Konzepte und Methoden zu allgemeinen, physikalisch basierten Animationssystemen auf der Grundlage der Lagrange-Faktoren-Methode

Dissertation

zur
Erlangung des akademischen Grades

Doktor–Ingenieur (Dr.-Ing.)

der Fakultät für Ingenieurwissenschaften
der Universität Rostock

vorgelegt von

Friedrich Wagner, geb. am 22. Mai 1968 in Osnabrück
aus Rostock

Rostock, Januar 2001

Gutachter der Arbeit

- Prof. Dr.-Ing. habil. Dietmar Jackèl
- Prof. Dr.-Ing. habil. Heidrun Schumann
- Prof. Dr.-Ing. Dr.-Ing. E.h. Wolfgang Straßer, Tübingen

Datum der Verteidigung: 20. Juli 2001

There is a theory which states
that if anyone discovers exactly
what the Universe is for and why it is here,
it will instantly disappear and be replaced
by something even more bizarre and inexplicable.
There is another theory which states
that this has already happened.

Douglas Adams (1952-2001)

Für die Betreuung dieser Arbeit möchte ich mich bei Herrn Prof. Dietmar Jackèl bedanken, der sich immer für mich eingesetzt hat und den ich in der gesamten Zeit als einen ausgesprochen menschlichen „Boss“ erfahren habe. Ein ganz herzlicher Dank gebührt auch Frau Prof. Heidrun Schumann, die sich nicht nur als Gutachterin zur Verfügung gestellt hat, sondern auch wertvolle Hinweise zum Gelingen dieser Arbeit beigetragen hat und es durch ihre fröhliche Art unmöglich machte, sich am Institut für Computergraphik *nicht* wohlzufühlen. Für die Übernahme des Gutachtens danke ich auch Herrn Prof. Wolfgang Straßer von der Universität Tübingen.

Einen ganz großen Dank möchte ich auch meinen Kolleginnen und Kollegen zusprechen, die immer hilfreich zur Seite standen und eine Atmosphäre geschaffen haben, in der es sich nicht nur sehr kreativ, sondern auch sehr unterhaltsam arbeiten ließ. Bei Thomas Nocke möchte ich mich dabei für die kritische Durchsicht großer Teile der Arbeit (und anregende philosophische Diskussionen) besonders bedanken.

Selbstständigkeitserklärung

Ich erkläre, daß ich die eingereichte Dissertation selbstständig und ohne fremde Hilfe verfaßt, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Friedrich Wagner Rostock, Juli 2001

Inhalt

Inhaltsverzeichnis **i**

Abbildungsverzeichnis **iv**

1 Einführung **1**

1.1 Hintergrund 1

1.2 Ziel und Aufbau der Arbeit 3

2 Grundlagen der physikalisch basierten Animation **5**

2.1 Verfahren zur Animationserstellung 5

2.2 Physikalisch basierte Animation 7

2.2.1 Die physikalische Naturbeschreibung 7

2.2.2 Der Grundablauf der physikalisch basierten Animation 8

2.2.3 Anforderungen an die physikalisch basierte Animation 10

2.3 Techniken zur Bewegungssteuerung 12

2.3.1 Allgemeine Betrachtungen 12

2.3.2 Grundlegende Steuertechniken 13

2.3.3 Eignung der Techniken für allgemeine Animationssysteme 21

2.4 Automatische Kontaktbehandlung 22

2.4.1 Kontaktbestimmung 23

2.4.2 Kollisionssimulation 24

2.4.3 Behandlung von Ruhekontakten 26

3 Verfahren zur Constraint-Behandlung **29**

3.1 Dynamik des freien Systems 29

3.2 Beschleunigungslineare Constraints 31

3.3 Penalty-Verfahren 33

3.4 Die Reduktionsmethode 34

3.5 Die Lagrange-Faktoren-Methode (LFM) 36

3.5.1 Grundlegender Verfahrensablauf 36
3.5.2 Ein Beispiel: Die Schiefe Ebene 38
3.5.3 Formale Notation für Mehrkörpersysteme 39
3.5.4 Das Stabilisierungsproblem 40
3.5.5 Der Baraff-Algorithmus 42
3.6 Andere Lösungsverfahren 44

4 Bewertung der Lösungsverfahren 47
4.1 Anwendung in der physikalisch basierten Animation 47
4.1.1 Allgemeine Anwendungen 47
4.1.2 Anwendungen für allgemeine Animationssysteme 51
4.2 Anwendungsaspekte für allgemeine Animationssysteme 53
4.3 Verknüpfung von Anwendungs- und Anforderungsaspekten 59

5 Konzepte zur Anwendung der LFM in allgemeinen Animationssystemen 63
5.1 Ein Konzept zur Einbindung von Constraints 63
5.1.1 Formulierung von holonomen Constraints 64
5.1.2 Formulierung von Geschwindigkeits-Constraints 66
5.1.3 Formulierung von Konnektoren 67
5.2 Ein Katalog grundlegender Constraints 68
5.2.1 Holonome Constraints 69
5.2.2 Geschwindigkeits-Constraints 71
5.3 Ein Konzept zur Umsetzung der LFM 72
5.3.1 Mathematischer Formalismus 73
5.3.2 Aufspaltung der LFM 74
5.3.3 Ein allgemeines Verfahren zur Bestimmung der Lagrange-Faktoren 75
5.4 Konzepte zur Anwendung von Steuertechniken 76
5.4.1 Kombinationen von Steuertechniken auf der Grundlage der LFM 76
5.4.2 Konzepte zur Behandlung überbestimmter Systeme 78
5.4.3 Konzepte zur Vermeidung von Bewegungsartefakten 81

6 EMPHAS – ein allgemeines, physikalisch basiertes Animationssystem 83
6.1 Übersicht über das System 83
6.2 Grundelemente 84
6.2.1 Dynamische Körper 84
6.2.2 Konnektoren 85
6.2.3 Constraints 88

6.2.4 Weitere Steuerelemente 91
6.3 Umsetzung der LFM 93
6.3.1 Mathematischer Formalismus 94
6.3.2 Bestimmung der Lagrange-Faktoren 94
6.3.3 Erweiterung um ein SVD-Verfahren 95
6.3.4 Stabilisierungs- und Integrationsverfahren 96
6.3.5 Anpassung der Lösungsverfahren 98
6.4 Automatische Kontaktbehandlung 99
6.4.1 Realisierung der Kontaktbehandlung 100
6.4.2 Kombinationen mit anderen Steuertechniken 102
6.5 Graphische Darstellung, Simulationssteuerung und Interaktion 104
6.5.1 Realisierung der Benutzungsschnittstelle 104
6.5.2 Graphische Darstellung der Grundelemente 105
6.5.3 Simulationssteuerung 108
6.5.4 Interaktionstechniken 109
6.5.5 Import- und Export-Möglichkeiten 113
6.6 Anwendung der Steuerelemente 115
6.6.1 Festlegung von Constraints und Controllern 115
6.6.2 Behandlung überbestimmter Systeme 118
6.6.3 Vermeidung von Bewegungsartefakten 119
6.7 Vergleich mit kommerziellen Animationssystemen 121

7 Zusammenfassung und Ausblick **123**
7.1 Zusammenfassung 123
7.2 Ausblick 124

Anhang **127**

A Architektur von EMPHAS **127**

B Ein Modell zur Echtzeit-Simulation **131**

C Grundzüge der Klassischen Mechanik **135**

D Ergänzungen zur Kontaktbehandlung **139**
D.1 Bestimmung von Kollisions-Kraftstößen 139
D.2 Bestimmung von Kompensationskräften für Ruhekontakte 141

E Rotationsbeschreibung mit Quaternionen **143**

F Numerische Verfahren **147**

F.1 Lösung gewöhnlicher Differentialgleichungen 147

F.2 SVD-Verfahren zur Lösung linearer Gleichungssysteme 149

G Ableitungsterme einiger Constraints **153**

H Notation **155**

I Glossar **157**

Literatur **159**

Thesen **167**

Abbildungsverzeichnis

2.1 Grundablauf der physikalisch basierten Animation. 9
2.2 Hierarchie der Steuertechniken. 12
2.3 Die Wirkungsweise der Direkten und der Inversen Dynamik. 14
2.4 Wirkungsweise von Controllern. 15
2.5 Eine schiefe Ebene. 19
2.6 Ein Kugelgelenk. 19
2.7 Ein explizit zeitabhängiger Constraint. 19
2.8 Ein Teilchen in einer quaderförmigen Box. 19
2.9 Ein Punktkontakt zweier Körper. 23
2.10 Kompensationskräfte bei einem Ruhekontaktproblem. 26

3.1 Kartesische und generalisierte Koordinaten bei der schiefen Ebene. 34
3.2 Ablaufschema der Reduktionsmethode. 35
3.3 Äußere Kräfte und Zwangskräfte bei der schiefen Ebene. 36
3.4 Ablaufschema der LFM. 37

4.1 Die Zwischenebene der allgemeinen, physikalisch basierten Animationssysteme. . . 52
4.2 Instantane und stetige, physikalisch basierte Constraint-Erfüllung. 57

5.1 Das Konnektoren-Konzept. 64
5.2 Ein überbestimmtes System. 79

6.1 Architektur von EMPHAS. 83
6.2 Realisierung des Konnektorenkonzeptes in EMPHAS. 86
6.3 Das in EMPHAS realisierte Prozeduren-Konzept. 92
6.4 Zwei Beispiele für die Kontaktbehandlung in EMPHAS. 100
6.5 Einbindung der Kollisionssimulation in EMPHAS. 101
6.6 Einbindung von Ruhekontakten in EMPHAS. 102
6.7 Zwei Beispielszenen in EMPHAS. 104
6.8 Eine Beispielszene in EMPHAS mit verschiedenen Steuerelementen. 107

6.9 Wirkung des kinematischen Körperverhaltens. 112
6.10 Behandlung überbestimmter Systeme. 118
6.11 Erfüllung eines *PointToPoint*-Constraints mit aktiviertem Assembling-Modus. 121

A.1 Die unterste Ebene der EMPHAS-Architektur. 128
A.2 Die mittlere Ebene der EMPHAS-Architektur. 129
A.3 Die oberste Ebene der EMPHAS-Architektur. 129

B.1 Belegung der Simulationsparameter für einen Simulationsablauf. 132

C.1 Beschreibung des starren Körpers in Referenzpunktkoordinaten. 137

F.1 Ein Anfangs- und ein Randwertproblem. 148

Kapitel 1

Einführung

1.1 Hintergrund

Der Begriff „Animation“ leitet sich von dem lateinischen Begriff „anima“ – *Leben* – ab und kann daher mit „ins Leben bringen“ übersetzt werden. Allgemein bezeichnet man damit eine Bildsequenz, die beim Betrachter den Eindruck einer Bewegung oder allgemein einer zeitlichen Veränderung hervorruft. Zu diesem Zweck läßt sich die Trägheit des menschlichen visuellen Systems ausnutzen, das aufeinanderfolgende Bilder nur dann als Einzelbilder auflösen kann, wenn ihr zeitlicher Abstand mehr als ca. 0.04 Sekunden beträgt. Bei einer Wiederholfrequenz von mindestens 25 Bildern pro Sekunde kann somit eine stetige zeitliche Veränderung suggeriert werden.

Die vielleicht bekannteste Form von Animationen stellen die Anfang des 20. Jahrhunderts erstmals entwickelten Zeichentrickfilme dar, um die sich vor allem Walt Disney große Verdienste erworben hat. Eine große Hilfe für den aufwendigen Prozess der Animationserstellung leistete dann die Anwendung des Computers, der heute das wichtigste Werkzeug zur Erstellung einer Animation darstellt. Unter dem Begriff *Computeranimation* soll dabei die computerunterstützte Erstellung von Animationen als Sequenzen computergenerierter Bilder verstanden werden.

Die Computeranimation hat sich zu einem wichtigen Teilgebiet der Computergraphik entwickelt. Vor allem in den finanzstarken Märkten der Unterhaltungs- und Werbeindustrie haben Computeranimationen als effektvolle und Aufmerksamkeit erzwingende Filmsequenzen einen nicht zu übersehenden Niederschlag gefunden. Aber auch zur Veranschaulichung komplizierter Sachverhalte, als Bestandteil graphischer Benutzungsoberflächen oder im Rahmen wissenschaftlicher Anwendungen, z.B. im Bereich der Virtual Reality oder des Virtual Prototyping, werden Computeranimationen mit großem Erfolg eingesetzt. Eine häufig gestellte Anforderung ist dabei die visuelle Plausibilität – die Animationen sollen möglichst realistisch wirken, d.h. den gleichen visuellen Eindruck wie reale Zeitabläufe vermitteln oder diese sogar täuschend echt imitieren.

Die Erstellung von Computeranimationen ist aber auch mit komplizierten Problemen verknüpft. Eine einminütige Animation besteht bereits aus mehr als Tausend Einzelbildern, die sämtlich in geeigneter Weise erstellt, d.h. gerendert werden müssen. Es stellt sich somit die Frage nach geeigneten Werkzeugen, die den Animateur in flexibler und intuitiver Weise bei seiner Arbeit unterstützen. Ein solches Werkzeug stellen *allgemeine Animationssysteme* dar.

Allgemeine Animationssysteme. Der Begriff *allgemeine Animationssysteme* soll Softwareprodukte bezeichnen, die eine allgemeine, d.h. nicht auf bestimmte Objekttypen oder zeitliche Entwicklungsarten zugeschnittene, flexible und einfach durchführbare Animationserstellung ermöglichen.

Sie sollten daher den folgenden Anforderungen gerecht werden:

- Effektivität
  Die Anforderung der Effektivität umfaßt nicht nur die Erzielung eines angemessenen Zeitverhaltens, sondern auch den möglichst sparsamen Einsatz von Benutzungseingaben (oder allgemein Betriebsmitteln) für ein vorgegebenes Ziel.

- Flexibilität
  Animationssysteme müssen dem Benutzer erlauben, auf verschiedenste Art in den Prozess der Animationserstellung einzugreifen, wobei er durch entsprechende Interaktionstechniken unterstützt werden sollte. Besonders wichtig ist daher das Ziel der *Interaktivität* und die Verwirklichung von möglichst flexiblen Steuertechniken.

- Intuitivität
  Gerade wegen den vielfältigen Interaktionsmöglichkeiten ist die einfache Bedienbarkeit von Animationssystemen von größter Bedeutung – das Programm sollte sich so intuitiv und effektiv wie möglich anwenden lassen. Eng verknüpft mit diesem Aspekt sind die Anforderungen der Einheitlichkeit (einmal erlernte Techniken sollten sich für verschiedene Teilaufgaben einsetzen lassen) und der Vorhersagbarkeit (die Benutzungsaktionen sollten zu den vom Animateur erwarteten Wirkungen führen), die beide zu einer intuitiven Programmbedienung beitragen.

- Generalität
  Ein allgemeines Animationssystem sollte nicht auf bestimmte Körpertypen, Steuermöglichkeiten oder andere anwendungsspezifische Aspekte zugeschnitten sein, sondern sich durch eine einfache und robuste Änder- und Erweiterbarkeit auszeichnen. Erreicht werden kann dieses Ziel vor allem durch einen hohen Grad an *Modularität.*

Die Konzeption und Realisierung derartiger Systeme stellt ein wichtiges Ziel der Computeranimation dar. Die derzeitigen kommerziellen Animationssysteme werden diesen Anforderungen nur teilweise und in sehr unterschiedlicher Form gerecht. Sie basieren fast ausschließlich auf dem sogenannten *Keyframing*-Verfahren – einzelne „Schlüsselszenen“ (*Keyframes*) müssen explizit vom Animateur vorgegeben werden und lassen sich dann automatisch mit Hilfe von Interpolationsverfahren zu einem kompletten Zeitverlauf ergänzen. Auf der Grundlage dieser Technik sind beeindruckende Computeranimationen entstanden, aber gerade die Erstellung von sehr realistisch wirkenden Animationen erfordert nach wie vor einen großen Einarbeitungsaufwand und je nach Zielvorgaben und Komplexitätsgrad auch ein gewisses Maß an Erfahrung und künstlerischem Geschick.

Der Ansatz der physikalisch basierten Animation. Eine interessante Alternative zum klassischen Keyframing, die in den letzten 10 Jahren eine zunehmend größere Verbreitung gefunden hat, stellt die sogenannte *physikalisch basierte Animation* dar. Dieser Ansatz basiert auf einer physikalischen Simulation, die den Zeitverlauf eines realen Systems durch die automatische Auswertung eines idealisierten Modells nachbildet, der dann anschließend graphisch dargestellt wird. Dieser Ansatz hat einen entscheidenden Vorteil: Schon mit einem sehr geringen Aufwand lassen sich auf diese Weise äußerst realistisch wirkende Animationen erzeugen. Aufgrund der automatischen Generierung weist dieses Verfahren zudem einen hohen Grad an Wiederverwendbarkeit auf. Die physikalisch basierte Animation profitiert außerdem von der enormen Leistungssteigerung heutiger Rechnersysteme, mit denen die Simulation immer komplexerer Systeme in immer kürzerer Zeit möglich ist.

Eine wichtige und nicht leicht zu lösende Aufgabe stellt bei diesem Ansatz allerdings die flexible und intuitive *Steuerung* der Zeitverläufe dar, die für die Animationserstellung unerläßlich ist. Die standardmäßigen Einflußmöglichkeiten für physikalische Simulationen in Form von Anfangs- oder

Randbedingungen besitzen nicht den Grad an Flexibilität, um für diese Aufgabenstellung geeignet zu sein. Dieses Problem stellt ein aktuelles Forschungsgebiet dar, das neben computergraphischen Aspekten auch Anleihen in Bereichen wie der Numerik oder der angewandten Mechanik erforderlich macht. Ein grundlegendes Hilfsmittel stellen in diesem Rahmen *Constraints* dar, unter denen man ganz allgemein Bedingungen oder Beschränkungen verstehen kann, die dem jeweiligen System auferlegt werden. Sie erlauben die Kombination einfacher Objekte zu zusammengesetzten Systemen, können aber auch einen wichtigen Beitrag für die flexible Steuerung der Zeitverläufe leisten.

Allgemeine, physikalisch basierte Animationssysteme. Es stellt sich nun die Frage, wie sich auf der Grundlage der physikalisch basierten Animation allgemeine Animationssysteme realisieren lassen, die den oben genannten Anforderungen gerecht werden. Dieses Problem ist bislang noch nicht befriedigend gelöst worden. Die physikalisch basierte Animation erfordert die Durchführung von Teilaufgaben wie der Aufstellung und Lösung zeitlicher Entwicklungsgleichungen, die oftmals einen komplizierten, mathematisch-physikalischen Hintergrund haben, was ihre einfache Umsetzung, aber auch ihre intuitive Anwendung sehr erschwert. Dieses Problem zeigt sich auch beim Einsatz dieser Techniken in kommerziellen Animationssystemen, wie er mittlerweile recht häufig als Ergänzung zum Keyframing angeboten wird. Dabei kommen in der Regel nur sehr anwendungsspezifische Techniken zum Einsatz, die einen großen Einarbeitungsaufwand erfordern und auf ganz spezielle Systeme oder Steueraufgaben zugeschnitten sind. Den Anforderungen der Flexibilität und Intuitivität werden diese Systeme daher nur sehr eingeschränkt gerecht.

Wie sich zeigt, sind die Probleme der effizienten Simulationsdurchführung, der Entwicklung flexibler Steuertechniken und der Realisierung einer allgemeinen und intuitiven Programmbedienung auf komplizierte Weise miteinander verknüpft. So ermöglichen bestimmte mathematische Formalismen zwar eine sehr zeiteffiziente Simulation, schränken aber die Wahl der Steuertechniken ein, die auf ihrer Grundlage realisierbar sind. Auf der anderen Seite können manche Steuertechniken einen enormen Simulationsaufwand verursachen und somit einer interaktiven Programmbedienung entgegenstehen. Einige spezifische Eigenschaften des Simulationsverfahren können zudem bis auf die Ebene der Programmentwicklung und -bedienung Auswirkungen zeigen – sowohl in unterstützender als auch in einschränkender Weise. Dies gilt auch für die Lösungsverfahren zur Behandlung von Constraints, die eine wichtige Komponente bei der physikalisch basierten Animation darstellen.

1.2 Ziel und Aufbau der Arbeit

Die oben angesprochenen Problemverknüpfungen bei der Anwendung physikalisch basierter Techniken wurden bislang nur unzureichend untersucht. Zwar lassen sich Abhängigkeiten zwischen Simulationsverfahren und konkreten Animationssystemen auch in der computergraphischen Literatur wiederfinden, sie werden aber in der Regel nur vor dem Hintergrund eines ganz speziellen Problems, z.B. einer bestimmten Steuertechnik, diskutiert und nicht in einen systematischen Zusammenhang gebracht.

Diese Lücke soll in der hier vorliegenden Arbeit zu einem großen Teil geschlossen werden. Für die wichtige Klasse der mechanischen Systeme wird gezeigt, wie die verschiedenen Verfahren zur Durchführung der Simulation, insbesondere zur Constraint-Einbindung, vor dem Anforderungshintergrund allgemeiner Animationssysteme zu bewerten sind und welche Möglichkeiten und Probleme sie beinhalten. Als Ergebnis dieser Analyse werden Argumente dafür erarbeitet, daß die sogenannte *Lagrange-Faktoren-Methode* (LFM) als ein klassisches Verfahren zur Lösung Constraint-behafteter Bewegungsgleichungen in besonderem Maße für die Realisierung eines allgemeinen, physikalisch basierten Animationssystems geeignet ist. Auf dieser Grundlage werden dann Konzepte und Methoden für die Umsetzung der LFM in einem solchen Rahmen entworfen. Hierzu wird das im Rahmen

dieser Arbeit entwickelte, modulare Animationssystem EMPHAS vorgestellt, das sich auf spezifische Eigenschaften der LFM stützt und eine allgemeine, vollständig interaktive Erstellung physikalisch basierter Animationen ermöglicht.

Der Aufbau der Arbeit soll nun stichpunktartig vorgestellt werden.

In Kapitel 2 wird zunächst eine ausführliche Übersicht über die Grundlagen der physikalisch basierten Animation gegeben, insbesondere über die verschiedenen Steuertechniken und die automatische Kontaktbehandlung.

Ein Überblick über die wichtigsten Lösungsverfahren für das komplizierte Problem der Constraint-Behandlung erfolgt in Kapitel 3, wobei auch die LFM genauer erläutert wird.

Als ein neuer Beitrag wird dann in Kapitel 4 eine systematische Untersuchung und Bewertung der Lösungsverfahren vor dem Hintergrund allgemeiner Animationssysteme durchgeführt.

Kapitel 5 umfaßt die im Rahmen dieser Arbeit entworfenen Konzepte für die Anwendung der LFM in einem allgemeinen Animationssystem. Dabei wird die Formulierung und automatische Einbindung von Constraints, die modulare Umsetzung der LFM und die flexible und intuitive Anwendung von Steuertechniken besprochen.

In Kapitel 6 werden das Animationssystem EMPHAS und die zu seiner Entwicklung realisierten Methoden vorgestellt. Zunächst werden hierzu die Grundelemente des Systems aufgeführt, die u.a. verschiedene Elemente zur Bewegungssteuerung umfassen. Anschließend erfolgt eine Beschreibung der Umsetzung der LFM, der in EMPHAS realisierten Kontaktbehandlung, der Benutzungsschnittstelle des Systems und der Aspekte bei der Anwendung der Steuerelemente.

Als Abschluß dieser Arbeit erfolgt dann eine Zusammenfassung und ein Ausblick.

Der Anhang enthält neben einem Glossar und mathematischen und physikalischen Abhandlungen, die für das Verständnis der Arbeit wichtig sind, eine Beschreibung des modularen Entwicklungskonzeptes von EMPHAS und ein Modell zu Echtzeit-Simulation.

Kapitel 2

Grundlagen der physikalisch basierten Animation

2.1 Verfahren zur Animationserstellung

Um eine Einordnung des Ansatzes der physikalisch basierten Animation vornehmen zu können, sollen in diesem Abschnitt die grundlegenden Verfahren zur Animationserstellung vorgestellt werden. Nach [Zel91] und [HA92] können sie in deskriptive, generative und verhaltensbasierte Ansätze unterteilt werden, wie nun genauer ausgeführt werden soll.

Deskriptive Verfahren. Deskriptive Verfahren sind durch direkte oder indirekte Animationsvorgaben ohne die Berücksichtigung kausaler Effekte gekennzeichnet. Mit dem *Keyframing* umfassen sie eine eine der ältesten und noch immer erfolgreichsten Techniken zur Animationserstellung. Bei diesem Ansatz werden einzelne Frames explizit vom Animateur vorgegeben, während in einem zweiten Verfahrensschritt eine automatische Interpolation auf der Grundlage dieser „Schlüsselszenen“ stattfindet, um eine komplette Animationssequenz zu erhalten. Das Keyframing stellt eine einfach anwendbare und sehr flexible Technik zur Animationserstellung dar und bildet die Grundlage praktisch aller kommerziell verfügbarer Animationssysteme.

Auch prozedurale Verfahren, die auf parametrischen und somit lediglich indirekten Vorgaben beruhen, werden zu den deskriptiven Ansätzen gezählt. Gelenkkörper, d.h. Systeme aus gelenkig verbundenen, starren Segmenten, können z.B. sehr effektiv durch die Vorgabe von Gelenkwinkeln (*Direkte Kinematik*) oder von Positionsangaben für einzelne Teilglieder (*Inverse Kinematik*) gesteuert werden, aus denen dann die Konfiguration des gesamten Körpers automatisch berechnet wird. Ein anderes Beispiel für prozedurale Verfahren stellen Animationsskripte auf der Grundlage spezieller Animationssprachen dar und auch die Modifikation und Kombination vorhandener Bewegungsabläufe läßt sich zu diesem Ansatz zählen ([C+99]). Die Grundbewegungen entstammen dabei Filmaufnahmen (*Rotoskopie*) oder Messungen an realen Systemen, z.B. menschlichen oder tierischen Bewegungen (*Motion Capturing*) oder Gesichtsausdrücken (*Facial Tracking*).

Generative Verfahren. Einen völlig anderen Ansatz zur Erstellung von Computeranimation stellen generative Verfahren dar. Sie basieren auf der Festlegung kausaler Beziehungen, mit deren Hilfe die Bewegung der Szenenobjekte (oder allgemein der Zeitverlauf der Animation) automatisch generiert werden kann. Zu diesem Ansatz läßt sich auch die physikalisch basierte Animation zählen. Die kausalen Beziehungen haben in diesem Fall die Form physikalischer Entwicklungsgesetze, die

der Beschreibung realer Systeme entnommen werden können, und erfordern somit die Modellierung *dynamischer* Objekteigenschaften wie Masse oder Massenverteilung. Für die Erstellung von Animationen bietet dieser Ansatz zwei wichtige Vorteile:

- Die generierten Zeitverläufe wirken sehr realistisch. Diese Eigenschaft resultiert aus der Verwendung der Entwicklungsgleichungen, die auch das reale „Vorbild“ des Systems beschreiben.
- Die Zeitverläufe werden automatisch generiert. Durch die indirekte Festlegung sind nur sehr wenige Angaben notwendig, um komplette Animationen generieren zu können. Über die Variierung physikalischer Parameter lassen sich die Zeitverläufe zudem auf einfache Weise modifizieren, was zu einem hohen Maß an Wiederverwendbarkeit führt.

Natürlich lassen sich neben physikalischen Gesetzen auch andere Kausalbeziehungen für den generativen Ansatz heranziehen. In diesem Fall muß aber häufig explizit dafür Sorge getragen werden, daß die generierten Zeitverläufe visuell plausibel erscheinen.

Ein klassisches Beispiel für den physikalisch basierten Ansatz ist die in Abschnitt 2.4 beschriebene Behandlung von Körperkontakten, mit der sich vor allem unplausibel wirkende Durchdringungen der Körper verhindern lassen. Verfahren dieser Art haben inzwischen auch in viele kommerzielle Animationssysteme Einzug gefunden und stellen eine große Hilfe für den Animateur dar. Physikalisch basierte Techniken können auch die Interaktion mit den virtuellen Szenenobjekten erleichtern, indem ihnen ein realistisches Verhalten wie Undurchdringlichkeit, Reibung oder Massenträgheit verliehen wird, um auf diese Weise intuitiver erfaßbar und effektiver manipulierbar zu sein ([FW88], [W$^+$90], [GW91], [Gle92], [Gle94a], [Sur92b], [Sur92a], [NC99]).

Verhaltensbasierte Verfahren. Von einer verhaltensbasierten Animationserstellung spricht man schließlich, wenn autonom agierende „Agenten“ konstruiert werden, die über Fähigkeiten zur Wahrnehmung der Szene oder Teile der Szene verfügen, nach bestimmten Regeln Entscheidungen treffen und darauf aufbauend Aktionen durchführen. Derartige Verfahren sind vor allem für die Nachbildung natürlicher Bewegungsabläufe geeignet, z.B. für die Animation von Tiergruppen ([Rey87], [TT94]), sie lassen sich aber auch als allgemeine Steuertechnik anwenden, um komplexe Bewegungen zu erzeugen ([Wil90], [SG93]). Eine Übersicht über aktuelle Arbeiten aus diesem Gebiet findet sich in [C$^+$99] und [Fun00].

Generative und verhaltensbasierte Verfahren dürfen aber nicht als sich gegenseitig ausschließende Ansätze angesehen werden. Letztere bilden lediglich eine höhere Abstraktionsebene für die Animationserstellung, um hochkomplexe Systeme nachbilden zu können. Der physikalisch basierte Ansatz ist dabei für die Konstruktion von relativ selbstständig agierenden Untereinheiten, die sich auf sehr realistisch wirkende Weise bewegen oder zeitlich verändern sollen, besonders gut geeignet. Er dient daher sehr häufig als Grundlage von abstrakteren Verfahren, so daß die in dieser Arbeit vorgestellten Ergebnisse für die einfache und allgemeine Anwendung physikalisch basierter Techniken auch für die einfachere Anwendung von verhaltensbasierten Verfahren einen wichtigen Beitrag leisten können.

2.2 Physikalisch basierte Animation

Diese Arbeit beruht auf dem Ansatz der physikalisch basierten Animation als wichtigstem Vertreter der generativen Verfahren. Wie im vorigen Abschnitt und in der Einführung dieser Arbeit erläutert wurde, weist dieser Ansatz einige gewichtige Vorteile für die Animationserstellung auf, läßt sich aber nicht leicht in ein allgemeines Animationssystem integrieren. Die Einbeziehung physikalischer Gesetzmäßigkeiten in den Prozess der Animationserstellung soll daher nun ausführlich erläutert werden. Zunächst wird dazu die Naturbeschreibung der Physik in groben Zügen wiedergegeben, insbesondere der Begriff des physikalischen *Modells*. Auf diese Ausführungen stützt sich der daran anschließende Abschnitt, in dem der Grundablauf der physikalisch basierten Animation vorgestellt wird. Andere einführende Darstellungen zu diesem Thema finden sich z.B. in [W+88], [Tha90], [Wil91], [Gre91], [R+91], [HA92], [JN95], [H+96].

Der Einfachheit halber werden dabei häufig die Begriffe *Bewegung*, *Bewegungsgleichung* usw. anstelle des allgemeineren Begriffs der *Zeitentwicklung* verwandt. Die Ausführungen gelten aber auch für Systeme, deren zeitliche Änderungen nicht in Form von Bewegungen stattfinden.

2.2.1 Die physikalische Naturbeschreibung

Ein wichtiger Aspekt für das Verständnis der physikalischen Naturbeschreibung ist der Begriff des mathematischen *Modells*. Ein Modell ist eine vereinfachte (idealisierte) Entsprechung eines realen Systems, das bestimmte Eigenschaften dieses Systems nachbildet. Ein wesentlicher Modellbestandteil sind dabei mathematische Gesetze, die seine zeitliche Entwicklung festlegen.

Modelle müssen im Rahmen einer physikalischen Theorie formuliert werden. Jede physikalische Theorie hat dabei einen bestimmten *Anwendungsbereich*. Zum Anwendungsbereich der Klassischen Mechanik gehören z.B. Objekte aus unserem alltäglichen Erfahrungsbereich, die durch einen bestimmten Größenbereich und Geschwindigkeiten weit unterhalb der Lichtgeschwindigkeit gekennzeichnet sind. Gerade für die Computergraphik ist diese Theorie daher sehr wichtig. Durch unsere Vertrautheit mit diesen Objekten werden Bewegungen, die nicht dem physikalisch korrekten Verhalten entsprechen, schnell als künstlich erkannt. Die realistische Bewegungserzeugung für Objekte dieses Anwendungsbereiches stellt daher eine besonders häufig gestellte Aufgabe dar.

Neben dem Modell und dem Anwendungsbereich besteht die dritte Komponente physikalischer Theorien in *Abbildungsregeln*, die das Modell mit den realen Objekten aus dem Anwendungsbereich verknüpfen. So kann man z.B. der Position eines Teilchens einen dreidimensionalen Vektor und seiner Masse eine skalare Größe zuordnen, während über andere Eigenschaften (Größe, Farbe usw.) abstrahiert wird. Die Festlegung solcher Abbildungsregeln ist keinesfalls trivial und kann nur vor dem Hintergrund des jeweiligen Modells erfolgen.

Im Rahmen eines solchen Modells müssen neben den üblicherweise betrachteten sichtbaren Objektattributen auch die *dynamischen* Eigenschaften der Objekte wie Masse oder Elastizität festgelegt werden. Bei flüssigen oder gasförmigen Substanzen könnten dies z.B. auch die Dichte oder die Temperatur sein. Dynamische Eigenschaften zeichnen sich durch ihren Einfluß auf die Zeitentwicklung des Objektes aus. Ebenfalls sehr wichtig ist die Festlegung der äußeren Kräfte, die in dem System wirken sollen. Gravitationskräfte und elektrische Anziehungs- oder Abstoßungskräfte stellen hierfür oft gewählte Beispiele dar. Diese Festlegung von dynamischen Eigenschaften und Objektbeziehungen bezeichnet man als *physikalisch basierte Modellierung*. Zuweilen wird dieser Begriff synonym zu dem der *physikalisch basierten Animation* verwandt, der im Rahmen dieser Arbeit aber weiter gefaßt ist und den gesamten Prozess der Animationserstellung einschließen soll, wie in Abschnitt 2.2.2 ausgeführt wird.

Auf der Basis dieser Modellierung läßt sich dann eine computerunterstützte *Simulation* durchführen, in der die Entwicklungsgesetze automatisch ausgewertet werden. Auf diese Weise ergibt sich ein zeitlicher Verlauf des Modells, der den Zeitverlauf des realen Systems in idealisierter Form widerspiegelt. Ein sehr wichtiger Aspekt ist dabei die Beziehung zwischen den Entwicklungsgesetzen und den daraus resultierenden speziellen Zeitverläufen. Diese ergeben sich nämlich nur dann eindeutig aus den Gesetzen, wenn bestimmte zusätzliche Randwerte festgelegt werden, die z.B. in der Angabe des Systemzustandes zu einem bestimmten Zeitpunkt bestehen können. Diese Trennung von allgemeingültigen Gesetzen und randwertabhängigen konkreten Lösungen zieht sich durch sämtliche physikalische Theorien und kann allgemein als Versuch angesehen werden, die vielfältigen Erscheinungsformen der Natur mit Hilfe räumlich und zeitlich invarianter Gesetze zu beschreiben.

Diese Art der Naturbeschreibung hat sich als erstaunlich erfolgreich erwiesen. Die einfache Beschreibung komplexer Naturvorgänge stellt auch den Hauptgrund dafür dar, daß physikalische Techniken so erfolgreich für die Erstellung von Computeranimationen eingesetzt werden können.

2.2.2 Der Grundablauf der physikalisch basierten Animation

Die Grundidee der physikalisch basierten Animation besteht darin, die physikalischen Modelle und Gesetze, die sich zur Naturbeschreibung bewährt haben, für die Animation virtueller Objekte zu verwenden, um ihnen so ein realistisches Verhalten zu verleihen – insbesondere ein realistisches Bewegungsverhalten. Dazu muß eine physikalische Simulation durchgeführt und in geeigneter Weise graphisch dargestellt werden. Die hierzu notwendigen Schritte sollen nun genauer beschrieben werden. Als erläuterndes Beispiel dient dabei die Animation von zwei kleinen, kugelförmigen Körpern, die als Folge einer Kollision voneinander abprallen sollen.

Eine schematische Darstellung des gesamten Ablaufes ist in Abbildung 2.1 dargestellt, wobei die grau hinterlegten Bereiche diejenigen Teilschritte anzeigen sollen, die einen weitreichenden Eingriff des Animateurs erfordern. Der Begriff *physikalisch basierte Animation* soll sich auf diesen gesamten Ablauf beziehen, der die physikalisch basierte Modellierung und die Simulation als Unterprobleme umfaßt.

Festlegung des physikalischen Modells. Die Auswahl bzw. Konstruktion eines geeigneten Modells, das die gewünschten Eigenschaften des realen Systems widerspiegelt, stellt den ersten und oftmals auch schwierigsten Schritt dar, der nicht selten eine gewisse Erfahrung und physikalische Intuition erfordert. Für das Beispiel der kollidierenden Körper liegt die Anwendung der Klassischen Mechanik nahe. Die Körper können hier näherungsweise als Punktkörper behandelt werden, die durch dreidimensionale Ortsvektoren $\mathbf{r}_1$ bzw. $\mathbf{r}_2$ und skalare Massenwerte m_1 bzw. m_2 gekennzeichnet sind und der Newton'schen Bewegungsgleichung $\mathbf{F} = m\,\frac{d^2}{dt^2}\mathbf{r}$ gehorchen. Zur Kollisionsbeschreibung läßt sich z.B. das in Abschnitt 2.4.2 vorgestellte Modell für den zentrischen Stoß anwenden.

Festlegung der dynamischen Parameter und Randwerte. Als nächstes müssen die dynamischen Parameter und die Randwerte der Simulation spezifiziert werden. Sie können Messungen an realen Systemen oder auch frei gewählten Vorgaben entsprechen.

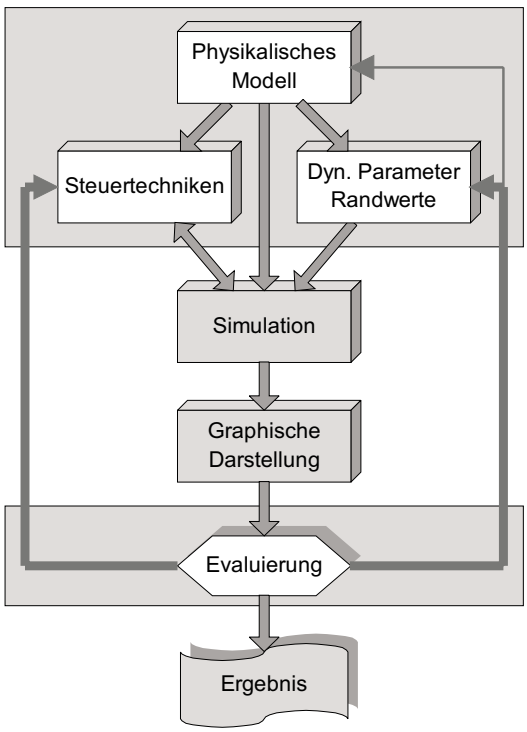

Abbildung 2.1: Grundablauf der physikalisch basierten Animation.

Zu den dynamischen Parametern gehören Objekteigenschaften wie Form, Masse oder Massenverteilung, aber auch Eigenschaften der äußeren Kräfte. Auf diese Spezifikation zusammen mit der Festlegung des physikalischen Modells soll sich auch der Begriff der *physikalisch basierten Modellierung* oder kurz *Modellierung* beziehen. Eine sehr formale Einführung in dieses Gebiet wird in [Bar92] gegeben.

In dem Kollisionsbeispiel müssen die Körpermassen, die Stoßzahl (siehe Abschnitt 2.4.2) und die äußeren Kräfte als dynamische Parameter festgelegt werden. Außerdem sind für die eindeutige Bestimmung der Körperbewegungen vier skalare Randwerte vorzugeben (vergl. Anhang F.1), z.B. die anfänglichen Positionen und Geschwindigkeiten der beiden Körper.

Festlegung von Steuertechniken. Eine sehr wichtige Komponente bei der physikalisch basierten Animation ist die geeignete Steuerung der Bewegungen. Die hierfür entwickelten Techniken werden in Abschnitt 2.3 ausführlich vorgestellt. Viele dieser Verfahren greifen dabei auf die aktuellen Zustandswerte des Systems zurück, die in dem anschließend erläuterten Schritt der Simulation automatisch generiert werden.

Für das betrachtete Beispiel könnte z.B. durch die Anwendung von speziellen Kraftfunktionen (vergl. Abschnitt 2.3.2.2) gesichert werden, daß sich die Körper an einem bestimmten Ort treffen.

Simulation. Die eigentliche Simulation besteht dann in der automatischen Auswertung der Entwicklungsgesetze, typischerweise z.B. durch eine numerische Lösung von Bewegungsgleichungen in Form von Differentialgleichungen. Diesen Vorgang bezeichnet man (bei bekannten Werten für die wirkenden Kräfte) auch als *Direkte Dynamik*. Als Ergebnis dieser Auswertung resultiert dann der gesuchte Zeitverlauf des Modells.

Auch in dem Beispiel der kollidierenden Körper bietet sich eine numerische Lösung der Bewegungsgleichungen an. Im Falle einer Kollision können dabei, wie in Abschnitt 2.4.2 beschrieben wird, neue Werte für die Körpergeschwindigkeiten berechnet und zugewiesen werden.

Graphische Darstellung. Der generierte Zeitverlauf muß nun noch auf geeignete Weise graphisch dargestellt werden, was durch eine Übertragung der simulierten Zustandswerte auf entsprechende virtuelle Objekte geschehen kann. Bei einer schrittweisen Simulation wie z.B. der numeri-

schen Lösung von Differentialgleichungen wird diese Übertragung in der Regel schon während der Auswertung der Entwicklungsgleichungen vorgenommen, z.B. nach jedem n-ten Simulationsschritt mit $n \geq 1$. Die vollständige, z.B. fotorealistische, graphische Darstellung erfolgt dann erst in einem Nachbearbeitungsschritt, wenn das Simulationsergebnis nicht zuvor verworfen wird.

Evaluierung. Schließlich muß eine Evaluierung der Animation vorgenommen werden, d.h. eine Analyse, ob die gewünschten Zielstellungen erfüllt worden sind. Bei einem nicht zufriedenstellenden Ergebnis muß zu der Festlegung der dynamischen Parameter und Randwerte oder der Festlegung der Steuertechniken zurückgegangen werden, um eine neue Simulation unter veränderten Bedingungen durchzuführen. Tatsächlich ist es oft sehr schwierig, den Einfluß dieser Größen auf die Zeitentwicklung des Modells einzuschätzen und richtig zu wählen, so daß dieser Schritt sehr häufig erforderlich ist. In dem Kollisionsbeispiel sind vor allem die Anfangspositionen und -geschwindigkeiten der beiden Körper wichtig, da sie darüber entscheiden, ob es überhaupt zu einer Kollision kommt. Zuweilen muß sogar das physikalische Modell modifiziert werden, was natürlich völlig andere Belegungen für die Simulationsparameter notwendig machen kann und in der Regel zu einem großen Mehraufwand beim Erstellungsprozess führt.

2.2.3 Anforderungen an die physikalisch basierte Animation

Das in Abbildung 2.1 skizzierte Ablaufschema der physikalisch basierten Animation besitzt auch für die Verwendung von Simulationen in einem rein technischen Umfeld Gültigkeit. Im Rahmen einer solchen Anwendung, die mit dem Begriff *Technische Simulation* bezeichnet werden soll, werden mit Hilfe der physikalischen Simulation bestimmte Informationen über die dynamischen Eigenschaften des zu Grunde gelegten Modells gewonnen. Das physikalische Modell und in der Regel auch die zum Modell gehörenden Steuertechniken [1] sind demnach fest vorgegeben, während das Ziel in der möglichst exakten Bestimmung des Zeitverlaufes besteht.

Wie in diesem Abschnitt genauer ausgeführt werden soll, weisen die beiden Anwendungen der physikalisch basierten Animation und der technische Simulation bestimmte Gemeinsamkeiten, aber auch wesentliche Unterschiede auf. Eine Abgrenzung dieser beiden Bereiche läßt sich dabei vor allem über die Untersuchung der Anforderungen erreichen, die an den physikalisch basierten Ansatz zu stellen sind – insbesondere vor dem Hintergrund allgemeiner Animationssysteme. Diese Anforderungen sollen nun genauer erörtert werden, wobei sich das Problem der flexiblen Bewegungssteuerung als besonders wichtig erweisen wird.

Effizienz und Robustheit. Zwei Teilziele sind sowohl für die physikalisch basierte Animation als auch für die technische Simulation von Bedeutung: Die zeitliche Effizienz und die Robustheit des Simulationsverfahrens. Bei der Erstellung von Computeranimationen kommen zwar meistens einfachere Systeme zum Einsatz ([AD92]), die Bewegungsgenerierung in Fast-Echtzeit stellt dafür aber ein viel wichtigeres Ziel dar. Nur auf diese Weise läßt sich ein interaktiver Umgang mit dem Animationssystem und eine direkte Rückkopplung für die getätigten Benutzungsaktionen erzielen. Eine mangelnde Robustheit auf der anderen Seite kann zu einem unvorhersehbaren und unkorrekten Verhalten oder sogar zu einem Abbruch der Simulation führen.

Automatisierung von Teilaufgaben. Die Erstellung einer Animation ist eher im künstlerischen als im technischen Bereich angesiedelt. Dieser Umstand wirkt sich auf das voraussetzbare Vorwissen des Anwenders aus und betont die Wichtigkeit der weitestgehenden Automatisierung der verschiedenen Teilaufgaben bei der physikalisch basierten Animation. Wie sich den Ausführungen

[1] Bei der Simulation von Kraftfahrzeugen stellt das Lenksystem z.B. einen integralen Bestandteil des jeweiligen Fahrzeugmodells dar.

im vorigen Abschnitt entnehmen läßt, sind diese Aufgaben sehr komplex und haben oftmals einen mathematisch-physikalischen Hintergrund. Die Einbindung in ein allgemeines Animationssystem, das sich ohne ein spezielles technisches Vorwissen bedienen läßt, hat daher für die physikalisch basierte Animation eine sehr große Bedeutung.

Da die physikalischen Modelle bei der physikalisch basierten Animation nicht wie bei der Technischen Simulation vorgegeben sind, sollte für die Umsetzung eines solchen Animationssystems zudem eine einfache Änder- und Erweiterbarkeit und damit ein möglichst hohes Maß an Modularität angestrebt werden. Grundsätzliche Beschränkungen auf bestimmte Körpertypen, Bewegungsabläufe oder Eingriffsmöglichkeiten sollten soweit wie möglich vermieden werden.

Wahrgenommene statt physikalische Korrektheit. Der Anspruch von Computeranimationen liegt in ihrer visuellen Glaubwürdigkeit bzw. in ihrer Ästhetik im Gegensatz zur physikalischen Korrektheit als Zielstellung der Technischen Simulation. Die physikalisch basierte Animation stellt daher viel höhere Ansprüche an die graphische Darstellung der generierten Zeitverläufe, während physikalisch unkorrekte Ergebnisse durchaus toleriert werden können, wenn sie nicht wahrnehmbar sind oder den Gesamteindruck der Animation nicht beeinträchtigen. Selbst im Rahmen realistisch wirkender Animationen ist die Übereinstimmung mit realen Zeitverläufen oftmals nicht für alle Szenenbestandteile erforderlich oder wünschenswert. Diese Verschiebung der Zielstellung, die in [AD92] und [B$^+$96] genauer beleuchtet wird[2], stellt einen wesentlichen Unterschied zwischen den beiden Anwendungsgebieten dar.

Bewegungssteuerung. Die Realisierung einer flexiblen und intuitiven *Steuerung* der generierten Bewegungen ist eine der wichtigsten Aufgaben bei der physikalisch basierten Animationund zugleich das wichtigste Unterscheidungskriterium zur Technischen Simulation. Auch dort werden Methoden zur automatischen Bewegungssteuerung eingesetzt, für die Erstellung von Animationen wird aber ein viel höheres Maß an Flexibilität, Intuitivität und Interaktivität gefordert. Die Vorstellungen des Animateurs, der sich in der Regel in einem künstlerischen und nicht in einem technischen Umfeld bewegt, müssen sich optimal umsetzen lassen. Dieser Sachverhalt ist auch auf die Formel „animation is simulation plus control“ ([Pla92]) gebracht worden. Die Steuertechniken sollten sich dabei in möglichst flexibler und leicht erlernbarer Weise anwenden lassen.

Das Problem der flexiblen Bewegungssteuerung läßt sich nicht leicht lösen und ist Untersuchungsgegenstand aktueller Forschungsarbeiten. Einige wichtige Ziele bei der Animationserstellung sind z.B. die Vorgabe einer Zielkonfiguration für bestimmte Teile des Systems, die Festlegung von Objektbeziehungen, die Minimierung von Kostenfunktionen wie der Summe der aufgewandten Kräfte oder die Angabe bestimmter Verhaltensmuster, wofür auch die automatische Kontaktbehandlung ein Beispiel wäre.

Im anschließenden Abschnitt wird nun eine ausführliche Übersicht über Techniken zur Steuerung mechanischer Systeme gegeben. Viele dieser Techniken lassen sich dabei auch auf nichtmechanische Systeme anwenden[3]. Da sich die weiteren Ausführungen dieser Arbeit aber auf mechanische Systeme beziehen, werden diese Aspekte des Gültigkeitsbereiches nicht im einzelnen diskutiert.

[2] In [B$^+$96] werden außerdem sogenannte *Variabilitäten* als zusätzliche, modellunabhängige Freiheitsgrade eingeführt, um natürlich wirkende Animationen erstellen zu können.

[3] Als generelle Voraussetzung müssen dabei zumindest dynamische Entwicklungsgleichungen vorliegen, die Kräfte und Zeitentwicklung miteinander in Beziehung setzen.

2.3 Techniken zur Bewegungssteuerung

2.3.1 Allgemeine Betrachtungen

Um eine Übersicht über die verschiedenen Ansätze zur Bewegungssteuerung zu gewinnen, soll zunächst eine Klassifikation der Steuertechniken über die Einführung verschiedener Abstraktionsebenen vorgenommen werden. Als Vorlage dieses Schemas diente die Klassifikation von David Zeltzer ([Zel91]). Er unterscheidet zwischen einer *strukturalen*, einer *prozeduralen*, einer *funktionalen* und einer *agentenbasierten* Ebene. Die strukturale Ebene entspricht der dynamischen Szenenbeschreibung, während die prozedurale Ebene strukturunabhängige Verfahren wie die Direkte und Inverse Dynamik sowie kinematische Verfahren umfaßt. Diese beiden Ebenen wurden für die hier erarbeitete Klassifikation übernommen.

Die höheren Abstraktionsebenen beziehen sich in [Zel91] aber in erster Linie auf die Steuerung biomechanischer Systeme. Die funktionale Ebene umfaßt laut [Zel91] Mittel, mit denen Prozeduren mit Objekten oder Teilobjekten verknüpft werden, um autonome Untersysteme in Form von Muskel-Gelenk-Gruppen zu bilden, die ganz bestimmte Bewegungsformen bewirken können. Die in der obersten Ebene angesiedelten „Agenten“ enthalten dann eine strukturale Beschreibung, eine Menge funktionaler Einheiten und Regeln zum Einsatz dieser Einheiten.

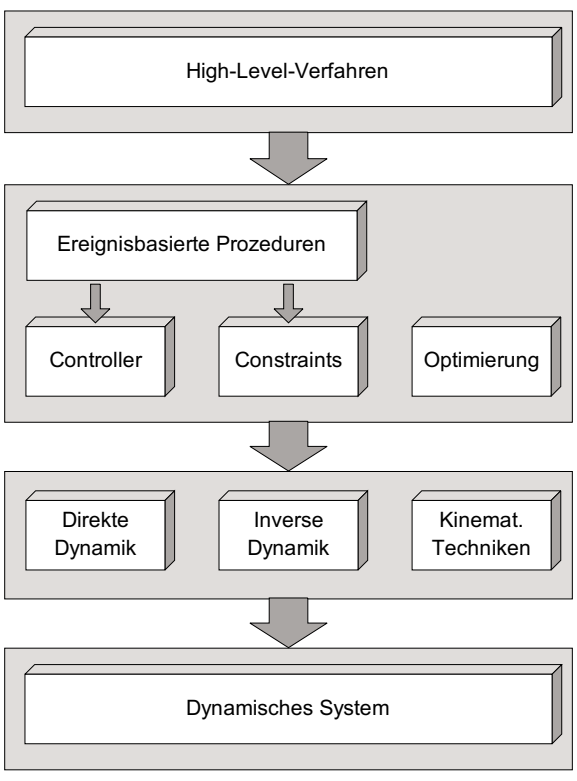

Abbildung 2.2: Hierarchie der Steuertechniken.

Es soll nun die in dieser Arbeit vorgenommene Klassifikation vorgestellt werden, in der dieser explizite Bezug auf biomechanische Systeme und damit die Beschränkung auf diese Anwendungsklasse vermieden wird. In Abbildung 2.2 ist diese Klassifikation schematisch dargestellt. Eine Übersicht über Techniken zur Steuerung von Gelenkkörpern, die sich zu einem großen Teil mit den hier vorgestellten Techniken decken, findet sich in [Wag95].

Auf der untersten Ebene befindet sich die Beschreibung des zu Grunde liegenden dynamischen Systems, in die z.B. die dynamischen Objekteigenschaften wie Masse oder Massenverteilung einfließen. Die hieraus resultierenden Bewegungsmöglichkeiten werden dabei in keiner Weise eingeschränkt. Dies geschieht erst in der darüberliegenden Abstraktionsebene, in der Techniken angesiedelt sind, die grundlegende Möglichkeiten zur Einflußnahme auf die zeitliche Entwicklung des dynamischen Systems darstellen. Neben prozeduralen kinematischen Techniken wie der Direkten und Inversen Kinematik gehören hierzu die Ansätze der Direkten und Inversen Dynamik, die im Abschnitt 2.3.2.1 näher beschrieben werden. Sie stellen keine Steuertechniken im eigentlichen Sin-

ne dar, sondern bezeichnen zwei Einsatzmöglichkeiten der dynamischen Systemgleichungen: Die Bewegungsgenerierung auf der Grundlage von Kräften und die Kraftbestimmung auf der Grundlage von Bewegungsdaten.

Die konkrete Anwendung dieser Ansätze auf die Elemente des dynamischen Systems kann mit Hilfe der drei wichtigsten Arten allgemeiner Steuertechniken geschehen: Controller, Optimierungstechniken und Constraints. Diese Techniken werden in den Abschnitten 2.3.2.2, 2.3.2.3 und 2.3.2.4 erläutert. Sie bilden auch die Grundlage für die in Abschnitt 2.3.2.5 vorgestellten ereignisbasierten Prozeduren.

Auf der höchsten Ebene können dann autonom agierende *Agenten*, die über eine Szenenwahrnehmung und die Möglichkeit zu regelbasierten Aktionen verfügen, oder andere abstrakte Steuertechniken, wie z.B. die Steuerung mit Hilfe natürlichsprachlicher Befehle, eingeordnet werden. Sie stellen keine Alternative zu den grundlegenden Steuerverfahren dar, sondern stützen sich auf diese, um komplexe Bewegungsmuster zu generieren. Die Untersuchung dieser Techniken, über die z.B. in [C+99] ein Überblick gegeben wird, stellt aber ein eigenes Forschungsgebiet dar, das nicht zum Themenbereich dieser Arbeit gehört.

Einen eigenständigen Ansatz stellt zudem die in [G+98] beschriebene Anwendung neuronaler Netze dar. Dabei wird ein neuronales Netz mit Hilfe eines dynamisch simulierten Systems trainiert, um seine physikalisch erlaubten Bewegungen zu erlernen. Im Anschluß an diesen Offline-Prozess können dann mit Hilfe des trainierten Netzes visuell plausible Bewegungen[4] für das jeweilige System generiert werden, die sich um ein Vielfaches schneller als bei der numerischen Lösung der Bewegungsgleichungen erstellen lassen. Über einen iterativen Annäherungsprozess können zudem bestimmte Zielvorgaben, z.B. die gewünschte Endstellung eines Pendels, in sehr effizienter Weise berücksichtigt werden.

2.3.2 Grundlegende Steuertechniken

2.3.2.1 Direkte und Inverse Dynamik

Unter der Direkten Dynamik versteht man die Bestimmung des Bewegungsverhaltens auf der Grundlage bekannter Kräfte. Diese auch als *Vorwärtssimulation* bezeichnete Auswertung der physikalischen Bewegungsgleichungen stellt das Standardverfahren bei der Simulation physikalischer Systeme dar. Bei mechanischen Systemen ist hierzu ein gewöhnliches Differentialgleichungssystem zu lösen, wofür z.B. numerische Verfahren zum Einsatz kommen können. Globale Kraftfelder wie die Schwerkraft oder elektrische Felder (bei geladenen Teilchen) können dabei ebenso unproblematisch eingebunden werden wie Kräfte, die an bestimmten Körperpunkten angreifen, z.B. benutzerspezifizierte Federn. Die Kräfte dürfen auch eine explizite Abhängigkeit vom Systemzustand oder von der Zeit besitzen.

Für die Realisierung einer flexiblen Bewegungssteuerung ist jedoch das Problem zu lösen, *welche* Kräfte angewandt werden müssen, um das gewünschte Bewegungsverhalten zu erzielen. Viele intuitive Steuervorgaben wie die Festlegung einer Zielkonfiguration oder die Vermeidung von Körperdurchdringungen treffen keine expliziten Aussagen über die im System wirkenden Kräfte.

Eine Möglichkeit würde darin bestehen, die Parameter der festgelegten Kräfte zu variieren und das dadurch erzielte mit dem gewünschten Bewegungsverhalten zu vergleichen. Auf diese Weise könnten die gesuchten Kräfte durch reines Probieren gefunden werden. Es stellt sich allerdings heraus,

[4] Laut [G+98] können die Unterschiede zu den physikalisch korrekten Bewegungen dabei unmerklich klein gemacht werden.

daß ein solches Vorgehen in den allermeisten praktischen Fällen nicht in einem akzeptablen Zeitrahmen durchführbar ist. Schon bei einfachen Gelenkkörpern ist die Zahl der Freiheitsgrade so groß, daß auf diese Weise keine koordinierten Bewegungen erzeugt werden können, schon gar nicht solch komplexe wie z.B. der menschliche Gang.

Als Steuertechnik hat die Direkte Dynamik daher einen sehr begrenzten Anwendungsbereich. Er umfaßt vor allem einfache Systeme, die sich unter der Einwirkung bekannter Kräfte wie der Schwerkraft entwickeln. Die Direkte Dynamik stellt aber die grundlegende Technik für die Generierung der physikalisch basierten Bewegungen dar (vergl. Abschnitt 2.2.2).

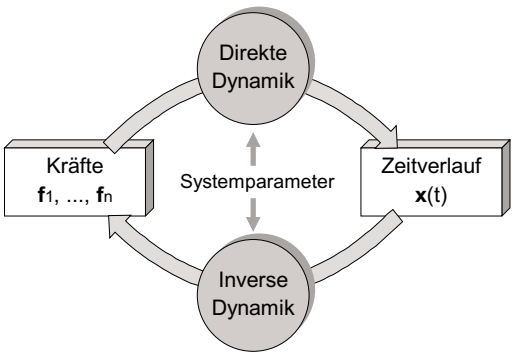

Abbildung 2.3: Die Wirkungsweise der Direkten und der Inversen Dynamik.

Die Inverse Dynamik ist eine direkte Umkehrung der Direkten Dynamik, wie in Abbildung 2.3 skizziert ist. Sie geht von einer gegebenen Konfiguration aus und zielt auf die Berechnung der Kräfte, die bei einer Vorwärtssimulation zu dieser Konfiguration führen würden. Dieser Ansatz liegt sehr vielen Steuertechniken zu Grunde. Bei Gelenkkörpern können den gesuchten Kräften z.B. Federn entsprechen, die an Gelenken angebracht sind und diese auslenken sollen. Mit Hilfe der Inversen Dynamik kann dann ermittelt werden, welche Federkräfte notwendig sind, um eine bestimmte Konfiguration oder einen bestimmten Bewegungsverlauf zu erzielen. Durch die Anwendung und gezielte Modifikation dieser Kräfte kann dann eine sehr effiziente Kontrolle über den Gelenkkörper erreicht werden (vergl. [KB96] und die in Abschnitt 2.3.2.2 angeführten Arbeiten). In [NC99] wurde zudem die Verwendung der Inversen Dynamik für die interaktive Bestimmung und Anwendung von Kräften für Force-Feedback-Eingabegeräte als Hilfe bei der physikalisch basierten Modellierung vorgestellt. Ein besonders weit verbreitetes Verfahren stellt die Inverse Dynamik außerdem in der Robotik dar.

Für die Anwendung dieses Ansatzes ergeben sich allerdings einige Probleme, die nicht leicht zu lösen sind:

- Anstelle eines Differentialgleichungssystems muß für die Bestimmung der Kräfte in den meisten Fällen ein nichtlineares Gleichungssystem aufgelöst werden, da die Gesamtkraft in komplizierter Weise von den einzelnen Wirkungskräften abhängt. Die Inverse Dynamik erfordert daher in der Regel einen wesentlich größeren Zeitaufwand als die Direkte Dynamik.

- Die vorgegebene Zielkonfiguration könnte physikalisch inkonsistent sein. In diesem Fall ließen sich keine Wirkungskräfte angeben, die zur vorgegebenen Konstellation führen würden.

- In vielen Fällen kann eine Zielkonfiguration nicht nur durch eine, sondern durch eine Vielzahl von Kombinationen wirkender Kräfte erzielt werden. Es stellt sich somit das Problem der geeigneten *Auswahl* einer Lösung. Bei der oben angesprochenen Gelenkkörper-Steuerung durch Federkräfte ergeben sich z.B. häufig unrealistisch große Kräfte, die sich nicht für eine plausibel wirkende Bewegungssteuerung eignen.

Als alleinige Grundlage zur Generierung von Animationen stellt die Inverse Dynamik außerdem keine Hilfe für die visuelle Plausibilität in Form realistischer Bewegungen dar. Der Zeitverlauf muß hier vorgegeben werden und resultiert nicht wie bei der Direkten Dynamik aus dem physikalischen

System. Die Inverse Dynamik erhält aber ihre Bedeutung in der Kombination mit anderen Techniken und in der Beschränkung auf einige wenige Systemfreiheitsgrade.

Mit Controllern, Optimierungsverfahren und Constraints sollen nun drei grundlegende Ansätze zur Bewegungssteuerung vorgestellt werden. Sie bilden das Grundgerüst für eine flexible Steuerung im Rahmen der physikalisch basierten Animation.

2.3.2.2 Controller

Als Controller bezeichnet man Kontrolleinheiten, die eine explizit zeitabhängige oder vom Systemzustand abhängige Kraft ausüben. Diese Kräfte können z.B. vor Simulationsbeginn als direkte Funktion der Zeit festgelegt werden, um einen direkten Einfluß auf die Szenendynamik zu erzielen. Wesentlich flexibler ist die Anwendung zustandsabhängiger Controller: Sie beinhalten eine funktionale Abhängigkeit von bestimmten Systemgrößen, deren Wert sich erst zur Simulationszeit ergibt. Man spricht daher auch von einer Rückkopplungskontrolle (*State Feedback Control*) oder von einem *Closed Loop*-Verfahren gegenüber einem *Open Loop*-Verfahren, bei dem diese Rückkopplung nicht stattfindet ([H+96]). Durch die Möglichkeit, auf die Systemkonfiguration zugreifen zu können, läßt sich vor allem eine sehr stabile Steuerung erzielen, die nicht durch kleine äußere Störungen außer Kraft gesetzt wird. Diese beiden Verfahrensweisen sind in Abbildung 2.4 skizziert.

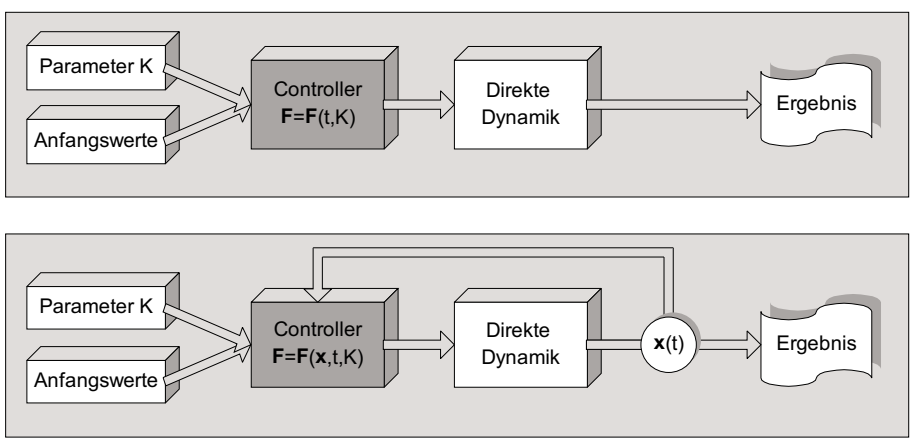

Abbildung 2.4: Wirkungsweise von Controllern als Open Loop- (oben) und als Closed Loop- Verfahren (unten).

Verbreitet sind Controller vor allem in der Robotik, wo sie das Standardverfahren zur Steuerung gelenkig verbundener Systeme darstellen. Hier wird jedes Gelenk durch einen Controller gesteuert, der Zugriff auf bestimmte Systemgrößen wie der Gelenkauslenkung hat und aus diesen eine Kraft errechnet, die dann auf das Gelenk angewandt wird. Wenn $\mathbf{x}(t)$ den Zustandsvektor eines dynamischen Systems und $\mathbf{x}^E$ eine Fehlerfunktion darstellt, z.B. $\mathbf{x}^E(t) := \mathbf{x}^{Ziel} - \mathbf{x}(t)$, läßt sich z.B. eine Kraft proportional zu $\mathbf{x}^E$ festlegen, die der Abweichung vom gewünschten Bewegungsverlauf entgegengerichtet ist. Um eine Stabilisierung der Kraftwirkung zu erreichen, werden bei den häufig eingesetzten *PID-Controller* zusätzlich ein differentieller Term und ein Integralterm berücksichtigt:

$$\mathbf{F} = -K_X \mathbf{x}^E + K_V \dot{\mathbf{x}}^E - K_I \int \dot{\mathbf{x}}^E \, dt \; .$$

Die zeitkonstanten Matrizen K_V, K_X und K_I müssen dabei geeignet gewählt werden, um das gewünschte Bewegungsverhalten zu erzielen. Ebenfalls sehr verbreitet sind *PD-Controller*, bei denen

Integralterm nicht berücksichtigt wird.

In dieser Form gehen die Eigenschaften des dynamischen Systems, insbesondere die Massenverhältnisse, allerdings nicht in die Bestimmung der Kontrollkräfte ein. Schwere Körper erfahren also die gleiche Kraft wie leichte. Mit Hilfe des sogenannten *Dynamic Control*-Ansatzes kann das dynamische System aber unter bestimmten Umständen für die Bestimmung der Kräfte einbezogen werden ([H+96]).

Controller wurde auch im Bereich der Computeranimation sehr häufig eingesetzt. Die zu jedem Zeitpunkt resultierenden Kräfte werden dabei einfach den äußeren Kräften hinzuaddiert, so daß die Bewegungsgleichungen wie gewohnt mit Hilfe der Direkten Dynamik gelöst werden können. Schon in den frühen Arbeiten zur physikalisch basierten Animation von Gelenkkörpern [AG85], [WB85], [A+87], [Wil87] wurden Controller für die Modellierung von Gelenkgrenzen und Bodenkontakte eingesetzt (für nähere Ausführungen siehe [Wag95]). Mit speziell auf die jeweiligen Bewegungsformen angepaßten Controllern konnten auch koordinierte Bewegungen für Gelenkkörper erzeugt werden, z.B. Sprünge eines einbeinigen Roboters ([RH91]), Gehbewegungen ([BC89], [MZ90], [RH91], [P+92a], [SC92a]), Tauchbewegungen ([WH96]) oder andere natürlich wirkende Bewegungsformen ([IC87], [P+92b], [H+92], [P+93]). Der Ansatz der Inversen Dynamik wird dabei häufig als Hilfsmittel für die Konstruktion der Controller angewandt. Zum Einsatz kommen Controller auch für die Steuerung von Gelenkkörpern, die nach biomechanischen Gesichtspunkten modelliert wurden (vergl. [Wag95]).

In [L+95] wird ein speziell angepaßter Controller eingesetzt, um kinematische Trajektorien für starre Körper vorgeben zu können, ohne das dynamische Verhalten des Körpers auszuschalten. Trifft ein Körper, der einer solchen Trajektorie folgt, z.B. auf einen anderen Körper, wird er zunächst durch eine physikalisch basierte Kontaktbehandlung abgelenkt und dann mit Hilfe des Controllers wieder auf die gewünschte Bahn gebracht. Auf diese Weise ist ein interessanter Kompromiß zwischen kinematischer und dynamischer Steuerung erzielt worden. Eine um Synchronisierungs-Constraints ergänzte Weiterentwicklung wurde in [LG96] vorgestellt.

Als allgemeine Steuertechnik sind Controller aber nur sehr bedingt geeignet. Dies liegt vor allem daran, daß nur lokale Bewegungsziele vorgegeben werden können, deren gegenseitiger Einfluß nicht berücksichtigt wird. Dieses Problem tritt z.B. bei mechanischen Kopplungen in Form von Gelenken auf, die für die Computeranimation sehr wichtig sind. Außerdem müssen die Matrizen K_V, K_X und K_I sowohl bei den PID-Controllern als auch beim *Dynamic Control*-Ansatz einzeln angepaßt werden, da sie sich nicht aus der Systemdynamik ergeben ([H+96]).

In [L+00] werden PD-Controller für die Gelenkauslenkung zweidimensionaler Gelenkkörper eingesetzt, die der direkten Kontrolle des Benutzers unterliegen, z.B. über die Zuordnung von Controller-Parametern zu Mauskoordinaten. Die Controller müssen aber auch für diese Anwendung für jedes Modell einzeln angepaßt werden, außerdem erfordert ihr effektiver Einsatz einen gewissen Lernaufwand ([L+00]). Sehr schwierig erscheint auch die Erweiterung auf dreidimensionale Modelle, da die hierfür notwendige Zahl von Controller-Parametern in der Regel um ein Vielfaches größer als die Zahl der gleichzeitig festlegbaren Eingabegrößen ist.

2.3.2.3 Optimierungsverfahren

Optimierungsverfahren basieren auf einem völlig anderen Ansatz als die anderen hier vorgestellten Steuerverfahren. Anstatt bestimmte kinematische Bedingungen oder Verhaltensregeln vorzugeben, können mit ihrer Hilfe Bewegungen erzeugt werden, die durch die Minimierung einer Kostenfunktion gekennzeichnet sind. Damit eignen sie sich vor allem für die Nachbildung natürlicher Bewegungsabläufe. Viele biomechanische Bewegungen wie der menschliche Gang, das Kriechverhalten

eines Wurms oder die Schwimmbewegungen eines Fisches sind nämlich durch die Minimierung bestimmter physikalischer oder biologischer Parameter ausgezeichnet, z.B. die Summe der aufgewandten Muskelkräfte. In der Robotik werden Optimierungsverfahren zur Vorausplanung optimaler Trajektorien verwandt, die dann mit Hilfe von Controllern realisiert werden.

Diese Verfahren basieren auf einer iterativen Annäherung an die gewünschte Lösung. Für ein System, das durch Zustandsvariablen **x** und Geschwindigkeitsvariablen **v** beschrieben wird, besteht z.B. ein häufig aufgestelltes Optimierungsproblem darin, für gegebene Anfangs- und Endzeiten t_0 und t_1 die Funktion

$$J = f(\mathbf{x}(t_1), \mathbf{v}(t_1)) + \int_{t_0}^{t_1} g(\mathbf{x}(t), \mathbf{v}(t), t)\, dt$$

unter Berücksichtigung der Newton'schen Bewegungsgleichungen zu minimieren. Dabei stellt g eine Kostenfunktion dar, z.B. die Summe der aufgewandten Kräfte, und f ein Funktional, das am Ende des Simulationszeitraums ausgewertet wird, z.B. die Differenz zu einem vorgegebenen Zielzustand $\mathbf{x}(t_1) - \mathbf{x}^{Ziel}$.

Man kann außerdem zwischen lokalen und globalen Optimierungsverfahren unterscheiden. Während erstere von einem bestimmten Anfangszustand ausgehen, um auf iterative Weise zu einer optimalen Lösung zu gelangen, operieren letztere auf dem gesamten Zustandsraum und suchen nach einer global optimalen Lösung.

In der Computeranimation wurden Optimierungsverfahren häufig wie in [Gir91], [P+92b], [P+92a], [P+93] für die Nachbildung menschlicher oder tierischer Bewegungen angewandt. Sie lassen sich aber auch für viele andere graphische Anwendungen einsetzen (vergl. [GB93]). In [P+90a] werden Controller angewandt, deren Parameter mit Hilfe eines Optimierungsverfahrens bestimmt werden, um den Bewegungsablauf so schnell wie möglich zu gestalten und gleichzeitig die Summe der aufgewandten Controller-Energien zu minimieren. Durch den Einsatz von Sensoren und Aktuatoren (Kontrolleinheiten, die bestimmte Bewegungen bewirken) können auf diese Weise sogar automatisch interessante Bewegungsmuster erzeugt werden ([PF93]).

Ein spezielle Verwendung von Optimierungsverfahren stellt das in [WK88] vorgestellte und in [Coh92], [NM93] weiterentwickelte Konzept der „Spacetime Constraints" dar. Die Bewegungsgleichungen werden hier nicht als dynamische Entwicklungsgleichungen, sondern als Nebenbedingungen angesehen, die zu jedem Zeitpunkt und an jedem Ort (daher der Name „Spacetime") erfüllt sein müssen. Zusätzlich können kinematische Nebenbedingungen spezifiziert werden, z.B. in Form von Start- und Zielkonfigurationen. Die Generierung der Bewegung erfolgt dann durch einen Optimierungsprozess. Für die Bewegung einer springenden Tischlampe, die einen hohen Bekanntheitsgrad erlangt hat, wurde in [WK88] z.B. die Energie der „Muskel"-Kräfte minimiert, die die Gelenke der Lampe auslenken. Allerdings erfordert diese Methode schon bei relativ einfachen Systemen einen extrem großen Rechenzeitaufwand, da die Bewegungsgleichungen zu jedem Zeitpunkt der Animation als Nebenbedingungen berücksichtigt werden müssen. Als sehr nützlich hat sich dieser Ansatz aber in jüngster Zeit für die Modifikation und Kombination vorhandener Bewegungsabläufe erwiesen ([Gle97], [PW99], [Pop00]).

Optimierungsverfahren erlauben keinen interaktiven Eingriff in den Prozess der Bewegungsgenerierung. Sie gehören damit zu *Offline*-Verfahren im Gegensatz zu den *Online*-Verfahren der auf Controllern und Constraints basierenden Steuerung ([H+96])[5]. Hierfür gibt es zwei Gründe:

- Zur Lösung des Optimierungsproblems müssen iterative Verfahren eingesetzt werden, die äußerst zeitaufwendig sein können. Ein Bewegungsgenerierung in Fast-Echtzeit ist für typische

[5] Für den Rahmen der physikalisch basierten Modellierung wurden in [Sur92b], [Sur92a], [NC99] allerdings interaktive Anwendungen des Optimierungsverfahrens vorgestellt.

in der Computeranimation eingesetzte Systeme daher in der Regel nicht möglich.

- Da die Kostenfunktion über den gesamten Animationsbereich minimiert wird, darf die Szene während der Bewegungsgenerierung nicht geändert werden.

Eine Kopplung mit anderen Verfahren zur Bewegungssteuerung ist aus diesem Grund sehr schwierig. Insbesondere führen ereignisgesteuerte Aktionen bei diesem Ansatz zu grundsätzlichen Schwierigkeiten. Als allgemeine Grundlage für die interaktive Erstellung von Animationen sind Optimierungsverfahren daher kaum geeignet.

2.3.2.4 Constraints

Unter *Constraints* versteht man ganz allgemein Bedingungen oder Beschränkungen bezüglich der Zustandsvariablen eines Systems. Sie werden daher auch als *Zwangsbedingungen* bezeichnet. Im Rahmen der physikalisch basierten Animationhaben sie die Interpretation von kinematischen Nebenbedingungen, die bei der Simulation berücksichtigt werden müssen.

