Animation komplexer Szenen

Thomas Jung

Die Computeranimation gewinnt innerhalb der Informatik mehr und mehr an Bedeutung. Neben immer größeren Anwendungsfeldern in der Unterhaltungsindustrie und der Werbung, eröffnen sich auch Einsatzmöglichkeiten in Lehre ("Educational systems") und Wissenschaft. Starke Bezüge existieren auch zwischen den Themengebieten "Scientific Visualization" und Computeranimation. Im folgenden Artikel werde ich die Techniken vorstellen, die in heutigen "State-Of-The-Art"-Animationssystemen verwendet werden und die daraus entstehenden Probleme diskutieren.

Die Ursprünge der Animation entstammen dem Trickfilm. Lange bevor das Medium Computer für die Generierung von Filmsequenzen benutzt werden konnte, wurden mit Hilfe von Zeichnungen, Marionetten, Knete, Schattenbildern, etc. Trickfilme produziert. Viele Begriffe und Techniken, die heute in der Computeranimation verwendet werden, haben ihre Wurzeln im konventionellen Trickfilm.
Im herkömmlichen Sinne bedeutet animieren, zum Leben erwecken, in der Computeranimation bezeichnet der Begriff die Veränderung der sichtbaren Dinge. Darunter fällt nicht nur die Bewegung von Objekten (Veränderung der Position), sondern auch die Veränderung von Attributen, wie Farbe, Oberflächeneigenschaften (Textur) oder Beleuchtung, oder auch Kamerafahrten.
Der Begriff der Szene hat in der deutschen Sprache eine vielschichtige Bedeutung. Einerseits ist damit der Schauplatz einer Handlung gemeint (Szenerie), des weiteren aber auch eine zeitliche Abfolge von Dingen ("jemandem eine Szene machen", "Szenenwechsel im Film", etc.). Im folgenden wird eine Szene als statische, strukturierte Ansammlung von Objekten betrachtet.
Wie sieht nun eine Szene auf dem Medium Computer aus ? Ziel einer Computeranimation ist es meist, einen Film zu produzieren. Ein Film besteht aus einer Folge von Bildern. Da Bilder nur zwei Dimensionen besitzen, wäre es denkbar, daß Szenen ebenfalls ausschließlich zweidimensional repräsentiert wären.
Betrachten wir zunächst die Szenen, die im konventionellen Trickfilm Verwendung finden: Dort gibt es sowohl zwei- als auch dreidimensionale Szenen. Während bei der Produktion von Zeichentrickfilmen in der Regel nur zweidimensionale Szenenrepräsentationen (Skizzen im sogenannten Storyboard, Umrißzeichnungen, kolorierte Bilder) verwendet werden, sich die dritte Dimension also nur im Kopf des Animateurs (Entwickler des Films) befindet, sind die im Puppentrickfilm verwendeten Marionetten von vornherein dreidimensional. Unterschiedliche Szenentypen (im Trickfilm besser Medien

genannt) verlangen auch unterschiedliche Animationstechniken. Ein Animateur, der Zeichentrickfilme entwickelt, ist nicht unbedingt ein guter Puppenspieler.

Auch in der Computeranimation werden unterschiedliche Ansätze verfolgt. Zunächst wurden nur zweidimensionale Verfahren realisiert, wobei sich die entstandenen Systeme eng an den Zeichentrickfilmtechniken orientierten. In den Computer eingegebene Umrißzeichnungen konnten halbautomatisch koloriert wurden, so daß sich der Aufwand für die Filmproduktion erheblich senken ließ. Auch das "Inbetweening" (Generierung von Zwischenbildern anhand von sogenannten Keyframes, die für ausgewählte Zeitpunkte angefertigt werden) konnte teilweise automatisiert werden.

Ein Nachteil zweidimensionaler Modelle ist mangelnde Plastizität. Zeichentrickbilder wirken selten plastisch, diese Einschränkung gilt analog auch auf dem Computer. Im Zeichentrickfilm wird dieser Nachteil in Kauf genommen, da geübte Zeichner intuitiv Charakteristika von Figuren und Bewegungen durch gekonnt gesetzte Striche (in zweidimensionalen Bildern) unterstreichen können. Dieser Vorteil existiert bei der automatischen Zwischenbildgenerierung auf dem Computer nicht. So werden in der Computeranimation heute in der Regel nur noch dreidimensionale Modelle verwendet. Techniken der 3D-Computer-Graphik (Raytracing, Radiosity, etc.) erlauben es, Bilder dreidimensionaler Modelle zu erzeugen, die ein hohes Maß an Plastizität aufweisen.

