„Interaktive Konstruktion von dreidimensionalen Szenen“

Diplomarbeit an der Universität Ulm

Fakultät für Informatik

vorgelegt von:

Michael Gösele

1. Gutachter: Prof. Dr. Wolfgang Reif

2. Gutachter: Prof. Dr. Wolfgang Stürzlinger

1998

Name: Michael Gösele

Matrikelnummer: 310 987

Erklärung

Ich erkläre, daß ich die Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 5. Dezember 1998

Zusammenfassung

Das Erstellen von Modellen für Anwendungen aus dem Bereich der Computer-Grafik ist ein sehr arbeitsintensiver Vorgang. Heutige CAD-Programme sind für das Erstellen von Modellen einzelner Objekte sehr gut geeignet, unterstützen aber das schnelle und exakte Zusammenfügen mehrerer solcher Modelle zu einer Szene kaum.

In dieser Arbeit wird ein System von Regeln für die Manipulation von Modellen in einer solchen Szene entwickelt, das auf dem „natürlichen“ Verhalten der Objekte beruht. Durch *Flächenbindungen* werden Position und Orientierung eines Objekts relativ zu anderen Objekten in der Szene festgelegt (ein Tisch steht auf dem Boden), mit der *Objektausrichtung* kann eine absolute Ausrichtung von Objekten in einer Szene erzwungen werden (ein Bild hängt gerade an der Wand) und durch die *Kollisionsvermeidung* wird gewährleistet, daß sich die Objekte in der Szene nicht gegenseitig durchdringen. Darüberhinaus wird eine Technik präsentiert, die aus den Flächenbindungen der Objekte eine dynamische Gruppierung ableitet.

Das IConS-System zur interaktiven Manipulation von Szenen unter Verwendung dieser Mechanismen wurde entwickelt und implementiert. Die Umsetzung der theoretischen Konzepte und die dabei aufgetretenen Probleme werden erläutert sowie die Ergebnisse eines kleinen Benutzertests mit dem System vorgestellt. Abschließend wird ein Ausblick auf weitere Verbesserungs- und Anwendungsmöglichkeiten dieser Technik gegeben.

Abstract

Generating object models for computer graphics applications is a labour-intensive process. Todays CAD systems address the problem of generating models of single objects quite well. But they support the task of combining several of these models to a scene in a fast and precise way only weakly.

This thesis introduces a system of constraints for manipulating object models within a scene, which is based on the "natural" behavior of the objects: *Surface constraints* are used to define an objects position and orientation relative to other objects in the scene (a table is on the floor). *Object orientation* specifies an absolute orientation of an object in the scene (a picture is hanging straight on the wall). *Collision avoidance* assures that there are no collisions between objects in the scene. Additionally there is a dynamic grouping mechanism, which is derived from the surface constraints.

The IConS system for interactive manipulation of scenes using these mechanisms was developed and implemented. The realization of the theoretical concepts and the hereby arising problems are discussed. The results of a small user test are presented. The conclusion shows possible directions for further research and additional applications of these techniques.

Vorwort

Wie so viele Dinge im Leben kann auch eine Diplomarbeit nicht von einem Einzelkämpfer, der in seinem Elfenbeinturm sitzt, bewältigt werden. Der Weg von einer Idee bis zur fertigen Diplomarbeit ist lang. Konzentriertes Arbeiten im stillen Kämmerlein ist dabei genauso wichtig wie intensives Diskutieren der Resultate mit anderen. Abgesehen davon ist auch die psychologische Unterstützung für den Schreibenden dringend notwendig... Deshalb bin ich allen, die mich bei meiner Arbeit so tatkräftig unterstützt haben, von Herzen dankbar.

Als Besonderheit kommt hinzu, daß diese Diplomarbeit an zwei Orten entstanden ist. Sie wurde während eines elfmonatigen Aufenthalts als Austauschstudent an der University of North Carolina at Chapel Hill in den USA begonnen und nach meiner Rückkehr nach Deutschland an der Universität Ulm abgeschlossen.

Mein besonderer Dank gilt meinem Betreuer Prof. Dr. Wolfgang Stürzlinger, der mich immer unterstützt und ermutigt hat. Besonders wertvoll war für mich die Freiheit, die er mir beim Forschen eingeräumt hat. Genauso möchte ich mich bei Prof. Dr. Wolfgang Reif bedanken, der sofort bereit war, das Wagnis einer „interkontinentalen Diplomarbeit" mitzutragen, und der mir in vielen Diskussionen geholfen hat, mit meiner Arbeit voranzukommen.

Beim ganzen Computer Science Department in Chapel Hill möchte ich mich bedanken für die freundliche Aufnahme und Unterstützung, die weit über das übliche Maß hinausgegangen ist. Hierbei seien besonders Prof. Dr. Ansalmo Lastra, Prof. Dr. Jan Prins und Prof. Dr. Gary Bishop sowie meine Mitstudenten Benjamin Lok, Nicholas Vallidis und Chris Wynn erwähnt. Außerdem gilt mein Dank dem Study Abroad Office in Chapel Hill und den akademischen Auslandsämtern in Ulm und Mannheim, die mir meinen Aufenthalt erst ermöglichten. Ganz privat möchte ich mich bei all den Mitmenschen bedanken, die mir die Zeit in Chapel Hill so angenehm gemacht haben: Vera Hart, Gary und Sandy Gaddy, Ken Gray, Sylvester Harrington, die Hoyles und viele andere mehr.

Auch eine Diplomarbeit ist ein Benutzerinterface – nämlich ein Benutzerinterface für die erarbeiteten Resultate. Deshalb habe ich sie – wie auch das hier vorgestellte IConS-System – einem ausführlichen Benutzertest unterworfen – ganz im Sinne von Houde (siehe Kapitel 5). Ich möchte mich an dieser Stelle bei allen Testbenutzern bzw. Korrekturlesern recht herzlich für ihre Mitarbeit und die vielen Anregungen, die sie mir gegeben haben, bedanken. Ganz besonders sei hier – aus akademischen und nicht-akademischen Gründen – Marcus Borst erwähnt.

Außerdem gilt mein ganz spezieller Dank meinen Eltern Erika und Helmut Gösele, meinen Großeltern und meinem ganzen Familien- und Freundeskreis, die mich auf meinem ganzen Weg so oft unterstützt haben. Ohne sie wäre all dies nicht möglich gewesen.

Michael Gösele, Dezember 1998

Inhaltsverzeichnis

1 Einleitung **5**

2 Grundlagen **7**
2.1 Grundlagen der Modellierung 7
2.1.1 Begriffsdefinitionen 7
2.1.2 Statische und dynamische Szenen 8
2.1.3 Bearbeitung von Objektmodellen und Szenen 8
2.2 Beschreibung der Problemstellung 9
2.2.1 Objektpositionierung in der Realität 9
2.2.2 Objektpositionierung am Computer 10
2.2.3 Schlußfolgerung 10
2.3 Arten von Modellen 10
2.3.1 Volumen- und Oberflächenmodelle 10
2.3.2 Ebene und gekrümmte Oberflächen 11
2.4 Vorgängersysteme 11
2.4.1 Systeme mit Constraints 11
2.4.2 Direkte Manipulation 12
2.4.3 Deklarative Systeme 13
2.5 Weiterentwicklung 13

3 Modellierung mit Constraints **14**
3.1 Überblick über das Constraint-Konzept 14
3.2 Flächenbindungen 15
3.2.1 Bindungs- und Angebotsflächen 15
3.2.2 Mehrere Bindungsflächen pro Objekt 16
3.2.3 Ausnutzung der Funktion von Gegenständen 16
3.2.4 Einschränkung der Bewegung durch erfüllte Flächenbindungen 17
3.3 Kollisionsvermeidung 17
3.3.1 Kollision bei erfüllten Flächenbindungen 18
3.3.2 Umgehung des Problems 18
3.4 Objektausrichtung 19
3.4.1 Funktionsweise 19
3.4.2 Vergleich mit Flächenbindungen 20
3.5 Weitere Hilfsmittel 20
3.5.1 Pseudo-Gravitation 20
3.5.2 Instantiierung von Objekten 20
3.6 Der Bindungsgraph 21
3.6.1 Flächenbindungen und dynamisches Gruppieren 22
3.6.2 Spezialfälle für den Bindungsgraphen 22
3.7 Der IConS-Szenengraph 24
3.7.1 Objektdaten und -instanzen 24
3.7.2 Dynamische Gruppierung 24

3.7.3 Umbau des Szenengraphen . . . 26
3.8 Formalisierung . . . 27
3.8.1 Einzelne Constraints . . . 27
3.8.2 Konsistenz einer Szene . . . 28
3.9 Interaktionsmodell . . . 28
3.9.1 Überblick über das Gesamtsystem . . . 29
3.9.2 Suche nach erfüllbaren Flächenbindungen . . . 30
3.9.3 Bewegungseinschränkung durch erfüllte Flächenbindungen . . . 32
3.9.4 Aufbruch von erfüllten Flächenbindungen . . . 34
3.10 Eingabe einer Transformation . . . 34
3.10.1 Grundprinzip der Eingabe . . . 34
3.10.2 Translation . . . 35
3.10.3 Rotation . . . 36

4 Implementierung **39**
4.1 Verwendete Hard- und Software . . . 39
4.1.1 OpenGL Optimizer . . . 39
4.2 Externe Software . . . 39
4.2.1 Kollisionserkennung mit V-COLLIDE . . . 40
4.2.2 Formatierte Ausgabe des Szenengraphen mit Graphplace . . . 40
4.2.3 Arcball-Rotationsinterface . . . 41
4.3 Bedienung des IConS-Systems . . . 41
4.3.1 Eingabe . . . 41
4.3.2 Bildschirmausgabe . . . 41
4.4 Leitlinien für die Definition von Angebots- und Bindungsflächen . . . 41
4.4.1 Rechenaufwand . . . 43
4.4.2 Physikalische Exaktheit . . . 43
4.4.3 Numerische Stabilität . . . 43
4.4.4 Ausreichende Flexibilität . . . 44
4.5 Vorverarbeitung der Modelle . . . 45
4.5.1 Objektdaten . . . 45
4.5.2 Erzeugung der Constraintdatei . . . 45
4.6 Zwei interessante Routinen . . . 47
4.6.1 Bestimmung einer Rotationsachse . . . 47
4.6.2 Positionierung von Objekten . . . 48
4.7 Numerische Probleme . . . 48
4.7.1 Programmierebene . . . 49
4.7.2 Benutzerebene . . . 49

5 Ergebnisse **50**
5.1 Benutzertest . . . 50
5.1.1 Durchführung . . . 50
5.1.2 Aufgabenstellung . . . 51
5.1.3 Ablauf der Benutzertests . . . 51
5.1.4 Gewonnene Ergebnisse . . . 52
5.1.5 Veränderungen am System . . . 53
5.2 Beispielinteraktionen . . . 54
5.3 Erweiterungsmöglichkeiten . . . 54

6 Zusammenfassung und Ausblick **62**
6.1 Zusammenfassung . . . 62
6.2 Ausblick . . . 62

A Materialien zum Benutzertest **64**
A.1 Aufgabenstellung 64

Kapitel 1

Einleitung

Durch die rasante Entwicklung auf dem Bereich der Grafikhardware und durch immer bessere Algorithmen kann man heute sehr realistische und detailreiche Bilder am Computer erzeugen. Inzwischen existiert bereits eine breite Palette von Anwendungen aus unterschiedlichen Bereichen wie Innenarchitektur, Computerspiele oder Kinofilme, die erstaunlich gute Bilder und Animationen – teilweise in Echtzeit – erzeugen.

Als Basis für diese Berechnungen dient das *Modell* einer Szene, das alle für die Erzeugung eines Bildes relevanten Daten enthält. Dazu gehören neben der reinen Geometrie der Szene je nach Situation auch Informationen über die Beschaffenheit von Oberflächen, Lichtverhältnisse und andere Parameter. Aufgrund der technischen Entwicklung ist es heute möglich, sehr detailliertere Modelle zu generieren, die eine große Zahl einzelner Objekte enthalten und dadurch auch realistisch wirken.

Betrachtet man zum Beispiel ein Büro, so enthält dies im wesentlichen einen Schreibtisch mit Stuhl, einige Regale oder Schränke und eine Lampe – mehr wird als Kulisse in vielen Theateraufführungen auch nicht eingesetzt. Das Büro wird aber steril und künstlich wirken, solange es nicht auch eine ganze Reihe von „Kleinigkeiten“ wie Stifte, Papier, ein Telefon, Bücher, einen gefüllten Papierkorb, ... enthält. In einem Kinofilm, der in einem realistischen Umfeld spielen soll, wird deshalb die Kulisse mit sehr viel Liebe zum Detail eingerichtet.

Das Erzeugen eines Modells einer solchen Szene ist ein sehr arbeitsintensiver Vorgang, der sich in zwei Schritte unterteilen läßt. Zuerst werden normalerweise Modelle der einzelnen Objekte erstellt, die dann zu einem Modell der ganzen Szene zusammengefaßt werden. Der erste Schritt wird schon heute sehr gut durch Konstruktionsprogramme unterstützt. Aufgrund der großen Ähnlichkeit dieses Schritts mit dem Konstruktionsprozeß im Maschinenbau kann bei der Entwicklung dieser Programme bereits auf einen umfangreichen Erfahrungsschatz zurückgegriffen werden. Außerdem ist eine große Zahl guter Modelle im Internet oder von kommerziellen Anbietern erhältlich.

Für den zweiten Schritt, den Zusammenbau des Modells einer ganzen Szene aus diesen Objektmodellen, gibt es dagegen wenige gut geeignete Systeme. Das Problem hierbei ist, daß eine große Zahl vorgefertigter Modelle schnell, exakt und möglichst intuitiv innerhalb der Szene plaziert werden muß. Dazu ist eine explizite Unterstützung durch das Modellierungssystem unerläßlich.

Ziel der vorliegenden Arbeit ist deshalb der Entwurf und die Implementierung eines Systems zur Erstellung und Bearbeitung eines Modells einer Szene aus vorgefertigten Objektmodellen. Dabei soll die Interaktion mit den Objektmodellen durch die Ausnutzung des „natürlichen“ Verhaltens der Objekte wesentlich vereinfacht und intuitiv gestaltet werden. Dazu gehört zum Beispiel, daß ein Tisch auf dem Boden steht und ein Bild an der Wand hängt, aber auch, daß sich alle Objekte, die auf einem Tisch stehen, beim Bewegen des Tischs mitbewegen. Auf diese Weise kann das System Beziehungen zwischen den einzelnen Objekten in der Szene herstellen und auf die Eingaben des Benutzers unter Berücksichtigung dieser Beziehungen reagieren.

Die Arbeit hat folgenden Aufbau: Kapitel 2 enthält Grundlagen der Objektmodellierung und der Objektpositionierung, eine ausführliche Beschreibung der Problemstellung und verschiedene Ansätze zum Bau von Szenen. In Kapitel 3 werden die theoretischen Prinzipien des Systems, das

im Rahmen dieser Arbeit entwickelt wurde, und ihre Umsetzung erläutert. Einige bei der Implementierung wichtigen Details sind in Kapitel 4 dargestellt. Kapitel 5 beschreibt die Ergebnisse eines informellen Benutzertests und zeigt beispielhaft den Ablauf einiger Interaktionen. Kapitel 6 enthält eine Bewertung des Systems und gibt einen Ausblick auf weitere Arbeiten ausgehend von dieser Grundlage.

Kapitel 2

Grundlagen

Zu Beginn werden hier die für das Verständnis wichtigen Grundlagen der Modellierung erläutert. Danach folgt eine ausführliche Darstellung der Problemstellung, ein Überblick über verschiedene Modellarten und über bereits existierende Arbeiten zur Modellierung von Szenen. Den Abschluß bildet eine kurze Übersicht darüber, wie diese Arbeiten in die vorliegende Diplomarbeit eingeflossen sind.

2.1 Grundlagen der Modellierung

2.1.1 Begriffsdefinitionen

Form und Funktion

Ein realer Gegenstand wird nach Rooney und Steadman durch zwei Eigenschaften beschrieben: *Form* und *Funktion* [RS90]. Unter der Form eines Gegenstandes versteht man unter anderem seine Geometrie, Farbe, Oberflächenbeschaffenheit oder Dichte. Die Funktion beschreibt, wie sich der Gegenstand verhält (ein Motor „erzeugt“ eine Bewegung) oder wie er normalerweise verwendet wird (ein Stuhl ist eine Sitzgelegenheit). Während sich die Form durch objektive Aussagen beschreiben läßt, besteht die Funktion zu großen Teilen aus subjektiven Eigenschaften und ist damit abhängig vom einzelnen Betrachter.

Die Beziehung zwischen Form und Funktion ähnelt dem Welle – Teilchen-Dualismus in der Physik. Jeder Gegenstand hat eine Form und eine Funktion, aber je nach Standpunkt und Intention des Betrachters steht eine der beiden Eigenschaften im Vordergrund [RS90]. In der Computergrafik ist meist nur die Form der Gegenstände von Interesse. Wenn aber der Benutzer mit ihnen interagieren soll, kann auch der Funktion eine große Bedeutung zukommen.

Modelle und Szenen

Ein *Modell* eines Gegenstandes ist eine – im allgemeinen nicht vollständige – Menge von Informationen über dessen Form und Funktion. Im Rahmen dieser Arbeit enthält ein *Objektmodell* eine Beschreibung der Geometrie eines Gegenstandes und Informationen über seine Funktion. Auf die verschiedenen Möglichkeiten, die Geometrie zu beschreiben, wird in Abschnitt 2.3 eingegangen und in Abschnitt 3.2.3 wird die Repräsentation der Funktion eines Objekts erläutert. Ein Objektmodell wird als unveränderbar und nicht in kleinere Einheiten aufteilbar betrachtet.

Oftmals ist eine Unterscheidung zwischen einem Gegenstand und dem zugehörigen Objektmodell nicht nötig. In diesem Fall wird für beide der Begriff *Objekt* verwendet.

Das Hauptaugenmerk dieser Arbeit liegt auf dem Zusammensetzen mehrerer Objektmodelle zu einer *Szene* und der Positionierung der Objektmodelle innerhalb der Szene. Neben den Objektmodellen, die die einzelnen Objekte beschreiben, enthält eine Szene auch Informationen über die Beziehungen der Objekte zueinander. Dazu gehört unter dem Aspekt der Form die Position

und Orientierung der Objekte relativ zueinander oder bezüglich des globalen Koordinatensystems der Szene. Eine Szene kann aber auch Informationen über funktionale Beziehungen zwischen den Objekten enthalten. Die Aussage „Das Glas steht auf dem Tisch“ ist ein Beispiel für eine solche funktionale Beziehung.

2.1.2 Statische und dynamische Szenen

Die Verwendung von Szenen läßt sich nach Abbildung 2.1 in zwei Kategorien unterteilen: Eine *statische* Szene wird einmal erzeugt und dann als Eingabe für eine Anwendung verwendet. Eine solche Anwendung kann zum Beispiel Bilder der Szene erzeugen oder es dem Benutzer ermöglichen, sich interaktiv durch die Szene zu bewegen. Wichtig ist, daß die Anwendung die Szene selbst nicht verändert.

Im Gegensatz dazu wird eine *dynamische* Szene durch eine Anwendung verändert. Objekte können innerhalb der Szene bewegt, gelöscht oder auch zur Szene hinzugefügt werden. Diese Anwendung kann es dem Benutzer zum Beispiel ermöglichen, sich nicht nur durch die Szene zu bewegen, sondern sie dabei auch zu verändern (siehe etwa [Buk95]).

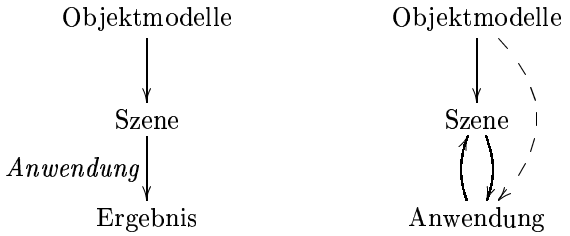

Abbildung 2.1: Statische (links) und dynamische Szenen. Eine statische Szene wird einmalig aus verschiedenen Objektmodellen zusammengesetzt. Diese Szene wird als Eingabe für eine Anwendung verwendet, die ein Ergebnis erzeugt, ohne die Szene selbst zu verändern. Eine dynamische Szene kann von der Anwendung und oft auch interaktiv vom Benutzer verändert werden.

Eine wichtige Beobachtung ist, daß auch eine statische Szene einen dynamischen Anteil hat. Bei der Erstellung einer statischen Szene aus den Objektmodellen handelt es sich um einen dynamischen Vorgang. Damit kommt dem Umgang mit einer dynamischen Szene eine sehr hohe Bedeutung zu, da er wesentlich für das Arbeiten mit jeder Art von Szenen ist.

2.1.3 Bearbeitung von Objektmodellen und Szenen

Die Anforderungen bei der Erstellung und Bearbeitung von Objektmodellen unterscheiden sich stark von den Anforderungen für das Arbeiten mit einer Szene. Daher müssen für beide Aufgabenstellungen unterschiedliche Strategien verfolgt werden.

Objektmodell

Bei der Erstellung und Bearbeitung eines Objektmodells wird ein neues Objekt im Computer aus einer Reihe sehr einfacher geometrischer Primitive (zum Beispiel Punkte, Polygone, Quader, Kegel) erschaffen. Je nach Aufgabenstellung und Typ des Modells (siehe dazu Abschnitt 2.3) kann eine Reihe von Operationen auf diese Primitive angewendet werden. Außerdem können bei der Erstellung eines Objektmodells normalerweise auch selbsterstellte oder in Bibliotheken erhältliche Objektmodelle geladen und mitverwendet werden.

Die Manipulation von Objektmodellen ist eine klassische Konstruktionsaufgabe. Die meisten Programme, die dafür eingesetzt werden, sind deshalb computergestützte Konstruktionssysteme (*computer aided design*, *CAD*). Bekannte Beispiele hierfür sind das Sharewareprogramm AC3D [Col98] oder das kommerziell erhältliche AUTOCAD. Neben einer großen Anzahl von Primitiven

und möglichst flexiblen Operationen, die auf die Primitive angewendet werden können, müssen diese Programme auch ein exaktes Positionieren unterstützen. Dies gilt nicht nur für die klassischen Konstruktionsaufgaben wie etwa bei der Konstruktion eines Motors sondern auch beim Erstellen von Modellen für die Computergrafik. Bereits kleinste Ungenauigkeiten oder Risse in einem Modell können in einem daraus erzeugten Bild ins Auge stechen.

Aus diesen Gründen ist die Erstellung eines Objektmodells ein recht arbeitsintensiver Vorgang. Allerdings läßt sich der erhebliche Aufwand damit rechtfertigen, daß ein fertiges Objektmodell immer und immer wieder eingesetzt werden kann. Außerdem bereitet die Zusammenstellung von Objekten zu einer Szene deutlich weniger Aufwand, wenn nicht auch noch auf Unzulänglichkeiten der einzelnen Objektmodelle Rücksicht genommen werden muß.

Szenen

Im Gegensatz dazu ist es beim Arbeiten an einer Szene sehr wichtig, zügig voran zu kommen. Eine realistische Szene kann leicht aus Hunderten oder gar Tausenden von Objekten bestehen, die zum Beispiel beim Einrichten eines virtuellen Hauses mit Möbeln und Alltagsgegenständen plaziert werden müssen.

Trotzdem kann dabei nicht auf ein exaktes Arbeiten verzichtet werden. Auch hier stechen beim Betrachten eines Bildes, das aus einer Szene erzeugt worden ist, Lücken zwischen den Objekten oder freischwebende Objekte ins Auge. Außerdem können Objekte nach der Anwendung von Rotationen oder bei der Verwendung spezieller Eingabehardware, die die Bewegung eines Objekts mit allen sechs Freiheitsgraden erlaubt, schräg im Raum stehen. Ohne explizite Unterstützung durch das System können solche Objekte kaum wieder gerade ausgerichtet werden.

In CAD-Systemen werden zum exakten Arbeiten Hilfsebenen und Gitter im Raum verwendet. In vielen Fällen ist sogar die manuelle Eingabe von Koordinaten notwendig. Dies kann zwar zu hoher Exaktheit führen, erlaubt aber sicher kein schnelles Interagieren mit der Szene. Andererseits können bei der Konstruktion von Szenen Beziehungen zwischen Objekten hergestellt und ausgenutzt werden, was wiederum beim Bau eines Objektmodells kaum möglich ist.

2.2 Beschreibung der Problemstellung

2.2.1 Objektpositionierung in der Realität

Abgesehen von der körperlichen Anstrengung ist es kein Problem, einen Schrank so in ein Zimmer zu stellen, daß er gerade auf dem Boden steht und bündig mit der Wand abschließt. Auch ein Kleinkind kann sein Spielzeug gerade auf den Boden stellen.

Diese Aufgaben sind vor allem deshalb so einfach, weil sie durch das physikalische Verhalten der Gegenstände unterstützt werden.

Physikalische Hilfen

Für das einfache Positionieren von Objekten sind im wesentlichen zwei physikalische Eigenschaften verantwortlich: das Bewegungsverhalten der Objekte, das hauptsächlich durch die Gravitation und die Reibung bestimmt wird, und die Tatsache, daß sich Objekte nicht gegenseitig durchdringen können.

Beide Effekte zusammen sorgen dafür, daß Objekte nach unten fallen und sich beim Auftreffen normalerweise mit einer Seitenfläche an den darunter liegenden Objekten ausrichten. Außerdem kann man einen Schrank parallel zu einer Wand ausrichten, indem man ihn gegen die Wand schiebt.

Grenzen

Es gibt allerdings Fälle, in denen es nicht möglich ist, diese Effekte auszunutzen. Ein Beispiel ist das gerade Aufhängen eines Bildes an der Wand. Dies erfordert entweder ein gutes Augenmaß

oder die Verwendung von Werkzeugen wie Wasserwaage oder Meterstab, denn das Bild kann nicht einfach wie der Schrank durch Schieben gegen den Boden ausgerichtet werden.

Die beiden Werkzeuge Wasserwaage und Meterstab entsprechen zwei verschiedenen Lösungsansätzen für das Problem. Mit einer Wasserwaage kann man eine absolute Orientierung an einem beliebigen Punkt im Raum festlegen und Dinge anhand dieser absoluten Orientierung ausrichten. Hängt man ein Bild dagegen unter Verwendung eines Meterstabs so auf, daß der Abstand der Unterkante zum Boden immer gleich ist, so wird der Boden als Referenz für die Ausrichtung des Bildes verwendet. Das Bild wird dabei relativ zum Boden ausgerichtet.

2.2.2 Objektpositionierung am Computer

In Abschnitt 2.1.3 wurde erläutert, daß es beim Positionieren von Objekten in einer Szene vor allem auf exaktes und schnelles Arbeiten ankommt. Gleichzeitig befindet man sich aber in einer noch schwierigeren Situation als beim Aufhängen eines Bildes. Ein CAD-Programm verhindert normalerweise nicht, daß sich Objekte gegenseitig durchdringen, und kann auch die Bewegung eines Objekts nicht simulieren, um bei der Orientierung und auch beim Herstellen eines Kontakts zwischen Objekt und Auflagefläche behilflich zu sein.

Dabei hat der Benutzer des Programms dieselben Freiheiten bei der Positionierung der Objekte wie in der Realität. Verwendet er nur ein zweidimensionales Eingabegerät (Maus, Grafiktablett, ...), so kann er Translationen und Rotationen voneinander trennen und hat damit auch weniger Probleme mit der Orientierung eines Objekts. Aber sobald Eingabegeräte mit sechs Freiheitsgraden verwendet werden, mit denen ein Objekt beliebig verschoben und rotiert werden kann, wird exaktes und zugleich schnelles Arbeiten ohne Unterstützung durch das System zur Illusion. Dies gilt auch dann, wenn ein System wie das WALKTHRU-System [Buk95] eine exakte Eingabe der Position und Orientierung von Objekten durch manuelle Eingabe von Koordinaten erlaubt. Ein optisches Feedback reicht für ein derartiges Arbeiten nicht aus.