Ein Beispiel eines Constraints ist das Vorhandenseins eines „Bodens“, der für die Objekte der Szene undurchdringlich sein soll. Mit Constraints lassen sich aber auch kinematische Beziehungen zwischen einzelnen Körpern formulieren. Insbesondere stellen sämtliche Gelenkarten wie Kugel-, Scharnier- oder Gleitgelenke Beispiele für Constraints dar. Für die Modellierung komplexer mechanischer Systeme sind sie daher ein unverzichtbares Hilfsmittel.

Constraints stellen aber auch eine sehr allgemeine Möglichkeit zur flexiblen Steuerung der Körperbewegungen dar. Vorgaben in Form von Bewegungsendpunkten, unpassierbaren Bewegungsgrenzen und exakten Pfadvorgaben für Teile des Systems sind Beispiele für diese Art der Steuerung. Die Grenze zur Modellierung ist dabei fließend. So könnte man einen Constraint, der zwei Körper durch ein Kugelgelenk verbindet, auch als Steueranweisung an zwei einzelne Körper ansehen, deren Bewegung auf diese Weise eingeschränkt wird. Die Notwendigkeit einer Trennung zwischen Modellierung und Bewegungssteuerung, wie sie sich bei vielen anderen Steuertechniken ergibt, besteht daher nicht, was einen großen Vorteil Constraint-basierter Verfahren darstellt, da sich diese beiden Teilaufgaben hierdurch in einheitlicher Weise bearbeiten lassen.

Die Verwendung von Constraints als Modellierungshilfen, insbesondere für die Beschreibung von Gelenkkörpern, zieht sich durch praktisch sämtliche Arbeiten zur physikalisch basierten Animation mechanischer Systeme (siehe hierzu die in den vorigen Abschnitten referenzierten Arbeiten). Verbreitet sind sie auch als Hilfsmittel für Zeichenprogramme (vergl. [Gle93], [Gle94b], [Alp93]) und für Anwendungen im Bereich des Computer Aided Design (vergl. [AM96], [HG96]). Als Steuertechnik wurden Constraints erstmals in [IC87], [IC88] als allgemeines Steuerverfahren und in [BC89] als Hilfsmittel für die Animation des menschlichen Ganges eingesetzt. Die rein kinematische Steuerung von Gelenkkörpern mit Hilfe von Constraints wurde z.B. in [KB82], [B$^+$87], [P$^+$90b], [PB91] beschrieben.

Es sollen nun einige typische Constraints vorgestellt und ihre mathematische Beschreibung erläutert werden.

In Abbildung 2.5 ist eine schiefe Ebene dargestellt, auf der sich ein punktförmiges Teilchen befindet. Der Constraint soll nun darin bestehen, daß sich das Punktteilchen nur auf dieser Ebene bewegen darf. Wie eine einfache geometrische Überlegung zeigt, ist dies genau dann der Fall, wenn der Ortsvektor des Teilchens $\mathbf{r} = (r_x, r_y, r_z)$ der folgenden Beziehung genügt:

$$r_x \cdot \tan\varphi = r_y \, .$$

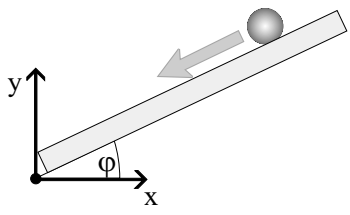

Abbildung 2.5: Eine schiefe Ebene.

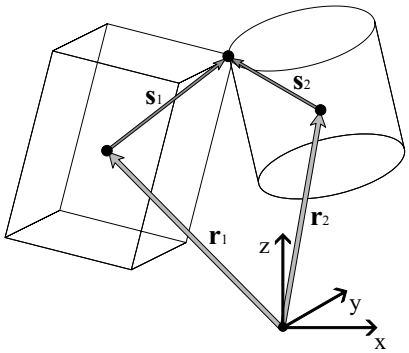

Abbildung 2.6: Ein Kugelgelenk.

Ein anderes einfaches Beispiel ist in Abbildung 2.6 skizziert.

Die beiden starren Körper mit den Schwerpunkt-Ortsvektoren $\mathbf{r}_1$ bzw. $\mathbf{r}_2$ sollen dem Constraint unterliegen, sich an den Körperpunkten mit den relativen Koordinaten $\mathbf{s}_1$ bzw. $\mathbf{s}_2$ zu berühren. Auf diese Weise kann ein Kugelgelenk an diesem gemeinsamen Punkt spezifiziert werden. Wenn die Rotation eines starren Körpers bezüglich des Schwerpunktes mit dem Ortsvektor $\mathbf{r}$ durch die Drehmatrix $\mathbf{R}$ beschrieben wird (siehe Abschnitt C), hat ein körperfester Punkt mit dem (zeitlich konstanten) relativen Ortsvektor $\mathbf{s}$ im Weltkoordinatensystem die Form $\mathbf{r}+\mathbf{R}\,\mathbf{s}$. Der hier festgelegte Constraint entspricht daher der Beziehung

$$\mathbf{r}_1+\mathbf{R}_1\,\mathbf{s}_1=\mathbf{r}_2+\mathbf{R}_2\,\mathbf{s}_2\ .$$

Für Gelenk-Constraints, bei denen die beiden Körper einen Flächenkontakt aufrecht erhalten, lassen sich genau 6 verschiedene Typen angeben (siehe z.B. [RS88], S. 104). Neben dem Kugel- und Scharniergelenk sind dies prismatische Gelenke, die eindimensionale translatorische Relativbewegungen erlauben, Schraubengelenke, die eine ebenfalls eindimensionale schraubenförmige Relativbewegung zulassen, zylindrische Gelenke, die den prismatischen Gelenken ein Rotationsfreiheitsgrad senkrecht zur Translationsachse hinzufügen, und planare Gelenke, die man durch den Kontakt zweier planarer Flächen modellieren kann und die zwei Translations- und einen Rotationsfreiheitsgrad besitzen.

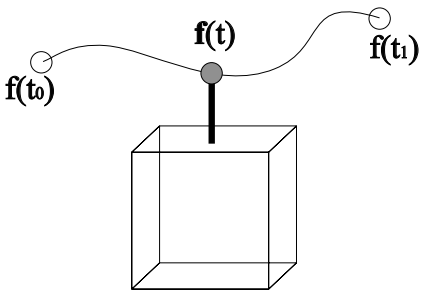

Abbildung 2.7: Ein explizit zeitabhängiger Constraint.

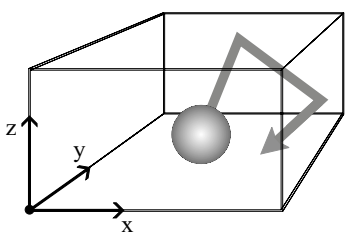

Abbildung 2.8: Ein Teilchen in einer quaderförmigen Box.

Constraints dürfen auch explizit von der Zeit abhängen. Sie heißen dann *rheonom* im Gegensatz zu nicht explizit zeitabhängigen *skleronomen* Constraints. So ist der Aufhängepunkt des in Abbildung 2.7 dargestellten Körpers dem Constraint unterworfen, der Trajektorie $\mathbf{f}=\mathbf{f}(t)$ zu folgen. Wenn $\mathbf{s}$ der relative Ortsvektor dieses Aufhängepunktes ist, ergibt sich demnach analog zum Kugelgelenk-Constraint

$$\mathbf{r}+\mathbf{R}\,\mathbf{s}=\mathbf{f}(t)\ .$$

Die festgelegten Bedingungen können auch in Form von Ungleichungen vorliegen. Ein solches Beispiel, das in Abbildung 2.8 skizziert ist, ist die Forderung an ein Teilchen, sich innerhalb einer quaderförmigen Box zu bewegen:

$$\mathbf{r}(t) - \mathbf{r}_{max} < 0 \,.$$

Holonome Constraints. Die meisten der bislang vorgestellten Constraints lassen sich als Funktion der Zustandsvariablen $\mathbf{x}$ und der Zeit t schreiben. Allgemein haben sie also die Form

$$\mathbf{C}(\mathbf{x},t) = 0 \,. \tag{2.1}$$

Bei den obigen Beispielen würde diese Gleichung z.B. mit den Festlegungen

$$\begin{aligned} C &:= r_x \tan\varphi - r_y \quad \text{(schiefe Ebene)} \\ \mathbf{C} &:= \mathbf{r}_1 + \mathbf{R}_1\,\mathbf{s}_1 - \mathbf{r}_2 - \mathbf{R}_2\,\mathbf{s}_2 \quad \text{(Kugelgelenk)} \\ \mathbf{C} &:= \mathbf{r} + \mathbf{R}\,\mathbf{s} - \mathbf{f}(t) \quad \text{(Aufhängepunkt)} \end{aligned}$$

resultieren. Solche Constraints, die auch sämtliche Gelenkverbindungen umfassen, nennt man *holonom*.

Eine sehr wichtige Eigenschaft dieser Constraints ist die aus ihrer Anwendung resultierende Einschränkung der Freiheitsgrade des Systems. Ein Punktteilchen im dreidimensionalen Raum hat z.B. drei Freiheitsgrade, die den drei Komponenten seines Ortsvektors entsprechen. Durch die Festlegung auf eine schiefe Ebene geht ein Freiheitsgrad verloren, das Teilchen wird auf einen zweidimensionalen Unterraum gezwungen. Der Zustand des Teilchens ließe sich damit durch nur zwei Zustandswerte beschreiben. Der oben vorgestellte Kugelgelenk-Constraint reduziert das freie System sogar um drei Freiheitsgrade, da die Constraint-Funktion $\mathbf{C}(\mathbf{x},t)$ aus drei voneinander unabhängigen Komponenten besteht.

Nichtholonome Constraints. Es gibt allerdings auch Constraints, die sich nicht in die Form von Gleichung (2.1) bringen lassen. Solche *nichtholonomen* Constraints können z.B. nur als Ungleichungen $\mathbf{C}(\mathbf{x},t) > 0$ formuliert werden wie bei dem Beispiel des in der Box eingesperrten Teilchens. Auch explizite Bedingungen bezüglich der zeitlichen Ableitungen der Zustandsvariablen in der Form $\mathbf{C}(\mathbf{x},\dot{\mathbf{x}},t) = 0$ gehören zu dieser Klasse[6]. Mit solchen Constraints können explizite Vorgaben für die Objekt-*Geschwindigkeiten* festgelegt werden. Die Forderung

$$\dot{r}_y - 2\,\dot{r}_x = 0$$

an ein Teilchen mit Ortsvektor $\mathbf{r}$ ist hierfür ein Beispiel.

Der oben erwähnte Zusammenhang mit den Freiheitsgraden des Systems besteht bei nichtholonomen Constraints nicht. Die Bewegung des in der Box eingesperrten Teilchens wird z.B. im Gegensatz zur schiefen Ebene nicht auf einen zweidimensionalen, sondern auf einen dreidimensionalen Unterraum eingeschränkt und muß daher wie im freien Systems durch drei Zustandswerte beschrieben werden. Nichtholonome Constraints sind daher in der Regel viel schwieriger zu behandeln als holonome.

2.3.2.5 Ereignisbasierte Prozeduren

Steuertechniken auf der Basis von Controllern und Constraints sind zustandsbasierte Verfahren, die in stetiger Weise von den aktuellen Werten der Zustandsvariablen abhängen. Eine ereignisbasierte

[6] Teilweise wird diese Form als allgemeine Definitionsgleichung für nichtholonome Constraints angegeben (z.B. in [Wit77] und [GB94]). In dieser Arbeit wird der Begriff aber wie in [Gol89] und [Sha98] in dem oben beschriebenen allgemeineren Sinne verwandt.

Steuerung wird dagegen nur bei genau festgelegten Bedingungen für einen bestimmten Zeitraum aktiv. Ein wichtiges Beispiel hierfür ist die Simulation von Kollisionen starrer oder deformierbarer Körper. Im Falle einer Kollision müssen hier Kollisionskräfte oder Kraftstöße angewandt werden, um ein realistisches Verhalten zu erzielen (vergl. Abschnitt 2.4.2). Sowohl Controller als auch Constraints können bei ereignisbasierten Prozeduren aber als Hilfsmittel eingesetzt werden, um das gewünschte Verhalten zu erzielen.

2.3.3 Eignung der Techniken für allgemeine Animationssysteme

Es stellt sich nun die Frage, welche der hier vorgestellten Steuertechniken für den Einsatz in allgemeinen Animationssystemen geeignet sind. Die Beantwortung dieser Frage muß unter Berücksichtigung der in der Einführung dieser Arbeit genannten Anforderungen erfolgen, die an derartige Systeme im Rahmen der physikalisch basierten Animation zu stellen sind.

Optimierungsverfahren werden den Ansprüchen aus mehreren Gründen nur sehr eingeschränkt gerecht. Zum einen ist ihre allgemeine Anwendung äußerst schwierig. Da sie auf einer „richtig“ gewählten Kostenfunktion beruhen, müssen sie für jedes neue System und jeden neuen Bewegungstyp geeignet angepaßt werden. Die Auswahl der Parameter, die bei der Animation zu minimieren oder zu maximieren sind, um das gewünschte Bewegungsverhalten zu erzielen, stellt zudem eine hochgradig nichttriviale Aufgabe dar. Ihre Durchführung kann außerdem einen sehr hohen Zeitaufwand erfordern und damit dem Ziel der Effektivität entgegenstehen. Am schwersten wiegt aber die in Abschnitt 2.3.2.3 erwähnte Unverträglichkeit mit Änderungen des Systemzustandes während der Simulationszeit. Ereignisbasierte Prozeduren wie die automatische Kontaktbehandlung, aber auch interaktive Eingriffe in den Simulationsvorgang durch den Animateur sind daher kaum mit Optimierungsverfahren zu kombinieren, so daß ihr Einsatz zu einer erheblichen Beschränkung der Eingriffsmöglichkeiten führen würde.

Auch Controller haben nicht den Grad an Allgemeinheit und Flexibilität, um als alleinige Grundlage zur Steuerung physikalisch basierter Bewegungen dienen zu können. Die mühsame Anpassung der Kontrollparameter und vor allem die fehlende Berücksichtigung von dynamischen Abhängigkeiten macht ihre Anwendung für viele Systeme äußerst schwierig. Im Gegensatz zu Optimierungsverfahren können sie aber problemlos mit anderen Steuertechniken kombiniert werden und erlauben auch Änderungen des Systemzustandes zur Simulationszeit. Als einfache Kraftfunktionen, z.B. in Form gedämpfter Federn, können Controller zudem intuitiv erfaßbare Grundeinheiten darstellen, die in sehr vielen physikalischen Systemen sinnvoll eingesetzt werden können.

Den höchsten Grad an Flexibilität bieten Steuerverfahren auf der Grundlage von Constraints. Die vielfältigen Möglichkeiten, die Constraints für die Modellierung komplexer Systeme und die Steuerung von Bewegungen bieten, wurden in Abschnitt 2.3.2.4 bereits angedeutet. Für den Einsatz in allgemeinen Animationssystemen sind sie zudem durch folgende Vorteile ausgezeichnet:

- Flexibilität
  Vorgaben in Form von Constraints stellen allgemeine Bedingungen an die Zustandsvariablen des Systems dar, so daß ein enormes Spektrum von Einflußmöglichkeiten erzielt werden kann.
- Generalität
  Es gibt im Grunde kein System, für das sich Constraints nicht anwenden lassen. Einfache Constraint-Typen, wie z.B. das Aufeinanderfallen zweier Körperpunkte, stellen zudem allgemeine Steuereinheiten dar, die für eine Vielzahl von Systemen in unveränderter Form und mit einer gleichbleibenden Interpretation angewandt werden können.

- Einheitlichkeit
  Constraints lassen sich für die Objektmodellierung und für die Bewegungssteuerung in gleicher Weise einsetzen und ermöglichen somit einen einheitlichen Zugang für diese beiden grundlegenden Teilaufgaben.

Auch auf Controller und Constraints aufbauende ereignisbasierte Prozeduren sind für eine flexible Bewegungssteuerung sehr wichtig. Obwohl ihre Einbindung in physikalisch basierte Systeme nicht trivial ist, insbesondere in der Kombination mit anderen Steuertechniken, stellt vor allem die automatische Simulation von Körperkontakten ein äußerst wichtiges Hilfsmittel bei der Animationserstellung dar.

Diese Einschätzung läßt sich auch für den in Abschnitt 2.3.1 beschriebenen Einsatz neuronaler Netze treffen. Der Trainingsprozess stellt einen relativ zeitaufwendigen Offline-Prozess dar und ist nur auf das als Vorlage verwandte System zugeschnitten ([G+98]). Die „interessanten“ Konfigurationen des Systems, die in das Training des Netzes als diskrete Eingabedaten eingehen, müssen zudem geeignet ausgewählt werden, was bei komplexen Systemen äußerst schwierig erscheint.

Wie den Ausführungen dieses Abschnitts entnommen werden kann, stellen Constraints eine besonders wichtige Technik zur Objektmodellierung und zur interaktiven Bewegungssteuerung dar. Sie bilden auch die Grundlage für die Realisierung von Steuertechniken mit einem höheren Abstraktionsgrad, wie die in Abschnitt 2.3.2.5 besprochenen ereignisbasierten Prozeduren. Die schwierige Aufgabe der Constraint-Behandlung, d.h. die Einbindung von Constraints in ein dynamisches System, wird in Kapitel 3 ausführlich besprochen.

Zuvor soll aber eine der wichtigsten Einsatzmöglichkeiten der physikalisch basierten Animationvorgestellt werden: Die automatische Behandlung von Kontakten starrer Körper. Dieser Einsatz stellt keine eigenständige Steuertechnik dar, sondern entspricht einer ganz speziellen Aufgabenstellung, zu deren Lösung bestimmte Steuertechniken wie Constraints oder ereignisbasierte Prozeduren eingesetzt werden können.

2.4 Automatische Kontaktbehandlung

Die automatische Behandlung von Kontakten starrer Körper ist eine der am häufigsten angewandten Methoden im Rahmen der physikalisch basierten Animation. Vorrangiges Ziel ist dabei die Vermeidung von Körperdurchdringungen, die ein äußerst unplausibel wirkendes Bewegungsverhalten darstellen. Im Falle eines Zusammenstoßes zweier Körper soll zudem eine realistisch wirkende Kollision erfolgen. Klassische Arbeiten zum Einsatz solcher Techniken im Bereich der Computeranimation sind z.B. [MW88], [Hah88], [Bar89], [Bar94]. Ein aktueller Ansatz für eine rein geometrische Vermeidung von Körperdurchdringungen, bei der die dynamischen Objekteigenschaften nicht berücksichtigt werden, wird in [CC00] beschrieben.

Es soll nun erläutert werden, welche grundsätzlichen Probleme bei der Kontaktbehandlung starrer Körper auftreten und welche Ansätze zur Lösung dieser Probleme existieren. Einige Ableitungen und weiterführende Aspekte werden dabei in Anhang D besprochen.

Der erste Schritt bei der automatischen Behandlung von Körperkontakten ist die *Erkennung* von Kontakten und die Bestimmung der Kontaktgeometrie. Diese beiden komplexen Probleme werden mit dem Begriff *Kontaktbestimmung* zusammengefaßt[7]. Als Ergebnis der Kontaktbestimmung erhält man die Information, ob zwei Körper miteinander in Kontakt stehen, und gegebenenfalls die

[7] Im englischen Sprachgebrauch werden hierfür die Begriffe *collision detection* und *contact determination* synonym verwandt ([LG98]).

zur automatischen Behandlung notwendigen Kontaktdaten. Diese Daten müssen für jeden Kontaktpunkt (ein Kontakt zweier Körper kann mehrere Kontaktpunkte beinhalten) die folgenden Größen enthalten: Die relativen Koordinaten $\mathbf{r}_A$ und $\mathbf{r}_B$ der jeweiligen Körperpunkte $\mathbf{P}_A$ und $\mathbf{P}_B$, die zum Kontaktzeitpunkt zusammenfallen, und die senkrecht zur Kontaktebene stehende Flächennormale $\mathbf{n}$ (siehe Abbildung 2.9).

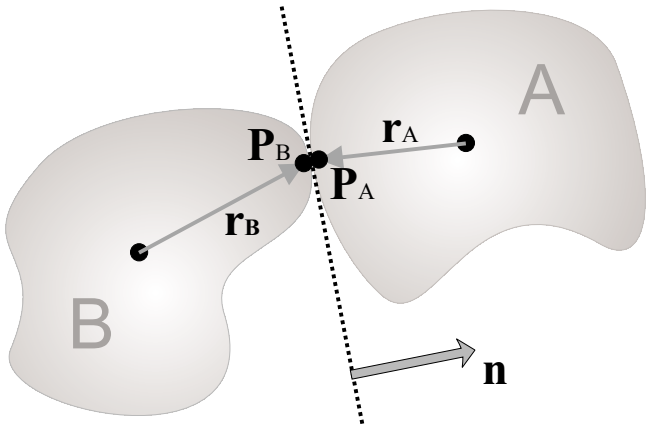

Abbildung 2.9: Ein Punktkontakt zweier Körper. Die Körperpunkte $\mathbf{P}_A$ und $\mathbf{P}_B$ berühren sich in der Ebene mit der Normalen $\mathbf{n}$.

Auf der Grundlage dieser Informationen läßt sich eine *Relativgeschwindigkeit* in der folgenden Form definieren:

$$v_{rel} := \mathbf{n} \cdot (\dot{\mathbf{P}}_A - \dot{\mathbf{P}}_B) \; .$$

Dabei stellen $\dot{\mathbf{P}}_A$ und $\dot{\mathbf{P}}_B$ die Geschwindigkeiten der jeweiligen Körperpunkte dar, die i.a. zum Kontaktzeitpunkt nicht übereinstimmen. Der Skalar v_{rel} bezeichnet demnach die relative Geschwindigkeit der beiden Körper am Kontaktpunkt in Richtung der Kontaktnormalen. Es können nun drei verschiedene Fälle auftreten (dabei soll angenommen werden, daß $\mathbf{n}$ in die Richtung des Körpers A zeigt):

- $v_{rel} > 0$
  Bei einer positiven Relativgeschwindigkeit bewegen sich die Körper voneinander weg, so daß keine Kollision auftritt. Eine Kontaktbehandlung ist in diesem Fall nicht notwendig.
- $v_{rel} < 0$
  Eine negative Relativgeschwindigkeit bedeutet, daß sich die Körper aufeinanderzu bewegen, es liegt also eine Kollision (ein Stoß) vor.
- $v_{rel} = 0$
  Bei einer verschwindenden Relativgeschwindigkeit sind die Körper relativ zueinander in Ruhe oder bewegen sich nur in der gemeinsamen Kontaktebene.

Zunächst soll das Problem der Kontaktbestimmung besprochen werden. Anschließend wird erläutert, wie sich Kollisionen auf realistisch wirkende Weise simulieren lassen, wobei der Begriff *Kollision* nur für Kontakte mit einer nichtverschwindenden Relativgeschwindigkeit angewandt werden soll. Das komplizierte Problem der Ruhekontaktbehandlung wird dann im daran anschließenden Abschnitt besprochen. Auf diese drei Teilaufgaben soll sich im Rahmen dieser Arbeit auch der Begriff *Kontaktbehandlung* beziehen. Sowohl die Kollisionssimulation als auch die Ruhekontaktbehandlung haben dabei die Kontaktbestimmung als Voraussetzung.

2.4.1 Kontaktbestimmung

Das Problem der Kontaktbestimmung unterteilt sich wie schon erwähnt in zwei Teilaufgaben: die Detektion eines Körperkontaktes, der auch als eine Durchdringung vorliegen kann, und die Bestimmung der Kontaktgeometrie in Form der in Abbildung 2.9 dargestellten Größen, die für die Behandlung des Kontaktes benötigt werden.

Die Durchführung dieser Aufgaben kann sich dabei als sehr schwierig erweisen. Die Kontaktbestimmung im Rahmen einer physikalisch basierten Animation stellt im Gegensatz zu einem *statischen* ein *dynamisches* Problem dar, da die raumzeitlichen Bahnen der Körper zum Simulationsbeginn unbekannt sind. Bei einer schrittweise durchgeführten Simulation kann daher z.B. das Problem auftreten, daß der Kontakt zweier Körper erst dann detektiert wird, wenn bereits eine Durchdringung stattgefunden hat. In diesem Fall läßt sich eine iterative Annäherung an den exakten Kontaktzeitpunkt durchführen, die allerdings einen zusätzlichen Zeitaufwand erfordert, der die Interaktivität bei der Erstellung der Animation entscheidend einschränken kann. Einfacher ist daher oftmals eine Verkleinerung der Schrittweite bei der Simulationsdurchführung, die allerdings ebenfalls zu Lasten der zeitlichen Effizienz geht.

Es soll nun umrissen werden, mit welchen Techniken eine effiziente Kontaktbestimmung durchgeführt werden kann.

In einem System mit n Körpern erfordert die Kontaktbestimmung im allgemeinen Fall einen zu n quadratischen Zeitaufwand. Praktisch läßt sich aber in den meisten Fällen eine lineare Abhängigkeit von n erreichen, da in der Regel nur ein kleiner Prozentsatz der Körper miteinander in Kontakt steht. Ein grundlegender Ansatz zur schnellen Ermittlung der *nicht* miteinander in Kontakt stehenden Körper ist die Festlegung von umschließenden Körpern (*bounding volumes*), z.B. Quader oder Kugeln, deren jeweilige Durchdringungen sehr schnell getestet werden können. In [C+95a] und [Bar95a] werden z.B. achsenparallele Quader eingesetzt, deren Überlappungen durch eine Sortierung der Quaderkoordinaten ermittelt werden. Ein zusätzlicher Effizienzgewinn läßt sich oftmals durch eine Raumzerlegung erzielen, durch die der Kontakt von Körpern, die sich nicht in der gleichen oder in benachbarten Raumzellen befinden, ausgeschlossen werden kann. Eine Übersicht über diese Techniken wird in [LG98] gegeben.

Für die Kontaktbestimmung dynamisch veränderlicher Körper, also insbesondere im Rahmen der physikalisch basierten Animation, ist es außerdem sehr wichtig, zeitliche Kohärenzen auszunutzen. Die Position eines Körpers ändert sich von einem Simulationsschritt zum nächsten in der Regel nur sehr wenig oder bleibt sogar unverändert. Diese Eigenschaft läßt sich als geometrische Kohärenz ausnutzen, indem z.B. durch die Analyse der relativen Geschwindigkeiten und Beschleunigungen ermittelt wird, welche Körper sich im darauffolgenden Zeitschritt nicht berühren können.

Die exakte Bestimmung der Kontaktgeometrie für die durch diese Techniken nicht aussortierten Körperpaare hängt dann sehr stark von der geometrischen Struktur ab, in der die Körper vorliegen. In [LG98] findet sich eine Übersicht über derartige Techniken. Sehr viele Bestimmungsverfahren sind z.B. auf polygonale Körper und somit auf konvexe oder konkave Polyeder oder auf unstrukturierte Polygon-Mengen (*polytope soups*) zugeschnitten. Auch zu diesem Zweck lassen sich neben einer ganzen Reihe anderer Verfahren hierarchisch strukturierte Begrenzungskörper einsetzen (siehe z.B. [V+98]). In [LG98] werden einige frei verfügbare Bibliotheken vorgestellt, die eine automatische Kontaktbestimmung ermöglichen und insbesondere für den Einsatz im Rahmen der physikalisch basierten Animation konzipiert wurden.

2.4.2 Kollisionssimulation

Es soll nun untersucht werden, wie das Kollisionsverhalten zweier Körper simuliert werden kann. Man unterscheidet dabei zwischen *zentrischen* und *exzentrischen* Stößen. Im Gegensatz zu exzentrischen Stößen verlaufen die Körpergeschwindigkeiten bei zentrischen Stößen (unmittelbar vor dem Stoß) in Richtung der jeweiligen Massenschwerpunkte. Bei zentrischen Stößen werden keine Dreh-

momente frei, d.h. keine Kräfte, die die Rotationsgeschwindigkeit der Körper beeinflussen [8].

Es soll nun das in [Bar95b] erläuterte Verfahren zur Kollisionssimulation vorgestellt werden, das auch für die automatische Kontaktbehandlung in dem Animationssystem, das im Rahmen dieser Arbeit entwickelt wurde, zum Einsatz kommt (siehe Abschnitt 6.4). Es läßt sich leicht anwenden und führt zu einem sehr realistisch wirkenden Kollisionsverhalten.

Die Dauer des Stoßvorganges Δt ist bei weicheren Materialien länger als bei härteren, während die auftretenden Kontaktkräfte $\mathbf{F}^{Koll}$ bei sehr harten Körpern am größten sind. Für die Simulation unverformbarer starrer Körper kann man sich die durch die Kollision verursachte Abstoßungskraft daher als den Grenzfall einer unendlich großen Kraft vorstellen, die in einem unendlich kleinen Zeitintervall wirkt. Für die Beschreibung dieses Prozesses verwendet man den Begriff des *Kraftstoßes* (siehe z.B. [Bar95b])

$$\mathbf{K} := \lim_{\Delta t \to 0} (\mathbf{F}^{Koll} \cdot \Delta t) \, .$$

Er hat die Dimension eines Impulses und stellt die durch den Stoß verursachte *Impulsdifferenz* dar. Bei einem Körper mit der Masse m und der anfänglichen Geschwindigkeit $\mathbf{v}^0$ bewirkt er eine Geschwindigkeitsänderung in der Form

$$\mathbf{v} = \mathbf{v}^0 + \frac{1}{m}\mathbf{K} \, . \tag{2.2}$$

Diese Größe $\mathbf{K}$ muß demnach bestimmt werden, um die modifizierten Geschwindigkeitswerte zu erhalten. Wie in Anhang D hergeleitet wird, hängt der Kraftstoß von den Massen und den anfänglichen Geschwindigkeiten der kollidierenden Körper ab. Bei einem zentrischen Stoß ergibt sich der folgende Ausdruck:

$$\mathbf{K} = -\frac{(1+\varepsilon)\, v^0_{rel}}{\frac{1}{m_A} + \frac{1}{m_B}} \mathbf{n} \, . \tag{2.3}$$

Dabei ist v^0_{rel} die Relativgeschwindigkeit vor dem Stoß, m_A und m_B sind die beiden Körpermassen, $\mathbf{n}$ bezeichnet die Kontaktflächennormale und ε stellt die sogenannte *Stoßzahl* dar. Die Stoßzahl hängt in erster Linie (aber nicht nur) vom Material der Körper ab und ist ein Maß für die Elastizität der Kollision. Ihr Wert liegt zwischen 0 und 1, wobei $\varepsilon = 0$ einem vollkommen unelastischen und $\varepsilon = 1$ einem vollkommen elastischen Stoß entspricht.

Beim exzentrischen Stoß müssen zusätzlich die Trägheitstensoren $\mathbf{I}_A$ und $\mathbf{I}_B$ der Körper berücksichtigt werden (vergl. Anhang C) und man gelangt auf den folgenden Ausdruck (siehe Anhang D):

$$\mathbf{K} = -\frac{(1+\varepsilon)\, v^0_{rel}}{\frac{1}{m_A} + \frac{1}{m_B} + \mathbf{n}\left((\mathbf{I}_A^{-1}(\mathbf{d}_A \times \mathbf{n})) \times \mathbf{d}_A\right) + \mathbf{n}\left((\mathbf{I}_B^{-1}(\mathbf{d}_B \times \mathbf{n})) \times \mathbf{d}_B\right)} \mathbf{n} \, . \tag{2.4}$$

Dabei ist $\mathbf{d}_A := \mathbf{r} - \mathbf{r}^A_{SP}$ und $\mathbf{d}_B := \mathbf{r} - \mathbf{r}^B_{SP}$ mit den Schwerpunktsvektoren $\mathbf{r}^A_{SP}$ und $\mathbf{r}^B_{SP}$ und dem Ortsvektor des Kontaktpunktes $\mathbf{r}$. In diesem Fall müssen auch die Winkelgeschwindigkeiten neue Werte erhalten. Hierzu dient die Beziehung

$$\omega = \omega^0 + \mathbf{I}^{-1}\left((\mathbf{r} - \mathbf{r}_{SP}) \times \mathbf{K}\right) . \tag{2.5}$$

Diese Gleichung und Gleichung (2.2) gelten bei den hier getroffenen Festlegungen für den Körper A, beim Körper B ist der Term $\mathbf{K}$ durch den Term $-\mathbf{K}$ zu ersetzen (siehe Anhang D).

Mit Hilfe dieses Ansatzes ist es demnach möglich, auf analytische Weise einen Kraftstoß zu bestimmen und mit seiner Hilfe eine instantane Veränderung der Linear- und Winkelgeschwindigkeiten

[8] Bei einem *geraden* Stoß bewegen sich die Körper zudem senkrecht zur Kontaktebene, bei *schiefen* Stößen ist das nicht der Fall.

vorzunehmen. Numerische Verfahren werden dabei nicht benötigt, so daß sich dieses Verfahren äußerst effizient durchführen läßt.

Äußere Kräfte wie die Schwerkraft werden dabei nicht berücksichtigt. Bei der Kollisionssimulation starrer Körper spielen sie aufgrund der infinitesimal kleinen Kontaktzeit keine Rolle, so daß ihre Einbindung einen völlig anderen Ansatz erfordern würde. Entscheidenden Einfluß haben sie aber im Rahmen der Ruhekontaktbehandlung, die nun als nächstes besprochen werden soll.

2.4.3 Behandlung von Ruhekontakten

In diesem Abschnitt sollen Konstellationen von Körpern betrachtet werden, die sich an N Kontaktpunkten berühren und an diesen Punkten die Relativgeschwindigkeit 0 haben. Dieser Fall darf bei der automatischen Kontaktbehandlung nicht ignoriert werden, denn beim Vorhandensein äußerer Kräfte (z.B. der Schwerkraft) kann es in einem solchen System zu einer ungewollten Durchdringung der Körper im unmittelbar folgenden Zeitschritt kommen.

Hier ist demnach die Bestimmung von *Kompensationskräften* erforderlich, die den äußeren Kräften entgegenwirken und Durchdringungen verhindern (siehe Abbildung 2.10). Dieses Problem ist sehr komplex, da sich die an einem Kontaktpunkt angreifende Kraft auch auf alle anderen Kontaktpunkte auswirken kann. In [Bar89] wurde erstmals eine analytische Lösung dieses Problems vorgestellt, die in [Bar91], [Bar93] auf Kontakte mit statischer und dynamischer Reibung erweitert wurde. Ein wesentlich einfachere und effizientere Bestimmung dieser Kompensationskräfte wurde dann in [Bar94] beschrieben (vergl. auch [Bar95a]).

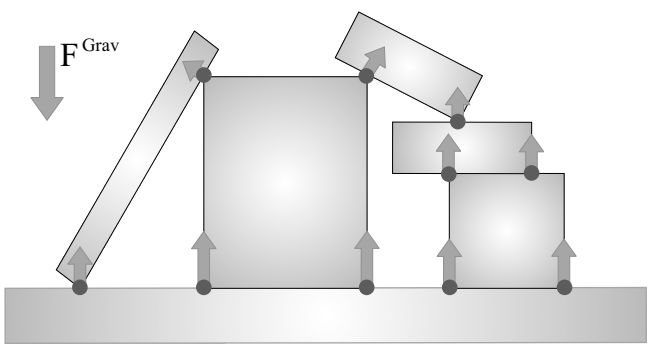

Abbildung 2.10: Kompensationskräfte bei einem Ruhekontaktproblem.

Diese kompensierenden Kräfte müssen senkrecht zur jeweiligen Kontaktebene wirken, um keine ungewollten Körperbewegungen parallel zu dieser Ebene zu erzeugen. Wenn $\mathbf{n}_i$ die jeweilige Kontaktnormale darstellt, haben diese Kräfte daher die Form

$$\mathbf{F}_i = f_i \mathbf{n}_i \quad , \qquad i = 1,\ldots,N \, .$$

Die zunächst unbekannten Skalare f_i müssen somit bestimmt werden, um eine automatische Ruhekontaktbehandlung durchführen zu können. Wie in Anhang D ausgeführt wird, können sie als Lösung eines sogenannten *Quadratic Programming*-Problems numerisch berechnet werden. Die Einbindung statischer oder dynamischer Reibungskräfte wird z.B. in [Bar94] beschrieben.

Ein ähnlicher Ansatz zur Berechnung der Kompensationskräfte wurde in [L+99] vorgeschlagen. Dabei werden aber anstatt der relativen Beschleunigungen die jeweiligen Abstandswerte als die zu den Kontaktkräften komplementären Größen behandelt, um ein stabileres Integrationsverhalten zu erzielen.

Ein anderer interessanter Ansatz ist die Kraftstoß-basierte Behandlung von Ruhekontakten, die in [Mir96b] und [MC95] vorgestellt wurde. Anstelle der Bestimmung von Kompensationskräften werden dabei „künstliche“ Kollisionen in Form von Kraftstößen zur Kontaktmodellierung simuliert, was i.a. wesentlich zeiteffizienter durchgeführt werden kann. Dieser Ansatz führt aber zu komplizierten

Problemen bei der Kombination mit Constraint-basierten Steuerverfahren, da explizite Regeln für das Umschalten zwischen den beiden Methoden spezifiziert werden müssen ([Mir95]).

Als grundlegender Ansatz zur Behandlung des Ruhekontaktproblems soll schließlich noch das Federmodell erwähnt werden. Grundidee dieses einfach anwendbaren Verfahrens ist die automatische Festlegung von lokal wirkenden Federkräften an den jeweiligen Kontaktpunkten, die ihrer Kontraktion eine Gegenkraft entgegensetzen und auf diese Weise eine Durchdringung der Körper an den jeweiligen Punkten verhindern. Sie können auch für die Simulation von Kollisionen eingesetzt werden (vergl. [MW88]). Dieses Verfahren hat allerdings mit einem grundsätzlichen Problem zu kämpfen: die Federkräfte sind nur dann in der Lage, deutlich sichtbare Durchdringungen zu verhindern, wenn sie durch eine sehr kurze Ruhelänge und einen großen Wert für die Federstärke ausgezeichnet sind. Derartige Konfigurationen führen aber zu sogenannten *steifen* Differentialgleichungen, die sich nicht mehr numerisch effizient lösen lassen (siehe Abschnitt F.1). Dieses Modell hat daher nur einen sehr eingeschränkten Anwendungsbereich.

Kapitel 3

Verfahren zur Constraint-Behandlung

In diesem Kapitel soll dargestellt werden, welche Lösungsverfahren für die Einbindung von Constraints in mechanische Systeme bekannt sind, d.h. für ihre Berücksichtigung bei physikalischen Simulationen. Zunächst wird dazu die Dynamik des freien Systems genauer untersucht und die wichtige Klasse der beschleunigungslinearen Constraints vorgestellt. Dann werden das Penalty-Verfahren, die Reduktionsmethode, die Lagrange-Faktoren-Methode und einige andere Verfahren erläutert, die grundverschiedene Ansätze zur Lösung des komplizierten Problems der Constraint-Behandlung darstellen.

Diese Übersicht, die sich im wesentlichen auf Literatur aus dem Bereich der Technischen Simulation stützt (insbesondere auf [RS88], [GB94], [Sha98]), kann dabei keinesfalls Vollständigkeit beanspruchen. Die Entwicklung und Umsetzung dieser Techniken stellt ein aktuelles Forschungsgebiet dar und konnte hier nur überblicksartig wiedergegeben werden. Wichtig ist für diese Arbeit die Erläuterung der grundlegenden Ansätze, da die hierauf basierenden Verfahren gerade vor dem Hintergrund allgemeiner Animationssysteme durch ganz spezifische Vor- und Nachteile ausgezeichnet sind, wie in Kapitel 4 ausführlich diskutiert wird.

3.1 Dynamik des freien Systems

Ein freies (nicht Constraint-behaftetes) mechanisches System läßt sich durch einen k-dimensionalen Zustandsvektor $\mathbf{x}$ beschreiben, der der Zeitentwicklung

$$\mathbf{M}(\mathbf{x})\,\ddot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}, t)$$

unterworfen ist. In der $(k \times k)$-dimensionalen Matrix $\mathbf{M}$ sind dabei die Masseneigenschaften und in dem k-dimensionalen Vektor $\mathbf{F}$ die Kraftwirkungen festgelegt. Neben äußeren Kräften wie z.B. der Schwerkraft gehören hierzu auch die im System wirkenden Inertialkräfte. Die Zustandsgrößen x_i werden als sogenannte *generalisierte* Koordinaten bezeichnet, da sie beliebige Koordinaten zur eindeutigen Festlegung des Systemzustandes darstellen können, während sich kartesische Koordinaten auf den üblichen 3D-Raum beziehen.

Dieses Differentialgleichungssystem zweiter Ordnung läßt sich auch als ein System erster Ordnung schreiben, wenn man den n-dimensionalen Geschwindigkeitsvektor $\mathbf{v}$ als zusätzliche, von $\mathbf{x}$ unabhängige Größe einführt (vergl. Anhang F.1):

$$\begin{aligned} \mathbf{M}(\mathbf{x})\,\dot{\mathbf{v}} &= \mathbf{F}(\mathbf{x},\mathbf{v},t) \\ \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x},\mathbf{v})\,. \end{aligned} \tag{3.1}$$

Die Funktion **f** spiegelt dabei die Beziehung zwischen **x** und **v** wieder. Dabei ist zu beachten, daß **M** hier die Dimension $(n \times n)$ und **F** die Dimension n hat, da die Dimensionalität von **x** und **v** verschieden sein kann.

Im Rahmen dieser Arbeit wurde die zweite Form gewählt, der nun für die Spezialfälle der massenbehafteten Punktkörper und der starren Körper eine explizite Form gegeben werden soll. Der Grund für diese Wahl wird bei der Beschreibung der starren Körper deutlich werden. Die weiteren Ausführungen dieses Kapitels beziehen sich allerdings nicht nur auf diese beiden Körpertypen, sondern auf alle mechanischen Systeme, deren Zeitentwicklung durch Gleichung (3.1) beschrieben wird.

Punktkörper. Ein massenbehafteter Punktkörper läßt sich durch lediglich drei Größen beschreiben. Sein Zustand wird durch die Angabe seines Ortsvektors **r** und seiner Geschwindigkeit $\dot{\mathbf{r}}$ festgelegt. Als einziger Parameter ist seine Masse m zu spezifizieren, die in der Regel als konstant angenommen wird.

Die für diesen Körpertyp geltende Bewegungsgleichung $\mathbf{F}^{ext} = m\ddot{\mathbf{r}}$ mit den äußeren Kräften $\mathbf{F}^{ext}$ entspricht daher unter den Festlegungen

$$\mathbf{x} := \mathbf{r}\,,\ \mathbf{v} := \dot{\mathbf{r}}\,,\ \mathbf{M} := m\mathbf{E} = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix},\ \mathbf{F} := \mathbf{F}^{ext}\,,\ \mathbf{f} := \mathbf{v}$$

der Form von Gleichung (3.1).

Starrer Körper. Die rein translatorische Bewegung des Schwerpunkts eines starren Körpers $\mathbf{r}_{SP}$ ergibt sich wie beim Punktteilchen aus der Gleichung $\mathbf{F}^{ext} = m\ddot{\mathbf{r}}_{SP}$ (vergl. Anhang C). Als komplizierter erweist sich aber die Beschreibung seines Rotationszustandes. Im Rahmen dieser Arbeit sollen hierzu Quaternionen **q** und Winkelgeschwindigkeiten ω eingesetzt werden (in Abschnitt 6.3.1 wird eine Begründung für diese Wahl gegeben). Diese Größen werden in Anhang E bzw. C genauer erläutert. Ein starrer Körper ist somit durch die Zustands- und Geschwindigkeitsvektoren

$$\mathbf{x} := \begin{pmatrix} \mathbf{r}_{SP} \\ \mathbf{q} \end{pmatrix},\ \mathbf{v} := \begin{pmatrix} \dot{\mathbf{r}}_{SP} \\ \omega \end{pmatrix}$$

gekennzeichnet, die man als *Referenzpunktkoordinaten* bezeichnet, da sie sich auf den Schwerpunkt des Körpers beziehen. Dabei ist zu beachten, daß eine Quaternion 4 Komponenten aufweist. Ein starrer Körper wird also durch einen 7-dimensionalen Zustandsvektor beschrieben, obwohl er nur 6 Freiheitsgrade hat. Der Geschwindigkeitsvektor **v** hat dagegen die Dimension 6 und ergibt sich nicht als direkte zeitliche Ableitung von **x**. Rotations-Quaternion und Winkelgeschwindigkeit sind vielmehr über die Beziehung

$$\dot{\mathbf{q}} = \frac{1}{2}\,\omega \cdot \mathbf{q}$$

miteinander verknüpft (siehe Anhang E). Die rechte Seite dieser Gleichung entspricht der Funktion **f** in Gleichung (3.1).

Die unterschiedliche Dimensionalität von **x** und **v** bei starren Körpern ist auch der Grund dafür, daß die Bewegungsgleichungen des freien Systems bezüglich $\dot{\mathbf{v}}$ und nicht bezüglich $\ddot{\mathbf{x}}$ formuliert wurden. Statt einem 6-dimensionalen müßte ansonsten bei der Anwendung auf einen einzelnen starren Körper ein 7-dimensionales Gleichungssystem gelöst werden, was schon aus Effizienzgründen vermieden werden mußte.

Die Zeitentwicklung der Rotationskomponenten wird durch die Gleichung $\mathbf{I}\ \dot{\omega} = \mathbf{N}^{ext} - \omega \times (\mathbf{I}\,\omega)$ beschrieben, wobei $\mathbf{N}^{ext}$ das äußere Drehmoment und $\mathbf{I}$ den Trägheitstensor des Körpers darstellt (siehe Anhang C). Gleichung (3.1) resultiert somit durch die Zuweisungen

$$\mathbf{M} := \begin{pmatrix} m\mathbf{E} & 0 \\ 0 & \mathbf{I} \end{pmatrix}, \quad \mathbf{F} := \begin{pmatrix} \mathbf{F}^{ext} \\ \mathbf{N}^{ext} - \omega \times (\mathbf{I}\,\omega) \end{pmatrix}.$$

Dabei ist zu beachten, daß $\mathbf{N}^{ext}$, ω und $\mathbf{I}$ nicht bezüglich des Weltkoordinatensystems, sondern bezüglich des körperfesten Koordinatensystems mit dem Körperschwerpunkt als Nullpunkt definiert sind, was ihre Interpretation und Spezifikation sehr erleichtert. Der Skalar m bezeichnet zudem die Gesamtmasse des Körpers.

Eine interessante Alternative zu dieser Beschreibung wurde in [vOB95] vorgestellt: Starre Körper werden darin durch eine (endliche) Menge von massenbehafteten Punktkörpern modelliert, die über Abstands-Constraints miteinander verbunden sind. Grundsätzliche Vorteile, wie z.B. eine Erhöhung der zeitlichen Effizienz bei der Simulation der Körper, konnten für diesen Ansatz aber nicht nachgewiesen werden.

Es stellt sich nun die Frage, wie das freie System modifiziert werden muß, um die in Abschnitt 2.3.2.4 beschriebenen Constraints berücksichtigen zu können. Zu diesem Zweck soll zunächst eine sehr allgemeine Klasse von Constraints eingeführt werden, die von den meisten der noch zu erläuternden Lösungsverfahren behandelt werden können.

3.2 Beschleunigungslineare Constraints

Unter beschleunigungslinearen Constraints versteht man Gleichungen bezüglich der Zustandsvariablen $\mathbf{x}$ eines Systems sowie ihrer ersten und zweiten zeitlichen Ableitungen $\dot{\mathbf{x}}$ und $\ddot{\mathbf{x}}$, die linear bezüglich $\ddot{\mathbf{x}}$ sind. Sie haben damit die folgende allgemeine Form[1]:

$$\mathbf{J}(\mathbf{x}, \dot{\mathbf{x}}, t) \cdot \ddot{\mathbf{x}} + \mathbf{c}(\mathbf{x}, \dot{\mathbf{x}}, t) = 0 \tag{3.2}$$

In einem n-dimensionalen System mit einem m-dimensionalen Constraint stellt $\mathbf{J}$ eine $(m \times n)$-Matrix und $\mathbf{c}$ einen m-dimensionalen Vektor dar. Ein beschleunigungslinearer Constraint ist somit durch die Angabe dieser beiden Größen $\mathbf{J}$ und $\mathbf{c}$ eindeutig festgelegt.

In Abschnitt 2.3.2.4 wurde der Begriff der *holonomen* Constraints eingeführt. Wie nun gezeigt werden soll, lassen sich diese Constraints in die Form der obigen Gleichung bringen, d.h. jeder holonome Constraint ist auch ein beschleunigungslinearer Constraint. Anschließend wird die zweite wichtige Untergruppe beschleunigungslinearer Constraints vorgestellt, die sogenannten *Geschwindigkeits-Constraints*.

Holonome Constraints. Um den Bezug zwischen holonomen Constraints und Gleichung (3.2) aufzuzeigen, soll die zweifache zeitliche Ableitung der Constraint-Funktion $\mathbf{C}(\mathbf{x}, t)$ betrachtet werden. Da die Funktion nicht nur an einem Punkt, sondern identisch gleich 0 sein soll (Gleichung (2.1)), muß diese Ableitung verschwinden. Es folgt somit

$$\begin{aligned} 0 &= \frac{d^2}{dt^2}\mathbf{C} \\ &= \frac{d}{dt}\left(\frac{\partial \mathbf{C}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \mathbf{C}}{\partial t}\right) \\ &= \frac{\partial \mathbf{C}}{\partial \mathbf{x}}\ddot{\mathbf{x}} + \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{C}}}{\partial t}. \end{aligned}$$

[1] Trotz der formalen Ähnlichkeit mit den Newton'schen Bewegungsgleichungen fließen in diese Definitionsgleichung dabei keine physikalischen Gesetzmäßigkeiten ein.

Diese Beziehung ist offensichtlich ein Spezialfall von Gleichung (3.2), womit gezeigt ist, daß holonome Constraints tatsächlich zur Klasse der beschleunigungslinearen Constraints gehören. Die Matrix **J** erhält dabei die Interpretation der Jacobi-Matrix von **C**. Wenn **C** die Dimension m und **x** die Dimension n hat, ist

$$\mathbf{J} = \frac{\partial \mathbf{C}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial C_1}{\partial x_1} & \cdots & \frac{\partial C_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial C_m}{\partial x_1} & \cdots & \frac{\partial C_m}{\partial x_n} \end{pmatrix} . \tag{3.3}$$

Für den Vektor **c** ergibt sich aus dem Vergleich mit Gleichung (3.2)

$$\mathbf{c}_j = \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{x}} \dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{C}}}{\partial t} = \sum_{i=1}^{n} \frac{\partial \dot{\mathbf{C}}}{\partial x_i} \dot{x}_i + \frac{\partial \dot{\mathbf{C}}}{\partial t} . \tag{3.4}$$

Geschwindigkeits-Constraints. Als *Geschwindigkeits-Constraints* sollen Constraints bezeichnet werden, die sich als Gleichung bezüglich der Zustandsvariablen, ihrer ersten Ableitungen und der Zeit schreiben lassen, d.h. als

$$\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}}, t) = 0 . \tag{3.5}$$

Mit diesen nichtholonomen Constraints ist es möglich, den im System definierten Geschwindigkeiten explizite Bedingungen aufzuerlegen.

Auch diese Constraints sind beschleunigungslinear. Um dies zu zeigen, soll die erste zeitliche Ableitung von **C** betrachtet werden, die aufgrund der obigen Constraint-Gleichung verschwinden muß:

$$\begin{aligned} 0 &= \frac{d}{dt} \mathbf{C} \\ &= \frac{\partial \mathbf{C}}{\partial \mathbf{x}} \dot{\mathbf{x}} + \frac{\partial \mathbf{C}}{\partial \dot{\mathbf{x}}} \ddot{\mathbf{x}} + \frac{\partial \mathbf{C}}{\partial t} . \end{aligned}$$

Gleichung (3.2) resultiert hier mit den Zuweisungen

$$\begin{aligned} \mathbf{J} &= \frac{\partial \mathbf{C}}{\partial \dot{\mathbf{x}}} \\ \mathbf{c} &= \frac{\partial \mathbf{C}}{\partial \mathbf{x}} \dot{\mathbf{x}} + \frac{\partial \mathbf{C}}{\partial t} . \end{aligned} \tag{3.6}$$

Mit Hilfe dieser Beziehungen läßt sich jeder Geschwindigkeits-Constraint in die allgemeine Form der beschleunigungslinearen Constraints überführen.

Formulierung bezüglich $\dot{\mathbf{v}}$. Beschleunigungslineare Constraints lassen sich auch bezüglich der Größen **v** und $\dot{\mathbf{v}}$ anstelle von $\dot{\mathbf{x}}$ und $\ddot{\mathbf{x}}$ formulieren:

$$\mathbf{J}(\mathbf{x}, \mathbf{v}, t) \cdot \dot{\mathbf{v}} + \mathbf{c}(\mathbf{x}, \mathbf{v}, t) = 0 . \tag{3.7}$$

Die Ausdrücke für **J** und **c**, d.h. der explizite Zusammenhang mit den Gleichungen der holonomen und der Geschwindigkeits-Constraints, ergeben sich in dieser Form ganz analog zu den obigen Ausführungen. Identisch sind diese Ausdrücke aber nur für den Fall $\dot{\mathbf{x}} = \mathbf{v}$. Mit der in Abschnitt 3.1 gewählten Notation für starre Körper hat **J** in der obigen Gleichung z.B. bei einem einzelnen Körper die Dimension $(7 \times m)$, während sich für die Matrix **J** in Gleichung (3.2) die Dimension $(6 \times m)$ ergeben würde. Dieser Umstand wird in Abschnitt 5.1.3 relevant, wo die explizite Bildung der Terme **J** und **c** besprochen wird.

Mit dem Penalty-Verfahren, der Reduktionsmethode und der Lagrange-Faktoren-Methode sollen nun drei der wichtigsten Verfahren zur Einbindung von Constraints vorgestellt werden. Einige weitere Lösungsansätze werden dann im daran anschließenden Abschnitt besprochen.

3.3 Penalty-Verfahren

Der Penalty-Ansatz stellt die einfachste Möglichkeit dar, Constraints in die Bewegungsgleichungen zu integrieren. Bei holonomen Constraints wird dem System dazu eine künstliche Kraft in der Form

$$\mathbf{F}^{Penalty} := -\alpha \mathbf{J}^T (\Omega^2 \mathbf{C} + 2\Omega\mu\dot{\mathbf{C}} + \ddot{\mathbf{C}})$$

hinzugefügt, wobei $\mathbf{J}$ die Jacobi-Matrix von $\mathbf{C}$ und α, Ω und μ konstante Parameter darstellen (siehe z.B. [GB94]). Diese Kraft wird bei einer Constraint-Verletzung ($\mathbf{C} \neq 0$) wirksam und ist dieser entgegengerichtet, so daß im idealen Fall wieder der Zustand $\mathbf{C} = 0$ erreicht wird. Der $\dot{\mathbf{C}}$- und der $\ddot{\mathbf{C}}$-Term werden dabei aus Stabilisierungsgründen eingefügt. Eine solche Kraft entspricht der Wirkung einer gedämpften Feder, wobei α der Federstärke, Ω der Eigenfrequenz und μ der Dämpfungsrate entspricht. Sie stellen in der Regel skalare Größen dar, können aber auch als Diagonalmatrizen zur komponentenweisen Festlegung dieser Parameter definiert werden.

Diese Kraft wird nun als zusätzliche äußere Kraft in die Bewegungsgleichungen eingefügt:

$$\mathbf{M}\dot{\mathbf{v}} = \mathbf{F} - \alpha\, \mathbf{J}^T (\Omega^2 \mathbf{C} + 2\Omega\mu\dot{\mathbf{C}} + \ddot{\mathbf{C}}) .$$

Die Lösung dieser Gleichung führt zu einer *näherungsweisen* Erfüllung der Constraints, nur für den Grenzfall eines unendlich großen Penalty-Terms α wäre das Verfahren exakt. Dieser Sachverhalt hat einen großen Einfluß auf die Verwendbarkeit des Verfahrens für die physikalisch basierte Animation, wie in Abschnitt 4.1.1 genauer erläutert wird.

Ersetzt man in der obigen Gleichung $\ddot{\mathbf{C}}$ durch den äquivalenten Ausdruck $\mathbf{J}\dot{\mathbf{v}} + \mathbf{c}$, erhält man

$$(\mathbf{M} + \alpha \mathbf{J}^T \mathbf{J})\,\dot{\mathbf{v}} = \mathbf{F} - \alpha \mathbf{J}^T (\Omega^2 \mathbf{C} + 2\Omega\mu\dot{\mathbf{C}} + \mathbf{c}) . \tag{3.8}$$

Diese Gleichung kann nun nach $\dot{\mathbf{v}}$ aufgelöst werden, um ein System von n gewöhnlichen Differentialgleichungen zu erhalten. Die Umformung ist wegen der positiven Definitheit von $\mathbf{M}$ selbst bei einer singulären $(\mathbf{J}^T\mathbf{J})$-Matrix problemlos möglich, was einen großen Vorteil des Verfahrens darstellt ([GB94]).

Ein solches System gewöhnlicher Differentialgleichungen resultiert bei fast allen Methoden zur Constraint-Einbindung. In Anhang F.1 werden numerische Verfahren vorgestellt, mit denen diese Systeme auf automatische Weise gelöst werden können. In [ESF98] wird eine Übersicht über die Anwendung dieser Verfahren für die Simulation von Mehrkörpersystemen gegeben.

Auch nichtholonome Geschwindigkeits-Constraints $\mathbf{C}(\mathbf{x}, \mathbf{v}, t) = 0$ können mit dem Penalty-Verfahren eingebunden werden. Dazu definiert man eine Kraft

$$\mathbf{F}^{Penalty} := -\alpha \left(\frac{\partial \mathbf{C}}{\partial \mathbf{v}}\right)^T (\mu\mathbf{C} + \dot{\mathbf{C}})$$

und gelangt in völlig analoger Weise zu

$$(\mathbf{M} + \alpha \left(\frac{\partial \mathbf{C}}{\partial \mathbf{v}}\right)^T \left(\frac{\partial \mathbf{C}}{\partial \mathbf{v}}\right))\,\dot{\mathbf{v}} = \mathbf{F} - \alpha \left(\frac{\partial \mathbf{C}}{\partial \mathbf{v}}\right)^T (\mu\mathbf{C} + \frac{\partial \mathbf{C}}{\partial \mathbf{x}}\,\dot{\mathbf{x}} + \frac{\partial \mathbf{C}}{\partial t}) . \tag{3.9}$$

Im Unterschied zur Einbindung holonomer Constraints wird das Bewegungsverhalten hier nur durch zwei Parameter, nämlich α und μ, beeinflußt. Das Penalty-Verfahren läßt sich dabei auch mit Verfahren zur Lösung anderer nichtholonomer Constraints, z.B. in Form von Ungleichungen, koppeln.

3.4 Die Reduktionsmethode

Die Reduktionsmethode macht sich die in Abschnitt 2.3.2.4 erwähnte Eigenschaft von holonomen Constraints zu Nutze, die Anzahl der Freiheitsgrade des Systems zu reduzieren. Ein punktförmiger Körper im dreidimensionalen Raum hat z.B. drei Freiheitsgrade, die man mit der *x*-, *y*- und *z*-Komponente seines Ortsvektors **r** identifizieren kann. Wird ein solcher Körper dem Constraint unterworfen, sich auf einer zweidimensionalen Ebene zu bewegen, geht ein Freiheitsgrad verloren. Es reicht dann z.B. die Angabe der *x*- und *z*- Komponenten von **r** zur eindeutigen Zustandsbeschreibung, während sich r_y aus der Constraint-Gleichung $r_x \tan\varphi = r_y$ (vergl. Abschnitt 2.3.2.4) ableiten läßt.

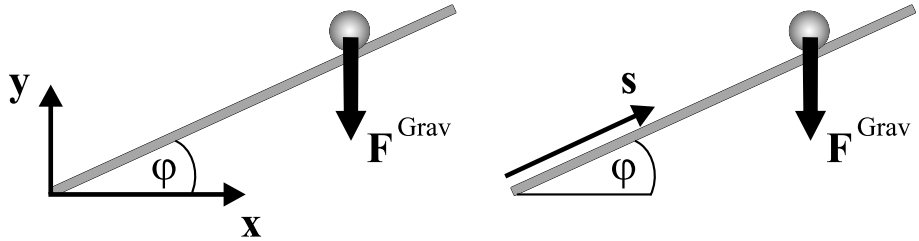

Abbildung 3.1: Kartesische (links) und generalisierte Koordinaten (rechts) bei der schiefen Ebene.

Allgemein werden *n* Freiheitsgrade eines freien Systems durch *m* (skalare) Constraints auf $n-m$ Freiheitsgrade reduziert. Man kann daher ein solches System durch $n-m$ unabhängige generalisierte Koordinaten q_i beschreiben. Der Satz der Zustandsvariablen x_i des freien Systems läßt sich dann als Funktion dieser Größen formulieren:

$$\begin{aligned} x_1 &= x_1(q_1,\ldots,q_{n-m}) \\ &\vdots \\ x_n &= x_n(q_1,\ldots,q_{n-m})\,. \end{aligned} \tag{3.10}$$

Beim Beispiel der schiefen Ebene ließe sich z.B. wie in Abbildung 3.1 gezeigt der Wegparameter *s* als generalisierte Koordinate festlegen. Die Transformationsgleichungen hätten dann die Form

$$\begin{aligned} r_x &= s\,\cos\varphi \\ r_y &= s\,\sin\varphi\,, \end{aligned}$$

wie man leicht geometrisch ableiten kann.

Die Reduktionsmethode besteht nun bei einem System mit *n* Freiheitsgraden und *m* Constraints aus den folgenden Schritten:

- Auswahl von $n-m$ unabhängigen generalisierten Koordinaten q_i.
- Ersetzung der Zustandsvariablen durch die generalisierten Koordinaten gemäß Gleichung (3.10). Dieser Schritt entspricht einer expliziten Parametrisierung des Systems.
- Aufstellung von $n-m$ Bewegungsgleichungen bezüglich der q_i. Dieser Schritt kann z.B. mit Hilfe des sogenannten *Lagrange-Formalismus* durchgeführt werden, der ein schematisch anwendbares und weit verbreitetes Verfahren darstellt (siehe z.B. [Gol89]).

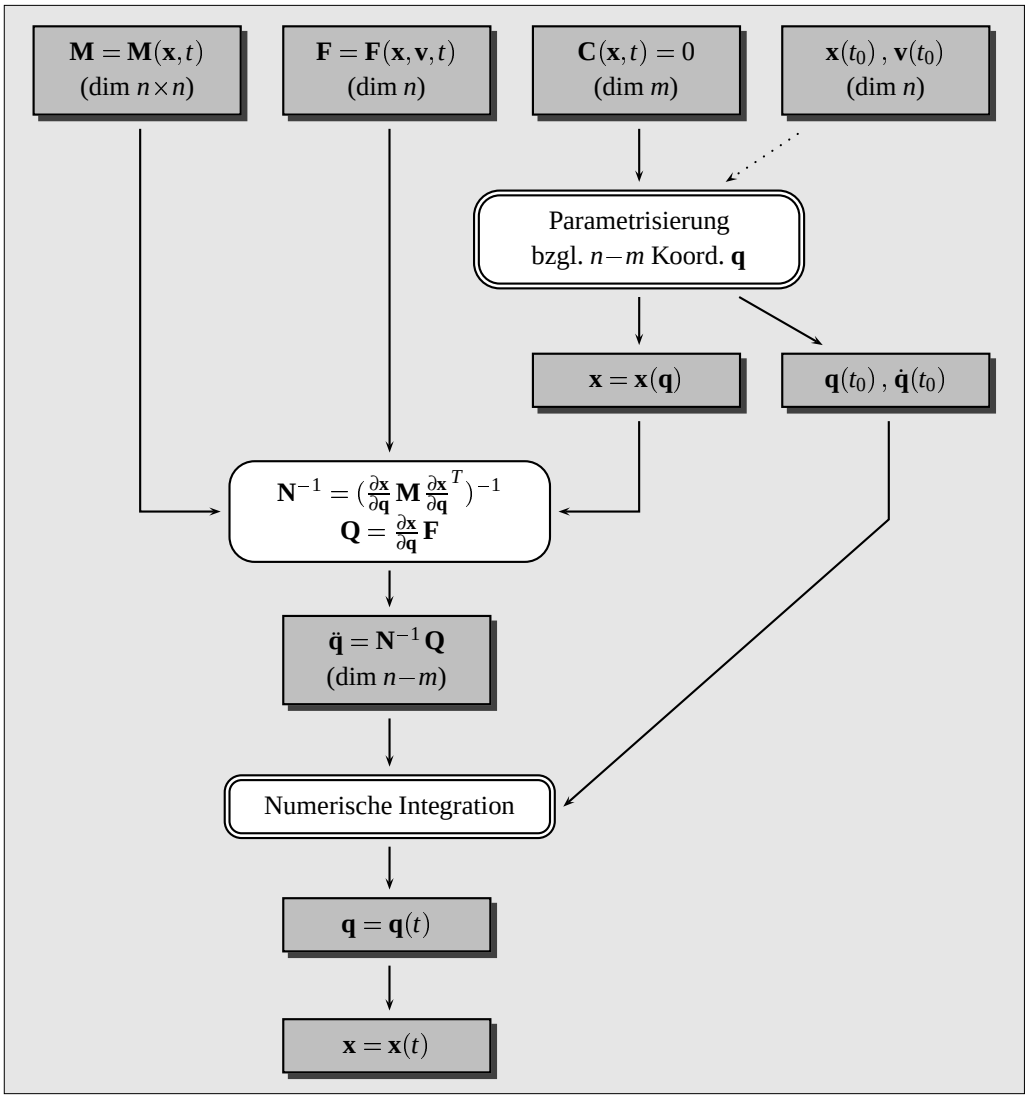

Abbildung 3.2: Ablaufschema der Reduktionsmethode.

- Standardmäßige Lösung des resultierenden $(n-m)$-dimensionalen Systems gewöhnlicher Differentialgleichungen, z.B. mit Hilfe numerischer Integrationsverfahren.

Die durch die Constraints verursachten Kräfte tauchen dabei im Gegensatz zu anderen Lösungsverfahren nicht explizit in den Bewegungsgleichungen auf. Dies stellt in vielen Fällen einen Vorteil dar, da normalerweise nur der resultierende Zeitverlauf $\mathbf{x}(t)$ von Interesse ist und explizit auftretende Constraint-Kräfte somit nur eine Verkomplizierung der Bewegungsgleichungen darstellen würden.

In Abbildung 3.2 ist der Ablauf der Reduktionsmethode schematisch dargestellt. Die schwierigsten und zeitaufwendigsten Teilaufgaben sind dabei die numerische Lösung des Differentialgleichungssystems, das in diesem Fall allerdings nur die Dimension $n-m$ hat, und die Parametrisierung bezüglich der unabhängigen Zustandskoordinaten, die auch durch die Anfangswerte der Koordinaten beeinflußt werden kann.

3.5 Die Lagrange-Faktoren-Methode (LFM)

3.5.1 Grundlegender Verfahrensablauf

Physikalisch betrachtet werden Constraints durch das Wirken von Kräften erfüllt. Die LFM beruht nun auf der expliziten Bestimmung dieser Zwangskräfte (Constraint-Kräfte), die anschließend dem System als zusätzliche Kräfte hinzugefügt werden. Als Constraints können dabei alle Beziehungen berücksichtigt werden, die sich in die Form $\mathbf{J}(\mathbf{x},\mathbf{v},t)\cdot\dot{\mathbf{v}}+\mathbf{c}(\mathbf{x},\mathbf{v},t)=0$ bringen lassen, d.h. alle beschleunigungslinearen Constraints.

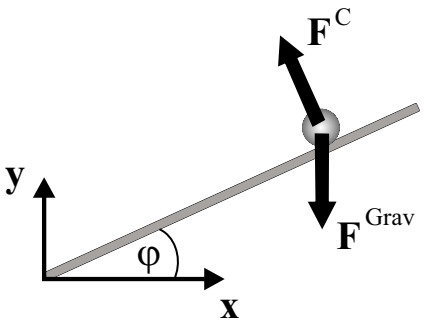

Abbildung 3.3: Äußere Kräfte und Zwangskräfte bei der schiefen Ebene.

Verfahren zur Behandlung anderer nichtholonomer Constraints, wie z.B. Bedingungen an die Zustandsvariablen in Form von Ungleichungen, lassen sich zudem problemlos in Kombination mit der LFM anwenden (vergl. [Pla92]).

Bei dem Beispiel der schiefen Ebene wirkt die durch den Constraint verursachte Zwangskraft z.B. senkrecht zur Ebene und verhindert so das Durchdringen des Körpers. Diesen Sachverhalt, der recht gut den tatsächlichen physikalischen Gegebenheiten entspricht, zeigt Abbildung 3.3.

Die zunächst noch unbekannten Constraint-Kräfte $\mathbf{F}^C$ lassen sich dabei mit Hilfe der sogenannten *Lagrange-Faktoren* ausdrücken. Ein physikalisches Prinzip sagt nämlich aus, daß derartige Zwangskräfte keine Arbeit leisten können ([Gol89]). Die mathematische Entsprechung dieser Forderung ist die Beziehung

$$\mathbf{F}^C=\mathbf{J}^T\lambda\,,$$

wobei λ einen (zunächst noch unbestimmten) Vektor darstellt, dessen Komponenten als *Lagrange-Faktoren* bezeichnet werden. In einem *n*-dimensionales System mit *m* Constraints läßt sich die Bestimmung des *n*-dimensionalen Kraftvektors $\mathbf{F}^C$ somit auf die Bestimmung des *m*-dimensionalen Lagrange-Faktoren-Vektors λ zurückführen.

Diese Constraint-Kräfte können wie äußere Kräfte behandelt werden, so daß sie sich zu den anderen wirkenden Kräften hinzuaddieren lassen. Die Bewegungs- und Constraint-Gleichungen bilden daher das Gleichungssystem

$$\begin{aligned}\mathbf{M}\dot{\mathbf{v}}-\mathbf{F}-\mathbf{J}^T\lambda &= 0\\ \mathbf{J}\dot{\mathbf{v}}+\mathbf{c} &= 0\,.\end{aligned}\tag{3.11}$$

Dies ist ein differentiellalgebraisches Gleichungssystem, das bei *n* Zustandsvariablen und *m* Constraints aus $n+m$ Gleichungen besteht. Um es zu lösen, geht man bei der LFM in zwei Schritten vor:

- Die erste Teilaufgabe besteht in der Bestimmung der Lagrange-Faktoren λ. Diese Faktoren und die daraus resultierenden Zwangskräfte werden bei der LFM also im Unterschied zu anderen Lösungsverfahren explizit berechnet.

  Das obige Gleichungssystem liefert dazu zunächst durch eine einfache Umformung die Gleichung

  $$(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)\lambda=-(\mathbf{J}\mathbf{M}^{-1}\mathbf{F}+\mathbf{c})\,.\tag{3.12}$$

Mit Hilfe dieses linearen Gleichungssystems läßt sich λ bestimmen, wenn die Constraint-Gleichungen in Form von $\mathbf{J}$ und $\mathbf{c}$ sowie die Massenmatrix $\mathbf{M}$ und die Kräfte $\mathbf{F}$ bekannt sind.

- Als zweiter Schritt kann nun λ in die Bewegungsgleichungen eingesetzt werden, so daß man die Gleichung

$$\dot{\mathbf{v}} = \mathbf{M}^{-1}(\mathbf{F} + \mathbf{J}^T \lambda) \tag{3.13}$$

erhält, in der die rechte Seite bekannt ist. Dies ist ein *n*-dimensionales System gewöhnlicher Differentialgleichungen, das standardmäßig gelöst werden kann.

In Abbildung 3.4 ist dieser Ablauf noch einmal schematisch dargestellt. Neben der Lösung des *n*-dimensionalen Differentialgleichungssystems muß bei diesem Verfahren ein *m*-dimensionales lineares Gleichungssystem gelöst werden.

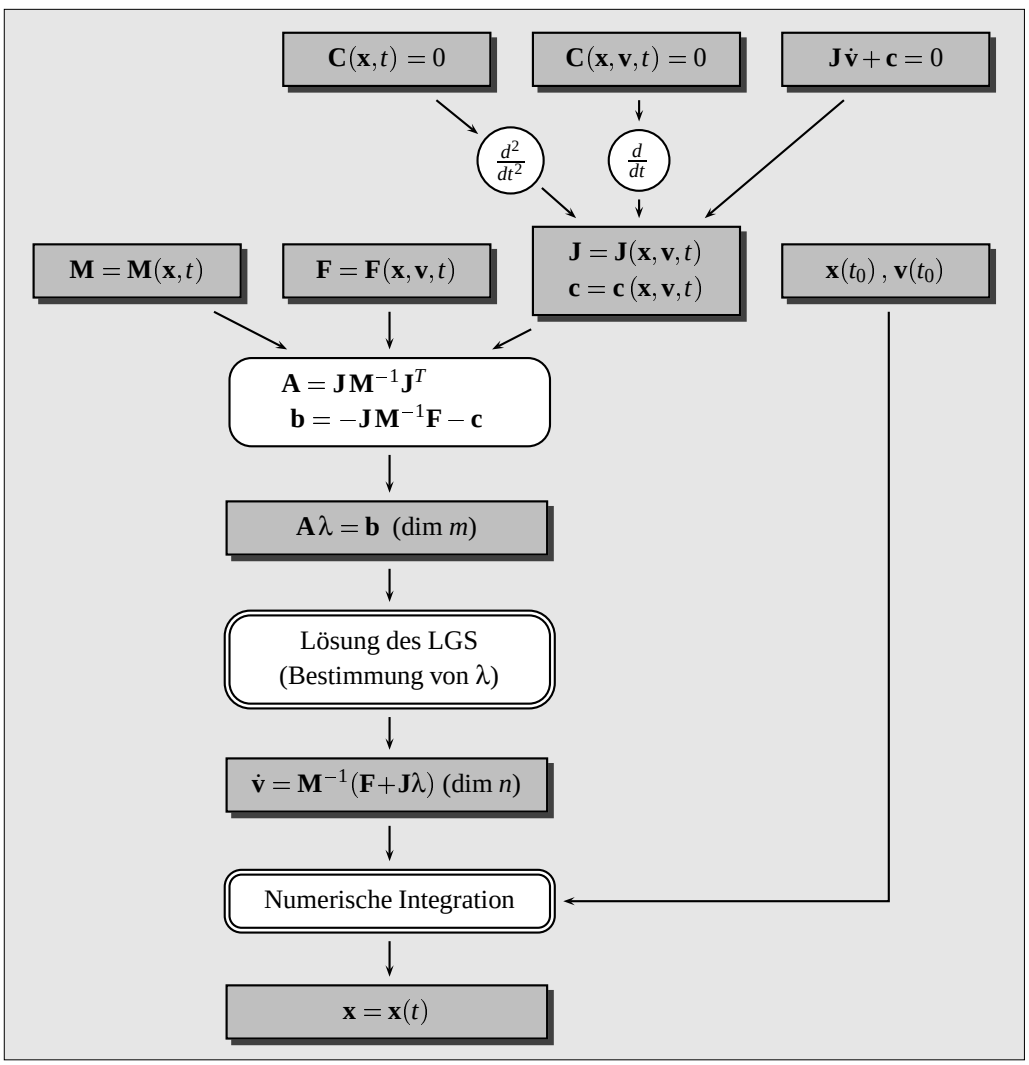

Abbildung 3.4: Ablaufschema der LFM.

