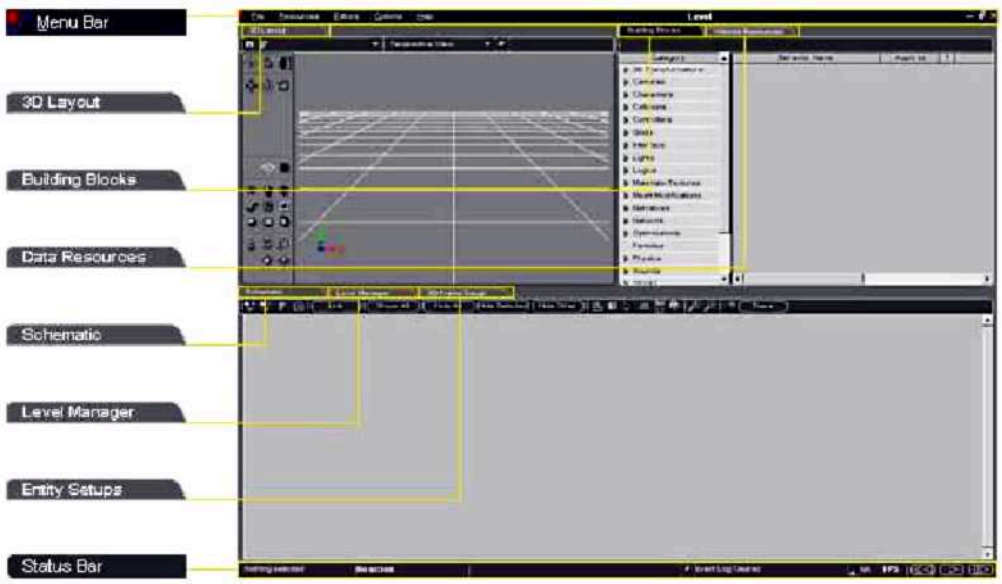


Bild 5.3: Die Virtools Benutzeroberfläche

5.2 Anwendungsszenario

Zur Verifikation des im Kapitel 4 auf Seite 93 eingeführten Konzepts wurde das Beispiel der crashaktiven Kopfstütze eines Fahrzeugsitzes herangezogen. Im Rahmen des Anwendungsszenarios soll der typische Ablauf einer Anpassungskonstruktion betrachtet werden, die in der Praxis den größten prozentualen Anteil ausmacht.

In der gegenwärtig angewandten Produktentwicklung eines Fahrzeugsitzes, wird vom Projektteam im ersten Projektschritt die Ist-Situation in einer detaillierten Umfrage festgehalten. Dabei stehen zum einen die Aufnahme der vorhandenen Best Practices, zum anderen die Aufnahme der bestehenden Defizite im Mittelpunkt. Daraufhin werden die vorhandenen Modelle und Techniken aus der vorangegangenen Produktreihe hinsichtlich ihrer Wiederverwendbarkeit analysiert. Dabei müssen die Möglichkeiten der vorhandenen Systeme in Bezug auf die Bedürfnisse der Kunden einerseits, sowie der etablierten Wertschöpfungskette andererseits berücksichtigt werden. Unterschiedliche, alternative Vorgehensweisen und Methodiken können gegebenenfalls untersucht und auf ihre Vor- und Nachteile hin geprüft werden. Basierend auf den gesammelten Daten wird die Produktgestalt mit den gängigen Methoden eines 3D-CAD Systems modelliert und abschließend firmenintern dokumentiert und archiviert. Der gesamte Produktentwicklungsprozess eines Fahrzeugsitzes kann dabei firmen- und länderübergreifend ablaufen, wobei je nach Komplexität des Sitzes ein hinreichender Aufwand an Personal, Geld und Zeit investiert werden muss.

Üblicherweise wird der Schritt der methodischen Funktionsmodellierung in der Produktentwicklung übersprungen. Die Integration von Funktionsstrukturen mit den Produktaufgabenstrukturen, wie sie aus der Anforderungsmodellierungsphase entstehen, oder in Produktstrukturen eines CAD Systems werden allenfalls am Rande umgesetzt oder sind gar nicht vorhanden. Funktionale Aspekte werden lediglich indirekt, durch angehängte Zusatzinformationen wie Toleranzangaben, Spaltmaße oder Spannstellenbeschreibungen als eine Information zur Erfüllung der Funktion *Verbindung schaffen* beschrieben (vgl. Kapitel 2.2.2 auf Seite 18).

Darüber hinaus beschreiben die während der Produktentwicklung verwendeten Systeme und Technologien insbesondere die technischen Aspekte, die zur Umsetzung der Produktidee notwendig sind. Innovative Technologien, wie die virtuelle Realität, werden in der Regel aus Kosten und Akzeptanz-

gründen nicht eingesetzt, zumal sich der Mehrwert nicht direkt quantitativ und in kurzer Zeit messen lässt.

Mit der Nutzung erweiterter, virtueller Umgebungen insbesondere in der frühen Phase der Funktionsmodellierung, wie sie in dieser Arbeit vorgestellt wurden, wird die gegenwärtige Vorgehensweise evolutioniert. Bild 5.4 fasst das zur Verifikation realisierte Verfahren zusammen.

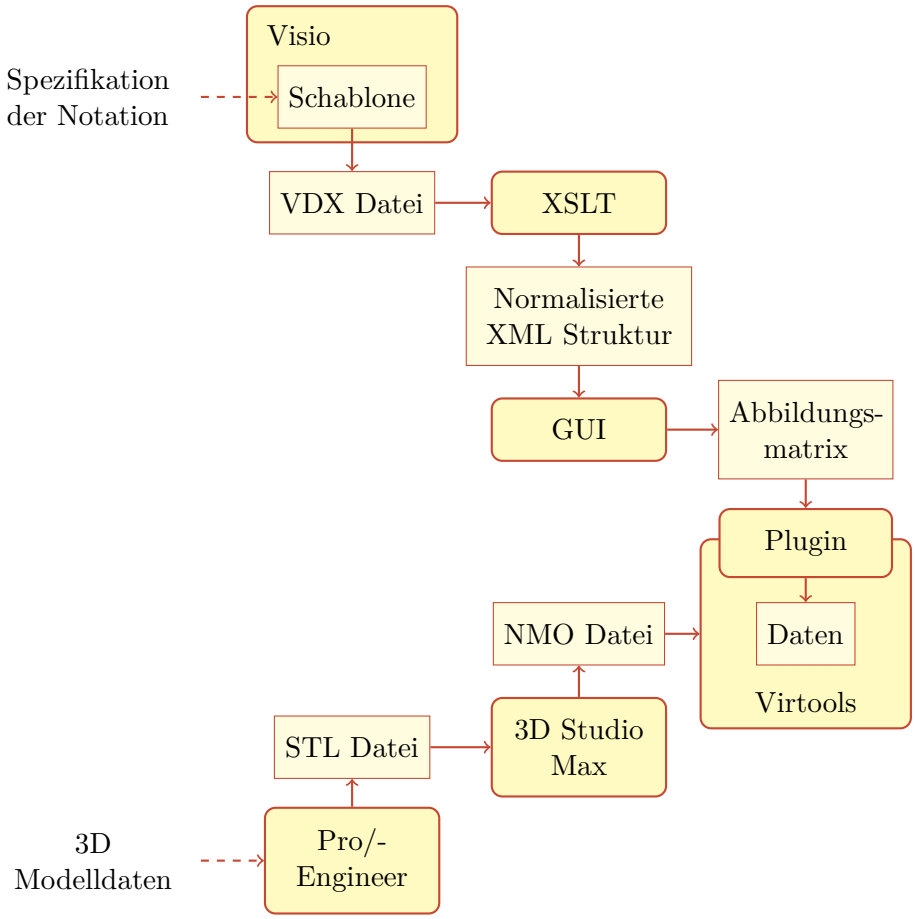


Bild 5.4: Zusammenfassung des im Rahmen der Verifiaktion eingesetzten Verfahrens

Dabei wurden die im Konzept erarbeiteten Schritte, Erstellung der 3D Umgebung, Erstellung des Funktionsmodells, Mapping der Daten in der virtuellen Umgebung, Integration der Interaktionsmetaphern sowie die Visualisierung des Funktionsmodells umgesetzt. Für die Aufarbeitung des CAD Modells wurde 3D Studio Max benutzt. Für die Darstellung der Funktionsmodelle wurde die proprietäre Modellierungssoftware Visio mit selbstentwi-

ckelten Diagrammobjekten (Shapes) verwendet. Das gezeichnete Funktionsmodell wurde daraufhin in dem Dateiformat VDX gespeichert und mit Hilfe von XSL Transformationen in ein einheitliches XML Format umgewandelt. Ein Plugin mit einer grafischen Bedienoberfläche wurde für die Abbildung mit einer Szene und die Einführung der Daten in der verwendeten virtuellen Umgebung Virtools realisiert und eingebunden. Die so in Virtools erzeugte Tabelle enthält alle notwendigen Daten der Funktionsstruktur, die abschließend mit den Interaktionsmetaphern versehen und visualisiert werden.

5.2.1 Bereitstellung des 3D Modells

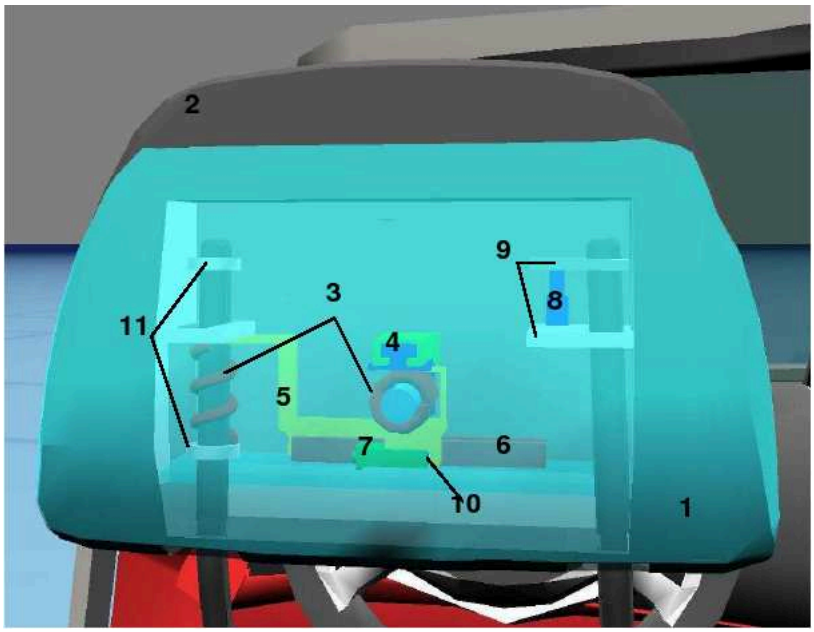
Zur Verifikation dieser Arbeit soll im Gesamtprodukt Pkw eine crashaktive Kopfstütze mit ihren spezifischen Funktionen näher untersucht werden. Dazu sind zuerst die fehlenden Elemente der Kopfstütze zu ermitteln, um diese dann in das zu großen Teilen schon bestehende, wiederverwendbare Fahrzeug zu integrieren. Da die Funktionsweise der crashaktiven Kopfstütze nicht in Details zugänglich beschrieben ist, werden die Komponenten und deren Funktionen anhand der Informationen der KÜS herausgefiltert und danach modelliert. Für einen nachvollziehbaren Funktionsablauf der Technik wurden einige Elemente noch selbständig modelliert.

Die identifizierten und modellierten Elemente sind folgende:

- Rückplatte
- Polster
- Feder horizontal/vertikal
- Verfahrschiene
- Auslöseblech
- Auslöseschiene
- Pyroaktuator
- Dämpfer
- Halterung des Dämpfers
- Auslösestift

- Federhalter oben/unten

Die modellierte crashaktive Kopfstütze mit den identifizierten Elementen zeigt Bild 5.5. Zum besseren Verständnis der Funktionsweise der Kopfstütze ist in dieser Arbeit die Rückplatte mit der Metapher halbtransparenter Objekte umgesetzt worden (vgl. Kapitel 4.3.2 auf Seite 168). Die Elemente wurden mit 3D Studio Max erstellt und in Virtools importiert.



Legende:

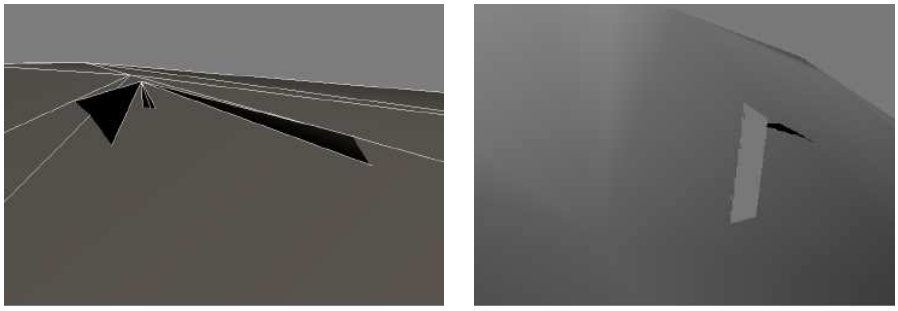
(1) Rückplatte (2) Polster (3) Federn horizontal/vertikal
(4) Verfahrschiene (5) Auslöseblech(6) Auslöseschiene
(7) Pyroaktuator (8) Dämpfer (9) Halterung Dämpfer
(10) Auslösestift (11) Federhalter oben/unten

Bild 5.5: Elemente der crashaktiven Kopfstütze

Um ein realitätsnahes Erlebnis in der VR Szene zu ermöglichen, soll neben der Realisierung der crashaktiven Kopfstütze, ebenso ein Fahrersitz, ein Fahrzeug sowie eine entsprechende Peripherie in der Szene modelliert werden. Ein im Raum schwebendes Modell wäre hier eine unbefriedigende Lösung. Als Umgebung, in der ein Fahrzeug präsentiert werden kann, sind vielerlei Möglichkeiten denkbar. Aus Gründen der Skalierbarkeit der PC Systeme wurde darauf geachtet, nicht zu detaillierte Modelle zu verwenden.

Deshalb ist eine einfache Ebene ergänzt durch einige Felsen, die zusätzlich zu der Funktionalität der VR Szene beitragen, gewählt worden. Die Modelle sind mit Texturen aufgewertet und auf Basis weniger Polygone modelliert, wie in Kapitel 4.1 auf Seite 96 beschrieben.

Da die vorhandenen Daten des Sitzes und des Fahrzeugs in einem CAD Format vorlagen, mussten diese zuerst in ein Modellierungstool überführt werden. Die Schritte zur Transformation des CAD Modells in die Virtools Szene sind in Bild 5.4 auf Seite 185 dargestellt. Diese Schritte sind notwendig, um die Daten mit Hilfe des Modellierungswerkzeuges 3D Studio Max aufzubereiten, damit zum einen die Performanz verbessert werden konnte (vgl. Kapitel 4.1.2 auf Seite 103), z.B. durch den LoD oder Culling Verfahren, sowie um verbesserte Oberflächen zu erhalten, und somit durch entsprechende Shader und Materialen eine realitätsnahe Erscheinung zu gewinnen. Insbesondere während des Exports und Imports der Daten traten dabei, aufgrund der verwendeten Algorithmen der Programme, Fehler in der Darstellung der Oberflächen auf. In Bild 5.6 sind Überlappungen von Polygonen bzw. das Durchstoßen der Oberfläche durch Polygone der Rückseite zu sehen. Fehler dieser Art wirken sich beim Rendern negativ aus, da keine eindeutige Orientierung der Oberfläche an diesen Stellen möglich ist. Es entstehen Reflexionen, die die Oberfläche uneben aussehen lassen.

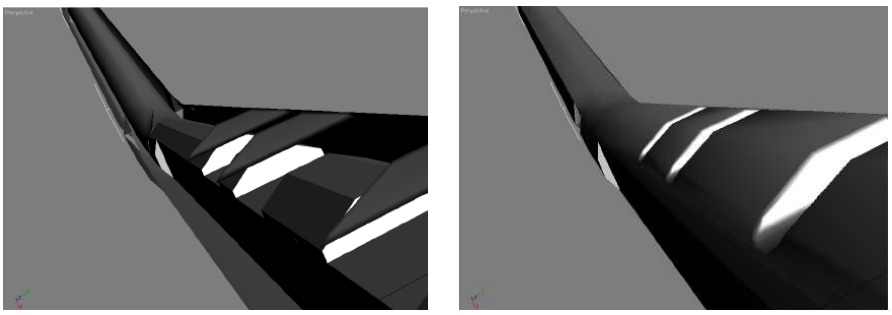


(a) Überlappte Polygone (b) Durchstoßende Polygone

Bild 5.6: Von Polygonen verursachte Oberflächenfehler

Ein weiterer, häufiger Fehler bestand in der Umkehrung der Flächennormalen der Polygone. Bei den Transformationsvorgängen von einem Dateiformat in ein anderes, werden die Flächennormalen häufig in die falsche Richtung ausgerichtet. Da die Renderer nur die positive Seite eines Polygons berechnen, werden diese Oberflächen dann durchsichtig dargestellt. Gleichzeitig wird allerdings die Innenseite dieser Oberfläche dargestellt. Ein Beispiel ist

in Bild 5.7 zu sehen. Da dieser Fehler nicht zwangsweise an allen Polygonen des Modells auftrat, war es häufig nötig, einzelne Polygone nachträglich manuell zu wenden.



(a) Fehlerbehaftet (b) Fehler korrigiert

Bild 5.7: Flächennormalenproblem

Dementsprechend muss jedes Teil hinsichtlich dieser Fehler überprüft und an den entsprechenden Stellen nachgebessert werden. Teilweise wurden dabei Objekte, mittels der Box Modelling Methode oder dem Subdivision Surface Modelling (vgl. Kapitel 4.1.1 auf Seite 99) nahezu neu modelliert, da die ursprüngliche Oberfläche nicht weiter zu gebrauchen war.

Sind alle Elemente für die Darstellung in einer virtuellen Umgebung optimiert, können diese bereits in dem Modellierwerkzeug in der Standardposition eingebaut werden. D.h. das Koordinatensystem des eingefügten Modells wird in Kongruenz mit dem Ursprungskoordinatensystem der Szene gebracht (vgl. Kapitel 4.1.1 auf Seite 99). Die so eingefügten Objekte können dann leicht neu orientiert und positioniert werden.

Für die Fahrzeuggeometrie existieren mehrere Versionen des Fahrzeugs, die sich in ihrer Detaillierung unterscheiden. Die Unterschiede liegen in der Menge der Polygone die benutzt werden (vgl. Kapitel 4.1.2 auf Seite 108). So existiert eine hoch aufgelöste Version (Hi Poly) mit über einer Million Polygonen. Die Version mit der geringsten Anzahl an Polygonen besteht aus ca. 120000 Polygonen. Es wurde zu Gunsten einer echtzeitfähigen Darstellung die Low Poly Variante des Fahrzeugs gewählt (vgl. Kapitel 2.3.2 auf Seite 30). Das Modell des Fahrzeugs beinhaltet eine Baugruppe, die aus mehreren Unterbaugruppen besteht. Bei diesen handelt es sich um Interieur, Exterieur, Türen und auch Fahrzeugsitze. Die vorhandenen Sitze, die reine

Designentwürfe ohne Funktion sind, wurden entfernt, um das Modell des hier behandelten Sitzes einzufügen (Mayer, 2006).

Nachdem Umgebung und Fahrzeug eingefügt und optimiert wurden, muss dies auch bei dem Fahrzeugsitz erfolgen. Der Sitz, dessen Teile über den in Bild 5.4 auf Seite 185 dargestellten Portierungsweg in das Modellierungswerkzeug 3D Studio max überführt wurde, bestand aus ca. 200.000 Polygonen. Diese Menge an Polygonen muss im Sinne der Skalierbarkeit wiederum reduziert werden, ohne das dabei die Geometrie zu stark verfälscht wird. Für die konzeptionell beschriebenen Methoden besteht in 3D Studio Max mittels Modifikatoren bereits die entsprechende Umsetzung. Um das Polygonnetz eines Modells zu vereinfachen, stellt das Werkzeug den Modifikator "optimieren" zur Verfügung. Dieser erlaubt es, über Parameter wie maximale Kantenlänge, die Anzahl der Polygone im Netz zu reduzieren. Dabei muss mit hoher Sorgfalt vorgegangen werden, da der Modifikator das Netz des Objekts unter Umständen stark verformt und verfälscht. Ästhetischen Gesichtspunkten gehorchend, wurden die Teile, die für die Gesamtszene wichtiger sind, weniger stark reduziert, als Teile, die zwar der Funktion nach zum Sitz gehören, jedoch für die Visualisierung eine untergeordnete Rolle spielen. Neben den Teilen der Kopfstütze waren als wichtige Teile die Lehne und das Sitzkissen eingestuft. Als eher unwichtig wurden Teile der Untergestellmechanik eingestuft. Sie sind zwar für die Funktion des Sitzes unentbehrlich, jedoch bei der Visualisierung der Szene meistens nicht sichtbar. Teile die in der Szene nicht zu sehen sind, wurden komplett weg gelassen.

Nachdem die Gesamtszene als 3D Modell erstellt, alle Objekte positioniert und optimiert wurden, muss abschließend sichergestellt werden, das alle Objekte kohärent bezeichnet wurden. Dazu erfolgt eine hierarchische Klassifizierung anhand der Produktstruktur (vgl. Kapitel 4.2.2 auf Seite 133). In der industriellen Anwendung ist die Produktstruktur bereits durch Konstruktionsabteilungen in einem PDM System hinterlegt. Im Rahmen der vorliegenden Arbeit wurde auf die Ausarbeitung Brenzingers zurückgegriffen, bei der eine Erkennung von Baugruppen und deren Überführung in eine hierarchische Klassifizierung bereits erfolgte (Brenzinger, 2007). Bild 5.8 zeigt die hierarchische Anordnung der Klassen eines Pkws. Dabei sind fünf Ebenen abgebildet, wobei die oberste Ebene, PKW, Ebene 0 darstellt. Dementsprechend folgt auf der 1. Ebene die Karosserie und der Powertrain. Da der Powertrain keine Rolle in der Verifikation spielt, wird er nicht weiter verfolgt. Auf der 2. Ebene unterhalb der Karosserie wird in Interieur und Exterieur unterschieden. Unterhalb der Klasse Exterieur sind insbesonde-

re die Stoßfänger interessant, da Funktionen der Kopfstütze hier ausgelöst werden können. Die 3. und 4. Ebene gliedern das Interieur und den Fahrersitz weiter auf.

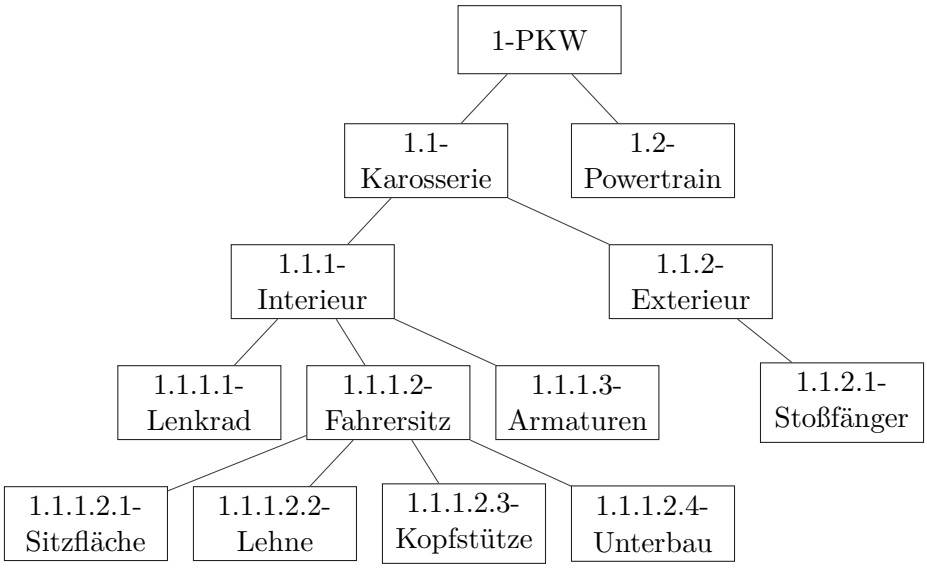


Bild 5.8: Aufbau der Klassifizierung der crashaktiven Kopfstütze

Da in der Verifikation insbesondere die Elemente der crashaktiven Kopfstütze eine zentrale Rolle spielen, wurden diese exakt klassifiziert. Bild 5.9 beschreibt die relevanten, zugehörigen Elemente zur Klasse Lenkrad, Armaturen und Stoßfänger.

1.1.1.1-Lenkrad
1.1.1.1.1-lenkrad
1.1.1.1.2-lenkachse
1.1.1.1.3-lenkrad_vorderteil
1.1.1.1.4-lenkrad_eintauchteil
1.1.1.1.5-lenkrad_dummy
1.1.1.1.6-lenkrad_dummy_mitte
1.1.1.1.7-lenkrad_steuergerät_aufprall
1.1.1.1.8-bedienelement_lenkrad_knopf1
1.1.1.1.9-bedienelement_lenkrad_knopf2
1.1.1.1.10-bedienelement_lenkrad_knopf3
1.1.1.1.11-bedienelement_lenkrad_knopfm

1.1.1.3-Armaturen
1.1.1.3.1-panel_vorn

1.1.2.1-Stoßfänger
1.1.2.1.1-sensor_heckaufprall

Bild 5.9: Elemente der 3. Klassifizierungsebene

Die Elemente der 4. Klassifizierungsebene zeigt Bild 5.10. Dazu gehören die Klasse Sitzfläche, Lehne, Lehne_motor und Kopfstütze. Die Klasse Unter-

bau ist ebenso klassifiziert, wurde aber aus Gründen der Übersichtlichkeit hier nicht dargestellt.

1.1.1.2.1-Sitzfläche
1.1.1.2.1.1-schiene_links
1.1.1.2.1.2-schiene_rechts
1.1.1.2.1.3-sitzschale_bezug
1.1.1.2.1.4-spmlh_gelenk
1.1.1.2.1.5-sp_mutter_links_hinten
1.1.1.2.1.6-sp_mutter_links_vorn
1.1.1.2.1.7-sp_mutter_rechts_hinten
1.1.1.2.1.8-sp_mutter_rechts_vorn
1.1.1.2.1.9-spindel_links
1.1.1.2.1.10-spindel_rechts
1.1.1.2.1.11-motor_schiene_links
1.1.1.2.1.12-motor_schiene_rechts
1.1.1.2.1.13-spmrv_gelenk
1.1.1.2.1.14-spmrh_gelenk
1.1.1.2.1.15-spmlv_gelenk
1.1.1.2.1.16-blende_bedienelement_rechts
1.1.1.2.1.17-blende_bedienelement_links
1.1.1.2.1.18-bedienelement_sitz_knopf_r
1.1.1.2.1.19-bedienelement_sitz_knopf_l

1.1.1.2.3-Kopfstütze
1.1.1.2.3.1-ks_bezug
1.1.1.2.3.2-ks_lager_lehne_links
1.1.1.2.3.3-ks_lager_lehne_rechts
1.1.1.2.3.4-ks_lagerung
1.1.1.2.3.5-ks_stäbe
1.1.1.2.3.6-rückplatte
1.1.1.2.3.7-feder_horizontal
1.1.1.2.3.8-feder_vertikal
1.1.1.2.3.9-verfahrschiene
1.1.1.2.3.10-verfahrschlitten
1.1.1.2.3.11-auslöseblech
1.1.1.2.3.12-dämpfer
1.1.1.2.3.13-aktuator
1.1.1.2.3.14.auslösestift
1.1.1.2.3.15-federhalter_stäbe_oben
1.1.1.2.3.16-federhalter_stäbe_unten
1.1.1.2.3.17-halter_dämpfer_stäbe

1.1.1.2.2-Lehne
1.1.1.2.2.1-lehne_bezug_vorn
1.1.1.2.2.2-lehne_bezug_hinten
1.1.1.2.2.3-lehne_tragegestell
1.1.1.2.2.4-lehne_griff
1.1.1.2.2.5-lehne_rotationsachse
1.1.1.2.2.6-lehne_motor

1.1.1.2.2.6-Lehne_motor
1.1.1.2.2.6.1-antrieb
1.1.1.2.2.6.2-anker
1.1.1.2.2.6.3-zahnrad

Bild 5.10: Elemente der 4. Klassifizierungsebene

Die Elemente dieser Klassen sind die 3D Entitys, welche die realen Objekte in der virtuellen Umgebung repräsentieren, und können ebenso in Virtools zugeordnet werden. Dies erfolgt im Level-Manager durch eine Umbenennung des Entity-Namens. Durch das Setzen der Initial Conditions (IC) wird gewährleistet, dass jedes Objekt wieder in seinen Ausgangszustand zurückgeführt werden kann. Dadurch ist es möglich, alle Funktionselemente wieder in die ursprüngliche topologische und geometrische Anordnung zu setzen.

Nach der Modellierung und Benennung der Objekte erfolgt der Export der Szene in die VR Entwicklungsumgebung Virtools. Der dazu verwendete Exporter erzeugt eine Datei im Virtoolsformat. Dabei ist es äußerst vorteilhaft, die komplette Szene in 3D Studio Max fertigzustellen und diese nicht erst in Virtools aus den einzelnen Elementen zusammenzubauen. Damit ist es möglich, zu einem späteren Zeitpunkt, falls nötig, einzelne Elemente der Szene

zu exportieren. Dies kann z.B. der Fall sein, wenn erst in Virtools festgestellt wird, dass ein Oberflächenfehler übersehen wurde. Dementsprechend kann in 3D Studio Max der Fehler behoben, und das Einzelteil erneut exportiert werden. Beim Import in Virtools wird daraufhin, über den in Bild 5.11 gezeigten Dialog abgefragt, wie mit dem vermeintlichen Duplikat verfahren werden soll. In dem geschilderten Fall ist ein Ersetzen aller vorhandener Objekte, Netze etc. gewünscht, da ein Fehler an dem Teil behoben wurde. Die geänderten Teile werden in Virtools dann an korrekter Stelle eingebaut. Ebenso bleibt die Funktionalität der Schematics, die das Teil betreffen, bestehen.

Bild 5.11: Virtools Import Dialog im Duplikatsfall

5.2.2 Darstellung und Mapping des Funktionsmodells

Nach der Erstellung der 3D Umgebung folgt der Schritt der Erstellung des Funktionsmodells und des Mappings der Funktionsmodelldaten in die virtuelle Umgebung (vgl. Bild 5.4 auf Seite 185). Für die Darstellung der Funktionsmodelle wird die proprietäre Modellierungssoftware Visio mit selbstentwickelten Diagrammobjekten (Shapes) anhand des ShapeStudios verwen-

det. Das gezeichnete Funktionsmodell wird daraufhin in dem Dateiformat VDX gespeichert und mit Hilfe von XSL Transformationen in ein einheitliches XML Format umgewandelt. Ein Plugin mit einer grafischen Bedienoberfläche wird für die Abbildung mit einer Szene und die Einführung der Daten in der verwendeten VR Entwicklungsumgebung Virtools realisiert und eingebunden.

Erstellung der Notationsschablonen mit ShapeStudio

Um die Funktionsmodellierung entsprechend der in Kapitel 4.2.1 auf Seite 120 und Kapitel 4.2.2 auf Seite 129 eingeführten Notation zu unterstützen, werden Schablonen für die Modellierungssoftware Visio, auf Basis des ShapeStudios, erstellt. Shapestudio soll im Folgenden hinsichtlich der Erstellung und Verwaltung von Schablonen kurz eingeführt werden. Danach wird gezeigt, wie diese Vorgehensweise für die Modellierung der Funktionshierarchie und der Funktionsstruktur der crashaktiven Kopfstütze angewandt wird, um die Geometrie, die Daten und das Verhalten der Diagrammelemente zu definieren.

Shapestudio ist ein Bestandteil des Visio SDK und erlaubt die Spezifikation eigener Diagrammelemente. Die damit erstellten Muster, sogenannte Master Shapes, werden in einer Datenbank versioniert und vor einer Auslieferung als Dokumentschablone kompiliert und validiert. Für die Verwendung von Shapestudio ist die Einrichtung einer Datenbank mit einer ODBC Schnittstelle notwendige Voraussetzung.

Die Generierung von Mustern mit Shapestudio wird in zwei Schritten durchgeführt. Zunächst werden die drei Aspekte Geometrie, Daten und Verhalten des Musters mit Hilfe der grafischen Oberfläche und mittels Menüs grob definiert. Danach werden diese mit Hilfe des Shapesheets verfeinert. Das Shapesheet ist eine tabellarische Darstellung der Eigenschaften eines Diagrammelements und erlaubt eine Spezifikation dieser Eigenschaften mit Hilfe von Formeln und Funktionen (vgl. Bild 5.1 auf Seite 179). Somit lassen sich sowohl geometrische als auch daten- und verhaltensbezogene Eigenschaften flexibel und eindeutig festlegen. Diese Eigenschaften sind in mehreren Tabellen (Sections) unterteilt. Eine Auflistung der in einem Shapesheet vorkommenden Sections und deren Inhalt wird im Anhang D auf Seite 291 gegeben.

Die Geometrie der Diagrammelemente wird der Notation aus Kapitel 4.2.1 auf Seite 120 und Kapitel 4.2.2 auf Seite 129 folgend definiert. Sie wird

unter Shapestudio als eine Zusammensetzung von elementaren Vektorgrafiken, wie Linien, Rechtecke, Ellipsen etc. sowie von importierten Bildern und anderen Mustern dargestellt. Grundlegend entspricht die Vorgehensweise, bei der geometrischen Spezifikation von Mustern, der Zeichnung eines Diagramms. Es werden hierfür ebenfalls grafische Elemente auf einem Zeichenblatt zusammengestellt mit den gleichen Möglichkeiten zur Festlegung der Eigenschaften, Festlegung der Ebene, Verbindung, Gruppenbildung und so weiter. Zusätzlich zu dem klassischen Zeichenblatt wird für die Gestaltung von Muster das Fenster "Master-Shape-Explorer" zur Verfügung gestellt. Dieses stellt die Struktur des Musters als Hierarchie seiner Komponenten dar und erleichtert den Zugriff auf jeden grafischen Baustein des Musters (vgl. Bild 5.12).

Die für die Unterstützung der spezifizierten Notation zu erstellenden Muster bestehen aus einfachen Vektorgrafiken wie Rechtecke, Linien und Textfelder, die durch Gruppierung zusammengeführt werden. Bild 5.12 zeigt das Zeichenblatt und den Master-Shape-Explorer des Diagrammelements zur Darstellung einer Funktionshierarchie. Wie in dem Master-Shape-Explorer zu sehen, besteht das Shape aus zwei Rechtecken ("Sheet.5" und "Sheet.6") und aus zwei Textfelder für die Bezeichnung des Funktionsnamens und der Position in der Hierarchie ("Überschrift 2" und "Überschrift 2.8"). Diese vier Grundelemente sind gruppiert ("Sheet.7") und bilden das Muster mit dem Namen "Funktionshierarchie_Objekt".

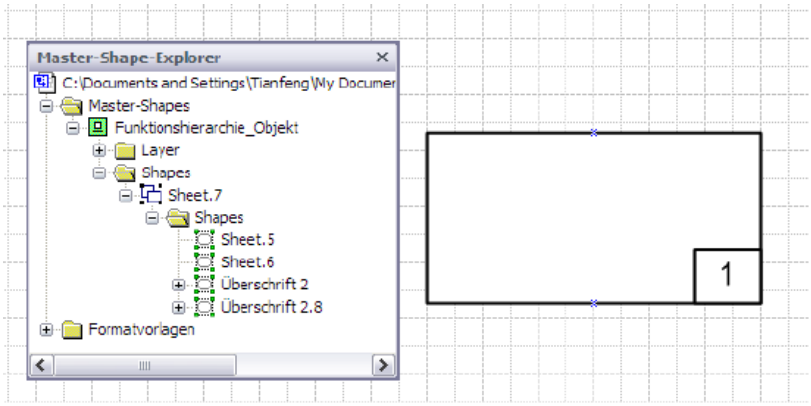


Bild 5.12: Zeichenblatt mit Master-Shape-Explorer am Beispiel des Diagrammelements der Funktionshierarchie

Die in der Funktionsstruktur verwendeten Verbindungen (Flüsse) entsprechen den bereits existierenden Verbindungen von Visio. Diese werden als

Muster eingeführt, so dass eine Unterscheidung zwischen Flüsse für Materie, Energie und Information je nach verwendeten Typ des Musters vorgenommen werden kann. Außerdem erlaubt die Spezialisierung der Standardverbindung das Einbringen der Verbindungen in dieselbe Schablone wie Funktionsverben und Funktionsobjekte und somit eine intuitive Einteilung der Diagrammelemente in der Zeichnungsumgebung.

Für die Modellierung der Funktionshierarchie werden vier geometrisch verschiedene Bausteine benötigt: Gesamtfunktion, Teilfunktion, unerwünschte Funktion und Elementarfunktion (vgl. Bild 4.2.1 auf Seite 120). Trotz der Verschiedenheit der Darstellungen dieser Bausteine beziehen sich diese auf die gleiche Semantik, und sollen daher als ein einziges konfigurierbares Diagrammelement dargestellt werden.

Dies wird durch die Einführung eines Parameters zur Feststellung des Typs und die Beschreibung eines Verhaltens, das die Geometrie des Bausteins verändert, realisiert. Somit wird ein einziges Muster für die vier Varianten erstellt und die Übersichtlichkeit der Schablone verbessert. Dabei können benutzerdefinierte Daten (Eigenschaften) an einem Diagrammelement angehangen werden. Dies kann über die Maske "Shape-Daten definieren" (vgl. Bild 5.13) erfolgen. Jede selbstdefinierte Eigenschaft hat eine frei definierbare Bezeichnung und einen Typ (Zeichenkette, Nummer, Feste oder variable Liste, Boolesch etc.).

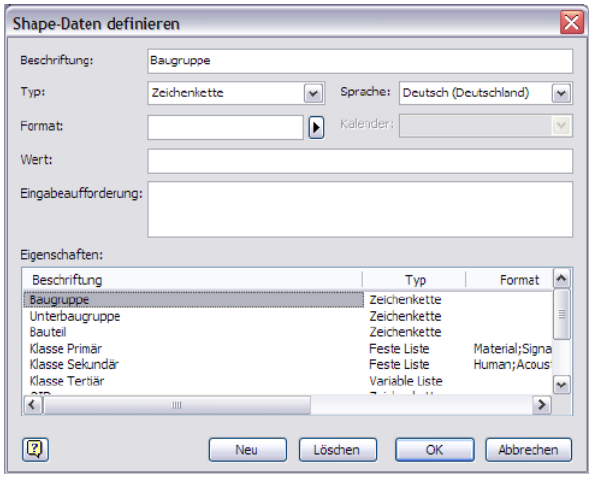


Bild 5.13: Dialog zur Definition von benutzerspezifischen Daten unter Visio

Tabelle 5.2, Tabelle 5.3 und Tabelle 5.4 fassen die jeweiligen definierten Daten für die Diagrammelemente Funktionsobjekt, Funktionsverb und Funk-

tionshierarchie zusammen. Die Eigenschaften zur Kennzeichnung, Nummerierung und Versionierung werden als Zeichenkette betrachtet, so dass eine hierarchische Nummerierung mit Hilfe von Punkten wie beispielsweise 1.1.2 dargestellt werden kann.

Eigenschaft	**Typ**	**Default**
Baugruppe	Zeichenkette	
Unterbaugruppe	Zeichenkette	
Bauteil	Zeichenkette	
Klasse Primär	Feste Liste	
Klasse Sekundär	Feste Liste	
Klasse Tertiär	Feste Liste	
O-ID	Zeichenkette	“1”
Position Funktionshierarchie	Zeichenkette	
Version	Zeichenkette	“1”
Attribut	Zeichenkette	

Tabelle 5.2: Eigenschaften des Funktionsobjekt

Eigenschaft	**Typ**	**Default**
Funktionsverb	Zeichenkette	
Klasse Primär	Feste Liste	
Klasse Sekundär	Feste Liste	
Klasse Tertiär	Feste Liste	
V-ID	Zeichenkette	“1”

Tabelle 5.3: Eigenschaften des Funktionsverbs

Eigenschaft	**Typ**	**Default**
Funktion	Zeichenkette	
Typ	Feste Liste	Teilfunktion
ID	Zeichenkette	“1”

Tabelle 5.4: Eigenschaften der Funktionshierarchie

Die Eigenschaften Klasse Primär, Klasse Sekundär und Klasse Tertiär der Diagrammelemente Funktionsobjekt und Funktionsverb stellen die NIST

Taxonomie für die Klassifizierung der Funktionsobjekte und Funktionsverben dar (vgl. Kapitel 4.2.2 und Anhang A). Um die Arbeit des Entwicklers zu erleichtern, aber auch um die Eingabe auf sinnvolle und bearbeitbare Werte einzuschränken, sollten hier Vorschläge für mögliche Werte vorgegeben werden. Daher werden diese Eigenschaften als feste Liste realisiert, die jeweils die Auswahlmöglichkeiten aus der NIST Taxonomie für jede Ebene anbieten. Da die Auswahlmöglichkeiten der Klasse Sekundär von den ausgewählten Elementen der Klasse Primär abhängig ist, soll die Liste dynamisch angepasst werden. Das gleiche gilt für die Auswahlmöglichkeiten der Klasse Tertiär, die von der Auswahl der Klasse Sekundär abhängig ist. Dazu wird auf das Shapesheet zurückgegriffen.

