Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-95-05

Eine Werkbank zur Erzeugung von 3D-Illustrationen

Georg Schneider

August 1995

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Sema Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ☐ Intelligent Engineering Systems
- ☐ Intelligent User Interfaces
- ☐ Computer Linguistics
- ☐ Programming Systems
- ☐ Deduction and Multiagent Systems
- ☐ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

Eine Werkbank zur Erzeugung von 3D-Illustrationen

Georg Schneider

DFKI-D-95-05

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-9400).

Eine Werkbank zur Erzeugung von 3D-Illustrationen

Georg Schneider

mail: schneide@dfki.uni-sb.de
www: http://www.dfki.uni-sb.de/~schneide

30. August 1995

Inhaltsverzeichnis

1 Einleitung 4

1.1 Ziel der Arbeit 5

2 Stand der Forschung 10

2.1 Überblick über verschiedene Grafiksysteme 10

2.1.1 Grafische Editoren 10

2.1.2 Diskussion 12

2.1.3 Halbautomatische Assistenzsysteme 14

2.1.3.1 Das System "Designer" 14

2.1.3.2 Diskussion 15

2.1.3.3 Ein System zur "Verschönerung" von Grafiken . 16

2.1.3.4 Diskussion 17

2.1.3.5 Das System "Chimera" 18

2.1.3.6 Diskussion 18

2.1.3.7 Ein System zur Generierung von Skizzen 19

2.1.3.8 Diskussion 19

2.1.4 Vollautomatische Systeme 20

2.1.4.1 Das System "APT" 20

2.1.4.2 Diskussion 21

2.1.4.3 Das System "BOZ" 21

2.1.4.4 Diskussion 22

2.1.4.5 Das System "ANDD" 22

2.1.4.6 Diskussion 24

2.1.4.7 Ein System zum Erzeugen von Bildern für Dokumentationen 24

2.1.4.8 Diskussion 25

2.1.4.9 Automatische Synthese von Grafiken aus Objektbeschreibungen 25

2.1.4.10 Diskussion 26

2.1.4.11 Das System "IBIS" 27

2.1.4.12 Diskussion 29

2.1.4.13 Die interaktive Version von "IBIS" 29

2.1.4.14 Diskussion 30

2.1.5 Resümee 30

2.2 Technisches Zeichnen 31

3 Grundkonzepte **34**
3.1 Illustratorische Techniken 34
3.2 Ausgangsdaten 34
3.2.1 3D-Objekte 35
3.2.2 Weltzustände 38
3.2.3 Pseudoobjekte 39
3.2.4 Illustratorische Szenen 40

4 Operationalisierung von Illustrationstechniken **43**
4.1 Aufrißtechnik 43
4.2 Explosionstechniken 50
4.3 Querschnitte 64
4.4 3D Trajektorienpfeile 67
4.5 Ghost Images 70
4.6 Virtuelle Böden 74
4.7 Einpassen einer Szene auf eine vorgegebene Abbildungsgröße . 77
4.8 Fokussieren durch Zentrieren von Objekten 79
4.9 Perspektivenauswahl 80
4.10 Erstellen von Insets 89
4.11 Abstraktion 91
4.12 Evaluierungsoperatoren 93

5 Konzeption der Werkbank TOPAS **96**
5.1 Systemarchitektur 96
5.2 Szenenverwaltung 97
5.3 Oberfläche 99
5.3.1 Interaktives Grafikfenster 99
5.3.2 Menüs 100
5.3.3 Kommandozeile 101
5.3.4 Skripteditor 101
5.4 Voraussetzungen an ein externes Grafiksubsystem 102

6 Arbeiten mit TOPAS **103**
6.1 Beispiele für Illustrationen mit **TOPAS** 103
6.2 Abstraktionsniveau der **TOPAS**-Befehle im Vergleich zu Grafikeditoren 107

7 Einsatzfelder von TOPAS 108
7.1 TOPAS als Assistenzsystem 108
7.1.1 Aufsatz für CAD-Systeme 108
7.1.2 Entwicklungsumgebung zur Operationalisierung von Illustrationstechniken 108
7.1.3 Key-Frame Editor 109
7.2 TOPAS als Teilkomponente intelligenter Präsentationssyteme 109
7.2.1 WIP 110
7.2.1.1 Überblick 110
7.2.1.2 Der Grafikgenerator von WIP 111
7.2.1.3 Die Rolle von TOPAS bei der Grafikgenerierung 113
7.2.2 PPP 113
7.2.3 Illustrationen als Benutzerschnittstelle 114

8 Zusammenfassung und Ausblick 115
8.1 Leistungsumfang von TOPAS 115
8.1.1 Operatoren 115
8.1.2 Betriebsmodi 116
8.2 Ausblick 117
8.2.1 Integration weiterer 3D-Techniken 117
8.2.2 Integration weiterer 2D-Techniken 118
8.2.3 Beleuchtung 118
8.2.4 Anbindung an kommerzielle CAD-Systeme 118

A Beispiele für das Arbeiten mit TOPAS 119

B Die TOPAS-Operatoren 129
B.1 Basisoperatoren zum Erstellen von Szenen 129
B.2 Techniken zur Manipulation von 3D-Modellen 131
B.3 Operatoren zur Perspektivenwahl 133
B.4 Operatoren zum Erstellen von virtuellen- und metagrafischen Objekten 135
B.5 Operatoren zu 2D-Techniken 137
B.6 Evaluierungsoperatoren 138

C Beispielskripte 139

1 Einleitung

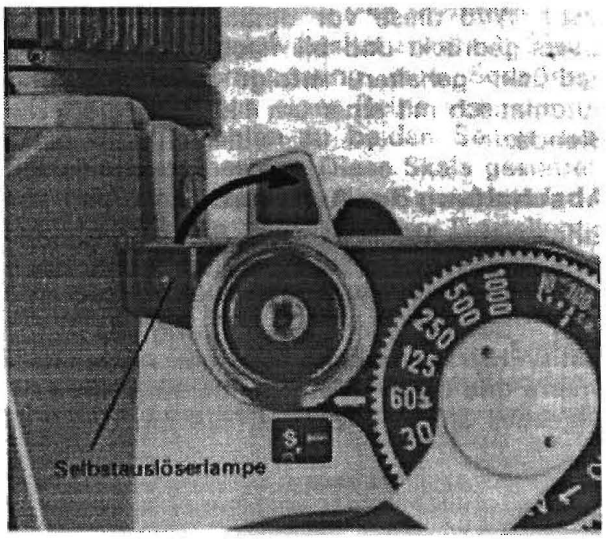

Abbildung 1: Die Abbildung zeigt die Bedienung des Selbstauslösers einer Fotokamera. Aus: [Canon 81].

Gebrauchsanweisungen für technische Geräte gehören sicherlich zu der in unseren Tagen am meisten gelesenen Prosa, da uns ständig neue Geräte das tägliche Leben erleichtern. Leider mangelt es diesen Schriftstücken oft an Aussagekraft, was den Umgang mit den Geräten erschwert. Ein Grund dafür sind häufig die mangelhaften Illustrationen, die kaum erkennen lassen, wie der Benutzer vorgehen soll. Nicht selten klafft eine große Lücke zwischen den Illustrationen, auf denen das Gerät zu sehen ist und dem realen Modell, das der Benutzer vor seinen Augen hat.
Die Abbildung 1 zeigt anhand der Bedienung des Selbstauslösers eines Fotoapparates, wie eine gelungene Illustration aussieht.

Mit einem Blick kann ein Betrachter herausfinden,

- wo sich die Bedienteile am Fotoapparat befinden,
- welche Teile für die Handlung wichtig sind und
- wie diese Handlung auszuführen ist,

ohne den zugehörigen Anleitungstext zu lesen.

Dies ist auch deshalb wichtig, weil viele Benutzer vor dem Lesen dicker Bedienungsanleitungen zurückschrecken. Die Vorgehensweise, nur die Überschriften zu lesen und die Bilder zu betrachten, ist heutzutage weit verbreitet.
Sollte hingegen obige Handlung ausschließlich sprachlich ausgedrückt werden, so wäre eine längere Ausführung vonnöten.
Im Anleitungstext wird die Handhabung des Selbstauslösers, ergänzend zum Bild, folgendermaßen beschrieben: "*Eingeschaltet wird der elektronische Selbstauslöser der AE1 durch Vorziehen eines Hebels.*" Aus [Canon 81].
Diese sehr kurze Beschreibung kann deshalb zur Beschreibung der Situation verwandt werden, weil aus der Abbildung 1 der genaue Sachverhalt bereits hervorgeht.
Speziell in dem hier beschriebenen Fall wird noch eine weitere wichtige Eigenschaft von Illustrationen deutlich - sie erhöhen die Robustheit von Betriebsanleitungen. Die unglücklich gewählte Bezeichnung "Vorziehen" führt ausschließlich auf Grund des Bildes nicht zu Fehlinterpretationen, handelt es sich doch bei der Bewegung viel mehr um eine Schwenkbewegung, als um eine Ziehbewegung.
Ein weiterer Vorteil von Illustrationen ist ihre weitgehende Sprachunabhängigkeit. In obigem Beispiel kann nach Austausch des Wortes "Selbstauslöserlampe" die gleiche Illustration ebenfalls für anderssprachige Bedienungsanleitungen genutzt werden. Würde beispielsweise die "Selbstauslöserlampe" symbolisch annotiert werden, beispielsweise durch "*", statt durch das deutsche Wort, so könnte die Illustration sogar ohne Änderungen in anderssprachige Dokumente übernommen werden.

1.1 Ziel der Arbeit

Die vorliegende Arbeit wird sich mit der computerbasierten Erstellung von Illustrationen für diese technischen Gebrauchs- und Bedienungsanleitungen beschäftigen.
Das Ziel ist, eine Werkbank zu entwerfen und zu implementieren, die ein Illustrator benutzen kann, um technische Illustrationen zu erzeugen. Es soll dabei möglich sein, die Illustrationen nicht nur von einem menschlichen Illustrator erstellen zu lassen. Ein Computersystem soll ebenfalls als Illustrator fungieren können.

In diesem Zusammenhang stellt sich die Frage, welche Art von Illustrationen für technische Gebrauchs- und Bedienungsanleitungen geeignet sind.
Technische Illustrationen sind Illustrationen, die dazu dienen, einem Leser technische Geräte in Dokumenten, wie beispielsweise Gebrauchs- und Bedienungsanleitungen, aber auch in Produktinformationen, Werbebroschüren oder ähnlichen Publikationen zu erklären. Ihre Aufgabe ist es, dementsprechend Information über diese Geräte zu vermitteln, über ihre Funktionsweise und ihre Handhabung. Dieses Ziel kann auf unterschiedliche Art und Weise erreicht werden. Die Auswahl, mit welchen Mitteln eine solche die Illustration erstellt wird, bleibt größtenteils dem Autor selbst überlassen. In vielen Fällen genügt eine einfache Skizze, während es in anderen Fällen zweckmäßiger ist, eine fotorealistische Darstellung zu benutzen.
Folgende Anforderungen sind an eine solche Werkbank zu stellen.

Breites Repertoire an Techniken

Der Illustrator soll ein breites Repertoire an Techniken zur Verfügung haben, wie

- Explosionszeichnungen,
- Aufrißdarstellungen,
- Querschnittdarstellungen,
- 3D-Pfeile,
- Insets,
- Ghost Images,
- Metagrafische Objekte (z.B. virtuelle Böden),
- Perspektivenwahl,
- Veränderung der Abbildungsgröße (zoom),

damit für jeden Zweck die geeigneten Werkzeuge zur Verfügung stehen. Hierbei soll besonderes Augenmerk auf 3D-Techniken gelegt werden, um die Voraussetzungen für eine variantenreiche Gestaltung von 3D-Illustrationen zu gewährleisten.

Halbautomatische Arbeitsweise

Einige Illustrationstechniken führen immer zu ähnlichen, vorhersehbaren Resultaten. Diese Techniken lassen sich weitgehend automatisieren. Andere Illustrationstechniken sind jedoch sehr situationsabhängig oder hängen stark vom ästhetischen Empfinden eines Illustrators ab. Diese Techniken sind schwer zu operationalisieren und können weniger stark automatisiert werden. Folglich würde eine halbautomatische Vorgehensweise eine flexible Gestaltung der Illustrationen ermöglichen und dem Illustrator größtmöglichen Einfluß auf den Gestaltungsprozeß sichern.

Variable Wissenstiefe

Der Benutzer soll selbst entscheiden können, welches bzw. wieviel Wissen er dem System zur Verfügung stellt. Damit wäre er selbst in der Lage, eine Auswahl über die Bereiche zu treffen, in denen er die Unterstützung des Systems wünscht. Eine Folgerung hieraus ist, daß das System schon mit geringem Wissen einsatzfähig sein muß. Wenn mehr Information zur Verfügung steht, soll es sich diese zu Nutzen machen.

Komfortable Oberfläche

Die Werkbank soll dem Benutzer effizientes Arbeiten ermöglichen, indem sie ihm sich komplementierende Arbeitsmodi zur Verfügung stellt und ihn unterstützt:

- Programmgesteuert
- Interaktiv

Eine Schnittstelle, die eine klar definierte Kommandosprache enthält, soll das programmgesteuerte Erstellen von Illustrationen einfach und durchschaubar machen.
Mit Hilfe dieser Sprache soll ein Benutzer die Werkbank programmieren und als Subsystem in seine Anwendung einbinden können. Sie soll ebenfalls dazu benutzt werden können, sogenannte "Makros" zu definieren. In diesem Fall wären das vorgefertigte Illustrationen für immer wiederkehrende Präsentationssituationen.
Die Werkbank soll es einem Illustrator ermöglichen, einfach und schnell Bilder mit den gewünschten Informationsinhalten zu erstellen. Hierfür soll zum

interaktiven Erstellen von Illustrationen eine bedienerfreundliche und übersichtliche Oberfläche zur Verfügung stehen.
Bei dieser Arbeitsweise soll der Benutzer im Dialog mit dem System die Illustrationen erstellen können. Hierbei sollen mehrere Arten der Interaktion möglich sein:

- Menügesteuert
- Direkte Manipulation
- Eingabeaufforderung

Bei der menügesteuerten Arbeitsweise soll der Benutzer das System über eine Oberfläche bedienen. Die Menüs sollen ihm die verschiedenen Manipulationsmöglichkeiten beim Erstellen und Verändern der Grafiken aufzeigen. Eine Verzahnung zwischen programmgesteuerter und interaktiver Arbeitsweise soll ebenfalls möglich sein. Weiterhin ist es wünschenswert, daß der Benutzer im menügesteuerten Modus Zugriff auf die oben beschriebenen Makros hat, ihre Realisierung verfolgen kann und das Resultat interaktiv weiterbearbeiten kann.
Techniken der direkten Manipulation sollen dem Benutzer die Möglichkeit geben, direkt in die Grafik einzugreifen und sie zu verändern. So sollen beispielsweise Objekte in der Grafik mit der Maus selektierbar sein, die dann unmittelbar modifiziert werden können.
Das System soll zusätzlich die Möglichkeit bieten, die Grafik durch Eingabe von Kommandos über eine Befehlszeile zu manipulieren.

Datenintegration

Die Illustrationen sollen möglichst aus den für die Produktion vorhandenen Daten direkt erstellt werden. Dadurch soll ein wiederholtes Erstellen des Modells, das ausschließlich zum Zwecke der Illustration dient, überflüssig werden. Diese Vorgehensweise ist weniger fehleranfällig und würde eine völlige Neugestaltung der Illustrationen bei geringfügigen Änderungen im Design erübrigen. Sie würde ebenfalls die Variantenkonstruktion erleichtern.

Repräsentation von Input und Output

Der Bildinhalt soll intern repräsentiert werden, um andere Systeme mit der

Information über die generierte Illustration zu versorgen. Gleichfalls muß eine Historie über gestellte Aufgaben angelegt werden, um zu wissen, welche Operationen bereits zum Einsatz kamen.

Breites Anwendungsgebiet

Das System soll domänenunabhängig sein. Die Techniken sollen nicht für spezielle Objekte entwickelt werden, sondern allgemeingültig für beliebige Modelle. Dadurch soll durch das System ein breites Anwendungsgebiet erschließen.

Weitgehende Soft- und Hardwareunabhängigkeit

Eine weitere wichtige Anforderung ist die nach weitestgehender Soft- und Hardwareunabhängigkeit. Es soll eine Schnittstelle zu vorhandener Software bestehen, und es sollen möglichst keine hardwareabhängigen Besonderheiten ausgenutzt werden. Somit kann der Portierungsaufwand des Systems auf andere Plattformen gering gehalten werden. Dies soll weiterhin gewährleisten, daß die Werkbank auf unterschiedliche 3D-Grafiksysteme aufsetzen kann.

2 Stand der Forschung

In diesem Kapitel werden verwandte Arbeiten hinsichtlich ihrer Vorgehensweise bei der Grafikgenerierung untersucht. Im ersten Teil werden Computersysteme vorgestellt, angefangen bei Editoren über Assistenzsysteme bis hin zu vollautomatischen Generatoren. Danach werden relevante Arbeiten aus einer benachbarten Disziplinen vorgestellt und ihr Einfluß auf die Erstellung von Illustrationen untersucht.

2.1 Überblick über verschiedene Grafiksysteme

Dieser Abschnitt stellt verschiedene Grafiksysteme vor, die mehr oder weniger gut zur Erstellung von Illustrationen für technische Gebrauchs- und Bedienungsanleitungen geeignet sind. Bei einem Teil der Systeme handelt es sich um kommerziell verfügbare Software, ein anderer Teil stammt aus Forschungseinrichtungen.
Zunächst werden Bildverarbeitungssysteme oder Editoren vorgestellt, die das konventionelle durch ein computerisiertes "Zeichenbrett" ersetzen.
Die halbautomatischen Assistenzsysteme unterstützen den Illustrator durch Verbesserungsvorschläge aktiv bei seiner Arbeit.
Die vollautomatischen Systeme erstellen die Grafiken automatisch und benötigen/ermöglichen keine Eingriffe des Benutzers.

2.1.1 Grafische Editoren

Grafische Editoren sind Grafiksysteme, die dazu dienen, die Erstellung von Illustrationen zu vereinfachen. In diesem Abschnitt wird versucht, ein breites Spektrum an kommerziell erhältlichen Editoren abzudecken, die zum Erzeugen von Illustrationen benutzt werden können. Die Editoren lassen sich in verschiedene Klassen unterteilen.
Einige Systeme beruhen ausschließlich auf 2D-Techniken, sie erlauben lediglich das Manipulieren einer Vorlage, die durch Abtasten eines Bildes mit einem "Scanner" ermittelt wurde. Ausgehend von dieser Vorlage können dann Illustrationen erstellt werden. Diese Systeme basieren hauptsächlich auf Veränderungen der Bildschirmpunkte und werden als bitmaporientierte Editoren bezeichnet.

Ein Beispiel hierfür ist der "ADOBE Photoshop" der Firma "Adobe Systems". Er ist "... *ein Fotoretusche-, Bildverarbeitungs- und Farbmalprogram* ..." (aus [Adobe Photoshop 90]).
Diese Art der Illustrationserstellung ist jedoch sehr unflexibel und umständlich, da für jede Präsentationssituation ein Foto des physikalischen Objekts gemacht und mit dem Scanner abgetastet werden muß. Jede dieser Vorlagen muß dann einzeln durch "Ausschneiden", "Einkleben", "Reinmalen" oder andere Bildmanipulationstechniken weiterbearbeitet werden.
Ein weiterer Nachteil dieser Systeme ist der, daß keine objektorientierte Beschreibung des Inhalts der Illustration möglich ist.
Andere Grafikeditoren arbeiten vektororientiert. Sie dienen dazu, Illustrationen für Präsentationen, bzw. Gebrauchs- und Werbegrafiken zu erstellen. Es werden jedoch keine spezifischen Funktionen für die Erstellung von technischen Illustrationen zur Verfügung gestellt, da diese Programme als universelle Zeicheneditoren konzipiert sind.
Ein Beispiel hierfür ist "CorelDraw 5.0" der Firma "Corel Corporation" (siehe [Liska 94]). Bei diesem Editor besteht bereits die Möglichkeit, einfache 3D-Objekte zu erzeugen und sie von Lichtquellen bescheinen zu lassen, um realistische Darstellungen zu erzeugen. Komplexere Objekte können jedoch nicht modelliert werden.
Zu diesem Programm werden sehr viele Cliparts und Ikonen mitgeliefert, ihr Inhalt ist jedoch weder genau beschrieben noch repräsentiert und kann nur durch "Nachschauen" ermittelt werden. Die Flexibilität einer genau auf die Präsentationssituation zugeschnittenen Illustration kann jedoch selbst durch eine noch so große Anzahl von Cliparts nicht erreicht werden.
Eine Programmierschnittstelle steht bei diesen Systemen nicht zur Verfügung, so daß sie nur über die Oberfläche bedient werden können.
Eine andere Klasse von Systemen wurde speziell zur Erstellung von Illustrationen für technische Gebrauchs- und Bedienungsanleitungen entwickelt. Beispiele hierfür sind "AutoSketch" von der Firma "Autodesk GmbH" (siehe [AutoSketch 94] und [Ebeling 94]) und "IsoDraw" von der Firma "Itedo Software GmbH" (siehe [Ehrmann 94] und [IsoDraw 94]).
Diese Systeme ermöglichen uneingeschränkt die Erstellung von 3D-Objekten. Perspektivisches Zeichnen in Fluchtpunkt- und Parallelperspektive und isometrisches Zeichnen werden ebenfalls unterstützt.
Es existieren Symbolbibliotheken für spezielle Anwendungen mit Normansichten von Schrauben, Muttern, Federn, was zwar für den Konstruktions-

prozeß hilfreich, für den Illustrationsprozeß jedoch von geringerer Bedeutung ist.
Einige vordefinierte Ansichten, wie Draufsicht, Vorderansicht, Seitenansicht, Isometrie oben/unten, können eingestellt werden, es fehlt jedoch auch hier die nötige Flexibilität, um maßgeschneiderte Illustrationen zu erstellen.
Beim isometrischen Zeichnen können Objekte auf zueinander parallelen Ebenen im Raum plaziert und in diesen Ebenen gespiegelt werden, was die Erstellung von Explosionsdarstellungen vereinfacht. Eine weitergehende Unterstützung des Illustrators findet jedoch nicht statt.
Die Systeme sind nur ansatzweise programmierbar, eine interne Programmiersprache ist nicht vorhanden, ein Zugriff auf die Daten der erstellten Objekte ist nicht möglich. Die Systeme können deshalb nicht programmgesteuert betrieben werden.
Im Gegensatz zu den zuvor erwähnten Grafikeditoren sind noch 3D-Editoren verfügbar, die zum Modellieren und Manipulieren von 3D-Objekten dienen. Beispiele hierfür sind "AutoCAD Release 12" von der Firma "Autodesk GmbH" (siehe [AutoCAD 94]) und "S-Geometry", "S-Render" von der Firma "Symbolics" (siehe [S-Geometry 90]).
Diese Systeme können durch einen Renderprozeß realistische 3D-Darstellungen erzeugen und bieten die Möglichkeit, beliebige räumliche Ansichten eines 3D-Modells zu generieren. Eine Kamera, die die erstellten Objekte aufnimmt, wird dabei frei im Raum plaziert.
"AutoCAD Release 12" verfügt über die eingebettete Programmiersprache "AutoLISP". Eine Schnittstelle zur Sprache "C" ist ebenfalls vorhanden, um Oberfläche und System zu programmieren.
Von besonderem Interesse für die vorliegende Arbeit sind "S-Geometry" und "S-Render", die in der Programmiersprache "Lisp" geschrieben sind. Eine Programmierung des Systems und der Oberfläche ist in dieser Sprache möglich, weiterhin ist sogar der Quelltext vorhanden. Sämtliche internen Daten sind repräsentiert und stehen auch extern zur Verfügung. Die Tabelle in Abbildung 2 führt die wichtigsten Merkmale der kommerziell verfügbaren Editoren auf.

2.1.2 Diskussion

Zusammenfassend kann gesagt werden, daß im kommerziellen Bereich kein System verfügbar ist, mit dem komfortabel Illustrationen erstellt werden

Programm	ADOBE Photoshop	CorelDraw 5.0	IsoDraw	AutoSketch	AutoCAD	S-Geometry/S-Render
Hersteller	Adobe Systems	Corel Corporation	Itedo Software GmbH	Autodesk GmbH	Autodesk GmbH	Symbolics
2D	✓	✓	✓	✓	✓	
3D		✓	✓	✓	✓	✓
Perspektive		✓	✓	✓	✓	✓
realist. Darstellungen		✓			✓	✓
Programmier-schnittstelle					✓	✓
Zugriff auf interne Strukturen						✓
Eigenschaften für den Illustrationsentwurf		22000 Cliparts	❋	❋ ❋	❋ ❋ ❋	Quellcode erhältlich

❋ Dient speziell zum Erstellen von technischen Illustrationen;
enthält Bibliothek mit 2000 Normansichten von Schrauben, Federn, usw.

❋ ❋ Dient speziell zum Erstellen von technischen Illustrationen;
erhältlich sind Applikationen und Objektbibliotheken für Elektronik, Maschinenbau, usw.

❋ ❋ ❋ Anschluß an SQL Datenbanken ist vorhanden;
berechnet Flächeninhalt, Umfang, Trägheitsmoment;
erhältlich sind Applikationen und Objektbibliotheken für Elektronik, Maschinenbau, usw.

Abbildung 2: Vergleich der Editoren

können. Bildverarbeitende Systeme, wie "ADOBE Photoshop" scheinen für diesen Zweck eher ungeeignet, da sie reine 2D-Systeme sind und die Erstellung von Illustrationen sich sehr mühsam gestaltet. Die anderen Systeme bieten zwar die Möglichkeit, in der dritten Dimension zu zeichnen, sie unterstützen einen Benutzer jedoch nicht bei der Erstellung der Illustrationen. Ansatzweise wird die Explosionstechnik beispielsweise bei dem Editor "IsoDraw" unterstützt, es besteht jedoch keine Möglichkeit, diese Unterstützung auszuweiten oder andere Techniken hinzuzufügen, da keine externe Programmierung des Systems möglich ist.
Die beiden zuletzt erwähnten Systeme ermöglichen zwar eine Programmie-

rung, es sind jedoch keine illustratorischen Techniken vorhanden. Sie kommen jedoch als Basissystem für eine Illustrationswerkbank in Frage. Vor allem "S-Geometry" und "S-Render" zeichnen sich dadurch aus, daß sie selbst in der Programmiersprache Lisp geschrieben sind und somit eine einfache Programmierung in Lisp ermöglichen. Weiterhin ist der Quellcode vorhanden und es besteht ein Zugriff auf sämtliche Daten des Systems.

2.1.3 Halbautomatische Assistenzsysteme

Während Editoren sich darauf beschränken, Werkzeuge zur Verfügung zu stellen, um den Entwurf von Illustrationen zu vereinfachen, versuchen halbautomatische Assistenzsysteme den Benutzer aktiv bei der Erstellung von Illustrationen zu unterstützen.

2.1.3.1 Das System "Designer"

In [Weitzman 86] wird ein System zum Entwurf grafischer Benutzerschnittstellen vorgestellt. Dieses System wird als Hilfesystem für "Steamer's Graphics Editor" (siehe Abbildung 3) eingesetzt. Dieser Editor ist ein computerbasierter Simulator für den Bereich Dampfantrieb. Der Simulator kann in verschiedenen Hierarchiestufen über eine farbige, grafische Benutzerschnittstelle beeinflußt werden. Er stellt viele verschiedene Sichten der Anlage zur Verfügung, begonnen mit detaillierten Sichten, die Druckanzeiger enthalten, bis hin zu abstrakteren Sichten (z.B. Netzdiagrammen). Der "Graphics Editor" ermöglicht es, Benutzern ohne Programmiererfahrung dynamische Sichten interaktiv und grafisch zu erzeugen.
Da die Benutzer selten mit den Grundlagen des Grafikdesign vertraut sind, und sie Stilkonventionen oft nicht über mehrere Sichten hinweg einhalten, wurde "Designer" entwickelt, um das Domänenwissen des "Graphics Editor" durch Wissen über gutes Grafikdesign zu erweitern. Sein Ziel ist es, Grafiken zu verschönern und den Benutzer bei deren Erstellung zu unterstützen.
"Designer" besteht aus drei, untereinander in Wechselbeziehung stehenden Komponenten: *Analyse*, *Kritik* und *Synthese*, wobei alle Komponenten mit einer Wissensbasis verknüpft sind.
Die *Analysekomponente* "parst" das Design.
Der *Kritiker* benutzt diese Information zusammen mit *Design Constraints*, um anzuzeigen wo das aktuelle Design erfolgreich ist oder fehlschlägt.

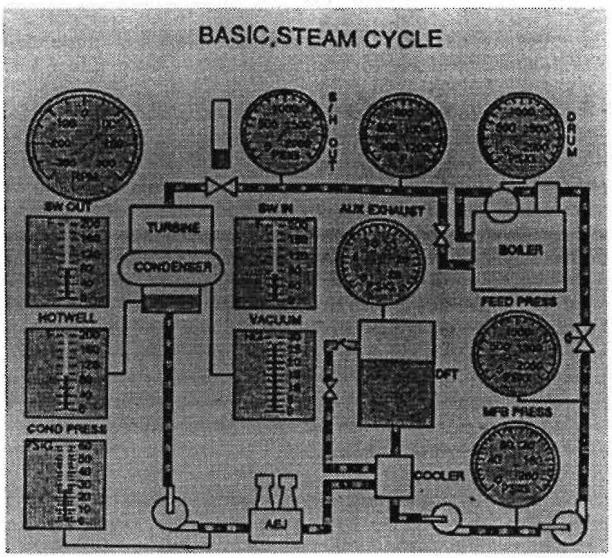

Abbildung 3: "Steamer's Graphics Editor". Aus: [Weitzman 86]

Die *Synthesekomponente* generiert daraufhin inkrementell mehrere mögliche Alternativen im entsprechenden Designstil, von denen der Benutzer dann eine auswählt (siehe Abbildung 4). Ein *Assumption-Based Truth Maintenance System* (ATMS[1]) wird benutzt, um alternative Designvorschläge aufrecht zu erhalten.

2.1.3.2 Diskussion

Das System "Designer" ist mit Domänenwissen und Designwissen ausgestattet, um so auch ästhetisch ansprechnede Grafiken zu erzeugen, bzw. Stilkonventionen zu befolgen. Eine Interaktionsmöglichkeit mit dem Benutzer ist ebenfalls vorhanden, so daß dieser letztendlich selbst entscheiden kann, ob er die vom System erzeugten Designalternativen akzeptiert oder verwirft.
Ein Nachteil des Systems ist jedoch das stark eingeschränkte Anwendungsgebiet. Es besteht außerdem keine Möglichkeit "Designer" programmgesteuert zu betreiben. Desweiteren sind die Grafiken auf den 2D-Bereich beschränkt,

[1]Eine genauere Beschreibung der Arbeitsweise von ATMS ist in [Wahlster 92] zu finden.

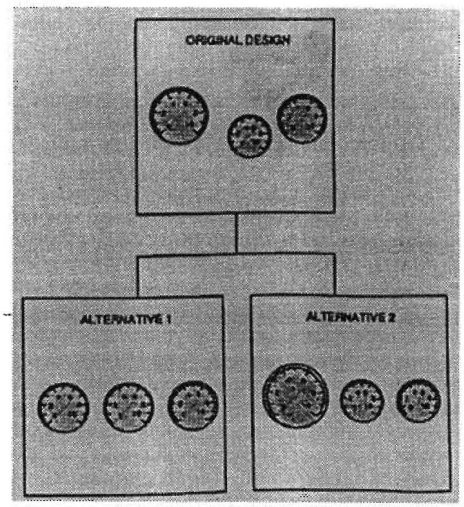

Abbildung 4: Vom System abgeleitete Alternativen. Aus: [Weitzman 86]

es können nur in der Wissensbasis abgelegte Ikonen zur Illustrationserstellung benutzt und keine neu erstellt werden.

2.1.3.3 Ein System zur "Verschönerung" von Grafiken

Ein System zur "Verschönerung" von mit der Hand gezeichneten Grafiken wurde im GMD Projekt Assistenzsysteme entwickelt (siehe [Bolz 93]), in dem innovative Methoden zur Mensch-Maschine Interaktion untersucht werden. Das "Beautifier-System" analysiert fertige oder unvollständige Zeichnungen, durch suchen nach Lücken, schiefen Linien und schlechter Ausrichtung (siehe Abbildung 5).
Weiterhin verfügt es über Wissen hinsichtlich "guten" Stils.
Nach einer *Analysephase* erstellt das System Vorschläge, um von ihm entdeckte "Fehler" zu berichtigen und versucht, sie in die Zeichnung zu integrieren. Hierfür existieren, ähnlich dem System "Designer", eine *Kritikphase* und eine *Korrekturphase.*
Die vom System als trivial eingestuften Fehler werden direkt verbessert. Entstehen widersprüchliche Anforderungen, kann der Benutzer befragt werden.

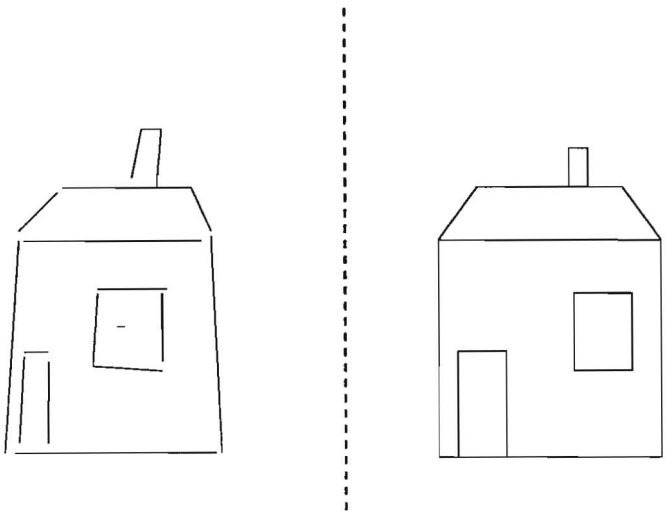

Abbildung 5: Eine Abbildung vor und nach einer Verschönerung, sinngemäß nach [Bolz 93].

Die Toleranzen in denen das System Verbesserungen vornimmt, lassen sich über Parameter einstellen (Winkel, Entfernungen und Abweichungen). Vom System generierte Veränderungen können über einen *Undo-Mechanismus* rückgängig gemacht werden.

2.1.3.4 Diskussion

Das System erzeugt aus einer relativ "unordentlichen" Vorlage eines Illustrators die "saubere" Zeichnungen. Es generiert Vorschläge, die es ihm offeriert und führt "triviale" Änderungen selbst durch. Diese Arbeitsweise scheint sinnvoll zu sein, um kooperativ mit einem Benutzer kompliziertere Illustrationen zu erstellen.
Ein Nachteil des Systems ist jedoch, daß die Vorlagen für sämtliche Illustrationen zuerst von einem menschlichen Illustrator vorgezeichnet werden müssen. Es werden keine Illustrationsvorschläge generiert. Das System ist nur auf den 2D-Bereich beschränkt und bietet keine externe Programmierschnittstelle zum Erzeugen von Grafiken oder illustratorischen Techniken.

2.1.3.5 Das System "Chimera"

In [Kurlander, Feiner 92] wird ein System vorgestellt, das einem Benutzer

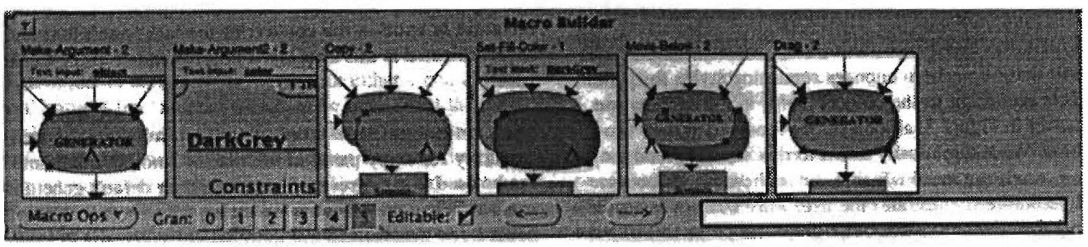

Abbildung 6: Ein Makro des Systems "Chimera". Aus: [Kurlander, Feiner 92]

ermöglicht, grafisch Makros für einen interaktiven 2D-Grafikeditor zu generieren, um so öfter wiederkehrende Operationsfolgen zu konservieren. Dies kann er durch Manipulation grafisch dargebotener Beispielmakros oder durch eine Auswahl von ihm selbst bei der Grafikerstellung benutzter Kommandos erreichen, die in einer grafischen History gespeichert werden. Sie werden als eine Art "Comicstrip" dargestellt.
Das Makro wird ebenfalls als "Comicstrip" visualisiert, die Einzeloperationen können durch selektieren eines Bildes dieser Sequenz verändert und mit sämtlichen Funktionen des Grafikeditors manipuliert werden.
Die Makros sind weiterhin parametrisierbar. Es können alle Objektattribute wie Farbe, Form, usw. der Objekte durch Parameter gesteuert werden, genauso wie die Argumente der einzelnen Operationen. Das System hilft in einem *Generalisierungsprozeß* beim Festlegen dieser Parameter.
Abbildung 6 zeigt ein Makro zur Erstellung eines Objektes mit Schatten.

2.1.3.6 Diskussion

"Chimera" bietet die Möglichkeit Makros grafisch darzustellen und jeden Schritt, bzw. jede Operation zu illustrieren. Dies ist von großer Bedeutung, wenn ein Illustrator die Erstellung einer Illustration verfolgen und eventuelle Fehler bei der Realisierung aufdecken möchte.

Zur Illustrationserstellung für technische Bedienungsanleitungen bietet "Chimera" jedoch keine Möglichkeiten, die über die eingangs beschriebenen kommerziellen Editoren hinausgehen. Das System bleibt auf den 2D-Bereich beschränkt, es stehen keine speziellen Techniken zur Illustrationserstellung zur Verfügung.

2.1.3.7 Ein System zur Generierung von Skizzen

Ein System zur semiautomatischen Erstellung von skizzenhaften Zeichnungen aus CAD-Modellen wird in [Emhardt, Strothotte 92] beschrieben. Der Renderprozeß wird in diesem System, abweichend von der üblichen Gebrauchsweise, nicht dazu benutzt, ein fotorealistisches Bild zu erzeugen, sondern um eine Zeichnung zu generieren, die vom Mensch erstellten Zeichnungen ähnelt.
Durch diese Vorgehensweise sollen kommunikative Ziele besser vermittelt werden, und es soll durch eine Reduktion der Datenmenge ein interaktives Arbeiten mit den Illustrationen ermöglicht werden.
Der Benutzer spezifiziert beim Arbeiten mit dem "Hyper-Renderer" interaktiv und direktmanipulativ *Sichtbarkeitsinformation* für einzelne Bildbereiche, die dann entsprechend seiner Vorgaben gerendert werden. Je nach Detaillierungsgrad wird dabei vom System die Anzahl der dargestellten Liniensegmente gewählt.
Durch Effektgeneratoren können beispielsweise Flächen zur Hervorhebung schraffiert werden, es können Pflanzen oder Bewegungslinien generiert werden und bestimmte Bildbereiche fokussiert werden.

2.1.3.8 Diskussion

Der "Hyper-Renderer" bietet die Möglichkeit, 3D-Objekte in unterschiedlichem Detaillierungsgrad darzustellen. Der Benutzer kann direktmanipulativ in die Illustration eingreifen, was für die Erstellung komplexer Illustrationen hilfreich ist.
Es stehen jedoch keine Illustrationstechniken zur Verfügung. Ein Benutzer wird bei der Illustrationserstellung nicht unterstützt, er müßte seine Illustrationen von Hand mit einem CAD-System anfertigen und sie dann dem "Hyper-Renderer" übergeben.

2.1.4 Vollautomatische Systeme

Vollautomatische Systeme generieren Illustrationen ohne weiteres Zutun eines Benutzers.

2.1.4.1 Das System "APT"

[Mackinlay 86] entwickelte das System "APT" (A Presentation Tool), um das Design von Geschäftsgrafiken effektiv zu automatisieren.
Er faßt eine Präsentation als Satz einer *grafischen Sprache* auf, die syntatktisch und semantisch genau definiert ist.
Mit Hilfe einer *Kompositionsalgebra*, bestehend aus *grafischen Sprachen* und zugehörigen *Kompositionsoperatoren*, können komplexe Präsentationen erzeugt werden.

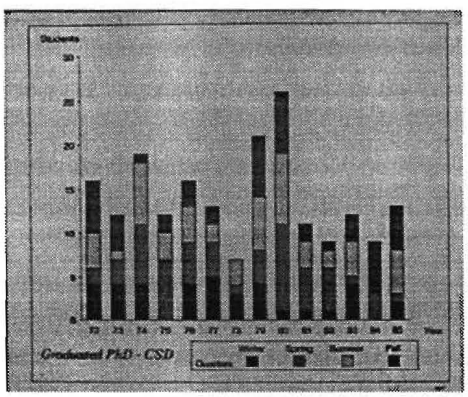

Abbildung 7: Ein durch "APT" generiertes Balkendiagramm. Aus: [Foley et al. 90]

Das in Abbildung 7 dargestellte Balkendiagramm ist beispielsweise ein Satz einer komplexen *grafischen Sprache*, der aus elementaren *grafischen Sprachen* mit Hilfe von *Kompositionsoperatoren* zusammengesetzt ist. Die elementaren *grafischen Sprachen* sind hierbei Achsen und Rechtecke in einer bestimmten Form und Farbe.
Durch diese Vorgehensweise kann die Menge der vorgefertigten grafischen Stile auf wenige Grundelemente reduziert werden.