Für die Abbildung dreidimensionaler Körper auf Bilder ist ausschließlich geometrische und Oberflächeninformation (Farbe, Rauheit, etc.) nötig. Zum Beispiel kann man sich vorstellen, über die Oberfläche jedes in der Szene vorkommenden Körpers ein Polygonnetz zu legen. Eine Liste aller in der Szene vorkommenden Polygone mit ihren Farben reicht aus, um eine plastische Darstellung der Szene generieren zu können. Für die Bildgenerierung ist es nicht nötig, die einzelnen Objekte in Bezug zueinander zu setzen. Anders stellt sich die Situation dar, wenn Szenen animiert werden. Wenn sich zum Beispiel ein Objekt bewegt, welches für einen Betrachter einen Oberarm eines Menschen repräsentiert, so erwartet der Betrachter, daß sich das Objekt, das er als Unterarm erkennt, mitbewegt und sich nicht etwa vom Oberarm trennt. Damit Algorithmen diese scheinbar selbstverständlichen Voraussetzungen gewährleisten können, müssen innerhalb der Szene strukturelle Beziehungen zwischen Einzelobjekten repräsentiert sein. Leider ist sowohl die Repräsentation dieser Beziehungen als auch die Entwicklung der entsprechenden Algorithmen nicht einfach. Die Modellierung der Beziehungen hängt einerseits von den auszuführenden Animationen andererseits von den dafür benutzten Algorithmen ab.

Diese strukturellen Beziehungen beschreiben nur die dreidimensionale Ausprägung von Szenen. Für die Produktion von Filmen ist es aber unerläßlich, darüberhinaus auch die zeitliche Veränderung von Szenen zu repräsentieren. Es gibt keine allgemein verwendetete Repräsentation vierdimensionaler Information, die für die Beschreibung von Animationen verwendet werden könnte. Meist orientiert sich die Repräsentation von Computeranimationen an den implementierten Animationstechniken und -algorithmen.

Simulation, Animation und Kontrolle

Bisher wurde nur beschrieben, wie Szenen repräsentiert werden können, wie Szenen aber durch den Animator entworfen und animiert werden können, wurde noch nicht diskutiert. Auf die Modellierung von Szenen wird im Rahmen dieses Aufsatzes nicht eingegangen, da das ein eigenes Forschungsgebiet ist und den Rahmen dieses Aufsatzes sprengen würde. Es werden nur die Aspekte der Szenenmodellierung betrachtet die für die Animation von Interesse sind.

Welche Techniken zur Animation von Szenen können also einem Animateur auf dem Medium Computer zur Verfügung gestellt werden ? Die ersten Animationssysteme für gelenkig verbundene Systeme waren Keyframing-Systeme. In Anlehnung an die Produktionstechniken im Zeichentrickfilm werden für ausgewählte Zeitpunkte ("Keyframes" oder "Schlüsselbilder"), die eine möglichst hohe Aussagekraft für die Gesamtbewegung haben sollten, die Stellungen aller Gelenke bzw. die Positionen aller Objekte durch einen Animateur vorgegeben. Für alle dazwischenliegenden Zeitpunkte wurden die Gelenkstellungen interpoliert ("Inbetweening"). Beim Entwurf der Keyframes kann der Animateur unterstützt werden, indem er eine vorgegebene Objektkonstellation durch Manipulation einiger weniger Gelenke der gewünschten Bewegung anpassen kann. Es entstanden weitere Animationstechniken, die es zum Beispiel erlauben, 3D-Raumkurven zu entwerfen, diese mit einem Zeitraster (Punkten auf der Raumkurve werden Zeitpunkte innerhalb der Animationssequenz zugeordnet) zu versehen und den Computer daraus Bewegungen generieren zu lassen. Im Prinzip basieren diese Ansätze alle auf der Interpolation von vorgegebenen, dreidimensionalen Gliedpositionen.

Vorteil dieser Ansätze ist die einfache Kontrolle der Animationen. Der Animateur kontrolliert Größen (3D-Positionen), deren Einfluss auf die Bewegung direkt nachvollziehebar ist, zumal der numerische Aufwand der Interpolationsalgorithmen relativ gering ist, und die entwickelten Bewegungen in Echtzeit (während der Manipulation durch den Animateur) visualisiert werden. Die Qualität der generierten Bewegungssequenzen hängt wesentlich vom Geschick des Animateurs ab. Erfahrungsgemäß müssen Keyframes in relativ engen zeitlichen Abständen definiert werden, um überhaupt einen akzeptablen Bewegungseindruck erzeugen zu können. Der Animateur muß ein "gutes Auge" für Bewegungen haben, denn durch das System erhält er bei der Erzeugung "realistischer" Bewegungsabläufe keine Unterstützung. Zum Beispiel bleibt die Gravitation, die menschliche Bewegungen wesentlich mitbeeinflusst, in solchen Systemen unberücksichtigt.