2.2.3 Schlußfolgerung

Zur Erstellung und Bearbeitung von Szenen am Computer ist eine weitgehende Unterstützung durch das verwendete System beim Positionieren und Orientieren der Objekte innerhalb der Szene unerläßlich. Es bietet sich an, die Art der Unterstützung an den natürlichen Eigenschaften realer Objekte zu orientieren, mit denen der Benutzer bereits vertraut ist. Dadurch kann dem Benutzer ein intuitives Arbeiten mit der Szene ermöglicht werden.

2.3 Arten von Modellen

Dieser Abschnitt beschäftigt sich mit den verschiedenen Möglichkeiten, wie die Geometrie eines Gegenstandes in einem Objektmodell repräsentiert werden kann. Dabei wird hauptsächlich auf die Details eingegangen, die im weiteren Verlauf dieser Arbeit wichtig werden. Für eine ausführlichere Darstellung, die den Rahmen dieser Arbeit sprengen würde, sei auf Standardwerke der Computergrafik wie etwa [FvDFH96] verwiesen.

2.3.1 Volumen- und Oberflächenmodelle

In einem Volumenmodell wird ein Gegenstand als geschlossenes Volumen dargestellt. Mit dieser Technik ist es möglich, das Volumen des Originals exakt zu repräsentieren. Ein Volumenmodell wird zum Beispiel benötigt, wenn man physikalische Eigenschaften wie Schwerpunkt oder Trägheitstensor berechnen möchte. Außerdem ist es nur so möglich, exakte Kollisionen zwischen verschiedenen Objekten zu berechnen und zuverlässig zwischen Punkten innerhalb, außerhalb oder auf der Oberfläche eines Objekts zu unterscheiden.

In der Computergrafik ist es in vielen Fällen aber gar nicht notwendig, alle diese Informationen zur Verfügung zu haben. Vielmehr genügt es zum Beispiel zum Erzeugen eines Bilds eines Objekts,

wenn die Außenflächen des Objekts bekannt sind. Die Verwendung von Oberflächenmodellen kann den Aufwand für die Erstellung eines Objektmodells stark verringern und auch die benötigte Datenmenge erheblich reduzieren. Eine weitere, in der Praxis oft genutzte Vereinfachungsmöglichkeit ergibt sich, wenn die möglichen Blickrichtungen, aus denen ein Objekt betrachtet werden kann, eingeschränkt sind. Die Unterseite eines Objekts, das nicht von unten betrachtet werden kann, braucht auch nicht modelliert zu werden.

In gewisser Weise läßt sich die Erstellung von Oberflächenmodellen mit dem Bau von Filmkulissen vergleichen. Anstatt eines ganzen Hauses wird nur die der Kamera zugewandte Fassade gebaut.

2.3.2 Ebene und gekrümmte Oberflächen

Um ein Modell eines realen Objekts zu erstellen, für das keine mathematisch exakte Beschreibung der Geometrie verfügbar ist, müßte man die Koordinaten der unendlich vielen Punkte verwenden, aus denen sich die Oberfläche zusammensetzt. Dies ist natürlich nicht möglich. In der Praxis werden deshalb die Oberflächen von Volumen- und Oberflächenmodellen durch geometrische Objekte approximiert.

Die einfachste Möglichkeit besteht in der linearen Approximation der Oberfläche durch ebene Polygone. Hierfür werden vorwiegend Dreiecke oder Vierecke eingesetzt, die sehr einfach zu zeichnen sind. Dreiecke haben den zusätzlichen Vorteil, daß sie immer konvex und planar sind – Eigenschaften, die für allgemeine Polygone im Raum nicht gelten.

Durch die Approximation mit Polygonen entstehen bei Objekten mit gekrümmten Oberflächen unerwünschte Kanten. Diese Artefakte lassen sich beim Generieren eines Bildes durch Beleuchtungs- und Schattierungsalgorithmen weitgehend unterdrücken. Daher wird diese Art der Oberflächenrepräsentation in sehr vielen Anwendungen eingesetzt.

Möchte man allerdings den Kotflügel eines Autos modellieren oder anhand eines Modells die Bewegung eines Eis auf einer schiefen Ebene simulieren, ist eine Approximation mit ebenen Oberflächen wenig sinnvoll. Um gute Ergebnisse zu erzielen, ist eine so große Anzahl von Dreiecken nötig, daß Approximationen höherer Ordnung wie zum Beispiel Freiformflächen effizienter sind (siehe [FvDFH96]).

2.4 Vorgängersysteme

In Abschnitt 2.2 wurde erläutert, daß der Benutzer beim Erstellen und Bearbeiten einer Szene vom System unterstützt werden sollte. Hierfür gibt es bereits eine Reihe von Lösungsansätzen, von denen einige in den folgenden Abschnitten vorgestellt werden.

2.4.1 Systeme mit Constraints

Sowohl bei der Objekterstellung als auch beim Arbeiten mit einer Szene kann man Zwangsbedingungen (*constraints*) für die Bewegung von ganzen Objekten oder von Teilen der Objekte aufstellen. Diese Constraints können entweder nur temporär während einer einzelnen Interaktion gelten oder dauerhafte Beziehungen zwischen verschiedenen Teilen einer Szene ausdrücken.

Temporäre Constraints

In klassischen, dreidimensionalen CAD-Systemen wie AUTOCAD oder AC3D stehen dem Benutzer mehrere zweidimensionale Ansichten eines Objekts zur Verfügung. Der Benutzer kann einzelne Punkte mit einem Eingabegerät (zum Beispiel Maus, Grafiktablett) auswählen und parallel zur Bildebene der verwendeten Ansicht verschieben. Durch die Auswahl der Ansicht für eine Interaktion wird damit ein temporärer Constraint erzeugt, der die Bewegung der Punkte auf eine Ebene beschränkt.

Ein anderes Konzept, das mit nur einer Ansicht auskommt, ist *Snap-Dragging in 3D* [Bie90]. Ein dreidimensionaler Cursor (*skitter*) wird in der Szene parallel zum Mauszeiger bewegt. Er wird

durch eine spezielle Gravitationsfunktion von den Ecken, Kanten und Seitenflächen der Objekte angezogen. Dadurch kann er zur Konstruktion und Positionierung relativ zu den bereits in der Szene vorhandene Objekten verwendet werden. Zusätzlich zu den existierenden Objekten können explizit *alignment objects* definiert werden, die genauso wie die anderen Objekte in der Szene als Anziehungspunkte für den *skitter* verwendet werden aber nicht zur eigentlichen Szene gehören.

Beide Ansätze ermöglichen das exakte Positionieren und Orientieren von Objekten in drei Dimensionen mit zweidimensionalen Ein- und Ausgabegeräten. Dabei erlaubt die Verwendung eines skitters in einer einzigen Ansicht eine Positionierung in drei Dimensionen in nur einem Arbeitsgang, womit ein intuitiveres Arbeiten als in einem klassischen CAD-System möglich ist. Der Benutzer muß allerdings noch explizit *alignment objects* definieren. Außerdem verursacht die große Zahl von Anziehungspunkten beim Arbeiten mit realen Objekten nicht nur einen erheblichen Rechenaufwand sondern führt auch zu einem Verlust an Übersichtlichkeit, da der Benutzer die Lage des *skitters* in der dreidimensionalen Szene nur noch schwer erkennen kann.

Dauerhafte Constraints

Der Objekt- und Szeneeditor Sced [Che98] arbeitet ebenfalls mit temporären Constraints zur Übersetzung der Mausbewegung in eine dreidimensionale Position. Der Benutzer kann dabei durch die Auswahl des Startpunktes bestimmen, welchen der temporären Constraints er verwenden möchte.

Darüberhinaus können in Sced auch explizit dauerhafte Constraints für die Objekte definiert werden. Jedes Objekt wird durch eine Reihe von ausgezeichneten Punkten (*features*) definiert. Dazu gehören zum Beispiel die Position, die durch den Mittelpunkt festgelegt wird, oder die Skalierung, die durch den Abstand eines Punktes vom Mittelpunkt bestimmt wird. Bei Freiformflächen existieren eine Reihe von Kontrollpunkten zur Festlegung der Form. Diese *features* können durch die Constraints in Beziehung zu absoluten Punkten in der Szene gesetzt werden. Außerdem können sie durch Beziehungen zwischen Punkten in verschiedenen Objekten definiert werden.

Bei jeder Veränderung der Szene werden die definierten Constraints aufrechterhalten. Dies geschieht – je nach Definition der Constraints – entweder durch die Einschränkung der Beweglichkeit und Veränderbarkeit eines Objekts oder dadurch, daß die Bewegung oder Veränderung eines Objekts auch zu einer Bewegung oder Veränderung anderer Objekte führt.

Durch eine gezielte Verwendung der zur Verfügung stehenden Constraints ist ein exaktes Erstellen von Objekten und Szenen möglich. Die Definition der einzelnen Constraints ist allerdings aufwendig und erfordert eine sorgfältige Planung. Das Erstellen einer Szene durch Laden, probeweises Positionieren und Löschen von Objekten ist kaum möglich.

2.4.2 Direkte Manipulation

Houde beschreibt die Entwicklung eines einfachen Interfaces zur Manipulation von dreidimensionalen Objekten mit einer Maus [Hou92]. Diesem System liegt der Gedanke zugrunde, für jedes Möbelstück eine natürliche Auflageebene zu definieren. Die Objekte verlassen diese Ebene nur, wenn sie explizit hochgehoben werden. Als Beispiele werden Tische und Stühle genannt, die normalerweise auf dem Boden stehen, oder Bilder, die an der Wand hängen. Davon ausgehend ist es möglich, ein Interaktionsmodell zu entwerfen, das aus nur drei Operationen besteht: Verschieben des Objekts parallel zur Auflageebene, explizites Bewegen senkrecht zur Auflageebene und Rotieren des Objekts um eine Achse senkrecht zur Auflageebene. Durch Anbringen von verschiedenen Griffen (*handles*) an den Objekten können alle Operationen einfach mit einer Ein-Tasten-Maus durchgeführt werden.

Dieses System führt zu einer Reduktion der Freiheitsgrade der Objekte. Darüberhinaus wird für jedes Objekt eine Ebene eingeführt, anhand derer sich Translationen und Rotationen ausrichten lassen.

Auf diesem Konzept aufbauend wurde das Konzept der *object associations* entworfen [BS95, Buk95]. Hierbei wird ebenfalls eine Auflageebene benutzt, in der ein Objekt per Definition verschoben werden kann. Die Positionierung eines Objekts ist in zwei Phasen aufgeteilt. Nach einer Verschiebung des Objekts innerhalb der Auflageebene (*relocation phase*) wird in einem zweiten

Schritt vom System eine Verschiebung senkrecht zu dieser Ebene erzeugt, um eine Berührung zwischen dem verschobenen Objekt und seiner Umgebung zu erzwingen (*association phase*). Wird etwa ein auf einem Tisch stehendes Objekt über die Tischkante hinausgeschoben, so fällt es auf den Boden herab. Wird ein Objekt unter einen Tisch geschoben, bis es von der Tischplatte verdeckt ist, so wird es auf die Tischplatte angehoben. Je nach verwendeter Ansicht kann es dadurch zum Beispiel unmöglich sein, einen Papierkorb unter einen Tisch zu stellen. Das vorgestellte Konzept arbeitet dabei mit einer Pseudo-Gravitation, durch die Objekte nicht nur nach unten fallen können, sondern auch nach oben (etwa für Deckenlampen) oder an die nächstliegende Wand.

Dieses System ermöglicht eine sehr einfache Interaktion mit einer Szene, da die Objekte automatisch zu den anderen Objekten innerhalb der Szene in Beziehung gesetzt werden. Durch die Möglichkeit, die Objekte in verschiedene Richtungen fallen zu lassen, hat die Funktion eines Objekts Einfluß auf seine Verhaltensweise.

2.4.3 Deklarative Systeme

Bei allen bisher vorgestellten Systemen gestaltet der Benutzer eine Szene durch direkte Interaktionen mit den einzelnen Objekten in der Szene. In einem *deklarativen System* wie [DH93] erstellt der Benutzer eine verbale Beschreibung der Objekte und der Szene. Die Beschreibung kann ganze Sätze wie „The table is seventy centimeters high and its width and depth are unfixed." oder „The table and the two chairs are on the carpet." enthalten. Das System versucht dann eine Szene zu erstellen, die dieser Beschreibung entspricht.

Der Aufbau einer Szene erfolgt normalerweise in mehreren Verfeinerungsschritten. Der Benutzer modifiziert die Beschreibung solange, bis die vom System erzeugte Szene seinen Vorstellungen entspricht. Das System muß dabei sowohl mit mehrdeutigen als auch mit widersprüchlichen Beschreibungen umgehen können.

Der interessante Punkt an diesem Ansatz ist, daß die Beschreibungen ein Ausdruck der Funktion sind, die die einzelnen Objekte für den Benutzer haben. In der vom System erstellten Szene sind damit alle Objekte entsprechend ihrer Funktion positioniert. Der Benutzer definiert bei jedem Verfeinerungsschritt die Funktion der Objekte genauer.

Ein großer Nachteil dieses Vorgehens ist, daß jede gewünschte Veränderung des Systems in der Beschreibungssprache formuliert werden muß. Dies erschwert ein intuitives Vorgehen und kann die Erstellung einer Szene sogar unmöglich machen, falls die Beschreibungssprache nicht mächtig genug ist, um eine Beschreibung dieser Szene zu erstellen. Außerdem erfordert die Erstellung der Szene einen erheblichen Rechenaufwand, so daß ein interaktives Arbeiten nur bei kleineren Beispielen möglich ist.

2.5 Weiterentwicklung

Im Rahmen dieser Diplomarbeit wurde das IConS-System (**I**nteractive **Con**struction of three-dimensional **S**cenes) entwickelt, in welchem eine Reihe der hier vorgestellten Ideen aufgegriffen, in einem System zusammengefaßt und weiterentwickelt wurden.

Als Basis dient die Idee der *object associations*, daß sich Objekte bei einer Bewegung an anderen Objekten in der Szene ausrichten und daß auch ein Kontakt zu diesen Objekten hergestellt wird. Dies geschieht allerdings nur, wenn die Objekte auch von ihrer Funktion her zusammenpassen. Um dies entscheiden zu können, wurde ein System entwickelt, das – ähnlich wie in deklarativen Systemen – Beziehungen zwischen Objekten ausdrücken kann. Mit diesem System können dauerhafte Constraints zwischen den Objekten abgeleitet werden, die die weitere Bewegungsfähigkeit der Objekte in der Szene einschränken.

Kapitel 3

Modellierung mit Constraints

In den ersten Abschnitten werden die für die Modellierung wichtigen theoretischen Grundkonzepte diskutiert. Abschnitt 3.5 beschreibt zwei weitere Hilfsmittel für das Arbeiten mit einer Szene. Durch die innere Struktur einer Szene ergibt sich eine dynamische Gruppierung, die in den Abschnitten 3.6 und 3.7 vorgestellt wird. Es folgt eine Formalisierung der Konzepte (Abschnitt 3.8), ein Überblick über die daraus resultierenden Konsequenzen für das Arbeiten mit IConS (Abschnitt 3.9) sowie eine Beschreibung der Benutzeroberfläche (Abschnitt 3.10).

3.1 Überblick über das Constraint-Konzept

Die Basis des IConS-Systems bilden eine Reihe von Constraints, die das Bewegungsverhalten der Objektmodelle innerhalb der Szene bestimmen. Diese Constraints gehen weit über die in Sced [Che98] vorgestellten geometrischen Constraints hinaus. Dort werden durch die Constraints rein geometrische Beziehungen zwischen einzelnen Objektmodellen in der Szene hergestellt, die bei jeder Bewegung erfüllt sein müssen. Im Gegensatz dazu ist das Ziel der hier vorgestellten Constraints, Teile des physikalischen und des durch die Funktion eines Gegenstands bestimmten Verhaltens zu imitieren, um damit dem Benutzer ein möglichst einfaches und intuitives Bearbeiten einer Szene zu ermöglichen.

Um dieses Ziel zu erreichen, wurden im IConS-System folgende Constraints implementiert:

- *Flächenbindungen*: Dieser Constrainttyp verwendet Informationen über die Funktion eines Gegenstands und seines Objektmodells, um Beziehungen zwischen den verschiedenen Objektmodellen in einer Szene herzustellen. Dies ermöglicht eine exakte Ausrichtung der Objektmodelle und die Gruppierung verschiedener Objektmodelle.
- *Kollisionsvermeidung*: Durch diesen Mechanismus wird verhindert, daß sich Objekte gegenseitig durchdringen.
- *Objektausrichtung*: Dieser Constrainttyp ermöglicht es, Objekte, die immer in einer festgelegten Weise orientiert sein sollen, präzise auszurichten.

Darüberhinaus stehen folgende Mechanismen zur Verfügung, die die Manipulation von Objekten in einer Szene vereinfachen sollen:

- *Pseudo-Gravitation*: Die Bewegung eines Objektmodells kann manchmal nicht durch eine Flächenbindung gesteuert werden, da die Flächenbindungen eng mit der Funktion des Gegenstands verknüpft sind. Durch die Pseudo-Gravitation kann dann das Herabfallen unter dem Einfluß der Gravitation simuliert werden.
- *Instantiierung*: Die Instantiierung erlaubt es, mehrere Instanzen eines Objekts in eine Szene einzubauen und durch die Manipulation des gemeinsamen Datenbestands alle Instanzen gleichzeitig zu verändern.

3.2 Flächenbindungen

Das Konzept der Flächenbindungen beruht auf den *object associations*, durch die eine Beziehung zwischen den einzelnen Objekten einer Szene hergestellt werden kann [BS95, Buk95]. IConS wendet diesen Mechanismus nicht auf komplette Objekte sondern nur auf einzelne Flächen der Objekte an. Es können mehrere Flächen pro Objekt als potentielle Bindungspartner definiert und mit einer feiner abgestuften Benennung versehen werden. Dadurch ist es möglich, feste Bindungen zwischen den beteiligten Objekten zu erstellen und die Bewegungsfreiheit der Objekte einzuschränken. Dies geschieht so, daß das Arbeiten mit dem System erleichtert wird und für den Benutzer durchschaubar bleibt.

3.2.1 Bindungs- und Angebotsflächen

Zu jedem Objekt können zweierlei Arten von Flächen definiert werden: Bindungs- und Angebotsflächen. Eine Bindungsfläche entspricht hierbei einer Stand- oder Auflagefläche eines Objekts. Einige Beispiele sind die Standfläche eines Tischs oder einer Stehlampe aber auch die Rückseite einer Wandtafel, die an der Wand aufliegt. Die Angebotsflächen sind das Gegenstück zu den Bindungsflächen. Durch sie werden diejenigen Teile von Objekten markiert, an denen andere Objekte an- oder aufgesetzt werden können. Dazu zählt etwa die Oberseite einer Tischplatte oder die Vorderseite einer Wand.

Zwischen den Flächen eines Objektmodells und den Angebots- und Bindungsflächen, die zu diesem Objektmodell definiert sind, besteht kein prinzipieller Zusammenhang. Abbildung 3.1 zeigt als Beispiel einen Tisch, eine Lampe und den Fußboden. Die Standfläche der Lampe ist auch die Bindungsfläche, die Oberseite des Tischs fällt mit der Angebotsfläche zusammen. Für die Standfläche des Tischs wurde aber nicht eine Bindungsfläche für jedes Tischbein sondern nur eine Bindungsfläche für den ganzen Tisch definiert. Die Oberseite des Fußbodens ist eine Angebotsfläche.

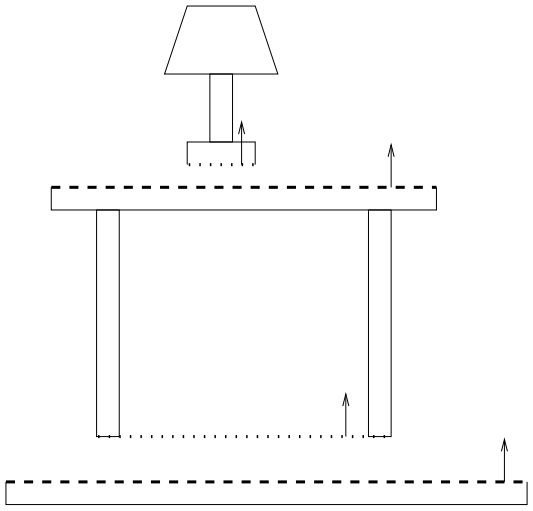

Abbildung 3.1: Tisch, Lampe und Fußboden mit Bindungsflächen (gepunktet) und Angebotsfläche (gestrichelt). Die Pfeile markieren die zu den Flächen definierten Normalen.

Soll nun die Lampe in Abbildung 3.1 unter Verwendung der Flächenbindungen auf den Tisch gestellt werden, so kann sie zunächst anhand der Flächennormalen so ausgerichtet werden, daß Bindungs- und Angebotsfläche parallel sind. Dann muß sie nur noch so verschoben werden, daß der Abstand zwischen den beiden Flächen verschwindet. Falls darüberhinaus sichergestellt wird, daß die Bindungsfläche ganz in der Angebotsfläche enthalten ist, kann ein ebener und physikalisch korrekter Kontakt zwischen zwei Objekten hergestellt werden, ohne daß eine physikalische Simulation verwendet werden muß.

Dieser Vorgang wird als die *Erfüllung* einer Flächenbindung bezeichnet. Durch eine erfüllte Flächenbindung wird eine feste Bindung zwischen den beteiligten Objekten erzeugt, die auch im weiteren Verlauf der Interaktion mit der Szene aufrechterhalten wird. Dies führt zu einer Einschränkung der Bewegungsfreiheit der beteiligten Objekte. Um die Erfüllung einer Flächenbindung rückgängig zu machen, muß sie explizit vom Benutzer *aufgebrochen* werden. Dieser Vorgang wird in Abschnitt 3.9.4 beschrieben.

3.2.2 Mehrere Bindungsflächen pro Objekt

Es gibt Fälle, in denen es sich anbietet, mehrere Bindungsflächen für ein Objekt zu definieren, die entweder alternativ oder gemeinsam erfüllt sein können.

Ein Spielwürfel kann zum Beispiel auf jeder der sechs Seitenflächen liegen, die damit auch alle als Bindungsflächen markiert sein sollten. Allerdings wird normalerweise immer nur eine der zugehörigen Flächenbindungen erfüllt sein. Andere Objektmodelle, wie etwa Schränke, können mehrere gleichzeitig erfüllte Flächenbindungen haben. Ein Schrank steht auf dem Boden und meist auch an der Wand.

3.2.3 Ausnutzung der Funktion von Gegenständen

Obwohl Objekte im allgemeinen aufgrund der Schwerkraft nach unten fallen, ist es nicht sinnvoll, dies in einem solchen System generell anzunehmen, denn eine Tafel hängt an der Wand und Lampen sind oft an der Decke befestigt. In den *object associations* wird darum jedem Objekt eine von drei möglichen Richtungen zugeordnet, in die es fallen kann: nach oben, nach unten oder in Richtung der nächsten Wand [BS95, Buk95].

Die meisten Objekte unterliegen aber im täglichen Leben stärkeren, über das rein physikalische Verhalten hinausgehenden Einschränkungen. Eine Tasse wird nicht auf den Boden gestellt, ein Tisch nicht auf einen Stuhl und eine Wandtafel hängt nicht vor einem Fenster. Diese Einschränkungen hängen mit der Funktion zusammen, die die Objekte für uns Menschen haben. Die Funktion eines Objekts bestimmt weitgehend die Art und Weise, wie wir mit diesem Objekt umgehen. Hierin liegt ein großes Potential zur Vereinfachung der Interaktion mit den Objektmodellen in einer Szene.

Um die Funktion eines Objekts bei der Interaktion mit einer Szene berücksichtigen zu können, wird jeder Bindungs- oder Angebotsfläche eine Bezeichnung zugeordnet, durch die die Funktion dieses Objekts möglichst exakt und vollständig erfaßt wird. Einige Beispiele sind *on-Floor* für die Standfläche von Tischen und Stühlen, *on-Wall* für die Rückseite einer Wandtafel oder *on-Table* für die Unterseite einer Tasse.

Dieses Vorgehen führt leicht zu einer schier unendlichen Anzahl von Bezeichnungen, die erzeugt und effizient verwaltet werden müssen. Das Problem wird noch dadurch verschärft, daß es für viele Objekte gar keinen eindeutigen Platz gibt. Ein Buch kann sowohl auf einem Tisch, einem Schränkchen oder dem Fernseher liegen. Die entsprechenden Bindungsflächen müßten mehrere Bezeichnungen haben.

Es bietet sich daher an, verallgemeinerte Bezeichnungen wie *on-Workspace*, *on-Store* einzuführen, die für ganze Klassen von Objekten bzw. Klassen von Flächen stehen. Zur weiteren Vereinfachung kann eine partielle Ordnung für diese Bezeichnungen erstellt werden, da zum Beispiel eine Arbeitsfläche (*on-Workspace*) meist auch als Ablagefläche (*on-Store*) verwendet wird. Als Beispiel dafür ist die hier verwirklichte partielle Ordnung $\leq_L$ in Abbildung 3.2 dargestellt.

Diese Ordnung enthält nur relative Bezeichnungen, die es ermöglichen, die Lage eines Objekts in Beziehung zur Lage eines anderen Objekts zu setzen. Zur Festlegung von absoluten Orientierungen von Objekten kann der Objektausrichtungs-Constraint verwendet werden, der in Abschnitt 3.4 vorgestellt wird.

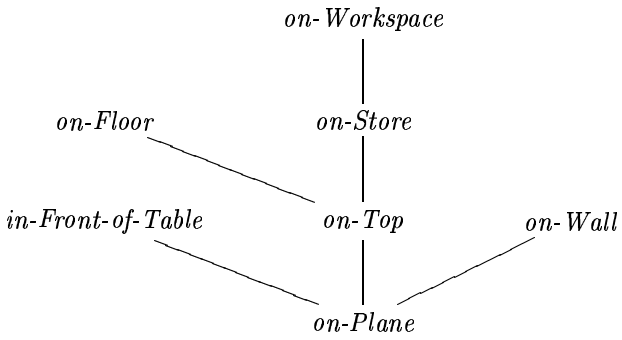

Abbildung 3.2: Hasse-Diagramm der partiellen Ordnung $\leq_L$ der im IConS-System implementierten Flächenbindungen. Für jede erfüllte Flächenbindung muß die Bezeichnung der Bindungsfläche $\leq_L$ der Bezeichnung der Angebotsfläche sein. Ein Objekt mit einer Bindungsfläche *on-Store* kann somit zum Beispiel auf eine Angebotsfläche mit der Bezeichnung *on-Store* oder *on-Workspace* gelegt werden. Die Bezeichnung *in-Front-of-Table* wurde eingeführt, um Stühle an einen Tisch stellen zu können (siehe auch Abschnitt 4.4.4).

3.2.4 Einschränkung der Bewegung durch erfüllte Flächenbindungen

Ein Objekt, das über keine erfüllten Bindungsflächen verfügt, kann beliebig bewegt werden – Translationen und Rotationen sind uneingeschränkt möglich. Jede erfüllte Flächenbindung verringert die Bewegungsfreiheit des Objekts, da vermieden werden muß, daß die Flächenbindung durch die Bewegung wieder aufgebrochen wird.

Hat ein Objekt genau eine erfüllte Bindungsfläche, so können Translationen nur noch parallel zu dieser Fläche erfolgen. Rotationen sind nur um eine Achse senkrecht zur Bindungsfläche möglich, um die Parallelität von Bindungs- und Angebotsfläche zu erhalten. Weiterhin wird die Bewegungsfreiheit noch durch die Forderung eingeschränkt, daß die Bindungsfläche vollständig in der Angebotsfläche enthalten sein soll.

Bei mehreren erfüllten Bindungsflächen pro Objekt kommt es für die Beweglichkeit auf die Anzahl der nicht-parallelen erfüllten Bindungsflächen an. Bei zwei nicht-parallelen Bindungsflächen ist eine Translation nur noch parallel zur Schnittgerade dieser Flächen möglich und eine Rotation bereits ausgeschlossen. Natürlich bleibt die Bedingung, daß alle Bindungsflächen ganz in den Angebotsflächen enthalten sein müssen, erhalten.

Durch drei nicht-parallele Bindungsflächen ist die Position eines Objekts eindeutig bestimmt, so daß weder Translationen noch Rotationen möglich sind.