Die Auswahl, mit welcher Technik eine Illustration realisiert wird, erfolgt nach sogenannten *Ausdruckskriterien* und *Effektivitätskriterien*:

- Eine Menge von Fakten ist in einer Sprache *ausdrückbar*, wenn sie einen Satz enthält, der alle Fakten der Menge beschreibt, und auch nur diese Fakten und keine zusätzlichen.
- Die *Effektivitätskriterien* bestimmen, wie gut die gewählte grafische Sprache die Möglichkeiten des Ausgabemediums und die Wahrnehmungfähigkeiten des Menschen ausschöpfen.

2.1.4.2 Diskussion

Obwohl im System "APT" ein für die automatische Grafikgenerierung vielversprechender Ansatz realisiert wurde, bleibt jedoch noch ungeklärt, inwieweit sich dieser Ansatz auch für die automatische Generierung von 3D-Illustrationen eignet.

2.1.4.3 Das System "BOZ"

Das System "BOZ" von [Casner 91] dient ebenfalls zur automatischen Erzeugung von abstrakten 2D-Diagrammen. In "BOZ" wird das Erzeugen dieser Präsentationen als eine Funktion der vom Benutzer zu bewältigenden Aufgabe angesehen. Ziel ist es, Präsentationsaufträge genau auf die gestellten Anforderungen zuzuschneiden.
In einer *logischen Beschreibung* wird hierbei zuerst die Problemstellung spezifiziert, die in der Grafik zum Ausdruck kommen soll.
In einem weiteren Schritt wird versucht, die verwendeten *logischen Operatoren* durch äquivalente *Wahrnehmungsoperatoren* zu ersetzen. Es können hierbei mehrere *Wahrnehmungsoperatoren* in Frage kommen. Soll beispielsweise gezeigt werden, daß eine von zwei Zahlen größer ist als die andere, so kann dies visuell auf einem Zahlenstrahl durch Eintragen der Zahlen an ihrer Position geschehen, es kann aber auch durch zwei verschiedengroße Quadrate visualisiert werden.
Im folgenden Schritt werden die Einzelprobleme der Problemstellung gruppiert, jenachdem, ob sie in einer gemeinsamen Grafik visualisiert werden können oder nicht.

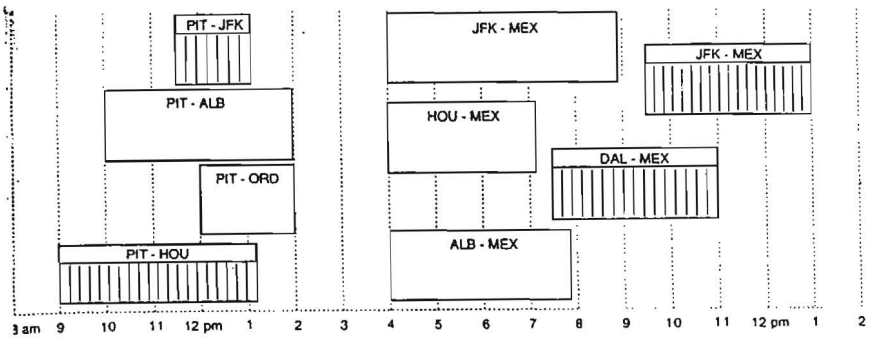

Abbildung 8: Die vom System "BOZ" generierte Grafik zeigt Flüge und deren Ankunfts-, bzw. Abflugzeiten. Sinngemäß nach [Casner 91].

Danach wird einer der gültigen *Wahrnehmungsoperatoren* für jede dieser Gruppen festgelegt.
In einem letzten Schritt wird die Präsentation, evtl. aus mehreren Illustrationen bestehend, erzeugt.
Abbildung 8 zeigt eine von "BOZ" erzeugte Präsentation, die Flüge und deren Ankunfts-, Abfluzeiten und die Dauer der Zwischenaufenthalte von Pittsburgh nach Mexico City zeigt.

2.1.4.4 Diskussion

Die Besonderheit von "BOZ" liegt darin, daß die vom Benutzer zu lösende Aufgabe in den Designprozeß einfließt.
Allerdings werden von "BOZ", genau wie in "APT", nur abstrakte 2D-Präsentationen erzeugt.

2.1.4.5 Das System "ANDD"

Ein System zum automatischen Erzeugen von Netzwerkdiagrammen wird in [Marks 91] beschrieben.
Als Eingabe erhält "ANDD" einen "Attributgraph", d.h. einen Graph mit Knoten und Kanten, die domänenabhängig, nominale, ordinale oder quanti-

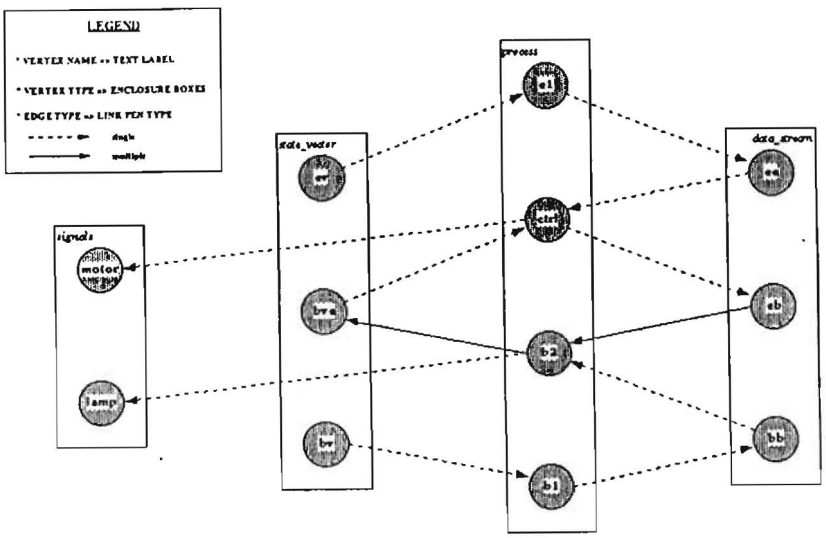

Abbildung 9: Ein vom System "ANDD" automatisch generiertes Netzwerk. Aus: [Marks 91]

tative Attribute besitzen können.
"ANDD" stehen zwei Verfahren des Layout zur Verfügung, ein schnell arbeitender, regelbasierter Algorithmus und ein langsamerer, jedoch robuster und mächtiger paralleler genetischer Algorithmus.
"ANDD" kann zusätzlich pragmatische Designregeln als Eingabe verarbeiten, z.B. das Ziel, eine Kante hervorzuheben.
Eine Hierarchie der Knoten kann durch eine Ordnungsrelation über die Attribute der Knoten erzielt werden, die dann beispielsweise durch eine breitere Knotenumrandung visualisiert wird.
Weiterhin kann "ANDD" zwischen verschiedenen grafischen Paletten auswählen, um die Präsentation dem Ausgabemedium anzupassen.
Beispiele hierfür sind Schraffierung, Form der Knoten, Farbe usw.
Abbildung 9 zeigt ein von "ANDD" erzeugtes Netzwerkdiagramm.

2.1.4.6 Diskussion

Mit "ANDD" können in relativ kurzer Zeit ansprechende Netzwerkdiagramme erzeugt werden. Dies ist vor allem deshalb möglich, weil das System pragmatische Designregeln verarbeiten kann, um so beispielsweise Zusammenhänge bzw. Gemeinsamkeiten mehrerer Knoten zu visualisieren.
Das System ist jedoch ausschließlich auf das Erzeugen dieser Netzpläne beschränkt.

2.1.4.7 Ein System zum Erzeugen von Bildern für Dokumentationen

Abbildung 10: Grundelemente aus der Datenbank. Aus: [Strothotte, Helms 95]

Abbildung 11: Aus den Grundelementen der Datenbank erzeugte Illustration. Aus: [Strothotte, Helms 95]

Mit dem System von [Strothotte, Helms 95] können Dokumentationen im

Personal Computer Bereich, die ausschließlich aus Bildern bzw. Bildfolgen bestehen, erzeugt werden.
Die Grafiken werden aus Bitmaps, die in einer Datenbank abgelegt sind, zusammengesetzt. Einige der Bitmaps repräsentieren Objekte, andere abstrakte grafische Symbole wie Pfeile oder Linien. Die Illustrationen bestehen aus diesen Grundbausteinen.
Ausgehend von einer komplexen Aktion, die visualisiert werden soll, dekomponiert der Grafikgenerator diese in primitive Aktionen.
Er wählt die zugehörigen Bitmaps aus der Datenbank und paßt deren Skalierung an. Sollten Objekte nicht wiedererkennbar sein, weil sie zu klein skaliert werden müssen, wählt er ein größer skaliertes Objekt und fügt eine Lupe hinzu.
Der Grafikgenerator legt danach das Layout der Präsentation fest.
Aus den Einträgen in der Datenbank (siehe Abbildung 10) generiert das System beispielsweise die Abbildung 11, eine Beschreibung für "Lösen sie die Schrauben an der Rückseite".

2.1.4.8 Diskussion

Dieses System erzeugt Illustrationen für Bedienungsanleitungen im PC-Bereich. Die Illustrationserstellung ist ein zielgerichteter Prozeß, ausgehend von einem komplexen Illustrationsauftrag erstellt das System selbstständig passende Illustrationen.
Da das System jedoch die Illustrationen als Montage von 2D-Bildern erstellt, fehlt ihm die nötige Flexibilität, diese Illustrationen unterschiedlichen Präsentationssituationen anzupassen. Für jede Perspektive, aus der ein Gerät betrachtet werden kann, müßte ein Bitmap in der Datenbank abgespeichert werden, ebenfalls für jede illustratorische Technik, die in dieser Situation möglich ist.

2.1.4.9 Automatische Synthese von Grafiken aus Objektbeschreibungen

In [Friedell 84] wird eine Sprache zur Objektbeschreibung vorgestellt, die dazu benutzt werden kann, 2D- und 3D-Grafiken in Echtzeit zu erstellen.
Diese Sprache wird im "View System" zur grafischen Datenbankanfrage eingesetzt, eine andere Anwendung ist ein System zur automatischen Erstellung

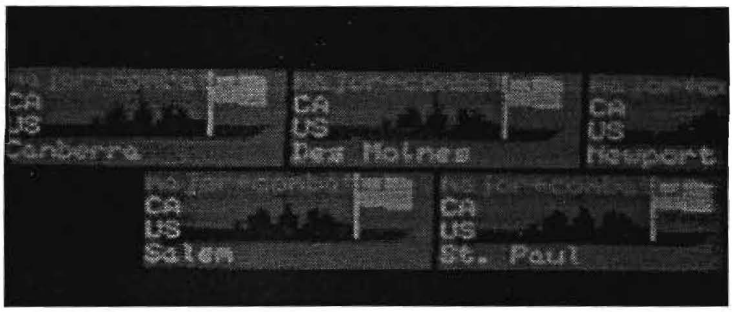

Abbildung 12: Verschiedene Schiffe, abgeleitet von demselben Prototyp. Aus: [Friedell 84]

von Hintergründen für komplexe 3D-Szenen (siehe Abbildung 12).
Die Sprache basiert auf der Manipulation prototypischer Vertreter einer Klasse. Die *Prototypen* werden durch *elementare Methoden* erzeugt. *Elementare Methoden* sind hierbei solche, die auf Punkte und Linien angewandt werden. Durch *Attribute* kann dann eine bestimmte Ausprägung des *Prototyps* spezifiziert werden, wobei für jedes *Attribut* eine Menge von Methoden, sogenannte *Modifier*, zuständig ist, die den "Umbau" des Prototyps bewirkt.
Ein Beispiel hierfür ist das Objekt "Ultralightflugzeug". Der *Prototyp* ist ein Flugzeug, das *Attribut* ist "Ultralight". Eine Menge von *Modifiern* für das *Attribut* "Ultralight" wandeln nun das prototypische Flugzeug in ein "Ultralightflugzeug" um.

2.1.4.10 Diskussion

Das System ermöglicht es, 3D-Illustrationen durch Manipulation prototypischer Vertreter zu erstellen. Der Generierungsprozeß ist sogar in Echtzeit möglich.
Das System stellt jedoch keine, speziell für die Illustrationserstellung für technische Betriebsanleitungen, geeigneten Techniken zur Verfügung. Diese müßten in der Beschreibungssprache programmiert werden.

2.1.4.11 Das System "IBIS"

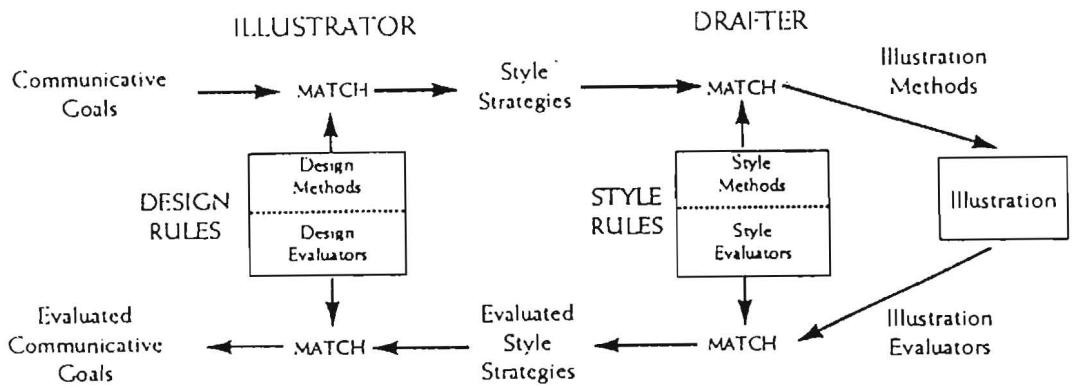

Abbildung 13: Der *generate-and-test* Ansatz bei "IBIS". Aus: [Feiner, Seligmann 91 (a)]

In [Feiner, Seligmann 91 (a)] wird ein automatischer regelbasierter Ansatz zur Erstellung von Illustrationen vorgestellt.
Als Illustrationen werden hierbei Bilder mit einer *kommunikativen Absicht* bezeichnet. Die Erstellung von Illustrationen wird als zielgerichteter Prozeß angesehen, mit dem Ziel, diese Absicht zu erfüllen.
"IBIS" basiert auf einem *generate-and-test* Verfahren (siehe Abbildung 13), das auf regelbasierten *Methoden* zur Erstellung der Illustrationen und *Evaluatoren* zur Beurteilung dieser Illustrationen beruht.
Eingaben für "IBIS" sind beispielsweise die Lage eines Objektes zeigen, die relative Lage zu anderen Objekten, die Eigenschaft eines Objektes, den Zustand und eine Veränderung des Zustandes.
Illustrationen werden erstellt indem die *kommunikative Absicht* vom System in *kommunikativen Ziele* umgesetzt wird. Für jedes dieser Ziele existiert zumindest eine *Designregel* in der Wissensbasis des Systems. Die *Designregeln* beinhalten eine Menge von *Stilstrategien*, die für deren Realisierung geeignet sind. Die *Stilstrategien* ihrerseits verursachen dann visuelle Effekte, wie beispielsweise ein Objekt durch stärkeres Beleuchten hervorzuheben.

Wie aus Abbildung 13 bereits hervorgeht, exisitieren zwei Arten von *Design Regeln*:

- *Methoden*
- *Evaluatoren*

Die *Methoden* bestimmen, wie ein Ziel erreicht werden soll, die *Evaluatoren* bewerten, wie gut es in der Illustration erreicht wurde.
Illustrationen werden in "IBIS" durch *Illustratoren* realisiert, die aus einer Menge von *kommunikativen Zielen* bestehen, für deren Erfüllung sie verantwortlich sind. Können die *Ziele* nicht in einer Illustration realisiert werden, kann der *Illustrator* neue *Illustratoren* erzeugen, denen er Untermengen seiner Ziele übergibt. Auf diese Weise können beispielsweise Bildsequenzen entstehen.

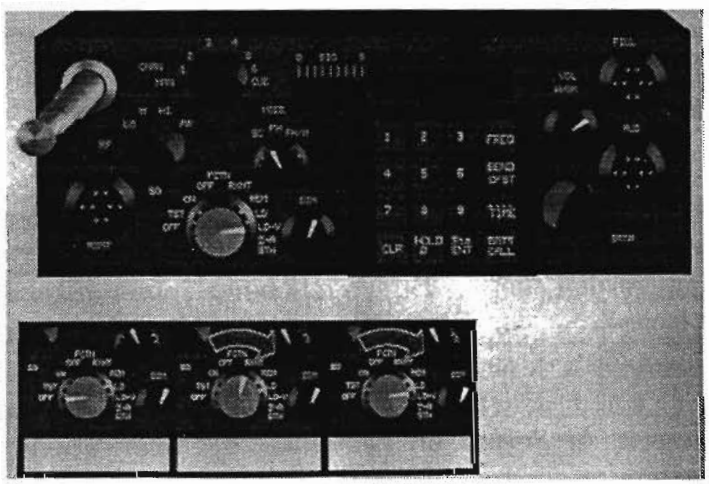

Abbildung 14: Eine von "IBIS" erzeugte Anleitung für ein Funkgerät. Aus: [Foley et al. 90]

Illustrationen enthalten *Illustrationsobjekte*, die speziell zum Zwecke der Illustration erstellt wurden. Sie stellen physikalische Objekte aus der Domänenwissensbasis dar, in der alle Farb-, Material-, geometrischen, abstrakten, etc. Eigenschaften der physikalischen Objekte abgelegt sind oder Metaobjekte, wie beispielsweise-Pfeile.

In einer Illustration bestehen Relationen zwischen *Illustrationsobjekten* und physikalischen Objekten. Physikalische Objekte können in einer Illustration durch mehrere *Illustrationsobjekte* dargestellt werden, die sie beispielsweise in verschiedenen Zuständen zeigen. Dies wird zu sogenannten "Ghosting Effekten," oder kurz "Ghost Images" benutzt.
Abbildung 14 zeigt eine von "IBIS" erzeugte Illustration für die Bedienung eines Funkgerätes mit Richtungspfeilen und Insets.

2.1.4.12 Diskussion

"IBIS" erstellt Illustrationen, die auf die einzelne Präsentationssituation zugeschnitten sind. Es handelt sich dabei um 3D-Grafiken, die in einem angeschlossenen Grafiksystem erzeugt werden. Somit ist auch die nötige Flexibilität bei der Illustrationserstellung, beispielsweise hinsichtlich der Perspektive, gewährleistet. "IBIS" stellt ebenfalls einige spezifische Techniken für die Erstellung von Illustrationen für technische Betriebsanleitungen zur Verfügung, wie Pfeile, Ghost Images, Lichter zum Hervorheben einzelner Objekte, Aufrißdarstellungen und Insets.

2.1.4.13 Die interaktive Version von "IBIS"

In [Seligmann, Feiner 93] wird das "IBIS" System hin zu interaktiven Ein-

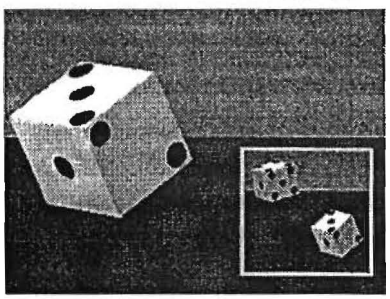

Abbildung 15: Die Abbildung zeigt eine Szene, bei der Hintergrundinformation durch ein Inset dargeboten wird. Aus: [Seligmann, Feiner 93]

griffsmöglichkeiten erweitert. In diesem Fall hat der Benutzer die Möglichkeit,

mit einer von "IBIS" erzeugten Illustration zu interagieren, beispielsweise durch Ändern des Betrachterstandpunktes.
Dazu werden *Metaregeln* hinzugefügt, die die Reihenfolge der *Evaluatoren* und *Methoden* bestimmen, die aktiviert werden sollen. *Meta Regeln* werden hierbei als permanent geltende Anforderungen angesehen.
Die *Metaregel*, die für Sichtbarkeit zuständig ist, hat höchste Priorität. So wird z.B. eine Aufrißdarstellung generiert, wenn Objekte beim Durchwandern der Szene verdeckt werden. Eine andere *Metaregel*, die für langsame Veränderungen der Illustration verantwortlich ist, sorgt dafür, daß dieser Aufriß durch langsames Ausblenden der verdeckenden Fläche entsteht.
Weiterhin kann der Benutzer Fragen an das System stellen, zum Beispiel über die Ziele, die in der aktuellen Illustration verfolgt wurden. In diesem Fall kann das System mit einer Einblendung eines kleineren "Hilfsbildes", einem sogenannten "Inset" antworten, in dem eine alternative Methode zur Zielerreichung gezeigt wird (siehe Abbildung 15).

2.1.4.14 Diskussion

Diese Erweiterung des Systems "IBIS" bietet dem Benutzer Interaktionsmöglichkeiten. Er kann nach Zielen des Systems fragen, oder eine Szene interaktiv "erforschen".
Der Benutzer hat jedoch keinen Einfluß darauf, wie ihm zusätzliche Information dargeboten wird. Nähert er sich beispielsweise in der Szene (siehe Abbildung 15) einem Objekt, erzeugt das System automatisch ein Inset, um mehr Hintergrundinformation zur Verfügung zu stellen. Hierbei wird keine Rücksicht darauf genommen, ob der Benutzer das will oder nicht. Er hat weiterhin keine Möglichkeit Größe und Position des Inset zu variieren oder dem System zu befehlen, das Inset ganz wegzulassen. Die Interaktion des Benutzers findet also nach dem automatischen Generierungsprozeß statt, auf ihn selbst hat der Benutzer jedoch weiterhin keinen Einfluß.

2.1.5 Resümee

Zusammenfassend kann gesagt werden, daß zwar kommerzielle Editoren speziell zur Erstellung von Illustrationen angepriesen werden. Eine Unterstützung des Benutzers bei der Illustrationserstellung, beispielsweise durch Vorschläge, ist jedoch nicht vorgesehen.

Demgegenüber erstellen vollautomatischen Systeme Illustrationen ohne das Mitwirken des Benutzers. Sie sind aber noch nicht soweit ausgereift, daß sie in jeder denkbaren Situation die passende Illustration erstellen können. Dies liegt vor allem an der komplexen Problematik der Illustrationserstellung im 3D-Bereich. Es werden jedoch interessante Anregungen geliefert, die zum Teil in dieser Arbeit aufgegriffen werden.
Inspirierend war vor allem die im System "IBIS" vorgestellten Illustrationstechniken und der hierbei verwendete *generate-and-test* Ansatz.
Halbautomatische Systeme unterstützen zwar den Benutzer, ein System speziell zur Erstellung von Illustrationen existiert jedoch noch nicht.

2.2 Technisches Zeichnen

Eng verwandt mit der Erstellung von Ilustrationen ist das technische Zeichnen. Das Ziel ist es, Zeichnungen für die Fertigung von Werkstücken zu erstellen. Es werden aber auch Zeichnungen erstellt, die zur Dokumentation, zum Zusammenbau oder zur Funktionsbeschreibung gedacht sind.
"*Die moderne Fertigung ist heute gekennzeichnet durch die weitgehende Arbeitsteilung. Die technische Zeichnung als Informationsträger ist dabei das Verständigungsmittel zwischen den einzelnen Abteilungen eines Werkes, ... in der technischen Zeichnung ist das räumliche Werkstück durch senkrechte Parallelprojektion in den notwendigen Ansichten in der Zeichenebene dargestellt. ... Ferner enthält die technische Zeichnung alle notwendigen Angaben ... so daß das Werkstück ohne Rückfragen hergestellt werden kann.*" (aus [Hoischen 88]).
Die in der vorliegenden Arbeit betrachteten Zeichnungen fallen gemäß der Begriffe aus dem Zeichnungs- und Stücklistenwesen (DIN 199 T1)[2] unter die folgenden Kategorien:

- Anordnungsplan: Technische Zeichnung, die die räumliche Lage von Gegenständen zueinander darstellt.
- Ergänzungszeichnung: eigenständige Zeichnung solcher Einzelheiten von Gegenständen, auf die in anderen Zeichnungen Bezug genommen wird.

[2]siehe [Geschke et al. 94]

- Gruppenzeichnung: maßstäbliche technische Zeichnung, die die räumliche Lage und die Form der zu einer Gruppe zusammengefaßten Teile darstellt.
- Montierte Zeichnung: Originalzeichnung, in die vorhandene oder getrennt erstellte Zeichnungen, Abbildungen, Texte eingesetzt sind.
- Schema: bei Zeichnungen deuten Wortkombinationen mit "-Schema-" auf eine stark abstrahierte oder symbolische Darstellung hin.
- Zusammenbauzeichnung: Technische Zeichnung zur Erläuterung von Zusammenbauvorgängen.

Die technische Zeichnung wird nach festen Regeln entworfen. Es existieren verschiedene festgelegte Ansichten:

- Vorderansicht
- Draufsicht
- Seitenansicht von links
- Seitenansicht von rechts
- Untersicht
- Rücksicht

Folgende DIN Normen sind zu beachten:

- DIN 30 Vereinfachte Darstellung in Zeichnungen
- DIN 6776 ISO - Normschrift
- DIN 6 Ansichten und Schnitte

Im Schnitt werden beispielsweise Hohlkörper dargestellt, um ihre innere Form zu zeigen. "*Man denkt sich bei der Schnittdarstellung einen Teil des Werkstückes weggeschnitten und zeichnet den übriggebliebenen Teil.*" (aus [Hoischen 88]). Dort wo der Schnitt ist, sind die Flächen schraffiert zu zeichnen. Die Schraffuren geben das Material des Werkstückes an. Es existieren verschiedene Schnittarten:

- Vollschnitt: die vordere Werkstückhälfte wurde herausgeschnitten.
- Halbschnitt: Ein Viertel des Hohlkörpers wurde herausgeschnitten.
- Teilschnitt: Nur ein Teil wird ausgeschnitten.
- Profilschnitt: die zugehörige Ansicht darf gedreht oder neben der Ansicht dargestellt werden.

Für eine Technik, die oft zu illustratorischen Zwecken benutzt wird, existieren ebenfalls Richtlinien: Explosionsdarstellungen werden als "*axonometrische Darstellungen von Gruppen, bei denen die Einzelteile einer Gruppe in Richtung der Koordinatenachsen auseinandergezogen angeordnet sind. Sie vermitteln einen 3D-Eindruck und erleichtern das Verständnis für das Zusammenwirken der einzelnen Teile (funktional und/oder lokal ...)*" (siehe [Geschke et al. 94]).

3 Grundkonzepte

In diesem Kapitel wird erklärt, welche Eingabedaten zum Erstellen von Illustrationen notwendig sind. Zwei zentrale Konzepte, *Weltzustände* und *illustratorische Szenen* werden vorgestellt und ihre Rolle bei der Grafikgenerierung wird erläutert. Die Begriffsbildung entspricht dabei der in [Rist, André 92], [André, Rist 93] und [Rist et al. 94].

3.1 Illustratorische Techniken

Illustratorische Techniken sind Techniken, die ein Illustrator dazu benutzt, um Sachverhalte darzustellen, die beispielsweise mit Fotos nicht oder nur sehr schwer darstellbar sind. Oftmals können Handlungen erst dann richtig ausgeführt werden, wenn einen Benutzer Mechanismen vor Augen geführt werden, die im Inneren des Gerätes verborgen sind, die er aber von außen betätigen muß.
Andererseits ist es mit stehenden Bildern problematisch Bewegungen zu zeigen. Es werden heute bereits Videos als Gebrauchsanweisung ausgeliefert. Dies ist jedoch nicht in allen Fällen sinnvoll oder möglich. Sollen solche Bewegungen in einer Zeichnung dargestellt werden, ist der Illustrator auf Tricks angewiesen, diese Bewegungen doch zu zeigen. Die Werkbank **TOPAS** stellt eine Vielzahl dieser als illustratorische Techniken bekannten Verfahren zur Verfügung.

3.2 Ausgangsdaten

Mit Hinblick auf die Inhalte von Illustrationen wird auf das Design von Illustrationen abgezielt, die Informationen über materielle Domänenobjekte, Weltzustände und Zustandsänderungen transportieren. Domänenobjekte sind dabei die realen Objekte, die zu einem Modell gehören, wie zum Beispiel einer Kaffeemaschine. Materielle Objekte sind Objekte, deren Repräsentation Spezifikationen geometrischer Objekteigenschaften enthält, vor allem der 3D Ausmaße.

3.2.1 3D-Objekte

Die 3D-Objekte werden im Grafiksystem als Drahtrahmenmodelle abgelegt (siehe [Fellner 88]). Zum Erstellen komplexer Modelle werden einfache Objekte zu komplexeren Objekten gruppiert.
Einfache Objekte sind in diesem Fall:

- Lamina: ein geschlossener Polygonzug
- Extruded-Lamina: extrudierter, entlang eines Vektors in die dritte Dimension "gezogener" Polygonzug
- Quader
- Loch-Quader
- Zylinder
- Loch-Zylinder
- Kugel
- Kopie: die Kopie eines Objektes

Die einfachen Objekte sind teilweise parametrisierbar. Es kann beispielsweise spezifiziert werden, aus wievielen Flächen eine Kugel oder ein Zylinder besteht etc. Folgende Beispiele zeigen, wie einfache, bzw. zusammengesetzte Objekte mit Hilfe von systemunabhängigen Makros definiert werden:

```
(create-primitive-object
  :NAME "oberteilkasten-1-r"
  :TYPE :LAMINA
  :VERTICES '((250.0 220.0 -80.0)
              (250.0 220.0 0.0)
              (250.0 290.0 0.0)
              (250.0 290.0 -80.0))
  :DIFFUSE-COLOR '(RENDER:RGB 0.8 0.8 0.8)  ; reflektierte Farbe
  :AMBIENT-COLOR '(RENDER:RGB 0.0 0.0 0.0)  ; Objekt-Leuchtfarbe
  :OPACITY 1.0                                    ; Transparenz
)
```

```
(create-compound-object
  :NAME "wasserauslauf-1"
  :OBJECT-LIST '(wasserauslaufummantelung-1
                 wasserauslaufoeffnung-1)
  :Merge t
  :vh '(0 0 1)
  :ou '(0 1 0)
  :PQUADER-PARENT "kaffeemaschine-1"
  :PQUADER-GROUP "oberteil-1"
)
```

Die Angaben ":vh" und ":ou" bestimmen die Ausrichtung des Objektes und werden zum Errechnen des umhüllenden *Objektquaders*, auch *Perspektivenquader* genannt, benötigt. Sie dienen zum Einstellen der Perspektive. Werden keine Angaben über die Ausrichtung des Objektes gemacht, ist die Unterstützung der Perspektivenwahl durch die Werkbank nicht möglich.
Über die Slots ":DIFFUSE-COLOR" ":AMBIENT-COLOR" ":OPACITY" können Farbattribute und Transparenz für die Objekte spezifiziert werden. Diese Angaben sind nötig, wenn aus dem Modell eine realistisch wirkende 3D-Darstellung erzeugt werden soll, beispielsweise durch Strahlenverfolgungsverfahren ("Ray Tracing")[3].
Zum Erstellen von Explosionsdarstellungen wird Wissen über den Zusammenbau der zu explodierenden Baugruppen benötigt (siehe Abschnitt 4.2). Dies wird durch eine 4-Tupel kodiert, bestehend aus *(Base-Object, Attached-Object, Side, No-Directions)* (siehe Abbildung 16). Das bedeutet: Das *Attached-Object* ist an der Seite: *Side* des *Base-Objectes* angefügt und läßt sich nicht in die Richtungen: *No-Directions* entfernen. Diese Spezifikationen implizieren eine Baumstruktur, die das zur Erstellung der Explosion benötigte Wissen enthält. Vorteilhaft ist dieses Verfahren speziell bei der Variantenkonstruktion, da sich hierbei die Zusammenbaustruktur einzelner Baugruppen nicht ändert, sondern nur ein spezielles Teil. Die Zusammenbauhierachie wäre weiterhin gültig.
In einer Wissensbasis (siehe [Heinsohn et al. 93]) wird eine Teil-Ganzes-Beziehung zwischen Domänenobjekten ("Part-of Hierarchie"), Art der Verbindungen zwischen den Objekten (trennbar, nicht trennbar) und Art der

[3]siehe [Fellner 88]

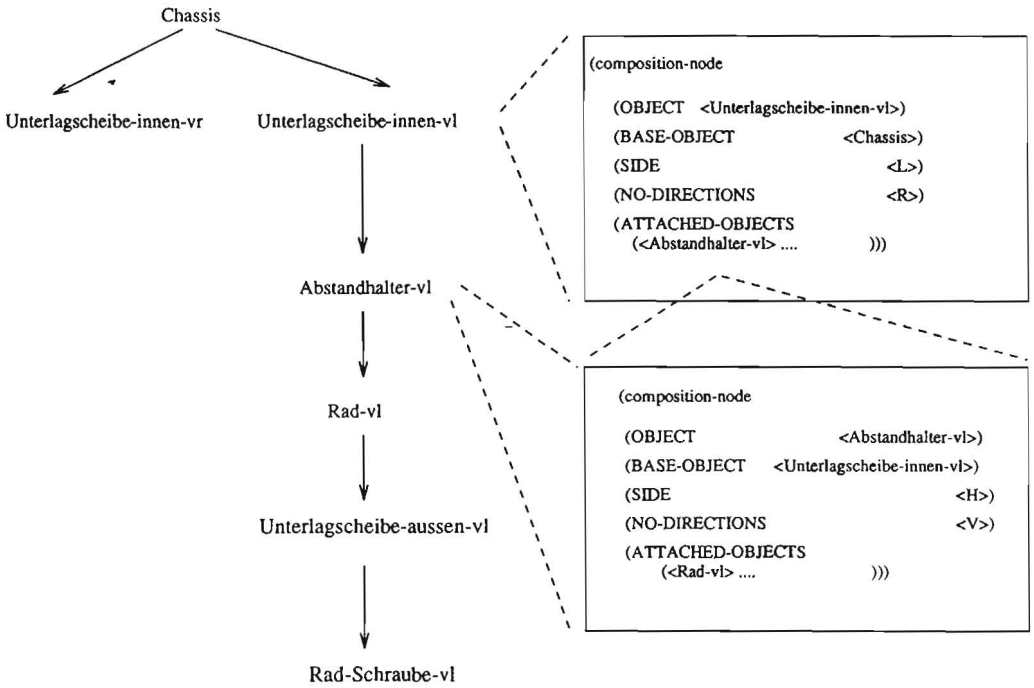

Abbildung 16: Zusammenbauhierarchie zum Generieren von explodierten Darstellungen.

Objekte selbst ("Has-Attribute Relation") abgelegt. Dies wird benötigt, wenn beispielsweise ein Fahrwerk explodiert oder ein Aufriß zum Sehen des Motors generiert werden soll. Damit nicht immer Teile eines solchen Konzeptes einzeln angesprochen werden müssen, wird dieses Wissen in der Wissensbasis abgelegt. Die "Has-Attribute" Relation beschreibt Eigenschaften, die das Objekt besitzt. Für die Werkbank ist das Wissen notwendig, ob ein Objekt als hohles oder massives Objekt zu klassifizieren ist. Diese Information wird benötigt, damit bei der Erstellung von Aufrißzeichnungen entschieden werden kann, ob ein Objekt durchbrochen wird oder nicht. Das Wissen über die Art der Verbindung wird benötigt, um zu wissen, ob eine Verbindung trennbar oder nicht trennbar ist. Nicht trennbare Verbindungen können beispielsweise nicht explodiert werden. Solche Verbindungen sind geklebt oder geschweißt und führen bei Versuchen sie doch zu trennen zur Zerstörung des Modells.

3.2.2 Weltzustände

Weltzustände (kurz WZ) beziehen sich konzeptuell auf Domänenobjekte, d.h. WZ sind bestimmt durch Eigenschaften, die Domänenobjekte zu einer gewissen Zeit besitzen. WZ werden dargestellt, als eine endliche Menge von relevanten Objekten zusammen mit der Spezifikation der relevanten Eigenschaften. Ein Übergang von einem WZ zum nächsten wird Änderung (englisch: "change") genannt. Alle Weltzustände werden von einem initialen Weltzustand, dem sogenannten "Ur-Weltzustand" abgeleitet. Sämtliche Schalter und Regler sind in ihren Ausgangspositionen oder den sogenannten "Defaultzuständen". Alle Veränderungen in der Welt sind Veränderungen gegenüber diesem Zustand.
Weltzustände beschreiben also die Zustände, die mit einem Modell, beispielsweise einer Kaffemaschine, in der realen Welt erreichbar sind. Eine Methode sämtliche Möglichkeiten zu beschreiben wäre das Anlegen eines Objektes für jeden dieser Zustände. Dies würde für einen Klappdeckel bedeuten, daß eine große Anzahl von Kopien von ihm existieren müßten, die Zuständen wie geschlossen, halb-offen, sehr weit geöffnet, offen, etc. entsprechen. Diese Vorgehensweise erscheint nicht sinnvoll. Sinnvoller ist es, das Wissen über räumliche Veränderungen beispielsweise durch Trajektorien zu beschreiben, entlang derer sich ein Objekt bewegt.
Durch folgende Struktur kann solch eine Bewegung beschrieben werden:

```
(create-movement :motionname "zuklappen"
 :objectname "oberteilkasten-1-deckel"
 :motiontype ':trajectory
 :orienttrajectory '((0 40 -40) (0 40 -30) (0 30 -10)
                     (0 10 0) (0 0 0))
 :alignto '(0 1 0)
 :axis 'x
 :amount 90
 :centerpoint '(125 290 -80)
 )
```

Aus den unter ":orienttrajectory" angegebenen Punkte wird in **TOPAS** eine Bézier-Kurve[4] errechnet, die die Trajektorie beschreibt. Somit sind durch

[4] siehe beispielsweise [Fellner 88]

diese Struktur alle Daten gegeben, um Objekte zu bewegen. Folglich können aus den gleichen Daten in **TOPAS** Trajektorienpfeile generiert werden und in BETTY (siehe [Butz 94]) Animationen.
Wird nun das Wissen über Objekte in bestimmten Weltzuständen mit den entsprechenden Trajektorien identifiziert, kann die Trajektorie als Übergangsfunktion angesehen werden, die ein Objekt von einer räumlichen Lage in eine andere überführt und dabei unendlich viele "Zwischen-Weltzustände" durchläuft.
In **TOPAS** liegt das Hauptaugenmerk auf der Veränderung der räumlich Lage, für andere Veränderungen gelten äquivalente Überlegungen und können durch Hinzufügen der Übergangsfunktion ebenfalls berücksichtigt werden. Jedes Objekt ist folglich zu einem bestimmten Zeitpunkt durch einen Weltzustand komplett beschrieben.

3.2.3 Pseudoobjekte

Pseudoobjekte sind Objekte, die zu verschiedenen illustratorischen Zwecken benutzt werden. Sie sind genauso repräsentiert wie materielle Domänenobjekte. Sie haben eine 3D-Geometrie und Materialeigenschaften. Obwohl Pseudoobjekte nicht zu einer Domäne dazugehören, können sie virtuell zu Anordnungen von Domänenobjekten hinzugefügt werden. Bei technischen Illustrationen kann ein Pseudoobjekt benutzt werden, als

- Ersatz für die reguläre Repräsentation eines Domänenobjektes zum Zwecke der Abstraktion von geometrischen oder materiellen Objekteigenschaften.
- Duplikat eines Domänenobjektes, zum Beispiel zum Erzeugen von "ghost" Effekten. Dies bedeutet, daß das Objekt in einem Bild in mehreren verschiedenen räumlichen Positionen dargestellt wird.
- Defaultdomänenobjekte sind Objekte, die in einer Domäne nicht explizit modelliert sind, deren Existenz jedoch als gesichert angenommen werden kann. Beispiel hierfür sind Böden und Wände, die als Hintergrundobjekte dienen können.
- Materialisierung eines abstrakten Domänenkonzeptes. Beispiele sind 3D-Pfeile, die eine Trajektorie zur Änderung der räumlichen Anordnung darstellen sollen, oder Symbole für Licht oder Ton.

3.2.4 Illustratorische Szenen

Illustrationen beziehen sich vor allem auf räumliche Objektanordnungen und visuelle Objekteigenschaften. Von einem syntaktischen Gesichtspunkt aus betrachtet, wird eine Illustration als ein Bild oder eine Menge von Bildern bezeichnet. Ein Bild besteht aus einem Rahmen und einer Menge von Abbildungen innerhalb dieses Rahmens, wobei eine Abbildung ein 2D-Objekt ist. Es wird durch eine festgelegte Region und eine Menge von Attributen, wie Form und Aussehen beschrieben. Der Zusammenhang zwischen Objekten, die illustriert werden sollen und Bildern, sind die Abbildungen, die durch Projektion von 3D-Objektanordnungen auf ebene 2D-Regionen entstehen. Es ist wichtig, dabei nicht zu vergessen, daß ein Objekt immer nur in einem bestimmten WZ projiziert wird. Illustrationen sind jedoch mehr als reine Projektionen von Objekten in bestimmten WZ. Um eine möglichst große Vielzahl von Illustrationen erzeugen zu können, erlauben wir die Existenz von Pseudoobjekten (siehe 3.2.3) und führen deshalb das Konzept der illustratorischen Szene ein.
Bilder, die aus der Projektion von WZ entstehen, sind vergleichbar mit Fotografien, die einen Schnappschuß einer bestimmten Objektanordnung zeigen. Der Begriff der illustratorischen Szene (kurz IS) wird eingeführt, um Bilder zu erzeugen, die folgendermaßen aussehen:

- Bilder, die virtuell modifizierte WZ zeigen. Anstatt einen WZ zu projizieren, projizieren wir eine IS. Die IS unterscheidet sich von den WZ dahingehend, daß einige Änderungen zu illustratorischen Zwecken gemacht werden können. Es können zum Beispiel Objekrepräsentationen ersetzt werden, Defaultobjekte können hinzugefügt werden oder es können Objektteile neu zusammengefügt werden.