In dem Shapesheet eines Diagrammelements werden die benutzerdefinierten Daten (Eigenschaften) in der Tabelle (Section) "Shape Data" zusammengefasst. Bild 5.14 illustriert der Inhalt der Tabelle am Beispiel des Musters Funktionsobjekt. Da die Werte in dem Shapesheet mit Hilfe von Funktionen beschrieben werden können, kann eine flexible Spezifikation der Auswahlmöglichkeiten vorgenommen werden.

Shape Data	Label	Prompt	Type	Format	Value	SortKey	Invisible	Ask	LangID	Calendar
Prop.Row_1	"Baugruppe"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_2	"Unterbaugruppe"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_3	"Bauteil"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_4	"Klasse Primär"	No Formula	1	"Material;Signal;Energy"	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_5	"Klasse Sekundär"	No Formula	1	GUARD(IF(STRSAME(Prop	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_6	"Klasse Tertiär"	No Formula	1	GUARD(IF(STRSAME(Prop	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_7	"OID"	No Formula	0	No Formula	1	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_8	"Position Funktionshier	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_9	"Version"	No Formula	0	No Formula	1	No Formula	No Formula	No Formula	1031	No Formula
Prop.Row_10	"Attribut"	No Formula	0	No Formula	""	No Formula	No Formula	No Formula	1031	No Formula

Bild 5.14: Tabelle der Eigenschaften Shape Data eines Funktionsobjekts in Shapesheet

Die Auswahlmöglichkeiten der Eigenschaften vom Typ Feste Liste werden in der Spalte Format als eine Liste von Werten, die durch Semikolon getrennt sind, dargestellt. Da für die Definition der Auswahlmöglichkeiten für die Eigenschaften Klasse Sekundär/ Tertiär jeweils mehrere solcher Listen, gemäß der Auswahl der oberen Ebene, definiert werden, benötigen diese entsprechend viele Zellen für die Definition. Zur Definition eigener Zellen steht unter Shapesheet die Tabelle "User-Defined Cells" zur Verfügung. Dort werden die unterschiedlichen Auswahllisten gespeichert.

Das Selektieren der passenden Auswahlliste wird durch eine Funktion in der Zelle Format der jeweiligen Eigenschaften realisiert. Folgend ist die Funktion

zur Auswahl der geeigneten Liste für die Eigenschaft Klasse Sekundär des Funktionsobjekts als Beispiel gegeben:

```
GUARD (IF(STRSAME(Prop.Row_4,"Material"), User.Material_S,
       IF(STRSAME(Prop.Row_4,"Signal"), User.Signal_S,
          User.Energy_S)))
```

Die Funktion GUARD sorgt dafür, dass die zur Laufzeit ausgewählte Liste, die innere Funktion nicht überschreibt, und dass immer das von der Funktion gelieferte Ergebnis verwendet wird. Prop.Row_4 bezeichnet den Text des ausgewählten Elements für die Eigenschaft Klasse Primär. Dieser wird mit der Zeichenkette Material verglichen. Sind die beiden Texte gleich, so wird die Liste aus der Zelle User.Material_S in der Tabelle User-Defined Cells gelesen und als Ergebnis geliefert. Ansonsten wird die Zeichenkette weiter verglichen, bis die entsprechende Liste ermittelt wurde.

Die in der Form von Eigenschaften angehängten Daten sollen entsprechend der Notation in dem Diagrammelement dargestellt werden, so dass das resultierende Funktionsmodell für Menschen lesbar sowie rechnerverarbeitbar bleibt. Dazu sollen die in den Textfeldern angezeigten Texte bei einer Änderung der Eigenschaften aktualisiert werden. Hierfür stellt Visio die Funktion Datengrafik bearbeiten zur Verfügung, mittels der Textfelder erzeugt werden können, die mit Objektdaten verknüpft sind und dadurch den Text automatisch zu den Daten anpassen.

Die menügeführten Einstellungen der Datengrafik sind jedoch nicht flexibel genug, um die komplette Konfiguration der Datengrafik zu unterstützen. Aus diesem Grund wird auf das Shapesheet der Datengrafik zurückgegriffen, um deren Einstellungen zu verfeinern. Die anzuzeigende Eigenschaft des Diagrammelements wird über die Zelle Prop.msvCalloutField in der Tabelle Shape Data referenziert. Durch das Ersetzen der Formel in dieser Zelle wird die Anpassung des anzuzeigenden Texts vorgenommen. Exemplarisch ist die Formel zur Verkettung der Eigenschaften Klasse Primär, Klasse Sekundär und Klasse Tertiär in der Kopfzeile des Funktionsobjekts angegeben:

```
User.Cut1 & IF(STRSAME(User.Cut2,""),"","/")} &
User.Cut2 & IF(STRSAME(User.Cut3,""),"","/")} &
User.Cut3
```

Der Operator & dient zur Verkettung von Zeichenketten. Dabei werden die Inhalte der Zellen User.Cut1, User.Cut2 und User.Cut3 verkettet, und das

Begrenzungszeichen / im Falle einer folgenden, nicht leeren Zeichenkette hinzugefügt. Die anzuzeigenden Eigenschaften werden nicht direkt referenziert, stattdessen der Inhalt der benutzerdefinierten Zellen. Diese Zellen verknüpfen die darzustellenden Texte und enthalten eine Formel, die dafür sorgt, dass die Länge der Texte eingeschränkt wird.

Neben Geometrie und Daten bietet Visio auch die Möglichkeit, das Verhalten eines Diagrammelements zu definieren. Damit kann z.B.mit einem Doppelklick auf dem Diagrammelement eine Aktion zugewiesen werden, oder festgelegt werden, welches Zeichenelement ausgewählt wird, wenn auf einer Gruppe geklickt wird. Mittels des Shapesheets kann auf die angebotenen Funktionalitäten im vollen Umfang zugegriffen werden.

Die Tabelle "Connection Points" spielt eine wichtige Rolle in der Verhaltensspezifikation, da hier die Verbindung zwischen zusammengesetzten Gruppen festgelegt wird. Somit ist das Anhängen eines Verbinders lediglich an die Gruppe möglich. Dies hat für die Visualisierung des Diagramms unter Visio keinen Einfluss, ist aber von starker Bedeutung bei der Extraktion der Semantik aus dem Funktionsmodell, da in diesem Schritt ermittelt werden soll, welche Instanzen von Mustern zusammen verbunden sind. Ist ein Verbinder nicht an einer Instanz selbst gebunden, sondern an einer seiner Teilkomponenten, so ist diese Ermittlung behindert, da der referenzierte Komponenten-IDs von der erwarteten ID abweicht.

Zudem können Anpassungen an die Geometrie eines Diagrammelements vorgenommen werden, falls sich die Daten (Eigenschaften) geändert haben. Dies ist z.B. der Fall, wenn der Typ des Diagrammelements für Funktionshierarchie entweder als Gesamtfunktion, Teilfunktion, unerwünschte Funktion oder Elementarfunktion gesetzt wird. Der Stil der Linien und/oder die Hintergrundfarbe werden der Notation entsprechend angepasst. Für die Einstellungen der genannten geometrischen Eigenschaften sind die Tabellen "Fill Format" und "Line Format" zuständig. Bei der Eingabe einer den Typ referenzierenden Formel in diesen Zellen ist eine dynamische Anpassung der Eigenschaften in Abhängigkeit von dem ausgewählten Typ möglich. Folgend ist die Formel zur Feststellung des Linienstils in der Zelle LinePattern beispielhaft angegeben:

```
GUARD (IF(STRSAME(Sheet.7!Prop.Row_2,"Elementarfunktion")
       ,2,1))
```

Wenn die Zelle Prop.Row_2 des Elements Sheet.7, d.h. die Eigenschaft Typ des Funktionshierarchieelements, die Zeichenkette Elementarfunktion

enthält, so wird der Linienstil auf “2” gesetzt, ansonsten auf “1”. Dabei entspricht der Linienstil vom Typ 1 einer vollen Linie und der Linienstil vom Typ 2 einer gestrichelten. Analog werden die Linienbreite und Hintergrundfarbe mit Formeln berechnet, so dass der Typ des Funktionshierarchieelements berücksichtigt wird.

Nachdem die Aspekte Geometrie, Daten und Verhalten der Diagrammelemente für die rechnerunterstützte Erstellung von Funktionshierarchie und Funktionsstruktur festgelegt wurden, konnte die Dokumentschablone vollständig entwickelt werden. Bild 5.15 zeigt die Schablone.

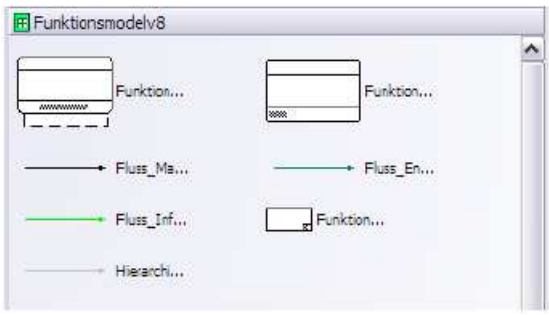


Bild 5.15: Visio Schablone zur Funktionsmodellierung

Durch die kohärente Bezeichnung der Objekte in Kapitel 5.2.1 lassen sich die bereits klassifizierten Elemente eindeutig zuordnen und können nun, mittels der Dokumentschablone (vgl. Bild 5.15) aus ihrem geometrischen Zusammenhang heraus als Funktionsstruktur am Rechner dargestellt werden. Damit diese Funktionsstruktur modelliert werden kann, müssen die funktionalen Zusammenhänge mittels der Funktionshierarchie ermittelt werden. In Bild 5.16 ist die notationsgerechte Funktionshierarchie für die crashaktive Kopfstütze abgebildet. Hierbei sind nicht alle Funktionen der Kopfstütze abgebildet, sondern nur der Teil, der sich mit dem Insassenschutz bei einem Heckaufprall befasst.

Die Funktionsstruktur behandelt das crashaktive Verfahren der Kopfstütze bei einem Heckaufprall auf einen PKW. Als Basis zur Findung der Funktionselemente dient die Funktionshierarchie aus Bild 5.16. Der Aufprall wird von einem Sensor detektiert und als Signal an ein Steuergerät weitergeleitet. Dort wird diese Information mit einem Wert verglichen, der als Schwelle für das Auslösen der Schutzmaßnahmen dient.

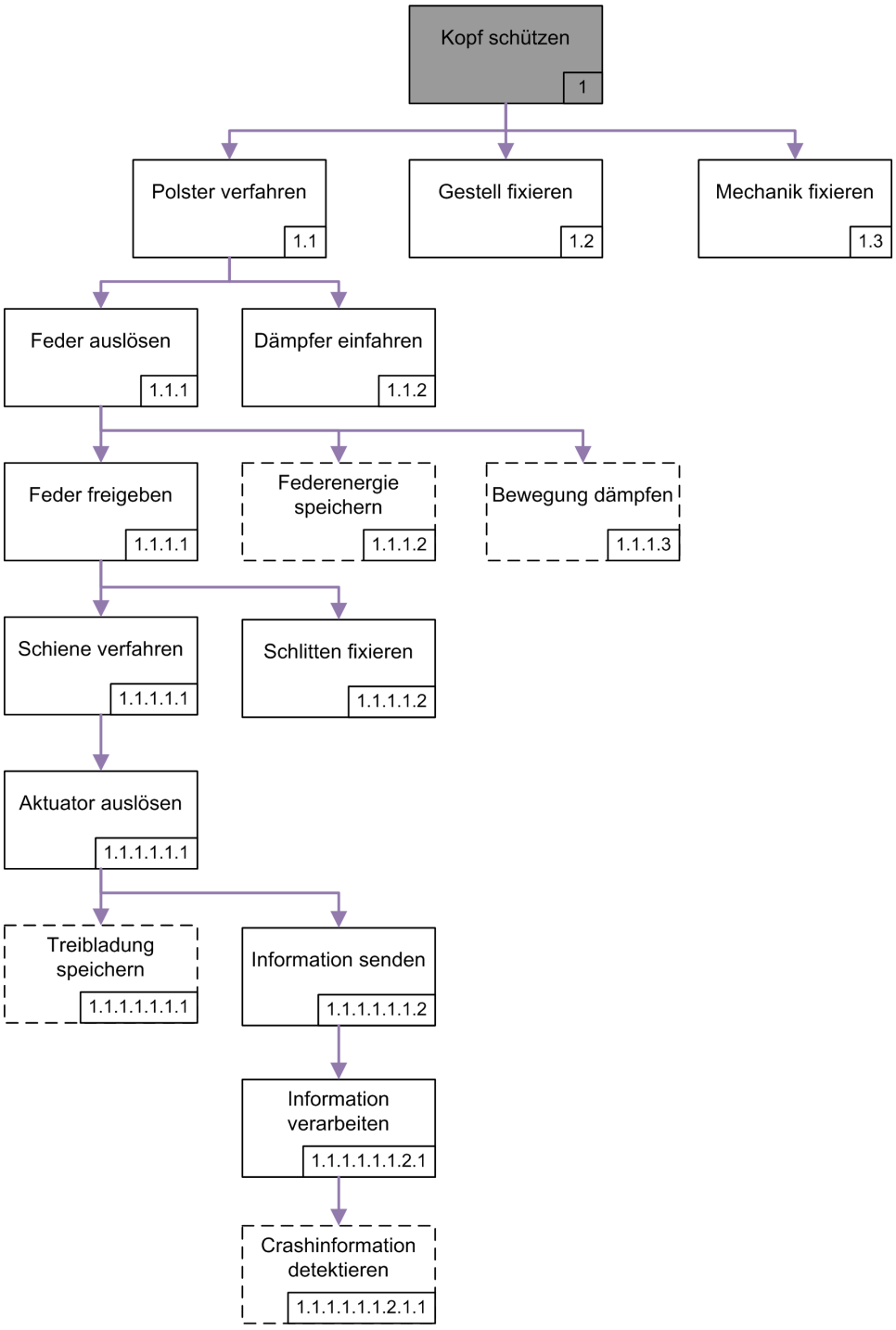


Bild 5.16: Funktionshierarchie der crashaktiven Kopfstütze

Bei Überschreitung dieser Schwelle wird ein Signal an einen Pyroaktuaor geleitet, der daraufhin ausgelöst wird. Dieser verfährt ein Auslöseblech, welches die gespeicherte Energie von zwei Schraubenfedern freigibt. Mittels Dämpfer und Führungsschienen wird die Bewegung der Kopfstütze gerichtet nach vorn (Richtung Insasse) und oben ausgeführt. Bild 5.17 und Bild 5.18 zeigen die konkrete Funktionsstruktur für das Verhalten eines Auslösevorgangs der Kopfstütze, welches nochmals untenstehend aufgeführt wird.

- Sensor im Heckstoßfänger detektiert einen Aufprall
- Sensor leitet das Signal an das Airbagsteuergerät
- Airbagsteuergerät vergleicht die Aufprallstärke mit einem Schwellenwert
- Airbagsteuergerät errechnet ein Überschreiten des Schwellenwertes
- Airbagsteuergerät schickt ein Signal an den Pyroaktuator
- Pyroaktuator verschiebt Auslöseblech
- Auslöseblech gibt Federenergien der Schraubenfedern frei
- Kraft der Schraubenfedern verfährt die Kopfstütze in horizontaler und vertikaler Richtung
- Dämpfer steuert die Bewegungskraft und Bewegungsgeschwindigkeit der Federn
- Führungsschienen steuern die Bewegungsrichtung

Bild 5.16, Bild 5.17 und Bild 5.18 stellen die gezeichnete Funktionshierarchie als auch die Funktionsstruktur anhand der Visio Dokumentschablone beispielhaft dar.

XSL Transformation

Nachdem die Funktionshierarchie und die Funktionsstruktur mit Hilfe von Visio und der entwickelten Schablone gezeichnet worden sind, kann das erstellte Diagramm im dem auf XML basierenden Visio VDX Dateiformat gespeichert werden. Da das Speicherformat VDX darstellungsorientiert ist

Bild 5.17: Funktionsstruktur der crashaktiven Kopfstütze (a)

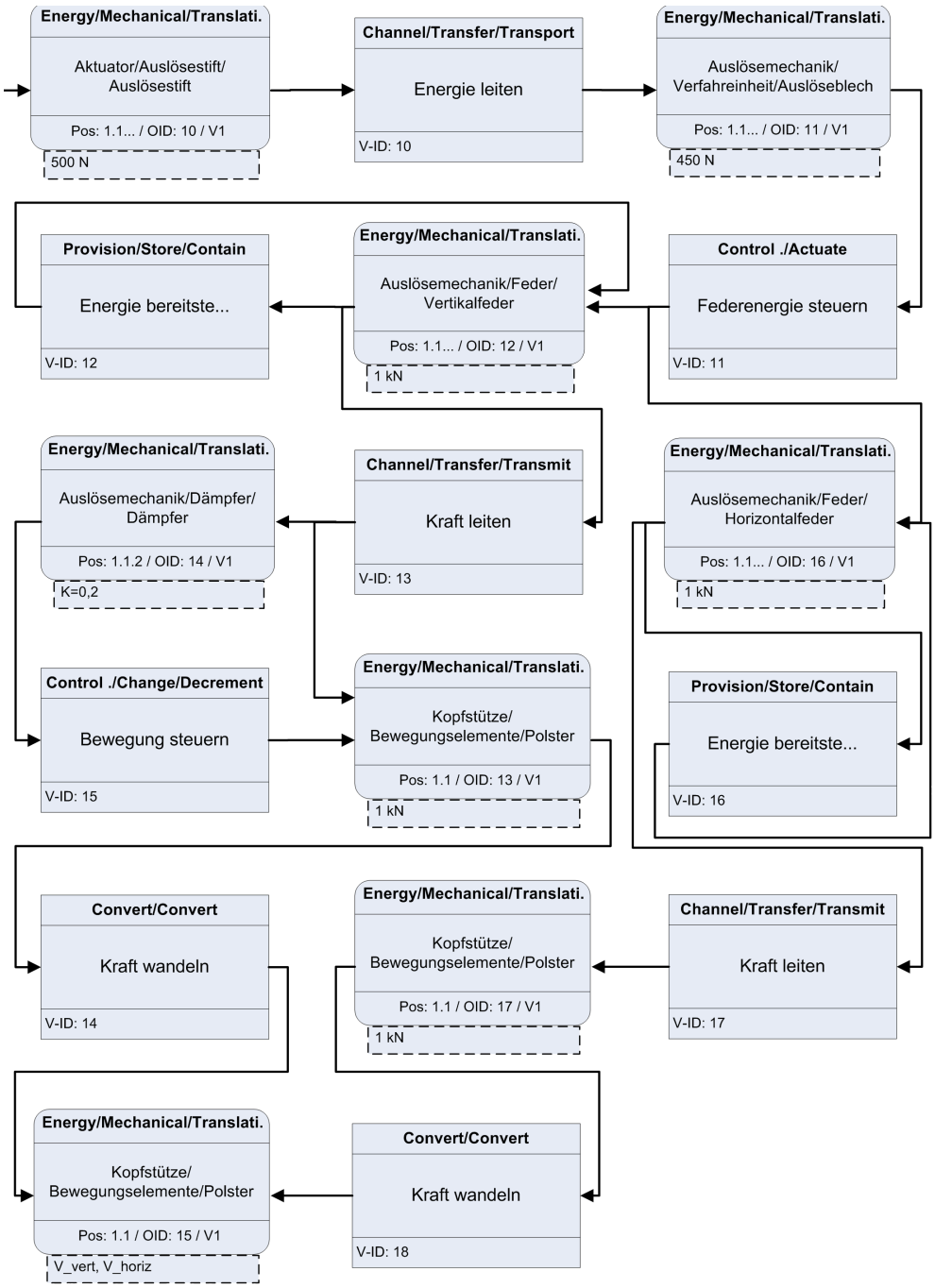


Bild 5.18: Funktionsstruktur der crashaktiven Kopfstütze (b)

(vgl. Kapitel 4.2.3 auf Seite 138), enthält dieses viele Informationen zur Gestaltung und Anzeige des Dokuments, die aber keine Semantik eines Funktionsmodells darstellen. Daher ist die Extraktion relevanter Daten bzw. Filterung der Datei hilfreich, bevor diese in die Szene gemappt werden kann.

VDX ist ein Datenformat für den Export von Visio Zeichnungen, welches in der Sprache XML definiert ist. Es enthält alle Charakteristiken einer mit Visio erstellten Zeichnung, so dass die Zeichnung nach einem erneuten Laden in ihrer Darstellung exakt wieder regeneriert werden kann. Der Inhalt einer VDX Datei kann mit Hilfe eines XML Standardeditors angezeigt und bearbeitet werden. Ein VDX Dokument besteht grundsätzlich aus Styles, Masters, Pages, Shapes und Connections (vgl. Bild 5.19).

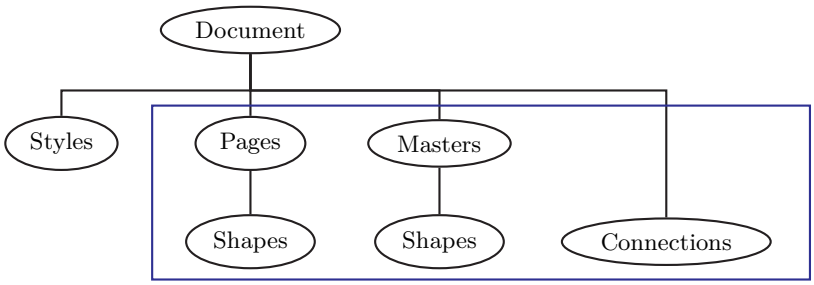


Bild 5.19: Visio XML Schema

Styles definieren die gesamte Gestaltung des Dokuments, z.B. die Eigenschaften der Texte, Linien, Ereignisse und so weiter. Masters enthält die Definition aller in dem Dokument verwendeten Muster. Ein Muster ist ein Template für ein Diagrammelement, das in mehreren Diagrammelementen rekursiv eingeteilt werden kann. Unter dem Knoten Pages befinden sich die aktuellen Zeichnungen, die aus Instanzen von Mustern (Shapes) bestehen. Shapes sind die Basiseinheit der Zeichnung, und bestehen aus verschiedenen geometrischen Primitiven und deren Eigenschaften, die in dem Muster definiert wurden. Zusätzlich kann ein Shape eigene Eigenschaften und geometrische Primitiva enthalten, unabhängig von seinem ursprünglichen Muster. Unter Connections werden Informationen über Verbindungen, d.h. wie die Shapes miteinander verknüpft sind, gespeichert. Bild 5.20 stellt einen Ausschnitt eines mit Visio gezeichneten Funktionsmodells in dem VDX Format dar.

Die Kommunikation zwischen dem Modellierungswerkzeug Visio und der VR Entwicklungsumgebung Virtools erfolgt über eine einheitliche XML

Schnittstelle, was die Verknüpfung anderer Werkzeuge ermöglicht, so dass Anpassungen an der Notation vorgenommen werden können, ohne die Übermittlungswerkzeuge neu zu implementieren. Dabei ist diese Schnittstelle in der Lage, alle im Rahmen der Funktionsmodellierung relevanten Informationen zu übermitteln (vgl. Kapitel 4.2.3 auf Seite 138).

Bild 5.20: VDX XML Struktur des Verifikationsbeispiels

Im Rahmen des Verifikationsbeispiels werden die Informationen aus der Funktionshierarchie und der Funktionsstruktur in Virtools abgebildet. Das Bild 5.21 auf der nächsten Seite illustriert die aus dem in Bild 4.32 auf Seite 141 im Konzept erarbeiteten Lösung für eine normierte Ausgangs XML Struktur. Unter der Wurzel Funktionsmodellierung werden die Daten aus der Funktionsstruktur nach dem Aspekt objektorientiert und aus der Funktionshierarchie nach dem Aspekt funktionsorientiert aufgeteilt. Jedes Objekt in der XML Struktur entspricht einem Diagrammobjekt (Shape) in der Visiozeichnung. Unter dem Knoten Eigenschaft sind die Attribute und Werte des Diagrammobjekts gespeichert. Die Verbindungen stellen die durch einen Funktionsfluss paarweise verbundenen Objekte dar, die nach der eingesetzten Notation jeweils ein Funktionsobjekt mit einem Funktionsverb verknüpfen. Der Knoten Funktion in der Funktionshierarchie speichert die Attribute Name, Typ und ID der in der Funktionshierarchie modellierten Funktionen.

Diese vereinheitlichte XML Struktur ist als Zwischensprache zwischen Modellierungswerkzeug (hier Visio) und virtuelle Umgebung (hier Virtools)

selbsterklärend und enthält nur die, für die Übermittlung der Semantik des Funktionsmodells, relevanten Informationen und erleichtert somit die Implementierung einer Schnittstelle zu der VR Entwicklungsumgebung, da die Informationen zur Darstellung abstrahiert werden können.

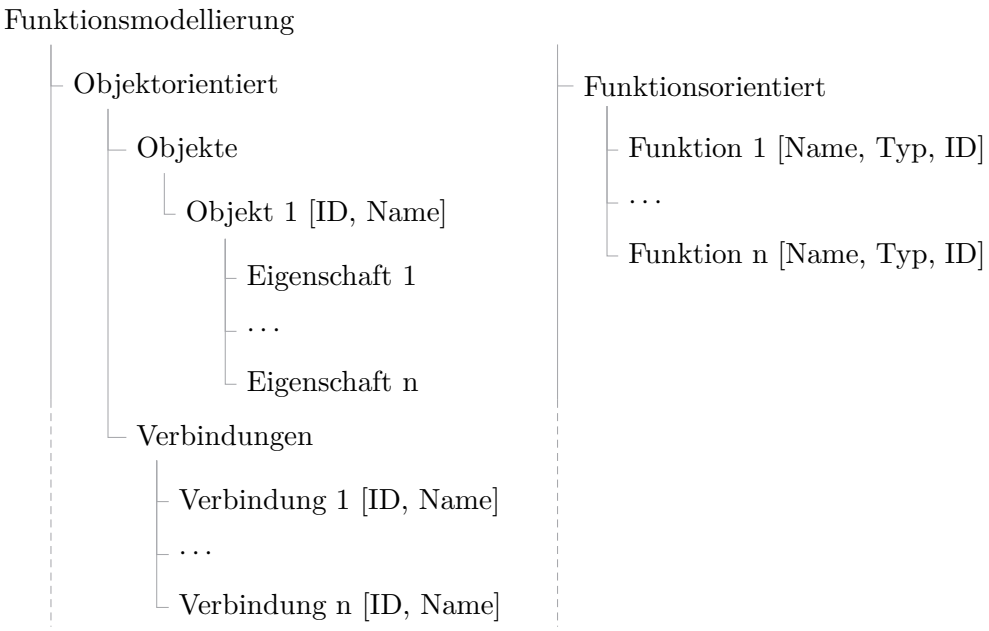


Bild 5.21: XML Schnittstelle des Verifikationsbeispiels

Wie in Kapitel 4.2.3 auf Seite 138 beschrieben wurde, ist das VDX Format darstellungsorientiert und enthält viele Informationen über die Gestaltung des Dokuments, die die eigentliche Logik des Diagramms verbergen. Mithilfe von XSLT können die notwendigen Informationen aus einer größeren Visiozeichnung abstrahiert und in die definierte XML Zielstruktur umgewandelt werden. Die XSL-Transformation wird auf das VisioXML Dokument angewandt, um ein neues XML Dokument nach der in Bild 5.21 eingeführten Struktur zu gewinnen.

Die Auswahl bestimmter zu transformierender Knoten in einem XML Dokument wird mit Hilfe von XPath Anfragen realisiert. Ein XPath Ausdruck adressiert Teile eines XML Dokuments, das heißt einen Knoten, eine Menge von Knoten oder auch Teilbäume. Die möglichen, adressierbaren Knoten eines XML Baumes sind die XML Elemente, XML Attribute, XML Textknoten, XML Kommentare, XML Namensräume und XML Verarbeitungsanweisungen.

Das Bild 5.22 illustriert die Umstrukturierung des VDX Dokuments in der definierten einheitlichen XML Struktur und die vorgenommenen Transformationen der gespeicherten Daten. Die mit gleicher Farbe markierten Knoten in den zwei XML Bäumen weisen gleichartige Informationen auf, d.h. dass die Knoten in der Ziel-XML-Struktur die Informationen der gleichfarbig gezeichneten Knoten der Quell-XML-Struktur enthalten.

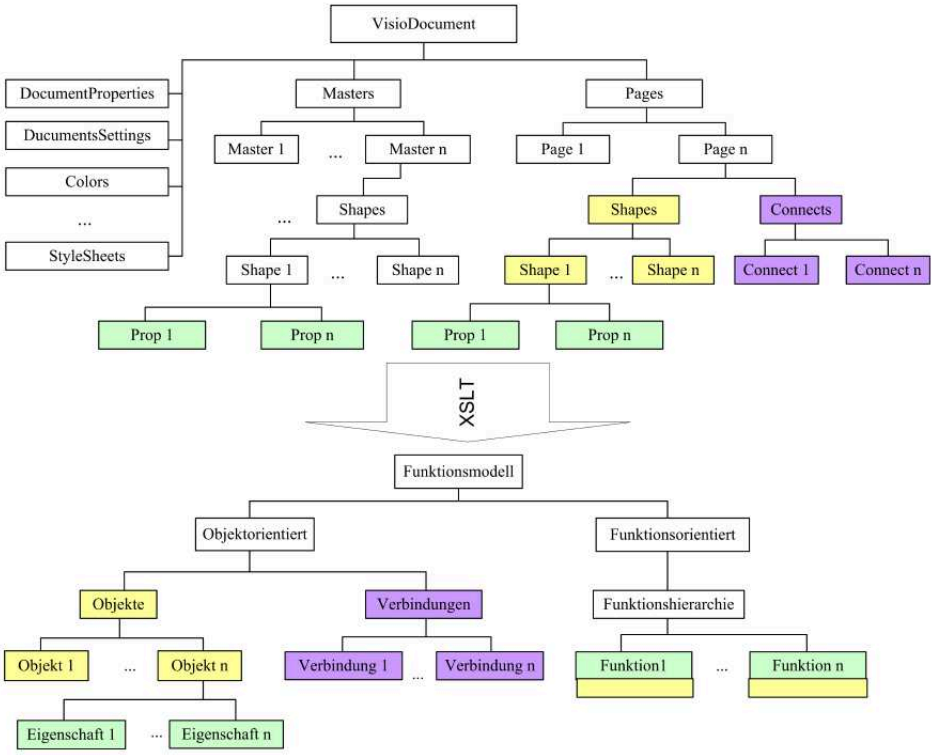


Bild 5.22: Struktur der XML Quelle und des XML Ziels

Die Funktionsverben, Funktionsobjekte und Flüsse sind im VDX Dokument als Schablone dargestellt (vgl. Kapitel 5.2.2 auf Seite 193). Die Informationen der Zeichnungsobjekte werden an zwei Stellen im VDX Dokument gespeichert. Unter Master befinden sich alle Informationen einer Klasse, wie z.B. die Defaultwerte von Eigenschaften etc. und unter Pages/Shapes befinden sich die instanzspezifischen Informationen wie z.B. die für eine Instanz eingegebenen Werte der Eigenschaften. Ein Großteil der Informationen der Zeichnungsobjekte sind in den Pages/Shapes abgelegt. Unter Shapes/Connections sind die eingebundenen Shapes paarweise durch deren ShapeID referenziert und gespeichert. Durch eine Suche nach der ShapeID

unter dem Knoten Pages/Shapes in der Quelldatei kann festgestellt werden, welche Zeichnungsobjekte gruppiert sind. Über die Typisierung des Konnektors (“BeginX” oder “EndX”) werden schliesslich die Verbindungen und deren Richtungen zwischen Funktionsverben und Funktionsobjekten ermittelt. Eine XSL-Transformation für die Übernahme der Eigenschaften in das Zielformat, in Bild 5.22 grün gekennzeichnet, sieht als Pseudocode beispielsweise so aus:

```
Für alle Objekte in der Quell-XML-Struktur
(/VisioDocument/Pages/Page/Shapes/Shape):
    Notiere Name des Masters M
    Füge die Eigenschaften
    (/VisioDocument/Pages/Page/Shapes/Shape/Prop)
    als Knoten in die Ziel-XML-Struktur hinzu.
Für alle Eigenschaften des Masters M
(/VisioDocument/Masters/Master/Shapes/Shape/Prop):
    Überprüfe, ob die Eigenschaft in der
    Quell-XML-Struktur bereits vorhanden ist.
         Ja:   Ignorieren
         Nein: Schreibe Eigenschaft in Ziel-XML-Struktur
```

Analog zu dem oben angeführten Pseudocode kann die XSL-Transformation implementiert werden. Eine Referenz der XSLT Befehle befindet sich im Anhang E auf Seite 297. Exemplarisch ist die resultierende XSL Regel nachfolgend dargestellt.

Ausschnit der XSL Regel ’objektorientiert’

```
<?xml version="1.0" encoding="iso-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:vis='http://schemas.microsoft.com/visio/2003/core'>

<xsl:template match="/">
 <Funktionsmodell>

  <Objektorientiert>
   <Objekte>
     <xsl:for-each select="/vis:VisioDocument/vis:Pages/vis:Page/
        vis:Shapes/vis:Shape[@Master=/vis:VisioDocument/vis:Masters/
        vis:Master[@NameU='Funktionsobjekt']/@ID or @Master=/
        vis:VisioDocument/vis:Masters/vis:Master[@NameU='
        Funktionsverb']/@ID or  @Master=/vis:VisioDocument/
        vis:Masters/vis:Master[@NameU='Fluss_Materien']/@ID or
```

```
      @Master=/vis:VisioDocument/vis:Masters/vis:Master[@NameU='
      Fluss_Energie']/@ID or @Master=/vis:VisioDocument/vis:Masters
      /vis:Master[@NameU='Fluss_Information']/@ID]">
    <xsl:variable name="currentID" select="./@ID">
    </xsl:variable>
    <xsl:variable name="currentMaster" select="./@Master">
    </xsl:variable>

   <Objekt>
    <xsl:attribute name="ID">
      <xsl:value-of select="./@ID"/>
    </xsl:attribute>
    <xsl:attribute name="Name">
      <!--
      <xsl:variable name="MasterName" select="/vis:VisioDocument/
      vis:Masters/vis:Master[@ID=$currentMaster]/@NameU">
      </xsl:variable>
      -->
    <xsl:value-of select="/vis:VisioDocument/vis:Masters/vis:Master
      [@ID=$currentMaster]/@NameU"/>
    <!--shapes kopieren-->
    <xsl:copy-of select="/vis:VisioDocument/vis:Pages/vis:Page/
      vis:Shapes/vis:Shape[@ID=$currentID]/vis:Prop[not(contains(
      @Del,1))]"/>
      <!--xsl:copy-of select="/vis:VisioDocument/vis:Pages/vis:Page
      /vis:Shapes/vis:Shape[@ID=$currentID]"/>-->
      <xsl:for-each select="/vis:VisioDocument/vis:Masters/
      vis:Master[@ID=/vis:VisioDocument/vis:Pages/vis:Page/
      vis:Shapes/vis:Shape[@ID=$currentID]/@Master]/vis:Shapes/
      vis:Shape/vis:Prop">
      <xsl:variable name="masterPropID" select="./@ID">
      </xsl:variable>
      <xsl:variable name="knotenVorhanden" select="count(/
      vis:VisioDocument/vis:Pages/vis:Page/vis:Shapes/vis:Shape[@ID
      =$currentID]/vis:Prop[@ID=$masterPropID])">
      </xsl:variable>
      <xsl:choose>
       <xsl:when test="$knotenVorhanden=0">
         <xsl:copy-of select="."/>
       </xsl:when>
       <xsl:otherwise>
         <!--xsl:copy-of select="/vis:VisioDocument/vis:Pages/
      vis:Page/vis:Shapes/vis:Shape[@ID=$currentID]/vis:Prop[@ID=$
      masterPropID]"/-->
       </xsl:otherwise>
    </xsl:choose>
      </xsl:for-each>
   </Objekt>
  </xsl:for-each>
</Objekte>

<Verbindungen>
        <xsl:for-each select="/vis:VisioDocument/vis:Pages/vis:Page
      /vis:Connects/vis:Connect[@FromCell='BeginX']">

   <Verbindung>
```

```
        <xsl:variable name="currentID" select="./@FromSheet">
        </xsl:variable>
        <xsl:attribute name="ID">
          <xsl:value-of select="./@FromSheet"/>
        </xsl:attribute>
        <xsl:attribute name="From">
          <xsl:value-of select="./@ToSheet"/>
        </xsl:attribute>
        <xsl:attribute name="To">
          <xsl:value-of select="/vis:VisioDocument/vis:Pages/vis:Page/
          vis:Connects/vis:Connect[@FromSheet=$currentID and @FromCell
          ='EndX']/@ToSheet"/>
        </xsl:attribute>
      </Verbindung>
      </xsl:for-each>
    </Verbindungen>

  </Objektorientiert>

 </Funktionsmodell>
</xsl:template>
</xsl:stylesheet>
```

Mit dem XPath Ausdruck /VisioDocument/Pages/Page/Shapes/Shape[Bedingung] wird auf alle Shape-Knoten <Shape>, die die gegebene Bedingung erfüllen, zugegriffen. Mit dem Befehl for-each kann über die Menge aller selektierten Knoten iteriert werden. Im obigen Beispiel werden alle Shape-Knoten, die Instanzen vom Master Funktionsobjekt oder Funktionsverb sind, selektiert und darüber hinaus auf diesen iteriert, um die Instanzen ins Ziel XML Dokument zu übernehmen. Durch das Element variable kann eine Variable definiert werden, die einen bestimmten Wert persistiert, d.h. dass dieser Wert somit auch nach der Durchführung eines neuen Befehls nicht beeinflusst wird. Das when-Element erlaubt eine bedingte Ausführung einer Regel, je nachdem ob die in dem Attribut test beschriebene Bedingung erfüllt ist oder nicht. Durch copy-of können die Knoten und Teilbäume von dem Quell-XML-Dokument ins Ziel-XML-Dokument kopiert werden. Zum Beispiel werden die Eigenschaften der Visio-Schablone kopiert und direkt in das Ziel XML Dokument übernommen.