Abhilfe kann der Einsatz von Simulationsalgorithmen schaffen. Wenn es gelingt, einerseits reale (Natur-) Gesetzmäßigkeiten auf das Medium Computer abzubilden, andererseits die Eigenschaften der Objekte bezüglich dieser Gesetze zu modellieren, können Simulationsalgorithmen sehr realistisch wirkende Animationssequenzen erzeugen. In einer so generierten "Newtonschen Welt" würden Objekte auf die Erde fallen, kollidieren, sich

deformieren, etc. Leider hat die Verwendung solcher Algorithmen auch Nachteile. Ein erster gravierender Nachteil ist der numerische Aufwand. Um relativ einfache physikalische Vorgänge (z. B. Kollision mit Deformation) zu simulieren, sind Algorithmen nötig, die auch unter Verwendung moderner Hardware für die Berechnung von ein paar Sekunden Film Stunden, Tage oder Monate an Rechenzeit benötigen. Ein weiteres Problem ergibt sich aus der Notwendigkeit, dem Animateur Eingriffsmöglichkeiten in die simulierten Bewegungsabläufe zu gewähren. Modelle, die den physikalischen Gesetzen folgen, können auch nur durch physikalische Größen (Kräfte oder Drehmomente) manipuliert werden. Ein Mensch würde, wenn er nicht kontrolliert seine Muskeln einsetzen würde, in sich zusammenfallen. Um nun zum Beispiel eine menschliche Laufbewegung zu definieren, müssen die an den Gelenken des menschlichen Modells wirkenden Drehmomente über die Zeit definiert werden. Leider ist der Zusammenhang zwischen den wirkenden Kräften (Drehmomenten) und den daraus resultierenden Bewegungen für einen Betrachter nur sehr schwer nachvollziehbar. Die Aufgabe ein menschliches Modell mit expliziten Gelenkkräften zu steuern, ist relativ aussichtslos.
Es wäre also günstig, dem Animateur Kontrolltechniken zur Verfügung stellen zu können, die für ihn leicht zu handhaben sind, und Algorithmen zu entwickeln, die anhand der Kontrollkommandos in der Lage sind, physikalisch basierte Modelle zu animieren. Am Ende dieses Aufsatzes werde ich auf dieses Problem zurückkommen. Zunächst einmal sollen die Voraussetzungen für den Entwurf leistungsfähiger Kontrolltechniken diskutiert werden.

Struktur, Freiheiten und Constraints

Bisher wurde nur erwähnt, daß das Szenenmodell strukturelle Information besitzen muß, eine nähere Beschreibung dieser Struktur wurde noch nicht gegeben. Wie muß eine zu modellierende Szene also strukturiert sein ? Um ein Animatiossystem zu bauen, müssen wir dieses Problem eher von einem technischen Standpunkt aus betrachten. Die Struktur des Szenenmodells muß so gewählt werden, daß Animationsalgorithmen Bewegungen erzeugen können, die den Ansprüchen des Animateurs genügen. Wenn man ein Modell des Menschen entwirft, ist es sicher nicht sinnvoll, die Struktur seiner Moleküle zu beschreiben, wahrscheinlich auch nicht die seiner inneren Organe. Wenn eine Laufbewegung dargestellt werden soll, kann vielleicht auch von den Bewegungsmöglichkeiten der Finger abstrahiert werden. Je (qualitativ) einfacher die Struktur gewählt wird, desto geringer ist der numerische Aufwand und desto einfachere Algorithmen können gewählt werden.

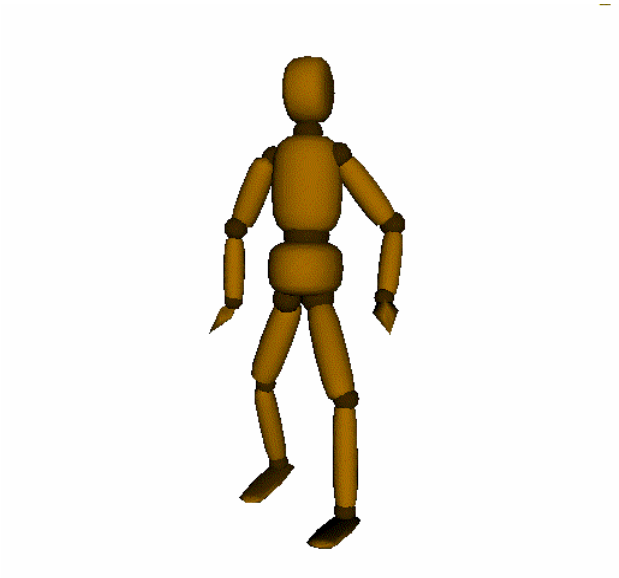

Abb. 1: Modell eines Menschen (Glieder in hellbraun, Gelenke in dunkelbraun)