- Bilder, die mehrere WZ darstellen. In diesem Falle entsprechen einer IS mehrere WZ. Wenn zum Beispiel eine Änderung der räumlichen Lage eines Objektes in einem einzelnen Bild dargestellt werden soll, dann kann der für die Illustration zuständige IS mehrere Objektkopien in verschiedenen räumlichen Lagen und damit WZ, enthalten.

Eine IS wird durch eine Menge von Objekten, die als WZ im 3D-Raum liegen, gebildet. Während Änderungen von einem WZ zum anderen durch physikalische Einwirkung verursacht werden, gehen IS auseinander durch die Anwendung illustratorischer Techniken hervor. Einige dieser Techniken ähneln

physikalischen Einwirkungen, zum Beispiel die Neuordnung der räumlichen Objektzusammenstellung. Andere, wie zum Beispiel das Hizufügen von Pseudoobjekten haben keine direkten Entsprechungen in der Domäne.
Der Zusammenhang zwischen WZ, IS, Abbildungen, Bildern und Illustrationen ist im einfachsten Fall wie folgt:
Es gibt einen einzigen WZ, der mit einem strukturell gleichartigen IS verknüpft wird. Eine Abbildung wird aus einer IS mittels einer Projektion gewonnen. Wird die Abbildung in ein Bild eingefügt, ist die Illustration fertig. Zur Vereinfachung wird in den folgenden Kapiteln für diese Konstellation der Begriff "Szene" benutzt und für die in der illustratorischen Szene enthaltenen Objekte "Szenenobjekte".
Kompliziertere Zusammenhänge treten in folgenden Fällen auf:

- Ein einzelner WZ wird durch mehrere IS dargestellt.
- Eine einzelne IS enthält mehrere WZ.
- Verschiedene Abbildungen werden aus einem einzelnen IS erhalten.
- Dieselbe Abbildung erscheint in verschiedenen Bildern.
- Ein Bild enthält dieselbe Abbildung mehrere Male.
- Eine Illustration besteht aus einer Bildsequenz.
- Ein Bild erscheint als ein "Inset" eines übergeordneten Bildes.

Abbildung 17 zum Beispiel zeigt eine Zusammenstellung aus zwei Bildern. Jedes Bild beinhaltet eine Abbildung einer IS. Jede IS gehört zu verschiedenen WZ. Die Abbildung des sogenannten "Inset", des eingeblendeten Bildes in der linken oberen Ecke, vergegenwärtigt das Öffnen des Deckels mit Hilfe von sogenannten "Ghost Images", gestrichelten Kopien des Deckels in verschiedenen Positionen. In der Abbildung wird der Deckel in mehreren aufeinanderfolgenden WZ dargestellt. Im Gegensatz dazu wird im Hauptbild dieselbe Sequenz von WZ durch ein Pseudoobjekt, nämlich einem 3D-Pfeil dargestellt, der die Bewegungstrajektorie des Deckels zeigt. Ein weiteres Pseudoobjekt ist der Boden, der hinzugefügt wurde, um den Eindruck räumlicher Tiefe zu vermitteln.

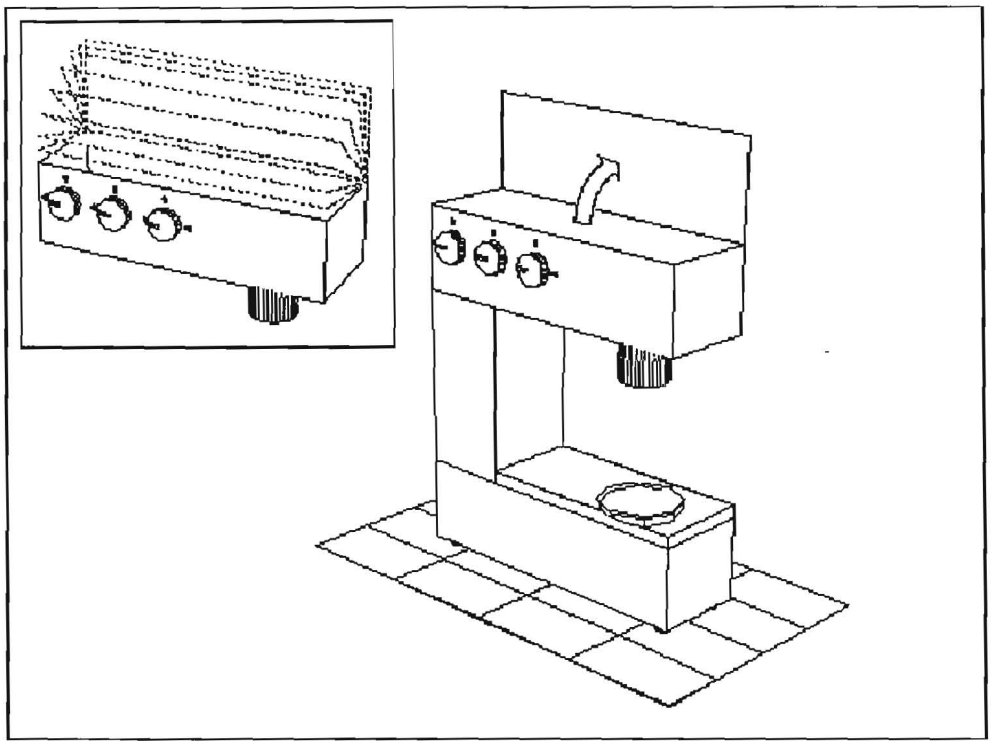

Abbildung 17: Kombination zwischen IS und WZ.

Diese Pseudoobjekte gehören zur illustratorischen Szene, da sie in der realen Welt nicht vorhanden sind und ausschließlich illustratorischen Zwecken dienen. Neben den Pseudoobjekten gehören aber auch die realen Objekte zur illustratorischen Szene. Schließlich ist sie es, die projiziert wird und die realen 3D-Objekte werden durch diesen Prozeß erst sichtbar gemacht.
Daraus folgt, daß die illustratorische Szene ebenfalls den Projektionsprozeß beschreiben muß. Sie beinhaltet den Projektionstyp, wie Drahtrahmenmodell oder "Hidden line"(bei dem die verdeckten Seiten des Modells entfernt wurden) oder realistisch wirkende 3D-Darstellung (durch "Rendering", "Ray Tracing", etc.), weiterhin die Kameraspezifikation (Brennweite, Kameraentfernung, etc.) und das Beleuchtungsmodell[5]. Ferner sind Verweise auf die Weltzustände, die durch die illustratorische Szene gezeigt werden, vorhanden.

[5]siehe [Foley et al. 90]

4 Operationalisierung von Illustrationstechniken

Im vorliegenden Kapitel werden Techniken vorgestellt, die zum Zweck der Illustration benutzt werden können. Der Schwerpunkt liegt dabei auf 3D-Techniken. Weiterhin wird jeweils ein Algorithmus vorgestellt, mit dem die entsprechende Technik implementiert werden kann (siehe hierzu auch [Schneider 95 a]).

4.1 Aufrißtechnik

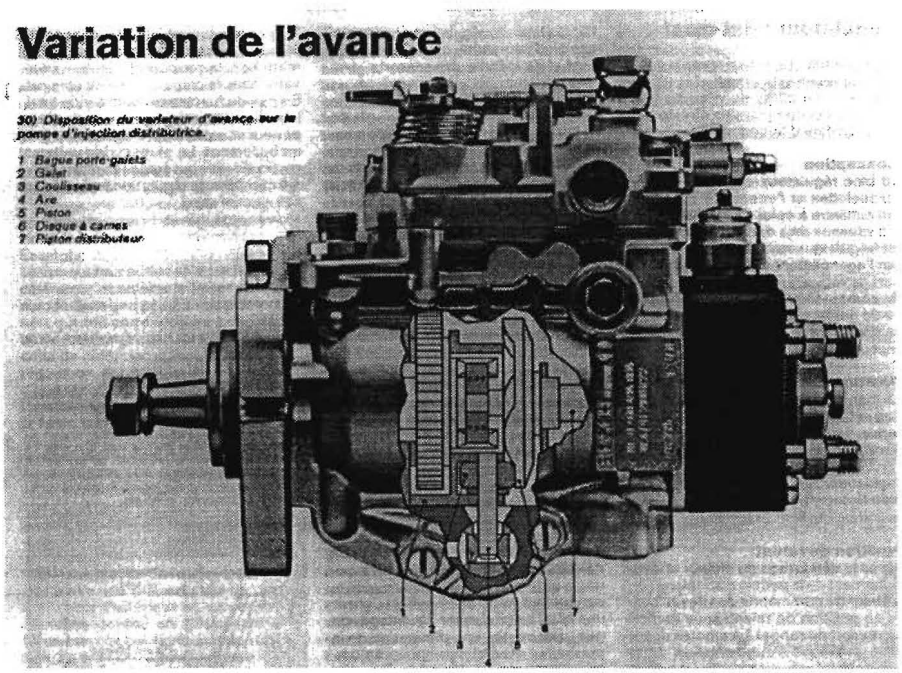

Abbildung 18: Aufrißdarstellung eines Motors. Aus: [Bosch 85 a].

Bei dieser Technik werden durch Objekte verdeckte Teile eines Modells sichtbar gemacht (siehe Abbildung 18). Die Aufrißtechnik (englisch: Cutaway) kann vielfältig eingesetzt werden. Neben dem Ziel ein Objekt sichtbar zu

machen, kann sie auch als Mittel eingesetzt werden, um z.B. Hohlräume zu zeigen.
Die Teile bleiben, anders als bei Explosionstechniken (siehe Abschnitt 4.2), an ihrem Platz und können daher im Zusammenspiel mit anderen Teilen betrachtet werden [Krüger et al. 91].
Gegeben ist eine Objektkonstellation, die aus einer bestimmten Perspektive betrachtet wird. Das Ziel ist es, ein vom Benutzer spezifiziertes Objekt sichtbar zu machen. Das wird erreicht, indem verdeckende Teile durchbrochen oder entfernt werden.
Die Technik wurde erfolgreich ausgeführt, wenn ein Betrachter das verdeckte Objekt sieht und die durch den Durchbruch entstandenen Löcher als virtuelle, also durch die Technik entstandene, Objekte erkennt.

Vorgehensweise

In [Feiner, Seligmann 91 (b)] wird eine "Z-Puffer" basierte Vorgehensweise zur Erstellung von Aufrißdarstellungen vorgestellt. Der Aufriß wird dadurch erreicht, daß zweimal gerendert wird. Beim ersten Renderprozeß wird allein das Objekt, das sichtbar werden soll, gerendert. Beim zweiten Renderprozeß wird der "Bildschirmspeicher" ausgeschaltet und eine "Cutawaymaske" in den Z-Puffer gerendert. Diese Maske umschließt das Objekt, das sichtbar werden soll, vollständig (beispielsweise ein umhüllender Quader) und wird auf eine z-Position gesetzt, näher als alle übrigen Objekte. Danach wird der Bildschirmspeicher eingeschaltet und alle verdeckenden Objekte werden gerendert. Dadurch werden alle Objekte außerhalb der Maske gezeichnet.
Diese Vorgehensweise hat jedoch den entscheidenden Nachteil, daß die Löcher erst beim Renderprozeß entstehen und in der 3D-Welt weder vorhanden noch repräsentiert sind. Sie exisitieren ausschließlich auf Bitmapebene.
Eine Vorgehensweise, die Technik am 3D-Objekt auszuführen und die verdeckenden Flächen durch neue Flächen, die ein Loch enthalten, zu ersetzen, scheint daher sinnvoller. Die Löcher können beispielsweise ebenfalls als Objekte modelliert werden. Referenzen auf diese Objekte sind dadurch möglich, eine Annotation der Löcher kann ebenfalls erfolgen.
Gegeben ist die Objektkonstellation, das Objekt, das sichtbar werden soll, das *Zielobjekt* und die Perspektive. Der Aufriß läßt sich nun in zwei Teilprobleme zerlegen:

- Bestimme die das *Zielobjekt* verdeckenden Flächen.

- Beseitige die Verdeckung durch Weglassen oder durch Ersetzen der verdeckenden Fläche durch Fläche(n) mit Loch.

Bestimmung der verdeckenden Flächen

Zur Lösung des ersten Problems wird ein schneller Algorithmus benötigt, der die verdeckenden Flächen bestimmt. Dadurch ist zu diesem Zeitpunkt schon eine Aussage über die ungefähre Berechnungsdauer des gesamten Algorithmus möglich. Fällt diese Schätzung zu negativ aus, kann ein alternatives Verfahren, oder eine günstigere Perspektive gewählt werden.
Eine Beschleunigung des Verfahrens ist durch Heuristiken und Vereinfachungen möglich.
Wird der umhüllende Quader als Objektvereinfachung benutzt, kann sehr schnell auf Verdeckung getestet werden. Die Fälle, in denen der Test positiv verläuft, müssen nochmals getrennt untersucht werden.
Um eine schnelle Näherungslösung zu erhalten, wird in **TOPAS** statt die exakte Überlappung des Zielobjektes mit den anderen Objekten zu berechnen, ein Strahlenverfolgungsverfahren benutzt, das anhand ausgezeichneter Teststrahlen die verdeckenden Flächen ermittelt.

Beseitigung der Verdeckung

Die Verdeckung soll durch "Stanzen von Löchern" in verdeckende Objekte beseitigt werden. In Fällen, in denen dies nicht sinnvoll ist, wird das verdeckende Objekt komplett beseitigt.
Solche Objekte sind "kleine" Objekte, wie zum Beispiel Schrauben, Unterlagscheiben, etc. und Objekte, die zwar aus vielen Einzelteilen bestehen, die aber intuitiv als Ganzes betrachtet werden, wie zum Beispiel ein Motor.
Im ersten Fall wird auf ein Durchstanzen verzichtet, weil von den Kleinteilen nur noch ein oder gar mehrere Fragmente übrigbleiben würden, was beim Betrachten der Zeichnung zu Verwirrungen führen kann.
Baugruppen, wie zum Beispiel ein Motor, werden ebenfalls weggelassen, wenn sie ein dahinterliegendes Objekt verdecken. Ein Loch durch eine solche Baugruppe würde nicht nur sehr viel Rechenzeit beanspruchen, sondern vor allem den Betrachter vom *Zielobjekt* ablenken und ihm den Blick für das Wesentliche verstellen.

Das Durchstanzen von Objekten läßt sich auf die Subtraktion von Polygonzügen zurückführen. Das *Zielobjekt* wird nacheinander auf alle verdeckenden Objekte projiziert und anschließend wird die Differenz zwischen verdeckendem Objekt und *Zielobjekt* betrachtet.
Durch eine Vereinfachung des *Zielobjektes* kann diese Phase ebenfalls beschleunigt werden. Es wird statt der Silhouette des *Zielobjektes* eine quadratische Umrandung, ähnlich einem Bilderrahmen, betrachtet. Die Umrandung wird für Projektion und Differenzbetrachtung benutzt, wodurch sich der Rechenaufwand für diese Operationen erheblich vermindert.
Zur Vereinfachung wird davon ausgegangen, daß das Modell keine verschränkten oder gekrümmten Flächen enthält. Diese müssen als aus mehreren Flächen zusammengesetzt modelliert werden.

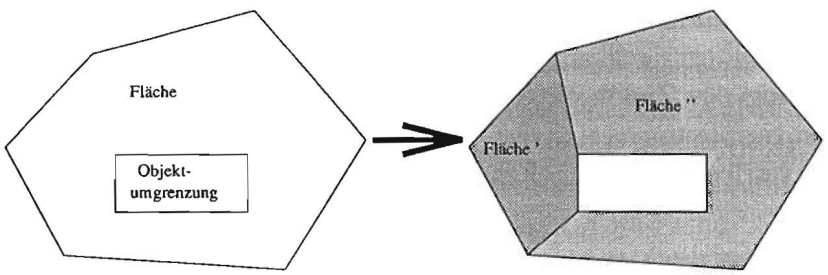

Abbildung 19: Die ursprüngliche Fläche wird durch zwei Flächen ersetzt, die zusammen die gewünschte "Lochfläche" ergeben.

Bei der Projektion können drei unterschiedlich Fälle auftreten:

1. Die Projektion liegt vollständig in der Fläche (vergleiche Abb. 19).
2. Die Projektion liegt teilweise in der Fläche (vergleiche Abb. 20).
3. Die Projektion ist größer als die Fläche (vergleiche Abb. 21).

Ein letzter Schritt dient dazu, das Loch als illustratorisches Stilmittel zu verdeutlichen. Der Betrachter muß zu jedem Zeitpunkt zwischen realen, zum Modell gehörenden und virtuellen Löchern unterscheiden können. Dies wird durch Zacken der Lochränder erreicht.

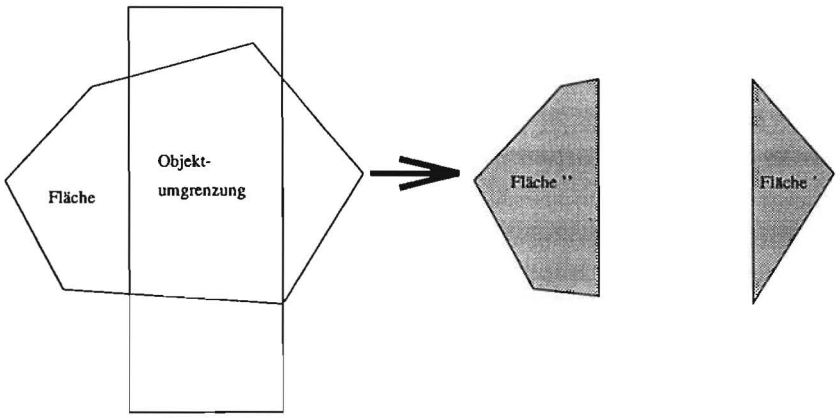

Abbildung 20: Im Fall 2 kann die Fläche in eine oder mehrere Teilflächen zerfallen.

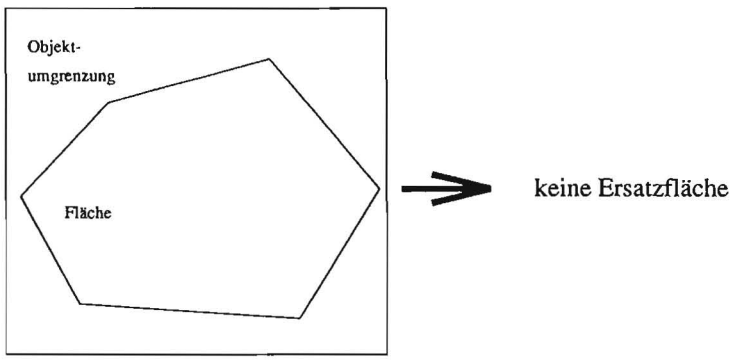

Abbildung 21: Im Fall 3 wird die Fläche ausgeblendet.

Implementierung

Die ausgezeichneten Teststrahlen werden vom Betrachtungsstandpunkt aus (Augpunkt) auf das *Zielobjekt* gerichtet und die getroffenen Flächen werden in einer Liste, dem *Aufrißplan*, gesammelt.

Die eingangs erwähnte Vereinfachung durch den umhüllenden Quader wird beim Schnitt mit massiven Objekten benutzt, da diese als aus mehreren Teilen zusammengesetzt modelliert sind. Die Information, daß es sich um massive Objekte handelt, wird aus der Wissensbasis (siehe Abschnitt 3.2) entnommen. Verstellen massive Objekte den Weg, werden sie später ausgeblendet.
Schließlich werden die Flächen des *Aufrißplanes* nach Entfernung zum *Zielobjekt* sortiert, um später eine geordnete Ersetzung der Flächen zu ermöglichen. Der folgende Pseudocode beschreibt den Algorithmus.

```
(1)  FOR Alle Objekte in Szene
(2)    DO
(3)      IF Objekt massiv
(4)        THEN
(5)          IF Schnitt mit zugehörigem PQuader?
(6)            THEN Objekt in Aufrißplan aufnehmen
(7)      FOR Alle Flächen von Objekt
(8)        DO
(9)          IF Schnitt zwischen Fläche und Teststrahlen?
(10)           THEN Fläche verdeckt
(11)                Objekt in Aufrißplan aufnehmen
```

Nachdem der *Aufrißplan* erstellt wurde, müssen die *Ersatzobjekte* angelegt werden. Das wird erreicht, indem Objektumgrenzung, zu durchbrechendes Objekt sowie Blickvektor in eine Koordinatenebene transformiert werden. Hier kann die Projektion der Umgrenzung entlang des Blickvektors auf die Objektfläche und die anschließende Ploygonbetrachtung, die bei der Differenzbildung nötig ist, sehr schnell berechnet werden. Die *Ersatzobjekte* werden gezackt, zurücktransformiert, der verdeckenden Fläche zugewiesen und statt ihrer angezeigt.
Der folgende Algorithmus beschreibt die genaue Vorgehensweise.

```
(1)  FOR Alle Elemente aus Aufrißplan
(2)    DO
(3)      IF Element massives Objekt?
(4)        THEN Kein Ersatzobjekt erstellen
(5)        ELSE
```

(6) Transformiere Fläche, Objektumgrenzung und Sehstrahl in x/y-Ebene
(7) Projiziere Objektumgrenzung enlang des Sehstrahls in die Fläche
(8) **CASE**
(9) **OF** Umgrenzung vollständig in Fläche?
(10) **THEN** Erstelle zwei *Ersatzobjekte*
(11) **OF** Umgrenzung schneidet Fläche?
(12) **THEN** Erstelle *Ersatzobjekt(e)*
(13) **OF** Umgrenzung überdeckt Fläche vollständig?
(14) **THEN** Erstelle kein Objekt
(15) Zacke Lochinnenseite zwischen Objektumgrenzung und expandierter Objektumgrenzung
(16) Weise Element das Ergebnis der Betrachtung zu

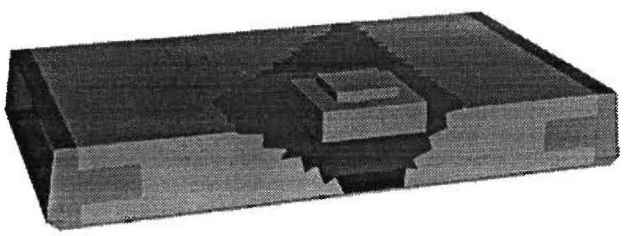

Abbildung 22: Durch den Aufriß wird der Transformator sichtbar.

In einem letzten Schritt werden dann die verdeckenden Objekte durch die *Ersatzobjekte* in der Grafik ausgetauscht. Kein *Ersatzobjekt* bedeutet dabei, daß das verdeckende Objekt ausgeblendet wird.

Lochgröße, Zackenhöhe und Zackenanzahl sind parametrisierbar. In der vorliegenden Implementierung werden die Defaultwerte großzügig gewählt, um sicherzustellen, daß das *Zielobjekt* auf jeden Fall vollständig durch das Loch sichtbar ist.
Das Beispiel 22 dient dazu, den Benutzer auf die Lage des Transformators hinzuweisen. Durch das Loch in der Abdeckung des Modems ist der Transformator vollständig zu sehen.

4.2 Explosionstechniken

Beim Erstellen technischer Bedienungsanleitungen ist häufig das Problem zu lösen, den Zusammenbau kompliziert zusammengesetzter Baugruppen zu erklären, um einem Betrachter das Zerlegen und Zusammensetzen dieser Gruppen zu ermöglichen. Techniken, die diese Aufgabe meistern können, sind zum Beispiel die Explosionstechniken (siehe [Krüger et al. 91]). Grundsätzlich lassen sich diese Techniken in drei Gruppen einteilen:

- Separation
- Isolation
- Explosion.

Separation

Die Separation ist die Grundtechnik. Zwei miteinander verbundene Teile werden räumlich getrennt, beispielsweise um die grundsätzliche Zerlegbarkeit einer Verbindung zu demonstrieren.
Zur Beschreibung der Technik wird im folgenden Teil zwischen *Separationsteil* und *separiertem Objekt* unterschieden.
Das *Separationsteil* ist das Teil, von dem die zu *separierenden Objekte* abgetrennt werden. Das *Separationsteil* muß vor der Separation bekannt sein. Die zu *separierenden Objekte* werden vom System entsprechend der Zusammenbauinformation errechnet.
Sofern die Verbindung zwischen den beiden Teilen lösbar ist, werden dann die zu *separierenden Objekte* als Block verschoben und eine Abtrennung vom *Separationsteil* ist erreicht.

Die relative Ausrichtung der verschobenen Teile zueinander wird durch die Separation nicht verändert (siehe Abbildung 23).

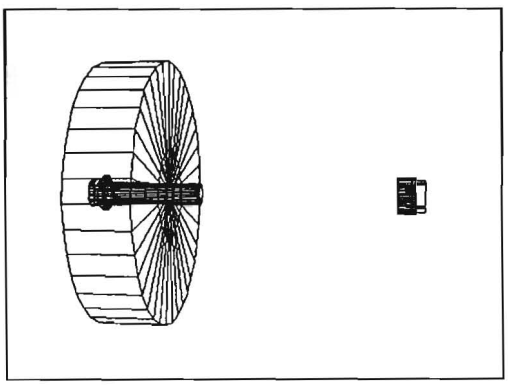

Abbildung 23: In dieser Illustration wurde das Rad vom restlichen Fahrwerk abgetrennt.

Isolation

Die Isolationstechnik wird verwandt, um ein Objekt einzeln zu zeigen, also von den restlichen Teilen einer Objektgruppe zu isolieren.
Im Gegensatz zu der Separation wird bei dieser Variante nicht nur die Objektgruppe abgetrennt. Das zu isolierende Teil wird komplett aus seiner Umgebung herausgelöst und kann nun von allen Seiten genau betrachtet und mit anderen, gleichartigen Objekten verglichen werden.
Die Technik wird häufig dazu benutzt, um zu zeigen, wie eine Baugruppe zerlegt werden muß, um ein Verschleißteil auszutauschen.
Die einfachste Möglichkeit diese Technik zu realisieren, ist die Rückführung auf die Separation. Bei der Isolation wird zu diesem Zweck zweimal separiert. Zuerst wird das *Isolationsteil* separiert, danach wird vom *Isolationsteil* selbst separiert, falls nötig. Nach Bekanntgabe des *Isolationsteils* wird das zur Ausführung der Technik benötigte Wissen aus der Zusammenbauinformation vom System ermittelt und die Isolation ausgeführt.

Sind die Verbindungen zwischen Isolationsteil und den anderen Teilen nicht zerstörungsfrei lösbar, wird entweder nur einmal oder überhaupt nicht separiert (siehe Abbildung 24).

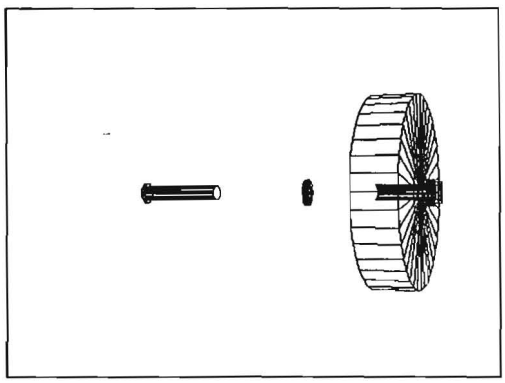

Abbildung 24: Dieses Beipiel zeigt die Isolation der Unterlagscheibe.

Explosion

Die Explosion dient dazu, einem Betrachter die vollständige Zerlegung einer Baugruppe vor Augen zu führen. Folglich werden alle Objekte dieser Baugruppe isoliert. Hiermit ist der Zusammenhang zwischen den drei Explosionstechniken hergestellt. In einer explodierten Darstellung erhält ein Betrachter sämtliche Informationen über den Zusammenbau einer Baugruppe. Er sieht, welche Verbindungen lösbar sind, wie die einzelnen Teile zusammenarbeiten und welche Funktion sie ausüben.
Wird bei der Isolation zweimal separiert, geschieht dies bei der Explosion höchstens: Anzahl {Teile der Baugruppe} mal.
Die folgenden Betrachtungen beschränken sich auf die Explosion, da die beiden anderen Techniken als Spezialfälle der Explosion betrachtet werden können (siehe Abbildung 25).

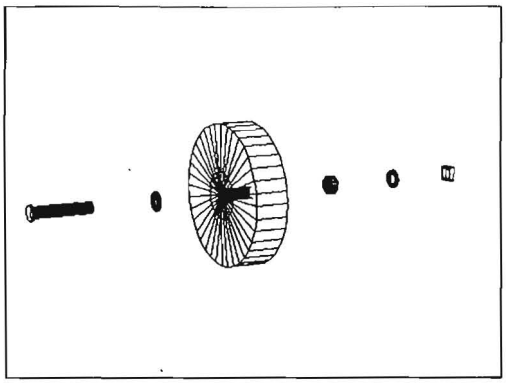

Abbildung 25: Explosion des gesamten Fahrwerks.

Vorgehensweise

Zuerst müssen die Einflußfaktoren auf Explosionsdarstellungen untersucht werden.
Zum Generieren einer Explosionsdarstellung genügt es nicht allein zu wissen, wie eine Objektgruppe zusammengebaut ist. Um eine realistische und aussagekräftige Darstellung der explodierten Baugruppe zu erreichen, müssen folgende Probleme zufriedenstellend gelöst werden:

- in welche Richtung soll ein Objekt verschoben werden?
- auf welcher Bahn soll verschoben werden?
- um welchen Betrag soll es verschoben werden?
- aus welcher Perspektive soll die neue Objektkonstellation gezeigt werden?

Explosionsrichtung

Die Richtung, in die die Objekte verschoben werden sollen, muß zuerst ermittelt werden. Um dieses Problem zu lösen, wurden Reparaturanleitungen untersucht, die von Hand erstellte Explosionszeichnungen enthalten, wie beispielsweise eine Anleitung zum Bedienen einer Küchenmaschine (siehe Abbildung 26).
In den meisten Fällen wird eine Baugruppe so explodiert, wie sie zusammengesetzt wurde und auch wieder auseinandergebaut werden muß. Wie in

Abschnitt 3.2 erklärt, ist diese Information bereits in den geometrischen Modellen kodiert.

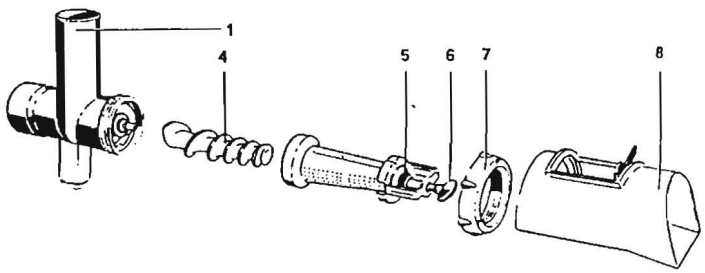

Abbildung 26: Explosionsdarstellung eines Fruchtpressenvorsatzes. Aus: [Braun].

Eine Heuristik, die die Explosionsrichtung bestimmt, verfolgt zwei Ziele:

- Bewege das zu explodierende Objekt auf der Flächennormalen seiner Anfügeseite von seinem Basisteil weg, falls möglich.
- Versuche dabei mit möglichst wenigen Richtungsänderungen auszukommen.

Explosionsbahn

Für die Wahl der Explosionsbahn existieren mehrere Möglichkeiten:

- gerade Bahn
- gekrümmte Bahn
- zackige Bahn

Die in technischen Dokumenten üblichste Bahn ist die gerade Bahn (siehe Abschnitt 2.2). In den meisten Explosionszeichnungen wird auf einer solchen Bahn explodiert. Die Objekte sind übersichtlich angeordnet und ermöglichen dem Betrachter ein einfaches Nachvollziehen des Auseinander-, bzw. Zusammenbaues.

Gekrümmte und gezackte Bahnen sind oftmals nicht so übersichtlich wie gerade Bahnen, sie ermöglichen andererseits eine platzsparendere Anordnung.

Explosionsbetrag

Unter dem Explosionsbetrag soll der Abstand verstanden werden, der nach Ausführung der Explosion zwischen einem Anfügeteil und seinem Basisteil liegt. Er läßt sich in zwei Komponenten zerlegen:

- *d-sep-3d*
  Dieser Betrag garantiert, daß alle Teile räumlich voneinander getrennt sind.

- *d-sep-2d*
  Falls eine Perspektive bereits vorgegeben wurde, ist dieser Betrag dafür verantwortlich, daß die explodierten Objekte aus der gewählten Perspektive als voneinander getrennt wahrgenommen werden können.

Der Abstand *d-sep-3d* läßt sich leicht mit Hilfe des Perspektivenquaders errechnen.
Das Ermitteln des Betrages *d-sep-2d* gestaltet sich hingegen schwieriger. Die exakte Vorgehensweise, für jedes zu explodierende Objekt den Betrag zu errechnen, der nötig ist, um alle Objekte bei gegebener Perspektive (und Projektion) voneinander zu trennen, erscheint wegen der intensiven Berechnungen als zu zeitaufwendig.
Aus diesem Grund wird *d-sep-2d* in **TOPAS** durch folgende Heuristik ermittelt: Jedem Objekt wird ein Betrag zugeordnet, der von seiner räumlichen Ausdehnung abhängt. Damit wird ebenfalls erreicht, daß größere Objekte weiter voneinander getrennt werden als kleinere Objekte. Zusätzlich wird *d-sep-2d* parametrisiert und durch Intervallgrenzen eingeschränkt. So kann verschiedenen Perspektiven Rechnung getragen werden.
Diese Vorgehensweise ist sehr schnell. Die Wahl von Parameter und Intervallgrenzen ermöglicht eine Einflußnahme auf die Darstellung. Im ungünstigsten Fall kann die Heuristik jedoch zu Illustrationen führen, bei denen sich die explodierten Teile überlappen.

Perspektive

Die Perspektive, aus der die neue Objektkonstellation zu sehen sein soll, kann bereits vorgegeben sein. Dann hat sie Einfluß auf die Wahl von Explosionsrichtung, -bahn und -betrag (*d-sep2d*).
Wird sie jedoch erst hinterher gewählt, dann müssen die ermittelten Explosionsrichtungen bei der Perspektivenwahl ausgeschlossen werden.

Einflußfaktoren auf die Operationalisierung

Neben diesen grundsätzlichen Überlegungen müssen zum Operationalisieren der Technik folgende Punkte in Erwägung gezogen werden:

- in welcher Reihenfolge sollen die Teile verschoben werden?
- wie oft soll jedes Teil während des Explosionsprozesses verschoben (Ausführungsart)?

Diese beiden Punkte werden in den beiden nachfolgenden Abschnitten näher untersucht.

Explosionsreihenfolge

Bei der Demontage realer Objekte ist die Reihenfolge bereits vorgegeben, wie folgendes Beispiel verdeutlicht.
Um einen Autoreifen zu wechseln, muß zuerst die Radkappe entfernt werden, die Schrauben sind zu lösen und schließlich kann das Rad abgezogen werden. Eine computererstellte Explosionsdarstellung muß sich zwar nicht an diese Reihenfolge halten, es könnte ohne weiteres zuerst das Rad entfernt werden und zum Schluß die Radkappe. Wichtig ist in diesem Fall nur, daß das Ergebnis korrekt ist.
Wird die komplette Explosion direkt berechnet, können die jeweils betroffenen Teile unmittelbar an ihren endgültigen Platz verschoben werden. Dazu werden $O(n)$ Schritte benötigt.
Im Hinblick darauf, daß die Arbeitsweise des Systems durch einen Trace nachvollzogen werden soll, bietet ein inkrementelles Verfahren die Möglichkeit, daß alle Zwischenschritte angezeigt werden können, die der Benutzer selbst beim Zerlegen erreichen würde.

Deshalb wurde dieses Verfahren gewählt. Das äußerste Teil einer Baugruppe. "Last Node" in der Zusammenbauhierarchie, wird dabei zuerst verschoben. Deshalb wird das Verfahren "LNF" (Last Node First) genannt.
Bei der Realisierung des inkrementellen Ansatzes wird das jeweils zu explodierende Teil um einen bestimmten Betrag von den anderen wegbewegt, der ausreicht, um es von der übrigen Baugruppe zu trennen. Die bereits explodierten Teile müssen ebenfalls um diesen Betrag wegbewegt werden, damit keine Kollisionen entstehen. Dazu sind jedoch im Vergleich zu dem anderen Verfahren mehr Verschiebeoperationen nötig, nämlich $O(n(n+1)/2)$.
Mit Hilfe dieses Verfahrens können jedoch alle Zwischenschritte sehr natürlich dargestellt werden. Die Möglichkeit, nach jedem Schritt zu evaluieren, ob alle Ausführungsbedingungen noch gelten und ob Restriktionen verletzt wurden. ist ebenfalls gegeben.
Weiterhin kann leicht eine Animation der Explosion generiert werden.
Die nachfolgende Bildsequenz (siehe Abbildungen 27) demonstriert die inkrementelle Vorgehensweise.

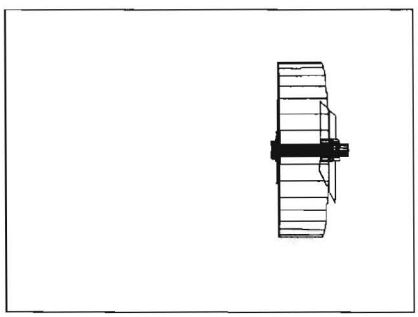

Abb.27 (a): 1. Schritt

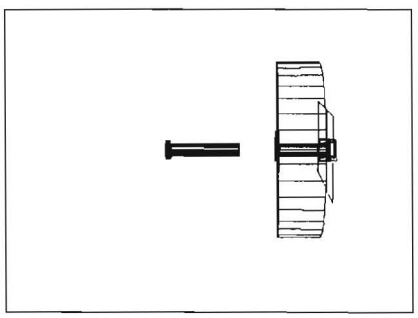

Abb.27 (b): 2. Schritt

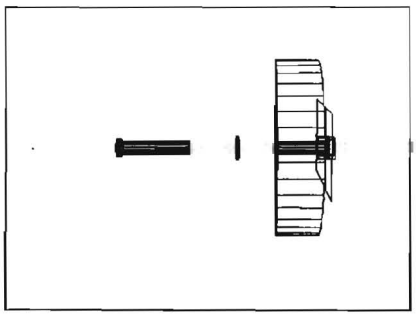

Abb.27 (c): 3. Schritt

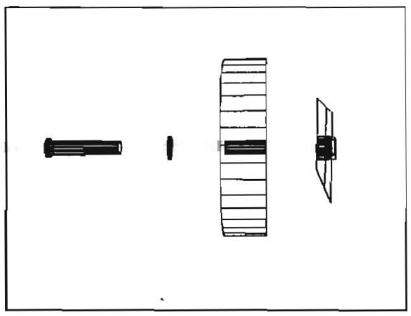

Abb.27 (d): 4. Schritt

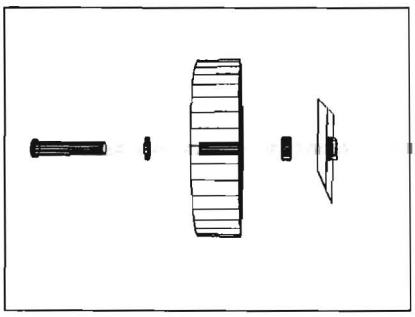

Abb.27 (e): 5. Schritt

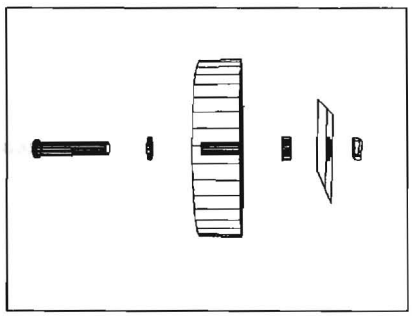

Abb.27 (f): 6. Schritt

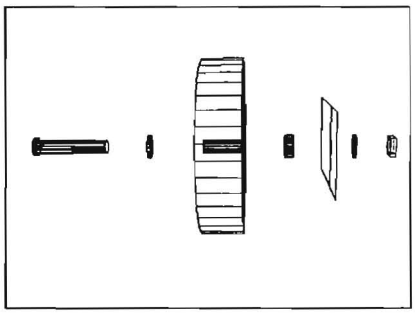

Abb.27 (g): 7. Schritt

Implementierung

Im Folgenden sei ein Pfad ein gerichteter zykelfreier Graph mit folgenden Eigenschaften:

- Jeder Knoten, außer der Quelle, hat genau einen Vorgänger.

- Jeder Knoten, außer der Senke, hat genau einen Nachfolger.
- es gibt nur Kanten von Knoten der Tiefe k nach Knoten der Tiefe k + 1.

Um die Explosionstechniken zu implementieren, sind folgende Probleme algorithmisch zu lösen:

1. Bestimmung der an der Explosion beteiligten Objekte. Dies geschieht durch Auswahl des zugehörigen Unterbaumes in der Zusammenbauhierarchie, dem *Explosionsbaum.*
2. Erzeugen eines Plans zur Ausführung der Verschiebeoperationen.

Der zweite Punkt wird dadurch realisiert, daß der in Schritt 1 ausgewählte *Explosionsbaum* unter folgender Bedingung immer wieder in bis auf die Wurzel disjunkte Teilbäume zerlegt wird:
Jeder Teilbaum wird, ausgehend von der Wurzel, in eine jeweils andere Richtung verschoben. Diese Zerlegung wird solange wiederholt, bis der Teilbaum in einzelne Pfade zerfallen ist, so daß alle Knoten eines Pfades in die gleiche Richtung explodiert werden.
Dies wird durch die nachfolgenden Schritte erreicht:

- Für das Erstellen disjunkter Teilbäume wird die Wurzel dubliziert.
- Von einem Teilbaum wird ein Pfad abgespalten, so daß alle Knoten eines Pfades in die gleiche Richtung explodiert werden.