3.5.2 Ein Beispiel: Die Schiefe Ebene

Als ein einfaches Beispiel für die Anwendung der LFM soll die in Abbildung 3.3 abgebildete schiefe Ebene betrachtet werden, auf der sich ein punktförmiges Massenteilchen mit den Ortsvektorkomponenten x und y sowie der Masse m unter dem Einfluß des Erdschwerefeldes mit der Gravitationsbeschleunigung g befindet (aus Gründen der Einfachheit im zweidimensionalen Raum). Die Massenmatrix und die externe Kraft sind von dem Ebenen-Constraint unabhängig und ergeben sich als

$$\mathbf{M} = \begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix}, \mathbf{F} = \begin{pmatrix} 0 \\ -mg \end{pmatrix}.$$

Der Constraint, der der schiefen Ebene mit dem Neigungswinkel φ entspricht, hat die Form

$$C(x,y) := x\tan\varphi - y = 0\,.$$

Gemäß den Gleichungen (3.3) und (3.4)[2] resultiert demnach als Jacobi-Matrix

$$\mathbf{J} := \left(\frac{\partial C}{\partial x}, \frac{\partial C}{\partial y}\right) = (\tan\varphi, -1)\,,$$

während sich für den Skalar c der Ausdruck

$$c := \frac{\partial \dot{C}}{\partial x}\dot{x} + \frac{\partial \dot{C}}{\partial y}\dot{y} + \frac{\partial \dot{C}}{\partial t} = 0$$

ergibt, da $\dot{C} = \dot{x}\tan\varphi - \dot{y}$ nur von $\dot{x}$ und $\dot{y}$ abhängt. Die Bestimmung des (in diesem Fall skalaren) Lagrange-Faktors λ kann nun mit Hilfe der Beziehung (3.12) durchgeführt werden. Dazu bestimmt man zunächst

$$\begin{aligned} \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T &= (\tan\varphi, -1)\begin{pmatrix} 1/m & 0 \\ 0 & 1/m \end{pmatrix}\begin{pmatrix} \tan\varphi \\ -1 \end{pmatrix} = \frac{\tan^2\varphi + 1}{m} \\ \mathbf{J}\mathbf{M}^{-1}\mathbf{F} + c &= (\tan\varphi, -1)\begin{pmatrix} 1/m & 0 \\ 0 & 1/m \end{pmatrix}\begin{pmatrix} 0 \\ -mg \end{pmatrix} + 0 = g\,. \end{aligned}$$

Die Auflösung nach λ ist in diesem Fall trivial und liefert das Ergebnis

$$\lambda = -\frac{mg}{\tan^2\varphi + 1} = -mg\cos^2\varphi\,.$$

Der Wert für λ wird nun gemäß Gleichung (3.13) in die Bewegungsgleichung eingesetzt:

$$\begin{aligned} \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} &= \begin{pmatrix} 1/m & 0 \\ 0 & 1/m \end{pmatrix}\left(\begin{pmatrix} 0 \\ -mg \end{pmatrix} + \begin{pmatrix} \tan\varphi \\ -1 \end{pmatrix}(-mg\cos^2\varphi)\right) \\ &= \begin{pmatrix} -g\sin\varphi\cos\varphi \\ -g\sin^2\varphi \end{pmatrix}. \end{aligned}$$

Dieses System aus 2 gewöhnlichen Differentialgleichungen läßt sich hier sogar analytisch lösen, wobei insgesamt 4 Integrationskonstanten $x_0, \dot{x}_0, y_0, \dot{y}_0$ festzulegen sind:

$$\begin{aligned} x(t) &= x_0 + \dot{x}_0 t - \frac{g\sin\varphi\cos\varphi}{2}t^2 \\ y(t) &= y_0 + \dot{y}_0 t - \frac{g\sin^2\varphi}{2}t^2\,. \end{aligned}$$

Dies ist der gesuchte Zeitverlauf für die beiden Zustandswerte x und y.

[2] Diese Gleichungen behalten hier trotz der Anmerkungen am Ende von Abschnitt 3.1 ihre Gültigkeit, da in diesem Beispiel $\dot{\mathbf{x}} = \mathbf{v}$ gilt.

3.5.3 Formale Notation für Mehrkörpersysteme

Für die Beschreibung der Umsetzung der LFM in Abschnitt 3.5.5 ist eine formale Notation notwendig, die auf ein System mehrerer durch Constraints verknüpfter Körper zugeschnitten ist. Eine solche Beschreibungsform, die an der Notation in [Bar96] angelehnt ist, soll nun vorgestellt werden.

Betrachtet werden soll ein System aus k Körpern, denen jeweils ein n_i-dimensionaler Geschwindigkeitsvektor $\mathbf{v}_i$ ($i = 1, \cdots, k$) entspricht, und l beschleunigungslinearen Constraints mit der Dimension m_j ($j = 1, \cdots, l$). Die freien Bewegungsgleichungen für einen einzelnen Körper lauten dann

$$\mathbf{M}_i \dot{\mathbf{v}}_i = \mathbf{F}_i$$

mit der $(n_i \times n_i)$-dimensionalen Massenmatrix $\mathbf{M}_i$ und dem n_i-dimensionalen Kraftvektor $\mathbf{F}_i$. Die j-te Constraint-Gleichung hat zudem die Form

$$\sum_{i=1}^{k} \mathbf{J}_{ji} \dot{\mathbf{v}}_i + \mathbf{c}_j = 0 \,,$$

wobei $\mathbf{J}_{ji}$ die Dimension $(m_j \times n_i)$ und $\mathbf{c}_j$ die Dimension m_j hat. Die auf den i-ten Körper wirkenden Zwangskräfte durch den j-ten Constraints ergeben sich als

$$\mathbf{F}_i^{C_j} = \mathbf{J}_{ji}^T \lambda_j \,.$$

Dabei hat $\mathbf{F}_i^{C_j}$ die Dimension n_i und der Lagrange-Faktoren-Vektor λ_j die Dimension m_j.

Mit Hilfe dieser Beziehungen bezüglich einzelner Körper und Constraints soll nun eine Beschreibung des Gesamtsystems vorgenommen werden. Die Gesamtdimensionen sollen dabei mit $n := \sum_{i=1}^{k} n_i$ und $m := \sum_{j=1}^{l} m_j$ bezeichnet werden. In Matrixschreibweise haben dann die freien Bewegungsgleichungen die Form

$$\mathbf{M}\dot{\mathbf{v}} = \mathbf{F} \tag{3.14}$$

mit der $(n \times n)$-Matrix $\mathbf{M}$ und den n-dimensionalen Vektoren $\mathbf{v}$ und $\mathbf{F}$. Diese Größen sind über die folgenden Zuweisungen mit den oben eingeführten Größen für einzelne Körper verknüpft:

$$\mathbf{M} := \begin{pmatrix} \mathbf{M}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{M}_k \end{pmatrix}, \quad \mathbf{v} := \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_k \end{pmatrix}, \quad \mathbf{F} := \begin{pmatrix} \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_k \end{pmatrix}. \tag{3.15}$$

Die Constraint-Gleichungen lassen sich außerdem als

$$\mathbf{J}\dot{\mathbf{v}} + \mathbf{c} = 0 \tag{3.16}$$

formulieren. Die $(m \times n)$-Matrix $\mathbf{J}$ und der m-dimensionale Vektor $\mathbf{c}$ sind dabei über

$$\mathbf{J} := \begin{pmatrix} \mathbf{J}_{11} & \cdots & \mathbf{J}_{1k} \\ \vdots & & \vdots \\ \mathbf{J}_{l1} & \cdots & \mathbf{J}_{lk} \end{pmatrix}, \quad \mathbf{c} := \begin{pmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_l \end{pmatrix} \tag{3.17}$$

definiert. Die Constraint-Kräfte haben die Form

$$\mathbf{F}^C = \mathbf{J}^T \lambda \,,$$

wobei für den *n*-dimensionalen Vektor $\mathbf{F}^C$ und den *m*-dimensionalen Vektor λ die Beziehungen

$$\mathbf{F}^C := \begin{pmatrix} \sum_{j=1}^{l} \mathbf{F}_1^{C_j} \\ \vdots \\ \sum_{j=1}^{l} \mathbf{F}_k^{C_j} \end{pmatrix}, \quad \lambda := \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_l \end{pmatrix}$$

gelten. Bewegungs- und Constraint-Gleichungen bilden somit das differentiellalgebraische Gleichungssystem

$$\begin{aligned} \mathbf{M}\dot{\mathbf{v}} - \mathbf{F} - \mathbf{J}^T\lambda &= 0 \\ \mathbf{J}\dot{\mathbf{v}} + \mathbf{c} &= 0 \quad . \end{aligned}$$

Dieses System läßt sich in die Form

$$\mathbf{A}\lambda = \mathbf{b} \tag{3.18}$$

mit den beiden Größen

$$\begin{aligned} \mathbf{A} &:= \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T \\ \mathbf{b} &:= -\mathbf{J}\mathbf{M}^{-1}\mathbf{F} - \mathbf{c} \end{aligned} \tag{3.19}$$

bringen, die eine $(m \times m)$-Matrix bzw. einen *m*-dimensionalen Vektor darstellen. Die Lagrange-Faktoren lassen sich dann durch Invertierung von $\mathbf{A}$ bestimmen und gemäß der Beziehung $\dot{\mathbf{v}} = \mathbf{M}^{-1}(\mathbf{F} + \mathbf{J}^T\lambda)$ in die Bewegungsgleichung einsetzen.

3.5.4 Das Stabilisierungsproblem

Die LFM hat mit einem Problem zu kämpfen, das aus der Einbindung holonomer und Geschwindigkeits-Constraints in der Form von Gleichung (3.7) resultiert. Mit Hilfe der LFM wird ein Zeitverlauf $\mathbf{x} = \mathbf{x}(t)$ gefunden, bei dem diese Gleichung erfüllt ist, die bei holonomen Constraints dem Verschwinden der zweiten Ableitung der Constraint-Funktion entspricht. Das Problem besteht nun darin, daß diese Bedingung nicht nur mit der gewünschten Lösung $\mathbf{C} = 0$, sondern allgemein mit der Beziehung

$$\mathbf{C} = \mathbf{k}_1 t + \mathbf{k}_2$$

verträglich ist, bei der $\mathbf{k}_1$ und $\mathbf{k}_2$ beliebige (i.a. vektorielle) Konstanten darstellen. Diese Lösung entspricht einer ständig wachsenden Verletzungen der Constraint-Bedingung! Mathematisch gesehen liegt hier ein Stabilitätsproblem vor, das nicht ignoriert werden darf.

Eine Möglichkeit zur Lösung dieses Problems besteht in der nachträglichen Korrektur der abhängigen Koordinaten nach jedem Integrationsschritt mit Hilfe der Constraint-Gleichungen. Abgesehen von dem zusätzlichen Berechnungsaufwand müßten dafür aber abhängige und unabhängige Koordinaten voneinander getrennt werden, was eine nicht leicht zu automatisierende Aufgabe darstellt. Außerdem wäre dieses Vorgehen mit vielen Integrationsverfahren unverträglich, da zusätzliche, unstetige Zustandsänderungen vorgenommen werden müßten ([Sha98], S. 251).

Einen besseren Lösungsansatz stellt die sogenannte *Baumgart-Stabilisierung* dar (siehe z.B. [RS88], [GB94]). Die Grundidee besteht hierbei in der Verwendung der Gleichung

$$\ddot{\mathbf{C}} + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C} = 0 \tag{3.20}$$

anstelle von $\ddot{\mathbf{C}} = 0$, wobei α und β konstante, skalare Parameter darstellen[3]. Diese Differentialgleichung hat die allgemeine Lösung

$$\mathbf{C} = \mathbf{a}_1 e^{-s_1 t} + \mathbf{a}_2 e^{-s_2 t} \ , \quad s_{1,2} := \alpha \mp \sqrt{\alpha^2 - \beta^2}$$

mit den vektoriellen Integrationskonstanten $\mathbf{a}_1$ und $\mathbf{a}_2$. Diese Funktion nähert sich für $s_{1,2} > 0$ exponentiell schnell an die gewünschte Vorgabe $\mathbf{C} = 0$ an und liefert somit ein stabiles Lösungsverhalten. Für $\alpha = \beta$ folgt $s_1 = s_2 = \alpha$ und es resultiert das kritisch gedämpfte Lösungsverhalten

$$\mathbf{C} = \mathbf{a} e^{-\alpha t} \ ,$$

während sich bei $\alpha < \beta$ imaginäre Werte für s_1 und s_2 ergeben, die zu einem oszillatorischen Verhalten führen.

Diese Stabilisierung läßt sich in den Formalismus der LFM integrieren, indem die allgemeine Constraint-Form (3.7), die bei holonomen Constraints der Beziehung $\ddot{\mathbf{C}} = 0$ entspricht, modifiziert wird:

$$\mathbf{J}\dot{\mathbf{v}} + \mathbf{c}_h = 0 \quad , \quad \mathbf{c}_h := \mathbf{c} + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C} \ . \tag{3.21}$$

Die Ausdrücke für **J** und **c** bleiben dabei erhalten. Mit dieser veränderten Constraint-Gleichung kann die LFM dann wie gewohnt angewandt werden. Den Stabilisierungsparametern α und β müssen dazu allerdings sinnvolle Werte vorgegeben werden, was einen gewissen Nachteil des Verfahrens darstellt.

Ganz analog verfährt man bei Geschwindigkeits-Constraints. Sie können in Form der stabilen Beziehung

$$\dot{\mathbf{C}} + \gamma\mathbf{C} = 0 \tag{3.22}$$

berücksichtigt werden, die die Constraint-Gleichung $\dot{\mathbf{C}} = 0$ ersetzt. Mit γ liegt in diesem Fall lediglich ein Stabilisierungsparameter vor. Die Einbindung in die LFM geschieht über die Gleichung

$$\mathbf{J}\dot{\mathbf{v}} + \mathbf{c}_g = 0 \quad , \quad \mathbf{c}_g := \mathbf{c} + \gamma\mathbf{C} \ . \tag{3.23}$$

Die Stabilisierung führt dabei lediglich zu einer *nahezu* exakten Lösung von holonomen und Geschwindigkeits-Constraints. Wie oben gezeigt wurde, läßt sich aber eine exponentiell schnelle Annäherung an den den exakten Lösungsverlauf erzielen, so daß die hieraus resultierenden Ungenauigkeiten nach einer kurzen Einpegelzeit verschwindend klein sind. In der Computergraphik wurde dieses Stabilisierungsverfahren erstmals im Rahmen des sogenannten *Dynamic Constraint* - Konzeptes ([BB88]) angewandt, das in Abschnitt 4.1.1.3 näher erläutert wird.

Eine etwas andere Herangehensweise an dieses Problem wurde in [W$^+$90], [WW90] gewählt. Die Lagrange-Faktoren werden hier zunächst mit Hilfe der unmodifizierten Constraint-Gleichung $\ddot{\mathbf{C}} = 0$ ermittelt. Bei der Aufstellung des Differentialgleichungssystems (3.13) werden die Stabilisierungsterme von Gleichung (3.20) dann dem System als zusätzliche Kräfte hinzugefügt:

$$\dot{\mathbf{v}} = \mathbf{M}^{-1}(\mathbf{F} + \mathbf{J}^T(\lambda + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C})) \ . \tag{3.24}$$

Grundsätzliche Unterschiede ergeben sich hieraus aber nicht. Auch dieser Ansatz beruht auf der Grundgleichung (3.20) der Baumgart-Stabilisierung, die Korrekturterme gehen hier lediglich in leicht veränderter Form in die Bewegungsgleichungen ein.

In [RE97] wird mit Gleichung (3.20) schließlich eine Modifikation der Lagrange-Funktion im Rahmen des Lagrange-Formalismus (vergl. hierzu [Gol89]) vorgenommen. Auf diese Weise gelangt

[3] Später wurde von Baumgart eine modifizierte, differentiell-integrale Form dieser Gleichung vorgeschlagen, die aber nur auf holonome Constraints anwendbar ist (vergl. [RE97]).

man zu einem $(n+m)$-dimensionalen System gewöhnlicher Differentialgleichungen bezüglich der zweiten Ableitungen der Zustandswerte und der Lagrange-Faktoren λ[4]. Für die Bestimmung der Lagrange-Faktoren muß also kein lineares Gleichungssystem aufgelöst werden. Laut [RE97] liefert dieser Ansatz in bestimmten Fällen, z.B. bei sehr schnell veränderlichen Systemen, bessere Resultate als das ursprüngliche Baumgart-Verfahren.

3.5.5 Der Baraff-Algorithmus

Der Ausgangspunkt zur Bestimmung der Lagrange-Faktoren ist Gleichung (3.18). Die quadratische Matrix $\mathbf{A} := \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ muß invertiert werden, um λ als Funktion der bekannten Größen $\mathbf{J}$, $\mathbf{M}$, $\mathbf{F}^{ext}$ und $\mathbf{c}$ formulieren zu können. Die Dimension von $\mathbf{A}$ ist $(m \times m)$, wobei m gleich der Summe der Dimensionen aller Constraints ist. Die Lösung eines solchen Gleichungssystems hat im allgemeinen Fall die Zeitkomplexität $O(m^3)$. Bei einer großen Anzahl von Constraints kann diese Aufgabe daher einen extremen Zeitaufwand erfordern.

In diesem Abschnitt soll der in [Bar96] vorgestellte Algorithmus erläutert werden, der eine einfache und sehr effiziente Lösung dieses Problems ermöglicht. Er bildet auch die Grundlage für das in Abschnitt 5.3 beschriebene Konzept zur Umsetzung der LFM. Zunächst müssen dazu die mathematischen Eigenschaften der oben angeführten Größen genauer besprochen werden.

Die Matrix $\mathbf{A}$ in Gleichung (3.18) hat oftmals eine bestimmte mathematische Struktur: sie ist schwachbesetzt, d.h. die meisten ihrer Komponenten sind gleich 0. Diese Eigenschaft ergibt sich aus der blockweise diagonalen Struktur von $\mathbf{M}$, die an Gleichung (3.15) abzulesen ist, und der Schwachbesetztheit von $\mathbf{J}$. Die Matrix $\mathbf{J}$ besteht nämlich aus den Jacobi-Matrizen des j-ten Constraints bezüglich des i-ten Körpers (Gleichung (3.16)) und spiegelt demnach die Konnektivität der Körper wieder. Da ein Constraint typischerweise nur auf ein oder zwei Körper wirkt, ergibt sich daher in den meisten Fällen eine sehr schwachbesetzte Struktur. Diese Eigenschaft der Matrix $\mathbf{A}$ kann nun von numerischen Invertierungsverfahren ausgenutzt werden. Einfache kinematische Ketten, die keine Abzweigungen aufweisen, lassen sich auf diese Weise z.B. mit einer linearen Zeitkomplexität lösen.

Komplizierter liegt der Fall, wenn Ketten mit mehreren Zweigen oder andere Systemtopologien vorliegen. Die Matrix $\mathbf{A}$ kann dann sogar vollkommen dicht besetzt sein[5]. Die optimale Behandlung solcher Systeme erfordert komplexe Methoden und stellt ein aktuelles Forschungsgebiet der Numerik dar. In der Computergraphik wurden z.B. wie in [Gle94a] konjugierte Gradientenverfahren angewandt, die typischerweise durch einen quadratischen Zeitaufwand gekennzeichnet sind.

In [Bar96] wurde nun ein einfaches Verfahren vorgestellt, mit dem vor allem Constraints, die auf genau zwei Körper wirken, sehr effizient gelöst werden können. Solche paarweisen Constraints sind für computergraphische Anwendungen sehr wichtig, da zu ihnen z.B. sämtliche Gelenkverbindungen und andere elementare Objektbeziehungen gehören. In [Bar96] wurden nun die folgenden Kernpunkte erarbeitet:

- Die Feststellung, daß alle paarweisen Constraints in einem System ohne geschlossene Schleifen mit linearer Zeitkomplexität gelöst werden können.
- Die Angabe eines einfachen Algorithmus zur Lösung dieser Aufgabe.
- Die Angabe eines erweiterten Algorithmus, mit dem auch Systeme mit geschlossenen Schleifen und Constraints, die sich auf einen oder mehr als zwei Körper beziehen, behandelt werden

[4] Die eindeutige Lösung dieses Systems erfordert somit auch zwei Randwerte für λ und $\dot{\lambda}$.
[5] Ein Beispiel für eine solches System wird in [Bar96] gegeben.

können.

Der erweiterte Algorithmus hat dabei ein Zeitverhalten von $O(p+pk)+O(k^3)$, wobei p die Anzahl der paarweisen und k die der anderen (beliebigen) Constraints ist[6].

Die wesentliche Idee bei dem Baraff-Verfahren besteht in der Aufstellung des Gleichungssystems

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J}^T \\ -\mathbf{J} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ -\mathbf{b} \end{pmatrix} \tag{3.25}$$

mit der Größe $\mathbf{b}$ aus Gleichung (3.18). Dieses System liefert die gleiche Lösung für λ wie Gleichung (3.12) (die obere Reihe liefert $\mathbf{M}\mathbf{y} - \mathbf{J}^\mathbf{T}\lambda = \mathbf{0}$, d.h. $\mathbf{y} = \mathbf{M}^{-1}\mathbf{J}^\mathbf{T}\lambda$, was eingesetzt in die untere Reihe $-\mathbf{J}\mathbf{y} = -\mathbf{b}$ zu $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda = \mathbf{b}$ führt). Da der Vektor $\mathbf{b}$ bekannt ist (Gleichung (3.19)), läßt sich also λ (und die nicht benötigte Variable $\mathbf{y}$) durch Invertierung der Matrix

$$\mathbf{H} := \begin{pmatrix} \mathbf{M} & -\mathbf{J}^T \\ -\mathbf{J} & 0 \end{pmatrix} \tag{3.26}$$

bestimmen. Diese Matrix hat eine größere Dimension als die $(m \times m)$-dimensionale Matrix $\mathbf{A} := \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$, nämlich $(m+n) \times (m+n)$, wobei n die Summe der Dimensionen der Zustandsvariablen der einzelnen Körper ist. Im Gegensatz zu $\mathbf{A}$ ist $\mathbf{H}$ aber bei paarweisen Constraints in einem nichtzyklischen System *immer* schwachbesetzt ([Bar96]). Graphentheoretisch gesehen entspricht $\mathbf{H}$ einem Baum (genauer gesagt, einer Liste von Bäumen). Um diese Eigenschaft ausnutzen zu können, ist i.a. eine Umindizierung von $\mathbf{H}$ erforderlich, die sich aber leicht durchführen läßt.

Diese Eigenschaft der Matrix $\mathbf{H}$ stellt auch den entscheidenden Grund dafür dar, Gleichung (3.25) anstelle von Gleichung (3.18) für die Bestimmung von λ einzusetzen: Derartige schwachbesetzten Gleichungssysteme lassen sich sehr effizient lösen. In [Bar96] wird ein einfacher Algorithmus als Pseudocode angegeben, der diese Aufgabe mit einer linearen Zeitkomplexität bewältigt.

Constraints, die sich nur auf einen oder auf mehr als zwei Körper beziehen oder die zu einem System mit geschlossenen Schleifen führen, sollen (der Begriffsbildung von [Bar96] folgend) als *sekundäre* im Gegensatz zu den paarweisen *primären* Constraints bezeichnet werden. Der erweiterte Algorithmus, mit dem sich beide Typen behandeln lassen, soll an dieser Stelle nur kurz umrissen werden.

Sekundäre Constraints lassen sich mit Hilfe der skalaren Funktionen

$$C_i^{sek} := \mathbf{J}_i^{sek}\dot{\mathbf{v}} + c_i^{sek}, \quad i = 1,\ldots,k$$

ausdrücken. Sie können sich auf eine beliebige Anzahl von Körpern beziehen und nicht nur als beschleunigungslineare Constraints $C_i^{sek} = 0$ vorliegen, sondern auch als Ungleichungen, z.B. in der Form $C_i^{sek} \geq 0$, $C_i^{sek}\lambda_i^{sek} = 0$, die sich für Kontaktkräfte ergibt (siehe Anhang D). Die Constraint-Kräfte haben zudem die Form $\mathbf{k}_i\lambda_i^{sek}$ mit n-dimensionalen Vektoren $\mathbf{k}_i$. Bei beschleunigungslinearen Constraints $C_i^{sek} = 0$ gilt z.B. $\mathbf{k}_i = (\mathbf{J}_i^{sek})^T$.

Die Größen $\mathbf{C}^{sek}$ und λ^{sek} sind über eine $(k \times k)$-Koeffizientenmatrix in linearer Weise miteinander verknüpft. Die Bestimmung von λ^{sek} über diese Beziehung wurde z.B. in [Bar94] behandelt und ist durch eine Zeitkomplexität von $O(k^3)$ gekennzeichnet (vergl. hierzu auch Fußnote 6). Noch aufwendiger ist aber i.a. die *Aufstellung* der Koeffizientenmatrix, da dabei auch die primären Constraints berücksichtigt werden müssen. In [Bar96] wird hierzu ein Verfahren mit einem Aufwand

[6] In [Fau99] wird ein iteratives Verfahren angegeben, das diese Aufgabe mit einer Zeitkomplexität von $O(p+pk)+O(k^2)$ löst und somit eine äußerst interessante Alternative für sehr große Systeme darstellt. Die in [Fau99] angegeben Vergleichszeiten für gelenkig verbundene Systeme legen allerdings den Schluß nahe, daß das Baraff-Verfahren für Systeme mit $k \leq 10$, die im Bereich der Computeranimation sehr häufig eingesetzt werden, mindestens ebenso effizient arbeitet.

von $O(pk)+O(k^2)$ angegeben. Da k in computergraphischen Anwendungen in der Regel viel kleiner als m ist, stellt dieser Vorgang eine weitaus zeitaufwendigere Aufgabe als die Bestimmung der Lagrange-Faktoren λ_i^{sek} dar.

In diesem erweiterten Algorithmus erfolgt als erstes eine Berechnung der zu den sekundären Constraints gehörenden Lagrange-Faktoren λ_i^{sek}. Dabei werden aber die Auswirkungen der primären Constraint-Kräfte, die sich als „Antwort“ auf die λ_i^{sek}-Kräfte ergeben, bereits berücksichtigt. Dann werden die Lagrange-Faktoren λ der primären Constraints unter der Wirkung der äußeren Kräften $\mathbf{F}^{ext}+\sum \mathbf{k}_i\lambda_i^{sek}$ bestimmt. Sie führen nicht zu einer Verletzung der sekundären Constraints, da ihre Wirkung bereits bei der Bestimmung von λ_i^{sek} berücksichtigt wurde. Die insgesamt resultierende Constraint-Kraft ergibt sich dann als $\mathbf{J}^T\lambda+\sum \mathbf{k}_i\lambda_i^{sek}$, die in gewohnter Weise in die Bewegungsgleichungen eingebunden werden kann.

3.6 Andere Lösungsverfahren

Es soll nun ein Überblick über einige weitere Lösungsverfahren zur Einbindung von Constraints gegeben werden, die sich in grundsätzlichen Punkten von den zuvor besprochenen Verfahren unterscheiden.

Direkte numerische Verfahren. Die auch bei der LFM aufgestellten n Bewegungsgleichungen $\mathbf{M}\,\dot{\mathbf{v}} = \mathbf{F}+\mathbf{J}^T\lambda$ bilden zusammen mit den m Constraint-Gleichungen ein $(n+m)$-dimensionales differentiell-algebraisches Gleichungssystem. Die Form der Constraints (und damit die Klasse der zugelassenen Constraints) bestimmt dabei den sogenannten *Index* des Gleichungssystems ([GB94]):

$$\begin{aligned} \mathbf{C}(\mathbf{x},t) &= 0 \qquad \text{(Index 3)} \\ \mathbf{J}(\mathbf{x},t)\,\mathbf{v} &= 0 \qquad \text{(Index 2)} \\ \mathbf{J}(\mathbf{x},\mathbf{v},t)\,\dot{\mathbf{v}}+\mathbf{c}(\mathbf{x},\mathbf{v},t) &= 0 \qquad \text{(Index 1)} \end{aligned}$$

Solche Systeme lassen sich nun auch direkt, d.h. ohne eine vorherige Umformung in ein gewöhnliches Differentialgleichungssystem, mit Hilfe numerischer Verfahren lösen. Auch Geschwindigkeits-Constraints können auf diese Weise gelöst werden und die Kombination mit Verfahren zur Einbindung anderer nichtholonomer Constraints ist ebenfalls möglich. Vor allem für Systeme mit Index 3 sind diese Verfahren aber weniger effizient und stabil als die anderen hier vorgestellten Lösungsansätze (vergl. [Val93] und [GB94], S. 163).

Koordinatenaufteilung. Ein Verfahren, das eine große Ähnlichkeit zur Reduktionsmethode hat, ist die sogenannte *Koordinatenaufteilung* (*Coordinate Partitioning*). Ausgangspunkt dieses Ansatzes, der z.B. in [Sha98] und [GB94] genauer beschrieben wird, sind die Bewegungsgleichungen in nichtreduzierter Form wie Gleichung (3.11) oder auch Gleichung (3.27). Mit Hilfe dieser Gleichungen wird dann eine Aufspaltung des Vektors der generalisierten Koordinaten in einen Satz von $n-m$ unabhängigen Koordinaten $\mathbf{q}_u$ und einen Satz von m abhängigen Koordinaten $\mathbf{q}_v$ vorgenommen. Die Integration geschieht dann nur bezüglich $\mathbf{q}_u$, während die $\mathbf{q}_v$-Werte in einem anschließenden Schritt aus den Constraint-Gleichungen bestimmt werden. Wie bei der Reduktionsmethode ergeben sich daher keine Stabilisierungsprobleme – die Constraints werden exakt und instantan gelöst. Im Unterschied zur dieser kann das Verfahren aber auch auf die Einbindung von Geschwindigkeits- und anderen nichtholonomen Constraints erweitert werden ([RS88], S. 210) und ermöglicht die Aufstellung der Bewegungsgleichungen in beliebigen Koordinaten. Ein schwieriges Problem stellt allerdings die automatische Bestimmung und Auswahl der unabhängigen Koordinaten dar, die bei der Reduktionsmethode schon vor der Aufstellung der Bewegungsgleichungen durchgeführt werden muß.

Obwohl die Koordinatenaufteilung nicht den Effizienzgrad der mit der Baumgart-Stabilisierung ergänzten LFM erreicht ([Val93]), stellt dieser Ansatz ein effizientes und in der Technischen Simulation häufig angewandtes Verfahren dar (vergl. [Sha98], [GB94]).

Projektionsmethode. Die LFM beruht auf der expliziten Bestimmung der Constraint-Kräfte. Wenn die Kenntnis dieser Größen nicht benötigt wird, läßt sich durch die Projektion des Zustandsraums auf den Unterraum der erlaubten Bewegungen eine Umformulierung des Gleichungssystems (3.11) durchführen, bei der die Zwangskraft-Terme eliminiert werden (siehe z.B. [GB94]). Hierzu muß zunächst eine Projektionsmatrix $\mathbf{R}$ bestimmt werden. Bei holonomen Systemen kann diese Matrix z.B. aus der Beziehung $\mathbf{J}\mathbf{R} = 0$ durch Triangulierung der Jacobi-Matrix $\mathbf{J}$ gewonnen werden. Als zweiter Schritt ist dann das Gleichungssystem

$$\begin{pmatrix} \mathbf{J} \\ \mathbf{R}^T\mathbf{M} \end{pmatrix} \dot{\mathbf{v}} = \begin{pmatrix} -\mathbf{c} \\ \mathbf{R}^T\mathbf{F} \end{pmatrix} \tag{3.27}$$

aufzulösen. Dies ist ein System von n gewöhnlichen Differentialgleichungen, das sich standardmäßig integrieren läßt. Das in Abschnitt 3.5.4 erläuterte Problem der Constraint-Stabilisierung muß dabei auch bei diesem Ansatz gelöst werden.

Im Bereich der Technischen Simulation werden Projektionsverfahren sehr oft eingesetzt, um über die obige Gleichung eine Trennung zwischen dynamischem und kinematischem Anteil der Bewegungsgleichungen zu erzielen. Für den Einsatz im Bereich der Computeranimation ergeben sich aus diesem Vorgehen aber keine grundsätzlichen Vorteile gegenüber der LFM. Abgesehen von der fehlenden Bestimmung der Constraint-Kräfte, deren Kenntnis durchaus nützlich sein kann, muß die Projektionsmatrix $\mathbf{R}$ bei jedem Simulationsschritt neu bestimmt werden, was keine triviale Aufgabe darstellt. Derartige Projektionsverfahren werden daher in dieser Arbeit nicht weiter betrachtet.

Gibbs-Appel-Formalismus. Der Gibbs-Appel-Formalismus hat in der Computergraphik eine gewisse Bekanntheit erlangt, da er in den frühen Arbeiten [WB85], [Wil87] zur Animation von Gelenkkörpern angewandt wurde. Er ist aber durch eine Zeitkomplexität von $O(n^4)$ gekennzeichnet und eignet sich daher nicht für eine effektive Constraint-Einbindung ([Wil88]).

Constraint-Entkopplung. Ein völlig anderer Ansatz zur Einbindung von Constraints wurde in [vO90], [vO91] sowie in [GG94] vorgeschlagen. Die wesentliche Idee besteht dabei in der Entkopplung von freier Bewegung und Constraint-Erfüllung. Zunächst werden die dynamischen Körper dazu ohne Constraints auf der Grundlage der freien Bewegungsgleichungen behandelt. Dann erst werden die berechneten Bewegungen gemäß den vorliegenden Constraints korrigiert. Diese Korrektur geschieht dabei durch ein iteratives Verfahren, das lediglich eine Approximation an die physikalisch korrekten Bewegungen liefert. Der Hauptvorteil dieser Herangehensweise liegt in einer äußerst schnellen Constraint-Behandlung sowie in der unproblematischen Einbindung überbestimmter Systeme. Die Korrektur der freien Bewegungen wurde dabei in den angeführten Arbeiten unterschiedlich gelöst.

In [vO90], [vO91] werden den Constraints Reaktionskräfte zugeordnet, die bei ihrer Verletzung wirksam werden und die wie zusätzliche äußere Kräfte behandelt werden. Diese Korrekturkräfte müssen allerdings in geeigneter Weise „per Hand“ für jeden einzelnen Constraint-Typ angepaßt werden. In [GG94] werden die Constraints dagegen direkt durch eine iterative Korrektur der Bewegungen in Form von „Verrückungs-Constraints“ (*Displacement Constraints*) ohne Verwendung der physikalischen Bewegungsgleichungen berücksichtigt (die weiterhin als Grundlage zur Generierung der freien Bewegung dienen). Die Schwierigkeit dieses Verfahrens liegt in der Angabe der Verrückungs-Terme, da diese nicht aus dem physikalischen System resultieren und trotzdem zu na-

türlich wirkenden Bewegungen führen sollen[7].

Als Grundlage von allgemeinen Animationssystemen ist der Ansatz der Constraint-Entkopplung daher kaum geeignet. Vor allem die Anforderungen der allgemeinen Anwendbarkeit und der einfachen Erweiterbarkeit lassen sich bei diesem Verfahren nur schwer erfüllen. Für jeden neu festgelegten Constraint müssen „korrekte“ Korrekturterme bzw. Reaktionskräfte konstruiert werden und selbst mit diesen läßt sich nur ein „plausibles“ Bewegungsverhalten (vergl. Fußnote 7) erzielen. Auf diesen Ansatz wird daher im weiteren Verlauf dieser Arbeit nicht mehr eingegangen. Stattdessen soll nun die Anwendung der wichtigsten Verfahren zur Lösung Constraint-behafteter Systeme im Rahmen einer allgemeinen Animationserstellung betrachtet werden.

[7] Für Point-to-Point-Constraints werden in [GG94] z.B. die Anforderungen genannt, daß sich schwere Objekte weniger als leichter bewegen sollten, daß die relativen Anteile von Rotation und Translation zu der Massenverteilung des jeweiligen Körpers korrespondieren müssen und daß Linear- und Drehimpuls erhalten bleiben.

Kapitel 4

Bewertung der Lösungsverfahren

Mit dem Penalty-Verfahren, der Reduktionsmethode, der LFM und der Koordinatenaufteilung wurden in Kapitel 3 vier Verfahren vorgestellt, die grundsätzlich für die Verwendung in der physikalisch basierten Animation geeignet sind. In diesem Kapitel soll nun untersucht werden, in welcher Form die Verfahren für *allgemeine Animationssysteme* geeignet sind, d.h. es soll eine Bewertung vor dem Hintergrund der in Abschnitt 1.1 aufgeführten Anforderungen an diese Systeme vorgenommen werden. Wie sich herausstellt, lassen sich dabei große Unterschiede zwischen den Ansätzen ausmachen.

Als erstes wird dazu eine Übersicht über die bisherige Anwendung der Verfahren im Bereich der Computeranimation gegeben, bei der auch (in Abschnitt 4.1.2) eine Abgrenzung dieser Arbeit zu bestehenden Arbeiten vorgenommen wird. Anschließend werden die mit diesen Verfahren verknüpften Anwendungsaspekte ausführlich analysiert und mit den Anforderungen an allgemeine Animationssysteme in Beziehung gesetzt.

4.1 Anwendung in der physikalisch basierten Animation

Die hier betrachteten Lösungsverfahren wurden in Kapitel 3 ausführlich vorgestellt. In Tabelle 4.1 sind einige ihrer Eigenschaften aufgeführt, die für ihre Anwendung im Bereich der physikalisch basierten Animation besonders wichtig sind. Der in der Literatur beschriebene Einsatz der Verfahren in diesem Bereich soll nun im einzelnen besprochen werden.

4.1.1 Allgemeine Anwendungen

4.1.1.1 Penalty-Verfahren

Das Penalty-Verfahren eignet sich für computergraphische Anwendungen vor allem durch seine einfache Umsetzbarkeit. Das numerische Problem reduziert sich bei diesem Ansatz auf die Integration eines Systems gewöhnlicher Differentialgleichungen, das standardmäßig gelöst werden kann. Außerdem ist das Penalty-Verfahren sehr allgemein: Nicht nur holonome, sondern auch Geschwindigkeits-Constraints können problemlos eingebunden werden.

Schwierig kann bei dieser Methode die geeignete Wahl der Parameter α, Ω und μ sein, die einen großen Einfluß auf das Lösungsverhalten haben. Sie müssen explizit vorgegeben werden, was zu einer Beeinträchtigung der einfachen Anwendbarkeit führen kann, da sie recht unintuitive Größen darstellen. Dieses Problem ergibt sich in ähnlicher Form auch für die LFM, bei der der Parameter α

	Penalty	LFM	Reduktion	Aufteilung	Direkt
Effizienzgrad	niedrig[a]	hoch	hoch	mittel	niedrig
Holonome Constraints	ja	ja	ja	ja	ja
Geschwindigkeits-Constraints	ja	ja	nein	zusätzlich	ja
Andere nichtholonome Constraints	zusätzlich	zusätzlich	nein	zusätzlich	zusätzlich
Bewegungsgl. in unabhängigen Koord.	nein	nein	ja	nein	nein
Bestimmung von unabhängigen Koord.	nein	nein	ja	ja	nein
Stabilisierung erforderlich	integriert	zusätzlich	nein	nein	integriert
Stabilisierungsparameter	3	2	0	0	0

[a] Bezogen auf einen Penalty-Term, der groß genug ist, um eine mit den anderen Verfahren vergleichbare Exaktheit zu erreichen.

Tabelle 4.1: Einige grundlegende Eigenschaften der Lösungsverfahren. Mit „zusätzlich“ werden dabei Eigenschaften bezeichnet, die nur durch zusätzliche Aufwendungen erzielt werden können.

allerdings kein Äquivalent hat.

Penalty-Verfahren haben aber vor allem mit einem grundsätzlichen Problem zu kämpfen: Sie erfüllen die Constraints nur näherungsweise. Nur für einen unendlich großen Penalty-Term α wären sie exakt. Der Grad der Constraint-Erfüllung ist dabei schwer vorherzusagen oder gar zu steuern, da die Federkräfte mit allen anderen Kräften des Systems interagieren. Um möglichst geringe Abweichungen von den festgelegten Constraints zuzulassen, muß α daher sehr groß gewählt werden. In diesem Fall können aber sogenannte *steife* Differentialgleichungen entstehen (siehe Anhang F.1), die nur mit speziellen numerischen Methoden und mit einem sehr großen Zeitaufwand gelöst werden können.

Aufgrund der einfachen Anwendbarkeit ist das Penalty-Verfahren in [MW88] für die Behandlung von Ruhekontakten eingesetzt worden (vergl. Abschnitt 2.4.3). In [W+87] wurde auf der Grundlage des Penalty-Ansatzes eine Umformulierung von Constraints als „Energiefunktionen“ vorgestellt, mit der die Modellierung und Simulation von parametrischen Modellen erleichtert werden soll. Als Energiefunktion werden dabei nichtnegative Funktionen auf dem Parameterraum des Modells verstanden, die dort Nullstellen haben, wo die zugehörigen Constraints erfüllt sind. Bei mehreren Constraints überlagern sich die Energiefunktionen additiv. Die Suche nach Parameterwerten, die die Constraints erfüllen, ist daher mit einer Minimierung dieser Energie gleichbedeutend und kann in intuitiver Weise vom Benutzer unterstützt werden. Als allgemeine Simulationsgrundlage für Animationssysteme sind Penalty-Verfahren aber aufgrund der erwähnten Probleme kaum geeignet. Zu diesem Ergebnis kommt auch die in [Pla92] vorgenommene Gegenüberstellung des Penalty-Ansatzes mit der Reduktionsmethode und der LFM.

Ein anderes Bild würde sich ergeben, wenn Weiterentwicklungen dieses Verfahrens auch für allgemeine Anwendungen den Grad an Stabilität und Effizienz erreichen, durch die z.B. die LFM und die Reduktionsmethode ausgezeichnet sind, wie in [GB94], S. 165, in Aussicht gestellt wird. Für die Analyse der Anwendungsaspekte bezüglich allgemeiner Animationssysteme darf dieses Verfahren daher nicht unberücksichtigt bleiben.

4.1.1.2 Reduktionsmethode

Für die Bestimmung der unabhängigen Koordinaten muß bei der Reduktionsmethode ein nichtlineares Gleichungssystem aufgelöst werden. Durch Singularitäten, die in der Regel erst während der Simulation auftreten, kann ein Satz unabhängiger Koordinaten zudem inadäquat werden, so daß er durch einen anderen Satz ersetzt werden muß. Die Automatisierung dieser Prozesse, die für Anwendungen im Bereich der Computeranimation unverzichtbar ist, stellt eine sehr schwierige Aufgabe dar. In [F+99] werden für diese Auswahl z.B. die Anforderungen genannt, daß numerisch günstige Gleichungsterme erzielt werden müssen und daß die Koordinaten zur effektiven Steuerung des Systems zu den unabhängigen Koordinaten zählen sollten, wofür ein expliziter Eingriff des Benutzers erfordert wird. Beachtet werden muß auch, daß zur Bestimmung von absoluten Objektkoordinaten, die z.B. für die graphische Darstellung erforderlich sein können, ein zusätzlicher Aufwand nötig ist.

Die Reduktionsmethode ist aber mathematisch gesehen sehr elegant und stellt trotz dieser Probleme ein sehr effizientes Verfahren dar. Insbesondere müssen statt der ursprünglichen n Differentialgleichungen des freien Systems nur $n-m$ Gleichungen gelöst werden. Schon 1990 wurde diese Methode in [SZ90] für die Animation Constraint-behafteter Systeme beliebiger Topologie mit einer linearen Zeitkomplexität angewandt. Die Reduktionsmethode hat daher gerade im Bereich der Computeranimation eine große Verbreitung gefunden.

In [AG85], [A+87] ist diese Methode erstmals für die Animation von Gelenkkörpern angewandt worden. Dabei wurde eine rekursive Beschreibung gewählt, die auf Kugelgelenke zugeschnitten ist und keine Einbindung von Gleitgelenken und allgemeineren Constraint-Typen erlaubt [1]. Das Verfahren zeichnet sich aber (im Gegensatz z.B. zu den in [WB85] und [Wil87] angewandten Verfahren) durch eine lineare Zeitkomplexität aus und ist daher in [FW88] zur Simulation von rein kinematisch gesteuerten Gelenkkörpern als Hilfsmittel für die interaktive Manipulation dieser Objekte eingesetzt worden. In [G+91] fand die Reduktionsmethode für die Animation deformierbarer Körper Anwendung. In [IC87] wurden Constraints als allgemeine Steuertechnik für hierarchische Systeme eingeführt, die auf der Grundlage der Reduktionsmethode mit einer kubischen Zeitkomplexität gelöst wurden. Eine auf dieser Methode basierende Simulation des menschlichen Ganges ist in [BC89] durchgeführt worden. Mit der sogenannten *Articulated Body Method* ([Fea87]) wurde dann eine rekursive Umsetzung für hierarchische Systeme beschrieben, die die Einbindung beliebiger holonomer Constraints mit einer linearen Zeitkomplexität ermöglicht. Anwendung für die Gelenkkörper-Animation fand sie u.a. in [MZ90] und [VC91]. Eine Erweiterung dieses Verfahrens auf nichthierarchische Systeme wurde in [SZ90] umgesetzt und für die Animation einiger einfacher mechanischer Systeme angewandt. In [HG95], [HG97] wurde ein hierauf basierendes interaktives Animationssystem vorgestellt, mit dem sich starre und deformierbare Körper animieren lassen.

[1] Scharniergelenke werden in dieser Arbeit durch Kugelgelenke mit zusätzlichen Drehmomenten zur Sperrung der entsprechenden Freiheitsgrade modelliert.

4.1.1.3 LFM

Im Vergleich zu der Reduktionsmethode hat das Differentialgleichungssystem bei der LFM die Dimension n statt $n-m$. Dies muß aber keinen Effizienzverlust bedeuten, da dieses Gleichungssystem i.a. günstigere numerische Eigenschaften aufweist. Tatsächlich ist in der Regel die Bestimmung der Lagrange-Faktoren als Lösung des linearen Gleichungssystems (3.12) der zeitaufwendigste Vorgang.

Die LFM stellt ein allgemeines und einfach anwendbares Verfahren dar und ist daher sehr häufig in der Computeranimation eingesetzt worden. In [A$^+$89a], [A$^+$89b] diente diese Methode z.B. als Grundlage für die Animation von Gelenkkörpern und anderen mechanischen Systemen. Die Einführung von Constraints als allgemeine Steuertechnik in [IC87] wurde in [IC88] mit Hilfe der LFM auf Systeme mit geschlossenen Schleifen erweitert[2]. Als sehr nützlich hat sich diese Methode auch für die Animation von deformierbaren Körpern ([PB88], [WW90]) und für kombinierte Systeme ([Met95], [C$^+$95b]) erwiesen, die sowohl starre als auch deformierbare Körper enthalten.

Eine besonders interessante Anwendung der LFM stellt das in [BB88] eingeführte *Dynamic Constraints*-Konzept dar. Da Constraints durch das Wirken von Kräften gelöst werden, geschieht der Prozess der Constraint-Erfüllung bei dieser Methode in einer stetigen, physikalisch basierten Weise. In [BB88] wurde für eine Reihe einfacher Constraints gezeigt, wie sich diese spezielle Art der Constraint-Erfüllung als Unterstützung für die physikalisch basierte Modellierung einsetzen läßt. In [W$^+$90] und [Pla92] ist dieser Ansatz auf die Einbindung beliebiger Constraints verallgemeinert worden.

In [Gle94a] bildet die LFM die Grundlage eines allgemeinen, auf differentiellen Manipulationen beruhenden Interaktionskonzeptes, das sich auf die pseudophysikalische Bewegungsgleichung $\mathbf{F} = m\mathbf{v}$ stützt. Die Anwendung dieses Konzeptes für ein Zeichenprogramm wurde in [Gle93] beschrieben. In [H$^+$95] ist die LFM zudem für eine kombinierte Behandlung von diskreten und kontinuierlichen Systemen angewandt worden und in [BW97] wurde eine Kapselung und effiziente Kombination von Simulationsverfahren für unterschiedliche Anwendungsbereiche auf der Grundlage der LFM vorgestellt.

Die effiziente Anwendung dieser Methode galt allerdings lange als sehr kompliziert. Häufig wurden Umsetzungen mit einem kubischen (wie in [IC88]) oder quadratischen Komplexitätsverhalten (wie in [Gle94a]) angewandt. Einen großen Fortschritt für den Einsatz der LFM stellt daher die einfache und zugleich numerisch effiziente Umsetzung von David Baraff dar ([Bar96]), deren grundlegender Ablauf in Abschnitt 3.5.5 vorgestellt wurde. Constraints, die Paare von Körpern beeinflussen, können mit diesem Verfahren mit einer linearen Zeitkomplexität gelöst werden.

4.1.1.4 Direkte numerische Lösung

Die direkte numerische Lösung der Constraint-behafteten Bewegungsgleichungen ist weniger effizient und robust als die anderen hier besprochenen Verfahren (vergl. Abschnitt 3.6). Dieser Ansatz stellt aber eine grundsätzliche Möglichkeit dar, die von Weiterentwicklungen im Bereich der Numerik profitieren kann und daher auch als Simulationsgrundlage eines allgemeinen Animationssystems in Frage kommt. Im Bereich der physikalisch basierten Animation wurde die Anwendung dieses Verfahrens bislang nicht beschrieben.

[2] Diese Arbeiten basieren dabei auf dem in [Wit77] vorgestellten Formalismus zur Beschreibung von Mehrkörpersystemen im Rahmen der Technischen Simulation.

4.1.1.5 Koordinatenaufteilung

Ein Ansatz, für dessen Anwendung in der Computeranimation ebenfalls keine Beispiele aufzufinden sind, ist das Verfahren der Koordinatenaufteilung. Dieses Verfahren stellt eine interessante Abwandlung der Reduktionsmethode dar. Die mögliche Einbindung nichtholonomer Constraints und die Verwendung beliebiger Koordinaten zur Formulierung der Bewegungsgleichungen, die sich auf diese Weise wesentlich einfacher aufstellen lassen, sind gerade vor dem Hintergrund allgemeiner Animationssysteme zwei wichtige Vorteile. Das Problem der Bestimmung und Auswahl von unabhängigen Koordinaten stellt sich allerdings auch für diesen Ansatz.

4.1.2 Anwendungen für allgemeine Animationssysteme

Die im vorigen Abschnitt angeführten Arbeiten haben in den allermeisten Fällen eine sehr spezielle Zielrichtung. Häufig wird z.B. die Realisierung bestimmter Bewegungsabläufe für geeignet modellierte Gelenkkörper untersucht oder es werden ganz spezifische Steuertechniken eingeführt. Die Frage, wie sich verschiedene Körpertypen und verschiedene Steuertechniken unter Einbeziehung der Besonderheiten des jeweiligen Lösungsverfahrens auf einheitliche Weise einsetzen und in ein Animationssystem integrieren lassen, wird dagegen kaum betrachtet[3].

Die hierzu in der Literatur erschienenen Beiträge, die zumeist aus den Arbeiten stammen, auf die im vorigen Abschnitt vor dem Hintergrund der Verbreitung der Lösungsverfahren verwiesen wurde, sollen nun kurz beschrieben und anschließend mit der Zielrichtung dieser Arbeit verglichen werden.

Eines der ersten Animationssysteme, das eine einfache Steuerung von Gelenkkörpern über eine graphische Benutzungsoberfläche erlaubt, wurde in [W+88] vorgestellt. In [A+89b] wurde ein Animationssystem auf der Grundlage der LFM beschrieben, das einen einheitlichen Zugang für die Modellierung und Bewegungssteuerung bietet und das sich über menü-, text- und skriptbasierte Befehle bedienen läßt. Die Constraint-behafteten Systeme werden hierzu mit Hilfe eines Graphen beschrieben und durch symbolische Verfahren ausgewertet. Auch in [B+92] wurde eine graphentheoretische Beschreibung von dynamischen Systemen eingeführt, die die einfache Festlegung von Constraints und die automatische Aufstellung der Bewegungsgleichungen erlauben soll. Ein allgemeines und einfach erweiterbares System zur Anwendung von Constraints auf der Grundlage der Reduktionsmethode, mit dem auch Systeme mit geschlossenen Schleifen effizient gelöst werden können, wurde in [SZ90] beschrieben. In [VC91] wurde eine ähnliche Anwendung der Reduktionsmethode vorgestellt, wobei zusätzlich Techniken zur Inversen Dynamik und zur einfachen Anwendung der Constraints realisiert wurden. Ein interaktives System zur Animation starrer und flexibler Körper auf der Grundlage von [SZ90] wurde in [HG95], [HG97] vorgestellt. Die interaktive Festlegung von Constraints, die an einem beliebigen Zeitpunkt und auch in Kombination mit einer Kollisionssimulation wirken können, wurde in [P+00] beschrieben.

Die kombinierte Anwendung mehrerer Steuertechniken wurde in [IC87] beschrieben und in [IC88] mit Hilfe der LFM auf Systeme mit geschlossenen Schleifen erweitert. Auch in [G+91] wurde ein System vorgestellt, das verschiedene Steuermöglichkeiten bereitstellt, allerdings für spezielle Gelenkkörper, die mit einer elastischen „Haut“ überzogen sind. Die Anwendung allgemeiner Prozeduren für ein Animationssystem wurde in [SC92a] dargelegt. In [DC95] wurden sowohl deskriptive als auch generative und verhaltensbasierte Steuertechniken in ein Simulations- und Animationssystem integriert. Die einheitliche Behandlung von starren und deformierbaren Körpern mit Hilfe eines Entity-Relationship-Modells wurde in [C+95b] beschrieben. In [Met95] wurde ein Formalismus auf

[3] Eine hierzu genau komplementäre Fragestellung wird in [Wen98] behandelt. Dort wird der Einsatz von Animationssystemen als Visualisierungswerkzeug zur Unterstützung Technischer Simulationen untersucht, wofür ein Metamodell zur Schaffung einer geeigneten Schnittstelle entworfen wird.

der Grundlage der LFM vorgestellt, der ebenfalls die einheitliche Anwendung auf diese beiden Körpertypen erlaubt. In [BW97] wurde ein Verfahren auf der Grundlage der LFM für die kombinierte Simulation und Steuerung verschiedener Systemtypen ausgenutzt.

Die Besonderheiten der jeweiligen Lösungsverfahren wurden bei diesen Arbeiten kaum oder überhaupt nicht berücksichtigt. In [BB88] wurde dagegen die Eigenschaft der LFM, Constraints durch einen stetigen, physikalisch basierten Prozess zu lösen, explizit für die intuitive Festlegung einer Reihe von Constraints ausgenutzt. Eine Erweiterung auf die Einbindung beliebiger Constraints wurde in den Arbeiten [W+90] und [Pla92] vorgestellt. Diese Arbeiten zeigen außerdem zusätzliche Möglichkeiten für den Einsatz der LFM auf: In [W+90] wird die voneinander unabhängige Formulierung von Körpern und Constraints und die Animation deformierbarer Körper beschrieben. Die hierauf basierende Steuerung deformierbarer Körper durch vorgegebene Trajektorien und skriptbasierte Techniken wurde in [WW90] genauer ausgeführt.

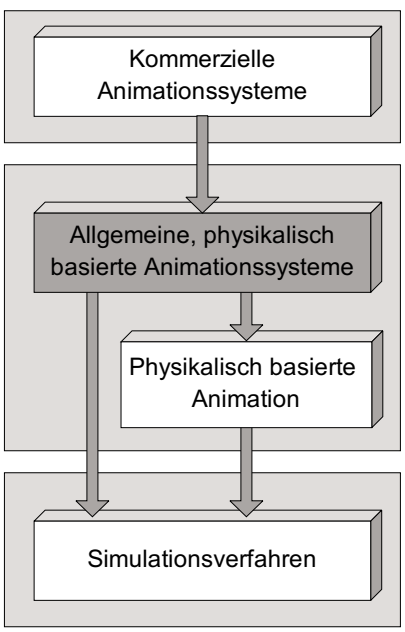

Abbildung 4.1: Die unzureichend untersuchte Zwischenebene der allgemeinen, physikalisch basierten Animationssysteme.

In [Pla92] wird die Eignung der Penalty-Methode, der Reduktionsmethode und der LFM als Grundlage eines Animationssystems verglichen und an einem Animationsbeispiel illustriert. Das in Abschnitt 3.5.5 erläuterte Verfahren zur einfachen und effizienten Umsetzung der LFM wurde in [Bar96] vorgestellt. In dieser Arbeit finden sich auch einige Anmerkungen über die grundsätzlichen Vorteile der LFM für die Animationserstellung wie z.B. die Möglichkeit zur Kapselung von Körpern und Constraints. Für die Animation von Kleidung werden in [BW98] schließlich vergleichende Betrachtungen für die Eignung der verschiedenen Lösungsverfahren für einen speziellen Typ deformierbarer Körper angestellt[4].

Die Realisierung eines vollständigen Animationssystems, das eine intuitive Bedienung aufweist und dem Animateur eine Reihe unterschiedlicher Modellierungs- und Steuertechniken zur Verfügung stellt, wird in diesen Arbeiten nicht behandelt. Die Arbeiten [W+88], [A+89b], [VC91], [W+90], [HG95], [HG97] kommen diesem Ziel noch am nähesten, auch sie enthalten aber z.B. keine Ausführungen über Probleme, die aus der interaktiven Bedienung des Animationssystems resultieren, wie z.B. die Behandlung von fehlerhaften Eingaben oder die Entwicklung intuitiver Interaktionstechniken. Derartige Techniken sind in kommerziellen Animationssystemen verwirklicht, die auch zunehmend physikalisch basierte Techniken zur Animationserstellung anbieten. Die über diese Systeme vorliegende Literatur beschreibt aber lediglich, durch welche Funktionalitäten die Systeme ausgezeichnet sind und wie sich sich anwenden lassen, nicht aber, welche Aspekte bei ihrer *Entwicklung* aufgetreten sind und wie ihre Umsetzung gelöst wurde.

Nur bei wenigen der oben angeführten Arbeiten werden zudem die Besonderheiten der jeweilig

[4] Dabei wird ein eigens entwickeltes Verfahren auf der Grundlage einer Modifikation der Masseneigenschaften eingeführt, das aber auf diesen speziellen Modelltyp zugeschnitten ist und sich nicht für die Simulation beliebiger mechanischer Systeme eignet.

angewandten Lösungsverfahren betrachtet und ausgenutzt. Lediglich in [BB88], [W+90], [Pla92], [Bar96] und [BW98] wird explizit auf derartige Eigenschaften verwiesen, wobei nur in den drei letztgenannten Arbeiten vergleichende Betrachtungen zu anderen Lösungsverfahren angestellt werden. Die systematische Untersuchung der Eignung der Lösungsverfahren für die Realisierung eines allgemeinen Animationssystems, das den in der Einführung dieser Arbeit genannten Anforderungen gerecht wird, ist dagegen noch nicht durchgeführt worden. Eine solche Analyse wird in dieser Arbeit vorgestellt. Anhand der Entwicklung eines allgemeinen Animationssystems auf der Grundlage der LFM wird diese Analyse dabei durch die Berücksichtigung konkreter Realisierungsaspekte unterstützt.

Für die Abgrenzung dieser Arbeit ergibt sich daher das in Abbildung 4.1 skizzierte Bild. Die Realisierung und Weiterentwicklung von Verfahren zur Lösung Constraint-behafteter dynamischer Systeme stellt ein aktuelles Forschungsgebiet der Technischen Simulation dar. Auch das Gebiet der physikalisch basierten Animation, das sich diese Verfahren zunutze macht, hat mittlerweile einen großen Niederschlag in der Literatur gefunden. Eine wichtige Zwischenebene zur Realisierung kommerzieller Animationssysteme, die wie gesehen bislang nur sehr unzureichend untersucht wurde, stellt aber die Entwicklung allgemeiner, physikalisch basierter Animationssysteme unter Berücksichtigung der Eigenschaften des jeweiligen Lösungsverfahrens dar. Die vorliegende Arbeit stellt mit der systematischen Bewertung der Verfahren zur Constraint-Einbindung und dem Entwurf von Konzepten und Methoden zur Realisierung eines allgemeinen Animationssystems auf der Grundlage der LFM einen neuen Beitrag für dieses wichtige Themenfeld dar.

Einige Ergebnisse dieser Arbeit wurden dabei bereits in [Wag99] und [Wag00] veröffentlicht. In [Wag99] wurde die grundsätzliche Eignung verschiedener Lösungsverfahren zur Constraint-Behandlung diskutiert und das Animationssystem EMPHAS vorgestellt, das in Kapitel 6 genauer beschrieben wird. In [Wag00] wurde eine Bewertung der LFM vor dem Hintergrund allgemeiner Animationssysteme vorgenommen und auf dieser Methode basierende Techniken vorgestellt, die im Rahmen des Animationssystems EMPHAS entwickelt worden sind.

4.2 Anwendungsaspekte für allgemeine Animationssysteme

Aus den Eigenschaften der hier betrachteten Lösungsverfahren ergeben sich interessante Aspekte für die Realisierung eines allgemeinen Animationssystems. Diese Anwendungsaspekte sollen nun aufgeführt und erläutert werden. Um einen einfacheren Überblick zu gewährleisten, werden diese Aspekte dabei so formuliert, daß sie eine *vorteilhafte* Eigenschaft des Lösungsverfahrens repräsentieren. In Tabelle 4.2 wird eine Übersicht über die Realisierbarkeit dieser Aspekte und ihre Relevanz bezüglich allgemeiner Animationssysteme, die nur als grobe Richtschnur gesehen werden kann, gegeben. Im Abschnitt 4.3 werden diese Aspekte dann den Anforderungen an allgemeine Animationssysteme zugeordnet.

1. Effizienz und Robustheit
   Die zeitliche Effizienz und die Robustheit des Lösungsverfahrens spielen nicht nur im Bereich der Technischen Simulation, sondern auch für Anwendungen im Bereich der Computeranimation eine wichtige Rolle, wie schon in Abschnitt 2.2.3 bemerkt wurde. Diese beiden Aspekte sind auch für allgemeine Animationssysteme sehr wichtig:

   (a) Effizienz
   Die Interaktivität stellt eine wesentliche Anforderung an allgemeine Animationssysteme dar und kann nur durch ein hinreichend effizientes Simulationsverfahren erreicht

	Penalty	LFM	Reduktion	Aufteilung	Direkt	Relevanz
1a	niedrig	hoch	hoch	mittel	niedrig	hoch
1b	mittel	mittel	mittel	mittel	niedrig	hoch
2a	ja	ja	nein	zusätzlich	ja	hoch
2b	zusätzlich	zusätzlich	nein	zusätzlich	zusätzlich	hoch
3a	(ja)[a]	ja	zusätzlich	zusätzlich	zusätzlich	mittel
3b	(ja)[a]	ja	zusätzlich	zusätzlich	zusätzlich	niedrig
4a	ja	ja	zusätzlich	zusätzlich	zusätzlich	mittel
4b	(ja)[b]	(ja)[b]	zusätzlich	zusätzlich	(ja)[b]	niedrig
4c	zusätzlich	zusätzlich	ja	ja	ja	mittel
5a	ja	ja	(nein)[c]	(nein)[c]	(nein)[c]	hoch
5b	ja	ja	(nein)[c]	(nein)[c]	(nein)[c]	hoch
5c	ja	ja	(nein)[c]	(nein)[c]	(nein)[c]	mittel
5d	(ja)[d]	(ja)[d]	ja	ja	ja	niedrig
6a	ja	ja	nein	ja	ja	mittel
6b	ja	ja	nein	ja	ja	hoch
6c	(nein)[e]	(nein)[e]	ja	(nein)[e]	(nein)[e]	mittel
7a	ja	ja	nein	nein	ja	mittel
7b	ja	ja	zusätzlich	ja	ja	hoch
7c	nein	nein	ja	ja	ja	mittel

[a] Keine exakten Constraint-Kräfte, sondern künstlich eingeführte Federkräfte.
[b] Bis auf die Anwendung unstetiger Zustandsänderungen bei der Kollisionssimulation.
[c] Nur durch wesentliche Erweiterungen der Verfahren realisierbar.
[d] Nahezu instantane Constraint-Erfüllung durch geeignete Wahl der Stabilisierungsparameter.
[e] Nur durch wesentliche Einschränkungen anderer Eigenschaften sinnvoll anwendbar.

Tabelle 4.2: Realisierbarkeit (ja, nein oder nur durch zusätzliche Aufwendungen möglich) und Relevanz der Anwendungsaspekte. Für die Punkte 1a und 1b ist dabei der Grad der Effizienz bzw. Robustheit angegeben.

werden. Die LFM und die Reduktionsmethode sind für diese Aufgabe besonders gut geeignet, wobei für die LFM mit dem in Abschnitt 3.5.5 vorgestellten Algorithmus sogar ein besonders einfaches Umsetzungskonzept vorliegt.

(b) Robustheit
Eine geringe Robustheit äußert sich vor allem in der Anfälligkeit gegenüber singulären oder numerisch ungünstigen Konfigurationen. Diese Probleme können bei allen der hier besprochenen Verfahren auftreten und sind sehr anwendungsspezifisch, so daß vergleichende Aussagen über den Grad der Robustheit äußerst schwierig sind[5]. Bis auf die direkte numerische Lösung gelten die hier besprochenen Lösungsverfahren aber als relativ robust ([Val93]).

2. Einbindung nichtholonomer Constraints
Wie Tabelle 4.1 entnommen werden kann, ermöglichen die meisten Lösungsverfahren neben

[5] Das Penalty-Verfahren bietet z.B. gewisse Vorteile für die Behandlung singulärer Konfigurationen (vergl. Abschnitt 3.3), führt aber auch zu steifen Differentialgleichungen, deren robuste Lösung sehr schwierig ist.

der Behandlung von holonomen Constraints auch die Einbindung nichtholonomer Constraints. Auf diese Weise können zwei wichtige Klassen von Constraints eingesetzt werden:

(a) Einbindung von Geschwindigkeits-Constraints
Mit Geschwindigkeits-Constraints lassen sich beliebige Gleichungen festlegen, die die Zustandsvariablen, ihre zeitlichen Ableitungen und die Zeit miteinander in Beziehung setzen. Sie können vor allem für die flexible Bewegungssteuerung eine wichtige Rolle spielen. Einige grundlegende Constraints dieser Art werden in Abschnitt 5.2.2 beschrieben.

(b) Einbindung von Ungleichungen
Auch Bedingungen an die Zustandsvariablen in Form von Ungleichungen gehören zur Klasse der nichtholonomen Constraints. Als besonders wichtiger Spezialfall gehört hierzu die Forderung der Undurchdringlichkeit für dreidimensionale Körper. Eine Möglichkeit zur Lösung dieses Problems im Rahmen einer automatischen Kontaktbehandlung wird in Abschnitt 6.4 vorgestellt.

3. Explizite Bestimmung der Zwangskräfte
Der physikalische Ausdruck eines festgelegten Constraints ist die Wirkung seiner zugehörigen Zwangskraft (vergl. Abschnitt 3.5.1). Bei der LFM werden diese Kräfte ebenso wie beim Penalty-Verfahren explizit bestimmt, wobei aber bei der LFM keine künstlich eingeführten Federkräfte, sondern die exakten Constraint-Kräfte ermittelt werden. Diese Berechnung ist auch bei den anderen Verfahren möglich, dort würde sie aber einen zusätzlichen Bearbeitungsschritt erfordern, durch den die zeitliche Effizienz des Verfahrens beeinträchtigt wäre. Die Kenntnis der Zwangskräfte kann in zweierlei Weise ausgenutzt werden:

(a) Anwendung für spezielle Steuertechniken
Die Größe der Zwangskräfte stellt eine zusätzliche dynamische Systemeigenschaft dar, die von speziellen Steuertechniken ausgenutzt werden kann. Ein Beispiel hierfür wäre die Simulation von Körpern, die bei einer bestimmten „Belastung“, die sich z.B. als Summe der am jeweiligen Körper anliegenden Zwangskräfte definieren ließe, auseinanderbrechen und in mehrere Teile zerfallen. Für bestimmte Ansätze zur Kontaktbehandlung muß außerdem die Richtung dieser Kräfte bekannt sein ([Pla92]).

(b) Realisierung einer Kraftrückkopplung
Die Bestimmung der Zwangskräfte könnte auch für die Realisierung einer interessanten Interaktionstechnik hilfreich sein: der Umwandlung von Systemkräften in reale Kräfte, die dem Benutzer über ein Force Feedback- Eingabegerät erfahrbar gemacht werden. Diese Anwendung der Zwangskräfte stellt einen bislang nicht betrachteten Vorteil der LFM und des Penalty-Verfahrens dar. In [NC99] wurde eine solche Interaktionstechnik im Rahmen von Modellierungsaufgaben beschrieben, bei der das Problem der Bewegungsgenerierung unberücksichtigt blieb. Dieser Aspekt wird noch einmal im Ausblick dieser Arbeit angesprochen.

4. Erfüllung der Constraints durch Kraftwirkungen
Bei der LFM und bei der Penalty-Methode werden Constraints durch das Wirken von Kräften erfüllt. Aus dieser Eigenschaft ergeben sich interessante Aspekte für die Verwendung in einem allgemeinen Animationssystem. Der physikalisch basierte Ablauf der Constraint-Erfüllung erweist sich dabei als ein besonders wichtiger Punkt und wird daher unten gesondert behandelt.

(a) Erhaltung der Stabilität bei elastischen Körpern
Eine wichtige Bedingung bei der Simulation elastischer Körper ist die Aufrechterhaltung ihrer Stabilität. Durch die stetigen Kraftanwendungen werden extrem große Ableitungen der Zustandsvariablen vermieden, die ansonsten zu einem instabilen Verhalten

führen könnten und daher durch zusätzliche Aufwendungen vermieden werden müßten ([Pla92]).

(b) Standardmäßige Einbindung von numerischen Integrationsroutinen
Bei einer Constraint-Erfüllung durch Kräfte ist eine standardmäßige Einbindung von numerischen Integrationsroutinen möglich. In [Pla92] und [GB94], S.172, wurde darauf aufmerksam gemacht, daß der numerische Integrationsprozess bei der Reduktionsmethode dagegen im Falle eines Wechsels der Koordinaten, der z.B. durch das Auftreten von Koordinatensingularitäten notwendig sein kann, neu gestartet werden muß, wodurch sich insbesondere bei Verfahren mit variabler Schrittweite Probleme ergeben.

Der hieraus resultierende Effizienzverlust ist aber in der Regel sehr klein, da ein solcher Wechsel nur selten erforderlich ist. Außerdem kann ein Neustart der Integrationsroutinen auch im Zuge der Kollisionssimulation notwendig sein (vergl. Abschnitt 6.4). Dieser Aspekt hat daher nur eine geringe praktische Relevanz.

(c) Vermeidung ungewollter Bewegungen
Ein Aspekt, der durch die Erfüllung der Constraints durch Kraftwirkungen erschwert wird, ist die Vermeidung von Bewegungen, die als Artefakte des Lösungsverfahrens entstehen. Bei der LFM und bei dem Penalty-Verfahren können durch die stabilisierungsbedingten Kraftwirkungen Bewegungen entstehen, die nicht vom Animateur vorgegeben wurden und oftmals auch nicht von ihm gewünscht sind. Dieser Aspekt ist bislang nicht berücksichtigt worden. Konzepte und Methoden, mit denen dieses Problem abgemildert werden kann, werden in den Abschnitten 5.4.3 (in dem das Problem genauer beschrieben wird) bzw. 6.6.3 vorgestellt.

5. Physikalisch basierter Prozess der Constraint-Erfüllung
Die Constraint-Erfüllung wird bei der LFM und bei der Penalty-Methode durch einen Prozess durchgeführt, der wie die generierten Bewegungsverläufe aus der Auswertung der physikalischen Bewegungsgleichungen resultiert. Bei den beiden anderen Verfahren geschieht diese Erfüllung dagegen instantan, d.h. innerhalb eines Simulationsschrittes. Für die Technische Simulation ist dieser Umstand kaum relevant, da in der Regel Systeme mit bereits erfüllten Constraints betrachtet werden. Vor dem Hintergrund allgemeiner Animationssysteme ergeben sich hieraus aber interessante Aspekte, denn bei der interaktiven Festlegung eines Constraints kann der Systemzustand beliebig weit von seiner Erfüllung entfernt sein.

In Abbildung 4.2 wird der physikalisch basierte Prozess der Constraint-Erfüllung am Beispiel eines Point-To-Point-Constraints, der zum Zeitpunkt t_0 aktiviert wird, illustriert und der instantanen Erfüllung gegenübergestellt. Die zeitliche Spanne bis zum gewünschten Endzustand (die in dem dargestellten Beispiel 3 Simulationsschritte umfaßt) kann dabei stark variieren. Sie hängt u.a. vom Ausgangszustand und von den dynamischen Eigenschaften der Körper ab.

Dieser Aspekt wurde wie in Abschnitt 4.1.1.3 erwähnt in [BB88] als *Dynamic Constraint*-Konzept zur Unterstützung der physikalisch basierten Modellierung eingeführt. Es soll nun dargestellt werden, welche Aspekte aus dieser speziellen Art der Constraint-Erfüllung für die *Animationserstellung* resultieren, die die Modellierung des Systems lediglich als ein (wenn auch sehr wichtiges) Teilproblem umfaßt.

(a) Keine Bewegungssprünge bei der Animationserstellung
Durch die spezielle Art der Constraint-Erfüllung werden durch das Aktivieren und Deaktivieren von Constraints keine Bewegungssprünge verursacht, wie auch in [Pla92] angemerkt wurde. Solche Sprünge werden in der Regel als sehr künstlich empfunden, da sie in realen Bewegungsabläufen nicht auftreten. Um den Anspruch der visuellen Plausibilität aufrechterhalten zu können, wäre es daher nicht möglich, Constraints während

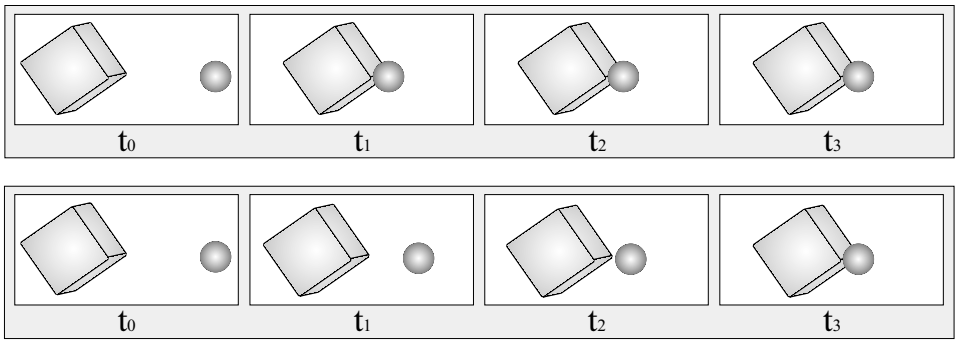

Abbildung 4.2: Instantane (oben) und stetige, physikalisch basierte Constraint-Erfüllung (unten).

einer Animation zu aktivieren oder zu deaktivieren, was eine erhebliche Einschränkung für die Flexibilität der Bewegungssteuerung darstellen würde. Im Rahmen dieser Arbeit (insbesondere in den Abschnitten 6.4.2, 6.5.4 und 6.6.1) spielt diese Eigenschaft, die auch für ein sehr allgemeines Prozedurenkonzept ausgenutzt werden konnte (vergl. Abschnitt 6.2.4.2), eine wichtige Rolle.

(b) Intuitive Anwendung von Constraints
Für die effektive Animationserstellung ist es äußerst wichtig, dem Animateur eine visuelle Rückkopplung über das Systemverhalten auf der Grundlage seiner Festlegungen zu ermöglichen. Der Prozess der Constraint-Erfüllung muß daher in einer interpretierbaren und nachvollziehbaren Weise ablaufen. Mit der stetigen, physikalisch basierten Constraint-Erfüllung wird diese Anforderung in natürlicher Weise erfüllt. Dieser Aspekt bezieht sich dabei nicht nur auf Modellierungsaufgaben, sondern auch auf die Anwendung von Constraints zur Bewegungssteuerung, wie z.B. in den Abschnitten 6.5.4 und 6.6.1 ersichtlich wird.

(c) Einheitliche Methoden für Modellierung und Bewegungssteuerung
Durch die spezielle Art der Constraint-Erfüllung wird ein sehr einheitlicher Zugang für die Teilaufgaben der Modellierung und der Bewegungssteuerung ermöglicht: Bei der Anwendung von Constraints läßt sich jeder Modelliervorgang auch als visuell plausible Animation verwenden. Für die einfache Bedienung des Animationssystems ist diese Einheitlichkeit sehr wichtig, um die einmal erlernten Techniken für möglichst viele Aufgaben einsetzen zu können (vergl. hierzu die Abschnitte 6.5.4 und 6.6.1).

(d) Möglichkeit zur instantanen Erfüllung der Constraints
Die instantane Erfüllung der Constraints führt wie oben erwähnt zu gewissen Schwierigkeiten bzw. Einschränkungen bei der Animationserstellung. In bestimmten Fällen kann ein solches Verhalten aber auch erwünscht sein, z.B. wenn die (in der Regel deutlich sichtbare) Verletzung der Constraints während des Erfüllungsprozesses vom Animateur als störend empfunden wird oder die instantane Erfüllung als gezielter visueller Effekt eingesetzt werden soll.

Bei der LFM und bei der Penalty-Methode kann der Animateur allerdings über die Festlegung der Stabilisierungsparameter den Prozess der Constraint-Erfüllung beeinflussen und damit insbesondere die dabei gewünschte Zeitspanne bis zum Erreichen des Zielzustandes festlegen (vergl. Abschnitt 6.3.4.1). Dieser Aspekt der instantanen Erfüllung, der vor allem aus Gründen der Vollständigkeit aufgeführt wurde, ist daher für die Bewertung der Verfahren weniger relevant.