Eine formale Beschreibung der eingeschränkten Bewegungsfreiheit durch erfüllte Flächenbindungen folgt in Abschnitt 3.9.3.

3.3 Kollisionsvermeidung

Neben den Flächenbindungen trägt die Kollisionsvermeidung wesentlich zur einfachen Bedienbarkeit des IConS-Systems bei. Dieser Constraint ist eine direkte Umsetzung der Tatsache, daß sich Objekte in der Realität nicht gegenseitig durchdringen können.

Hierfür muß bei jeder Veränderung der Position eines Objekts geprüft werden, ob das Objekt an der neuen Position mit anderen Objekten der Szene kollidiert. Dies bezieht sich nicht nur auf vom Benutzer eingegebene Transformationen, sondern auch auf vom System generierte, wie etwa beim Laden eines Objekts oder wenn ein Objekt zur Erfüllung einer Flächenbindung neu positioniert werden soll. Kommt es hierbei zu einer Kollision, so wird in den meisten Fällen die komplette Aktion rückgängig gemacht und das System auf den letzten kollisionsfreien Zustand zurückgesetzt. Nur in wenigen Fällen, wie beim Einfügen eines neuen Objekts in eine Szene, sucht

das System selbst einen Platz für das neue Objekt, an dem es mit keinem anderen kollidiert.

3.3.1 Kollision bei erfüllten Flächenbindungen

Ein Sonderfall liegt vor, wenn zwischen zwei Objekten eine erfüllte Flächenbindung besteht. In diesem Fall berühren sich die Bindungs- und die Angebotsfläche. Da diese in der Regel auch zum Objektmodell gehörende Flächen enthalten, kann es zu einer Berührung dieser Flächen und damit zu einer Kollision kommen.

In einem System, das mit Volumenmodellen arbeitet, kann dieses Problem umgangen werden, indem man nicht schon bei einer Berührung sondern erst beim Durchdringen der Objekte eine Kollision auslöst. Dies ist auch bei der Verwendung von Oberflächenmodellen möglich, wenn die einzelnen Flächenelemente eine geschlossene Oberfläche bilden und damit zwischen dem Inneren und dem Außenraum eines Objekts unterschieden werden kann. Da die frei erhältlichen Objekte diese Bedingung in der Regel nicht erfüllen, mußte eine andere Lösung für dieses Problem gefunden werden.

3.3.2 Umgehung des Problems

Um die Kollision von Objekten mit erfüllten Flächenbindungen zu vermeiden, wäre eine gesonderte Behandlung aller in den Bindungs- und Angebotsflächen enthaltenen realen Flächen nötig. Aber auch dann ist nicht gewährleistet, daß Flächen, die nur an der Bindungs- oder Angebotsfläche ansetzen, korrekt behandelt werden (siehe Abbildung 3.3).

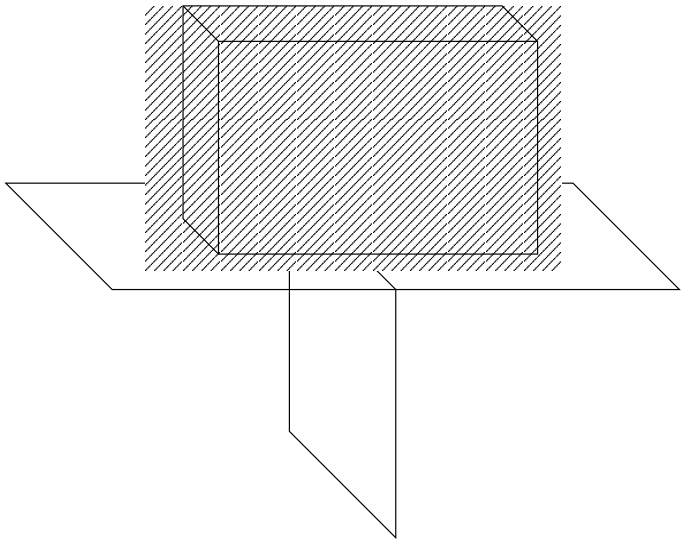

Abbildung 3.3: Problemfall für die Kollision: Der Block kollidiert nicht nur mit dem waagerechten Rechteck, auf dem er aufliegt, sondern auch mit dem senkrechten, da die Kante dieses Rechtecks den Block berührt.

Auf eine Sonderbehandlung dieser Flächen wurde verzichtet, da sie nicht nur eine Aufspaltung der Objektmodelle in einzelne Flächen erfordert hätte, sondern auch einen erheblichen Arbeitsaufwand verursachen würde. Stattdessen wird die Kollisionserkennung zwischen Objekten, die durch eine Flächenbindung verbunden sind, deaktiviert. Der Nachteil dieses Vorgehens ist, daß – besonders bei komplexeren Objekten – Kollisionen mit anderen Teilen der Objekte auftreten können (siehe Abbildung 3.4).

Abbildung 3.4: Kollisionen zwischen einem Schreibtisch und den darauf stehenden Objekten (links eine rote Flasche, rechts eine braune Lampe). Durch das Ausschalten der Kollisionserkennung zwischen Objekten mit erfüllten Flächenbindungen können die Kollisionen vom System nicht erkannt werden.

3.4 Objektausrichtung

Durch einen Objektausrichtungs-Constraint kann ein Objekt bezüglich des Weltkoordinatensystems und damit absolut ausgerichtet werden. Dies macht vor allem in zwei Fällen Sinn:

- Mit diesem Mechanismus kann man erzwingen, daß der Boden immer horizontal ist. Durch die Ausrichtung der Objekte bei der Erfüllung von Flächenbindungen wird diese Referenzausrichtung an alle anderen Objekte in der Szene „vererbt“. So kann eine Basis geschaffen werden, auf der alle anderen Operationen aufbauen können.

- Es gibt eine Reihe von Objekten, die zum Beispiel aus physikalischen oder ästhetischen Gründen immer in einer bestimmten Weise ausgerichtet sind. Eine Hängelampe, die an einem Kabel aufgehängt ist, zeigt aufgrund der Schwerkraft immer nach unten; ein Bild oder ein Spiegel wird normalerweise gerade aufgehängt. Während das erste Beispiel in den meisten Fällen durch eine *on-Ceiling*-Flächenbindung abgehandelt werden kann, ist dies im zweiten Beispiel nicht möglich. Durch die Einführung der Objektausrichtung wird dies erreicht.

Hierbei ist zu beachten, daß die Einführung einer absoluten Ausrichtung für einzelne Objekte innerhalb einer Szene nur dann sinnvoll ist, wenn auch die Szene selbst, zum Beispiel unter Verwendung eines immer horizontalen Bodens, absolut ausgerichtet ist.

3.4.1 Funktionsweise

Zur Umsetzung der Objektausrichtung werden zwei Vektoren benötigt. Der Vektor $\vec{n}$ definiert eine Richtung innerhalb des lokalen Koordinatensystems (LKS) des Objekts. Das LKS ist das Koordinatensystem, in dem das Objektmodell definiert ist. Es wird zusammen mit dem Objekt in der Szene verschoben und rotiert. Der zweite Vektor $\vec{w}$ legt eine absolute Richtung im Weltkoordinatensystem (WKS) fest. Das WKS ist das globale Koordinatensystem einer Szene, das nicht verändert werden kann und als Referenz dient. Abbildung 3.5 zeigt zwei Objekte innerhalb einer Szene mit den zugehörigen LKS und dem WKS.

Ein Objekt, das einen Objektausrichtungs-Constraint hat, wird vom System immer so ausgerichtet, daß $\vec{n}$ nach einer Transformation in das WKS in dieselbe Richtung zeigt wie der Vektor $\vec{w}$. Ist ein Objekt bereits korrekt orientiert, so gilt $\vec{n} = \vec{w}$ und man bräuchte nur einen Vektor angeben.

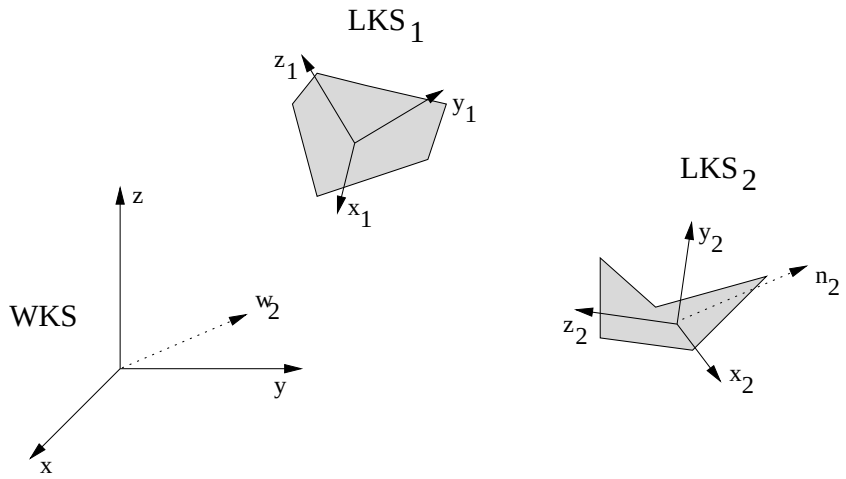

Abbildung 3.5: Lokale Koordinatensysteme (LKS) und Weltkoordinatensystem (WKS). Das WKS ist die Referenz in einer Szene, auf die sich alle Transformationen beziehen. Jedes Objekt innerhalb der Szene hat ein LKS, das zusammen mit dem Objekt transformiert wird. Objekt 2 ist durch einen Objektausrichtungs-Constraint gebunden, so daß $\vec{n_2}$ immer parallel zu $\vec{w_2}$ sein muß.

Durch die Vorgabe einer Ausrichtung für ein Objekt wird die Bewegungsfreiheit des Objekts eingeschränkt und es kann nur noch um eine Achse rotiert werden, die parallel zu $\vec{n}$ ist (siehe Abschnitt 3.9.3).

3.4.2 Vergleich mit Flächenbindungen

Wenn man die beiden Vektoren $\vec{n}$ und $\vec{w}$ als Normalen auf zwei Flächen auffaßt, dann entspricht ein Objektausrichtungs-Constraint der Bedingung, daß die beiden Flächen parallel sein müssen. Vergleicht man dies mit den Bedingungen für die Erfüllung einer Flächenbindung, so kann man erkennen, daß die Objektausrichtung dem Rotationsanteil einer erfüllten Flächenbindung entspricht. Dieser Zusammenhang wird auch im Algorithmus zur Berechnung der Einschränkung der Bewegungsfähigkeit eines Objekts durch Constraints in Abschnitt 3.9.3 ausgenutzt.

3.5 Weitere Hilfsmittel

3.5.1 Pseudo-Gravitation

Ein Objekt, das keine erfüllte Flächenbindung besitzt, kann auf diese Weise auch nicht an anderen Objekten ausgerichtet werden. Dieser Fall tritt besonders dann auf, wenn ein Objekt in einer Funktion verwendet werden soll, die bei der Erstellung des Objektmodells nicht vorgesehen wurde. Der Benutzer kann dann die Pseudo-Gravitation aktivieren.

Jedes Objekt, das durch eine Eingabe des Benutzers bewegt wird, fällt dann am Ende der Eingabe nach unten (d.h. in Richtung der negativen z-Achse des Systems), bis es auf einem anderen Objekt zu liegen kommt. Befindet sich kein Objekt darunter, so fällt das Objekt auch nicht nach unten.

3.5.2 Instantiierung von Objekten

Möchte man mehrere gleiche Objekte in eine Szene einfügen (etwa mehrere gleiche Stühle um einen Tisch stellen), so kann man dasselbe Objektmodell mehrmals in die Szene hereinladen. Man erhält dann voneinander völlig unabhängige Objekte, die zwar großteils dieselben Eigenschaften

(Geometrie, Farbe, ...) haben, aber sonst nichts mehr miteinander zu tun haben. Der Nachteil dieses Vorgehens ist, daß eventuelle Änderungen an den Objektdaten an jedem Objekt einzeln vorgenommen werden müssen. Dies kann recht aufwendig sein, wenn etwa alle Stühle in einem Gebäude durch ein anderes Modell ersetzt werden sollen.

Es ist aber auch möglich, beim ersten Laden eines Objekts zwar das komplette Objekt zu laden, dann aber für alle weiteren Objekte nur noch einen Verweis auf die entsprechenden Daten des ersten Objekts zu erstellen. Damit werden die Objektdaten nur einmal im Speicher gehalten. Eine Veränderung dieses Datensatzes führt automatisch auch zu einer Veränderung bei allen Objekten, die einen Verweis auf diese Daten enthalten.

Dieses Konzept führt zu einer Aufspaltung des Objektmodells in zwei Teile: die *Objektdaten* und die einzelnen *Instanzen*. Alle Daten, die für alle Instanzen gemeinsam gelten, werden in den Objektdaten abgelegt. Dazu gehören die grundlegenden Eigenschaften wie Geometrie, Farbe und Transparenz. Alle Daten, die sich von Instanz zu Instanz unterscheiden können, werden hingegen in den einzelnen Instanzen abgelegt. Dies sind im IConS-System vor allem die Position und Orientierung eines Objekts sowie alle Informationen über erfüllte Flächenbindungen.

Diese Form der Instantiierung von Objekten ähnelt den heute in objektorientierten Programmiersprachen üblichen Konzepten von Klassen und Instanzen, bei dem jede Instanz auch nur die für sie charakteristischen Daten enthält. Gemeinsame Eigenschaften, wie die Methoden, werden in der Klasse selbst zentral abgelegt und können damit auch zentral verändert werden.

Wenn im folgenden nun von einem Objekt gesprochen wird, so soll damit ein implizit in Objektdaten und -instanzen aufgespaltenes Objekt bezeichnet werden. Nur an Stellen, an denen eine Unterscheidung zwischen Objektdaten und -instanzen notwendig ist, wird diese im Text explizit gemacht.

3.6 Der Bindungsgraph

Eine einfache Möglichkeit zur Darstellung der zwischen den Objekten erfüllten Flächenbindungen ist der sogenannte Bindungsgraph. Seine Knoten sind die in der Szene enthaltenen Objekte. Eine erfüllte Flächenbindung wird durch eine gerichtete Kante vom anbietenden Objekt zum Objekt, dessen Flächenbindung durch das Angebot erfüllt wird, dargestellt. Abbildung 3.6 zeigt eine kleine Szene und Abbildung 3.7 den zugehörigen Bindungsgraphen.

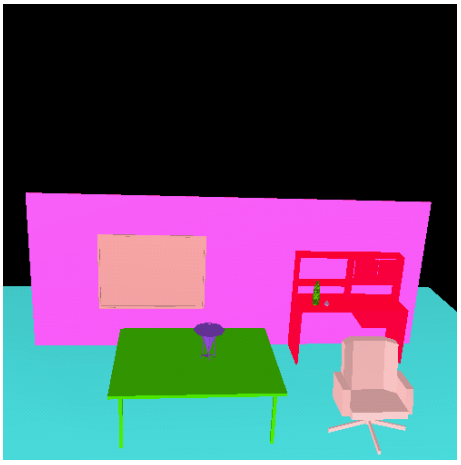

Abbildung 3.6: Bild einer Szene. Der zugehörige Bindungsgraph ist in Abbildung 3.7 dargestellt.

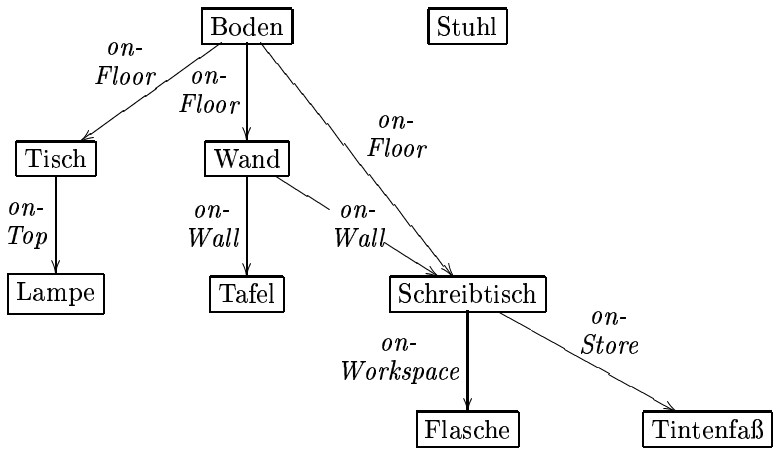

Abbildung 3.7: Bindungsgraph zu Abbildung 3.6. Der Typ der erfüllten Flächenbindungen ist in kursiver Schrift angegeben. Der Stuhl hat keine erfüllte Flächenbindung und ist daher auch nicht mit dem Rest des Graphen verbunden.

3.6.1 Flächenbindungen und dynamisches Gruppieren

Durch erfüllte Flächenbindungen wird eine dynamische Gruppierung erzeugt. Beim Bewegen eines Objekts werden alle Objekte, die mit diesem Objekt durch eine erfüllte Flächenbindung verbunden sind, so mitbewegt, daß sie sich relativ zueinander nicht bewegen. Sollten diese Objekte wieder mit weiteren Objekten verbunden sein, so werden auch sie mitbewegt. Die erzeugte Gruppierungsrelation ist damit eine transitive Relation.

Der Bindungsgraph enthält alle erfüllten Flächenbindungen. Bildet man die transitive Hülle des Bindungsgraphen, so erhält man die Gruppierungsrelation.

Die dynamische Gruppierung unterscheidet sich von der in vielen Grafikprogrammen enthaltenen Gruppierung dadurch, daß die gruppierten Objekte nicht starr miteinander verbunden sind. Bei der Bewegung eines Objekts mit erfüllten Flächenbindungen werden die anbietenden Objekte nicht mitbewegt. Ein Buch, das auf einem Tisch liegt, wird beim Bewegen des Tischs mitverschoben. Bewegt man aber das Buch, so bleibt der Tisch stehen. Die Gruppierungsrelation des IConS-Systems ist somit, im Gegensatz zu der meist in Grafikprogrammen enthaltenen Art der Gruppierung, nicht symmetrisch.

3.6.2 Spezialfälle für den Bindungsgraphen

Durch gerichtete Kreise oder ungeordnete Vorgängerknoten im Bindungsgraphen kann es zu unerwarteten und unerwünschten Effekten bei der Bedienung des Systems kommen. Eine gesonderte Behandlung dieser Fälle ist deshalb zur Steigerung der Bedienungsfreundlichkeit des Systems notwendig.

Gerichtete Kreise

Durch einen gerichteten Kreis im Bindungsgraphen kommt es, wie man leicht durch Bildung der transitiven Hülle des Bindungsgraphen erkennen kann, zu einer vollständigen Gruppierung aller Objekte, die in diesem Kreis enthalten sind. Bei der Bewegung eines Objekts aus der Gruppe, die durch diesen Kreis gebildet wird, bewegen sich alle anderen Objekte der Gruppe mit.

Allerdings sollte die Bewegungsfreiheit der ganzen Gruppe nicht durch die Lage der innerhalb der Gruppe erfüllten Flächenbindungen beeinflußt werden, da durch erfüllte Flächenbindungen nur die Bewegung von Objekten relativ zueinander eingeschränkt wird. Dies erfordert eine gesonderte Behandlung der so gruppierten Objekte.

IConS unterstützt diese Art der Gruppierung nicht, sondern erlaubt die Erfüllung von Flächenbindungen nur dann, wenn dadurch keine gerichteten Kreise im Bindungsgraphen entstehen. Dies gilt auch für den Sonderfall, in dem Angebots- und Bindungsfläche einer zu erfüllenden Flächenbindung zum selben Objekt gehören.

Ungeordnete Vorgängerknoten

Abbildung 3.8 zeigt eine Szene mit vier Objekten: Ein Tisch und eine Wand stehen auf dem Fußboden. Das L-Stück ist durch je eine erfüllte Flächenbindung mit der Wand und dem Tisch verbunden. Der Bindungsgraph zu dieser Szene ist in Abbildung 3.9 dargestellt. Das besondere an dieser Szene ist, daß zwischen den beiden Vorgängern des L-Stücks (Wand und Tisch) keine Ordnung existiert. Trotzdem besteht durch das L-Stück als gemeinsamem Nachfolger beider Objekte ein Zusammenhang zwischen der Bewegungsfreiheit von Wand und Tisch.

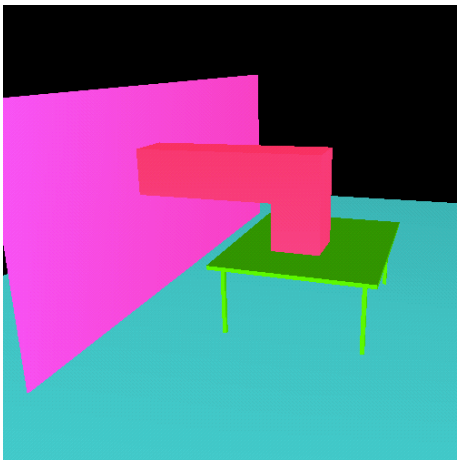

Abbildung 3.8: Szene mit ungeordneten Vorgängerknoten. Die Wand und der Tisch stehen auf dem Fußboden. Das L-Stück ist sowohl mit der Wand als auch mit dem Tisch durch eine erfüllte Flächenbindung verbunden (siehe dazu auch den zugehörigen Bindungsgraphen in Abbildung 3.9). Zwischen der Wand und dem Tisch besteht keine Ordnung und damit kommt es zu den im Text beschriebenen Problemen.

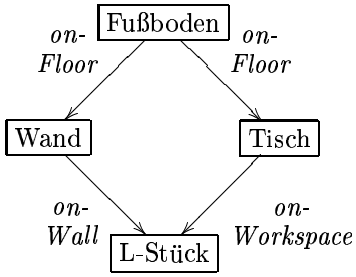

Abbildung 3.9: Bindungsgraph zu Abbildung 3.8. Zwischen den beiden Vorgängern des L-Stücks besteht keine Ordnung.

Wird zum Beispiel der Tisch verschoben, so bewegt sich auch das L-Stück. Dabei muß gewährleistet sein, daß die Flächenbindung zwischen L-Stück und Wand erhalten bleibt. Dies kann wie folgt geschehen: Entweder wird die Wand nicht mitverschoben und schränkt damit die Bewegungsfreiheit des Tischs auf eine Verschiebung parallel zur Wand ein. Alternativ dazu kann aber auch die Gruppierung zwischen L-Stück und Wand während dieser Bewegung invertiert werden, so daß sich die Wand zusammen mit dem Tisch bewegt.

Beide Möglichkeiten führen implizit eine Ordnung zwischen den Vorgängerknoten ein und sind zulässige Lösungen. Allerdings ist im konkreten Einzelfall meist eine der beiden Lösungen besser geeignet, um eine bestimmte Aufgabe zu erfüllen. Eine generelle Festlegung auf eine der Lösungsstrategien kann sich somit als problematisch für die Bedienung erweisen und es kann nötig werden, eine erfüllte Flächenbindung aufzubrechen, um die Objekte in einer bestimmten Weise anzuordnen. Da sich der entstehende Mehraufwand für den Benutzer aber in vertretbaren Grenzen hält, wurde im IConS-System die erste Variante fest eingebaut, die sich natürlich aus den sonstigen Eigenschaften der Gruppierung ergibt.

3.7 Der IConS-Szenengraph

In den vorhergehenden Abschnitten wurden zwei dieser Arbeit zugrundeliegende Ideen zur Darstellung und Speicherung von Objekten erläutert. Dies sind die Aufspaltung der Objekte in Objektdaten und -instanzen sowie die dynamische Gruppierung von Objekten, die sich aus den erfüllten Flächenbindungen ergibt. Diese beiden Konzepte spiegeln sich auch im Aufbau der internen Datenstrukturen zum Speichern einer Szene wieder.

IConS speichert alle Daten einer Szene in einem *Szenengraphen*. Ein solcher Szenengraph wird in den meisten Programmpaketen zum Arbeiten mit einer Szene bereitgestellt, wobei es konzeptuelle Unterschiede im Aufbau des Szenengraphen gibt. Das hier vorgestellte Konzept orientiert sich an dem in OpenGL Optimizer 1.1 verwendeten Szenengraph [Eck97] (siehe auch Abschnitt 4.1.1).

Ein Szenengraph besteht aus einer Reihe verschiedener Knotentypen, von denen jede eine bestimmte Funktion erfüllt. Ein *Objektknoten* enthält die Beschreibung eines Objekts, ein *Transformationsknoten* wendet eine Transformation auf seine Nachfolgerknoten an und durch einen *Gruppierungsknoten* können mehrere Teilgraphen zu einem einzigen Szenengraphen zusammengefaßt werden. Abbildung 3.10 zeigt einen Beispielszenengraphen, auf dessen Struktur weiter unten noch eingegangen wird.

Die Unterstützung der Aufspaltung in Objektdaten und -instanzen wird durch eine Abbildung auf Knotentypen erreicht. Die dynamische Gruppierung ergibt sich aus der speziellen Struktur des Szenengraphen. In den beiden folgenden Abschnitten wird dies näher erläutert.

3.7.1 Objektdaten und -instanzen

Wie bereits in Abschnitt 3.5.2 dargestellt, erfolgt durch die Trennung in Objektdaten und Objektinstanz auch eine Trennung der Daten aufgrund ihres Geltungsbereichs. Diejenigen Daten, die für alle Instanzen gleich sind, werden in den Objektdaten gehalten. Dies sind Geometrie und Aussehen der Objekte sowie die zu diesen Objekten gehörenden Bindungs- und Angebotsflächen. Dagegen werden Daten, die für jede Instanz unterschiedlich sind, bei den Instanzen gehalten. Dies sind die Position und Orientierung einer Instanz und die für diese Instanz erfüllten Flächenbindungen.

Aufgrund dieser Überlegungen bietet es sich an, die Objektdaten durch einen Objektknoten zu repräsentieren, der bereits die Daten über Geometrie und Aussehen eines Objekts enthält. Die einzelnen Objektinstanzen können dann durch Transformationsknoten dargestellt werden, wodurch jede Instanz einzeln positioniert werden kann.

Alle Knoten einer Szene werden durch einen Gruppierungsknoten in einem einzigen Szenengraph zusammengefaßt. Ein solcher Szenengraph ist in Abbildung 3.10 dargestellt. Abbildung 3.11 zeigt, wie dieser Szenengraph zu interpretieren ist.

3.7.2 Dynamische Gruppierung

Bei der Gruppierungsrelation handelt es sich um eine asymmetrische Relation. Ein auf einem Tisch stehendes Glas sollte zusammen mit dem Tisch verschoben werden, aber der Tisch soll sich nicht bewegen, wenn das Glas verschoben wird. Damit werden aber gruppierte Objekte nicht wie in vielen Anwendungen zu einem großen Objekt zusammengefaßt sondern in eine Gruppierungshierarchie eingeordnet.

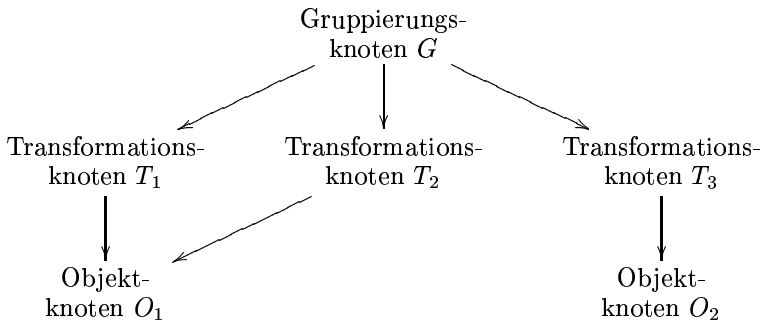

Abbildung 3.10: Ein Szenengraph. Abbildung 3.11 erläutert die Beziehung zwischen dem Szenengraph und der Umsetzung des Objektkonzepts.

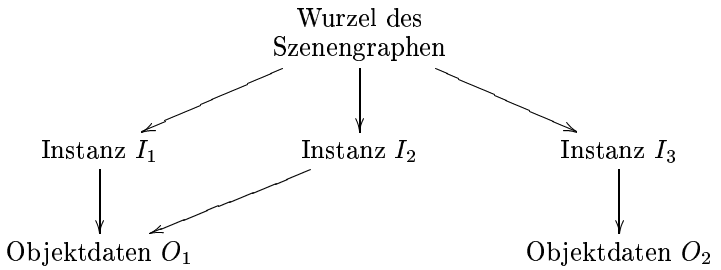

Abbildung 3.11: Interpretation des Szenengraphen aus Abbildung 3.10. Der Gruppierungsknoten G ist die Wurzel des Szenengraphen. Die Objektdaten sind in den Objektknoten abgelegt und ihre Instanzen werden durch die Transformationsknoten repräsentiert.