Die gesamte Transformation wird entsprechend dem Beispiel mit Hilfe von XSLT implementiert und durchgeführt. Nach der XSL-Transformation sind die Funktionshierarchie (vgl. Bild 5.23) und die Funktionsstruktur (vgl. Bild 5.24) so strukturiert, wie vorab zu Beginn des Kapitels 5.2.2 auf Seite 203 beschrieben.

```
- <Funktionsmodell>
  - <Objektorientiert>
    + <Objekte></Objekte>
    + <Verbindungen></Verbindungen>
    </Objektorientiert>
  - <Funktionsorientiert>
    - <Funktionshierarchie>
        <Funktion ShapeID="1" Name="Kopf schützen" Typ="Gesamtfunktion" ID="1"> </Funktion>
        <Funktion ShapeID="6" Name="Polster verfahren" Typ="Teilfunktion" ID="1.1"> </Funktion>
        <Funktion ShapeID="11" Name="Gestell fixieren" Typ="Teilfunktion" ID="1.2"> </Funktion>
        <Funktion ShapeID="16" Name="Mechanik fixieren" Typ="Teilfunktion" ID="1.3"> </Funktion>
        <Funktion ShapeID="24" Name="Feder auslösen" Typ="Teilfunktion" ID="1.1.1"> </Funktion>
        <Funktion ShapeID="29" Name="Dämpfer einfahren" Typ="Teilfunktion" ID="1.1.2"> </Funktion>
        <Funktion ShapeID="36" Name="Feder freigeben" Typ="Teilfunktion" ID="1.1.1.1"> </Funktion>
        <Funktion ShapeID="41" Name="Federenergie speichern" Typ="Elementarfunktion" ID="1.1.1.2"> </Funktion>
        <Funktion ShapeID="46" Name="Bewegung dämpfen" Typ="Elementarfunktion" ID="1.1.1.3"> </Funktion>
      </Funktionshierarchie>
    </Funktionsorientiert>
  </Funktionsmodell>
```

Bild 5.23: Funktionshierarchie nach der XSL Transformation

```
- <Funktionsmodell>
  - <Objektorientiert>
    - <Objekte>
      + <Objekt ID="1" Name="Funktionsobjekt"></Objekt>
      + <Objekt ID="12" Name="Funktionsverb"></Objekt>
        <Objekt ID="21" Name="Fluss_Materien"/>
      + <Objekt ID="23" Name="Funktionsverb"></Objekt>
      + <Objekt ID="32" Name="Funktionsobjekt"></Objekt>
        <Objekt ID="22" Name="Fluss_Materien"/>
        <Objekt ID="43" Name="Fluss_Materien"/>
        <Objekt ID="44" Name="Fluss_Materien"/>
      + <Objekt ID="45" Name="Funktionsobjekt"></Objekt>
        <Objekt ID="56" Name="Fluss_Materien"/>
      + <Objekt ID="57" Name="Funktionsverb"></Objekt>
        <Objekt ID="66" Name="Fluss_Materien"/>
      + <Objekt ID="67" Name="Funktionsobjekt"></Objekt>
        <Objekt ID="78" Name="Fluss_Materien"/>
      + <Objekt ID="79" Name="Funktionsverb"></Objekt>
        <Objekt ID="88" Name="Fluss_Materien"/>
        <Objekt ID="89" Name="Fluss_Materien"/>
      </Objekte>
    + <Verbindungen></Verbindungen>
    </Objektorientiert>
  </Funktionsmodell>
```

(a) Objekte

```
- <Funktionsmodell>
  - <Objektorientiert>
    + <Objekte></Objekte>
    - <Verbindungen>
        <Verbindung ID="22" From="1" To="23"/>
        <Verbindung ID="43" From="23" To="32"/>
        <Verbindung ID="44" From="32" To="12"/>
        <Verbindung ID="56" From="12" To="45"/>
        <Verbindung ID="66" From="45" To="57"/>
        <Verbindung ID="78" From="57" To="67"/>
        <Verbindung ID="88" From="67" To="79"/>
        <Verbindung ID="89" From="79" To=""/>
      </Verbindungen>
    </Objektorientiert>
  </Funktionsmodell>
```

(b) Verbindungen

Bild 5.24: Funktionsstruktur nach der XSL Transformation

Da im Rahmen dieser Arbeit die Funktionshierarchie und Funktionsstruktur separat gezeichnet und dementsprechend in unterschiedlichen Dateien gespeichert werden, wird die XSL-Transformation für jede Datei durchgeführt. Da eine einzelne vollständige XML Struktur als Schnittstelle zum Mapping zwischen dem Funktionsmodell und der Szene notwendig ist, werden die anhand der XSL-Transformationen gewonnenen XML Ausgangsdateien zusammengefasst.

Durch die XSL-Transformation ist die Semantik der Funktionsmodelle in einem einheitlichen XML Schnittstellenformat gespeichert und steht für eine Abbildung mit dem Szenengraph zur Verfügung.

Mapping zwischen Funktionsmodell und Szenengraph

Nachdem im Kapitel 4.2.3 auf Seite 138 entwickelten Konzept, werden die in dem einheitlichen Schnittstellenformat transformierten Funktionsmodelle mit Hilfe einer graphischen Bedienoberfläche in die VR Entwicklungsumgebung abgebildet. Die Aufgaben dieser Benutzerschnittstelle sind:

- Die Analyse und Manipulation der Daten aus dem XML Speicherformat und dem Virtools Szenengraph.
- Die Darstellung aller Informationen, die für das Mapping von Funktionsmodell und Szenengraph benötigt werden, um ein manuelles bzw. semi-automatisiertes Mapping vornehmen zu können.
- Die Datenablage und Bewertung der eingegebenen Abbildungsmatrix.
- Die Realisierung des Abgleichs zwischen Funktionsmodell und Szene.

Um das Funktionsmodell der crashaktiven Kopfstütze in die virtuelle Umgebung zu überführen, ist eine enge Orientierung an der Visualisierung des Funktionsmodells der Szene anzustreben. Dazu sind aus der Funktionshierarchie und der Funktionsstruktur folgende Informationen in die 3D Szene einzubringen:

- die Funktionen und deren Position in der Funktionshierarchie
- die Funktionsobjekte und deren Eigenschaften
- die Funktionsverben

- die Verbindungen zwischen Funktionsverben und Funktionsobjekte sowie die Richtung dieser Verbindungen
- die Visio-Shape-ID der Funktionsverben

Die Shape-IDs gehören nicht zur Semantik der Funktionsmodelle, übermitteln aber die eindeutige Position jedes Funktionsverbes bei der Darstellung in der Szene. Wie in Kapitel 4.2.2 auf Seite 126 erläutert, können Funktionsverben mit gleicher Verb-ID in einer Funktionsstruktur mehrmals dargestellt werden, um die Lesbarkeit des Modells zu erhöhen und die Eindeutigkeit der Flüsse zu gewährleisten. In diesem Fall kann es von Vorteil sein, diese verdoppelte Darstellung in dem Szenengraph mit zu importieren, so dass die Darstellung des Funktionsmodells in der Szene ebenfalls eindeutig und klar erfolgt. Dafür wird die von Visio vergebene eindeutige Shape-ID genutzt.

Um die Abbildungsmatrix zu etablieren und später den Import der Daten in den Szenengraph zu ermöglichen, ist ein PlugIn für die VR Entwicklungsumgebung Virtools implementiert worden. Bild 5.25 stellt die allgemeine Architektur des Plugins in der Form eines Komponentendiagramms dar.

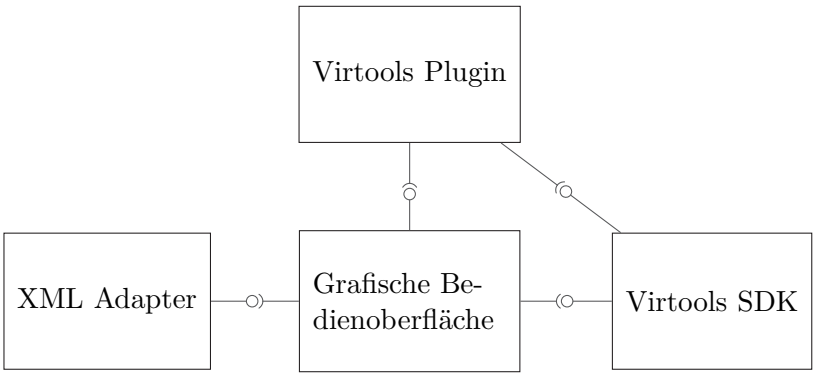


Bild 5.25: Architektur des implementierten Virtools PlugIn

Das entwickelte Plugin ist in vier Komponenten unterteilt:

1. Das eigentliche Virtools PlugIn, das an der Softwareschnittstelle von Virtools implementiert ist, so dass es von Virtools erkannt und geladen wird. Durch dieses werden die neu implementierten Funktionalitäten in der Form eines zusätzlichen Menüs eingebunden sowie die Schnittstelle mit dem Virtools SDK an die grafische Oberfläche weitergelei-

tet, so dass diese mit der VR Entwicklungsumgebung kommunizieren kann.

2. Die grafische Bedienoberfläche, welche die Schnittstelle zu dem Benutzer repräsentiert. Diese stellt die für das Mapping von Funktionsmodell und Szenengraph erforderlichen Funktionalitäten zur Verfügung und implementiert eine Abbildungsmatrix zwischen Funktionsmodell und Szenengraph.

3. Der XML Adapter, stellt Funktionalitäten für Abfragen und Bearbeitungen des im XML Format gespeicherten Funktionsmodells zur Verfügung.

4. Das Virtools SDK, stellt Funktionalitäten für den Zugriff auf Elemente der Szene und deren Bearbeitung zur Verfügung.

Die Kommunikation zwischen der grafischen Oberfläche und dem XML Adapter bzw. zwischen der Oberfläche und dem Virtools SDK erfolgt über definierte Schnittstellen. Somit ist die entwickelte Oberfläche für die Benutzung mit anderen Werkzeugen wiederverwendbar. Zum Beispiel könnte anstatt des XML Adapters ein Adapter für Datenbanken mit der gleichen Schnittstelle implementiert und verwendet werden, ohne Einfluss auf die Implementierung der grafischen Oberfläche zu haben. Somit ist die Wiederverwendbarkeit der Komponente für die Implementierung von anderen Lösungen mit anderen Werkzeugen gewährleistet. Unterstützt die verwendete VR Entwicklungsumgebung die Einbindung eines selbstdefinierten Dialogs im Rahmen seines Frameworks nicht, so kann die Oberfläche ohne Anpassung in ein eigenständiges, externes Werkzeug übernommen werden.

Die Gestaltung der Benutzeroberfläche wurde als Windows Forms realisiert. Windows Forms ist ein Formularpaket, mit dem Entwickler windowsbasierte Anwendungen erstellen können und ist Bestandteil des Microsoft .NET Frameworks. Dieses Paket stellt Standardkomponenten für die Realisierung von grafischen Oberflächen zur Verfügung und ermöglicht somit eine schnelle Entwicklung von Benutzerschnittstellen. Da das Virtools SDK lediglich für eine Anbindung in der Programmiersprache C++ dokumentiert ist, wurde hier das Windows Forms Paket in der Sprache C++ verwendet.

Die implementierte Benutzeroberfläche besteht aus den vier Registerreitern Mapping, Funktionsobjekt, Funktionsverb und Einstellungen vom Typ System.Windows.Forms.TabControl, die im folgenden erläutert werden.

Entsprechend dem im Konzept Kapitel 4.2.4 auf Seite 144 beschriebenen Sequenzdiagramm (vgl. Bild 4.35 auf Seite 147), das den Ablauf des Mappings von Funktionsmodell auf Szenengraph zeigt, wird zunächst die Abbildung von Funktionsobjekte aus dem Modell auf Objekte des Szenengraphs durchgeführt. Hierfür steht die Reiterkarte Mapping (vom Typ System.Windows.Forms.TabPage) zur Verfügung. Die Funktionsobjekte aus dem Funktionsmodell werden in Textfelder (vom Typ System.Windows.Forms.TextBox) auf der Reiterseite angezeigt. Die 3D Objekte aus der Szene werden in einer Combobox (vom Typ System.Windows.Forms.ComboBox) aufgelistet. Da die Anzahl der Funktionsobjekte in Funktionsmodellen variieren kann, soll dieser Reiter dynamisch gestaltet werden. Das Bild 5.26 veranschaulicht die Gestaltung des Reiters Mapping. Die linke Seite ist der initiale Zustand des Reiters (leer), solange kein Funktionsobjekt vorhanden ist. Nach dem Einlesen der Funktionsmodelle wird die Seite angepasst und mit Funktionsobjekte aus dem Funktionsmodell befüllt (Bild 5.26 rechts).

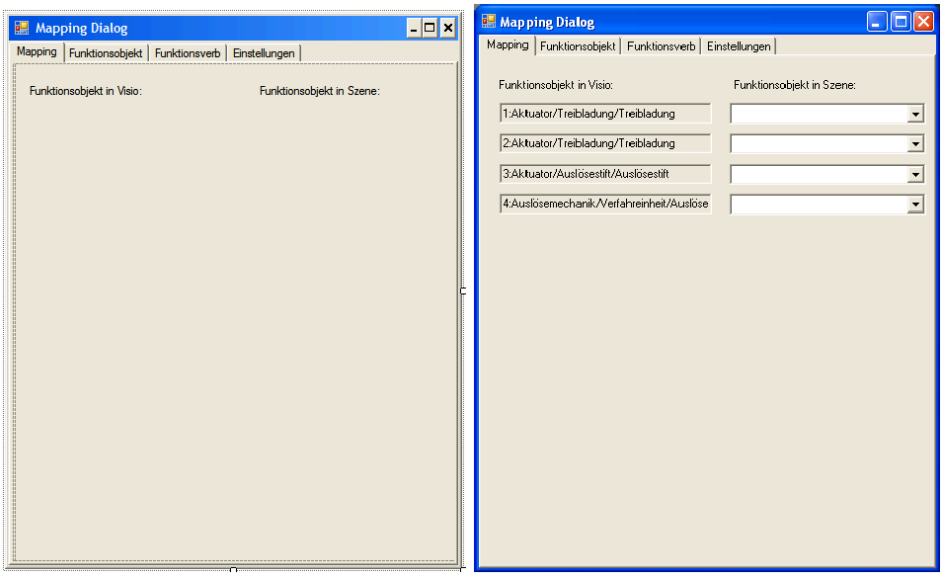


Bild 5.26: GUI: Abbildung der Funktionsobjekte

Sind die Funktionsobjekte auf Objekte in der Szene abgebildet, können die Eigenschaften der abgebildeten Objekte ebenfalls verknüpft werden. Der Reiter Funktionsobjekt (vgl. Bild 5.27) ist für das Mapping der Eigenschaften der Funktionsobjekte zuständig. Mit Hilfe einer Combobox wird ein Funktionsobjekt aus der Liste ausgewählt. Grüne Pfeile neben der ComboBox ermöglichen eine Navigierung zwischen den vorhandenen Funktionsob-

jekten. Im unteren Bereich befinden sich zwei weitere Bereiche (vom Typ System.Windows.Forms.GroupBox), die eine dynamische Auflistung der Eigenschaften aus dem Funktionsmodell bzw. aus der Szene enthalten. Ist ein Funktionsobjekt in der ComboBox gewählt worden, werden der oberen GroupBox die Eigenschaften des ausgewählten Funktionsobjektes aus Visio übergeben (vgl. Bild 5.27 rechts). Durch einen Klick auf der Schaltfläche Speichern wird die Abbildung der Eigenschaften des aktuell angezeigten Objekts in der 3D Szene gespeichert. Falls eine Abbildung der Eigenschaften definiert ist, wird der Attributwert von dem Objekt in der Szene mit dem im Funktionsmodell definierten Wert überschrieben. Ist dies nicht der Fall werden die Eigenschaften der Funktionsobjekte aus dem Funktionsmodell als neue Attribute des Objekts in der Szene eingefügt.

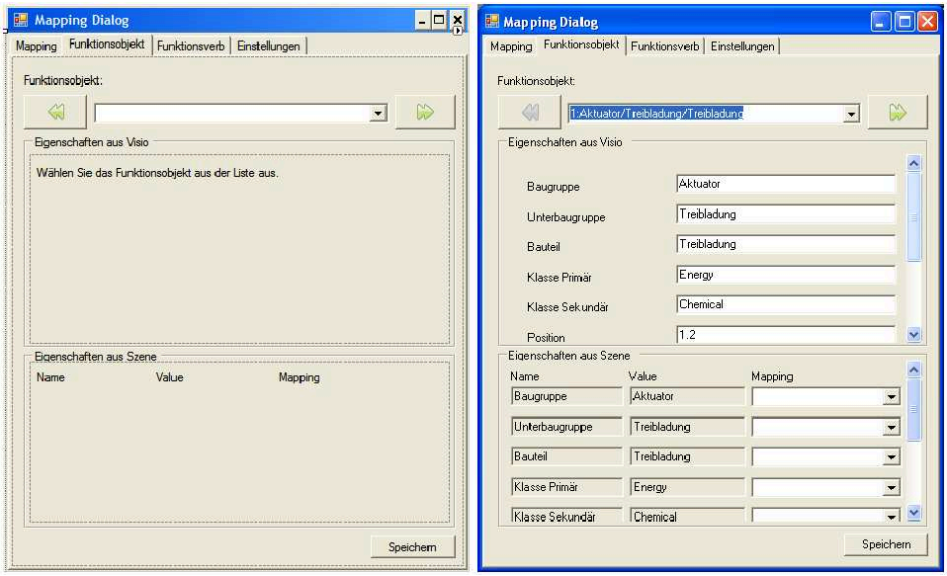


Bild 5.27: GUI: Eigenschaften der Funktionsobjekte

Der Reiter Funktionsverb, den Bild 5.28 zeigt, unterstützt den Konstrukteur darin, mittels Funktionsverben eine Übersicht vom Funktionsmodell zu entwicklen. In diesem Reiter werden keine Abbildungsentscheidungen getroffen, jedoch können hier die Funktionshierarchie und die Funktionsstruktur in der Szene gespeichert werden. Außerdem wird ein Überblick über die unterschiedlichen, beteiligten Funktionsverben gegeben. Durch die Auswahl eines Funktionsverbs in der ComboBox werden die notwendigen Informationen ermittelt. In der RichTextBox Beschreibung (von Typ System.Windows.-Forms.RichTextBox) wird die Eigenschaft Beschreibung aus dem Funkti-

onsmodell geladen. Im unteren Bereich vom Typ System.Windows.Forms.-ListBox werden die Funktionsobjekte, die mit dem ausgewählten Funktionsverb verbunden sind, aufgelistet. Durch einen Klick auf ein Element im unteren Feld wird der Reiter Funktionsobjekt aufgerufen. Dabei werden die Eigenschaften des ausgewählten Funktionsobjektes automatisch geladen. Diese Funktionalität ermöglicht eine erleichterte Abbildung, beispielsweise bei Anpassungen an eine bestehende, bereits gemappte Funktionsstruktur, indem die abzubildenden Funktionsverben ausgewählt und dadurch die zugehörigen Funktionsobjekte ermittelt werden.

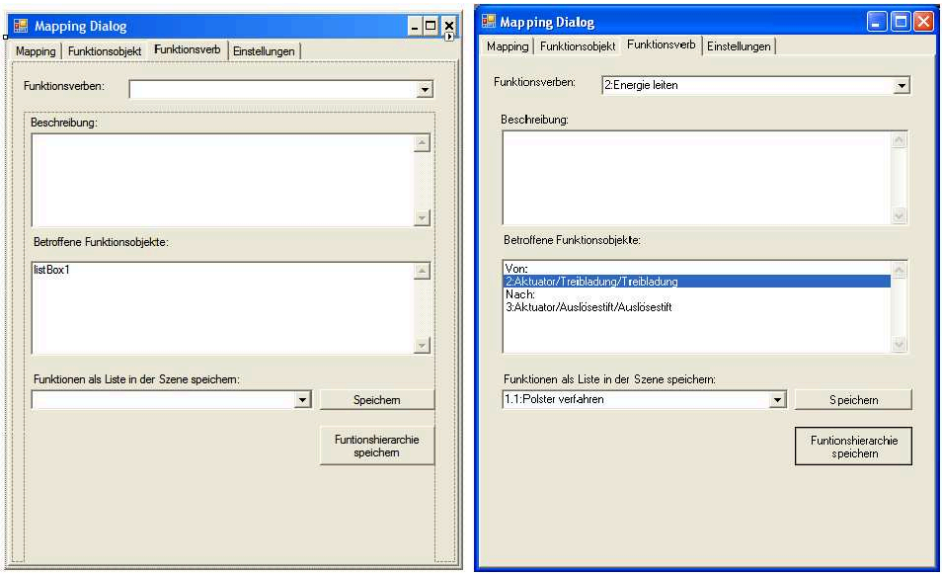


Bild 5.28: GUI: Funktionsverben

Da vor dem Mapping des Funktionsmodells auf den Szenengraph die VDX Quelldateien der Funktionshierarchie und der Funktionsstruktur dem System zur Verfügung gestellt werden müssen, wurde der Reiter Einstellung implementiert (vgl. Bild 5.29. In diesem Reiter sollen der Verzeichnispfad der Quelldatei der Funktionshierarchie bzw. der Funktionstruktur in VDX angegeben werden. Dabei wird angenommen, dass die Funktionshierarchie und die Funktionsstruktur, entsprechend der beschriebenen Vorgehensweise (vgl. Kapitel 4.2 auf Seite 116), in verschiedenen Dateien gespeichert werden. Nach einem Klick auf OK werden die Daten aus der Visiozeichnung und aus der Szene in den Mapping Dialog überführt. Nach dem Laden wird der Reiter Mapping automatisch eingeblendet und mit den eingelesenen Da-

ten aus dem Funktionsmodell bzw. aus der Szene befüllt.

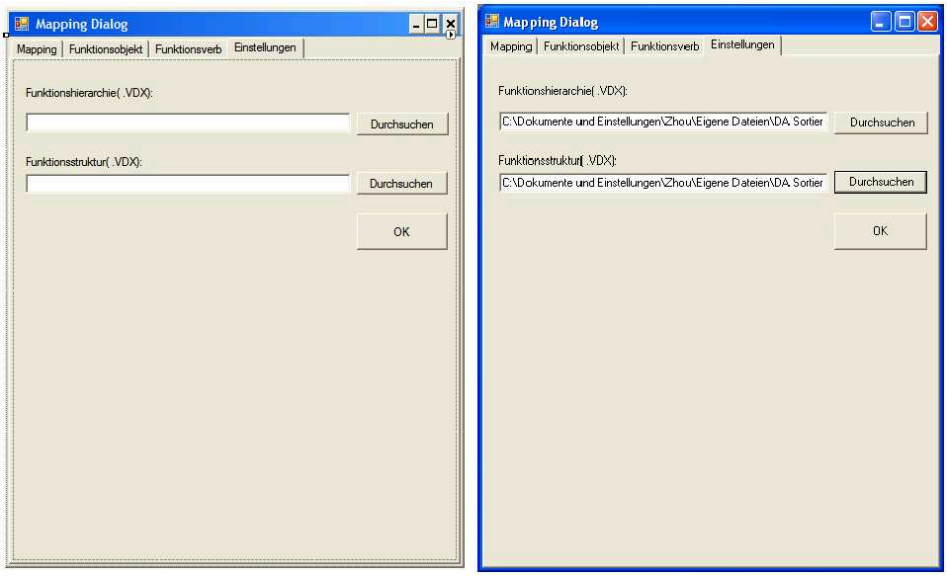


Bild 5.29: GUI: Einstellung

Nachdem die Komponente grafische Bedienoberfläche entwickelt wurde, sollen die weiteren Komponenten besprochen werden. Der Zugriff auf das XML Speicherformat erfolgt über die von der Komponente XML Adapter angebotene Schnittstelle (vgl. Bild 5.25). Diese Komponente kann in künftigen Arbeiten durch eine andere Komponente, die die gleiche Funktionalitäten anbietet, ersetzt werden, ohne dass eine Änderung an der grafischen Oberfläche vorgenommen werden muss. Dies erleichtert z.B. die Entwicklung eines Datenbankadapters, um eine Datenbank als einheitliche Schnittstelle zwischen Modellierungswerkzeug und VR Entwicklungsumgebung zu unterstützen (vgl. Kapitel 4.2.3).

Der XML Adapter wurde im Rahmen der Arbeit in der Programmiersprache C# entwickelt, da diese über eine Bibliothek von XML Funktionalitäten des .NET Frameworks System.XML verfügt. Somit werden die XML Elemente (Knoten, Teilbäume, Attribute) als Objekte (vom Typ XmlNode) dargestellt, die Funktionen zur Abfrage und Bearbeitung der Inhalte anbieten. Die Suche von Daten in einem XML Dokument erfolgt beispielsweise über die Funktion

`SelectSingleNode(string XPath)` bzw.
`SelectNodes(string XPath)`

wobei der Parameter XPath eine Abfrage in der Sprache XPath bezeichnet.

Die Komponente XML Adapter implementiert folgende Funktionen :

- *Load(string filename)*, um die gewünschte Datei zu laden.
- *Filter(string filename, string filtername)*, führt die in der Datei filtername enthaltene XSL-Transformation auf die Datei filename durch. Diese Funktion wird aufgerufen, um die in Kapitel 5.2.2 eingeführte Transformation automatisch auszuführen, bevor eine Bearbeitung stattfinden kann.
- *getFH()*, liefert die gesamte Funktionshierarchie in einer selbstdefinierten und von dem Speicherformat unabhängigen Struktur zurück.
- *getObjekte(string Typ)*, liefert die Liste aller Namen der in dem Funktionsmodell enthaltenenen Objekte vom Typ (Funktionsverb, Funktionsobjekt, usw.) zurück.
- *getProp(string objektname)*, liefert die Liste aller Eigenschaften des Elements mit dem Namen *objektname* und deren Werte zurück.
- *getConnectionInfo(string FVID)*, liefert die Liste aller mit dem Funktionsverb *FVID* verbundenen Funktionsobjekte, sowie die Richtung der Verbindung.

Neben den Funktionen Load und Filter ist die Signatur aller angebotenen Funktionen von der Implementierung der Datenspeicherung unabhängig und somit für die Implementierung eines Adapters für Datenbankzugriffe ebenso geeignet.

Um mit der Virtools Entwicklungsumgebung zu kommunizieren, sind in der konzipierten Architektur zwei Schnittstellen entwickelt worden (vgl. Bild 5.25). Zum Einen das Virtools Plugin selbst, das mittels Menüsteuerung in die Oberfläche von Virtools integriert wird, zum Anderen die entwickelte grafische Benutzeroberfläche, über die ebenfalls mit der geladenen Szene interagiert werden kann.

Das Virtools Plugin ist unter Visual Studio .NET mit Hilfe des hierfür vorgesehenen AppWizard entwickelt worden. Dieser erstellt ein neues Projekt mit allen notwendigen Voreinstellungen und Dateien, so dass nach dem

Kompilieren des Projekts das erzeugte Assembly als Plugin in Virtools geladen werden kann. Konkret werden somit zwei Quelldateien erzeugt : VirtoolsUIPlugin2Editor und VirtoolsUIPlugin2Callback. Die Datei VirtoolsUIPlugin2Editor implementiert die Funktion *PluginInfo* GetVirtoolsPluginInfo(int index)* die die Informationen, die Virtools benötigt, um das Plugin zu laden und mit ihm zu kommunizieren in der Struktur *PluginInfo* liefert. Die Struktur *PluginInfo* definiert die Eigenschaften des implementierten Plugins, wie Name, Typ, Version, und Callback-Funktion.

Die Callback-Funktion definiert einen Zeiger auf eine Funktion des Plugins, die von Virtools aufgerufen werden soll, um eine Benachrichtigung an das Plugin weiterzureichen. Diese Funktion wird mittels der Datei VirtoolsUIPlugin2Callback realisiert und implementiert folgende Signatur :

```
void PluginCallback(PluginInfo::CALLBACK_REASON reason,
                    PluginInterface* plugininterface)
```

Bei einem Aufruf der Callback-Funktion übergibt Virtools den Grund des Rückrufs (reason), sowie eine Schnittstelle zu sich selbst, so dass das Plugin auf diesen Rückruf reagieren kann und ebenfalls mit Virtools kommuniziert. Über diesen Callback-Mechanismus ist eine bidirektionale Kommunikation zwischen Virtools und dem geladenen Plugin realisiert.

Nachdem das Plugin erfolgreich von der Entwicklungsumgebung geladen wird, erfolgt über einen Aufruf der Callback-Funktion mit dem Grund PluginInfo::CR_LOAD eineBenachrichtigung. Dadurch bekommt das Plugin die Schnittstelle zu Virtools und kann somit seine Funktionalitäten registrieren, z.B. durch die Erstellung eines Menüs.

Die bei einem Rückruf gelieferte Schnittstelle plugininterface bietet die folgenden Funktionen zur Verwaltung eines selbstdefinierten Menüs in Virtools an:

- *AddPluginMenu* um einen selbstdefinierten Menüeintrag in der Menüleiste der Umgebung hinzuzufügen.
- *AddPluginMenuItem* um eine Funktion in dem Menü hinzuzufügen.
- *UpdatePluginMenu* um das Menü zu aktualisieren, so dass die letzten Änderungen übernommen werden.
- *ClearPluginMenu* um ein selbstdefiniertes Menü zurückzusetzen.

Die zweite Schnittstelle wird durch die grafische Benutzeroberfläche realisiert, die die Informationen über die geladene Szene darstellen kann. Bei dem Aufruf des Dialogs wird das plugininterface an die grafische Benutzeroberfläche weitergereicht. Über die von der Schnittstelle angebotene Funktion GetCKContext, wird auf dem aktuellen Kontext der Umgebung (geladene Szene, Daten etc.) zugegriffen. Dabei verwendet die Funktion CKContext, um auf die aktuell geladene Szene zuzugreifen, den Funktionsaufruf GetCurrentScene(). Danach liefert die Funktion GetObjectIterator() einen Iterator, mit dem die Identifikatoren aller Objekte in der Szene ermittelt werden. Anhand dieser Identifikatoren werden über die Funktion GetObject des CKContext die Objekte und deren Eigenschaften ermittelt. Bild 5.30 fasst das Vorgehen bei der Auflistung der Objekte aus der Szene in Form eines Sequenzdiagramms zusammen.

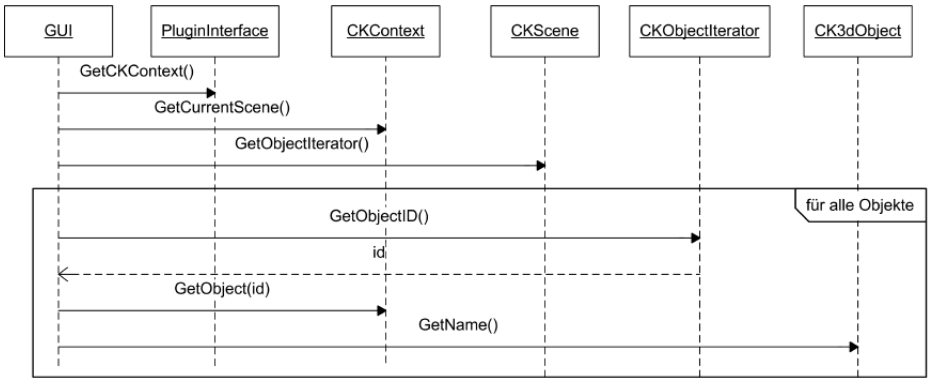


Bild 5.30: Sequenzdiagramm des Zugriffs auf die Objekte in der Szene

Wird ein Objekt in der grafischen Benutzeroberfläche ausgewählt, so sollen dessen Eigenschaften aus der Szene ausgelesen und angezeigt werden. Hierfür wird zunächst das ausgewählte Objekt in der Szene anhand der Funktion GetObjectByName des CKContext aufgerufen. Diese bietet die Funktion GetAttributeList an, welche eine Liste eines 2-gliedrigen Tupels {CKAttributeType, CK_ID} zurückgibt, wobei CKAttributeType den Typ des Attributs bezeichnet und CK_ID eine Referenz auf die eigentliche Instanz des Attributs darstellt. Um auf diese Instanz zugreifen zu können,wird die Instanz anhand des CKAttributeManager aufgerufen. Der CKAttributeManager ist über die Funktion GetAttributeManager des CKContext abrufbar. Bild 5.31 fasst das Vorgehen bei der Auflistung der Attribute eines Objekts

aus der Szene in Form eines Sequenzdiagramms zusammen.

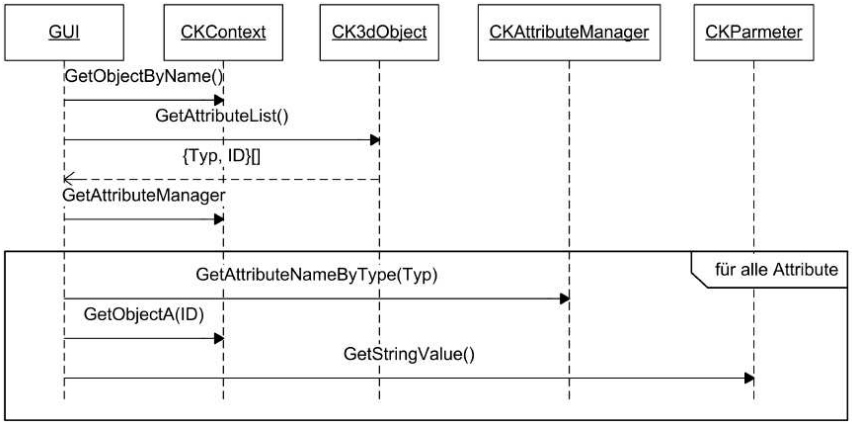


Bild 5.31: Sequenzdiagramm des Zugriffs auf die Attribute eines Objekts in der Szene

Import des Funktionsmodells in die Szene

Nachdem die Abbildungsentscheidungen getroffen und gespeichert wurden, muss im finalen Schritt der Abgleich zwischen Funktionsmodell und Szenengraph vorgenommen werden. Diese Funktionalitäten wurden ebenfalls in der eingeführten Benutzeroberfläche eingebettet (vgl. Kapitel 5.2.2 auf Seite 214).

Zur Erstellung und Speicherung der Datenstrukturen erlaubt Virtools, zusätzlich zur Definition selbstdefinierter Attribute grafischer Objekte, eine Definition eigener Datenbestände in tabellarischer Form (vgl. Kapitel 4.2.5 auf Seite 147). Dabei sollen die folgenden Informationen aus dem Funktionsmodell für eine Darstellung in der Szene importiert werden:

- Eigenschaften der Funktionsobjekte aus der Funktionsstruktur
  Die im Funktionsmodell definierten Eigenschaften wie Baugruppe, Materialien, Kosten etc. werden in der Virtoolsszene als Attribute des abgebildeten 3D Objekts eingeführt.

- Funktionsverben und Flüsse aus der Funktionsstruktur
  Da die Funktionsverben nicht in der Szene vorhanden sind, werden die Informationen über diese, sowie deren Relationen zu den abgebildeten

Objekten der Szene in einer Tabelle in der Szene gespeichert. Jeder Funktionsstruktur entspricht eine Funktion in der Funktionshierarchie. Diese wird separat in einer neuen Tabelle gespeichert, so dass eine Verlinkung zwischen Funktionshierarchie und Funktionsstruktur vorgenommen werden kann.

- Sichten aus der Funktionshierarchie
  Da eine Funktionsstruktur eine Sicht einer Funktion in der Funktionshierarchie darstellt, werden die Informationen über die Funktionshierarchie auch in der Szene in Form einer Tabelle eingeführt. Dadurch wird eine Übersicht auf die Funktionen eines Produkts geschaffen und die Darstellung der Funktionsmodelle kann hierarchisch aufgebaut werden.

Auf diese Daten wird von den Buildingblocks in Virtools zugegriffen, und diese als Eingangsdaten zu den Verhaltensblocks eingeführt. Da die Implementierung der Verhaltensblöcke auf vordefinierte Eingangsvariablen basiert, können die Tabellen nicht vom Benutzer beliebig benannt werden. Dementsprechend wird die Bezeichnung der Tabllen festgelegt und ist damit den Verhaltensblöcken bekannt. Die Funktionshierarchie wird in eine Tabelle mit festen Namen "Funktionshierarchie" gespeichert und besteht aus den drei Spalten Funktion, Typ und ID (vgl. Tabelle 5.5). Jede Zeile entspricht der Sicht einer Funktion in der Funktionshierarchie, und jede Sicht kann durch eine Funktionsstruktur beschrieben werden, welche ebenfalls in einer Tabelle gespeichert wird (vgl. Tabelle 5.6. Die Tabelle der Funktionsstruktur wird mit dem Namen der Sicht einer Funktion (Eintrag in der Tabelle der Funktionshierarchie) benannt. Darüber hinaus muss bei der Implementierung eines Verhaltensblocks die Tabelle "Funktionshierarchie" bekannt sein, die Tabellen über Funktionsstrukturen werden automatisch durch das Einlesen der Einträge in der Funktionshierarchie erkannt.

Die Tabelle 5.6 beschreibt die Datenstruktur der aus einer Funktionsstruktur importierten Daten. Die erste Spalte Funktionsobjekt besteht aus Verweisen auf die Objekte aus Virtools (Typ:CK3dObject*). Die Spalte FO(Visio) enthält den Objektnamen aus der Funktionsstruktur, die in Visio erstellt wurde. In den nächsten drei Spalten Funktionsverb, Verb ID und ShapeID sind die Eigenschaften des Funktionsverbs abgebildet. Die Spalte Richtung beschreibt den Fluss zwischen Funktionsobjekt und Funktionsverb und kann den Wert “In” oder “Out” annehmen. “In” bedeutet, dass das Funktionsobjekt vor dem Funktionsverb steht, “Out”, dass das Funk-

tionsobjekt nach dem Funktionsverb steht. FHID entspricht dem Attribut Position in der Funktionshierarchie bei dem Funktionsobjekt (vgl. Bild 4.25 auf Seite 129). Jede Zeile entspricht einem Fluss in der Funktionsstruktur.

Funktionshierarchie		
Funktion	Typ	ID
Funktion 1		
...		
Funktion n		

Tabelle 5.5: Funktionshierarchie in Virtools

Funktionsstruktur *Funktion 1*						
Funktions-objekt	FO (Visio)	Funktions-verb	Verb-ID	Shape-ID	FH-ID	Rich-tung

Tabelle 5.6: Funktionsstruktur in Virtools der Sicht *Funktion 1*

Für die Implementierung werden nach der Mapping der Funktionsobjekte auf Objekte des Szenengraphs, die während der Funktionsmodellierung eingegebenen Eigenschaften der Objekte an das 3D Objekt angehängt (vgl. Kapitel 4.2.5 auf Seite 147). Hierfür müssen zunächst die Attribute einem in Virtools bekannten Typ zugeordnet werden. Ist dies nicht möglich, muss einen neuen Typ erstellt werden. Diese Attributtypen besitzen unter Virtools entsprechende Datentypen (string, integer etc.) und werden in Kategorien eingeordnet. In diesem Fall wird eine Kategorie erstellt, die alle aus dem Funktionsmodell importierten Attributtypen enthält. Daraufhin wird ein neuer Typ für jede Eigenschaft der Funktionsmodellelemente (Bauteil, Unterbauteil, Klasse Primär etc.) in dieser Kategorie erstellt.

In der Praxis wird die neue Kategorie mit Hilfe des CKAttributeManager und durch die Funktion AddCategory angelegt. Danach können die neuen Attributtypen erzeugt und registriert werden. Dies erfolgt über die Funktion RegisterNewAttributeType des CKAttributeManager. Diese Funktion nimmt als Parameter den Namen des neuen Typs, den zugehörigen Datentyp sowie den Typ des Objektes, an den dieses Attribut angehängt werden

soll. Als Datentyp wird im Rahmen des Verifikationsbeispiel immer CKPGUIG_STRING verwendet. Dies entspricht einer Zeichenkette. Der Attributtyp wird mit der Eingabe des Parameters CKCID_3DOBJECT für die 3D Objekte in der Szene freigegeben. Somit wird ein neuer Attributtyp erstellt, der anschließend in die bereits erzeugte Kategorie über die Funktion SetAttributeCategory eingeordnet wird. Ist der Attributtyp erzeugt und eingeordnet, kann dieser in der Liste der Eigenschaften der Objekte in die Szene hinzugefügt werden. Dies erfolgt über die Funktion SetAttribute des entsprechenden CK3dObject. Bild 5.32 fasst das Vorgehen des Imports von Objekteigenschaften aus dem Funktionsmodell in die Szene zusammen und Bild 5.33 zeigt die Liste der Attribute eines Szenenobjektes nach dem Import in Virtools.

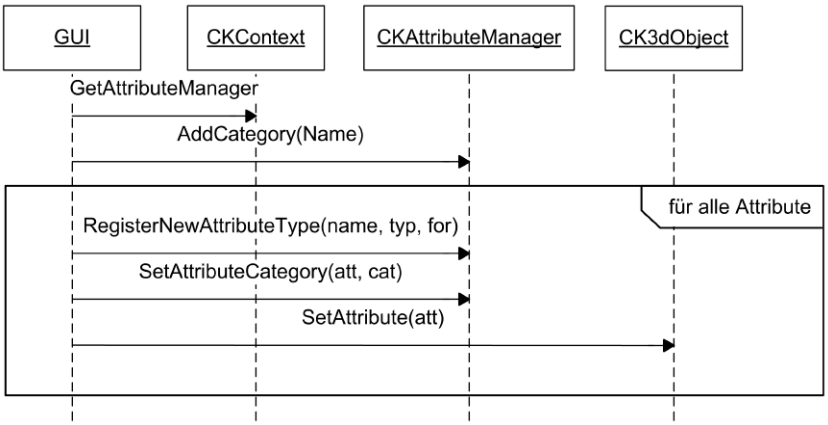


Bild 5.32: Sequenzdiagramm des Imports von selbstdefinierten Attributen in die Szene

Name	Category	Value
Attribut	Visio Import	Chem. Formel
Baugruppe	Visio Import	Aktuator
Bauteil	Visio Import	Treibladung
Klasse Primär	Visio Import	Energy
Klasse Sekundär	Visio Import	Chemical
Klasse Tertiär	Visio Import	
OID	Visio Import	1
Position Funktionshierarchie	Visio Import	1.2
Unterbaugruppe	Visio Import	Treibladung
Version	Visio Import	1

Bild 5.33: Importierte Objektattribute

Die Funktionshierarchie und Funktionsstruktur sollen in Form von Tabellen mit der vorab beschriebenen Datenstruktur in der Szene gespeichert werden. Dazu stellt das CKContext die Funktion CreateObject zur Verfügung.