Diese beiden Schritte greifen ineinander. Stellt sich beim Erstellen des Pfades heraus, daß ein Knoten mehrere Söhne hat, wird mit Hilfe des ersten Schrittes vom Pfad ein Teilbaum abgespaltet, der später weiterzerlegt wird.
Hierdurch werden Explosionsreihenfolge, Explosionsrichtung und die zu explodierenden Objekte festgelegt.
Der Explosionsbetrag wird bei Ausführung im Grafiksystem errechnet.
Als fest, aber beliebig für eine spezielle Explosion, werden die Parameter Explosionsbahn und Explosionstechnik betrachtet.
Folgendes Beispiel veranschaulicht die Arbeitsweise der nachfolgenden Algorithmen (siehe Abbildungen 28, 29, 30 und 31).

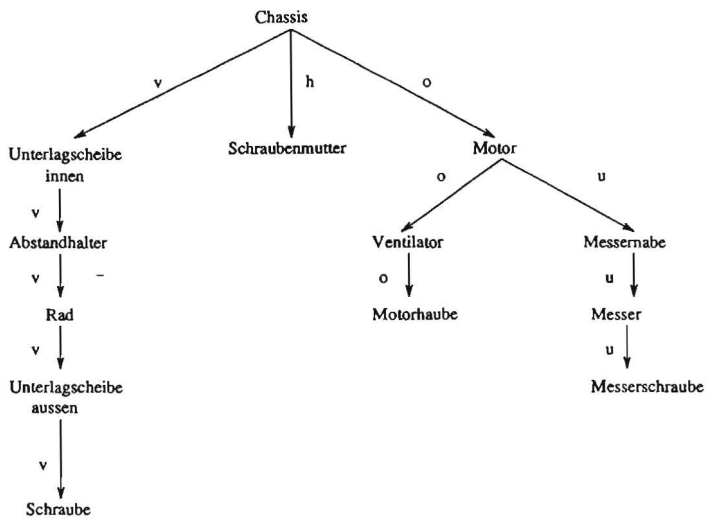

Abbildung 28: Ein Beipiel einer Zusammenbauhierarchie.

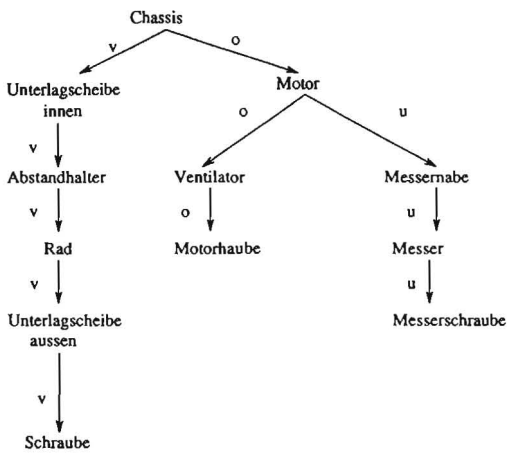

Abbildung 29: Der daraus generierte Explosionsbaum.

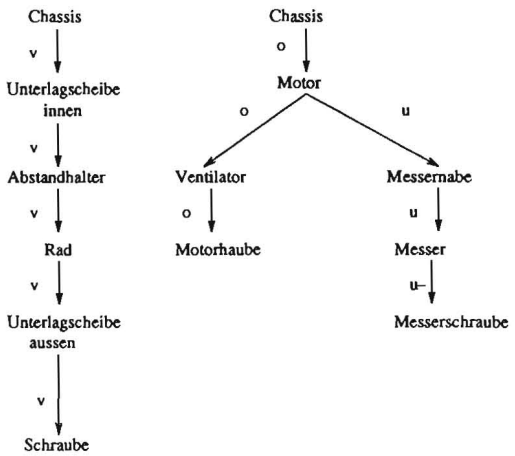

Abbildung 30: Nach der ersten Sequentialisierung: Eine Wurzel wurde gesplittet, ein Pfad wurde bereits erstellt.

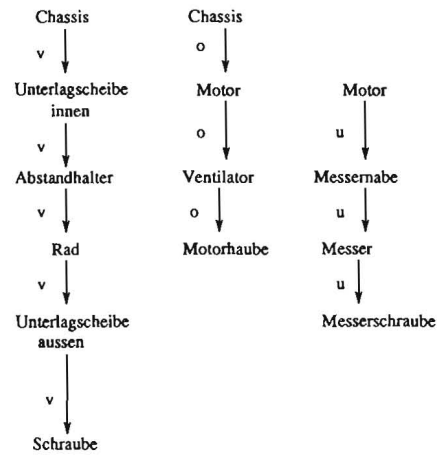

Abbildung 31: Nach dem zweiten Konvertierungsschritt: Die Konvertierung ist beendet. Der Baum wurde komplett in Pfade zerlegt.

Bestimmung der an der Explosion beteiligten Objekte

Der Algorithmus zur Bestimmung aller beteiligten Objekte muß in der Zusammenbauhierarchie den Unterbaum auswählen, der alle vom Benutzer als Explosionsteile spezifizierten Knoten enthält.
Diese Aufgaben bewältigt folgender Algorithmus, E_k sei dabei ein beliebiger Teilbaum der Zusammenbauhierarchie.

(1) Sei w die Wurzel des Baumes E_k, dann ist w ein zu explodierendes Teil
(2) **FOR** Alle Söhne v von w
(3) **DO**
(4) **IF** Ist v ein zu explodierendes Teil?
(5) **THEN** w:=v
(6) Gehe zu (2)
(7) **ELSE** Bestimme zu v die reflexive, transitive Hülle H
(8) Vermerke w und H in der Liste L_w
der zu explodierenden Teile

Zerlegen des Unterbaumes

Nachdem der Unterbaum ausgewählt wurde, muß der Baum durch Dublizieren der Wurzel in Teilbäume zerlegt werden, sofern es mehr als einen Sohn gibt.
Dies wird durch folgenden Algorithmus erreicht:

(1) **FOR** Alle Bäume E_k
(2) **DO** Sei w die Wurzel des Baumes E_k
(3) Bestimme die Anzahl j $\leq$ 6 der *Anfügeseite*
(4) **IF** j > 0?
(5) **THEN**
(6) **FOR** i from 1 to j - 1
(7) **DO** Mache neuen Baum E_{ki} und füge w
als Wurzel zu E_{ki} hinzu
(8) Streiche i aus der Menge der Anfügeteile

Generieren der Pfade

Um die Bildung der Pfade zu beeinflussen, wird als Hauptexplosionsrichtung für einen Unterbaum die Anfügeseite des Sohnes der Wurzel gewählt. Dadurch wird in den meisten Fällen das Erstellen langer Pfade erreicht, was das Ausführen der Explosion vereinfacht.
Sobald ein Knoten mehrere Söhne hat, wird der Unterbaum weiter zerlegt und der Teil in Hauptexplosionsrichtung an den vorhandenen Pfad angefügt, während der andere Teil einen neuen Unterbaum bildet.
Am Ende ist der *Explosionsbaum* vollständig in Pfade mit jeweils gleicher Explosionsrichtung zerlegt.

(1) Sei E eine Liste mit allen *Explosionsbäumen*
(2) **FOR** Alle Teilbäume E_k
(3) **DO** w:=Wurzel(E_k)
(4) v:=Sohn(w)
(5) Hauptrichtung:=*Anfügeseite*(w,v)
(6) **WHILE** Anzahl{Söhne(v)} > 0
(7) **DO**
(8) **FOR** Alle Söhne v_l von v
(9) **DO**
(10) **IF** *Anfügeseite*(v, v_l) $\neq$ Hauptrichtung?
(11) **THEN** $E_{k'}$:= Unterbaum mit Wurzel w und Sohn v_l
(12) Füge Unterbaum $E_{k'}$ am Ende von E ein
(13) Lösche $E_{k'}$ in E_k;
v bleibt allerdings als Knoten in E_k enthalten
(14) **ELSE** $next_v$:=v_l
(15) v:=$next_v$; $next_v$:=nil

Das Beispiel 32 illustriert eine explodierte Darstellung des Rasenmähers. Es wurde auf fünf unterschiedlichen Achsen explodiert. Die Radgruppen und die Motorgruppe sind vollständig zerlegt.

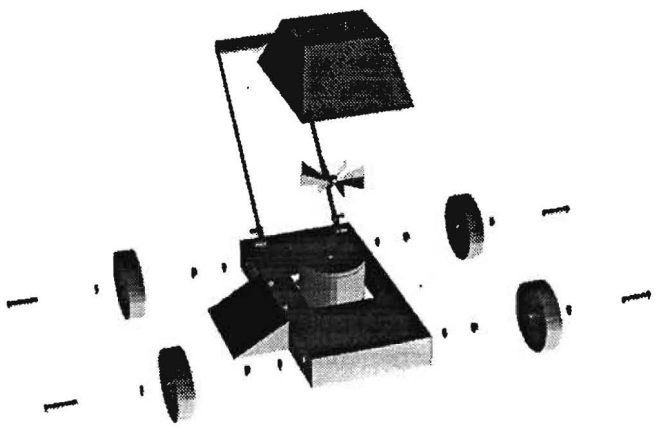

Abbildung 32: Eine Explosionsdarstellung des Rasenmähers.

4.3 Querschnitte

Querschnitte sind Darstellungen, bei denen ein Objekt entlang einer Ebene, die durch dieses Objekt verläuft, durchtrennt wird (siehe Abbildung 33). Sie

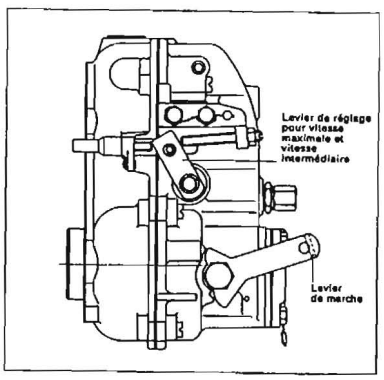

Abbildung 33: Querschnitt durch eine Pumpe. Aus: [Bosch 85 b].

dienen dazu, das "Innenleben", speziell von komplexen Objekten zu zeigen und ermöglichen dem Betrachter einen Einblick, den er sich am Modell selbst nicht verschaffen kann, ohne es dabei zu zerstören. Dadurch können Zusammenhänge, die sonst im Inneren eines Modells verborgen sind, vom Betrachter erkannt werden und tragen zum besseren Verständnis der Funktionsweise bei.

Vorgehensweise

In [Rossignac et al. 92] wird ein Ansatz zur Erstellung von Querschnitten vorgestellt. Er beruht darauf, daß ein Hilfs-"clipping plane" in das Objekt hineingeschoben wird. Daraus ergibt sich, daß ein Teil des Objektes nicht mehr im Sehvolumen enthalten und somit nicht mehr sichtbar ist. Da bei dieser Vorgehensweise keine Veränderung des 3D-Modells erfolgt, ist eine Repräsentation der entfernten Objektteile nicht vorhanden. Dementsprechend kann hierauf in einem zugehörigen Text kein Bezug genommen werden.
Aus diesem Grund wurde eine Vorgehensweise gewählt, die den Querschnitt durch Ersetzen am Modell realisiert.
Zum Erstellen einer Querschnittdarstellung wird neben dem Modell die *Querschnittebene*, die das Objekt zertrennen soll, benötigt. Dabei kann diese Ebene beliebig im Raum liegen und ist nicht auf die üblichen achsenparallelen *Querschnittebenen* beschränkt.
Ausgehend von dieser Ebene muß entschieden werden, welche Objekte unversehrt bleiben, welche entfernt werden, und welche durchtrennt werden. Das wird erreicht, indem für jedes Objekt seine Lage bezüglich der Ebene bestimmt wird.
Für die Objekte, die die Ebene durchdringen, werden *Ersatzobjekte* angelegt. Diese werden errechnet, indem die Strecken der Polygonzüge, die die Flächen der Objekte bilden, mit der *Querschnittebene* geschnitten werden. Die Schnittpunkte werden in den Polygonzug aufgenommen und die Punkte, die hinter der Ebene liegen, werden entfernt.
Da jedes einzelne Objekt des Modells betrachtet werden muß, kann auch mit dieser Vorgehensweise ein beträchtlicher Geschwindigkeitsgewinn erzielt werden, wenn in einem ersten Schritt die umhüllenden Quader der Objekte betrachtet werden.
Damit kann die genaue Schnittbetrachtung mit der Ebene auf eine kleine Zahl von Objekten reduziert werden.

Implementierung

Der Algorithmus läßt sich folgendermaßen beschreiben:

```
(1)  FOR Alle Objekte in Szene
(2)    DO
(3)      CASE
(4)        OF Umhüllender Quader vor Querschnittebene qe?
(5)          THEN Objekt bleibt erhalten
(6)        OF Umhüllender Quader hinter qe?
(7)          THEN Objekt entfernen
(8)        OF Umhüllender Quader in qe?
(9)          THEN
(10)           FOR Alle Flächen des Objektes
(11)             DO
(12)               CASE
(13)                 OF Fläche vor qe?
(14)                   THEN Fläche unverändert
(15)                 OF Fläche hinter qe?
(16)                   THEN Fläche entfernen
(17)                 OF Fläche in qe?
(18)                   THEN Berechne Schnittpunkte sp mit qe
(19)                     Sammle sp und vor qe
                         liegenden Punkte in einer Liste
(20)                     Erstelle neue Fläche mit Liste
(21)               Lege Ersatzobjekt mit Flächen
                   aus (14) und (19) an
```

Die Abbildung 34 zeigt einen Querschnitt durch ein komplexes Objekt, den Rasenmäher. Es ist das Innere aller Objekte, die auf der Schnittebene liegen, zu sehen, wie Motor, Messergruppe, Ventilator und Gehäuse.

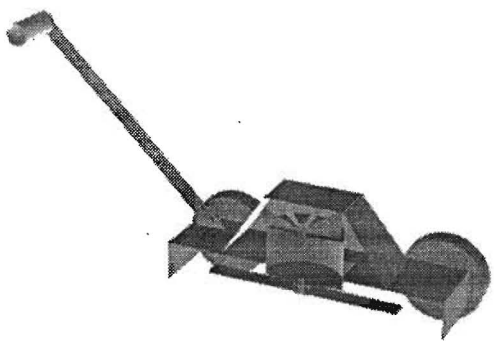

Abbildung 34: Der Rasenmäher im Querschnitt.

4.4 3D Trajektorienpfeile

Bewegungen können normalerweise nur durch Bilderfolgen oder Animationen gezeigt werden (siehe Abbildung 35).

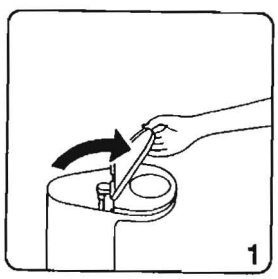

Abbildung 35: Trajektorienpfeil zum Zeigen einer Bewegung. Aus: [HIT]

In Betriebs- und Bedienungsanleitungen besteht jedoch ein großes Interesse, diese Bewegungen auch in "stehenden Bildern" zu visualisieren. Dies geschieht mit Hilfe von Pfeilen, die sich entlang der auszuführenden Bewegung ("Trajektorie") erstrecken, sogenannten Trajektorienpfeilen.

Trajektorienpfeile sind Pseudoobjekte, da sie in der realen Welt nicht vorhanden sind.

Vorgehensweise

Für die Bewegungen sind Trajektorien vorhanden (siehe Abschnitt 3.2.2), die zur Berechnung der Pfeile benutzt werden.
Die Länge eines Pfeiles ist durch die Trajektorie bestimmt, nicht aber die Dicke des Schaftes, die Dicke der Spitze und die Länge der Spitze. Im Extremfall können Pfeile gewünscht werden, die nur aus Spitze bestehen und deshalb die Form eines Dreiecks haben.
Welche Art von Pfeil geeignet ist, hängt einerseits von der Trajektorie, andererseits aber auch vom persönlichen Geschmack eines Illustrators ab. Deshalb sollen diese Faktoren beeinflußt werden können.
Um immer beste Sichtbarkeit auf den Pfeil zu garantieren, wird er senkrecht zum Blickvektor erstellt.
Dazu wird in jedem Stützpunkt eine Strecke oberhalb und unterhalb des Stützpunktes abgetragen, die senkrecht zu Blickvektor und "Expansionsvektor" ($<aktuellerStützpunkt, \vec{nachfolgenderStützpunkt}>$), steht und damit eindeutig bestimmt ist.
Die oben erwähnten Einflußfaktoren, wie Pfeilschaftdicke, Pfeilspitzenbreite und Pfeilspitzenlänge sind parametrisierbar.
Es können gekrümmte oder ebene Pfeilspitzen generiert werden, je nachdem, ob der Expansionsvektor nach der ersten Berechnung der Pfeilspitze fest bleibt, oder mitverändert wird.
Um eine gleichmäßige Pfeilspitze zu erhalten, muß dann, ausgehend vom Startwert, bei jedem Durchgang die aktuelle Pfeilspitzenbreite neu berechnet werden.

Implementierung

Durch Vorgabe der Trajektorie steht bereits Anfang und Ende des Pfeiles fest. Falls vom Benutzer nicht angegeben, besteht die Aufgabe von **TOPAS** darin, geeignete Werte für die Parameter zu finden.
Folgende Werte wurden hierbei gewählt: Die Pfeilschaftdicke beträgt 1/10 der Gesamtlänge der Trajektorie, die Pfeilspitzenbreite ist doppelt so groß wie die Pfeilschaftdicke und die Pfeilspitzenlänge beträgt 1/4 der Gesamtpfeillänge.

Der Algorithmus, mit dem Trajektorienpfeile erstellt werden können sieht wie folgt aus:

(1) **FOR** Alle Stützpunkte v_i (i=i...n) von Trajektorie
(2) **DO**
(3) **IF** $v_i \neq v_n$?
(4) **THEN** Berechne Vektorprodukt:
$\vec{v_s} := \vec{Blickvektor} \times (v_i - \vec{v_{i+1}})$
(5) Normiere $\vec{v_s}$
(6) **IF** Pfeilschaft?
(7) **THEN** $p_o := v_i + 1/2 Pfeilschaftdicke \cdot \vec{v_s}$
(8) $p_u := v_i - 1/2 Pfeilschaftdicke \cdot \vec{v_s}$
(9) Sammle p_o und p_u in Liste
(10) **IF** Pfeilspitze?
(11) **THEN** $p_o := v_i + 1/2 Pfeilspitzenbreite_{v_i} \cdot \vec{v_s}$
(12) $p_o := v_i + 1/2 Pfeilspitzenbreite_{v_i} \cdot \vec{v_s}$
(13) Sammle p_o und p_u in Liste
(14) Berechne $Pfeilspitzenbreite_{v_{i+1}}$
(15) Nehme v_n in Liste auf
(16) Erstelle Pfeil aus Liste

Das in Abbildung 36 gezeigte Beispiel soll den Benutzer dazu auffordern die Kaffeetasse am Henkel anzugreifen und unter die Kaffeemaschine zu stellen.

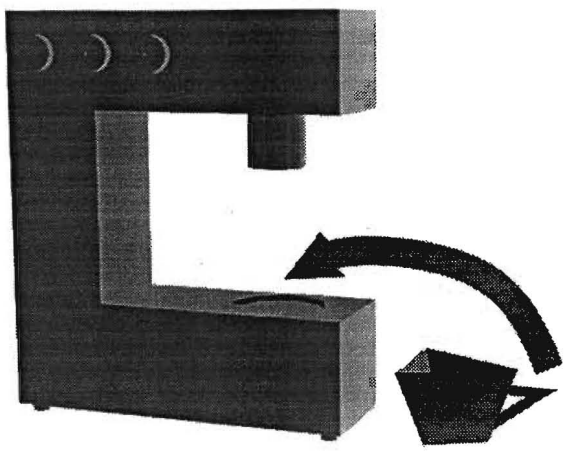

Abbildung 36: Kaffeemaschine mit Trajektorienpfeil.

4.5 Ghost Images

Ghost Images werden benutzt, um in einer Illustration die Visualisierung eines Sachverhaltes aus einer der folgenden Gruppen zu erreichen:

- Zeigen eines Objektes in zwei verschiedenen Zuständen (WZ)
- Zeigen von mehreren gleichzeitig möglichen Zuständen (WZ)
- Zeigen eines Übergangs von einem Zustand in einen anderen

Im ersten Fall sollen zwei zeitlich aufeinanderfolgende Zustände gezeigt werden. Ein Beispiel hierfür ist, das Öffnen eines Deckels zu visualisieren. Der Deckel wird in diesem Fall in der "Geschlossen-Position" und in der "Offen-Position" dargestellt.
Im zweiten Fall wird eine Situation illustriert, in der ein Betrachter mehrere Auswahlmöglichkeiten zwischen verschiedenen Zuständen des Objektes hat. Er soll sich für eine Möglichkeit entscheiden und diesen Zustand wie im ersten Fall mit dem Objekt realisieren. Beispielsweise soll eine Radioantenne in die Position gebracht werden, in der ein Sender am besten zu empfangen ist (siehe dazu die Abbildungen 37).

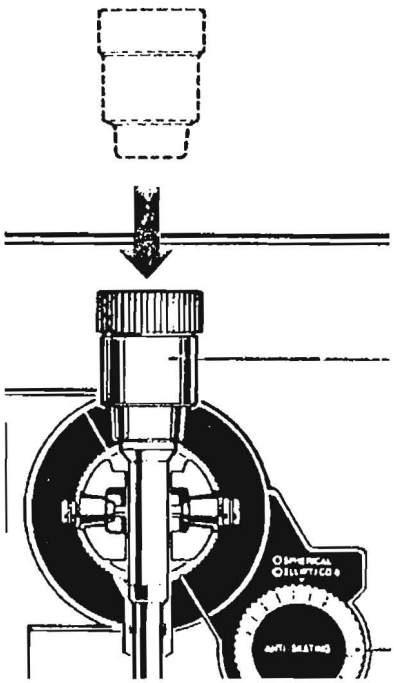

Abb.37 (a): Übergang zwischen zwei Zuständen. Aus: [Philips a].

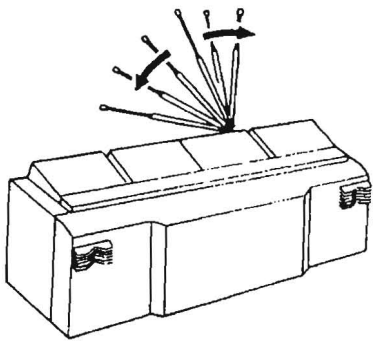

Abb.37 (b): mehrere gleichzeitig mögliche Zustände. Aus: [Philips b].

Im letzten Fall soll eine Bewegung durch ein Bild erklärt werden. Anders als bei den Trajektorienpfeilen, bei denen die Bewegung als Pfeil dargestellt wurde, wird das Gewünschte bei dieser Technik durch eine Anzahl von Ghost Images erreicht, die die Zwischenzustände des zu bewegenden Objektes zeigen.
Ghost Images sind offensichtlich Pseudoobjekte. Sie sind Duplikate eines Objektes in einer Abbildung in verschiedenen räumlichen Positionen.

Vorgehensweise

Ghost Images können durch Erstellen von Kopien des ursprünglichen Objektes erzeugt werden.
Um diese Kopien als Pseudoobjekte kenntlich zu machen, empfiehlt es sich, ihre Abbildungsart gegenüber dem Original zu verändern. Dies ist insbeson-

ders deshalb wichtig, damit der Betrachter diese Objekte als illustratorisches Mittel erkennt.
Bei gerenderten Darstellungen kann dies durch Änderung von Materialbeschaffenheit oder Farbe geschehen. Bei Drahtrahmenmodellen können die Ghost Images beispielsweise durch Stricheln der Objektumrandung (siehe Abschnitt 4.11) gekennzeichnet werden.
Es lassen sich grundsätzlich zwei unterschiedliche Vorgehensweisen unterscheiden, abhängig davon, ob ein, bzw. mehrere Zustände oder ob eine Bewegung eines Objektes dargestellt werden soll.
Der erste Fall ist einfacher zu realisieren, da die verschiedenen Zustände als Weltszenen vorhanden sind. Um die Abbildung zu erstellen muß vom aktuellen Zustand zu dem Zustand gewechselt werden, in dem das Objekt als Ghost Image in der Abbildung erscheinen soll. Ist der Zustand hergestellt, wird eine Kopie des Objektes angelegt, wieder in den Ausgangszustand zurückgewechselt und die Kopie zum Modell hinzugefügt.
Werden Ghost Images dazu verwandt, eine Bewegung zu visualisieren, muß eine andere Vorgehensweise gewählt werden. Anders als im ersten Fall sind hierbei die räumlichen Positionen für die Ghost Images nicht explizit als Zustände repräsentiert. Die Bewegung ist jedoch als Trajektorie gegeben. Ähnlich wie bei den Trajektorienpfeilen wird sie auch bei dieser Technik benutzt. Die Positionen, die ein Objekt bei der Bewegung entlang dieser Trajektorie annimmt, lassen sich als diskrete Punkte auf dieser Trajektorie errechnen.
Wieviele Ghost Images nötig sind, um beim Benutzer den Eindruck einer Bewegung zu erwecken, ist vom ästhetischen Empfinden des Grafikdesigners, der die Illustration entwirft.
Aus diesem Grund ist die Anzahl der Ghost Images, die entlang der Trajektorie erstellt werden sollen, parametisierbar.

Implementierung

Ein Algorithmus für den ersten Fall läßt sich folgendermaßen formulieren:

(1) **FOR** Alle Zustände WZ_i (i=1..n), in denen Ghost Images des Objektes o erstellt werden sollen
(2) **DO**
(3) Wechsle von der Ursprungsszene WZ_u auf WZ_i

(4) Erstelle Kopie als Ghost Image von o
(5) Wechsle auf die WZ_u zurück
(6) Füge zu WZ_u das Ghost Image hinzu

Im zweiten Fall tritt eine weitere Schwierigkeit auf. Je nachdem, ob es sich bei der Bewegung um eine Drehbewegung, um eine Schiebebewegung oder um eine kombinierte Bewegung handelt, muß die Position, bzw. der Rotationswinkel oder beides berechnet werden.
Abhängig von der Anzahl der Länge der Trajektorie errechnet das System einen Defaultwert.
Der Algorithmus für diesen Fall sieht wie folgt aus:

(1) **FOR** Anzahl der Ghost Images (i=1..n) für das Objekt O in Szene WZ_u
(2) **DO**
(3) Berechne Punkt P_i auf der Trajektorie: P_i := j-ter Stützpunkt der Trajektorie j := i · Anzahl{Stützpunkte}
(4) Berechne Rotationswinkel α von O für P_i über das Skalarprodukt
(5) Bewege O zu P_i
(6) Rotiere O um Rotationsachse um α
(7) Erstelle Kopie als Ghost Image von O
(8) Bringe O in Ausgangsposition
(9) Füge Ghost Image zu WZ_u hinzu

Abbildung 38 fordert den Benutzer dazu auf, den Deckel der Kaffeemaschine zu öffnen. Die Ghost Images zeigen dabei die erreichbaren Zwischenzustände.

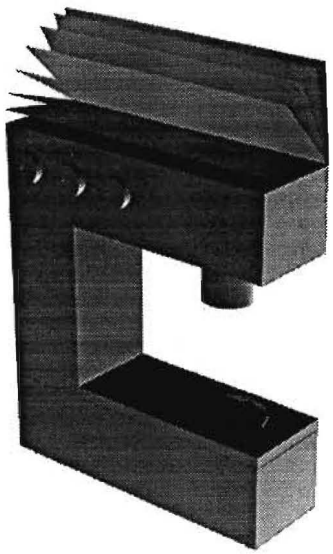

Abbildung 38: Kaffeemaschine mit Ghost Images zum Zeigen der Öffnenbewegung.

4.6 Virtuelle Böden

Es fällt einem Benutzer einfacher die räumliche Anordnung von Objekten wahrzunehmen, wenn ein Boden in einer Illustration vorhanden ist (siehe Abbildung 39). Hierbei wird der 3D-Effekt einer Szene unterstrichen.

Abbildung 39: Der Boden als zur Szene gehöriges Defaultobjekt. Aus: [Philips c].

In vielen Abbildungen kann das Vorhandensein eines Bodens als gesichert betrachtet werden, auch, wenn er nicht dargestellt wird. Objekte, wie Kaffeemaschinen, Radios, usw. schweben schließlich nicht in der Luft.

Böden werden normalerweise dann hinzugefügt, wenn es entweder für den Sachverhalt von besonderer Wichtigkeit ist, daß ein Objekt auf einem Untergrund steht oder wenn bei einem Benutzer wirklich der Eindruck entstehen würde, daß die entsprechenden Objekte in der Luft schweben. Dies ist jedoch ein sehr subjektives Kriterium.
Böden werden auch dann hinzugefügt, wenn ein Objekt beispielsweise auf dem Kopf steht. Einem Betracher wir dadurch sofort klar, daß es sich bei dieser Illustration nicht um einen Irrtum handelt, sondern daß das Objekt absichtlich in dieser Form abgebildet wurde.
Wird dem Boden weiterhin eine Struktur verliehen, wie zum Beispiel bei einem gekachelten Boden oder wird der Boden mit verschiedenen Farben eingefärbt, so kann ein Betrachter neben der räumlichen Konstellation ebenfalls problemlos Schlüsse über die Größenverhältnisse der Objekte ziehen.
Die perspektivische Verzerrung der Struktur erleichtert ihm das Abschätzen der räumlichen Ausdehnung.
Virtuelle Böden sind folglich Defaultdomäneobjekt. Wie in Kapitel 3 beschrieben, handelt es sich hierbei um Objekte, die in einer Domäne nicht explizit modelliert sind, deren Existenz jedoch als gesichert angenommen werden kann. Defaultdomäneobjekte dienen normalerweise als Hintergrundobjekte.

Vorgehensweise

Virtuelle Böden werden normalerweise an der Unterseite eines Objektes erstellt. Wurde ein Objekt auf den Kopf oder zur Seite gedreht, um zu zeigen, daß der Benutzer das Modell auf die entsprechende Seite legen soll, so ist es in diesem Fall nicht mehr wünschenswert, den Boden an der Unterseite des Objektes anzubringen, sondern an der jetzt untengelegenen Seite.
Aus diesem Grund ist die Seite, an der der Boden generiert werden soll, parametrisierbar.
Der Boden soll weiterhin größer als das Objekt sein, um den Eindruck eines Bodens zu verstärken. Außerdem können keine Kanten und Ecken durch daraufstehende Objekte verdeckt werden.

Implementierung

Als Ausgangsdaten für das Erstellen eines virtuellen Bodens werden der umhüllende Quader und die Seite, an der der Boden erstellt werden soll, benötigt. Als Defaultwert wird die Unterseite des Objektes angenommen.
Die Größe des Bodens läßt sich leicht aus dem für jede Szene vorhandenen Szenequader errechnen. Die Maße der spezifizierten Seite werden entnommen und um 20% vergrößert.
Mit diesen Werten wird ein Boden aus gleichgroßen Kacheln erstellt, die abwechselnd schwarz und weiß, wie ein Schachbrett, eingefärbt werden.
In **TOPAS** wird dabei ein rekursiver Ansatz verfolgt, durch Erhöhen der Rekursionstiefe läßt sich die Anzahl der Kacheln erhöhen. Als Defaultwert werden 16 Kacheln erstellt.
Der Algorithmus hierzu lautet folgendermaßen:

(1) Entnehme die Ausdehnung der spezifizierten Seite
    aus dem umhüllenden Quader
(2) Vergrößere den Boden um 20%
(3) Zerteile Boden in 4 gleichgroße Teile
(4) **FOR** Alle Teile T_i (i=1..4)
(5) **DO**
(6) Zerteile T_i in 4 Teile T_{ij} (j=1..4)
(7) Weise T_{ij} eine Farbe zu
(8) Füge T_{ij} zum Boden zusammen
(9) Nehme Boden in Szene auf

Die Abbildung 40 zeigt eine Abbildung mit einem virtuellen Boden. Der Boden ermöglicht die räumliche Lage der Kaffeemaschine, sowie die der davorstehenden Tasse zu identifizieren.
Weiterhin erleichtert er das Abschätzen der Größenverhältnisse zwischen Kaffeemaschine und Tasse.
Das Schachbrettmuster verleiht der Szene zusätzlich räumliche Tiefe.

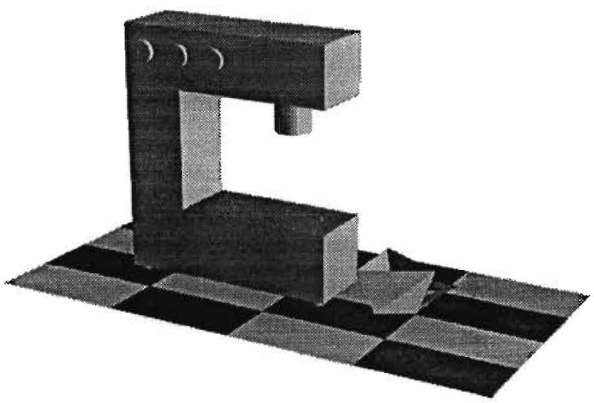

Abbildung 40: Der virtuelle Boden zeigt die Lage der Maschine im Raum.

4.7 Einpassen einer Szene auf eine vorgegebene Abbildungsgröße

Bei der Erstellung von technischen Bedienungsanleitung sind fast immer Rahmenbedingungen bezüglich des Layouts zu beachten. Muß das Dokument auf eine A4- oder A5-Seite passen, soll es auf einem bestimmten Bildschirm wiedergegeben werden, usw.
Diese Constraints schränken ebenfalls die Illustrationen ein. Offensichtlich darf eine Abbildung nicht größer sein, als das Seitenlayout. Meistens werden strengere Anforderungen gestellt. Die Abbildung soll auf den für sie reservierten Platz im Dokument passen. Um dies zu erreichen, muß eine Funktion zur Verfügung stehen, die die Abbildung auf eine vorgegebene Größe bringt. Ein anderer Fall, in dem Abbildungen auf eine vorgegebene Größe gebracht werden müssen, sind Insets. Zur genauen Beschreibung dieser Technik siehe Abschnitt 4.10.

Vorgehensweise

Für eine Funktion, die die oben beschriebenenen Aufgaben erfüllt, werden als Ausgangsdaten die Abbildung und die vorgegebene Abbildungsgröße zur

Verfügung gestellt.
In einem Grafikeditor, der der Kamerametapher folgt, ist eine Verkleinerung oder Vergrößerung eines Bildes dadurch zu erreichen, daß die Kamera von den Objekten "Weggefahren", bzw. "Hingefahren" wird.

Implementierung

Für die Kamera muß entsprechend dem vorher gesagten eine neue Position berechnet werden.
Die neue Position kann exakt errechnet werden, wenn die Art der Abbildung und die genauen Parametereinstellungen bekannt sind.
Ist dies nicht der Fall, so kann zumindest anhand der bekannten Einstellungen und dem Kosinussatz eine pessimistische Näherung berechnet werden. Ausgehend von diesen Werten wird die Kameraposition zu den Szeneobjekten iterativ verkleinert und getestet, ob die Abbildung auf die vorgegebene Größe gebracht wurde.
Dieses Verfahren hat den Vorteil, daß beim Benutzer der Eindruck des "Heranfahrens" an die Objekte erzeugt wird, wie er es bei realen Foto- und Filmkameras kennt. Diese Vorgehensweise kann als eine einfache Art der Animation angesehen werden.
In Pseudocode kann man das folgendermaßen ausdrücken:

```
(1) Berechne Näherung für die neue Kameraposition
    mit Hilfe des Kosinussatzes
(2) Stelle neue Position im Grafikeditor ein
(3) WHILE Szene noch nicht auf vorgegebener Größe
(4)   DO
(5)     Vergrößere Kameraentfernung um 5%
(6)     Stelle neue Position im Grafikeditor ein
```

Das Beispiel in den Abbildungen 41 illustriert das Einpassen einer Szene.

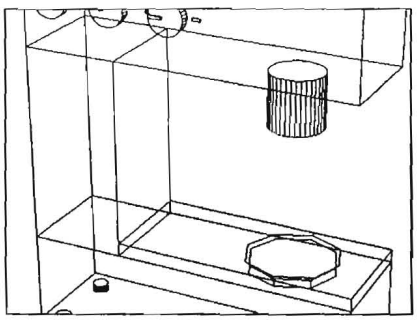

Abb.41 (a): Das Objekt ist nicht vollständig im Grafikfenster zu sehen.

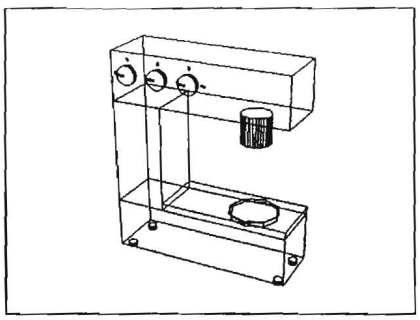

Abb.41 (b): Das Grafikfenster nach dem Einpassen in des Objektes.

4.8 Fokussieren durch Zentrieren von Objekten

Es existieren viele Mittel, um in Illustrationen die Aufmerksamkeit des Betrachters auf ein bestimmtes Objekt zu lenken. Einige davon sind Farbe, Pfeile, Lupen usw.
Ein weiteres Mittel, um die Aufmerksamkeit des Betrachters auf ein bestimmtes Objekt zu lenken ist, das Objekt in den Bildmittelpunkt zu schieben, also zu zentrieren.

Vorgehensweise

Ausgangsdaten für diese Operation sind die Szene und ein Objekt, das fokussiert werden soll.
Sinnvoll ist eine Vorgehensweise, bei der keine unerwünschten Seiteneffekte auftreten und somit die Operation für den Illustrator durchschaubar machen. Deshalb wird die Abbildungsgröße beibehalten.
Ähnlich wie bei dem vorangegangenen Operator wird das Fokussieren durch eine Änderung der Kameraposition zu den Szeneobjekten erreicht. Die Kamera wird so ausgerichtet, daß sie auf das angegebene Objekt zielt.

Implementierung

Damit die Abbildungsgröße beibehalten wird, müssen die Kameraparameter ausgelesen werden. Sie geben Position und Ziel (*Augpunkt* und *Zielpunkt*) der Kamera an.

Der *Blickvektor* ist der Vektor, der sich aus Betrachterposition (bzw. Position der Kamera oder kurz: *Augpunkt*) und dem *Zielpunkt* der Kamera ergibt. Anhand dieser Daten kann nun eine neue Kameraposition errechnet werden. Der neue *Zielpunkt* der Kamera ist das zu fokussierende Objekt, der neue *Augpunkt* wird errechnet, indem ausgehend vom neuen *Zielpunkt* der alte *Blickvektor* abgetragen wird.
Ein Algorithmus dazu sieht folgendermaßen aus:

(1) Lies Kameradaten aus (*Augpunkt*, *Zielpunkt*)
(2) Berechne *Blickvektor*: $\vec{Blickvektor}$:= *Augpunkt* - *Zielpunkt*
(3) Berechne neue Kameraposition:
    neuer *Zielpunkt* := Position des zu fokussierenden Objektes
    neuer *Augpunkt* := neuer *Zielpunkt* + $\vec{Blickvektor}$
(4) Stelle Kameraparameter ein

Das Beispiel (siehe Abbildungen 42) illustriert das Zentrieren von Szeneobjekten. In diesem Beispiel wird linke Knopf in den Mittelpunkt der Szene gerückt.

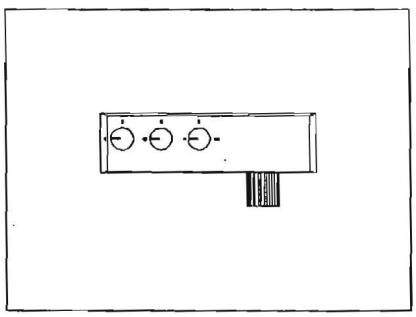

Abb.42 (a): Szene vor dem Zentrieren.

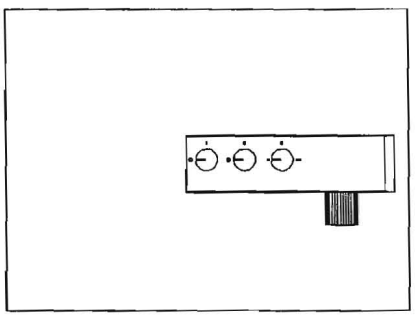

Abb.42 (b): Der linke Knopf wurde zentriert.

4.9 Perspektivenauswahl

Die Wahl der Perspektive, aus der ein Objekt oder eine aus mehreren Objekten bestehende Szene gezeigt werden soll, ist speziell für die automatische

Erzeugung von Illustrationen eines der Hauptprobleme. Das Wissen über Objekte und Ziele, die mit der Illustration beabsichtigt werden, spielen hierbei eine entscheidende Rolle (siehe [Rist, André 90]).
Beispiel für zu berücksichtigende Faktoren in diesem Zusammenhang sind:

- Funktionalität/Zugänglichkeit hervorheben.

  Bei technischen Geräten wird meistens die Seite hervorgehoben, an der die meisten Bedienelemente angebracht sind.

- aktuelle Beobachterposition berücksichtigen.

  Soll ein Objekt vom Benutzer weggedreht werden, so ist in der Illustration darauf zu achten, daß die Seite, die er vorher betrachtet hat, auch weiterhin sichtbar bleibt. Dadurch wird dem Betrachter die Orientierung erleichtert.

- Bewegungsrichtung betonen.

  Soll sich ein Objekt in eine bestimmte Richtung bewegen, wird eine Perspektive, die zur Bewegungsrichtung parallele Objektseiten zeigen.

- Räumlichkeit vermitteln.

  Es wird eine Ansicht gewählt, bei der möglichst viele Seiten des Objektes zu sehen sind, um der Illustration Volumen zu verleihen

- ein bestimmtes Objektteil zeigen.

  Soll ein bestimmtes Objekt, zum Beispiel ein Regler, gezeigt werden, so muß eine Perspektive gewählt werden, aus der der Regler sichtbar ist

- Identifizierung erleichtern.