6. Bewegungsgleichungen bezüglich beliebiger Zustandskoordinaten

Bei den meisten Verfahren können beliebige Koordinaten für die Aufstellung der Bewegungsgleichungen zum Einsatz kommen, sofern sie den Zustand des Systems eindeutig beschreiben. Bei der Reduktionsmethode müssen sie dagegen in unabhängigen Koordinaten formuliert werden. Aus diesem Umstand lassen sich einige wichtige Aspekte vor dem Hintergrund allgemeiner Animationssysteme ableiten:

(a) Schematische Aufstellung der Bewegungsgleichungen
Das Aufstellen der Bewegungsgleichungen läßt sich bei der Verwendung beliebiger Koordinaten sehr schematisch durchführen und ist daher im Rahmen eines Animationssystems leicht umzusetzen. Beiträge zur Durchführung dieser Aufgabe auf der Grundlage der LFM finden sich in dieser Arbeit vor allem in den Abschnitten 5.1 und 5.3. Die Formulierung dieser Gleichungen mit Hilfe unabhängiger Koordinaten ist dagegen wesentlich schwieriger ([RS88], S. 400).

(b) Unabhängige Formulierung von Körpern und Constraints
Durch die Verwendung beliebiger Zustandskoordinaten lassen sich Körper und Constraints unabhängig voneinander formulieren. Dieser Umstand erleichtert nicht nur die Aufstellung der Bewegungsgleichungen, sondern ermöglicht auch eine *Kapselung* von Körpern und Constraints in separate Objekte. Auf diese Weise können auch komplette Untersysteme abgespalten und z.B. separat gespeichert werden. Dieser Aspekt, der in [W+90] ausgenutzt und in [Bar96] explizit angesprochen wurde, ist äußerst wichtig für die Realisierung eines Animationssystems, das dem Anspruch der *Modularität* gerecht werden soll. Wie in den Abschnitten 6.5.4 und 6.6.1 ersichtlich wird, kann auf diese Weise auch die *Anwendung* von Constraints auf beliebige Körper wesentlich vereinfacht werden. In Abschnitt 5.1 wird ein sehr allgemeines Konzept vorgestellt, mit dem sich diese Anwendung ohne Aufhebung der Modularität durchführen läßt.

(c) Integrierte Bestimmung relativer Koordinaten
Als ein Nachteil kann sich die fehlende Bestimmung relativer Zustandskoordinaten auswirken, die bei der Reduktionsmethode explizit bestimmt werden. Beispiele solche relativen Koordinaten sind z.B. die Auslenkungswinkel von Scharniergelenken. Sie stellen einfach interpretierbare Zustandsgrößen dar, die insbesondere für die Bewegungssteuerung wichtig sein können, um eine effektive Einflußnahme auf das System zu ermöglichen. Ein Scharniergelenk läßt sich z.B. oftmals viel einfacher durch die Vorgabe dieses Winkels steuern, als die durch das Gelenk verbundenen Körper separat zu bewegen.

Relative Koordinaten lassen sich bei Verfahren auf der Grundlage beliebiger Zustandskoordinaten wie der LFM durch einen zusätzlichen Berechnungsschritt bestimmen, was für konkrete Fälle durchaus sinnvoll sein kann. Als allgemeine Erweiterung dieser Verfahren eignet sich diese Möglichkeit jedoch nicht. Neben dem dafür anfallenden Zeitaufwand wären damit die unter den Punkten 6a, 6b und 7a genannten Vorteile hinfällig, da sie gerade auf der Vermeidung einer solchen Koordinatentransformation beruhen. Insbesondere ist nicht zu sehen, wie die schematische, zustands- und topologieunabhängige Anwendung der LFM in einfacher Weise für die Lösung dieser Aufgabe erweitert werden kann und wie sich die Kapselung von Körpern und Constraints aufrechterhalten ließe. Der Aufgabe, auch ohne relative Koordinaten eine flexible Bewegungssteuerung zu erzielen, kommt daher eine große Bedeutung zu.

7. Verfahrensspezifische Aspekte

Es sollen nun einige Aspekte besprochen werden, die sich aus spezifischen Eigenschaften der Lösungsverfahren ergeben. Sie betreffen vor allem die schematische Anwendbarkeit und damit die einfache Automatisierbarkeit des Verfahrens, die eine unverzichtbare Voraussetzung für die Realisierung eines physikalisch basierten Animationssystems darstellt.

(a) Vermeidung einer Trennung von abhängigen und unabhängigen Koordinaten
Die meisten der hier betrachteten Lösungsverfahren erfordern keine explizite Trennung von abhängigen und unabhängigen Koordinaten. Bei der Reduktionsmethode und beim Verfahren der Koordinatenaufteilung müssen die unabhängigen Koordinaten dagegen explizit bestimmt werden, da nur die Komponenten der Bewegungsgleichungen bezüglich dieser Koordinaten integriert werden. Bei der Reduktionsmethode ist diese Bestimmung dabei schon vor der Aufstellung der Bewegungsgleichungen erforderlich (vergl. Punkt 6). Die Constraint-Gleichungen müssen dazu nach den Zustandsvariablen aufgelöst werden, was eine sehr schwer zu automatisierende Aufgabe darstellt, da diese Gleichungen hochgradig nichtlinear sein können (vergl. Abschnitt 4.1.1.2). Diese Umformung kann zudem auch während der Simulation notwendig werden, wenn Änderungen der Systemtopologie oder Koordinatensingularitäten auftreten.

(b) Anwendung auf beliebige Systemtopologien
Die Reduktionsmethode läßt sich standardmäßig nur auf hierarchische Systeme anwenden. Systeme mit geschlossenen Schleifen (zyklischen Verbindungen), die bei der Animationserstellung sehr häufig vorliegen und oftmals erst während der Simulation entstehen, erfordern die Anwendung zusätzlicher Bearbeitungsschritte, um eine geeignete Aufspaltung der Schleifen zu erreichen. Die anderen Verfahren lassen sich dagegen unabhängig von der Topologie anwenden, d.h. Systeme mit geschlossenen Schleifen können in gleicher Weise wie hierarchische Systeme behandelt werden[6]. Diese Eigenschaft trägt zu einer einfacheren Umsetzbarkeit der Verfahren bei, da eine Beschränkung auf hierarchische Systeme der angestrebten Generalität des Animationssystems entgegen stehen würde.

(c) Vermeidung zusätzlicher Stabilisierungsparameter
Die Stabilisierungsverfahren stellen bei der LFM wegen der nur indirekten Einbeziehung von holonomen und Geschwindigkeits-Constraint eine notwendige Erweiterung des Grundablaufes dar. Ihre Wirkungsweise wird dabei durch einen oder auch mehrere Parameter beeinflußt, deren Modifikation zu deutlich sichtbaren Unterschieden bei der Verfahrensanwendung führen kann. Es stellt sich somit das Problem der geeigneten Festlegung dieser Stabilisierungsparameter, ohne dabei die allgemeine Anwendung der LFM einschränken zu müssen oder ein tiefes technisches Verständnis des Animateurs zu erfordern (vergl. Abschnitt 6.3.4.1). Das gleiche Problem ergibt sich auch bei dem Penalty-Verfahren, das mit dem Penalty-Faktor sogar die Festlegung eines zusätzlichen Parameters erfordert.

4.3 Verknüpfung von Anwendungs- und Anforderungsaspekten

Es soll nun eine Zuordnung der im letzten Abschnitt angeführten Anforderungsaspekten zu den in der Einführung dieser Arbeit genannten Anforderungen an allgemeine Animationssysteme vorgenommen werden. Auf dieser Grundlage läßt sich auch eine allgemeine Einschätzung über die Eignung der Lösungsverfahren vor diesem Hintergrund treffen.

- Effektivität
  Ein hoher Grad an zeitlicher Effektivität (Punkt 1a) stellt einen äußerst wichtigen Gesichts-

[6] Die konkrete numerische Umsetzung der Verfahren kann allerdings eine Unterscheidung zwischen hierarchischen und nichthierarchischen Systemen notwendig machen. Das Treffen dieser Unterscheidung und die Behandlung des zyklischen Systems läßt sich aber auch in diesem Fall problemlos durchführen, wie z.B. in Abschnitt 5.3.3 bei der Besprechung einer Umsetzung der LFM ersichtlich wird.

punkt dar. Bezüglich dieser Eigenschaft haben die LFM, die Reduktionsmethode und mit gewissen Einschränkungen auch die Methode der Koordinatenaufteilung Vorteile gegenüber den beiden anderen Verfahren.

- Flexibilität
  Durch die Einbindung nichtholonomer Constraints (Punkte 2a und 2b) läßt sich eine große Klasse von Constraints für die Modellierung und Bewegungssteuerung einsetzen. Für die Anforderung der Flexibilität ist diese Möglichkeit, die bei der Anwendung der Reduktionsmethode nicht besteht, daher sehr wichtig. Auch die Kenntnis der Zwangskräfte kann wie unter Punkt 3a beschrieben von speziellen Steuertechniken ausgenutzt werden und damit zu einer Erhöhung der Flexibilität beitragen. Durch die Vermeidung von Bewegungssprüngen (Punkt 5a) lassen sich Constraints zudem sehr flexibel einsetzen, da sie auch während einer laufenden Simulation aktiviert und deaktiviert werden können, was einen wichtigen Vorteil der LFM und des Penalty-Verfahrens darstellt.

  Die unter Punkt 5d genannte instantane Constraint-Erfüllung läßt sich im Prinzip mit allen Lösungsverfahren erreichen. Eine große Hilfe für die Realisierung flexibler Steuertechniken kann aber die Bestimmung relativer Koordinaten darstellen, die nur bei der Reduktionsmethode vor der Aufstellung der Bewegungsgleichungen durchgeführt wird (Punkt 6c). Wie anhand der in den Abschnitten 6.2.3 und 6.2.4 beschriebenen Steuertechniken gezeigt wird, führt der Verzicht auf die Verwendung dieser Größen aber zu keinen grundsätzlichen Einschränkungen bezüglich der Flexibilität der Bewegungssteuerung.

- Intuitivität
  Gerade für die einfache und intuitive Bedienung von Animationssystemen bieten die im vorigen Abschnitt aufgeführten Anwendungsaspekte interessante Möglichkeiten. In erster Linie ist dabei die stetige Art der Constraint-Erfüllung zu nennen, mit der sich eine sehr intuitive Festlegung von Constraints erzielen läßt (Punkt 5b). Sehr wichtig ist auch die hiermit verknüpfte Einheitlichkeit bezüglich Modellierungs- und Steueraufgaben (Punkt 5c). Auch die unabhängige Formulierung von Constraints und Körpern (Punkt 6b) kann für eine einfache und leicht erlernbare Anwendung von Constraints ausgenutzt werden. Die unter Punkt 3b genannte Möglichkeit könnte zudem einen ganz speziellen Beitrag für eine intuitive Programminteraktion liefern.

  Für den Gesichtspunkt der Intuitivität bieten die LFM und das Penalty-Verfahren daher wesentliche Vorteile. Einen gewissen Nachteil für die Anwendung dieser Verfahren stellt die Anpassung der Stabilisierungsparameter (Punkt 7c) und das Auftreten ungewollter Bewegungen (Punkt 4c) dar.

- Generalität und einfache Umsetzbarkeit
  Auch die einfache Umsetzbarkeit der Lösungsverfahren im Rahmen eines allgemeinen Animationssystems, das nicht auf spezielle Anwendungen oder Systemtypen zugeschnitten ist, steht mit vielen der genannten Anwendungsaspekte in Verbindung. Wichtige Beispiele hierfür stellen die schematische Aufstellung der Bewegungsgleichungen (Punkt 6a), die Vermeidung des aufwendigen Schrittes zur Trennung von abhängigen und unabhängigen Koordinaten (Punkt 7a), die standardmäßige Einbindung von Integrationsroutinen (Punkt 4b) sowie die einfache Einbindung von elastischen Körpern (Punkt 4a) und Systemen mit geschlossenen Schleifen (Punkt 7b) dar. Auch die Robustheit der Lösungsverfahren (Punkt 1b) trägt entscheidend für eine einfache Umsetzbarkeit bei. Eine wesentliche Bedeutung für das Ziel der Modularität hat außerdem der unter Punkt 6b genannte Aspekt. Auf ihn fußt auch die modulare Konzeption und Implementierung des im Rahmen dieser Arbeit entwickelten Animationssystems EMPHAS. Die LFM und das Penalty-Verfahren bieten für diese Aspekte der Generalität und einfachen Umsetzbarkeit die größten Vorteile, wie Tabelle 4.2 entnommen werden kann.

Wie die Ausführungen in diesem Kapitel gezeigt haben, insbesondere die oben durchgeführte Verknüpfung mit den Anforderungsaspekten, bietet die LFM somit hervorragende Voraussetzungen für den Einsatz im Rahmen eines allgemeinen Animationssystems. Das Penalty-Verfahren teilt sehr viele Anwendungsaspekte mit der LFM, läßt sich aber bislang[7] bei weitem nicht so effizient umsetzen. Die anderen hier betrachteten Verfahren, insbesondere die Reduktionsmethode, führen dagegen zu wesentlichen Einschränkungen und zusätzlichen Aufwendungen bei der Umsetzung eines solchen Systems. Die direkte numerische Lösung ist zudem durch eine geringere Effizienz und Robustheit gekennzeichnet.

Die praktische Relevanz dieser Aspekte wird in den folgenden Kapiteln ersichtlich, in denen Konzepte und Methoden zur Entwicklung eines allgemeinen Animationssystems auf der Grundlage der LFM vorgestellt werden.

[7] Vergl. hierzu die Anmerkung zur Effizienz des Penalty-Verfahrens in Abschnitt 4.1.1.1.

Kapitel 5

Konzepte zur Anwendung der LFM in allgemeinen Animationssystemen

In diesem Kapitel werden Konzepte für die Realisierung allgemeiner Animationssysteme auf der Grundlage der LFM vorgestellt. Zunächst wird dazu ein sehr allgemeines Konzept für die Formulierung von beschleunigungslinearen Constraints erläutert, das ihre automatische Einbindung und modulare Realisierung ermöglicht. Auf dieser Grundlage können dann einige wichtige Constraint-Typen beschrieben werden. Anschließend wird ein modulares Konzept zur Umsetzung der LFM vorgestellt, das eine effiziente und robuste Bewegungsgenerierung erlaubt. Schließlich werden die Probleme bei der Anwendung von Constraints und anderen Steuertechniken besprochen und Konzepte zu ihrer Lösung entworfen.

5.1 Ein Konzept zur Einbindung von Constraints

Die Anwendung der LFM basiert auf der Kenntnis der Größen **J** und **c** von Gleichung (3.7), die sich aus der mathematischen Form der festgelegten Constraints ergeben (vergl. Abschnitt 3.2). Für die Umsetzung eines allgemeinen Animationssystems ergibt sich hieraus ein schwieriges Problem: Für die Constraints muß eine Formulierung gefunden werden, die eine automatische Generierung dieser Terme ermöglicht. Ein zu diesem Zweck entworfenes Konzept soll nun vorgestellt werden.

Constraints legen Beziehungen für bestimmte Zustandsvariablen fest. Da jeder Körpertyp wie Punktkörper, starrer oder flexibler Körper durch einen eigenen Satz von Zustandsvariablen gekennzeichnet ist, kann die Formulierung der Constraints nicht unabhängig vom Körpertyp erfolgen. Eine weitgehende Unabhängigkeit wäre aber sehr wünschenswert. Ein Point-to-Point-Constraint hat z.B. für alle Körper die gleiche Interpretation, nämlich das Aufeinanderfallen zweier Körperpunkte. Schon aus Gründen der einfachen Erweiterbarkeit um neue Körpertypen wäre es äußerst ungünstig, einen solchen Constraint für jeden Körpertyp getrennt formulieren und implementieren zu müssen. Verschärft wird dieses Problem noch dadurch, daß zur Einbindung der Constraints die partiellen Ableitungen bezüglich der Zustandsvariablen (vergl. Abschnitt 3.2) zur Verfügung stehen müssen.

Eine Lösung für dieses Problem bietet das in [W+90] vorgestellte *Konnektoren*-Konzept[1]. Konnektoren repräsentieren körperfeste Punkte oder andere allgemeine Anknüpfungspunkte[2]. Ihr Nutzen

[1] In [Wit94] wurden einige Hinweise zur Implementierung dieses Konzeptes gegeben.

[2] In den späteren Arbeiten [GW93], [Gle94a] wurde der Begriff *Konnektor* als sehr allgemeines Konzept für die automatische Auswertung mathematischer Funktionen verwandt. Ein expliziter Bezug zu Constraints und zur Anwendung der

liegt darin, daß sich sämtliche Constraints mit Hilfe dieser Anknüpfungspunkte formulieren lassen, ohne Bezug auf die tatsächlich vorliegenden Körper nehmen zu müssen. Ein Point-to-Point-Constraint, der auf zwei Konnektoren $\mathbf{P}_1$ und $\mathbf{P}_2$ wirkt, kann z.B. durch die einfache Formulierung $\mathbf{C} = \mathbf{P}_1 - \mathbf{P}_2$ spezifiziert werden.

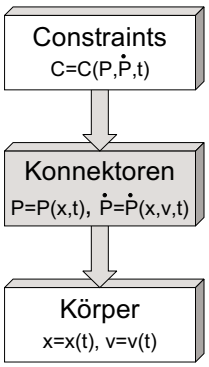

Abbildung 5.1: Das Konnektoren-Konzept.

In dieser Arbeit wurde das Konzept in zweierlei Hinsicht verallgemeinert. Zunächst wurden nicht nur holonome Constraints, sondern auch Geschwindigkeits-Constraints mit Hilfe dieser Konnektoren umformuliert, um auch diese Constraint-Typen einbinden zu können. Außerdem wurde im Gegensatz zu [W+90] eine explizite Zeitabhängigkeit der Konnektoren zugelassen. Auf diese Weise können z.B. auch Punkte, die sich auf einer explizit festgelegten raumzeitlichen Bahn bewegen, als Anknüpfungsobjekte dienen, ohne eine Sonderbehandlung notwendig zu machen. Ein Beispiel für ein solches Objekt stellt der in Abschnitt 6.2.2 beschriebene *FctConnector* dar.

Die Zugriffshierarchie von Constraints, Konnektoren und dynamischen Körpern ist in Abbildung 5.1 schematisch dargestellt. Mit Hilfe dieses Konzeptes lassen sich holonome und Geschwindigkeits-Constraints in einer sehr einfachen und allgemeinen Weise formulieren, ohne Bezug auf die verwendete Klasse von dynamischen Körpern Bezug nehmen zu müssen (vergl. hierzu die Abschnitte 5.2 und 6.2.3). Wie in Abschnitt 6.6.1 gezeigt wird, konnte auf dieser Grundlage auch eine sehr intuitive Anwendung der Constraints realisiert werden.

Im folgenden soll nun gezeigt werden, wie sich die beiden Typen der holonomen und Geschwindigkeits-Constraints im Rahmen des verallgemeinerten Konnektoren-Konzeptes formulieren lassen und welche Informationen die Konnektoren selber bereitstellen müssen.

5.1.1 Formulierung von holonomen Constraints

Holonome Constraints haben im Rahmen dieses Konzeptes die mathematische Gestalt

$$\mathbf{C}\,(\mathbf{P}_1, \ldots, \mathbf{P}_k, t\,) = 0\,. \tag{5.1}$$

Die Constraint-Funktionen können also von k Konnektoren $\mathbf{P}_i$ und der Zeit t abhängen. Die Konnektoren $\mathbf{P}_i$ stellen dabei dreidimensionale Raumpunkte dar, die sich als Funktion der Zustandsvariablen ergeben. In dieser verallgemeinerten Form können sie außerdem explizit von der Zeit abhängen:

$$\mathbf{P}_i = \mathbf{P}_i(\mathbf{x}, t)\,. \tag{5.2}$$

Die obige Constraint-Funktion $\mathbf{C}$ stellt aber eine reine Funktion der $\mathbf{P}_i$ (und der Zeit) dar, hängt also *nicht* von der Zustandsvariablen $\mathbf{x}$ ab.

Es stellt sich nun die Frage, wie diese nur indirekte Abhängigkeit von den Zustandsvariablen bei der Bildung der partiellen Ableitungen der Constraint-Funktionen aufrechterhalten werden kann. Zur Aufstellung der Gleichung (3.7) müssen die Terme $\mathbf{J}$ und $\mathbf{c}$ bekannt sein. Ein direkter Zugriff auf die Zustandsvariablen würde das Konzept aber außer Kraft setzen. Dieses Problem läßt sich durch eine Aufspaltung in körperunabhängige Constraint-Ableitungen und Constraint-unabhängige Konnektoren-Ableitungen lösen. Zu diesem Zweck soll analog zu dem Vorgehen in Abschnitt 3.2 die zweifache totale Ableitung der holonomen Constraint-Funktionen nach der Zeit betrachtet werden.

physikalisch basierten Animation wurde dabei aber nicht mehr aufgestellt.

Dabei ist zu beachten, daß $\dot{\mathbf{P}}$ im Unterschied zum ursprünglichen Konnektoren-Konzept in [W+90] eine Funktion von $\mathbf{x}$, $\mathbf{v}$ *und* t ist, so daß $\ddot{\mathbf{P}} = \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{v}}\dot{\mathbf{v}} + \frac{\partial \dot{\mathbf{P}}}{\partial t}$ gilt.

$$\begin{aligned} 0 &= \frac{d^2}{dt^2}\mathbf{C} \\ &= \frac{d}{dt}\left(\sum_{i=1}^{k}\frac{\partial \mathbf{C}}{\partial \mathbf{P}_i}\dot{\mathbf{P}}_i + \frac{\partial \mathbf{C}}{\partial t}\right) \\ &= \sum_{i=1}^{k}\frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_i}\dot{\mathbf{P}}_i + \sum_{i=1}^{k}\frac{\partial \mathbf{C}}{\partial \mathbf{P}_i}\left(\frac{\partial \dot{\mathbf{P}}_i}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{P}}_i}{\partial \mathbf{v}}\dot{\mathbf{v}} + \frac{\partial \dot{\mathbf{P}}_i}{\partial t}\right) + \frac{\partial \dot{\mathbf{C}}}{\partial t}\,. \end{aligned}$$

Um auf Gleichung (3.7) zu gelangen, ergeben sich aus dieser Gleichung die Zuweisungen

$$\begin{aligned} J_{ij} &= \sum_{l=1}^{k}\frac{\partial C_i}{\partial \mathbf{P}_l}\frac{\partial \dot{\mathbf{P}}_l}{\partial v_j} \\ c_i &= \sum_{l=1}^{k}\frac{\partial \dot{C}_i}{\partial \mathbf{P}_l}\dot{\mathbf{P}}_l + \sum_{l=1}^{k}\frac{\partial C_i}{\partial \mathbf{P}_l}\left(\frac{\partial \dot{\mathbf{P}}_l}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{P}}_l}{\partial t}\right) + \frac{\partial \dot{C}_i}{\partial t}\,. \end{aligned} \tag{5.3}$$

Für die in Abschnitt 3.5.4 beschriebene Anwendung der Baumgart-Stabilisierung müssen für die Behandlung der Constraints zudem die Terme $\mathbf{C}$ und $\dot{\mathbf{C}}$ für jeden Zeitpunkt bekannt sein, wobei sich $\dot{\mathbf{C}}$ wie oben gesehen aus den Termen $\frac{\partial \mathbf{C}}{\partial \mathbf{P}}$, $\dot{\mathbf{P}}$ und $\frac{\partial \mathbf{C}}{\partial t}$ ableiten läßt.

Auf diese Weise ist die gewünschte Aufspaltung vollzogen worden. Die Ableitungen der Constraint-Gleichungen bezüglich der Konnektoren $\frac{\partial C_i}{\partial \mathbf{P}_j}$, $\frac{\partial \dot{C}_i}{\partial \mathbf{P}_j}$ und $\frac{\partial \dot{C}_i}{\partial t}$ können nun ohne Kenntnis der Zustandsvariablen $\mathbf{x}$ durchgeführt werden und die Terme $\frac{\partial \dot{\mathbf{P}}_i}{\partial v_j}$, $\frac{\partial \dot{\mathbf{P}}_i}{\partial x_j}$ und $\frac{\partial \dot{\mathbf{P}}_i}{\partial t}$ sind unabhängig von den Constraints. Zusammengefaßt müssen für die Formulierung von holonomen Constraints die folgenden Größen bekannt sein:

- die Dimension (Komponentenanzahl) m des Constraints
- die Komponenten C_i der Constraint-Funktion (Skalare)
- die Anzahl k der mit dem Constraint verknüpften Konnektoren $\mathbf{P}_j$
- die Ableitungen $\frac{\partial C_i}{\partial \mathbf{P}_j}$ von C_i bezüglich des j-ten Konnektors (3D-Vektoren)
- die Ableitungen $\frac{\partial \dot{C}_i}{\partial \mathbf{P}_j}$ von $\dot{C}_i$ bezüglich des j-ten Konnektors (3D-Vektoren)
- die Ableitungen $\frac{\partial \dot{C}_i}{\partial t}$ von $\dot{C}_i$ bezüglich der Zeit (Skalare)
- die Komponenten $\dot{C}_i$ der differenzierten Constraint-Funktion (Skalare)

Anstatt der letztgenannten Größe, die für die Stabilisierung der LFM mit Hilfe der Baumgart-Methode wichtig ist, ließe sich auch der Term $\frac{\partial C_i}{\partial t}$ angeben, aus dem der Wert für $\dot{C}_i$ aber jedesmal berechnet werden müßte.

Dieser Formalismus läßt sich anhand eines einfachen Beispieles verdeutlichen. Einem Point-to-Point-Constraint entspricht die dreidimensionale Funktion $\mathbf{C} = \mathbf{P}_1 - \mathbf{P}_2$, d.h. es gilt $m = 3$ und $k = 2$. Die Ableitungen von $\mathbf{C}$ ergeben sich hier in trivialer Weise:

$$\dot{\mathbf{C}} = \dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2\,, \quad \frac{\partial \mathbf{C}}{\partial \mathbf{P}_1} = -\frac{\partial \mathbf{C}}{\partial \mathbf{P}_2} = \mathbf{E}\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_1} = \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_2} = 0\,.$$

Die Ableitungen der Konnektor-Funktionen auf der anderen Seite ergeben sich gerade *nicht* aus der speziellen Form des Constraints. Ihre Ausformulierung wird in Abschnitt 5.1.3 vorgenommen. Zunächst soll aber gezeigt werden, wie sich Geschwindigkeits-Constraints als andere wichtige Unterklasse beschleunigungslinearer Constraints im Rahmen dieses Konnektoren-Konzeptes darstellen.

5.1.2 Formulierung von Geschwindigkeits-Constraints

Bei der Formulierung von Geschwindigkeits-Constraints sind ähnliche Probleme wie im vorigen Abschnitt zu lösen. Hier sind es die Terme der Gleichung (3.6), die in eine geeignete Form gebracht werden müssen.

Geschwindigkeits-Constraints können im Gegensatz zu holonomen Constraints sowohl von den Zustandsvariablen selber als auch von ihren zeitlichen Ableitungen abhängen. Auch diese Beziehung läßt sich indirekt ausdrücken, indem die Constraints als Funktion der Konnektoren $\mathbf{P}_i$, ihrer zeitlichen Ableitungen $\dot{\mathbf{P}}_i$ (der Konnektoren-Geschwindigkeit) und der Zeit geschrieben werden:

$$\mathbf{C}(\mathbf{P}_1,\ldots,\mathbf{P}_k,\dot{\mathbf{P}}_1,\ldots,\dot{\mathbf{P}}_k,t)=0\,. \tag{5.4}$$

Auch hier wird eine explizite Zeitabhängigkeit der Konnektoren zugelassen:

$$\mathbf{P}_i=\mathbf{P}_i(\mathbf{x},t)\,,\quad \dot{\mathbf{P}}_i=\dot{\mathbf{P}}_i(\mathbf{x},\mathbf{v},t)\,. \tag{5.5}$$

Die zeitliche Ableitung der Constraint-Funktion liefert somit den Ausdruck

$$\begin{aligned}
0 &= \frac{d}{dt}\mathbf{C}\\
&= \sum_{i=1}^{k}\frac{\partial\mathbf{C}}{\partial\mathbf{P}_i}\dot{\mathbf{P}}_i+\sum_{i=1}^{k}\frac{\partial\mathbf{C}}{\partial\dot{\mathbf{P}}_i}\ddot{\mathbf{P}}_i+\frac{\partial\mathbf{C}}{\partial t}\\
&= \sum_{i=1}^{k}\frac{\partial\mathbf{C}}{\partial\mathbf{P}_i}\dot{\mathbf{P}}_i+\sum_{i=1}^{k}\frac{\partial\mathbf{C}}{\partial\dot{\mathbf{P}}_i}\left(\frac{\partial\dot{\mathbf{P}}_i}{\partial\mathbf{x}}\dot{\mathbf{x}}+\frac{\partial\dot{\mathbf{P}}_i}{\partial\mathbf{v}}\dot{\mathbf{v}}+\frac{\partial\dot{\mathbf{P}}_i}{\partial t}\right)+\frac{\partial\mathbf{C}}{\partial t}\,.
\end{aligned}$$

Die Größen **J** und **c** ergeben sich demnach als

$$\begin{aligned}
J_{ij} &= \sum_{l=1}^{k}\frac{\partial C_i}{\partial\dot{\mathbf{P}}_l}\frac{\partial\dot{\mathbf{P}}_l}{\partial v_j}\\
c_i &= \sum_{l=1}^{k}\frac{\partial C_i}{\partial\mathbf{P}_l}\dot{\mathbf{P}}_l+\sum_{i=1}^{k}\frac{\partial C_i}{\partial\dot{\mathbf{P}}_l}\left(\frac{\partial\dot{\mathbf{P}}_l}{\partial\mathbf{x}}\dot{\mathbf{x}}+\frac{\partial\dot{\mathbf{P}}_l}{\partial t}\right)+\frac{\partial C_i}{\partial t}\,.
\end{aligned} \tag{5.6}$$

Für die Anwendung der Stabilisierungsverfahren müssen lediglich die Constraint-Komponenten C_i bekannt sein. Zur Formulierung von Geschwindigkeits-Constraints sind somit die folgenden Informationen erforderlich:

- die Dimension m des Constraints
- die Komponenten C_i der Constraint-Funktion (Skalare)
- die Anzahl k der zugehörigen Konnektoren $\mathbf{P}_j$
- die Ableitungen $\frac{\partial C_i}{\partial\mathbf{P}_j}$ von C_i bezüglich des j-ten Konnektors (3D-Vektoren)
- die Ableitungen $\frac{\partial C_i}{\partial\dot{\mathbf{P}}_j}$ von C_i bezüglich der j-ten Konnektor-Geschwindigkeit (3D-Vektoren)

- die Ableitungen $\frac{\partial C_i}{\partial t}$ von C_i bezüglich der Zeit (Skalare)

Mit Hilfe dieser Terme lassen sich auch Geschwindigkeits-Constraints in die physikalische Simulation einbinden. Als nächstes sollen nun die Ableitungen der Konnektor-Funktionen $\mathbf{P}$ und $\dot{\mathbf{P}}$ untersucht werden.

5.1.3 Formulierung von Konnektoren

An den Beziehungen in den beiden vorherigen Abschnitten läßt sich erkennen, daß ein Konnektor für holonome und Geschwindigkeits-Constraints die gleichen Informationen bereitstellen muß:

- seine Position $\mathbf{P}$ bezüglich des Weltkoordinatensystems (3D-Vektor)
- seine Geschwindigkeit $\dot{\mathbf{P}}$ bezüglich des Weltkoordinatensystems (3D-Vektor)
- die Anzahl n der Zustandsvariablen des zugehörigen Körpers
- die Ableitungen $\frac{\partial \dot{\mathbf{P}}}{\partial x_i}$ von $\dot{\mathbf{P}}$ bezüglich der Zustandsvariablen x_i (3D-Vektoren)
- die Ableitungen $\frac{\partial \dot{\mathbf{P}}}{\partial v_i}$ von $\dot{\mathbf{P}}$ bezüglich der Geschwindigkeits-Variablen v_i (3D-Vektoren)
- die Ableitung $\frac{\partial \dot{\mathbf{P}}}{\partial t}$ von $\dot{\mathbf{P}}$ bezüglich der Zeit (3D-Vektor)

Diese Terme sollen nun für die Klassen der punktförmigen und starren Körper konkretisiert werden, die zwei besonders wichtige Körpertypen darstellen und auch bei dem im Rahmen dieser Arbeit entwickelten Animationssystem zum Einsatz kommen.

5.1.3.1 Konnektoren für Punktkörper

Für punktförmige Körper ergeben sich die oben angeführten Ausdrücke in trivialer Weise aus der dreidimensionalen Zustandsvariable $\mathbf{x}$, denn $\mathbf{P}$ fällt mit der Körperposition $\mathbf{x}$ und $\dot{\mathbf{P}}$ mit $\mathbf{v} = \dot{\mathbf{x}}$ zusammen. Als partielle Ableitungen resultieren demnach

$$\frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{v}} = \mathbf{E} \, , \, \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{x}} = 0 \, . \tag{5.7}$$

Eine explizite Zeitabhängigkeit von $\dot{\mathbf{P}}$ liegt zudem nicht vor, d.h. es gilt $\frac{\partial \dot{\mathbf{P}}}{\partial t} = 0$.

5.1.3.2 Konnektoren für starre Körper

Für die Zustandsbeschreibung starrer Körper sollen an dieser Stelle die in Abschnitt 3.1 eingeführten Referenzpunktkoordinaten verwendet werden. Ein Konnektor auf einem derartig beschriebenen starren Körper mit dem Schwerpunktsvektor $\mathbf{x} := \mathbf{r}_{SP}$ entspricht einem körperfesten Punkt, der durch einen zeitunabhängigen Ortsvektor $\mathbf{p}$ bezüglich des Koordinatensystems des Körpers festgelegt ist. Mit Hilfe der Rotationsmatrix $\mathbf{R}$ läßt sich die absolute Position eines solchen Punktes durch die Beziehung

$$\mathbf{P}(t) = \mathbf{x}(t) + \mathbf{R}(t)\,\mathbf{p}$$

bestimmen (vergl. Anhang E)[3]. Als zeitliche Ableitung resultiert hieraus unter Anwendung der Gleichung (E.1) aus Anhang E

$$\dot{\mathbf{P}} = \dot{\mathbf{x}} + \omega \times (\mathbf{R}\,\mathbf{p}) \, . \tag{5.8}$$

[3]Der Ausdruck $\mathbf{R}(t)\,\mathbf{p}$ ließe sich genauso mit Hilfe der Quaternion $\mathbf{q}$, die die Rotation des Körpers beschreibt, als $\mathbf{q}(t)\,\mathbf{p}\,\bar{\mathbf{q}}(t)$ formulieren (siehe Anhang E), was aber für die Ausführungen in diesem Abschnitt nicht relevant ist.

Mit Hilfe des Sternoperators (Definition (G.1) in Anhang G) läßt sich diese Gleichung als $\dot{\mathbf{P}} = \dot{\mathbf{x}} - (\mathbf{R}\,\mathbf{p}) \times \omega = \dot{\mathbf{x}} - (\mathbf{R}\,\mathbf{p})^\star\,\omega$ schreiben. Faßt man nun noch ω und $\dot{\mathbf{x}}$ in den Geschwindigkeitsvektor $\mathbf{v}$ zusammen, erhält man

$$\dot{\mathbf{P}} = \begin{pmatrix} \mathbf{E} \\ -(\mathbf{R}\,\mathbf{p})^\star \end{pmatrix} \mathbf{v}\,, \quad \mathbf{v} = \begin{pmatrix} \dot{\mathbf{x}} \\ \omega \end{pmatrix}.$$

Da $\mathbf{R}$ nicht von den Geschwindigkeitsvariablen abhängt, läßt sich aus dieser Beziehung die Ableitung nach den Geschwindigkeits-Variablen ableiten, die in dieser Form einer (3×6)-Matrix entspricht:

$$\frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{v}} = \begin{pmatrix} \mathbf{E} \\ -(\mathbf{R}\,\mathbf{p})^\star \end{pmatrix}. \tag{5.9}$$

Auch hier gilt zudem wieder $\frac{\partial \dot{\mathbf{P}}}{\partial t} = 0$.

Als letztes muß schließlich noch die Ableitung $\partial\dot{\mathbf{P}}/\partial\mathbf{x}$ bestimmt werden, die bei einem 7-dimensionalen Zustandsvektor einer (3×7)-Matrix entspricht. Da die Rotationsmatrix $\mathbf{R}$ in sehr komplizierter Weise von den Rotations-Zustandsvariablen (in Form der Quaternionen-Komponenten) abhängt, ist es aber günstiger, stattdessen einen Ausdruck für den Term $\frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{x}}\dot{\mathbf{x}}$ herzuleiten. Wie den beiden vorherigen Abschnitten entnommen werden kann, geht die Ableitung von $\dot{\mathbf{P}}$ nach den Zustandsvariablen nur in dieser Form in die Constraint-Formulierungen ein.

Um diesen Term bestimmen zu können, soll nun die zweite totale Ableitung von $\mathbf{P}$ nach der Zeit betrachtet werden:

$$\begin{aligned} \ddot{\mathbf{P}} &= \ddot{\mathbf{x}} + \dot{\omega} \times (\mathbf{R}\,\mathbf{p}) + \omega \times \frac{d}{dt}(\mathbf{R}\,\mathbf{p}) \\ &= \ddot{\mathbf{x}} - (\mathbf{R}\,\mathbf{p})^\star\,\dot{\omega} + \omega \times (\omega \times (\mathbf{R}\,\mathbf{p}))\,. \end{aligned}$$

Vergleicht man diesen Ausdruck mit der allgemeinen Beziehung $\ddot{\mathbf{P}} = \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{x}}\dot{\mathbf{x}} + \frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{v}}\dot{\mathbf{v}} + \frac{\partial \dot{\mathbf{P}}}{\partial t}$, ergibt sich der gesuchte Term offensichtlich als

$$\frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{x}}\dot{\mathbf{x}} = \omega \times (\omega \times (\mathbf{R}\,\mathbf{p}))\,, \tag{5.10}$$

denn in diesem Fall ist $\frac{\partial \dot{\mathbf{P}}}{\partial t} = 0$ und gemäß Gleichung (5.9) gilt $\frac{\partial \dot{\mathbf{P}}}{\partial \mathbf{v}}\dot{\mathbf{v}} = \ddot{\mathbf{x}} - (\mathbf{R}\,\mathbf{p})^\star\,\dot{\omega}$. Mit diesen Beziehungen sind nun alle Größen bekannt, die für die Formulierung von beschleunigungslinearen Constraints im Rahmen des verallgemeinerten Konnektoren-Konzeptes benötigt werden.

5.2 Ein Katalog grundlegender Constraints

Constraints legen Beziehungen für bestimmte Zustandsvariablen fest und sind daher ein grundlegendes Hilfsmittel für die Modellierung und Bewegungssteuerung. In diesem Abschnitt sollen nun einige grundlegende Constraints und ihre mathematische Formulierung vorgestellt werden. Im Rahmen des in Abschnitt 5.1 beschriebenen Konzeptes greifen sie dazu auf Konnektoren zurück, die feste Raumpunkte, körperfeste Punkte eines starren Körpers oder andere Anknüpfungspunkte repräsentieren. Auf diese Weise erhalten sie eine einfache mathematische Form, die zudem auf beliebige Klassen dynamischer Körper anwendbar ist. Die Auswahl dieser Constraints, die natürlich keinen Anspruch auf Vollständigkeit erheben kann, da sich *jede* Beziehung zwischen den Zustandsvariablen als Constraint formulieren läßt, geschah dabei unter zwei Gesichtspunkten: die Einfachheit der Constraints und ihre Bedeutung für die Modellierung und Bewegungssteuerung im Rahmen der physikalisch basierten Animation.

Die größte Gruppe stellen dabei die holonomen Constraints dar, die zuerst betrachtet werden. Im Bereich der Technischen Simulation wird dieser Constraint-Typ überwiegend (wie in [RS88]) oder

sogar ausschließlich (wie in [ESF98]) betrachtet. Anschließend werden einige grundlegende Geschwindigkeits-Constraints vorgestellt. Dabei wird für jeden Typ die von den jeweiligen Konnektoren und ggf. dem Zeitparameter t abhängige Funktion angegeben, durch die der Constraint eindeutig bestimmt ist. Die hieraus resultierenden Ableitungen, die für die Einbindung der Constraints benötigt werden (vergl. Abschnitt 5.1), sind in Anhang G aufgeführt. Eine tabellarische Übersicht über die hier besprochenen Constraints findet sich am Ende dieses Abschnitts.

Ein bestimmter Constraint kann durch mehrere Constraint-Funktionen festgelegt werden, die zwar eine identische physikalische Interpretation haben, aber für die numerische Lösung der Constraints sehr unterschiedlich geeignet sein können. Einige grundlegende Richtlinien für die Wahl dieser Funktionen wurden in [Gle94a], S. 137, angegeben. Als besonders wichtige Aspekte lassen sich die drei folgenden Punkte anführen:

1. Einfache Funktionen führen zu besseren Ergebnissen als komplizierte.
2. Jeder Freiheitsgrad sollte durch eine einzelne Gleichung beschränkt werden (z.B. $x = 0, y = 0$ statt $x^2 + y^2 = 0$).
3. Die Ableitungen der Funktion sollten nicht verschwinden, wenn der Constraint erfüllt ist (wie es z.B. bei $x^2 + y^2 = 0$ der Fall wäre).

Diese Richtlinien wurden auch bei den hier vorgestellten Constraints berücksichtigt.

5.2.1 Holonome Constraints

PointToPoint. Dies ist ein einfacher, aber sehr grundlegender Constraint-Typ, der sich sowohl für die Modellierung als auch für die Bewegungssteuerung einsetzen läßt. Er fordert das räumliche Zusammenfallen zweier Punkte $\mathbf{P}_1$ und $\mathbf{P}_2$ und damit das Verschwinden der Funktion

$$\mathbf{C}(\mathbf{P}_1, \mathbf{P}_2) = \mathbf{P}_1 - \mathbf{P}_2 .$$

Wenn $\mathbf{P}_1$ und $\mathbf{P}_2$ körperfeste Punkte auf zwei Körpern sind, wird auf diese Weise ein Kugelgelenk spezifiziert. Falls einer der beiden Punkte einen benutzerdefinierten Wert hat, erhält der Constraint die Interpretation einer räumlichen Zielvorgabe für einen Körperpunkt. Diese beiden Einsatzmöglichkeiten machen deutlich, wie universell *PointToPoint*-Constraints bei der Animationserstellung eingesetzt werden können.

Die Constraint-Funktion $C(\mathbf{P}_1, \mathbf{P}_2) = |\mathbf{P}_1 - \mathbf{P}_2|$, die ebenfalls einem solchen Constraint entspricht, würde dagegen zu einem sehr instabilen Verhalten führen und stellt ein Beispiel für die Verletzung der Punkte 2 und 3 der oben angegebenen Richtlinien dar.

Distance. Dieser skalare Constraint legt den räumlichen (euklidischen) Abstand zweier Punkte $\mathbf{P}_1$ und $\mathbf{P}_2$ auf einen festen Wert d fest:

$$C(\mathbf{P}_1, \mathbf{P}_2) = |\mathbf{P}_1 - \mathbf{P}_2| - d .$$

Auf diese Weise lassen sich z.B. zwei Objekte wie durch ein starres Segment mit einer vorgebbaren Länge miteinander verbinden.

Auch dieser Constraint-Typ ist leicht interpretierbar und sehr universell einsetzbar. Bei der Bewegungssteuerung und vor allem bei der Modellierung wird er daher sehr häufig benötigt.

Orientation. Oftmals ergibt sich der Wunsch, einer körperfesten Achse eine bestimmte Orientierung, d.h. Richtung, zu geben. Eine solche Achse läßt sich durch die Angabe zweier räumlich getrennter Körperpunkte $\mathbf{P}_1$ und $\mathbf{P}_2$ auf dieser Achse spezifizieren. Ein *Orientation*-Constraint fordert nun die parallele Ausrichtung mit einer zweiten Achse, die analog durch zwei Punkte $\mathbf{P}_3$ und $\mathbf{P}_4$ festgelegt ist. Diese Bedingung ist gegeben, wenn das Kreuzprodukt der beiden Vektoren $\mathbf{d}_1 := \mathbf{P}_1 - \mathbf{P}_2$ und $\mathbf{d}_2 := \mathbf{P}_3 - \mathbf{P}_4$ verschwindet. Als mathematische Entsprechung des Constraints würde sich also

$$\mathbf{C}(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4) = \mathbf{d}_1 \times \mathbf{d}_2$$

anbieten. In dieser Form ist die Constraint-Gleichung aber nicht geeignet – sie hat die Dimension 3, obwohl durch ein Orientierungs-Constraint nur 2 Freiheitsgrade verloren gehen. Tatsächlich ist es ausreichend, eine solche Beziehung für die zweidimensionalen *Projektionen* der beiden Achsen auf einer Fläche senkrecht zu einer der beiden Achsen zu fordern. Wenn $\mathbf{f}_1$ und $\mathbf{f}_2$ Projektionsvektoren darstellen, die einen beliebigen dreidimensionalen Vektor auf die erste bzw. zweite Komponente des auf die Fläche projizierten zweidimensionalen Vektors abbilden, lassen sich die zwei Constraint-Komponenten durch

$$\begin{aligned} C_1(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4) &= (\mathbf{d}_1 \times \mathbf{d}_2) \cdot \mathbf{f}_1 \\ C_2(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4) &= (\mathbf{d}_1 \times \mathbf{d}_2) \cdot \mathbf{f}_2 \end{aligned}$$

definieren.

InLine. Um die Position eines Punktes auf eine beliebig im Raum liegende Gerade zu beschränken, läßt sich ein zweidimensionaler *InLine*-Constraint anwenden. Die Gerade wird dabei durch zwei Körperpunkte oder einen konstanten Vektor festgelegt. Ein solcher Constraint läßt sich durch die dreidimensionale Funktion

$$\mathbf{C}(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3) = (\mathbf{P}_1 - \mathbf{P}_2) \times (\mathbf{P}_3 - \mathbf{P}_1)$$

präzisieren. Auch hier kann aber wieder analog zum *Orientation*-Constraint eine Projektion mit Hilfe zweier Projektionsvektoren $\mathbf{f}_1$ und $\mathbf{f}_2$ durchgeführt werden:

$$\begin{aligned} C_1(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3) &= \mathbf{C}(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3) \cdot \mathbf{f}_1 \\ C_2(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3) &= \mathbf{C}(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3) \cdot \mathbf{f}_2 \,. \end{aligned}$$

Ortho. Ein für zwei Körperachsen festgelegter *Ortho*-Constraint bewirkt, daß diese senkrecht (orthogonal) aufeinander stehen. Auf diese Weise wird ein Freiheitsgrad eingeschränkt. Wenn $\mathbf{d}_1 := \mathbf{P}_1 - \mathbf{P}_2$ und $\mathbf{d}_2 := \mathbf{P}_3 - \mathbf{P}_4$ die beiden Achsen bezeichnen, ist dieser Constraint durch die Beziehung

$$C(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4) = \mathbf{d}_1 \cdot \mathbf{d}_2$$

gekennzeichnet.

Als eigenständiger Constraint wird eine solche Vorgabe nur selten benötigt. Mit seiner Hilfe können aber z.B. komplexere Constraint-Typen wie das *Universal*-Gelenk realisiert werden.

Hinge. Mit diesem Constraint lassen sich Scharniergelenke festlegen. Als Relativbewegung sind dabei nur Drehungen um eine feste Achse erlaubt, so daß das System um 5 Freiheitsgrade reduziert wird. Zusammen mit Kugelgelenken stellt diese Gelenkart das wichtigste Konstruktionsmittel zur Modellierung von Gelenkkörpern dar.

Die mathematische Form dieses Constraint-Typs entspricht der Kombination von einem *PointToPoint*- und einem *Orientation*-Constraint:

$$\begin{aligned} C_1^{Hinge} &= C_1^{PtP}\,, \quad C_2^{Hinge} = C_2^{PtP}\,, \quad C_3^{Hinge} = C_3^{PtP} \\ C_4^{Hinge} &= C_1^{Ori}\,, \quad C_5^{Hinge} = C_2^{Ori}\,. \end{aligned}$$

Slide. Auch Gleitgelenke (*cylinder joints*) stellen ein nützliches Bindeglied zwischen zwei Körpern dar. Sie beschränken die relative Bewegung auf eine eindimensionale Translation entlang einer Verbindungslinie und erlauben nur Rotationen um diese Linie. Ein solches Gelenk kann durch den *Slide*-Constraint modelliert werden. Er läßt sich so wie der *Hinge*-Constraint als Kombination zweier einfacher Constraints formulieren, in diesem Fall von zwei *Inline*-Constraints:

$$\begin{aligned} C_1^{Slide} &= C_1^{Inline1}, C_2^{Slide} = C_2^{Inline1} \\ C_3^{Slide} &= C_1^{Inline2}, C_4^{Slide} = C_2^{Inline2}. \end{aligned}$$

Universal. Ein mittels *PointToPoint*-Constraint modelliertes Kugelgelenk erlaubt drei relative Freiheitsgrade. Oftmals ist es wünschenswert, bei einer solchen Verbindung die Drehungen um die durch den Verbindungspunkt gehenden Achsen zu verbieten. Eine solche Verbindung läßt sich mit dem sogenannten *Universal*-Constraint erzielen.

Dieser Constraint entspricht der Kombination aus einem *PointToPoint*- und einem *Ortho*-Constraint:

$$\begin{aligned} C_1^{Uni} &= C_1^{PtP}, C_2^{Uni} = C_2^{PtP}, C_3^{Uni} = C_3^{PtP} \\ C_4^{Uni} &= C^{Ortho}. \end{aligned}$$

PointToFunction. Alle bislang vorgestellten Constraints sind implizit zeitabhängig, da sie sich auf Raumpunkte beziehen, die zeitlich variieren können. Der Zeitparameter tritt aber nicht explizit in den Constraint-Gleichungen auf. Um für einen Körperpunkt eine explizite funktionale Bahn vorzugeben, ist diese explizite Berücksichtigung der Zeit aber unvermeidlich. Eine solche Steuermöglichkeit bietet der *PointToFunction*-Constraint. Mathematisch wird er durch die Funktion

$$\mathbf{C}(\mathbf{P}, t) = \mathbf{P} - \begin{pmatrix} f_x(t) \\ f_y(t) \\ f_z(t) \end{pmatrix}$$

beschrieben, wobei f_x, f_y und f_z beliebige Funktionen der Zeit t darstellen.

5.2.2 Geschwindigkeits-Constraints

Geschwindigkeits-Constraints entsprechen im Rahmen des Konnektorenkonzeptes Funktionen, die neben der Zeit t und Konnektoren $\mathbf{P}_i$ auch von Konnektorgeschwindigkeiten $\dot{\mathbf{P}}_i$ abhängen können. Auch sie können für die flexible Bewegungssteuerung eine wichtige Rolle spielen. Einige grundlegenden Beispiele dieser Constraints sollen nun vorgestellt werden.

EqualVelocity. Als wichtiger Grundtyp ist der dreidimensionale *EqualVelocity*-Constraint durch die Beziehung

$$\mathbf{C}(\dot{\mathbf{P}}_1, \dot{\mathbf{P}}_2) = \dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2$$

gekennzeichnet, wobei $\dot{\mathbf{P}}_1$ und $\dot{\mathbf{P}}_2$ die Geschwindigkeiten zweier Anknüpfungspunkte bezeichnen. Mit seiner Hilfe läßt sich die Geschwindigkeit von zwei Körperpunkten auf den gleichen Wert beschränken. Durch die Anwendung einer Konnektor-Geschwindigkeit mit einem zeitlich konstanten Wert kann einem Körperpunkt außerdem eine feste Geschwindigkeitsvorgabe auferlegt werden.

Ein solcher Constraint läßt sich vor allem zur Bewegungssteuerung einsetzen, um mehreren Objekten das gleiche Bewegungsverhalten zu verleihen. Er kann aber auch als Modellierhilfe Verwendung finden: Mit seiner Hilfe kann ein Körper einem anderen Körper als „begleitendes Objekt“ zugeordnet werden – durch die Verknüpfung bewegen sich die Körper stets parallel zueinander, wobei ihr

Abstand (im Gegensatz z.B. zu einer Verknüpfung mit einem *Distance*-Constraint) jederzeit ohne Beeinträchtigung der Bewegungen vom Animateur geändert werden kann.

EqualVelocityNorm. Dieser Constraint legt den Geschwindigkeits*betrag* zweier Punkte auf den gleichen Wert fest. Er ist somit eindimensional und wird durch die Gleichung

$$C(\dot{\mathbf{P}}_1, \dot{\mathbf{P}}_2) = |\dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2|$$

beschrieben[4]. Die *Richtung* der jeweiligen Geschwindigkeitsvektoren wird auf diese Weise nicht eingeschränkt.

Dieser Constraint kann z.B. zu interessanten Bewegungen führen, wenn er auf eine ganze Gruppe von Körpern in der Art eines Partikelsystems angewandt wird.

VelocityToFunction. Analog zum *PointToFunction*-Constraint lassen sich auch Geschwindigkeiten durch eine explizit zeitabhängige Funktion vorgeben. Eine solche Vorgabe ermöglicht der *VelocityToFunction*-Constraint:

$$\mathbf{C}(\dot{\mathbf{P}}, t) = \dot{\mathbf{P}} - \begin{pmatrix} f_x(t) \\ f_y(t) \\ f_z(t) \end{pmatrix}.$$

Die Funktionen f_x, f_y und f_z stellen auch hier beliebige Funktionen der Zeit dar. Sie entsprechen den *Ableitungen* der gleichnamigen Funktionen des *PointToFunction*-Constraints. In ihren Einsatzmöglichkeiten sind diese beiden Constraints aber dennoch nicht identisch. Mit dem *PointToFunction*-Constraint wird die *absolute* Position des Körperpunktes als Funktion der Zeit vorgegeben, während der *VelocityToFunction*-Constraint die Bewegung *relativ* zur aktuellen Position des Körperpunktes festlegt.

In Tabelle 5.1 sind die hier vorgestellten Constraints noch einmal aufgeführt.

Constraint	Dim.	holon.	rheon.	Constraint	Dim.	holon.	rheon.
PointToPoint	3	ja	nein	*Slide*	4	ja	nein
Distance	1	ja	nein	*Universal*	4	ja	nein
UnitDistance	1	ja	nein	*PointToFunction*	3	ja	ja
Orientation	2	ja	nein	*EqualVelocity*	3	nein	nein
InLine	2	ja	nein	*EqualVelocityNorm*	1	nein	nein
Ortho	1	ja	nein	*VelocityToFunction*	3	nein	ja
Hinge	5	ja	nein				

Tabelle 5.1: Ein Katalog grundlegender Constraints. Dabei ist angegeben, welche Dimension der Constraint hat, ob er holonom ist und ob er rheonom, d.h. explizit zeitabhängig ist.

5.3 Ein Konzept zur Umsetzung der LFM

Es soll nun ein modulares Konzept zur Umsetzung der LFM vorgestellt werden, das die Einbindung der oben beschriebenen und aller anderen beschleunigungslinearen Constraints ermöglicht. Zunächst

[4]Um den Punkt 3 der oben angeführten Richtlinien nicht zu verletzen, muß die Ableitung dieser Constraint-Funktionen dabei für den Fall $C(\dot{\mathbf{P}}_1, \dot{\mathbf{P}}_2) = 0$ auf einen kleinen festen Wert gesetzt werden.

muß dazu die Frage geklärt werden, welche grundlegenden Möglichkeiten zur Beschreibung des Systemzustandes bestehen, mit welchen Formalismen die Bewegungsgleichungen aufgestellt werden können und wie diese Ansätze zu bewerten sind. Diese Untersuchung, die sich auf Arbeiten aus dem Bereich der Technischen Simulation stützt, stellt in diesem Rahmen einen neuen Beitrag dar.

Als Grundlage des hier vorgestellten Konzeptes wird dann eine Aufspaltung des LFM-Verfahrens in verschiedene, voneinander unabhängige Teilaufgaben besprochen. Anschließend wird eine allgemeine und robuste Anwendung des in Abschnitt 3.5.5 beschriebenen Baraff-Verfahrens vorgestellt. Diese Aspekte wurden in der ursprünglichen Arbeit [Bar96] nicht erörtert.

5.3.1 Mathematischer Formalismus

5.3.1.1 Koordinaten für die Zustandsbeschreibung

Die Auswahl eines Satzes von Zustandskoordinaten, die den Systemzustand eindeutig beschreiben, erscheint auf den ersten Blick trivial, da die verschiedenen Koordinatensätze physikalisch gesehen vollkommen äquivalent sind und sich zudem durch Transformationen ineinander überführen lassen. Tatsächlich können sich aus der Wahl der Zustandsvariablen aber erhebliche Unterschiede für die computerunterstützte *Umsetzung* der Lösungsverfahren ergeben. Wie bei den Verfahren zur Einbindung von Constraints in Kapitel 3 zu sehen war, ist diese Auswahl sogar eng mit dem jeweiligen Lösungsansatz verknüpft. Während z.B. die Reduktionsmethode auf dem minimalen Satz unabhängiger Koordinaten operiert, können bei der LFM auch voneinander abhängige Koordinaten eingesetzt werden. In der Technischen Simulation ist dieses Problem wohlbekannt (siehe z.B. [GB94]). Im Bereich der Computeranimation wird dieser Aspekt dagegen kaum behandelt.

Für Verfahren wie die LFM, die prinzipiell mit beliebigen Zustandskoordinaten durchführbar sind, kommen drei Typen von Koordinatensätzen in Betracht:

- Relative Koordinaten
  Bei relativen Koordinaten wird der Zustand eines Körpers bezüglich eines anderen Körpers festgelegt. Der Auslenkungswinkel eines Scharniergelenks zwischen zwei Körpern ist z.B. eine solche Koordinate. Wenn die Konfiguration eines der beiden Körper bekannt ist, kann die Konfiguration des zweiten eindeutig aus diesem Winkel abgeleitet werden.

- Absolute Koordinaten
  Absolute Koordinaten stellen eine von anderen Körpern unabhängige Körperbeschreibung dar. Zu ihnen gehören z.B. solche intuitiven Zustandsgrößen wie die Position des Körperschwerpunktes bezüglich eines Weltkoordinatensystems.

- Redundante Koordinaten
  Bei miteinander gekoppelten Körpern liefern im Grunde schon absolute Koordinaten eine redundante Systembeschreibung, da die Zahl der Koordinaten größer als die der Freiheitsgrade ist. Unter Umständen kann es aber sinnvoll sein, selbst für die Beschreibung einzelner Körper mehr Zustandskoordinaten als notwendig zu verwenden. Ein solcher Ansatz, der sich für eine effiziente Simulationsdurchführung ausnutzen läßt, wird z.B. in [GB94] propagiert [5].

Natürlich sind auch Mischformen dieser Koordinatensätze möglich. In [AH93] werden z.B. relative Koordinaten für Ketten gekoppelter Körper verwandt, die aber durch „Abschneiden“ der Ketten in

[5] Die in [GB94] beschriebenen sogenannten *natürlichen* Koordinaten führen dabei zu einer Drehmatrix, die lediglich linear von den Zustandsvariablen abhängt, während Quaternionen zu quadratischen und Euler-Winkel zu transzendentalen Abhängigkeiten führen. Auch die Auswertung der Jacobi-Matrix und der Constraint-Gleichungen läßt sich bei diesem Ansatz beschleunigen.

regelmäßigen Intervallabständen durch zusätzliche absolute Koordinaten ergänzt werden.

Für die Umsetzung eines Simulationsverfahrens auf der Grundlage der LFM ergeben sich aus der Wahl dieser Koordinatentypen große Unterschiede. Tatsächlich sind die relativen Koordinaten für eine solche Anwendung kaum geeignet. Nach jeder Änderung der Szenentopologie muß bei dieser Beschreibungsform ein neuer Koordinatensatz bestimmt werden, der zudem für bestimmte Systemkonfigurationen inadäquat werden kann. Bei nichtreduzierten Systemgleichungen führen sie daher im Vergleich zu absoluten Koordinaten bis auf wenige Ausnahmesituationen zu unnötigen Verkomplizierungen ([RS88], S. 401). Noch schwerwiegender ist die durch relative Koordinaten verursachte Aufhebung der Unabhängigkeit von Körpern und Constraints, die in Abschnitt 4.2 als ein wesentlicher Vorteil der LFM ausgemacht wurde.

Für absolute und redundante Koordinaten ergeben sich diese Probleme nicht. Der Einsatz redundanter Koordinaten verspricht dabei einen möglichen Effizienzzuwachs für das Simulationsverfahren. Absolute Koordinaten haben dagegen den Vorteil, sehr intuitiven Größen zu entsprechen. Aufgrund ihrer weiten Verbreitung wurden zudem viele effiziente Verfahren entwickelt, die auf diese Beschreibungsform zugeschnitten sind.

5.3.1.2 Formalismen für die Aufstellung der Bewegungsgleichungen

Für die Aufstellung der physikalischen Bewegungsgleichungen lassen sich verschiedene mathematische Formalismen einsetzen. Der Euler-Formalismus bezeichnet z.B. die klassische Form der Bewegungsgleichungen ($\mathbf{F} = \mathbf{M}\,\mathbf{a}$), die auch in Kapitel 3 gewählt wurde. Der (nicht mit der Lagrange-Faktoren-Methode zu verwechselnde) Lagrange-Formalismus stellt dagegen eine physikalisch gesehen völlig äquivalente Formulierung mit den Größen der kinetischen und potentiellen Energie als Ausgangspunkt dar. Dieser Formalismus, der z.B. in [Gol89] beschrieben wird, kann vor allem bei der Verwendung generalisierter Koordinaten in einer sehr schematischen Weise angewandt werden und stellt daher ein häufig gewähltes Standardverfahren dar. Der Lagrange-Formalismus erster Art beinhaltet dabei die explizite Berechnung der Constraint-Kräfte, während der Lagrange-Formalismus zweiter Art, der nur die Einbindung holonomer Constraints erlaubt, diese aus den Bewegungsgleichungen eliminiert. Große Ähnlichkeit zu diesem Ansatz haben zudem die Formulierungen auf der Grundlage des d'Alembert-Prinzips und des Jourdain-Prinzips, die z.B. in [RS88] erläutert werden[6].

Für die Anwendung der LFM ist der Euler-Formalismus, der Lagrange-Formalismus erster Art und der Jourdain-Formalismus gleichermaßen für die Aufstellung der Bewegungsgleichungen geeignet. Der Lagrange-Formalismus zweiter Art würde aufgrund der ausschließlichen Behandlung von holonomen Constraints zu einer wesentlichen Einschränkung führen und auch das d'Alembert-Prinzip ist nur für bestimmte Klassen nichtholonomer Constraints anwendbar. Ansonsten sind die Formalismen aber nicht nur physikalisch äquivalent, sondern führen auch auf die selben Bewegungsgleichungen ([Fea87]).

5.3.2 Aufspaltung der LFM

Der Verfahrensablauf der LFM wurde in Abschnitt 3.5 ausführlich beschrieben. Dieser Ablauf umfaßt die Lösung ganz unterschiedlicher mathematischer Probleme wie z.B. die Lösung linearer Gleichungssysteme und die Integration gewöhnlicher Differentialgleichungen. Für eine Umsetzung der

[6] Diese Formalismen vermeiden zudem die Systembeschreibung mit Hilfe des kinetischen Energieterms, dessen Verwendung beim Lagrange-Formalismus zu überflüssigen Berechnungen führt ([RS88], S. 171).

LFM wäre es daher aus Gründen der Modularität wünschenswert, eine voneinander getrennte Bearbeitung dieser Probleme zu ermöglichen. Ein Konzept für eine solche Aufspaltung soll nun vorgestellt werden.

Die bei der Durchführung der LFM anfallenden Verarbeitungsschritte lassen sich in drei Teilaufgaben aufspalten, die weitgehend unabhängig voneinander bearbeitet werden können:

- Bestimmung der Lagrange-Faktoren
- Anwendung von Stabilisierungsverfahren
- Lösung der gewöhnlichen Differentialgleichungen

Durch die getrennte und voneinander unabhängige Realisierung von Lösungsverfahren für diese Teilaufgaben läßt sich eine sehr robuste und leicht erweiterbare Simulationsumgebung für mechanische Systeme realisieren. In Abschnitt 6.3 wird mit dem Animationssystem EMPHAS eine konkrete Realisierung dieses Konzeptes vorgestellt, die sich durch einen hohen Grad an Modularität auszeichnet – für jedes der drei Probleme wurde eine ganze Reihe von Lösungsverfahren implementiert, die sich jederzeit gegeneinander austauschen lassen. Diese Aufspaltung stellt daher einen wichtigen Beitrag für die Anwendung der LFM im Rahmen allgemeiner Animationssysteme dar.

Im folgenden Abschnitt soll nun dargelegt werden, wie sich die Bestimmung der Lagrange-Faktoren auf der Grundlage des Baraff-Algorithmus, der in Abschnitt 3.5.5 besprochen wurde, durchführen läßt. Beispiele für konkrete Verfahren zur Constraint-Stabilisierung und zur Lösung von gewöhnlichen Differentialgleichungen werden in den Abschnitten 6.3.4.1 und 6.3.4.2 bei der Besprechung des Animationssystems EMPHAS vorgestellt.

5.3.3 Ein allgemeines Verfahren zur Bestimmung der Lagrange-Faktoren

Die Bestimmung der Lagrange-Faktoren stellt die weitaus schwierigste Teilaufgabe bei der Umsetzung der LFM dar. Der hierfür notwendige Rechenaufwand ist in den meisten Fällen wesentlich größer als z.B. für die numerische Lösung der Differentialgleichungen, so daß der Einsatz hochgradig effizienter Verfahren erforderlich ist. Wie noch genauer erläutert wird, müssen zudem besondere Vorkehrungen getroffen werden, um die Robustheit der Lösungsverfahren zu garantieren.

Ein einfach umsetzbares und gleichzeitig sehr effizientes Verfahren zur Bestimmung der Lagrange-Faktoren stellt das in Abschnitt 3.5.5 erläuterte Baraff-Verfahren ([Bar96]) dar. Dieses Verfahren soll dem hier vorgestellten Konzept zu Grunde gelegt werden. Um einen möglichst allgemeinen Einsatz zu erzielen, muß dabei eine Anwendung auf Systeme mit geschlossenen Schleifen und eine robuste Behandlung singulärer Konstellationen verwirklicht werden, wie nun genauer ausgeführt werden soll.

Systeme mit geschlossenen Schleifen können nicht durch den Grundalgorithmus des Baraff-Verfahrens zur Behandlung primärer Constraints berücksichtigt werden, da die Matrix **H** ihre Baumstruktur verliert (vergl. Abschnitt 3.5.5). In [Bar96] wird aber eine einfache Lösung dieses Problems angegeben: Einzelne primäre Constraints, die Teil einer geschlossenen Schleife sind, müssen als sekundäre Constraints deklariert werden. Auf diese Weise kann jede einfache Schleife durch die Wegnahme von nur einem primären Constraint geöffnet werden. Die umdeklarierten Constraints können dabei genauso wie die anderen sekundären Constraints mit Hilfe des in Abschnitt 3.5.5 angesprochenen erweiterten Algorithmus behandelt werden. Unterschiede bezüglich der Systemtopologie gibt es bei diesem Verfahren nicht. Selbst komplett aus sekundären Constraints bestehende Schleifen (die sich nicht auf die oben beschriebene Weise öffnen lassen) erfordern keine Sonderbehandlung.

Diese schematische Behandlung von Systemen mit geschlossenen Schleifen beruht auf einer allgemeinen Eigenschaft der LFM: der Verwendung nichtreduzierter Koordinaten. Selbst ein System, das ausschließlich holonome Constraints umfaßt, kann nicht direkt mit Hilfe reduzierter Koordinaten beschrieben werden, wenn geschlossene Schleifen vorliegen. Diese Systemtopologie, die bei der Animationserstellung sehr häufig und oft erst während des Simulationsverlaufes auftritt, erfordert daher bei der Reduktionsmethode den Einsatz zusätzlicher Verfahren (siehe hierzu z.B. [Lei92]). Bezüglich der einfachen Umsetzung des Simulationsverfahrens stellt dieser Aspekt daher einen großen Vorteil der LFM dar.

Wie für andere Lösungsverfahren auch können für das Baraff-Verfahren außerdem Systemkonfigurationen auftreten, die eine Fortführung der Simulation unmöglich machen. Dieses Problem wird in [Bar96] nicht erörtert. Mathematischer Ausdruck einer solchen Konfiguration ist eine singuläre Matrix, die im Laufe des Verfahrens invertiert werden muß. Diese Konfiguration kann dann entstehen, wenn die Ableitung der Constraint-Funktion verschwindet, ohne daß der Constraint erfüllt ist, also auch bei Systemzuständen, die physikalisch gesehen wohldefiniert sind.

Koordinatensingularitäten lassen sich vermeiden, wenn die festgelegten Constraints annähernd erfüllt sind, da in diesem Fall die Ableitungen bei günstig definierten Constraint-Funktionen (siehe Abschnitt 6.2.3) nicht verschwinden können. Diese Bedingung darf aber bei der interaktiven Erstellung physikalisch basierter Animationen nicht vorausgesetzt werden – bei der Festlegung eines Constraints kann das System zunächst beliebig weit von seiner Erfüllung entfernt sein.

Eine Möglichkeit zur Behandlung dieses Problems besteht in der Anwendung kleiner Verrückungen, mit denen der Systemzustand minimal verändert wird, um auf diese Weise die Koordinatensingularität aufzulösen. Diese Methode läßt sich leicht umsetzen und kann zu einer wesentlichen Erhöhung der Verfahrensrobustheit führen. Um systematische Verfälschungen des Systemzustandes zu vermeiden, können die Verrückungen dabei mit Hilfe eines (Pseudo-)Zufallsprozesses erzeugt werden.

Zusammen mit diesen Ergänzungen ermöglicht das Baraff-Verfahren somit nicht nur eine sehr zeiteffiziente, sondern auch eine robuste und für beliebige mechanische Systeme anwendbare Bestimmung der Lagrange-Faktoren.

5.4 Konzepte zur Anwendung von Steuertechniken

Das oben beschriebene Konzept zur Umsetzung der LFM ermöglicht die Einbindung beliebiger beschleunigungslinearer Constraints. Für die konkrete *Anwendung* von Constraints und anderen Steuertechniken im Rahmen eines allgemeinen Animationssystems ergeben sich aber wichtige zusätzliche Aspekte, wie in diesem Abschnitt genauer ausgeführt werden soll.

5.4.1 Kombinationen von Steuertechniken auf der Grundlage der LFM

Im Abschnitt 2.3 wurden verschiedene Ansätze zur Steuerung von Bewegungen vorgestellt. Ein äußerst wichtiger Aspekt vor dem Hintergrund allgemeiner Animationssysteme ist die Möglichkeit zur *Kombination* dieser Techniken. Die Bewegungssteuerung wäre stark eingeschränkt, wenn die verschiedenen Ansätze nur jeweils ausschließlich angewandt und nicht miteinander kombiniert werden könnten. Durch die verschiedene Art der Einflußnahme auf die Körperbewegungen (z.B. über Kräfte, kinematische Vorgaben oder ereignisgesteuerte Aktionen) ist diese Aufgabe allerdings mit komplizierten Problemen verknüpft, die auch mit den Eigenschaften des Simulationsverfahrens zusammenhängen.

Drei grundlegende Steuerverfahren, die sich für den Einsatz in einem allgemeinen Animationssystem besonders gut eignen (vergl. Abschnitt 2.3.3) und auch in dem Animationssystem EMPHAS realisiert wurden, stellen Controller, Constraints und ereignisbasierte Prozeduren dar. Die Möglichkeiten zur gemeinsamen Anwendung dieser Techniken auf der Grundlage der LFM sollen nun im einzelnen besprochen werden. Die aufgezeigten Zusammenhänge mit den Eigenschaften der LFM stellen dabei einen neuartigen Beitrag für dieses wichtige Problem dar.

Die Kombination von Controllern untereinander und Constraints untereinander stellt zunächst kein grundsätzliches Problem dar. Controller führen zu zusätzlichen Kräften mit einem bestimmten funktionalen, z.B. zustandsabhängigen, Verhalten. Diese Kräfte können problemlos zu einer Gesamtkraft aufsummiert werden. Auch Constraints lassen sich in der Regel problemlos miteinander kombinieren. Bei der Festlegung mehrerer Constraints können allerdings überbestimmte Systeme entstehen: Die Constraints könnten sich widersprechen und wären nicht gleichzeitig erfüllbar. Die Behandlung derartiger Konstellationen wird in Abschnitt 5.4.2 besprochen. Wie dort erläutert wird, können sie eine wichtige Rolle für die interaktive Animationserstellung spielen.

Controller lassen sich auch gemeinsam mit Constraints anwenden. Die Controller-Kräfte werden bei jedem Zeitschritt unabhängig von den Constraints ermittelt und dem dynamischen System hinzugefügt. Sie werden daher wie alle anderen dynamischen Eigenschaften bei der Behandlung der Constraints berücksichtigt und führen nicht zu deren Verletzung.

Durch die Festlegung von Constraints kann allerdings die *Auswirkung* der Controller auf die von ihnen beeinflußten Körper verändert werden. Dieser Umstand muß vor allem bei rein zeitabhängigen Controllern, d.h. beim *Open Loop*-Verfahren (siehe Abschnitt 2.3.2.2) beachtet werden, da in diesem Fall keine Rückkopplung über den Systemzustand stattfindet. Vorteilhaft wirkt sich dabei eine spezielle Eigenschaft der LFM aus: der stetige, physikalisch basierte Prozess bei der *Erfüllung* der Constraints (Punkt 5 in Abschnitt 4.2). Dieser Prozess findet daher in einer leicht nachvollziehbaren Weise statt und führt zu Auswirkungen auf die Controller, wie sie sich auch bei anderen dynamischen Abläufen ergeben würden. Bei Verfahren wie der Reduktionsmethode würde dieser Vorgang dagegen instantan ablaufen, was zu einem unvorhersehbaren Verhalten der hiervon beeinträchtigten Controller führen kann.

Eine ganz analoge Beeinflussung tritt bei der gemeinsamen Anwendung von Controllern mit ereignisbasierten Prozeduren auf. Die Berücksichtigung von Controllern durch die Prozeduren stellt aus den gleichen Gründen wie bei der gemeinsamen Anwendung mit Constraints kein Problem dar. Problematischer ist aber der umgekehrte Einfluß. Da die Wirkung derartiger Prozeduren von bestimmten Ereignissen abhängt, die i.a. zum Simulationsbeginn unbekannt sind, ist eine solche Kombination nur für *Closed Loop*-Controller sinnvoll. Falls die Wirkung der Prozeduren Festlegungen von Constraints einschließen, wirken sich außerdem auch in diesem Fall die oben angeführten Unterschiede der Lösungsverfahren bezüglich des Prozesses der Constraint-Erfüllung aus.

Abschließend soll nun die Kombination von Constraints und ereignisbasierten Prozeduren besprochen werden. Da Constraints sogar neben Controllern intern von Prozeduren eingesetzt werden können, um das gewünschte Bewegungsverhalten zu erzielen, scheint diese Möglichkeit sehr nahe zu liegen. Zumindest für die per Constraints festgelegten *Ziele*, d.h. die angestrebten Zustände, bei denen die Constraints erfüllt sind, läßt sich dieses Problem auf die oben besprochene Kombination von Constraints und Controllern reduzieren. Ein schwieriges Problem stellt aber i.a. die prozedurale Auswertung des *Erfüllungsprozess* der Constraints dar. Bei Lösungsverfahren wie der Reduktionsmethode würde der Prozess in einer solch unvorhersehbaren und verfahrensspezifischen Weise ablaufen, daß er kaum sinnvoll auswertbar ist. Die Wirkungen sämtlicher Prozeduren, die eine physikalisch basierte Bewegungsgenerierung voraussetzen, müßten daher für diesen Vorgang ausgesetzt werden, um nicht absurde Ergebnisse zu liefern. Diese Einschränkung kann zu einer erheblichen Be-

einträchtigung der visuellen Plausibilität führen. Auf der Grundlage der LFM sind dagegen nicht nur die Ziele der Constraints mit den prozeduralen Verfahren vereinbar, sondern auch ihr Erfüllungsprozess. Dieser Vorgang, der je nach Systemzustand und festgelegtem Constraint eine unterschiedlich lange Zeitdauer in Anspruch nimmt, ist also genauso prozedural auswertbar wie jeder andere Simulationsvorgang. Diese Anwendung der LFM wurde auch in [Wag00] besprochen.