Zunächst werden die Komponenten betrachtet, die durch die Struktur verbunden werden. In erster Annäherung sollten die Komponenten der Struktur als in sich völlig unbeweglich angenommen werden. Betrachten wir wieder ein Modell des Menschen. Die Körperglieder werden zu Strukturkomponenten (Oberschenkel, Unterschenkel, Fuß, Hand, etc). Nachfolgend werde ich diese Komponenten generell Glieder nennen. Bewegungen des menschlichen Modells beschränken sich auf Relativbewegungen zwischen den Gliedern. Solche gelenkig verbundenen Systeme starrer Körper werden sehr oft zur Animation sowohl von technischen (Roboter, Maschinen, etc), als auch von natürlichen Modellen (Wirbeltiere, Insekten, etc) verwendet. Mit Hilfe dieser Modelle können tatsächliche Bewegungen natürlicher Modelle allerdings nur approximiert werden. Muskuläre Bewegungen haben auch Veränderungen der Hautoberfläche zur Folge, so daß Glieder von Menschen oder Tieren real nicht starr sind. Das Problem der korrekten Darstellung der Haut kann aber (abhängig von den Gelenkstellungen der Glieder) isoliert betrachtet werden. Andere Ansätze ermöglichen es, deformierbare Modelle zu definieren, jedoch ist der numerische Aufwand der dafür nötigen Algorithmen (z. B. Fenite Elemente Methoden) wesentlich höher als der für die Animation von festen gelenkig verbundenen Systemen, so daß nachfolgend nur letztere betrachtet werden.
Als erstes ist zu untersuchen, welche Art von Ordnung zwischen den Gliedern definiert werden kann. Generell sind Tiere und Menschen hierarchisch strukturiert, es gibt keine zyklischen Gelenkverbindungen bei Wirbeltieren. Allerdings ist die Wahl der Wurzel der Hierarchie nicht festgelegt, vielfach wird bei menschlichen Modellen das Becken als Wurzel verwendet, da sich dort der Schwerpunkt befindet und diese Eigenschaft relevant für manche Algorithmen

ist. Bei technischen Systemen hingegen existieren häufig zyklische Verbindungen. Als Beispiel sei hier ein hydraulisch betriebener Kran zu nennen.

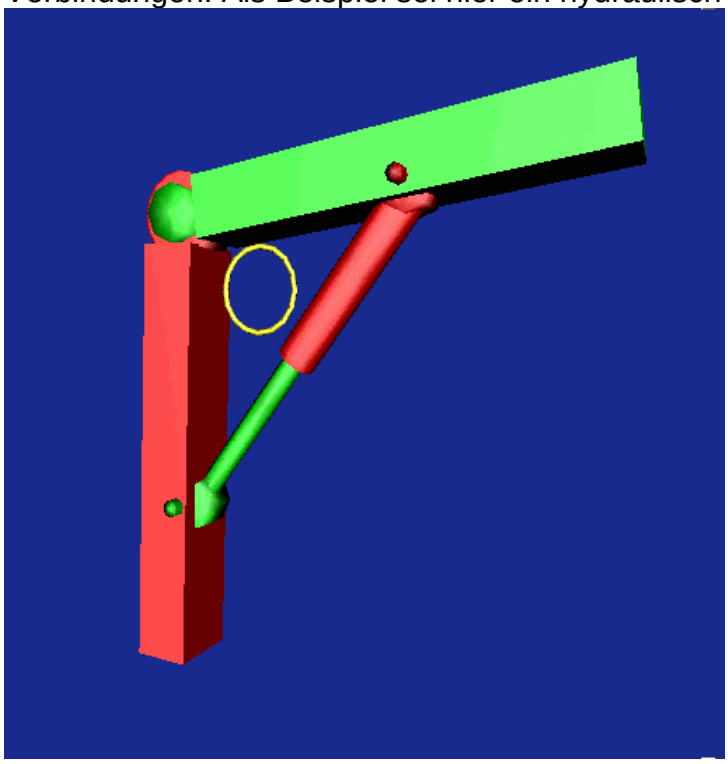

Abb. 2: zyklische Verbindungskette (der gelbe Kreis symbolisiert den Zyklus)

Neben den dauerhaften Gelenkverbindungen können während einer Animation vielfältige Wechselwirkungen zwischen weiteren Gliedern entstehen. Nimmt zum Beispiel ein Mensch einen Gegenstand in die Hand, so ist die Bewegung des Gegenstandes nicht mehr unabhängig von der der Hand. Solange der Gegenstand festgehalten wird, sind die Verbindungen zwischen Hand und Gegenstand, sowie zwischen Hand und Unterarm qualitativ gleichwertig. Die durch Interaktion mehrerer, zunächst nicht verbundener Subsysteme entstehenden Verbindungen, führen in der Regel zu Zyklen. Falls zum Beispiel ein Mensch mit beiden Füßen auf dem Boden steht, schließt der Boden einen Kreis über das Becken und die beiden Beine.