  Die Perspektive ist zu wählen, aus der ein Objekt vom Betrachter als solches erkannt wird. Das Erkennen eines Bleistiftes fällt beispielsweise leichter, wenn er von der Seite gezeigt wird, als wenn er von vorne oder hinten gezeigt wird.

- Wirkungen erzielen.

  Ein aus der Vogelperspektive betrachtetes Motiv wird als dem Betrachter unterlegen empfunden, ein aus der Froschperspektive betrachtetes Motiv wird als dem Betrachter überlegen empfunden (siehe dazu [Espe 86]).

Auch bei der interaktiven Erstellung von Illustrationen steht der Illustrator vor dem Problem die Perspektive so zu wählen, daß in einer Szene die wichtigen Details zu sehen sind und den Einsatz von illustratorischen Techniken, wie Aufriß, Querschnitt usw. nicht nötig machen. Im ungünstigsten Fall kann es erforderlich sein, daß sogar eine zweite Illustration erstellt wird.
Um den Illustrator bei der Auswahl zu unterstützen und rechenintensive Kamerabewegungen im Grafikeditor zu vermeiden, ist es wünschenswert, ein Hilfsmittel vom System zur Verfügung gestellt zu bekommen.
Auftreten können in diesem Zusammenhang zum Beispiel Aufträge, wie ein Objekt von vorne zeigen oder von dessen Ober- und Vorderseite. Langwieriges Suchen "von Hand" nach der geeignetsten Betrachterposition auf die Szene bleiben dem Illustrator erspart.

Vorgehensweise

Die Information über die Zuordnung der Grundperspektiven zu den Objekten ist durch die umhüllenden Quader und die dazugehörige Struktur gegeben (siehe Kapitel 3).
Davon ausgehend muß eine einheitliche Perspektivenwahl für die Szene gefunden werden. Ist diese Wahl erfolgt, kann, ähnlich dem umhüllenden Quader einzelner Objekte, ein umhüllender Quader für alle Objekte einer Szene, der *Szenenquader*, automatisch erstellt werden. Mögliche Perspektiven auf eine Szene werden in diesem Fall über die Seiten des *Szenenquaders* benannt.
Anhand dieser Zuordnung kann dann die Perspektivenwahl erfolgen.
Soll zum Beispiel ein Objekt in einer Szene von seiner Hinterseite gezeigt werden, so muß die Objektperspektive auf die Szenenperspektive umgerechnet werden und die Szene ist aus der entsprechenden Ansicht zu zeigen.
Wie kann eine Seitenzuordnung zu einer Szene erfolgen? Umfaßt eine Szene nur ein einziges Objektmodell, so ist die Zuordnung offensichtlich. Die Vorderseite der Szene ist dann gleich der Vorderseite des Modells.

Sind jedoch mehrere Teile des Objektes zu sehen, muß eine andere Vorgehensweise gewählt werden. Die Vorderseite eines Objektes muß nicht immer gleich der Vorderseite der Szene sein. Dies gilt beispielsweise für einen Knopf oder Regler, der an der Hinterseite eines Radios angebracht ist. Je nachdem, welche Objekte noch in dieser Szene zu sehen sind, würde die Knopfvorderseite entweder gleich der Szenenvorderseite sein, oder der Szenenhinterseite. Ist das komplette Radio zu sehen, käme der Knopf an der Hinterseite der Szene zu liegen. Ist jedoch nur die Radiohinterseite zu sehen, so würde die Szenenvorderseite der Knopfvorderseite zugeordnet werden. Wird beispielsweise die Rückwand des Radios ausgebaut, so ist die Vorderseite dieses Bauteils gleich der Hinterseite des Radios. Die intuitive Zuordnung der Szenenseiten ist folglich abhängig von den abgebildeten Modellteilen. Sie entspricht offensichtlich der Zuordnung der Baugruppe zu der die Objekte gehören, oder anders ausgedrückt, dem ersten gemeinsamen "Oberkonzept"[6].
Die vier Grundperspektiven (vorne, hinten, oben, unten, rechts, links) können somit festgelegt werden.
Es sollen aber auch Mischperspektiven möglich sein, sodaß eine Betrachterposition auf jedem Punkt einer Kugel um die Szene eingenommen werden kann. Diese lassen sich durch Kombination der Normalenvektoren, die auf den Quaderseiten senkrecht stehen, errechnen.
In Abbildung 43 sind alle möglichen Betrachterpositionen um die linke Quaderseite dargestellt. Diese Positionen lassen sich gleichfalls um die übrigen Seiten einnehmen. Wie in der Abbildung zu sehen ist, bleibt die Entfernung des Betrachters zur Seite gleich, um keine unerwünschten Nebeneffekte auf die Abbildungsgröße zu erzielen.
Soll beispielsweise ein Szene hauptsächlich von der Vorderseite gezeigt werden, so soll verhindert werden, daß sich die Betrachterposition nach einigen Veränderungen an der Perspektive plötzlich auf der Rückseite befindet. In diesem Fall müßten beispielsweise zwei Illustrationen erstellt werden.
Bei der automatischen Erzeugung von Illustrationen ist es jedoch sinnvoll, den Suchraum einzuschränken. Die unendlich vielen Perspektiven können hierbei auf 26 Klassen eingeschränkt werden (Quaderseiten, -ecken und -kanten) (siehe [Rist, André 90]), die sogenannten *Quader-Grundperspektiven.* Im technischen Zeichnen wird die Anzahl der Perspektiven sogar auf sechs

[6] siehe [Strube et al. 93]

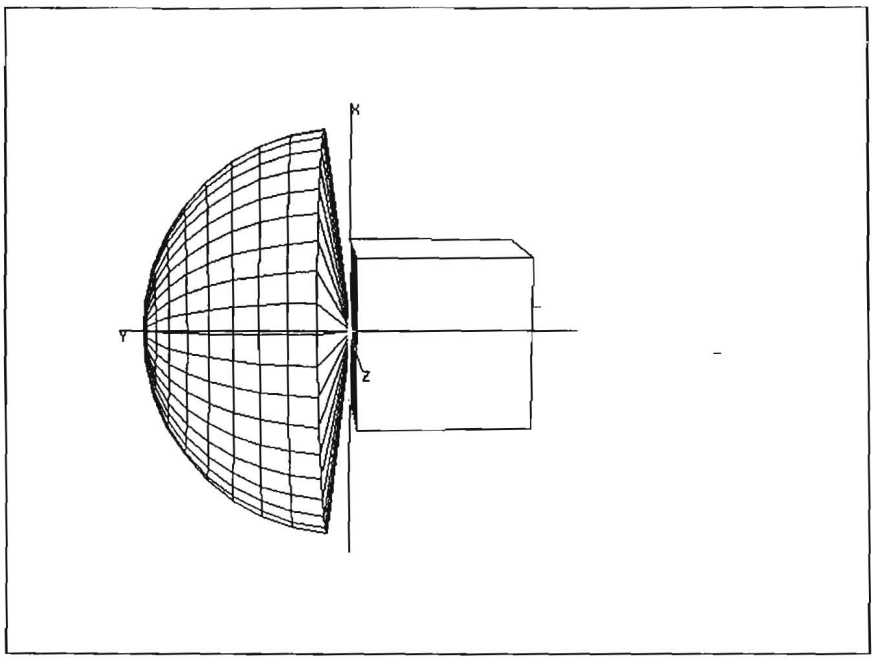

Abbildung 43: Mögliche Betrachterpositionen um die invers dargestellte linke Quaderseite.

beschränkt[7].
Werden obige Illustrationsziele mit einer oder mehreren *Quader-Grundperspektiven* verknüpft, bzw. werden Perspektiven zum Erreichen bestimmter Ziele verboten, kann ein Regelinterpreter in einer konkreten Präsentationssituation eine Entscheidung für eine bestimmte Perspektive treffen. Die so erhaltene *Quader-Grundperspektive* muß dann mit einer (Misch-) Perspektive identifiziert und im Grafikeditor eingestellt werden.

Implementierung

Ein Algorithmus, der die Zuordnung der Perspektiven zu den *Szenequaderseiten* vornimmt, sieht in Pseudocode folgendermaßen aus:

(1) **FOR** Objekte der Szene
(2) **DO**

[7]siehe Abschnitt 2.2

(3) Suche gemeinsame Baugruppe aller Objekte,
bzw. gemeinsames Oberkonzept
(4) Weise *Szenenquader* die Perspektiven der Baugruppe zu

Ein Algorithmus zur Perspektivenwahl muß sämtliche Perspektiven auf einer Art Halbkugel (siehe Abbildung 43) um eine Seite erreichen können. Hierzu wird zuerst eine "Hauptseite" gewählt. Von dieser Seite ausgehend kann dann die Betrachterposition in maximal zwei weitere Richtungen geändert werden, ohne daß widersprechende Richtungen auftreten können.
Folgendes Beispiel verdeutlicht die gewählte Vorgehensweise.
Wird die erste Seite gewählt, z.B. die Vorderseite, ist dies die Hauptseite. Der neue *Zielpunkt* der Kamera ist der Mittelpunkt der Szene bzw. des *Szenenquaders*. Die neue Kameraposition ergibt sich durch addieren des Normalenvektors der Hauptseite mit Länge des alten *Blickvektors* zum neuen *Zielpunkt* der Kamera.
Ausgehend von der Vorderseite soll auch die Oberseite eines Objektes sichtbar werden.
Die neue Betrachterposition wird folgendermaßen errechnet: Zum *Blickvektor* wird der Normalenvektor der Oberseite mit der Länge 1/2 *Blickvektor* addiert. Der Vektor zwischen *Zielpunkt* und errechnetem Punkt wird auf die Länge des *Blickvektors* gekürzt. Der resultierende Punkt ist die neue Kameraposition.
Die verschiedenen Aufträge sollen auf diese Weise kummulierbar sein, so daß ein Illustrator die Perspektive stets weiter verfeinern kann.
Wird nach Vorderseite und Oberseite wieder die Vorderseite als Richtung ausgewählt, so wird ein Punkt auf der Hälfte der Strecke zwischen Hauptseite (Vorderseite) und der aktuellen Kameraposition gewählt. Die Kameraentfernung zum Objekt wird beibehalten.
Eine Konsequenz hieraus ist, daß die Perspektivenwahl nicht kommutativ ist.
In Pseudocode läßt sich der Algorithmus folgendermaßen formulieren:

(1) **IF** Ist die angegebene Seite gültig?
(2) **THEN**
(3) *Blickvektor* = *Augpunkt* - *Zielpunkt*; $\vec{b} := a - z$
(4) $bl :=$ Länge von $\vec{b}$

(5) $\vec{Perspektivenvektor} := \vec{Normalenvektor der Seite}$
(aus PQuader-Struktur)
(6) $\vec{np}$:= normiere $\vec{Perspektivenvektor}$
(7) z := Mittelpunkt der Szene
(8) **CASE**
(9) **OF** Noch keine Hauptseite gewählt?
(10) **THEN** $a := z + bl \cdot \vec{NP}$
(11) **OF** Hauptseite bereits gewählt und
neue Richtung $\neq$ Hauptseite?
(12) **THEN** $a := bl \cdot (normiere$
$< z, (a + \vec{1/2} \cdot bl \cdot \vec{np}) >)$
(13) **OF** Hauptseite gewählt und neue Richtung = Hauptseite
und bereits andere Richtungen gewählt?
(14) **THEN** $a := bl \cdot (normiere < z, a + 1/2 \cdot < \vec{a}, z + \vec{np} \cdot bl > >)$
(15) **OF** Hauptseite gewählt, neue Richtung = Hauptseite
und keine anderen Richtungen gewählt?
(16) **THEN** keine Änderung von a
(17) **ELSE** Fehler

Für eine genaue Beschreibung des Regelinterpreters zur regelbasierten Perspektivenwahl siehe [Zimmermann 93].
Die 26 möglichen *Quader-Grundperspektiven* müssen mit einer Sequenz von Richtungsbefehlen identifiziert und dann im Grafikeditor eingestellt werden. Folgende Präsentationsziele (siehe [Rist, André 90]) werden berücksichtigt:

- Zeige Objekt
- Zeige Objekt von Seite
- Räumlichkeit zeigen
- Bevorzuge Vogelperspektiven
- Bevorzuge Froschperspektiven
- Aktuelle Perspektive

Bei den Abbildungen 44, 45 und 46 wurde die Perspektive vom Regelinterpreter nach Eingabe folgender Präsentationsziele abgeleitet. Sie zeigen, wie sich eine eingestellte Perspektive durch Nachschieben von Präsentationszielen ändert.

- Zeige *knopf-1* von Seite *vorne*
  Daraus wird abgeleitet: Zeige die Szene von vorne.
- Bevorzuge Froschperspektiven
  Wird übersetzt zu: Zeige die Szene zusätzlich von unten.
- Zeige Räumlichkeit
  Bedeutet: Zeige die Szene weiterhin so, daß drei Seiten zu sehen sind.

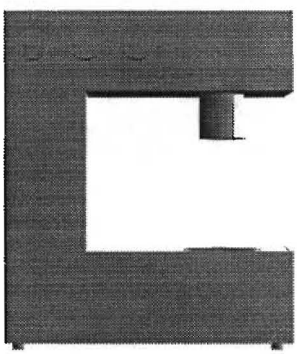

Abbildung 44: Durch den Regelinterpreter abgeleitete Perspektive nach Abarbeiten des Zieles: Zeige *knopf-1* von Seite *vorne*.

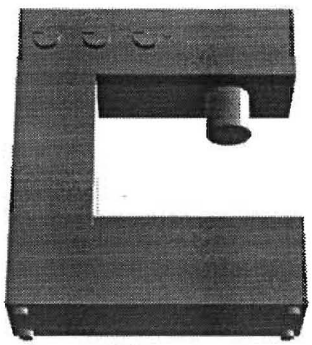

Abbildung 45: Durch den Regelinterpreter abgeleitete Perspektive nach Abarbeiten des Zieles: Bevorzuge Froschperspektiven.

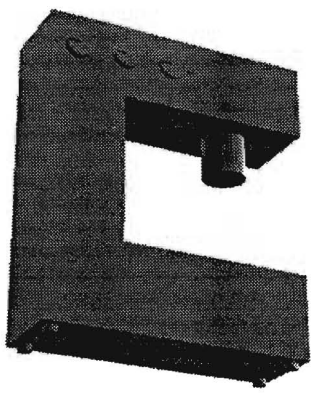

Abbildung 46: Durch den Regelinterpreter abgeleitete Perspektive nach Abarbeiten des Zieles: 3D-Präsentation.

4.10 Erstellen von Insets

Oft steht ein Illustrator vor der Aufgabe, mehrere Handlungen gleichzeitig zu visualisieren, um ein Ziel zu erreichen. Beispielsweise das gleichzeitige Drücken zweier Knöpfe auf der Vorder- und Hinterseite eines Gerätes. Werden diese beiden Handlungen in getrennten Bildern visualisiert, so geht beim Betrachten der Illustration die Gleichzeitigkeit der Handlungen verloren. In diesem Fall greift ein Illustrator auf sogenannte Insets zurück. Dies sind Montagen von ein oder mehreren Bildern ineinander.

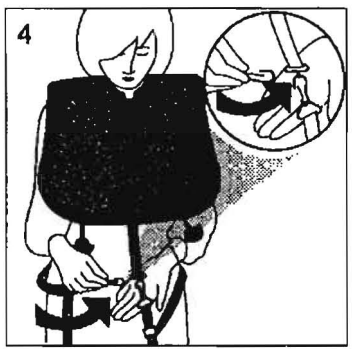

Abbildung 47: das Inset dient zur Ausschnittvergrößerung. Aus: [Lufthansa 93].

Die Technik wird auch dann angewandt, wenn in einem Bild eine Ausschnittvergrößerung (siehe Abbildung 47) notwendig ist, um bei einer Handlung sowohl den Überblick über die Szene zu vermitteln (wo am Objekt findet die Handlung statt), als auch die Details der Handlung (wo genau und/oder wie wird vorgegangen) zu zeigen.
Insets dienen weiterhin dazu, sogenannte "Vorher-Nachher Bilder" zu erstellen, die Ursache und Wirkung, einen Zustand vor und nach einer Handlung etc. visualisieren.
Schließlich werden die Insets auch zum Zeigen von verbotenen Handlungen benutzt. Sie werden dann meistens rot unterlegt oder durchgestrichen.

Vorgehensweise

Ein Inset ist die Montage eines Bildes in ein anderes Bild. Aus diesem Grund müssen für ein Inset zwei Szenen vorhanden sein[8].
Eine Szene, die *Sohnszene* wird "ausgeschnitten" und in die andere Szene, die *Vaterszene* "eingeklebt". Um nicht zu viel Platz in der *Vaterszene* zu beanspruchen wird die *Sohnszene* in den meisten Illustrationen in eine der Ecken kopiert. Die *Sohnszene* muß folglich zuerst angelegt und auf die gewünschte Abbildungsgröße gebracht werden. Dann wird untersucht, welcher Teil des Grafikfensters von der Szene benutzt wird, und ein Bitmap der Region wird erstellt. Dieses Bitmap wird um einen Rahmen erweitert, um die Kopie der *Sohnszene* als illustratorische Technik kenntlich zu machen. Dieses Bitmap wird dann in die für es vorgesehene Ecke der *Vaterszene* kopiert. Es ist möglich eine beliebige Ecke auszuwählen und es können mehrere Insets (≤ 4) erstellt werden.

Implementierung

Das Erstellen der beiden Szenen und Bearbeitungsschritte, wie zum Beispiel Einpassen der Szenen auf eine bestimmte Abbildungsgröße, können mit den bereits beschriebenen Operationen erreicht werden.
Um für die *Sohnszene* herauszufinden, welche Region im Grafikausgabefenster besetzt ist, wird nach dem ersten und letzten gesetzten Pixel in x- und y-Richtung dieses Fensters gesucht. Ein genügend großes "Bitmap-array" um die Szene und den Rahmen aufzunehmen, wird angelegt und die Region wird hineinkopiert.
In der *Vaterszene* wird dieses Bitmap in die vom Benutzer spezifizierte Ecke kopiert.
Ein Algorithmus, der ausgehend von einer aktuellen Szene (*Vaterszene*) eine Insetdarstellung mit einer beliebigen *Sohnszene* erstellt, sieht in Pseudocode folgendermaßen aus:

(1) Bestimme Ecke (oben-rechts, oben-links, unten-rechts, unten-links) und *Sohnszene*
(2) Schalte um zur *Sohnszene*

[8]Zur Vereinfachnung wird bei der Beschreibung der Technik eine Szene, die beispielsweise aus zwei unterschiedlichen Perspektiven gezeigt wird, als zwei getrennte Szenen betrachtet.

(3) Bestimme die besetzte Region im Grafikfenster
(4) Lege Bitmap-array an
(5) Lies die Information aus dem Grafikfenster aus und
kopiere sie in das Bitmap-array
(6) Füge einen Rahmen hinzu
(7) Schalte um zur *Vaterszene*
(8) Kopiere das Bitmap-array in die spezifizierte Ecke des Grafikfensters

In Abbildung 48 wird eine komplexere zusammenhängende Handlung durch ein Inset illustriert. Es verdeutlicht, daß erst die Klappe geschlossen und dann der Knopf gedreht werden soll.

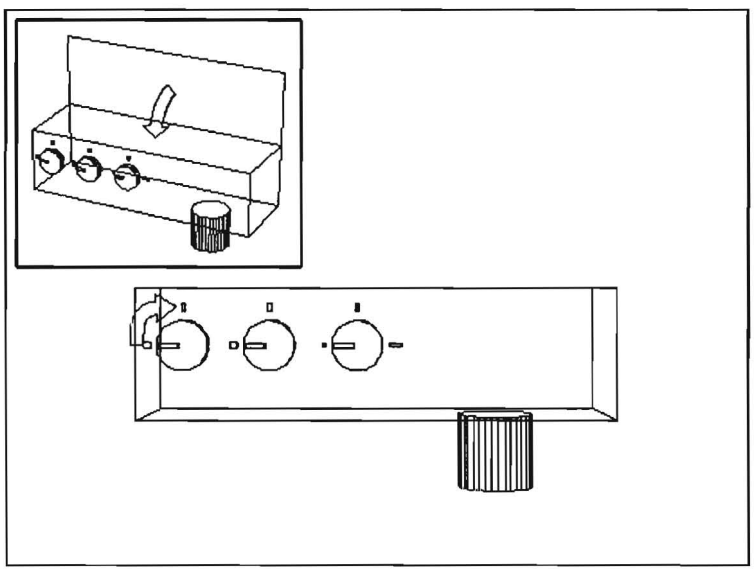

Abbildung 48: Kaffeemaschine mit einem Inset.

4.11 Abstraktion

In den meisten Bedienungsanleitungen wird auf die fotorealistische Abbildung der Modelle zugunsten simplerer Abbildungsarten verzichtet. Wie in

den Abbildungen 35, 37 und 39 zu sehen ist, werden beispielsweise Bleistiftzeichnungen in technischen Betriebs- und Bedienungsanleitungen oft benutzt. Diese sehr vereinfachten Darstellungsarten werden als abstrahierte Darstellungen oder kurz als Abstraktion bezeichnet. Der Grund weshalb abstrahiert wird, ist die Verringerung der Informationsmenge, um sie auf das Notwendigste zu beschränken. Somit wird eine Überforderung des Betrachters vermieden (siehe [Schneider, Krüger 93]).
Eine sehr verbreitet Abstraktionstechnik ist beispielsweise das Stricheln von Linienzügen. "Weniger wichtige" Objekte können durch Anwendung dieser Technik "in den Hintergrund geschoben" werden. Besonders im Zusammenhang mit Ghost Images (siehe Abschnitt 4.5) finden sie oft Verwendung, um Unterscheidungen zu erleichtern.
Diese Technik wird von der Werkbank zur Verfügung gestellt. Sollen mächtigere Abstraktionen generiert werden, ist dies durch das Abstraktionssystem PROXIMA (siehe [Krüger 95]) möglich, das über eine Schnittstelle angesteuert werden kann.

Vorgehensweise

Das Stricheln von Linienzügen kann erreicht werden, indem die Projektion eines Modells nachgezeichnet wird. Ein Grafiksystem, das es ermöglicht, die Art und Weise, wie eine Linie gezeichnet wird, zu spezifizieren, erübrigt die Arbeit an der Projektion. Es müssen nur noch Objekt und Art, bzw. Größe der Striche angegeben werden.

Implementierung

Ein Algorithmus läßt sich in Pseudocode folgendermaßen formulieren:

(1) Spezifiziere das zu strichelnde Objekt
(2) Spezifiziere Art der Strichelung
(3) Übergebe die Parameter dem Grafiksystem

Durch das Stricheln der Ghost Images beim Öffnen des Deckels in Abbildung 49 werden diese als virtuelle Objekte kenntlich gemacht. Sie charakterisieren die Zwischenzustände, die beim Öffnen der Klappe erreicht werden.

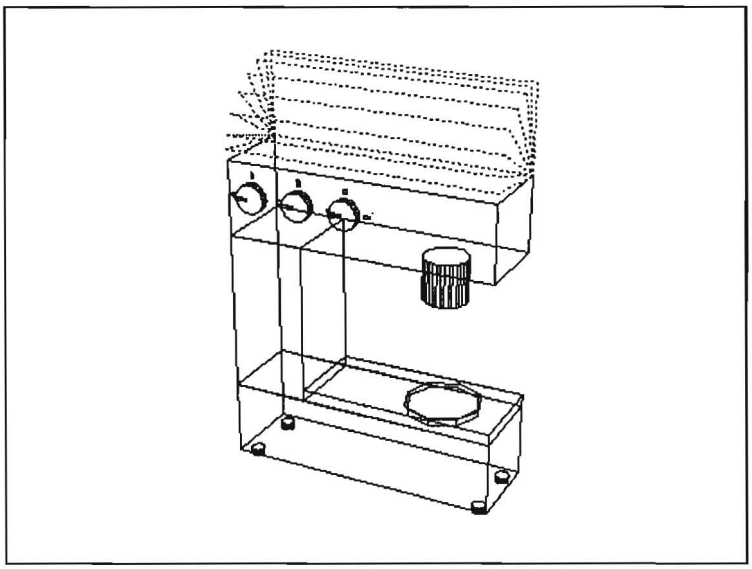

Abbildung 49: Das Stricheln der Ghost Images verdeutlicht ihre Funktion als virtuelle Objekte.

4.12 Evaluierungsoperatoren

Evaluierungsoperatoren sind Operatoren, die die erstellte Illustration hinsichtlich der Ziele, die mit ihr erreicht werden soll, bewerten. Bei der Erstellung von Illustrationen durch einen Grafikdesignexperten scheinen diese Operatoren anfangs überflüssig. Es existieren jedoch eine Anzahl von Regeln über Bildgestaltung, bei deren Beachtung ein Illustrator vom System unterstützt werden könnte.
Wird eine Illustration automatisch erstellt, so ist es nicht möglich, die Ergebnisse sämtlicher illustratorischer Techniken und ihrer wechselseitigen Beziehungen zu antizipieren. Evaluierungsoperatoren geben Aufschluß über die Ergebnisse der angewandten Techniken und sind wesentlicher Bestandteil des in WIP verfolgten Generierungs-Rückkopplungsschleife.
Ein weiterer Vorteil dieser Vorgehensweise ist ein Geschwindigkeitsgewinn. Die Evaluierung der Illustration muß nicht nach Anwendung jeden Schrittes ausgeführt werden, wenn die Illustration noch weiterentwickelt und verändert werden soll. Sie kann weiterhin auf die zur Übermittlung der Information

relevanten Teile der Illustration beschränkt werden [André, Rist 93]).
Elementare Operatoren zur Evaluierung der Szene sind das Vorhandensein eines Objektes in einer Szene *included* und die Sichtbarkeit eines Objektes in einer Illustration *visible*.

Vorgehensweise

In [Feiner, Seligmann 91 (a)] wird eine Technik beschrieben, wie die Sichtbarkeit eines Objektes festgestellt werden kann. Der hardwaremäßig implementierte Z-Puffer wird mit dem Objekt geladen, der Rest des Z-Puffers im Bereich eines umhüllenden Quaders um dieses Objekt wird auf den nächstmöglichen Z-Wert gesetzt. Für jedes übrige Objekt kann das System nun durch den Z-Puffer entscheiden, ob ein Teil dieses Objektes sichtbar ist. Hieraus läßt sich eine binäre Relation für die Sichtbarkeit gewinnen.
Ist jedoch kein Z-Puffer vorhanden, muß auf eine andere Vorgehensweise zurückgegriffen werden. Um das Vorhandensein eines Objektes in einer Szene festzustellen, genügt ein Nachschauen in der Szeneverwaltung (siehe Abschnitt 5.2).
Die Sichtbarkeit eines Objektes kann evaluiert werden, indem ein Strahlenverfolgungsverfahren, ähnlich dem von Abschnitt 4.1, verwandt wird. Hierbei ist es sinnvoll, die prozentuale Sichtbarkeit eines Objektes zu errechnen, um eine genauere Information über den Verdeckungs-, bzw. Sichtbarkeitsgrad zu erhalten.

Implementierung

der Operator *included* läßt sich folgendermaßen implementieren:

```
(1) FOR Spezifiziertes Objekt
(2)   DO
(3)     IF Objekt in der Szene enthalten?
(4)       THEN True
(5)       ELSE Nil
```

Der Opertator *visible* kann durch untenstehendes Verfahren kodiert werden:

```
(1) FOR Spezifiziertes Objekte in Szene
```

```
(2)   DO
(3)     IF Objekt massiv?
(4)       THEN Benutze zugehörigen PQuader für Strahlenverfolgung
(5)       ELSE Benutze Objekt selbst für Strahlenverfolgung
(6)     FOR Alle Teststrahlen vom Augpunkt zu einem Punkt des Objektes
(7)       DO
(8)         IF Schnitt zwischen davorliegenden Objekten und Teststrahl?
(9)           THEN Verminderung des Sichtbarkeitsgrades
(10)          ELSE Objektteil sichtbar
```

5 Konzeption der Werkbank TOPAS

In diesem Kapitel wird der Aufbau der Werkbank **TOPAS** vorgestellt. Zuerst wird auf die Systemarchitektur näher eingegangen, danach wird die Realisierung der Szenenverwaltung näher erläutert. Zum Schluß werden die Bestandteile der interaktiven Benutzeroberfläche vorgestellt.

5.1 Systemarchitektur

Die Architektur wird in dem folgenden Schaubild (siehe Abbildung 50) verdeutlicht. Die Werkbank **TOPAS** baut auf ein 3D-Grafiksystem auf, wie beispielsweise [S-Geometry 90] oder [Müller 95]. Dieses Grafiksystem stellt elementare Funktionen zum Modellieren und Projizieren von 3D-Objekten zur Verfügung. Das Ergebnis der Projektion wird in einem interaktiven Grafikfenster, welches Teil der Benutzeroberfläche ist, dargestellt.
Die externe Wissensbasis enthält die Daten, die zum Erstellen der Weltzustände nötig sind: 3D-Modelle, Zustandsbeschreibungen und Trajektorien. Der Zusammenhang zwischen Weltzuständen und illustratorischen Szenen wurde in Kapitel 3 erklärt.
Da jedoch mehrere Weltzustände und mehrere illustratorische Szenen gleichzeitig existieren, es aber nur einen Arbeitsspeicher gibt, ist eine Szenenverwaltung nötig, die dieses Nebeneinander ermöglicht und die Konsistenz einer Szene sicherstellt. Bei einem Illustrationsauftrag an **TOPAS** werden die zu manipulierende illustratorische Szene und der Weltzustand (bzw. die Weltzustände) in dieser Szenenverwaltung ausgesucht und dem 3D-Arbeitsbereich übergeben. Dieser Arbeitsbereich tauscht über eine Schnittstelle die Daten mit dem zugrundeliegenden externen Grafiksystem aus.
Der Kommandointerpreter übersetzt Illustrationsaufträge in werkbankinterne Befehle, indem er beispielsweise eine illustratorische Technik aus der internen Wissensbasis auswählt. Die interne Wissensbasis enthält darüberhinaus Auswahlregeln (beispielsweise für die Perspektive) und Evalutoren zum Bewerten der Illustration. Dadurch entstandene Veränderungen des 3D-Arbeitsbereiches werden ebenfalls über die Schnittstelle an das externes Grafiksystem weitergegeben.
Ein direkter Zugriff auf den Projektor ist nötig, um 2D-Techniken zu realisieren.

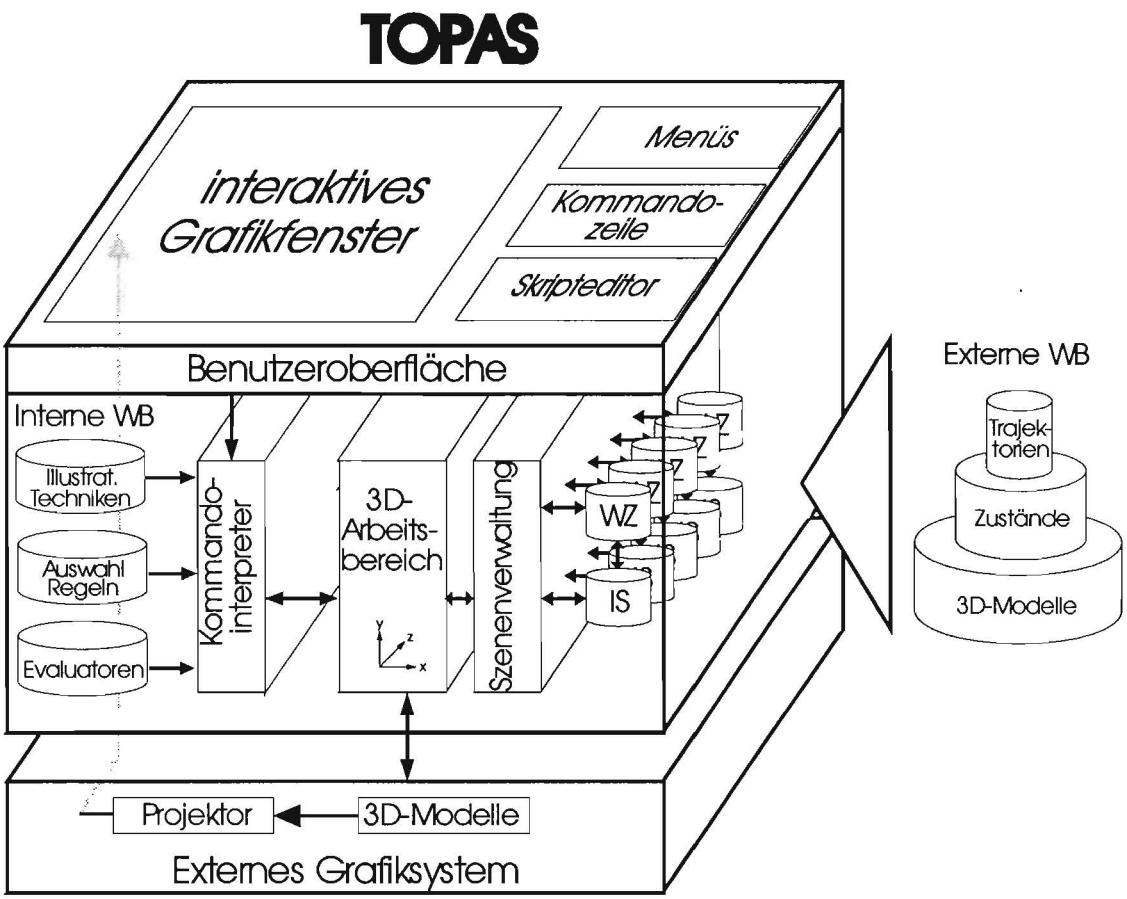

Abbildung 50: Die Architektur von **TOPAS**.

5.2 Szenenverwaltung

Die technische Umsetzung der Hauptbestandteile der Szenenverwaltung, die Weltzustände und die illustratorischen Szenen, werden im Folgenden näher erläutert. Sie sind in LISP als Objekte realisiert.
Ein Weltzustand hat folgende Merkmale:

- Verweis auf den Ur-Weltzustand
- Domäneobjekte
- Zustand der Domäneobjekte

- Verweis auf die illustrorische(n) Szenen

Dadurch läßt sich ein Modell in jedem möglichen Zustand beschreiben.

Die illustratorische Szene hat folgende Merkmale:

- in die Szene aufgenommene (sichtbare) Domäneobjekte der Form: (Objekt, Repräsentant, WZ)
- Perspektivenspezifikation
- Projektionsspezifikation
- History
- Verweis auf die WZ(s)
- Szenenquader
- Objektquader

und einige weitere Slots zur einfacheren internen Verwaltung der illustratorischen Techniken.
Die Ausplittung in Objekt, Repräsentant und WZ wurde gewählt, um ein Objekt in mehreren Weltzuständen in eine illustratorische Szene aufnehmen zu können, beispielsweise als Ghost Image.
Zum Anlegen einer Szene wird zuerst geprüft, ob die Domäne bereits im Arbeitsspeicher verfügbar ist.

- Wenn ja, wird mit Hilfe der Historie zu ihr umgeschaltet.
- Anderenfalls wird das Modell geladen und es wird ein Ur-Weltzustand angelegt, jedoch nur beim erstmaligen Laden der Domäne. Da von diesem Zustand alle anderen Zustände abgeleitet werden können, wird schließlich der gewünschte Weltzustand erstellt und abgeleitet und ein Verweis auf den Ur-Weltzustand eingetragen. Weiterhin werden im *Zustandsslot* die Veränderungen der Domäneobjekte gegenüber dem Ur-Weltzustand vermerkt.

Damit später von einem Weltzustand zu einem anderen umgeschaltet werden kann, wird das Modell mit Hilfe der History auf den Ur-Weltzustand zurückgeführt und dann in den neuen Zustand überführt.
Um das Modell sichtbar zu machen, muß eine illustratorische Szene angelegt werden, die die Domäneobjekte in bestimmten Weltzuständen zeigt. Dazu werden diese Objekte in die illustratorische Szene eingetragen.
In den Slot *Objektquader* der illustratorischen Szene werden die Perspektivenquader der in dieser Szene aufgenommen Objekte, soweit vorhanden, eingetragen. Der Slot *Szenenquader* enthält den aktuellen Perspektivenquader der Szene.
Die Slots für *Perspektive* und *Projektion* sind zu Beginn mit Defaultwerten belegt. Sie können durch die entsprechenden illustratorischen Techniken verändert oder über Menüs direkt modifiziert werden.
Im *History Slot* werden alle Veränderungen der illustratorischen Szene mitprotokolliert. Für jede dieser Veränderung existiert eine inverse Veränderung. Somit existiert ein "Undo-Mechanismus", der ebenfalls zum Umschalten zwischen Szenen benutzt werden kann. Hierbei werden alle Veränderungen schrittweise in umgekehrter Folge rückgängig gemacht, während zum Herstellen der anderen illustratorischen Szene die Veränderungen gemäß der History aktiviert werden.

5.3 Oberfläche

Abbildung 51 zeigt die Oberfläche von **TOPAS**. Sie besteht aus 4 Regionen:

- das Grafikausgabefenster
- die Menüs
- die Kommandozeile
- der Skripteditor

5.3.1 Interaktives Grafikfenster

Im interaktiven Grafikfenster werden die Illlustrationen angezeigt.
Es ist möglich Objekte zu selektieren und ihren Namen vom System zu erfragen.

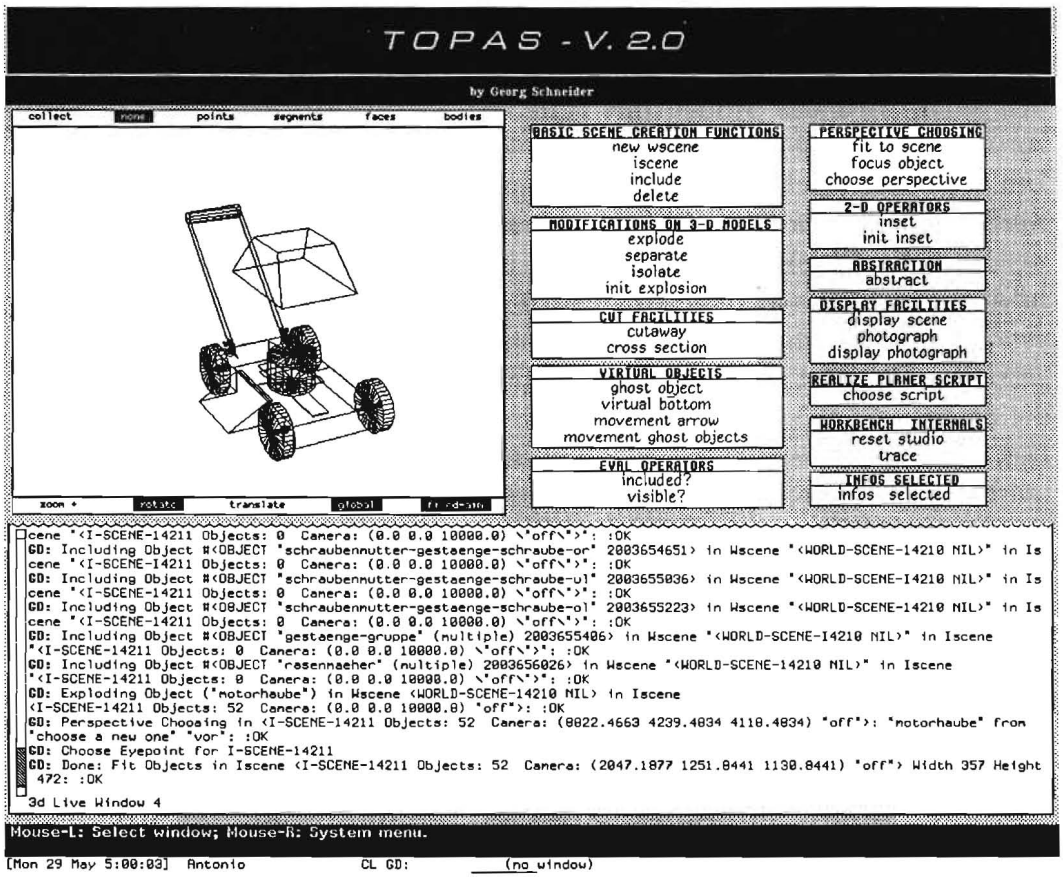

Abbildung 51: Die Oberfläche von **TOPAS**.

Durch Techniken der direkten Manipulation können selektierte Objekte, sofern sie explodiert werden können, entlang ihrer Explosionsachse verschoben werden, wobei Mausbewegungen vom Kommandointerpreter als Bewegungen entlang ihrer Explosionsachse verstanden werden.
Weiterhin kann die Perspektive direktmanipulativ verändert werden, indem das Kamerafenster angeklickt und darin die Kamera um die Objekte einer Szene bewegt wird.

5.3.2 Menüs

Die Menüs sind unterteilt in verschieden Gruppen, entsprechend ihrer Funktionalität. Die wichtigsten Gruppen sind:

- Basisfunktionen zum Erstellen von Szenen

- Explosionstechniken
- Schnittechniken
- Erstellen virtueller Objekte
- Evaluierungsoperatoren
- Perspektivenauswahl
- 2D Operatoren
- Abstraktion

Die einzelnen Menüpunkte verzweigen teilweise weiter in Untermenüs, um die Auswahl von Techniken, Objekten und Präsentationszielen so komfortabel wie möglich zu gestalten. Die Inhalte der Menüs werden situationsabhängig aufbereitet, so daß der Benutzer nur gültige Möglichkeiten zur Auswahl hat und nicht erst durch Werbankfehlermeldungen erfährt, daß die von ihm getroffene Wahl zu diesem Zeitpunkt nicht unterstützt wird.

5.3.3 Kommandozeile

In diesem Fenster hat der Benutzer die Möglichkeit **TOPAS**-Befehle direkt einzugeben.
Weiterhin werden in diesem Fenster Meldungen des Systems (beispielsweise Tracemeldungen) ausgegeben.