In der Technischen Simulation wird dieser Aspekt nicht berücksichtigt, da Systeme mit nahezu erfüllten Constraints vorausgesetzt werden. Wie schon häufiger in dieser Arbeit betont wurde, ist diese Voraussetzung aber bei der interaktiven Animationserstellung nicht gegeben. Die Anwendung von Prozeduren auf Systeme mit nichterfüllten Constraints stellt daher einen wichtigen und bislang in der Literatur nicht erwähnten Beitrag für den Einsatz der LFM im Rahmen eines allgemeinen Animationssystems dar. Bei der Vorstellung des Animationssystems EMPHAS wird dieser Aspekt in den Abschnitten 6.2.4.2 und 6.4.2 aufgegriffen.

5.4.2 Konzepte zur Behandlung überbestimmter Systeme

Überbestimmte Systeme entstehen bei der gleichzeitigen Festlegung von sich widersprechenden Constraints. Wie Koordinatensingularitäten (vergl. Abschnitt 5.3.3) führen sie zu einer singulären Lösungsmatrix. Im Unterschied zu diesen haben sie aber eine echte physikalische Unbestimmtheit als Ursache und können daher auch nicht durch eine kleine Modifikation des Systemzustandes aufgelöst werden. In diesem Abschnitt soll die Bedeutung dieser Systeme im Rahmen der Animationserstellung untersucht und Möglichkeiten zu ihrer Behandlung aufgezeigt werden.

Die Behandlung überbestimmter Systeme stellt nicht nur im Bereich der Computeranimation ein häufig untersuchtes Problem dar. Eine allgemeine Untersuchung vor dem Hintergrund allgemeiner Animationssysteme, wie sie im folgenden durchgeführt wird, ist in der computergraphischen Literatur aber nicht aufzufinden.

5.4.2.1 Bedeutung im Rahmen allgemeiner Animationssysteme

Bei einem genau festgelegten Bewegungsverhalten, z.B. bei der Erstellung einer zuvor festgelegten Animation, können überbestimmte Systeme durch den korrekten Einsatz von Constraints schon in der Planung der Animationserstellung vermieden werden. Bei der interaktiven Erstellung im Rahmen eines allgemeinen, interaktiven Animationssystems ist dies aber nicht immer möglich. Überbestimmte Systeme können hier sehr schnell entstehen, ohne daß der Animateur dies gewollt hätte. Wird z.B. die Geschwindigkeit eines Körpers auf einen konstanten Wert festgelegt und gleichzeitig ein *PointToPoint*-Constraint zwischen diesem Körper und einem festen Punkt vorgegeben, ist eine solche überbestimmte Situation entstanden, bei der sich der Körper zwischen einem der beiden Constraints „entscheiden“ müßte. Ein wichtiger Aspekt ist dabei auch das Vorwissen des Animateurs, das sich wie in Abschnitt 2.2.3 erwähnt von dem Betreiber einer technischen Simulation unterscheidet. Das Auftreten überbestimmter Systeme darf daher nicht ignoriert werden.

Aus diesen Ausführungen kann eine wesentliche Anforderung an allgemeine Animationssysteme abgelesen werden:

- Robuste Durchführung der Simulation
  Das Auftreten überbestimmter Systeme darf nicht zu einem unkontrollierten Verhalten des Simulationsverfahrens führen.

Eine mögliche Strategie zur Behandlung solcher Fälle wäre ein mit einem entsprechenden Hinweis an den Animateur versehener Abbruch der Simulation im Falle einer Überbestimmtheit. Dieses

Vorgehen erscheint aber für interaktive Animationssysteme nicht adäquat. Stattdessen soll dafür argumentiert werden, daß die Simulation überbestimmter Systeme eine durchaus *sinnvolle* Aufgabe bei der interaktiven Animationserstellung darstellen kann. Hierfür lassen sich zwei wichtige Gründe anführen:

- Unterstützung des Benutzers bei der Fehlersuche
  Durch die geeignete Simulation des überbestimmten Systems kann dem Benutzer durch die graphische Rückkopplung vermittelt werden, *welche* Bewegungsvorgaben miteinander unverträglich sind, d.h. welchen *Grund* die Überbestimmheit hat, und somit eine Unterstützung bei der Beseitigung dieses Zustandes liefern. Der Begriff der Überbestimmheit muß dem Animateur dabei nicht einmal geläufig sein.

- Einsatz als Modellierungs- oder Steuertechnik
  Bei einer geeigneten Behandlung können zu überbestimmten Systemen führende Vorgaben *bewußt* als Hilfsmittel bei der Modellierung und Bewegungssteuerung eingesetzt werden.

Um diese Ziele, für die im folgenden Abschnitt 5.4.2.2 ein Beispiel gegeben wird, erreichen zu können, müssen Techniken zur geeigneten Behandlung überbestimmter Systeme realisiert werden. Dabei lassen sich drei grundsätzliche Herangehensweisen unterscheiden, die nun genauer erläutert und anschließend bewertet werden sollen: Das Auffinden einer optimalen Lösung, die selektive Einschränkung von Constraints und die Einführung zusätzlicher Freiheitsgrade.

5.4.2.2 Bestimmung einer Näherungslösung

Eine Möglichkeit zur Behandlung überbestimmter Systeme besteht in der Berechnung einer „näherungsweisen“ Lösung, die bestimmten Kriterien gerecht wird und damit einer „optimalen“ Lösung entspricht. Dieser Ansatz läßt sich an dem in Abbildung 5.2 dargestellten Beispiel erläutern. Die beiden *PointToPoint*-Constraints, die die Mittelpunkte der beiden Zylinder-Grundflächen mit ortsfesten Punkten verknüpfen sollen, können nicht gleichzeitig erfüllt werden.

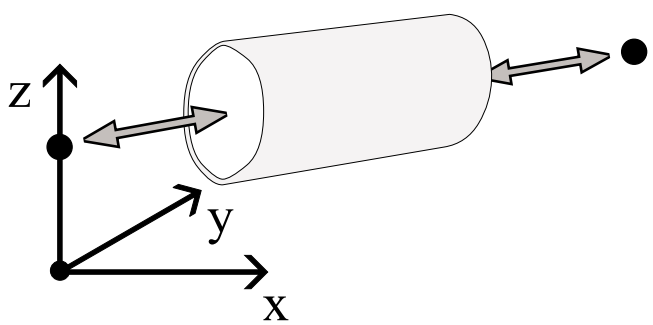

Abbildung 5.2: Ein überbestimmtes System. Die beiden *PointToPoint*-Constraints können nicht gleichzeitig erfüllt werden.

Eine optimale Lösung könnte bei diesem Beispiel z.B. in dem Verharren des Zylinders zwischen den beiden Raumpunkten bestehen, so daß die Constraints zumindest so gut wie möglich erfüllt werden. Zu diesem Zweck lassen sich z.B. numerische Optimierungsverfahren einsetzen. Das Verhalten des Zylinders kann dann auf zweierlei Art vom Animateur ausgewertet werden. Zum einen wird ihm hierdurch auf intuitive Weise die Unverträglichkeit der von ihm festgelegten Constraints vermittelt. Durch die Deaktivierung einer der beiden Constraints könnte er nun z.B. die Überbestimmtheit auflösen, falls er dies wünscht. Auf der anderen Seite könnte die hier entstandene Konfiguration aber auch dem *Ziel* seiner Festlegungen entsprechen. Um den Zylinder in dieser Weise zwischen den beiden Punkten zu positionieren, stellt dieses Vorgehen sogar eine der *einfachsten* Möglichkeiten dar. Feste Abstands-Constraints wären hierfür kaum geeignet, da

die exakten Abstände zu den jeweiligen Raumpunkten bekannt sein müßten. Schon anhand dieses einfachen Beispiels ist somit erkennbar, wie wichtig die Behandlung überbestimmter Systeme für die interaktive Animationserstellung ist.

5.4.2.3 Selektive Einschränkung von Constraints

Ein anderer grundsätzlicher Ansatz zur Behandlung überbestimmter Systeme ist die selektive Einschränkung von Constraints. Die einfachste Variante dieser Methode besteht in der Deaktivierung derjenigen Constraints, die für die Überbestimmtheit verantwortlich sind. Auf diese Weise läßt sich jedes überbestimmte in ein wohldefiniertes und damit standardmäßig lösbares System überführen. Oftmals läßt sich dieses Ziel aber auch durch eine Beschränkung der Constraints selber erreichen. Eine Möglichkeiten hierfür stellt die Unterdrückung einzelner Constraint-Komponenten und damit die Verringerung der Dimensionalität des Constraints dar. In Abschnitt 6.6.2 wird ein konkretes Beispiel für eine solche Herangehensweise vorgestellt.

5.4.2.4 Einführung zusätzlicher Freiheitsgrade

Anstelle der Einschränkung von Constraints kann auch der umgekehrte Weg eingeschlagen werden – die Einführung zusätzlicher Freiheitsgrade. In [P+00] wird z.B. ein Konzept zur interaktiven Festlegung von Constraints beschrieben, bei dem Überbestimmtheiten durch die Einführung zusätzlicher Variabilitäten aufgelöst werden können. Diese können sich z.B. auf die Masseneigenschaften eines Körpers oder auf die Lage von Kontaktflächennormalen beziehen[7]. Die Auswahl und Justierung dieser Variabilitäten stellt aber ein schwieriges Problem dar, das sich nur zum Teil automatisieren läßt. Zum einen sind die zusätzlichen Freiheitsgrade künstliche (d.h. nicht physikalisch abgeleitete) Modifikationen des physikalisch basierten Modells und können daher zu unplausibel wirkenden und damit unerwünschten Bewegungen führen. Für die Auflösung überbestimmter Systeme muß ihre Anwendung außerdem sehr gezielt erfolgen, d.h. sie muß auf den jeweiligen Fall angepaßt sein.

5.4.2.5 Bewertung der Ansätze

Es stellt sich nun die Frage, wie diese Ansätze, die beim Auftreten überbestimmter Systeme nicht zu einem Simulationsabbruch führen, vor dem Hintergrund der eingangs genannten Ziele zu bewerten sind. Eine solche Bewertung soll nun vorgenommen werden.

Die Bestimmung einer Näherungslösung bietet den Vorteil, prinzipiell für jedes überbestimmte System auf automatische Weise eine sinnvolle Lösung zu liefern. Für eine robuste Durchführung der Simulation ist diese Eigenschaft sehr wichtig. Der bewußte Einsatz überbestimmter Systeme mit Hilfe dieses Ansatzes ist allerdings durch zwei Probleme eingeschränkt. Zum einen kann die Bestimmung einer optimalen Lösung sehr zeitaufwendig sein. Die hierfür einsetzbaren numerischen Verfahren sind in der Regel weniger effizient als die standardmäßigen Methoden zur Bewegungssimulation. Noch gravierender ist die Einschränkung der *Vorhersagbarkeit*, die als Unterpunkt der Intuitivität zu den Anforderungen an allgemeine Animationssysteme gehört. Die Einschätzung, welches Bewegungsverhalten bei diesem Ansatz für die überbestimmten Systeme resultiert, ist sehr schwierig und hat nicht immer eine solch einfache und intuitiv verständliche Form wie in dem oben angeführten Beispiel. Eine äußerst nützliche Anwendung dieser Methode stellt aber die Unterstützung des Animateurs bei der Aufhebung der Überbestimmung dar, wie an dem in Abbildung 5.2 gezeigten Beispiel verdeutlicht wurde.

[7] Derartige Variabilitäten wurden auch in [B+96] angewandt. Dort dienten sie aber als zusätzliche Einflußmöglichkeit auf die Szenendynamik und als Mittel zur Vermeidung unnatürlich wirkender Bewegungsabläufe.

Die selektive Einschränkung von Constraints erfordert dagegen einen expliziten Eingriff des Animateurs. Je nach System und Zielstellung kann die angemessene Durchführung dieser Aufgabe sehr schwierig sein und ein gewisses Vorwissen des Animateurs erfordern. Diese Methode läßt auch das Problem der versehentlichen Erzeugung eines überbestimmten Zustandes unberücksichtigt – lediglich eine nachträgliche Korrektur ist mit Hilfe dieser Methode möglich. Auf der anderen Seite führt die geeignete Einschränkung von Constraints zu nichtsingulären Bewegungsgleichungen, die sich standardmäßig und ohne Beeinträchtigung der zeitlichen Effizienz lösen lassen. Außerdem ist die Vorhersagbarkeit und damit die Nachvollziehbarkeit des hierdurch erzielten Bewegungsverhaltens wesentlich größer. Die Überbestimmtheit wird nicht durch einen komplexen mathematischen Vorgang aufgelöst, sondern durch die gezielte Einschränkung von Constraint-Vorgaben. Diese Technik stellt daher eine sehr flexible Möglichkeit für den bewußten Einsatz überbestimmter Systeme dar.

Eine ganz ähnliche Einschätzung ergibt sich für die Einführung zusätzlicher Freiheitsgrade. Auch dieser Ansatz stellt eine Möglichkeit dar, ein überbestimmtes System in ein wohldefiniertes System zu transformieren. Die Umsetzung dieses Ansatzes ist aus den in Abschnitt 5.4.2.4 angeführten Gründen aber wesentlich schwieriger.

	Abbruch der Simulation	Näherungs-lösung	Constraint-Einschränkung	Zusätzliche Freiheitsgrade
Robuste Simulation	ja	ja	nein	nein
Korrektur-Unterstützung	nein	ja	nein	nein
Bewußter Einsatz	nein	(ja)	ja	ja
Realisierungs-Aufwand	niedrig	mittel	mittel	hoch

Tabelle 5.2: Ansätze zur Behandlung überbestimmter Systeme.

Tabelle 5.2 umfaßt eine Gegenüberstellung dieser Aspekte. Aus dieser Tabelle geht auch hervor, daß die hier besprochenen Ansätze keine sich gegenseitig ausschließenden Möglichkeiten darstellen. Ein allgemeines, physikalisch basiertes Animationssystem sollte daher möglichst sowohl Techniken zur Bestimmung einer Näherungslösung als auch zur gezielten Constraint-Einschränkung oder zur Einführung zusätzlicher Freiheitsgrade bereitstellen.

5.4.3 Konzepte zur Vermeidung von Bewegungsartefakten

Bei der Festlegung von Constraints kann es zu einem Problem kommen, das bereits in Abschnitt 4.2 angesprochen wurde. Die Stabilisierungskräfte, die bei der LFM nach der Festlegung eines Constraints berechnet und angewandt werden, können zu Bewegungsartefakten führen, die nicht vom Animateur erwünscht sind. Die eigentliche Bewegung der mit dem Constraint verknüpften Körper wäre dabei auch bei anderen Lösungsverfahren unvermeidlich (sofern man auch einen instanten Erfüllungsprozess als Bewegung ansieht): Um einen *PointToPoint*-Constraint zwischen einem Körper und einem raumfesten Ortspunkt erfüllen zu können, muß die Position des Körpers z.B. in aller Regel modifiziert werden. Problematisch sind aber zusätzliche Bewegungen, die nur in impliziter Weise aus dem Constraint resultieren, vor allem wenn sie auch nach dem Prozess der Constraint-Erfüllung noch wahrnehmbar sind, wenn der festgelegte Constraint bereits erfüllt ist. Bei dem Beispiel des *PointToPoint*-Constraints könnte es z.B. zu einer Rotation des Körpers um den ausgewählten Raumpunkt kommen, ohne den Constraint dabei zu verletzen.

In der Literatur waren zu diesem Problem keine Ausführungen auffindbar. Wie die Erfahrungen mit

dem Animationssystem EMPHAS, das in Kapitel 6 vorgestellt wird, gezeigt haben, treten solche Artefakte des Stabilisierungsverfahrens aber durchaus häufig auf und können insbesondere den Modellierungsprozess erheblich erschweren. Mit der Ausnahme des Prozesses der Constraint-Erfüllung (vergl. Abschnitt 4.2) sind bei der Durchführung von Modellierungsaufgaben in der Regel überhaupt keine Bewegungen erwünscht. Dieses Problem darf daher nicht unberücksichtigt bleiben, zumal es sich aus den speziellen Eigenschaften der LFM ergibt.

Zur Lösung dieses Problems lassen sich drei grundsätzliche Herangehensweisen unterscheiden, die nun erläutert werden sollen. Diese Konzepte stellen einen neuen Beitrag für die Anwendung der LFM im Rahmen allgemeiner Animationssysteme dar.

Explizite Zustandsänderung. Mit einem direkten Eingriff in das System lassen sich z.B. die Komponenten der Körpergeschwindigkeit nach dem erfolgten Prozess der Constraint-Erfüllung auf Null setzen. Wenn die Bewegung mehrerer über Constraints miteinander verknüpfter Körper beeinflußt werden soll, ist dieses Vorgehen aber kaum anwendbar, da sich die derart abgestoppten Körper in der Regel aufgrund der durch den Constraint bedingten Wechselwirkungen mit den anderen Körpern sofort wieder in Bewegung setzen. Die bis zum Interaktionszeitpunkt stattgefundene Translation und Rotation der Körper aufgrund der Stabilisierungsartefakte können bei dieser Herangehensweise außerdem überhaupt nicht verhindert werden.

Anwendung von kompensierenden Kräften. Eine andere Möglichkeit besteht in der Anwendung von zusätzlichen Kräften, die den berechneten Constraint-Kräften in der Weise entgegenwirken, daß die unerwünschten Bewegungsartefakte unterdrückt werden, ohne dabei die Wirkung der jeweiligen Constraints zu verfälschen. Die automatische Bestimmung solcher Kräfte ist aber äußerst schwierig, da sich der Anteil der Constraint-Kräfte, der für die Erfüllung der Constraints unverzichtbar ist, und der ungewollte zusätzliche Anteil praktisch nicht voneinander trennen läßt[8]. Durchaus möglich ist aber z.B. eine allgemeine Abbremsung sämtlicher Bewegungen, wofür in Abschnitt 6.6.3 ein konkretes Beispiel vorgestellt wird.

Selektive Einschränkung von Freiheitsgraden. Eine sehr flexible Möglichkeit zur Unterdrückung von Bewegungsartefakten stellt die selektive Einschränkung von Freiheitsgraden dar. Viele Constraints lassen sich z.B. auch durch reine Translationen der zugehörigen Körper erfüllen. Indem die Transformationen der Körper bei der Erfüllung des Constraints auf reine Translationsbewegungen beschränkt werden, können dann unerwünschte Rotationsbewegungen gänzlich vermieden werden. Im Rahmen eines allgemeinen Animationssystems muß allerdings dafür Sorge getragen werden, daß sich die auf diesem Ansatz basierenden Techniken in einfacher Weise anwenden lassen, ohne ein tiefgreifendes technisches Verständnis des Animateurs zu erfordern.

Eine Einschränkung von Freiheitsgraden läßt sich auch durch die Festlegung zusätzlicher Constraints erzielen. Die automatische Erstellung von Constraints, deren Einfluß genau in der Unterdrückung der unerwünschten Bewegungen besteht, stellt aber ein ebenso großes Problem wie die oben angesprochene Berechnung kompensierender Kräfte dar.

Für das Problem der unerwünschten Bewegungsartefakte lassen sich somit mehrere Lösungskonzepte angeben, die wie schon bei der Behandlung überbestimmter Systeme nicht einer ausschließenden, sondern in einer ergänzenden Weise angewandt werden sollten.

In diesem Kapitel wurden Konzepte für die Anwendung der LFM im Rahmen allgemeiner Animationssysteme vorgestellt. Die konkrete Realisierung eines solchen Systems, für das Methoden auf der Grundlage dieser Konzepte entworfen wurden, soll im nun folgenden Kapitel beschrieben werden.

[8] Dieser ungewollte Anteil stellt nur einen *Teil* der Stabilisierungskräfte dar, deren Unterdrückung zu einer deutlich sichtbaren Verletzung der Constraints führen kann (vergl. Abschnitt 3.5.4).

Kapitel 6

EMPHAS – ein allgemeines, physikalisch basiertes Animationssystem

6.1 Übersicht über das System

In diesem Kapitel soll das im Rahmen dieser Arbeit realisierte Animationssystem EMPHAS (**E**asy-to-use **M**odular **PH**ysically-based **A**nimation **S**ystem) vorgestellt werden, das auf der Grundlage der im vorigen Kapitel eingeführten Konzepte entwickelt wurde. EMPHAS stellt ein modulares System zur einfachen Erstellung physikalisch basierter Animationen mechanischer Systeme auf der Grundlage der LFM dar. Es läßt sich vollständig interaktiv bedienen und ermöglicht eine sehr effiziente Bewegungsgenerierung. In [Wag99] und [Wag00] wurden bereits einige Grundzüge dieses Systems vorgestellt.

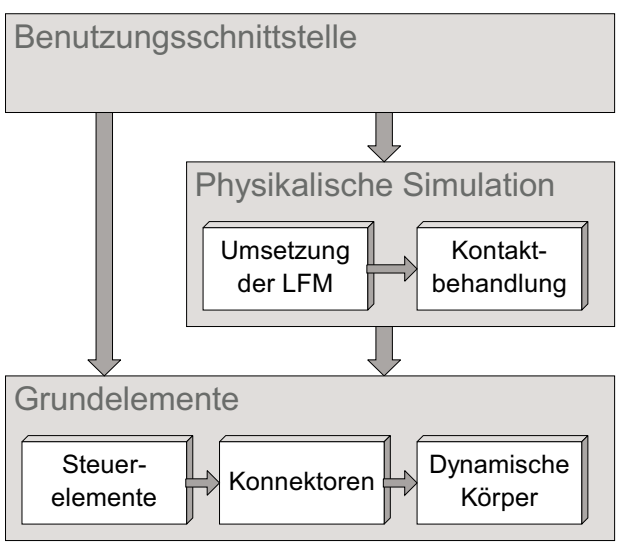

Abbildung 6.1: Architektur von EMPHAS.

Die Modularität von EMPHAS bezieht sich sowohl auf die objektorientierte Implementation als auch auf die Anwendung der Modellierungs- und Steuertechniken: Dynamische Körper und Steuerelemente können als voneinander unabhängige Module behandelt werden, die sich wie in einem Baukastensystem beliebig miteinander kombinieren lassen. Zur Bewegungssteuerung können holonome Constraints, Geschwindigkeits-Constraints, Controller, Kraftfelder und ereignisbasierte Prozeduren eingesetzt werden. Außerdem wurde eine automatische Behandlung von Kollisionen und Ruhekontakten verwirklicht. Die Bedienung des Systems wird zudem durch eine Reihe von Interaktionstechniken unterstützt. Eine grobe Darstellung der Architektur von EMPHAS zeigt Abbildung 6.1. In Anhang A wird diese Architektur detaillierter beschrieben.

Die Grundelemente von EMPHAS werden im anschließenden Abschnitt 6.2 vorgestellt: Dynamische Körper, Konnektoren und modular konzipierte Techniken zur Modellierung und Bewegungssteuerung. Dann wird die Umsetzung der LFM und die automatische Kontaktbehandlung beschrieben. Anschließend werden in Abschnitt 6.5 die Aspekte der graphischen Darstellung, der Simulationssteuerung und der Interaktion behandelt. Auf der Grundlage dieser Ausführungen läßt sich dann auch in Abschnitt 6.6 die Anwendung der Steuerelemente genauer erläutern und ein Vergleich zu kommerziellen Animationssystemen ziehen.

6.2 Grundelemente

Die in EMPHAS realisierten dynamischen Körper, Konnektoren und Steuerelemente sollen zusammengenommen als *Grundelemente* bezeichnet werden. Sie werden nun im einzelnen vorgestellt.

6.2.1 Dynamische Körper

Die Objekte in EMPHAS, deren Zustandswerte sich als Resultat der physikalischen Simulation ergeben, werden *dynamische Körper* genannt. Sie sind die eigentlichen Grundelemente der physikalisch basierten Animation und die Verleihung eines realistisch wirkenden und den Wünschen des Animateurs entsprechenden Bewegungsverhaltens stellt das Ziel der Animationserstellung mit EMPHAS dar.

Für mechanische Systeme spiegelt sich der Zustand eines solchen Körpers in seiner räumlichen Lage und seiner Geschwindigkeit wieder. Andere Körpereigenschaften wie Masse oder Massenverteilung werden dagegen in der Regel als zeitlich konstante Größen behandelt. Sie können aber einen großen *Einfluß* auf die Bewegung des Körpers haben, d.h. sie stellen dynamische Körpereigenschaften dar (vergl. Abschnitt 2.2.1).

Die wichtigsten Grundobjekte mechanischer Systeme sind massenbehaftete Punktteilchen und dreidimensionale starre Körper. Diese beiden Typen wurden in EMPHAS in Form der Objekte *PointBody* und *RigidBody* realisiert, die in diesem Abschnitt genauer besprochen werden sollen.

Auch eine große Klasse von Partikelsystemen läßt sich mit Hilfe dieser beiden Körpertypen simulieren. Derartige Systeme bestehen aus Gruppen einfacher Körper, die punktförmig oder auch ausgedehnt sein können, und sich damit mit Hilfe der Typen *PointBody* und *RigidBody* modellieren lassen, sofern sich die Bewegungen der einzelnen Partikel aus den Gesetzen der Klassischen Mechanik ergeben sollen. Methoden zur einfachen *Steuerung* von Partikelsystemen, die z.B. die Festlegung von „Quellen“ und „Senken“ für die Partikel umfassen, sind in EMPHAS aber nicht implementiert worden und wurden weiterführenden Arbeiten vorbehalten. Die physikalisch basierte Animation dieser Systeme wird daher nur sehr eingeschränkt unterstützt und im weiteren Verlauf dieser Arbeit nicht explizit diskutiert.

Prinzipiell ist auch die Einbindung deformierbarer Körper in EMPHAS möglich, da die Umsetzung der LFM und auch die Steuerelemente unabhängig vom jeweiligen Körpertyp realisiert werden konnten. Für ihre mathematische Beschreibung und ihre Integration in die Benutzungsschnittstelle von EMPHAS wäre aus dieser Erweiterung aber ein Mehraufwand erwachsen, der im Hinblick auf die Zielstellung dieser Arbeit nicht gerechtfertigt erschien.

PointBody. Ein massenbehaftetes Punktteilchen hat die Zustandsgrößen Position und Geschwindigkeit, die für EMPHAS bezüglich des Weltkoordinatensystems definiert sind (also absolute Koordinaten darstellen). Sie werden durch zwei dreidimensionale Vektoren repräsentiert. Die Masse des Teilchens wird als ein zeitlich konstanter, skalarer Parameter behandelt. Wie alle Objektparameter kann er jederzeit über die Benutzungsoberfläche von EMPHAS modifiziert werden.

RigidBody. Position und Geschwindigkeit stellen auch für starre Körper Zustandsgrößen dar. Zusätzlich muß hier aber die Beschreibung der Körperrotation und deren Geschwindigkeit berücksichtigt werden. Wie in Abschnitt 6.3.1 ausgeführt wird, kommt in EMPHAS hierzu eine Quaternion und ein dreidimensionaler Rotationsgeschwindigkeitsvektor zum Einsatz. Im Gegensatz zu Punktkörpern wird die Gesamtmasse eines starren Körpers *nicht* als Parameter, sondern als abgeleitete Größe betrachtet. Das gilt auch für den Trägheitstensor, der ebenfalls für die Aufstellung der Bewegungsgleichungen bekannt sein muß. Stattdessen dienen die als räumlich (und zeitlich) konstant angenommene Dichte und die geometrische Form des Körpers als Körperparameter. Sie lassen sich in EMPHAS jederzeit ändern, im Fall der geometrischen Form z.B. auch durch eine uniforme Skalierung. Die Gesamtmasse und der Trägheitstensor werden nach einer solchen Änderung automatisch aus diesen Größen berechnet.

Um die dreidimensionale geometrische Form der starren Körper beschreiben zu können, wurde in EMPHAS die Klasse *Shape* eingeführt, auf die die *RigidBody*-Klasse via Delegation zugreift. Neben der Verwaltung der Formparameter stellt sie u.a. Funktionen zur Volumenberechnung, zur Skalierung und zur Übertragung in eine Polyeder-Datenstruktur zur Verfügung, die für die Kontaktbehandlung benötigt wird. Als Grundformen wurden in EMPHAS Kugeln, Quader, Würfel, Zylinder und Kegel als *Shape*-Unterklassen implementiert[1]. Diese Typen wurden ausgewählt, da sie Grundformen darstellen, die in sehr vielen Animationen benötigt werden, und da sie auf einfache Weise graphisch dargestellt werden können (vergl. Abschnitt 6.5.2.2). Für den Einsatz komplexerer Formen wurde ein sogenanntes *PolyederShape*-Objekt realisiert, mit dem beliebige Polyeder beschrieben werden können. Es dient als Grundlage für den Import geometrischer Körper aus anderen Animationssystemen, der in Abschnitt 6.5.5 vorgestellt wird. Die interaktive Festlegung oder Editierung solcher Objekte in Form einer geometrischen Modellierung ist in EMPHAS aber nicht vorgesehen worden. Diese Aufgabe kann als weitgehend unabhängig von der physikalisch basierten Animation betrachtet werden und weist insbesondere keine Anknüpfungspunkte zu der Beurteilung und Anwendung der LFM auf. Die Realisierung eines flexiblen und einfach anwendbaren Werkzeuges zur geometrischen Modellierung hätte zudem den Rahmen dieser Arbeit gesprengt.

6.2.2 Konnektoren

Konnektoren sind als Bindeglieder zwischen Körpern und Constraints kein Bestandteil des dynamischen Systems, sondern stellen Hilfsobjekte dar, die die Anwendung von Constraints und Controller wesentlich erleichtern (vergl. Abschnitt 5.1). Wie in Abschnitt 6.6.1 genauer ausgeführt wird, konnte auf diese Weise auch eine sehr intuitive, „baukastenartige“ Modellierung und Bewegungssteuerung verwirklicht werden.

[1] Eine *PointShape* genannte Klasse repräsentiert zudem einen ausdehnungslosen Punkt und dient als ein *Shape*-Platzhalter für punktförmige Körper.

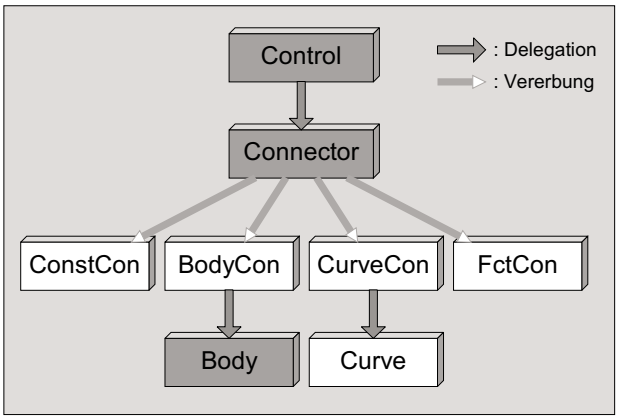

Abbildung 6.2: Realisierung des Konnektorenkonzeptes in EMPHAS.

Constraints und Konnektoren wurden in EMPHAS in separate Klassen gekapselt, die Funktionen zur Bestimmung und Rückgabe der in Abschnitt 5.1 aufgelisteten Informationen bereitstellen. Die Schnittstellen sind dabei über die abstrakten Basisklassen *Constraint* und *Connector* festgelegt, während in einer leicht erweiterbaren Reihe von Unterklassen die eigentlichen Funktionalitäten realisiert sind. Der Zugriff der Constraints auf die Konnektoren und der Konnektoren auf die dynamischen Körper wird dabei in Abbildung 6.2 ersichtlich, wobei die Klasse *Control* sowohl die Constraints als auch die Controller umfaßt. Auch Controller können daher in EMPHAS über Konnektoren auf dynamische Körper zugreifen und die von ihnen bereitgestellten Informationen nutzen. In Anhang A werden diese Klassenbeziehungen genauer beschrieben.

Die in EMPHAS realisierten Konnektortypen sollen nun vorgestellt werden. Um als Anknüpfungspunkt zu dienen, müssen sie zumindest Funktionen zur Rückgabe der Position und Geschwindigkeit des Konnektors bezüglich des Weltkoordinatensystems bereitstellen. Sie verfügen aber auch über ganz spezielle weitere Funktionalitäten, wie nun genauer ausgeführt werden soll.

ConstConnector. Der einfachste Konnektortyp ist der *ConstConnector*, der als Position und Geschwindigkeit die Werte zurückliefert, die zuvor über entsprechende Set-Funktionen festgelegt wurden (bzw. Standardwerte, wenn dies nicht geschehen ist). Er entspricht damit einem ortsfesten oder gleichförmig bewegten Raumpunkt. Auf diese Weise ist eine parametrische Einflußnahme auf den jeweiligen Constraint oder Controller möglich, ohne das modulare Konzept durchbrechen zu müssen.

BodyConnector. Der *BodyConnector* entspricht körperfesten Anknüpfungspunkten und stellt den wichtigsten Konnektortyp dar. Jeder Constraint oder Controller muß auf mindestens einen *BodyConnector* zugreifen, um in das dynamische System eingreifen zu können. Neben der aktuellen Position und der Geschwindigkeit liefert er auch die partiellen Ableitungen dieser Größen bezüglich der Zustandsvariablen des mit ihm verknüpften Körpers zurück. Diese Informationen werden für die Anwendung der Constraints benötigt (vergl. Abschnitt 5.1.3). Der *BodyConnector* ist auch der einzige Konnektortyp, der in EMPHAS graphisch dargestellt wird (vergl. Abschnitt 6.5.2). Auf diese Weise kann er in einfacher Weise selektiert und für die Anwendung von Constraints und Controllern ausgewählt werden.

Jedem Körper werden in EMPHAS bei seiner Erzeugung eine Reihe dieser Konnektoren zugeordnet, die symmetrisch auf diesem angeordnet sind. Die Anzahl und Art der Anordnung ist dabei fest vor-

gegeben und hängt vom Typ und von der geometrischen Form des Körpers ab. Zusätzlich wird dem Schwerpunkt des Körpers ein Konnektor zugeordnet. Die interaktive Modifikation und Erweiterung um zusätzliche Anknüpfungspunkte, die vor allem ein Problem im Rahmen der Benutzungsschnittstelle darstellt, wurde dabei weiterführenden Arbeiten vorbehalten.

Den in EMPHAS realisierten Konnektoren dieses Typs wurden zudem zwei zusätzliche Funktionalitäten verliehen. Zum einen stellt ein *BodyConnector* Informationen über bestimmte Parameterwerte der mit ihm verknüpften Körper zur Verfügung, die auf diese Weise von Constraints oder Controllern, die auf den Konnektor zugreifen, ausgewertet werden können. Um die Modularität von Constraints und Körpern nicht aufzuheben, werden dabei aber nur Parameter sichtbar gemacht, die nicht körperspezifisch sind, wie z.B. der bei jedem Körpertyp definierte Massenwert. Der Vorteil dieses Vorgehens liegt in einer größeren Flexibilität für die einsetzbaren Steuertechniken, die diese Informationen nutzen können. Der in Abschnitt 6.2.4.1 beschriebene *TrajectForce*-Controller stellt hierfür ein Beispiel dar, da er Zugriff auf die Größe der Körpermassen haben muß.

Die zweite Funktionalität besteht in der Möglichkeit, die vom Konnektor beeinflußten Translations- und Rotationsfreiheitsgrade (separat voneinander) sperren zu können. Diese Sperrung wird über einen konnektoreigenen Parameter festgelegt, der die Werte 0 (gesperrte Rotation), 1 (gesperrte Translation) oder 2 (keine Sperrung) annehmen kann. Hierzu werden gegebenenfalls die partiellen Ableitungen nach den jeweiligen Zustandsvariablen gleich 0 gesetzt. Auf diese Weise kann erreicht werden, daß der über diese Konnektoren wirkende Constraint nur die Translation oder nur die Rotation des zugehörigen Körpers beeinflußt.

Auch diese Funktionalität läßt sich für eine höhere Flexibilität der Steuerelemente einsetzen, wie in Abschnitt 6.2.3.2 gezeigt wird. Sie stellt aber auch eine Möglichkeit zur Lösung des in Abschnitt 5.4.3 diskutierten Problems der unerwünschten Bewegungen im Anschluß einer Constraint-Festlegung dar, wie in Abschnitt 6.6.3 ausgeführt wird.

CurveConnector. Mit dem *CurveConnector* lassen sich Kurven als Anknüpfungsobjekte für Constraints und Controller festlegen, um feste Verbindungen zwischen Kurven- und Körperpunkten oder auch Pfadvorgaben verwirklichen zu können. In EMPHAS wurde dazu eine Klasse *Curve* implementiert, mit der sich parametrisierte Bezier-Splines festlegen lassen. Ein *CurveConnector* kann entweder mit einem festen Kurvenparameterwert oder mit der Angabe eines dreidimensionalen Punktes initialisiert werden. Im ersten Fall wird der zum Kurvenparameter gehörende Kurvenpunkt als Konnektorposition festgelegt, während im zweiten Fall derjenige Kurvenpunkt bestimmt wird, der dem vorgegebenen Punkt (der i.a. nicht auf der Kurve liegt) am nähesten ist. Auf diese Weise kann ein „Abstand“ eines Punktes zu einer Kurve definiert werden, was z.B. für die Festlegung des *Distance*-Constraints wichtig ist. Der *CurveConnector* ermöglicht somit einen sehr flexiblen Einsatz von Kurven, die in sehr vielen Steuerelementen zur Anwendung kommen können (vergl. Abschnitt 6.6.1).

FctConnector. Der *FctConnector* ermöglicht schließlich die Constraint-Verbindung mit einem Punkt, der einer explizit vorgegebenen raumzeitlichen Kurve folgt. Zu diesem Zweck kodiert der Konnektor drei skalare, polynomiale Funktionen der Zeit [2], deren aktueller Funktionswert den drei Komponenten der Konnektorposition zugeordnet werden. Dazu kann der Konnektor vor jedem Simulationsschritt mit dem aktuellen Wert des Zeitparameters initialisiert werden. Er liefert außerdem die ersten und zweiten Ableitungen dieser Funktionen, die bei der Einbindung derjenigen Constraints benötigt werden, die auf den *FctConnector* zugreifen.

Als funktionale Abhängigkeiten können bei dem *FctConnector* beliebige polynomiale Beziehungen festgelegt werden, die sich durch die Angabe einer Zeichenkette spezifizieren lassen. Eine über

[2] Die Funktionen sind im Grunde bezüglich eines beliebigen skalaren Parameters definiert, dem aber bei dem *Einsatz* des Konnektors in EMPHAS ausschließlich die Interpretation der Zeit zugewiesen wird.

polynomiale Beziehungen hinausgehende Formelinterpretation, z.B. mit Hilfe einer speziellen Beschreibungssprache, wurde dagegen weiterführenden Arbeiten vorbehalten. Da auch die Ableitungen der Funktionen automatisch berechnet werden müßten, ist hierfür der Einsatz computeralgebraischer Verfahren erforderlich.

6.2.3 Constraints

In Abschnitt 5.2 wurden einige Constraints beschrieben, die eine grundlegende Bedeutung für die Computeranimation haben. Mit Hilfe des verallgemeinerten Konnektorenkonzeptes (Abschnitt 5.1) konnten in EMPHAS sämtliche dieser Constraints und zwei zusätzliche Varianten in modularer Weise realisiert und in das Gesamtsystem eingebunden werden. Sie stellen in EMPHAS das wichtigste Konstruktionsmittel zur Modellierung und Bewegungssteuerung dar.

Zunächst wird die Umsetzung dieser Constraints genauer erläutert. Mit der selektiven Sperrung von Freiheitsgraden und Constraint-Komponenten werden dann zwei Funktionalitäten vorgestellt, die zu ihrer flexiblen Anwendung beitragen. Schließlich werden die Erweiterungsmöglichkeiten um zusätzliche Constraint-Typen besprochen.

6.2.3.1 Realisierung der Constraints

Aufgrund der einfachen mathematischen Form, die die Constraints im Rahmen des Konnektorenkonzeptes annehmen, konnten die in Abschnitt 5.2 vorgestellten holonomen Constraints *PointToPoint*, *Distance*, *Orientation*, *InLine*, *Ortho* und die Geschwindigkeits-Constraints *EqualVelocity* und *EqualVelocityNorm* problemlos in entsprechende gleichnamige Klassen übertragen werden. Die Projektionsvektoren, die für die Festlegung der *Orientation*- und *InLine*-Constraints erforderlich sind, werden in EMPHAS dabei automatisch aus der jeweiligen räumlichen Lage der ersten Achse bestimmt und bei jedem Zeitschritt neu berechnet. Die Ableitungen der Constraint-Funktionen, die ebenfalls zur Definition der jeweiligen Klasse gehören, sind in Anhang G aufgeführt.

Die Typen *Hinge*, *Slide* und *Universal* konnten durch Rückgriff auf diese Typen realisiert werden. Der *Hinge*-Constraint wurde z.B. durch die Kombination von einem *PointToPoint*- und einem *Orientation*-Constraint mit Hilfe einer Delegation an die beiden entsprechenden Klassen implementiert. Der *Slide*- und der *Universal*-Constraint wurden in ganz analoger Weise mit Hilfe einer Delegation an zwei *Inline*-Klassen (im Falle des *Slide*-Constraints) bzw. an eine *PointToPoint*- und eine *Ortho*-Klasse (im Falle des *Universal*-Constraints) verwirklicht.

Für die Umsetzung des *PointToFunction*- und des *VelocityToFunction*-Constraints wurde der in Abschnitt 6.2.2 vorgestellte *FctConnector* eingesetzt, mit dem sich beliebige polynomiale Funktionen der Zeit als Zielfunktion festlegen lassen. Die entsprechenden Funktionsparameter können dabei jederzeit vom Animateur geändert werden. Diese beiden Constraints stellen daher ein sehr flexibles Hilfsmittel zur Bewegungssteuerung dar.

Neben diesen in 5.2 aufgeführten Constraints wurden in EMPHAS zudem zwei zusätzliche Typen als Varianten des *Distance*- bzw. des *PointToPoint*-Constraints eingeführt: der *UnitDistance*- und der *PointToPath*-Constraint.

Als standardmäßige Längenvorgabe dient bei *Distance*-Constraints die aktuelle Distanz zwischen den beiden Punkten zum Zeitpunkt der Constraint-Erstellung. Als einziger Unterschied zu diesem Typ erhält der *UnitDistance*-Constraint standardmäßig eine Einheitslänge als Vorgabe für den Abstandswert. Eine solche Vorgabe erweist sich zuweilen als nützlichere Möglichkeit. Die mathematische Form ist aber mit der des *Distance*-Constraints identisch.

Der *PointToPath*-Constraint ist ein reiner Steuer-Constraint. Mit seiner Hilfe kann einem Körperpunkt ein Pfad in Form einer zuvor erstellten Kurve zugewiesen werden, auf der er sich gleichförmig bewegen soll. Für die Verknüpfung mit der Kurve kommt dabei der in Abschnitt 6.2.2 vorgestellte *CurveConnector* zum Einsatz. Sehr nützlich ist eine solche Vorgabe z.B. für den Aufhängepunkt eines Pendels oder eines anderen komplexen Gebildes, das unter dem Einfluß der Schwerkraft steht. Als Parameter besitzt dieser Constraint die (skalare) Größe der Geschwindigkeit dieser Bewegung sowie einen boolschen Wert, der das Verhalten bei Erreichen des Kurvenendes festlegt – entweder Bewegungsstopp oder Richtungsumkehr mit der gleichen Absolutgeschwindigkeit.

	Verknüpfung	Dim.	holon.	rheon.
PointToPoint	2 Punkte	3	ja	nein
PointToPath	1 Punkt, 1 Kurve	3	ja	nein
Distance	2 Punkte	1	ja	nein
UnitDistance	2 Punkte	1	ja	nein
Orientation	2 Achsen	2	ja	nein
InLine	1 Punkt, 1 Achse	2	ja	nein
Ortho	2 Achsen	1	ja	nein
Hinge	2 Punkte, 2 Achsen	5	ja	nein
Slide	2 Punkte, 2 Achsen	4	ja	nein
Universal	2 Punkte, 2 Achsen	4	ja	nein
PointToFunction	1 Punkt	3	ja	ja
EqualVelocity	2 Punkte	3	nein	nein
EqualVelocityNorm	2 Punkte	1	nein	nein
VelocityToFunction	1 Punkt	3	nein	ja

Tabelle 6.1: Die in EMPHAS realisierten Constraints.

In EMPHAS wurde dieser Constraint als Unterklasse der *PointToPoint*-Klasse realisiert, um die dort implementierten Funktionalitäten mitnutzen zu können. Als Erweiterung wird bei jedem Zeitschritt ein Zielpunkt auf der zugehörenden Kurve als zweiter Bezugspunkt bestimmt. Dazu wird zunächst der Punkt der Kurve ermittelt, der dem Körperpunkt am nächsten ist. Dann wird der Kurvenparameter dieses Punktes entsprechend der aktuellen Geschwindigkeit modifiziert, wodurch sich der gewünschte Zielpunkt ergibt.

Tabelle 6.1 umfaßt eine Auflistung der Constraints, die in EMPHAS realisiert wurden. Neben der Dimension des Constraints und den Angaben, ob er holonom oder rheonom (d.h. explizit zeitabhängig) ist, wurde dabei auch die Information aufgenommen, auf welche geometrischen Eingangsdaten sich der Constraint bezieht.

Spezielle nichtholonome Constraints in Form von Ungleichungen wurden in EMPHAS dagegen nicht implementiert. Hierfür sprachen im wesentlichen praktische Gründe. Das in Kapitel 5 beschriebene Konzept zur Umsetzung der LFM ist lediglich für beschleunigungslineare Constraints in standardmäßiger Weise anwendbar. Die Behandlung von Ungleichungs-Constraints ist dagegen nicht trivial, wie z.B. an dem in Anhang D skizzierten Verfahren für das Ruhekontaktproblem ersichtlich wird. Für die Modellierung und Bewegungssteuerung im Rahmen der physikalisch basierten Animation ist ihre Bedeutung außerdem deutlich geringer als die der holonomen und Geschwindigkeits-Constraints einzuschätzen. Im Rahmen weiterführender Arbeiten wäre eine Erweiterung um diesen Constraint-Typ aber durchaus möglich.

6.2.3.2 Selektive Sperrung von Freiheitsgraden

Eine zusätzliche in EMPHAS realisierte Möglichkeit für die Anwendung von Constraints ist die selektive Sperrung von Freiheitsgraden auf der Grundlage der in Abschnitt 6.2.2 beschriebenen Funktionalität des *BodyConnectors*. Für jeden Constraint kann interaktiv festgelegt werden, ob er sich auf die Translations-, die Rotations- oder (was die Standardvorgabe darstellt) auf alle Freiheitsgrade der mit ihm verknüpften Körper bezieht. Diese Auswahlmöglichkeit läßt sich dabei für jeden Konnektor des Constraints treffen, der einem dynamischen Körper zugeordnet ist. Bei bestimmten Constraint-Typen und Zustandskonfigurationen kann die Wegnahme von Freiheitsgraden zu unerwünschten Effekten oder überbestimmten Systemen führen. In vielen Situationen ist diese Umschaltmöglichkeit aber sehr sinnvoll. Ein *PointToPoint*- oder ein *Distance*-Constraint, der sich nur auf die Translationsfreiheitsgrade der zugehörigen Körper bezieht, läßt sich z.B. in gewohnter Weise interpretieren, beeinflußt aber die Rotationszustände der Körper nicht. Er führt somit zu reinen Translationsbewegungen, die sich ohne diese Umschaltmöglichkeit nur sehr schwer (z.B. durch die Anwendung zusätzlicher Constraints) erzielen ließen. Diese selektive Sperrung von Freiheitsgraden konnte daher in EMPHAS für eine wesentliche Erweiterung der Steuermöglichkeiten ausgenutzt werden.

6.2.3.3 Selektive Sperrung von Constraint-Komponenten

Jeder Constraint ist durch eine mathematische Funktion $\mathbf{C}(\mathbf{x},t)$ bzw. $\mathbf{C}(\mathbf{x},\mathbf{v},t)$ festgelegt, deren Komponentenanzahl seiner Dimension entspricht. In EMPHAS wurde nun die selektive Sperrung einzelner Constraint-Komponenten ermöglicht, die interaktiv durchgeführt werden kann. Auf diese Weise läßt sich z.B. die Wirkung eines *PointToPoint*-Constraints auf eine beliebige Koordinatenachse oder -ebene beschränken. Auch diese Funktionalität führt zu einer wesentlich höheren Flexibilität bei der Anwendung der Constraints Diese Technik ist auch für die Behandlung von überbestimmten Systemen nützlich, wie in Abschnitt 6.6.2 erläutert wird.

In EMPHAS kann außerdem jeder Constraint deaktiviert werden, ohne dabei seine Verknüpfung zu den jeweiligen Konnektoren zu verlieren. Auf diese Weise läßt er sich problemlos zu einem späteren Zeitpunkt wieder aktivieren.

6.2.3.4 Erweiterung um zusätzliche Constraints

Zusammen mit den Controllern bilden die hier beschriebenen Constraints das Grundgerüst für die Modellierung und Bewegungssteuerung mit EMPHAS. Es soll nun noch die Frage untersucht werden, wie sich eine Erweiterung um zusätzliche Constraints vornehmen läßt.

Für die Realisierung eines neuen Constraints müssen in EMPHAS die zu den jeweiligen Constraints gehörenden Funktionen und deren Ableitungen explizit gebildet und implementiert werden. Eine Formulierung von Constraints mit Hilfe einer speziellen Beschreibungssprache und die automatische Generierung der Ableitungen, wie sie z.B. in [GW93], [Gle94a] angewandt wurde, hätte den Rahmen dieser Arbeit gesprengt und ist daher weiterführenden Arbeiten vorbehalten worden. Mit Hilfe des Konnektorenkonzeptes können die Constraints aber unabhängig von den verwandten Konnektor- und Körpertypen formuliert werden, was die Implementation der Constraint-Funktionen sehr vereinfacht. Durch die modulare Konzeption von EMPHAS ist die Bereitstellung und Anwendung dieser zusätzlichen Constraints nach der erfolgten Funktionsfestlegung außerdem problemlos realisierbar.

6.2.4 Weitere Steuerelemente

6.2.4.1 Controller

Controller liefern für jeden Zeitpunkt eine Kraft, die über die mit ihnen verknüpften Konnektoren an den jeweiligen Körpern angreift. Diese Kräfte können dem System dann als zusätzliche äußere Kräfte hinzugefügt werden. Die abstrakte Klasse *ForceConstraint* legt daher Funktionen zur Rückgabe der Anzahl der verknüpften Konnektoren und der am Konnektor i wirkenden Kraft $\mathbf{f}_i$ fest. Die eigentliche Kraftbestimmung ist in den daraus abgeleiteten Unterklassen implementiert, die nun vorgestellt werden sollen.

VectorForce. Eine sehr flexible Art, die Bewegung eines Körpers im Rahmen der physikalisch basierten Animation zu beeinflussen, ist die Festlegung von direkten Kraftwirkungen. Dies leistet der *VectorForce*-Controller, mit dem jedem Körperpunkt ein konstanter Kraftvektor zugeordnet werden kann. Die Größe und Richtung dieser Kraft lassen sich dabei als Parameter des Controllers jederzeit ändern. Da eine konstante Kraft eine ständig wachsende Geschwindigkeit des Körpers bewirken würde, ist ein solcher Constraint nach seiner Erstellung voreinstellungsgemäß deaktiviert. Er kann dann vom Animateur für den gewünschten Zeitraum aktiviert werden.

SpringForce. Dieser Controller stellt eine Verbindung zwischen einem Körperpunkt und einem anderen Anknüpfungspunkt mit Hilfe eines Federmodells her: die Auslenkung von der Ruhelänge bewirkt eine Kraft, die der Bewegung entgegengesetzt ist und proportional zur Größe der Auslenkung ist. Sie wird dabei durch einen Dämpfungsterm abgeschwächt, der wiederum proportional zur Relativgeschwindigkeit der beiden Punkte ist. Wenn $\mathbf{r}_{rel}$ den Relativvektor der beiden Punkte, v_{rel} die Relativgeschwindigkeit (entlang der Verbindungsrichtung $\mathbf{r}_{rel}$), d_0 die Ruhelänge der Feder, α die Federstärke und β den Dämpfungsgrad bezeichnet, ergibt sich die Federkraft demnach aus der Beziehung

$$\mathbf{F} = \alpha\,(|\mathbf{r}_{rel}| - d_0)\,\mathbf{r}_{rel} - \beta\,|v_{rel}|\,\mathbf{r}_{rel}\ .$$

Die Größen α, β und d_0 stellen dabei Parameterwerte des Controllers dar.

Derartige Federverbindungen sind ein wichtiges Hilfsmittel zur Konstruktion komplexer Körper, aber auch als Steuertechnik können sie sehr sinnvoll eingesetzt werden.

TrajectoryForce. Der *TrajectoryForce*-Controller stellt ein Steuerelement dar, das an der in [L+95] beschriebenen sogenannten *Trajectory Control*-Technik angelehnt ist. Er dient dem Ziel, einen Körper auf einer vom Animateur vorgegebenen Trajektorie bewegen zu lassen, ohne die dynamischen Einflüsse durch andere Körper zu ignorieren. Falls es z.B. zu einem Zusammenstoß mit einem anderen Körper kommt, wird zunächst die übliche Kollisionssimulation angewandt, die den Körper i.a. von der Zieltrajektorie wegbewegt. Nach einer kurzen Zeit, die sich parametrisch beeinflussen läßt, wird er dann aber wieder auf diese Bahn zurückgeführt. Um ein solches Verhalten zu ermöglichen, wendet der *TrajectoryForce*-Controller auf die beiden Körperpunkte $\mathbf{P}_1$ und $\mathbf{P}_2$ die folgende Kraft an[3]:

$$\mathbf{F} = \alpha m_2\,(\mathbf{P}_1 - \mathbf{P}_2) - \beta\sqrt{2\alpha}\,m_2\,\dot{\mathbf{P}}_2\ .$$

Dabei sind α und β zwei skalare Parameter und m_2 steht für die (Gesamt-) Masse des zweiten Körpers. Die Massenabhängigkeit ist wichtig, damit leichte Körper in ähnlicher Weise wie schwere Körper auf die Zieltrajektorie zurückgeführt werden können.

Dieser Controller ermöglicht eine sehr intuitive Bewegungssteuerung. Die Zieltrajektorie wird dabei in der Regel mit Hilfe eines anderen Körpers festgelegt, dem dann der vom Controller beeinflußte

[3] Die mathematische Form dieser Kraftfunktion wurde dabei [L+95] entnommen.

Körper nachfolgt, ohne unplausibel wirkende Abweichungen vom dynamischen Verhalten zu verursachen.

6.2.4.2 Prozeduren

Wie in Abschnitt 2.3.2.5 bereits angesprochen wurde, können ereignisbasierte Prozeduren die Flexibilität der Bewegungssteuerung erheblich erhöhen. In EMPHAS wurden Prozeduren als Erweiterung der auf Constraints und Controllern basierenden Steuermöglichkeiten verwirklicht.

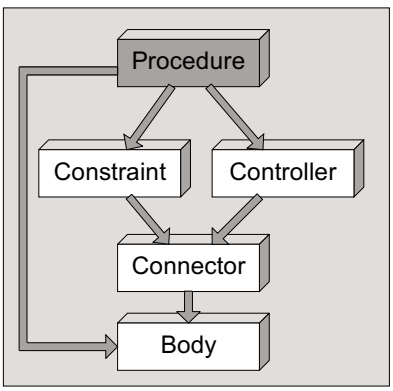

Abbildung 6.3: Das in EMPHAS realisierte Prozeduren-Konzept.

Dabei konnte ein sehr allgemeines Prozeduren-Konzept realisiert werden, das auch in [Wag00] beschrieben wurde. Die Prozeduren sind in EMPHAS keine Alternative zu Controllern und Constraints, sondern stellen eine höhere Steuertechnik dar, die sich dieser Module bedient, aber auch einen direkten Körperzugriff ermöglicht (siehe Abbildung 6.3). Dabei können nicht nur die Zustandswerte der Körper, sondern auch sämtliche Parameter von Körpern, Controllern und Constraints zur Simulationszeit prozedural modifiziert werden. Auf diese Weise lassen sich Constraints z.B. auch aktivieren und deaktivieren. Aufgrund der in Abschnitt 4.2 besprochenen Eigenschaften der LFM werden hierdurch keine unnatürlich wirkenden Bewegungssprünge verursacht. Dieser Aspekt ist für die Animationserstellung sehr wichtig. Der hierdurch ermöglichte flexible Einsatz von Constraints eröffnet dem Animateur eine Vielzahl von Möglichkeiten, mit Hilfe von Prozeduren auf die Körperbewegungen einzuwirken.

Als ein einfaches Beispiel einer solchen Prozedur wurde die sogenannte *DistanceProcedure* implementiert. Sie wird für ein Paar von Körpern aktiv, deren Distanz einen bestimmten Schwellwert d_0 unterschreitet, der als Abstand der zugehörigen umschließenden Kugeln definiert ist. Für diese Körper wird dann ein *Distance*-Constraint aktiviert, dessen Distanzparameter dem Produkt von d_0 und einem Parameter k gleichgesetzt wird. Für $k \geq 1$ resultiert auf diese Weise eine Abstoßung der beiden Körper, die aber nur solange wirkt, bis ihr Abstand wieder den Wert d_0 überschreitet. Im Fall $k < 1$ pegelt sich das System dagegen auf einen festen Abstand ($< d_0$) ein. Für dieses Körperpaar bleibt die Prozedur somit aktiv, bis die Körper durch andere Einflüsse voneinander getrennt werden.

Mit dieser Prozedur lassen sich interessante Bewegungsabläufe erzielen. Auf der einen Seite wird (bei $k > 1$) ein „weiches“ Kollisionsverhalten modelliert, das eine nützliche Alternative zu der standardmäßigen Kollisionssimulation auf der Grundlage instantan wirkender Kraftstöße darstellt. Da die Körper dabei schon vor ihrer Berührung abgelenkt werden, kann die *DistanceProcedure* auch als eine Technik zur effektiven Kontakt*vermeidung* eingesetzt werden. Andererseits führt das oben beschriebene Einfangen von Körpern (bei $k < 1$) zu einem interessanten Gruppenverhalten, das sich z.B. für partikelsystemartige Gebilde anwenden läßt.

6.2.4.3 Kraftfelder

Mit den in EMPHAS realisierten Controllern können zustandsabhängige Kräfte spezifiziert werden, die an den jeweils festgelegten Körperpunkten angreifen. Mit dem Konzept der *Kraftfelder* lassen

sich dagegen globale Kräfte festlegen, denen sämtliche dynamische Körper unterworfen sind. Das wichtigste Beispiel eines solchen Feldes stellt die Schwerkraft dar, die auf jeden Körper eine zur Körpermasse proportionale Kraft in Richtung der negativen z-Achse („unten“) ausübt. Die Modellierung einer solchen Kraft ist für die realistische Nachbildung mechanischer Bewegungsabläufe unverzichtbar.

Ein anderes in EMPHAS implementiertes Kraftfeld ist das *FrictionForceField*. Es übt auf jeden Körper eine Kraft aus, die proportional zur Masse und zum Betrag der Geschwindigkeit ist und seiner Bewegung entgegengerichtet ist. Auf diese Weise wird eine Art von Reibung erzeugt, die die Bewegung des Körpers langsam abbremst, bis sie (bei Abwesenheit anderer Einflüsse) zum Stillstand kommt. Die Stärke dieser Bremskraft läßt sich dabei über die Festlegung eines entsprechenden Parameterwertes steuern. Obwohl dieses Konzept kein physikalisch korrektes Reibungsmodell darstellt, führt es in vielen Systemen zu einer höheren Plausibilität der Körperbewegungen. Vor allem bei der Modellierung trägt das Reibungsfeld dazu bei, unerwünschte Bewegungen abklingen zu lassen (siehe Abschnitt 6.6.3). Wie die Erfahrungen mit EMPHAS gezeigt haben, stellt dieses Feld daher eine wichtige Komponente bei der Animationserstellung dar.

Neben diesen standardmäßig erzeugten Kraftfeldern, die sich auf einfache Weise aktivieren und deaktivieren lassen, können in EMPHAS auch noch weitere globale Felder festgelegt werden. Als ein einfaches und gleichzeitig sehr nützliches Beispiel eines solchen Kraftfeldes wurde hierzu das *RadialForceField* implementiert. Es ist (im Unterschied zu den zuvor genannten Feldern) an einem bestimmten Raumpunkt lokalisiert und legt eine radiale Kraft fest, die umgekehrt proportional zum Abstand des jeweiligen Körpers zu diesem Raumpunkt ist. Ein Körper mit dem Ortsvektor $\mathbf{r}$ erfährt also von einem solchen Feld an der Position $\mathbf{q}$ die Kraft

$$\mathbf{F} = k \cdot \frac{\mathbf{r} - \mathbf{q}}{|\mathbf{r} - \mathbf{q}|^2} \, .$$

Durch den Parameter k ist die Stärke, aber auch die Richtung der Kraft festgelegt. Für $k > 0$ ergibt sich eine abstoßende Kraft, für $k < 0$ eine anziehende. Dieses Kraftfeld ist somit sehr universell einsetzbar. Für die flexible Bewegungssteuerung von EMPHAS stellt es ein nützliches und häufig angewandtes Steuerelement dar.

6.3 Umsetzung der LFM

In Abschnitt 5.3 wurde ein Konzept zur Umsetzung der LFM vorgestellt, das auf der Aufspaltung in die Teilaufgaben der Lagrange-Faktoren-Bestimmung, der Anwendung von Stabilisierungsverfahren und der Lösung gewöhnlicher Differentialgleichungen beruht. In EMPHAS wurde für jede dieser Aufgaben eine ganze Reihe von Verfahren realisiert, die voneinander unabhängige Module darstellen und sich jederzeit gegeneinander austauschen lassen. Die Verfahren sind dabei nicht auf die in EMPHAS realisierten punktförmigen und starren Körper zugeschnitten, sondern eignen sich für die Simulation beliebiger mechanischer Systeme (deren Zeitentwicklung durch Gleichung (3.1) beschrieben wird). Damit konnte eine Umsetzung der LFM verwirklicht werden, die sich durch einen hohen Grad an Modularität und Generalität auszeichnet.

Als Voraussetzung einer solchen Umsetzung muß zunächst ein Koordinatensatz zur Zustandsbeschreibung und ein Formalismus zum Aufstellen der Bewegungsgleichungen ausgewählt werden. Dieser Aspekt soll nun genauer besprochen werden.

6.3.1 Mathematischer Formalismus

Wie in Abschnitt 5.3.1 erläutert wurde, lassen sich als grundsätzliche Möglichkeiten zur Zustandsbeschreibung relative, absolute und redundante Koordinaten einsetzen. Relative Koordinaten kamen wegen den dort angesprochenen Schwierigkeiten nicht für die Zustandsbeschreibung in EMPHAS in Betracht. Auch die hochgradig redundanten Koordinatentypen wurden nicht für diese Aufgabe ausgewählt, obwohl sie möglicherweise Vorteile für eine effizientere Simulationsdurchführung bieten (vergl. Abschnitt 5.3.1). Hierfür sprachen im wesentlichen praktische Gründe, da die Ausarbeitung von effektiven Verfahren, die auf die Eigenheiten dieser Beschreibungsart zugeschnitten sind, den Rahmen dieser Arbeit gesprengt hätte.

Für starre Körper fiel die Wahl stattdessen auf die in Abschnitt 3.1 und Anhang C erläuterten Referenzpunktkoordinaten, bei denen der Rotationszustand mit Hilfe von Quaternionen beschrieben wird. Sie entsprechen somit im wesentlichen absoluten Koordinaten, weisen aber auch eine Redundanz auf: Die drei Freiheitsgrade der Körperrotation werden durch vier Quaternionen-Komponenten beschrieben. Diese Redundanz hätte z.B. durch die Verwendung von Euler-Winkeln vermieden werden können. Für die Simulation der Zustandsvariablen – nicht nur im Rahmen der physikalisch basierten Animation – bieten Quaternionen aber große Vorteile, wie in Anhang C ausgeführt wird. Vor allem kann mit ihnen eine numerische „Drift“ vermieden werden, die zu einer ständig wachsenden Abweichung vom korrekten Bewegungsverlauf führen würde. Bei der Beschreibung der Rotations*geschwindigkeit* treten diese Probleme nicht auf. Hierfür wurde daher der dreidimensionale Vektor der Winkelgeschwindigkeit ω ausgewählt, der eine gebräuchliche und einfach interpretierbare Größe darstellt.

Der Zustandsvektor $\mathbf{x}$ hat somit die Dimension 7 und umfaßt die drei Komponenten der (absoluten) Schwerpunktsposition $\mathbf{r}_{SP}$ und die vier Komponenten der Quaternionen $\mathbf{q}$ für die Rotationsfestlegung bezüglich dieses Schwerpunktes. Der Vektor $\mathbf{v}$ zur Beschreibung der Geschwindigkeit ist dagegen nur 6-dimensional:

$$\mathbf{x} := \begin{pmatrix} \mathbf{r}_{SP} \\ \mathbf{q} \end{pmatrix}, \; \mathbf{v} := \begin{pmatrix} \dot{\mathbf{r}}_{SP} \\ \omega \end{pmatrix}.$$

Die Zustandsbeschreibung von punktförmigen Körpern ist wesentlich unproblematischer. Sein dreidimensionaler Zustandsvektor $\mathbf{x}$ entspricht dem Ortsvektor $\mathbf{r}$ des Körpers und seine Geschwindigkeit $\mathbf{v}$ ist gleich $\dot{\mathbf{r}}$.

Als Formalismus zur Aufstellung der Bewegungsgleichungen kommen bei der LFM wie in Abschnitt 5.3.1 erwähnt mehrere gleichwertige Möglichkeiten in Betracht. Für EMPHAS wurde der Euler-Formalismus ausgewählt, da er im Vergleich zum Lagrange- und zum Jourdain-Formalismus eine vertrautere Form für die Bewegungsgleichungen liefert und aufgrund der Verwendung absoluter Koordinaten ebenso einfach anzuwenden ist. Ein sehr wichtiger Grund für diese Wahl ist außerdem der Umstand, das der Euler-Formalismus auch der Arbeit [Bar96] zu Grunde gelegt wurde, in der das Baraff-Verfahren beschrieben wird.

6.3.2 Bestimmung der Lagrange-Faktoren

In EMPHAS wurden zwei Ansätze zur Bestimmung der Lagrange-Faktoren realisiert. Sie sollen nun genauer vorgestellt werden.

Um eine einfache Referenzmethode zur Verfügung zu haben, wurde mit dem *DirectSolver* ein Verfahren implementiert, das auf der direkten Auflösung von Gleichung (3.12) beruht, ohne dabei die blockweise Zusammensetzung der Matrizen $\mathbf{J}$ und $\mathbf{M}$ auszunutzen. Aus Gründen der Einfachheit

wurde dazu das von der *LEDA*-Bibliothek ([Näh93]) bereitgestellte Lösungsverfahren für reellwertige Matrizen des Typs *matrix* ausgewählt, der in EMPHAS für die Matrixcodierung eingesetzt wird. Da die Matrix $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ in Gleichung (3.12) beim Vorhandensein von m skalaren Constraints die Dimension $(m \times m)$ hat, ist für diese Lösung ein maximaler Zeitaufwand von $O(m^3)$ zu veranschlagen[4]. Als Voraussetzung zur Anwendung des Verfahrens muß diese Matrix zudem maximalen Rang haben, d.h. sie darf nicht singulär sein.

Der Baraff-Algorithmus verspricht dagegen wie in Abschnitt 3.5.5 erwähnt eine wesentlich zeiteffizientere Bestimmung der Lagrange-Faktoren. Um ihn für das EMPHAS-System einsetzen zu können, mußte der in [Bar96] vorgestellte Pseudocode-Algorithmus implementiert und in geeigneter Weise eingebunden werden. Um auch Systeme mit geschlossenen Schleifen lösen zu können (vergl. Abschnitt 5.3.3), wurde dabei eine Möglichkeit geschaffen, einzelne primäre Constraints auf automatische Weise als sekundäre Constraints zu deklarieren, die dann an den erweiterten Algorithmus weitergegeben werden. Die Detektion geschlossener Schleifen und die daran anschließende Auswahl der umzudeklarierenden primären Constraints wird dabei im Zuge der Erstellung der Baumstruktur der Matrix $\mathbf{H}$ durchgeführt: Eine geschlossene Schleife liegt genau dann vor, wenn bei der (rekursiven) Traversierung der Kinder wieder ein ursprüngliches Element gefunden wird.

Ein ebenfalls in Abschnitt 5.3.3 angesprochenes Problem stellt das Auftreten von Koordinatensingularitäten dar. Durch den Umstand, daß jeder in EMPHAS erzeugter Körper zunächst den gleichen Satz von Default-Werten erhält, treten singuläre Konfigurationen, die ansonsten extrem unwahrscheinlich wären, sogar relativ häufig auf. Dieses Problem durfte daher nicht unberücksichtigt bleiben. Reine Koordinatensingularitäten lassen sich durch eine minimale Veränderung der Zustandswerte auflösen. In EMPHAS werden daher bei einer detektierten Singularität kleine, (pseudo-)zufällig verteilte Verrückungen der Translations- und Rotationszustandswerte vorgenommen. Um den ursprünglichen Systemzustand so wenig wie möglich zu verändern, werden dabei nur diejenigen Körper berücksichtigt, die zu dem Constraint bzw. den Constraints gehören, die die Koordinatensingularität verursacht haben. Diese Auflösung der singulären Zustände geschieht völlig automatisch und erfordert keinen Eingriff des Benutzers.

6.3.3 Erweiterung um ein SVD-Verfahren

Die oben beschriebene Umsetzung des Baraff-Verfahrens ermöglicht eine sehr effiziente Bewegungsgenerierung. Überbestimmte Systeme, die in [Bar96] keine Erwähnung fanden, lassen sich mit diesem Verfahren aber nicht behandeln. Wie die Ausführungen in Abschnitt 5.4.2 gezeigt haben, dürfen diese Konstellationen bei der Realisierung eines interaktiven Animationssystems aber nicht ignoriert werden. In EMPHAS konnte eine Lösung für dieses Problem gefunden werden. Es soll nun eine Erweiterung der oben beschriebenen Lösungsverfahren vorgestellt werden, mit der sich auch diese Systeme in sinnvoller Weise simulieren lassen und die zudem zu einer höheren Verfahrensrobustheit beiträgt.

Sowohl für das direkte als auch für das Baraff-Verfahren wurde eine Erweiterung um ein SVD-(Singular Value Decomposition-) Verfahren durchgeführt. Diese in Anhang F.2 näher beschriebene Methode zur Lösung eines linearen Gleichungssystems $\mathbf{A}\mathbf{x} = \mathbf{b}$ ist selbst dann sinnvoll anwendbar, wenn die Matrix $\mathbf{A}$ singulär ist (vergl. hierzu auch [Mac90]). Dabei können drei Sonderfälle unterschieden werden, die in EMPHAS wie folgt behandelt werden:

- Keine Lösung

[4] Über die Zeiteffizienz des *LEDA*-Lösungsverfahren wird in der Dokumentation dieser Bibliothek keine Aussage gemacht, so daß nur dieser maximale Zeitaufwand angegeben werden kann.

Wenn es aufgrund einer Überbestimmung keine exakte Lösung des Gleichungssystems gibt, wird ein Lösungsvektor **x** zurückgeliefert, der den Ausdruck $|\mathbf{Ax} - \mathbf{b}|$ minimiert und damit die dichteste Annäherung an die Erfüllung des Systems darstellt.

- Mehrere Lösungen
  Singuläre Matrizen können auch zu einer Vielzahl möglicher Lösungen führen, genauer gesagt zu unendlich vielen. In diesem Fall wird der „kleinste“ der Lösungsvektoren **x** ausgewählt, d.h. derjenige Vektor mit der kleinsten Norm $|\mathbf{x}|$.

- Schlecht konditionierte Lösung
  Mit Hilfe des SVD-Verfahrens können in EMPHAS schlecht konditionierte Fälle, bei denen numerische oder rundungsbedingte Fehler zu völlig falschen Ergebnissen oder sogar zu einem Simulationsabbruch führen würden, in sinnvoller Weise abgefangen werden. Dazu wurde die in Anhang F.2 geschilderte Annulierung von schlecht konditionierten Komponenten unter Berücksichtigung eines vorgebbaren Toleranzwertes realisiert.

Mit Hilfe dieses Verfahrens konnte die Robustheit der Simulation aufgrund der Detektion und Behebung schlecht konditionierter Lösungen erheblich erhöht werden. Die Anwendung dieser Technik für die Behandlung überbestimmter Systeme wird in Abschnitt 6.6.2 beschrieben.

Für die Erweiterung des Baraff-Verfahrens wurde dabei eine spezielle Eigenheit dieses Ansatzes ausgenutzt: die Trennung zwischen primären und sekundären Constraints. Bei rein primären Constraints können überbestimmte Systeme nicht auftreten (sofern die einzelnen Constraints nicht für sich genommen unerfüllbar sind). Das SVD-Verfahren wird daher nur für die Lösung der sekundären Constraints angewandt. Die primären Constraints, deren Anzahl in der Regel weit höher als die der sekundären ist, werden weiterhin durch direkte Methoden behandelt, da sie einen höheren Effizienzgrad als das SVD-Verfahren besitzen ([P+92c]). Diese auf die Eigenschaften des Baraff-Verfahrens abgestimmte Erweiterung ist ein neuer Beitrag für die effiziente und robuste Bestimmung der Lagrange-Faktoren, die einen optimalen Kompromiß zwischen Zeiteffizienz und Robustheit darstellt.

Die Anwendung des SVD-Verfahrens wurde in den Modulen *DirectSVD* und *BaraffSVD* als Erweiterung der beiden Grundverfahren implementiert. Bei allen vier Lösungsverfahren kommt zudem die oben beschriebene automatische Auflösung singulärer Konfigurationen zum Tragen.

Zusammenfassend kann festgestellt werden, daß für das schwierige Problem der Lagrange-Faktoren-Bestimmung insbesondere mit dem *BaraffSVD*-Verfahren eine allgemeine, zeiteffiziente und robuste Umsetzung realisiert werden konnte.

6.3.4 Stabilisierungs- und Integrationsverfahren

6.3.4.1 Stabilisierungsverfahren

Die Grundlage der Constraint-Stabilisierung bildet in EMPHAS das in Abschnitt 3.5.4 erläuterte Baumgart-Verfahren. Dabei wurde sowohl die in [BB88] beschriebene Anwendung als auch die in [W+90] vorgeschlagene modifizierte Variante (vergl. Abschnitt 3.5.4) in Form der Verfahren *Baumgart* bzw. *Spring* realisiert. Die Parameter α und β, die in beiden Verfahren benötigt werden, können dabei zur Programmlaufzeit angepaßt werden. Wie die Erfahrungen mit EMPHAS gezeigt haben, können beide Verfahren die in Abschnitt 3.5.4 aufgeworfenen Stabilitätsprobleme erfolgreich lösen. Die Implementation des in [RE97] beschriebenen Verfahrens wurde dagegen weiterführenden Arbeiten vorbehalten, da es auf dem Lagrange-Formalismus beruht und daher weitreichende Änderungen bei der Aufstellung der Bewegungsgleichungen erforderlich gemacht hätte.

Die Anpassung des Stabilisierungsverfahrens geschieht in EMPHAS über die Modifikation der Parameter α und β von Gleichung (3.20). Der Parameter β wird dabei indirekt über die Vorgabe des Quotienten $\gamma := \beta/\alpha$ festgelegt, da diese Größe leichter interpretierbar ist: Für $\gamma = 1$ resultiert ein kritisch gedämpftes Bewegungsverhalten mit einer exponentiell schnellen Annäherung an die exakte Constraint-Erfüllung[5], während für γ-Werte größer als 1 ein Einschwingvorgang stattfindet, der schneller zum gewünschten Verlauf führt, aber auch als störend empfunden werden kann. Standardmäßig hat γ den Wert 1.

Der Parameter α ist ein Maß dafür, wie schnell die Annäherung an die exakte Lösung geschieht. Für $\gamma = 1$ ist sie in der Regel nach einer Zeit von etwa $\frac{5}{\alpha}$ - $\frac{10}{\alpha}$ Sekunden abgeschlossen. Diesem Parameter kommt daher eine große Bedeutung bei der Modellierung und Bewegungssteuerung zu, da der Prozess der Constraint-Erfüllung ein grundlegender Bestandteil bei der Animationserstellung mit EMPHAS ist. Als Default-Vorgabe wurde für α der Wert 5 gewählt.