Wird diese bei dem Aufruf CKCID_DATAARRAY als Parameter eingegeben, so wird eine neue Tabelle erzeugt. Diese neu erzeugte Tabelle soll noch in der aktuell geöffneten Szene registriert werden. Alle Elemente einer Szene werden in einer sogenannten composition zusammengefasst und durch einen Level verwaltet. Die Tabelle soll dementsprechend innerhalb des Levels eingetragen werden, so dass diese mit der Szene zusammen gespeichert wird. Bild 5.34 fasst das Vorgehen bei der Erstellung einer Tabelle in der Szene zusammen. Bild 5.35 und Bild 5.36 beschreibt die erzeugten Tabellen für die Speicherung der Funktionshierarchie bzw. der Funktionsstruktur nach dem Import.

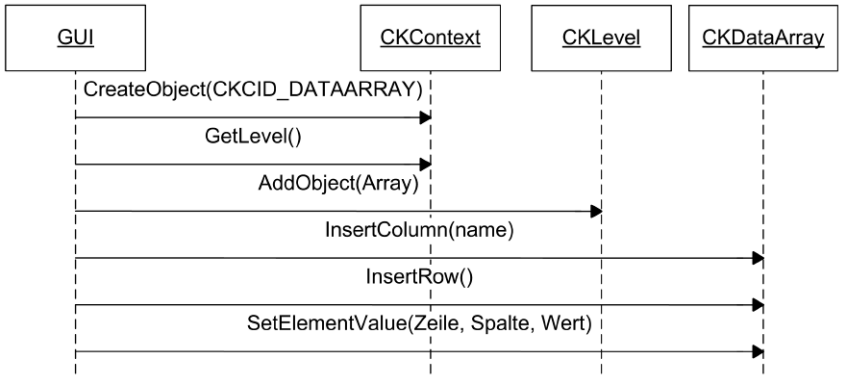


Bild 5.34: Sequenzdiagramm eines Imports von selbstdefinierten Daten in tabellarischer Form in Virtools

	0 : Funktion	1 : Type	2 : ID
0	Kopf schützen	Gesamtfunktion	1
1	Polster verfahren	Teilfunktion	1.1
2	Gestell fixieren	Teilfunktion	1.2
3	Mechanik fixieren	Teilfunktion	1.3
4	Feder auslösen	Teilfunktion	1.1.1
5	Dämpfer einfahren	Teilfunktion	1.1.2
6	Feder freigeben	Teilfunktion	1.1.1.1
7	Federenergie speichern	Elementarfunktion	1.1.1.2
8	Bewegung dämpfen	Elementarfunktion	1.1.1.3

Bild 5.35: Importierte Funktionshierarchie

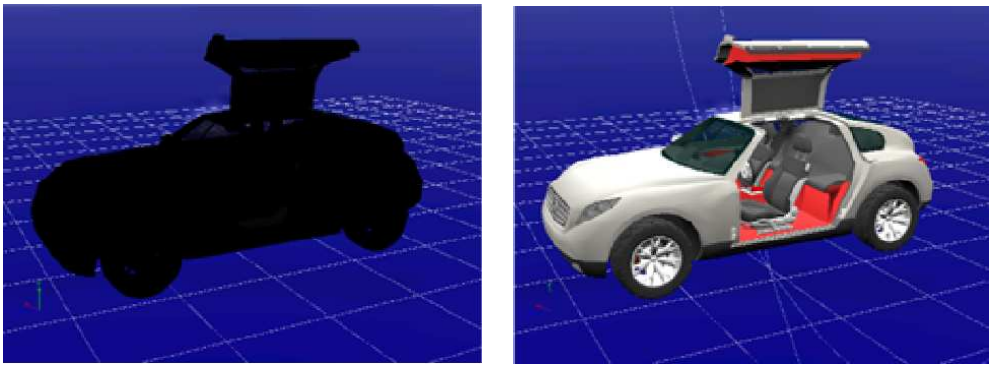
	0 : Funktionsobjekt	1 : FO(Visio)	2 : Funktionsverb	3 : Verb ID	4 : Verb ShapeID	5 : FHID	6 : Richtung
0	1.1.1.2.3.12-aktuator	o-2-Aktuator/Treibladung/Treibladung	Energie leiten	f-2	12	fh_1.2	In
1	1.1.1.2.3.13-ausloesestift	o-3-Aktuator/Auslösestift/Auslösestift	Energie leiten	f-2	12	fh_1.2	Out
2	1.1.1.2.3.12-aktuator	o-1-Aktuator/Treibladung/Treibladung	Energie wandeln	f-1	23	fh_1.2	In
3	1.1.1.2.3.12-aktuator	o-2-Aktuator/Treibladung/Treibladung	Energie wandeln	f-1	23	fh_1.2	Out
4	1.1.1.2.3.13-ausloesestift	o-3-Aktuator/Auslösestift/Auslösestift	Energie leiten	f-3	57	fh_1.2	In
5	1.1.1.2.3.10-ausloeseblech	o-4-Auslösemechanik/Verfahreinheit/Auslöseblech	Energie leiten	f-3	57	fh_1.2	Out
6	1.1.1.2.3.10-ausloeseblech	o-4-Auslösemechanik/Verfahreinheit/Auslöseblech	Federenergie steuern	f-4	79	fh_1.2	In
7	1.1.1.2.3.10-ausloeseblech	o--1-None	Federenergie steuern	f-4	79	fh_1.2	Out

Bild 5.36: Importierte Funktionsstruktur

5.2.3 Interaktive Visualisierung des Funktionsmodells in der virtuellen Umgebung

Nach der Erstellung der 3D Umgebung, des Funktionsmodells und des erfolgten Mappings der Funktionsmodelldaten in die virtuelle Umgebung, folgt die Virtualisierung der Gesamtszene (vgl. Bild 5.4 auf Seite 185). Dafür wird die proprietäre VR Entwicklungsumgebung Virtools verwendet. Die für die immersiven Interaktionen verantwortlichen Schematics werden zuerst für die Integration der Interaktionsmetaphern verwendet. Daraufhin wird der Bewegungsablauf der crashaktiven Kopfstütze entwickelt. Im nächsten Schritt werden die Funktionsverben modelliert und abschließend die interaktive Funktionsstruktur visualisiert.

Nach dem Starten von Virtools wird zunächst eine leere Komposition geladen. In dieser Komposition existiert außer einem Koordinatensystem und einer Ebene nichts. Das bedeutet unter anderem, dass eingefügte Elemente zunächst nur als schwarze Körper zu erkennen sind, da kein Licht vorhanden ist, mit dem die Shader ein gerendertes Bild erzeugen können (vgl. Bild 5.37). Es wird also, neben den Modellen die eine Szene füllen, auch eine Beleuchtung der Szene benötigt.



(a) Szene ohne Licht (b) Szene mit Licht

Bild 5.37: Importierte Szene in Virtools

Dazu bietet Virtools verschiedene Arten von Lichtquellen, wie sie standardisiert zur Verfügung stehen, an (vgl. Bild 4.11 auf Seite 106). Um die Szene gleichmäßig zu beleuchten, wurden vier Punktlichter über dem Fahrzeug im Raum positioniert, und ihre Parameter entsprechend der Komposition eingestellt. Die Position und weitere Parameter, wie Range (Reichweite) und Color (Farbe), wurden dabei entsprechend angepasst. In Virtools werden

Modelle über *Ressources* → *Import File* eingefügt. Es erscheint ein Dialogfenster, in dem zu importierende 3D Modelle in der persönlichen Ordnerstruktur zu finden sind. Falls das zu importierende 3D Modell in der Szene noch nicht vorhanden ist, wird es daraufhin eingefügt, und zwar mit der Orientierung und Position, wie diese in dem Modellierungswerkzeug hinterlegt wurde. Im Fall des Auftretens einer doppelten Namensbelegung von Objekten in der Szene erscheint ein menügesteuerter Hilfedialog. Dabei steht es dem Benutzer frei, wie mit dem Duplikat verfahren werden soll. Dies ermöglicht auch nachträglich Modellierungsarbeiten auszuführen und die Ergebnisse in die VR Szene einzufügen.

Umsetzung der Interaktionsmetaphern anhand von Schematics

Zum Erweitern der Szene mit interaktiven Elementen steht in Virtools die VR Library zur Verfügung. Diese ist das Erweiterungspaket, das Virtools ermöglicht, eine virtuelle Umgebung zu entwickeln, die beispielsweise in einem Cluster visualisiert wird. Ein bereits erstelltes Cluster wird mit den VR BuildingBlocks gesteuert und bedient. VR Devices, wie Trackingsysteme und Powerwalls werden ebenfalls durch die VR Library bereitgestellt. Dazu stehen BBs zur Verfügung, die mit diesen Devices kommunizieren, d.h. diese erwarten Ein- und Ausgangswerte. Für die Powerwall wird beispielsweise automatisch das richtige Bild für jeden Projektor erzeugt. Es ist dabei nicht nötig, dass manuell für jedes Auge des Users eine Kamera mit entsprechender Position und Orientierung erstellt werden muss. Viele der Funktionen der VR Library werden nicht nur über BBs, sondern auch über Konfigurationsdateien benutzt. Jede VR Szene benötigt zumindest ein Configfile mit der Standardbenennung VRpack.cfg. In dieser Datei wird z.B. festgelegt, welche Auflösung jeder Host des Cluster ausgeben soll.

Wichtigster Bestandteil von Virtools bilden die Schematics. Alle Schematics in einer Komposition sind in den Levelskripten enthalten. Dies erhöht die Übersicht über die Skripte und zwingt gleichzeitig dazu allen BBs von Anfang an die korrekten Targets zuzuweisen. Falls Funktionen in Teilskripts erstellt und danach in ein anderes Skript kopiert werden, kann es zu unerwünschten Effekten kommen, beispielsweise im Falle falsch zugewiesender Target Parameter in den BBs. Das Levelskript ist in mehrere Behavior-Graphs aufgeteilt, die jeder in sich geschlossen eine Funktion erfüllen und untereinander nur über Parameteraustausch in Verbindung stehen. So gibt es z.B. einen BG dessen Aufgabe es ist, ständig zu kontrollieren, ob durch

Bewegungen der Objekte in der Szene Kollisionen entstanden sind. Im Falle einer Kollision wechselt ein Boolscher Parameter seinen Wert von False auf True.

Um die Interaktion zu ermöglichen, werden zur Realisierung des Verifikationsbeispiels im Folgenden die notwendigen BGs und die zugehörigen BBs erläutert und ihre Umsetzung beschrieben.

BehaviourGraph Cluster und Distribution Dieser BG ist für das Clustermanagement und die Distribution, d.h. die Verteilung der Daten im Cluster, zuständig. Bild 5.38 illustriert den Aufbau des BGs aus den BBs VR Distrib Add-Remove, VR Stereo Settings, VR Distrib Get HostID, Key Event, VR Player Exit, Nop, VR Distribute und Text Display.

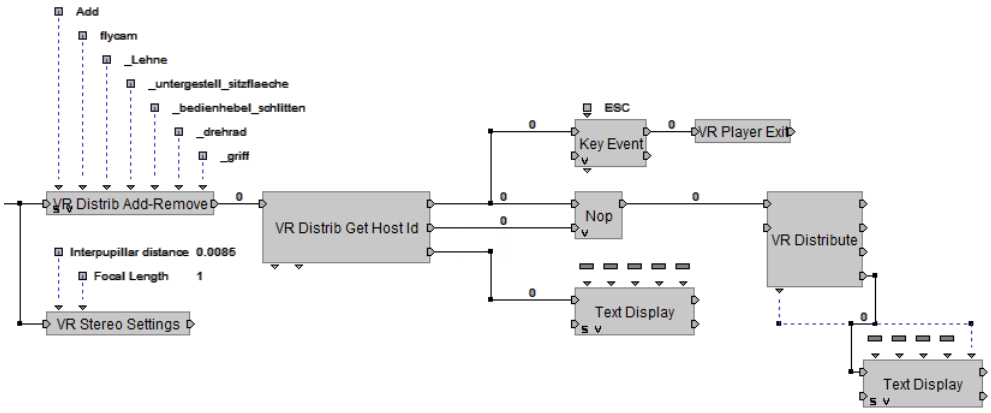


Bild 5.38: BehaviourGraph Cluster and Distribution

Der VR Distrib Add-Remove BB bestimmt von welchen Objekten welche Parameter vom Master an die Slaves übermittelt werden. Dies ist insbesondere bei kritischen Teilen, bei denen eine absolut eindeutige Position und Orientierung nötig ist, wichtig. Falls beispielsweise die Worldmatrix des Polsters der Kopfstütze nicht auf diese Art an die Slaves gesendet wird, kann es passieren, dass auf einem Slave das Polster eine andere Orientierung annimmt als auf einem Anderen. Die Auswahl der zu verteilenden Objekte muss vorab überlegt werden, da eine zu hohe Anzahl von Objekten zu hohem Transfervolumen im Netzwerk führt, was dann wiederum Performanceprobleme mit sich bringt.

Für das Verifikationsbeispiel werden folgende Objekte und Parameter auf die Slaves verteilt:

- Die flycam, da die Position dieser Kamera auf allen beteiligten Hosts gleich sein muss.
- Die Interaktionsmetpahern, Avarworm, HilfeAvatar ViMa, Ballsteuerung mit GoGo, Interaktive räumliche Sphären, Window in World, Kollisionserkennung sowie halbtransparente Objekte, da jeder Host die Positionen der Interaktionsmetaphern kennen muss, um entsprechend die Interaktionen korrekt auszuführen.
- Alle Objekte der Kopfstütze, des Fahrersitzes sowie die Autotür und der Stein in der Peripherie, da diese zentrale Elemente der Funktion des Sitzes sind und somit in jeder Situation korrekt distributiert werden müssen.

Mit Hilfe des BBs VR Stereo Settings kann der Pupillenabstand und die Brennweite des Nutzers hinterlegt werden. Virtools erzeugt daraufhin, die entsprechenden Bilder für jeden Projektor. Konfigurationen an diesen Werten sind notwendig, um das Immersionsgefühl in der virtuellen Umgebung zu verbessern. Von Virtools werden die Standardwerte $85mm$ als Pupillenabstand und $1m$ als Brennweite vorgeschlagen.

Je nach Host ID des PC, auf dem die Komposition läuft, wird entweder der Ausgang Master oder Slave aktiviert. Falls Fehler auftreten wie z.B. das Fehlen des Configfile, wird der Ausgang Error aktiviert. Dabei wird der BB VR Distrib Get Host ID dazu verwendet, den BB VR Player Exit auf dem Master zu aktivieren, nicht jedoch auf den Slaves. Die lokalen Versionen des VR Publisher werden dann vom VR Remote Daemon beendet.

Die Parametereingabe des Key Event BBs entspricht dem Tastendruck auf der Tastatur des PCs. Durch Drücken der Escape Taste kann beispielsweise der BB VR Player Exit aktiviert werden. Dieser BB beendet den VR Publisher. Falls das Cluster zu diesem Zeitpunkt korrekt gestartet ist, werden vom VR Remote Daemon auch die lokal laufenden Versionen auf den Slaves geschlossen. Da der Master aber stets schneller startet als die Slaves, kann es vorkommen, dass der Benutzer den VR Publisher beendet bevor das Cluster komplett gestartet ist. In dem Fall muss der Nutzer die Player auf den Slaves über eine Stapelverarbeitungsdatei mit dem VRRemoteController.exe componame.cmo -f Token beenden. Der Key Event BB bleibt selbständig aktiv. Er wird dementsprechend nicht in eine Schleife eingebunden.

Der Nop BB kann sehr flexibel verwendet werden, und verfügt über beliebig viele Ein- und Ausgänge. Immer wenn einer der Eingänge aktiviert wird,

werden alle Ausgänge aktiviert. Nop hat also keine wirkliche Funktion. Er bündelt lediglich den Signalfluss. Der BB wird dazu verwendet, um die Skripte zu ordnen und einen besseren Überblick über die Signalflüsse zu erhalten.

Der BB VR Distribute löst die Distribution der vorher ausgewählten Parameter und Objekte aus. Er bewirkt ein Senden der Werte beim Master, und ein Warten auf die Werte bei den Slaves. Dabei ist zu beachten, dass die zu distributenden Parameter vor dem BB mit Hilfe des VR Distrib Add-Remove BB bestimmt werden. Diese können erst nach der Verteilung der Daten verwendet werden.

Der Text Display BB ermöglicht es, Texte auf dem Bildschirm auszugeben. Wie in Bild 5.39 zu sehen ist, sind dabei die Position des Textfeldes, die Schriftfarbe, die Ausrichtung des Textes im Feld und die Schriftgröße einstellbar. Der wichtigste Parameter ist der Text, der dargestellt werden soll. Dieser kann als Zeichenkette direkt angegeben oder als Parameter aus einem anderen BB übernommen werden. Dabei werden alle Parameter beim Verlinken automatisch in Text umgewandelt und dargestellt. In dem Verifikationsbeispiel sind zwei Text Display BBs verbaut. Der Erste wird bei einem Fehler der Host ID Zuweisung aktiviert und gibt dann als Text VR Distrib Get Host Id Error: Host not in the list aus. Der Zweite wird aktiviert, wenn ein Fehler mit dem VR Distribution BB auftritt, und gibt dann keine vorgegebene Zeichenkette als Fehlermeldung aus, sondern übernimmt vom VR Distribution BB dessen Parameter Error und gibt diesen dann aus.

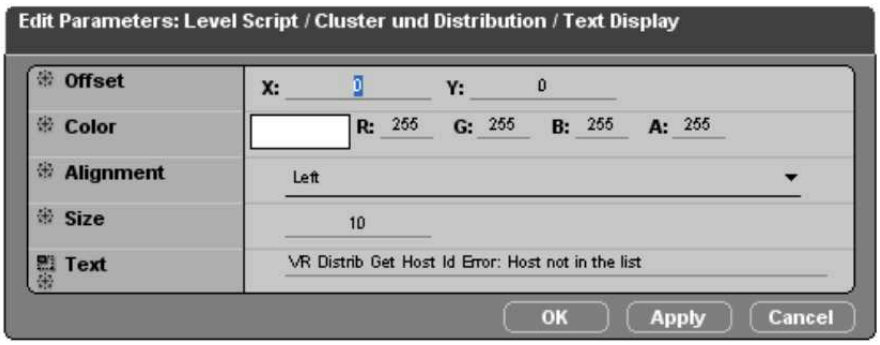


Bild 5.39: Parametereinstellungen des BB Text Display

In Virtools werden nicht die Zustände synchronisiert, sondern nur Frames und Zeit. Wird eine Variable auf einem der Rechner geändert, behält sie im

Normalfall auf den anderen Rechnern ihren alten Wert, es sei denn, es gibt im gleichen Moment auf den anderen Rechnern ebenfalls dieses Ereignis, welches zum Ändern der Variable veranlasst. Dementsprechend müssen Variablen, die synchronisiert werden sollen, entsprechend deklariert werden. Generell gilt, dass die Ursachen für Bewegungen (Eingabegeräte) nur am Master stattfinden, nicht an den angeschlossenen Slaves. Somit muss die Position aller Objekte, die vom Master aus bewegt werden, synchronisiert werden. Wird allerdings ein Objekt bewegt, weil etwa der Avarworm es anstößt, findet die Ursache auf allen Rechnern statt und somit wird auch ohne Synchronisation das gleiche Resultat auftreten.

BehaviourGraph Tracking Neben dem BG Cluster and Distribution ist der BG Tracking der zweite, zentrale Baustein der VR Library von Virtools, und essentiell zur Interaktion mit der virtuellen Umgebung. Der BehaviourGraph Tracking ist für das Verarbeiten der Tracking Informationen zuständig. Er beinhaltet die BBs Vrpn Init, Vrpn Tracker und Set Position.

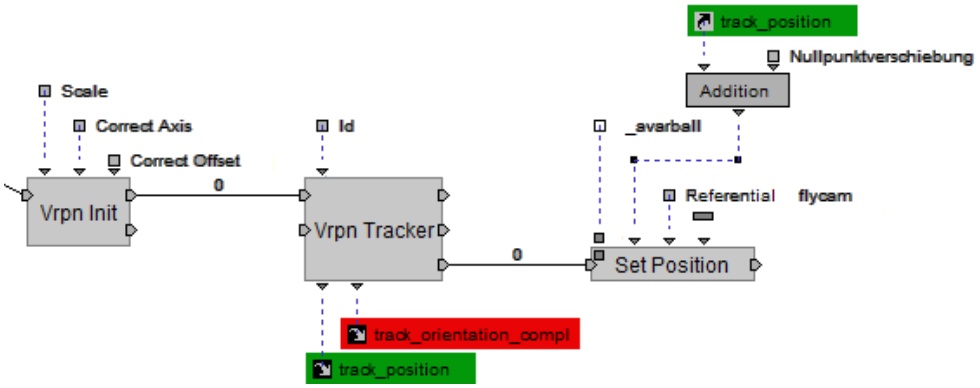


Bild 5.40: BehaviourGraph Tracking

Der Vrpn Init BB erstellt mit Hilfe der Configfiles, die Verbindung zum Vrpn Server. Dieser ist für das Virtual Reality Periphal Network zuständig. Konnte eine Verbindung aufgebaut werden, wird der obere bOut OK aktiviert, im anderen Fall der untere bOut Error. BOut OK aktiviert den Vrpn Tracker BB. Der Vrpn Tracker erhält vom Vrpn Server die Werte des Trackingsystems und gibt diese als Positionsvektor und Quaternion* aus.

*Quaternionen werden in der VR Entwicklung und in der 3D Animationen dazu verwendet, Orientierungen im Raum zu beschreiben

Mithilfe der bIns On und Off kann der BB ein- und ausgeschaltet werden. Er hat drei bOuts: Out on, Out off und Trigger. Die ersten beiden werden jeweils aktiviert wenn der BB ein- bzw. ausgeschaltet wird. Der letzte Ausgang wird in jedem Frame, nachdem die neuen Werte ausgegeben wurden, aktiviert. Dieser Ausgang ist verbunden mit dem Set Position BB.

Der Set Position BB hat die pIns Target, Position, Referential, Hierarchy und jeweils einen bIn und bOut. Set Position setzt den Positionsvektor eines Objektes innerhalb eines bestimmten Koordinatensystems. Für das Verifikationsbeispiel wird dem BB Set Position als Target der Avarworm und als Position eine Summe aus der Position die der Vrpn Tracker BB ausgibt und einem Korrekturwert übergeben. Als Referenzkoordinatensystem wurde die flycam gewählt. Dementsprechend wird bei jedem Aktivieren des BB der Avarworm im Koordinatensystem der flycam auf die aktuelle Position des Tracker gesetzt. Der Avarwom bleibt stets vor der Kamera, auch wenn diese sich bewegt.

Avarworm Der für den Avarworm entwickelte BG sorgt dafür, dass die vier Kugeln aus denen der Avarworm besteht (vgl. Kapitel 4.3.2 auf Seite 158), stets einen bestimmten Abstand voneinander halten. Die letzten drei Kugeln sollen immer der ersten Kugel folgen. Deren Position wird wiederum vom Set Position BB im Tracking BG gesetzt. Dadurch erhält der Avarworm sein typisches wurmartiges Erscheinungsbild. Bild 5.41 zeigt den BehaviourGraph.

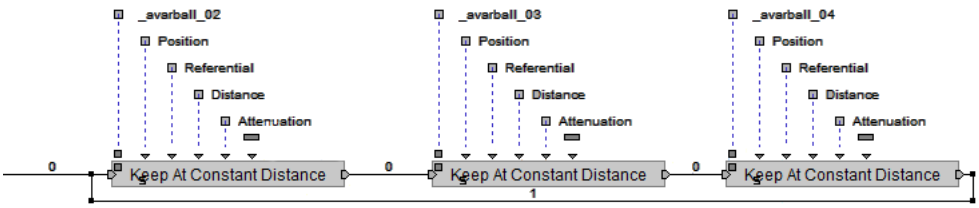


Bild 5.41: BehaviourGraph Avarworm

Das Skript besteht aus mehreren Keep At Constant Distance BBs. Dieser besitzt die pIns Target, Position, Referential, Distance, Attenuation und Hierarchy. Jeder einzelne BB sorgt dafür, dass der im Target festgelegte

Ball seinem Vorgänger mit einem bestimmten Abstand (Distance Parameter) und mit einstellbarer Verzögerung (Attenuation Parameter) folgt. So folgt z.B. der *_avarball_02* dem *_avarball* in einem Abstand von $0,15m$ mit einem Verzögerungswert von 3. Alle Werte wurden durch experimentelle Sichtüberprüfung so festgelegt, dass der Avarworm sowohl bei langsamer, als auch bei schneller Bewegung, stets seine charakteristischen Eigenschaften beibehält. Bild 5.42 illustriert die Umsetzung des implementierten Avarworms.

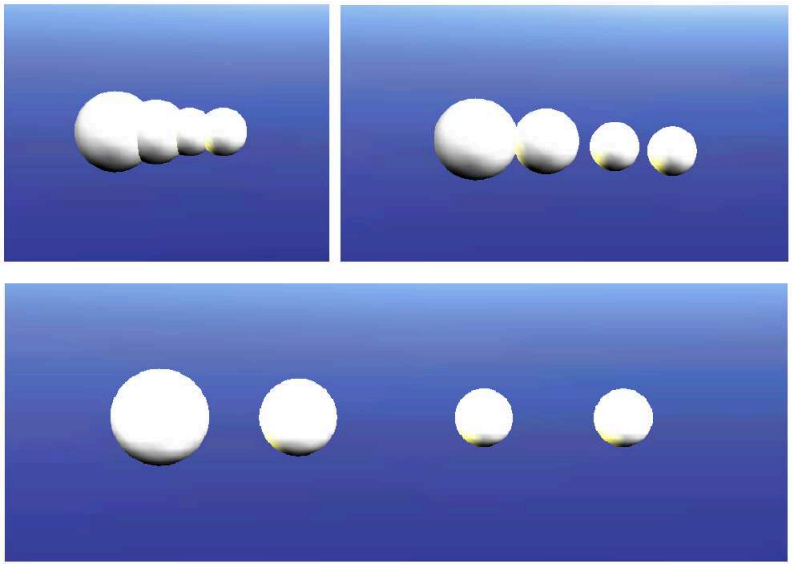
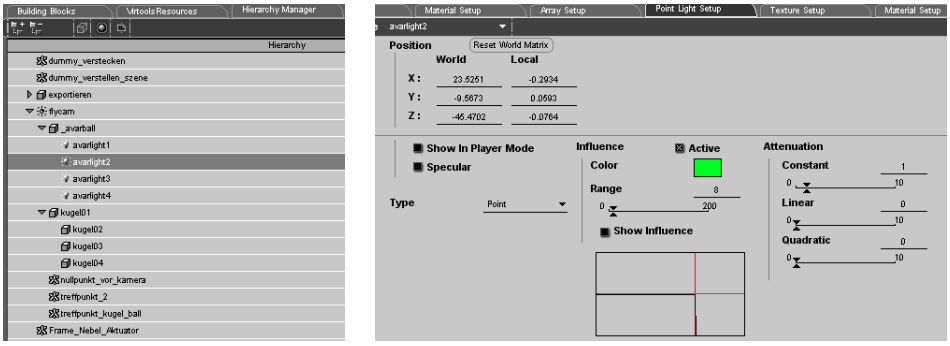


Bild 5.42: Umsetzung des Avators “Avarworm”

Zur Definition der Zusammenstöße zwischen einem Objekt und dem Avarworm gibt es prinzipiell zwei Möglichkeiten, die in Betracht zu ziehen sind. Zum Einen der Face Face Intersection BB, zum Anderen der Collision Detection BB. Der Unterschied liegt darin, dass für die Face Face Intersection schon bei der Erstellung der Schematics bekannt sein muss, welche Objekte kollidieren können. Bei wenigen Objekten in einer Szene stellt diese Methode ein probates Mittel dar. Ist die Szene komplex, wird die Face Face Intersection schwer durchschaubar und unflexibel. Dementsprechend ist diese in dem Verifikationsbeispiel lediglich für die Erkennung von Kollisionen zwischen Avarworm und Schaltflächen zur Steuerung des Fahrersitzes verwendet worden. Die zweite Möglichkeit, die aber mit Vorbereitungen und Uneindeutigkeiten verbunden ist, ist die Collision Detection. Dabei wird einer beliebigen Anzahl von Objekten ein Attribut (Moving Obstacle) zuge-

wiesen. Anschließend wird der Collision Detection BB mit dem Parameter des beteiligten Objektes, wie beispielsweise des Avarworm, gestartet. Im Falle einer Kollision des Avarworms mit einem Objekt, welches das obige Attribut (Moving Obstacle) besitzt, schaltet ein Behaviourausgang am Buildingblock. Zusätzlich wird am Parameterausgang der Name des Objektes, mit welchem der Ball kollidiert ist, ausgegeben. Dabei ist darauf zu achten, dass explizit nur diejenigen Objekte mit dem Attribut (Moving Obstacle) belegt sind, bei denen eine Weiterbearbeitung nach einem Zusammenstoß auch einen Sinn ergibt.

Zur Umsetzung der Tiefenerkennung (vgl. Kapitel 4.3.2 auf Seite 158) wurden virtuelle Kugeln aus Licht um die Avarballs angeordnet, die im Bereich $1 < r < 3$, je nach Abstand zum Zielobjekt, mit unterschiedlichen Farben leuchten. Dabei wurde ein Lichtpunkt im Zentrum jedes Avarballs gesetzt, so dass insgesamt vier zusätzliche Lichtquellen erstellt wurden. Pro Avarball werden im Point Light Setup die Parameter Color und Range entsprechend der Anordnung der Avarballs gewählt. Avarlight2 wird dementsprechend die Farbe Grün und der Leuchtradius 2 zugewiesen. Zudem wurde eine konstante Dämpfung des Lichts gesetzt, um die Übergänge der Farben zu nuancieren. Bild 5.43 beschreibt die Umsetzung und Bild 5.44 illustriert diese.



(a) Avarlight im Hierarchiemanager

(b) Setup des Avarlights

Bild 5.43: Umsetzung der Lichtkugeln am Avarworm

Die Synchronisierung des Avarworms auf dem Cluster geschieht anhand des VR Distrib Add-Remove BBs, in dem die zu synchronisierende Variable übergeben wird. Diese enthält die Änderung der Position anhand der getrackten Bewegung, die nur am Master-Rechner stattfindet. Da der Avarworm mit allen seinen Eigenschaften synchronisiert wird, ändert sich die

Position entsprechend auf allen Clustern.

Bild 5.44: Umsetzung der Lichtkugeln bei Annäherung des Avarworms an ein Objekt

Hilfe-Avatar ViMa gibt ihre Hilfestellungen als Sprache aus, die universell in jedem BG zum Einsatz kommen kann (vgl. Kapitel 4.3.2 auf Seite 161). Dazu beinhaltet der BG ViMa-Start den BB Test, der den weiteren BG ViMa spricht genau dann startet, wenn eine Wave-Datei dem Parameter was vima sprechen soll zugeordnet wird. Diese Zuordnung kann in allen BGs mit dem BB Identity erfolgen.

Der BG Vima spricht, der in Bild 5.45 dargestellt ist, beinhaltet zunächst zwei Op BBs, von denen der erste die aktuelle Position der Benutzeransicht (flycam) mit der Operation Get Position berechnet.

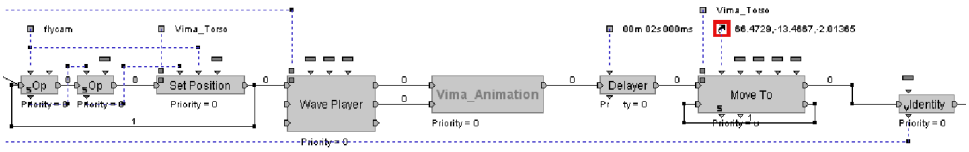


Bild 5.45: BehaviourGraph Vima Spricht

Der zweite BB Op addiert zu dieser Position einen Abstandsvektor, welcher durch den BB Set Position der Metapher ViMa zugeordnet wird. Somit befindet sich ViMa immer im selben Abstand vom Benutzer, egal, wohin dieser sich innerhalb der virtuellen Umgebung bewegt. Der nachfolgende BB Wave Player spielt die zugeordnete Wave-Datei des Parameters was

ViMa sprechen soll ab. Solange diese Datei abgespielt wird, schickt der BB Wave Player sein Ausgangssignal auf den BG ViMa_Animation, welcher im nachfolgenden Abschnitt beschrieben wird.

Mit Beendigung des Abspielens geht das Signal nach einer Zeitverzögerung (Delayer) in den BB Move to, der ViMa wieder an ihre ursprüngliche, vom Nutzer nicht sichtbare Position bewegt. Danach wird der Parameter was_vima_sprechen_soll durch den BB Identity mit einem Stringwert überschrieben. Dieser wird vom BB Test im BG ViMa Start nicht als Wave Datei erkannt und verhindert somit ein mehrmaliges Abspielen der zuletzt zugeordneten Datei.

Der BG ViMa Animation (vgl. Bild 5.46) beinhaltet die Bewegungen im Gesicht von ViMa, die durch elementare BBs beschrieben werden.

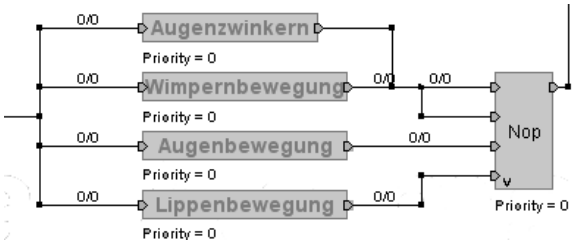


Bild 5.46: BehaviourGraph Vima Animation

Die elementaren BBs, die Teil der untergeordneten BGs des BG ViMa Animation sind, sollen im Folgenden kurz abstrahiert beschrieben werden.

1. BG Augenzwinkern: periodisches Erscheinen einer schwarzen Ellipsenebene in den Augen
2. BG Wimpernbewegung: periodisches Auf- und Abbewegen der Wimpern
3. BG Augenbewegung: periodisches Bewegen der jeweiligen Iris
4. BG Lippenbewegung: periodisches Auf- und Abbewegen der Lippen

Der vollständig implementierte Hilfe-Avatar ViMa ist in Bild 5.47 dargestellt. Die Aufgabe von ViMa besteht darin, innerhalb der Funktionsvisualisierung Hilfestellung zu leisten. Dabei spricht sie nochmals die nicht sofort

intuitiv visuell verständlichen Verbnamen aus und nennt dabei deren Führungsgrößen.

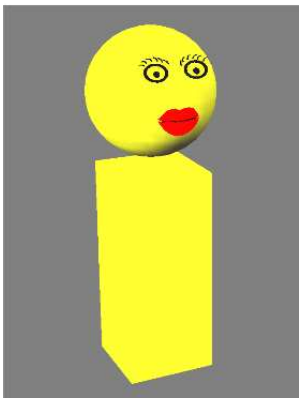


Bild 5.47: Realisierung des audiovisuellen Hilfe-Avatars ViMa

Ballsteuerung mit GoGo Um die räumliche Teife besser zu nutzen, wurde im Konzept die ballgesteuerte Systemsteuerung entwickelt, welche Steuerungsfunktionalitäten der Szene in die Tiefe des Raumes verlagert, von wo sie bei Bedarf jederzeit in den Vordergrund geblendet werden können (vgl. Kapitel 4.3.2 auf Seite 162). Als Bedienelemente wurden farblich getrennte Kugeln verwendet, da diese sich von dem Interaktionsavatar Avarworm besser selektieren lassen, als vergleichbare Elemente. Bild 5.48 zeigt die Visualisierung der Bedienelemente.

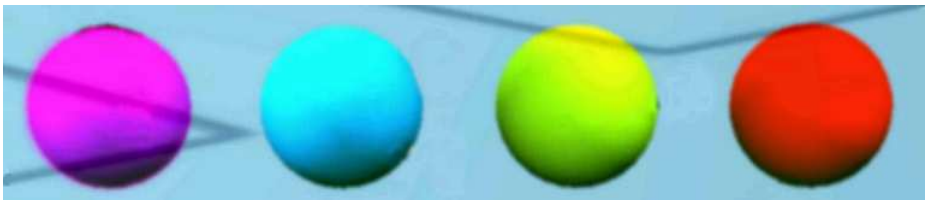


Bild 5.48: Realisierung der ballgesteuerten Systemsteuerung

Die ballgesteuerte Systemsteuerung mittels der Go-Go Metapher dient als Mechanismus zur Auswahl von Sichten von Funktion, oder zur Manipulation einzelner 3D Objekte. Der Bediener soll in der Lage sein, durch die Auswahl einzelner Komponenten in beliebiger Reihenfolge einzelne Funktionen

auszuwählen und so beispielsweise eine detailierte Visualisierung auszulösen (vgl. Kapitel 4.3.2 auf Seite 162). Bild 5.49 zeigt den BehaviourGraph der Ballsteuerung.

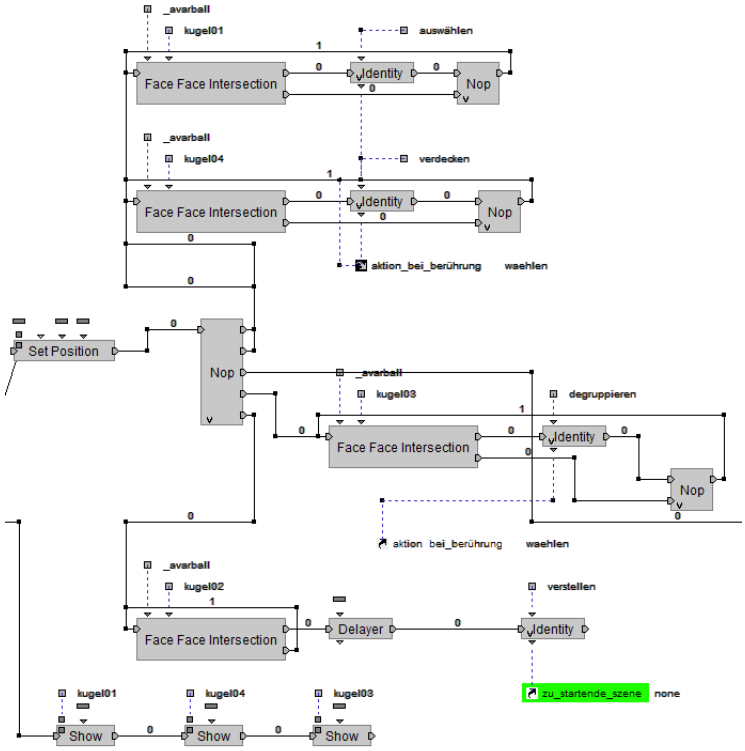


Bild 5.49: BehaviourGraph der Ballsteuerung

In dem BG sind lediglich drei Kugeln Aktionen zugewiesen. Angezeigt werden diese mit dem BB Show. Die jeweiligen Aktionen werden durch den BB Face Face Intersection mit dem Avarworm als Trigger ausgelöst. Dabei werden alle funktionalen Zusammenhänge in einem Array gespeichert, welches zu Beginn der Szene kopiert wird, um es durchgängig verändern zu können, und anschließend wieder zurück in die Ausgangssituation zu finden.

Zu der Auswahl eines Objektes, wird eine Liste aller aktuell visualisierbarer Funktionen erstellt. Diese wird analysiert und mit dem Funktionsvisualisierungs_array aus Bild 5.36 auf Seite 228 verglichen. Dabei wird nach einer Zeile gesucht, in welcher das Objekt zusammen mit der Funktion auf der Liste vorkommt. Wird eine solche Zeile gefunden, wird diese Funktion von der Liste gelöscht. Die Liste beschreibt also genau ein Negativ dessen, was eigentlich angestrebt wird. Dementsprechend wird die Liste durchgegangen

und im Array wird jede Zeile, welche die aktuelle Funktion der Liste enthält, gelöscht.

Soll ein Objekt verdeckt werden, wird im Array nach dem ersten Vorkommen des Objektes gesucht. Ist dies geschehen, wird in dieser Zeile die Funktionsspalte ausgelesen. Aus dem Array wird nun jede Zeile gelöscht, die in der Funktionsspalte den ausgelesenen Wert aufweist, da diese Zeilen zu einer Funktion gehören, die aufgrund der Abwahl (verdecken) des Objektes nicht mehr visualisierbar ist. Danach wird wiederum nach dem Objekt gesucht, beispielsweise wenn dies an mehreren Funktionen beteiligt ist, und entsprechend verfahren. Ist die Anzahl der verschiedenen visualisierbaren Funktionen im Array gleich 1, hat der Bediener seine Auswahl getroffen und die verbliebene Funktion wird visualisiert.

Um zu gewährleisten, das die Bedienelemente für den Benutzer immer erreichbar sind, wurde das Schema mit der GoGo Metapher, wie Bild 5.50 zeigt, ergänzt.