5.3.4 Skripteditor

Skripte bestehen aus einer Folge von **TOPAS**-Operatoren. Eine genaue Beschreibung der Operatoren befindet sich im Anhang B. Beipielskripte sind im Anhang C zu finden.
Wenn der Benutzer ein vorgefertigtes oder selbsterstelltes Skript auswählt das realisiert werden soll, wird der Skripteditor geöffnet. Er kann sich nun schrittweise die Abarbeitung dieses Skriptes betrachten. Ist das Abarbeiten des Skriptes abgeschlossen, kann er die daraus resultierende Illustration interaktiv weiterbearbeiten, er kann aber auch das Skript in einen Editor laden, verändern und die Änderungen kontrollieren, indem er dieses Skript nochmals aufruft.

5.4 Voraussetzungen an ein externes Grafiksubsystem

Zur Realisierung der Werkbank ist eine schnelle Hardware nötig, um die Grafiken in einer akzeptablen Zeit zu erstellen.
Wünschenswert ist eine Grafik-Workstation mit einer CPU, die mindestens einige Millionen MIPS ausführt, einer großen Festplatte und einem Monitor, der eine Auflösung von mindestens 1000 mal 800 besitzt. Insbesondere ist ein Hardware Z-Puffer zum Entfernen verdeckter Kanten und Flächen von Vorteil (siehe [Foley et al. 90]). Es muß genügend Speicher zur Verfügung stehen um die Modelle im Arbeitsspeicher zu halten, was von der Art der Objektrepräsentation abhängt.
Die Software muß ermöglichen, daß

- 3D-Objekte erstellt, verschoben und rotiert werden können.
- Objektpunkte, -kanten, -flächen und Objekttransformationen müssen repräsentiert und über eine Schnittstelle zugänglich sein.
- beliebige Perspektiven möglich sind.
- Mehrere Projektionsarten zur Verfügung stehen, so daß die Objekte beispielsweise als Drahtrahmenmodelle oder als Darstellungen, bei denen die verdeckenden Flächen entfernt sind gezeigt werden können.

Möglichkeiten zum Erzeugen realistischer Darstellungen durch Schattierung ("Shading"), "Ray Tracing" oder Strahlungsmethoden ("Radiosity")[9] etc. sind von Vorteil.

[9] siehe [Foley et al. 90]

6 Arbeiten mit TOPAS

Im folgenden Kapitel wird anhand mehrerer Beispiele gezeigt, wie Illustrationen mit **TOPAS** erstellt werden.
Die Beschreibungen können dabei einem Benutzer gleichfalls als Leitfaden dienen, die Illustrationen interaktiv über die **TOPAS**-Oberfläche zu erstellen. Mit ihrer Hilfe kann er mit der Werbank umgehen lernen und die Erstellung der Illustrationen nachvollziehen.

6.1 Beispiele für Illustrationen mit TOPAS

Die mit **TOPAS** erstellten Illustrationen vermitteln Information über Funktionszusammenhänge: Wie funktioniert etwas? Was passiert darin? Um solche Fragen zu beantworten, können die Schnittechniken oder die Explosionstechniken benutzt werden. Aufforderungen, Regler zu drehen und Knöpfe zu drücken, können durch Trajektorienpfeile visualisiert werden. Komplexe Handlungen lassen sich durch Insets und Bilderfolgen zeigen. Dies sind typische Situation für Gebrauchs- und Bedienungsanleitungen, für die **TOPAS** als Illustrationseditor benutzt werden kann.
Die genaue Beschreibung der programmgesteuerten Illustrationserstellung der folgenden Abbildungen befindet sich im Anhang C. Es folgt eine kurze Erläuterung, aus welchen Elementen die Illustrationen bestehen und wie sie erstellt wurden.
Die Bildunterschriften könnten typischerweise aus Bedienungsanleitungen stammen und wurden zur Vervollständigung hinzugefügt.
Mit der Bildsequenz 52 wird beispielhaft demonstriert, wie eine Gebrauchsanweisung zum Bedienen einer Kaffeemaschine mit Hilfe von **TOPAS** illustriert werden könnte.

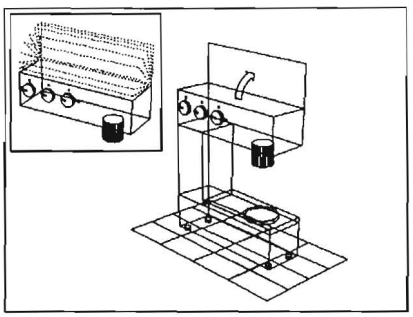

Abb.52 (a): Öffnen Sie den Deckel des Wasserbehälters der Kaffeemaschine!

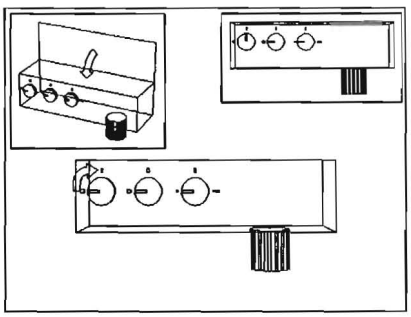

Abb.52 (b): Schließen Sie den Deckel und schalten Sie die Maschine ein!

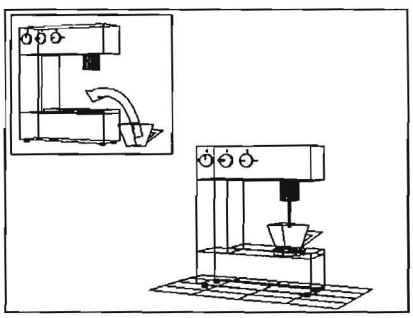

Abb.52 (c): Stellen Sie die Tasse unter den Kaffeeauslauf!

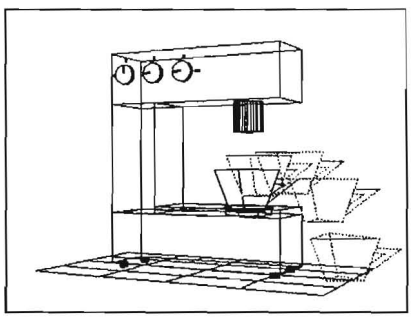

Abb.52 (d): Der Kaffee ist fertig. Sie können die Tasse entfernen.

Für das Bild 52 (a) müssen zwei Szenen angelegt werden, eine für das Inset in der oberen linken Ecke, eine andere für das Hauptbild. Im Hauptbild muß die komplette Kaffeemaschine im Weltzustand: "Klappe offen" zu sehen sein. Es muß ein Trajektorienpfeil für die Öffnenbewegung und ein virtueller Boden hinzugefügt werden. Die Kaffeemaschine muß danach auf ein geeignete Abbildungsgröße gebracht werden. Das Inset zeigt ebenfalls den Weltzustand "Klappe offen", im Gegensatz zum Hauptbild wurde hier nur das Oberteil der Kaffeemaschine illustriert. Hierfür muß eine neue Szene angelegt werden. Die Öffnenbewegung wird durch multiple, gestrichelte Ghost Images verdeutlicht. Das Oberteil muß danach auf eine kleinere Abbildungsgröße gebracht werden, damit es in die Ecke paßt. Schließlich muß diese Illustration als Inset in das Hauptbild aufgenommen werden.

Der Illustration 52 (b) liegen drei Szenen zugrunde, in denen jeweils das Oberteil der Kaffeemaschine in unterschiedlichen Zuständen zu sehen ist. Das Inset in der oberen linken Ecke zeigt die Kaffeemaschine im Weltzustand: "Deckel offen". Eine Szene zur Illustration des Oberteils der Kaffeemaschine muß angelegt werden. Ein Trajektorienpfeil für die Schließenbewegung wird hizugefügt und die Illustration auf eine passende Abbildungsgröße gebracht. Das rechte obere Inset zeigt die Kaffeemaschine im Weltzustand: "Kaffeemaschine angeschaltet". Eine Szene zu deren Illustration muß angelegt und ebenfalls auf eine geeignete Abbildungsgröße gebracht werden. Das Hauptbild besteht aus der Kaffeemaschine im Ur-Zustand mit geschlossenem Deckel und alle Knöpfen auf Ausgangsposition. Nachdem das Objekt auf eine geeignete Abbildungsgröße gebracht wurde, werden die anderen beiden Szenen als Insets hinzugefügt.
Das Bild 52 (c) besteht ebenfalls aus zwei Szenen, die die komplette Kaffeemaschine im Weltzustand: "Kaffeemaschine angeschaltet" illustrieren. In der Szene für das Inset muß die Tasse in der Position: "Neben der Kaffeemaschine" hinzugefügt werden. Weiterhin ist ein Trajektorienpfeil für die Bewegung: "Tasse unter den Auslauf stellen" hinzuzufügen. Das Hauptbild besteht aus Kaffeemaschine mit Tasse in Position: "Unter dem Auslauf". Auslaufender Kaffee muß hinzugefügt werden. Ein virtueller Boden muß ebenfalls erstellt und hinzugefügt werden. Danach muß die Kaffeemaschine auf eine geeignete Abbildungsgröße gebracht und der Einschaltknopf fokussiert werden, um Platz für das Inset zu schaffen. Schließlich wird die erste Szene als Inset in die Illustration aufgenommen.
Das Bild 52 (d) zeigt die Kaffeemaschine im Weltzustand: "Kaffeemaschine angeschaltet". Die Tasse in Position: "Unter dem Auslauf" wird hinzugefügt, ebenso mutiple Ghost Images für die Bewegung: "Tasse wegnehmen", die danach gestrichelt gezeichnet werden. Schließlich muß noch ein virtueller Boden hinzugefügt und die Szene auf die gewünschte Abbildungsgröße gebracht werden.

Die Sequenz 53 könnte aus einer Reparaturanleitung stammen. Erzeugt das Modem keinen Signalton, soll der Lautstärkeregler nach rechts gedreht werden. Ist der Fehler dadurch nicht behoben, ist das Modem zu öffnen und der Lautsprecher, der hinter dem Transformator liegt, auszutauschen.

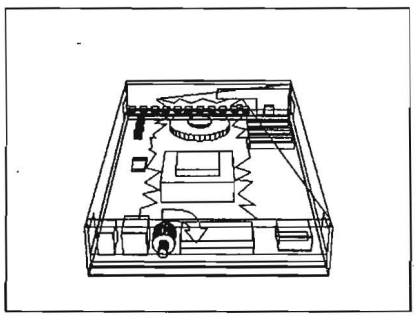

Abb.53 (a): Stellen sie die Lautstärke Modem höher.

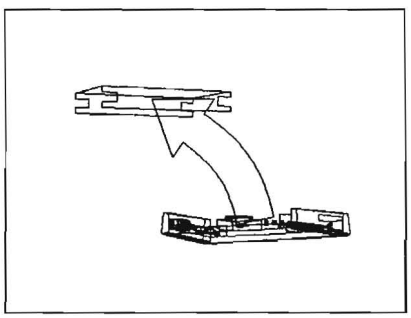

Abb.53 (b): Öffnen sie das Modem.

Das Bild 53 (a) zeigt das Modem im Ur-Zustand von hinten. Ein Trajektorienpfeil zeigt die Bewegung: "Lautstärkeregler nach rechts". Zwei Aufrißdarstellungen auf den Transformator und den Lautsprecher zeigen das Innenleben und die Anordnung der Bauteile auf der Platine.
Im Bild 53 (b) ist das Modem im Weltzustand: "Deckel geöffnet" zu sehen. Ein Trajektorienpfeil verdeutlicht die Öffnenbewegung.
Die Bildsequenz 54 zeigt dem Benutzer eines Rasenmähers, wie er den Ventilator austauschen und das Rad wechseln kann.

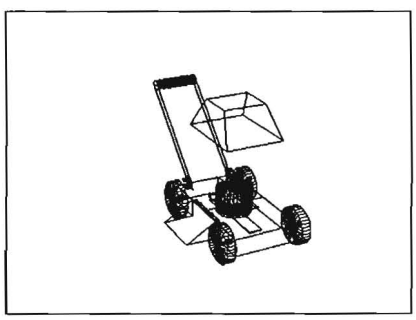

Abb.54 (a): Die Abbildung zeigt die Position des Motors und des Ventilators.

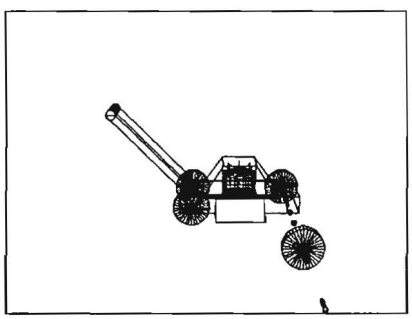

Abb.54 (b): Bauen sie das Fahrwerk aus, um den Reifen zu wechseln.

Abbildung 54 (a) zeigt, wie mit Hilfe der Explosionstechnik Motor und Ventilator sichtbar gemacht wurden.
Das Bild 54 (b) illustriert durch eine Explosionsdarstelllung den Zusammenbau des Fahrwerks.
Im Anhang A befinden sich Beipiele für das Arbeiten mit dem Skripteditor, das menügesteuerte Erstellen von Illustrationen und für das direktmanipulative Arbeiten im interaktiven Grafikfenster.

6.2 Abstraktionsniveau der TOPAS-Befehle im Vergleich zu Grafikeditoren

Am Beispiel der Querschnittdarstellung soll demonstriert werden, in wieviele Einzelbefehle ein **TOPAS**-operator im zugrundeliegenden Grafiksystem zerfällt (siehe [S-Geometry 90]). Soll der Querschnitt für einen Zylinder mit 76 Seiten berechnet werden, um ihn quer zu durchtrennen, so sind 76 Befehle notwendig um die Flächendaten zu extrahieren. Danach werden 75 Seiten entfernt und 74 neue Seiten erstellt. Diese Seiten werden danach sichtbar gemacht. Dem zufolge entspricht schon bei einem einfachen Beispiel, wie einem Zylinderobjekt, ein Werkbankbefehl $76 + 75 + 74 + 74 = 299$ Befehlen des Grafiksystems, die ein Benutzer von Hand eingeben müßte. Rechenbefehle wurden nicht berücksichtigt. Ein komplexeres Modell, wie der Rasenmäher, besteht aus über 110 Teilen.

7 Einsatzfelder von TOPAS

In diesem Kapitel werden die verschiedenen Einsatzfelder für die Werkbank **TOPAS** beschrieben. Zuerst werden die Möglichkeiten aufgezeigt, **TOPAS** als Assistenzsystem einzusetzen, danach wird der Einsatz von **TOPAS** zum vollautomatischen Erstellen von Illustrationen erläutert (siehe hierzu auch [Schneider 95 b]).

7.1 TOPAS als Assistenzsystem

TOPAS kann als Assistenzsystem eingesetzt werden. Hierbei interagiert die Werkbank mit externen Systemen, die seine Fähigkeiten beispielsweise zur Szenenerstellung und Verwaltung ausnutzen.

7.1.1 Aufsatz für CAD-Systeme

TOPAS geht in seiner Funktionalität hinsichtlich der Erstellung von Illustrationen weit über die bisherigen Grafikeditoren hinaus. Der naheliegenste Einsatzbereich ist daher **TOPAS** als Werkbank zum Erstellen von Illutrationen zu verwenden. Hierbei dient die Werkbank als hochentwickelte Editierhilfe, die einem menschlichen Illustrator in die Lage versetzen, prägnante und aussagekräftige Präsentationen ohne großen Aufwand zu erstellen. Der Illustrator wird dabei von automatischen Funktionen unterstützt, es stehen ihm aber auch die interaktiven Eingriffsmöglichkeiten zur Verfügung, um die Illustration nach seinem Geschmack zu verändern.

7.1.2 Entwicklungsumgebung zur Operationalisierung von Illustrationstechniken

Als Beispiel sei hier das Abstraktionssystem PROXIMA (siehe [Krüger 95]) genannt. Wie bereits in Abschnitt 4.11 erwähnt, können mit der Werkbank einfachere Abstraktionen, wie das Stricheln von Objektumrandungen, vorgenommen werden.
Über eine Schnittstelle zu diesem Abstraktionssystem können Daten ausgetauscht werden. Die mit **TOPAS** erstellte Szene wird dann von PROXIMA weiterbearbeitet.

7.1.3 Key-Frame Editor

Eine weitere Anwendung in Verbindung mit technischen Dokumentationen ist die Erstellung von Animationen. Bei der herkömmlichen Art der Erstellung von Animationen werden "Key-frames" angelegt, die die wichtigsten Eigenschaften einer Szene, wie Kameraposition, Lichter, etc. zu diskreten Zeitpunkten beschreiben. Von diesen Key-frames ausgehend, werden die dazwischenliegenden Zustände durch Interpolationstechniken abgeleitet.
Eine ähnliche Vorgehensweise wird auch im Animationssystem BETTY (siehe [Butz 94]) verwandt, wobei die einzelnen Szenen mit Hilfe von **TOPAS** angelegt werden.
Die Werkbank zum Erstellen der Key-frames zu verwenden, bietet mehrere Vorteile:

- **TOPAS** stellt die internen Repräsentationen aller Eigenschaften der Objekte einer Szene zur Verfügung. Die Zwischenzustände können direkt aus diesen Daten interpoliert werden.

- Die Ausführung einiger Illustrationstechniken von **TOPAS** können direkt animiert werden.

Als Beispiel sind hier die Explosionstechniken zu nennen. Durch den inkrementellen Ansatz bei der Erstellung explodierter Darstellungen wird von **TOPAS** nicht nur die fertige Darstellung, es werden auch alle Zwischenzustände errechnet. Davon ausgehend, kann die Explosion direkt animiert werden, wobei die Zwischenzustände als eine Art Plan für die Animation fungieren.
Diese Vorgehensweise ausnutzend, animiert beispielsweise das System BETTY die Explosionsdarstellungen.
Für andere Illustrationstechniken müssen jedoch kompliziertere Interpolationsalgorithmen benutzt werden. Auf- und Querschnitte können beispielsweise nur durch Morphing Techniken (siehe [Kent et al. 92]) animiert werden.

7.2 TOPAS als Teilkomponente intelligenter Präsentationssyteme

TOPAS wird als Grafikgenerierungskomponente für multimediale Dokumente, wie sie von WIP erzeugt werden, eingesetzt.

7.2.1 WIP

Im folgenden Kapitel wird das System WIP beschrieben und der Einsatz von **TOPAS** in diesem Intellimediasystem erläutert.

7.2.1.1 Überblick

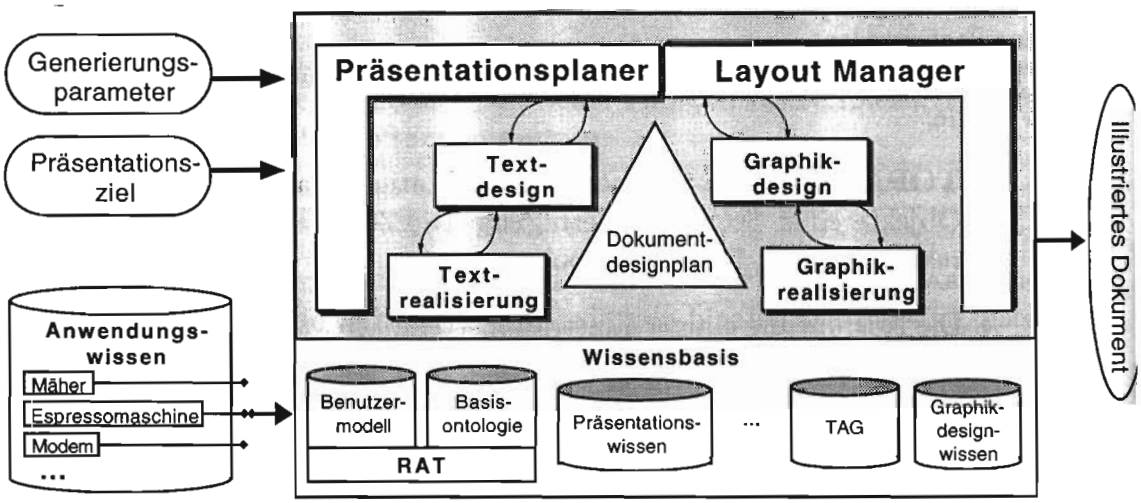

Abbildung 55: Die Architektur von WIP

Das System WIP (siehe [Wahlster et al. 93] und Abbildung 55) wurde als unidirektionale Schnittstelle zwischen einer Anwendung und einem Benutzer. Die Information, die durch das System präsentiert werden soll, wird durch die Anwendung bereitgestellt, dabei kann es sich um ein Expertensystem, ein Hilfesystem oder einen Leitstand handeln. Abhängig von Generierungsparametern, wie Benutzerwissen, Sprache und Ausgabemedium wird von WIP eine multimediale Präsentation erstellt.
Als Beispieldomäne dienen illustrierte Bedienungsanleitungen für technische Geräte, wie Rasenmäher, Espressomaschine und Modem.
WIP hat Zugriff auf anwendungsspezifisches Wissen, wie eine Ontologie der Domänenobjekte und Aktionen. Pläne über Problemlösen in der Domäne

gehören ebenfalls zum Wissen von WIP, genauso wie Beschreibungen der Geometrie und Konfiguration der 3D-Objekte.
Die Hauptkomponenten von WIP sind:

- Präsentationsplaner (siehe [André, Rist 93]).
  Er entscheidet über die Inhalte, die Kombination der Modi und wählt die entsprechenden Generatoren für den gewählten Modus aus.
- Layout Manager (siehe [Graf 93]).
  Er stellt die generierten Ausgaben in-einem Dokument zusammen.
- Textgenerator (siehe [Kilger, Finkler 95]).
- Grafikgenerator (siehe [Rist, André 92]).
- Komponenten zur Verwaltung und zum Schlußfolgern über repräsentiertes Domänenwissen (siehe [Heinsohn et al. 93]).

Die Generatoren bestehen aus einer Design- und einer Realisierungskomponente. Sowohl Text als auch Grafikgenerator werden vom Planer angesteuert. Das Resultat des Planungsprozesses ist ein hierarchisch strukturierter Plan des Dokumentes. Die Wurzel ist dabei ein komplexes Ziel, während die Blätter direkt als Eingabe für die Generatoren dienen.
Ein Merkmal von WIP ist es, daß die Generatoren ihre Aufträge stückweise erhalten und sofort mit der Auswertung beginnen. Diese Vorgehensweise spart Zeit, was für on-line Präsentationen unerläßlich ist.

7.2.1.2 Der Grafikgenerator von WIP

Im Folgenden wird näher auf die Arbeitsweise des *Grafikgenerators* eingegangen, um den Einsatz von **TOPAS** in diesem Prozeß zu verdeutlichen. Der *Grafikgenerators* besteht aus den Teilen: *Grafikdesignkomponente* und *Grafikrealisierungskomponente*. Die *Designkomponente* erhält die Aufträge stückweise von einem *Präsentationsplaner* und steuert den *Grafikrealisierer* an. Sie ist folglich verantwortlich für die Art und Weise der Realisierung dieser Aufträge durch den *Grafikrealisierer*, sie bestimmt beispielsweise, ob ein Bild genügt oder ob mehrere Bilder erstellt werden müssen, um die Aufträge

auszuführen. Weiterhin ist sie für die Kommunikation mit dem *Planer* über die Realisierung der Aufträge verantwortlich, um Verweise von der Sprache auf die Grafik zu ermöglichen.
Ein Bild kann als Komposition aus einem Rahmen und einer Menge von Abbildungen in diesem Rahmen angesehen werden. Es kann zwischen verschiedenen Abbildungstypen unterschieden werden:

- Abbildungen von 2D Konzepten, wie Punkten, Linien, etc. die in 3D Illustrationen oft als metagrafische Objekte benutzt werden.
- Abbildungen, die durch die Darstellung von Zeichen oder Symbolen entstanden sind.
- Abbildungen, die durch die Projektion eines 3D Modells auf eine 2D Ebene entstanden sind.

Um Grafiken zu erzeugen, die verschiedene Abbildungstypen enthalten, wurde ein *Grafikrealisierer* entwickelt, dessen Operatoren sich in drei Klassen zerteilen lassen:

- Operatoren zum Erstellen und Manipulieren von Drahtrahmenmodellen von 3D Objekten.
- Operatoren, die Projektionsparameter einschränken und Drahtrahmenmodelle in Abbildungen überführen.
- Operatoren, die diese Abbildungen manipulieren.

Da bei der Grafikgenerierung Seiteneffekte von Operatoren auf bereits erfüllte Ziele schwierig zu antizipieren sind, müssen es neben den Operatoren zu Erstellung von Illustrationen Operatoren exisitieren, die die Illustrationen bewerten. Aus diesem Grund basiert die Grafikgenerierung bei WIP ebenfalls auf dem *generate-and-test* Ansatz.
Das Designwissen wird in WIP als *grafische Constraints* abgelegt. Die Motivation dafür ist, daß Präsentationsaufträge sich nicht direkt auf grafische Präsenationen beziehen, sondern auf eine Menge von *grafischen Constraints*. *Grafische Constraints* beschränken Lichtquellen, Projektionsart und Abbildungen in einem Bild. Folglich wird keine Illustration erstellt, sondern ein breiter Lösungsraum von gültigen Designmöglichkeiten für eine Illustration

aufgespannt. Somit kann der *Grafikdesigner* flexibel auf die Präsentationsaufträge reagieren und, wenn möglich, mehrere in einer Illustration erfüllen. *Grafische Designstrategien* werden dazu benutzt, eine Verbindung zwischen elementaren Aufträgen und *Constraints* für die Grafik herzustellen. Einige *Constraints* sind direkt mit Operatoren der *Realisierungskomponente* verbunden, andere mit weiteren *Designstrategien*. Durch Hinzunahme der *grafischen Designstrategien*, ist das Grafikdesign prinzipiell zu einem zielgerichteten Prozeß geworden.
Präsentationsaufträge wurden mit Constraints verknüpft und nach mehreren Verfeinerungen wird eine Sequenz von Operatoren für die *Grafikrealisierung* erhalten.
Um die Flexibilität des *Grafik Designers* zu erhalten, sollten auf jeden Fall so wenig *Constraints* wie möglich für einen Auftrag aufgestellt werden, damit der Lösungsraum nicht unnötig eingeschränkt wird.

7.2.1.3 Die Rolle von TOPAS bei der Grafikgenerierung

Der Präsentationsplaner erteilt Visualisierungsaufträge an den Grafikgenerator. Die Grafigdesignkomponente wählt daraufhin Enkodierungen für diese Aufträge aus, wobei ein strategischer Ansatz verfolgt wird. Als Ergebnis gibt die Grafikdesignkomponente Realisierungspläne an die Grafikrealisierungskomponente weiter, die die Form von Skripten haben.
In dem System WIP wird **TOPAS** als Grafikrealisierungskomponente eingesetzt.
Da Illustrationstechniken Seiteneffekte, die schwierig vorherzusehen sind, haben können, sind Design- und Realisierungsphase verzahnt. Evaluierungsoperatoren werden dazu benutzt, um herauszufinden, ob die angewandten Techniken die gewünschten Auswirkungen auf die Illustration haben.

7.2.2 PPP

Neuerliche Erweiterungen des WIP Systems zielen auf Präsentationen ab, die den Betrachter einer Präsentation aktiv mit einbeziehen (siehe [Rist, André 94]). An bestimmten Punkten einer Präsentation wird der Betrachter dazu aufgefordert, selbst das Modell zu untersuchen. Beispielsweise erstellt das System zuerst ein Bild eines Objektes, wie im WIP System und fordert dann den Betrachter auf, das dargestellte Objekt mit den Illustrati-

onstechniken und den Techniken zur direkten Manipulation, die von **TOPAS** zur Verfügung gestellt werden, zu untersuchen.

7.2.3 Illustrationen als Benutzerschnittstelle

Für viele technische Geräte wurden Programme zur Simulation ihrer Funktionalität entwickelt, die Lehr- und Trainingsaufgaben unterstützen. Da Knöpfe, Schalter, Schieberegler, etc. als Benutzerschnittstelle dienen, liegt es nahe, Abbildungen von ihnen als Schnittstelle zu einem Simulationssystem zu benutzen. Die am weitesten verbreiteten Beispiele hierfür sind Flugsimulatoren. Wird **TOPAS** an ein Simulationssystem angeschlossen, kann der Benutzer mit diesem System durch Illustrationen kommunizieren.
Dies kann anhand eines einfachen Simulationsprogrammes für die Modem Domäne verdeutlicht werden. Verschiedene Knopfpositionen sind mit unterschiedlichen internen Zuständen des Modems verknüpft. Jeder Zustand wird als Vektor der verschiedenen Objekteigenschaften dargestellt, die durch neue Eingaben verändert werden können. So beeinflußt beispielsweise eine Änderung der Position des "An/Aus" Schalters auf die Postion "Aus" nicht nur den aktuellen Zustand des Schalters, sondern auch den Zustand der LED's, die dadurch erlöschen.
Durch die Verwendung von **TOPAS** als Schnittstelle zum Simulationsprogramm werden zwei Arten der Veränderung der Grafik durch den Benutzer ermöglicht.

- Veränderungen, die als Eingabe für die Simulation interpretiert werden und solche,
- die dazu dienen, die Benutzerschnittstelle nach dem eigenen Geschmack zu gestalten.

Zum Beispiel kann der Benutzer die Aufrißtechnik benutzen, um im Inneren verborgene Schalter zugänglich zu machen. Hiermit würde er die Schnittstelle nach seinen Wünschen gestalten. Nun kann er Veränderungen dieser Schalter vornehmen, wobei sich die Arbeitsart von **TOPAS** vom "Illustrationsmodus" auf den "Simulationsbetrieb" ändert. Diese Veränderungen werden als Eingabe für die Simulation benutzt. Durch Zugriff auf die Wissensbasis können **TOPAS** die Manipulationen auf sinnvolle Veränderungen beschränkt werden, umgekehrt müssen nach jeder Manipulation die internen Zustände aktualisiert werden.

8 Zusammenfassung und Ausblick

Dieses Kapitel faßt die mit der Konzeption und Implementierung erzielten Ergebnisse der Werkbank **TOPAS** zusammen.
Desweiteren werden einige Erweiterungsmöglichkeiten aufgezeigt.

8.1 Leistungsumfang von TOPAS

Ein wichtiges Merkmal der Werkbank ist die Domänenunabhängigkeit. Zur Zeit wird mit drei verschiedene Testdomänen gearbeitet: Ein Rasenmäher, ein Modem und eine Kaffeemaschine.
Die Szenenverwaltung der Werkbank erlaubt es sowohl mehrere Szenen einer Domäne, als auch die verschiedenen Domänen zur gleichen Zeit zu bearbeiten. Durch einfaches Umschalten kann die gewünschte Szene aktiviert und weiterbearbeitet werden.
Die situationsabhängige Menüsteuerung von **TOPAS** ist ein Punkt, der die Bedienerfreundlichkeit der Oberfläche auszeichnet. Einem Benutzer werden für die von ihm ausgewählte Funktion ausschließlich die Objekte in Menüs angezeigt, die zu diesem Zeitpunkt manipuliert werden können. Dies verhindert Verwirrungen beim Benutzer und vermeidet weitestgehend Werkbankfehlermeldungen.
Grundsätzlich kann **TOPAS** jedoch in Bezug auf zwei Gesichtspunkte betrachtet werden,

- dem Repertoire an implementierten Operatoren zum Erstellen von Illustrationen und
- den unterstützten Betriebsmodi, die ein effektives und komfortables Arbeiten ermöglichen.

Um die Funktionalität der Werkbank **TOPAS** kurz zusammenzufassen, werden im Folgenden die wichtigsten Operatoren nochmals aufgeführt:

8.1.1 Operatoren

Es steht eine Vielzahl von 3D-Techniken zur Verfügung:

- Explodieren, Separieren, Isolieren

- Aufrißtechnik
- Querschnitt
- Ghost Images (einfach oder entlang einer Trajektorie)
- Virtuelle Böden
- 3D-Trajektorienpfeile
- Perspektivenauswahl (am Objekt orientiert oder an Präsentationszielen orientiert)

Es werden aber auch verschiedene 2D-Techiken unterstützt:

- Insets
- einfache Abstraktion durch Stricheln von Objekten
- "Fotografieren" und Anzeigen von Szenen durch Erstellen von Bitmaps

Ein Illustrator wird weiterhin durch Operatoren zum Bewerten einer Szene unterstützt, den sogenannten Evaluierungsoperatoren:

- Ist ein Objekt in einer Szene enthalten?
- Prozentuale Sichtbarkeit eines Objektes

Weiterhin steht ein Trace Operator zur Verfügung, der die interne Verarbeitung der Befehle verdeutlicht.

8.1.2 Betriebsmodi

TOPAS unterstützt verschiedene Betriebsmodi:

Realisieren von Skripten

Beim interaktiven Arbeiten mit der Werkbank kann der Benutzer zusätzlich von den Vorteilen der programmgesteuerten Arbeitsweise profitierten, indem er Skripte mit Hilfe der **TOPAS**-operatoren erstellt. Durch Anwählen des entsprechenden Menüpunktes kann er die schrittweise Realisierung des Skriptes verfolgen.

Post Editing

Die oben beschriebenen Skripte können weiterhin als eine Art Makro angesehen werden, die vordefinierte Illustrationen erstellt. Diese "Illustrationsvorschläge" kann der Illustrator dann interaktiv weiterbearbeiten und verfeinern.

Direkte Manipulation

Durch Techniken zur direkten Manipulation können bei **TOPAS** beispielsweise Explosionszeichnungen durch anklicken eines explodierten Teiles bearbeitet werden. Mausbewegungen werden hierbei als Bewegungen des angewählten Teiles auf der Explosionsachse interpretiert. Damit kann ein Illustrator Explosionsdarstellungen seinen Vorstellungen entsprechend modifizieren.
Der Benutzer kann durch Anklicken des interaktiven Grafikfensters die Kamera um Szeneobjekte herumbewegen und somit die Perspektive direktmanipulativ verändern.
Weiterhin kann er, ähnlich wie bei Hypertextdokumenten (siehe [Hypercard]) Objekte im Grafikausgabefenster selektieren und ihre Namen erfragen.

Programmierschnittstelle

Über eine Programmierschnittstelle in der Programmiersprache "LISP"[10] kann **TOPAS** als Subsystem für Präsentationssysteme, wie WIP oder PPP fungieren. Alle Operatoren werden exportiert. Zugriffe auf die internen Datenstrukturen sind ebenfalls möglich.

8.2 Ausblick

In dem folgenden Kapitel werden mögliche Erweiterungen der Techniken und Operationen von **TOPAS** erläutert.

8.2.1 Integration weiterer 3D-Techniken

Der Funktionsumfang von **TOPAS** könnte hinsichtlich der vorhandenen 3D-Techniken ausgeweitet werden. Zum Abstraktionssystem PROXIMA existiert bereits eine Schnittstelle, über die Daten ausgetauscht werden können. In einer nachfolgenden Version von **TOPAS** könnten die Techniken des Abstrak-

[10] siehe [Steele 90]

tionssystems komplett in die Werkbank integriert werden, so daß die Werkbank alle Abstraktionsmöglichkeiten, sowohl programmgesteuert als auch interaktiv, zur Verfügung stellt.

8.2.2 Integration weiterer 2D-Techniken

Die Integration des AnnA II Systems (siehe [Zimmermann 93]) zum automatischen Annotieren von Grafiken würde den Funktionsumfang von **TOPAS** hinsichtlich der 2D-Techniken erweitern. Somit stünde eine Werkbank zur Verfügung, mit der Grafiken nicht nur erstellt, sondern auch annotiert werden könnten. Speziell als vorgeschrittener Grafikeditor zum Erstellen von Illustrationen im Bereich technischer Bedienungsanleitungen wären dann alle wichtigen Funktionen vorhanden.

8.2.3 Beleuchtung

Ein Illustrator könnte bei der Auswahl und Plazierung der Beleuchtung einer Szene unterstützt werden. Bisher bleibt das Ausleuchten der Szene dem Benutzer überlassen. Neben dieser Unterstützung können Lichter zum Fokussieren von Objekten, beispielsweise mit Hilfe von Punktstrahlern benutzt werden. In Verbindung mit Farbe können Lichter dazu benutzt werden, Gegenstände farbig anzustrahlen, um Temperaturen, Gefahren etc. zu visualisieren.

8.2.4 Anbindung an kommerzielle CAD-Systeme

Eine Anbindung an kommerzielle CAD-Systeme, beispielsweise durch Importmöglichkeiten des AutoCAD Formates DWG (siehe [AutoCAD 94]), könnte **TOPAS** einerseits im kommerziellen Bereich eingesetzt werden, es könnten andererseits auch mit herkömmlichen CAD-Systemen erstellte Modelle mit der Werkbank bearbeitet werden.

A Beispiele für das Arbeiten mit TOPAS

Die Abbildungen 56, 57, 58 und 59 sollen das Arbeiten mit dem Skripteditor und das schrittweise Abarbeiten des Skriptes zum Erstellen der Illustration 53 (a) (siehe Seite 106) verdeutlichen.
Die Abbildungen 60, 61 und 62 zeigen, wie die Illustration 53 (b) (siehe Seite 106) beispielsweise interaktiv, mit Hilfe der Menüsteuerung erstellt werden kann.
Anhand von Abbildung 54 (b) (siehe Seite 106) soll gezeigt werden, wie Techniken der direkten Manipulation dazu verwandt werden können, um eine Illustration zu verbessern. Bedingt durch die Darstellung des Rasenmähers als Drahtrahmenmodell ist die Unterlagscheibe in dieser Abbildung mit dem Rad als Hintergrund schlecht zu erkennen. Sie kann im interaktiven Grafikfenster durch die Maus selektiert und auf der Explosionsachse verschoben werden, wobei Mausbewegungen als Bewegungen auf dieser Achse interpretiert werden. In Abbildung 63 wird die Selektierung der Unterlagscheibe gezeigt. Abbildung 64 zeigt den Rasenmäher nach der Verschiebung.

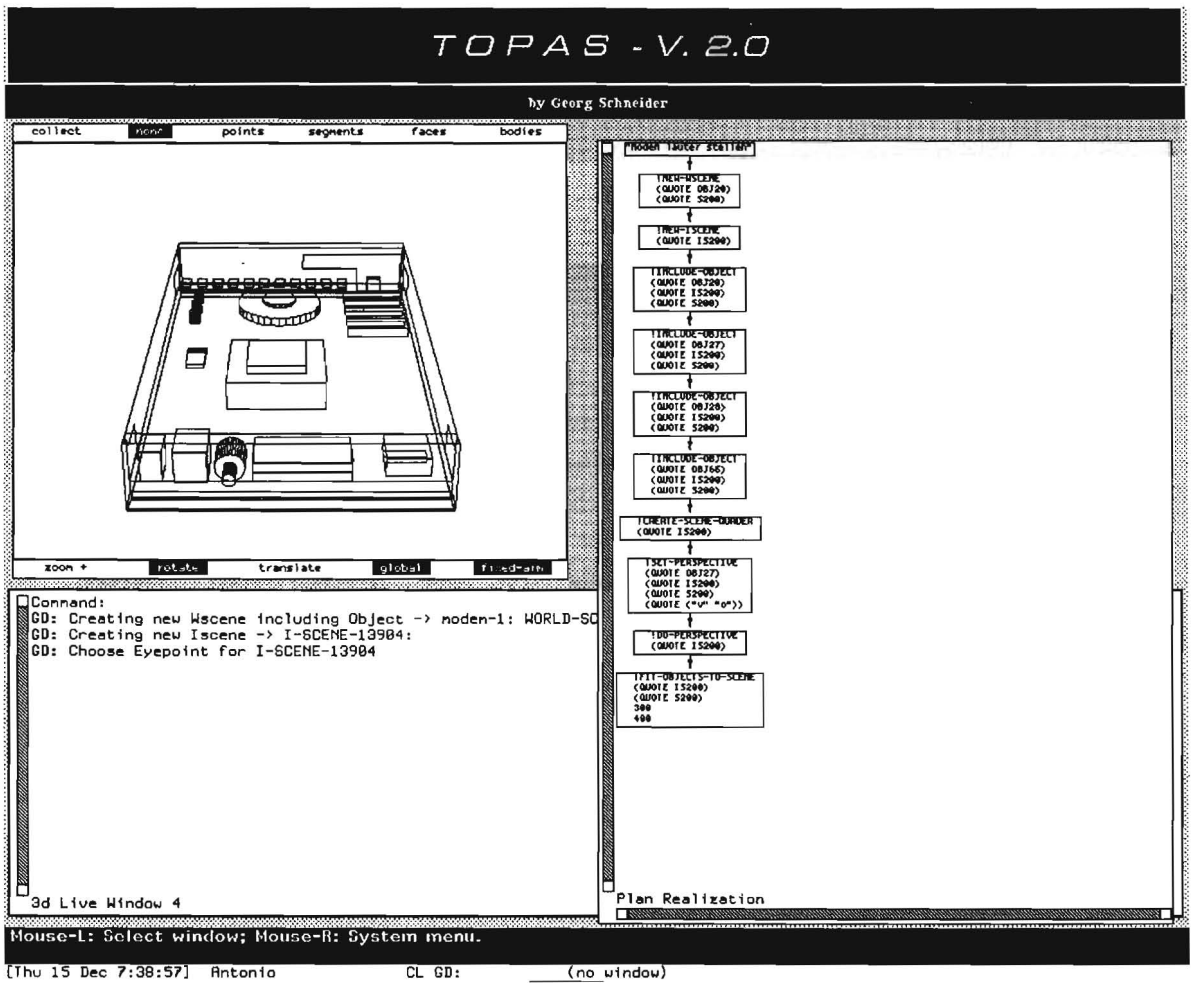

Abbildung 56: Die Hinterseite des Modems mit dem Lautstärkeregler ist zu sehen, das Modem wurde bereits in das Grafikfenster eingepaßt.