Diese Hierarchie läßt sich direkt durch die Struktur des Szenengraphen umsetzen, solange jede Instanz nur eine erfüllte Flächenbindung besitzt und damit auch nur zu einer Instanz gruppiert wird. Die Verallgemeinerung auf die Gruppierung zu mehreren anderen Instanzen ist im allgemeinen nur durch eine dynamische Umstrukturierung des Szenengraphen möglich. Auf diese beiden Fälle wird in den zwei folgenden Abschnitten eingegangen.

Objekte mit einem eindeutigen Vorgänger

Wenn jede Objektinstanz nur über eine erfüllte Flächenbindung verfügt, kann sie eindeutig zu der Objektinstanz gruppiert werden, die die Flächenbindung erfüllt.

Im IConS-System wird eine Objektinstanz I_B genau dann zu einer Objektinstanz I_A gruppiert, wenn eine Bindungsfläche von I_B durch ein Angebot von I_A erfüllt wird. Als Effekt dieser Gruppierung soll bei einer Bewegung von I_A die Instanz I_B so mitbewegt werden, daß sich die Position der beiden Instanzen relativ zueinander nicht verändert. Andererseits soll eine Bewegung von I_B nicht zu einer Bewegung von I_A führen.

Diese Eigenschaft ist automatisch garantiert, wenn I_B im Szenengraph ein Nachfolgerknoten von I_A ist. Da Objektinstanzen durch Transformationsknoten repräsentiert werden, die eine beliebige Zahl von Nachfolgerknoten haben können, ist eine Umsetzung einfach. Wenn eine Flächenbindung einer Instanz erfüllt wird, so wird diese Instanz im Szenengraphen zu einem Nachfolgerknoten der Instanz mit dem korrespondierenden Bindungsangebot gemacht. Dieser Vorgang ist in Abbildung 3.12 dargestellt. Aufgrund des Zusammenhangs zwischen der Erfüllung von Flächenbindungen und der Gruppierung hat der hierbei entstehende Szenengraph dieselbe Struktur wie der Bindungsgraph der Szene.

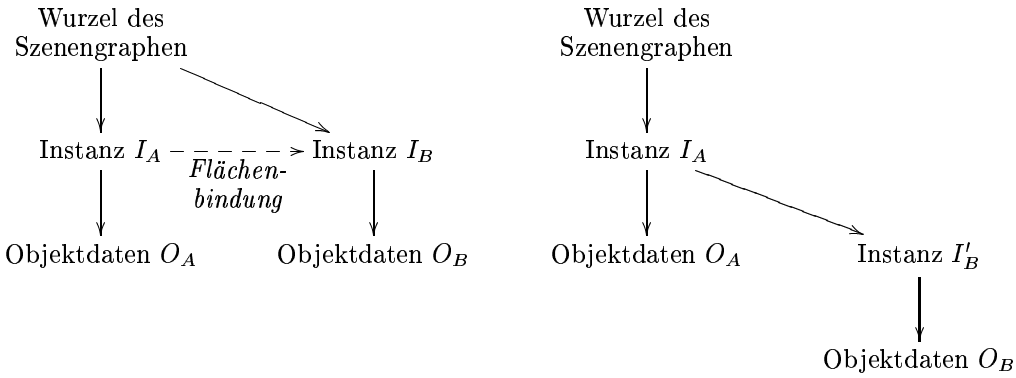

Abbildung 3.12: Umbau des Szenengraphs bei der Erfüllung einer Flächenbindung. I_B wird zum Nachfolger von I_A gemacht und dadurch eine Gruppierung erreicht. Der Übergang von I_B zu I'_B deutet an, daß sich die Transformation der Instanz verändert (siehe Abschnitt 3.7.3)

Objekte mit mehreren Vorgängern

Leider ist es nicht möglich, dieses Vorgehen direkt auf den Fall zu übertragen, in dem mehrere Flächenbindungen einer Instanz erfüllt sind. Durch die Einführung mehrerer Vorgänger würde die betroffene Instanz mehrfach in die Szene eingefügt.

Um dieses Problem zu umgehen, wurde von einer starren Struktur des Szenengraphen abgegangen. Stattdessen wird der Szenengraph vor jedem Verschieben einer Objektinstanz I umstrukturiert. Dazu werden alle Instanzen, die zu I gruppiert sind, so im Szenengraph verschoben, daß sie direkte Nachfolgerknoten von I sind. Aufgrund der Transitivität der Gruppierungsrelation muß dies auch mit allen Instanzen, die über eine oder mehrere andere Instanzen zu I gruppiert sind, gemacht werden. Als Ergebnis sind dann alle Instanzen, die sich mit I mitbewegen sollen, direkte Nachfolgerknoten von I.

Durch dieses Vorgehen existiert kein direkter Zusammenhang mehr zwischen dem Bindungsgraphen und dem Szenengraphen einer Szene.

3.7.3 Umbau des Szenengraphen

Durch das Verschieben innerhalb des Szenengraphen verändert sich auch die Position der verschobenen Instanz innerhalb der Szene. Deshalb muß die im Transformationsknoten der Instanz gespeicherte Transformation entsprechend geändert werden. Dies ist in Abbildung 3.12 durch den Übergang von I_B zu I'_B angedeutet.

Sei T_A die Transformation vom Knoten I_A bis zur Wurzel des Graphen und T_B die vor der Gruppierung gültige Transformation von I_B bis zur Wurzel. Dann ist die neue Transformation T'_B des Knotens I'_B

$$T'_B = T_A^{-1} \cdot T_B$$

Für die Transformation eines Vektors $\vec{x}$ aus dem LKS von I_B in das WKS gilt vor Umordnung des Szenengraphen:

$$\vec{X} = T_B \cdot \vec{x}$$

Danach gilt für denselben Vektor:

$$\vec{X}' = T_A \cdot T'_B \cdot \vec{x}$$

Damit kann gezeigt werden, daß sich die Position des Vektors im WKS durch die Umordnung des Szenengraphen nicht verändert:

$$\vec{X} = T_B \cdot \vec{x} = T_A \cdot T_A^{-1} \cdot T_B \cdot \vec{x} = T_A \cdot T'_B \cdot \vec{x} = \vec{X}'$$

3.8 Formalisierung

Eine Szene S besteht aus einer Menge von Objekten: $S = \{O_1, O_2, \ldots\}$. Jedes Objekt ist ein 7-Tupel $O_i = (G, T, R, A, B, N, W)$. Es enthält die Geometrie $G = \{D_1, D_2, \ldots\}$ als eine Menge von Dreiecken, eine Translation T und eine Rotation R, die die aktuelle Position und Ausrichtung beinhalten, eine Menge von Angebotsflächen $A = \{A_1, A_2, \ldots\}$ und eine Menge von Bindungsflächen $B = \{B_1, B_2, \ldots\}$ sowie eine Objektnormale N und eine Normale W im Weltkoordinatensystem für den Objektausrichtungs-Constraint.

Eine Angebotsfläche $A_i = (P, L)$ besteht aus einem ebenen Polygon im Raum P und einer Bezeichnung L. Eine Bindungsfläche $B_i = (P, L, E)$ enthält zusätzlich einen Verweis auf eine Angebotsfläche E, falls eine erfüllte Flächenbindung zwischen B_i und E besteht.

Zur Verbesserung der Lesbarkeit werden die einzelnen Elemente in Großbuchstaben auch als Selektorfunktion verwendet. Ein Element in Kleinbuchstabe markiert eine konkrete Instanz eines Elements. Damit drückt zum Beispiel

$$\mathit{gebunden}(a, b) \quad \Leftrightarrow \quad E(b) = a$$

aus, daß das Prädikat *gebunden* mit einer Angebotsfläche a und einer Bindungsfläche $b = (p, l, e)$ als Argument genau dann wahr ist, wenn $e = a$ gilt.

3.8.1 Einzelne Constraints

Flächenbindungen

Die Flächenbindungen zwischen zwei Objekten sind genau dann zulässig, wenn für jedes Angebots–Bindungs-Paar gilt, daß entweder keine erfüllte Flächenbindung besteht oder daß mit einer Flächenbindung auch alle für eine erfüllte Flächenbindung notwendigen Bedingungen erfüllt sind:

$$\begin{aligned} \mathit{bindok}(o, o') \quad \Leftrightarrow \quad & (\forall a \in A(o), b \in B(o') : \neg \mathit{gebunden}(a, b) \vee \mathit{erfüllt}(a, b)) \wedge \\ & (\forall a \in A(o'), b \in B(o) : \neg \mathit{gebunden}(a, b) \vee \mathit{erfüllt}(a, b)) \end{aligned}$$

Dabei drückt $\mathit{gebunden}(a, b)$ aus, daß die Flächenbindung zwischen der Angebotsfläche a und der Bindungsfläche b erfüllt ist:

$$\mathit{gebunden}(a, b) \quad \Leftrightarrow \quad E(b) = a$$

Das Prädikat $\mathit{erfüllt}(a, b)$ enthält alle notwendigen Bedingungen für eine erfüllte Flächenbindung zwischen einer Angebotsfläche a und einer Bindungsfläche b:

$$\begin{aligned} \mathit{erfüllt}(a, b) \quad \Leftrightarrow \quad & \mathit{parallel}(P(a), P(b)) \wedge \mathit{distanz}(P(a), P(b)) = 0 \wedge \mathit{enthält}(P(a), P(b)) \wedge \\ & \neg \mathit{vorgänger}(\mathit{objekt}(b), \mathit{objekt}(a)) \wedge \\ & L(b) \leq_L L(a) \end{aligned}$$

Die erste Zeile legt die Position der Polygone der Angebots- und Bindungsfläche fest: $\mathit{parallel}(p, p')$ gilt, wenn die Normalen auf den beiden Polygonen identisch sind. Dadurch sind die Polygone nicht nur parallel sondern haben auch dieselbe Orientierung. $\mathit{distanz}(p, p')$ ist nur für parallele Polygone definiert und gibt den Abstand der beiden durch die Polygone definierten Ebenen an. Wenn beide Prädikate wahr sind, liegen die beiden Polygone in einer gemeinsamen Ebene im Raum. $\mathit{enthält}(p, p')$ ist dann wahr, wenn p in p' enthalten ist.

Zeile zwei bezieht sich auf die Struktur des Bindungsgraphen: Dabei gibt $\mathit{objekt}(a)$ bzw. $\mathit{objekt}(b)$ das Objekt zurück, das zu einer Angebots- oder Bindungsfläche gehört. $\mathit{vorgänger}(o, o')$ ist erfüllt, wenn $o = o'$ gilt oder wenn o im Bindungsgraphen ein Vorgänger von o' ist.

Die letzte Zeile legt fest, daß zwischen den Bezeichnungen von Angebots- und Bindungsfläche die in Abschnitt 3.2.3 festgelegten Beziehungen gelten müssen.

Kollisionsvermeidung

Das Prädikat $nokoll(o, o')$ ist genau dann wahr, wenn zwischen zwei Objekten keine unerlaubte Kollision existiert:

$$\begin{aligned} nokoll(o, o') \quad \Leftrightarrow \quad & o = o' \vee \neg kollision(o, o') \vee \\ & \exists a \in A(o), b \in B(o') : (gebunden(a, b) \vee wargebunden(a, b)) \vee \\ & \exists a \in A(o'), b \in B(o) : (gebunden(a, b) \vee wargebunden(a, b)) \end{aligned}$$

Eine Kollision zwischen zwei Objekten liegt genau dann vor, wenn mindestens ein Dreieck des einen Objekts ein Dreieck des anderen Objekts schneidet. Damit gilt:

$$kollision(o, o') \quad \Leftrightarrow \quad \exists d \in G(o), d' \in G(o') : d \text{ und } d' \text{ schneiden sich}$$

$wargebunden(a, b)$ steht für den in Abschnitt 3.3.2 beschriebenen Sonderfall, daß zwei Objekte nach dem Bruch einer Flächenbindung solange kollidieren dürfen, bis sie einmal komplett voneinander getrennt waren.

Objektausrichtung

Ein Objekt ist dann bezüglich der Objektausrichtung korrekt ausgerichtet, wenn die Objektnormale N nach einer Transformation in das Weltkoordinatensystem parallel zur Normalen W ist oder wenn kein solcher Constraint für das Objekt definiert ist:

$$ausgerichtet(o) \quad \Leftrightarrow \quad N(o) = \vec{0} \vee welt(N(o)) = W(o)$$

Die Funktion $welt(n)$ transformiert dabei einen Vektor vom lokalen Koordinatensystem eines Objekts in das Weltkoordinatensystem.

3.8.2 Konsistenz einer Szene

Mit den oben definierten Prädikaten kann man nun das Prädikat $konsistent(s)$ für eine Szene definieren. Eine Szene ist konsistent, wenn sich alle Objekte einer Szene in einem Zustand befinden, der durch die drei definierten Constraints erlaubt ist:

$$konsistent(s) \quad \Leftrightarrow \quad \forall o \in s, o' \in s : (bindok(o, o') \wedge nokoll(o, o') \wedge ausgerichtet(o))$$

Beim Start von IConS befinden sich keine Objekte in der Szene. Damit ist die Szene automatisch konsistent. Wenn nachgewiesen werden kann, daß die Szene auch bei jedem weiteren Bearbeitungsschritt konsistent bleibt, so ist damit auch gezeigt, daß jede von IConS erzeugte Szene immer konsistent ist.

Beim bloßen Hinzufügen eines Objekts durch Laden oder Instantiierung genügt es, die Objekte entsprechend der Objektausrichtung zu orientieren und sie an eine Position zu setzen, an der sie mit keinem anderen Objekt in der Szene kollidieren. Auch das Entfernen eines Objekts aus der Szene erfordert nur ein Löschen der von ihm erfüllten Flächenbindungen. Dahingegen ist der Nachweis, daß auch beim Erfüllen oder Aufbrechen einer Flächenbindung und beim Bewegen eines Objekts die Konsistenz erhalten bleibt, nicht trivial zu erbringen. Auf diese Punkte wird in den Abschnitten 3.9.2 bis 3.9.4 eingegangen.

3.9 Interaktionsmodell

Ein Interaktionsmodell beschreibt, auf welche Weise ein Benutzer mit einem System umgehen kann und wie das System auf die Benutzereingaben reagiert. Dieser Abschnitt soll nicht nur einen Überblick über das IConS-System geben, sondern auch wichtige Details beleuchten.

3.9.1 Überblick über das Gesamtsystem

Das IConS-System befindet sich zu jedem Zeitpunkt in einem von drei *Modi*: *eingeschränkte Interaktion*, *freie Interaktion* oder *Navigation*.

Das Bearbeiten von Szenen findet meist im eingeschränkten Interaktionmodus statt, in dem alle Bindungsmechanismen aktiviert sind. Im freien Interaktionsmodus sind dagegen die Flächenbindungen deaktiviert. Für jedes Objekt, das in diesem Modus bewegt wird, werden alle erfüllten Flächenbindungen aufgebrochen und neue Flächenbindungen können nicht erfüllt werden. Dieser Modus wird hauptsächlich dazu verwendet, Objekte von einer Position mit erfüllten Flächenbindungen an eine andere Position zu verschieben. So kann zum Beispiel ein Regal von einer Wand an eine andere verschoben werden. Im Navigationsmodus kann sich der Benutzer durch die Szene bewegen, um sie von verschiedenen Positionen aus betrachten und bearbeiten zu können.

Die Auswahl dieser Modi erfolgt durch Drücken der zugehörigen Taste auf der Tastatur. Außerdem werden einige weitere Aufgaben wie das Laden oder Löschen von Objekten über Tastatureingaben gesteuert.

Eingeschränkter Interaktionsmodus

In diesem Modus können vom Benutzer drei verschiedene Transformationen eingegeben werden: Translationen parallel zur Bildschirmebene, Translationen senkrecht zur Bildschirmebene und Rotationen. Der Benutzer wählt durch Anklicken mit der Maus ein Objekt aus, erzeugt eine Reihe einzelner Transformationen durch Bewegen der Maus und beendet die Eingabe durch Loslassen der Maustaste. Jede der drei Operationen kann durch das Drücken einer der drei Maustasten ausgewählt werden. Auf die Eingabe von Transformationen wird in Abschnitt 3.10 eingegangen.

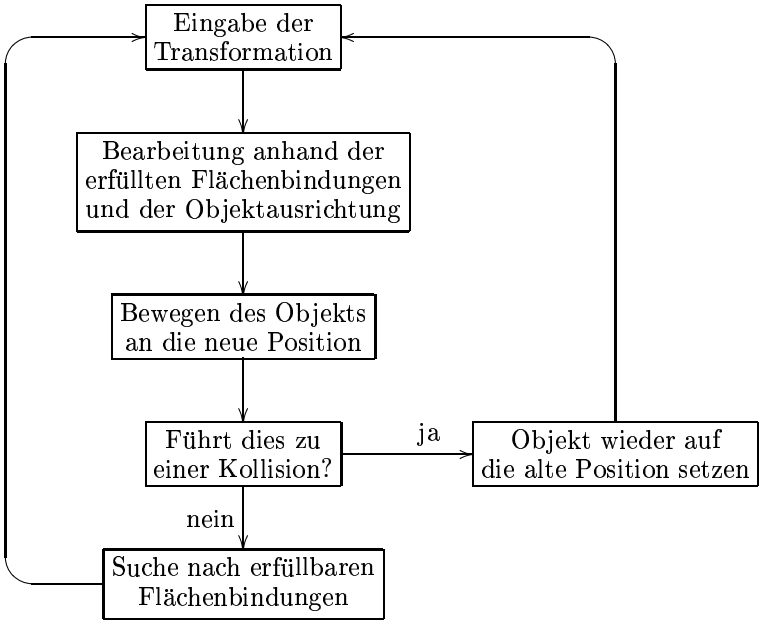

Abbildung 3.13: Schritte bei der Verarbeitung einer Transformation im eingeschränkten Interaktionsmodus.

Abbildung 3.13 zeigt, in welchen Schritten eine einzelne Transformation vom IConS-System verarbeitet wird. Im ersten Schritt wird die Transformation anhand der erfüllten Flächenbindungen des Objekts bearbeitet (siehe Abschnitt 3.9.3). Dann wird das Objekt probehalber an die neue Position gesetzt und getestet, ob es dort mit einem anderen Teil der Szene kollidiert. Ist dies der Fall, so wird das Objekt an die Ausgangsposition zurückgesetzt. Die Verarbeitung dieser Transformation ist damit zu Ende.

Nur wenn das Objekt an der neuen Position nicht mit einem anderen kollidiert, wird diese Position als gültig gespeichert. Dann wird versucht, eventuell noch nicht erfüllte Flächenbindungen zu erfüllen. Dieser Vorgang wird in Abschnitt 3.9.2 beschrieben.

Freier Interaktionsmodus

Das Arbeiten in diesem Modus unterscheidet sich vom Arbeiten im eingeschränkten Interaktionsmodus in folgenden Punkten: Beim Anklicken eines Objekts werden alle erfüllten Flächenbindungen dieses Objekts aufgebrochen, so daß es wieder frei bewegt werden kann (siehe Abschnitt 3.9.4). Da es keine erfüllten Flächenbindungen für dieses Objekt mehr gibt, brauchen die Transformationen nicht entsprechend bearbeitet werden und müssen nur noch an die Objektausrichtung angepaßt werden. Zudem wird nicht versucht, unerfüllte Flächenbindungen zu erfüllen.

Navigationsmodus

Im Navigationsmodus wird die Szene relativ zum Betrachter verschoben oder rotiert. Um eine möglichst einheitliche Benutzeroberfläche zu schaffen, werden dieselben Techniken für die Eingabe von Translationen und Rotationen wie für die Interaktionsmodi verwendet. Allerdings beziehen sich die Transformationen nicht auf ein einzelnes Objekt sondern immer auf die Kamera.

3.9.2 Suche nach erfüllbaren Flächenbindungen

An zentraler Stelle steht der Algorithmus, der für ein einzelnes Objekt mit einem oder mehreren unerfüllten Flächenbindungen nach korrespondierenden Angebotsflächen sucht. Dabei darf nur das Objekt und die mit ihm durch erfüllte Flächenbindungen gruppierten Objekte bewegt werden. Es handelt sich damit um einen lokalen Algorithmus und nicht, wie etwa in deklarativen Modellierungs-Systemen üblich, um ein globales Verfahren, bei dem alle Objekte in einer Szene bewegt werden dürfen.

Eine Angebotsfläche kann zur Erfüllung einer Flächenbindung verwendet werden, wenn die Bezeichnungen der Flächen zusammenpassen, dadurch kein entarteter Bindungsgraph entsteht (siehe Abschnitt 3.6.2) und wenn die Geometrie der beiden Flächen es zuläßt. Allerdings würde es die Bedienung eines interaktiven Systems erheblich erschweren, wenn Objekte durch die Erfüllung einer Flächenbindung „unkontrollierbar“ durch die Szene verschoben werden, ohne daß ein direkter Zusammenhang zwischen der Ausgangsposition und der Endposition zu erkennen ist.

Deshalb kann eine Flächenbindung nur dann erfüllt werden, wenn, nachdem die Bindungsfläche parallel zur Angebotsfläche ausgerichtet wurde, die Projektion der Bindungsfläche auf die Ebene der Angebotsfläche vollständig in der Angebotsfläche enthalten ist. Anschaulich bedeutet dies, daß sich die Bindungsfläche nach der Ausrichtung in einem unendlichen, senkrechten Prisma befinden muß, das sich, wie in Abbildung 3.14 dargestellt, von beiden Seiten der Angebotsfläche aus in den Raum erstreckt.

Der folgende Algorithmus findet alle Angebotsflächen, die auf diese Weise im Zusammenhang mit einer unerfüllten Bindungsfläche eines Objekts stehen:

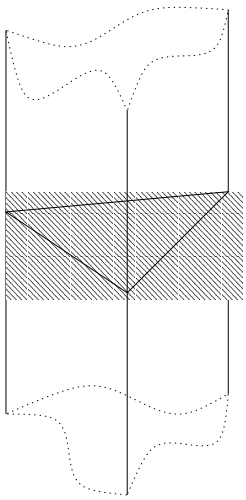

Abbildung 3.14: Das unendliche Prisma einer Angebotsfläche, in dem sich die Bindungsflächen befinden müssen.

Algorithmus 1 Suche nach Angeboten für nichterfüllte Bindungsflächen von Objekt o

sucheBindungen(o, s)

```
1:  angebote ⇐ ∅
2:  for all b ∈ B(o) do
3:    if E(b) = NULL then
4:      for all o' ∈ O(s) do
5:        if ¬vorgänger(o, o') then
6:          for all a ∈ A(o') do
7:            if L(a) ≥ L(b) then
8:              r = erzeugeRotation(welt(normale(P(b))), welt(normale(P(a))))
9:              wende r auf o an
10:             if enthält(P(a), P(b)) then
11:               t = erzeugeTranslation(P(b), P(a))
12:               angebote ⇐ angebote ∪ (o', a, b, t, r)
13:             wende −r auf o an
14: return angebote
```

Die Funktion *erzeugeRotation*(n, n') berechnet dabei eine Rotation, die die Richtung des ersten Arguments in die des zweiten überführt. Dazu wird eine spezielle Routine verwendet, die in Abschnitt 4.6.1 beschrieben ist. *normale*(p) berechnet eine Normale auf einem Polygon p.

Für alle Elemente aus *angebote* gilt damit, daß nach Anwendung der Transformationen t und r auf o das Prädikat *erfüllt*(a, b) wahr ist: Angebots- und Bindungsflächen sind parallel, haben den Abstand Null und die Bindungsfläche ist in der Angebotsfläche enthalten. Ferner stimmen auch die Struktur des Bindungsgraphen und die Bezeichnungen der beiden Flächen.

Vor der Erfüllung der Flächenbindung bleibt damit nur noch zu prüfen, ob die Transformationen aufgrund schon erfüllter Flächenbindungen und der Objektausrichtung überhaupt auf o angewendet werden dürfen. Außerdem darf es durch diese Anwendung zu keiner Kollision kommen. Dies wird von der folgenden Routine erledigt:

Algorithmus 2 Erfüllung einer Flächenbindung von Objekt o

erfülleBindung(o, s)

1: $angebote \Leftarrow sucheBindungen(o, s)$
2: **if** $|angebote| > 0$ **then**
3: selektiere ein $x = (o', a, b, t, r)$ aus $angebote$
4: deaktiviere Kollisionstest zwischen o' und o
5: $t' \Leftarrow t$
6: $r' \Leftarrow r$
7: $(t', r') \Leftarrow testeBewegung(o, t', r')$
8: **if** $(t' = t) \wedge (r' = r)$ **then**
9: $E(b) = a$
10: **for all** $o'' \in O(s)$ **do**
11: **if** $vorgaenger(o, o'')$ **then**
12: wende t und r auf o an
13: **else**
14: aktiviere Kollisionstest zwischen o' und o

Hierbei wird die in Abschnitt 3.9.3 vorgestellte Funktion $testeBewegung(o, t, r, s)$ verwendet, die testet, ob eine Transformation auf ein Objekt angewendet werden darf. Wenn $sucheBindungen(o, s)$ mehr als ein Angebot zurückgibt, wird dasjenige gewählt, für das der Abstand und damit $|t|$ am geringsten ist.

Durch die Verwendung dieses Algorithmus kann garantiert werden, daß alle drei für die Konsistenz einer Szene entscheidenden Prädikate $bindok(o, o')$, $nokoll(o, o')$ und $ausgerichtet(o)$ nach der Erfüllung einer Flächenbindung für alle Objektkombinationen erfüllt sind, sofern sie das auch vor der Erfüllung waren.

3.9.3 Bewegungseinschränkung durch erfüllte Flächenbindungen

Vor der Anwendung einer Transformation auf ein Objekt wird von der Funktion $testeBewegung(o, t, r)$ (Algorithmus 3, Seite 33) getestet, ob dadurch Probleme mit erfüllten Flächenbindungen oder mit einem der anderen Constraints auftreten können. Je nach Situation wird die Transformation dann entweder zurückgewiesen oder so verändert, daß keine Probleme mehr auftreten.

Die aus der Translation t und der Rotation r bestehende Transformation wirkt nicht nur auf das Objekt o selbst sondern durch die dynamische Gruppierung auch auf alle anderen Objekte, die im Bindungsgraph Nachfolger von o sind. Damit können alle erfüllten Flächenbindungen zwischen o und seinen Nachfolgern vernachlässigt werden, da sich ihre Position relativ zueinander nicht verändert. Dafür müssen aber alle erfüllten Flächenbindungen zwischen o oder einem seiner Nachfolger und allen anderen Objekten bei der Bearbeitung von t und r berücksichtigt werden.

Durch eine erfüllte Flächenbindung zwischen zwei Objekten kann das Objekt mit der Bindungsfläche nur noch parallel zur Bindungsfläche verschoben werden, da sonst der Abstand zwischen den beiden Flächen verändert würde. In den Zeilen 1 bis 13 werden deshalb alle Normalen auf relevanten und erfüllten Bindungsflächen gesammelt und die Transformation t entsprechend verändert. Außerdem kann ein Objekt mit erfüllten Flächenbindungen nur noch um eine Achse rotiert werden, die senkrecht zu den erfüllten Bindungsflächen steht, da Bindungs- und Angebotsfläche sonst nicht mehr parallel sind. Sollte es für ein Objekt auch eine Objektausrichtung geben, so kann das Objekt ebenfalls nur noch um eine Achse parallel zu diesem Vektor rotiert werden, um die Ausrichtung des Objekts nicht zu verändern. Die Rotation r wird deshalb in den Zeilen 14 bis 21 entsprechend bearbeitet. In den Zeilen 22 bis 30 wird dann geprüft, ob bei der Anwendung der resultierenden Translation und Rotation alle Bindungsflächen noch in den entsprechenden Angebotsflächen enthalten sind. Fällt dieser Test positiv aus, bleibt sowohl $bindok(o, o')$ als auch $ausgerichtet(o)$ für alle Objekte in der Szene erhalten.