Wie auch in [BB88] angemerkt wurde, reduziert sich das Problem der Parameteranpassung bei der Anwendung des kritisch gedämpften Bewegungsverhaltens, das einem γ-Wert von 1 entspricht, auf die Festlegung dieses Parameters α, der sich wiederum als Zeitmaß für den Erfüllungsprozess interpretieren läßt. In EMPHAS konnte somit eine relativ leicht verständliche Festlegung der Stabilisierungsparameter verwirklicht werden, ohne dabei auf die Möglichkeit für einen Einschwingvorgang (für einen γ-Wert größer als 1) zu verzichten.

6.3.4.2 Integrationsverfahren

Als dritte wichtige Teilaufgabe mußte schließlich eine automatische Integration der gewöhnlichen Differentialgleichungen realisiert werden, die nach dem Einsetzen der berechneten Constraint- und Stabilisierungskräfte in die Bewegungsgleichungen entstehen. Dabei stellte sich die Frage, ob hierfür numerische oder symbolische Verfahren zum Einsatz kommen sollten. Der numerische Ansatz hat dabei den großen Vorteil, einfach umsetzbar und allgemein zu sein, während symbolische Verfahren nicht für alle Differentialgleichungen geeignet sind (vergl. Abschnitt F.1). Da insbesondere die Generalität ein wichtiges Ziel bei der Entwicklung von EMPHAS darstellt, wurden daher eine Reihe numerischer Standardverfahren implementiert, mit denen gute praktische Ergebnisse erzielt werden konnten.

Durchaus denkbar wäre allerdings eine *Ergänzung* durch symbolische Verfahren. Durch die Vereinfachung oder in vielen Fällen sogar exakte Lösung von Differentialgleichungen könnte auf diese Weise ein großer Effizienzgewinn erzielt werden. Ein solches Unterfangen ist allerdings nicht ganz einfach umzusetzen und stünde auch außerhalb des Problemkreises dieser Arbeit, da keine Anknüpfungspunkte zu den speziellen Charakteristika der physikalisch basierten Animation im allgemeinen und der LFM im besonderen zu erkennen sind.

Die folgenden numerischen Lösungsverfahren, die in Anhang F.1 genauer erläutert werden, wurden in EMPHAS implementiert:

- Euler-Verfahren
- Runge-Kutta 2. Ordnung
- Runge-Kutta 4. Ordnung
- Runge-Kutta mit variabler Schrittweite

[5] Exaktheit ist hier im Sinne der visuellen Ununterscheidbarkeit vom physikalisch korrekten Bewegungsverlauf zu verstehen.

- Burlisch-Stoer mit variabler Schrittweite

Zur Anpassung der adaptiven Verfahren kann dabei ein Genauigkeitswert vorgegeben werden, der bei der Bestimmung der Schrittweite berücksichtigt wird.

Jedes der Verfahren ist durch bestimmte Vor- und Nachteile ausgezeichnet, die mit dem simulierten System und der gewünschten Zielstellung verknüpft sind. Neben der Ordnung des Schrittweitenfehlers liegt z.B. ein wichtiger Unterschied in der Anzahl der Funktionsaufrufe zur Berechnung der Zustandsableitungen. Diese Auswertung kann bei der physikalisch basierten Animation sehr aufwendig sein, da in dieser Funktion die gesamte Dynamik des Systems festgelegt ist. Bei jedem Aufruf muß also insbesondere die Constraint-Behandlung in Form der Lagrange-Faktoren-Bestimmung durchgeführt werden.

Wie die Erfahrungen mit EMPHAS gezeigt haben, werden die Runge-Kutta-Verfahren zweiter und vierter Ordnung den Anforderungen der Effizienz und Robustheit am besten gerecht. Äußerst eingeschränkt ist vor allem die Anwendung des Euler-Verfahren, da es (bekanntermaßen) sehr schnell instabil wird. Sinnvoll einsetzbar ist es vor allem zu Testzwecken. Das Burlisch-Stoer-Verfahren kann bei sehr gleichmäßigen, insbesondere singularitätsfreien, Bewegungsverläufen zu einem hohen Grad an Genauigkeit führen ([P+92c]). Für die oftmals sehr unregelmäßigen Bewegungen im Rahmen einer Animationserstellung, für die zudem nur ein geringer Exaktheitsgrad erforderlich ist, ist es aber weniger geeignet. Das Runge-Kutta-Verfahren mit variabler Schrittweite liefert hier oftmals bessere praktische Ergebnisse.

Eine genauere Untersuchung dieser Problematik und die Einbindung weiterer Lösungsverfahren wäre sicherlich möglich, hätte aber den Rahmen dieser Arbeit gesprengt. Neben einem möglichen Effizienzgewinn wären von einer solchen Erweiterung keine grundsätzlichen Aspekte für die Zielstellung dieser Arbeit zu erwarten. Stattdessen wurde bei der Umsetzung der Verfahren ein großes Gewicht auf ihre unproblematische Auswahl gelegt, wie im folgenden Abschnitt genauer ausgeführt wird. Eine Evaluierung numerischer Integrationsverfahren im Rahmen der physikalisch basierten Animation findet sich z.B. in [Gre91], [Jun94], [Jun98]. Eine allgemeine Übersicht über die Anwendung dieser Verfahren im Bereich der Technischen Simulation wird in [ESF98] gegeben.

6.3.5 Anpassung der Lösungsverfahren

Die hier beschriebene Simulationsumgebung wurde in EMPHAS in hohem Maße anpaßbar gestaltet. Die Grundlage hierfür bildet die modulare Realisierung der einzelnen Teilverfahren, wie sie oben beschrieben worden ist (für die objektorientierte Implementation dieser Module siehe Anhang A). Im Rahmen der physikalisch basierten Animation ist die Möglichkeit zur Umkonfiguration sehr wichtig, da die Zeiteffizienz und Robustheit der Verfahren sehr unterschiedlich ausfallen kann und stark von den Eigenschaften des jeweiligen simulierten Systems abhängt. Ein großer Vorteil der modularen Umsetzung ist auch die einfache Erweiterbarkeit um zusätzliche Verfahren, was aufgrund der ständigen Weiterentwicklung von zeiteffizienten Lösungstechniken sehr wichtig ist.

Das wichtigste Mittel zur Beeinflussung des Simulationsverfahrens besteht in EMPHAS in der unproblematischen Auswahl des jeweiligen Teilverfahrens zur Bestimmung der Lagrange-Faktoren, zur Anwendung der Stabilisierungstechniken und zur Integration der Differentialgleichungen. Die hierfür entwickelten Module sind in Tabelle 6.2 zusammengefaßt, wobei auch die Anzahl der Systemparameter aufgeführt sind, die in die jeweiligen Verfahren einfließen (sofern solche vorliegen).

Sämtliche dieser Module lassen sich unabhängig voneinander zur Lösung der jeweiligen Teilaufgabe auswählen, d.h. sie können beliebig miteinander kombiniert werden. Diese Auswahl ist nicht fest kodiert, sondern kann über die in Abschnitt 6.5 beschriebene Benutzungsschnittstelle zur Programm-

λ-Bestimmung		Stabilisierung		Integration	
Direct	-	*Baumgart*	2	*Euler*	-
DirectSVD	1	*Spring*	2	*RK2*	-
Baraff	-			*RK4*	-
BaraffSVD	1			*RKad*	1
				BSad	1

Tabelle 6.2: In EMPHAS realisierte Module zur Simulationsdurchführung.

laufzeit vorgenommen werden. Ein Wechsel der Verfahren läßt sich sogar während einer laufenden Simulation durchführen. Intern wird die Bewegungsgenerierung hierzu automatisch gestoppt und mit der neuen Konfiguration wieder gestartet, was für den Benutzer aber unsichtbar bleibt.

Das Auffinden einer *optimalen* Konfiguration ist dagegen sehr zeitaufwendig, da hierfür umfangreiche Testdurchläufe für das jeweilige System durchgeführt werden müssen. Die hier vorgestellten Verfahren ermöglichen aber (abgesehen von den oben besprochenen Einschränkungen bezüglich bestimmter Integrationsverfahren und bestimmter Verfahren zur Bestimmung der Lagrange-Faktoren) in den allermeisten praktischen Fällen in jeder Kombination eine effiziente und robuste Bewegungsgenerierung. Als Voreinstellung sind das *BaraffSVD*-, das *Baumgart*- und das *RK4*-Verfahren aktiviert. Auf ebenso einfache Weise läßt sich die Größe der Systemparameter festlegen, die einen großen Einfluß auf die jeweiligen Verfahren haben können. Auch diese Modifikationsmöglichkeit ist in die Benutzungsschnittstelle von EMPHAS integriert und kann zu jedem Zeitpunkt wahrgenommen werden.

Zusammengenommen wurden somit sehr flexible Anpassungsmöglichkeiten für die Simulationsumgebung realisiert, die sich zur Programmlaufzeit und ohne spezielle Anforderungen an das technische Verständnis des Animateurs anwenden lassen.

Diese weitreichende Konfigurierbarkeit der Simulationsumgebung stellt auch einen Grund für den Verzicht auf eine quantitative Analyse der zeitlichen Effizienz dieser Umsetzung dar, die für eine Vielzahl von Parameter-Kombinationen hätte erfolgen müssen. Ein noch wichtigerer Grund für diesen Verzicht ergibt sich aus der Zielstellung bei der Entwicklung von EMPHAS im Rahmen dieser Arbeit. Es konnte eine allgemeine und robuste Anwendung von Lösungsverfahren realisiert werden, die nach dem derzeitigen Forschungsstand zu den effizientesten Verfahren gehören und eine hinreichend schnelle (d.h. eine interaktive Animationserstellung ermöglichende) Bewegungsgenerierung für typische in der Computeranimation betrachtete Systeme erlauben. Ein hierüber hinausgehender Grad an zeitlicher Effizienz wird für die hier realisierte Umsetzung nicht in Anspruch genommen. Für sämtliche der in den Abbildungen 6.4, 6.7, 6.8, 6.9, 6.10 und 6.11 dargestellten Szenen ermöglicht EMPHAS aber eine stabile Simulation in Echtzeit.

6.4 Automatische Kontaktbehandlung

Die automatische Behandlung von Körperkontakten stellt wie in Abschnitt 2.4 beschrieben eine sehr wichtige Technik zur Erstellung visuell plausibel wirkender Animationen dar. In EMPHAS wurde sowohl eine Simulation von Kollisionen als auch eine automatische Behandlung von Ruhekontakten verwirklicht. Zwei Screenshots, die die Anwendung dieser Techniken zeigen, sind in Abbildung 6.4 dargestellt.

Das Problem der Kontaktbehandlung gliedert sich in drei Teilaufgaben (vergl. Abschnitt 2.4): die Kontaktbestimmung, die Simulation von Kollisionen und die Behandlung von Ruhekontakten. Im

folgenden soll dargestellt werden, wie diese Aufgaben in EMPHAS gelöst wurden und wie die Kontaktbehandlung in die zuvor beschriebene Umsetzung der LFM eingebunden werden konnte.

6.4.1 Realisierung der Kontaktbehandlung

6.4.1.1 Kontaktbestimmung

In EMPHAS wurde mit Hilfe der frei verfügbaren C++ - Bibliothek *I-COLLIDE* ([C+95a], [Man00]) eine automatische Bestimmung von Körperkontakten für beliebige konvexe Polyeder realisiert, die die Detektion von Kontakten und die Bestimmung der jeweiligen Kontaktgeometrie ermöglicht. Neben der reinen Detektion von Körperkontakten, die im Rahmen der physikalisch basierten Animation i.a. *Durchdringungen* der Körper gleichkommen, stellt diese Bibliothek Funktionen zur Bestimmung der Geometrie-Elemente (Ecke, Kante oder Fläche) der beteiligten Körper zur Verfügung, die *vor* dem Kontakt den geringsten Abstand voneinander hatten.

Dabei wird die Detektion eines Kontaktes in *I-COLLIDE* durch die Verwendung achsenparalleler, dynamisch skalierter Bounding Boxes beschleunigt, indem eine Koordinatensortierung vorgenommen wird. Dabei wird berücksichtigt, daß sich die Positionen der Szenenobjekte im Zuge einer physikalischen Simulation von einem Frame zum nächsten nur geringfügig ändern, so daß die Sortierreihenfolge oftmals unverändert bleibt. Der hierfür notwendige Zeitaufwand wird in [C+95a] als linear abhängig von der Anzahl der Körper angegeben. Für die Abstandsbestimmung wird ein Verfahren mit einem erwarteten konstanten Zeitverhalten bezüglich der Anzahl der Polygoneckpunkte eingesetzt[6].

Die Einbindung der *I-COLLIDE*-Bibliothek in EMPHAS wurde dabei so modular gestaltet, daß die Ersetzung oder Erweiterung durch andere Kontaktbestimmungsverfahren, die z.B. auf allgemeinere Objektklasse wie unstrukturierte Polygon-Mengen zugeschnitten sind, problemlos möglich ist (siehe hierzu auch Anhang A).

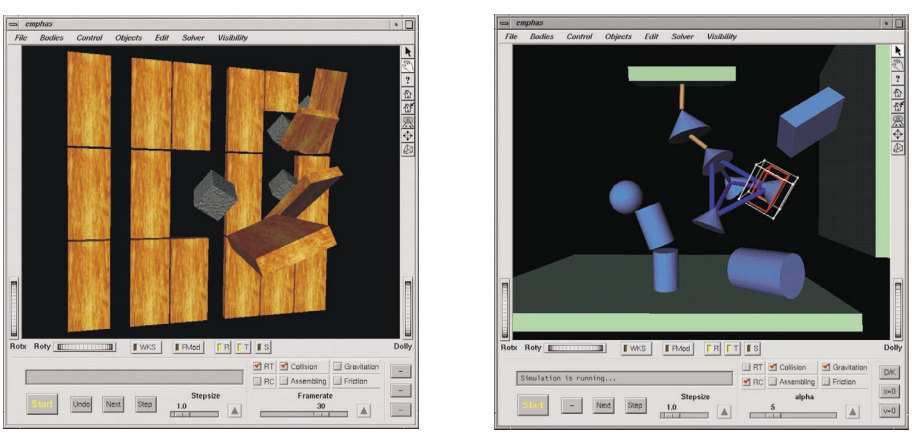

Abbildung 6.4: Zwei Beispiele für die Kontaktbehandlung in EMPHAS. Die linke Szene zeigt die Anwendung der Kollisionssimulation und die rechte die kombinierte Anwendung verschiedener Steuertechniken, bei der insbesondere eine Ruhekontaktbehandlung durchgeführt wird.

[6] Laut [C+95a] ermöglicht die Bibliothek eine Kontaktbestimmung von mehr als 1000 Polyedern mit jeweils mehr als 50 Flächen in weniger als 1/20 Sekunden (auf einer HP 9000/750).

6.4.1.2 Kollisionssimulation

Auf der Basis der Kontaktpunkt- und Kontaktnormalen-Bestimmung konnte in EMPHAS eine physikalisch basierte Kollisionssimulation auf der Grundlage einer Kraftstoß-Berechnung realisiert werden, wie sie in Abschnitt 2.4.2 beschrieben wurde. Zwei Beispielszenen, in denen diese Technik angewandt wurde, sind in Abbildung 6.4 dargestellt. Für die in diese Simulation einfließende Größe der Stoßzahl dient dabei aus Gründen der Einfachheit ein szenenglobaler Parameter, der zwischen 0 und 1 liegt und zur Programmlaufzeit modifiziert werden kann. Eine körperweise Zuweisung erschien wenig sinnvoll, da diese Größe keine echte Materialkonstante darstellt. Als Erweiterung wäre ein Kollisionsmodell, bei der die Stoßzahl für jeden Körperkontakt individuell angepaßt wird, aber durchaus denkbar.

Die Kollisionssimulation wird in EMPHAS nach jedem detektierten Körperkontakt mit negativer Relativgeschwindigkeit (vergl. Abschnitt 2.4) aktiv. In diesem Fall wird (gemäß Gleichung (2.4)) ein Kraftstoß berechnet, mit dem sich neue Werte für die Körpergeschwindigkeiten ergeben, die dazu führen, daß sich die Körper voneinander wegbewegen. Auf diese Weise läßt sich ein sehr plausibel wirkendes Kollisionsverhalten erzielen, ohne auf zeitaufwendige numerische Verfahren zurückgreifen zu müssen.

Ein Problem ergibt sich allerdings aus der unstetigen Änderung der Geschwindigkeiten. Die numerischen Integrationsverfahren zur Lösung der Bewegungsgleichungen setzen nämlich stetige Zustandsänderungen voraus und können daher als Folge dieser Modifikationen fehlerhafte Resultate liefern. In EMPHAS erfolgt daher nach jedem Kollisionsereignis ein Neustart des Integrationsverfahrens mit den berechneten Geschwindigkeiten als neuen Anfangswerten. Diese auch in [Bar95b] vorgeschlagene Vorgehensweise ist in Abbildung 6.5 schematisch dargestellt.

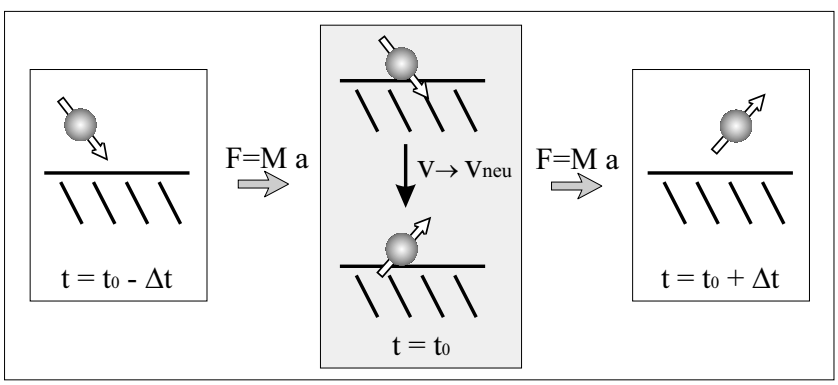

Abbildung 6.5: Einbindung der Kollisionssimulation in EMPHAS. Nach einer Kollision wird die Simulation mit den berechneten Geschwindigkeitswerten neu gestartet.

Ein interessante Alternative, deren Umsetzung den Rahmen dieser Arbeit gesprengt hätte, stellt der in [Mir00b] beschriebene Ansatz dar. Die Grundidee besteht dabei in der Aufhebung der Synchronizität der Simulation, so daß die Detektion eines einzelnen Kollisionskontaktes nicht mehr zu einer Unterbrechung des gesamten Simulationsprozesses führt. Auf diese Weise läßt sich vor allem für die Animation mehrerer Hundert oder Tausend Körper eine wesentlich effizientere Kollisionssimulation realisieren, die sich zudem auch leichter parallelisieren läßt.

6.4.1.3 Ruhekontaktbehandlung

Ein weitaus schwierigeres Unterfangen stellt die Implementation einer automatischen Ruhekontaktbehandlung dar, die ein Durchdringen von relativ zueinander ruhenden Körpern verhindern soll (vergl. Abschnitt 2.4.3). Derartige Konstellationen enthält z.B. die in Abbildung 6.4 rechts dargestellte Szene. Hierzu ist die Bestimmung von *Kompensationskräften* erforderlich, die den äußeren Kräften genau in der Form entgegenwirken, daß Durchdringungen verhindert werden. Die im Rahmen der Kontaktbestimmung gewonnenen Größen des Kontaktpunktes und der Kontaktflächennormalen fließen dabei auch in diese Berechnung ein.

Die in EMPHAS realisierte Lösung basiert auf dem in in Anhang D.2 angegebenen Verfahren, das eine sehr effiziente Ruhekontaktbehandlung ermöglicht. Reibungseffekte wurden dabei allerdings weiterführenden Arbeiten vorbehalten, da ihre Einbindung relativ kompliziert ist (siehe hierzu z.B. [Bar94]).

Die korrekte und effiziente Berechnung der Kompensationskräfte stellt wie in Abschnitt 2.4.3 angesprochen ein schwieriges Problem dar. Die *Einbindung* der Ruhekontaktbehandlung in eine physikalische Simulation ist aber unproblematischer als die der Kollisionssimulation – die berechneten Kompensationskräfte werden einfach dem System als zusätzliche äußere Kräfte hinzugefügt. Ein nachfolgender Neustart des numerischen Lösungsverfahrens ist also nicht erforderlich.

Ein einfaches Beispiel für die Anwendung der Ruhekontaktbehandlung ist in Abbildung 6.6 dargestellt. Die Kugel soll auf dem Boden liegen bleiben, so daß die auf sie wirkende Schwerkraft kompensiert werden muß, um eine Durchdringung zu verhindern. Dies gelingt durch die Berechnung und Anwendung von kompensierenden Kräften, die den Ruhezustand der Kugel aufrechterhalten.

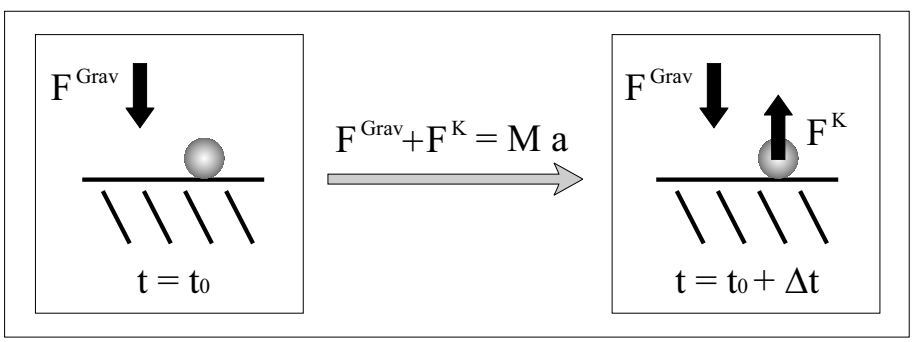

Abbildung 6.6: Einbindung von Ruhekontakten in EMPHAS. Die Simulation wird mit den Kompensationskräften $\mathbf{F}^K$ als zusätzlichen äußeren Kräften weitergeführt.

In EMPHAS wurde somit eine automatische Behandlung von Körperkontakten für beliebige konvexe Polyeder verwirklicht, die durch die physikalisch basierte Simulation von Kollisionen und die Berücksichtigung von Ruhekontakten einen großen Beitrag für die visuelle Plausibilität der generierten Bewegungen liefert.

6.4.2 Kombinationen mit anderen Steuertechniken

Es soll nun die Frage geklärt werden, welche Möglichkeiten zur gemeinsamen Anwendung der hier beschriebene Kontaktbehandlung mit den in EMPHAS realisierten Steuertechniken bestehen. Hierzu muß wieder zwischen der Simulation von Kollisionen und der Behandlung von Ruhekontakten unterschieden werden.

6.4.2.1 Kombinationen mit der Kollisionssimulation

Die Simulation einer Körperkollision wird durch eine instantane Modifizierung der Geschwindigkeiten im Falle eines Kollisionsereignisses durchgeführt. Die Kollisionssimulation kann somit als ein Beispiel für die in Abschnitt 6.2.4.2 eingeführten ereignisbasierten Prozeduren angesehen werden, die ebenfalls eine direkte Änderung der Zustandsgrößen erlauben. Über die Kombination dieses Ansatzes mit anderen Steuertechniken gilt somit das in Abschnitt 5.4.1 gesagte: Sowohl Controller als auch Constraints lassen sich aufgrund der Eigenschaften der LFM problemlos mit der Kollisionssimulation kombinieren. Selbst wenn *während* des Erfüllungsprozesses eines Constraints eine Kollision detektiert werden sollte, kann diese wie gewohnt simuliert werden. Diese Eigenschaft stellt ein wichtiges Beispiel für die Vorteile der LFM im Rahmen eines allgemeinen Animationssystems dar.

6.4.2.2 Kombinationen mit der Ruhekontaktbehandlung

Die Behandlung von Ruhekontakten dient der Vermeidung von Körperdurchdringungen. Bei dem in Abschnitt 2.4.3 beschriebenen Ansatz, der in EMPHAS realisiert wurde, werden die hierfür notwendigen Bedingungen als Ungleichungen formuliert, die sich auf die Zustandskoordinaten der Körper beziehen und damit zu der Klasse der nichtholonomen Constraints gehören. Die Konzepte in Abschnitt 5.4.1 bezüglich der Kombination von Constraints mit anderen Steuertechniken lassen sich daher auch für diesen Fall anwenden. Die Anknüpfungspunkte dieser Constraints sind dabei allerdings nicht fest, sondern ergeben sich aus der jeweiligen Kontaktgeometrie. Aus diesem Umstand und aus der großen Bedeutung des Ziels der Undurchdringlichkeit ergeben sich einige wichtige zusätzliche Aspekte.

Für die Kombination mit Controllern ergeben sich keine Schwierigkeiten, da die durch sie wirkenden Kräfte bei der Ruhekontaktbehandlung berücksichtigt werden. Auch Constraints und ereignisbasierte Prozeduren lassen sich mit diesem Ansatz zur Ruhekontaktbehandlung kombinieren. Als problematisch kann sich dabei allerdings das Auftreten überbestimmter Systeme erweisen. Die in Abschnitt 5.4.2 beschriebenen Ansätze zur Behandlung solcher Konstellationen ließen sich für diesen Fall nur sehr eingeschränkt anwenden, da die Constraints, die der Ruhekontaktbehandlung zu Grunde liegen, in Form von Ungleichungen vorliegen und keinen interaktiv beeinflußbaren Steuerelementen entsprechen. In EMPHAS wurde daher ein anderer Ansatz angewandt, der nun vorgestellt werden soll.

Anstatt das kombinierte System aus Constraints und Ruhekontakten zu lösen, erfolgt *erst* eine Lösung der festgelegten Constraints und *dann* gegebenenfalls eine Korrektur aufgrund der vorliegenden Ruhekontakte. Theoretisch können die Constraints durch dieses Vorgehen verletzt werden, da sie bei der nachträglichen Korrektur nicht explizit berücksichtigt werden [7]. Dieses Problem hat aber eine sehr geringe praktische Relevanz, da die bei der Ruhekontaktbehandlung berechneten Kräfte lediglich eine Durchdringung der Körper verhindern und somit eine rein statische Wirkung haben. Als Vorteil dieser Vorgehensweise wird das Auftreten überbestimmter Systeme, die aus der gemeinsamen Anwendung von Constraints und Ruhekontaktbehandlung hervorgehen, gänzlich vermieden. Der Undurchdringlichkeit wird auf diese Weise eine höhere Priorität zugeordnet, so daß die „Entscheidung“ zwischen diesem Ziel und der Erfüllung von Constraints, die mit dem dafür erforderlichen Bewegungsverhalten nicht kompatibel sind, in eindeutiger Weise (zugunsten der Undurchdringlichkeit) getroffen werden kann. Für die einfache Umsetzung eines allgemeinen Animationssystems hat dieser Ansatz daher große Vorteile.

[7] Dieser Ansatz stellt daher eine spezielle Variante der selektiven Einschränkung von Constraints (vergl. Abschnitt 5.4.2) dar.

In Abbildung 6.4 ist mit der rechts dargestellten Szene ein Beispiel für eine solche kombinierte Anwendung abgebildet. Darin werden sowohl Controller und (zyklische) Constraints, als auch die hier besprochenen Methoden zur Kollisionssimulation und zur Ruhekontaktbehandlung angewandt, um das gewünschte Bewegungsverhalten zu erzielen.

6.5 Graphische Darstellung, Simulationssteuerung und Interaktion

6.5.1 Realisierung der Benutzungsschnittstelle

Ein wichtiges Ziel dieser Arbeit ist der Nachweis, daß sich physikalisch basierte Techniken in einer einfachen und intuitiven Weise für die Animationserstellung einsetzen lassen. Der Benutzungsschnittstelle als Bindeglied zwischen der physikalischen Simulationsumgebung und dem Animateur fällt damit eine besondere Rolle zu. Über sie müssen u.a. die folgenden Aufgaben abgewickelt werden:

- Graphische Darstellung von dreidimensionalen Objekten mit einer Bildwiederholfrequenz von mindestens 20 s^{-1}
- Flexible Festlegung des Betrachterstandpunktes, d.h. der virtuellen Kamera
- Steuerung der Simulation (Start, Stop, Undo usw.)
- Interaktive Manipulation von dreidimensionalen Objekten (Select, Transform, Copy usw.)
- Festlegung von Simulations-, Darstellungs- und Objektparametern
- Anwendung von Techniken zur Modellierung und Bewegungssteuerung

In EMPHAS wurde eine Benutzungsschnittstelle auf der Grundlage der *Open Inventor*-Bibliothek ([SC92b], [Wer94]) realisiert, mit der sich diese Aufgaben in effektiver und leicht erlernbarer Weise lösen lassen. Zwei beispielhafte Screenshots von EMPHAS sind in Abbildung 6.7 dargestellt.

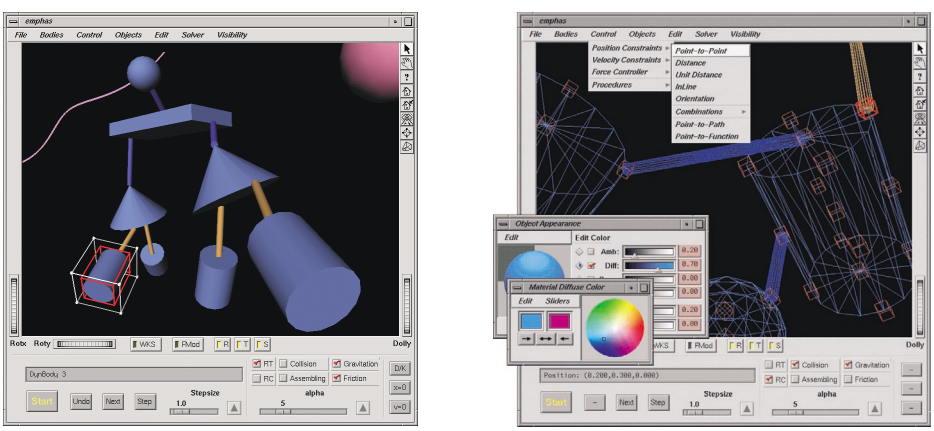

Abbildung 6.7: Zwei Beispielszenen in EMPHAS.

Open Inventor operiert auf der *Open GL*- Bibliothek und basiert auf der Beschreibung dreidimensionaler Szenen durch einen Szenengraphen, in dem nicht nur die darzustellenden Körper, sondern

auch Objekte wie Lichtquellen oder Kameras integriert sind. Auch Interaktionselemente, durch die Teile des Szenengraphen untereinander oder mit Nutzereingaben gekoppelt werden können, sind ein integraler Bestandteil der Szenenbeschreibung. Diese Bibliothek stellt zudem Hilfsmittel zur interaktiven Auswahl und Manipulation dreidimensionaler Objekte zur Verfügung und ermöglicht eine hinreichend zeiteffiziente Darstellung. Mit ihrer Hilfe konnte in EMPHAS auch eine sehr flexible Festlegung des Betrachterstandpunktes verwirklicht werden. Zu diesem Zweck wurde die Xt-Komponente *SoXtExaminerViewer* von *Open Inventor* eingesetzt, die einfach anwendbare Funktionen zur Erfüllung dieser Aufgabe bereitstellt. Entwicklungsumgebungen mit einem ähnlichen Anwendungsbereich werden z.B. in [Z+91], [E+94] und [GB95] beschrieben.

Die in EMPHAS realisierte Benutzungsschnittstelle umfaßt neben der graphischen Darstellung der Grundelemente eine Vielzahl von menübasierten und anderen graphischen Bedienelementen, die eine flexible Steuerung der physikalischen Simulation und eine weitreichende Beeinflussung des dynamischen Systems ermöglichen. Diese Aspekte sollen nun in den folgenden Abschnitten besprochen werden.

6.5.2 Graphische Darstellung der Grundelemente

Für die graphische Darstellung der in Abschnitt 6.2 vorgestellten Grundelemente wurde in EMPHAS auf die von *Open Inventor* bereitgestellten Render-Funktionalitäten zurückgegriffen. Ein flexibel parametrisiertes Rendering-Modul zur Erzeugung fotorealistischer Bilder, wie es in vielen kommerziellen Animationssystemen integriert ist, wurde dagegen nicht realisiert. Neben dem dazu notwendigen Aufwand gibt es hierfür zwei wichtige Gründe. Zum einen liegen die hiermit verknüpften Aspekte außerhalb des Problembereiches dieser Arbeit, da das Rendering (weitestgehend) unabhängig von den Methoden zur Bewegungserzeugung und -steuerung durchgeführt werden kann. Zum anderen bildet das interaktive Arbeiten bei der Modellierung und Bewegungssteuerung einen integralen Bestandteil von EMPHAS und ist nur dann in sinnvoller Weise möglich, wenn sich das Rendern der Szene in weniger als ca. $0.1\,s$ durchführen läßt. Für das fotorealistische Rendering ist dies in aller Regel nicht der Fall.

Als *Nachbearbeitungsschritt* kann diese Möglichkeit aber sehr wichtig sein. In EMPHAS wurde daher eine Exportmöglichkeit realisiert, mit der die erstellten Bewegungsabläufe auf kommerzielle Animationssysteme wie *3D Studio Max* übertragen und dort gerendert werden können. Diese Funktionalität wird in Abschnitt 6.5.5 beschrieben. Eine engere Kopplung mit diesen Systemen wäre dabei im Rahmen weiterführender Arbeiten durchaus denkbar.

6.5.2.1 Erstellung und graphische Darstellung von Kurven

Die in Abschnitt 6.2.2 bei der Vorstellung des *CurveConnectors* angesprochenen Kurven können in EMPHAS als Anknüpfungsobjekte für dynamische Körper eingesetzt werden. Es stellte sich daher das Problem, eine einfache Festlegung dieser Kurven zu ermöglichen sowie diese graphisch darzustellen.

Für die Erstellung der Kurven läßt sich in EMPHAS ein vollständig interaktiv durchführbares Verfahren anwenden: Nach Aktivierung eines entsprechenden Modus kann eine beliebige Anzahl von Kontrollpunkten für eine Kurve erzeugt werden, deren räumliche Lage der jeweiligen Position eines beliebigen, zuvor ausgewählten Körpers entnommen wird (genauer gesagt der Position seines Schwerpunktes). Dabei werden automatisch Bezier-Spline-Kurven festgelegt, für die bei mehr als 3 Kontrollpunkten eine kubische, bei 3 Punkten eine quadratische und bei 2 Punkten eine lineare Ordnung gewählt wird. Diese Art von Kurven wurde gewählt, da sie sich leicht festlegen und manipulie-

ren lassen. Die Koordinaten der Kontrollpunkte lassen sich dabei als Parameter des Kurvenobjektes auch nachträglich ändern.

Dargestellt werden Kurven mit Hilfe der *Open Inventor*-Klasse *SoNurbsCurve*. Um die Sichtbarkeit der Kurven zu garantieren, werden sie dabei mit einer Liniendicke angezeigt, die invariant bezüglich des gewählten Betrachterstandpunktes ist.

6.5.2.2 Graphische Darstellung von Körpern und Konnektoren

Die dynamischen Körper in den Varianten Punktkörper und starrer Körper stellen die wichtigsten Objekte bei der Animationserstellung dar. Die geometrischen Formen der dreidimensionalen starren Körper werden dabei über die in Abschnitt 6.2.1 angesprochene *Shape*-Klasse beschrieben. Neben den geometrischen Grundformen Kugel, Quader, Würfel, Zylinder und Kegel können dabei auch beliebige Polyeder eingesetzt werden.

Den Grundformen werden in EMPHAS entsprechende *SimpleShapes*-Klassen von *Open Inventor* zugeordnet, die eine effektive und einfach durchführbare graphische Darstellung der entsprechenden Körper ermöglichen. Die allgemeinen Polyeder werden mit Hilfe der *SoIndexedFaceSet*-Klasse abgebildet. Für punktförmige Körper wurden als Darstellung Kugeln mit einem festen Radius gewählt.

Da sich die Translations- und Rotationswerte der dynamischen Körper als Ergebnis der physikalischen Simulation ergeben, müssen die aktuellen Zustandswerte nach jedem Simulationsschritt (bzw. nach jedem *n*-ten Simulationsschritt, vergl. Anhang B) an diese Darstellungsobjekte übertragen werden, um ihnen das richtige Bewegungsverhalten zu verleihen. Dieser Vorgang soll im Rahmen dieser Arbeit als *Darstellungsschritt* bezeichnet werden. Auf der anderen Seite müssen Manipulationen der Körper, die in Abschnitt 6.5.4.1 beschrieben werden, an die physikalische Simulation weitergeleitet werden. In Abschnitt A wird beschrieben, wie diese Aufgaben in EMPHAS gelöst werden konnten, ohne die Modularität von Objekten der physikalischen Simulation einerseits und Darstellungsobjekten andererseits verletzen zu müssen.

Auch Konnektoren in Form körperfester Anknüpfungspunkte werden in EMPHAS graphisch dargestellt, was vor allem ihrer einfachen Selektierbarkeit mit Hilfe der Maus dient. Der in Abschnitt 6.6.1 genauer beschriebene Vorgang der Anwendung von Constraints konnte auf diese Weise sehr intuitiv und effektiv gestaltet werden. Für Konnektoren wurde eine Würfel-Darstellung gewählt, da lediglich Raumpunkte zu symbolisieren sind und Kugeln einen größeren Rendering-Aufwand verursachen würden. Beispiele für die graphische Darstellung von Konnektoren finden sich in den Abbildungen 6.8 und 6.10.

6.5.2.3 Graphische Darstellung von Steuerelementen

Eine wichtige Aufgabe stellt auch die graphische Darstellung der Steuerelemente dar. Zum einen dient dies der leichteren Selektierbarkeit der Elemente, die sich auf diese Weise durch ein direktes Anwählen mit der Maus durchführen läßt (vergl. Abschnitt 6.5.4.1). Bei Constraints und Controllern kann außerdem durch die graphische Darstellung angezeigt werden, *welche* Körper oder Anknüpfungspunkte direkt durch dieses Steuerelement beeinflußt werden. Diese Information ist bei der Modellierung und Bewegungssteuerung natürlich sehr wichtig. Ein geeignet dargestelltes Steuerelement kann zudem im Rahmen der Animationserstellung als ein eigenständiges, zu dynamischen Körpern gleichberechtigtes Szenenobjekt betrachtet werden. Ein *SpringForce*-Connector, der die Interpretation einer Feder-Verbindung hat, wird in EMPHAS z.B. als langgestreckter Zylinder dargestellt, der von einem Anknüpfungspunkt zum anderen verläuft. Er kann somit ganz intuitiv als ein zusätzlich

eingeführtes Objekt mit der Interpretation einer Feder interpretiert werden.

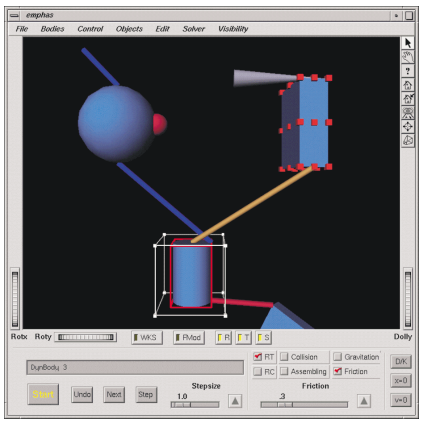

Abbildung 6.8: Eine Beispielszene in EMPHAS mit verschiedenen Steuerelementen.

Die Darstellung als Zylinder mit einem kleinen, festen Radius wurde in EMPHAS auch für die meisten anderen Constraint- und Controller-Typen gewählt, da diese Form eine gute Symbolisierung für allgemeine *Verbindungen* liefert. Die räumliche Position der Zylinderenden (als Mittelpunkte der Grundflächen) werden dabei bei jedem Simulationsschritt den jeweiligen Anknüpfungspunkten angepaßt. *PointToFunction*-Constraints und Geschwindigkeits-Constraints bezüglich nur eines dynamischen Körpers entsprechen keinen Verbindungen und werden daher als Kugeln am Ort des jeweiligen Anknüpfungspunktes dargestellt. Ein *VectorForce*-Controller wird durch einen Kegel symbolisiert, der in die Richtung der Kraftrichtung zeigt. Controller, holonome Constraints und Geschwindigkeits-Constraints können zudem durch verschiedene Farbgebungen voneinander unterschieden werden. Über die Farbgebung wird auch angezeigt, ob der Constraint aktiv ist – inaktive Constraints werden grau dargestellt.

Beispiele für diese Art der Darstellung finden sich in Abbildung 6.8. Neben den dynamischen Körpern sind darin ein Controller (gelb), zwei holonome Constraints (blau), zwei Geschwindigkeits-Constraints (rot) und ein inaktiver *VectorForce*-Controller (grau) dargestellt.

Das der Schwerkraft entsprechende Kraftfeld, das Kraftfeld *FrictionForceField* und die Prozedur *DistanceProcedure* haben im Gegensatz zu den oben besprochenen Elementen keinen Bezug zu einzelnen Anknüpfungs- oder Raumpunkten. Sie werden daher in EMPHAS nicht graphisch dargestellt. Das Kraftfeld *RadialForceField* übt dagegen einen vom Kraftzentrum abhängigen Einfluß auf die dynamischen Körper aus und wird daher durch eine Kugel mit einem festen Radius repräsentiert. Auf diese Weise läßt sich die räumliche Position des Feldzentrums leicht erkennen und interaktiv manipulieren.

6.5.2.4 Einflußmöglichkeiten auf die graphische Darstellung

In EMPHAS wurde eine Reihe von Einflußmöglichkeiten für die Darstellung der Grundelemente verwirklicht. Mit Hilfe der Xt-Komponente *SoXtMaterialEditor* lassen sich z.B. die graphischen Oberflächeneigenschaften einzelner Darstellungsobjekte festlegen, wobei die Farbwerte für die verschiedenen Reflexionsarten und die Transparenz über intuitive Bedienelemente eingestellt werden können. Nicht nur die Darstellung von dynamischen Körpern, sondern auch das Erscheinungsbild von Constraints, Controllern, Kraftfeldern und Kurven läßt sich auf diese Weise individuell anpassen. Für dynamische Körper wurde zudem die Verwendung einer als Bitmap vorliegenden Textur ermöglicht (eine solche Anwendung ist z.B. in Abbildung 6.4 zu sehen).

Die graphische Darstellung von Konnektoren ist für die Festlegung von Constraints, die sich auf derartige Anknüpfungspunkte beziehen sollen, unverzichtbar. Ansonsten ist sie aber meistens unerwünscht, insbesondere während einer laufenden Simulation, wo sie auch zu Einbußen der Render-Performance führen kann. Aus diesem Grund wurde eine Möglichkeit zur Umschaltung der Konnektoren-Sichtbarkeit realisiert, die sich entweder auf die Konnektoren eines ausgewählten Körpers oder (falls kein Körper selektiert wurde) auf die Konnektoren sämtlicher in der Szene befindlichen Körper bezieht. Zudem kann festgelegt werden, daß die Konnektoren während einer Simulation automatisch

ausgeblendet werden, um nach dem Ende des Simulationslaufes wieder sichtbar zu werden, was in EMPHAS als standardmäßige Vorgabe genutzt wird. Die ursprüngliche Sichtbarkeitsfestlegung, d.h. die Wahl der Körper, für die die Konnektoren sichtbar sein sollen, bleibt dabei erhalten. Schließlich kann sich auch die graphische Darstellung von Controllern und Constraints in vielen Fällen als störend erweisen. Daher wurde auch für diese Elemente eine einfache Möglichkeit zur Umschaltung der Sichtbarkeit zur Verfügung gestellt.

6.5.3 Simulationssteuerung

Die Körperbewegungen werden in EMPHAS mit Hilfe einer physikalischen Simulation generiert. Die geeignete Aktivierung, Deaktivierung und Anpassung der Simulation stellt daher eine sehr wichtige Aufgabe dar, zumal sämtliche Modellier-, Steuer- und Interaktionsvorgänge zur Simulationszeit durchgeführt werden können. Die in EMPHAS realisierten Konzepte für eine solche Steuerung sollen nun vorgestellt werden.

6.5.3.1 Festlegung der Simulationsparameter

In Anhang B werden die Simulationsparameter *Simulationsrate*, *Oversampling* und *Framerate* eingeführt und ihre gegenseitigen Abhängigkeiten erläutert. Die Festlegung dieser Parameter bildet die Grundlage für eine flexible Einflußnahme auf die Durchführung der Simulation, wie nun genauer erläutert wird.

In EMPHAS stellen die Simulationsrate R_S und der Oversampling-Wert n_O frei vorgebbare Parameter dar, die zur jeder Zeit geändert werden können. Standardmäßig haben sie die Werte 30 bzw. 1. Eine Erhöhung von R_S trägt zu einem stabileren Lösungsverhalten bei, führt aber auch zu einem größeren Simulationsaufwand, so daß sämtliche Körperbewegungen langsamer erscheinen. Wenn der gesamte zeitliche Ablauf schneller gestaltet werden soll, kann der Wert für n_O vergrößert werden, was vor allem bei einer zeitaufwendigen graphischen Darstellung der Objekte (z.B. beim Vorhandensein von Texturen) sinnvoll ist. Dabei wird allerdings die Framerate verringert, was zu „ruckeligen“ Bewegungen führen kann (vergl. Anhang B)[8].

In EMPHAS wurde außerdem ein jederzeit aktivierbarer Echtzeit-Modus verwirklicht. Um ein Echtzeit-Verhalten erzielen zu können, muß die Simulationsrate während der Simulation automatisch angepaßt werden . Zu diesem Zweck wird die zeitliche Dauer jedes Simulations- und Darstellungsschrittes gemessen und R_S gemäß Gleichung (B.1) aus Anhang B angepaßt. Dabei wurde allerdings eine untere Schranke $R_S^{min} = 10$ festgelegt, um ein Mindestmaß an Simulationsrobustheit zu garantieren. Fällt die zur Echtzeit notwendige Simulationsrate unter diesen Wert, wird R_S gleich R_S^{min} gesetzt und somit das Echtzeit-Verhalten aufgehoben.

Bei aktiviertem Echtzeit-Modus wird zudem der Oversampling-Wert n_O automatisch angepaßt. Hierzu läßt sich ein Parameterwert R_F^{opt} festlegen, der die gewünschte Framerate vorgibt. Der Wert von n_O wird dann so klein gewählt, daß diese vorgegebene Framerate gerade nicht unterschritten wird. Die Information, welche Werte für R_S und n_O zur Aufrechterhaltung der Echtzeit ermittelt wurden, wird dabei während der Simulation angezeigt. Da n_O nicht kleiner als 1 sein kann, ist dieses Ziel allerdings nicht immer erfüllbar. Im Vergleich zu einem konstanten n_O-Wert gleich 1 erreicht man auf diese Weise aber, daß unnötig hohe Frameraten, die zu überflüssigen Darstellungsschritten und zu einer Verlangsamung des Simulationsablaufes führen würden, vermieden werden. Die Größe

[8] Dabei scheint es nahe zu liegen, durch eine einheitliche Skalierung aller Körpergeschwindigkeiten das gleiche Simulationsergebnis mit einem schnelleren Ablaufverhalten zu erzielen, ohne R_S verringern zu müssen. Dieses Vorgehen verspricht aber keinen Erfolg, da die auftretenden Ungenauigkeiten und Instabilitäten proportional zur Änderungsrate des Systems und damit auch proportional zur Größe der Körpergeschwindigkeiten sind.

R_F^{opt} stellt somit neben der Simulationsrate R_S und dem Oversampling-Wert n_O den dritten vorgebbaren Simulationsparameter dar. Mit diesen relativ leicht interpretierbaren Größen konnte somit in EMPHAS eine sehr flexible Beeinflussung der Simulation verwirklicht werden. Die eigentliche Steuerung der Simulation soll nun als nächstes besprochen werden.

6.5.3.2 Schrittweise Simulation

Um einen Simulationslauf mit einer fest definierten Länge durchführen zu können, stehen die Bedienelemente *Next* und *Step* zur Verfügung. Mit *Next* wird genau ein Simulationsschritt ausgeführt. Die Schrittweite hängt dabei von der aktuell gültigen Simulationsrate ab. Über das Bedienelement *Step* läßt sich ein Simulationslauf starten, dessen Länge bezüglich der Systemzeit durch den interaktiv veränderbaren Parameter *Stepsize* gegeben ist. Bei aktiviertem Echtzeit-Modus ist auf diese Weise auch eine direkte Realzeitvorgabe möglich.

6.5.3.3 Laufende Simulation

Die interessanteste Möglichkeit zur Simulationssteuerung stellt die sogenannte *laufende Simulation* dar. Über einen kombinierten *Start/Stop*-Button kann die Simulation jederzeit gestartet und wieder angehalten werden. Im Gegensatz zur schrittweisen Simulation bleiben dabei während der Simulation sämtliche Interaktionsmöglichkeiten von EMPHAS erhalten. Die Bewegungsgenerierung wird bei einem solchen Eingriff automatisch angehalten und dann wieder automatisch gestartet. Dieses interne Verhalten bleibt aber unsichtbar, so daß sich sämtliche Interaktionen ohne sichtbare Beeinträchtigung des Bewegungsablaufes durchführen lassen.

Die laufende Simulation hat daher nicht nur für die eigentliche Bewegungserzeugung, sondern gerade auch für reine Modelliervorgänge eine fundamentale Bedeutung. Nur durch die unterbrechungsfreie Fortführung der Simulation läßt sich die stetige Constraint-Erfüllung der LFM ausnutzen, die eine große Hilfe für den Animateur darstellt (vergl. Abschnitt 6.6.1.3). Außerdem lassen sich auf diese Weise Änderungen der Systemparameter, Objektmanipulationen, Modelliervorgänge und Steueraktionen während einer laufenden Simulation völlig interaktiv und ohne Aufgabe der dynamischen Integrität, d.h. innerhalb des Prozesses der physikalisch basierten Bewegungsgenerierung, durchführen, wie in Abschnitt 6.5.4.1 genauer ausgeführt wird.

6.5.3.4 Undo-Konzept

Da das Simulationsergebnis natürlich nicht immer dem gewünschten Verlauf entspricht, ergibt sich oftmals die Notwendigkeit, zu dem Systemzustand vor dem Simulationsstart zurückkehren zu können. In EMPHAS wurde daher eine Undo-Funktion realisiert, die diese Aufgabe bewältigt und sowohl für die schrittweise als auch für die laufende Simulation angewandt werden kann. Falls während einer laufenden Simulation Aktionen durchgeführt werden, die eine Initialisierung des Systems notwendig machen, z.B. das Erzeugen oder Löschen von Körpern, wird dabei die neue Konfiguration automatisch als Ausgangszustand gespeichert.

6.5.4 Interaktionstechniken

In diesem Abschnitt sollen die in EMPHAS realisierten Interaktionstechniken vorgestellt werden. Zunächst werden die Möglichkeiten zur interaktiven Manipulation der Grundelemente beschrieben. Anschließend werden zwei Techniken erläutert, die einen direkten Einfluß auf die Bewegungen der dynamischen Körper haben und auch in [Wag00] beschrieben wurden.

6.5.4.1 Interaktive Manipulation der Grundelemente

Eine sehr wichtige Aufgabe bei der Erstellung von Computeranimationen im allgemeinen und der physikalisch basierten Animation im besonderen ist die interaktive *Manipulation* von Szenenobjekten. Die in der Szene vorhandenen dynamischen Körper sollten z.B. in möglichst einfacher Weise transformiert werden können. Auch die mit einem Objekt verknüpften Parameterwerte müssen sich in geeigneter Form verändern lassen. Die in EMPHAS realisierten Techniken zur Erfüllung dieser Aufgaben sollen nun vorgestellt werden.

Für die Objektinteraktion wurde in EMPHAS ein sehr modulares Konzept realisiert. Dynamische Körper, Konnektoren, Kurven und Steuerelemente werden als allgemeine Szenenobjekte behandelt, die von vielen Interaktionstechniken in gleicher Weise angesprochen werden können. Auf diese Weise konnte eine sehr einheitliche und einfach anwendbare Objektinteraktion verwirklicht werden.

Selektion. Der erste Schritt bei der Manipulation von Objekten besteht in der Auswahl des gewünschten Objektes. In EMPHAS wurde eine solche Selektionsmöglichkeit auf der Basis der von *Open Inventor* bereitgestellten *Picking*-Funktionalität realisiert, das die Auswahl über das Anklicken mit der Maus ermöglicht. Um auch die Selektion von nicht sichtbaren, z.B. verdeckten, Objekten zu gewährleisten, kann die Auswahl auch über eine Liste der aktuell vorhandenen Objekte eines bestimmten Typs erfolgen. Diese beiden Techniken sind für Körper, Konnektoren vom Typ *BodyConnector* (nur über Picking[9]), Kurven, Prozeduren (nur listenbasiert), Kraftfelder vom Typ *RadialForceField*, Controller und Constraints in gleicher Weise anwendbar.

Modifikation der Objektparameter. Jedes Objekt in EMPHAS ist durch eine Reihe von Parametern ausgezeichnet. Das Verhalten von punktförmigen und starren Körpern wird z.B. von der Größe der Masse beeinflußt und auch Constraints und anderen Objekten sind bestimmte Objektparameter zugeordnet, die auf einfache Weise veränderbar sein sollten. In EMPHAS wurde das Editieren sämtlicher Parameterwerte eines Objektes über ein entsprechendes Eingabefenster ermöglicht. Diese Modifikation kann auch zur Simulationszeit durchgeführt werden und geschieht für alle Objekttypen einheitlich. Auf diese Weise lassen sich auch die Translations- und Rotationsgeschwindigkeiten von dynamischen Körpern jederzeit ändern, die einen Teil der Zustandsbeschreibung darstellen.

Transformation. Für die dynamischen Körper (und das Kraftfeld *RadialForceField*) wurde in EMPHAS eine interaktiv durchführbare Transformationsmöglichkeit verwirklicht. Grundlage dieser Interaktion ist das *Dragger*-Konzept von *Open Inventor*. Durch Anklicken eines Körpers wird ein sogenanntes Handle-Objekt erzeugt, das den Körper umschließt und über bestimmte Interaktionselemente verfügt, die mit der Maus angesteuert werden können. In EMPHAS wird dazu ein Objekt vom Typ *TransformBoxDragger* eingesetzt. In den Screenshots von Abbildung 6.4 (rechts), 6.7 (links) und 6.8 ist dieses Objekt als weißer, quaderförmiger Rahmen um einen dynamischen Körper zu erkennen. Auf diese Weise lassen sich die Translation, Rotation und Skalierung von Körpern und die Translation von Kraftfeldern durchführen. Die Transformationstypen können dabei auch einzeln deaktiviert werden, um unbeabsichtigte Eingaben zu vermeiden.

Sämtliche Transformationsvorgaben beziehen sich dabei auf das körperfeste Koordinatensystem. Bei starren Körpern wird außerdem nach einer Skalierung eine automatische Korrektur der Gesamtmasse und des Trägheitstensors vorgenommen, da von einer konstanten Dichte ausgegangen wird (vergl. Abschnitt 6.2.1). Möglich ist auch die numerische Festlegung der Transformationen über ein entsprechendes Eingabefenster und die einfache Rücksetzung der Werte auf den Ursprungszustand.

Das Handle-Objekt wird normalerweise zusammen mit dem zugehörigen Körper rotiert, wodurch

[9] Bei starren Körpern ist der Konnektor am Körperschwerpunkt i.a. nicht sichtbar. Er kann aber in EMPHAS über die Selektion des Körpers selber angewählt werden.

Translationen auf das Koordinatensystem des Körpers bezogen werden. Oftmals ist es aber wünschenswert, eine Translation entlang der Achsen des Weltkoordinatensystems oder eines anderen Koordinatensystems vornehmen zu können. In EMPHAS wurde daher eine Umschaltmöglichkeit realisiert, mit der sich dieses Ziel realisieren läßt. Das Handle-Objekt folgt dann weiterhin dem zugehörigen Körper, seine Rotation ist von diesem aber abgekoppelt und kann frei gewählt werden (in diesem Modus kann das Handle-Objekt somit nicht zur Rotation des Körpers eingesetzt werden). Die Transformation bezieht sich dann auf das Koordinatensystem des Handle-Objektes, das standardmäßig mit dem Weltkoordinatensystem ausgerichtet wird, und kann auf diese Weise sehr flexibel festgelegt werden.

Automatische Verrückung. In bestimmten Fällen kann es außerdem nützlich sein, die Position oder Rotation eines Körpers um einen zufälligen Wert zu verändern. Durch dieses Vorgehen lassen sich z.B. singuläre Konfigurationen in nichtsinguläre verwandeln (vergl. Abschnitt 5.3.3). In EMPHAS besteht daher zum einen die Möglichkeit, eine kleine, (pseudo-) zufällige Verrückung bezüglich der Translation und der Rotation für die dynamischen Körper der Szene durchführen zu lassen. Zum anderen können den Körpern zufällige und für jeden Körper verschiedene Werte für den Rotationszustand zugewiesen werden. Die Translationszustände bleiben bei dieser zweiten Variante, die im Gegensatz zur ersten zu deutlich sichtbaren Veränderungen führt, unangetastet.

Kopieren und Löschen von Objekten. Das Kopieren von kompletten Objekten stellt eine oft benötigte Funktionalität bei der Animationserstellung dar. In EMPHAS wurde daher eine Möglichkeit geschaffen, über die Aktionen *Copy*, *Cut* und *Paste* das Kopieren und Einfügen von Körpern und Kraftfeldern mitsamt ihrer zugehörigen Parameterwerte zu ermöglichen. Über einen *Remove*-Befehl läßt sich zudem jedes Objekt aus der aktuellen Szene entfernen. Dabei werden Constraints, die mit einem zu löschenden Objekt verknüpft sind, ebenfalls automatisch gelöscht, um keinen undefinierten Zustand entstehen zu lassen.

Interaktionen zur Simulationszeit. Sämtliche der oben vorgestellten Interaktionstechniken lassen sich auch zur Simulationszeit durchführen. Die Möglichkeit für einen solchen Eingriff wurde bereits in Abschnitt 6.5.3.3 angesprochen. Mit Hilfe der interaktiven Objekttransformation lassen sich z.B. die Körperpositionen während einer laufenden Simulation direkt beeinflussen. Durch einen solchen Eingriff können Constraints verletzt werden, da eine rein benutzerspezifizierte und von der Szenendynamik unbeeinflußte Interaktion vorliegt. Aufgrund der in Abschnitt 4.2 besprochenen Eigenschaften der LFM können in EMPHAS aber auch Systeme, die durch eine starke Verletzung der festgelegten Constraints gekennzeichnet sind, problemlos behandelt werden. Nach jedem Interaktionsvorgang reagiert das System auf die neuen Körperkonfigurationen und erreicht nach kurzer Zeit wieder einen Zustand, bei dem die Constraints erfüllt sind, ohne dabei unnatürlich wirkende Bewegungssprünge zu verursachen. Das gleiche gilt für die Editierung von Objektparametern und sogar für die Erstellung und Löschung von Körpern, Constraints und anderen Objekten, die ebenfalls zur Simulationszeit möglich sind.

Diese Art der Einflußnahme ist in kommerziellen Animationssystemen, die auf dem Verfahren des Keyframing beruhen, nicht anzutreffen. Sie stellt einen wichtigen Bestandteil der Interaktivität von EMPHAS dar. Die einheitliche Verwendung der interaktiven Manipulationstechniken sowohl für die Modellierung, die normalerweise vor der Bewegungssimulation geschieht als auch für die Steuerung der Bewegungen während der Simulation erleichtert den Umgang mit dem System und damit den Prozess der Animationserstellung erheblich. Die modulare Konzeption der Szenenobjekte trägt dabei ihren Teil zu dem hohen Maß an Einheitlichkeit und Intuitivität bei.

6.5.4.2 Kinematisches Körperverhalten

Oftmals ist es wünschenswert, daß ein Körper unbeeinflußt von äußeren Kräften im Ruhezustand verbleibt oder sich gleichmäßig fortbewegt. Man möchte dem Körper also ein rein *kinematisches* Verhalten verleihen, das nur durch die festgelegten Zustands- und Geschwindigkeitswerte festgelegt ist. Auf diese Weise kann er z.B. selbst in einem Gravitationsfeld an einem bestimmten Raumpunkt verharren.

In EMPHAS wurde die Möglichkeit geschaffen, für jeden Körper zwischen diesem kinematischen und dem üblichen dynamischen Verhalten umschalten zu können. Dieser Wechsel ist auch jederzeit während einer laufenden Simulation möglich. Dabei wurde eine wichtige Anforderung erfüllt: Die mit dem Körper verknüpften Constraints bleiben auch nach der Umschaltung erhalten. Auf den ersten Blick erscheint dies wenig sinnvoll, denn kinematische Körper werden nicht mehr durch Constraints beeinflußt. Es gibt aber zwei Gründe für dieses Vorgehen. Zum einen wären die Constraints ansonsten nach dem Zurückschalten zum dynamischen Verhalten gelöscht und müßten wieder neu festgelegt werden. Vor allem ist auf diesem Weg aber eine sehr flexible Möglichkeit geschaffen worden, durch die Translation und Rotation des kinematischen Körpers die Bewegungen derjenigen Körper zu steuern, die über Constraints mit ihm verknüpft ist. Diese Art der Beeinflussung ist sehr intuitiv, da der kinematische Körper wie ein fester Bezugspunkt für den Constraint wirkt. Bei der Animationserstellung mit EMPHAS wird diese Umschaltmöglichkeit daher sehr häufig genutzt.

Ein kinematischer Körper bleibt zudem Bestandteil der Kollisions- und Ruhekontaktbehandlung. Zwar werden die auf ihn selber wirkenden Kräfte ignoriert, seine eigene Wirkung im Falle eines Kontaktes bleibt aber erhalten[10].

Die Wirkungsweise dieser Umschaltmöglichkeit läßt sich an einem Beispiel verdeutlichen. In Abbildung 6.9 ist eine Szene dargestellt, in der zwei Körper, die durch einen *Distance*-Constraint miteinander verbunden sind und die gleiche Anfangsgeschwindigkeit besitzen, auf ein Hindernis stoßen. In der unteren Sequenz wurde bei gleichen Anfangsbedingungen für den kugelförmigen Körper ein kinematisches Verhalten eingestellt. Er bleibt daher im Zustand der gleichförmigen Bewegung, ohne seine Wirkung als Constraint-Bezugspunkt für den anderen Körper zu verlieren.

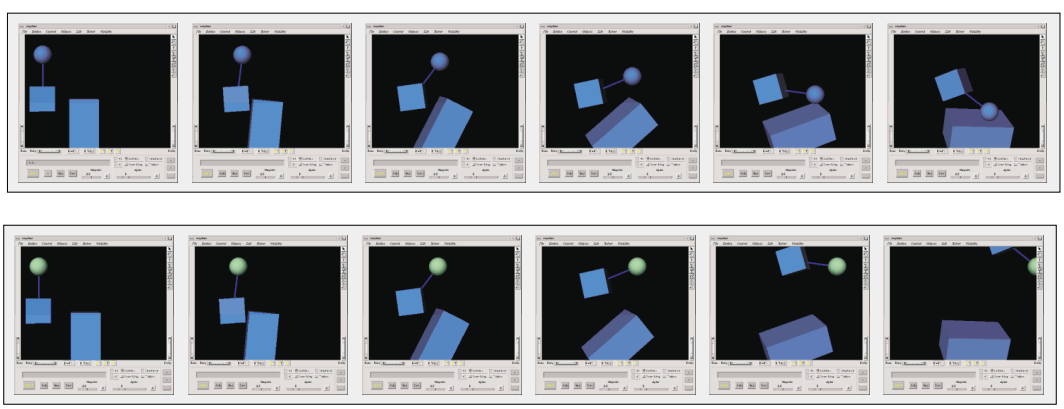

Abbildung 6.9: Wirkung des kinematischen Körperverhaltens. In der unteren Sequenz wurde für die Kugel als einzige Änderung zur oberen Sequenz ein kinematisches Körperverhalten festgelegt.

Eine weitere Anwendung dieser Umschaltmöglichkeit wird in Abschnitt 6.6.3 als Technik zur Unterdrückung von Bewegungsartefakten vorgestellt.

[10] Im Rahmen der Kollisionssimulation werden kinematischen Körpern dazu wie dynamische Körper mit einer unendlich großen Masse (genauer gesagt einem inversen Trägheitstensor gleich der Nullmatrix) behandelt.

Um das kinematische Verhalten zu ermöglichen, wird ein Constraint, der auf zwei Körper wirkt, bei der Umschaltung eines dieser Körper so modifiziert, daß seine partiellen Ableitungen nicht mehr in die Simulationsroutinen einfließen (genauer gesagt in die Bestimmung der Jacobi-Matrix **J**) [11]. Die entsprechenden Konnektoren werden also wie konstante Raumpunkte behandelt, deren Positions- und Geschwindigkeitswerte aber aus der Lage des jeweiligen Körpers resultieren und laufend aktualisiert werden.

6.5.4.3 Interaktive Kraftanwendung

Eine in EMPHAS realisierte Interaktionstechnik, die nur zur Simulationszeit angewandt werden kann, ist die direkte Anwendung von Kräften. Durch diese Technik können zuvor ausgewählte Körper sowohl verschoben als auch rotiert werden. Im Gegensatz zu der in Abschnitt 6.5.4.1 beschriebenen direkten Manipulation der Körpertransformationen werden dabei keine Constraints verletzt, da die vom Benutzer festgelegten Kräfte bei der Simulation berücksichtigt werden. Die Erfüllung der vorhandenen Constraints bleibt damit gewahrt, was oftmals ein sehr wünschenswertes Verhalten darstellt.

Die direkte Kraftanwendung stellt außerdem eine sehr intuitive Interaktionstechnik dar. Sie kommt einem instantanen „Anstoßen“ des ausgewählten Körpers gleich, der dann entsprechend der Stärke und Richtung dieser Kraft in eine gleichförmige Bewegung versetzt wird (sofern er nicht durch andere Einflüsse daran gehindert wird). Durch diese direkte visuelle Rückkopplung kann sehr schnell erlernt werden, wie sich die Kraftanwendungen auswirken und wie sich mit ihnen das gewünschte Bewegungsverhalten erzielen läßt.

Realisiert wurde diese Technik wie schon die direkte Transformationsmanipulation mit Hilfe des *Open Inventor*-Interaktionselementes *TransformBoxDragger*. Während einer Simulation läßt sich dabei jederzeit zwischen den beiden Interaktionsmodi umschalten. Im „Kraftmodus“ kann das Handle-Objekt mit Hilfe der Maus in eine beliebige Richtung verschoben werden (die Verknüpfung mit dem zugehörigen Körper wird dabei während dieser Aktion aufgelöst). Nach Beendigung des Ziehvorgangs wird dann eine Kraft bestimmt, deren Stärke und Richtung der Differenz von alter und neuer Handle-Position entspricht und am Schwerpunkt des ausgewählten Körpers angreift. In analoger Weise geschieht die Festlegung von Drehmomenten, die eine Rotation des Körpers relativ zu seinem Schwerpunkt verursachen.

Mit Hilfe der hier vorgestellten Interaktionstechniken wurde eine flexible und intuitive Einflußnahme auf die Szenendynamik ermöglicht. Sehr wichtig ist aber auch die Anbindung von EMPHAS an kommerzielle Animationssysteme, um die von ihnen bereitgestellten Funktionalitäten nutzen zu können. Eine solche Anbindung wird nun vorgestellt.

6.5.5 Import- und Export-Möglichkeiten

Es sollen nun die in EMPHAS realisierten Möglichkeiten zum Import und Export von Szenendaten beschrieben werden. Neben dem Abspeichern und Laden über ein internes Dateiformat wurde dabei auch ein Importmöglichkeit und ein Export von Geometrie- und Bewegungsdaten verwirklicht.

[11] Constraints, die sich nur auf einen Körper beziehen, werden dagegen automatisch deaktiviert, solange für den Körper ein kinematisches Verhalten festgelegt ist.

6.5.5.1 Speichern und Laden dynamischer Szenen

Um das Abspeichern und Laden von Szenen, die mit EMPHAS erstellt wurden, zu ermöglichen, wurde ein internes Dateiformat entworfen, mit dem nicht nur die in der Szene befindlichen Körper, sondern auch Kurven, sämtliche Steuerelemente und Verknüpfungen zwischen diesen Objekten beschrieben werden können. Neben den geometrischen werden dabei auch die dynamischen Eigenschaften der Körper und alle Parameterwerte, die den jeweiligen Objekten zugewiesen wurden, erfaßt. Nur mit Hilfe dieser Informationen läßt sich die modellierte Szenendynamik rekonstruieren.

Über dieses Format läßt sich die aktuelle Szene in EMPHAS jederzeit mit den üblichen *Save*-, *SaveAs*- und *Open*-Funktionalitäten abspeichern und zu einem späteren Zeitpunkt wieder aufnehmen. Als zusätzliche Möglichkeit kann zudem eine zuvor abgespeicherte mit der aktuellen Szene *gemischt* werden. Die jeweiligen Körper, Kurven und Steuerelemente werden dann (mitsamt ihren Verknüpfungen) der aktuellen Szene hinzugefügt, anstatt diese zu ersetzen. Diese Funktionalität läßt sich vor allem für die Speicherung und Übertragung von einzelnen Körpern oder autonomen Untersystemen ausnutzen.

6.5.5.2 Import von Polyedern

In Abschnitt 6.2.1 wurde mit dem *PolyederShape* ein *Shape*-Objekt vorgestellt, das sich zur Beschreibung beliebiger Polyeder eignet. In EMPHAS lassen sich dynamische Körper mit dieser geometrischen Form über den Import aus einer VRML-Datei erzeugen. Dieses Dateiformat wurde vor allem aufgrund seiner Verbreitung gewählt – fast alle der kommerziell verfügbaren Modellierungs- und Animationssysteme erlauben einen Geometrie-Export dreidimensionaler Körper in diese Beschreibungssprache. Die Funktionalität dieser Systeme zur geometrischen Modellierung läßt sich auf diese Weise in EMPHAS für den Einsatz eines sehr allgemeinen Geometrietyps ausnutzen.

Für die Einbindung solcher Körper in die Simulationsumgebung müssen allerdings einige zusätzliche Größen berechnet werden. Neben den Normalenvektoren der einzelnen Polygone, die für die Durchführung der Kontaktbehandlung benötigt werden, müssen das Volumen, aus dem sich die Gesamtmasse ableitet, und der Trägheitstensor des Körpers bekannt sein. Diese Größen werden in EMPHAS automatisch berechnet. Für die Bestimmung des Trägheitstensors wurde dabei auf das in [Mir96a] und [Mir00a] beschriebene, frei verfügbare C-Programm zurückgegriffen.

6.5.5.3 Export von Geometrie- und Bewegungsdaten

In kommerziellen Animationssystemen werden eine Vielzahl von Techniken angeboten, die das fotorealistische Rendering von Animationen unterstützen. Um diese Funktionalitäten nutzen zu können, wurde in EMPHAS eine Exportmöglichkeit verwirklicht, mit der die simulierten Bewegungsabläufe als Keyframe-Animationen in das Animationssystem *3D Studio Max* übertragen und dort weiterverarbeitet werden können. Zu diesem Zweck werden zunächst die Zustandswerte, die die dynamischen Körper nach jedem k-ten Simulationsschritt annehmen, zusammen mit anderen Szenenparametern in eine Datei mit einem internen Format geschrieben. Anschließend kann diese Datei dann mit Hilfe eines in der *MaxScript*-Sprache implementierten Skriptes von *3D Studio Max* ausgewertet werden kann. Für jeden Datensatz werden dann entsprechende Keyframes für jedes Objekt erzeugt. Die Intervallgröße k (mit $k \geq 1$) kann dabei frei vorgegeben werden. Ein Wert für k größer als 1 führt zu einer Verkleinerung der Export-Datei, die bei längeren Animationen ein beträchtliches Ausmaß annehmen kann. Die zwischenliegenden Frames werden in diesem Fall allerdings in *3D Studio Max* durch eine rein geometrische Interpolation erzeugt, was zumindest bei sehr großen k-Werten zu einer deutlich sichtbaren Verfälschung der simulierten Bewegungen führen kann.

Für die Auswertung der Datei durch das Skript-Programm wurden zwei verschiedene Möglichkeiten realisiert. Die erste besteht in der Übertragung der gesamten Szene, wobei nicht nur für die dynamischen Körper, sondern auch für die Steuerelemente entsprechende geometrische Primitive von *3D Studio Max* erzeugt werden, deren Form der Darstellung dieser Elemente in EMPHAS (vergl. Abschnitt 6.5.2) entspricht. Neben den Translations- und Rotations-Zustandsgrößen werden daher in der Export-Datei auch die drei Parameter, die der Skalierung dieser Elemente entsprechen, zu jedem Keyframe abgespeichert und entsprechend ausgewertet. Auf diese Weise können die Steuerelemente wie zusätzliche Körper eingesetzt werden, was eine sehr wichtige Möglichkeit darstellt. Die Erzeugung der Primitive und die Festlegung ihrer Eigenschaften zu jedem vorgegebenen Keyframe geschieht dabei völlig automatisch. Anschließend läßt sich die Animation dann in *3D Studio Max* als Keyframe-Animation weiterverarbeiten und gerendert ausgeben.

Als zweite Möglichkeit wurde eine selektive Übertragung von simulierten Zustandsdaten auf beliebige in *3D Studio Max* vorliegende Körper verwirklicht. Diese Zuweisung läßt sich innerhalb von *3D Studio Max* interaktiv über entsprechende Bedienelemente durchführen. Mit Hilfe dieser Funktionalität kann auch Körpern, die aus anderen Anwendungen oder Animationssystemen stammen und eine beliebig komplexe Gestalt haben können, ein mit EMPHAS generiertes, physikalisch basiertes Bewegungsverhalten verliehen werden.

Falls der Körper nicht die gleiche geometrische Form wie der zugehörige simulierte Körper hat, können dabei natürlich Abweichungen zum physikalisch korrekten Bewegungsverhalten resultieren. Gerade bei der Anwendung der automatischen Kontaktbehandlung, bei der die geometrische Form eine entscheidende Rolle spielt, ist eine solche Übertragung daher nur unter ganz bestimmten Umständen sinnvoll (z.B. wenn die beiden Körper über die gleiche konvexe Hülle verfügen). Selbst bei sehr großen Form-Unterschieden ergeben sich aber in vielen Fällen keine deutlich sichtbaren oder überhaupt keine Verfälschungen aus diesem Umstand. Bei reinen Translationsbewegungen geht die geometrische Form z.B. nur indirekt über die Größe des Körpervolumens, das für die Gesamtmasse entscheidend ist, in die physikalischen Bewegungsgleichungen ein (vergl. Anhang C). Diese selektive Übertragung stellt daher eine äußerst nützliche Export-Möglichkeit dar, die sich sehr flexibel einsetzen läßt.

6.6 Anwendung der Steuerelemente

Wie in Abschnitt 6.2 ausgeführt wurde, sind für EMPHAS eine ganze Reihe verschiedener Steuertechniken realisiert worden. Es soll nun die *Anwendung* dieser Elemente unter Ausnutzung der in Abschnitt 6.3 und 6.5 vorgestellten Methoden zur physikalisch basierten Bewegungsgenerierung bzw. zur Schaffung eines intuitiv bedienbaren Animationssystems beschrieben werden.

6.6.1 Festlegung von Constraints und Controllern

6.6.1.1 Das „Baukasten“-Konzept

Für eine einfache interaktive Animationserstellung mußte die Anwendung von Controllern und Constraints so allgemein und intuitiv wie möglich gestaltet werden. Dieses Ziel konnte durch die Umsetzung eines einfach handhabbaren „Baukasten“-Konzeptes verwirklicht werden, das auch in [Wag00] beschrieben wurde und nun vorgestellt werden soll.

Durch die Anwendung des in Abschnitt 5.1 beschriebenen Konnektoren-Konzeptes ließen sich in EMPHAS dynamische Körper und Constraints als voneinander unabhängige Module realisieren. Auf dieser Grundlage konnte auch die *Festlegung* von Constraints, d.h. ihre konkrete Anwendung

auf bestimmte dynamische Körper, sehr modular gestaltet werden. Starre und punktförmige Körper, raumfeste Punkte und Kurven können gleichermaßen als Anknüpfungsobjekte für Constraints dienen. Aber auch die Steuerelemente selber können in sehr einheitlicher Weise angewandt werden. Nicht nur holonome Constraints, sondern auch Geschwindigkeits-Constraints und Controller können mit diesen Anknüpfungsobjekten verknüpft werden. Durch diese Modularität wird der Umgang mit den Steuerelementen sehr vereinfacht und auf eine intuitive Ebene gestellt.