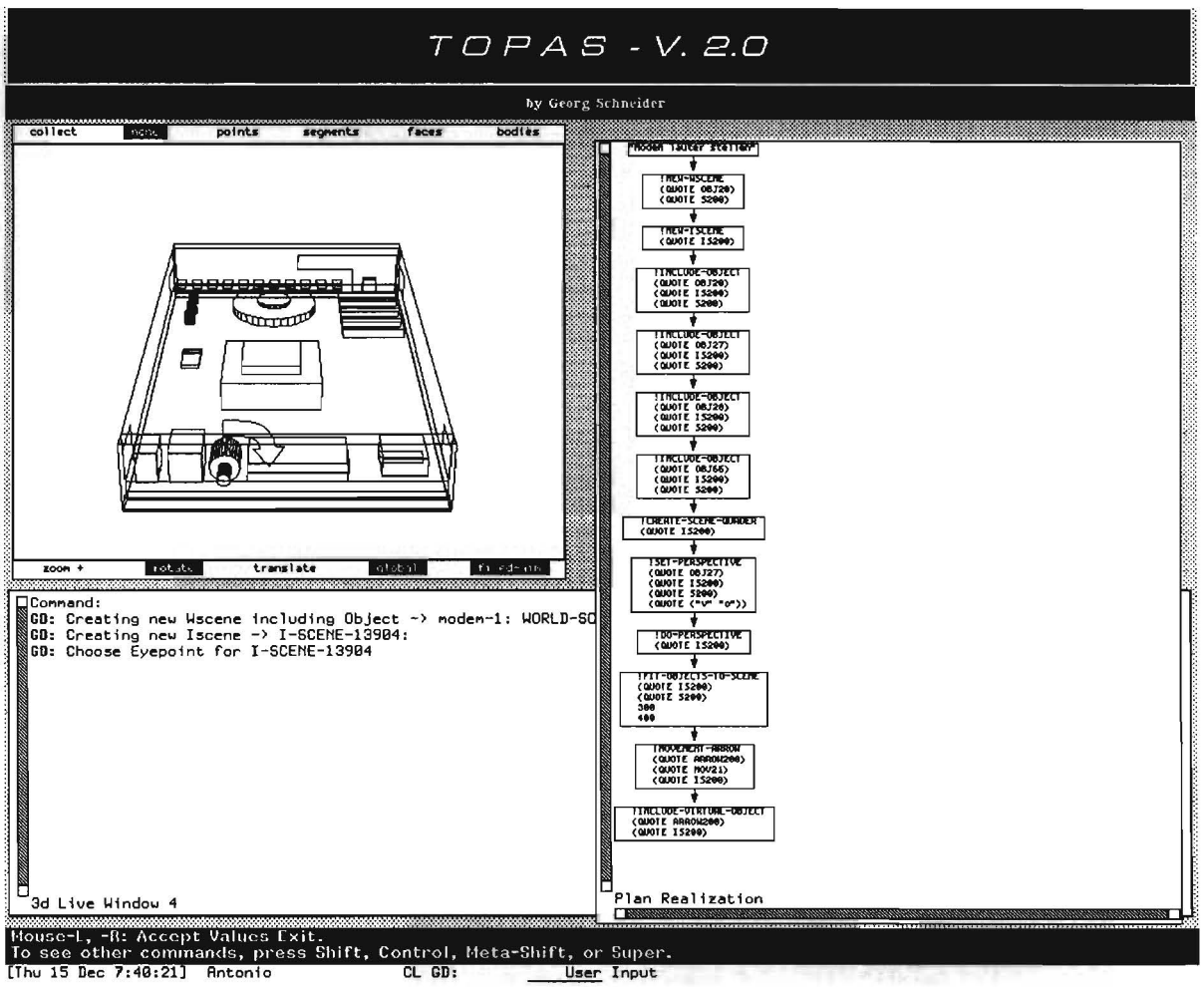

Abbildung 57: Der Trajektorienpfeil, der die Bewegung "Lautstärkeregler nach rechts" symbolisiert wurde erstellt und zur Illustration hinzugefügt.

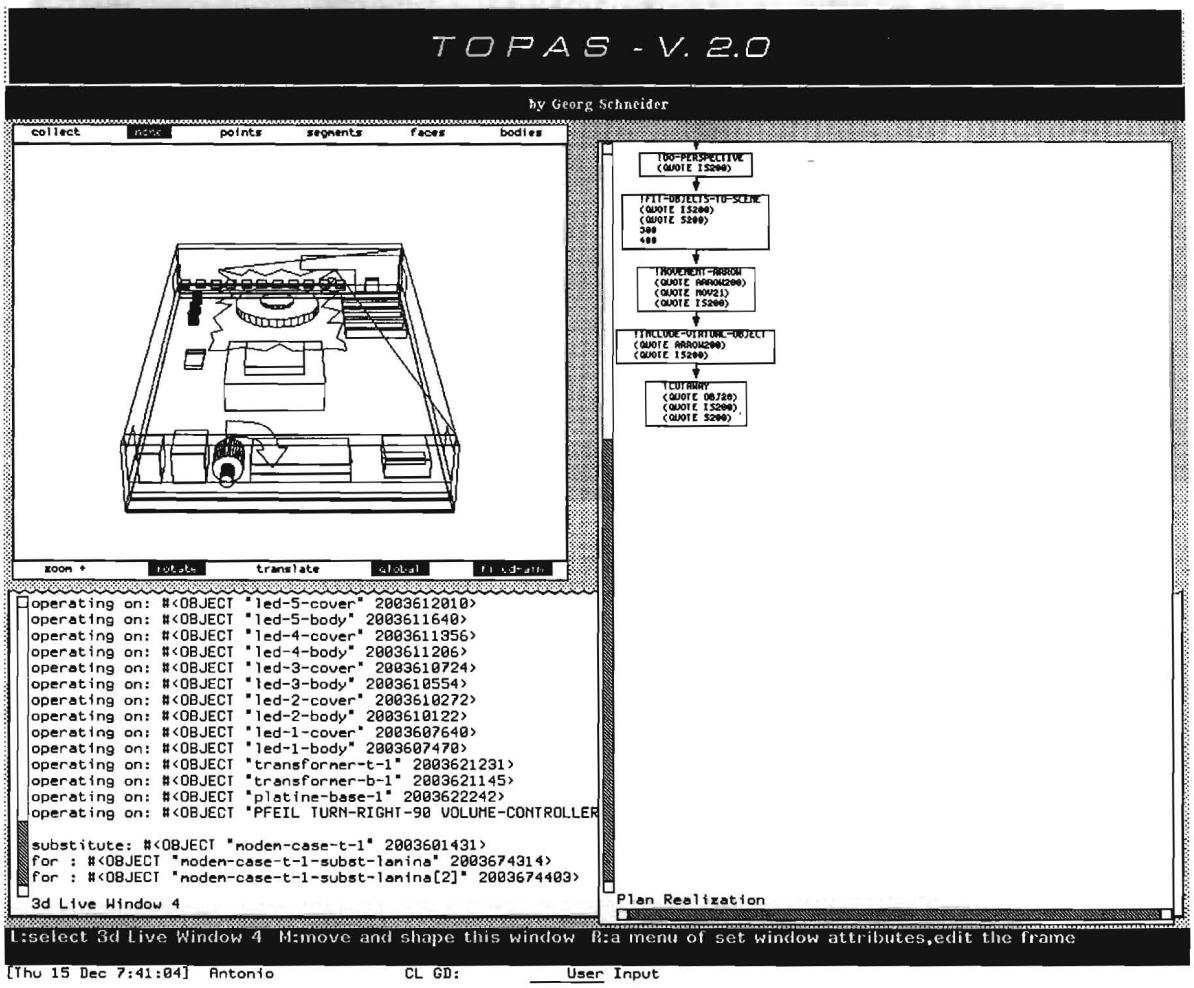

Abbildung 58: Der Aufriß auf den Lautstärkeregler wurde erzeugt.

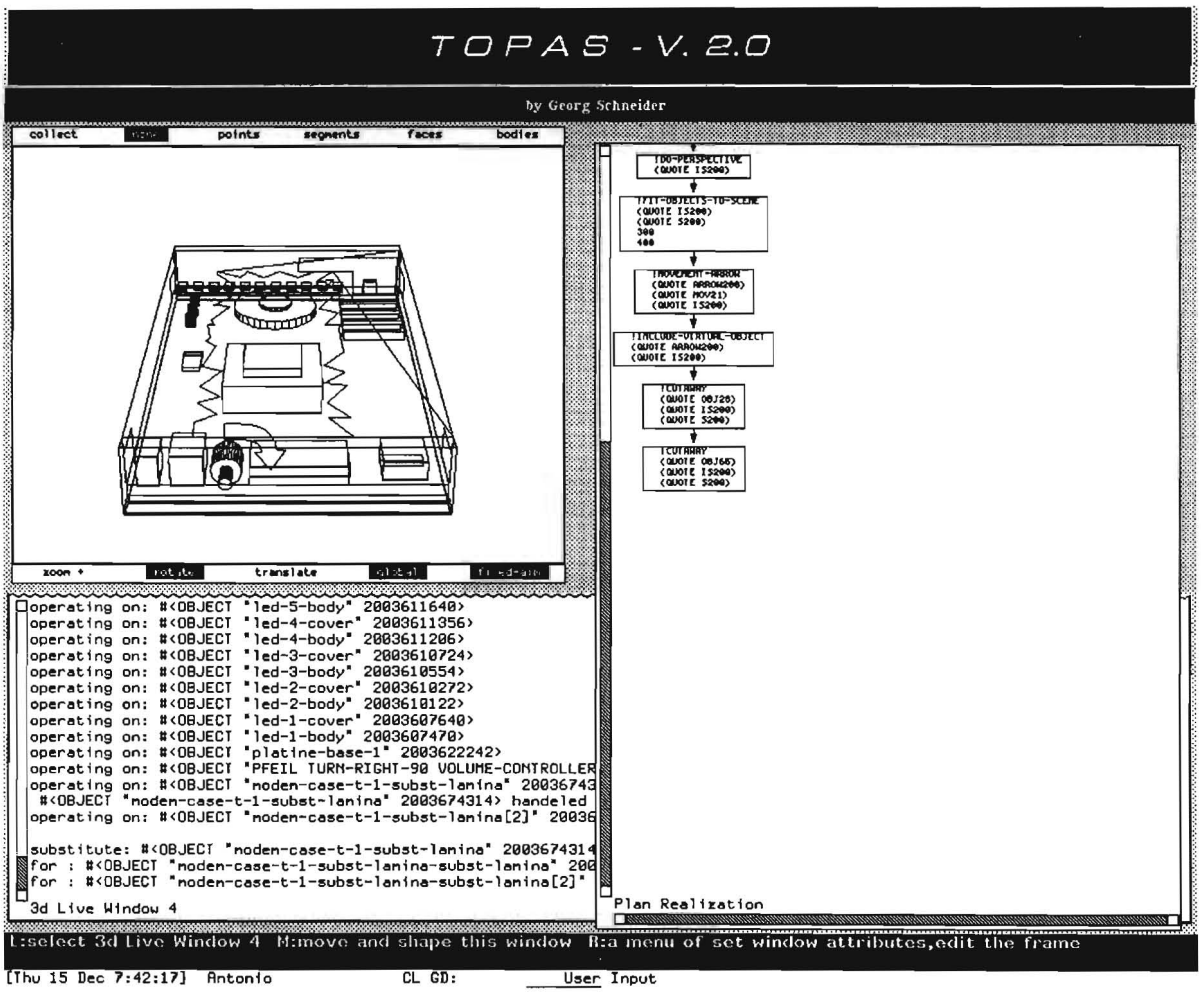

Abbildung 59: In einer weitergehenden Aufrißdarstellung ist nun auch der Transformator zu sehen.

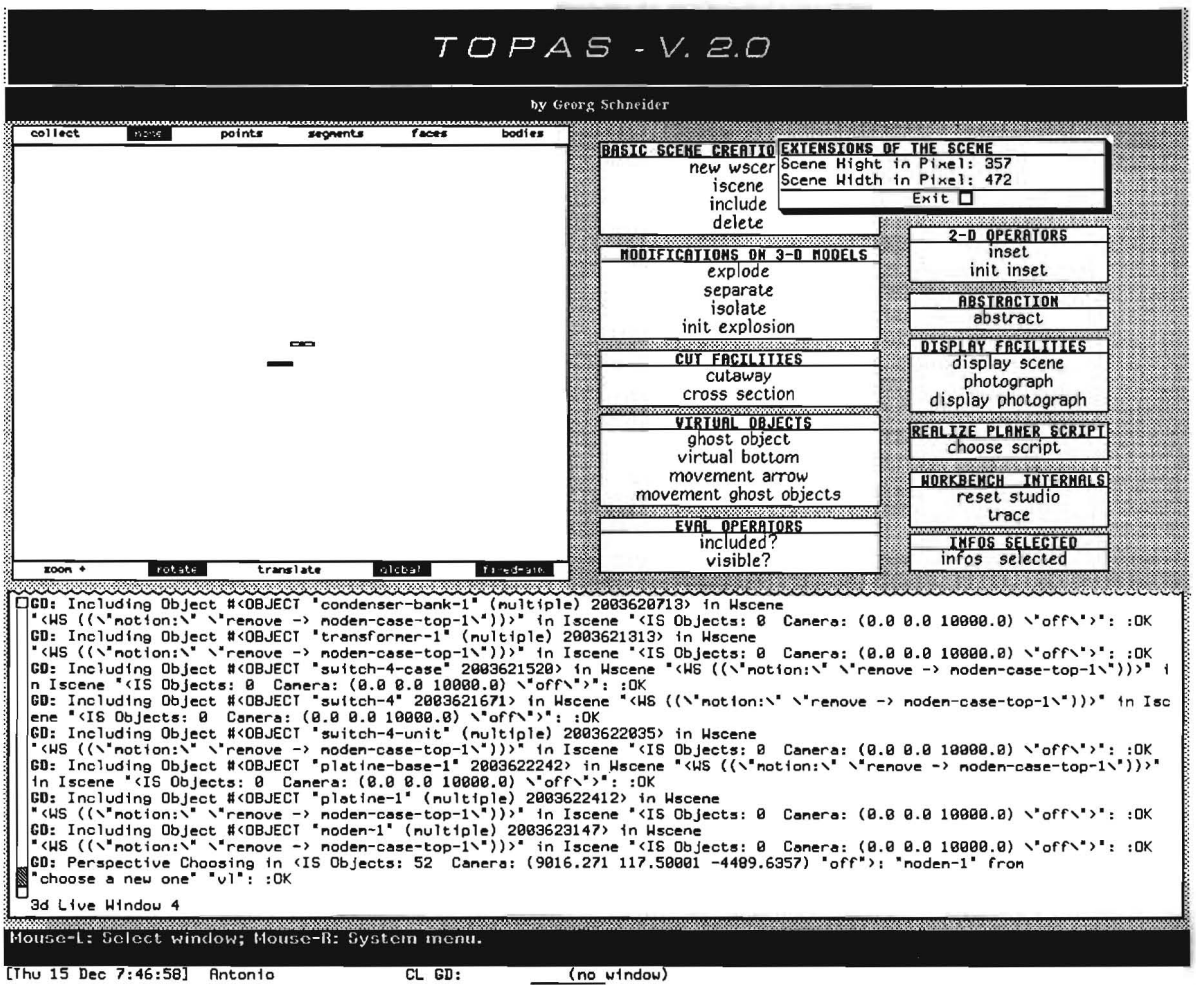

Abbildung 60: Eine Szene mit dem Modem im Zustand "Deckel geöffnet" wurde erzeugt und soll auf eine bestimmte Abbildungsgröße gebracht werden.

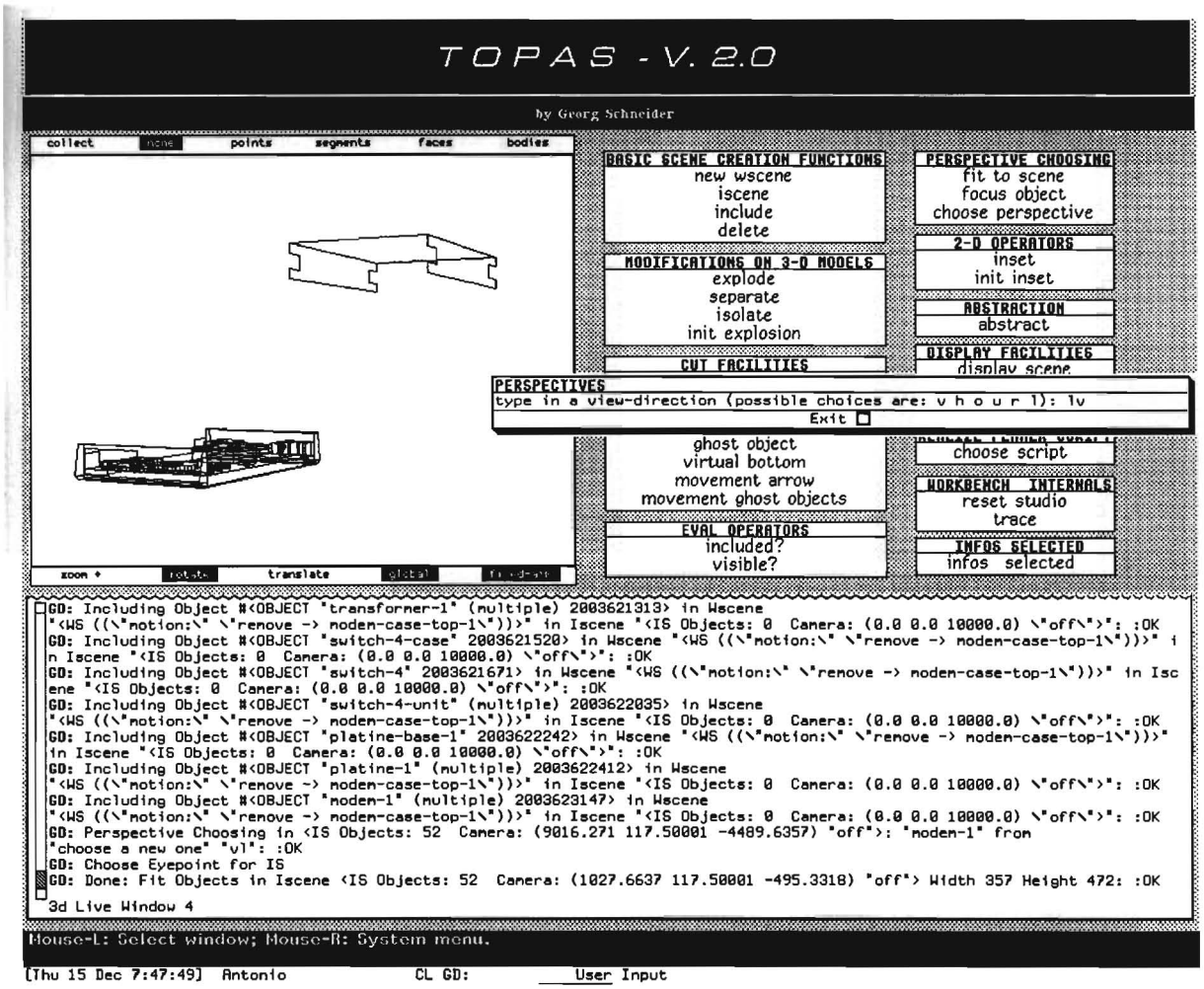

Abbildung 61: Die Perpektive "links" "vorne" soll eingestellt werden.

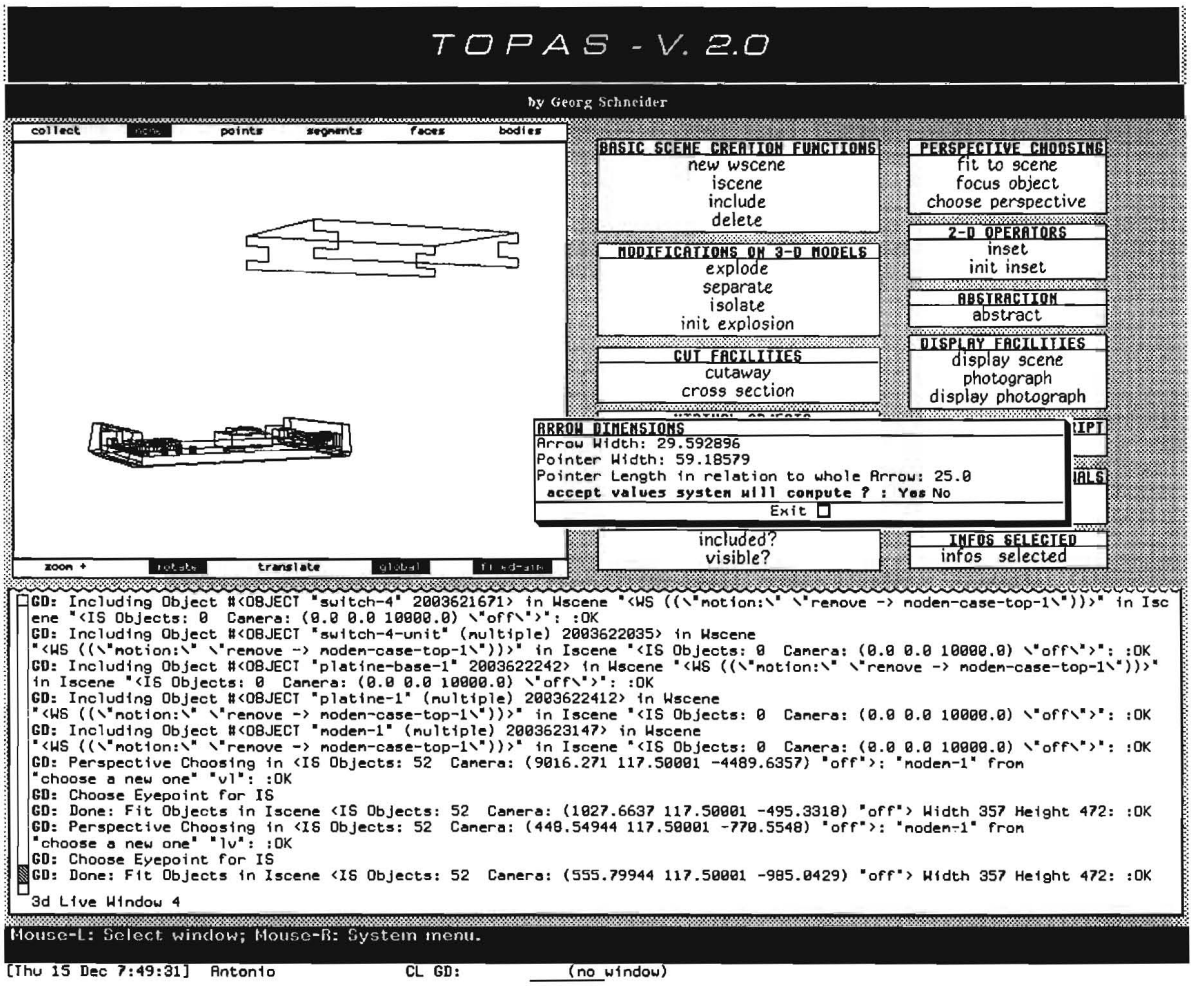

Abbildung 62: Ein Trajektorienpfeil, der die Öffnenbewegung symbolisiert, soll hinzugefügt werden.

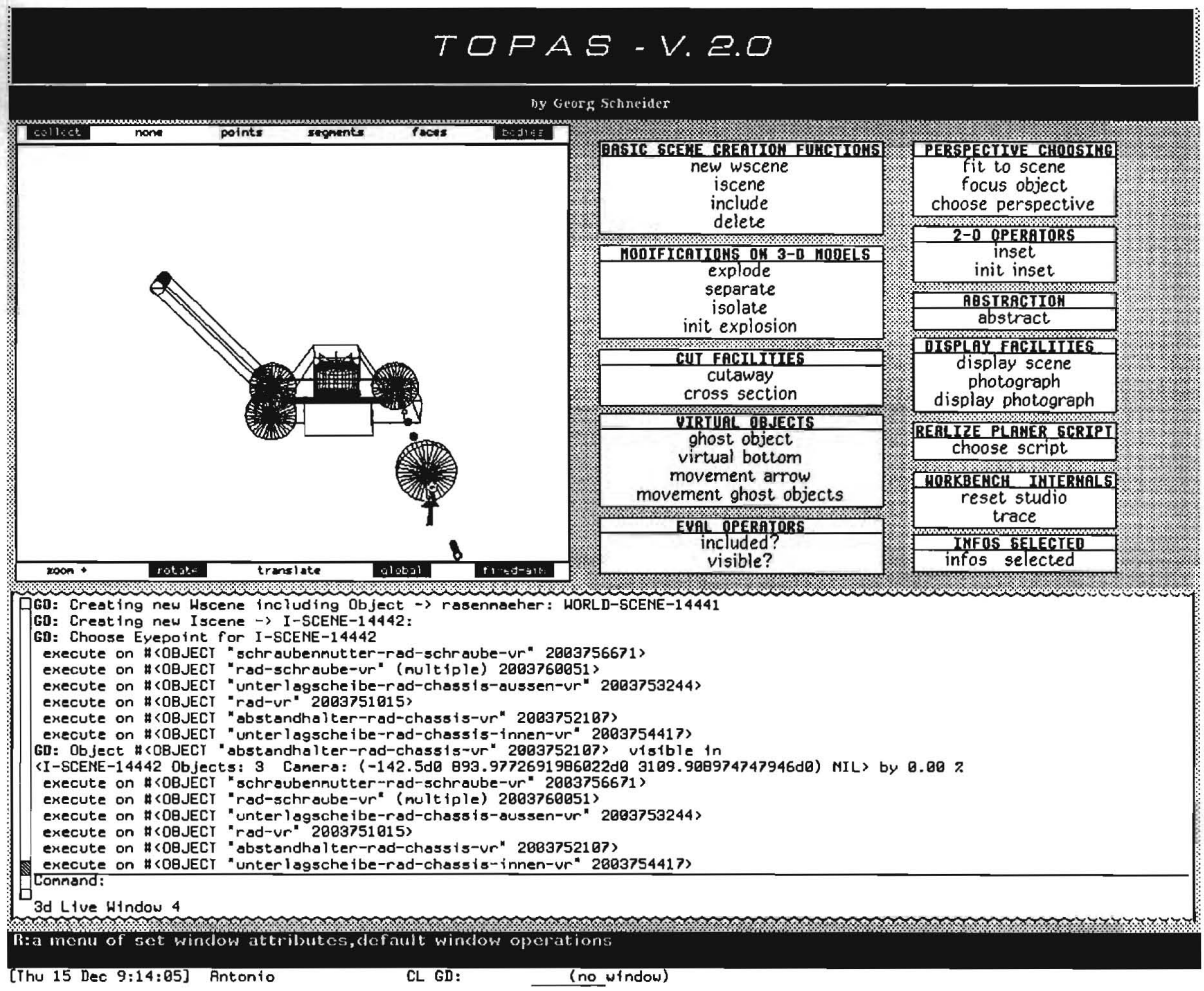

Abbildung 63: Die Unterlagscheibe wird mit der Maus selektiert.

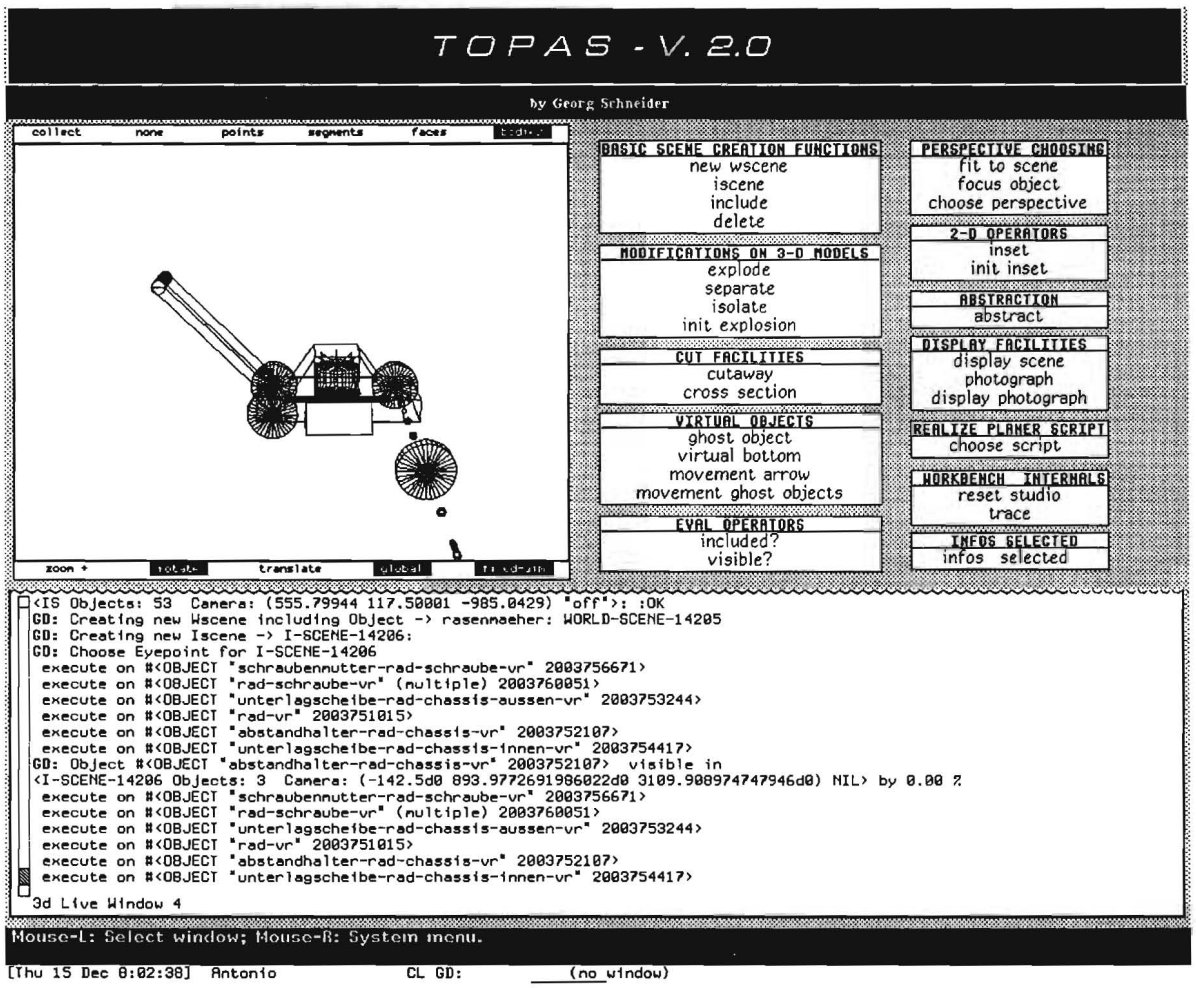

Abbildung 64: Die Explosionsdarstellung nach dem Verschieben der Unterlagscheibe.

B Die TOPAS-Operatoren

In diesem Teil des Anhangs werden alle **TOPAS**-Operatoren mit Parameterbeschreibung vorgestellt. Ein Benutzer kann mit ihrer Hilfe Skripte schreiben oder die Werkbank programmgesteuert betreiben.

B.1 Basisoperatoren zum Erstellen von Szenen

- \!new-wscene <object-symbol> <wscene-symbol> &optional <state-description>

  Legt eine neuen Weltzustand (WZ) an.

  Prüft, ob für <object-symbol> ein Modell vorhanden ist, das heißt, es gibt bereits einen Ur-Weltzustand (WZ-0) der ein zu <object-symbol> gehöriges Objekt enthält. Falls ja, wird der neue WZ mit Hilfe der <state-description> vom Zustand WZ-0 abgeleitet. Falls nein, wird das Studio mit neuem WZ-0 initialisiert, der das zu <object-symbol> gehörige Objekt enthält. Von diesem wird dann ein WZ abgeleitet, der <state-description> erfüllt.
  Schließlich wird der WZ an <wscene-symbol> gebunden.

- \!wscene <object-symbol> <wscene-symbol>

  Sucht nach geeignetem WZ.

  Prüft, ob für zu <object-symbol> gehöriges Objekt ein WZ vorhanden ist der <state-description> genügt. Falls ja, wird dieser ausgewählt und an <wscene-symbol> gebunden. Falls nein, wird :FAIL geliefert.

- \!new-iscene <iscene-symbol>

  Legt eine neue ilustratorische Szene (IS) an.

  Eine neue Datenstruktur vom Typ "Iscene" wird angelegt und an <iscene-symbol> gebunden.

- \!iscene <iscene-symbol> <object-symbol> <wscene-symbol>

  Suche nach geeigneter IS.

  Prüft, ob IS <iscene-symbol> vorhanden ist, die <object-symbol> im

Zustand <wscene-symbol> bereits enthält. Falls ja, wird diese ausgewählt und an <iscene-symbol> gebunden. Falls nein, wird :FAIL geliefert.

- \!include-object <object-symbol> <iscene-symbol> <wscene-symbol> &key :ghost <ghost-symbol>

  Nimmt <object-symbol>, das sich in einem durch <wscene-symbol> charakterisierten Zustand befindet, in <iscene-symbol> auf.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> im Zustand <wscene-symbol> vorhanden sind. Wenn ja, wird Objekt <object-symbol> im Zustand <wscene-symbol> in die IS <iscene-symbol> aufgenommen, sonst Rückgabewert :FAIL.
  Ghost Images müssen mit dem Schlüsselwort ":ghost" übergeben werden, da es sich bei ihnen um spezielle Objekte handelt.
  Bemerkungen:

  - ein und dasselbe Objekt kann nicht mehrmals in eine IS aufgenommen werden. Wird dies gewünscht, muß eine Kopie (d.h. ein Ghost Image) angelegt werden.
  - Alle Objekte aus IS sind aus gleichem WZ (Ausnahme: Ghost Images)
  - Ghost Images werden mit dem Schlüsselwort ":ghost" aufgenommen.

- \!delete-object <object-symbol> <iscene-symbol> <wscene-symbol> &key :ghost <ghost-symbol> :virtual <virtual-object-symbol>

  Löscht das Objekt, das sich in einem durch <wscene-symbol> charakterisierten Zustand befindet, aus IS <iscene-symbol>.

  Prüft, ob Objekt <object-symbol> im Zustand <wscene-symbol> in IS <iscene-symbol> vorhanden ist. Wenn ja, wird dieses gelöscht. Wenn nein, Rückgabewert :FAIL.
  Sollen Ghost Images gelöscht werden, so müssen sie mit dem Schlüsselwort ":ghost" übergeben werden, virtuelle Objekte, wie zum Beispiel Böden, werden mit dem Schlüsselwort ":virtual" übergeben.

- \!include-virtual-object <virtual-object-symbol> <iscene-symbol>

  In die Iscene wird ein virtuelles Objekt eingefügt.

  Prüft, ob das virtuelle Objekt <virtual-object-symbol> und IS <iscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Bemerkung: Virtuelle Objekte sind

  - Böden
  - Trajektorienpfeile
  - Ghost Images, die durch "movement-ghosts" entstanden sind

B.2 Techniken zur Manipulation von 3D-Modellen

- \!cutaway <object-symbol> <iscene-symbol> <wscene-symbol>

  Schneidet Löcher in Objekte, die die Sicht zu einem spezifizierten Objekt <object-symbol> verstellen.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> im Zustand <wscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Testet, welche Szeneobjekte den Blick auf das Objekt <object-symbol> versperren und ersetzt diese durch Objekte mit Loch.

- \!init-cutaway <object-symbol> <iscene-symbol> <wscene-symbol>

  Initialisiert \!cutaway.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> im Zustand <wscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Die Objekte mit Loch werden wieder gegen die ursprünglichen Objekte getauscht.
  Bemerkungen:

  - Die erstellten Aufrißdarstellungen müssen immer in der umgekehrten Reihenfolge rückgängig gemacht werden, wie sie aufgerufen wurden, d.h. der Jüngste zuerst, da bei späteren Aufrufen eventuell bereits ersetzte Objekte nochmals ersetzt werden müssen.

- \!cross-section <object-symbol> <iscene-symbol> <wscene-symbol> <side-specification>

  Generiert Querschnittdarstellungen einer Szene.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> im Zustand <wscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Die Querschnittebene liegt im Mittelpunkt des spezifizierten Objektes <object-symbol> und <side-specification> gibt an, welche Seite des Objektes <object-symbol>, ausgehend von der Ebene, entfernt werden soll und ersetzt die zu durchschneidenden Objekte.

- \!init-cross-section <iscene-symbol>

  Initialisiert \!cross-section.

  Prüft, ob IS <iscene-symbol> vorhanden ist, sonst Rückgabewert :FAIL.
  Tauscht die durchschnittenen Objekte wieder zurück.

- \!explode-objects <object-symbol-list> <iscene-symbol> <wscene-symbol>

  Explodiert Objekte aus <object-symbol-list>.

  Prüft, ob IS <iscene-symbol> und Objekte <object-symbol-list> im Zustand <wscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Erstellt eine Explosionszeichnung für die Objekte aus <object-symbol-list>, sofern eine Zusammenbauinformation für sie existiert. Die betroffenen Objekte werden verschoben.

- \!separate-object <object-symbol> <iscene-symbol> <wscene-symbol>

  Separiert ein Objekt <object-symbol-list>.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> im Zustand <wscene-symbol> vorhanden ist, sonst Rückgabewert :FAIL.
  Separiert das Objekt <object-symbol>, sofern eine Zusammenbauinformation für es existiert. Die betroffenen Objekte werden verschoben.

- \!isolate-object <object-symbol> <iscene-symbol> <wscene-symbol>

  Isoliert ein Objekt <object-symbol-list>.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> im Zustand <wscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL. Isoliert das Objekt <object-symbol>, sofern eine Zusammenbauinformation für es existiert. Die betroffenen Objekte werden verschoben.

- \!init-explosion <explosion-technique> <object-symbol-list> <iscene-symbol> <wscene-symbol>

  Initialisiert die Explosionstechniken \!explode-objects \!separate-object und \!isolate-object.

  Prüft, ob IS <iscene-symbol> und Objekt(e) <object-symbol-list> im Zustand <wscene-symbol> vorhanden sind und ob die Explosionstechnik <explosion-technique> mit ihnen durchgeführt wurde, sonst Rückgabewert :FAIL.
  Verschiebt die betroffenen Objekte wieder an die Position vor Aufruf der Explosionstechnik <explosion-technique>.

B.3 Operatoren zur Perspektivenwahl

- \!create-scene-quader <iscene-symbol>

  Legt einen Szenequader für die IS <iscene-symbol> an.

  Existiert diese nicht, Rückgabewert :FAIL.

- \!fit-objects-to-scene <iscene-symbol> <wscene-symbol> <width> <height>

  Objekte, die sich in einer durch <wscene-symbol> und <iscene-symbol> charakterisierten Szene befinden, werden in das Grafikfenster in der durch <width> und <height> vorgegebenen Pixelgröße eingepaßt.

  Prüft zuerst, ob Zustand <wscene-symbol> in der IS <iscene-symbol> enthalten ist, sonst Rückgabewert :FAIL.
  Danach werden die Szeneobjekte auf die entsprechende Abbildungsgröße gebracht.

- \!set-perspective <object-symbol> <iscene-symbol> <wscene-symbol> <perspective-specification(s)>

  Trägt die gewünschte Perspektive in die Perspektivenliste ein.

  Prüft, ob Objekt <object-symbol> im Zustand <wscene-symbol> in IS <iscene-symbol> vorhanden ist, bzw. ob <object-symbol> gleich <iscene-symbol> ist, sonst Rückgabewert :FAIL.
  Die Perspektive wird ausgewählt, indem Seite(n) des Objektes die sichtbar sein sollen, angegeben werden oder wenn <object-symbol> gleich <iscene-symbol>, indem Seite(n) der IS, die sichtbar sein sollen, angegeben werden.
  Bemerkung:

  - der Szenequader muß vorhanden sein.

- \!do-perspective <iscene-symbol>

  "do-perspective" wertet die Einträge der Perspektivenliste aus.

  Prüft, ob IS <iscene-symbol> vorhanden ist und stellt die Perspektive ein. Sind widersprüchliche Angaben in der Liste, beispielsweise "oben" und "unten" oder ist die IS <iscene-symbol> nicht vorhanden, so wird :FAIL zurückgeliefert.

- \!choose-perspective <iscene-symbol> <wscene-symbol> <perspective-specifications>

  Wählt eine Perspektive mit Hilfe des Regelinterpreters aus.

  Prüft, ob WZ <wscene-symbol> und IS <iscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Die Perspektive aus den in <perspective-specifications> angegebenen Zielen wird abgeleitet und mit einem *certainty factor* bewertet. Die ausgewählte Perspektive wird in die Perspektivenliste eingetragen.

- \!clear-perspective <iscene-symbol>

  Löscht die Einträge der Perspektivenliste.

  Prüft, ob IS <iscene-symbol> vorhanden ist, sonst Rückgabewert :FAIL.

- \!focus <object-symbol> <iscene-symbol>

  Rückt ein Szeneobjekt in den Bildmittelpunkt.

  Prüft, ob Objekt <object-symbol> und IS <iscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Behält die Betrachterentfernung zur Szene bei und verschiebt den Blickvektor so, daß sich das Objekt <object-symbol> im Bildmittelpunkt befindet.

B.4 Operatoren zum Erstellen von virtuellen- und metagrafischen Objekten

- \!add-virtual-bottom <bottom-symbol> <iscene-symbol> <side-specification>

  Legt einen virtuellen Boden für die IS <iscene-symbol> an.

  Prüft, ob IS <iscene-symbol> und <side-specification> vorhanden sind, sonst Rückgabewert :FAIL.
  Ein virtueller Boden an der Seite <side-specification> der IS <iscene-symbol> wird angelegt und an <bottom-symbol> gebunden.
  Bemerkung:

  - der Szenequader mit der entsprechenden Seitenreferenz muß vorhanden sein.