Es bleibt dann noch zu testen, ob durch t und r eine unerlaubte Kollision in der Szene entsteht.

Algorithmus 3 Test der Zulässigkeit einer Bewegung eines Objekts

testeBewegung(o, t, r, s)

1: $normalen = \emptyset$ {Bearbeite die Translation t}
2: **for all** $o' \in O(s)$ **do**
3: **if** *vorgänger*(o, o') **then**
4: **for all** $b \in B(o')$ **do**
5: **if** $E(b) \neq$ NULL $\wedge \neg$*vorgaenger*$(o,$ *objekt*$(E(b)))$ **then**
6: $normalen = normalen \cup$ *welt*(*normale*$(P(b))$)
7: entferne alle Normalen aus *normalen*, die zu einer anderen Normalen, die noch in *normalen* ist, parallel oder antiparallel sind
8: **if** |normalen| $= 1$ **then**
9: setze t senkrecht zu *normalen*[0]
10: **else if** $|normalen| = 2$ **then**
11: setze t senkrecht zu *normalen*[0] und *normalen*[1]
12: **else if** $|normalen| \geq 3$ **then**
13: $t \Leftarrow (0, 0, 0)$

14: **for all** $o' \in O(S)$ **do** {Bearbeite die Rotation r}
15: **if** *vorgänger*$(o, o') \wedge N(o') \neq (0, 0, 0)$ **then**
16: $normalen = normalen \cup W(o')$
17: entferne alle Normalen aus *normalen*, die zu einer anderen Normalen, die noch in *normalen* ist, parallel oder antiparallel sind
18: **if** $|normalen| = 1$ **then**
19: *Rotationsachse*$(r) =$ *normalen*[0]
20: **else if** $|normalen| \geq 2$ **then**
21: $r = (0, 0, 0, 0)$

22: **if** $t = (0, 0, 0) \wedge r = (0, 0, 0, 0)$ **then** {t und r sind endgültig}
23: **return** (t, r)
24: **else**
25: $ok =$ TRUE {Bleibt die Flächeninklusion erhalten?}
26: **for all** $o' \in O(S)$ **do**
27: **if** *vorgänger*(o, o') **then**
28: **for all** $b \in B(o')$ **do**
29: **if** $E(b) \neq$ NULL $\wedge \neg$*vorgänger*$(o,$ *objekt*$(E(b))) \wedge \neg$*enthaelt*$(P(E(b)), P(b))$ **then**
30: $ok =$ FALSE

31: **if** ok = FALSE **then**
32: **return** $((0, 0, 0), (0, 0, 0, 0))$
33: **else**
34: **for all** $o' \in O(s)$ **do** {Gibt es eine Kollision durch t und r?}
35: **if** *vorgänger*(o, o') **then**
36: wende t und r auf o' an
37: **if** es gibt eine unerlaubte Kollision in der Szene **then**
38: $ok =$ FALSE
39: **for all** $o' \in O(s)$ **do**
40: **if** *vorgänger*(o, o') **then**
41: wende $-r$ und $-t$ auf o' an
42: **if** $ok =$ FALSE **then**
43: **return** $((0, 0, 0), (0, 0, 0, 0))$
44: **else**
45: **return** (t, r)

Falls ja werden beide gleich Null gesetzt. Damit wird auch $nokoll(o, o')$ durch diesen Algorithmus erhalten.

3.9.4 Aufbruch von erfüllten Flächenbindungen

Sobald ein Objekt im freien Interaktionsmodus angeklickt wird, werden alle erfüllten Flächenbindungen dieses Objekts aufgebrochen, so daß es wieder frei in der Szene bewegt werden kann. Dazu wird folgender Algorithmus verwendet:

Algorithmus 4 Aufbruch aller erfüllten Flächenbindungen eines Objekts o

entferneBindungen(o)

```
1: for all b ∈ B(o) do
2:   if E(b) ≠ NULL then
3:     setze wargebunden(objekt(E(b)), o)
4:     aktiviere Kollisionstest zwischen objekt(E(b)) und o
5:     E(b) = NULL
```

Der Kollisionstest kann in Zeile 4 aktiviert werden, da es nach Beendigung des Algorithmus keine erfüllten Flächenbindungen zwischen den beteiligten Objekten mehr geben kann. Alle Flächenbindungen, bei denen o die Bindungsfläche gestellt hat, sind gerade aufgebrochen worden. Da keine gerichteten Kreise im Bindungsgraphen erlaubt sind, kann es zudem keine erfüllten Flächenbindungen zwischen o und $objekt(E(b))$ gegeben haben, bei denen eine Angebotsfläche von o verwendet wurde.

Der Algorithmus erhält außerdem die Konsistenz einer Szene: $bindok(o, o')$ bleibt erhalten, da Bindungen nur aufgehoben aber keine neuen erzeugt wurden. $ausgerichtet(o)$ ist ebenfalls erhalten, weil kein Objekt bewegt wurde. Durch die Aktivierung des Kollisionstests in Zeile 4 können neue Kollisionen auftreten. Diese werden aber durch $wargebunden(o, o')$ aufgefangen. Damit bleibt auch $nokoll(o, o')$ erhalten.

3.10 Eingabe einer Transformation

Das IConS-System kann als Eingabe beliebige Transformationen, die aus Translationen und Rotationen im Raum bestehen, verarbeiten. Solche Transformationen können zum Beispiel durch mechanische, magnetische oder optische Positions- und Orientierungssensoren erzeugt werden, die als Eingabegeräte für Anwendungen aus dem Bereich der Virtual Reality verwendet werden.

Da während der Entwicklung des Systems keine derartige Ausrüstung zur Verfügung stand, können Transformationen im IConS-System nur mit einer Maus eingegeben werden. Dazu muß die zweidimensionale Mauseingabe in Translationen und Rotationen im Raum umgesetzt werden, wobei aus jeder Bewegung der Maus, abhängig von der gedrückten Maustaste, entweder eine Translation oder eine Rotation erzeugt wird. In den nächsten Abschnitten wird das Maus-Interface für das IConS-System beschrieben.

3.10.1 Grundprinzip der Eingabe

Um eine Transformation für ein Objekt eingeben zu können, muß auch klar sein, für welches Objekt die Transformation gelten soll. Hierfür gibt es zwei grundlegende Möglichkeiten:

- Das Objekt wird zuerst in einem getrennten Schritt selektiert. Alle danach eingegebenen Transformationen beziehen sich solange auf das ausgewählte Objekt, bis ein anderes Objekt selektiert wird.
- Das Objekt, für das die Transformation gelten soll, wird bei der Eingabe der Transformation implizit ausgewählt. Dazu muß das System jede Transformation eindeutig und in einer für den Benutzer durchschaubaren Weise einem Objekt zuordnen.

Das Maus-Interface des IConS-Systems ist eine Umsetzung der zweiten Variante. Bei der Eingabe einer Translation kann das Objekt direkt durch Anklicken mit der Maus selektiert werden. Zur Eingabe einer Rotation wird eine Heuristik verwendet, um das entsprechende Objekt auszuwählen.

3.10.2 Translation

Die Benutzerschnittstelle eines Modellierungssystems sollte die Eingabe von Translationen in allen drei Richtungen des Raumes erlauben. Mit einer Maus können aber nur ebene Translationen eingegeben werden. Aus diesen wird eine Translation in einer Ebene errechnet, die je nach verwendeter Maustaste parallel oder senkrecht zum Bildschirm ist.

Parallel zum Bildschirm

Zur Eingabe einer Translation wird das Objekt mit der Maus angeklickt und dadurch ausgewählt. Wird die Maus dann mit gedrückter Maustaste verschoben, so erzeugt IConS eine Translation für das Objekt. Ein Objekt, das keine erfüllten Flächenbindungen hat, bewegt sich dabei so, daß der Mauszeiger beim Verschieben immer auf dieselbe Stelle des Objekts zeigt. Das Objekt haftet sozusagen an der Maus.

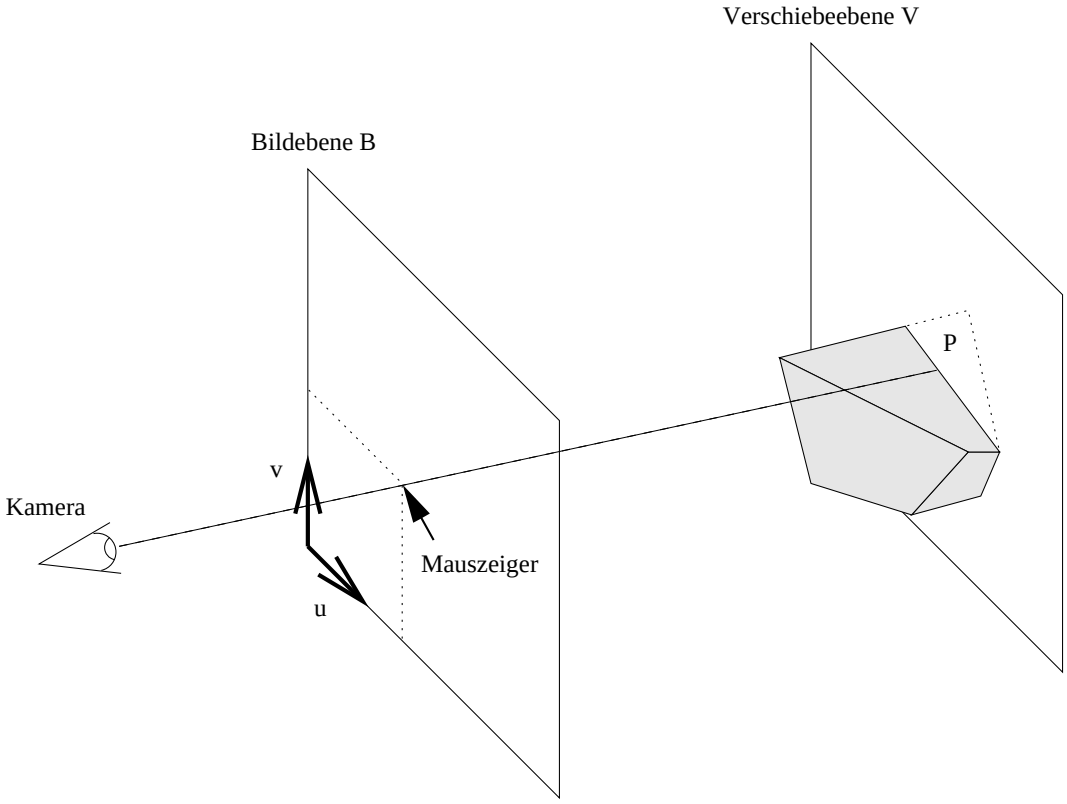

Abbildung 3.15: Eingabe einer Translation. Durch die Positionen von Kamera und Mauszeiger wird eine Gerade im Raum – der sogenannte *Mausstrahl* erzeugt. Der am nächsten bei der Kamera liegende Schnittpunkt zwischen dem Mausstrahl und einem Objekt in der Szene ist Punkt P. Durch diesen Punkt wird die *Verschiebeebene* V parallel zur *Bildebene* B gelegt. Das Objekt wird dann so verschoben, daß P die Verschiebeebene V nicht verläßt und sich immer auf dem Mausstrahl befindet.

Abbildung 3.15 zeigt, wie dieses Verhalten erreicht wird. Beim Klicken in das IConS-Fenster wird ein von der Position des Mauszeigers ausgehender Strahl erzeugt und mit der Szene geschnit-

ten. Trifft dieser Mausstrahl auf ein Objekt, so wird dieses Objekt selektiert und eine Verschiebeebene V durch den Auftreffpunkt P gelegt. Diese Ebene ist parallel zur Bildebene der Kamera und damit parallel zum Bildschirm. Beim Verschieben der Maus wird das Objekt so mitverschoben, daß sich P immer unter dem Mauszeiger befindet und die Ebene V nicht verläßt.

Senkrecht zum Bildschirm

Um ein Objekt auch „in der Tiefe“, also senkrecht zum Bildschirm, verschieben zu können, wird der auf dieselbe Weise gewonnene Translationsvektor um 90 Grad um die horizontale Achse u der Bildebene B rotiert. Eine Bewegung der Maus in der Vertikalen entspricht somit einer Bewegung des Objekts in der Tiefe.

Dadurch wird die Problematik umgangen, eine geeignete Skalierungsfunktion für die Bewegung in die Tiefe zu finden. Durch die Tatsache, daß das Objekt am Mauszeiger haftet, ist bei einer Bewegung parallel zum Bildschirm nicht nur die Richtung der Translation sondern auch die Länge vorgegeben. Verschiebt man ein Objekt in der Tiefe, so kann man nie ausschließen, daß sich die Projektion eines Punkts auf dem Bildschirm nicht verändert. Damit kann auch der obige Mechanismus, bei dem das Objekt am Mauszeiger haftet, nicht zur Bestimmung der Länge der Verschiebung verwendet werden.

Durch die Rotation des Translationsvektors ist es zudem möglich, das Objekt auch zusätzlich in der Horizontalen zu verschieben. Dies erweist sich besonders dann als praktisch, wenn ein Objekt eine erfüllte Flächenbindung hat. Die Translationen werden dann in die Ebene der Flächenbindung projiziert und es ist so möglich, mit beiden Translationsarten auf ähnliche Weise zu arbeiten.

3.10.3 Rotation

Die Eingabe von Rotationen wird dadurch erschwert, daß die Objekte um eine beliebige Rotationsachse rotiert werden können. Damit muß nicht nur ein Winkel sondern auch eine Rotationsachse aus der Mauseingabe errechnet werden. Wenn ein Objekt allerdings durch eine erfüllte Flächenbindung gebunden ist, hat es auch eine eindeutige Rotationsachse. Ein Rotationsinterface sollte daher für beide Fälle eine möglichst einheitliche Bedienung erlauben.

In der Literatur gibt es hierfür bereits eine Reihe von Lösungen, wie beispielsweise das von Shoemake vorgestellte Arcball-Interface [Sho92, Sho94]. Dieses gilt allgemein als eines der am einfachsten zu bedienenden Rotationsinterfaces. Es enthält darüberhinaus einen Mechanismus, mit dem die Rotationen auf eine Achse eingeschränkt werden können und der gleichzeitig eine einheitliche Bedienung erlaubt.

Arcball-Interface

Beim Arcball-Interface werden die Mauseingaben auf eine Kugel projiziert und aus dem Bogen auf der Kugeloberfläche eine Rotation errechnet. Abbildung 3.16 verdeutlicht dies: Aus dem Startpunkt A und dem Endpunkt B wird ein zu einem Großkreis gehörender Bogen berechnet. Die Rotationsachse $\vec{x}$ ist dann senkrecht zu diesem Großkreis, der Winkel der Rotation beträgt 2θ. Start- und Endpunkte, die außerhalb der Kugel liegen, werden auf die Außenseite des durch die Kugel gebildeten Kreises projiziert und ermöglichen eine Rotation um eine Achse, die senkrecht zum Bildschirm ist.

Die so erzeugten Rotationen hängen nur vom Start- und Endpunkt des Bogens ab und sind unabhängig vom dazwischenliegenden Pfad des Mauszeigers. Durch die Verdoppelung des Winkels ist es möglich, ein Objekt in einem Zug um jede beliebige Achse um 360 Grad zu drehen.

Die Rotationen können auf eine bestimmte Achse eingeschränkt werden, indem die Start- und Endpunkte des Bogens zuerst auf den zu der Achse gehörenden Großkreis und dann wieder auf die Kugeloberfläche projiziert werden.

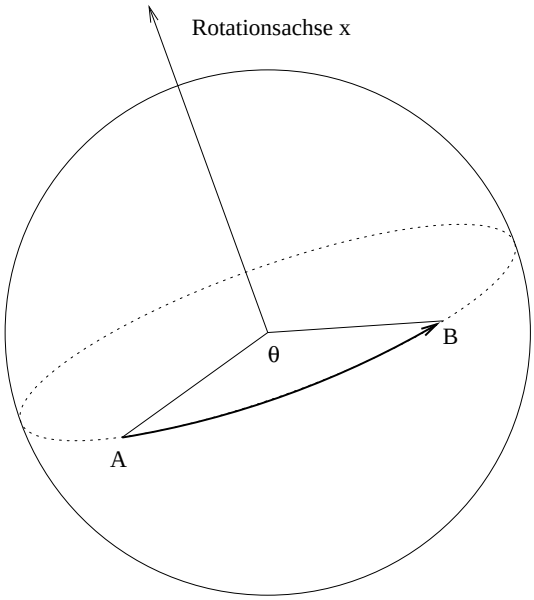

Abbildung 3.16: Funktion des Arcball-Interfaces. Bei einer Bewegung der Maus von Punkt A nach Punkt B wird eine Rotation um die Achse $\vec{x}$ um den Winkel 2θ erzeugt. Hierzu muß sowohl der Mittelpunkt als auch der Radius der Kugel festliegen, auf die die Bildschirmeingabe projiziert wird.

Einsatz des Interfaces

Im IConS-System wird die kleinste Kugel, die das jeweilige Objekt enthält, als Kugel für das Arcball-Interface verwendet (siehe Abbildung 5.18). Dadurch ist eine Selektion des Objekts durch Anklicken wie bei einer Translation nicht mehr möglich. Der Startpunkt eines eingegebenen Bogens sollte möglichst frei wählbar sein, um die möglichen Rotationsachsen und -winkel nicht bereits durch den Selektionsmechanismus zu beschränken. Insbesondere sollte es sogar möglich sein, an einem Punkt außerhalb der Kugel zu beginnen, um eine Rotation senkrecht zur Bildebene eingeben zu können.

Deshalb wurde für die Rotationen eine Heuristik entwickelt, die bei der Wahl des Startpunktes ein Objekt selektiert, auf das die Rotation angewendet werden soll. Diese Heuristik wird im nächsten Abschnitt beschrieben.

Heuristische Selektion

Wenn der Benutzer an eine Stelle des Bildschirms klickt, wird eine Liste L aller Objekte erstellt, die für die Rotation in Frage kommen. Dies sind alle Objekte, deren um den Faktor $f > 1$ vergrößerte Umkugel vom Mausstrahl geschnitten wird (siehe dazu auch Abbildung 3.15). f muß größer als 1 sein, damit die Objekte auch um eine Achse senkrecht zum Bildschirm rotiert werden können. In der Praxis hat sich $f = 1.4$ bewährt. Wenn sich das System in einem der beiden Interaktionsmodi befindet, muß es außerdem möglich sein, das Objekt überhaupt zu rotieren. Dazu müssen erfüllte Flächenbindungen und die Objektausrichtung des Objekts und der mit ihm dynamisch gruppierten Objekte untersucht werden.

Dann werden alle Objekte aus L entfernt, die von der Kamera aus unsichtbar sind, wobei ein Objekt als unsichtbar gilt, wenn sein Zentrum von der Kamera aus nicht sichtbar ist. Außerdem werden auch alle Objekte aus L entfernt, für die die Projektion des Umkugelradius auf den Bildschirm einen minimalen Wert – hier 25 Pixel – unterschreitet, da das Arcball-Interface umso empfindlicher ist, je kleiner die Projektion der Umkugel ist. Diese beiden Maßnahmen dienen dazu, unerwartete oder schwer zu handhabende Fälle aus L zu entfernen. Würde L dadurch keine

Objekte mehr enthalten, d.h. alle Objekte sind entweder unsichtbar oder zu klein, so wird L nicht verändert, damit überhaupt eine Rotation möglich ist.

L wird dann nach aufsteigendem Radius der Umkugeln sortiert, damit kleinere Objekte bevorzugt ausgewählt werden, da sie auch nur in einem kleineren Bildschirmbereich selektiert werden können. Wenn mehrere Objekte in L denselben Umkugelradius haben (meist sind dies mehrere Instanzen desselben Objektmodells), so werden sie nach aufsteigendem Abstand von der Bildschirmmitte sortiert. Diesem Vorgehen liegt die Annahme zugrunde, daß der Benutzer am ehesten den Teil einer Szene manipulieren will, auf den er direkt blickt. In Abbildung 3.17 ist ein Beispiel für die Wirkungsweise des Sortierteils der Heuristik angegeben.

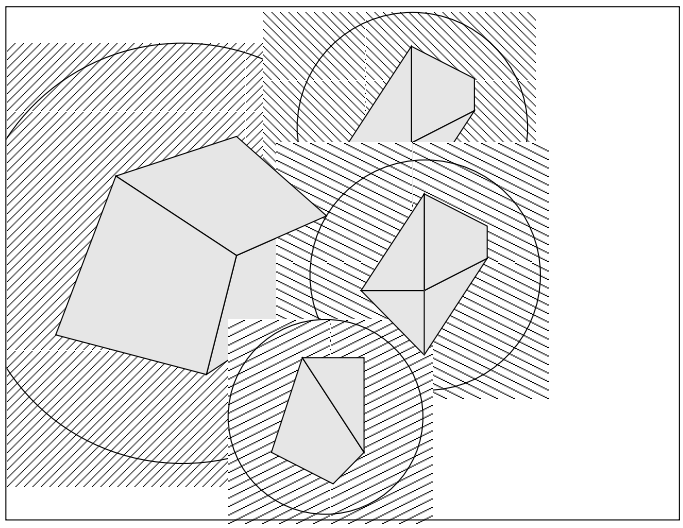

Abbildung 3.17: Erläuterung des Sortierteils der Heuristik zur Selektion beim Rotieren. Alle Objekte haben denselben Abstand von der Kamera. Dadurch entspricht die Größe auf dem Bildschirm auch der absoluten Größe der Objekte. Die gefüllten Kreise markieren den Bereich, in dem das jeweilige Objekt selektiert wird. Der rechteckige Rahmen stellt das Bildschirmfenster dar.

Kapitel 4

Implementierung

Dieses Kapitel geht auf einige Details der Implementierung und Bedienung des IConS-Systems ein. Zunächst wird die verwendete Hard- und Software beschrieben. In Abschnitt 4.3 wird die Bedienung des Systems erläutert. Die folgenden beiden Abschnitte beschäftigen sich mit der Erzeugung der Objektmodelle. Das Kapitel endet mit der Beschreibung zweier Routinen des Systems (Abschnitt 4.6) und einem Überblick über numerische Probleme, die bei der Programmierung angegangen werden mußten (Abschnitt 4.7).

4.1 Verwendete Hard- und Software

Zur Entwicklung von IConS wurde eine Indigo2 High Impact Workstation von Silicon Graphics (SGI) mit einem MIPS R4400 Prozessor und 256 MB Hauptspeicher verwendet. Die Programmierung erfolgte in der Programmiersprache C++ unter Verwendung des *OpenGL Optimizer*-Paketes von SGI. Außerdem wurden noch einige weitere Softwarepakete verwendet, auf die in Abschnitt 4.2 näher eingegangen wird. Der IConS-Quellcode umfaßt ohne diese Pakete rund 6300 Zeilen.

4.1.1 OpenGL Optimizer

Das Paket OpenGL Optimizer (kurz: Optimizer) ist eine Grafik-API, die für die Visualisierung von sehr umfangreichen Modellen optimiert ist. Es beruht auf der Grafiksprache OpenGL und stellt darauf aufbauend Werkzeuge zur Speicherung, Bearbeitung und Darstellung von großen Modellen zur Verfügung [Eck97, Wen97].

Der wichtigste Grund für die Verwendung von Optimizer zur Programmierung von IConS ist die Art, wie die Daten einer Szene in einem Szenengraphen abgespeichert werden. In Abschnitt 3.7 wurde dargestellt, daß die Konzepte zur Gruppierung und Instantiierung in IConS durch die Abbildung der Szene auf den Optimizer-Szenengraphen sehr einfach realisiert werden können. Wäre die Grafikprogrammierung direkt in OpenGL erfolgt, dann wäre es nötig gewesen, einen ähnlichen Mechanismus von Hand zu implementieren. Andere Grafik-APIs wie etwa Open Inventor verwenden zwar auch Szenengraphen, die aber eine andere Struktur haben und dadurch für die direkte Umsetzung von Gruppierung und Instantiierung nicht geeignet sind.

4.2 Externe Software

Für einige Teilaufgaben, nämlich für die Kollisionserkennung, die Formatierung der Ausgabe des Szenengraphen und die Eingabe von dreidimensionalen Rotationen, wurde bei der Programmierung von IConS auf externe Software zurückgegriffen. Auf diese Weise konzentrierte sich der Entwicklungsaufwand auf die Erstellung des eigentlichen Systems, ohne – etwa für die Kollisionserkennung – auf Techniken, die dem aktuellen Forschungsstand entsprechen, verzichten zu müssen.

In den folgenden Abschnitten werden die verwendeten Pakete mit den wesentlichen Eigenschaften kurz vorgestellt.

4.2.1 Kollisionserkennung mit V-COLLIDE

Die Kollisionserkennung erfolgt mit dem V-COLLIDE Paket [HLC+97], das an der University of North Carolina at Chapel Hill entwickelt worden ist. Es wurde für den Einsatz in VRML-Browsern optimiert und erfüllt die grundlegenden Anforderungen an ein interaktives und offenes System:

- *Dynamische Veränderbarkeit des Szenengraphen*: Objekte können bewegt, hinzugefügt oder entfernt werden.
- *Interaktive Geschwindigkeit*: Das System erreicht die für die Interaktion unter Echtzeitbedingungen nötige Geschwindigkeit für hinreichend große Modelle.
- *Allgemeinheit der Modelle*: Es werden keine Annahmen über die Geometrie der Modelle (zum Beispiel Konvexität oder keine Selbstdurchdringungen) gemacht.

Diese Eigenschaften sind für das IConS-System sehr wichtig. Da die Kollisionserkennung für jede Bewegung eventuell sogar mehrfach aufgerufen wird, spielt die Geschwindigkeit eine entscheidende Rolle.

Schnittstelle zu V-COLLIDE

Die Eingabe eines Objektmodells in V-COLLIDE erfolgt als eine Menge von Dreiecken (*triangle soup*), aus denen interne Datenstrukturen für eine effiziente Verwaltung und Kollisionserkennung erzeugt werden. Ein Objekt kann jederzeit durch die Eingabe einer Transformation positioniert und orientiert werden. Die Transformationen sind auf Translationen und Rotationen beschränkt. Eine Skalierung der Modelle wird von V-COLLIDE nicht unterstützt und wäre damit nur durch eine komplette Neueingabe des skalierten Objektmodells als *triangle soup* möglich. Die Skalierung von Objekten ist im IConS-System daher nicht vorgesehen.

Auf Anforderung errechnet V-COLLIDE eine Liste von Objektpaaren, zwischen denen eine Kollision besteht. Eine Ausgabe der kollidierenden Dreiecke der Objekte erfolgt nicht und ist für dieses System auch nicht nötig.

Berücksichtigung von Ausnahmen

In Abschnitt 3.3.2 wurde beschrieben, daß für Objekte, zwischen denen eine erfüllte Flächenbindung besteht, eine Ausnahmebehandlung benötigt wird. Diese wird folgendermaßen umgesetzt:

V-COLLIDE ermöglicht es, einzelne Objektpaare von der Kollisionserkennung auszunehmen, so daß für diese Objektpaare keine Kollisionsmeldungen mehr erzeugt werden. Dieser Mechanismus wird verwendet, um die Kollisionserkennung zwischen Objekten, die durch eine erfüllte Flächenbindung verbunden sind, zu deaktivieren.

Wird eine Flächenbindung wieder aufgebrochen, so können die beteiligten Objekte trotzdem immer noch kollidieren und dadurch jede weitere Interaktion blockieren. Daher werden Objektpaare nach dem Bruch einer Flächenbindung zunächst auf eine Ausnahmeliste gesetzt. Alle Kollisionen von Objektpaaren auf dieser Liste werden ignoriert. Ein Objektpaar wird automatisch aus der Liste gelöscht, sobald für eine gültige Objektkonfiguration in der Szene keine Kollision zwischen den beiden Objekten besteht.

Dies führt in der Praxis dazu, daß sich zwei Objekte nach dem Aufbrechen einer sie verbindenden Flächenbindung solange durchdringen können, bis sie einmal vollständig getrennt wurden.