Wie in einem Baukastensystem können diese Elemente zudem beliebig miteinander kombiniert werden. Die Grundlagen für diese Möglichkeit wurden in Abschnitt 5.4.1 besprochen, wobei sich die speziellen Eigenschaften der LFM in Form des physikalisch basierten Prozesses der Constraint-Erfüllung als besonders wichtig erwiesen haben. Für die kombinierte Anwendung mit Kraftfeldern, die in Abschnitt 5.4.1 nicht betrachtet wurden, ergeben sich zudem keine Schwierigkeiten, da diese Felder lediglich zu zusätzlichen äußeren Kräften führen und selber nicht von der Szenendynamik beeinflußt werden. Grundsätzlich lassen sich in EMPHAS *alle* der in Abschnitt 6.2 vorgestellten Steuerelemente beliebig miteinander kombinieren. Dabei wurde auch eine gemeinsame Anwendung mit der automatischen Kollisionssimulation und Ruhekontaktbehandlung verwirklicht, wie in Abschnitt 6.4.2 ausgeführt wurde.

Sehr wichtig ist dabei auch, daß in EMPHAS keine künstliche Trennung zwischen der Anwendung dieser Elemente zur Modellierung und zur Bewegungssteuerung besteht. Dieser Aspekt wird in Abschnitt 6.6.1.3 genauer besprochen.

6.6.1.2 Festlegung von Constraints und Controllern

Es soll nun geschildert werden, wie die konkrete Festlegung von Constraints und Controllern bei EMPHAS durchgeführt werden kann. Für Kraftfelder und Prozeduren ist eine solche Objektverknüpfung nicht notwendig, da sich ihre Wirkung prinzipiell auf alle in der Szene vorhandenen Körper bezieht.

Um ein Steuerelement erzeugen zu können, müssen zunächst sämtliche Anknüpfungspunkte ausgewählt werden, auf die sich das Element beziehen soll. Die Auswahl der jeweiligen Körper, Konnektoren oder Kurven kann in einfacher Weise mit Hilfe der in Abschnitt 6.5.4.1 beschriebenen Selektionstechniken durchgeführt werden. Die Anzahl und Art dieser Objekte hängt dabei vom gewünschten Steuerelement ab, wie unten genauer ausgeführt wird.

Konnektoren vom Typ *ConstConnector* (vergl. Abschnitt 6.2.2) können dabei indirekt über die Anzahl der selektierten Anknüpfungsobjekte ausgewählt werden. Bei der Erzeugung eines *PointToPoint*-Constraints mit nur einem ausgewählten Körperpunkt wird z.B. automatisch ein *ConstConnector* erzeugt und dem Constraint zusammen mit dem zum Körperpunkt gehörenden *BodyConnector* zugeordnet. Die Koordinaten des *ConstConnector*-Raumpunktes werden dem Constraint dabei als Parameter zugeordnet, die sich jederzeit modifizieren lassen.

Für jeden Constraint oder Controller muß außerdem mindestens ein Anknüpfungspunkt angegeben werden, der mit einem dynamischen Körper verknüpft ist. Ansonsten wäre kein Eingriff in das dynamische System möglich. Bezüglich Art und Anzahl weiterer Anknüpfungsobjekte bestehen aber große Unterschiede zwischen den einzelnen Steuerelementen. Daher soll nun aufgeführt werden, welche weiteren Angaben bei der Festlegung der in Abschnitt 6.2.4.1 und 6.2.3 beschriebenen Controller und Constraints notwendig sind.

Für *PointToFunction*-Constraints *VelocityToFunction*-Constraints und *VectorForce*-Controller sind (neben dem körperfesten Anknüpfungspunkt) keine weiteren Angaben notwendig, da sie sich auf genau einen Körperpunkt beziehen. Ein *PointToPath*-Constraint verbindet einen Körperpunkt mit

Element	Verkn.	Element	Verkn.	Element	Verkn.
PointToPoint	P,PP,PK	*Orientation*	A,AA	*EqualVelocity*	P,PP
Distance	P,PP,PK	*InLine*	A,AP	*EqualVelocityNorm*	P,PP
UnitDistance	P,PP,PK	*Ortho*	A,AA	*VelocityToFunction*	P
PointToPath	PK	*Hinge*	A,AA	*SpringForce*	P,PP,PK
PointToFunction	P	*Slide*	A,AA	*VectorForce*	P
		Universal	A,AA	*TrajectoryForce*	P,PP,PK

Tabelle 6.3: Anknüpfungsarten der Constraints und Controller in EMPHAS. Dabei bezeichnet „P“ einen körperfesten Anknüpfungspunkt, „A“ eine durch zwei Punkte festgelegte Achse und „K“ eine Kurve.

einer Kurve, die den Raumpfad für den Punkt vorgibt, und erfordert daher die Auswahl eines Körperpunktes und einer Kurve. *EqualVelocity*- und *EqualVelocityNorm*-Constraints lassen sich wahlweise auf zwei oder nur auf einen Körperpunkt anwenden. Im letzteren Fall wird der zweite Punkt automatisch einem *ConstConnector* zugeordnet, so daß der erste Körperpunkt parametrisch gesteuert werden kann. Diese Möglichkeit besteht auch für die Constraint-Typen *PointToPoint*, *Distance* und *UnitDistance* sowie die Controller *SpringForce* und *Trajectory*. Sie erlauben aber zusätzlich die Verbindung eines Körperpunktes mit einer Kurve, genauer gesagt mit einem konkreten Punkt auf dieser Kurve. Der Kurvenparameter dieses Punktes wird dabei durch einen skalaren Wert zwischen 0 und 1 festgelegt, der dem Steuerelement als Parameter zugeordnet wird.

Ein *Inline*-Constraint gibt eine Beziehung zwischen einer *Achse* und einem Punkt wieder. Eine solche Achse wird durch die Angabe zweier körperfester Punkte spezifiziert. Zusätzlich kann ein weiterer Anknüpfungspunkt auf einem anderen Körper angegeben werden, der ansonsten durch einen *ConstConnector* ersetzt wird. Auch für die Festlegung der Constraints *Orientation*, *Hinge* und *Slide* müssen mindestens zwei Punkte auf einem Körper ausgewählt sein. Falls nur diese beiden Punkte vorgegeben werden, wird dem *Orientation*-Constraint ein dreidimensionaler Vektor als Parameter zugeordnet, bei den beiden anderen Constraints ein Vektor und zusätzlich ein Raumpunkt. Wenn dies nicht gewünscht ist, müssen zwei zusätzliche Punkte auf einem anderen Körper ausgewählt werden.

Tabelle 6.3 faßt diese Anknüpfungsmöglichkeiten noch einmal zusammen.

Die Anwendung dieser Steuerelemente konnte somit sehr flexibel gestaltet werden. Wie die Erfahrungen mit EMPHAS zeigen, kann der Umgang mit ihnen sehr schnell erlernt werden, da das hier beschriebene Baukasten-System in intuitiver Weise erfaßbar ist und für eine Vielzahl von Modellierungs- und Steuermöglichkeiten eingesetzt werden kann. Sehr wichtig ist aber auch noch ein anderer Aspekt, der nun genauer erläutert werden soll – die stetige Constraint-Erfüllung.

6.6.1.3 Stetige Constraint-Erfüllung

Die Constraints werden in EMPHAS nach ihrer Erstellung durch den Animateur nicht instantan, sondern durch einen stetigen, physikalisch basierten Vorgang erfüllt. Die Grundlage hierfür bildet die Eigenschaft der LFM, Constraints durch entsprechend berechnete Kräfte zu erfüllen (vergl. Abschnitt 4.2). In EMPHAS konnte dieses Verhalten, dessen Nutzen für den Modellierungsprozess erstmals in [BB88] beschrieben wurde, für eine sehr intuitive Anwendung von Constraints ausgenutzt werden, wie auch in [Wag99] erläutert wurde. Durch die stetige Erfüllung der Constraints kann leicht nachvollzogen werden, mit welchen Körpern sie verknüpft sind (nicht immer kann dies durch die graphische Darstellung der Constraints eindeutig angezeigt werden) und in welcher Weise sie auf diese Körper wirken. Diese Anwendung wird somit auch der Anforderung der *Vorhersagbarkeit*

gerecht, die für die Nutzung eines allgemeinen Animationssystems sehr wichtig ist.

Ein anderer wesentlicher Aspekt ist der hierdurch erzielte Grad an *Einheitlichkeit*: Controller und Constraints können in vollkommen einheitlicher Weise zu Modellierungs- und Steueraufgaben herangezogen werden. Diese Eigenschaft gründet sich auf dem Umstand, daß Constraints und andere Steuerelemente in EMPHAS auch während einer laufenden Simulation neu festgelegt werden können. Da der oben beschriebene Prozess der Constraint-Erfüllung einer dynamisch korrekten Einflußnahme auf die Körperbewegungen entspricht, läßt er sich somit auch als Mittel zur Bewegungssteuerung interpretieren und entsprechend einsetzen. Ein *PointToPoint*-Constraint kann auf diese Weise z.B. auch für die Vorgabe eines Zielpunktes für einen bestimmten Körper eingesetzt werden.

Der durch die LFM in natürlicher Weise ermöglichte physikalisch basierte Prozess der Constraint-Erfüllung liefert somit einen wichtigen Beitrag für die einfache Durchführung von Modellierungs- und Steueraufgaben im Rahmen eines allgemeinen Animationssystems.

6.6.2 Behandlung überbestimmter Systeme

Die Behandlung überbestimmter Systeme stellt wie in Abschnitt 5.4.2 begründet eine wichtige Aufgabe im Rahmen allgemeiner Animationssysteme dar. In EMPHAS wurde neben der Möglichkeit zum Simulationsabbruch sowohl die Bestimmung einer Näherungslösung als auch die Anwendung einer Technik zur Einschränkung von Constraints verwirklicht. Die Einführung zusätzlicher Freiheitsgrade wurde dagegen wegen des großen Realisierungsaufwandes weiterführenden Arbeiten vorbehalten.

6.6.2.1 Abbruch der Simulation

Falls bei den Verfahren *Direct* oder *Baraff*, für die die Erweiterung um das SVD-Verfahren nicht durchgeführt wurde, ein überbestimmter Zustand detektiert wird, erfolgt ein Abbruch der Simulation mit einer entsprechenden Fehlermeldung. Der Benutzer kann dann die Überbestimmtheit auflösen, z.B. durch die Deaktivierung einzelner Constraints, und anschließend die Simulation fortführen. Als Hilfe für den Animateur wird dabei einer der Constraints, die zu der Überbestimmtheit geführt haben, automatisch deaktiviert (und entsprechend farblich gekennzeichnet, vergl. Abschnitt 6.5.2).

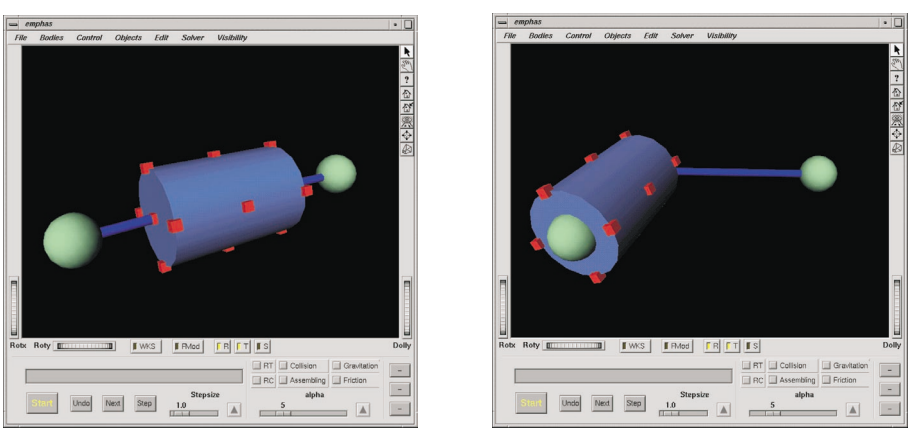

Abbildung 6.10: Behandlung überbestimmter Systeme durch Bestimmung einer Näherungslösung (links) und durch Einschränkung von Constraints (rechts).

6.6.2.2 Bestimmung einer Näherungslösung

Bei den Verfahren *DirectSVD* und *BaraffSVD* wird die Simulation im Falle eines überbestimmten Systems nicht abgebrochen, sondern vielmehr eine optimale Lösung für das Problem gesucht. Die Grundlage hierfür stellt die in Abschnitt 6.3.3 vorgestellte Anwendung des SVD-Verfahrens dar. Für das lineare Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ liefert es in diesem Fall denjenigen Lösungsvektor $\mathbf{x}$, der den Ausdruck $|\mathbf{Ax} - \mathbf{b}|$ minimiert.

In Abbildung 6.10 ist ein Beispiel für die Anwendung dieser wichtigen Technik dargestellt. Als Grundlage diente hierfür das in Abbildung 5.2 betrachtete Problem: Die beiden *PointToPoint*-Constraints, die den Zylinder mit zwei raumfesten Punkten verknüpfen sollen, führen zu einem überbestimmten System. Mit Hilfe des SVD-Verfahrens wird in diesem Fall eine Lösung gefunden, bei der der Zylinder genau zwischen den beiden Punkten verharrt. Die Behandlung solcher Systeme mit diesem Verfahren ist daher sowohl als Hilfe für den Animateur zur Auflösung der Überbestimmtheit als auch als bewußt eingesetzte Erweiterung der Modellierungs- und Steuertechniken einsetzbar.

Einen großen Vorteil für den Einsatz des SVD-Verfahren stellt dabei die in Abschnitt 6.3.3 beschriebene eingeschränkte Anwendung auf die sekundären Constraints dar, da dieses Verfahren nicht den Effizienzgrad anderer numerischer Verfahren zur Lösung linearer Gleichungssysteme erreicht.

6.6.2.3 Selektive Einschränkung von Constraints

Ein ganz gezielter Einsatz von überbestimmten Systemen ist mit Hilfe der in EMPHAS realisierten selektiven Sperrung von Constraint-Komponenten möglich, die in Abschnitt 6.2.3 angesprochen wurde. Ein Beispiel für die Anwendung dieser Technik ist in Abbildung 6.10 rechts dargestellt. Einer der beiden *PointToPoint*-Constraints wurde dabei mit Hilfe dieser selektiven Sperrung in seiner Wirkung auf die y-z-Ebene beschränkt. Auf diese Weise lassen sich *beide* Constraints erfüllen, ohne daß eine singuläre und damit unbestimmte Konfiguration auftritt. Der hieraus resultierende Einfluß auf den Zylinder ist auf intuitive Weise nachzuvollziehen und kann ein durchaus wünschenswertes Verhalten darstellen. Diese Technik führt daher zu einer wichtigen Erweiterung der Anwendungsmöglichkeiten von Constraints zur Modellierung und Bewegungssteuerung. Zusammen mit der Anwendung des SVD-Verfahrens konnte somit ein robuster und sehr flexibler Umgang mit überbestimmten Systemen verwirklicht werden.

6.6.3 Vermeidung von Bewegungsartefakten

In Abschnitt 5.4.3 wurde das Problem der Bewegungsartefakte erörtert, das sich aus den speziellen Eigenschaften der LFM ergibt. Wie im folgenden dargestellt wird, konnten im Rahmen dieser Arbeit Techniken realisiert werden, die eine wirkungsvolle Unterdrückung dieser Artefakte erlauben und die sich in EMPHAS sehr bewährt haben. Ihre Konzeption und Realisierung stellt einen wichtigen neuen Beitrag für die Animationserstellung auf der Grundlage der LFM dar.

Explizite Zustandsänderung. Eine sehr direkte Möglichkeit zur Behandlung dieses Problems stellt die nachträgliche Korrektur der Bewegungen dar. In EMPHAS ist es z.B. sehr einfach, über die Benutzungsoberfläche einen Körper zu selektieren und seine Geschwindigkeit auf 0 zu setzen. In Abschnitt 5.4.3 wurden aber bereits die Probleme dieser Herangehensweise angesprochen: Die Korrektur kann sich durch die Kopplung der Körper als sehr schwierig erweisen und ist zudem erst nach dem Prozess der Constraint-Erfüllung möglich.

Abbremsung durch ein Kraftfeld. Eine einfache, aber oftmals sehr nützliche Alternative stellt

die Anwendung des in Abschnitt 6.2.4.3 beschriebenen *FrictionForce*-Kraftfeldes dar. Dieses globale Kraftfeld bewirkt eine Abbremsung sämtlicher Bewegungen und ähnelt daher in seiner Wirkung einer starken allgegenwärtigen Reibungskraft. Durch die fehlende Möglichkeit zur körper- oder bewegungsspezifischen Anpassung dieser Wirkungen können hieraus natürlich auch unerwünschte Effekte resultieren. Gerade für Modellierungsvorgänge, bei denen die durch die Abbremsung bedingten Bewegungsverfälschungen nicht ins Gewicht fallen, stellt diese Anwendung aber ein sehr nützliches Verfahren dar.

Umschalten des Bewegungsverhaltens. Für Constraints, die auf mehr als einen Körper wirken, läßt sich zudem ein anderes Vorgehen anwenden, das gute praktische Resultate liefert: Das Umschalten einzelner Körper auf ein „kinematisches“ Bewegungsverhalten. Durch diese in Abschnitt 6.5.4.2 besprochene Interaktionsmöglichkeit bleiben die entsprechenden Körper unbeeinflußt von äußeren Kräften. Sie werden somit auch nicht durch die Stabilisierungskräfte beeinflußt, sondern verharren in Ruhe oder in einer gleichförmigen Bewegung, die sich leicht (ohne Rückwirkung anderer Körperbewegungen) abstoppen läßt. Dieses Umschalten kann sowohl vor als auch nach der Constraint-Festlegung in sinnvoller Weise angewandt werden.

Sperrung von Freiheitsgraden. Eine weitere Möglichkeit, die ebenfalls auf dem Ansatz der Freiheitsgrad-Einschränkung beruht, besteht in der Ausnutzung der in Abschnitt 6.2.2 beschriebenen Sperrung von Translations- und Rotationsfreiheitsgraden. Ein einzelner *PointToPoint*-Constraint zwischen zwei Körperpunkten läßt sich z.B. auch durch eine reine Translation der Körper erfüllen. Die Sperrung von Rotations- bzw. Translationsfreiheitsgraden stellt für diesen und ähnliche Fälle eine sehr elegante Möglichkeit dar, unerwünschte Rotationen bzw. Translationen der Körper zu verhindern. In EMPHAS läßt sich diese Sperrung für jeden Konnektorpunkt getrennt über die Parameterliste des jeweiligen Constraints durchführen (vergl. Abschnitt 6.2.3.2), so daß sich diese Technik in sehr flexibler Weise einsetzen läßt.

Assembling-Modus. Eine besonders nützliche Technik zur Unterdrückung von Bewegungsartefakten stellt der in EMPHAS realisierte *Assembling-Modus* dar. Im Gegensatz zu den oben besprochenen Ansätzen erfordert diese Methode, die sich für Constraints anwenden läßt, die auf genau zwei Körper wirken, keinen zusätzlichen Eingriff des Animateurs. Wenn dieser Modus aktiv ist, bleibt der Körper, der bei der Constraint-Festlegung zuletzt selektiert wurde, solange unbeeinflußt von äußeren Kräften (und damit in seinem ursprünglichen Bewegungszustand), bis der Constraint erfüllt ist[12]. Wenn das der Fall ist, wird die Geschwindigkeit des zuerst selektierten Körpers auf 0 gesetzt und die Simulation wie gewohnt fortgesetzt. Als Ergebnis dieses Vorgehens wird der Constraint erfüllt, ohne zusätzliche Bewegungen der beiden Körper entstehen zu lassen. Sie bleiben somit nach dem Erfüllungs-Prozess in Ruhe, bzw. in einer gleichförmigen Bewegung, falls der zuletzt selektierte Körper vor der Constraint-Festlegung in Bewegung war. Dieser Vorgang geschieht völlig automatisch, ein Eingriff des Animateurs ist also nicht erforderlich.

Ein Beispiel für die Anwendung dieser Technik ist in Abbildung 6.11 dargestellt. Die Festlegung eines *PointToPoint*-Constraints für zwei in Ruhe befindliche Körper kann wie in der oberen Sequenz gezeigt zu starken Bewegungsartefakten führen, während beim aktivierten Assembling-Modus zunächst die Bewegung eines Körpers erfolgt, die dann nach der Erfüllung des Constraints automatisch abgestoppt wird.

Man kann abschließend feststellen, daß die aus dem Stabilisierungsverfahren resultierenden Bewegungsartefakte ein spezifisches Problem der LFM darstellen, das aber mit den hier vorgestellten

[12] Tatsächlich erfolgt dieser Wechsel dann, wenn der Constraint *nahezu* erfüllt ist (bezüglich eines bestimmten Toleranzwertes), da keine völlig exakte Constraint-Erfüllung vorausgesetzt werden kann.

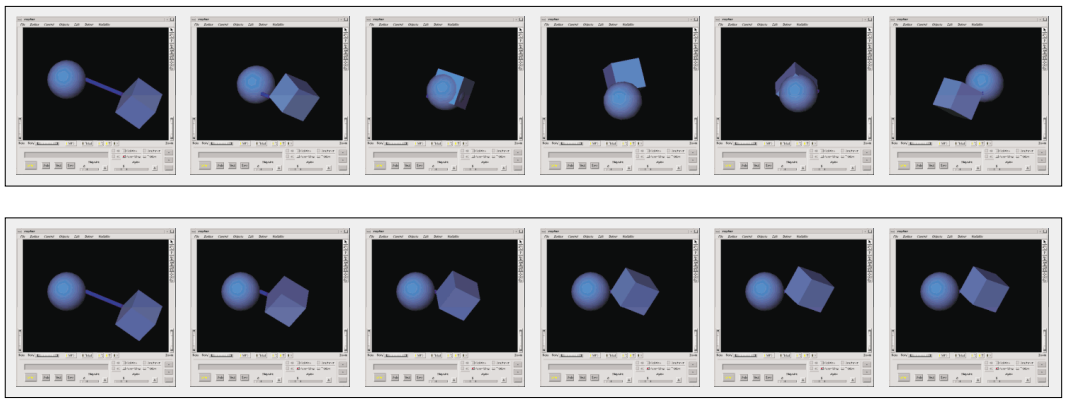

Abbildung 6.11: Erfüllung eines *PointToPoint*-Constraints ohne (oben) und mit aktiviertem Assembling-Modus (unten).

Techniken, die sich auch kombiniert anwenden lassen, in soweit gelöst werden konnte, daß nur ein unwesentlicher zusätzlicher Aufwand bei der Animationserstellung in Kauf genommen werden muß.

6.7 Vergleich mit kommerziellen Animationssystemen

In diesem Abschnitt soll das Animationssystem EMPHAS gegenüber derzeitigen kommerziellen Animationssystemen abgegrenzt werden. Neben einem grundsätzlichen Vergleich der bereitgestellten Funktionalitäten soll dabei vor allem die Zielstellung bei der Entwicklung von EMPHAS verdeutlicht werden.

Für die Animation dreidimensionaler Szenen sind mittlerweile eine Vielzahl von kommerziellen Animationssystemen verfügbar. Die zentrale Rolle bei der Bewegungserstellung nimmt dabei das klassische Verfahren des Keyframings ein. Viele Systeme stellen aber auch Techniken zur physikalisch basierten Animation von mechanischen Systemen in Form starrer Körper zur Verfügung. Zu ihnen zählen die Animationssysteme[13] *3D Studio MAX* (von der Firma *Discreet*), *Animation:Master* (*Hash*), *Cinema 4D* (*Maxon*), *Lightwave* (*NewTek*), *Maya* (*Alias Wavefront*) und *SoftImage XSI* (*Avid Technology*) sowie die Plugin-Module *Digimation Plugins* (*Digimation*), *Falling Bodies* (*Animats*), *Havok* (*Havok.com*) und *ReelMotion* (*Motional Realms*). Die physikalisch basierte Animation flexibler Körper erlauben dabei *3D Studio MAX*, *Lightwave*, *Maya*, *SoftImage XSI*, *Digimation Plugins* und *Havok*.

Diese Systeme stellen hochkomplexe Softwareprodukte dar, die mit einem enormen Entwicklungsaufwand produziert werden. Ihre Funktionalitäten bezüglich der geometrischen Modellierung, der Oberflächen- und Lichtgestaltung, der Anwendung von Spezialeffekten und der Möglichkeiten für ein fotorealistisches Rendering übersteigen die von EMPHAS bei weitem. Im folgenden sollen daher nun einige wichtige Aspekte für die Abgrenzung von EMPHAS gegenüber diesen Systemen angeführt werden:

- Der Einsatz der derzeitigen kommerziellen Animationssysteme erfordert aufgrund ihrer Komplexität einen sehr großen Einarbeitungsaufwand. Für die Anwendung der bereitgestellten physikalisch basierten Techniken gilt dies in besonderem Maße. EMPHAS ermöglicht dagegen eine einfache, vollständig interaktive Erstellung physikalisch basierter Animationen,

[13] Diese Aufstellung entstammt einer studentischen Arbeit von Gunnar Wallmann.

die sich durch äußerst realistisch wirkende Bewegungen auszeichnen. Eine wesentliche Rolle spielen dabei die verschiedenen, beliebig kombinierbaren Steuertechniken, die für dieses System realisiert wurden.

- In EMPHAS wurde ein Bedienungskonzept verwirklicht, das die einfache Anwendung von Modellierungs-, Steuer- und Interaktionstechniken *während* des Prozesses der physikalisch basierten Bewegungsgenerierung erlaubt. Diese äußerst intuitive Art der Einflußnahme besteht bei keinem der oben erwähnten kommerziellen Animationssysteme.

- Sehr wichtig sind außerdem die Aspekte, die die *Umsetzung* von EMPHAS betreffen. Die entwickelte Simulationsumgebung auf der Grundlage der LFM, aber auch die dynamischen Körper, die Constraints und die anderen Steuerelemente sind durch einen hohen Grad an Generalität und Modularität ausgezeichnet. Auf diese Weise wurde auch eine einfache Erweiterbarkeit um zusätzliche Komponenten erzielt. Ein Vergleich mit kommerziellen Animationssystemen ist dabei allerdings kaum möglich, da sich die zu diesen Systemen erschienene Literatur auf die Erläuterung ihrer Funktionalitäten beschränkt (vergl. Abschnitt 4.1.2).

Ein wesentlicher Unterschied besteht vor allem in der Zielsetzung bei der Entwicklung dieser Systeme. Bei der Realisierung von EMPHAS stand nicht ein möglichst großer Funktionsumfang im Vordergrund, sondern die Ausnutzung der in Kapitel 4 angeführten Eigenschaften der LFM und die Entwicklung von Techniken auf der Grundlage der in Kapitel 5 vorgestellten Konzepte zur Umsetzung der LFM[14], wie in diesem Kapitel 6 ausführlich dargelegt wurde. Auf diese Weise konnte ein allgemeines Animationssystem verwirklicht werden, das den in der Einführung dieser Arbeit genannten Anforderungen gerecht wird und die grundsätzlichen Vorteile der LFM für die Erstellung physikalisch basierter Animationen aufzeigt.

[14] Aus diesem Grund wurde für EMPHAS auch wie schon in Abschnitt 6.3.5 erwähnt keine quantitative Analyse der zeitlichen Effizienz des Simulationsverfahrens durchgeführt.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

In dieser Arbeit wurde eine systematische Untersuchung über die Anwendung von Lösungsverfahren zur Simulation mechanischer, Constraint-behafteter Systeme vor dem Hintergrund allgemeiner Animationssysteme durchgeführt, in der bislang unbeachtet gebliebene Problemverknüpfungen aufgezeigt werden konnten. Die Lagrange-Faktoren-Methode (LFM) erwies sich dabei für eine solche Anwendung als besonders geeignet. Im weiteren Verlauf der Arbeit wurden dann Konzepte und Methoden für die Realisierung allgemeiner, physikalisch basierter Animationssysteme auf der Grundlage der LFM vorgestellt.

Zunächst wurde hierzu ein Konzept entworfen, das die automatische Einbindung von Constraints und ihre Kapselung in separate, von den dynamischen Körpern unabhängige, Module ermöglicht. Das in der Literatur beschriebene Konnektoren-Konzept wurde hierfür auf die Einbindung einer allgemeineren Constraint-Klasse und auf explizit zeitabhängige Konnektoren erweitert. Auf dieser Grundlage wurde einigen wichtigen Constraint-Typen eine mathematische Form gegeben.

Mit der Aufspaltung der LFM in verschiedene Teilaufgaben und der allgemeinen Anwendung des in der Literatur beschriebenen Baraff-Verfahrens, das eine sehr effiziente Bestimmung der Lagrange-Faktoren erlaubt, wurde dann ein modulares Konzept zur Umsetzung der LFM vorgestellt. Dabei wurden auch die Möglichkeiten zur Beschreibung des Systemzustandes und zur Aufstellung der Bewegungsgleichungen untersucht und bewertet.

Eine schwierige Aufgabe stellt die Kombination verschiedener Steuertechniken dar. Für dieses Problem konnten Lösungen aufgezeigt werden, die sich insbesondere für die gemeinsame Anwendung von Constraints und prozeduralen Verfahren auf spezielle Eigenschaften der LFM stützen. Zudem wurde die Bedeutung überbestimmter Systeme vor dem Hintergrund allgemeiner Animationssysteme untersucht und eine bewertende Aufstellung von Konzepten zu ihrer Behandlung gegeben. Mit der Entstehung unerwünschter Bewegungsartefakte bei der Festlegung von Constraints wurde außerdem ein LFM-spezifisches, bislang gänzlich unbeachtetes Problem diskutiert. Zur Vermeidung dieser Artefakte wurden einige grundsätzliche Lösungskonzepte entworfen.

Ein wesentlicher Bestandteil dieser Arbeit ist die Entwicklung des modularen Animationssystems EMPHAS, das auf dem Lösungsverfahren der LFM basiert und eine effektive und vollständig interaktive Erstellung physikalisch basierter Animationen ermöglicht. Hierfür wurden spezielle Methoden auf der Grundlage der oben besprochenen Konzepte und unter Ausnutzung spezifischer Eigenschaften der LFM entworfen.

Für die Umsetzung der LFM wurden eine Reihe von Verfahren zur Lösung der verschiedenen Teilaufgaben realisiert, die sich für die effiziente Simulation beliebiger mechanischer Systeme einsetzen lassen. Mit der speziell angepaßten Erweiterung des Baraff-Algorithmus um ein Singular Value Decomposition- Verfahren wurde dabei auch die robuste Behandlung überbestimmter und schlecht konditionierter Systeme ermöglicht.

Als dynamische Körper sind in EMPHAS massenbehaftete Punktkörper und dreidimensionale starre Körper implementiert worden. Für die Modellierung und Steuerung dieser und anderer mechanischer Körpertypen wurden modular konzipierte Steuerelemente in Form von Controllern, holonomen und Geschwindigkeits-Constraints, Kraftfeldern und ereignisbasierten Prozeduren entwickelt. Insbesondere konnten dabei die Eigenschaften der LFM für ein sehr allgemeines Prozeduren-Konzept ausgenutzt werden, das einen direkten Zugriff auf Körper und Constraints gestattet. Zudem wurden Methoden für die Simulation von Kollisionen und die automatische Behandlung von Ruhekontakten realisiert, die einen großen Beitrag für die visuelle Plausibilität der generierten Bewegungen liefern. Ein wichtiger Beitrag stellt dabei auch die geschaffene Möglichkeit zur kombinierten Anwendung mit sämtlichen der in EMPHAS realisierten Steuertechniken dar.

Für die Anwendung der Steuerelemente wurde ein intuitives „Baukasten“-Konzept verwirklicht: Dynamische Körper und Steuerelemente können als voneinander unabhängige Module behandelt werden, die sich wie in einem Baukastensystem beliebig miteinander kombinieren lassen. Die Teilaufgaben der Modellierung und der Bewegungssteuerung können dabei in einer sehr einheitlichen Weise gelöst werden. Auf der Grundlage der zuvor entwickelten Konzepte wurden außerdem Techniken realisiert, die eine robuste Behandlung überbestimmter Systeme und eine weitgehende Vermeidung der unerwünschten Bewegungsartefakte ermöglichen. Für das Artefakte-Problem wurden dabei neuartige Lösungsansätze vorgestellt, die sich auch miteinander kombinieren lassen.

Für EMPHAS wurde eine Benutzungsschnittstelle entwickelt, mit der die flexible Steuerung der physikalischen Simulation, die graphische Darstellung der dynamischen Körper, Konnektoren und Steuerelemente und die Anwendung intuitiver Interaktionstechniken realisiert werden konnten. Im Rahmen der Simulationssteuerung wurde dabei ein Echtzeit-Modus und das wichtige Konzept der „laufenden Simulation“ verwirklicht, mit dessen Hilfe sämtliche Benutzungseingaben auch *während* einer Simulation getätigt werden können. Die Interaktionstechniken, insbesondere in Form des kinematischen Körperverhaltens und der interaktiven Kraftanwendung, erlauben einen effektiven und intuitiven Umgang mit dem System. Mit dem Import von Geometriedaten und dem skriptgesteuerten Export von Geometrie- und Bewegungsdaten ist zudem eine Anbindung an kommerzielle Animationssysteme geschaffen worden, die die Weiterverarbeitung und Übertragung der mit EMPHAS generierten Bewegungsabläufe ermöglicht.

7.2 Ausblick

Es sollen nun abschließend einige Möglichkeiten zur Weiterentwicklung der in dieser Arbeit erlangten Ergebnisse aufgezeigt werden. Neben einer Erweiterung des Animationssystems EMPHAS werden dabei auch Anwendungen außerhalb des Bereiches der Computeranimation diskutiert.

Ein Animationssystem stellt ein sehr offenes System dar, das sich in vielfältiger Weise um zusätzliche Funktionalitäten ergänzen läßt. Dies zeigt sich auch an den derzeitigen kommerziellen Animationssystemen, die ein enormes Spektrum von Hilfsmitteln für Modellierungs-, Steuerungs-, Rendering- und anderen Aufgaben bereitstellen. Für die Frage, wie das im Rahmen dieser Arbeit entwickelte Animationssystem EMPHAS erweitert und verbessert werden kann, sollen daher nur einige wenige, aber sehr wichtig erscheinende Möglichkeiten angesprochen werden.

Eine grundsätzliche Erweiterung von EMPHAS würde die Anwendung auf allgemeinere Klassen mechanischer Systeme darstellen, insbesondere auf deformierbare Körper. Ihr Zustand und ihre zeitliche Entwicklung müßten dazu in geeigneter Weise mathematisch beschrieben werden. Sowohl die Simulationsumgebung von EMPHAS als auch die realisierten Steuertechniken ließen sich für diese Körpertypen prinzipiell in unveränderter Form anwenden. Um ihre effektive Animation zu ermöglichen, müßten allerdings zusätzliche Modellierungs- und Steuertechniken entworfen und das Problem ihrer Einbindung in die Benutzungsschnittstelle gelöst werden.

Die Entwicklung neuer Techniken zur Modellierung und flexiblen Steuerung stellt auch für sich genommen eine interessante Aufgabe dar. Die in dieser Arbeit vorgestellten Konzepte zur Formulierung, Kombination und Anwendung solcher Techniken, bei denen spezielle Eigenschaften der LFM ausgenutzt wurden und die sich auch als Grundlage für abstraktere Steuertechniken eignen, bieten hierfür vielversprechende Möglichkeiten. Erweiterungsmöglichkeiten bietet auch die automatische Kontaktbehandlung – neben der Einbeziehung von Reibungseffekten käme z.B. eine Erweiterung der Kontaktbestimmung auf allgemeinere Geometrieklassen in Betracht.

Weiter verbessern läßt sich auch die Benutzungsschnittstelle von EMPHAS. Als eine neue Technik könnte dabei die Interaktion mit Hilfe eines Force Feedback- Eingabegerätes auf der Grundlage der berechneten Constraint-Kräfte dienen.

Die in dieser Arbeit vorgestellten Konzepte und Methoden wurden im Hinblick auf eine allgemeine Erstellung physikalisch basierter Animationen entworfen. Über die interaktive Nachbildung, Manipulation und graphische Darstellung physikalisch simulierter Systeme läßt sich aber auch die Wirkungsweise der zu Grunde liegenden physikalischen Gesetze erfassen bzw. anschaulich demonstrieren. Das EMPHAS-System bietet daher auch interessante Möglichkeiten, um als „virtuelles Physik-Labor“ für wissenschaftliche Ausbildungszwecke eingesetzt zu werden.

Anhang A

Architektur von EMPHAS

In diesem Kapitel soll die Architektur des in der Programmiersprache *C++* implementierten Animationssystems EMPHAS skizziert werden. Der hierbei erzielte Grad an Modularität ermöglicht eine einfache Anpassung und Erweiterung des Systems. So können z.B. neue Körpertypen, Anknüpfungsobjekte und Steuerelemente unabhängig voneinander eingefügt werden und auch die Erweiterung um zusätzliche Lösungsverfahren für die verschiedenen Teilprobleme bei der Simulationsdurchführung läßt sich problemlos durchführen. Diese Eigenschaften der *Generalität* und *Modularität* sind wichtige Anforderungen für die Realisierung allgemeiner Animationssysteme. Eine besondere Relevanz erhält diese modulare Architektur zudem durch den Bezug zu den Eigenschaften der LFM: Die Kapselung von Körpern und Constraints wird durch diese Methode in natürlicher Weise nahegelegt und auch die Realisierung der Lösungsverfahren profitiert wie in Kapitel 5.3 beschrieben von der schematischen Anwendbarkeit der LFM.

Es soll nun dargestellt werden, wie sich die in Abschnitt 6.2 vorgestellten Grundelemente, die in den Abschnitten 6.3 und 6.4 beschriebenen Lösungsverfahren und die in Abschnitt 6.5 vorgestellte Benutzungsschnittstelle in die Gesamtarchitektur von EMPHAS eingliedern.

Auf der in Abbildung A.1 skizzierten untersten Architekturebene sind die dynamischen Körper und die Controller- und Constraint-Steuerelemente, die über Konnektoren auf die Körper zugreifen, angesiedelt. Wie bei fast allen der hier beschriebenen Module stellen sie abstrakte Basisklassen dar, von denen die eigentlichen Objekte abgeleitet sind. Eine Erweiterung um neue Typen ist auf diese Weise leicht möglich. Derartige Basisklassen sind in den Abbildungen dieses Kapitels dunkel gezeichnet.

Dynamische Körper sind in der Klasse *Body* gekapselt, die via Delegation auf eine *Shape*-Klasse zugreift, in der die geometrische Form der Körper verwaltet wird. Die Klasse *Connector* stellt das Verbindungselement zwischen Constraints und Controllern auf der einen und dynamischen Körpern auf der anderen Seite dar. Als wichtigste Unterklasse greift der *BodyConnector* hierzu via Delegation auf die *Body*-Klasse zu. Der Zugriff auf diese Konnektoren geschieht durch die Klasse *Control*, von der die Klassen *Constraint* und *Controller* abgeleitet sind. Neben der Verwaltung der Konnektoren, die für die Anwendung von Constraints und Controllern notwendig sind, stellt sie einige zusätzliche Funktionen zur Verfügung, mit der die Steuerelemente z.B. als aktiv oder inaktiv gekennzeichnet werden können.

In der *Constraint*-Klasse sind die Schnittstellen genau diejeniger Funktionen festgelegt, die nach den Ausführungen in Abschnitt 5.1 für die Formulierung von holonomen Constraints notwendig sind. Analoges gilt für die Klasse *VConstraint*, in der die Geschwindigkeits-Constraints gekapselt sind.

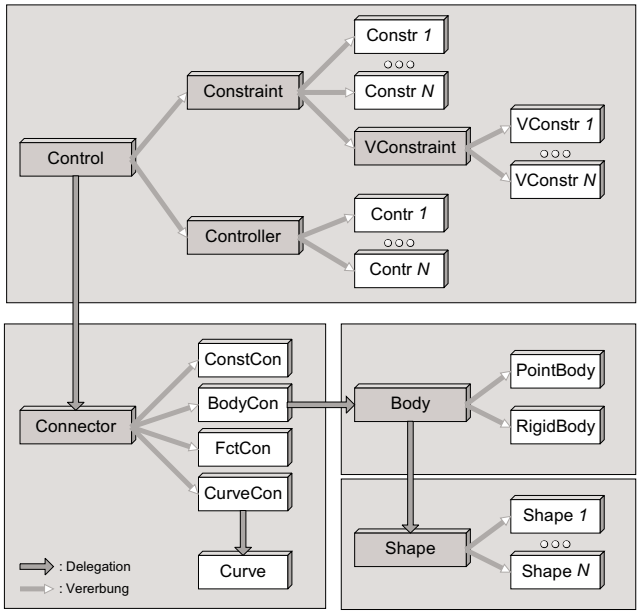

Abbildung A.1: Die unterste Ebene der EMPHAS-Architektur. Constraints und Controller greifen über Konnektoren auf dynamische Körper zu.

Dabei stellte sich das Problem, daß die beiden Klassen unterschiedliche Schnittstellen haben, so daß die Behandlung durch die LFM eine umständliche Fallunterscheidung notwendig gemacht hätte. Die *VConstraint*-Klasse wurde daher als Unterklasse der *Constraint*-Klasse formuliert, wobei die von ihr vererbten Funktionsschnittstellen in geeigneter Weise umdefiniert wurden. Die Funktion, die den Term $\frac{\partial C_i}{\partial \mathbf{K}_j}$ für holonome Constraints zurückliefern soll, liefert nun $\frac{\partial C_i}{\partial \mathbf{K}_j}$ für Geschwindigkeits-Constraints, die für $\frac{\partial \dot{C}_i}{\partial \mathbf{K}_j}$ liefert $\frac{\partial C_i}{\partial \mathbf{K}_j}$ und die für $\frac{\partial \dot{C}_i}{\partial t}$ liefert $\frac{\partial C_i}{\partial t}$. Die Funktion zur Rückgabe von C_i behält ihre Interpretation, die zur Bestimmung von $\dot{C}_i$ wird nicht benötigt. Grundlage dieser Identifizierungen sind die Beziehungen 3.3, 3.4 und 3.6, an denen man erkennen kann, daß sich holonome und Geschwindigkeits-Constraints auf diese Weise völlig analog behandeln lassen.

Auf der zweiten Architekturebene, die in Abbildung A.2 dargestellt ist, befinden sich die von der Klasse *Solver* eingesetzten Module. Als Steuerelemente gehören zu ihnen die in *ForceField* implementierten und auf die Klasse *Body* zugreifenden Kraftfelder sowie die in *Procedure* gekapselten Prozeduren, die sowohl auf die *Control*- als auch auf die *Body*-Klasse zugreifen können. In dieser Ebene sind auch die Module zur Lösung Constraint-behafteter Bewegungsgleichungen und zur Durchführung der automatischen Kontaktbehandlung.

Die Verfahren zur Bestimmung der Lagrange-Faktoren wurden dabei als Unterklassen von *CSolver* realisiert. In dieser Klasse sind auch die Aufgaben, die bei jedem Lösungsverfahren durchgeführt werden müssen, z.B. die Bestimmung der Jacobi-Matrix aus den aktuellen Constraint-Werten, implementiert. Auf diese Weise wird eine Erweiterung um zusätzliche Lösungsverfahren sehr erleichtert. Das gilt auch für die Verfahren zur Lösung der gewöhnlichen Differentialgleichungen in Form der *ODESolver*- und für die Stabilisierungsverfahren in Form der *StabSolver*-Klasse.

Die Klasse *RFSolver* stellt Funktionen zur Behandlung von Ruhekontakten zur Verfügung, während die in analytischer Form mögliche Kollisionssimulation in *Solver* integriert wurde. Für die Kontaktbestimmung wurde wie in Abschnitt 6.4.1 erwähnt die frei verfügbare Bibliothek *I-COLLIDE* eingesetzt. Der Zugriff auf diese Bibliothek geschieht über eine Unterklasse der abstrakten Adapter-

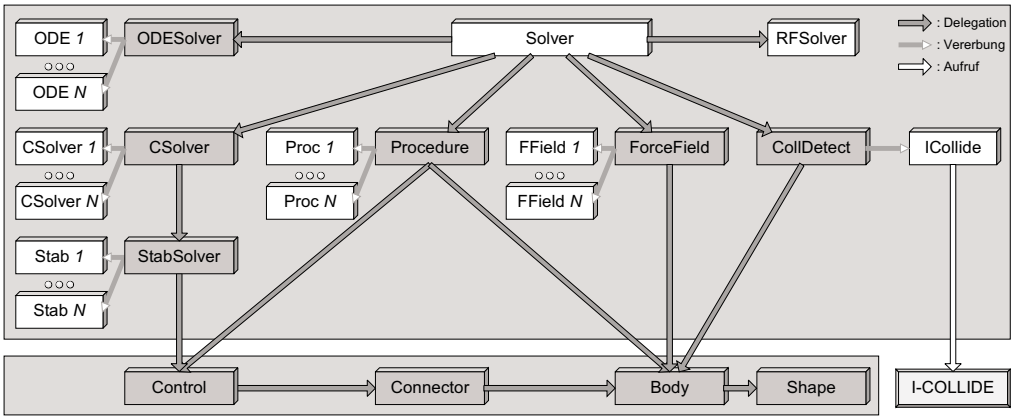

Abbildung A.2: Die mittlere Ebene der EMPHAS-Architektur. Neben den Steuerelementen der Prozeduren und Kraftfelder sind hier die Module zur Durchführung der physikalischen Simulation und der Kontaktbehandlung angesiedelt.

Klasse *CollDetect*, um die Einbindung anderer oder zusätzlicher Kontaktbestimmungs-Bibliotheken zu erleichtern.

Auf der obersten Ebene mußte schließlich eine Verbindung zur Benutzungsoberfläche von EMPHAS geschaffen werden. Wie in den Abschnitten 6.5.2 und 6.5.4 erläutert wurde, mußte dabei insbesondere die graphische Darstellung der Körper und der Steuerelemente, aber auch die Manipulation dieser Objekte ermöglicht werden. Aus dieser engen Kopplung von Simulations- und Darstellungsobjekten resultierte ein schwieriges Problem: Die von *Body*, *Connector* und *Control* abgeleiteten Klassen mußten unabhängig von der graphischen Darstellung und der Programminteraktion spezifizierbar bleiben, um eine einfache Erweiterbarkeit zu gewährleisten. Zudem sollte vor allem aus Gründen der Portabilität eine möglichst kleine Schnittstelle zur *Open Inventor*-Bibliothek gebildet werden, mit deren Hilfe die Benutzungsoberfläche von EMPHAS realisiert wurde.

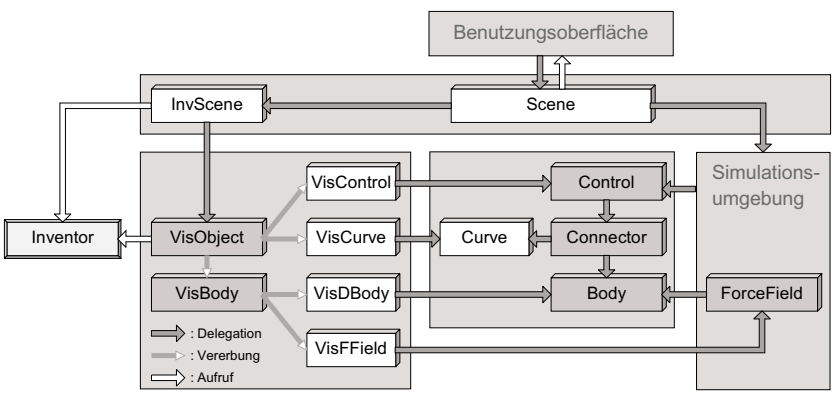

Abbildung A.3: Die oberste Ebene der EMPHAS-Architektur. Die Grundelemente und die Module zur Simulationsdurchführung werden mit der Benutzungsschnittstelle verknüpft.

Gelöst wurde dieses Problem durch die Definition der abstrakten Klasse *VisObject*. Sie stellt eine Update-Funktion zur Verfügung, mit deren Hilfe die aktuellen Zustandswerte der jeweiligen Simulationsobjekte wie Körper oder Constraints an ihre graphischen Äquivalente übertragen werden kön-

nen, und die nach jedem Simulationsschritt aufgerufen wird. Die zu den verschiedenen Objekttypen gehörenden Unterklassen von *VisObject* greifen dazu wie in Abbildung A.3 gezeigt via Delegation auf das jeweilige Objekt zu[1].

Diese Unterklassen von *VisObject* leiten zudem die Resultate der verschiedenen Interaktionstechniken, die im wesentlichen in der Klasse *InvScene* implementiert sind, an die jeweiligen Simulationsobjekte weiter. *InvScene* dient auch als Adapter für *Open Inventor*, um die Klasse *Scene* unabhängig von dieser Bibliothek implementieren zu können. In *Scene* werden die Benutzungseingaben in Form von Callbacks verwaltet und an die entsprechenden Darstellungs- und Simulationsmodule weitergeleitet. Die Benutzungsoberfläche wurde mit Hilfe der Entwicklungsumgebung *RapidApp* auf der Basis des XWindows-Fenstersystems implementiert[2]. Sie ermöglicht eine einfache und flexible Bedienung des Systems.

[1] Um die Implementationsdetails und die Abhängigkeit von *Open Inventor* zu verstecken, fungiert zudem eine abstrakte Klasse als Wrapper für *VisObject*.

[2] Ein Problem bestand dabei in der Notwendigkeit, von der Klasse *Scene* aus bestimmte Interaktionsfenster öffnen und schließen zu können sowie ihnen bestimmte Parameterobjekte zu übergeben, ohne eine zu enge Kopplung an die XWindows-Umgebung eingehen zu müssen. Daher wurde eine Wrapper-Klasse als Basisklasse der XWindows-Klassen spezifiziert, die diese Funktionalitäten bereitstellt.

Anhang B

Ein Modell zur Echtzeit-Simulation

In diesem Kapitel soll ein Modell zur Echtzeit-Simulation im Rahmen einer physikalisch basierten Animation vorgestellt werden, die sich auf eine schrittweise Auswertung von Entwicklungsgleichungen stützt[1]. Zunächst müssen dazu zwei verschiedene Zeitbegriffe eingeführt und der Begriff der Echtzeit erläutert werden. Mit der Simulationsrate, der Oversampling-Rate und der Framerate werden dann drei wichtige Simulationsparameter besprochen und ihre gegenseitigen Abhängigkeiten untersucht.

Systemzeit, Realzeit, Echtzeit

Eine physikalische Simulation besteht in dem hier betrachteten Rahmen in der automatischen Auswertung zeitlicher Entwicklungsgleichungen. Die *Systemzeit* t_S soll dem in diesen Gleichungen enthaltenen Zeitparameter t entsprechen[2]. Bei einer schrittweise durchgeführten Simulation wird dieser Parameter bei jedem Simulationsschritt um einen bestimmten Δt-Wert erhöht. Diese Diskretisierung ist z.B. ein integraler Bestandteil der numerischen Lösung gewöhnlicher Differentialgleichungen (vergl. Anhang F.1). Der Schrittweitenwert kann große Auswirkungen auf das Simulationsergebnis haben, wie unten genauer besprochen wird. Als Einheit des Zeitparameters t soll die Sekunde gewählt werden, d.h. einem Wert von $t = 1$ entspricht die Systemzeit $t_S = 1\,s$, was lediglich einer nützlichen Bezeichnungsart entspricht.

Mit *Realzeit* t_R soll die tatsächlich ablaufende Zeit bezeichnet werden, die während einer Simulation vergeht. Sie hängt u.a. von der Geschwindigkeit der Simulation und damit auch von dem Rechnersystem ab, auf dem die Simulation durchgeführt wird. Sie kann während einer Simulation schneller oder auch langsamer als die Systemzeit ablaufen.

Falls beide Zeiten genau gleichschnell ablaufen, soll der Begriff *Echtzeit-Zustand* oder kurz *Echtzeit* verwandt werden. Er entspricht somit keinem Zeitparameter, sondern bezeichnet eine bestimmte Konstellation der Simulationsumgebung. Im Echtzeit-Zustand läuft die Simulation genauso schnell wie das reale System ab, dessen Zeitentwicklung durch die Simulation nachgebildet wird. Dieses Verhalten ist bei der physikalisch basierten Animation in der Regel sehr wünschenswert, da ein unrealistisch langsames oder schnelles Zeitverhalten vermieden wird und sich der visuelle Eindruck der Animation direkt überprüfen läßt. Dieses Ziel kann allerdings nicht immer erreicht werden. Zur Klärung dieses Sachverhaltes ist die Einführung einiger weiterer Begriffe erforderlich.

[1] Eine ähnliche, aber weitaus weniger detaillierte Diskussion dieser Aspekte findet sich in [DC95].

[2] Häufig ist der Zeitparameter nur implizit in den Entwicklungsgleichungen enthalten. Dieser Umstand ist aber für die hier angestellten Überlegungen nicht relevant, da sich dieser Parameter auch über die *Änderungsraten* des Systems identifizieren läßt.

Simulationsrate, Oversampling, Framerate

Die oben bereits erwähnte Schrittweite Δt, mit der die Lösung der Bewegungsgleichungen entwickelt wird, ist ein frei wählbarer Parameter der Simulation. Er beeinflußt den Simulationsverlauf in zweierlei Weise: Je kleiner der Wert für Δt gewählt wird, um so größer ist der zeitliche Aufwand für die Simulation. Große Δt-Werte führen auf der anderen Seite auch zu größeren Ungenauigkeiten bei der numerischen Lösung oder sogar zu einem instabilen Lösungsverhalten, das einen Abbruch der Simulation zur Folge haben kann (vergl. Anhang F.1). Der geeigneten Wahl dieser Schrittweite kommt daher eine große Bedeutung zu.

Die sogenannte *Simulationsrate* R_S gibt nun an, wieviele Simulationsschritte in einer Sekunde Systemzeit durchgeführt werden: $R_S = 1/\Delta t$. Diese Größe ist damit ein Maß für die Ablaufgeschwindigkeit der Simulation, aber auch für die verfahrensbedingten Ungenauigkeiten und Instabilitäten. Nach den Erfahrungen mit EMPHAS sind für viele typischerweise in der Animation eingesetzte mechanische Systeme Simulationsraten von ca. 10 - 30 notwendig, um eine stabile Simulation zu gewährleisten, was aber von sehr vielen Faktoren abhängig ist.

Um die Simulationsergebnisse graphisch darzustellen, kann der Darstellungsschritt, d.h. die Übertragung der berechneten Zustandswerte auf die graphischen Repräsentanten der Körper (vergl. Abschnitt 6.5.2.2), direkt nach jedem Simulationsschritt vollzogen werden. Oftmals ist es aber sinnvoll, zwischen zwei Darstellungsschritten, und damit zwischen zwei Frames, mehrere Simulationsschritte durchzuführen, um z.B. die Stabilität des Verfahrens zu erhöhen. Man spricht in diesem Fall von *Oversampling* (Überabtastung). Die Anzahl von Simulationsschritten, nach denen ein Darstellungsschritt durchgeführt wird, soll mit n_O bezeichnet werden, wobei $n_O \geq 1$ gilt. Pro Sekunde Systemzeit ergeben sich demnach durchschnittlich $\frac{R_S}{n_O}$ Darstellungsschritte.

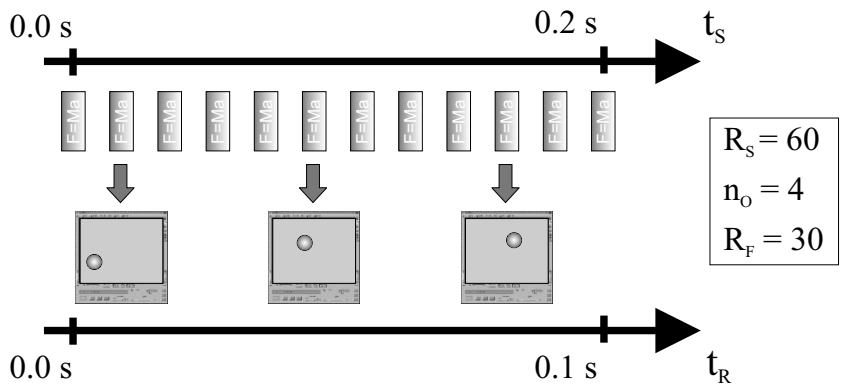

Abbildung B.1: Belegung der Simulationsparameter R_S, n_O und R_F für einen Simulationsablauf, in dem nach jedem vierten Simulations- ein Darstellungsschritt erfolgt.

Eine andere wichtige Größe ist die Anzahl der Frames, die in einer Sekunde realer Zeit erzeugt werden. Sie wird als *Framerate* R_F bezeichnet. Bei einer Framerate, die deutlich unter einem Wert von ca. 30 liegt, werden die Frames als einzelne Bilder aufgelöst, d.h. die Bewegungen werden „ruckelig“. Die Parameter R_S, n_O und R_F stellen daher wichtige Kenngrößen bei der physikalisch basierten Animation dar.

Ein Beispiel für die Belegung dieser Simulationsparameter ist in Abbildung B.1 dargestellt. In 0.2 Sekunden Systemzeit werden hierbei 12 Simulationsschritte durchgeführt, wobei nach jedem 4. Schritt ein Darstellungsschritt und damit eine graphische Ausgabe erfolgt. Wenn die Systemzeit wie in dem Beispiel gezeigt doppelt so schnell wie die Realzeit abläuft (was vom verwendeten Rechnersystem und vielen anderen Faktoren abhängt), ergibt sich daraus eine Rate von 30 Darstellungs-

schritten pro Sekunde Realzeit. Falls der Zeitaufwand für das Rendern des Frames vernachlässigt werden kann, würde demnach eine Framerate von 30 resultieren.

Beziehungen zwischen den Simulationsparametern

Es stellt sich nun die Frage nach den gegenseitigen Abhängigkeiten dieser Simulationsparameter. Für die Steuerung der Simulation sind vor allem die Größen der Simulationsrate und der Framerate sowie die Bedingungen für ein Echtzeit-Verhalten von Bedeutung. Zunächst muß daher untersucht werden, welche reale Zeit für die Simulation einer Sekunde Systemzeit vergeht.

In einer Sekunde Systemzeit werden R_S Simulations- und durchschnittlich $\frac{R_S}{n_O}$ Darstellungsschritte durchgeführt. Die reale zeitliche Dauer eines einzelnen Simulationsschrittes hängt zwar z.B. vom simulierten System und von der Hardware-Umgebung ab, nicht aber von der Rate R_S selber[3]. Dieses Zeitintervall, das während einer Simulation nur empirisch bestimmt werden kann, soll mit T_S bezeichnet werden. Analoges gilt für die Zeitdauer T_D, die sowohl die Zeitdauer eines Darstellungsschrittes als auch die (in der Regel wesentlich größere) Zeit zum Rendern eines Frames umfassen soll. Einer Sekunde Systemzeit entspricht dann $R_S \cdot T_S + \frac{R_S}{n_O} \cdot T_D$ Sekunden Realzeit, d.h.

$$t_R = R_S \, (T_S + \frac{1}{n_O} \cdot T_D) \, t_S \, . \tag{B.1}$$

Das Verhältnis von Realzeit und Systemzeit ist also direkt proportional zur Simulationsrate. Aus dieser Beziehung läßt sich auch die Bedingung für ein Echtzeit-Verhalten ablesen: Wenn die Simulationsrate auf den Wert $(T_S + \frac{T_D}{n_O})^{-1}$ gesetzt wird, laufen Realzeit und Systemzeit genau gleich schnell ab.

Da $\frac{R_S}{n_O}$ die Anzahl der Darstellungsschritte pro Sekunde Systemzeit wiedergibt, kann die Framerate, die sich auf die Realzeit bezieht, als $R_F = \frac{R_S}{n_O} \frac{t_S}{t_R}$ geschrieben werden. Durch Einsetzen von Gleichung (B.1) ergibt sich hieraus

$$R_F = \frac{1}{n_O T_S + T_D} \, . \tag{B.2}$$

Die Framerate R_F ist demnach unabhängig von der Simulationsrate R_S und kann durch den Wert der Oversampling-Zahl n_O beeinflußt werden.

Die in diesem Kapitel abgeleiteten Beziehungen zwischen den Simulationsparameter R_S, n_O und R_F, insbesondere Gleichung (B.2) und die oben angeführte Bedingung für ein Echtzeit-Verhalten, stellen eine wichtige Hilfe für die geeignete Festlegung dieser Größen im Rahmen der physikalisch basierten Animationserstellung dar.

[3] Verfahren mit adaptiven Schrittweiten stellen hier eine gewisse Ausnahme dar, was in dieser Analyse aber vernachlässigt werden soll.

Anhang C

Grundzüge der Klassischen Mechanik

Physikalische Theorien, die als wesentlichen Bestandteil ein mathematisches Modell enthalten, bilden die Grundlage physikalischer Simulationen und damit auch der physikalisch basierten Animation. In diesem Kapitel soll die Theorie umrissen werden, die für computergraphische Anwendungen besonders häufig eingesetzt wird: die Klassische (Newtonsche) Mechanik[1]. Zurückgegriffen wurde dazu auf die klassischen Lehrbücher [Gol89] und [Sch88]. Eine Einführung in die Dynamik von Mehrkörpersystemen wird in [Wit77] und [Sha98] gegeben.

Die Dynamik von Punktmassen

Newtons Bewegungsgesetz für ein einzelnes Punktteilchen unter dem Einfluß einer äußeren Kraft $\mathbf{F}$ lautet $\mathbf{F} = \dot{\mathbf{p}}$. Dabei ist $\mathbf{p} := m\mathbf{v}$ der Impuls des Teilchens, $\mathbf{v} := \dot{\mathbf{r}}$ seine Geschwindigkeit, $\mathbf{r}$ sein Ortsvektor und m seine Masse. Bei einer konstanten Masse, was in den meisten Fällen gegeben ist und im folgenden stets angenommen werden soll, erhält man somit die bekannte Form

$$\mathbf{F} = m\,\mathbf{a}$$

mit der Beschleunigung $\mathbf{a} := \dot{\mathbf{v}} = \ddot{\mathbf{r}}$. Einem bewegten Teilchen kann man außerdem einen *Drehimpuls* und ein *Drehmoment* zuordnen, die sich auf seine Drehbewegung bezüglich eines bestimmten Punktes im dreidimensionalen Raum beziehen. Der Drehimpuls bzgl. eines Punktes $\mathbf{P}$ wird durch die Beziehung

$$\mathbf{L} := (\mathbf{r} - \mathbf{P}) \times \mathbf{p}$$

festgelegt. Das Drehmoment bzgl. $\mathbf{P}$ ist durch

$$\mathbf{N} := (\mathbf{r} - \mathbf{P}) \times \mathbf{F}$$

definiert. Setzt man die Newtonsche Beziehung zwischen $\mathbf{p}$ und $\mathbf{F}$ ein, ergibt sich $\dot{\mathbf{L}} = \dot{\mathbf{r}} \times (m\dot{\mathbf{r}}) + (\mathbf{r} - \mathbf{P}) \times \dot{\mathbf{p}} = (\mathbf{r} - \mathbf{P}) \times \mathbf{F}$, d.h.

$$\dot{\mathbf{L}} = \mathbf{N}\,.$$

Diese Beziehung ist vor allem als Gesetz der Drehimpulserhaltung bekannt: Bei verschwindenden äußeren Drehmomenten bleibt der Drehimpuls konstant.

[1] Ein etwas ausführlicherer Überblick über dieses Thema, bei dem auch der Lagrange-Formalismus erläutert wird, findet sich in [Wag95].

Die Dynamik von Punktmassen-Systemen

Es soll nun ein System aus n Punktteilchen betrachtet werden. Das i-te Teilchen dieses Systems mit der Masse m_i und der Beschleunigung $\mathbf{a}_i$ wird dabei sowohl von der äußeren Kraft $\mathbf{F}_i$ als auch von Kräften $\mathbf{F}_{ij}$, die von einem anderen Teilchen j herrühren, beeinflußt. Es gilt somit

$$\mathbf{F}_i + \sum_{j=1}^{n} \mathbf{F}_{ij} = m_i \mathbf{a}_i \,, \quad i = 1, \ldots, n \,. \tag{C.1}$$

Diese Gleichung läßt sich durch die Einführung des Begriffs des *(Massen-) Schwerpunkts* vereinfachen. Der Schwerpunktsvektor $\mathbf{r}_{SP}$ ist durch

$$\mathbf{r}_{SP} := \frac{\sum_{j=1}^{n} m_i \mathbf{r}_i}{M} \,, \quad M := \sum_{j=1}^{n} m_i \,.$$

definiert. Summiert man nun Gleichung (C.1) über alle i und beachtet, daß der Term $\sum_{i,j=1}^{n} \mathbf{F}_{ij}$ wegen des „actio = reactio“ - Gesetzes verschwindet, erhält man $\sum_{i=1}^{n} \mathbf{F}_i = \sum_{i=1}^{n} m_i \ddot{\mathbf{r}}_i$, d.h.

$$M \ddot{\mathbf{r}}_{SP} = \sum_{i=1}^{n} \mathbf{F}_i := \mathbf{F} \,. \tag{C.2}$$

Der Schwerpunkt bewegt sich also so, als ob die ganze Masse in ihm konzentriert wäre, die inneren Kräfte haben keinen Einfluß auf seine Bewegung.

Der Gesamtimpuls des Systems hat mit Hilfe dieser Größe die Form

$$\mathbf{P} := \sum_{i=1}^{n} m_i \dot{\mathbf{r}}_i = M \dot{\mathbf{r}}_{SP} \,. \tag{C.3}$$

Der Gesamtdrehimpuls setzt sich aus dem Drehimpuls des Schwerpunkts (relativ zum Weltkoordinatensystem) und den Drehimpulsen der Teilchen bezüglich des Schwerpunkts zusammen:

$$\mathbf{L} = \mathbf{r}_{SP} \times \mathbf{P} + \sum_{i=1}^{n} (\mathbf{r}_i - \mathbf{r}_{SP}) \times (m_i (\dot{\mathbf{r}}_i - \dot{\mathbf{r}}_{SP})) \,.$$

Für ihn kann in ganz analoger Weise die Gleichung

$$\dot{\mathbf{L}} = \sum_{i=1}^{n} \mathbf{N}_i =: \mathbf{N}$$

abgeleitet werden. Diese Beziehungen bilden auch die Grundlage für die dynamische Beschreibung eines anderen wichtigen Körpertyps: der starre Körper.

Die Kinematik starrer Körper

Dreidimensionale Körper, deren geometrische Form zeitlich konstant bleibt, nennt man *starre* Körper. Gerade für die Computergraphik haben sie eine große Bedeutung, da sich viele alltägliche Objekte näherungsweise wie starre Körper verhalten und ihre Modellierung wesentlich einfacher als die von deformierbaren Körpern ist.

Ein formkonstanter Körper hat 6 Freiheitsgrade, 3 bezüglich der Translations und 3 bezüglich der Rotation. In den sogenannten *Referenzpunktkoordinaten* wird er durch die Ortsangabe eines Körperpunktes und den Rotationszustand relativ zu diesem Punkt beschrieben. Normalerweise wird der Schwerpunkt des Körpers als ein solcher Referenzpunkt gewählt, da er in eindeutiger Weise ausgezeichnet ist und die Beschreibung der Rotation vereinfacht wird. Sie kann z.B. mit Hilfe der

(3×3)-dimensionalen Drehmatrix **R** (vergl. Anhang E) ausgedrückt werden. Für die Beschreibung der Drehgeschwindigkeit läßt sich die Größe der Winkelgeschwindigkeit ω einsetzen. Dieser dreidimensionale Vektor legt durch seine Richtung die Drehachse fest, während sein Betrag die Größe der Drehgeschwindigkeit angibt. Abbildung C.1 zeigt die Interpretation dieser Größen, wobei der Ortsvektor des Schwerpunktes mit **r** und die zugehörige Translationsgeschwindigkeit mit **v** bezeichnet wurden.

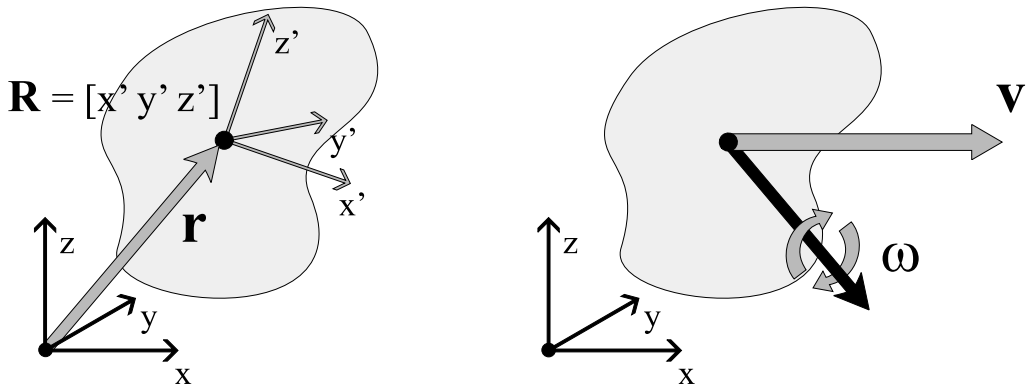

Abbildung C.1: Beschreibung des starren Körpers in Referenzpunktkoordinaten.

Die drei Spaltenvektoren der Matrix **R** lassen sich als die Koordinatenachsen des körperfesten Koordinatensystems bezüglich des Weltkoordinatensystems interpretieren. Eine für viele Zwecke günstigere Beschreibung der Körperrotation läßt sich allerdings mit *Quaternionen* erreichen, wie in Anhang E näher erläutert wird.

Der lineare Impuls eines starren Körpers entspricht dem eines Punktteilchens, in dem die gesamte Masse des Körpers konzentriert ist:

$$\mathbf{p} = m\mathbf{v}\,.$$

Herleiten kann man diese Beziehung zum Beispiel unter Benutzung von Gleichung (C.3), indem man einen starren Körper als Grenzfall eines Vielteilchensystems ansieht.

Die Beschreibung des Drehimpulses eines starren Körpers stellt eine kompliziertere Aufgabe dar. Vor allem muß dabei beachtet werden, daß Massen einen größeren Beitrag zum Drehimpuls liefern, wenn sie weit von der Drehachse entfernt sind. Hier spielt also nicht nur die Gesamtmasse, sondern auch die Massen*verteilung* eine Rolle. Die Beziehung zwischen Drehimpuls **L** und Winkelgeschwindigkeit ω hat die Form

$$\mathbf{L} = \mathbf{I}\,\omega\,,$$

wobei **I** keinen Skalar, sondern eine (3×3)-Matrix darstellt, den sogenannten *Trägheitstensor*. Seine Komponenten sind über die Beziehung

$$I_{ij} := \int_V \rho(\mathbf{r})\,(\mathbf{r}^2\,\delta_{ij} - r_i\,r_j)\,d\mathbf{r}\,,$$

definiert, wobei $\rho(\mathbf{r})$ die (möglicherweise ortsabhängige) Massendichte des Körpers und V sein Volumen darstellt. Wie jede reelle symmetrische Matrix läßt sich **I** durch eine orthogonale Transformation auf Diagonalform bringen, d.h. man kann ein Koordinatensystem wählen, in dem **I** die Form

$$\mathbf{I} = \begin{pmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{pmatrix}$$

hat. Die Diagonalelemente nennt man *(Haupt-) Trägheitsmomente*. Der Nullpunkt dieses Koordinatensystems liegt im Körperschwerpunkt und die Achsen auf den sogenannten *Hauptträgheitsachsen*, die bei symmetrischen Körpern mit den Symmetrieachsen zusammenfallen.

In den meisten Fällen ist die Massendichte konstant und kann daher vor das Integral gezogen werden. Die Trägheitsmomente haben dann die Form

$$I_1 = mK_1^2\,,\; I_2 = mK_2^2\,,\; I_3 = mK_3^2\,,$$

wobei m die Gesamtmasse des Körpers ist und K_1, K_2, K_3 rein geometrische Größen darstellen, die nur von der Form des Körpers abhängen. Einige Beispiele für diese Formfaktoren sind im folgenden angegeben:

- Kugel mit Radius r: $K_1 = K_2 = K_3 = \frac{2}{5}r^2$
- Zylinder mit Radius r und Länge l: $K_1 = K_2 = \frac{r^2}{4} + \frac{l^2}{12}$, $K_3 = \frac{r^2}{2}$
- Quader mit Kantenlängen a, b, c: $K_1 = \frac{b^2+c^2}{12}$, $K_2 = \frac{a^2+c^2}{12}$, $K_3 = \frac{a^2+b^2}{12}$

Der Trägheitstensor eines Körpers, der durch das Zusammenfügen mehrerer starrer Körper entsteht, ist zudem gleich der Summe der Trägheitstensoren dieser Körper.

Die Dynamik starrer Körper

Der Schwerpunkt eines starren Körpers bewegt sich unabhängig von seiner Rotation. Diese wichtige Eigenschaft entspricht der oben für Punktteilchensysteme abgeleiteten Beziehung (C.2). Bei einer äußeren Kraft $\mathbf{F}$ gilt daher wie im Fall eines Punktteilchens

$$\mathbf{F} = \dot{\mathbf{p}} = m\ddot{\mathbf{r}}\,.$$

Analog gilt bei einem äußeren Drehmoment $\mathbf{N}$ die Beziehung

$$\mathbf{N} = \dot{\mathbf{L}} = \frac{d}{dt}\left(\mathbf{I}(t)\,\omega(t)\right)\,.$$

Diese Gleichung ist allerdings schwer auszuwerten, da sich die darin vorkommenden Größen sowie die zeitliche Ableitung auf das Weltkoordinatensystem beziehen. Günstiger ist es, ein körperfestes Koordinatensystem mit dem Schwerpunkt als Nullpunkt als Grundlage zu wählen. Der Trägheitstensor ist in körperfesten Koordinaten zeitunabhängig, da die Massenverteilung des Körpers zeitlich konstant ist. Bei der Wahl der Hauptträgheitsachsen als Koordinatenachsen ist er sogar diagonal. Die hierfür notwendige Koordinatentransformation liefert die folgende Beziehung, die als *Euler-Gleichung* bekannt ist:

$$\mathbf{N}_K = \mathbf{I}_K\,\dot{\omega}_K + \omega_K \times (\mathbf{I}_K\,\omega_K)\,.$$

Der Index K soll dabei andeuten, daß sich alle Größen auf das körperfeste Koordinatensystem beziehen.

Mit Hilfe dieser Beziehungen kann das Bewegungsverhalten starrer Körper eindeutig berechnet werden, wenn seine Masseneigenschaften, die äußeren Kräfte und geeignete Anfangs- oder Randwerte bekannt sind.

Anhang D

Ergänzungen zur Kontaktbehandlung

Inhalt dieses Kapitels sind ergänzende Ausführungen für die in Abschnitt 2.4 besprochene automatische Kontaktbehandlung. Zunächst soll die Form des Kraftstoßes aus Gleichung (2.3) und (2.4) abgeleitet werden. Anschließend wird die Bestimmung von Kompensationskräften im Rahmen des Ruhekontaktproblems genauer untersucht. Für beide Probleme wurde dabei auf die Arbeit [Bar95b] zurückgegriffen.

D.1 Bestimmung von Kollisions-Kraftstößen

Eine Kollision läßt sich durch die Anwendung eines *Kraftstoßes* **K** simulieren, der eine Impulsdifferenz bewirkt:

$$\mathbf{p} = \mathbf{p}^0 + \mathbf{K} \,.$$

Ein Kraftstoß kann als Grenzfall einer unendlich großen Kraft gesehen werden, die in einem unendlich kleinen Zeitintervall wirkt. Er ist daher vor allem für die Kollisionssimulation *starrer* Körper geeignet, die durch eine beliebig kleine Kontaktzeit gekennzeichnet sind.

Für die nähere Bestimmung des Kraftstoßes kann man sich einige grundlegende physikalische Eigenschaften von Kollisionen zu Nutze machen, die nun erläutert werden sollen:

- Die Kollisionskraft wirkt bei allen Kollisionen senkrecht zur Kontaktebene. Wenn **n** die Flächennormale dieser Ebene darstellt, hat **K** demnach die Form

  $$\mathbf{K} = k\,\mathbf{n} \,,$$

  wobei k eine (zunächst unbekannte) skalare Größe darstellt.

- Aufgrund der Impulserhaltung muß der Impulsübertrag für beide Körper A und B symmetrisch sein. Für die jeweiligen Kraftstöße $\mathbf{K}_A$ und $\mathbf{K}_B$ folgt daher

  $$\mathbf{K}_A = -\mathbf{K}_B \,.$$

  Für die Änderungen der Translationsgeschwindigkeiten ergeben sich somit die Ausdrücke

  $$\mathbf{v}_A \quad = \quad \mathbf{v}_A^0 + \frac{k}{m_A}\,\mathbf{n}$$

$$\mathbf{v}_B = \mathbf{v}_B^0 - \frac{k}{m_B}\mathbf{n}\,.$$

Dabei sind $\mathbf{v}_A^0$ und $\mathbf{v}_B^0$ die Geschwindigkeiten vor dem Stoß, m_A und m_A stellen die Körpermassen dar. Das Problem reduziert sich demnach auf die Angabe eines Ausdrucks für den Skalar k, der mit Hilfe der obigen Beziehungen die neuen Körpergeschwindigkeiten eindeutig festlegt.