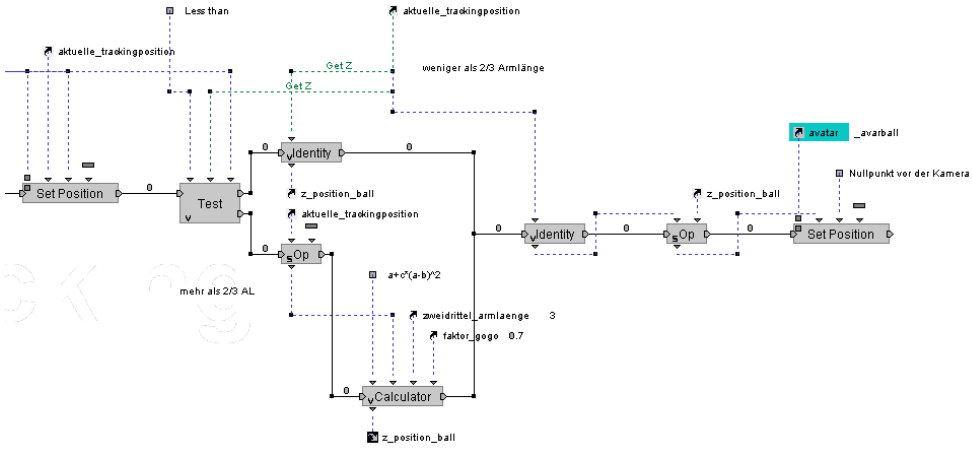


Bild 5.50: Ballsteuerung mit GoGo Metapher

Die GoGo Metapher wird dabei folgendermaßen umgesetzt. Beim Szenenstart drückt der Bediener einen Knopf an dem getrackten Interaktionsdevice. Dadurch wird die Position des Trackings in diesem Moment zur Nullposition gesetzt. Diese stellt den Körper des Bedieners dar. Alle späteren Positionen werden mit dieser Nullposition verrechnet. Daraus errechnet sich die aktuelle Position des Trackings relativ zur Nullposition. Dieser Abstand entspricht der Position der Hand relativ zum Körper aus der sich schließen lässt, wie weit der Arm ausgestreckt ist. Ist der Abstand Körper-Hand kleiner als $^2/_3$ der durchschittlichen Standardarmlänge, wird der Ball zur

Handposition bewegt, ist der Abstand größer, wird die Tiefenkoordinate des Balles (im Verifikationsbeispiel die z-Koordinate) entsprechend der Beschreibung des GoGo vergrößert (vgl. Kapitel 2.6.4 auf Seite 71), der Ball fliegt der Hand davon.

Window in World Bei der exozentrischen WiM arbeitet der Benutzer mit einer verkleinerten Darstellung der virtuellen Umgebung, die ikonische Repräsentationen aller Objekte enthält und vor allem den Vorteil bietet, viel einfacher manipulierbar zu sein als eine real skalierte Welt (vgl. Kapitel 4.3.2 auf Seite 166). Zur Realisierung der Interaktionsmetapher Window in World wird zunächst eine Kamera in die virtuelle Umgebung integriert und derart verschoben, dass ihr Blickfeld die Seitenansicht der crashaktiven Kopfstütze beinhaltet. Dies geschieht mittels der Menü-Symbole auf der linken oberen Seite innerhalb der Virtools Entwicklungsumgebung (vgl. Bild 5.51)

Bild 5.51: Menü Icons für die Umsetzung der Window in World Metapher

Die positionierte Kamera wird dem BB Additional View zugeordnet. Ebenso werden mit diesem BB die Parameter für die Größe und Positionierung des Window in World festgelegt. Dabei ist darauf zu achten, dass die Größe dieses Fensters nicht dynamisch zur Projektionsfläche zunimmt, sondern an die jeweilige Projektion angepasst werden muss. Bild 5.52 auf der nächsten Seite zeigt die Implementierung des BB Additional View. Durch die Signalschleife ist gewährleistet, dass das Window in World solange aktiv ist, bis

die Bewegung der Kopfstütze und eine angekoppelte Zeitverzögerung (Delayer) abgelaufen ist. Die Schleife wird mittels des BB-Eingangs Loop in geschlossen. Nach der Bewegungsanimation wird der BB Additional View wieder inaktiv und schickt das Signal durch seinen Ausgang (Out) an den nachfolgenden BB.

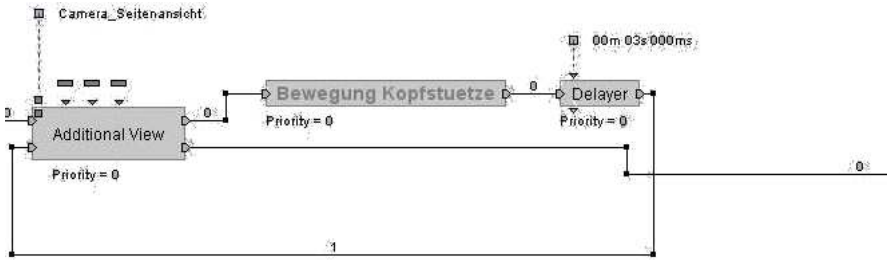


Bild 5.52: BehaviourGraph Additional View

Halbtransparente Objekte Für das Verifikationsbeispiel wurde die Virtual X-Ray Metapher in abgewandelter Form genutzt, die es erlaubt, Objekte dynamisch, halbtransparent darzustellen, und so beispielsweise Zugang zu verdeckt liegenden Objekten zu erhalten (vgl. Kapitel 4.3.2 auf Seite 168). Bild 5.53 zeigt die Umsetzung transparenter Objekte, um dahinterliegende Objekte nutzbar zu machen, am Beispiel der crashaktiven Kopfstütze.

Diese Metapher wird hier zur Unterstützung der Sicht der Funktionen verwendet. Die Auswahl einer Funktion, und der dazugehörigen Objekte erfolgt durch das Funktionsvisualisierungsarray (vgl. Bild 5.36 auf Seite 228). Wird eine Funktion ausgewählt, sollen lediglich die an der Funktion beteiligten Objekte sichtbar sein. Die Umgebung soll jedoch nicht komplett verschwinden, sondern den Gesamtzusammenhang weiter darstellen. Dazu wird allen Objekte außerhalb der gewählten Funkiton ein semitransparentes Material zugewiesen. Bild 5.54 zeigt den BehaviourGraph.

Der BB Group Iterator enthält alle gruppierten Objekte der Gesamtszene. Mittels des BB Row Search wird das Funktionsvisualisierungs_array nach Objekten aus der gewählten Funktion durchsucht. Ist ein Objekt an der Funktion nicht beteiligt, wird es mittels des BG Object Copy kopiert und der Kopie des Objektes ein halbtransparentes Material zugewiesen. D.h., dass

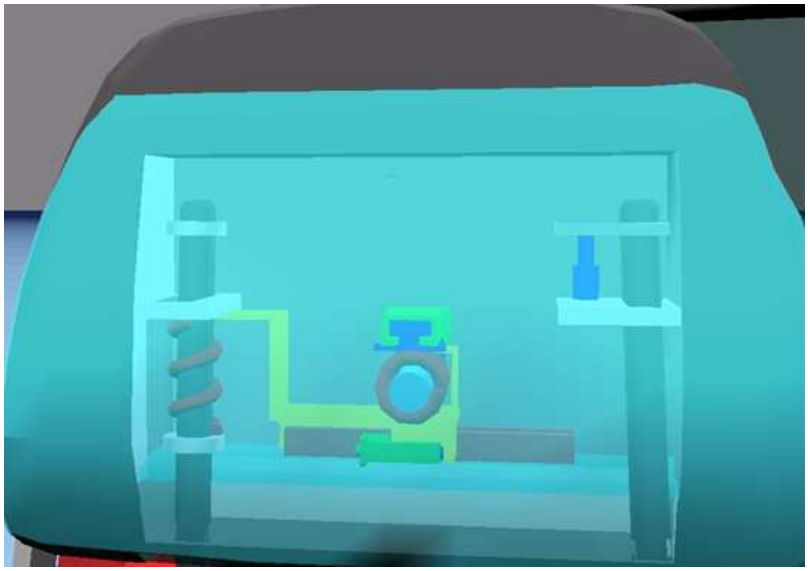


Bild 5.53: Halbtransparent dargestellte Abdeckung der Kopfstütze

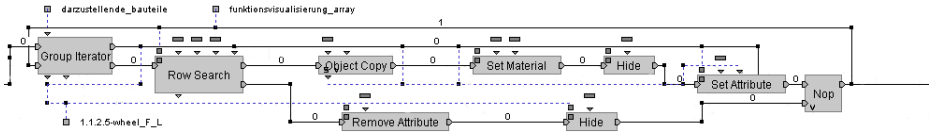


Bild 5.54: BehaviourGraph zur Zuweisung halbtransparenter Materialen auf die Objekte

bei erfolgreicher Suche nach der Zeile des Funktionsvisualisierungs_arrays, in dem sich ein Objekt einer Funktion befindet, geschieht nichts. Befindet sich keine Zeile im Array, bekommt das Objekt ein semitransparentes Material zugewiesen und außerdem wird das Attribut (Moving Obstacle) mittels des BB Set Attribute oder Remove Attribute weggenommen. Somit reagiert es nicht mehr auf Kollisionen mit dem Avatar. Der Schritt der Objektkopie ist notwendig, um den Ausgangszustand des Objektes wiederherstellen zu können. Wenn die Anzahl der verschiedenen visualisierbaren Funktionen im Array 1 ist, hat der Bediener seine Auswahl getroffen. Alle nicht gewählten Objekte sind nun halbtransparent dargestellt.

Interaktive räumliche Sphären Um ein Verirren in der Szene zu vermeiden, wurden transparente Sphären um die virtuelle Umgebung gelegt, die bei Berührung durch den Avatar bestimmte definierbare Aktionen triggern (vgl. Kapitel 4.3.2 auf Seite 165). Die äußere Hülle wurde im Verifikationsbeispiel

als unsichtbare Halbkugel definiert, die in einem festgelegten Abstand um die crashaktive Kopfstütze platziert wurde. Berührt der Trigger Avarworm diese, wird das Ziel der Kamera automatisch auf das Zentrum der Szene zurück gerichtet. Wird der Boden berührt, erfolgt zusätzlich mittels eines Move BBs die Wiederausrichtung der Kamera in der Ausgangsposition. Die innere Objektsphäre liegt um das Auto herum und wird erst dann aktiv, wenn die Distanz zwischen Avarworm und Fahrersitz entprechend klein ist. Durch die zusätzliche Funktionalität innerhalb der inneren Objektsphäre dient diese vor allem dazu bei Berührung wiederum die Ausgangssituation herzustellen. Solche berührungsempfindlichen Objektsphären liegen unsichtbar ebenfalls um die Bedienelemente des Fahrersitzes, mit dem Zweck die Steuerung mit dem Avarworm zu erleichtern.

Kollisionserkennung Kollisionen sind ein wesentlicher Teil bei einer physikalisch korrekten, realitätsnahen Abbildung (vgl. Kapitel 4.3.2 auf Seite 167). Der entwickelte BG Kollisionserkennung ist ein kleiner BG, der einen Op BB (Or paramOp), einen While BB und eine Not paramOp beinhaltet. Der Op gibt ein True aus, falls eine Kollision entsteht. Solange keine Kollision vorliegt, wird der Wert False ausgegeben. Da der While BB seinen bOut Loop Out nur mit einem True aktiviert, wird der paramOut des Op mit einer Not paramOp negiert. Der bLink am Ausgang des While BB wird dementsprechend nur aktiviert, wenn keine Kollision vorliegt. Bild 5.55 illustriert den BehaviourGraph.

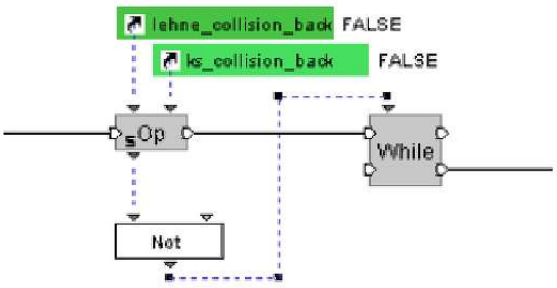


Bild 5.55: BehaviourGraph Kollisionserkennung

Die Auswirkung der Positionsänderung, die durch eine Kollision mit einem anderen Objekt ausgelöst wird, findet im Gegensatz zu Positionsänderungen, die durch ein Trackingdevice verursacht wurden, unabhängig auf allen

Rechnern statt, da die gleichen Algorithmen verwendet werden, denn Ereignisse werden über die Schematics gleich behandelt.

Bewegungsablauf der crashaktiven Kopfstütze

Zum besseren Verständnis der Funktionen durch den Nutzer wird der Bewegungsablauf der Kopfstütze in Virtools modelliert und soll als Kopplung zwischen der geometrischen und funktionalen Sicht innerhalb Virtools dienen. Dies geschieht mittels BehaviorGraphs innerhalb der Programmierumgebung von Virtools. Dabei wurden die Bewegungsabläufe eines Auslösevorgangs dieser Kopfstütze nacheinander programmiert. Das Auslösen der Kopfstütze geschieht durch einen Heckaufprall, der im Verifikationsbeispiel durch einen Stein und eine deformierte Heckseite der Fahrzeugkarosserie visualisiert wurde.

Die Bewegung startet durch Berührung des Steins durch den Avarworm (vgl. Kapitel 4.3.2 auf Seite 158). Nach dieser Berührung bewegt sich der Stein auf die Karosserie zu und kollidiert mit dieser. Daraufhin verfährt die crashaktive Kopfstütze, was im BehaviorGraph Bewegung Kopfstütze realisiert wird.

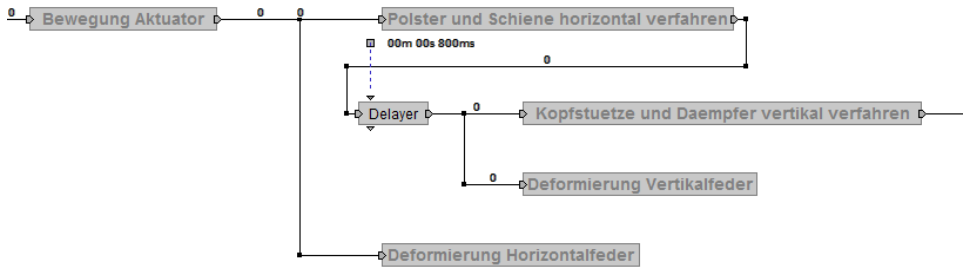


Bild 5.56: BehaviourGraph Bewegung Kopfstütze

In Bild 5.56 ist die Untergliederung dieses BG in seine, im Folgenden genannten, untergeordneten Graphen zu sehen.

- Bewegung Aktuator
- Polster und Schiene horizontal verfahren
- Deformierung Horizontalfeder

- Kopfstütze und Dämpfer vertikal verfahren
- Deformierung Vertikalfeder

Der zwischen der horizontalen und vertikalen Bewegung der Kopfstütze eingebaute BB Delayer realisiert eine Zeitverzögerung zwischen diesen beiden Bewegungen und dient der Veranschaulichung. Die BGs Bewegung Aktuator, Polster und Schiene horizontal verfahren und Kopfstütze und Dämpfer vertikal verfahren führen lineare Bewegungen aus. Wichtiges Element dieser Bewegungen ist der BB Move to, der Objekte in lineare Richtungen verfahren lässt. Hierzu wurde das Objekt dem BB als Target-Parameter zugeordnet. Diese Vorgehensweise wird ebenfalls bei allen folgenden BBs durchgeführt und garantiert deren Wiederverwendbarkeit in anderen BGs. Bild 5.57 zeigt den BG für das horizontale Verfahren des Bezugs (oberer Zweig) und der Führungsschiene (unterer Zweig). Der jeweils erste Operator-BB (Op) mit der Operation Get Position liefert die aktuelle Position der Objekte als Vektor, der mit einem Verschiebungsvektor durch den Befehl Addition im darauffolgenden BB Op summiert wird. Das Ergebnis ist die Zielposition, an die das jeweilige Objekt verschoben werden soll.

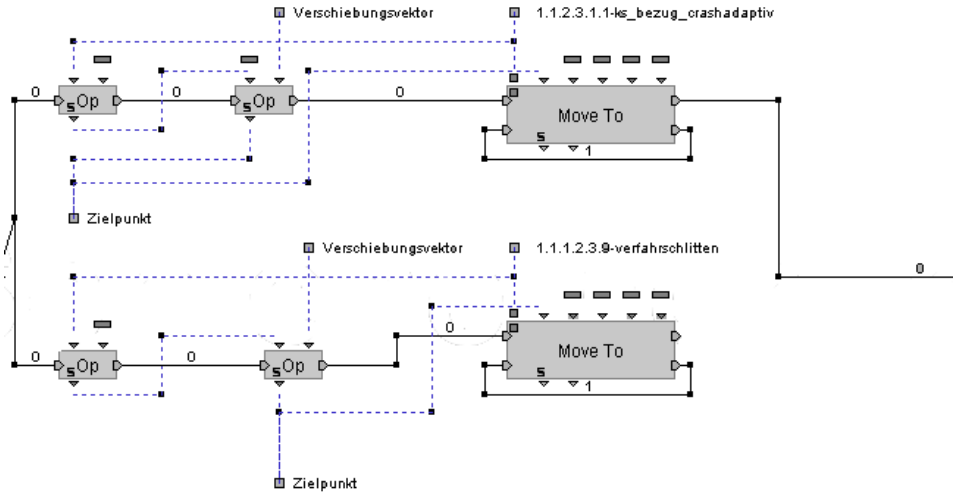


Bild 5.57: BehaviourGraph Polster und Schiene horizontal verfahren

Die Deformierung der Federn wird durch eine zeitliche Veränderung ihrer Weltmatritzen realisiert, welche die Form und Position der jeweiligen Feder beinhaltet. Bild 5.58 zeigt den BG für die Deformierung der Horizontalfeder mit den folgenden inhaltlichen Elementen.

- Der erste BB Op liefert als Ausgangsparameter die Weltmatrix der undeformierten Feder mit der Operation Get World Matrix.
- Der zweite Operator addiert auf diese Matrix eine Verformungsmatrix und liefert als Ergebnis die Zielmatrix der deformierten Feder.
- Der Interpolator BB erhält als Eingangsgrößen Ausgangsmatrix und Zielmatrix und nähert diese zeitlich mittels der Bezier Progression an.
- Der BB Set World Matrix erhält als Eingang den zeitlich veränderten Wert des Interpolator-BB und visualisiert die Veränderung der Ausgangsmatrix in die Zielmatrix.

Der BG zur Deformierung der Vertikalfeder ist äquivalent erstellt worden und wird dementsprechend nicht näher erläutert.

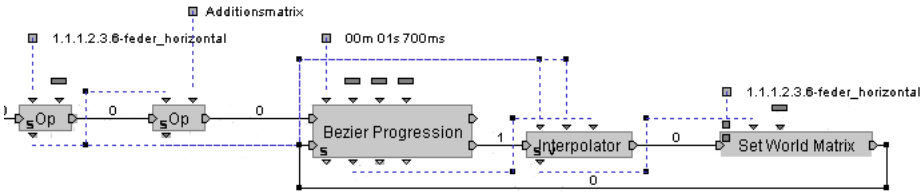


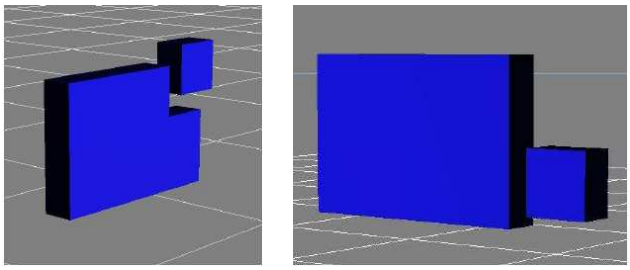
Bild 5.58: BehaviourGraph Deformierung der Horizontalfeder

Modellierung der Funktionsverben

Das in der Konzeptphase durchgeführte Brainstorming lieferte zwölf Metaphern für die Modellierung von Funktionsverben (vgl. Kapitel 4.3.3 auf Seite 170). Die Geometrien der Verben sind mit 3D Studio Max erstellt und nach Virtools importiert worden. Die Animationen der Funktionsverben werden in Virtools mittels BehaviorGraphs programmiert. Aus der Menge der zur Verfügung stehenden Funktionsverben, wurden alle Primären, sowie die für das Verifikationsbeispiel benötigten sekundären Verben ausgewählt. Die Realisierung der Funktionsverben wird im Folgenden erläutert.

Abtrennen und Hinzufügen Das Grundgerüst dieser Funktionsverben besteht aus einem kleinen und einem großen Quader. Für die Visualisierung des Abtrennens wird der kleine Quader vom großen entfernt, für das Hinzufügen wird er an ihn herangefügt. Beide Objektverhalten werden mit dem

BB Move to realisiert und entsprechen dem Prinzip aus Bild 5.57. Die Visualisierungen der beiden Funktionsverben sind in Bild 5.59 zu sehen.



(a) Abtrennen (b) Hinzufügen

Bild 5.59: Verbmetaphern Abtrennen und Hinzufügen

Importieren, Exportieren und Übermitteln Die in Kapitel 4.3.3 auf Seite 170 erarbeiteten Metaphern für die Funktionsverben Importieren und Exportieren beschreiben "einen Raum, in dem etwas hineingebracht (Importieren) oder hinausgebracht (Exportieren) wird". Dies wird mittels eines Behälters und eines Pfeils dargestellt. Für das Verb Importieren bewegt sich der Pfeil in das Behältnis, beim Exportieren bewegt er sich aus ihm hinaus. Beim Übermitteln bewegt sich der Pfeil von einem Behälter in den Anderen. Die Visualisierungen der Funktionsverben sind in Bild 5.60 zu sehen.

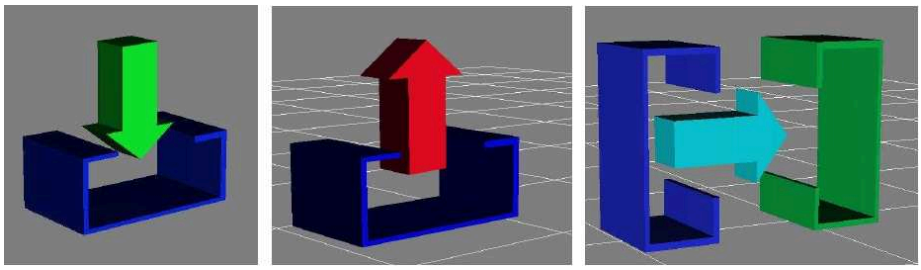


Bild 5.60: Verbmetaphern Importieren, Exportieren und Übermitteln

Zur besseren Veranschaulichung der Pfeilbewegung wurden die Behältnisse ohne Seitenwände modelliert. Die Bewegung der Pfeile wurde mit Hilfe des BB Move to erstellt und entspricht dem Prinzip aus Bild 5.57.

Sichern, Speichern und Leiten Die erstellten dreidimensionalen Metaphern für diese Funktionsverben entsprechen in ihrer Visualisierung den Korrespondierenden aus Bild 4.52 auf Seite 173: Vorhängeschloss, Diskette und Verkehrsschild. Die modellierten Metaphern dieser Funktionsverben sind in Bild 5.61 zu sehen.

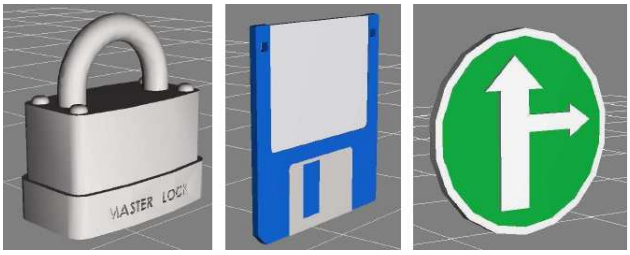


Bild 5.61: Verbmetaphern Sichern, Speichern und Leiten

An diesen Verben erfolgen keine direkten Animationen. Damit die Metaphern vom Nutzer als animierte Verben wahrgenommen werden, glänzen diese in periodischen Abständen. Um ein wiederkehrendes glänzendes Objekt zu visualisieren, wird ein Punktlicht über die Oberfläche dieser Objekte bewegt. Das Punktlicht ist als Menüsymbol hinterlegt und muss für das jeweilige Objekt positioniert werden (vgl. Bild 5.51). Zur Bewegung des Punktlichtes dient wiederum der BB Move to (vgl. Bild 5.57).

Verknüpfen Die Metapher für das Funktionsverb Verknüpfen wird, wie im Konzept in Kapitel 4.3.3 auf Seite 170 erarbeitet, als Knoten dargestellt. Die Metapher des Verbs Verknüpfen ist in Bild 5.62 zu sehen.

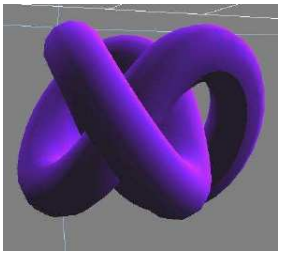


Bild 5.62: Verbmetapher Verknüpfen

Zur Animation dieses Knotens wird dessen Worldmatrix derart verändert, dass sich der Knoten zusammenzieht und wieder löst. Die Erstellung des

BG zur Animation des Knotens ist äquivalent zur Federdeformierung (vgl. Bild 5.58) und wird deshalb nicht näher erläutert.

Melden Das Funktionsverb Melden wird anhand des Konzeptes (vgl. Kapitel 4.3.3 auf Seite 170) mittels Schallwellen realisiert. Diese sind um eine Antenne gruppiert und erscheinen periodisch. Dazu müssen die modellierten Wellen an die Antenne platziert werden und in skalierter Größenreihenfolge erscheinen. Die Realisierung erfolgt mit den BBs Show und Hide, welche die Wellen sichtbar bzw. unsichtbar machen. Hierzu werden zuerst alle Schallwellen unsichtbar gemacht (Hide). Daraufhin werden diese einzeln, mit zeitlichen Abständen, sichtbar (Show). Die zeitlichen Abstände werden mit dem BB Delayer realisiert, dem eine Zeitspanne als Eingangsgröße zugeordnet wird. Bild 5.63 zeigt das Verb im Zustand aller sichtbaren Wellen, in Bild 5.64(a) ist der dazugehörige BG dargestellt.

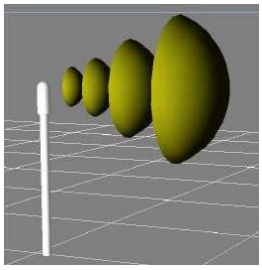
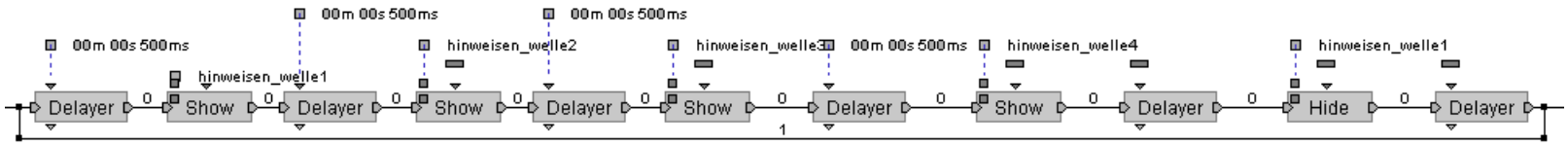
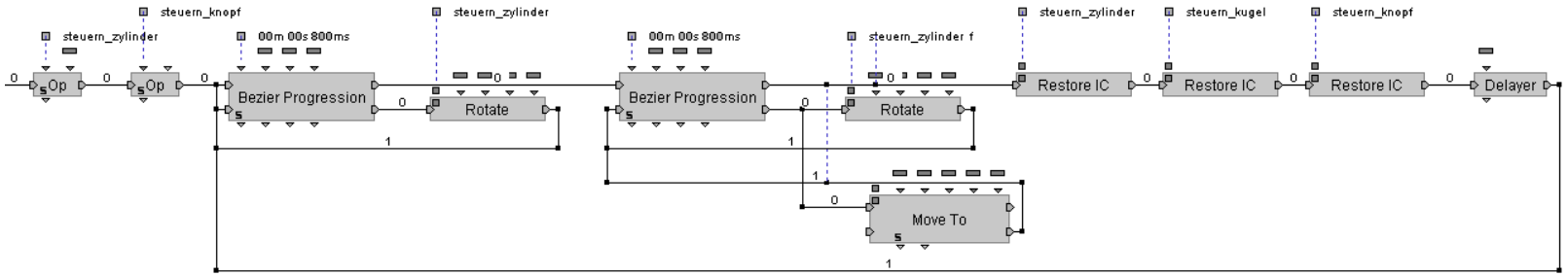


Bild 5.63: Verbmetapher Melden

Steuern Die im Konzept (vgl. Kapitel 4.3.3 auf Seite 170) erarbeitete Metapher eines Steuerknüppels wird als Joystick realisiert und dementsprechend modelliert. Für die Animation werden zuerst die Ausgangspositionen des Steuerhebels und des Feuerknopfes mittels der beiden Op BBs am Anfang des BGs festgehalten. Daraufhin wird die zeitliche Abfolge der Drehbewegung des Steuerknüppels durch eine Bezier Progression, die den BB Rotate beinhaltet modelliert. Mittels einer zweiten Bezier Progression wird der Steuerknüppel wieder auf seine Ausgangsposition gedreht. Innerhalb dieser zweiten Bezier Progression führt der BB Move to die Bewegung des Feuerknopfes aus. Da sich beim Ausführen dieses BG die Objekte nicht immer auf ihre exakte Ausgangsposition zurückbewegen, sind die BBs Set Position implementiert, die dem Steuerhebel und dem Feuerknopf wieder ihre Ausgangspositionen zuweisen, die sie von den oben genannten Op BBs erhalten.



(a) BehaviourGraph zum Funktionsverb Melden



(b) BehaviourGraph zum Funktionsverb Steuern

Bild 5.64: Die BehaviourGraphen der Funktionsverben Melden und Steuern

Das Bild 5.65 zeigt die Metapher für das Verb Steuern. Der dazugehörende BG ist in Bild 5.64(b) visualisiert.

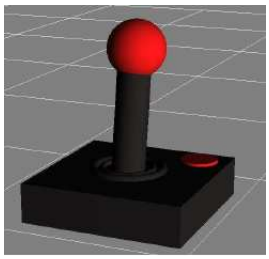


Bild 5.65: Verbmetapher Steuern

Umwandeln Zur Visualisierung des Funktionsverbs Umwandeln wurden ein Würfel und eine Pyramide gewählt. Die Umwandlung des Würfels in eine Pyramide erfolgt mittels des BB Mesh Morpher innerhalb einer Linear Progression, welche die lineare zeitliche Abfolge der Umwandlung steuert. Zu beachten ist, dass der BB Mesh Morpher nur zwischen Objekten ausgeführt werden kann, die dieselbe Anzahl an Eckpunkten haben. Da ein Würfel acht und eine Pyramide vier Eckpunkte besitzen, musste zur Nutzung des Mesh Morpher BBs die Pyramide approximativ aus acht Eckpunkten bestehen.

Dementsprechend besteht die Spitze der Pyramide nicht aus einem Punkt, sondern aus vier derart angenäherten Punkten, dass diese für den Betrachter nur als ein Punkt erkennbar sind, wie in der Draufsicht in Bild 5.66 schematisch zu sehen ist.

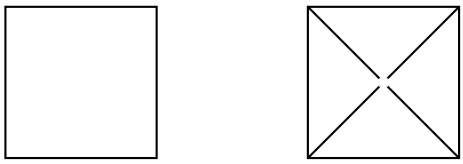


Bild 5.66: Draufsicht der Elemente des Verbs umwandeln

Somit besitzt die Pyramide ebenfalls acht Eckpunkte und hat die Bedingung für das Anwenden des BB Mesh Morpher erfüllt. Die Eingangsparameter

dieses BB werden durch die beiden vorangestellten Op BBs mit der Operation Get Mesh errechnet. Nach Beendigung der Linear Progression wird der Würfel mittels des BB Set World Matrix wieder auf seine Ausgangsmatrix gesetzt, die er durch den ersten BB Op erhält. Nach einer kurzen Verweilzeit (delayer) startet der Vorgang erneut. Bild 5.67 zeigt Ausschnitte aus der Umwandlung des Würfels in eine Pyramide. Der dazugehörende BG ist in Bild 5.68 zu sehen.

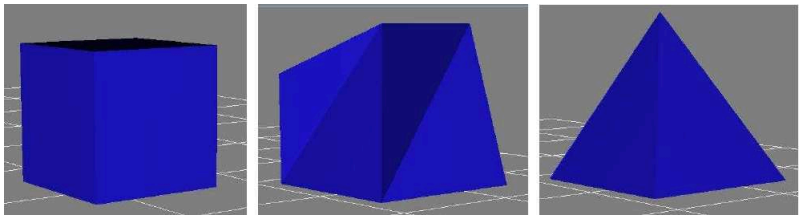


Bild 5.67: Verbmetapher Umwandeln

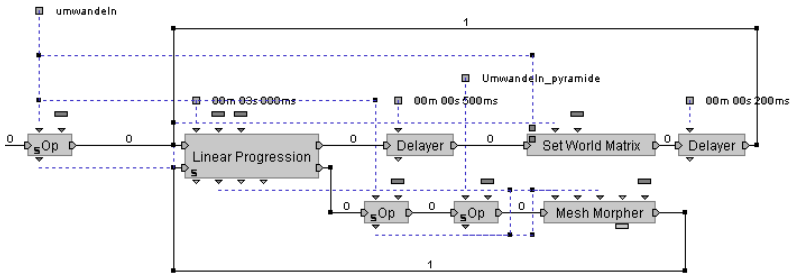


Bild 5.68: BehaviourGraph zum Funktionsverb Umwandeln

Interaktive Visualisierung der Funktionsstruktur

Die modellierte Funktionsstruktur, die bereits in Bild 5.17 auf Seite 204 und Bild 5.18 auf Seite 205 dargestellt ist, wird in Virtools mittels eines Array, in dem sowohl Informationen zur Funktionsstruktur als auch dessen Darstellung gefüllt ist, überführt (vgl. Kapitel 4.2.4 auf Seite 144 sowie Kapitel 5.2.2 auf Seite 193. Bild 5.69 zeigt dieses Übergabe-Array mit dem Namen funktionsvisualisierung_array, das die Elemente beinhaltet, die für die Modellierung der Funktionsstruktur des Verifikationsbeispiels von Bedeutung sind (vgl. Kapitel 4.3.3 auf Seite 169). Dabei ist der Array bereits in Sichten der Teilfunktionen der Funktionshierarchie (vgl. Bild 5.69, Spalte

5: FHID) geordnet worden.

	0 : Funktionsobjekt	1 : FO(Visio)	2 : Funktionsverb	3 : Verb ID	4 : Verb ShapeID	5 : FHID	6 : Richtung
0	1.1.1.2.3.12-aktuator	o-2-Aktuator/Treibladung/Treibladung	Energie leiten	f-2	12	fh_1.2	In
1	1.1.1.2.3.13-ausloesestift	o-3-Aktuator/Auslösestift/Auslösestift	Energie leiten	f-2	12	fh_1.2	Out
2	1.1.1.2.3.12-aktuator	o-1-Aktuator/Treibladung/Treibladung	Energie wandeln	f-1	23	fh_1.2	In
3	1.1.1.2.3.12-aktuator	o-2-Aktuator/Treibladung/Treibladung	Energie wandeln	f-1	23	fh_1.2	Out
4	1.1.1.2.3.13-ausloesestift	o-3-Aktuator/Auslösestift/Auslösestift	Energie leiten	f-3	57	fh_1.2	In
5	1.1.1.2.3.10-ausloeseblech	o-4-Auslösemechanik/Verfahreinheit/Auslöseblech	Energie leiten	f-3	57	fh_1.2	Out
6	1.1.1.2.3.10-ausloeseblech	o-4-Auslösemechanik/Verfahreinheit/Auslöseblech	Federenergie steuern	f-4	79	fh_1.2	In
7	1.1.1.2.3.10-ausloeseblech	o--1-None	Federenergie steuern	f-4	79	fh_1.2	Out

Bild 5.69: Funktionsvisualisierungs_array

Die benötigten Elemente zur Visualisierung der Funktionsstruktur sind folgendermaßen im Array aufgelistet:

- Spalte 0: Funktionsobjekte (Datentyp Entity)
- Spalte 2: Funktionsverben (Datentyp: String)
- Spalte 3: Gruppierung in Elementarfunktionen (Datentyp String)
- Spalte 6: Funktionsflussrichtung (Datentyp: String)

Zur Darstellung der Kreisanordnung anhand des Konzeptes aus Bild 4.54 auf Seite 175 wird folgendermaßen vorgegangen:

- Identifizierung der Funktionsverben im BG darzustellende Verben suchen
- Funktionsverben in das Array funktionsvisualisierung_verben schreiben
- Identifizierung der Eingangs- und Ausgangsfunktionsobjekte im BG Objekte anordnen
- Eingangsfunktionsobjekte in das Array funktionsvisualisierung_objekte_IN schreiben
- Ausgangsfunktionsobjekte in das Array funktionsvisualisierung_objekte_OUT schreiben
- Berechnen der neuen Positionen der Objekte und Verben im BG Anordnen
- Kopieren der Objekte und Verben im BG Anordnen

- Den kopierten Elementen die neuen Positionen zuweisen (BG Anordnen)

Bild 5.70 zeigt den BG Funktionsvisualisierung, dessen untergeordnete BGs darzustellende Verben suchen und Objekte anordnen sind.

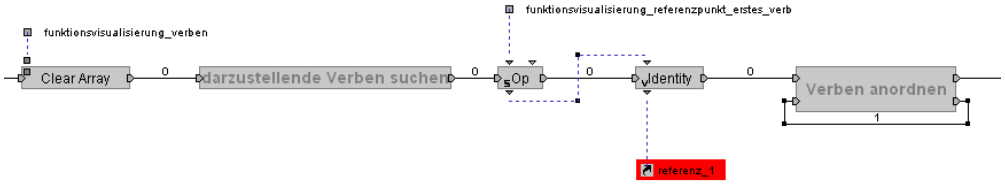


Bild 5.70: BehaviourGraph Funktionsvisualisierung

Der oben genannte BG Anordnen ist in der Unterstruktur von Objekte anordnen zu finden. Die beiden Arrays für die Eingangs- und Ausgangsobjekte entsprechen der ersten Spalte von Bild 5.69 auf der vorherigen Seite mit den jeweiligen zugeordneten Funktionsobjekten. Die beiden BBs Op und Identity ordnen dem Parameterwert referenz_1 einen ersten Fixpunkt in der virtuellen Umgebung zu, auf den das erste Verb der Funktionsvisualisierung platziert werden soll. Dabei nutzt der BB Op die Operation Get Position, die durch den BB Identity der Referenz zugeordnet wird, welche eine Position in Form eines Vektors darstellt und innerhalb des BB dynamisch verändert wird. Die blau unterlegten Elemente in Bild 5.70 stellen die Winkel und Radien für die Kreisstruktur dar und werden den Berechnungen innerhalb BG Anordnen als Shortcuts zugewiesen. Die Durchführung der oben aufgeführten Schritte zur Funktionsvisualisierung wird im Folgenden genauer erläutert.

Die Identifizierung der Funktionsverben erfolgt durch die zweite Spalte des funktionsvisualisierung_array, in der die Namen der verwendeten Funktionsverben und die dritte Spalte, in der die zugehörige Identifikationsnummer angegeben ist (vgl. Bild 5.69). Diese Verb-ID teilt die vom Array beschriebene Funktion auf und ordnet die Eingangs- und Ausgangsobjekte je einem Verb zu. Dies bedeutet, dass pro Identifikationsnummer ein Verb aus Spalte zwei zu visualisieren ist. Zu deren Identifikation muss das Array funktionsvisualisierung_verben nach den Elementen aus Spalte drei, den Verb-IDs, durchsucht werden. Dies geschieht mittels eines Iterator BB, der periodisch jede Zeile des Arrays funktionsvisualisierung_array ausliest und

die Werte aller Zeilen der aktuellen Zeile als Ausgang liefert. Dabei ist zur Ordnung der Verben nur der Wert aus Spalte drei (Verb-ID) erforderlich. Die erhaltene ID wird dem Eingang des BB Row Search zugeordnet, welcher das weitere Array funktionsvisualisierung_verben nach genau diesem Wert durchsucht. Falls die ID nicht gefunden wird, wird diese mit Hilfe des BB Add Row als neue Zeile in dieses Array eingefügt. Der dazugehörige Verbname aus Spalte zwei (vgl. Bild 5.69) wird ebenfalls mit diesem BB in eine weitere Spalte dieser Zeile eingefügt. Falls die Verb-ID bereits im Array funktionsvisualisierung_verben vorhanden ist, wird keine neue Zeile hinzugefügt. Der BB Nop beinhaltet keine Parameter oder Funktionen und dient lediglich einer übersichtlichen Bündelung der Signalflüsse. Der Suchvorgang dauert so lange, bis alle Zeilen des Arrays funktionsvisualisierung_array durch den Iterator durchlaufen worden sind. Das Array funktionsvisualisierung_verben muss zu Beginn des Suchvorgangs leer sein, was bereits am Anfang des BG Funktionsvisualisierung durch den BB clear array sichergestellt wird (vgl. Bild 5.70).