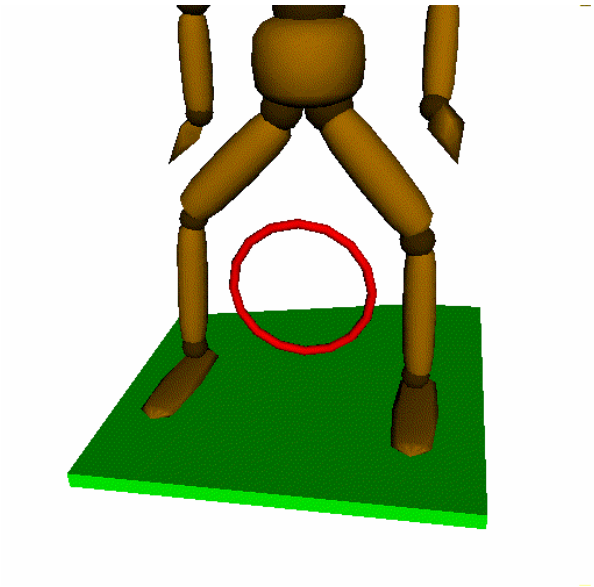

Abb. 3: Zyklus durch Bodenkontakt

Unglücklicherweise sind die Algorithmen für die Behandlung zyklischer Verbindungen komplizierter und in der Regel aufwendiger als die für die Behandlung hierarchischer Vebindungen.
Ein großes Problem stellt auch die Erkennung und Klassifizierung der während der Animation entstehenden, neuen Verbindungen dar. Setzt man zum Beispiel einen Kreuzschlitzschraubenzieher (mit Druck) an eine entsprechende Schraube an, so können beide Objekte sich nur gleichzeitig drehen, ein normaler Schraubenzieher kann hingegen seitlich abrutschen. Bei dem Einsetzen einer Kerze in einen Kerzenständer, was geometrisch kaum von den anderen beiden Beispielen zu unterscheiden ist, kann die Kerze unabhängig gedreht werden, falls die Reibung zwischen beiden Objekten nicht zu groß ist.
Wie schon aus den obigen Beispielen ersichtlich ist, können ganz unterschiedliche Bewegungsmöglichkeiten zwischen Gliedern existieren. Die prinzipielle Möglichkeit, um eine Gelenkachse zu rotieren, oder sich entlang einer Achse zu bewegen (Translation), wird Freiheit genannt. Ein völlig frei bewegliches Objekt besitzt genau sechs Freiheiten. Da wir uns in einem dreidimensionalen Raum bewegen, gibt es drei Translationsfreiheiten (für jede Dimension eine). Des weiteren gibt es drei Möglichkeiten, ein Objekt zu rotieren. Als Rotationsachsen können zum Beispiel die drei Koordinatenachsen eines kartesischen Koordinatenssystem angenommen werden, jedoch sind auch andere Achsenkombinationen möglich. Zwischen zwei gelenkig verbundenen Gliedern existieren maximal die sechs oben beschriebenen Freiheiten, in der Regel aber wesentlich weniger.

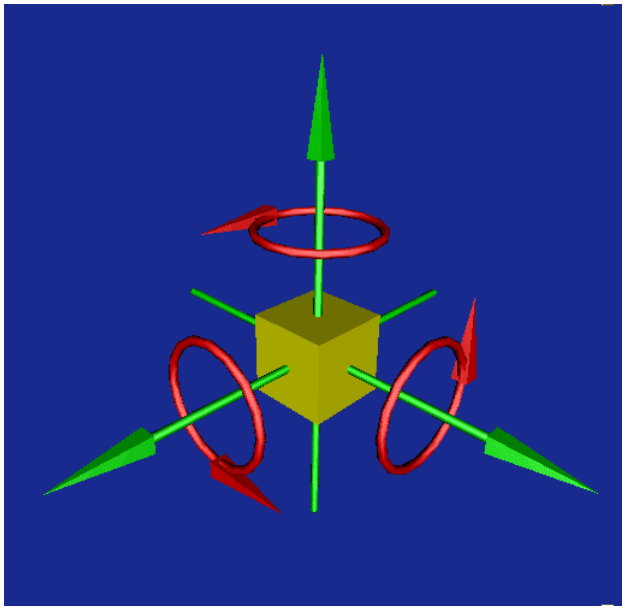

Abb. 4: Ein Würfel mit 6 Freiheiten (Translation in grün, Rotation in rot)

Eine Teleskopantenne hat zum Beispiel eine Translationsfreiheit, ein menschliches Knie eine Rotationsfreiheit. Zwei Glieder, zwischen denen überhaupt keine Freiheit existiert, können als ein Glied betrachtet werden.
Eine Freiheit bezeichnet nur die prinzipielle Bewegungsmöglichkeit. Eine genauere Beschreibungen der Verbindung zwischen zwei Gliedern erfolgt durch sogenannte Constraints ("Gezwungenheiten"). Constraint ist ein wichtiger Begriff in der Computeranimation, eine genaue Definition dieses Begriffs existiert jedoch nicht. Constraints können Relationen, Regeln, Randbedingungen oder im einfachsten Fall der Minimal- und Maximalwert eines Rotationswinkels zwischen zwei Gliedern sein.