4.2.2 Formatierte Ausgabe des Szenengraphen mit Graphplace

Die Formatierung der Ausgabe des Szenengraphen im rechten Teil des Fensters des IConS-Systems (siehe Abbildung 4.1) wird von Graphplace [vE94] übernommen. Dieses Programm kann beliebige, gerichtete Graphen formatieren. Graphplace erhält als Eingabe eine Liste der Knoten und

gerichteten Kanten des Graphen und erzeugt daraus eine um die Koordinaten der Knoten und der Kantenenden angereicherte Liste.

Graphplace ordnet die Knoten zunächst in der Vertikalen entsprechend ihrem Abstand von der Wurzel des Graphen an, wobei es auch mit gerichteten Kreisen im Graphen umgehen kann. Danach werden die Knoten entsprechend einer Heuristik in der Horizontalen plaziert. Da die darzustellenden Szenengraphen keine gerichteten Kreise enthalten, entsteht durch diesen Algorithmus ein Hasse-Diagramm des Szenengraphen und es kann auf die explizite Angabe der Richtungen der einzelnen Kanten verzichtet werden.

Die Kommunikation zwischen Graphplace und IConS erfolgt über die von UNIX bereitgestellte Standardeingabe und -ausgabe. Dazu wird für jede Formatierung ein neuer Graphplace-Prozeß gestartet.

4.2.3 Arcball-Rotationsinterface

Die Funktionsweise des Arcball-Rotationsinterface wurde bereits in Abschnitt 3.10.3 vorgestellt und erläutert. Die in IConS verwendete Implementierung basiert auf dem in [Sho94] enthaltenen Code, der an die speziellen Erfordernisse der Transformationseingabe in IConS angepaßt wurde, ohne die wesentlichen Eigenschaften des Arcball-Rotationsinterfaces zu verändern.

4.3 Bedienung des IConS-Systems

4.3.1 Eingabe

Die Bedienung des Systems erfolgt nur über Maus- und Tastatureingaben. Tabelle 4.1 listet die Mauseingabe auf, Tabelle 4.2 zeigt die Tastaturbefehle. Bei letzteren ist die Groß- und Kleinschreibung unerheblich.

4.3.2 Bildschirmausgabe

Abbildung 4.1 zeigt das Hauptfenster des IConS-Systems, das in drei Teile unterteilt ist:

- In der linken Hälfte ist eine perspektivische Ansicht der gerade bearbeiteten Szene zu sehen, in welcher der Benutzer mit der Maus die Szene verändern kann.

  Die Farben der einzelnen Objekte werden in Anlehnung an [ZHH96] zufällig vergeben und sind in den Objektdaten gespeichert. Damit haben auch alle Instanzen dieser Objektdaten, wie hier die vier Stühle am Tisch, dieselbe Farbe.

- Im rechten Teil des Fensters wird der aktuelle Szenengraph der Szene angezeigt. Der Benutzer kann daraus die Instantiierungen in der Szene und die aktuelle dynamische Gruppierung ablesen. Wenn der Benutzer mit der Maus auf den Knoten einer Objektinstanz klickt, wird diese Instanz zusammen mit allen dazu gruppierten Instanzen im linken Teil des Fensters hervorgehoben.

- Die Titelzeile des Fensters zeigt den gerade aktiven Modus an.

4.4 Leitlinien für die Definition von Angebots- und Bindungsflächen

Die Verwendung der Flächenbindungen macht einen wesentlichen Teil des IConS-Systems aus. Damit dieser Mechanismus effektiv genutzt werden kann, muß zu den verwendeten Objektmodellen ein sinnvolles und in sich schlüssiges System von Angebots- und Bindungsflächen definiert werden. Ob ein solches System sinnvoll und schlüssig ist, hängt sehr stark von der Funktion ab, die die verschiedenen Objekte für den Benutzer haben. Da die Funktion eines Objekts ein sehr subjektives

Maustaste	**eingeschränkter und freier Interaktionsmodus**	**Navigationsmodus**
linke	Verschiebung parallel zur Bildebene	Verschiebung parallel zur Bildebene
mittlere	Rotation	Rotation
rechte	Verschiebung in der Horizontalen senkrecht zur Bildebene	Verschiebung der Kamera entlang der Blickrichtung

Tabelle 4.1: IConS-Mauskommandos. Hierbei besteht kein Unterschied zwischen dem eingeschränkten und dem freien Interaktionsmodus. Die Mauskommandos des Navigationsmodus entsprechen weitgehend denen der Interaktionsmodi.

Taste	**Befehl**
i	Umschalten in den eingeschränkten Interaktionsmodus
u	Umschalten in den freien Interaktionsmodus
n	Umschalten in den Navigationsmodus
l	Laden eines neuen Objekts
o	Instantiieren eines existierenden Objekts
d	Löschen eines Objekts
w	Speichern des kompletten Szenengraphen (unter Verlust der Bindungsfunktionalität)
p	Aktivieren oder Deaktivieren der Pseudo-Gravitation
x	Zurücksetzen der Kameraposition für beide Fensterhälften
h	Anzeigen einer kurzen Hilfe
ESC	Beenden des Programms

Tabelle 4.2: IConS-Tastaturkommandos. Die ersten drei Befehle werden zur Auswahl des Modus verwendet. Die zweite Gruppe dient zur Manipulation einzelner Objekte. In der dritten Gruppe sind sonstige Befehle zusammengefaßt.

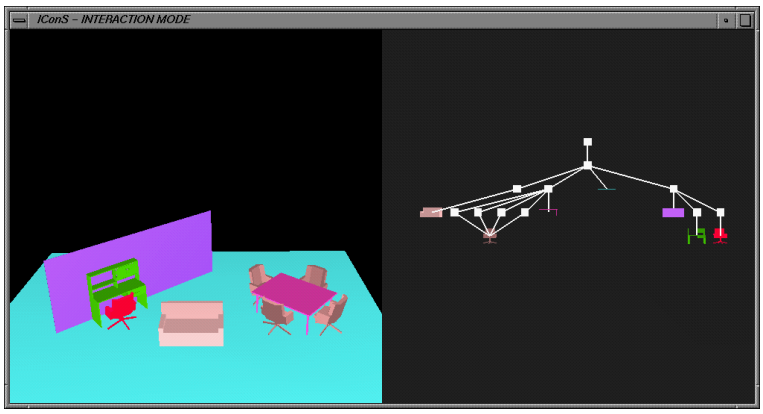

Abbildung 4.1: Beispiel einer Bildschirmausgabe des IConS-Systems. Der linke Teil des Fensters zeigt die aktuelle Szene. Im rechten Teil ist der zugehörige Szenengraph dargestellt. Die Titelzeile zeigt den gerade aktiven Modus an.

Kriterium ist, müssen sowohl die Definitionen der Angebots- und Bindungsflächen als auch die Ordnung der Bezeichnungen an das Verständnis der einzelnen Benutzer angepaßt werden.

Darüberhinaus spielen eine Reihe objektiver Überlegungen, wie der dadurch verursachte Rechenaufwand und die Empfindlichkeit für numerische Fehler, bei der Definition der Angebots- und Bindungsflächen eine wichtige Rolle. In den folgenden Abschnitten wird auf einige objektive Gesichtspunkte eingegangen.

4.4.1 Rechenaufwand

Bei der Suche nach erfüllbaren Flächenbindungen für ein Objekt wird jede nicht erfüllte Angebotsfläche mit jeder Bindungsfläche getestet. Der Aufwand für diese Suche ist $O(a \cdot b)$, wobei a die Anzahl der Angebotsflächen und b die Anzahl der Bindungsflächen in einer Szene ist. Da bereits jeder einzelne Test zwischen zwei Flächen einen erheblichen Rechenaufwand verursacht, ist ein wichtiges Ziel, bei der Definition der Flächen die Anzahl der Flächen klein zu halten.

Die Standfläche eines Tisches mit vier Tischbeinen kann zum Beispiel entweder durch vier einzelne Bindungsflächen repräsentiert werden oder aber durch eine einzige Bindungsfläche, welche die vier Tischbeine enthält. Unter dem Gesichtspunkt des Rechenaufwands ist die zweite Möglichkeit klar zu bevorzugen.

Der Rechenaufwand für die einzelnen Tests zwischen zwei Flächen kann zusätzlich durch die Verwendung möglichst spezieller Bezeichnungen für die Flächen stark verringert werden. Der erste Schritt in diesem Test ist der Vergleich der beiden Bezeichner. Je mehr Paare bereits hier ausgeschlossen werden können, desto geringer ist der Aufwand.

4.4.2 Physikalische Exaktheit

Durch eine größere Zahl von Flächen wird nicht unbedingt gewährleistet, daß die erzeugte Szene physikalisch korrekter ist. Betrachtet man wieder das im vorherigen Abschnitt erwähnte Beispiel eines Tisches, so steht der Tisch auf jeden Fall sicher auf einem Fußboden, wenn er mit der gesamten Standfläche und damit mit allen vier Tischbeinen darauf steht. Hat er dagegen vier getrennte Bindungsflächen, so kann er zum Beispiel mit nur einem Tischbein auf dem Fußboden stehen und sich damit in einer physikalisch nicht plausiblen Lage befinden.

Allerdings ist es mit vier Bindungsflächen theoretisch auch möglich, den Tisch auf vier Klötze zu stellen. Im gegenwärtigen System scheitert dies aber an numerischen Ungenauigkeiten. Beim Versuch, die Bindungsflächen der Reihe nach zu erfüllen, kam es bisher immer zu einer Kollision zwischen dem Tisch und einem der Klötze (siehe Abschnitt 4.7.2).

4.4.3 Numerische Stabilität

In Situationen, in denen bei einer erfüllten Flächenbindung die Bindungsfläche am äußersten Rand der Angebotsfläche ist, kann es zu numerischen Problemen kommen. Abbildung 4.2 zeigt einen solchen Fall, bei dem ein Regal sowohl an einer Wand und als auch auf dem Fußboden steht und mit diesen auch durch erfüllte Flächenbindungen verbunden ist. Die Bindungsfläche für die Flächenbindung mit der Wand befindet sich ganz am unteren Rand der Angebotsfläche.

Durch die Flächenbindungen kann das Regal noch parallel zu Wand und Fußboden – also in Blickrichtung – verschoben werden. Durch numerische Ungenauigkeiten kann es vorkommen, daß die Bewegung nicht ganz parallel zum Fußboden ist. Das Regal dringt damit in den Fußboden ein. Um die Bewegung trotzdem zu ermöglichen, wurde bereits die Kollisionserkennung zwischen Objektpaaren mit einer erfüllten Flächenbindung ausgeschaltet. Hier wird die Bewegung aber trotzdem verhindert, da die Bindungsfläche für die Flächenbindung mit der Wand nicht mehr in der Angebotsfläche enthalten ist.

Dieses Problem kann umgangen werden, indem man die Bindungsfläche an der Rückwand des Regals nicht bis ganz zum unteren Ende hin definiert. Das Regal wird trotzdem korrekt positioniert, wenn es an der Wand und auf dem Fußboden steht. Steht das Regal nur an der Wand aber nicht auf dem Boden, so kann es dadurch auch etwas nach unten überstehen. Diese Konfiguration tritt

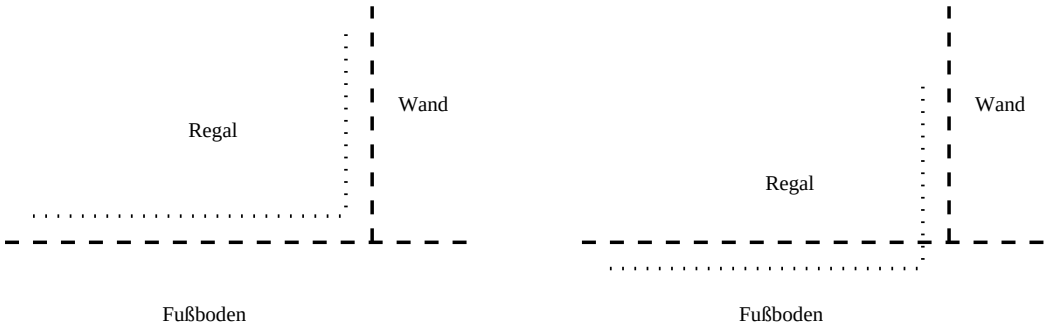

Abbildung 4.2: Eine numerisch instabile Situation. Im linken Teil ist ein Regal zu sehen, das je eine erfüllte Flächenbindung mit der Wand und dem Fußboden hat. Die Bindungsflächen (gepunktet) sind nur zur Verdeutlichung nicht direkt auf den Angebotsflächen (gestrichelt) gezeichnet. Der rechte Teil zeigt, wie das Regal durch numerische Ungenauigkeiten durch eine Bewegung parallel zu Wand und Decke unterhalb des Fußbodens gelangen würde. Dies wird von der Kollisionserkennung nicht verhindert, da sie durch die erfüllte Flächenbindung deaktiviert ist. Die Flächenbindung mit der Wand wäre aber nicht mehr erfüllt und die Bewegung wird verhindert, obwohl sie eigentlich zulässig ist.

in den meisten Anwendungen entweder gar nicht auf, da die Wände immer auf dem Fußboden stehen, oder kann vernachlässigt werden.

Aus diesem Grund ist es ratsam, mehrere Bindungsflächen eines Objekts, die gleichzeitig erfüllt werden können, so zu definieren, daß beide von einer eventuellen gemeinsamen Kante einen kleinen Abstand haben.

4.4.4 Ausreichende Flexibilität

Beschränkt man die Position von Angebots- und Bindungsflächen nicht auf in den Objektmodellen enthaltene Flächen, so kann man viel Flexibilität gewinnen. Abbildungen 4.3 und 4.4 zeigen ein Beispiel für diese Vorgehensweise anhand von Definitionen für Flächenbindungen des Typs *in-Front-of-Table*.

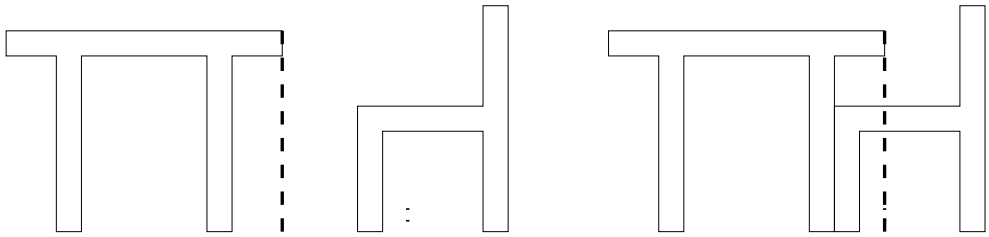

Abbildung 4.3: Beispiel für den Bindungstyp *in-Front-of-Table*. Im linken Teil sind ein Tisch und ein Stuhl mit den beiden Flächen zu sehen. Der rechte Teil zeigt die Lage der Objekte, nachdem die Flächenbindung erfüllt worden ist. Die Angebotsfläche (gestrichelt) ist die gesamte Seite des Tisches. Die Bindungsfläche (gepunktet) hat zwar die Breite des Stuhls, aber nur eine geringe Höhe. Diese Größe reicht aus, um einen Stuhl vor einen Tisch stellen zu können. Sie ermöglicht es aber auch, den Stuhl an einen niedrigeren Tisch zu stellen.

Die Bindungsfläche am Stuhl hat die Breite des ganzen Stuhls aber nur eine sehr geringe Höhe. Dies ist sinnvoll, da ein Stuhl mit der gesamten Breite vor einem Tisch stehen sollte. Der Abstand der beiden Objekte zum Fußboden soll aber durch diese Flächenbindung nicht festgelegt werden. Außerdem ermöglicht es die geringe Höhe der Bindungsfläche, den Stuhl an große und

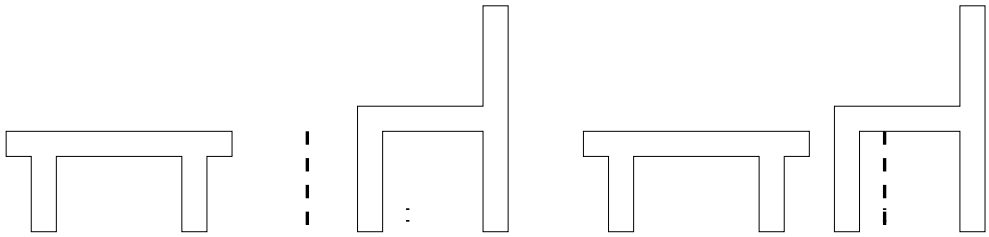

Abbildung 4.4: Dasselbe Beispiel wie in Abbildung 4.3 mit einem niedrigeren Tisch.

kleine Tische zu stellen. Dies wird auch erst durch den Abstand zwischen dem Tisch und der zugehörigen Angebotsfläche möglich.

4.5 Vorverarbeitung der Modelle

Damit ein Modell mit dem System verwendet werden kann, muß es im richtigen Dateiformat vorliegen, eine passende Struktur haben und korrekt skaliert sein. Außerdem sollte noch eine Constraintdatei angelegt werden, in der die zum Arbeiten mit den Flächenbindungen benötigten Daten enthalten sind.

4.5.1 Objektdaten

Optimizer 1.1 ist in der Lage, Dateien im Format von VRML 2.0 und Open Inventor einzulesen. Viele frei erhältliche Modelle sind bereits in diesen Formaten verfügbar. Ein Objektmodell, das nicht in einem dieser Formate vorhanden ist, kann mit einem Konvertierungsprogramm wie etwa Crossroads 3D [Rul98] entsprechend umgewandelt werden.

Bevor ein Objektmodell im IConS-System verwendet werden kann, muß es in das von Optimizer verwendete csb-Format umgewandelt und bearbeitet werden. Dies kann durch die beiden Skripte *wrl2csb* und *iv2csb* geschehen, die sich nur im Format der Eingabedatei unterscheiden.

Da das System von einer speziellen Form des Szenengraphen ausgeht, darf ein Objektmodell nur die Geometriedaten in einem einzigen Objektknoten enthalten. Das Zusammenfassen der Objektknoten des Eingabemodells wird von dem mit Optimizer mitgelieferten Programm *opoptimize* durchgeführt. Das Programm *extract* erzeugt aus dessen Ausgabe einen Szenengraphen, der nur aus diesem einzelnen Objektknoten besteht.

In weiteren Schritten wird das Modell mit *center* zentriert und mit *scale* skaliert und an die interne Maßeinheit Zentimeter angepaßt.

Das interne Koordinatensystem des Systems entspricht der Konvention x-Achse nach rechts, y-Achse nach hinten und z-Achse nach oben. Viele Modelle verwenden aber ein in der Computergrafik gebräuchliches Koordinatensystem, bei dem die y-Achse nach oben und die z-Achse nach vorne zeigt. In diesem System entspricht die negative z-Achse dann der Blickrichtung des Betrachters. Da durch die Flächenbindungen eine automatische Ausrichtung der Modelle erreicht wird, kann man ohne weiteres auch mit gemischten Koordinatensystemen arbeiten. Möchte man eine korrekte Ausrichtung des Objekts bereits beim Laden erreichen, so kann man durch Drehen des Objektmodells mit *rotatex* die beiden Koordinatensysteme zur Deckung bringen.

4.5.2 Erzeugung der Constraintdatei

Die Syntax der Constraintdatei ist in Tabelle 4.3 angegeben. Sie enthält eine Auflistung von Flächen mit den dazugehörigen Bezeichnungen. Die zu den Flächen gehörenden Normalen können entweder direkt angegeben werden oder vom System aus den ersten drei Eckpunkten der Fläche berechnet werden.

DATEI:	CONSTRAINT$^+$ ANGEBOT$^+$ AUSRICHTUNG | <end-of-file>
CONSTRAINT:	<surface-constraint> FLÄCHE <end-surface-constraint>
ANGEBOT:	<offer-constraint> FLÄCHE <end-offer-constraint>
FLÄCHE:	<none> | KBLOCK
KBLOCK:	BEZEICHNUNG POLYGON NORMALE
BEZEICHNUNG:	onFloor | onPlane | onTop | onStore | onWorkspace | onWall | inFrontOfTable
POLYGON:	NAT NAT *Zeilen, von denen jede eine* KOORDINATE *ist*
AUSRICHTUNG:	<orientation> KOORDINATE KOORDINATE <end-of-file>
NAT:	*eine positive, natürliche Zahl*
NORMALE:	<calc-normal> | KOORDINATE
KOORDINATE:	REAL REAL REAL *(entsprechend der x, y und z Koordinate eines Punktes)*
REAL:	*eine reelle Zahl*

Tabelle 4.3: Syntax der Constraintdatei wie in Abschnitt 4.5.2 beschrieben. Das Startsymbol ist DATEI. Jede Zeile entspricht einer Zeile der Constraintdatei. Kommentare sind in kursiver Schrift angegeben.

Der schwierigste Teil der Erstellung einer Constraintdatei ist die Eingabe der Flächen. Das Skript *getSG*, das automatisch von *wrl2csb* und *iv2csb* aufgerufen wird, erzeugt aus einem Modell drei Dateien, die die Koordinaten aller im Modell vorhandenen Eckpunkte sortiert nach x-, y- und z-Koordinaten enthalten. Da die Bindungs- und Angebotsflächen in den meisten Fällen parallel zu einer der drei Koordinatenebenen sind, lassen sich die entsprechenden Koordinaten einfach bestimmen. Die Koordinaten müssen dann noch sortiert werden, so daß ihre Abfolge dem Umriß der Fläche entspricht.

Die Normale zu einer Angebotsfläche sollte in Richtung der erfüllenden Bindungsfläche und damit nach außen zeigen, für eine Bindungsfläche sollte sie von der Angebotsfläche weg und damit nach innen zeigen (siehe Abbildung 3.1). Falls die Normale nicht eingegeben sondern vom System berechnet werden soll (durch Angabe von <calc-normal> in der Constraintdatei), müssen die ersten drei Eckpunkte beim Blick in Richtung der Normalen im Uhrzeigersinn angeordnet sein. Dies kann bei nichtkonvexen Flächen eine Umordnung der Reihenfolge der Eckpunkte nötig machen.

Die Eingabe einer Objektausrichtung erfolgt durch zwei Vektoren. Der erste gibt die Richtung im lokalen Koordinatensystem des Objekts an, der zweite die Richtung im Weltkoordinatensystem.

4.6 Zwei interessante Routinen

Dieser Abschnitt beschreibt zwei Routinen, die zwar nicht direkt mit dem Constraint-System von IConS zusammenhängen aber trotzdem die Benutzerfreundlichkeit des Systems wesentlich erhöhen.

4.6.1 Bestimmung einer Rotationsachse

Bei der Suche nach erfüllbaren Flächenbindungen muß die betrachtete Bindungsfläche so ausgerichtet werden, daß die Normalen auf der Bindungs- und der Angebotsfläche identisch sind (siehe dazu Abbildung 3.1). Wenn dies nicht bereits der Fall ist, muß das Objekt, zu dem die Bindungsfläche gehört, rotiert werden.

Möchte man einen Richtungsvektor $\vec{x}$ durch Rotation um eine Achse $\vec{r}$ in einen Richtungsvektor $\vec{y} \neq \vec{x}$ überführen, so kann man die Richtung Rotationsachse $\vec{r}$ beliebig innerhalb der von $\vec{x} \times \vec{y}$ und $\vec{x} + \vec{y}$ aufgespannten Mittelebene M wählen (wenn $\vec{x}$ und $\vec{y}$ parallel sind, ist M die Ebene, zu der $\vec{x}$ senkrecht ist, siehe auch Abbildung 4.5).

Der Grund dafür ist, daß die Orientierung eines Objekts relativ zu einem anderen durch die Parallelität eines Paars von Normalen nicht vollständig bestimmt ist. Die Objekte können immer noch um die Normale rotiert werden, ohne daß die Bedingung der Parallelität verletzt wird. Durch die Wahl der Rotationsachse $\vec{r}$ wird implizit auch eine Rotation um diese Normale ausgewählt. Um dem Benutzer das Arbeiten mit dem System zu erleichtern, sollte die Achse so gewählt werden, daß sich die Objekte beim Ausrichten so verhalten, wie sie es auch in Natur tun würden.

Naturgetreues Verhalten

Für ein Objekt, das unter dem Einfluß der Schwerkraft ausgerichtet wird, kann man eine Rotationsachse verwenden, die senkrecht zur Schwerkraft und damit senkrecht zur z-Achse des WKS der Szene ist. Durch diese Wahl findet keine Rotation um die z-Achse statt, die eine Rotation um eine Achse parallel zur Schwerkraft wäre.

Diese gilt allerdings nicht für Objekte, die wie Bilder oder Spiegel an der Wand hängen. Ihre Ausrichtung erfolgt nicht unter dem direkten Einfluß der Schwerkraft. Verwendet man auch hierfür eine Rotationsachse, die senkrecht zur z-Achse ist, so kommt es auch hier zu ungewollten Rotationen. Für solche Objekte empfiehlt es sich deshalb, eine passende Rotationsachse durch die Objektausrichtung vorzugeben.

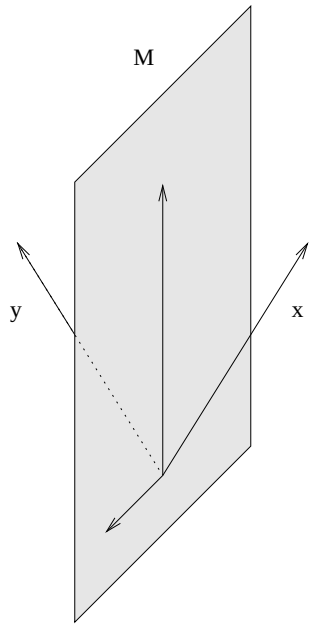

Abbildung 4.5: Mittelebene M zwischen zwei Vektoren. Um den Vektor $\vec{x}$ in den Vektor $\vec{y}$ durch eine Rotation zu überführen, kann eine beliebige Rotationsachse innerhalb von M, die denselben Ursprung wie die beiden Vektoren hat, verwendet werden.

4.6.2 Positionierung von Objekten

Wenn ein neues Objekt durch Laden oder Instantiieren in die Szene eingefügt wird, hat der Benutzer die Möglichkeit, eine Position anzugeben, an der es plaziert werden soll. Das System versucht dann einen Platz in der Nähe dieser Position zu finden, an dem das Objekt unter Berücksichtigung von Kollisionsvermeidung und Objektausrichtung positioniert werden kann.

Dazu markiert der Benutzer durch Klicken in die Szene eine Position, an die das Objekt gesetzt werden soll. Trifft der dabei erzeugte Mausstrahl auf ein Objekt, so wird das Zentrum des neuen Objekts an diese Stelle gesetzt und das Objekt anhand einer eventuell vorhandenen Objektausrichtung ausgerichtet. Falls das Objekt an dieser Stelle eine Kollision verursacht, wird es in kleinen Schritten nach oben (in Richtung der z-Achse) angehoben, bis es nicht mehr kollidiert.

Wenn kein Objekt in der Szene vom Mausstrahl geschnitten wird, muß der Abstand des neuen Objekts von der Kamera auf andere Weise bestimmt werden. Dabei wird die Tatsache ausgenutzt, daß zur Darstellung der Szene zwei *clip planes* angegeben werden müssen, die den minimalen und maximalen Abstand von der Kamera definieren, in dem Objekte in der Szene gezeichnet werden. Diese *clip planes* werden zur Initialisierung des *z-Buffers* benötigt, mit dem das System feststellt, welche Objekte von anderen verdeckt werden. Das einzufügende Objekt wird in diesem Falle so positioniert, daß sich sein Zentrum in der Mitte zwischen den beiden clip planes und damit etwa im Zentrum der Szene befindet. Das Objekt wird dann genauso wie im ersten Fall ausgerichtet und in eine kollisionsfreie Position gebracht.

Nach diesen Schritten befindet sich das Objekt in einer Position, die die Konsistenz der Szene nicht beeinträchtigt. Danach wird noch der Algorithmus zur Erfüllung von Flächenbindungen (siehe Abschnitt 3.9.2) auf das Objekt angewendet.

4.7 Numerische Probleme

Bei der Programmierung von IConS spielten numerische Ungenauigkeiten auf den unterschiedlichsten Ebenen eine große Rolle. Dieser Abschnitt enthält Beispiele für daraus resultierende Probleme

auf der Programmier- und auf der Benutzerebene.