In Bild 5.71 ist der BG darzustellende Verben suchen zu sehen, der die Verben aus dem Array funktionsvisualisierung_array anhand ihrer IDs im Array funktionsvisualisierung_verben ordnet, welches in Bild 5.72 zu sehen ist.

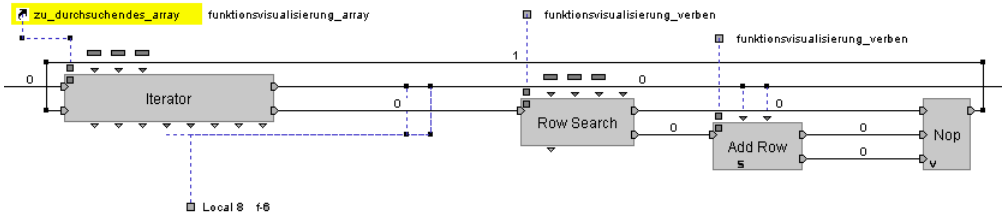


Bild 5.71: BehaviourGraph darzustellende Verben suchen

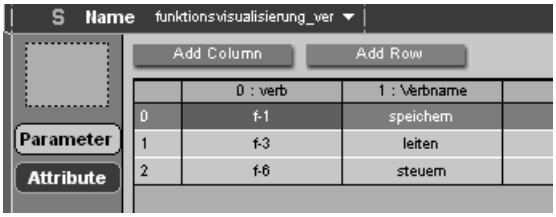


Bild 5.72: Array funktionsvisualisierung_verben

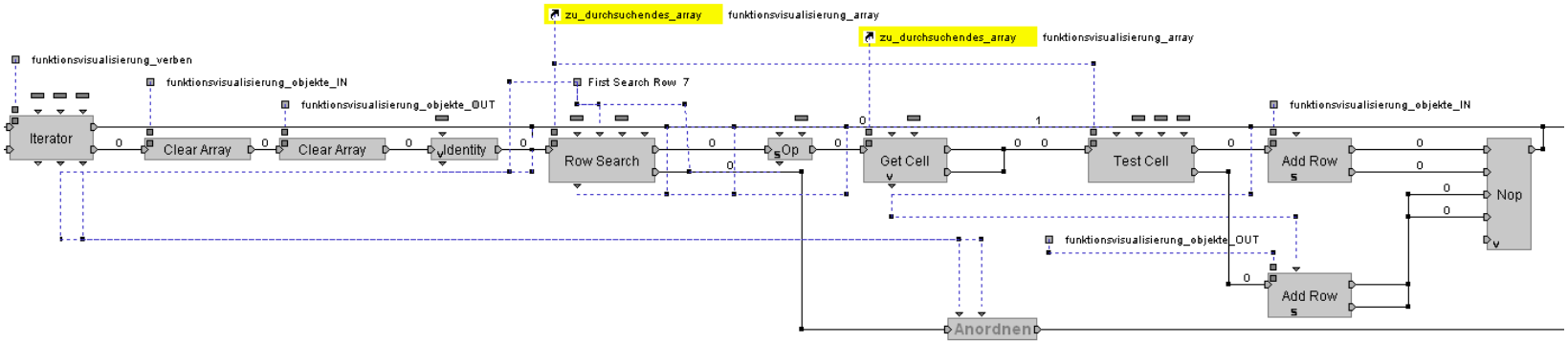
Die Funktionsobjekte sollen den anhand des BGs darzustellende Verben (vgl. Bild 5.71) geordneten Funktionsverben zugewiesen werden. Zur Iden-

tifizierung der Funktionsobjekte sind zunächst die Arrays für die Eingangsobjekte (funktionsvisualisierung_objekte_IN) sowie der Ausgangsobjekte (funktionsvisualisierung_Objekte_OUT) zu erstellen. Die Zuordnung zum jeweiligen Array erfolgt innerhalb des BG Objekte anordnen, dessen Aufbau in Bild 5.73(a) zu sehen ist und nachfolgend beschrieben wird.

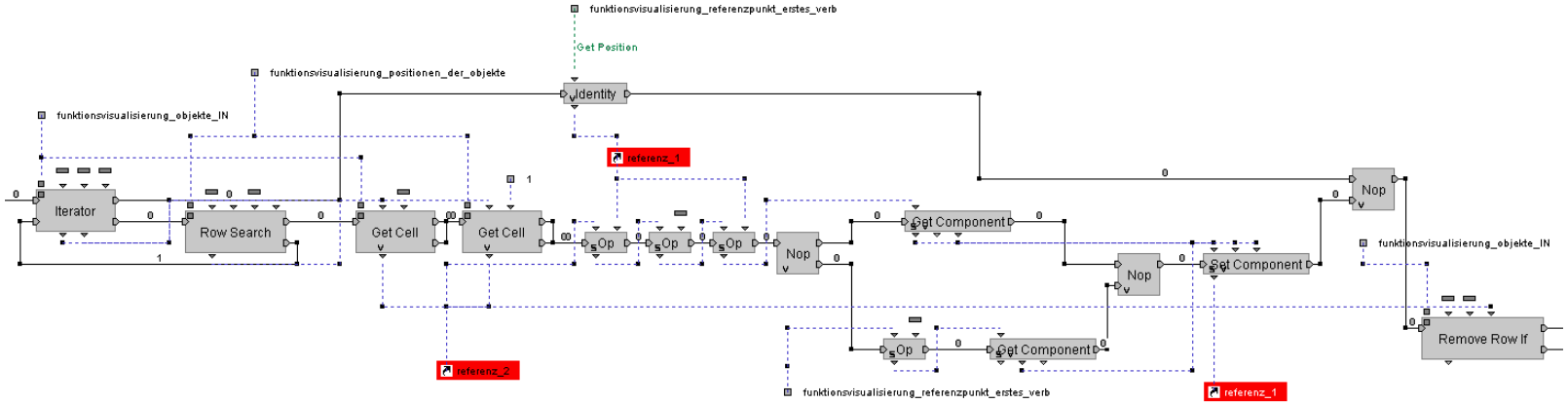
Zur Identifizierung der Objekte für das jeweilige Array der Eingangs-/ Ausgangsobjekte werden zunächst die Zeilen des Arrays funktionsvisualisierung_array mit einem Iterator BB durchlaufen, der als Ausgangswert die Verb-ID aus Spalte drei liefert. Diese wird mittels des Row Search BB mit den Werten in Spalte null des Arrays funktionsvisualisierung_array verglichen. Bevor die Zeilen mit Row Search durchsucht werden, werden zunächst die beiden Arrays für die Funktionsobjekte mit dem BBs Clear Array geleert. Der BB Identity setzt für alle weiteren zu suchenden Verb-IDs die erste zu durchsuchende Zeile für den BB Row Search auf den Wert Null. Bei Übereinstimmung der Verb-IDs aus den Arrays funktionsvisualisierung_array und funktionsvisualisierung_verben wird, mittels des BB Get Cell, das Funktionsobjekt der übereinstimmenden Zeile aus dem Array funktionsvisualisierung_array als Ausgangsparameter zugeordnet. Danach wird mit dem BB Test Cell überprüft, ob in der Spalte sechs innerhalb dieser Zeile als Richtung “IN” angegeben ist (vgl. Bild 5.69). Falls dies der Fall ist, wird das Objekt mit dem BB Add Row in das Array funktionsvisualisierung_objekte_IN geschrieben, ansonsten in das Array funktionsvisualisierung_objekte_OUT. Danach wird das Array funktionsvisualisierung_array erneut mit dem BB Row Search nach den Verb-IDs durchsucht, aber diesmal anfangend von der nächstfolgenden der zuletzt gefundenen Zeile. Dies wird durch eine Addition im BB Op sichergestellt, in dem immer die Zahl 1 aufaddiert und das Ergebnis dem BB Row Search dynamisch als erste zu durchsuchende Zeile zugewiesen wird. Dieser Vorgang läuft so lange ab, bis der BB Row Search keine Übereinstimmung mehr mit dem Wert des Iterators und dem Array funktionsvisualisierung_array findet. Das Signal wird dann an den BG Anordnen geschickt.

Der BG Anordnen schickt sein Ausgangssignal wieder zurück an den Iterator BB, welcher dann für die obenstehend erläuterten Vorgänge die nächste Verb-ID aus dem Array funktionsvisualisierung_array liefert.

Die Anordnung der Objekte wird zweifach betrachtet. Zum Einen folgt sie dem Aufbau der Produktstruktur (vgl. Kapitel 4.2.2 auf Seite 133), um die Wiedererkennung des Produktes zu gewährleisten. Zum Anderen sollen die betrachteten Baugruppen und Unterbaugruppen beliebig gruppierbar



(a) BehaviourGraph Objekte anordnen



(b) BehaviourGraph Position des Verbs berechnen

Bild 5.73: Die BehaviourGraphen Objekte anordnen und Position des Verbs berechnen

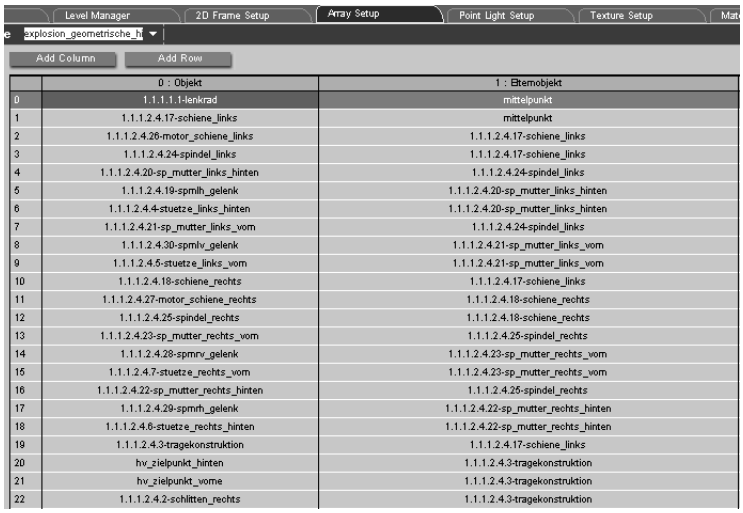
sein, um die für die Sichten der Funktionen aus der Funktionshierarchie notwendigen topologischen Anordnungen zu erstellen (vgl. Kapitel 4.3.1 auf Seite 155).

Um in einer bestimmten Sicht von Funktionen nicht alle Teile gleichzeitig zu sehen, sowie die korrespondierenen Teile übersichtlich im Raum anzuordnen werden im Rahmen der Objektanordnung die bestehenden 3D Objekte und Baugruppen in einer kreisförmig angeordneten Explosionsansicht überführt. Die Explosion der Baugruppen beruht auf einer Kombination des Hierarchiemanagers von Virtools und einer Hierarchisierung durch den Objektnamen. Durch den Objektnamen lassen sich Zusammengehörigkeiten der Baugruppen ableiten, durch den Hierarchiemanager ist es möglich, die Bauteile unter Beibehaltung ihrer Relativpositionen untereinander gleichzeitig an eine Zielposition zu verschieben. Der Vorgang der Explosion von Baugruppen wurde in Virtools folgendermaßen umgesetzt.

Zuerst wird mit Set Parent (none), angewendet auf alle Objekte, die alte Hierarchie zerstört. Wichtig ist, dass die Hierarchie, die für andere interaktive Szenen definiert wurde, vorab gespeichert wird. Anschließend wird allen sichtbaren Bauteilen das Attribute Moving Obstacle hinzugefügt. Daraufhin muss die Kollision zwischen Avarworm und Objekt festgestellt werden. Dazu wird das berührte Objekt ausgelesen und anschließend in einer lokalen Variable gespeichert. In der Szene wird jetzt ein Signal gesendet und ein entsprechender BehaviourGraph aktiviert. Die Arrays der Referenzen, Auslesehilfen und Zielpositionen müssen gelöscht werden. Ist dies durchgeführt, werden die Objekte einzeln analysiert um herauszufinden, ob sie bis zur Stelle der aktuellen Ebene in ihrer Namenshierarchie mit dem Referenzobjekt übereinstimmen. Ist das der Fall, wird das jeweilige Objekt zu einer weiteren Gruppe hinzugefügt. Außerdem wird der Wert an der Stelle $(aktuelle\_ebene + 1)$ in ein Hilfsarray geschrieben. Mittels des größten Wertes aus diesem Array kann auf die Anzahl der darzustellenden Baugruppen geschlossen werden. Stimmt jedoch der alte Wert nicht mit dem Referenzobjekt überein, wird das Bauteil versteckt und aus der Gruppe anzuordnende Bauteile entfernt. Sind alle Objekte analysiert worden, wird der Winkel ziwschen den entstandenen Baugruppen bestimmt durch die Formel $\varphi = {}^{2\pi}/_{n}$. Anschließend wird die Zielpositionen ermittelt. Bild 5.74 zeigt das Array Zielposition.

In der Gruppe anzuordende Bauteile sind nach dem letzten Schritt alle Objekte, die dargestellt werden sollen und vielleicht wiederum verschiedene Baugruppen bilden, also nicht am Stück bleiben sollen, sondern ex-

plodieren und sich auf einem Kreis anordnen. Gemeinsames Merkmal von Objekten der gleichen Objektgruppe ist ihr Hierarchiewert an der Stelle $aktuelle\_ebene+1$, der nun mit dem Group Iterator (anzuordnende bauteile) in Kombination mit einem VSL-Skript für jedes Objekt herausgefunden wird. Im Array Referenzen wird jetzt nach einem Eintrag unter diesem Wert gesucht. Ist dieser nicht vorhanden, existiert noch kein Referenzobjekt für die jeweilige Objektgruppe. Es muss also ein solcher Eintrag erzeugt werden. Dazu wird eine Zeile zum Array Referenzen hinzugefügt, die aus dem Wert, der die Baugruppe kennzeichnet und dem Objekt besteht. Nun wird dem Objekt eine Zielposition auf dem Kreis zugewiesen, jedoch wird das Objekt nicht dorthin bewegt, sondern die Position wird lediglich mit dem Objektnamen zusammen in das Array Zielpositionen geschrieben. Ist im Array Referenzen ein Eintrag für die Objektgruppe des aktuellen Objektes vorhanden, so wird das Objekt mit Set Parent (aktuelles objekt, referenzobjekt aus dem array) als Kind des Referenzobjektes deklariert und in der Schleife fortgefahren. Sind nun alle Objekte durchgegangen, wird das Array Zielpositionen anaylisiert. Zeile für Zeile werden die eingetragenen Objekte an ihre Zielposition verschoben. Da die Objekte aber auch Kindobjekte haben, werden diese mit verschoben.



	0 : Objekt	1 : Elternobjekt
0	1.1.1.1.1-lenkrad	mittelpunkt
1	1.1.1.2.4.17-schiene_links	mittelpunkt
2	1.1.1.2.4.26-motor_schiene_links	1.1.1.2.4.17-schiene_links
3	1.1.1.2.4.24-spindel_links	1.1.1.2.4.17-schiene_links
4	1.1.1.2.4.20-sp_mutter_links_hinten	1.1.1.2.4.24-spindel_links
5	1.1.1.2.4.19-spmlh_gelenk	1.1.1.2.4.20-sp_mutter_links_hinten
6	1.1.1.2.4.4-stuetze_links_hinten	1.1.1.2.4.20-sp_mutter_links_hinten
7	1.1.1.2.4.21-sp_mutter_links_vorn	1.1.1.2.4.24-spindel_links
8	1.1.1.2.4.30-spmlv_gelenk	1.1.1.2.4.21-sp_mutter_links_vorn
9	1.1.1.2.4.5-stuetze_links_vorn	1.1.1.2.4.21-sp_mutter_links_vorn
10	1.1.1.2.4.18-schiene_rechts	1.1.1.2.4.17-schiene_links
11	1.1.1.2.4.27-motor_schiene_rechts	1.1.1.2.4.18-schiene_rechts
12	1.1.1.2.4.25-spindel_rechts	1.1.1.2.4.18-schiene_rechts
13	1.1.1.2.4.23-sp_mutter_rechts_vorn	1.1.1.2.4.25-spindel_rechts
14	1.1.1.2.4.28-spmrv_gelenk	1.1.1.2.4.23-sp_mutter_rechts_vorn
15	1.1.1.2.4.7-stuetze_rechts_vorn	1.1.1.2.4.23-sp_mutter_rechts_vorn
16	1.1.1.2.4.22-sp_mutter_rechts_hinten	1.1.1.2.4.25-spindel_rechts
17	1.1.1.2.4.29-spmrh_gelenk	1.1.1.2.4.22-sp_mutter_rechts_hinten
18	1.1.1.2.4.6-stuetze_rechts_hinten	1.1.1.2.4.22-sp_mutter_rechts_hinten
19	1.1.1.2.4.3-tragekonstruktion	1.1.1.2.4.17-schiene_links
20	hv_zielpunkt_hinten	1.1.1.2.4.3-tragekonstruktion
21	hv_zielpunkt_vorne	1.1.1.2.4.3-tragekonstruktion
22	1.1.1.2.4.2-schlitten_rechts	1.1.1.2.4.3-tragekonstruktion

Bild 5.74: Array Explosionsdarstellung der Objektgruppen

Zur Auswahl und Darstellung der Objekte nach Sichten wurde zu jedem Objekt für jede Sicht, in welcher es sichtbar sein soll, ein Attribut zugewiesen. Das Attribut selbst hat keinen Wert, nur einen Namen. Grundlage

zur Zuordnung des Attributes zum Objekt ist das Array Sichtenmanager, welches in Zeilen alle Funktionsobjekte und in Spalten alle möglichen Sichten enthält. In den Zeilen ist nun jeweils ein boolescher Wert für sichtbar / unsichtbar in der jeweiligen Sicht hinzugefügt. Da das Array beim Auslesen über eine unbekannte Anzahl von Spalten verfügt, wird zuerst das zum Spaltenkopf passende Attribut über den Sichtenmanager_verfuegbare_sichten Array herausgefunden und dann Zeile für Zeile durchgegangen. In der ersten Spalte befindet sich immer das Objekt, in der n-ten Spalte der true/false-Wert, welcher darüber entscheidet, ob das Attribut zugewiesen wird oder nicht. Sind alle Zeilen durchgegangen, wird mit der nächsten Spalte ebenso verfahren und zwar so lange, bis alle Spalten durchgegangen sind.

Die Sichten werden entweder über die Ballsteuerung oder automatisch innerhalb der Gesamtfunktion ausgewählt. Dazu wird eine Nachricht Sicht hinzufügen oder Sicht entfernen versendet, die bewirkt, dass ein Attribut-Objekt in einer Variablen “hinzuzufügende oder zu entfernende Sicht” zum Array Sichtenmanager_aktuelle_ sicht hinzugefügt und entfernt wird. Anschließend wird die Gruppe Sichtbare Bauteile geleert und das Array wird geparst. Dazu werden Zeile für Zeile alle Objekte, die das aktuelle Attribut besitzen, wieder zur Gruppe hinzugefügt. Danach werden per Group Iterator (darzustellende bauteile) alle Funktionsobjekte der Szene durchgegangen. Ist ein Objekt in der jeweiligen Sicht vorhanden, wird es mit Show sichtbar gemacht, ansonsten wird es mit Hide versteckt.

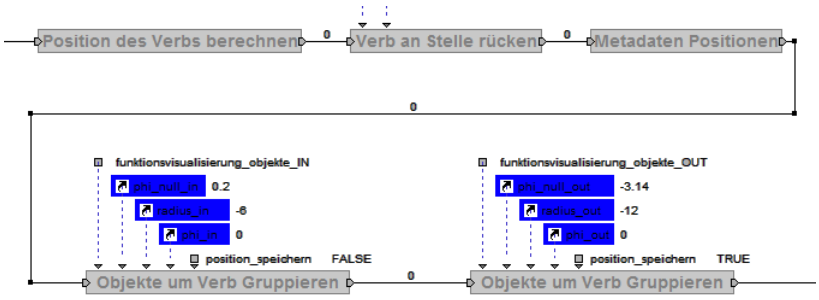


Bild 5.75: BehaviourGraph Anordnen

Der BG Anordnen positioniert alle Objekte und Verben in einer Kreisanordnung (vgl. Bild 4.54) innerhalb der virtuellen Umgebung. Da die Positionierungen zyklisch verlaufen, werden die Verbnamen als Eingangsparameter durch den BB Iterator des BG Objekte anordnen dem BG Verb an Stelle rücken zugeordnet. In Bild 5.73(a) ist die Steuerlinie zu sehen, die vom Ite-

rator BB in den BG Anordnen führt. Innerhalb des BG Anordnen führt diese Linie in den BG Verb an Stelle rücken, wie in Bild 5.75 zu sehen ist.

Bevor die jeweiligen Verben platziert werden, bestimmt der BG Positionen des Verbs berechnen deren Positionen. Bild 5.73(b) zeigt diesen BG, der die Berechnung der Positionsvektoren folgendermaßen ausführt. Zunächst wird das Array mit dem Namen funktionsvisualisierung_positionen_der_objekte benötigt, das die Namen der Eingangs- und Ausgangsobjekte und deren Positionen beinhaltet. Dieses Array wird in den beiden BG Objekte um Verb gruppieren zyklisch gefüllt (siehe Signalschleife in Bild 5.75). Der BB Iterator im BG Position des Verbs berechnen durchläuft das Array funktionsvisualisierung_objekte_IN und liefert als Ausgang die Namen der Verben. Diese werden durch den BB Row Search im Iteratorsignalkreis so lange mit den Werten des Arrays funktionsvisualisierung_positionen_der_objekte verglichen, bis sie übereinstimmen. Dann wird durch den BB Get Cell die hinterlegte Position des gefundenen Objektes dem Positionsvektor referenz_2 zugeordnet. Die Position des darauffolgenden Verbs in der Funktionsstruktur wird mit Hilfe der nächsten Op BBs berechnet. Die Position des letzten Eingangsobjektes (referenz_2) wird von der Position des letzen Verbs (referenz_1) durch den ersten BB Op subtrahiert. Dadurch ist der Abstand zwischen dem letzten Objekt und dem dazugehörigen Verb berechnet, der durch eine Multiplikation mit dem Faktor 2 im nächsten BB Op verdoppelt wird. Das Produkt wird dann mit der letzten Position des Verb (referenz_1) addiert, deren Summe die Position des nächsten Verbs in X- und Z-Richtung ergibt. Da das erste zu platzierende Verb kein vorhergehendes besitzt, wird diesem mit dem BB Identity als refernzpunkt_1 eine feste Position in der virtuellen Umgebung durch den Parameter funktionsvisualisierung_referenzpunkt_erstes_verb gesetzt (vgl. Bild 5.73(b) oben). Bild 5.76 veranschaulicht nochmals die von den drei Op BBs durchgeführte Vektorrechnung.

Aus Gründen der Veranschaulichung sollen die Verben der Funktionsstruktur in der virtuellen Umgebung auf gleicher Höhe sein. Daher wird der Positionsvektor referenz_1 mit den nachfolgenden BBs auf einen einheitlichen Wert gesetzt (vgl. Bild 5.73(b)). Die Position in den X- und Z-Koordinaten für die nachfolgenden Verben wird periodisch errechnet, durch den BB Get Component aus dem Ergebnis der Vektorrechnung ausgelesen und dem BB Set Component als Werte eines neuen Vektors zugeordnet. Dieser wird mittels des BB Identiy dem zyklisch veränderlichen Positionsvektor referenz_1 zugeordnet. Die neu zu bestimmende Y-Komponente wird dadurch

bestimmt, dass der BB Op im unteren Zweig durch die Operation Get Position den Positionsvektor des ersten Verbs an den BB Get Component weitergibt (vgl. Bild 5.73(b)). Dessen Ausgang für den Y-Wert wird der fehlenden Y-Koordinate des BB Set Component zugeordnet. Der BB Nop beinhaltet keine Berechnungen und dient lediglich der Bündelung der Signalflüsse.

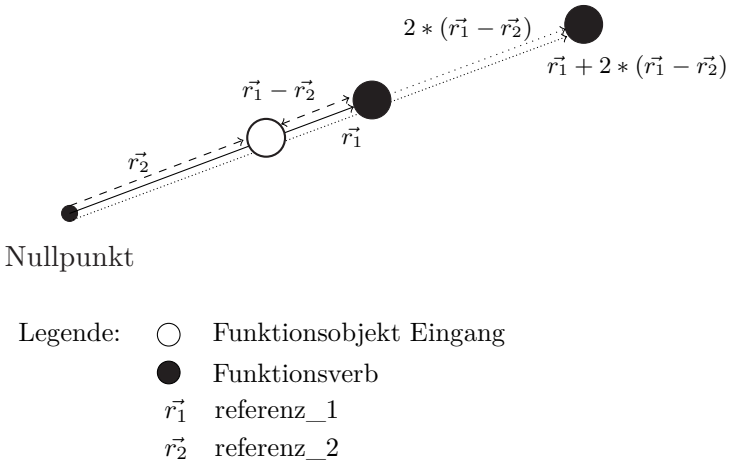


Bild 5.76: Vektorrechnung zum BG Position des verbs berechnen

Die errechnete Position wird mittels des BG Verb an Stelle rücken angeordnet, welcher in Bild 5.77 zu sehen ist.

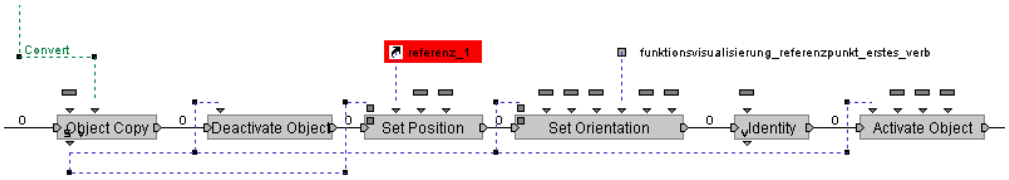


Bild 5.77: BehaviourGraph Verb an Stelle verschieben

Zuerst wird das jeweilige Funktionsverb aus dem BB Iterator des BG Objekte anorden (vgl. Bild 5.73(a)) mittels des BB Object copy kopiert. Die Kopie ist nötig, da Verben innerhalb von Funktionsstrukturen mehrmals auftreten können (vgl. Bild 5.17 auf Seite 204 und Bild 5.18 auf Seite 205). Die Animation jedes kopierten Verbs (vgl. Kapitel 4.3.3 auf Seite 169) wird zunächst durch den BB Deactivate Object gestoppt, bevor ihm der dazugehörende

Positionsvektor referenz_1, der im BG Position des Verbs berechnen errechnet wurde, zugeordnet wird. Das Stoppen der Animation für kopierte Objekte soll Rechenoperationen verringern und so die Rechenleistung für die Abläufe nicht unnötig belasten. Nachdem die Position des kopierten Verbs festgelegt ist, wird dessen Orientierung mit dem BB Set Orientation in eine einheitliche Richtung gesetzt. Dadurch ist die Funktionsstrukturrichtung der platzierten Objekte erkennbar (vgl. Kapitel 4.3.3 auf Seite 169). Nachdem das kopierte Verb platziert und ausgerichtet ist, wird dessen Animation mit dem BB Activate Object gestartet.

Da durch den BG Verb an Stelle rücken die Position für das Verb festgelegt ist, können nun die Positionsvektoren der Eingangs- und Ausgangsobjekte berechnet und zugeordnet werden. Dies geschieht durch die beiden BGs Objekte um Verb gruppieren im BG Anordnen (vgl. Bild 5.75 auf Seite 263), von denen der erste die Positionen der Eingangsobjekte und der zweite die Positionen der Ausgangsobjekte errechnet und zuweist. Die Objekte wurden bereits im BG Objekte anordnen den Arrays funktionsvisualisierung_objekte_IN und funktionsvisualisierung_objekte_OUT zugeordnet. Bild 5.78 zeigt den BG Objekte anordnen für die Eingangsobjekte, dessen Aufbau nachfolgend beschrieben wird.

Im ersten BG Objekte um Verb gruppieren wird zunächst die Anzahl der Objekte im Array funktionsvisualisierung_objekte_IN ausgelesen. Dies geschieht mittels des ersten BB Op, der die Operation get row count auf den Array ausführt. Das Ergebnis ist die Anzahl der Reihen aller Eingangsobjekte für das jeweilige Verb, die auf einem Halbkreis um dieses platziert werden sollen (vgl. Bild 4.54 auf Seite 175). Dafür wird zunächst der Winkel zwischen den einzelnen Objekten auf diesem Halbkreis mit dem BB Calculator durch die Formel π/n errechnet und durch den BB Identity dem Parameterwert aktuelles_phi zugeordnet. Dieser Winkel muß sich für jedes weitere Objekt auf dem Halbkreis verdoppeln.

Hierzu wird mittels des BB Iterator das Array funktionsvisualisierung_objekte_IN ausgelesen und jedem Objekt durch den nachfolgenden BB Op eine Multiplikation des Winkels aktuelles_phi mit der Zeilennummer jedes Objektes im Array zuordnet. Das Produkt wird durch den darauffolgenden BB Op mit dem vorgegeben Winkel phi_null_in addiert, welcher die Anfangsposition des ersten Verbs festlegt. Dieser Winkel wird nun mittels der nachfolgenden Calculator BBs und dem festgelegten Parameter radius_in in kartesische X- und Z-Koordinaten umgewandelt. Dies erfolgt für die X-Koordinate durch die Operation $r \cdot \sin(\phi)$ und für die Z-Koordinate durch

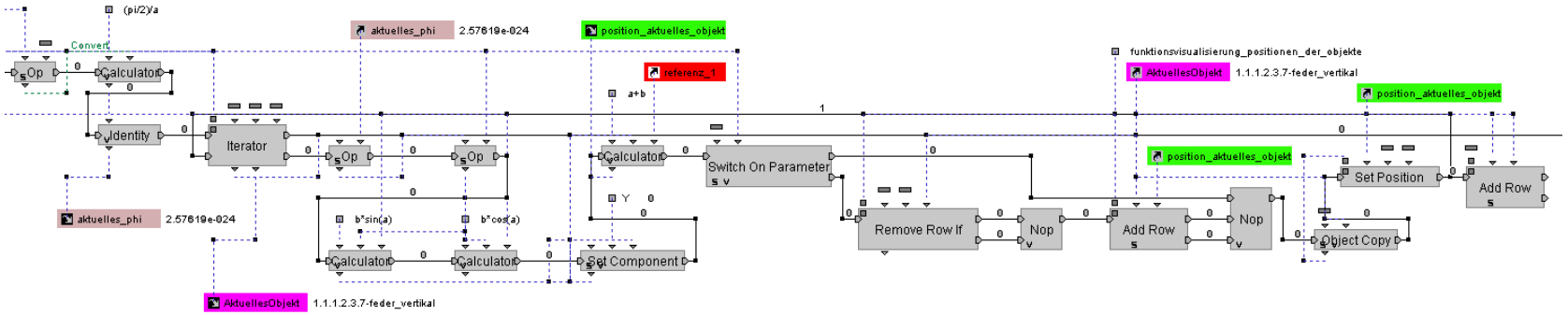


Bild 5.78: BehaviourGraph Objekt um Verb gruppieren

die Operation $r \cdot \cos(\phi)$. Der nachfolgende BB Set Component überführt die Koordinaten in einen Vektor, wobei die Y-Koordinate auf den festen Wert Null gesetzt wird. Das Ergebnis dieses BB ist ein Vektor, der den Abstand zwischen dem aktuellen Eingangsobjekt und dem dazugehörigen Verb beschreibt. Damit dieser Abstand eine Position im Weltkoordinatensystem einnehmen kann, wird dieser Vektor durch den nächsten BB Calculator mit der Position des aktuell korrespondierenden Verbs (referenz_1) addiert. Der neue Positionsvektor position_aktuelles_objekt wird dann mit dem BB Set Position einer Kopie des aktuellen Objektes zugeordnet, welche durch den BB Object Copy erstellt wird. Durch das Kopieren der Objekte können diese aus ihrer topologischen Anordnung herausgelöst werden.

Der zweite BG Objekte anordnen positioniert die Ausgangsobjekte um das Verb und entspricht in seinem Aufbau dem vorangegangenen BG. Die einzigen Unterschiede liegen darin, dass der Ausgangsradius radius_out der Kreisanordnung einen größeren Wert als der Eingangsradius besitzt und der Winkel phi_null_out um 180° zu phi_null_in versetzt ist, was durch die Anforderungen für die Kreisanordnung aus Kapitel 4.3.3 auf Seite 169 vorgegeben ist.

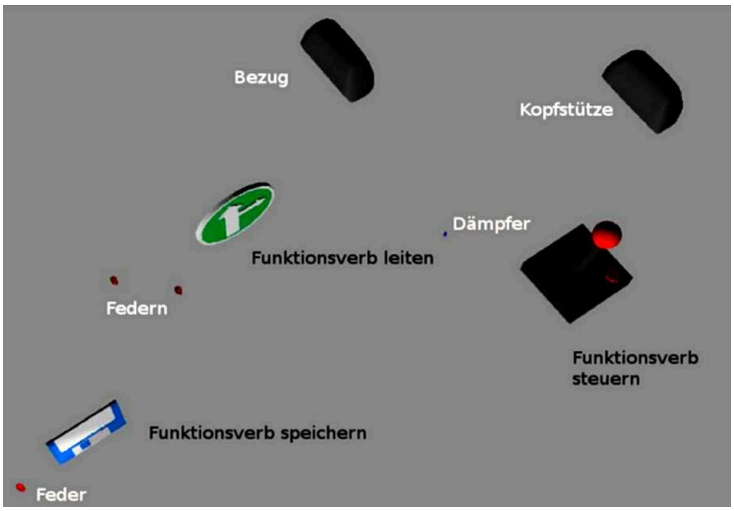


Bild 5.79: Positionierte Objekte und Verben in der virtuellen Umgebung

In Bild 5.79 ist die Anordnung der positionierten Funktionsverben und Funktionsobjekte zu sehen, die das Resultat der zuvor beschriebenen Schritte darstellt. Dieses beschreibt eine Teilfunktion der Funktionsstruktur des

Verifikationsbeispiels (vgl. Bild 5.17 auf Seite 204 und Bild 5.18 auf Seite 205).

- Verb: Federenergie speichern; Eingangsobjekt: Feder; Ausgangsobjekt: Feder
- Verb: Kraft leiten; Eingangsobjekt: Feder; Ausgangsobjekte: Kopfstützenbezug; Dämpfer
- Verb: Bewegung steuern; Eingangsobjekt: Dämpfer; Ausgangsobjekt: Kopfstützenbezug

Die Führungsgrößen (Kraft, Energie etc.) sind nicht visualisiert, da in der virtuellen Umgebung auf das Anwenden von Schriftzeichen verzichtet wurde (vgl. Tabelle 3.2 auf Seite 92). Diese werden durch den Hilfe-Avatar ViMa (vgl. Kapitel 4.3.2 auf Seite 161 und Bild 5.45 auf Seite 238) mittels Sprache ausgegeben.

KAPITEL 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Unternehmen im Maschinen- und Anlagenbau leiden neben dem für alle gültigen Kostendruck auch, bedingt durch Globalisierung, unter einer erheblich verschärften Wettbewerbssituation. Auf Marktanforderungen muss flexibel reagiert werden bei gleichzeitig wachsender Komplexität der Produkte. Unternehmenskonzentration und Auslagerung von Prozessen oder kompletten Unternehmensbereichen haben in den letzten Jahren den Bedarf geweckt, die Kernprozesse der Fertigungsindustrie besser miteinander zu integrieren, zu optimieren und dazu die Möglichkeiten moderner Technologien noch effektiver zu nutzen. Die wachsende Komplexität der Produkte und die immer stärkere Produktindividualisierung führen, bei immer kürzeren Produktentwicklungszeiten, zu einem außerordentlich schwer zu beherrschenden Abhängigkeitsgeflecht von Informationen in der Produktentwicklung.

Dem Erkennen und Verstehen dieser Abhängigkeiten durch Visualisierung, auch über die klassischen Domänengrenzen hinweg, kommt deshalb eine zentrale Bedeutung bei der Produktdefinition zu. Insbesondere existieren heute keine Ansätze Informationen aus frühen Konstruktionsphasen, wie z.B. der Funktionsmodellierung, integriert mit der sich daraus ergebenden Bauteilgestalt und den gegenwärtigen technischen Möglichkeiten wie den Technologien der virtuellen Realität zu visualisieren.

Unter Berücksichtigung dieser Randbedingungen muss ein entsprechender, neuartiger Ansatz dem Konstrukteur ein Werkzeug der Informationsvisualisierung an die Seite stellen, das diese wesentlichen Kernpunkte widerspiegelt. Dabei müssen die Informationen insbesondere aus den frühen Konstruktionsphasen wie der Funktionsmodellierung sowie neuartige Technologien stärker in die existierende rechnerunterstütze Produktentwicklung integriert werden. Im Rahmen dieser Arbeit wurde an dieser Stelle angesetzt. Ziel war es, Funktionsmodelle in virtuellen Umgebungen zu übertragen und dort interaktiv nutzbar zu machen. Die interaktive, immersive Visualisierung der Zusammenhänge der Gesamt- und Teilfunktionen mit der Produktstruktur spielt für das interdisziplinäre Gesamtverständnis komplexer Produkte eine tragende Rolle.

Um dies umzusetzen, wurde eine Methode zum Mapping der semantischen Informationen der Funktionsmodellierung eines Produkts auf den zugehörigen Szenengraph einer virtuellen Umgebung entwickelt, auf deren Basis ein virtueller Prototyp erstellt werden kann, der das Verhalten der Komponenten sowie die Interaktionen zwischen den Bauteilen visualisiert. Durch diese Simulation kann das Produkt bereits zu einem sehr frühen Zeitpunkt analysiert werden, um beispielsweise Entwurfsentscheidungen zu treffen, oder fehlerhafte Entwürfe vorzeitig zu erkennen. Es wurde im Verlauf der Recherche festgestellt, dass es bislang keine vergleichbare Arbeit gibt, die die Thematik einer semantischen Abbildung von Funktionsmodellen hinsichtlich der Simulation in einer virtuellen Umgebung diskutiert hat. In vorherigen Arbeiten wurden zwar Lösungen vorgeschlagen, um die Semantik eines Funktionsmodells einheitlich zu beschreiben, dabei stand jedoch meist die Suche von Entwürfen von Produkttypen in Design Repositories oder ähnliches im Vordergrund. Ferner waren die vorgeschlagenen Konzepte oftmals auf eine bestimmte Notation zur Funktionsmodellierung beschränkt. Das im Rahmen dieser Arbeit eingeführte Konzept hat den Anspruch, allgemeingültig genug zu sein, um sowohl auf verschiedene Notationen als auch verschiedene Werkzeuge anwendbar zu sein.

Durch die Entwicklung allgemeiner Metaphern zur Darstellung von einem Satz von Funktionsverben ist eine breite Anwendungsmöglichkeit der Funktionsmodelle gegeben. Virtuell vorhandene Funktionsobjekte (Produktkomponenten) können so aus ihren geometrischen und topologischen Zusammenhängen herausgelöst, in die Anordnung der Funktionsstruktur überführt und wieder in ihre ursprüngliche Anordnung zurückversetzt werden. Anhand der Entwicklung geeigneter Interaktionsmetaphern, soll dem Benutzer die

speziell für virtuelle Umgebungen angepasste Steuerung, Navigation und Manipulation intuitiv zugänglich gemacht werden. Dabei wurde Wert auf einen hohen Immersionsgrad sowie auf ästhetische Interaktionselemente gelegt, um die Akzeptanz des Benutzers zu unterstützen. Die Anordnung in virtuellen Funktionsstrukturen liefert dem Nutzer einen Mehrwert an Informationen über die funktionalen Zusammenhänge der Produktkomponenten. Dabei wurde darauf geachtet, dass unterschiedliche Funktionsstrukturen anhand des virtuellen Modells realisiert werden können und somit unterschiedliche Sichten auf die Produktkomponenten zulassen. Diese Strukturen können in einer Auflistung (Array) innerhalb der VR Entwicklungsumgebung abgelegt werden und bilden die Grundlage für die jeweilige Sicht. Somit ist auch hier durch den generischen Ansatz eine Erweiterbarkeit für andere Anwendungen gegeben.

Durch die Definition einer einheitlichen Datenschnittstelle ist das im Rahmen dieser Arbeit vorgeschlagene Konzept systemunabhängig, und erlaubt eine freie Auswahl des Modellierungswerkzeugs, der Modellierungsmethode und des 3D Visualisierungswerkzeugs. Die bidirektionale Datenkommunikation bzw. die Abbildungsentscheidungen werden durch eine graphische Benutzeroberfläche unterstützt. Wegen der Hierarchisierung und Unterteilung der Funktionen eignet sich das Konzept auch für die Modellierung komplexer Produkte.