- \!new-ghost-object <ghost-symbol> <object-symbol> <wscene-symbol>

  Legt neues Ghost Image <ghost-symbol> vom Objekt <object-symbol> im WZ <wscene-symbol> an.

  Prüft, ob Objekt <object-symbol> und WZ <wscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Das Ghost Image wird als Kopie des Objektes <object-symbol> im WZ <wscene-symbol> erstellt und an <ghost-symbol> gebunden.

- \!movement-arrow <arrow-symbol> <movement-symbol> <iscene-symbol> &key :pfeilschaftdicke [integer] :pfeilspitzenbreite [integer]

:pfeilspitzenlaenge [%]

Legt neues Objekt vom Typ <pfeil-object> als Visualisierung der Bewegung <movement-symbol> an.

Prüft, ob Bewegung <movement-symbol> und IS <iscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
Eine Datenstruktur vom Typ <pfeil-object> wird angelegt, in der neben dem Pfeil als 3D-Objekt selbst auch Daten über die Abmessungen des Pfeils und der aktuellen Perspektive gespeichert sind. Der Pfeil wird an <arrow-symbol> gebunden. Die Schlüsselworte ":pfeilschaftdicke", ":pfeilspitzenbreite" und ":pfeilspitzenlaenge" ermöglichen dem Benutzer eigene Werte für die Pfeildimensionen zu liefern. Sind diese Parameter nicht spezifiziert, werden sie vom System errechnet.
Bemerkungen:

- Die Pfeilschaftdicke steht für die Breite des rechteckigen Teils des Pfeiles.
- Die Pfeilspitzenbreite bestimmt die Ausdehnung der breitesten Stelle der Pfeilspitze.
- Die Pfeilspitzenlänge gibt an, wieviel Prozent die Spitze verglichen mit dem gesamten Pfeil einnimmt.

- \!movement-ghosts <ghost-symbol> <movement-symbol> <iscene-symbol> &key :number-of-ghosts [integer]

  Legt neues <ghost-object> als Visualisierung der Bewegung <movement-symbol> an.

  Prüft, ob Bewegung <movement-symbol> und IS <iscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Die Ghost Images werden als Kopien des Objektes, für das die Bewegung definiert ist, an diskreten, vom System errechneten Zwischenzuständen erstellt. Mit dem Schlüsselwort ":number-of-ghosts" kann die Anzahl der Zwischenzustände, die visualisiert werden, vom Benutzer beeinflußt werden. Die Kopien werden zu einem Objekt zusammengefaßt und an <ghost-symbol> gebunden.

B.5 Operatoren zu 2D-Techniken

- \!draw-object-dashed <object-symbol> <iscene-symbol> &key :type [small, normal, big]

  Stellt ein Objekt gestrichelt dar.

  Prüft, ob Objekt <object-symbol> und IS <iscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Wird keine Strichlänge durch das Schlüsselwort ":type" spezifiziert, wird der Typ "normal" verwandt.

- \!init-dashing <object-symbol> <iscene-symbol>

  Initialisiert \!draw-object-dashed

  Prüft, ob das Objekt <object-symbol> und IS <iscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.

- \!inset <iscene-symbol> &key :inset <inset-symbol> :position [ol, or, ul, ur]

  Generiert ein Inset der IS <inset-symbol> an die angegebene Position der IS <iscene-symbol>.

  Prüft, ob die IS <iscene-symbol>, die IS <inset-symbol> und die Position vorhanden sind, sonst Rückgabewert :FAIL.
  Schaltet auf die IS <inset-symbol> um, erstellt ein Bitmap der von der Szene eingenommenen Region und legt einen Rahmen darum. Danach wird auf die IS <iscene-symbol> umgeschaltet und das Bitmap an die durch das Schlüsselwort ":position" spezifizierte Position kopiert.

- \!init-inset <iscene-symbol> &key :inset <inset-symbol> :position [ol, or, ul, ur]

  Entfernt ein Inset.

  Prüft, ob IS <iscene-symbol> und Inset <inset-symbol> auf der entsprechenden Position vorhanden sind. Wenn ja, wird das Inset gelöscht, wenn nein, Rückgabewert :FAIL.

- \!photograph <iscene-symbol>

  Erstellt ein Bitmap der IS <iscene-symbol>.

  Prüft, ob IS <iscene-symbol> vorhanden ist, sonst Rückgabewert :FAIL.
  Erstellt das Bitmap.

B.6 Evaluierungsoperatoren

- \?included <object-symbol> <iscene-symbol>

  Testet, ob das Objekt <object-symbol> in der IS <iscene-symbol> enthalten ist.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Liefert TRUE, wenn das Objekt <object-symbol> enthalten ist.

- \?visible <object-symbol> <iscene-symbol>

  Testet, zu wieviel Prozent das Objekt <object-symbol> in der IS <iscene-symbol> sichtbar ist.

  Prüft, ob IS <iscene-symbol> und Objekt <object-symbol> im Zustand <wscene-symbol> vorhanden sind, sonst Rückgabewert :FAIL.
  Liefert die prozentuale Sichtbarkeit des Objektes <object-symbol>.

C Beispielskripte

In diesem Teil des Anhangs sind die Beispielskripte für die in dieser Arbeit mit **TOPAS** erzeugten Illustrationen aufgelistet.
Skript zu: *Klappe öffnen*, siehe Abbildung 52 (a).

```
(\!new-wscene 'obj1 's100)
(\!new-iscene 'is100)
(\!include-object 'obj15 'is100 's100)
(\!include-object 'obj12 'is100 's100)
(\!include-object 'obj9 'is100 's100)
(\!include-object 'obj1 'is100 's100)
(\!create-scene-quader 'is100)
(\!fit-objects-to-scene 'is100 's100 200 300)
(\!set-perspective 'obj1 'is100 's100 ("v" "o" "r"))
(\!do-perspective 'is100)
(\!add-virtual-bottom 'bot100 'is100 "u")
(\!include-virtual-object 'bot100 'is100)
(\!new-wscene 'obj1 's101 (mov11))
(\!new-iscene 'is101)
(\!include-object 'obj15 'is101 's101)
(\!new-ghost-object 'ghost101 'obj15 's101)
(\!include-object 'obj15 'is100 's101 ':ghost 'ghost101)
(\!movement-arrow 'arrow100 'mov11 'is100)
(\!include-virtual-object 'arrow100 'is100)
(\!create-scene-quader 'is100)
(\!set-perspective 'obj1 'is100 's100 ("v" "o" "r"))
(\!do-perspective 'is100)
(\!fit-objects-to-scene 'is100 's100 300 400)
(\!focus 'obj9 'is100)
(\!include-object 'obj12 'is101 's101)
(\!create-scene-quader 'is101)
(\!set-perspective 'obj12 'is101 's101 ("v" "o" "r"))
(\!do-perspective 'is101)
(\!fit-objects-to-scene 'is101 's101 300 400)
(\!movement-ghosts 'ghost101 'mov12 'is101)
(\!include-virtual-object 'ghost101 'is101)
```

```
(\!draw-object-dashed 'ghost101 'is101 :type "small")
(\!create-scene-quader 'is101)
(\!fit-objects-to-scene 'is101 's101 200 200)
(\!inset 'is100 :inset 'is101 :position "ol")
```

Skript zu: *Deckel schließen und Kaffeemaschine anschalten*, siehe Abbildung 52 (b).

```
(\!new-wscene 'obj1 's110 (mov11))
(\!new-iscene 'is110)
(\!include-object 'obj12 'is110 's110)
(\!create-scene-quader 'is110)
(\!set-perspective 'obj1 'is110 's110 ("v" "o" "r"))
(\!do-perspective 'is110)
(\!fit-objects-to-scene 'is110 's110 200 200)
(\!movement-arrow 'arrow110 'mov12 'is110)
(\!include-virtual-object 'arrow110 'is110)
(\!new-wscene 'obj1 's111)
(\!new-iscene 'is111)
(\!include-object 'obj2 'is111 's111)
(\!include-object 'obj15 'is111 's111)
(\!include-object 'obj12 'is111 's111)
(\!create-scene-quader 'is111)
(\!set-perspective 'obj1 'is111 's111 ("v"))
(\!do-perspective 'is111)
(\!fit-objects-to-scene 'is111 's111 200 300)
(\!movement-arrow 'arrow111 'mov3 'is111)
(\!include-virtual-object 'arrow111 'is111)
(\!focus 'obj15 'is111)
(\!new-wscene 'obj1 's112 (mov3))
(\!new-iscene 'is112)
(\!include-object 'obj12 'is112 's112)
(\!create-scene-quader 'is112)
(\!set-perspective 'obj1 'is112 's112 ("v"))
(\!do-perspective 'is112)
(\!fit-objects-to-scene 'is112 's112 200 200)
(\!inset 'is111 :inset 'is110 :position "ol")
(\!inset 'is111 :inset 'is112 :position "or")
```

Skript zu: *Tasse unterstellen*, siehe Abbildung 52 (c).

```
(\!new-wscene 'obj1 's120 (mov1 mov3))
(\!new-iscene 'is120)
(\!include-object 'obj1 'is120 's120)
(\!include-object 'obj18 'is120 's120)
(\!create-scene-quader 'is120)
(\!set-perspective 'obj1 'is120 's120 ("v" "r"))
(\!do-perspective 'is120)
(\!fit-objects-to-scene 'is120 's120 200 200)
(\!movement-arrow 'arrow120 'mov2 'is120
  :pfeilschaftdicke 10
  :pfeilspitzenbreite 25
  :pfeilspitzenlaenge 25)
(\!include-virtual-object 'arrow120 'is120)
(\!new-wscene 'obj1 's121 (mov3))
(\!new-iscene 'is121)
(\!include-object 'obj1 'is121 's121)
(\!include-object 'obj2 'is121 's121)
(\!include-object 'obj18 'is121 's121)
(\!include-object 'obj19 'is121 's121)
(\!create-scene-quader 'is121)
(\!set-perspective 'obj1 'is121 's121 ("v" "r"))
(\!do-perspective 'is121)
(\!add-virtual-bottom 'bot121 'is121 "u")
(\!include-virtual-object 'bot121 'is121)
(\!fit-objects-to-scene 'is121 's121 200 200)
(\!focus 'obj2 'is121)
(\!inset 'is121 :inset 'is120 :position "ol")
```

Skript zu: *Tasse wegnehmen*, siehe Abbildung 52 (d).

```
(\!new-wscene 'obj1 's130 (mov3))
(\!new-iscene 'is130)
(\!include-object 'obj1 'is130 's130)
(\!include-object 'obj18 'is130 's130)
(\!create-scene-quader 'is130)
(\!set-perspective 'obj1 'is130 's130 ("v" "r"))
```

```
(\!do-perspective 'is130)
(\!fit-objects-to-scene 'is130 's130 290 390)
(\!movement-ghosts 'ghost130 'mov1 'is130 :NUMBER-OF-GHOSTS 4)
(\!include-virtual-object 'ghost130 'is130)
(\!draw-object-dashed 'ghost130 'is130 :type "small")
(\!add-virtual-bottom 'bot130 'is130 "u")
(\!include-virtual-object 'bot130 'is130)
```

Skript zu: *Modem lauter stellen*, siehe Abbildung 53 (a).

```
(\!new-wscene 'obj20 's200)
(\!new-iscene 'is200)
(\!include-object 'obj20 'is200 's200)
(\!include-object 'obj27 'is200 's200)
(\!include-object 'obj28 'is200 's200)
(\!include-object 'obj65 'is200 's200)
(\!create-scene-quader 'is200)
(\!set-perspective 'obj27 'is200 's200 ("v" "o"))
(\!do-perspective 'is200)
(\!fit-objects-to-scene 'is200 's200 300 400)
(\!movement-arrow 'arrow200 'mov21 'is200)
(\!include-virtual-object 'arrow200 'is200)
(\!cutaway 'obj28 'is200 's200)
(\!cutaway 'obj65 'is200 's200)
```

Skript zu: *Modem öffnen*, siehe Abbildung 53 (b).

```
(\!new-wscene 'obj20 's210 (mov22))
(\!new-iscene 'is210)
(\!include-object 'obj20 'is210 's210)
(\!include-object 'obj27 'is210 's210)
(\!include-object 'obj28 'is210 's210)
(\!include-object 'obj21 'is210 's210)
(\!create-scene-quader 'is210)
(\!set-perspective 'obj27 'is210 's210 ("l" "v"))
(\!do-perspective 'is210)
(\!fit-objects-to-scene 'is210 's210 300 400)
(\!movement-arrow 'arrow210 'mov22 'is210)
(\!include-virtual-object 'arrow210 'is210)
```

Skript zu: *Rasenmäher: Fahrwerk ausbauen*, siehe Abbildung 54 (b).

```
(\!new-wscene 'obj80 's300)
(\!new-iscene 'is300)
(\!include-object 'obj97 'is300 's300)
(\!include-object 'obj116 'is300 's300)
(\!include-object 'obj80 'is300 's300)
(\!create-scene-quader 'is300)
(\!set-perspective 'obj80 'is300 's300 ("r" "o" "r"))
(\!do-perspective 'is300)
(\!fit-objects-to-scene 'is300 's300 200 300)
(\!explode-objects (obj116) 'is300 's300)
(\?visible 'obj97 'is300)
(\!explode-objects (obj116) 'is300 's300)
```

Skript zu: *Position Motor und Ventilator*, siehe Abbildung 54 (a).

```
(\!new-wscene 'obj80 's310)
(\!new-iscene 'is310)
(\!include-object 'obj81 'is310 's310)
(\!include-object 'obj84 'is310 's310)
(\!include-object 'obj90 'is310 's310)
(\!include-object 'obj80 'is310 's310)
(\!create-scene-quader 'is310)
(\!set-perspective 'obj80 'is310 's310 ("v" "o" "r"))
(\!do-perspective 'is310)
(\!fit-objects-to-scene 'is310 's310 300 400)
(\!cutaway 'obj84 'is310 's310)
(\!cutaway 'obj81 'is310 's310)
(\!init-cutaway 'obj81 'is310 's310)
(\!init-cutaway 'obj84 'is310 's310)
(\!explode-objects (obj90) 'is310 's310)
```

Abbildungsverzeichnis

1 Die Abbildung zeigt die Bedienung des Selbstauslösers einer Fotokamera. Aus: [Canon 81]. 4
2 Vergleich der Editoren 13
3 "Steamer's Graphics Editor". Aus: [Weitzman 86] 15
4 Vom System abgeleitete Alternativen. Aus: [Weitzman 86] . . 16
5 Eine Abbildung vor und nach einer Verschönerung, sinngemäß nach [Bolz 93]. 17
6 Ein Makro des Systems "Chimera". Aus: [Kurlander, Feiner 92] 18
7 Ein durch "APT" generiertes Balkendiagramm. Aus: [Foley et al. 90] 20
8 Die vom System "BOZ" generierte Grafik zeigt Flüge und deren Ankunfts-, bzw. Abflugzeiten. Sinngemäß nach [Casner 91]. 22
9 Ein vom System "ANDD" automatisch generiertes Netzwerk. Aus: [Marks 91] 23
10 Grundelemente aus der Datenbank. Aus: [Strothotte, Helms 95] 24
11 Aus den Grundelementen der Datenbank erzeugte Illustration. Aus: [Strothotte, Helms 95] 24
12 Verschiedene Schiffe, abgeleitet von demselben Prototyp. Aus: [Friedell 84] 26
13 Der *generate-and-test* Ansatz bei "IBIS". Aus: [Feiner, Seligmann 91 (a)] 27
14 Eine von "IBIS" erzeugte Anleitung für ein Funkgerät. Aus: [Foley et al. 90] 28
15 Die Abbildung zeigt eine Szene, bei der Hintergrundinformation durch ein Inset dargeboten wird. Aus: [Seligmann, Feiner 93] 29
16 Zusammenbauhierarchie zum Generieren von explodierten Darstellungen. 37
17 Kombination zwischen IS und WZ. 42
18 Aufrißdarstellung eines Motors. Aus: [Bosch 85 a]. 43
19 Die ursprüngliche Fläche wird durch zwei Flächen ersetzt, die zusammen die gewünschte "Lochfläche" ergeben. 46
20 Im Fall 2 kann die Fläche in eine oder mehrere Teilflächen zerfallen. 47
21 Im Fall 3 wird die Fläche ausgeblendet. 47

22 Durch den Aufriß wird der Transformator sichtbar. 49
23 In dieser Illustration wurde das Rad vom restlichen Fahrwerk abgetrennt. 51
24 Dieses Beipiel zeigt die Isolation der Unterlagscheibe. 52
25 Explosion des gesamten Fahrwerks. 53
26 Explosionsdarstellung eines Fruchtpressenvorsatzes. Aus: [Braun]. 54
27 Explosionssequenz 58
28 Ein Beipiel einer Zusammenbauhierarchie. 60
29 Der daraus generierte Explosionsbaum. 60
30 Nach der ersten Sequentialisierung: Eine Wurzel wurde gesplittet, ein Pfad wurde bereits erstellt. 61
31 Nach dem zweiten Konvertierungsschritt: Die Konvertierung ist beendet. Der Baum wurde komplett in Pfade zerlegt. . . . 61
32 Eine Explosionsdarstellung des Rasenmähers. 64
33 Querschnitt durch eine Pumpe. Aus: [Bosch 85 b]. 64
34 Der Rasenmäher im Querschnitt. 67
35 Trajektorienpfeil zum Zeigen einer Bewegung. Aus: [HIT] . . . 67
36 Kaffeemaschine mit Trajektorienpfeil. 70
37 Zwei Ghost Images 71
38 Kaffeemaschine mit Ghost Images zum Zeigen der Öffnenbewegung. 74
39 Der Boden als zur Szene gehöriges Defaultobjekt. Aus: [Philips c]. 74
40 Der virtuelle Boden zeigt die Lage der Maschine im Raum. . . 77
41 Einpassen eines Objektes 79
42 Fokussieren durch Zentrieren 80
43 Mögliche Betrachterpositionen um die invers dargestellte linke Quaderseite. 84
44 Durch den Regelinterpreter abgeleitete Perspektive nach Abarbeiten des Zieles: Zeige *knopf-1* von Seite *vorne*. 87
45 Durch den Regelinterpreter abgeleitete Perspektive nach Abarbeiten des Zieles: Bevorzuge Froschperspektiven. 88
46 Durch den Regelinterpreter abgeleitete Perspektive nach Abarbeiten des Zieles: 3D-Präsentation. 88
47 das Inset dient zur Ausschnittvergrößerung. Aus: [Lufthansa 93]. 89

48 Kaffeemaschine mit einem Inset. 91
49 Das Stricheln der Ghost Images verdeutlicht ihre Funktion als virtuelle Objekte. 93
50 Die Architektur von **TOPAS**. 97
51 Die Oberfläche von **TOPAS**. 100
52 Sequenz: Kaffeemaschine bedienen 104
53 Sequenz: Modem bedienen 106
54 Sequenz: Rasenmäher warten 107
55 Die Architektur von WIP 110
56 Die Hinterseite des Modems mit dem Lautstärkeregler ist zu sehen, das Modem wurde bereits in das Grafikfenster eingepaßt. 120
57 Der Trajektorienpfeil, der die Bewegung "Lautstärkeregler nach rechts" symbolisiert wurde erstellt und zur Illustration hinzugefügt. 121
58 Der Aufriß auf den Lautstärkeregler wurde erzeugt. 122
59 In einer weitergehenden Aufrißdarstellung ist nun auch der Transformator zu sehen. 123
60 Eine Szene mit dem Modem im Zustand "Deckel geöffnet" wurde erzeugt und soll auf eine bestimmte Abbildungsgröße gebracht werden. 124
61 Die Perpektive "links" "vorne" soll eingestellt werden. 125
62 Ein Trajektorienpfeil, der die Öffnenbewegung symbolisiert, soll hinzugefügt werden. 126
63 Die Unterlagscheibe wird mit der Maus selektiert. 127
64 Die Explosionsdarstellung nach dem Verschieben der Unterlagscheibe. 128

Literatur

[Adobe Photoshop 90] Adobe Photoshop, Basic User Guide; Advanced User Guide, London: Adobe, 1990

[André, Rist 93] André, E., Rist, T., The Design of Illustrated Documents as a Planning Task, in: Maybury, M. (Hrsg.), Intelligent Multimedia Interfaces, AAAI/MIT Press, S. 94-116, 1993

[AutoCAD 94] AutoCAD Release 12, München: Autodesk GmbH, 1994

[AutoSketch 94] AutoSketch für Windows, Release 2.0, Produktübersicht, Das universelle Zeichenprogramm mit der Präzision von CAD, München: Autodesk GmbH, 1994

[Bolz 93] Bolz, D., Some Aspects of the User Interface of a Knowledge Based Beautifier for Drawings, in: Proc. of the International Workshop on Intelligent User Interfaces in Orlando, New York: ACM, S. 45-52, 1993

[Bosch 85 a] Bosch, Pompe d'injection distributrice, Typ VE, Cahier technique, Stuttgart: Robert Bosch GmbH, 1985

[Bosch 85 b] Bosch, Equipements d'injection Diesel, Régulateurs pour pompes d'injection en ligne, Cahier technique, Stuttgart: Robert Bosch GmbH, 1985

[Braun] Braun, Alleskönner, Produktbeilage einer Küchenmaschine

[Butz 94] Butz, A., BETTY: Ein System zur Planung und Generierung informativer Animationssequenzen, Diplomarbeit, Universität des Saarlandes, Saarbrücken, Fachbereich für Informatik, Prof. Dr. W. Wahlster, 1994

[Canon 81] Canon, AE1, Bedienungsanleitung - deutsche Ausgabe, Amsterdam: Canon, 1981

[Casner 91] Casner, S. M., A Task-Analytic Approach to the Automated Design of Graphic Presentations, in: ACM Transactions on Graphics, Vol. 10, No. 2, S. 111-151, 1991

[Ebeling 94] Ebeling, A., Das Nesthäkchen, Low-Cost-CAD mit AutoSketch für Windows, Release 2, in: c't September 1994, Hannover: Heinz Heise, S. 52, 1994

[Ehrmann 94] Ehrmann, S., Powerful Software, Programme für den Native Mode der Power Macs, in: c't Juli 1994, Hannover: Heinz Heise, S. 144-146, 1994

[Emhardt, Strothotte 92] Emhardt, J., Strothotte, T., Hyper-Rendering, in: Proc. Graphics Interface '92, Vancouver, S. 37-43, 1992

[Espe 86] Espe, H., Zum Einfluß von Kamerawinkeln auf die Beurteilung von Portraitfotografien - eine Erkundungsstudie, in: Espe, H. (Hrsg.), Semiotische Studien zur Kommunikation, Band 2, Visuelle Kommunikation: Empirische Analysen, Hildesheim: G. Olms, S. 166-182, 1986

[Feiner, Seligmann 91 (a)] Feiner, S. K., Seligmann, D. D., Automated Generation of Intent-Based 3D-Illustrationen, in: Computer Graphics (ACM) 25(4), S. 123-132, 1991

[Feiner, Seligmann 91 (b)] Feiner, S. K., Seligman, D. D., Dynamic 3D Illustrations with Visibility Constraints, in: Proc. Computer Graphics International '91, Cambridge, 1991

[Fellner 88] Fellner, W. D., Computer Grafik, Reihe Informatik, Band 58, Mannheim: BI-Wissenschaftsverlag, 1988

[Foley et al. 90] Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F., Computer Graphics: Principles and Practice, (2nd Edition), New York: Addison-Wesley, 1990

[Friedell 84] Friedell, M., Automatic Synthesis of Graphical Object Descriptions, in: Computer Graphics (ACM) 18(3), S. 53-62, 1984

[Geschke et al. 94] Geschke, H. W., Heller, W., Wehr, W., Technisches Zeichnen, 22. neubearbeitete und erweiterte Auflage, Stuttgart: Teubner, 1994

[Graf 93] Graf, W., LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations, in: Salvendy, G., Smith, M. J. (Hrsg.), Human-Computer Interaction: Software and Hardware Interfaces (19 B), New York: Elsevier, S. 446-451, 1993

[Heinsohn et al. 93] Heinsohn, J., Kudenko, D., Nebel, B., Profitlich, H. J., RAT: Representation of Actions using Terminological Logics, in: Proc. of the DFKI Workshop on Taxonomic Reasoning, DFKI Document D-92-08, Saarbrücken: DFKI, S. 16-22, 1992

[HIT] HIT, Kaffee-Automat KA 1100, Produktbeilage, Aschau-Werk

[Hoischen 88] Hoischen, H., Technisches Zeichnen, Grundlagen, Normen, Beispiele, Darstellende Geometrie, Düsseldorf: Cornelsen Girardet, 1988

[Hypercard] HyperCard Benutzerhandbuch München: Apple Computer GmbH

[IsoDraw 94] IsoDraw Produktfibel, Die Lösung für die technische Illustration, ITEDO IsoDraw, Inside Technical Illustration, Siegburg: ITEDO Software GmbH, 1994

[Kent et al. 92] Kent, J. R., Carlson, W. E., Parent, R. E., Shape Transformation for Polyhedral Objects, in: Computer Graphics (ACM) 26(2), New York: ACM, S. 47-54, 1992

[Kilger, Finkler 95] Kilger, A., Finkler, W., TAG-Based Inkremental Generation, in: Computational Linguistics, MIT Press, to appear 1995

[Krüger 95] Krüger, A., PROXIMA: Ein System zur Generierung graphischer Abstraktionen, Diplomarbeit, Universität des Saarlandes, Saarbrücken, Fachbereich für Informatik, Prof. Dr. W. Wahlster, 1995

[Krüger et al. 91] Krüger, A., Schneider, G., Schneiderlöchner, F., Schommer, C., Wissensbasierte Graphikgenerierung. Abschlußbericht FOPRA WS 90/91, Universität des Saarlandes, 1991

[Kurlander, Feiner 92] Kurlander, D., Feiner, S. K., A History-Based Macro By Example System, in: Proc. ACM Symposium on User Interface

Software and Technology, Monterey, California, New York: ACM, S. 99-106, 1992

[Liska 94] Liska, J., ...zum fünften, Grafikpaket CorelDraw 5.0, in: c't September 1994, Hannover: Heinz Heise, S. 49, 1994

[Lufthansa 93] Lufthansa, Für Ihre Sicherheit, B 737, B737-200/300/500, TN 413-0200 10/93, 1993

[Mackinlay 86] Mackinlay, J., Automating the Design of graphical Presentations of Relational Information, in: Proc. ACM Transactions on Graphics, 5 (2), S. 110-141, 1986

[Marks 91] Marks, J., The Competence of an Automated Graphic Designer, in: Proc. Long Island Conference on Artificial Intelligence and Computer Graphics, NYIT, New York: S. 53-61, 1991

[Müller 95] Müller, J., GeoDisplay: Ein flexibles Visualisierungstool für LISP-Applikationen, Diplomarbeit, Universität des Saarlandes, Saarbrücken, Fachbereich für Informatik, Prof. Dr. W. Wahlster, 1995

[Philips a] Philips Plattenspieler AF877, Produktbeilage

[Philips b] Philips Radiorecorder N2643, Produktbeilage

[Philips c] Philips: Café Gourmet, Kaffeemaschine, Produktbeilage

[Rist et al. 94] Rist, T., Krüger, A., Schneider, G., Zimmermann, D., AWI: A Workbench for Semi-Automated Illustration Design, in: Catarci, T., Costabile, M. F., Levialdi, S., Santucci, G. (Hrsg.) Proc. of the Workshop on Advanced Visual Interfaces AVI '94, New York: ACM, S. 59-68, 1994

[Rist, André 94] André, E., Rist, T., Multimedia Presentations: The support of Passive and Active Viewing, in: Working Notes of the AAAI Spring Symposium on Intelligent Multi-Media Multi-Modal Systems Stanford University, S. 22-29, 1994

[Rist, André 92] André, E., Rist, T., Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP, in: Catarci,

T., Costabile, M. F., Levialdi, S. (Hrsg.), Advanced Visual Interfaces, London: World Scientific, 1992

[Rist, André 90] Rist, T., André, E., Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen, in: Kansy, K., Wißkirchen, P. (Hrsg.), Graphik und KI, IFB 239, Berlin: Springer, S. 48-57, 1990

[Rossignac et al. 92] Rossignac, J., Megahed, A., Schneider, B. O., Interactive Inspection of Solids: Cross-sections and Interferences, in: Computer Graphics (ACM) 26(2), S. 353-360, 1992

[S-Geometry 90] S-Geometry, User Reference Manual, Revision 6.0, Symbolics, 1990

[Schneider, Krüger 93] Schneider, G., Krüger, A., Eine Werkbank zur Erzeugung von 3D-Illustrationen, in: Iwainsky, A. (Hrsg.) 3. Workshop Computergrafik und automatisierte Layoutsynthese, Berlin: IIEF, S. 115-135, 1993

[Schneider 95 a] Schneider, G., Eine Werkbank zur Erzeugung von 3D-Illustrationen, in: Iwainsky, A. (Hrsg.) Ableitung technischer Grafiken aus rechnerinternen Modellen, Vieweg, erscheint voraussichtlich 1995

[Schneider 95 b] Schneider, G., Wissensbasierte Grafikgenerierung: Eine Werkbank zum Erzeugen von 3D-Illustrationen, in: Künstliche Intelligenz, FBO, erscheint 1995

[Seligmann, Feiner 91] Seligmann, D. D., Feiner, S. K., Automated Generation of Intend-Based 3D Illustrations, in: Computer Graphics (ACM) 25(4), S. 123-132, 1991

[Seligmann, Feiner 93] Feiner, S. K., Seligman, D. D., Supporting Interactivity in Automated 3D Illustrations, in: Proc. of the 1993 International Workshop on Intelligent User Interfaces, Orlando: S. 37-44, 1993

[Steele 90] Steele, G. L., Common Lisp - The Language (2nd Edition), Bedford: Digital Press, 1990

[Strothotte, Helms 95] Strothotte, T., Helms, C., Seeing Between the Pixels, to appear 1995

[Strube et al. 93] Strube, G., Habel, C., Hemforth B., Konieczny, L., Becker, B., Kognition, in: Görz, G. (Hrsg.) Einführung in die künstlich Intelligenz, Bonn: Addison-Wesley, S. 303-365, 1993

[Wahlster 92] Wahlster, W., Expertensysteme, unveröffentlichtes Vorlesungsskript, Universität des Saarlandes, Saarbrücken, Sommersemester, 1992

[Wahlster et al. 93] Wahlster, W., André, E., Finkler, W., Profitlich, H. J., Rist, T., WIP: Plan-based Integration of Natural Language and Graphics Generation, in: Artificial Intelligence 63, Amsterdam: Elsevier, S. 387-427, 1993

[Weitzman 86] Weitzman, L., Designer: A Knowledge-based Graphic Design Assistant, Tech. Report No. ICS-8609, Institute for Cognitive Science, Univ. of California, San Diego, La Jolla, Ca, 1986

[Zimmermann 93] Zimmermann, D., AnnA II: Ein wissensbasiertes System zur automatischen Annotation von Grafiken, Diplomarbeit, Universität des Saarlandes, Saarbrücken, Fachbereich für Informatik, Prof. Dr. W. Wahlster, 1993

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
PF 2080
67608 Kaiserslautern
FRG

Telefon (0631) 205-3506
Telefax (0631) 205-3210

e-mail
dfkibib@dfki.uni-kl.de
WWW
http://www.dfki.uni-sb.de/dfkibib

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder (so sie als per ftp erhaeltlich angemerkt sind) per anonymous ftp von ftp.dfki.uni-kl.de (131.246.241.100) im Verzeichnis pub/Publications bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or (if they are marked as obtainable by ftp) by anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) in the directory pub/Publications.
The reports are distributed free of charge except where otherwise noted.

DFKI Research Reports

1995

RR-95-11
Anne Kilger, Wolgang Finkler
Incremental Generation for Real-Time Applications
47 pages

RR-95-09
M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf
A Refined Architecture for Terminological Systems: Terminology = Schema + Views
71 pages

RR-95-07
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt
The Complexity of Concept Languages
57 pages

RR-95-04
M. Buchheit, H.-J. Bürckert, B. Hollunder, A. Laux, W Nutt,
M. Wójcik
Task Acquisition with a Description Logic Reasoner
17 pages

RR-95-03
Stephan Baumann, Michael Malburg, Hans-Guenther Hein, Rainer Hoch,
Thomas Kieninger, Norbert Kuhn
Document Analysis at DFKI
Part 2: Information Extraction
40 pages

RR-95-02
Majdi Ben Hadj Ali, Frank Fein, Frank Hoenes, Thorsten Jaeger,
Achim Weigel
Document Analysis at DFKI
Part 1: Image Analysis and Text Recognition
69 pages

1994

RR-94-39
Hans-Ulrich Krieger
Typed Feature Formalisms as a Common Basis for Linguistic Specification.
21 pages

RR-94-38
Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne,
Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger,
Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P Spackman.
DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling.
13 pages

RR-94-37
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 1: Overview.
54 pages

RR-94-36
Manfred Meyer
Issues in Concurrent Knowledge Engineering. Knowledge Base and Knowledge Share Evolution.
17 pages

RR-94-35
Rolf Backofen
A Complete Axiomatization of a Theory with Feature and Arity Constraints
49 pages

RR-94-34
Stephan Busemann, Stephan Oepen, Elizabeth A. Hinkelman,
Günter Neumann, Hans Uszkoreit
COSMA - Multi-Participant NL Interaction for Appointment Scheduling
80 pages

RR-94-33
Franz Baader, Armin Laux
Terminological Logics with Modal Operators
29 pages

RR-94-31
Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-23
Gert Smolka
The Definition of Kernel Oz
53 pages

RR-94-20
Christian Schulte, Gert Smolka, Jörg Würtz
Encapsulated Search and Constraint Programming in Oz
21 pages

RR-94-18
Rolf Backofen, Ralf Treinen
How to Win a Game with Features
18 pages

RR-94-17
Georg Struth
Philosophical Logics– A Survey and a Bibliography
58 pages

RR-94-16
Gert Smolka
A Foundation for Higher-order Concurrent Constraint Programming
26 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr
Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-13
Jana Koehler
Planning from Second Principles—A Logic-based Approach
49 pages

RR-94-12
Hubert Comon, Ralf Treinen
Ordering Constraints on Trees
34 pages

RR-94-11
Knut Hinkelmann
A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-10
Knut Hinkelmann, Helge Hintze
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-08
Otto Kühn, Björn Höfling
Conserving Corporate Knowledge for Crankshaft Design
17 pages

RR-94-07
Harold Boley
Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-06
Dietmar Dengler
An Adaptive Deductive Planning System
17 pages

RR-94-05
Franz Schmalhofer, J. Stuart Aitken, Lyle E. Bourne jr.
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-03
Gert Smolka
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-02
Elisabeth André, Thomas Rist
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-01
Elisabeth André, Thomas Rist
Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

1993

RR-93-48
Franz Baader, Martin Buchheit, Bernhard Hollunder
Cardinality Restrictions on Concepts
20 pages

RR-93-46
Philipp Hanschke
A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

RR-93-45
Rainer Hoch
On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt
Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-43
M. Bauer, G. Paul
Logic-based Plan Recognition for Intelligent Help Systems
15 pages

RR-93-42
Hubert Comon, Ralf Treinen
The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-41
Winfried H. Graf
LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt,
Andrea Schaerf
Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-38
Stephan Baumann
Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck,
Ralf Legleitner, Gabriele Schmidt
Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.)
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

RR-93-34
Wolfgang Wahlster
Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-33
Bernhard Nebel, Jana Koehler
Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-32
David R. Traum, Elizabeth A. Hinkelman
Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman
Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman
Corporate Agents
14 pages

RR-93-29
Armin Laux
Representing Belief in Multi-Agent Worlds via Terminological Logics
35 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker
Feature-Based Allomorphy
8 pages

RR-93-27
Hans-Ulrich Krieger
Derivation Without Lexical Rules
33 pages

RR-93-26
Jörg P. Müller, Markus Pischel
The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-25
Klaus Fischer, Norbert Kuhn
A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-24
Rainer Hoch, Andreas Dengel
Document Highlighting — Message Classification in Printed Business Letters
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy
Comparative Study of Connectionist Simulators
20 pages

RR-93-22
Manfred Meyer, Jörg Müller
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-20
Franz Baader, Bernhard Hollunder
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-18
Klaus Schild
Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-17
Rolf Backofen
Regular Path Expressions in Feature Logic
37 pages

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz
Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-15
Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles,
Markus A. Thies, Wolfgang Wahlster
PLUS - Plan-based User Support Final Project Report
33 pages

RR-93-14
Joachim Niehren, Andreas Podelski, Ralf Treinen
Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-13
Franz Baader, Karl Schlechta
A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-12
Pierre Sablayrolles
A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-11
Bernhard Nebel, Hans-Jürgen Bürckert
Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-10
Martin Buchheit, Francesco M. Donini, Andrea Schaerf
Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-09
Philipp Hanschke, Jörg Würtz
Satisfiability of the Smallest Binary Program
8 pages

RR-93-08
Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer
COLAB: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-07
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux
Concept Logics with Function Symbols
36 pages

RR-93-06
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux
On Skolemization in Constrained Logics
40 pages

DFKI Documents

1995

D-95-07
Ottmar Lutzy
Morphic - Plus
Ein morphologisches Analyseprogramm für die deutsche Flexionsmorphologie und Komposita-Analyse
74 pages

D-95-06
Markus Steffens, Ansgar Bernardi
Integriertes Produktmodell für Behälter aus Faserverbundwerkstoffen
48 Seiten

D-95-05
Georg Schneider
Eine Werkbank zur Erzeugung von 3D-Illustrationen
157 Seiten

D-95-03
Christoph Endres, Lars Klein, Markus Meyer
Implementierung und Erweiterung der Sprache *ACCP*
110 Seiten

D-95-02
Andreas Butz
BETTY
Ein System zur Planung und Generierung informativer Animationssequenzen
95 Seiten

D-95-01
Susanne Biundo, Wolfgang Tank (Hrsg.)
Beiträge zum Workshop „Planen und Konfigurieren".
Februar 1995
169 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

1994

D-94-15
Stephan Oepen
German Nominal Syntax in HPSG
— On Syntactic Categories and Syntagmatic Relations —
80 pages

D-94-14
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 2: User Guide.
72 pages

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.)
Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'94 Workshop: KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 pages

Note: This document is no longer available in printed form.

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.)
Working Notes of the 1994 International Workshop on Description Logics
118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-94-09
Technical Staff
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-08
Harald Feibel
IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-07
Claudia Wenzel, Rainer Hoch
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

D-94-06
Ulrich Buhrmann
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 Seiten

D-94-04
Franz Schmalhofer, Ludger van Elst
Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 Seiten

D-94-03
Franz Schmalhofer
Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung
54 Seiten

Note: This document is no longer available in printed form.

D-94-02
Markus Steffens
Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-01
Josua Boon (Ed.)
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

1993

D-93-27
Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman,
Hans Uszkoreit (Eds.)
Report of the EAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken
110 pages

D-93-26
Frank Peters
Unterstützung des Experten bei der Formalisierung von Textwissen INFOCOM - Eine interaktive Formalisierungskomponente
58 Seiten

D-93-25
Hans-Jürgen Bürckert, Werner Nutt (Eds.)
Modeling Epistemic Propositions
118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-93-24
Brigitte Krenn, Martin Volk
DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen
36 Seiten

D-93-22
Andreas Abecker
Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

Note: This document is no longer available in printed form.

D-93-21
Dennis Drollinger
Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-20
Bernhard Herbig
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt
Design & KI
74 Seiten

D-93-15
Robert Laux
Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-14
Manfred Meyer (Ed.)
Constraint Processing - Proceedings of the International Workshop at CSAM'93, St.Petersburg, July 20-21, 1993
264 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-93-12
Harold Boley, Klaus Elsbernd, Michael Herfert, Michael Sintek,
Werner Stein
RELFUN Guide: Programming with Relations and Functions Made Easy
86 pages

D-93-11
Knut Hinkelmann, Armin Laux (Eds.)
DFKI Workshop on Knowledge Representation Techniques — Proceedings
88 pages

Note: This document is no longer available in printed form.

D-93-10
Elizabeth Hinkelman, Markus Vonerden, Christoph Jung
Natural Language Software Registry (Second Edition)
174 pages

D-93-09
Hans-Ulrich Krieger, Ulrich Schäfer
TDL ExtraLight User's Guide
35 pages

D-93-08
Thomas Kieninger, Rainer Hoch
Ein Generator mit Anfragesystem für strukturierte Wörterbücher zur Unterstützung von Texterkennung und Textanalyse
125 Seiten

D-93-07
Klaus-Peter Gores, Rainer Bleisinger
Ein erwartungsgesteuerter Koordinator zur partiellen Textanalyse
53 Seiten

D-93-06
Jürgen Müller (Hrsg.)
Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz, Saarbrücken, 29. - 30. April 1993
235 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-93-05
Elisabeth André, Winfried Graf, Jochen Heinsohn, Bernhard Nebel, Hans-Jürgen Profitlich, Thomas Rist, Wolfgang Wahlster
PPP: Personalized Plan-Based Presenter
70 pages

D-93-04
Technical Staff
DFKI Wissenschaftlich-Technischer Jahresbericht 1992
194 Seiten

D-93-03
Stephan Busemann, Karin Harbusch (Eds.)
DFKI Workshop on Natural Language Systems: Reusability and Modularity - Proceedings
74 pages

D-93-02
Gabriele Schmidt, Frank Peters, Gernod Laufkötter
User Manual of COKAM+
23 pages

D-93-01
Philipp Hanschke, Thom Frühwirth
Terminological Reasoning with Constraint Handling Rules
12 pages

Eine Werkbank zur Erzeugung von 3D-Illustrationen

Document

D-95-05

Georg Schneider