4.7.1 Programmierebene

So gilt zum Beispiel für das Skalarprodukt zweier Vektoren

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos \phi,$$

wobei ϕ der zwischen $\vec{a}$ und $\vec{b}$ eingeschlossene Winkel ist. Verwendet man für $\vec{a}$ und $\vec{b}$ Einheitsvektoren, so gilt

$$-1 \leq \vec{a} \cdot \vec{b} = \cos \phi \leq 1.$$

Aufgrund numerischer Fehler kann es vorkommen, daß diese Ungleichung falsch wird. Bei der Anwendung der Arcuscosinus-Funktion auf einen Wert $x > 1$ oder $x < -1$ kommt es dann zu einem Fehler, der durch den vorherigen Test auf $-1 \leq x \leq 1$ abgefangen werden muß.

4.7.2 Benutzerebene

Neben Fehlern, die für den Benutzer unsichtbar sind, solange sie vom System abgefangen werden können, gibt es auch Fehler, die die Arbeit des Benutzer direkt beeinflussen. In Abschnitt 4.4.3 wurde beschrieben, wie die Bewegungsfreiheit von Objekten durch numerische Fehler eingeschränkt wird. Dieses Problem kann durch eine passende Definition der Bindungsflächen umgangen werden.

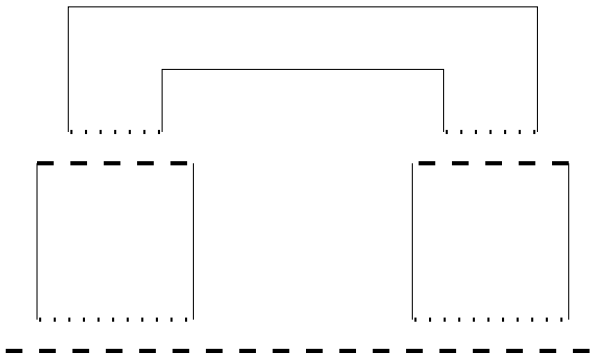

Abbildung 4.6: Brücke mit numerischem Problem. Werden die beiden identischen Blöcke durch erfüllte Flächenbindungen auf den Boden gestellt, so sollte es auch kein Problem mehr sein, die beiden Flächenbindungen der Brücke zu erfüllen, indem sie auf die Blöcke gesetzt wird. Die ist allerdings aufgrund numerischer Probleme bei der Abstandsberechnung zwischen Angebots- und Bindungsflächen nicht möglich.

Ein weiterer Problemfall ist in Abbildung 4.6 zu sehen: Aufgrund der Geometrie der Objekte und der Anordnung der Angebots- und Bindungsflächen dürfte es kein Problem sein, die Flächenbindungen zwischen den Blöcken und dem Boden und zwischen der Brücke und den Blöcken zu erfüllen. Durch numerische Fehler bei der Berechnung der Abstände zwischen Angebots- und Bindungsflächen an der Brücke ist es aber nicht möglich, beide Enden der Brücke auf die Blöcke aufzusetzen.

Kapitel 5

Ergebnisse

Dieses Kapitel ist in drei Abschnitte unterteilt: Im ersten Abschnitt werden die Ergebnisse eines Benutzertests mit IConS vorgestellt. Danach wird das Arbeiten mit dem System anhand einer Reihe von Bildschirmausgaben demonstriert und abschließend auf Erweiterungsmöglichkeiten hingewiesen.

5.1 Benutzertest

Im Laufe der Arbeiten am IConS-System wurde ein kleiner Benutzertest mit einer vorläufigen Version durchgeführt, die von der hier beschriebenen Version hauptsächlich in drei Punkten abwich: Die Flächenbindungen waren beim Laden von Objekten noch nicht aktiv, es gab noch keinen Objektausrichtungs-Constraint und die Heuristik für die Selektion beim Rotieren verhielt sich anders (siehe dazu auch Abschnitt 5.1.5).

Der Benutzertest beschäftigte sich mit zwei Fragestellungen:

- Inwieweit hilft das implementierte Constraintsystem den Testpersonen bei der Erstellung einer Szene?
- Wie hilfreich oder hinderlich ist das gewählte Rotationsinterface für das Arbeiten mit dem System?

Der Aufbau des Tests orientierte sich an einer von Gomoll entwickelten Methodik [Gom90]. Sie wurde mit dem Ziel entwickelt, die zukünftigen Benutzer in allen Stadien des Entwicklungsprozesses eines Programms miteinzubeziehen. Dieser Test entspricht nicht unbedingt strengen, wissenschaftlichen Kriterien, kann aber nützliche Hinweise auf Stärken und Schwächen eines Programms geben.

Durch die Verwendung von Tastenkürzeln und die Eingabe von Dateinamen in einem separaten Fenster ist die IConS-Benutzerschnittstelle nicht intuitiv bedienbar. Bei der Durchführung des Tests wurde deshalb an Stellen, an denen die Testperson nur mit der eigentlichen Benutzerschnittstelle Probleme hatte, Hilfestellung geleistet.

Die folgenden Abschnitte geben einen Überblick über die Durchführung eines Benutzertests und beschreiben die Aufgabenstellung. Danach wird der Ablauf der beiden Tests beschrieben und die Ergebnisse zusammengefaßt. Der letzte Abschnitt beschreibt die Veränderungen, die aufgrund dieser Ergebnisse am System vorgenommen wurden.

5.1.1 Durchführung

Die Durchführung des Benutzertests kann in folgende Schritte unterteilt werden:

1. *Allgemeine Beschreibung des Tests*: Dabei wird ausdrücklich betont, daß das Ziel ist, Fehler und Ungereimtheiten im System zu finden und nicht die Testperson zu testen. Außerdem kann der Test von der Testperson jederzeit abgebrochen werden.

Die Testperson hat dadurch die Freiheit, ehrlich zu sein und gefundene Fehler auch anzugeben. Sie unterliegt nicht dem Zwang, den Test zu Ende führen zu müssen.

2. *Vertrautmachen mit der Testumgebung*: Das Ziel des IConS-System, durch Umsetzung physikalischer Eigenschaften von Objekten in Regeln das Arbeiten mit dem Modell einer Szene zu erleichtern, wird erläutert. Dabei wird nicht auf die Art und Funktion der Regeln eingegangen.

   Außerdem wird der Aufbau des Bildschirmfensters, die drei Modi des Systems und die Verwendung der Maustasten zur Auswahl der Operation erläutert. Die Testperson erhält ein Blatt mit den IConS-Befehlen (Tabellen 4.1 und 4.2) und einer Liste der verfügbaren Objekte.

3. *Erläuterung des „Laut-Denkens“*: Die Testperson wird gebeten, während des Tests laut zu denken und auftretende Fragen zu stellen, obwohl die Fragen normalerweise erst nach Ende des Tests beantwortet werden können.

4. *Ausgabe der schriftlichen Aufgabenstellung* (siehe Abschnitt 5.1.2 und Anhang A): Eventuell vorhandene Fragen werden noch beantwortet.

5. *Bearbeiten der Aufgabenstellung*:

6. *Abschluß des Tests*: Hintergrund und Ziele des Tests werden erläutert und eventuelle Fragen beantwortet.

5.1.2 Aufgabenstellung

In Anhang A sind die für den Benutzertest verwendeten Aufgaben abgedruckt. Die beiden ersten Aufgaben sollen dem Benutzer die Einarbeitung in das System ermöglichen. In dieser Phase wurden deshalb auch alle Fragen zur reinen Bedienung des Systems beantwortet.

Die dritte Aufgabe zielt auf die Erstellung einer Szene unter Verwendung der Flächenbindungen ab. Sie kann ohne Benutzung des Rotationsinterfaces erstellt werden.

Aufgabe vier basiert auf dem Ergebnis der dritten Aufgabe und erfordert die Verwendung des Rotationsinterfaces.

5.1.3 Ablauf der Benutzertests

An dem Test nahmen zwei Testpersonen teil, die im folgenden als Benutzer A und Benutzer B bezeichnet werden. Benutzer A arbeitete während des Tests zum ersten Mal mit dem IConS-System und hatte auch noch nie mit einem anderen Modelliersystem gearbeitet. Benutzer B hatte bereits etwas Erfahrung mit CAD-Systemen und hatte bereits eine ältere Version des IConS-Systems gesehen. In Tabelle 5.1 ist die Bearbeitungsdauer für die einzelnen Aufgaben angegeben.

Aufgabe	**Benutzer A**	**Benutzer B**
1. und 2.	31 min.	19 min.
3.	22 min.	11 min.
4.	4 min. (*)	8 min.

Tabelle 5.1: Bearbeitungsdauer für die einzelnen Aufgaben des Benutzertests. Die mit (*) markierte Aufgabe mußte wegen einer vom Programm nicht abgefangenen Fehleingabe abgebrochen werden.

Aufgaben 1. und 2.

Benutzer A mußte sich in dieser Phase erst mit dem System vertraut machen. Besondere Schwierigkeiten bereitete dabei das Verständnis der drei Modi des Systems. Benutzer B fiel der Einstieg wesentlich leichter und er startete sehr bald mit der Erkundung der Funktionsweise der Constraints und einzelner Systemteile wie dem Löschen von Objekten.

Beiden Benutzern fiel auf, daß die Flächenbindungen beim Laden eines Objekts noch nicht aktiv sind. Die dadurch auftretenden Kollisionen beim Verschieben und die fehlende Gruppierung verwirrten Benutzer A. Benutzer B hatte das Problem, daß ein Stuhl durch die Flächenbindungen nicht, wie von ihm gewünscht, zum Tisch sondern zum Schreibtisch gruppiert wurde, da er nicht ganz vor dem Tisch stand.

Aufgabe 3.

Benutzer A versuchte die Szene von hinten nach vorne aufzubauen. Jedes geladene Objekt wurde sofort positioniert. Der Benutzer beherrschte an dieser Stelle zwar den Navigationsmodus, hatte aber große Probleme mit der Verwendung des Rotationsinterface zur Rotation der ganzen Szene. Der Unterschied zwischen freiem und eingeschränktem Interaktionsmodus war nicht klar.

Die Lampe konnte nicht auf den Schreibtisch gestellt werden, da entweder der Lampenschirm mit dem Aufbau des Schreibtischs kollidierte oder die Lampe auf den Boden herabfiel, weil in der gewählten Ansicht nicht klar war, ob sich die Lampe über dem Schreibtisch befindet oder nicht. Trotz des Hinweises, daß die Positionierung in der Draufsicht einfacher ist, konnte die Lampe nicht auf den Schreibtisch gestellt werden.

Benutzer B hat zuerst alle Objekte in die Szene hereingeladen und positionierte sie danach. Die Erstellung der Szene verlief flüssig, aber auch er hatte große Probleme mit der Positionierung der Lampe auf dem Schreibtisch.

Aufgabe 4.

Der Umbau der Szene und die damit verbundenen Rotationen von Tisch und Wand um eine festgelegte Achse bereiteten Benutzer A keine Probleme. Das Programm wurde dann aufgrund einer nicht abgefangenen Fehleingabe beim Laden eines Objekts unbedienbar und der Test mußte abgebrochen werden.

Benutzer B hatte Probleme mit dem Aufhängen der Wandtafel. In der gewählten Ansicht war nicht klar, daß sich die Tafel nicht vor der Wand befand. Er versuchte deshalb, die Tafel parallel zur Wand auszurichten und hatte damit auch keinen Erfolg. Nachdem die Ansicht gewechselt wurde, hatte sich das Problem erledigt. Die Positionierung der Bücher auf dem Regal des Schreibtischs unterblieb aus Zeitgründen.

5.1.4 Gewonnene Ergebnisse

Die in diesem Versuch gewonnenen Ergebnisse lassen sich in konzeptuelle und implementierungsspezifische Ergebnisse unterteilen. Außerdem erfolgt noch eine generelle Beurteilung anhand von allgemeinen Kriterien für Benutzeroberflächen.

Konzeptuelle Ergebnisse

Aus konzeptueller Sicht hat sich das Constraintsystem als sehr hilfreich erwiesen. Allerdings kann es durch die gewählte Ansicht schwierig sein, die räumliche Anordnung von Objekten zu erkennen. Dadurch kommt es auch zu unerwarteten Effekten bei der Erfüllung von Flächenbindungen.

Das Rotationsinterface bereitete keine Probleme, wenn die Rotationsachse durch eine erfüllte Flächenbindung feststand. Die freie Rotation war aber besonders bei kleinen Objekten schwierig. Die Heuristik zur Selektion des rotierten Objekts war ebenfalls bei kleinen Objekten für die Benutzer nicht ganz vorhersehbar.

Implementierungsspezifische Ergebnisse

Die Eingabe von Tastenkürzeln und Dateinamen bereitete erwartungsgemäß große Probleme. Weitere Punkte sind die beim Laden noch nicht aktiven Flächenbindungen, die Markierung der gruppierten Objekte beim Bearbeiten, die eine Unterscheidung der Objekte unmöglich macht, und der Stop der Rotationseingabe, nachdem eine Flächenbindung erfüllt worden ist.

Bewertung der Benutzeroberfläche

Die Bewertung der Benutzeroberfläche kann anhand von fünf Kriterien erfolgen [FvDFH96]. Diese sind in der folgenden Aufzählung zusammen mit einer kurzen Bewertung für das IConS-System genannt. Eine wirklich aussagekräftige Bewertung läßt sich nur durch wesentlich aufwendigere Versuche gewinnen.

- *Lerngeschwindigkeit*: Beide Benutzer konnten erstaunlich schnell mit dem System umgehen. Eine nähere Erläuterung der Arbeitsweise der Flächenbindungen kann den Lernprozeß sicher noch beschleunigen.
- *Arbeitsgeschwindigkeit*: Beide Benutzer erreichten nach Ende der Einarbeitungszeit eine gute Arbeitsgeschwindigkeit. Bei der Bearbeitung der Aufgaben entfiel auch einiger Aufwand auf das Laden der Objekte.
- *Fehlerrate*: Die Benutzer machten noch einige Fehler. Mit zunehmender Erfahrung mit dem System nimmt deren Anzahl aber ab. Trotzdem werden Aufgaben wie das Positionieren der Lampe auf dem Schreibtisch schwierig bleiben.
- *Geschwindigkeit beim Wiedereinstieg*: Die Frage, wie schnell ein Benutzer nach längerer Pause wieder mit dem System arbeiten kann, kann mit diesem Versuch nicht beantwortet werden.
- *Attraktivität*: Der spontane Kommentar eines Benutzers am Ende des Tests war: "Es hat Spaß gemacht." Die wirkliche Attraktivität einer Benutzeroberfläche kann aber nur in vergleichenden und detaillierten Tests festgestellt werden.

5.1.5 Veränderungen am System

Nach Abschluß des Benutzertests wurden noch drei wesentliche Veränderungen am System vorgenommen:

- *Aktivierung der Flächenbindungen beim Einfügen eines Objekts*: Im Testsystem wurde beim Einfügen der Algorithmus aus Abschnitt 4.6.2 verwendet, ohne nach Abschluß der Positionierung nach erfüllbaren Flächenbindungen zu suchen. Dadurch wurden die neuen Objekte zwar in die Szene eingefügt aber zur Aktivierung der Flächenbindungen mußte jedes Objekt nochmals im eingeschränkten Interaktionsmodus angeklickt werden.

  Durch die Aktivierung der Flächenbindungen verhalten sich die Objekte jetzt beim Einfügen in eine Szene und beim Verschieben innerhalb der Szene gleich und werden bereits beim Einfügen exakt positioniert und gruppiert. Das Positionieren der Lampe auf dem Schreibtisch wird dadurch zumindest beim Einfügen der Lampe sehr erleichtert.
- *Einführung des Objektausrichtungs-Constraints*: Dieser Constraint war im damaligen System nicht vorhanden. Aufgrund der Erfahrungen beim Positionieren der Wandtafel und wegen grundsätzlicher Überlegungen wurde dieser Constraint eingeführt.
- *Veränderung der Heuristik zur Selektion bei Rotationen*: Die damals implementierte Heuristik hatte keine Untergrenze für die Größe der selektierten Objekte auf dem Bildschirm und wählte immer das kleinste Objekt aus.

  Durch Einführung einer Untergrenze sowie eine Verbesserung des Umgangs mit Objekten, deren Zentrum nicht sichtbar war, wurde die Heuristik berechenbarer und bedienerfreundlicher.

5.2 Beispielinteraktionen

Die folgenden Abbildungen zeigen das Arbeiten mit IConS anhand eines Beispiels. Beim Start des Systems erhält der Benutzer eine leere Szene (Abbildung 5.1). In den Abbildungen 5.2 bis 5.6 werden Objekte zu der Szene hinzugefügt und die Szene von verschiedenen Ansichten aus betrachtet. Nachdem die Objekte in der Szene verschoben worden sind (Abbildung 5.7), werden in den Abbildungen 5.8 bis 5.10 zwei Spezialfälle beim Laden und Instantiieren von Objekten betrachtet. Die Abbildungen 5.11 und 5.12 zeigen den Bruch einer Flächenbindung und die Ersetzung eines Objekts. In den Abbildungen 5.13 und 5.14 werden die Veränderungen durch das Löschen eines Objekts gezeigt und in den Abbildungen 5.15 bis 5.17 das Arbeiten mit eingeschalteter Pseudo-Gravitation demonstriert. Abschließend ist in Abbildung 5.18 das Arbeiten mit dem Rotationsinterface dargestellt.

In den Bildunterschriften zu den einzelnen Abbildungen wird der Inhalt der Abbildung näher erläutert und auf Besonderheiten eingegangen.

5.3 Erweiterungsmöglichkeiten

Aus den letzten beiden Abschnitten und den Erfahrungen mit dem IConS-System ergeben sich eine Reihe von Erweiterungs- und Verbesserungsmöglichkeiten. Dabei sollte das generelle Ziel die Beschleunigung und Vereinfachung der Benutzung sein.

Anpassung der Benutzeroberfläche

Bei der Besprechung des Benutzertests wurde bereits auf die Probleme der Benutzer mit der Steuerung des Programms über die Tastatur hingewiesen. Die Integration einer graphischen Benutzeroberfläche mit Schaltflächen, Menüs und Dialogen könnte die Bedienung des Systems durch unerfahrene Benutzer wesentlich erleichtern.

Aufbau einer Bibliothek von Objektmodellen

Mit dem Aufbau einer Bibliothek von Objektmodellen ergeben sich mehr Möglichkeiten zur Ausgestaltung eines Raumes. Gleichzeitig ist dies aber auch ein Test für den Bezeichnungsmechanismus der Flächenbindungen und kann zu Veränderungen Anlaß geben. Neben einer größeren Anzahl von Bezeichnungen kann es auch sinnvoll sein, die Ordnung der Bezeichnungen aus Abschnitt 3.2 zu verändern oder – falls die Ordnung sonst zu komplex werden würde – mehrere Bezeichnungen für eine Fläche zuzulassen.

Interaktive Veränderung der Flächenbindungen

Die Flächenbindungen sind ein Versuch, das „natürliche“ Verhalten eines Objekts zu charakterisieren. Da dies aber niemals vollständig gelingen kann und der Benutzer manchmal ein Objekt auch in einem anderen Zusammenhang verwenden möchte, sollte es möglich sein, einzelne Angebots- und Bindungsflächen zu deaktivieren oder ihre Bezeichnungen interaktiv zu verändern.

Umsetzen von Objekten

Möchte man die Lampe von einem Tisch auf einen anderen und damit von einer Position mit einer erfüllten Flächenbindung an eine andere Position mit einer erfüllten Flächenbindung umsetzen, so ist das nur durch Aufbrechen der Flächenbindung im freien Interaktionsmodus möglich. Dazu gibt es mehrere Alternativen. Zum einen könnte eine Flächenbindung aufgebrochen werden, sobald der Benutzer ein Objekt weit genug von einer erfüllten Position weggezogen hat. Im Beispiel würde die Lampe dann aber wahrscheinlich auf den Fußboden unter den Tischen fallen und müßte in einem zweiten Schritt wieder auf den Tisch gestellt werden.

Dies könnte vermieden werden, indem die erfüllten Flächenbindungen temporär deaktiviert werden und erst wieder durch ein Angebot mit derselben oder einer höheren Bezeichnung (im Sinne der Ordnung der Bezeichnungen aus Abbildung 3.2) aktiviert werden. Eine weitere Möglichkeit ist die Einführung einer Funktion, mit der Objekte unter Verwendung der Positionierungsfunktion beim Laden (siehe Abschnitt 4.6.2) versetzt werden können.

Umbau des Szenengraphen

Die Existenz von Objekten mit mehreren erfüllten Flächenbindungen erfordert nach Abschnitt 3.7.2 den Umbau des Szenengraphen. Dadurch wird der Zusammenhang zwischen Bindungs- und Szenengraph weitgehend zerstört. Tatsächlich ist ein Umbau aber nur dann notwendig, wenn der Bindungsgraph einen Knoten mit ungeordneten Vorgängern besitzt (siehe 3.6.2). Enthält der Bindungsgraph aber keine ungeordneten Vorgänger, so kann der Szenengraph analog dazu aufgebaut werden, ohne daß die Funktionalität der dynamischen Gruppierung verlorengeht.[1]

[1] Vielen Dank an Marcus Borst für diesen Hinweis.

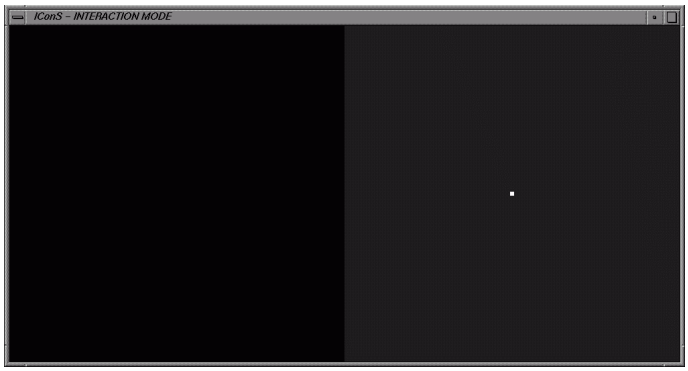

Abbildung 5.1: Ansicht von IConS direkt nach dem Start. Die Szene enthält noch kein Objekt. Ein einzelnes Quadrat in der rechten Hälfte des Fensters zeigt den Wurzelknoten des Szenengraphens. Das System befindet sich im eingeschränkten Interaktionsmodus.

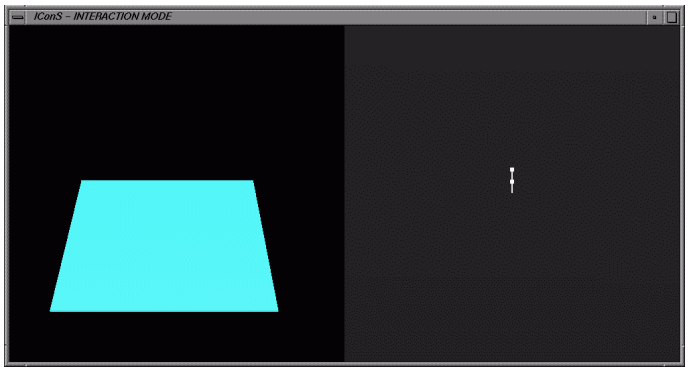

Abbildung 5.2: Der Fußboden wurde geladen und positioniert. Die Positionierung erfolgte mit Hilfe des Algorithmus aus Abschnitt 4.6.2 in der Mitte zwischen den beiden clip planes. Der Szenengraph besteht jetzt aus der Wurzel (oberes Quadrat), der Objektinstanz (unteres Quadrat) und den Objektdaten. Die Objektdaten werden durch eine verkleinerte Ansicht des Fußbodens repräsentiert, die aber nicht sichtbar ist, da sie den Fußboden genau von der Seite her zeigt. In Abbildung 5.3 wurde der Szenengraph vom Benutzer vergrößert und neu positioniert, so daß der Fußboden zu erkennen ist.

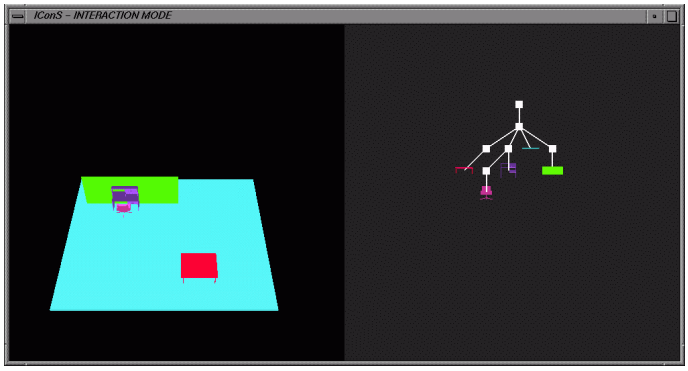

Abbildung 5.3: Weitere Objekte wurden geladen. Die Objekte wurden in der Reihenfolge „von hinten nach vorne“ geladen und mit dem Algorithmus aus Abschnitt 4.6.2 auf dem Fußboden plaziert. Außerdem wurden einige Flächenbindungen erfüllt. Dies ist auch im Szenengraph zu erkennen, der zudem noch vom Benutzer vergrößert und anders positioniert wurde, um den Fußboden auch in der Seitenansicht erkennen zu können (vergleiche dazu auch Abbildung 5.2).

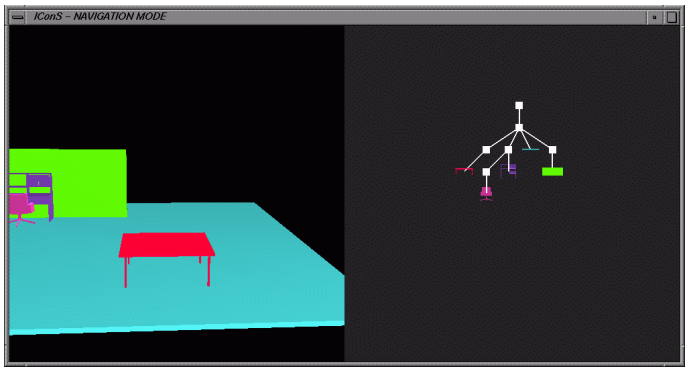

Abbildung 5.4: Arbeiten im Navigationsmodus. Nach einer Umschaltung in den Navigationsmodus wurde die Ansicht auf die Szene verändert, um im nächsten Schritt einen besseren Überblick über diesen Teil der Szene zu haben.

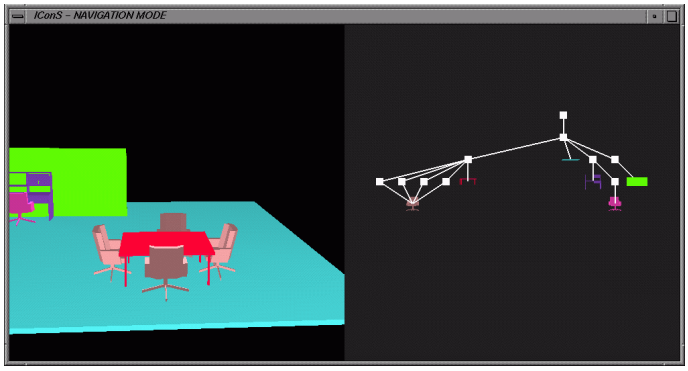

Abbildung 5.5: Laden und Instantiieren von Objekten. Ein Stuhl wurde geladen und an den Tisch gestellt. Danach wurden noch drei weitere Instanzen dieses Stuhls erzeugt und ebenfalls an den Tisch gestellt. Alle vier Stühle haben durch die Instantiierung dieselbe Farbe. Der Szenengraph zeigt auch, daß es sich um vier Instanzen handelt, die dieselben Objektdaten haben. Das System befindet sich immer noch im Navigationsmodus.

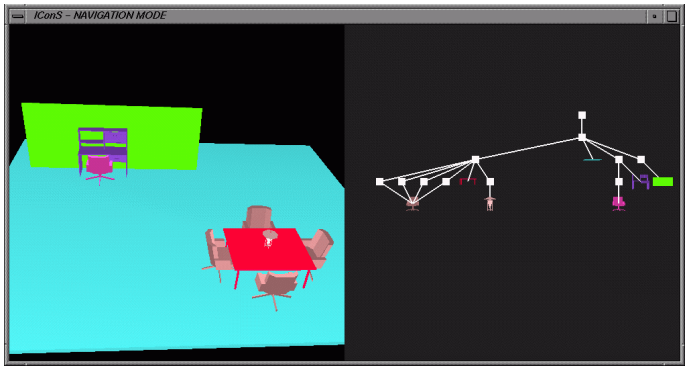

Abbildung 5.6: Hinzufügen einer Lampe und wechseln der Ansicht.