Die im Konzept entwickelte Methode wurde am Beispiel einer crashaktiven Kopfstütze, die in der Umgebung Fahrersitz eingebettet ist, verifiziert. Zur Funktionsmodellierung wurde das Modellierungswerkzeug Visio verwendet, und eine Schablone mit vordefinierten Abbildungvorschriften realisiert. Die semantischen Informationen wurden in die 3D Szene des Verifikationsbeispiels in der VR Entwicklungsumgebung Virtools gemappt. Das Mapping erfolgte über eine in einer grafischen Benutzeroberfläche implementierten Abbildungsmatrix. Diese GUI wurde als PlugIn in der virtuellen Entwicklungsumgebung Virtools integriert, so dass der Benutzer hierdurch an dem Abbildungsvorgang beteiligt wurde und so seine Entscheidung dem System mitteilen kann. Über das von Virtools angebotene SDK wurden die semantischen Informationen in der Szene gespeichert, so dass diese von Verhaltensblöcke zugegriffen werden können. Die Interaktionsmetpahern wurden in den virtoolseigenen Verhaltensblöcken erstellt und mittels Tracking und Powerwall virtuell ausgegeben.

6.2 Ausblick

Das hier vorgestellte Konzept beschäftigt sich mit der semantischen Abbildung von Informationen aus Funktionsmodellen auf Szenengraphen virtueller Umgebungen, sowie die Nutzung von Interaktionsmetaphern zur intuitiven Kommunikation in der virtuellen Umgebung. Durch die Anwendung der neuartigen Technologien der virtuellen Realität ist die Erweiterung und Optimierung des implementierten Prototyps insbesondere hinsichtlich der prozessnahen Nutzung in der Industrie notwendig. Um beispielsweise einen vollständigen virtuellen Prototyp zu entwickeln, der eine komplette Simulation der Produktfunktionen erlaubt, sind die bisher abgebildete Informationen nicht ausreichend. Daher können in Zukunft weiterführende Untersuchungen angestellt werden, um das eingeführte Konzept dergestalt zu erweitern, dass eine interaktive Szene basierend auf der eingeführten Semantik das Produktverhalten physikalisch korrekt abbildet. Dafür sollte untersucht werden, wie die bisher fehlende Semantik in das Funktionsmodell integriert werden kann, so dass diese durch eine Erweiterung des hier vorgeschlagenen Einsatzes ebenfalls in einem Szenengraph abgebildet werden können.

Das im Rahmen dieser Arbeit verwendete Verifikationsbeispiel stellt eine prototypische Umsetzung des Konzepts dar und erzielt dadurch den Nachweis der Gültigkeit des Konzepts. Dementsprechend besteht in vernachlässigten Aspekten wie der Benutzerfreundlichkeit weiteres Verbesserungspotential, z.B. durch die Einführung von Verfahren zur Automatisierung bzw. Unterstützung der Abbildungsentscheidungen. Die entwickelten Visioschablonen können ebenfalls erweitert werden, so dass die Eingabe stärker unterstützt erfolgen kann, und dass eine semantiche Prüfung der Korrektheit des Funktionsmodells durchgeführt wird. Ferner wurde das Konzept nur an dem eingeführten Beispiel validiert und für eine bestimmte Notation und bestimmtes Werkzeuge implementiert. Um den Nachweis der Allgemeingültigkeit des Konzepts zu verstärken und die allgemeine Verfügbarkeit der gewonnene Ergebnisse zu gewährleisten, sollte das Konzept für andere Notationen, Werkzeuge und Umgebungen implementiert werden.

Desweiteren ließen sich die Ergebnisse dahingehend erweitern, dass eine semantische Suche von Szenengraphen in Design Repositories ermöglicht wird, so dass eine Vernetzung von Funktion, Prinzip und Verhalten standardisierter Bauteile automatisiert vom Rechner übernommen werden kann. Dafür sollte die in dem Konzept eingeführte einheitliche Schnittstelle zwischen Modellierungswerkzeug und virtueller Umgebung an eine für die Definiti-

on und Verarbeitung semantischer Netze notwendigen Standardnotation wie OWL orientiert werden. Diese Erweiterung könnte als eine Standardisierung des Mappingprozesses betrachtet werden, und somit zu einer nachhaltigen Verbreitung des Konzepts führen.

Da diese Arbeit den Grundstein für die Visualisierung von Funktionsmodellen in virtuellen Umgebungen darstellt, sind keine zweidimensionalen Visualisierungstechniken zum Einsatz gekommen. Deren Einsatz sollte im Kontext der Benutzerfreundlichkeit zukünftig überprüft werden. Denkbar sind beispielsweise Linienverbindungen zwischen Funktionsobjekten und Funktionsverben, um dem Nutzer die Zuordnung der Komponenten innerhalb einer komplexen Funktionsstruktur in der virtuellen Umgebung zu erleichtern.

Ebenso sollte untersucht werden, ob die Weitergabe von Metainformationen von einem aktuell rein statischen Zustand in einen dynamischen Zustand überführt werden kann. Denkbar ist dabei, dass die Sprachausgabe des Hilfeavatars ViMa dynamisch an neue Funktionsstrukturen angepasst wird. Darüberhinaus sollte die Möglichkeit eines dynamischen Mappings von Metainformationen in die Funktionsstruktur der virtuellen Umgebung weiter untersucht werden.

Zusammenfassend kann gesagt werden, dass diese Arbeit im Bereich der domänenübergreifenden virtuellen Produktentwicklung den Sprung von der rechnerunterstützen Übertragung der Funktionsmodelle in die virtuelle Umgebung schafft. Dabei bietet sie eine allgemeine erweiterbare Möglichkeit zur Funktionsvisualisierung in VR an, die mittels intuitiver Interaktionsmetaphern und Verwendung ästhetischer Interaktionselemente, die gegenwärtigen Technologien der virtuellen Realität dem Ingenieur näher bringt. Zudem legt sie den Grundstein zur Visualisierung von Produktsichten für die Abbildung weiterer Produktlebenzyklusphasen.

Appendices

ANHANG A

Taxonomie der Funktionsstrukturen

Class (Primary)	*Secondary*	*Tertiary*	*Power conjugate complements*	
			Effort analogy	*Flow analogy*
Energy			Effort	Flow
	Human		Force	Velocity
	Acoustic		Pressure	Particle velocity
	Biological		Pressure	Volumetric flow
	Chemical		Affinity	Reaction rate
	Electrical		Electromotive force	Current
	Electromagnetic		Effort	Flow
		Optical	Intensity	Velocity
		Solar	Intensity	Velocity
	Hydraulic		Pressure	Volumetric flow
	Magnetic		Magnetomotive force	Magnetic flux rate
	Mechanical		Effort	Flow
		Rotational	Torque	Angular velocity
		Translational	Force	Linear velocity
	Pneumatic		Pressure	Mass flow
	Radioactive/Nuclear		Intensity	Decay rate
	Thermal		Temperature	Heat flow

Bild A.1: Funktionsfluss Energie anhand des Vergleichs Leistung/Fluss

Class (Primary)	*Secondary*	*Tertiary*	*Correspondents*
Material	Human		Hand, foot, head
	Gas		Homogeneous
	Liquid		Incompressible, compressible, homogeneous,
	Solid	Object	Rigid-body, elastic-body, widget
		Particulate	
		Composite	
	Plasma		
	Mixture	Gas-gas	
		Liquid-liquid	
		Solid-solid	Aggregate
		Solid-Liquid	
		Liquid-Gas	
		Solid-Gas	
		Solid-Liquid-Gas	
		Colloidal	Aerosol
Signal	Status	Auditory	Tone, word
		Olfactory	
		Tactile	Temperature, pressure, roughness
		Taste	
		Visual	Position, displacement
	Control	Analog	Oscillatory
		Discrete	Binary
Energy	Human		
	Acoustic		
	Biological		
	Chemical		
	Electrical		
	Electromagnetic	Optical	
		Solar	
	Hydraulic		
	Magnetic		
	Mechanical	Rotational	
		Translational	
	Pneumatic		
	Radioactive / Nuclear		
	Thermal		

Overall increasing degree of specification →

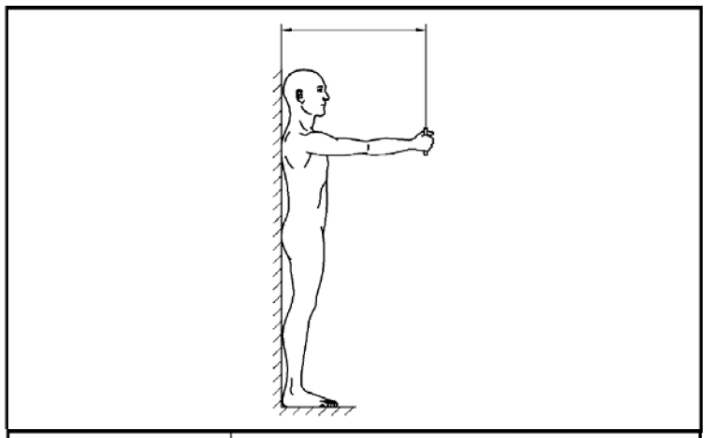
Bild A.2: Mögliche Objektklassen von Funktionsobjekten

Class (Primary)	*Secondary*	*Tertiary*	*Correspondents*
Branch	Separate		Isolate, sever, disjoin
		Divide	Detach, *isolate*, release, sort, split, disconnect, subtract
		Extract	Refine, filter, purify, percolate, strain, *clear*
		Remove	Cut, drill, lathe, polish, sand
	Distribute		Diffuse, dispel, disperse, dissipate, diverge, scatter
Channel	Import		Form entrance, *allow*, input, *capture*
	Export		Dispose, eject, *emit*, empty, *remove*, destroy, eliminate
	Transfer		Carry, deliver
		Transport	Advance, lift, move
		Transmit	Conduct, convey
	Guide		Direct, shift, steer, straighten, switch
		Translate	Move, relocate
		Rotate	Spin, turn
		Allow DOF	*Constrain*, unfasten, unlock
Connect	Couple		Associate, connect
		Join	Assemble, fasten
		Link	Attach
	Mix		Add, blend, coalesce, combine, pack
Control Magnitude	Actuate		Enable, initiate, start, turn-on
	Regulate		Control, equalize, limit, maintain
		Increase	*Allow*, open
		Decrease	Close, delay, interrupt
	Change		Adjust, modulate, *clear*, demodulate, invert, normalize, rectify, reset, scale, vary, modify
		Increment	Amplify, enhance, magnify, multiply
		Decrement	Attenuate, dampen, reduce
		Shape	Compact, compress, crush, pierce, deform, form
		Condition	Prepare, adapt, treat
	Stop		End, halt, pause, interrupt, restrain
		Prevent	Disable, turn-off
		Inhibit	Shield, insulate, protect, resist
Convert	Convert		Condense, create, decode, differentiate, digitize, encode, evaporate, generate, integrate, liquefy, *process*, solidify, transform
Provision	Store		Accumulate
		Contain	*Capture*, enclose
		Collect	Absorb, consume, fill, reserve
	Supply		Provide, replenish, retrieve
Signal	Sense		Feel, determine
		Detect	Discern, perceive, recognize
		Measure	Identify, *locate*
	Indicate		Announce, show, denote, record, register
		Track	Mark, time
		Display	*Emit*, expose, select
	Process		Compare, calculate, check
Support	Stabilize		Steady
	Secure		*Constrain*, hold, place, fix
	Position		Align, *locate*, orient
Overall increasing degree of specification →			

Bild A.3: Mögliche Verbklassen von Funktionsverben

ANHANG B

Standardarmlänge nach DIN-33402



Altersgruppen	Reichweite nach vorn (Griffachse) mm					
	Männer			Frauen		
	Perzentil					
Jahre	5	50	95	5	50	95
18–65	**685**	**740**	**815**	**625**	**690**	**750**
18–25	700	760	825	635	695	760
26–40	685	745	820	630	690	750
41–60	680	735	810	620	685	745
61–65	675	730	800	615	680	740

Bild B.1: Reichweite nach vorn (Griffachse)

Anmerkung: Das 5. Perzentil repräsentiert die Körpergröße *klein*, d. h. nur 5% der Werte liegen unterhalb des unteren Grenzwerts. Das 95. Perzentil

repräsentiert die Körpergröße *groß*, d. h. nur 5% der Werte liegen über diesem oberen Grenzwert. Das 50. Perzentil repräsentiert die Körpergröße *mittelgroß*, d. h. 50% der Werte liegen über oder unter diesem Median.

ANHANG C

VR Software

In den letzten Jahren haben sich mehrere VR Softwareprodukte auf dem Markt etabliert. Zu den weit verbreiteten Systemen zählen Quest 3D der Firma Act-3D, RTT Deltagen der Firma Realtime Technology, EON Professional der Firma EON Reality und Virtools von Dassault Systems, welches bereits in Kapitel 5.1.2 auf Seite 180 ausführlich beschrieben wurde. Alle Entwicklungsumgebungen werden in Industrie und Forschung für den Entwurf von dreidimensionalen Szenarien genutzt. Virtools und Quest 3D werden ebenso für die Entwicklung von Computerspielen genutzt. Nachfolgend werden die oben genannten Entwicklungsumgebungen kurz erläutert und abschließend kurz tabellarisch zusammengefasst.

Quest 3D

Quest3D ist ein Multimedia-Baukastensystem auf der Basis von DirectX. Einsatzgebiete sind insbesondere interaktive 3D-Echtzeitanwendungen. Programmierbasis ist eine graphische datenflussorientierte Baumstruktur. Die Szenegraphen werden im Gegensatz zu Virtools hierarchisch angeordnet. Verzweigungen entstehen durch das Hinzufügen von Verhaltensbausteinen an einen in einer Ebene. Jeder Knoten des Baumes besitzt sowohl eine Elternbeziehung zu den Nachfolgenden als auch eine Kindbeziehung zum nächsthöheren Knoten. Ausnahmen bilden da logischerweise die Wurzel des Verhaltensbaumes, dass nur einen Elternknoten darstellt und die Knoten in der untersten Ebene, die nur Kinder darstellen. Während die Ablaufsteuerung unidirektional von den Eltern zu den Kindern erfolgt, kann eine

Werteübergabe auch bidirektional vom Kind zum Elternteil erfolgen. In Bild C.1 ist zu sehen, dass im linken Zweig des Graphen eine Werteübergabe stattfinden und das endgültige Setzen der Daten in Ablaufrichtung auf der rechten Seite vonstatten geht. Dies wird durch die Pfeilanimationen verdeutlicht, die auf den Signallinien sitzen.

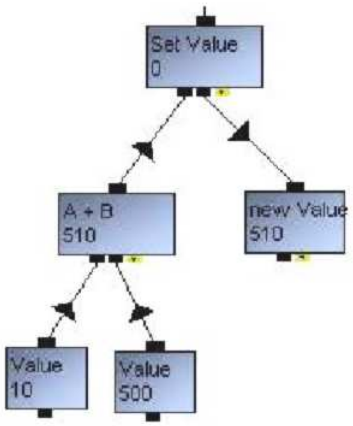


Bild C.1: Ablaufrichtung und Datenübergabe bei Quest3D

Im Gegensatz zu den 400 implementierten Verhaltensbausteine bei Virtools Dev, besitzt Quest 3D nur eine Bibliothek mit 100 sogenannten Channels. Diese sind ebenfalls über SDK erweiterbar. Die weiteren Funktionalitäten und die Benutzeroberfläche ähneln denen von Virtools, wobei Quest 3D das Produktportfolio von Dassault Systems nicht integrieren kann und sich beim Import von CAD Modellen aktuell noch auf das Programm AutoCAD beschränkt. (Maser, 2008)

RTT Deltagen

RTT DeltaGen ist ein System, dessen Hauptaugenmerk auf der Aufbereitung komplexer 3D CAD Daten für die Echtzeitvisualisierung liegt. Geometrische Daten können aufbereitet, mit visuellen Effekten versehen und in virtuelle Umgebungen integriert werden. Durch die hochauflösende Visualisierung der Daten und komplexen Tools für Licht- und Schattenbrechung erreicht RTT Deltagen eine realistische Darstellung von Objekten in der virtuellen Umgebung. Daher hat sich dieses System im Bereich der Produktpräsentationen in sogenannten Showrooms durchgesetzt.

Die Datenverarbeitung kann im Batch-Modus (RTT RealBatch) oder interaktiv durchgeführt werden. Der Benutzer kann auf einfache Weise Animationen und Modellvarianten definieren, sowie hochaufgelöste Bilder, Videos und interaktive QuickTime VR Filme rendern. RTT Deltagen kann die Formate CSB, STL, VRML und IV sowie Strukturdateien im PLMXML-Format einlesen. Weiterhin können CAD Daten wie Wire, Catia V4 und V5, Parasolid, Pro/E, IGES, JT, STEP, und VDA konvertiert werden.

Trotz der hochauflösenden Grafik und vielen Schnittstellen zu CAD Systemen ist RTT Deltagen für die Bildung einer interaktiven, immersiven virtuellen Umgebung weniger gut geeignet als beispielsweise Virtools. Dies liegt einerseits an der hohen benötigten Rechenleistung aufgrund der hochauflösenden Grafiken. Andererseits sind die Möglichkeiten zur Modellierung von Animationen eingeschränkt. Bild C.2 zeigt ein CAD Modell, das mit RTT Deltagen durch Licht-/Schattenverhältnisse und Oberflächenreflexionen aufbereitet wurde. (Maser, 2008)

Bild C.2: Hochauflösende Grafik von RTT Deltagen

EON Professional

EON Professional ist ein Komplettpaket von Werkzeugen für die Entwicklung interaktiver 3D Modelle und VR-Applikationen. Neben dem Authoring Tool EON Studio und dem Raptor-Plugin für den Import von 3D Studio Max Dateien sind im System noch folgende Werkzeuge enthalten.

EON Visual Effects ordnet den Objekten Oberflächeneigenschaften zu und enthält eine Reihe von Schattierungen wie phong-shading, bumpmapping, darkmapping, hatch shading etc.

EON Physics ermöglicht die Zuordnung und Simulation physikalischer Eigenschaften wie Kinematik, Kollisionserkennung, Gravitation, Reibung und Dynamik. Dieses Tool entspricht dem Physics Pack von Virtools.

EON CAD ist ein Integrationswerkzeug für den Import von CAD- und 3D-Modellen und unterstützt unter anderem die Dateiformate von CATIA, UG, Pro-E, STEP, Microstation, AutoCAD, 3D StudioMax, Alias und Maya.

EON Human ist eine Bibliothek von vordefinierten 3D Menschmodellen mit vordefinierten Bewegungsanimationen. Weiterhin ist eine Importfunktion für die Körperdaten realer Nutzer (Real People Format) implementiert.

Die Benutzeroberfläche von EON Studio ähnelt der von Virtools und Quest 3D. Animationen werden ebenfalls durch sogenannte Verhaltensblöcke (Behaviors) animiert und der Signalfluss erfolgt durch Verbindungslinien. Die Elemente sind, ebenfalls wie bei Virtools, nicht hierarchisch angeordnet. Bild C.3 zeigt exemplarisch eine Anordnung von Behaviors in EON Studio. (Maser, 2008)

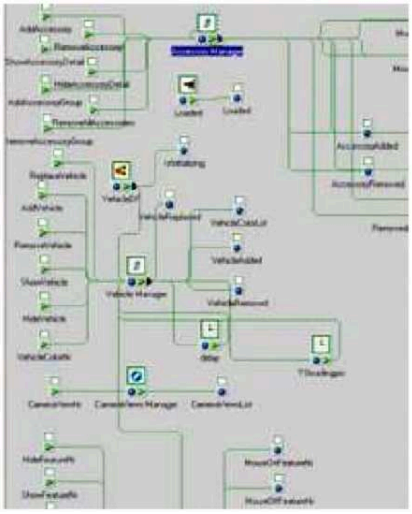


Bild C.3: Behaviours in EON Studio

Tabelle C.1 fasst die wichtigsten Produkte aus dem Bereich der VR Software übersichtlich zusammen, und fügt zu jedem Produkt eine Beschreibung der gegenwärtig vorhandenen Funktionalitäten hinzu. Dabei kann diese Tabelle als Ergänzung zu Tabelle 5.1 auf Seite 181 betrachtet werden.

Software	**Funktionalität**
Virtools/ Dassault	VR Pack, API Schnittstelle, Schematic Programmierung, VSL Skriptsprache, Integration in Dassault Produktportfolio über 3D XML, Plugins für verschiedene 3D Modeller
Quest 3D/ Act-3D	VR Hardwareunterstützung, API Schnittstelle, Schematic Programming, Plugins für verschiedene 3D CAD und Modeller
IDO:Software/ IC:IDO	VR Hardwareunterstützung, API Schnittstelle, Integration in den Konstruktionsprozess, Plugins für verschiedene 3D CAD und Modeller
CoVisE/ Visenso	Unterstützt 3D Eingabegeräte, OpenGL basierte API Schnittstelle, keine Integration in den Konstruktionsprozess
DeltaGEN/ RTT	VR Hardwareunterstützung, Modularer Aufbau, kommerzialisiert, integriert in den Konstruktionsprozess
dVise/ Division	Laufzeitumgebung dVS, eigene Programmiersprache, OpenGL basierte API Schnittstelle, keine Integration in den Konstruktionsprozess
EON Professional/ EON Reality	VR Hardwareunterstützung, Modular aufgebautes Softwarepaket, Behaviour Programming mit EON Studio, API Schnittstelle, Plugins für verschiedene 3D CAD und Modeller
WorldToolkit/ Sense8	Grafisches Entwicklungswerkzeug WorldUp, C-Programmierschnittstelle WorldToolKit, hohe Flexibilität
VR Juggler/ Infiscape	Portable Run-Time, C-Programmierschnittstelle nutzt OpenGL und OpenSG, plattformunabhängig, hohe Modularität, sehr umfangreich und komplex

Tabelle C.1: Querschnitt exemplarisch ausgewählter VR Software

ANHANG D

ShapeSheet Reference: Sections

1-D Endpoints Section Contains x- and y-coordinates of the begin point and endpoint of a 1-D shape. This section appears for 1-D shapes only.

Actions Section Contains rows that describe menu items on a shortcut or smart tag menu of a shape or page. The TagName cell is used to associate an action with a row in the Smart Tags section, where smart tags are defined.

Alignment Section Indicates the alignment of a shape with respect to the guide or guide point to which the shape is glued. The Alignment section appears only for shapes that are glued to guides.

Annotation Section Contains information about comments inserted into a document page. See Also Reviewer Section.

Character Section Shows the formatting attributes for the shape's text, such as font, color, text style, case, position relative to the baseline, and point size.

Connection Points Section Contains a row and cells for each connection point defined for the shape.

Controls Section Contains a row and cells for each control handle defined for the shape.

Delete a Section Select any cell in the section you want to delete, and then on the Edit menu, click Delete Section. Or right-click a cell, and then click Delete Section on the shortcut menu.

Document Properties Section Contains cells for a document that control preview quality, scope, and output format. You can also set these values using the Properties dialog box. Or, open the Drawing Explorer (View menu), right-click a drawing, and then click Properties on the shortcut menu.

Events Section Contains formulas that control shape events. Event cells are evaluated only when the event occurs, not upon formula entry.

Fill Format Section Shows the current fill formatting attributes for the shape and the shape's drop shadow, including pattern, foreground color, and background color. Fill formats can be set in the Fill dialog box (Format menu), by applying a fill style, or by making an entry in a Fill Format cell.

Foreign Image Info Section Contains the width and height of an object from another program used in a Microsoft Office Visio drawing and appears in the ShapeSheet window only for such objects. It also indicates the distance the object's image is offset within its borders.

Geometry Section Contains rows that list the coordinates of the vertices for the lines and arcs that make up the shape. If the shape has more than one path, there is a Geometry section for each path.

Glue Info Section Contains formulas generated for a 1-D shape when the 1-D shape is glued to other shapes.

Group Properties Section Contains cells for a group that control how you add shapes to a group, move members of a group, and select groups. You can also set these values on the Behavior tab in the Behavior dialog box (click Behavior on the Format menu).

Hyperlinks Section Contains cells for creating multiple jumps between a shape or drawing page and another drawing page, another file, or a Web site.

Image Properties Section Contains the gamma, brightness, contrast, blur, sharpen, and denoise values for bitmaps (an image that consists of pixels) and appears in the ShapeSheet window only for such objects.

Layer Membership Section Contains one row that lists each layer to which the shape is assigned.

Layers Section Shows all layers defined for the page and the properties set for each layer. This section is a page property that can be viewed only in the ShapeSheet window for a page. Its cells correspond to the options in the Layer Properties dialog box (click Layer Properties on the View menu).

Line Format Section Contains cells that control line attributes for a shape, such as pattern, weight, and color. They determine whether the line ends are formatted (for example, with an arrowhead), the size of line end formats, radius of the rounding circle applied to the line, and line cap style (round or square).

Miscellaneous Section Contains various attributes of shapes and groups, such as control selection highlighting and visibility.

Page Layout Section Contains cells that control the page settings for shapes and connectors, such as the spacing between all shapes on the page, spacing between all connectors on the page, and routing style for all connectors on the page.

Page Properties Section Contains cells that control page attributes, such as the page width, height, and scale.

Paragraph Section Shows the paragraph formatting attributes for the shape's text, such as indents, line spacing, bullets, and horizontal alignment of paragraphs.

Print Properties Section Contains the cells that control how the drawing page is formatted (appears) on the printer page.

Protection Section Shows the current lock settings set with the Protection command plus a few additional locks that can be set only in a ShapeSheet window.

Reviewer Section Contains identifying information about each document reviewer.

Ruler & Grid Section Shows the current settings of the page's rulers and grid.

Scratch Section A work area for entering and testing formulas that can be referred to by other cells.

Shape Data Section Contains one row for each item of shape data and cells for associating data with a shape.

Shape Layout Section Contains cells that control shape placement and connector routing settings.

Shape Transform Section Contains general positioning information about a shape: for example, its width, height, angle, and center of rotation (pin); whether the shape has been flipped; and how the shape should behave when resized within a group.

Show or Hide Sections

1. Click in the ShapeSheet window to make it active.
2. On the View menu, click Sections. Or right-click in the ShapeSheet window, and then click View Sections.
3. Select the check boxes for the sections you want to show, or clear the check boxes for the sections you want to hide.
4. Click OK.

Smart Tags Section Contains a row for each smart tag defined for a shape or page.

Style Properties Section Includes cells that control style behavior, such as whether a style includes text, line, and fill attributes.

Tabs Section Contains cells for shapes or styles that control tab stop position and alignment.

Text Block Format Section Contains cells that control the alignment, margins, and default tab stop position of text in a shape's text block.

Text Fields Section Displays functions and formulas inserted in the shape's text using the Field dialog box.

Text Transform Section Contains positioning information about a shape's text block.

User-defined Cells Section A work area for entering formulas in user-specific cells that can be referred to by other cells and add-on tools.

Quelle: Microsoft MSDN (Microsoft, 2008b)

ANHANG E

XSLT Befehlsreferenz

<xsl:apply-imports> wird in Verbindung mit dem importierten Stylesheet verwendet. Ist ein spezifisches Template des importierten Stylesheets überschrieben worden, kann der Prozessor mit dieser Instruktion trotzdem zur Ausführung dieses Templates gezwungen werden.

<xsl:apply-templates> selektiert ein Knotenset des Input-Dokuments und bearbeitet jeden einzelnen Knoten mit einer entsprechenen Templateregel. Der Knoten, der durch das spezifische Template bearbeitet wird, wird zum aktuellen Knoten des jeweiligen Templates. Kommen mehrere Templates in Frage, wird dasjenige mit der höchsten Priorität ausgewählt.

<xsl:attribute> innerhalb des Template-Bodys: erzeugt ein Attribut an die aktuelle Position des Output-Dokuments. Die Anweisung ist nur dann erfolgreich, wenn sie unmittelbar nach der Erzeugung eines Elements ausgeführt wird. Der Template-Body der Anweisung bestimmt den zugehörigen Wert des Attributs. Wird das Attribut öfters definiert, wird der Wert der letzten Definition übernommen.

<xsl:attribute-set> definiert einen Set, das aus einer Ansammlung von Attributen besteht, die mit Hilfe der Anweisung <xsl:attribute> festgelegt werden. Bereits erzeugte Attribut-Sets können mit Hilfe des Attributs [use-attribute-sets] verwendet werden. Ist ein Attribut mehrmals definiert worden, wird der Wert der letzten Definition übernommen. Einem beliebigen

Element können die Attribute mit Hilfe des Attributs [xsl:use-attribute-sets] zugewiesen werden. Die Instuktionen <xsl:element> und <xsl:copy> können das Set verwenden, um mehrere Attributknoten gleichzeitig an der aktuellen Position des Output-Dokuments zu erzeugen.

<xsl:call-template> ruft ein Template mit einem vorgegebenen Namen auf. Wird ein <xsl:with-param> Element verwendet und das zugehörige Template weist ein <xsl:param> Element auf, das den gleichen Namen hat, wird der zugehörige Wert des Parameters übergeben.

<xsl:choose> umschließt eine Anzahl von Alternativen, die mit Hilfe der <xsl:when> Instruktion definiert worden sind. Es wird diejenige ausgewählt und ausgeführt, deren Bedingung als erstes zutrifft. Nachfolgende Alternativen, deren Bedingungen auch zutreffen, werden ignoriert. Ist von allen Alternativen keine Bedingung zutreffend, wird die <otherwise> Alternative ausgeführt, sofern sie definiert worden ist.

<xsl:comment> schreibt einen Kommentar an die aktuelle Positon des Output-Dokuments. Es werden lediglich Textknoten generiert. Elemente, Attribute oder eingebettete Kommentare werden ignoriert. Der Kommentar darf nicht die Sequenz "–" beinhalten und nicht mit "–" abgeschlossen werden, da dies in einem XML-Kommentar nicht erlaubt ist.

<xsl:copy> kopiert den aktuellen Knoten des Input-Dokuments in die aktuelle Position des Output-Dokuments. Die Knoten und Attribute, die von dem zu kopierenden Knoten abhängig sind, werden ignoriert.

<xsl:copy-of> kopiert den Knoten und das abhängige Knotenset, mit den zugehörigen Namespaces und in der entsprechenden Reihenfolge innerhalb des Dokuments, an die aktuelle Position des Output-Dokuments.

<xsl:decimal-format> definiert Zeichen und Symbole, die bei der Konvertierung von Zahlen in Strings verwendet werden sollen. Die Umwandlung erfolgt durch die Funktion format-number(). Andere Funktionen sind durch die hier vorgenommenen Definitionen nicht betroffen.

<xsl:element> generiert einen Elementknoten an die aktuelle Position des Output-Dokuments.

<xsl:fallback> definiert einen Prozess, der ausgeführt werden soll, wenn der XSLT-Prozessor die Version des Stylesheets nicht unterstützt oder wenn ihm eine herstellerspezifische Funktion unbekannt ist.

<xsl:for-each> selektiert mit XPath ein Knotenset relativ zum aktuellen Knoten und wendet den Template-Body auf jeden einzelnen Knoten an. Sind <xsl:sort> Elemente definiert worden, werden die einzelnen Knoten des Sets in einer spezifischen Reihenfolge behandelt, ansonsten werden sie gemäß der Reihenfolge im Input-Dokument abgearbeitet.

<xsl:if> erlaubt die Ausführung des Template-Bodys nur dann, wenn die spezifische Bedingung zutrifft. Ist der Datentyp der Bedingung nicht boolesch, so wird er nach den Regeln der boolean() Funktion umgewandelt.

<xsl:import> sorgt für den Import eines weiteren Stylesheets in das ursprüngliche. Das importierte Modul besitzt allerdings eine geringere Importpriorität als das ursprüngliche Stylesheet.

<xsl:include> importiert ein zusätzliches Stylesheet an die Position des <xsl:include> Elements. Das neu eingebundene Stylesheet hat die gleiche Importpriorität, wie das ursprüngliche. Es handelt sich somit um eine reine textuelle Einbindung. Enthält das importierte Stylesheet relative URIs, beziehen sie sich auf den Basis-URI des HauptStylesheets. Es werden nur die im Haupt-Stylesheet definierten Namespaces benutzt, sofern die Definition in beiden Stylesheets vorkommt. Ist eine Template-Definition mehrmals vorhanden, wird in der Regel die letzte Definition bevorzugt.

<xsl:key> definiert einen Schlüssel, welcher mit der key() Funktion zugänglich ist und der ein Knotenset beinhalten kann. Desweiteren kann der Name eines Schlüssels mehrmals vorkommen. Importierte Schlüssel werden dem Haupt-Stylesheet hinzugefügt und besitzen keinen anderen Status. Das Element ist hilfreich, da der XSLT-Prozessor ein Index von Schlüsseln erstellt und damit die Suche nach bestimmten Elementen erleichtert.

<xsl:message> wird erzeugt, um während des Transformationsprozesses eine Meldung auszugeben. Sie wird in der Regel zur Fehlerbehandlung verwendet.

<xsl:namespace-alias> erlaubt, dass ein spezifischer Namespace, der im Stylesheet verwendet wird, einen anderen Namespace im Output-Dokument erhält. In der Regel wird dieses Element benutzt, um ein Stylesheet in ein anderes Stylesheet zu transformieren.

<xsl:number> kann eine Sequenzzahl, basierend auf der aktuellen Postion des Knotens, ermitteln und / oder eine Zahl für den Output formatieren. Die Berechnung der Sequenzzahl ergibt sich aus der spezifischen Level-Regel. Ist keine angegeben, wird als Level Single angenommen. Nachdem die Zahl ermittelt worden ist, wird sie durch die anderen Attribute formatiert und an die aktuelle Position des Output-Dokuments als String ausgegeben.

<xsl:otherwise> erlaubt die Ausführung des Template-Bodys nur dann, wenn keine der Bedingungen zutreffen, die innerhalb des <xsl:choose> Elements mit <xsl:when> definiert worden sind.

<xsl:output> definiert das Outputformat, d.h. nachdem das XSLT Stylesheet zunächst einen Output Baum erzeugt hat, kontrolliert das Element in welcher Form dieser in die Outputdatei geschrieben wird.

<xsl:param> definiert einen Parameter, der einen leeren String enthält, sofern ihm kein Wert zugewiesen worden ist. Falls das [select] Attribut vorhanden ist, darf kein Template-Body innerhalb des Elements auftauchen. Dann handelt es sich um einen Default-Wert, der mittels eines Template-Aufrufs, in Verbindung mit dem <xsl:with-param> Element, uberschrieben werden kann.

<xsl:preserve-space> definiert, wie auch das <xsl:strip-space> Element, die Art der Bearbeitung von Whitespace-Textknoten. Mit dem <xsl:strip-space> Element können diejenigen Knoten bezeichnet werden, deren Whitespaces gelöscht und mit dem <xsl:preserve-space> Element diejenigen, die

nicht gelöscht werden sollen. Werden weitere Stylesheets importiert, welche die gleichen Elemente behandeln, wird die Regel angewand, welche die höchste Importpriorität besitzt.

<xsl:processing-instruction> generiert einen Prozessinstruktionsknoten an die aktuelle Position des Output-Dokument

<xsl:sort> definiert innerhalb der Elemente <xsl:apply-templates> oder <xsl:for-each> in welcher Reihenfolge die Knoten bearbeitet werden. Es können mehrere Sortierungsschlüssel festgelegt werden, wobei die Elemente zunächst nach dem ersten, dann nach dem zweiten usw. sortiert werden.

<xsl:strip-space> definiert, wie auch das <xsl:preserve-space> Element, die Art der Bearbeitung von Whitespace-Textknoten. Whitespace Zeichen sind Leertaste, Tabulator, Carriage Return und Linefeed (#x20, #x9, #xD und #xA), die laut Grundeinstellung nichtignoriert werden. Mit diesem Element können diejenigen Knoten bezeichnet werden, deren Whitespaces gelöscht und mit dem <xsl:preserve-space> Element diejenigen, die nicht gelöscht werden sollen. Werden weitere Stylesheets importiert, welche die gleichen Elemente behandeln, wird die Regel angewand, welche die höchste Importpriorität besitzt.

<xsl:stylesheet> wird sowohl im Haupt-Stylesheet, als auch in den möglichen Import- oder Include-Stylesheets verwendet.

<xsl:template> definiert ein Template zur Generation des Outputs. Es wird entweder explizit über einen definierten Namen oder durch einen Vergleich mit einem Namen eines Knotens im Input-Dokument aufgerufen.

<xsl:text> erzeugt einen Textknoten an die aktuelle Position des Output-Dokuments.

<xsl:transform> wird sowohl im Haupt-Stylesheet, als auch in den möglichen Import- oder Include-Stylesheets verwendet.

<xsl:value-of> erzeugt einen String an die aktuelle Position des Output-Dokuments. Handelt es sich bei dem Ausdruck um einen Knotenset, wird nur der erste Knoten betrachtet. Attribute der einzelnen Knoten werden dabei ignoriert.

<xsl:variable> definiert eine Variable. Sofern das [select] Attribut vorhanden ist, darf innerhalb des Elements kein Template-Body erscheinen. Nach ihrer Definition innerhalb des Template-Bodys kann ihr Wert nicht mehr geändert werden. Handelt es sich um eine lokale Varibale, ist sie nur innerhalb des Template-Bodys erreichbar.

<xsl:when> erlaubt die Ausführung des Template-Bodys nur dann, wenn die Bedingung der Alternative zutrifft. Sind innerhalb des <xsl:choose> Elements mehrere Alternativen definiert worden, wird diejenige ausgewählt und ausgeführt, deren Bedingung als erstes zutrifft. Nachfolgende Alternativen, deren Bedingungen auch zutreffen, werden ignoriert. Ist der Datentyp der Bedingung nicht boolesch, so wird er nach den Regeln der boolean() Funktion umgewandelt.

<xsl:with-param> weist dem Parameter einen Wert zu, der innerhalb des aufgerufenen Templates verfügbar ist. Dies setzt allerdings voraus, dass der Parameter im aufgerufenen Template erneut mittels <xsl:param> definiert worden ist. Sollte er jedoch hier bereits einen Wert enthalten, wird dieser nun mit dem neuen Wert uberschrieben. Ist das [select] Attribut vorhanden, darf innerhalb des Elements kein Template-Body auftauchen.