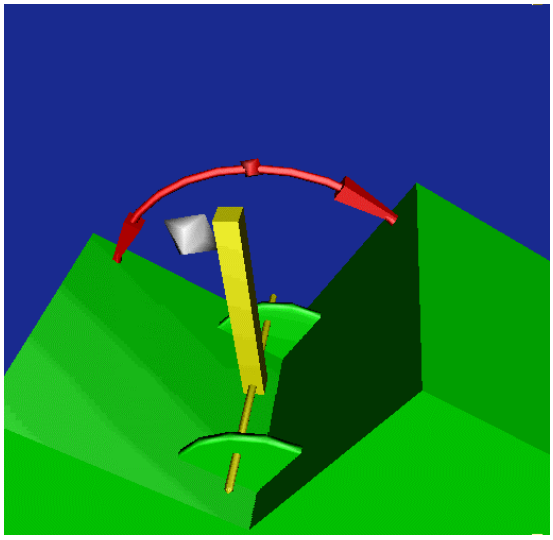

Abb. 5: Rotationsbeschränkung (der rote Pfeil symblisiert den erlaubten Rotationswinkel)

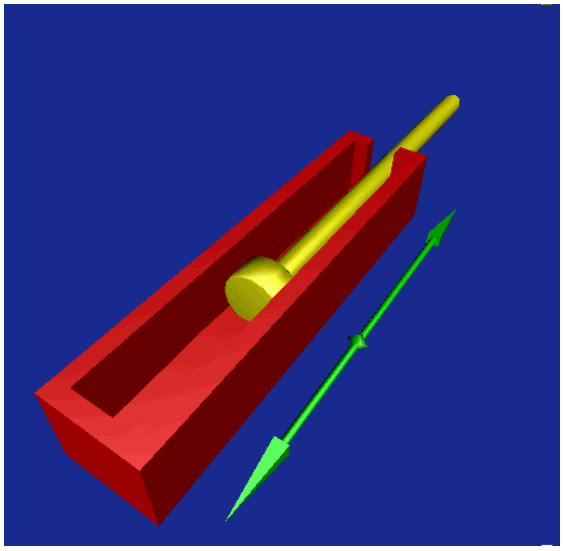

Abb.6: Translationsbeschränkung der grüne Pfeil symbolisiert den erlaubten Bewegungsbereich)

Leider lassen sich die Beziehungen zwischen Gliedern in der Regel nur approximieren. Ein Kniegelenk mag noch durch einen einzelnen begrenzten Rotationswinkel relativ genau zu beschreiben sein, bei einem Oberschenkelgelenk existieren nicht triviale Wechselwirkungen zwischen den Extremwerten der einzelnen Rotationsfreiheiten. Je weiter man das Knie hebt, desto weniger kann man es nach außen drehen. Je nach auszuführender Bewegung sollte man die Constraints mehr oder weniger genau modellieren. Die Wahl der Constraints legt die zu deren Lösung oder Einhaltung nötigen Algorithmen fest und bestimmt damit wesentlich den numerischen Aufwand für die Berechnung von Animationssequenzen. Genauso wird die Benutzerschnittstelle (und somit die Kontrollmöglichkeiten des Animateurs) wesentlich von den im System definierbaren Constraints mitbestimmt.
Zuletzt werden die Glieder selbst betrachtet. Wenn man einen rein kinematischen Ansatz (ohne physikalische Gesetzmäßigkeiten) wählt, besteht ein (starres) Glied ausschließlich aus der Geometrie- und Oberflächeninformation. Die kinematischen Algorithmen können bei der Betrachtung der Verbindungen zwischen den Gliedern völlig von der Beschaffenheit der Glieder abstrahieren. Ein Glied hat dann ein eigenes Koordinatensystem und eine Oberflächenbeschreibung (Polygone, Farbe, Textur, etc.) bezüglich dieses Koordinatensystems, die während der Bewegungen als konstant angesehen werden kann.
Wenn ein dynamisches Modell verwendet wird, benötigt man zusätzliche Informationen über die Massenverteilung des Gliedes. Körper die nur translative Bewegungen ausführen (also sich nicht drehen), können durch Massenpunkte approximiert werden. Man benötigt dann nur die Masse des gesamten Gliedes. Glieder starrer, gelenkig verbundener Systeme besitzen in der Regel aber auch Rotationsfreiheiten. Glücklicherweise lässt sich die Massenverteilung starrer Körper mit relativ wenig Daten ausreichend beschreiben. Zusätzlich zur Masse muß der Schwerpunkt und ein sogenannter Trägheitstensor, welcher durch eine symmetrische 3x3-Matrix (also sechs Zahlen) repräsentiert werden kann, bekannt sein. Diese drei Größen sind für feste Körper konstant, und können relativ leicht aus der Geometrie (näherungsweise) bestimmt werden.
Würde man von der Annahme der Starrheit der Glieder ablassen, würde eine wesentlich komplexere Repräsentation der Massenverteilung benötigt werden. Es existieren Ansätze zur Repräsentation deformierbarer Körper zum Beispiel unter Verwendung von Finiten Elemente Methoden, auf die hier aber nicht eingegangen wird.