- Für reibungslose Kollisionen gibt es ein empirisches Gesetz, das die Relativgeschwindigkeit $v_{rel} := \mathbf{n}\,(\mathbf{v}_A - \mathbf{v}_B)$ der beiden Körper vor und nach dem Stoß in Beziehung setzt ([Bar95b]):

$$v_{rel} = -\varepsilon\, v_{rel}^0\,.$$

Der Parameter ε stellt dabei die in Abschnitt 2.4.2 erläuterte Stoßzahl dar.

Mit diesen Beziehungen kann nun einen Ausdruck für k hergeleitet werden. Dies soll am Beispiel des zentrischen Stoßes erfolgen, bei dem keine Drehmomente frei werden (vergl. Abschnitt 2.4.2). Die entsprechende Herleitung für den exzentrischen Stoß würde völlig analog verlaufen.

Am einfachsten gelangt man durch die Auswertung von $-\varepsilon\, v_{rel}^0 = v_{rel} := \mathbf{n}\,(\mathbf{v}_A - \mathbf{v}_B)$ zu einem Ausdruck für k:

$$\begin{aligned}
-\varepsilon\, v_{rel}^0 &= \mathbf{n}\,(\mathbf{v}_A - \mathbf{v}_B) \\
&= \mathbf{n}\,((\mathbf{v}_A^0 + \frac{k}{m_A}\mathbf{n}) - (\mathbf{v}_B^0 - \frac{k}{m_B}\mathbf{n})) \\
&= \mathbf{n}\,(\mathbf{v}_A^0 - \mathbf{v}_B^0) + \mathbf{n}\,(\frac{1}{m_A} + \frac{1}{m_B})\,k\,\mathbf{n} \\
&= v_{rel}^0 + (\frac{1}{m_A} + \frac{1}{m_B})\,k\,.
\end{aligned}$$

Aufgelöst nach k erhält man somit das Ergebnis

$$k = -\frac{(1+\varepsilon)\, v_{rel}^0}{\frac{1}{m_A} + \frac{1}{m_B}}\,.$$

Dies ist die gewünschte Beziehung zwischen k und den Massen und Anfangsgeschwindigkeiten der kollidierenden Körper.

Für die Betrachtung exzentrischer Stöße bleiben die grundlegenden physikalischen Beziehungen $\mathbf{K} = k\,\mathbf{n}$, $\mathbf{K}_A = -\mathbf{K}_B$ und $v_{rel} = -\varepsilon\, v_{rel}^0$ erhalten. Zusätzlich zum (linearen) Kraftstoß $\mathbf{K}$ wirkt in diesem Fall aber ein „Drehmoment-Stoß“

$$\mathbf{N} = (\mathbf{r} - \mathbf{r}_{SP}) \times \mathbf{K}$$

auf die beiden Körper. Dabei bezeichnet $\mathbf{r}$ den Kontaktpunkt und $\mathbf{r}_{SP}$ den Ortsvektor des Körperschwerpunktes. Dieser Ausdruck verschwindet bei zentrischen Stößen, bei denen der Vektor $\mathbf{r} - \mathbf{r}_{SP}$ parallel zur Stoßnormalen und damit zum Kraftstoß verläuft.

Bei exzentrischen Stößen muß also zusätzlich die Änderung der Winkelgeschwindigkeit in der Form $\omega = \omega^0 + \mathbf{I}^{-1}\,\mathbf{N}$ berücksichtigt werden, wobei $\mathbf{I}$ den Trägheitstensor des Körpers darstellt, der in Anhang C beschrieben wurde. Zusammengenommen ergeben sich somit die Beziehungen

$$\begin{aligned}
\omega_A &= \omega_A^0 + \mathbf{I}_A^{-1}\,(\mathbf{d}_A \times \mathbf{n})\,k \\
\omega_B &= \omega_B^0 - \mathbf{I}_B^{-1}\,(\mathbf{d}_B \times \mathbf{n})\,k
\end{aligned}$$

mit $\mathbf{d}_A := \mathbf{r} - \mathbf{r}_{SP}^A$ und $\mathbf{d}_B := \mathbf{r} - \mathbf{r}_{SP}^B$.

Ein Ausdruck für k kann unter Beachtung der Beziehung $\dot{\mathbf{r}} = \mathbf{v} + \omega \times (\mathbf{r} - \mathbf{r}_{SP})$ nach dem gleichen Muster wie im Falle des zentrischen Stoßes hergeleitet werden (siehe hierfür [Bar95b]). Das Ergebnis ist

$$k = -\frac{(1+\varepsilon)\, v_{rel}^0}{\frac{1}{m_A} + \frac{1}{m_B} + \mathbf{n}\,(\mathbf{I}_A^{-1}(\mathbf{d}_A \times \mathbf{n})) \times \mathbf{d}_A + \mathbf{n}\,(\mathbf{I}_B^{-1}(\mathbf{d}_B \times \mathbf{n})) \times \mathbf{d}_B} \,.$$

Mit Hilfe dieser Beziehung kann somit auch das Problem des exzentrischen Stoßes in geschlossener Form gelöst werden.

D.2 Bestimmung von Kompensationskräften für Ruhekontakte

Beim Vorhandensein von Ruhekontakten, d.h. Berührungspunkten mit verschwindender Relativgeschwindigkeit senkrecht zur Kontaktebene, lassen sich im Rahmen der physikalisch basierten Animation Kompensationskräfte berechnen, die eine Durchdringung der Körper verhindern. Diese Kräfte haben dabei an jedem Punkt die Form

$$\mathbf{F}_i = f_i\, \mathbf{n}_i \,,$$

d.h. sie wirken in Richtung der Kontaktnormalen $\mathbf{n}_i$. Die Skalare f_i müssen daher bestimmt werden, um die Kompensationskräfte anwenden zu können.

Um einen Ausdruck für diese Kraftskalare zu erlangen, lassen sich drei Bedingungen für eine plausible Behandlung von Ruhekontakten formulieren. Dazu soll das Abstandsmaß

$$d_i := \mathbf{n}_i\,(\mathbf{r}_i^A - \mathbf{r}_i^B)$$

zwischen zwei Körpern A und B eingeführt werden, wobei $\mathbf{r}_i^A$ und $\mathbf{r}_i^B$ die Ortsvektoren derjenigen Punkte von Körper A bzw. B darstellen, die miteinander in Kontakt stehen. Nach Voraussetzung ist zum Zeitpunkt des Kontaktes $d_i = 0$ und $\dot{d}_i = v_i^{rel} = 0$. Die Bedingungen haben nun die folgende Form ([Bar94], [Bar95b]):

- $\ddot{d}_i \geq 0$
  Eine negative Relativbeschleunigung würde im nächsten Zeitschritt die Körper aufeinanderzu streben lassen, so daß es zu einer Durchdringung käme. Dieser Fall muß daher ausgeschlossen werden.

- $f_i \geq 0$
  Die Constraint-Kräfte sollen nur abstoßend auf die Körper wirken. Eine Kontaktkraft, die die Körper wie ein Gummiband zusammenhält, wäre höchst unplausibel.

- $f_i\, \ddot{d}_i = 0$
  Diese Beziehung sagt aus, daß entweder die Kontaktkräfte f_i oder die Relativbeschleunigungen $\ddot{d}_i$ verschwinden müssen. Auf diese Weise stellt man sicher, daß bei $\ddot{d}_i > 0$ keine Kontaktkräfte wirken. Dieser Fall entspricht einem abbrechenden Kontakt, bei dem die Körper auseinander streben. Eine nicht verschwindende Kontaktkraft würde dann eine Beschleunigung der Körper bewirken und so die physikalische Simulation verfälschen.

Um diese Bedingungen erfüllen zu können, benötigt man noch eine Beziehung zwischen den Kontaktkräften f_i und den Relativbeschleunigungen $\ddot{d}_i$. In diese Beziehung gehen die Eigenschaften

der Körper (Massen bzw. Massenverteilungen, Zustandsvektoren und Geschwindigkeiten) sowie die Geometrie der Kontaktpunkte ein. Außerdem muß hierfür das Newton'sche Bewegungsgesetz $\mathbf{F} = m\,\mathbf{a}$ berücksichtigt werden, das Kräfte und Beschleunigungen miteinander in Beziehung setzt. Als allgemeine Form ergibt sich dabei die Gleichung

$$\begin{pmatrix} \ddot{d}_1 \\ \vdots \\ \ddot{d}_N \end{pmatrix} = \mathbf{K} \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix} + \mathbf{b}\,.$$

Dabei stellt $\mathbf{K}$ bei N Kontaktpunkten eine $(N \times N)$-Matrix und $\mathbf{b}$ einen N-dimensionalen Vektor dar. Die Herleitung dieser Gleichung und die explizite Ausformulierung der Terme $\mathbf{K}$ und $\mathbf{b}$ läßt sich (mitsamt einer Programmcode-Umsetzung) in [Bar95b] nachlesen und soll an dieser Stelle nicht ausgeführt werden.

Das so formulierte Problem, das in einer Lösung für die Kraftskalare f_i unter Berücksichtigung der obigen Bedingungen besteht, stellt ein sogenanntes *Quadratic Programming*-Problem dar. Zu seiner Bewältigung können spezielle numerische Verfahren eingesetzt werden, die auf solche Probleme zugeschnitten sind, wie in [Bar95a] näher beschrieben wird. Sie liefern explizite Werte für die Kontaktkräfte f_i, die alle genannten Bedingungen für eine plausible Behandlung von Ruhekontakten erfüllen.

Anhang E

Rotationsbeschreibung mit Quaternionen

Die mathematische Beschreibung einer dreidimensionalen Rotationstransformation stellt eine relativ komplizierte Aufgabe dar. Mit den Quaternionen sollen in diesem Kapitel mathematische Größen vorgestellt werden, die sich für eine solche Aufgabe besonders gut eignen. Weitergehende Erläuterungen für den Einsatz von Quaternionen findet man in [Sho85] und [Mai90]. Ein Vergleich mit neueren Ansätzen wird in [Gra98] durchgeführt.

Eine übliche Art zur Beschreibung dreidimensionaler Rotationen ist die Verwendung einer (3×3)-dimensionalen Rotationsmatrix **R**, mit der ein Vektor **p** über

$$\mathbf{p}' = \mathbf{R}\mathbf{p}\,.$$

in den transformierten Vektor $\mathbf{p}'$ überführt wird. Die 9 Komponenten dieser Matrix lassen sich dabei durch 3 Euler-Winkel α, β und γ parametrisieren, die orthogonalen Rotationen bezüglich der *x*-, *y*- und *z*-Achse des Weltkoordinatensystems entsprechen. Mit der Winkelgeschwindigkeit ω stehen diese Größen über die Gleichungen

$$\begin{aligned}\omega_x &= -\dot{\alpha}\sin\beta\cos\gamma + \dot{\beta}\sin\gamma\\ \omega_y &= \dot{\alpha}\sin\beta\sin\gamma + \dot{\beta}\cos\gamma\\ \omega_z &= \dot{\alpha}\cos\beta + \dot{\gamma}\end{aligned}$$

in Verbindung und für die zeitliche Ableitung der Rotationsmatrix läßt sich

$$\dot{\mathbf{R}}\mathbf{p} = \omega \times (\mathbf{R}\mathbf{p}) \tag{E.1}$$

ableiten (siehe z.B. [Wit77]).

Diese Art der Rotationsbeschreibung ist allerdings durch eine Reihe von Problemen gekennzeichnet, die insbesondere für computergraphische Anwendungen ins Gewicht fallen (vergl. [Sho85], [Gra98]):

- Koordinatensingularitäten
  Wenn zwei der durch die Euler-Winkel beschriebenen Rotationsachsen aufeinanderfallen, geht ein Freiheitsgrad verloren, es liegt also eine Koordinatensingularität vor. Dieses Problem tritt bei allen dreiparametrigen Rotationsbeschreibungen auf.

- Interpolierbarkeit
  Rotationen auf der Grundlage von Euler-Winkeln lassen sich sehr schlecht interpolieren. Die lineare Interpolationen dieser Winkel führt zu einer äußerst unregelmäßigen Rotation, da die Abhängigkeiten zwischen den Rotationsachsen nicht berücksichtigt werden.
- Numerische Drift
  Bei der numerischen Berechnung von Rotationsmatrizen, z.B. im Zuge einer Integration von Differentialgleichungen, sind kleine Ungenauigkeiten unvermeidbar. Sie können zu einer Matrix $\mathbf{R}$ führen, die keine Rotation beschreibt und somit eine numerische Drift verursacht, die zu einer ständig wachsenden Abweichung vom exakten Lösungsverlauf führt.

Ein für viele Anwendungen besser geeigneter Ansatz stellt die Rotationsbeschreibung mit *Quaternionen* dar. Diese 1843 von Sir William Hamilton eingeführten mathematischen Größen bestehen aus einem skalaren Realteil und drei skalaren Imaginärteilen. Für sie soll die folgende Schreibweise verwandt werden:

$$\mathbf{q} := (s\,,\mathbf{v}) = s + v_x\,i + v_y\,j + v_z\,k\,.$$

Die Addition zweier Quaternionen ist durch

$$(s_1\,,\mathbf{v}_1) + (s_2\,,\mathbf{v}_2) := (s_1 + s_2\,,\mathbf{v}_1 + \mathbf{v}_2)$$

definiert und ihre (nichtkommutative) Multiplikation durch

$$(s_1\,,\mathbf{v}_1) \circ (s_2\,,\mathbf{v}_2) := (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2\,,\, s_1\,\mathbf{v}_2 + s_2\,\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)\,.$$

Die Transponierte, der Betrag und die inverse Quaternion sind zudem durch

$$\bar{\mathbf{q}} := (s\,,-\mathbf{v})\,, \quad |\mathbf{q}|^2 := \mathbf{q}\,\bar{\mathbf{q}} = s^2 + |\mathbf{v}|^2\,, \quad \mathbf{q}^{-1} := \frac{\bar{\mathbf{q}}}{|\mathbf{q}|^2}$$

festgelegt. Für eine Einheitsquaternion ($|\mathbf{q}| = 1$) gilt daher $\mathbf{q}^{-1} = \bar{\mathbf{q}}$.

Eine Rotation um den Winkel θ, die bezüglich einer beliebig gelegene Achse entlang des Einheitsvektors $\mathbf{u}$ durchgeführt wird, kann durch eine Einheitsquaternion

$$\mathbf{q} = (\cos\frac{\theta}{2}\,,\sin\frac{\theta}{2}\,\mathbf{u})$$

beschrieben werden. Die Rotationstransformation eines Punktes $\mathbf{p}$ in einen Punkt $\mathbf{p}'$ läßt sich dann durch

$$\mathbf{p}' = \mathbf{q}\,\mathbf{p}\,\bar{\mathbf{q}}$$

ausdrücken. Man kann außerdem zeigen, daß sich die zeitliche Ableitung von $\mathbf{q}$ über die Beziehung

$$\dot{\mathbf{q}} = \frac{1}{2}\,\omega\,\mathbf{q}$$

ergibt.

Für die beiden letzten Gleichungen ist dabei eine vereinfachende Schreibweise angewandt worden. Wenn ein dreidimensionaler Vektor $\mathbf{a}$ mit Quaternionen gleichgesetzt oder mit diesen multipliziert wird, steht $\mathbf{a}$ für die Quaternion $(0\,,\mathbf{a})$. Diese Schreibweise wird sehr oft angewandt und führt in der Regel zu keinen Mehrdeutigkeiten.

Eine Quaternion $(s\,,\mathbf{v})$ läßt sich auch direkt in eine Rotationsmatrix umrechnen:

$$\mathbf{R} = \begin{pmatrix} 1-2v_y^2-2v_z^2 & 2v_xv_y-2sv_z & 2v_xv_z+2sv_y \\ 2v_xv_y+2sv_z & 1-2v_x^2-2v_z^2 & 2v_yv_z-2sv_x \\ 2v_xv_z-2sv_y & 2v_yv_z+2sv_x & 1-2v_x^2-2v_y^2 \end{pmatrix}.$$

Diese Umrechnung kann eine nützliche Möglichkeit darstellen, um Quaternionen als interne Rotationsbeschreibung zu verwenden, z.B. im Rahmen einer physikalischen Simulation, ohne auf die übliche Anwendung von Rotationen in der Form $\mathbf{p}' = \mathbf{R}\,\mathbf{p}$ verzichten zu müssen.

Es stellt sich nun die Frage nach den Unterschieden, die sich aus dieser Art der Rotationsbeschreibung ergeben. Zunächst bestehen Quaternionen aus 4 Komponenten im Gegensatz zu den drei Parametern der Euler-Winkel. Bei einer Integration der zeitlichen Ableitungen dieser Komponenten, wie sie z.B. im Rahmen einer physikalischen Simulation notwendig ist, müssen daher 4 statt 3 Gleichungen gelöst werden.

Diesem Nachteil stehen aber gewichtige Vorteile gegenüber. Zum einen führt die Normierung $|\mathbf{q}| = 1$, die z.B. im Anschluß eines Integrationsschrittes durchgeführt werden kann, zu einer Einheitsquaternion, die eine reine Rotation beschreibt. Die oben angesprochene Drift kann daher nicht entstehen. Sehr wichtig ist auch der Umstand, daß Quaternionen nicht zu Koordinatensingularitäten führen können, was ihre Verwendung wesentlich vereinfacht. Auch für Interpolationsaufgaben ist dieser Ansatz sehr gut geeignet. Mit Quaternionen lassen sich Interpolationen leicht durchführen und führen zu sehr gleichmäßigen Rotationsbewegungen. Gerade für Anwendungen aus dem Bereich der Computeranimation stellen Quaternionen daher mittlerweile die standardmäßige Beschreibung für die Festlegung von Rotationen dar.

Anhang F

Numerische Verfahren

F.1 Lösung gewöhnlicher Differentialgleichungen

Gewöhnliche Differentialgleichungen sind Gleichungen oder Gleichungssysteme, die eine gesuchte Funktion $x(t)$ mit den Variablen t und den Ableitungen der Funktion $x(t)$ miteinander in Beziehung setzen. Die *Ordnung* einer Differentialgleichung gibt dabei den größten auftretenden Ableitungsgrad an. Derartige Gleichungen lassen sich stets auf ein System erster Ordnung zurückführen, das durch eine Auflösung nach dem Differentialterm[1] die folgende allgemeine Form hat:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}, t) \; . \tag{F.1}$$

Dabei ist $\mathbf{f}(\mathbf{x}, t)$ gegeben und $\mathbf{x}(t)$ ist auf einem Intervall $[t_0, t_1]$ gesucht. Liegt eine Gleichung höherer Ordnung vor, kann diese Form durch Einführung zusätzlicher Variablen erreicht werden. Die Newton'sche Bewegungsgleichung $\mathbf{F} = m\,\frac{d^2}{dt^2}\mathbf{r}$ läßt sich z.B. in die Form

$$\frac{d}{dt}\mathbf{v} = m^{-1}\,\mathbf{F} \; , \quad \frac{d}{dt}\mathbf{r} = \mathbf{v}$$

bringen, indem man $\mathbf{x}$ und $\mathbf{v}$ als voneinander unabhängige Größen behandelt.

Zur eindeutigen Lösung eines n-dimensionalen Systems von Differentialgleichungen erster Ordnung müssen n Lösungspunkte bekannt sein. Wenn diese durch die Variablenwerte zum Parameterwert t_0 gegeben sind, spricht man von einem *Anfangswertproblem*. Diese Formulierung ist der Standardfall bei der physikalisch basierten Animation und läßt sich auch am einfachsten numerisch lösen. Sogenannte *Randwertprobleme*, bei denen Variablenwerte an den Intervallrändern t_0 und t_1 gegeben sind, machen dagegen den Einsatz von Optimierungsverfahren erforderlich. Diese beiden Ausgangssituationen sind in Abbildung F.1 für das Beispiel eines schiefen Wurfes dargestellt.

Für die automatische Lösung von Differentialgleichungen lassen sich zwei verschiedene Verfahrensweisen unterscheiden: numerische und symbolische. Symbolische oder computeralgebraische Methoden bestehen in der computerunterstützten Anwendung mathematischer Umformungen, durch die die Gleichungen so lange vereinfacht werden, bis sie einer exakten Integration zugänglich sind.

Numerische Verfahren gehen auf eine völlig andere Weise vor. Sie beruhen auf einer schrittweisen oder auch iterativen Annäherung an die exakte Lösung. Bei einem Anfangswertproblem wird z.B. eine für einen Variablenwert t_0 bekannte Lösung durch schrittweise Auswertung der Differentialgleichungen bis zu einem Endwert t_E entwickelt. Das Ziel solcher Verfahren ist die Bestimmung einer

[1] Wenn diese Auflösung nicht möglich ist, spricht man von *impliziten* im Gegensatz zu *expliziten* Differentialgleichungen.

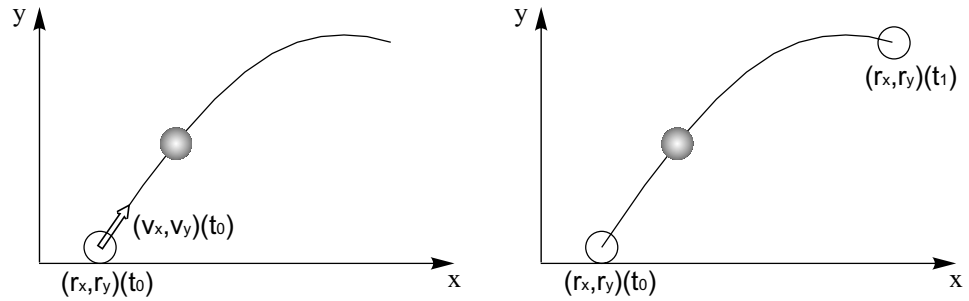

Abbildung F.1: Ein Anfangs- (links) und ein Randwertproblem (rechts).

Lösungskurve auf einem bestimmten Intervall, die der exakten Lösung möglichst genau entspricht. Dieser Ansatz soll nun genauer erläutert werden.

Numerische Lösungsverfahren

Die Anwendung von numerischen Integrationsverfahren stellt eine sehr verbreitete Möglichkeit zur automatischen Lösung eines Gleichungssystems in Form von Gleichung (F.1) dar. Die Grundidee dieser Verfahren, die z.B. in [P+92c] und [ESF98] genauer erläutert werden, liegt in der Überführung dieser Gleichung in eine algebraische Form, wozu der Zeitparameter t diskretisiert wird. Ersetzt man $\frac{d\mathbf{x}}{dt}$ durch $\frac{\Delta\mathbf{x}}{\Delta t}$, erhält man $\Delta\mathbf{x} = \mathbf{f} \cdot \Delta t$ und mit $h := \Delta t$, $t_i := t_0 + i \cdot h$ und $\mathbf{x}_i := \mathbf{x}(t_i)$ somit

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}(\mathbf{x}_i, t_i) \cdot h \,.$$

Dies ist die Näherungsgleichung des *Euler-Verfahrens*, mit dem ausgehend von den Anfangswerten $\mathbf{x}(t_0)$ die Lösung bis zum gewählten Endzeitpunkt entwickelt werden kann. Der dabei auftretende Fehler hängt von der gewählten Schrittweite h ab – wie man zeigen kann, ist er proportional zu h^2. Nur für den unerreichbaren Grenzfall $h \to 0$ würde sich eine exakte Lösung ergeben.

Ein besseres Konvergenzverhalten haben die sogenannten *Runge-Kutta-Verfahren*, die Terme mit höherer Ordnung von h berücksichtigen. Das Runge-Kutta-Verfahren zweiter Ordnung ist z.B. durch die Gleichung

$$\begin{aligned} \mathbf{k}_1 &:= \mathbf{f}(\mathbf{x}_i, t_i)\, h \\ \mathbf{k}_2 &:= \mathbf{f}(\mathbf{x}_i + \frac{1}{2}\mathbf{k}_1, t_i + \frac{1}{2}h)\, h \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \mathbf{k}_2 \end{aligned}$$

festgelegt und hat ein Konvergenzverhalten von $O(h^3)$. Sehr häufig wird auch das Runge-Kutta-Verfahren vierter Ordnung eingesetzt, das durch einen $O(h^5)$-Fehler und

$$\begin{aligned} \mathbf{k}_1 &:= \mathbf{f}(\mathbf{x}_i, t_i)\, h \\ \mathbf{k}_2 &:= \mathbf{f}(\mathbf{x}_i + \frac{1}{2}\mathbf{k}_1, t_i + \frac{1}{2}h)\, h \\ \mathbf{k}_3 &:= \mathbf{f}(\mathbf{x}_i + \frac{1}{2}\mathbf{k}_2, t_i + \frac{1}{2}h)\, h \\ \mathbf{k}_4 &:= \mathbf{f}(\mathbf{x}_i + \mathbf{k}_3, t_i + h)\, h \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \frac{1}{6}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4 \end{aligned}$$

gekennzeichnet ist.

Eine erhebliche Verbesserung bezüglich Effizenz und Robustheit läßt sich oftmals durch sogenannte *Mehrschrittverfahren* erzielen. Sie legen die Größe der Schrittweite h an jedem Integrationsschritt

neu fest. Dadurch können numerisch unproblematische Bereiche mit einer großen *h*-Wert durchschritten werden, während sich beim Auftreten von Instabilitäten eine kleinere Schrittweite wählen läßt. Sogenannte *Burlisch-Stoer*- und Runge-Kutta- Verfahren mit variabler Schrittweite stellen Beispiele für eine solche Herangehensweise dar. Sie werden zusammen mit einigen anderen Beispielen solcher Verfahren in [P+92c] erläutert.

Spezielle Techniken erfordern außerdem sogenannte *steife* Differentialgleichungen. Sie sind durch Lösungsterme gekennzeichnet, die ein extrem unterschiedliches Zeitverhalten aufweisen. Ein solcher Fall entsteht z.B. häufig bei der Anwendung steifer Federn (d.h. Federn mit einer großen Federkonstanten) auf mechanische Systeme, da sich die zeitlichen Veränderungen des mechanischen Systems i.a. auf einer sehr viel größeren Zeitskala als die Schwingungszustände der Federn abspielen. Um die Stabilität der numerischen Lösungsverfahren aufrechterhalten zu können, müssen daher extrem kleine Schrittweiten angewandt werden, was den Lösungsprozess sehr ineffizient macht. In [P+92c] und [ESF98] werden einige sogenannter *impliziter* Verfahren vorgestellt, die auf steife Differentialgleichungen zugeschnitten sind und eine robuste numerische Lösung ermöglichen. Sie sind aber deutlich ineffizienter als die standardmäßigen Verfahren und auch durch einen geringeren Grad an Genauigkeit ausgezeichnet, so daß sie nur für diesen speziellen Differentialgleichungstyp eingesetzt werden sollten. Ein solcher adaptiver Einsatz wird allerdings durch den Umstand erschwert, daß die *Detektion* eines steifen Systems sehr schwierig ist.

Vergleich mit symbolischen Verfahren

Im Vergleich zum numerischen Ansatz haben symbolische Verfahren zwei wichtige Vorteile: sie liefern exakte Lösungen und sie lassen sich i.a. wesentlich zeiteffizienter durchführen. Dieser Ansatz hat allerdings auch mit einigen gewichtigen Problemen zu kämpfen:

- Die Verfahren können einen enormen Speicheraufwand erfordern, da die Gleichungen während des Umformungsprozesses extrem „aufgebläht“ werden.
- Die Implementation computeralgebraischer Verfahren ist nicht trivial und sehr anwendungsspezifisch.
- Bei einer Änderung des Systems, z.B. in Form einer veränderten Körperanzahl, müssen die Differentialgleichungen neu formuliert und gelöst werden. Dieser Fall tritt bei der physikalisch basierten Animation sehr häufig auf.
- Symbolische Verfahren sind nicht für beliebige Systeme anwendbar, d.h. nicht für alle Differentialgleichungen kann mit ihrer Hilfe eine exakte Lösung gefunden werden.

Vor allem der letzte Punkt stellt im Rahmen der physikalisch basierten Animation ein entscheidendes Kriterium dar. Im Gegensatz zum symbolischen Ansatz können numerische Verfahren für jeden Differentialgleichungstyp eine angenäherte Lösung liefern und lassen sich zudem in einfacher Weise umsetzen. Als Technik zur Lösung ganz spezieller Gleichungssysteme hat der computeralgebraische Ansatz, der sich auch mit numerischen Verfahren kombinieren läßt, aber eine große Bedeutung.

F.2 SVD-Verfahren zur Lösung linearer Gleichungssysteme

Ein lineares Gleichungssystem hat die Form

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}\ ,$$

wobei der Vektor **x** bei gegebenen Werten für die Matrix **A** und den Vektor **b** gesucht ist. In diesem Abschnitt soll **A** dabei einer quadratischen, $(n \times n)$-dimensionalen Matrix und **x** und **b** n-dimensionalen Vektoren entsprechen. Bei der Lösung eines solchen Systems können drei Fälle auftreten:

- Es gibt eine eindeutige Lösung für **x**.
- Es gibt keine Lösung für **x** (Überbestimmtheit).
- Es gibt unendlich viele Lösungen für **x** (Unterbestimmtheit).

Eine eindeutige Lösung ergibt sich bei einer nichtsingulären Matrix **A**. In diesem Fall läßt sich **A** invertieren und **x** ergibt sich aus $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. Die beiden anderen Fälle resultieren aus einer singulären Matrix und sind wesentlich schwieriger zu behandeln.

Für die Lösung linearer Gleichungssysteme sind eine Vielzahl von Verfahren entwickelt worden. Einige Standardverfahren sind z.B. in [P+92c] beschrieben. Sie unterscheiden sich in ihrer Zeiteffizienz und Speicherauslastung, vor allem aber auch in der Behandlung singulärer Systeme. Ein wichtiger Faktor ist dabei die Robustheit des Verfahrens bei singulären oder fastsingulären Konfigurationen. Gerade im Rahmen der physikalisch basierten Animation dürfen sie nicht zu unsinnigen Ergebnissen oder zu einem Abbruch der Simulation führen. Ein Verfahren, das unter diesen Aspekten besonders interessante Eigenschaften hat, soll nun vorgestellt werden: das SVD- (*Singular Value Decomposition*-) Verfahren.

Dieses Verfahren beruht auf der folgenden Zerlegung (siehe z.B. [Mac90], [P+92c]): Jede $(n \times n)$-dimensionale quadratische Matrix **A** kann als Produkt

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \omega_1 & & 0 \\ & \ddots & \\ 0 & & \omega_n \end{pmatrix} \mathbf{V}^T, \quad \omega_i \geq 0$$

mit orthogonalen $(n \times n)$-Matrizen **U** und **V** geschrieben werden. Diese Zerlegung bildet den ersten Schritt des SVD-Verfahrens, d.h. für die gegebene Matrix **A** des Gleichungssystems $\mathbf{Ax} = \mathbf{b}$ werden zunächst die Matrizen **U** und **V** sowie die ω_i-Skalare ermittelt.

Da orthogonale Matrizen durch die Beziehung $\mathbf{U}\mathbf{U}^T = \mathbf{E}$ gekennzeichnet sind, läßt sich die zu **A** inverse Matrix formal durch

$$\mathbf{A}^{-1} = \mathbf{V} \begin{pmatrix} \frac{1}{\omega_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\omega_n} \end{pmatrix} \mathbf{U}^T$$

angeben. Dieser Ausdruck ist wohldefiniert, falls alle ω_i-Wert größer als 0 sind. In diesem Fall ist $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ die gesuchte eindeutige Lösung. Anderenfalls ist die Matrix **A** *singulär*. Ferner ist **A** *schlecht konditioniert* (*ill-conditioned*), wenn die ω_i-Werte sehr unterschiedlich sind, d.h. wenn das Verhältnis von größtem und kleinsten ω_i-Wert sehr groß ist. Wenn das Reziproke dieses Verhältnisses in der Größenordnung der Gleitkommazahlen-Ungenauigkeit des Computers liegt, kann das zugehörige Gleichungssystem nicht mehr gelöst werden. Das Lösungsverfahren würde in diesem Fall abbrechen oder eine völlig falsche Lösung liefern. Ein Vorteil des SVD-Verfahrens gegenüber anderen Verfahren besteht demnach darin, diese beiden Fälle der Singularität und der schlechten Konditioniertheit anzeigen zu können. Es ermöglicht aber auch eine sinnvolle Behandlung dieser Konfigurationen, wie nun genauer erläutert werden soll.

Falls **A** singulär ist, können zwei Fälle auftreten:

- unendlich viele Lösungen
  In diesem Fall stellt jeder Spaltenvektor $\mathbf{V}_i$ von $\mathbf{V}$, dessen zugehöriger ω_i-Wert gleich 0 ist, eine Lösung dar. Zusätzlich ist die Bestimmung einer speziellen Lösung für $\mathbf{x}$ sinnvoll, die möglichst „klein“ sein sollte, d.h. einen kleinen $|\mathbf{x}|^2$-Wert besitzt. Eine solche Lösung läßt sich bei dem SVD-Verfahren in der Form des Ausdrucks

$$\mathbf{x} = \mathbf{V} \begin{pmatrix} \frac{1}{\omega_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\omega_n} \end{pmatrix} (\mathbf{U}^T \mathbf{b}) \; .$$

  gewinnen, wenn zuvor alle $1/\omega_i$-Terme, für die $\omega_i = 0$ gilt, durch 0 ersetzt werden.

- keine Lösung
  Selbst für überbestimmte Systeme kann das SVD-Verfahren eine sinnvolle Lösung liefern. Bestimmt man $\mathbf{x}$ mit Hilfe des gleichen Ausdrucks wie beim oben behandelten Fall der Unterbestimmtheit, erhält man einen Vektor, der $|\mathbf{Ax} - \mathbf{b}|$ minimiert und somit den bestmöglichen „Lösungsvektor“ darstellt.

Bei schlecht konditionierten Matrizen besteht das beste Vorgehen in der Regel darin, die extrem kleinen ω_i-Werte gleich 0 zu setzen und in der oben beschriebenen Weise fortzufahren ([P+92c]). Auf diese Weise werden genau die Lösungskomponenten unterdrückt, die aufgrund von Rundungsfehlern zu falschen Resultaten führen würden. Mit dem SVD-Verfahren lassen sich daher sowohl unbestimmte als auch numerisch ungünstige Gleichungssysteme in robuster Weise lösen.

Anhang G

Ableitungsterme einiger Constraints

In Abschnitt 5.2 wurden einige grundlegende Constraints vorgestellt. Für ihre Einbindung mit Hilfe der LFM müssen neben den eigentlichen Constraint-Funktionen auch bestimmte partielle Ableitungen dieser Funktionen bekannt sein, wie in Abschnitt 5.1 erläutert wurde. Für holonome Constraints in der Form $\mathbf{C}(\mathbf{P}_1,\ldots,\mathbf{P}_k,t)=0$ sind dies die Terme $\dot{\mathbf{C}}$, $\partial\mathbf{C}/\partial\mathbf{P}_i$, $\partial\dot{\mathbf{C}}/\partial\mathbf{P}_i$ und $\partial\dot{\mathbf{C}}/\partial t$. Für Geschwindigkeits-Constraints, die auch von den Konnektor-Geschwindigkeiten $\dot{\mathbf{P}}_i$ abhängen können, müssen die Terme $\partial\mathbf{C}/\partial\mathbf{P}_i$, $\partial\mathbf{C}/\partial\dot{\mathbf{P}}_i$ und $\partial\mathbf{C}/\partial t$ bekannt sein. Ihre explizite Form für die jeweiligen Constraints soll nun nachgereicht werden. Die Constraints *UnitDistance*, *PointToPath*, *Hinge*, *Slide* und *Universal* werden dabei nicht aufgeführt, da sie wie in Abschnitt 6.2.3 erläutert auf die anderen Constraint-Typen und damit auf die hier angegebenen Terme zurückgeführt werden können.

Für die Ableitungen des *Orientation*- und des *InLine*-Constraints wird außerdem der durch die Beziehung

$$\mathbf{a}^\star := \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \tag{G.1}$$

definierte *Sternoperator* verwendet. Er erlaubt die Umformulierung des Kreuzproduktes als $\mathbf{a}\times\mathbf{b} = \mathbf{a}^\star\,\mathbf{b}$, was sich in diesem Rahmen als nützlichere Schreibweise erweist.

PointToPoint

$$\mathbf{C} := \mathbf{P}_1 - \mathbf{P}_2\,, \quad \dot{\mathbf{C}} = \dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2\,, \quad \frac{\partial\dot{\mathbf{C}}}{\partial t} = 0$$

$$\frac{\partial\mathbf{C}}{\partial\mathbf{P}_1} = -\frac{\partial\mathbf{C}}{\partial\mathbf{P}_2} = \mathbf{E}\,, \quad \frac{\partial\dot{\mathbf{C}}}{\partial\mathbf{P}_1} = \frac{\partial\dot{\mathbf{C}}}{\partial\mathbf{P}_2} = 0$$

Distance

$$C := |\mathbf{P}_1 - \mathbf{P}_2| - d\,, \quad \dot{C} = \frac{\mathbf{P}_1 - \mathbf{P}_2}{|\mathbf{P}_1 - \mathbf{P}_2|}\,(\dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2)\,, \quad \frac{\partial\dot{C}}{\partial t} = 0$$

$$\begin{aligned} \frac{\partial C}{\partial\mathbf{P}_1} = -\frac{\partial C}{\partial\mathbf{P}_2} &= \frac{(\mathbf{P}_1 - \mathbf{P}_2)^T}{|\mathbf{P}_1 - \mathbf{P}_2|} \\ \frac{\partial\dot{C}}{\partial\mathbf{P}_1} = -\frac{\partial\dot{C}}{\partial\mathbf{P}_2} &= \frac{(\dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2)^T}{|\mathbf{P}_1 - \mathbf{P}_2|} - \frac{((\mathbf{P}_1 - \mathbf{P}_2)\,(\dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2))}{|\mathbf{P}_1 - \mathbf{P}_2|^3}\,((\mathbf{P}_1 - \mathbf{P}_2)^T) \end{aligned}$$

Orientation

Mit $\mathbf{d}_1 := \mathbf{P}_1 - \mathbf{P}_2$ und $\mathbf{d}_2 := \mathbf{P}_3 - \mathbf{P}_4$ gilt

$$\mathbf{C} := \mathbf{d}_1 \times \mathbf{d}_2\,, \quad \dot{\mathbf{C}} = \dot{\mathbf{d}}_1 \times \mathbf{d}_2 + \mathbf{d}_1 \times \dot{\mathbf{d}}_2\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial t} = 0$$

$$\frac{\partial \mathbf{C}}{\partial \mathbf{P}_1} = -\frac{\partial \mathbf{C}}{\partial \mathbf{P}_2} = -\mathbf{d}_2^\star\,, \quad \frac{\partial \mathbf{C}}{\partial \mathbf{P}_3} = -\frac{\partial \mathbf{C}}{\partial \mathbf{P}_4} = \mathbf{d}_1^\star$$

$$\frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_1} = -\frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_2} = -\dot{\mathbf{d}}_2^\star\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_3} = -\frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_4} = \dot{\mathbf{d}}_1^\star$$

Diese Ableitungen ergeben sich aus $\mathbf{C} = \mathbf{d}_1 \times \mathbf{d}_2 = \mathbf{d}_1^\star\,\mathbf{d}_2 = -\mathbf{d}_2^\star\,\mathbf{d}_1$ mit dem oben definierten Sternoperator.

InLine

Mit $\mathbf{d}_1 := \mathbf{P}_2 - \mathbf{P}_3$ und $\mathbf{d}_2 := \mathbf{P}_2 - \mathbf{P}_1$ gilt

$$\mathbf{C} := \mathbf{d}_1 \times \mathbf{d}_2\,, \quad \dot{\mathbf{C}} = \dot{\mathbf{d}}_1 \times \mathbf{d}_2 + \mathbf{d}_1 \times \dot{\mathbf{d}}_2\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial t} = 0$$

$$\frac{\partial \mathbf{C}}{\partial \mathbf{P}_1} = -\mathbf{d}_1^\star\,, \quad \frac{\partial \mathbf{C}}{\partial \mathbf{P}_2} = \mathbf{P}_1^\star - \mathbf{P}_3^\star\,, \quad \frac{\partial \mathbf{C}}{\partial \mathbf{P}_3} = \mathbf{d}_2^\star$$

$$\frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_1} = -\dot{\mathbf{d}}_1^\star\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_2} = \dot{\mathbf{P}}_1^\star - \dot{\mathbf{P}}_3^\star\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}_3} = \dot{\mathbf{d}}_2^\star$$

Ortho

Mit $\mathbf{d}_1 := \mathbf{P}_1 - \mathbf{P}_2$ und $\mathbf{d}_2 := \mathbf{P}_3 - \mathbf{P}_4$ gilt

$$C := \mathbf{d}_1 \cdot \mathbf{d}_2\,, \quad \dot{C} = \dot{\mathbf{d}}_1 \cdot \mathbf{d}_2 + \mathbf{d}_1 \cdot \dot{\mathbf{d}}_2\,, \quad \frac{\partial \dot{C}}{\partial t} = 0$$

$$\frac{\partial C}{\partial \mathbf{P}_1} = -\frac{\partial C}{\partial \mathbf{P}_2} = -\mathbf{d}_2^T\,, \quad \frac{\partial C}{\partial \mathbf{P}_3} = -\frac{\partial C}{\partial \mathbf{P}_4} = \mathbf{d}_1^T$$

$$\frac{\partial \dot{C}}{\partial \mathbf{P}_1} = -\frac{\partial \dot{C}}{\partial \mathbf{P}_2} = -\dot{\mathbf{d}}_2^T\,, \quad \frac{\partial \dot{C}}{\partial \mathbf{P}_3} = -\frac{\partial \dot{C}}{\partial \mathbf{P}_4} = \dot{\mathbf{d}}_1^T$$

PointToFunction

$$\mathbf{C} := \mathbf{P} - \mathbf{f}(t)\,, \quad \dot{\mathbf{C}} = \dot{\mathbf{P}} - \dot{\mathbf{f}}(t)\,, \quad \frac{\partial \mathbf{C}}{\partial \mathbf{P}} = \mathbf{E}\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial \mathbf{P}} = 0\,, \quad \frac{\partial \dot{\mathbf{C}}}{\partial t} = -\ddot{\mathbf{f}}(t)$$

EqualVelocity

$$\mathbf{C} := \dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2\,, \quad \frac{\partial \mathbf{C}}{\partial t} = 0\,, \quad \frac{\partial \mathbf{C}}{\partial \mathbf{P}_1} = \frac{\partial \mathbf{C}}{\partial \mathbf{P}_2} = 0\,, \quad \frac{\partial \mathbf{C}}{\partial \dot{\mathbf{P}}_1} = -\frac{\partial \mathbf{C}}{\partial \dot{\mathbf{P}}_2} = \mathbf{E}$$

EqualVelocityNorm

$$C := |\dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2|\,, \quad \frac{\partial C}{\partial t} = 0\,, \quad \frac{\partial C}{\partial \mathbf{P}_1} = \frac{\partial C}{\partial \mathbf{P}_2} = 0\,, \quad \frac{\partial C}{\partial \dot{\mathbf{P}}_1} = -\frac{\partial C}{\partial \dot{\mathbf{P}}_2} = \frac{(\dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2)^T}{|\dot{\mathbf{P}}_1 - \dot{\mathbf{P}}_2|}$$

VelocityToFunction

$$\mathbf{C} := \dot{\mathbf{P}} - \mathbf{f}(t)\,, \quad \frac{\partial \mathbf{C}}{\partial t} = -\dot{\mathbf{f}}(t)\,, \quad \frac{\partial \mathbf{C}}{\partial \mathbf{P}} = 0\,, \quad \frac{\partial \mathbf{C}}{\partial \dot{\mathbf{P}}} = \mathbf{E}$$

Anhang H

Notation

In der untenstehenden Tabelle sind die wichtigsten der in dieser Arbeit verwandten mathematischen Symbole aufgeführt. Vektorielle Größen und Matrizen werden dabei stets durch fettgedruckte Buchstaben repräsentiert, um sie von skalaren Größen unterscheiden zu können. Die Tabelle enthält außerdem die Komponentenanzahl der jeweiligen Größe (d.h. ihre Dimension), die sie im Rahmen dieser Arbeit annimmt.

	Interpretation	Dim.
E	Einheitsmatrix	$n \times n$
t	Zeitparameter	1
x	Zustandsvektor	3 / n
v	Geschwindigkeit	3 / n
q	Quaternion	4
R	Rotationsmatrix	3×3
ω	Winkelgeschwindigkeit	3
p	Linearimpuls	3
L	Drehimpuls	3
F	Kraft	3 / n
N	Drehmoment	3
K	Kraftstoß	3
m	Masse	1
I	Trägheitstensor	3×3
M	Massenmatrix	$n \times n$
g	Gravitationsbeschleunigung	1
n	Richtungsvektor	3
r	Ortsvektor	3
P	Raumpunkt/Konnektor	3
C	Constraint-Funktion	m
J	Jacobi-Matrix	$m \times n$
c	Constraint-Vektor	m
λ	Lagrange-Faktoren-Vektor	m

Anhang I

Glossar

Animation: Aus einer Bildsequenz bestehendes Medium, das den Eindruck einer zeitlichen Entwicklung, z.B. in Form von Bewegungen, vermittelt.

Animationssystem: Softwareprodukt, das die Erstellung von Computeranimationen ermöglicht.

Allgemeines Animationssystem: Animationssystem, das nicht auf bestimmte Objekttypen oder zeitliche Entwicklungsarten zugeschnitten ist, und sich flexibel und einfach anwenden läßt.

Bewegungsgleichungen: Mathematische Gleichungen, die die zeitliche Entwicklung eines physikalischen Modells beschreiben.

Computeranimation: Computergenerierte Animation bzw. Teilgebiet der Informatik, das sich mit ihrer Erstellung beschäftigt.

Constraints: Allgemeine Bedingungen an die Zustandsvariablen eines Systems.

Darstellungsschritt: Im Anschluß eines Simulationsschrittes erfolgende Übertragung der berechneten Zustandswerte der dynamischen Körper auf ihre graphischen Repräsentanten.

Dynamische Objekteigenschaften: Eigenschaften eines Objektes, die Einfluß auf seine zeitliche Entwicklung haben (z.B. Masse oder Elastizität).

Generalisierte Koordinaten: Beliebige Zustandskoordinaten, die sich nicht wie kartesische Koordinaten auf den üblichen 3D-Raum beziehen müssen.

Geschwindigkeits-Constraints: Constraints, die sich als Gleichung der Zustandsvariablen, ihrer zeitlichen Ableitungen und der Zeit schreiben lassen.

Holonome Constraints: Constraints, die sich (im Gegensatz zu *nichtholonomen* Constraints) als Gleichung der Zustandsvariablen und der Zeit schreiben lassen.

Kollision (Stoß): Zusammenstoß zweier Körper mit nichtverschwindender Relativgeschwindigkeit.

Lagrange-Faktoren-Methode (LFM): Klassisches Verfahren zur Lösung Constraint-behafteter mechanischer Bewegungsgleichungen, das auf der expliziten Bestimmung der Zwangskräfte beruht.

Lagrange-Formalismus: Mathematischer Formalismus zur Aufstellung von Bewegungsgleichungen im Rahmen der Klassischen Mechanik, der von den Größen der kinetischen und potentiellen Energie ausgeht.

Physikalisch basierte Animation: Mit Hilfe einer physikalischen Simulation generierte Computeranimation bzw. Teilgebiet der Informatik, das sich mit einer solchen Erstellung beschäftigt.

Physikalisch basierte Modellierung: Konstruktion oder Auswahl eines physikalischen Modells und Festlegung der dynamischen Objekteigenschaften.

Physikalisches Modell: Idealisierte Entsprechung eines realen Systems, das bestimmte Eigenschaften von diesem nachbildet.

Physikalische Simulation: Automatische Auswertung eines physikalischen Modells, z.B. in Form der Bestimmung der zeitlichen Entwicklung.

Rheonome Constraints: Constraints, die in expliziter Weise von der Zeit abhängen.

Ruhekontakt: Berührung zweier Körper mit verschwindender Relativgeschwindigkeit.

Singular Value Decomposition- (SVD-) Verfahren: Numerisches Verfahren zur Lösung linearer Gleichungssysteme, das auf einer Zerlegung in orthogonale Matrizen beruht.

Technische Simulation: Teilgebiet der Physik und der Ingenieurwissenschaften, das sich mit der Ermittlung von Modelleigenschaften mit Hilfe physikalischer Simulationen beschäftigt.

Zwangskraft (Constraint-Kraft): Kraft, die von einem Constraint ausgeübt wird bzw. deren Wirkung zur seiner Erfüllung führt.

Literatur

[A+87] W. Armstrong et al. Near-real-time control of human figure models. *IEEE Computer Graphics and Applications*, 7(6):52–61, 1987.

[A+89a] B. Arnaldi et al. Animation control with dynamics. In *Proc. Computer Animation 1989*, pages 113–123, 1989.

[A+89b] B. Arnaldi et al. Dynamics and unification of animation control. *Visual Computer*, 5(1/2):22–31, 1989.

[AD92] B. Arnaldi and G. Dumont. Vehicle simulation versus vehicle animation. In G. Hegron and D. Thalmann, editors, *Computer Animation and Simulation 1992*, Eurographics, pages 1–13, Cambridge, 1992.

[AG85] W. Armstrong and M. Green. The dynamics of articulated rigid bodies for purposes of animation. *Visual Computer*, 1(4):231–240, 1985.

[AH93] M. Anantharaman and M. Hiller. Dynamic analysis of complex multibody systems using methods for differential-algebraic equations. In W. Schiehlen, editor, *Advanced Multibody System Dynamics*, pages 173–194. Kluwer Academic Publisher, 1993.

[Alp93] S.R. Alpert. Graceful interaction with graphical constraints. *IEEE Computer Graphics and Applications*, 13(2):82–91, 1993.

[AM96] R. Anderl and R. Mendgen. Modelling with constraints: theoretical foundation and application. *Computer-aided Design*, 28(3), 1996.

[B+87] N.I. Badler et al. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications*, 7(6):28–38, June 1987.

[B+92] G. Baciu et al. A formal approach to modeling and animation of physically based systems. In G. Hegron and D. Thalmann, editors, *Computer Animation and Simulation 1992*, Eurographics, Cambridge, 1992.

[B+96] R. Barzel et al. Plausible motion simulation for computer graphics animation. In R. Boulic and G. Hegron, editors, *Computer Animation and Simulation '96*, pages 184–197. Eurographics, Springer-Verlag, 1996.

[Bar89] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics (Proc. SIGGRAPH´89)*, 23(3):223–232, 1989.

[Bar91] D. Baraff. Coping with friction for nonpenetrating rigid body simulation. *Computer Graphics (Proc. SIGGRAPH´91)*, 25(4):31–41, 1991.

[Bar92] R. Barzel. *Physically-Based Modeling for Computer Graphics*. Academic Press, London, 1992.

[Bar93] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[Bar94] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. *Computer Graphics (Proc. SIGGRAPH´94)*, 28:23–34, July 1994.

[Bar95a] D. Baraff. Interactive simulation of solid rigid bodies. *IEEE Computer Graphics and Applications*, 15(3):63–75, 1995.

[Bar95b] D. Baraff. Rigid body simulation. In *Siggraph Tutorials 1995*, page 23. ACM, 1995.

[Bar96] D. Baraff. Linear-time dynamics using lagrange multipliers. *Computer Graphics (Proc. SIGGRAPH´94)*, 30:137–146, 1996.

[BB88] R. Barzel and A. Barr. A modeling system based on dynamic constraints. *Computer Graphics (Proc. SIGGRAPH´88)*, 22(4):179–188, 1988.

[BC89] A. Bruderlin and T. Calvert. Goal-directed, dynamic animation of human walking. *Computer Graphics (Proc. SIGGRAPH´89)*, 23(3):233–242, 1989.

[BW97] D. Baraff and A. Witkin. Partitioned dynamics. Technical Report CMU-RI-TR-97-33, Carnegie Mellon University, Pittsburgh, Pennsylvania, march 1997.

[BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. *Computer Graphics (Proc. SIGGRAPH´98)*, 32:43–54, August 1998.

[C+95a] J.D. Cohen et al. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In P. Hanrahan and J. Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 189–196. ACM SIGGRAPH, 1995.

[C+95b] R. Cozot et al. A unified model for physically based animation and simulation. In *Applied modelling, simulation and optimization*, pages 213–216, Anaheim, 1995. IASTED.

[C+99] E. Cerezo et al. Motion and behavior modelling: State of art and new trends. *Visual Computer*, 15:124–146, 1999.

[CC00] M. Choi and J.F. Cremer. Geometrically-aware interactive object manipulation. *Computer Graphics forum*, 19(1):65–76, 2000.

[Coh92] M. Cohen. Interactive spacetime control for animation. *Computer Graphics (Proc. SIGGRAPH´92)*, 26(2):293–302, 1992.

[DC95] S. Donikian and R. Cozot. General animation and simulation platform. In D. Terzopoulos and D. Thalmann, editors, *Computer Animation and Simulation '95*, pages 197–209. Eurographics, Springer-Verlag, September 1995.

[E+94] C. Elliott et al. TBAG: A high level framework for interactive, animated 3D graphics applications. *Computer Graphics (Proc. SIGGRAPH´94)*, 28:421–434, July 1994.

[ESF98] E. Eich-Soellner and C. Führer. *Numerical Methods in Multibody Dynamics*. Teubner-Verlag, Stuttgart, 1998.

[F+99] P. Fisette et al. Symbolic modelling for the simulation, control and optimization of multibody systems. In A. Kecskeméthy et al., editors, *Advances in Multibody System and Mechatronics*, pages 139–173. TU Graz, 1999.

[Fau99] F. Faure. Fast iterative refinement of articulated solid dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):268–276, 1999.

[Fea87] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[Fun00] J. Funge. Cognitive modeling for games and animation. *Communications of the ACM*, 43(7):40–48, July 2000.

[FW88] D. Forsey and J. Wilhelms. Techniques for interactive manipulation of articulated bodies using dynamics analysis. In *Proc. Graphics Interface 1988*, pages 8–15, 1988.

[G+91] M. Gascuel et al. A modeling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2(3):82–91, 1991.

[G+98] R. Grzeszczuk et al. Neuroanimator: Fast neural network emulation and control of physics-based models. *Computer Graphics (Proc. SIGGRAPH´98)*, 32:9–20, 1998.

[GB93] J. Goldsmith and A. Barr. Applying constrained optimization to computer-graphics. *SMPTE Journal - Society of Motion Picture and Television Engineers*, 102(10):910–912, 1993.

[GB94] J. Garcia de Jalon and E. Bayo. *Kinematic and dynamic simulation of multibody systems*. Springer-Verlag, New York, 1994.

[GB95] E. Gobbetti and J.-F. Balaguer. An integrated environment to visually construct 3D animations. *Computer Graphics (Proc. SIGGRAPH´95)*, 29:395–398, 1995.

[GG94] J. Gascuel and M. Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *Visual Computer*, 10(4):191–204, 1994.

[Gir91] M. Girard. Constrained optimization of articulated animal movement in computer animation. In Norman I. Badler et al., editors, *Making them Move*, pages 209–232. Morgan Kaufmann, 1991.

[Gle92] M. Gleicher. Integrating constraints and direct manipulation. *Computer Graphics (Proc. SIGGRAPH´92)*, 25(2):171–174, 1992.

[Gle93] M. Gleicher. A graphics toolkit based on differential constraints. In *Proceedings of the 6th Annual Symposium on User Interface Software and Technology*, pages 109–120, New York, 1993. ACM Press.

[Gle94a] M. Gleicher. *A Differential Approach to Graphical Manipulation*. Phd thesis, Carnegie Mellon University, 1994.

[Gle94b] M. Gleicher. Practical issues in graphical constraints. In V. Saraswat and P. van Hentenryck, editors, *Principles and Practice of Constraint Programming*, pages 407–426. MIT Press, 1994.

[Gle97] M. Gleicher. Motion editing with spacetime constraints. In Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 139–148. ACM SIGGRAPH, April 1997.

[Gol89] H. Goldstein. *Klassische Mechanik*. AULA-Verlag, Wiesbaden, 1989.

[Gra98] F. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[Gre91] M. Green. Using dynamics in computer animation: Control and solution issues. In Norman I. Badler et al., editors, *Making them Move*, pages 281–314. Morgan Kaufmann, 1991.

[GW91] M. Gleicher and A. Witkin. Differential manipulation. In *Proceedings of Graphics Interface '91*, pages 61–67, 1991.

[GW93] M. Gleicher and A. Witkin. Supporting numerical computations in interactive contexts. In *Proceedings of Graphics Interface '93*, pages 138–146, Toronto, Canada, 1993. Canadian Information Processing Society.

[H+92] J. Hodgins et al. Generating natural-looking motion for computer animation. In *Proceedings of Graphics Interface '92*, pages 265–272, May 1992.

[H+95] M. Harada et al. Interactive physically-based manipulation of discrete/continuous models. *Computer Graphics (Proc. SIGGRAPH´95)*, 29:199–208, 1995.

[H+96] G. Hégron et al. Dynamic simulation and animation. In *Interactive Computer Animation*, pages 71–99. ed. Magnenat-Thalmann, N. and Thalmann, D., 1996.

[HA92] G. Hegron and B. Arnaldi. Computer animation: Motion and deformation control. In *Eurographics Technical Report Series*. Eurographics, 1992.

[Hah88] J. Hahn. Realistic animation of rigid bodies. *Computer Graphics (Proc. SIGGRAPH´88)*, 22(4):299–308, 1988.

[HG95] A. Hummel and B. Girod. FRANCIS – Ein interaktives computergraphisches Simulationssystem zur schnellen Animation starrer und verformbarer Körper. In *Workshop Visualization – Dynamic and Complexity 1995*, Bremen, 1995.

[HG96] W. Hower and W. H. Graf. A bibliographical survey of constraint-based approaches to CAD, graphics, layout, visualization, and related topics:. *Knowledge-Based Sytems*, 9:449–464, 1996.

[HG97] A. Hummel and B. Girod. Fast dynamic simulation of flexible and rigid bodies with kinematic constraints. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST-97)*, pages 125–132, New York, 1997. ACM Press.

[IC87] P. Isaacs and M. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics (Proc. SIGGRAPH´87)*, 21(4):215–224, 1987.

[IC88] P. Isaacs and M. Cohen. Mixed methods for complex kinematic constraints in dynamic figure animation. *Visual Computer*, 4(6):296–305, 1988.

[JN95] D. Jackèl and S. Neunreither. Aspects of simulation in computer graphics. In *8th Symposium on Simulation Techniques*, pages 277–289, Berlin, 1995.

[Jun94] T. Jung. *Entwicklung einer Plattform zur interaktiven physikalisch basierten Animation gelenkig verbundener Systeme*. Dissertation, TU Berlin, 1994.

[Jun98] T. Jung. An algorithm with logarithmic time complexity for interactive simulation of hierarchically-articulated bodies using a distributed-memory architecture. *Journal of Real Time Imaging*, 4(1):81–96, 1998.

[KB82] J.U. Korein and N.I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics and Applications*, 2(9):71–74, 76–81, November 1982.

[KB96] H. Ko and N. Badler. Animationg human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(3):50–59, 1996.

[L+95] A. Lamouret et al. Combining physically-based simulation of colliding objects with trajectory control. *Journal of Visualization and Computer Animation*, 6(2):71–90, 1995.

[L+99] C. Lennerz et al. A framework for collision detection and response. In *11th European Simulation Symposium 1999*, pages 309–314. ESS, 1999.

[L+00] J. Laszlo et al. Interactive control for physically-based animation. *Computer Graphics (Proc. SIGGRAPH´2000)*, 34:201–208, 2000.

[Lei92] G. Leister. *Beschreibung und Simulation von Mehrkörpersystemen mit geschlossenen kinematischen Schleifen*. Fortschritt-Berichte VDI, Reihe 11, Nr. 167, 1992.

[LG96] A. Lamouret and M. Gascuel. Scripting interactive physically-based motions with relative paths and synchronization. *Computer Graphics Forum*, 15(1):25–34, 1996.

[LG98] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. IMA Conference on Mathematics of Surfaces 1998*, 1998.

[Mac90] A.A. Maciejewski. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications*, 10(3):63–71, 1990.

[Mai90] P. Maillot. Using quaternions for coding 3d transformations. In A. Glassner, editor, *Graphics Gems*, pages 498–515. Academic Press, 1990.

[Man00] D. Manocha. I-collide. <http://www.cs.unc.edu/~geom/I_COLLIDE.html>, 2000.

[MC95] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. In P. Hanrahan and J. Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 181–188. ACM, 1995.

[Met95] D. Metaxas. Articulated figure dynamics, control and shape capture. In *Siggraph Tutorials 1995, Course 11*. ACM, 1995.

[Mir95] B. Mirtich. Hybrid simulation: Combining constraints and impulses. In *First Workshop on Simulation and Interaction in Virtual Environments*, July 1995.

[Mir96a] B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2), 1996.

[Mir96b] B. Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems*. Phd thesis, University of California, Berkeley, 1996.

[Mir00a] B. Mirtich. Computing polyhedral mass properties . <http://www.merl.com/projects/rigidBodySim/massProps/>, 2000.

[Mir00b] B. Mirtich. Timewarp rigid body simulation. *Computer Graphics (Proc. SIGGRAPH´00)*, 34:193–200, 2000.

[MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics (Proc. SIGGRAPH´88)*, 22(4):289–298, 1988.

[MZ90] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *Computer Graphics (Proc. SIGGRAPH´90)*, 24(4):29–38, 1990.

[NC99] D.D. Nelson and E. Cohen. Interactive mechanical design variation for haptics and CAD. *Computer Graphics Forum*, 18(3), September 1999.

[Näh93] S. Näher. LEDA: a library of efficient data types and algorithms. In P. Enjalbert et al., editors, *Proceedings of the Symposium on Theoretical Aspects of Computer Science 1993*, volume 665, pages 710–723, Berlin, February 1993. Springer-Verlag.

[NM93] J. Ngo and J. Marks. Spacetime constraints revisited. *Computer Graphics (Proc. SIGGRAPH '93)*, 27:343–350, August 1993.

[P+90a] M. van de Panne et al. Reusable motion synthesis using state-space controllers. *Computer Graphics (Proc. SIGGRAPH´94)*, 24(4):225–234, 1990.

[P+90b] C.B. Phillips et al. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics (Proc. SIGGRAPH´90)*, 24(2):245–250, March 1990.

[P+92a] M. van de Panne et al. Control techniques for physically-based animation. In G. Hegron and D. Thalmann, editors, *Computer Animation and Simulation 1992*, Eurographics, pages 1–15, Cambridge, 1992.

[P+92b] J. Park et al. Realistic animation using musculotendon skeletal dynamics and suboptimal control. In G. Hegron and D. Thalmann, editors, *Computer Animation and Simulation '92*, Eurographics, 1992.

[P+92c] W.H. Press et al. *Numerical Recipes in C, 2nd. edition*. Cambridge University Press, 1992.

[P+93] M. van de Panne et al. Physically-based modeling and control of turning. *CVGIP - Graphical Models and Image Processing*, 55(6):507–521, 1993.

[P+00] J. Popovic et al. Interactive manipulation of rigid body simulations. *Computer Graphics (Proc. SIGGRAPH 2000)*, pages 209–218, 2000.

[PB88] J.C. Platt and A.H. Barr. Constraint methods for flexible models. *Computer Graphics (Proc. SIGGRAPH´88)*, 22(4):279–287, 1988.

[PB91] C.B. Phillips and N.I. Badler. Interactive behaviors for bipedal articulated figures. *Computer Graphics (Proc. SIGGRAPH´91)*, 25(4):359–362, July 1991.

[PF93] M. van de Panne and E. Fiume. Sensor-actuator networks. *Computer Graphics (Proc. SIGGRAPH´93)*, 27:335–342, 1993.

[Pla92] J. Platt. A generalization of dynamic constraints. *CVGIP - Graphical Models and Image Processing*, 54(6):516–525, 1992.

[Pop00] Z. Popović. Controlling physics in realistic character animation. *Communications of the ACM*, 43(7):50–58, July 2000.

[PW99] Z. Popović and A. Witkin. Physically based motion transformation. *Computer Graphics (Proc. SIGGRAPH´99)*, 33:11–20, 1999.

[R+91] H. Ruder et al. Kinematics and dynamics for computer animation. In *Eurographics Technical Report Series, Note 12*. Eurographics, 1991.

[RE97] A. Rosen and E. Eddelstein. Investigation of a new formulation of the lagrange method for constrained dynamic systems. *Journal of Applied Mechanics*, 64:116–122, March 1997.

[Rey87] C.W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (Proc. SIGGRAPH '87)*, 21:25–34, July 1987.

[RH91] M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics (Proc. SIGGRAPH´91)*, 25(4):349–358, 1991.

[RS88] R. Roberson and R. Schwertassek. *Dynamics of Multibody Systems*. Springer-Verlag, Berlin, 1988.

[SC92a] J. Stewart and F. Cremer. Beyond keyframing: An algorithmic approach to animation. *Proc. Graphics Interface 1992*, pages 273–281, 1992.

[SC92b] P.S. Strauss and R. Carey. An object-oriented 3D graphics toolkit. *Computer Graphics (Proc. SIGGRAPH´92)*, pages 341–349, 1992.

[Sch88] F. Scheck. *Mechanik*. Springer-Verlag, Berlin, 1988.

[SG93] H. Sun and M. Green. The use of relations for motion control in an environment with multiple moving objects. In *Proceedings of Graphics Interface '93*, pages 209–218, Toronto, Canada, 1993. Canadian Information Processing Society.

[Sha98] A. Shabana. *Dynamics of Multibody Systems, 2nd Edition*. Wiley, New York, 1998.

[Sho85] K. Shoemake. Animation rotations with quaternion curves. *Computer Graphics (Proc. SIGGRAPH´85)*, 19(3):245–254, 1985.

[Sur92a] M.C. Surles. An algorithm with linear complexity for interactive, physically-based modeling of large proteins. *Computer Graphics (Proc. SIGGRAPH´92)*, 26(2):221–230, 1992.

[Sur92b] M.C. Surles. Interactive modeling enhanced with constraints and physics with applications in molecular modeling. *Computer Graphics (Proc. SIGGRAPH´92)*, 25(2):175–182, 1992.

[SZ90] P. Schröder and D. Zeltzer. The virtual erector set: Dynamic simulation with linear recursive constraint propagation. *Computer Graphics (Proc. 1990 Symposium on Interactive 3D Graphics)*, 24(2):23–31, 1990.

[Tha90] D. Thalmann. Robotic methods for task-level and behavioral animation. In D. Thalmann, editor, *Scientific Visualization and Graphics Simulation*, pages 129–147, 1990.

[TT94] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. *Computer Graphics (Proc. SIGGRAPH´94)*, 28:43–50, July 1994.

[V+98] B.C. Vemuri et al. Fast collision detection algorithms with applications to particle flow. *Computer Graphics Forum*, 17(2):121–134, 1998.

[Val93] M. Valasek. Integration of recursive equations of motion for multibody systems with loops. In W. Schiehlen, editor, *Advanced multibody system dynamics*, pages 439–444. Kluwer, Dordrecht, 1993.

[VC91] N. Vasilonikolidakis and G.J. Clapworthy. Inverse Lagrangian dynamics for animating articulated models. *Journal of Visualization and Computer Animation*, 2(3):106–113, 1991.

[vO90] C. van Overveld. A technique for motion specification in computer animation. *The Visual Computer*, 6(2):106–116, 1990.

[vO91] C. van Overveld. An iterative approach to dynamic simulation of 3-d rigid-body motion for real-time interactive computer animation. *The Visual Computer*, 7:29, 1991.

[vOB95] K. van Overveld and B. Barenbrug. All you need is force: a constraint-based approach for rigid body dynamics in computer animation. In D. Terzopoulos and D. Thalmann, editors, *Computer Animation and Simulation '95*, pages 80–94. Eurographics, Springer-Verlag, September 1995.

[W+87] A. Witkin et al. Energy constraints on parameterized models. *Computer Graphics (Proc. SIGGRAPH´87)*, 21(4):225–232, 1987.

[W+88] J. Wilhelms et al. Dynamic animation: Interaction and control. *Visual Computer*, 4(6):283–295, 1988.

[W+90] A. Witkin et al. Interactive dynamics. *Computer Graphics (Proc. SIGGRAPH´90)*, 24(2):11–21, 1990.

[Wag95] F. Wagner. *Physikalisch basierte Animation von Gelenkkörpern*. Preprint CS-10-95, Universität Rostock, 1995.

[Wag99] F. Wagner. EMPHAS - ein modulares System zur einfachen Erstellung physikalisch basierter Animationen. In *Tagungsband 2. Graphiktag*, Rostock, 1999.

[Wag00] F. Wagner. Physikalisch basierte Animation auf der Grundlage der Lagrange-Faktoren-Methode. In *Proc. SimVis'2000*, pages 239–252, Magdeburg, 2000.

[WB85] J. Wilhelms and B. Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. In *Proc. Graphics Interface 1985*, pages 97–104, 1985.

[Wen98] S. Wenzel. *Verbesserung der Informationsgestaltung in der Simulationstechnik unter Nutzung autonomer Visualisierungswerkzeuge*. Verlag Praxiswissen, Dortmund, 1998.

[Wer94] J. Wernecke. *The Inventor Mentor*. Addison-Wesley, 1994.

[WH96] W. Wooten and J. Hodgins. Animation of human diving. *Computer Graphics Forum*, 15(1):3–13, 1996.

[Wil87] J. Wilhelms. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications*, 7(6):12–27, 1987.

[Wil88] J. Wilhelms. Dynamics for computer graphics: A tutorial. *Computing Systems*, 1(1):63–94, 1988.

[Wil90] J. Wilhelms. Behavioral animation using an interactive network. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation 1990*, pages 95–106. Springer-Verlag, Tokyo, 1990.

[Wil91] J. Wilhelms. Dynamic experiences. In Norman I. Badler et al., editors, *Making them Move*, pages 265–279. Morgan Kaufmann, 1991.

[Wit77] J. Wittenburg. *Dynamics of Systems of Rigid Bodies*. Teubner, Stuttgart, 1977.

[Wit94] A. Witkin. Connectors. <http://www.cs.cmu.edu/~aw/courses.html>, 1994. Course 15-860 on Physically Based Modeling.

[WK88] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics (Proc. SIGGRAPH´88)*, 22(4):159–168, 1988.

[WW90] A. Witkin and W. Welsh. Fast animation and control of nonrigid structures. *Computer Graphics (Proc. SIGGRAPH´90)*, 24(4):243–252, 1990.

[Z+91] R.C. Zeleznik et al. An object-oriented framework for the integration of interactive animation techniques. *Computer Graphics (Proc. SIGGRAPH´91)*, 25(4):105–111, 1991.

[Zel91] D. Zeltzer. Task-level graphical simulation: Abstraction, representation, and control. In Norman I. Badler et al., editors, *Making them Move*, pages 3–33. Morgan Kaufmann, 1991.

Thesen

1. Die Erstellung von realistisch wirkenden Computeranimationen ist ein wichtiges Teilgebiet der Computergraphik.

2. Die physikalisch basierte Animation stützt sich auf eine physikalische Simulation, mit deren Hilfe der zeitliche Verlauf eines zuvor modellierten Systems automatisch generiert und anschließend graphisch dargestellt wird. Gegenüber traditionellen Techniken hat dieser Ansatz den Vorteil, schon mit einem sehr geringen Aufwand äußerst realistisch wirkende Animationen erstellen zu können.

3. Ein schwieriges Problem besteht dabei in der flexiblen Steuerung der generierten Bewegungen. Zu diesem Zweck muß insbesondere die Behandlung von Constraints, d.h. allgemeiner Nebenbedingungen an den Zustand des Systems, gelöst werden. Ein bewährtes Verfahren stellt hierfür die Lagrange-Faktoren-Methode (LFM) dar. Die Anwendung des Ansatzes verlangt zudem die Einbindung in ein allgemeines Animationssystem, das eine effiziente und einfach durchführbare Animationserstellung ermöglicht, ohne auf spezielle Aspekte der Anwendung zugeschnitten zu sein.

4. Bei einer solchen Einbindung ergeben sich komplizierte Verknüpfungen zwischen den Problemen der effizienten Simulationsdurchführung, der Entwicklung flexibler Steuertechniken und der Realisierung einer intuitiven Programmbedienung. Diese Verknüpfungen wurden bislang nur unzureichend untersucht.

5. In der vorliegenden Arbeit wird eine systematische Analyse und Bewertung von Verfahren zur Simulation Constraint-behafteter mechanischer Systeme vor diesem Hintergrund durchgeführt. Die LFM erweist sich dabei als besonders geeignet, um als Grundlage eines allgemeinen, physikalisch basierten Animationssystems zu dienen.

6. Auf der Grundlage der LFM wird ein verallgemeinertes Konzept vorgestellt, das die automatische Einbindung einer sehr allgemeinen Constraint-Klasse und ihre Kapselung in separate Module ermöglicht. In diesem Rahmen wird zudem einigen grundlegenden Constraints eine mathematische Form gegeben.

7. Für die Umsetzung der LFM wird ein modulares Konzept entworfen, das auf einer Aufspaltung in verschiedene Teilaufgaben und der allgemeinen Anwendung eines in der Literatur beschriebenen, sehr zeiteffizienten Verfahrens beruht. Dabei wurden auch die Probleme bei der Beschreibung des Systemzustandes und der Formulierung der Bewegungsgleichungen berücksichtigt.

8. Für die kombinierte Anwendung von Constraints und verschiedenen anderen Steuertechniken werden Möglichkeiten aufgezeigt, die sich auf spezielle Eigenschaften der LFM stützen. Es werden Anwendungsmöglichkeiten für überbestimmte Systeme im Rahmen allgemeiner Animationssysteme und grundlegende Ansätze zu ihrer Behandlung vorgestellt. Außerdem werden Lösungskonzepte für das bislang gänzlich unbeachtete, LFM-spezifische Problem der Bewegungsartefakte, d.h. unerwünschter zusätzlicher Bewegungen bei der Festlegung von Constraints, beschrieben.

9. Auf der Grundlage dieser Konzepte wurde das allgemeine Animationssystem EMPHAS (**E**asy-to-use **M**odular **PH**ysically-based **A**nimation **S**ystem) realisiert, das ein modulares System zur einfachen, interaktiven Erstellung physikalisch basierter Animationen mechanischer Systeme auf der Grundlage der LFM darstellt.

10. Als dynamische Körper dienen in EMPHAS massenbehaftete Punktkörper und dreidimensionale starre Körper. Durch die Realisierung verschiedener Steuerelemente konnte eine flexibel durchführbare Modellierung und Bewegungssteuerung verwirklicht werden, die sich auf diese Körpertypen, aber auch auf andere mechanische Systeme anwenden läßt. Zu diesen Elementen zählen sehr allgemeine Typen von Constraints, aber auch Controller, die die Anwendung zustandsabhängiger Kräfte erlauben, und Kraftfelder, mit denen globale Kraftwirkungen festgelegt werden können. Zusätzlich wurde ein sehr allgemeines Konzept für eine prozedurale Einflußnahme auf die Szenendynamik realisiert.

11. In EMPHAS wurde eine effiziente Umsetzung der LFM verwirklicht, die sich durch einen hohen Grad an Generalität und Modularität auszeichnet. Als wesentlicher Bestandteil wurde dabei eine Erweiterung um ein spezielles numerisches Lösungsverfahren durchgeführt, das auch die Behandlung überbestimmter Systeme ermöglicht.

12. Es wurde eine realistisch wirkende Simulation von Kollisionen und eine automatische Behandlung von Ruhekontakten realisiert und in das EMPHAS-System eingebunden. Dabei konnte auch das schwierige Problem gelöst werden, die kombinierte Anwendung mit sämtlichen der in EMPHAS realisierten Steuertechniken zu ermöglichen.

13. Mit Hilfe eines sehr intuitiven „Baukasten“-Konzeptes wurde eine einfache und frei kombinierbare Anwendung der Steuerelemente verwirklicht. Dabei wurden auch Techniken zur Behandlung überbestimmter Systeme und zur Vermeidung unerwünschter Bewegungsartefakte auf der Grundlage der zuvor entwickelten Konzepte realisiert.

14. Für die Benutzungsschnittstelle von EMPHAS wurden eine Reihe von Techniken entwickelt, die eine einfach erlernbare, flexible und vollständig interaktive Programmbedienung erlauben. Durch eine Import-Funktionalität und den Export von Geometrie- und Bewegungsdaten ist zudem eine Anbindung an kommerzielle Animationssysteme geschaffen worden.