Quelle: XSL Befehlreferenz (Schfer, 2008)

Verzeichnis der Abkürzungen

Abkürzung	Beschreibung
3DS	Proprietäres Autodesk Datenformat von 3D Studio
3Dc	3D Compressionstechnik
API	Application Programming Interface
AR	Augmented (erweiterte) Realität
ARIS	ARchitektur integrierter InformationsSysteme
B-REP	Boundary Representation
BB	BuildingBlock
BG	BehaviourGraph
BHV	Bounding Volume Hierarchie
Blobs	Binary Large OBjectS
BOOM	Binocular Omni-Orientation Monitor
BVH	BioVision-Format zur Übertragung von Animationen
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAS	Computer-Aided Styling
CASE	Computer-Aided Software Engineering
CAVE	Cave Automatic Virtual Environment
CAx	Computer-Aided Technologies
CPU	Central Processing Unit
CRM	Customer Relationship Management
CSCW	Computer-Supported Cooperative Work
CSG	Constructive Solid Geometry
CSV	Comma Seperated Value
DSS	Decision Support System
DXTC	DirectX Texture Compression
EIS	Executive Information System
EPK	Ereignisgesteuerte ProzessKette
ERD	Entity Relationship Diagram
ERP	Enterprise Ressource Planning
ETIM	ElektroTechnisches InformationsModell

Abkürzung	Beschreibung
FAST	Functional Analysis Systems Technique
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HMD	Head-Mounted Display
ICAM	Integrated Computer Aided Manufacturing
ICOM	Input, Control, Output, Mechanism
IDEF	Icam DEFinition language
IKT	Informations- und Kommunikations Technologien
IPC	Inter-Process Communication
IRVE	Information-Rich Virtual Environments
JT	Jupiter Tessilation
KÜS	Kraftfahrzeug-Überwachungsorganisation freiberuflicher Kfz-Sachverständiger
LoD	Level of Detail
MIS	Marketing Information System
MOM	Message Oriented Middleware
NIAM	Nijssens Information Analysis Method
NIST	National Institute of Standards and Technology
NURBS	Non-Uniform Rational B-Splines
ODBC	Open DataBase Connectivity
OBJ	Wavefront Dateiformat zur Geometrieübertragung
OMT	Object Modelling Technique
PDM	Product Data Management
PLM	Product Lifecycle Management
RAM	Random Access Memory
RDF	Ressource Description Framework
RS232	Recommended Standard 232
S3TC	S3 Texture Compression
SADT	Structures Analysis and Design Technique
SCM	Supply Chain Management
SDK	Software Development Kit
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
STEP	STandard for the Exchange of Product model data
STL	Standard Tessellated Language
U3D	Universal 3D
UML	Unified Modelling Language
VDX	Visio Drawing Xml
VIS	Visualisation
VR	Virtual Reality

Abkürzung	Beschreibung
VRAM	Video Random Access Memory
VRD	Virtual Retinal Displays
VRML	Virtual Reality Modeling Language
VSL	Virtools Scripting Language
W3C	World Wide Web Consortium
WiM	World in Miniature
WiW	Window in World
XML	eXtensible Markup Language
XSLT	extensible Stylesheet Language Transformation

Verzeichnis der Abbildungen

Verzeichnis der Tabellen

Verzeichnis der Stichworte

Verzeichnis der Autoren

Literaturverzeichnis

G.S. Altshuller. *Creativity as an Exact Science: The Theory of the Solution of Inventive Problems*. Gordon and Breach: New York, 1984. 13, 19, 24

M.M. Andreasen and L. Hein. *Integrated Product Development*. IFS (Publications) Ltd., Bedford ,Berlin, 1987. 13

L.B. Archer. The Inplifications for the Study of Design Methods of Recent Developments in Neighbouring Disciplins. In *Proceedings of IECD 85; Schriftenreihe WDK 1*. HEURISTA, Zrich, 1985. 13

Peter Astheimer, Klaus Böhm, Wolfgang Felger, Martin Göbel, and Stefan Müller. Die Virtuelle Umgebung - Eine neue Epoche in der Mensch-Maschine-Kommunikation, Teil I: Einordnung, Begriffe und Geräte. *Informatik Spektrum*, 17(5):281–290, 1994. 53, 56

Steve Aukstakalnis and David Blatner. *Silicon mirage*. Peachpit Press, 1992. ISBN 0-938151-82-7. 53, 61

Richard Bartle. *Designing virtual worlds*. Berkeley: New Riders, nachdruck edition, 2006. ISBN 0-1310-1816-7. 84

Christian Bauer. *Nutzenorientierter Einsatz von Virtual Reality im Unternehmen*. Computerwoche-Verl., 1996. ISBN 3-930377-21-7. 67

Benjamin B. Bederson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21(4):833–854, 2002. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/571647.571649. 47

Kirsty A. Beilharz and Rabee M. Reffat. Thematic design and efficiency in virtual environments using metaphors. In *MMM*, page 371, 2004. 67

Steve Benford, Dave Snowdon, Chris Greenhalgh, Rob Ingram, Ian Knox, and Chris Brown. VR-VIBE: A Virtual Environment for Co-operative Information Retrieval. *Computer Graphics Forum*, 14(3):349–360, 1995. 48

Tomas Michael Benz. *Funktionsmodellieren als Basis zur Lösungsfindung in CAD-Systemen*. Shaker, 1990. 16

Jörg Bergbauer. *Entwicklung eines Systems zur interaktiven Simulation von Produktionssystemen in einer virtuellen Umgebung*. Aachen: Shaker, 2002. ISBN 3-8322-0553-5. 53, 56, 59

Mark Billinghurst. Put that where? voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 32(4):60–63, 1998. ISSN 0097-8930. doi: http://doi.acm.org/10.1145/307710.307730. 73

Oliver Bimber and Ramesh Raskar. Modern approaches to augmented reality. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 1, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1185657.1185796. 63

Frank Biocca. The Cyborg's Dilemma: Progressive Embodiment in Virtual Environments. *J. Computer-Mediated Communication*, 3(2), 1997. 59, 61

Chuck Blanchard, Scott Burgess, Young Harvill, Jaron Lanier, Ann Lasko, Mark Oberman, and Mike Teitel. Reality built for two: a virtual reality tool. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 35–36, New York, NY, USA, 1990. ACM. ISBN 0-89791-351-5. doi: http://doi.acm.org/10.1145/91385.91409. 57

James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/357306.357310. 30

Arbeitsgruppe BMBF. Nationaler IT-Gipfel. Bekanntmachung, BMBF - Hasso Plattner Institut, DEC 2006. Arbeitsgruppe Hightech-Strategie fr die Informationsgesellschaft. 1, 2

Joan De Boeck, Tom De Weyer, Chris Raymaekers, and Karin Coninx. Using the non-dominant hand for selection in 3d. In *3DUI '06: Proceedings of the 3D User Interfaces (3DUI'06)*, pages 53–58, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 1-4244-0225-5. doi: http://dx.doi.org/10.1109/VR.2006.140. 69, 70

Frank Bongers. *XSLT 2.0*. Galileo Press, 1. aufl. edition, 2004. ISBN 3-89842-361-1. 37

Sven Bormann. *Virtuelle Realität*. Addison-Wesley, 1. aufl. edition, 1994. ISBN 3-89319-707-9. 61, 63, 64

D. Bowman and L. Hodges. Toolsets for the Development of Highly Interactive and Information-Rich Virtual Environments. *International Journal of Virtual Reality*, 3(2):12–20, 1997. 73, 74

Doug A. Bowman, Chris North, Jian Chen, Nicholas F. Polys, Pardha S. Pyla, and Umur Yilmaz. Information-rich virtual environments: theory, tools, and research agenda. In *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 81–90, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-569-6. doi: http://doi.acm.org/10.1145/1008653.1008669. 83

Matthias Brenzinger. Aufbereitung und Visualisierung eines 3D CAD Modells in einer VR Umgebung am Beispiel eines Fahrersitzes. Studienarbeit, Universität Karlsruhe - IMI, July 2007. Betreut durch Hardy Krappe. 62, 63, 65, 88, 158, 190

Brockhaus. *Brockhaus Enzyklopädie in 30 Bänden*. F.A. Brockhaus GmbH, Leipzig, Bibliografisches Institut und F.A. Brockhaus AG, Mannheim, 21 edition, 2006. 38, 39, 56, 90

Frederick P. Brooks, Jr. Three great challenges for half-century-old computer science. *J. ACM*, 50(1):25–26, 2003. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/602382.602397. 87

Manfred Broy, editor. *VDI-Lexikon Informatik and Kommunikationstechnik*. Springer, 2., erw. and neubearb. aufl. edition, 1999. ISBN 3-540-63249-2. 15

M. Bruls, K. Huizing, and J. van Wijk. Squarified treemaps, 2000. URL `citeseer.ist.psu.edu/bruls99squarified.html`. 47

Steve Bryson. Virtual reality in scientific visualization. *Commun. ACM*, 39(5): 62–71, 1996. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/229459.229467. 74

Tim Bücker. Real-Time Rendering - Semesterarbeit. Technical report, Universität Koblenz-Landau, Koblenz, September 2003. 4, 105

Hans-Joachim Bungartz, Michael Griebel, and Christoph Zenger. *Introduction to computer graphics*. Charles River Media, 2. ed. edition, 2004. ISBN 1-58450-332-7. 27, 31

Grigore Burdea and Philippe Coiffet. *Virtual reality technology*. Wiley, 1994. ISBN 0-471-08632-0, 2-866601-386-f7. 52, 54, 55, 57, 61

Marshall Burns. Automated Fabrication. retrieved on the World Wide Web: http://www.ennex.com/ fabbers/StL.asp, 1989. 113

Stuart K. Card, George G. Robertson, and William York. The webbook and the web forager: an information workspace for the world-wide web. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 111–ff., New York, NY, USA, 1996. ACM. ISBN 0-89791-777-4. doi: http://doi.acm.org/10.1145/238386.238446. 41, 44

A. Chakrabarti, P. Langdon, Liu Y.-C., and P. Bligh. *An approach to compositional synthesis of mechanical design concepts using computers*. London Berlin: Springer, 2002. 25

B. Chandrasekaran, A.K. Goel, and Y. Iwasaki. Functional Representation as Design Rational. *IEEE Computer*, 1, 1993. 16, 25

H.J. Charwat. *Lexikon der Mensch-Maschine-Kommunikation*. Oldenbourgh, 1994. ISBN 3-486-22618-5. 38

Peter P. Chen and Heinz-Dieter Knöll. *Der Entity-Relationship-Ansatz zum logischen Systementwurf*. BI-Wiss.-Verl., 1991. ISBN 3-411-15311-3. 24

Charles Clark. *Brainstorming*. Verl. Moderne Industrie, gek. taschenbuchausg., 3. aufl. edition, 1973. 171

classmate. Simus Classmate Produktportfolio. retrieved from the World Wide Web: http://www.simus-systems.com/. 126

ComputerBase. Konstruktive Festkörpergeometrie. retrieved on the World Wide Web: http://www.computerbase.de/lexikon/Constructive28

Klaus-Jörg Conrad. *Grandlagen der Konstruktionslehre*. Hanser, 1998. ISBN 3-446-19467-3. 10

Nigel Cross. *Engineering Design Method*. J. Wiley and Suns Ltd., Chichester, 1985. 13

IBM Dassault Systèmes. 3D XML, a leightweight xml-based format. retrieved on the World Wide Web: http://www.3ds.com/de/products-solutions/3dvia/3d-xml/. 114

Rolf Dässler. Informationsvisualisierung - Stand, Kritik and Perspektiven. Technical report, FH Potsdam, 1999. 41, 42, 51

Rolf Dässler and Hartmut Palm. *Virtuelle Informationsräume mit VRML*. Dpunkt Verl., 1. aufl. edition, 1998. ISBN 3-920993-78-0. 38, 39

Michael F. Deering. The holosketch vr sketching system. *Commun. ACM*, 39 (5):54–61, 1996. 66

Stephan Diehl, editor. *Software visualization*, volume 2269 of *Lecture Notes in Computer Science*. Springer, 2002. 82

DIN-199-1. Cad-modelle, zeichnungen und stücklisten, teil 1: Begriffe. Technical report, Deutsches Institut für Normung, 2002. Technische Produktdokumentation. 133

Dive. The Distributed Interactive Virtual Environment Project, 1991. URL `http://www.sics.se/dive/dive.html`. 48

J.R. Dixon. *On Research Methodology Towards A Scientific Theory of Engineering Design*. Design Theory 88. Springer, New York, 1988. 13

Marek Doniec. Funktionsweise und Einsatzmöglichkeiten von Virtual Retinal Displays. Seminararbeit, Universität Karlsruhe, November 2003. 63

Dudenredaktion. *Duden, die deutsche Rechtschreibung: auf der Grundlage der neuen amtlichen Rechtschreibregeln*. Dudenverl., Mannheim [u.a.], 24., völlig neu bearb. und erw. aufl. edition, 2006. ISBN 3-411-04014-9. 80

N.I. Durlach and A.S. Mavor. *Virtual Reality: Sientific and Technological Challenges*. National Research Concil. Washington, DC: National Academy Press, 1995. 52, 59

eClass. eClass - Das führende Klassifikationssystem. retrieved from the World Wide Web: http://www.eclass.de/index.html. 126

Klaus Ehrlenspiel. *Integrierte Produktentwicklung*. Hanser, 2., überarb. aufl. edition, 2003. ISBN 3-446-22119-0. 10, 17, 170

Niklas Elmqvist, Ulf Assarsson, and Philippas Tsigas. Employing Dynamic Transparency for 3D Occlusion Management: Design Issues and Evaluation. In *INTERACT (1)*, pages 532–545, 2007. 72

José L. Encarnação and Wolfgang Felger. International activities and future perspectives of virtual reality. In *Computer Graphics International*, pages 9–, 1996. 53, 54

José L. Encarnação, Wolfgang Strasser, and Reinhard Klein. *Graphische Datenverarbeitung*. Oldenbourg, 1995. 29

H. Englberger. Computergestütze informationsvisualisierung. Diplomarbeit, Universität München - Fachbereich Informatik, September 1995. 37, 41, 44, 47, 50

Etim. ETIM 4.0 und neuer Leitfaden. retrieved from the World Wide Web: http://www.etim.de/Home.1.0.html. 126

Dieter Eulenbach. *Konstruktionssystem WISKON*. VDI-Verl., als ms. gedr. edition, 1990. ISBN 3-18-143220-2. 25

H. Falter, M. Rtting, and J. Springer. Zur Problematik der Verwendung von Displays zur dreidimensionalen Wahrnehmung. *Zeitschrift für Arbeitswissenschaft*, 47(4):198–205, 1993. 63

Jean-Daniel Fekete and Catherine Plaisant. Interactive information visualization of a million items. In *INFOVIS '02: Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, page 117, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1751-X. 40, 42

FileFormat. FileFormat.info - Informationen zum 3DS Dateiformat. retrieved on the World Wide Web: http://www.fileformat.info/format/3ds. 113

Andrew Forsberg, Kenneth Herndon, and Robert Zeleznik. Aperture based selection for immersive virtual environments. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 95–96, New York, NY, USA, 1996. ACM. ISBN 0-89791-798-7. doi: http://doi.acm.org/10.1145/237091.237105. 70

Theodore C. Fowler. *Value Analysis in Design*. Van Nostrand Reinhold, 1981. 21, 23, 120

G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM. ISBN 0-89791-180-6. doi: http://doi.acm.org/10.1145/22627.22342. 44, 45

Jürgen Gausemeier, Peter Ebbesmeyer, and Ferdinand Kallmeyer. *Produktinnovation*. Hanser, 2001. ISBN 3-446-21631-6. 53

Matthias Gebauer. *Kooperative Produktentwicklung auf der Basis verteilter Anforderungen*. Shaker, 2001. ISBN 3-8265-9200-X. 9, 10, 12, 15

Kerstin Geiger. *Neue Wege zur Produktentwicklung*. RAABE, Stuttgart, 1997. ISBN 3-88649-347-4. Herausgegeben von Hans Grabowski. 2

Franz-Josef Gierse. *Begleittext zum Seminar Einsatz der Wertanalyse im Konstruktionsprozess beim Niederrheinischen Bezirksverein des VDI*. Düsseldorf: VDI-Verlag, 1984. 16

Hans Grabowski. Rechnerunterstütztes Konstruieren and Erstellen von Fertigungsunterlagen I/II. Technical report, RPK, Karlsruhe, March 2000. 26, 27, 29

John Shapley Gray. *Interprocess communications in UNIX*. Prentice Hall PTR, 1997. ISBN 0-13-186891-8. 34

S.A. Gregory. *Creativity in Engineering*. Butterworth, London, 1970. 13

Wulf R. Halbach. *Interfaces. Medien- und Kommunikationstheoretische Elemente einer Interface-Theorie*. Fink, München, 1994. ISBN ISBN 3-7705-2934-0. 33, 82

Joan O'C. Hamilton, Emily T. Smith, Gary McWilliams, Evan I. Schwartz, and John Carey. Virtual Reality - How a Computer generated World could Change the Real World. *BusinessWeek*, 38(3), 1997. 52

Elliotte Rusty Harold. *Die XML-Bibel*. mitp-Verl., 2. und aktualis. aufl. edition, 2002. ISBN 3-8266-0821-6. 35, 36

S.L. Havre, A. Shah, C. Posse, and B.M. Webb-Robertson. Diverse Information Integration and Visualization. In *In Visualization and Data Analysis 2006 (EI10). SPIE The International Society for Optical Engineering, San Jose, CA*, 2006. 47

Christopher G. Healey, Kellogg S. Booth, and James T. Enns. Visualizing real-time multivariate data using preattentive processing. *ACM Trans. Model. Comput. Simul.*, 5(3):190–221, 1995. ISSN 1049-3301. doi: http://doi.acm.org/10.1145/217853.217855. 42

P. Heckel. Conceptual models and metaphor in software design. In *Compcon Spring '91. Digest of Papers*, pages 498–499, 1991. ISBN 0-8186-2134-6. 67

Jack Heflin, Anita Komolodi, Nakul Pasricha, and Theen-Theen Tan. WebTOC: Evaluation of a hierarchical browsing interface for the World Wide Web. Retrieved on the World Wide Web: http://www.otal.umd.edu/SHORE/bs11/, 1998. 44

Michael Heim. *The metaphysics of virtual reality*. Oxford Univ. Press, 1993. ISBN 0-19-508178-1. 54

Matthias Hemmje, Clemens Kunkel, and Alexander Willett. Lyberworld - a visualization user interface supporting fulltext retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 249–259, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X. 48

Alexander Henning. *Die andere Wirklichkeit*. Addison-Wesley, 1. aufl. edition, 1997. ISBN 3-8273-1102-0. 52, 54, 87

Julie Hirtz, Robert B. Stone, Daniel A. McAdams, Simon Szykman, and Kristin L. Wood. A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts. *Research in Engineering Design*, 13(2):65–82, 2002. doi: 10.1007/s00163-001-0008-3. 19, 21, 128, 130, 131

T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. Exploration of full-text databases with self-organizing maps. In *Proceedings of the ICNN96, International Conference on Neural Networks*, volume I, pages 56–61, Piscataway, NJ, 1996. IEEE Service Center. URL `citeseer.ist.psu.edu/honkela96exploration.html`. 46

Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise Smooth Surface Reconstruction. *Computer Graphics*, 28(Annual Conference Series):295–302, 1994. 29

Mei Huang. *Funktionsmodellierung and Lösungsfindung mechatronischer Produkte*. Shaker, 2002. ISBN 3-8322-0059-2. 16, 19

Markus Huber. Sprachen zur 3D-Visualisierung im Web. Proseminar, Universitt Tübingen, September 2002. 100

Ralf Huber. *Wissensbasierte Funktionsmodellierung als Grundlage zur Gestaltsfindung in Konstruktionssystemen*. Shaker, als ms. gedruckt edition, 1994. ISBN 3-86111-798-3. 16, 18

V. Hubka and W. E. Eder. Theory of technical systems: A total concept theory for engineering design (2nd revised and enlarged edition). *Shock and Vibration*, 1988. 10, 11

M. S. Hundal. A Systematic Method for Developing Function Structures, Solutions and Concept Variants. *Mech. Mach. Theory*, 25(3):243–256, 1990. 19

Andrew Hutchison. Back to the holodeck: new life for virtual reality? In *DIMEA '07: Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, pages 98–104, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-708-7. doi: http://doi.acm.org/10.1145/1306813.1306838. 56

L. Ibanez, C. Hamitouche, and C. Roux. Structured deformable models application of metaballs to 3D medicalimage segmentation. In *Engineering in Medicine and Biology Society*, 2000. ISBN 0-7803-6465-1. 30

Rob Ingram and Steve Benford. Legibility enhancement for information visualisation. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 209, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7187-4. 48

C. Irgens. Unterstützung der qualitätsgerechten Konstruktion. *Erfolgreiche Anwendung wissensbasierter Systeme in Entwicklung and Konstruktion*, 1991. 25

Bernhard Jenny. 3D Grafik - Vorlesungsbegleitende Unterlagen. Technical report, ETH Zürich, Zürich, March 2007. 105

Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press. ISBN 0-8186-2245-8. 47

Roy S. Kalawsky. *The science of virtual reality and virtual environments*. Addison-Wesley, 1. print. edition, 1993. ISBN 0-201-63171-7. 56

Daniel A. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002. ISSN 1077-2626. doi: http://dx.doi.org/10.1109/2945.981847. 41, 51, 78

Daniel A. Keim, Jörn Schneidewind, and Mike Sips. Circleview: a new approach for visualizing time-related multidimensional data sets. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 179–182, New York, NY, USA, 2004. ACM. ISBN 1-58113-867-9. doi: http://doi.acm.org/10.1145/989863.989891. 47

Roland Kläger. *Modellierung von Produktanforderungen als Basis für Problemlösungsprozesse in intelligenten Konstruktionssystemen*. Shaker, 1993. ISBN 3-86111-622-7. 11, 12

Simon Knebel. Simulation von Schwarmverhalten autonomer Objekte. Studienarbeit, Universitt Koblenz Landau, September 2003. 100, 111

Eugenia M. Kolasinski. Simulator Sickness in Virtual Environments. Final technical report 68, Army research institute for the behavioral and social sciences, Alexandria VA, May 1995. Accession Number: ADA295861. 85

Rudolf Koller. *Konstruktionslehre für den Maschinenbau*. Springer, 4., neubearb. and erw. aufl. edition, 1998. ISBN 3-540-63037-6. 10, 16, 128, 170

Hardy Krappe. Entscheidungsfindung für VR Investitionen. *CAD-CAM Report*, 24(10):46–49, 2005. 89

Hardy Krappe and Milan Marinov. Funktionsmodellierung in der virtuellen Produktentwicklung. *CAD-CAM Report*, 26(3):24–27, 2007. 18, 81, 117

Hardy Krappe, Milan Marinov, and Jivka Ovtcharova. Anwendungen der Funktionsmodellierung. *CAD-CAM Report*, 26(6):84–87, 2007. 19

Myron W. Krueger. *Artificial reality*. Addison-Wesley, 1983. ISBN 0-201-04765-9. 52

Peter Krumhauer. *Rechnerunterstützung für die Konzeptphase der Konstruktion*. Springer, 1974. 128

Rainer Kuhlen. *Informationsmarkt*. UVK, Univ.-Verl., 1995. ISBN 3-87940-528-X,3-87940-529-8. 39

Kunstuniversity Linz. Grundlagen der 3D Modellierung. retrieved on the World Wide Web: http://www.dma.ufg.ac.at/app/link/Grandlagen:3D-Grafik/. 26, 32

Harald Kunze. *Ein Beitrag zur Gewinnung von Funktionen für mechanische Produkte aus 3D-CAD-Gestaltmodelldaten*. Shaker, 2002. ISBN 3-8322-0766-X. 11, 12, 17, 18, 25, 27, 31

Ray Kurzweil. The Law of Accelerating Returns. *KurzweilAI.net*, March 2001. URL `http://www.kurzweilai.net/articles/art0134.html?printable=1`. 89

Detlef Kuttig. *Rechnerunterstützte Funktions- and Wirkstrukturverarbeitung beim Konzipieren*. TU, Univ.-Bibliothek, Abt. Publ., 1993. ISBN 3-7983-1545-0. 25

John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: http://doi.acm.org/10.1145/223904.223956. 44

Gerd Langlotz. *Ein Beitrag zur Funktionsstrukturentwicklung innovativer Produkte*. Shaker, 2000. ISBN 3-8265-7246-7. 2, 16, 21, 128, 129

Hoffman LaRoche, editor. *Roche-Lexikon Medizin*. Urban and Schwarzenberg, 4., neubearb. u. erw. aufl. edition, 1999. ISBN 3-541-17134-0,3-541-11214-X,3-541-17114-6. 85, 96

Brenda Laurel. *The art of human-computer interface design*. Addison-Wesley, 4th print. edition, 1992. ISBN 0-201-51797-3. 4

Joseph J. LaViola, Jr. A discussion of cybersickness in virtual environments. *SIGCHI Bull.*, 32(1):47–56, 2000. ISSN 0736-6906. doi: http://doi.acm.org/10.1145/333329.333344. 85

Helen Leemhuis. *Funktionsgetriebene Konstruktion als Grundlage verbesserter Produktentwicklung*. PhD thesis, Berlin, Techn. Univ., Diss., 2005. 3, 14, 16, 24, 25

Stefan Leinenbach. *Interaktive Geschäftsprozessmodellierung*. Dt. Univ.-Verl., 1. aufl. edition, 2000. ISBN 3-8244-9042-0. 54, 59, 61, 64, 75

J. Leston, K. Ring, and E. Kyral. *Virtual Reality - Business Applications, Markets and Opportunities*. Ovum Reports, Ovum Ltd., 1996. 52, 53, 56, 57, 65, 66, 67

Vladimir I. Levenshtein. *Binary codes capable of correcting deletions, insertions, and reversals*. 10(8) edition, 1966. 146

X. Lin. Visual Sitemap - A SOM application to textual information visualization. In *Proceedings of WSOM'97, first International Conference on Kohonen's Self-organizing Algorithm*, June 1997. 47

Ralf-Stefan Lossack. *Konstruktionsarbeitsräume als Grundlage zur integrierten Produktmodellierung*. Shaker, als ms. gedr. edition, 1997. ISBN 3-8265-2565-5. 18

Bernd Lutz. *Konzepte für den Einsatz von virtueller and erweiterter Realität zur interaktiven Wissensvermittlung*. PhD thesis, Darmstadt, Techn. Univ., Diss., 2004. 86

Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: detail and context smoothly integrated. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176, New York, NY, USA, 1991. ACM. ISBN 0-89791-383-3. doi: http://doi.acm.org/10.1145/108844.108870. 41, 46

Sal Mangano. *XSLT Cookbook*. O'Reilly, 2. ed. edition, 2006. ISBN 0-596-00974-7. 37

Katerina Mania, Stephen R. Ellis, Mark Billinghurst, and Anthony Steed. Tutorial 1: Usability evaluation techniques for virtual reality technologies. In *VR*, pages 299–, 2002. 63

David A. Marca and Clement L. MacGowan. *SADT*. McGraw-Hill, 2nd print. edition, 1988. ISBN 0-07-040235-3. 21

Ren Martin. *Microsoft Visio 2007 Programmierung*. Hanser, 2007. ISBN 978-3-446-41084-8, 3-446-41084-8. 179

Christoph Maser. Abbildung des Verhaltens von Komponenten eines Fahrzeugsitzes in einem virtuellen Funktionsmodell. Diplomarbeit, Universität Karlsruhe - IMI, February 2008. Betreut durch Hardy Krappe. 161, 162, 172, 183, 286, 287, 288

Günter Matthiessen and Michael Unterstein. *Relationale Datenbanken und SQL*. Addison-Wesley, [3. aktualis. aufl.] edition, 2003. ISBN 3-8273-2085-2. 34

Oliver Mayer. Entwicklung einer Methode zur Datenübertragung von Modellen aus einem CAS- in ein CAD-System anhand unterschiedlicher Designstudien am Beispiel eines Fahrersitzes. Diplomarbeit, Universität Karlsruhe - IMI, September 2006. Betreut durch Hardy Krappe. 190

Arno Michelis. *Aufbau von integrierten Lösungsspeichern für die Produktentwicklung im Maschinenbau*. Shaker, 2002. ISBN 3-8322-0085-1. 11

Microsoft. Getting started developing shapes in visio 2003 shapestudio. Technical report, Microsoft, 2008a. 178

Microsoft. Msdn library: Microsoft office visio 2003 sdk documentation. Technical report, Microsoft, 2008b. 179, 295

Mark R. Mine. ISAAC: A Virtual Environment Tool for the Interactive Construction of Virtual Worlds. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995. 70

Stefan Mintert. Extensible markup language (xml) 1.1, w3c recommendation 04. february 2004. Technical report, W3C, 2004. 35

Kenneth Mitchell and Jessie Kennedy. The Perspective Tunnel: An Inside View on Smoothly Integrating Detail and Context. In *In 8th EG Workshop on ViSC, Boulogne sur Mer*, 1997. 44

Tamara Munzner and Paul Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. In *VRML '95: Proceedings of the first symposium on Virtual reality modeling language*, pages 33–38, New York, NY, USA, 1995. ACM. ISBN 0-89791-818-5. doi: http://doi.acm.org/10.1145/217306.217311. 45

G. Nadler. *The Planning and Design Approach*. Wiley, New York, 1981. 13

Martha Najarro. *Visualisierung von Informationsräumen.* PhD thesis, Ilmenau, Techn. Univ., Diss., 2003. 38

nDimension. Multidimensionale Navigationssysteme in Hypermedien. Retrieved on the World Wide Web: http://syntheti.cc/ndimensions/index.html, 2007. 44

NIST. Integration definition for information modeling (IDEF0). Technical report, Computer Systems Laboratory, National Institute of Standards and Technology, 1993. 21

W.M. Odrin. *Morphologische Synthese von Systemen: Aufgabenstellung Klassifikation, Morphologische Suchmethoden.* Institut für Kybernetik, Kiew, 1986. 13

Rory O'Neil and Eden Muir. Information corridor. Technical report, Columbia State University, 1997. 49

B. Ostrofsky. *Design, Planning and Development Methodology.* New Jersey: Prentice-Hall Inc., 1977. 13

Jivka Ovtcharova. Virtual Engineering I/II - Vorlesungsbegleitendes Skriptum des Sommersemesters 2006. Technical report, IMI, Karlsruhe, March 2006a. URL `http://www.imi.uni-karlsruhe.de/289.php`. 60, 85, 86, 134

Jivka Ovtcharova. PLM, Product Lifecycle Management - Vorlesungsbegleitendes Skriptum des Sommersemesters 2006. Technical report, IMI, Karlsruhe, March 2006b. URL `http://www.imi.uni-karlsruhe.de/275.php`. 79, 80

Gerhard Pahl. *Konstruktionslehre.* Springer, 7. aufl. edition, 2007. ISBN 3-540-34060-2. 10, 17, 18, 19, 94, 170

Jeffrey S. Pierce, Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. Image plane interaction techniques in 3d immersive environments. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 39–ff., New York, NY, USA, 1997. ACM. ISBN 0-89791-884-3. doi: http://doi.acm.org/10.1145/253284.253303. 69

U. Pighine. Methodological Design of Machin Elements. In Zrich Heurista, editor, *Proceedings of ICED 90; Schriftenreihe 19*, 1990. 13

Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *UIST '96: Proceedings of the 9th annual ACM symposium on User*

interface software and technology, pages 79–80, New York, NY, USA, 1996. ACM. ISBN 0-89791-798-7. doi: http://doi.acm.org/10.1145/237091.237102. 71

proficlass. Proficlass - Wir beschreiben Standards. retrieved from the World Wide Web: http://www.proficlass.de/. 126

Verein ProSTEP/iViP. STEP Definition des ProSTEP iViP Vereins. retrieved on the World Wide Web: http://www.prostep.org/de/glossar/s.htm. 113

Erik T. Ray. *Learning XML*. O'Reilly, 2. ed. edition, 2003. ISBN 0-596-00420-6, 978-0-596-00420-0. 35

G. Reinhold. *Reinhold Soziologie-Lexikon - Interaktion*. G. Reinhold & S. Lamnek (Hrsg.), München: Oldenbourg, 1997. 55

Jun Rekimoto and Mark Green. The information cube: Using transparency in 3d information visualization. In *In Proceedings of the Third Annual Workshop on Information Technologies and Systems (WITS93)*, pages 125–132, 1993. URL `citeseer.ist.psu.edu/rekimoto93information.html`. 49

Howard L. Resnikoff. *The illusion of reality*. Springer, 1989. ISBN 3-540-96398-7,0-387-96398-7. 46

Martin Richter. *Eine Plattform zur Visualisierung von einander abhängigen Produktdaten im Produktlebenslauf*. PhD thesis, Karlsruhe, Techn. Univ., Diss., November 2005. 67, 90

George G. Robertson and Jock D. Mackinlay. The document lens. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 101–108, New York, NY, USA, 1993. ACM. ISBN 0-89791-628-X. doi: http://doi.acm.org/10.1145/168642.168652. 41, 42, 50

George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 189–194, New York, NY, USA, 1991. ACM. ISBN 0-89791-383-3. doi: http://doi.acm.org/10.1145/108844.108883. 41, 46

Wolf G. Rodenacker. *Methodisches Konstruieren*. Springer, 4., überarb. aufl. edition, 1991. ISBN 3-540-53977-8, 0-387-53977-8. 10, 128

Stefan Roettger, Wolfgang Heidrich, Philip Slusallek, and Hans-Peter Seidel. Real-time generation of continuous levels of detail for height fields. In *Procceedings of WSCG '98*, pages 315–322, 1998. 108

Karl-Heinz Roth. *Konstruktionslehre*. Springer, 3. aufl., erw. and neu gestaltet. edition, 2000. ISBN 3-540-67142-0. 10, 12, 16, 17, 18, 19, 120, 128, 170

Stefan Rude. *Rechnerunterstützte Gestaltfindung auf der Basis eines integrierten Produktmodells*. VDI-Verl., als ms. gedr. edition, 1991. ISBN 3-18-145220-3. 11, 12

Stefan Rude. *Wissensbasiertes Konstruieren*. Shaker, als ms. gedr. edition, 1998. ISBN 3-8265-3985-0. 16, 17, 19

Ingrid Rügge. Alternativen zum Desktop: Virtual, Augmented and Real Reality. Fachaufsatz, Universität Bremem, July 1999. 4, 52

James Rumbaugh and Doris Märtin. *Objektorientiertes Modellieren and Entwerfen*. Studienzentrum für Sehgeschädigte, 1994. 21, 23

Andreas Rutz. *Konstruieren als gedanklicher Prozess*. PhD thesis, Technische Universität München, 1985. 10

Hermann Sauer. *Relationale Datenbanken - Theorie und Praxis*. Addison-Wesley, 5., aktualis. u. erw. aufl. edition, 2002. ISBN 3-8273-2060-7. 34

August-Wilhelm Scheer. *Architektur integrierter Informationssysteme*. Springer, 2., verb. aufl. edition, 1992. ISBN 3-540-55401-7, 0-387-55401-7. 21, 22, 89

Martin Scherer. Einarbeitung von inhaltlichen and sichtenorientierten Zusatzinformationen in eine bestehende VR Umgebung am Beispiel eines Fahrzeugsitzes. Studienarbeit, Universität Karlsruhe - IMI, February 2007. Betreut durch Hardy Krappe. 41, 46, 50, 63, 71, 72, 73, 74, 83

Jan Scheurich. VRML/X3D unter Unix/Linux: 3D-Welten im WWW. retrieved on the world wide web: http://vrml.cip.ica.uni-stuttgart.de/linuxtag/index.html, 2007. 65

Timo Schfer. XSL Befehlsreferenz. Retrieved on the World Wide Web: http://xml.cnec.org/xsl, 2008. 302

Karsten Schmidt. *Methodik zur integrierten Grobplanung von Abläufen und Strukturen mit digitalen Fabrikmodellen*. PhD thesis, RWTH Aachen, 2002. 118

Jörg Schmittler, Daniel Pohl, Tim Dahmen, Christian Vogelgsang, and Philipp Slusallek. Realtime ray tracing for current and future games. In *GI Jahrestagung (1)*, pages 149–153, 2004. 59

Peter Schubert. Systematischer Entwurf eines kostenoptimierten elektrischen Kraftstoffpumpenantriebkonzepts. Diplomarbeit, Universität Karlsruhe - IMI, July 2007. Betreut durch Milan Marinov. 20, 21

Hans-Joachim Schulz. *Experimentalphysik für Ingenieure*. Vieweg, 1996. ISBN 3-528-04934-0. 19

H. Schumann and W. Müller. *Informationsvisualisierung: Methoden and Perspektiven*. Oldenbourg Verlag, 2004. 40, 51

Johannes Schwall. 3D-Interaktion. Arbeit zum projektseminar interaktive 3d-stadtplanung, Westfälische Wilhelms-Universität Münster, 2004. 67, 69, 71

Wenzel Seiler. *Technische Modellierungs- and Kommunikationsverfahren für das Konzipieren and Gestalten auf der Basis der Modell-Integration*. VDI-Verlag, als ms. gedr. edition, 1985. ISBN 3-18-144910-5. 11

Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7508-X. 40, 83

Ben Shneiderman and Martin Wattenberg. Ordered Treemap Layouts. In *INFOVIS '01: Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, page 73, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1342-5. 47

John E. Simpson. *XPath and XPointer*. O'Reilly, 2002. ISBN 0-596-00291-2. 37

Mel Slater, Anthony Steed, and Yiorgos Chrysanthou. *Computer Graphics and Virtual Environments: From Realism to Real - Time*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. ISBN 0201624206. 98

Dept. Dave Snowdon. Www3d: A 3d multi-user web browser, 1996. URL `citeseer.ist.psu.edu/679930.html`. 48

Tom Sperlich. Interaktivität ist ein Witz, Jaron Lanier im Gespräch mit Tom Sperlich. *c'T*, 6:p. 68, 1995. 5, 55

Sprut. Grundlagen, Datenleitung, Handshake der RS232 Schnittstelle. retrieved on the World Wide Web: http://www.sprut.de/electronic/interfaces/rs232/rs232.htm. 123

Jonathan Steuer. Defining virtual reality: dimensions determining telepresence. *Communication in the age of virtual reality*, pages 33–56, 1995. 53

Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual reality on a WIM: interactive worlds in miniature. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: http://doi.acm.org/10.1145/223904.223938. 68

Nam P. Suh. *The Principles of Design*. Oxford University Press: New York, Oxford, 1990. 13

Alexander Suhm. *Produktmodellierung in wissensbasierten Konstruktionssystemen auf der Basis von Lösungsmustern*. Shaker, 1993. ISBN 3-86111-574-3. 11

Ivan E. Sutherland. Sketchpad - A Man-Machine Graphical Communication System. *SJCC ' 63*, pages 329–345, 1963. 57

Matthias Svoboda and Pamela Ravasio. Die Entwicklung der Desktopmetapher - Semesterarbeit. Technical report, ETH, Zrich, September 2003. 157

Andrew S. Tanenbaum. *Modern operating systems*. Prentice Hall, 2. ed. edition, 2001. ISBN 0-13-031358-0. 33

Andrew S. Tanenbaum. *Moderne Betriebssysteme*. Pearson Studium, 2., überarb. a. edition, 2002. ISBN 978-3827370198. 34

Klaus-Dieter Thies. *Multitasking*. Hanser, 1994. ISBN 3-446-17506-7. 33

K. R. Thórisson, D. B. Koons, and R. A. Bolt. Multimodal Natural Dialogue. *SIGCHI '92 Proceedings*, pages 139–140, 1992. 73

T. Tomiyama, T. Kiriyama, H. Takeda, D. Xue, and H. Yoshikawa. Metamodel: A Key to Intelligent CAD Systems. *Research in Engineering Design*, 1989. 25

Tetsuo Tomiyama and Hiroyuki Yoshikawa. Knowledge engineering and CAD. *Future Gener. Comput. Syst.*, 1(4):237–243, 1985. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/0167-739X(85)90012-3. 15, 16

Tetsuo Tomiyama, Deyi Xue, Yasushi Umeda, Hideaki Takeda, Takashi Kiriyama, and Hiroyuki Yoshikawa. Systematizing Design Knowledge for Intelligent CAD Systems. In *Proceedings of the IFIP TC5 / WG5.3 Eight International PROLAMAT Conference on Human Aspects in Computer Integrated Manufacturing*, pages 237–248, Amsterdam, The Netherlands, The Netherlands, 1992. North-Holland Publishing Co. ISBN 0-444-89465-9. 13, 15

Ozgur Turetken and Ramesh Sharda. Visualization of web spaces: state of the art and future directions. *SIGMIS Database*, 38(3):51–81, 2007. ISSN 0095-0033. doi: http://doi.acm.org/10.1145/1278253.1278260. 48

David G. Ullman. A Taxonomy of Mechanical Design. *Research in Engineering Design*, 3(3):179–189, 1992. 128

VDI 2210. *Analyse des Konstruktionsprozesses im Hinblick auf den EDV Einsatz*. Düsseldorf: VDI-Verlag, 1975. 11

VDI 2221. *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*. Berlin: Beuth Verlag, 1993. 10, 16

John Vince. *Virtual reality systems*. Addison-Wesley, 1. pr. edition, 1995. ISBN 0-201-87687-6. 56

K.M. Wallace and C. Hales. Systematic Design in Practice. In HEURISTA, editor, *Proceedings of ICED 91, Schriftenreihe WDK 20*, 1991. 13

Weixin Wang, Hui Wang, Guozhong Dai, and Hongan Wang. Visualization of large hierarchical data by circle packing. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 517–520, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7. doi: http://doi.acm.org/10.1145/1124772.1124851. 47

Colin Ware. Graph Visualizer 3D. Retrieved on the World Wide Web: http://www.omg.unb.ca/hci/projects/gv3d/, 1998. 48

Martin Wattenberg. Visualizing the stock market. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 188–189, New York, NY, USA, 1999. ACM. ISBN 1-58113-158-5. doi: http://doi.acm.org/10.1145/632716.632834. 47

Mark J. Weal. 3D Interfaces to Hypermedia Information Spaces. Master's thesis, University of Southampton, 1996. 44

Jean-Jacques V. Wintraecken. *The NIAM information analysis method*. Kluwer Academic, 1990. ISBN 0-7923-0263-X. 24

Matthias M. Wloka and Eliot Greenfield. The virtual tricorder: a uniform interface for virtual reality. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 39–40, New York, NY, USA, 1995. ACM. ISBN 0-89791-709-X. doi: http://doi.acm.org/10.1145/215585.215647. 72

Andrew Wood, Nick Drew, Russell Beale, and Bob Hendley. HyperSpace: Web Browsing with Visualisation. Retrieved on the World Wide Web under: `http://igd.fhg.de/archive/1995_www95/proceedings/posters/35/`, 1995. 3rd International WWW Conference'95, Darmstadt, Germany. 48

V. Wünsche. Visualisierung strukturierter Informationen mit VRML. Studienarbeit, Universität Rostock - Fachbereich Informatik, February 1997. 38

H. Yoshikawa. Design theory for cad. In H. Yoshikawa and E.A. Warman, editors, *Design Theory for CAD Proceedings of the IFIP WG5.2 Working Conference Tokyo, Japan*, page 464 page. NORTH-HOLLAND, 1987. ISBN 0-444-70151-6. 13, 15

Stefan Zerbst. *3D Spieleprogrammierung mit DirectX in C/C++*. Books on Demand GmbH, 2000. ISBN 3831105936. 106

Tianfeng Zhou. Semantische Abbildung von Funktionsmodellen auf Szenengraphen am Beispiel von Komponenten eines Fahrzeugsitzes. Diplomarbeit, Universität Karlsruhe - IMI, February 2008. Betreut durch Hardy Krappe. 33, 34, 35, 36, 37, 178, 179, 181, 182

ISSN: 1860-5990
ISBN: 978-3-86644-380-8

www.uvka.de