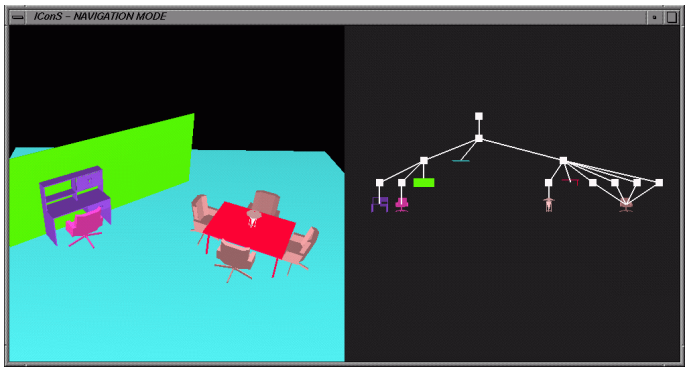

Abbildung 5.7: Bewegen von Objekten. Nach einer Umschaltung in den eingeschränkten Interaktionsmodus wurden die Wand und der Tisch verschoben und rotiert. Alle Objekte, die direkt oder indirekt mit diesen Objekten gruppiert waren, wurden dabei mitverschoben. Im Falle der Wand mußte hierzu der Szenengraph, wie in Abschnitt 3.7.2 beschrieben, umgebaut werden. Alle zur Wand gruppierten Objekte wurden direkte Nachfolger der Wand im Szenengraphen. Danach wurde die Ansicht im Navigationsmodus verändert.

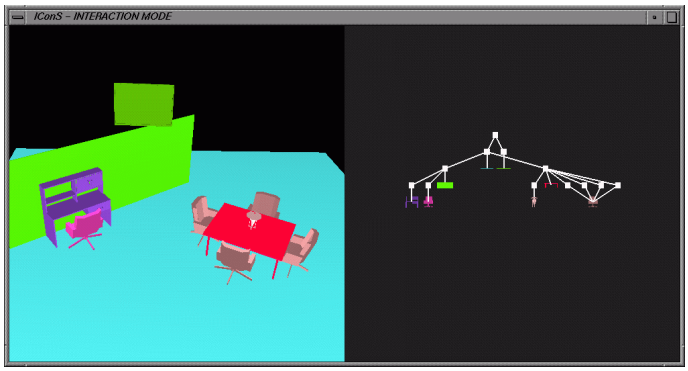

Abbildung 5.8: Laden der Wandtafel. Beim Positionieren wurde auf die Wand geklickt. Da es dadurch zu einer Kollision zwischen Wand und Tafel kam, wurde die Tafel nach oben verschoben. Dadurch war es aber nicht möglich, eine Flächenbindung zwischen Wand und Tafel zu erfüllen.

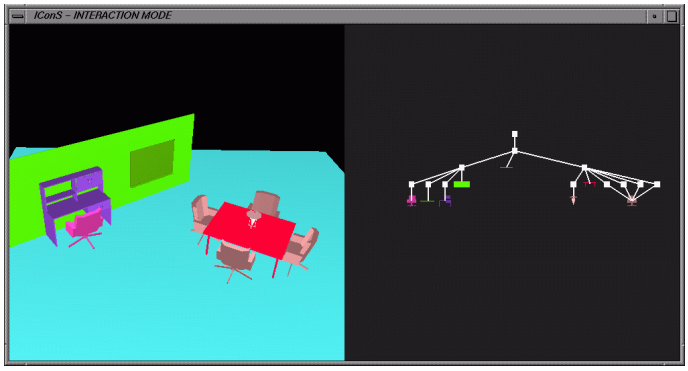

Abbildung 5.9: Positionieren der Wandtafel an der Wand. Durch Verschieben der Tafel vor die Wand wird die Flächenbindung zwischen Wand und Tafel erfüllt.

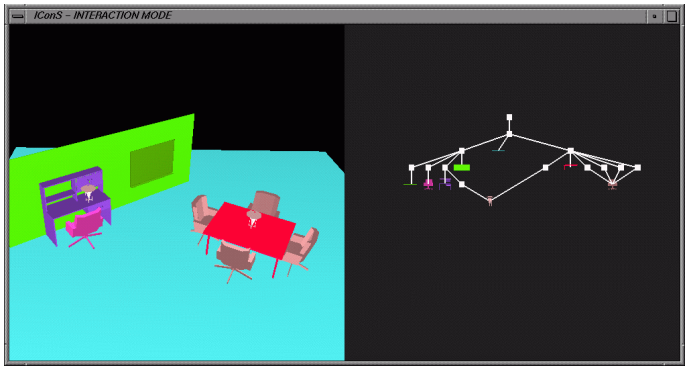

Abbildung 5.10: Instantiieren einer Lampe und Positionierung auf dem Schreibtisch. Diese Operation bereitete während des Benutzertests noch erhebliche Schwierigkeiten (siehe Abschnitt 5.1.3). Durch die Aktivierung der Flächenbindungen bereits beim Einfügen eines Objekts konnte diese Aufgabe wesentlich vereinfacht werden.

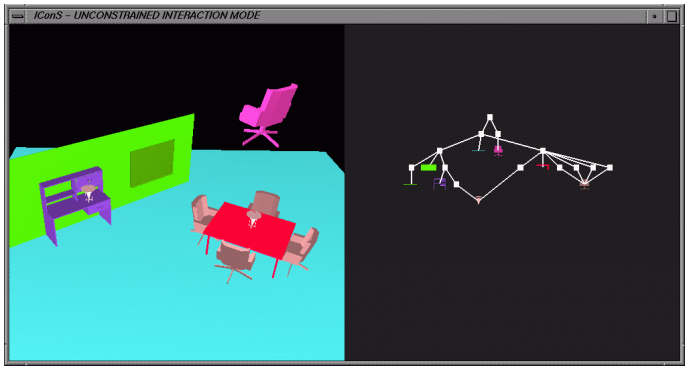

Abbildung 5.11: Bruch einer Flächenbindung und Bewegen eines Objekts im freien Interaktionsmodus. Die Flächenbindungen zwischen Stuhl und Schreibtisch und zwischen Stuhl und Fußboden wurden im freien Interaktionsmodus aufgebrochen. Dadurch ist der Stuhl im Szenengraphen ein direkter Nachfolger der Wurzel geworden. Er kann nun frei in der Szene bewegt werden.

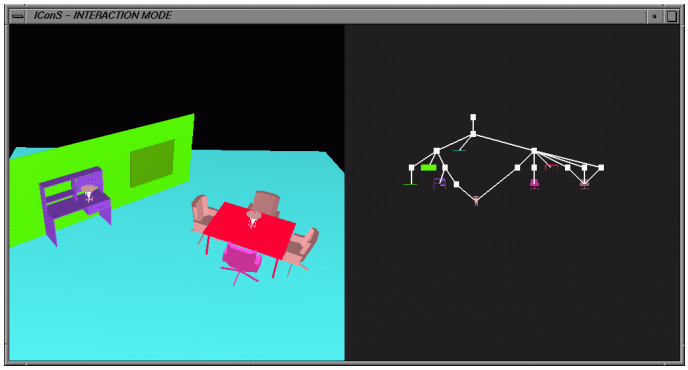

Abbildung 5.12: Ersetzen eines Stuhls durch einen anderen. Der vordere Stuhl am Tisch wurde gelöscht und dann im eingeschränkten Interaktionsmodus durch den in Abbildung 5.11 vom Schreibtisch entfernten Stuhl ersetzt. Man kann auch im Szenengraphen erkennen, daß nun nur noch drei Instanzen des Stuhls zum Tisch gruppiert sind.

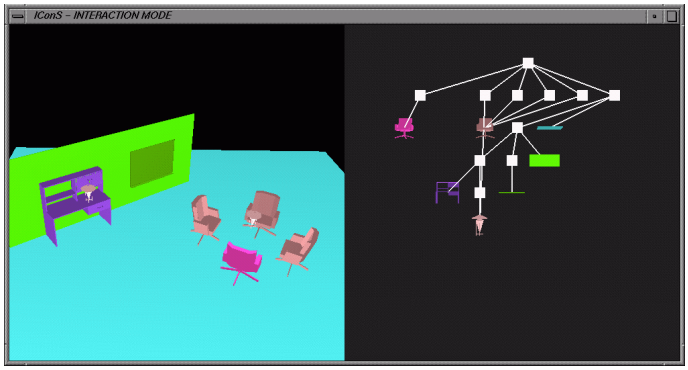

Abbildung 5.13: Löschen des Tischs. Durch das Löschen des Tischs werden auch alle Flächenbindungen, die zwischen dem Tisch und anderen Objekten in der Szene existierten, gelöscht. Dabei werden andere Objekte, wie zum Beispiel die Lampe, nicht bewegt sondern können in der Luft schwebend verharren. Alle Objekte, die Nachfolger des Tischs im Szenengraph waren (siehe Abbildung 5.12), sind nun direkte Nachfolger der Wurzel, auch wenn sie eventuell, wie die Stühle, noch über erfüllte Flächenbindungen verfügen. Graphplace hat Probleme mit der formatierten Ausgabe dieses Szenengraphen. Leider ist es nicht möglich, die Art der Formatierung zu beeinflussen und eine bessere Ausgabe zu erreichen.

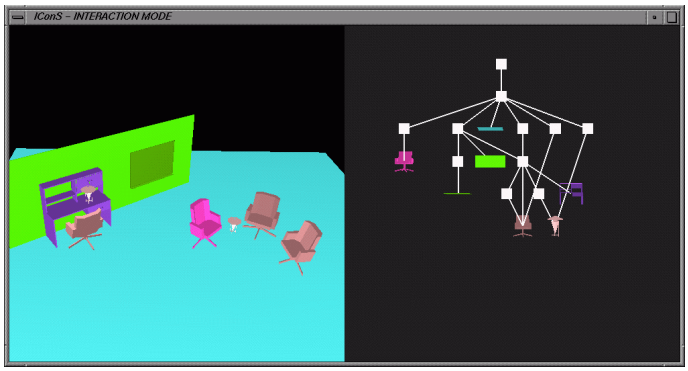

Abbildung 5.14: Bewegen der Objekte. Nachdem die Stühle und die Lampe bewegt und einige Flächenbindungen erfüllt worden sind, sind alle Objekte wieder entsprechend der erfüllten Flächenbindungen im Szenengraph angeordnet.

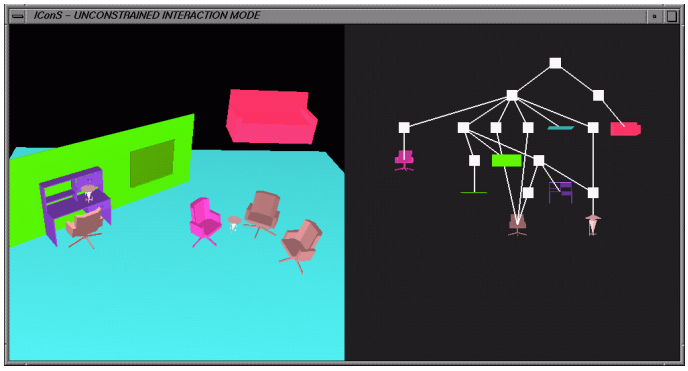

Abbildung 5.15: Laden im freien Interaktionsmodus. Nach dem Laden schwebt die Couch in der Luft. Da sich das System im freien Interaktionsmodus befindet, wurde nicht versucht, eine Flächenbindung zu erfüllen. Die Couch ist deshalb im Szenengraph als direkter Nachfolger der Wurzel eingefügt worden.

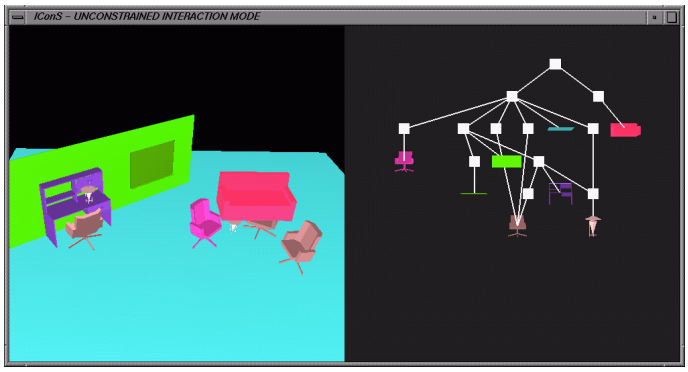

Abbildung 5.16: Aktivieren der Pseudo-Gravitation. Entsprechend dem Algorithmus für die Pseudo-Gravitation (siehe Abschnitt 3.5.1) fällt die Couch am Ende einer Bewegung nach unten. Dabei trifft sie auf einem Stuhl auf und bleibt in einer Position, die physikalisch und auch von der Funktion her nicht sinnvoll ist. Der Szenengraph bleibt bei dieser Operation unverändert.

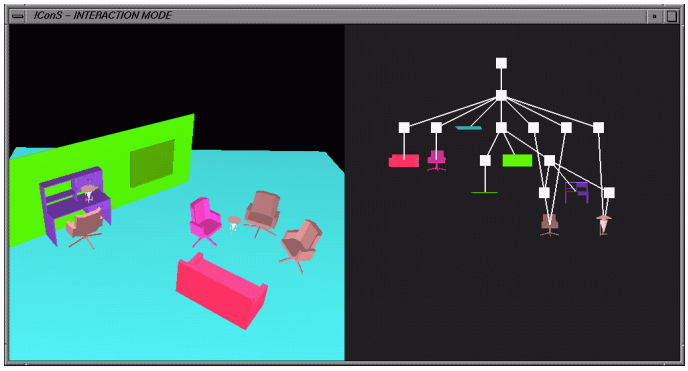

Abbildung 5.17: Bewegen der Couch im eingeschränkten Interaktionsmodus. Die Couch wurde an eine Stelle in der Szene verschoben, an der eine Flächenbindung zwischen der Couch und dem Fußboden zustande kommen konnte.

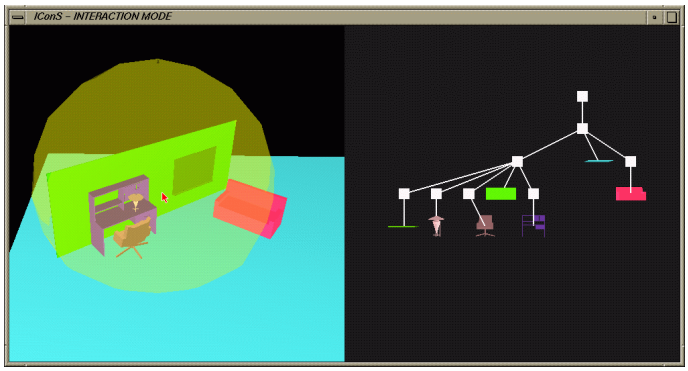

Abbildung 5.18: Rotation der Wand. Durch die Heuristik zur Selektion bei Rotationen wurde die Wand selektiert: Lampe und Couch sind zu weit entfernt. Schreibtisch und Stuhl können wegen erfüllter Flächenbindungen nicht rotiert werden und die Wandtafel wird zusätzlich durch die Objektausrichtung blockiert. Damit bleibt die Wand und der Fußboden übrig, wodurch die Wand als kleineres der beiden Objekte selektiert wird. Als Hilfe für die Bedienung des Arcball-Interfaces wird eine halb-transparente Kugel in die Szene eingeblendet.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Mit der Entwicklung des IConS-Systems konnte gezeigt werden, daß das Arbeiten mit einer Szene stark vereinfacht werden kann, wenn sich die Objekte in der Szene „natürlich“ verhalten. Dieses Verhalten läßt sich auf ein System von Constraints abbilden, das die Verwendung einer physikalischen Simulation von Objektbewegungen überflüssig macht und damit wesentlich allgemeiner und mit geringerem Rechenaufwand realisierbar ist.

Die Erfahrungen, die während eines Benutzertests gesammelt wurden, belegen, daß ein solches System auch von ungeübten Personen vergleichsweise schnell bedient werden kann und daß ein effizientes und exaktes Zusammenstellen einer Szene dadurch wesentlich erleichtert wird. Durch den Benutzertest wurden aber auch eine Reihe von Schwachstellen des Systems deutlich, die zum Teil schon mit geringem Aufwand entfernt werden konnten. Die Durchführung von Benutzertests sollte deshalb ein fester Bestandteil beim Entwurf von Benutzerschnittstellen sein.

6.2 Ausblick

Das Constraint-Konzept, das in dieser Arbeit vorgestellt wird, kann auch in anderen Kontexten verwendet werden, von denen hier drei vorgestellt werden. Dabei ist es sicherlich notwendig, die Constraints an die jeweilige Situation anzupassen.

Große Szenen

Bei der Verwendung der Constraints in einer umfangreichen Szene gibt es zwei Probleme: Zum einen wird die Suche nach erfüllbaren Flächenbindungen sehr bald zu einer starken Verlangsamung des Systems führen, so daß das System nicht mehr interaktiv ist. Zum anderen kann es bei der Erfüllung von Flächenbindungen zu einem sehr unintuitiven Verhalten kommen, wenn Objekte durch Wände und Decken verschoben werden.

Beide Probleme können durch eine Begrenzung der Reichweite der Bindungs- und Angebotsflächen zumindest teilweise gelöst werden. Wenn Wände und Decken als Barrieren für Flächenbindungen wirken, wird das Verhalten intuitiver und die Anzahl der möglichen Bindungspaare reduziert sich drastisch.

Eingabe mit Force-Feedback-Geräten

Ein Eingabegerät ist ein *Force-Feedback*-Gerät, wenn das Gerät eine Kraft (etwa auf die Hand des Benutzers) ausüben kann (siehe [MRF+96]). Diese Art der Rückmeldung vom System an den Benutzer kann die Bedienung eines Modelliersystems erheblich vereinfachen [HPGK94].

Durch die Constraints wird die Bewegungsfreiheit der Objekte in der Szene eingeschränkt. Dies wird bereits vom Arcball-Rotationsinterface ausgenutzt, das es erlaubt, die Rotation eines Objekts auf eine vorgegebene Achse einzuschränken. Mit einem Force-Feedback-Gerät ist es möglich, die Bewegung des Eingabegerätes beliebig einzuschränken. Dem Benutzer kann auf diese Weise das Gefühl vermittelt werden, daß das Eingabegerät fest mit dem Objekt verbunden ist, das er gerade bewegt. Er kann damit die Auswirkungen der Constraints nicht nur sehen sondern auch spüren.

Diese Art von Eingabe verhindert, daß ein Objekt in eine anderes hineingeschoben wird (wie zum Beispiel einen Tisch in die Wand). Dies führt zu einem realistischeren Verhalten der Objekte, das aber auch kontraproduktiv sein kann, wenn etwa ein großes Objekt durch eine schmale Tür bewegt werden soll.

Dreidimensionale Ausgabegeräte

Es gibt bereits eine Reihe von Modelliersystemen, in denen der Benutzer ganz von Projektionswänden umgeben ist oder einen Datenhelm verwendet, um eine dreidimensionale Ausgabe der Szene zu ermöglichen (siehe [BDHO92, Min95]). Der Benutzer kann dann auf verschiedenen Wegen (zum Beispiel Position und Orientierung der Hand, Spracheingabe) mit dem System interagieren.

Durch die Integration der Constraints in ein solches Modelliersystems kann der Benutzer auf sehr natürliche und intuitive Weise mit einer Szene arbeiten.

Anhang A

Materialien zum Benutzertest

A.1 Aufgabenstellung

Bitte lesen Sie sich die einzelnen Aufgabenstellungen genau durch, bevor Sie mit dem Arbeiten beginnen. Denken Sie während des Arbeitens bitte laut darüber nach, was Sie gerade machen und welchen Schritt Sie als nächstes unternehmen wollen. Stellen Sie auch Fragen, obwohl diese erst nach dem Versuch beantwortet werden. Sie haben beliebig viel Zeit für die Aufgaben zur Verfügung.

1. Machen Sie sich mit der Funktionsweise des IConS-Systems vertraut, indem Sie mit den verfügbaren Modellen experimentieren. Sie können Objekte in das System hereinladen, die Objekte verschieden anordnen, sie löschen. Außerdem können Sie sich durch die Anordnung bewegen und diese drehen.

2. Richten Sie sich ein Zimmer nach Ihren Vorstellungen mit den verfügbaren Objekten ein. Das Zimmer braucht keine vier Wände zu haben.

   Wenn Sie fertig sind, speichern Sie das Ergebnis durch Drücken der Taste „w“ unter dem Namen „zimmer.csb“ ab. Beenden Sie das IConS-System mit der Taste „Esc“ (Escape-Taste).

3. Starten Sie das IConS-System erneut durch Eingabe von „project“. Der Versuchsleiter zeigt Ihnen eine Abbildung einer Anordnung (Abbildung A.1). Bauen Sie diese nach.

   Wenn Sie fertig sind, speichern Sie das Ergebnis durch Drücken der Taste „w“ unter dem Namen „anordnung1.csb“ ab.

4. Der Versuchsleiter zeigt Ihnen eine weitere Abbildung einer Anordnung (Abbildung A.2). Bauen sie die Anordnung aus dem vorherigen Versuch in diese Anordnung um.

   Wenn Sie fertig sind, speichern Sie das Ergebnis durch Drücken der Taste „w“ unter dem Namen „anordnung2.csb“ ab. Beenden Sie das IConS-System mit der Taste „Esc“ (Escape-Taste).

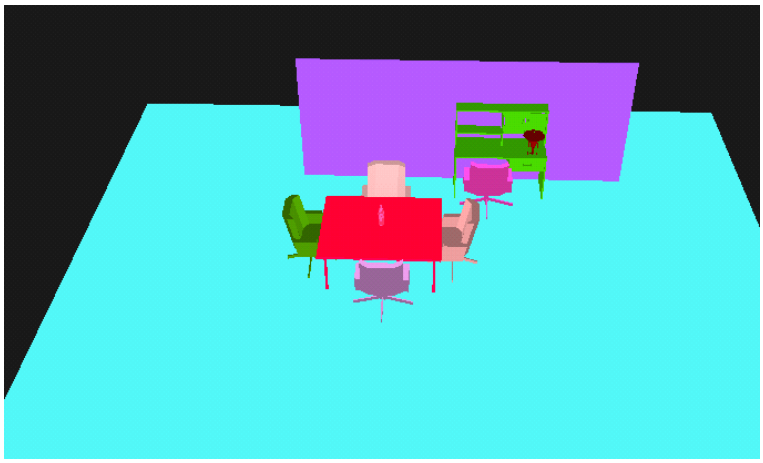

Abbildung A.1: Erste Anordnung für den Benutzertest

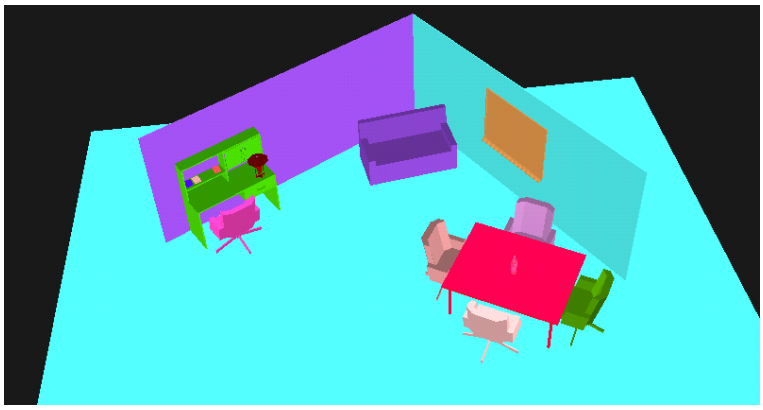

Abbildung A.2: Zweite Anordnung für den Benutzertest

Literaturverzeichnis

[BDHO92] Jeff Butterworth, Andrew Davidson, Stephen Hench, and T. Marc Olano. 3DM: A Three Dimensional Modeler Using a Head-Mounted Display. In *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, pages 135–138, 1992.

[Bie90] Eric A. Bier. Snap-Dragging in Three Dimensions. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, pages 193–204, 1990.

[BS95] Richard W. Bukowski and Carlo H. Sequin. Object Associations: A Simple and Practical Approach to Virtual 3D Manipulation. In *Computer Graphics (1995 Symposium on Interactive 3D Graphics)*, pages 131–138, 1995.

[Buk95] Richard W. Bukowski. The WALKTHRU Editor: Towards Realistic and Effective Interaction with Virtual Building Environments. Master's thesis, University of California, Berkeley, 1995.

[Che98] Stephen Chenney. *Sced: Constraint Based Scene Editing Users Guide.* University of California, Berkeley, 1998.

[Col98] Andy Colebourne. AC3D, 1998. Programm erhältlich im Internet unter der Adresse http://www.comp.lancs.ac.uk/computing/users/andy/ac3d.html.

[DH93] Stéphanie Donikian and Gérard Hégron. A Declarative Design Method for 3D Scene Sketch Modeling. In R.J. Hubbold and R. Juan, editors, *EUROGRAPHICS 93.* Blackwell Publishers, 1993.

[Eck97] George Eckel. *Cosmo 3D Programmers Guide.* Silicon Graphics, Inc., 1997.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: principles and practice.* Addison-Wesley, 1996.

[Gom90] Kathleen Gomoll. Some Techniques for Observing Users. In B. Laurel, editor, *The Art of Human-Computer Interface Design*, pages 85–90. Addison-Wesley, 1990.

[HLC+97] Thomas C. Hudson, Ming C. Lin, Jonathan Cohen, Stefan Gottschalk, and Dinesh Manocha. V-COLLIDE: Accelerated Collision Detection for VRML. In *Proceedings of VRML '97*, 1997.

[Hou92] Stephanie Houde. Iterative Design of an Interface for Easy 3-D Direct Manipulation. In *Proceedings of the ACM Conference on Human Factors in Operating Systems - CHI 92*, pages 135–141, 1992.

[HPGK94] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. Design Hints for Spatial Input, Course Notes – Developing Advanced VR Applications. In *SIGGRAPH 1994*, 1994.

[Min95] Mark R. Mine. ISAAC: A Virtual Environment Tool for the Interactive Construction of Virtual Worlds, Course Notes. In *SIGGRAPH 1995*, 1995.

[MRF+96] William R. Mark, Scott C. Randolph, Mark Finch, James M. Van Verth, and Russel M. Taylor II. Adding Force Feedback to Graphics Systems: Issues and Solutions. In *Computer Graphics (SIGGRAPH 1996)*, pages 447–452, 1996.

[RS90] Joe Rooney and Philip Steadman, editors. *CAD: Grundlagen von Computer Aided Design.* R. Oldenbourg Verlag, 1990.

[Rul98] Keith Rule. Crossroads 3D, 1998. Programm erhältlich im Internet unter der Adresse http://www.europa.com/~keithr/Crossroads/index.html.

[Sho92] Ken Shoemake. ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse. In *Proceedings of Graphics Interface 92*, pages 151–156, 1992.

[Sho94] Ken Shoemake. Arcball Rotation Control. In Paul S. Heckbert, editor, *Graphics Gems*, volume 4, pages 175–192. Academic Press, 1994.

[vE94] Jos van Eijndhoven. Graphplace, 1994. Programm erhältlich im Internet unter der Adresse ftp://ftp.es.ele.tue.nl/pub/down/graphplace.tar.Z.

[Wen97] Leif Wennerberg. *OpenGL Opimizer Programmers Guide: An Open API for Large-Model Visualisation.* Silicon Graphics, Inc., 1997.

[ZHH96] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An Interface for Sketching 3D Scenes. In *Computer Graphics (SIGGRAPH 1996)*, 1996.

Lebenslauf

Dezember 1972	Geburt in Heidenheim an der Brenz
September 1979 bis Juli 1983	Besuch der Grundschule in Steinheim
September 1983 bis Mai 1992	Besuch des Hellenstein-Gymnasiums in Heidenheim
Juni 1992 bis August 1993	Zivildienst im Kindergarten der Lebenshilfe für behinderte Menschen, Heidenheim
seit Oktober 1993	Informatikstudium an der Universität Ulm
August 1997 bis Mai 1998	Austauschstudent an der University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, USA