Bewegung von Festkörpern

Im Zusammenhang mit der Animation fester Körper werden meistens vier Probleme beschrieben, die nachfolgend erläutert werden.
In kinematischen Systemen ist, um Bewegungen gelenkig verbundener Systeme darzustellen, das sogenannte "Forward-Kinematics"-Problem zu lösen. Gesucht sind die absoluten Positionen aller Glieder innerhalb der Welt des Betrachters, bekannt sind jedoch nur die Stellungen der Gelenke (in den

Koordinatensystemen der Glieder). Dieses Problem ist zum Glück relativ einfach zu lösen. Die algorithmische Umsetzung ist mit Hilfe einiger Matrixmultiplikationen (Aufmultiplikation aller beteiligten Gelenktransformationen der Glieder entlang einer Gliederkette) zu leisten.
Soll eine Bewegung dargestellt werden, bei der ein Mensch einen Gegenstand greift, ist das umgekehrte Problem ("Inverse Kinematics") zu lösen. Die Position der Hand ist schon vorgegeben ("dort, wo der Gegenstand ist"), die Stellungen der Gelenke zum Erreichen dieser Position müssen bestimmt werden. Man kann sich leicht vorstellen, daß dieses Problem nicht eindeutig zu lösen ist. Man kann den Ellenbogen ganz unterschiedlich bewegen, um mit der Hand einen Gegenstand zu erreichen. Andererseits gibt es Gegenstände, die zu weit entfernt sind, um sie zu erreichen (ohne sich zuerst in die Nähe des Gegenstandes zu bewegen). Es können also zu einem Inverse-Kinematics-Problem mehrere oder auch gar keine Lösungen existieren. Unglücklicherweise existieren keine Algorithmen, die für alle gelenkig verbundenen Systeme immer alle Lösungen finden können. Die existierenden Algorithmen sind numerisch aufwendig, finden in der Regel nur eine Lösung und nähern diese nur an.
In physikalisch basierten Systemen existieren zusätzlich zwei analoge Probleme. Um ein Forward-Dynamics-Problem zu lösen, sind die Beschleunigungen der Glieder zu bestimmen (bei gegebenen Gelenkstellungen und -geschwindigkeiten), wenn Gelenk- (oder auch externe) kräfte an das verbundene System angreifen. Ein Inverse-Dynamics-Problem löst man, indem die Gelenkkräfte bestimmt werden (ebenfalls Gelenkstellungen und -geschwindigkeiten bekannt), die zu gegebenen Beschleunigungen führen. Für die Lösung beider Probleme gibt es Algorithmen, die eine genaue Lösung bestimmen, wobei (im Gegensatz zu den Kinematics-Problemen) die Lösung eines Inverse-Dynamics-Problems einfacher als die eines Forward-Dynamics-Problems ist.
Für die Simulation von Bewegungen physikalisch basierter, gelenkig verbundener Systeme sind zwei Probleme zu lösen. Für das Forward-Dynamics-Problem sind wie schon oben beschrieben, Kräfte, Geschwindigkeiten und Gelenkstellungen nötig, um die Beschleunigungen zu bestimmen. Das zweite Problem besteht darin, die durch die Beschleunigungen verursachten Veränderungen der Geschwindigkeiten und Gelenkstellungen zu bestimmen. Um das zweite Problem zu lösen, muß eine Differentialgleichung zweiten Grades gelöst werden, die leider nicht explizit vorliegt. Das zweite Problem wird als Anfangswertproblem bezeichnet. Für einen Folgezeitpunkt werden die neuen Gelenkstellungen und Geschwindigkeiten anhand der alten Werte und der neuen Beschleunigungen bestimmt. Diese Betrachtung kann nur mit Hilfe eines zeitlichen Rasters durchgeführt werden. Die Rasterabstände müssen in der Regel sehr klein (Millisekundenbereich) gewählt werden, um den Gesamtfehler der Simulation in einem akzeptablen Rahmen zu halten. Für jeden Rasterzeitpunkt ist das Forward-Dynamics-Problem ein- oder mehrmals zu lösen, so daß insgesamt ein erheblicher numerischer Aufwand entsteht. Unglücklicherweise sind die Verfahren für die Lösung von Anfangswertproblemen numerisch oft instabil, so daß die simulierten

Bewegungen durch aufsummierte Rechenfehler eskalieren. Starke Beschleunigungen durch plötzlich auftretende Kräfte (Kontrollkomandos des Animateurs, Kollisionen innerhalb der Szene) verschärfen die numerischen Probleme erheblich.
Ein weiteres großes Problem ist die Behandlung von Kollisionen. Da nicht im voraus bestimmt werden kann, wohin die Objekte sich bewegen, muß für jeden Simulationszeitpunkt untersucht werden, ob und wenn ja welche Objekte mit einander kollidieren, da daraus Kräfte bestimmt werden müssen, die die Abstoßung zwischen den Objekten realisieren. Problematisch ist dabei die zeitliche Rasterung. Die Kollisionen werden erst zu einem "Rasterzeitpunkt" entdeckt, haben in der Regel aber schon im davorliegenden Intervall stattgefunden. Für die Berechnung der Abstoßungskräfte ist aber die exakte geometrische Konstellation zum Zeitpunkt der Kollision nötig, die nicht mehr ohne weiteres rekonstruiert werden kann. Die Erkennung von Kollisionen ist in erster Linie ein geometrisches Problem und wird deshalb hier nicht weiter betrachtet.

Kontrolle der Bewegungen

Am Anfang dieses Aufsatzes wurde eine Benutzerschnittstelle gefordert, die den Animateur von der Definition schwer zu kontrollierender Gelenkkräfte befreit und High-Level-Kontrollmöglichkeiten zur Verfügung stellt.
Es wird davon ausgegangen, daß der Animateur speziell Interaktionen und Bewegungen auf höchster Ebene günstig durch natürliche Sprache beschreiben kann. Im Gegensatz dazu geht man davon aus, daß Bewegungen und Geometrieinformation auf unterer Ebene besser interaktiv über analoge Eingabegeräte (Computer mouse, Spaceball, etc.) zu kontrollieren sind. Infolgedessen stellen die meisten leistungsfähigen Animationssysteme Benutzerschnittstellen auf unterschiedlichen Abstraktionsebenen zur Verfügung. Die Produktion einer größeren Animation erfolgt außerdem meist in Zusammenarbeit unterschiedlich qualifizierter Animateure.
Auf der höchsten Ebene stehen sogenannte Task-Level-Kontrollschnittstellen zur Verfügung. Darunter fallen sprachliche Kontrollkonzepte wie zum Beispiel "Laufen", "Greifen", "Tanzen", etc. Für diese Konzepte stehen schon fertige, systemimmanente Umsetzungen zur Verfügung, die es ermöglichen, komplexe Bewegungsabläufe durch die Manipulation weniger Parameter zu kontrollieren. Diese Ebene erlaubt es zum Beispiel künstlerisch vorgebildeten Animateuren, sich relativ systemunabhängig auszudrücken.
Leider gibt es schon bei der Kontrolle physikalisch basierter Modelle auf höchster Ebene prinzipielle Probleme. Es kann generell nicht verhindert werden, daß ein Benutzer das System durch widersprüchliche Angaben vor unlösbare Probleme stellt. Was soll zum Beispiel passieren, wenn ein Benutzer eine Wand und einen Ball modelliert, die beide massiv sein sollen und dann fordert, daß der Ball durch die Wand fliegen soll ? Die Eigenschaften von Ball und Wand legen fest, daß der Ball von der Wand abprallt, der Benutzer fordert aber das Durchdringen. Diese Situation ist durch den Animateur leicht zu

vermeiden, jedoch kann der Benutzer generell nicht wissen, ob es die physikalischen Eigenschaften seiner Modelle erlauben, seine Kontrollanweisungen auszuführen, so daß Konfliktsituationen nicht zu vermeiden sind. Dieses Problem kann auf verschiedene Weise adressiert werden. So können zum Beispiel Präzedenzen zwischen Objekteigenschaften und Kontrollanweisungen definiert werden, die den Konflikt auflösen oder auch eine geschickte Benutzerführung entwickelt werden, die hilft, Konfliktsituationen zu vermeiden.
Abgesehen von dieser Problematik wird es in der Regel nicht ausreichen, die fertigen Task-Level-Konzepte zu benutzen, da schon bald Varianten dieser Konzepte nötig werden, die durch die bisherigen Implementierungen der Konzepte nicht abgedeckt sind. Ein Animateur stellt zum Beispiel fest, daß, wenn er einen Akteur in seiner Szene schleichen lassen will, das Konzept "Laufen" bei Wahl einer geringen Geschwindigkeit für diesen Fall keine befriedigenden Ergebnisse liefern kann. Auf unterer Ebene muß nun das Konzept "Schleichen" eingeführt werden. Hier könnte zum Beispiel eine interaktive Schnittstelle existieren, mit Hilfe derer Keyframes manipuliert werden können, durch deren Interpolation der visuelle Eindruck einer Schleichbewegung entsteht. Aus der interpolierten Bewegungssequenz könnten die jeweiligen Gelenkkräfte durch Inverse-Dynamics-Algortihmen bestimmt werden, die es dann später erlauben, unterschiedliche menschliche Modelle in einer physikalisch modellierten Umgebung schleichen zu lassen. Leider entspricht dieses Szenario nur der Vision eines zukünftigen, Animationssystems. Momentan wäre es realistischer, anzunehmen, daß ein Programmierer das Konzept Schleichen auf aller unterster Programmebene einführt ("dazuprogrammiert"). Generell gilt in heutigen Animationssystemen eine Programmierschnittstelle als unerläßlich.

Ausblick

Zukünftige Animationssysteme sollten homogene Kontrollstrukturen über alle Abstraktionsebenen unterstützen. Dem Animateur sollte es ermöglicht werden, in einer Top-Down-Vorgehensweise unter Verwendung von existierenden komplexen Modellen und Bewegungskonzepten neue Animationen zu entwickeln. Um dieses Ziel zu erreichen, werden Konzepte der künstlichen Intelligenz (Wissensbasen, Expertensysteme, etc.) in Animationssystemen eine immer wichtigere Rolle spielen. Durch neue Gerätetechnologien ("Virtual Reality") wird der Animateur ein besseres Gefühl ("Feedback") für die entworfenen Animationen entwickeln können. Voraussetzungen für alle diese Fortschritte sind jedoch effizientere Algorithmen auf leistungsfähigeren Hardware-Architekturen.