

# Seminar 3D-Modelling und Rendering

## Animation

Roland Reichle  
Fakultät für Informatik  
Universität Ulm  
rr4@informatik.uni-ulm.de

### ABSTRACT

Diese Arbeit beschäftigt sich mit dem Themengebiet der Animation. Nach einer kurzen allgemeinen Einführung sollen wichtige Begriffe erklärt und Anwendungsgebiete aufgezeigt werden. Anschließend werden die Entwicklungsphasen einer Animation skizziert und die wichtigsten Animationsmethoden vorgestellt. Außerdem werden auch die in Animationen häufig anzutreffenden Prinzipien der Inversen Kinematik und Collision Detection behandelt.

## 1. EINLEITUNG

### 1.1 Allgemeine Einführung

Das tägliche Leben ist geprägt durch dynamische Abläufe und bewegte Objekte. Deshalb ist es bei der Visualisierung von Prozessen unerlässlich, zeitliche Veränderungen des Zustands und der Position von Objekten zu berücksichtigen und geeignet darzustellen. Dies stellt das Ziel der Animation dar.

Im 19. Jahrhundert wurde dabei erkannt, dass dem Menschen durch eine schnelle Abfolge von Einzelbildern, die jeweils geringfügige Änderungen zum vorhergehenden Bild aufweisen und die jeweilige Konfiguration von Objekten zu gewissen Zeitpunkten darstellen, der Eindruck einer Bewegung vermittelt werden kann<sup>1</sup>. Ein allseits bekannter Ansatz hierzu ist das sogenannte Daumenkino oder ein Verfahren von Joseph Plateau (1801-1883), der Zeichnungen auf eine mit Schlitzen versehene Scheibe aufbrachte, die bei schneller Rotation und parallel zu einem Spiegel gehalten ebenfalls die Illusion bewegter Bilder hervorrief. Diese Konstruktion ist unter dem Namen *Phenakistoskop* bekannt (siehe Abbildung (1)). Charakteristisch für dieses Verfahren ist, dass schon für relativ kurze Sequenzen eine sehr grosse Anzahl von Einzelbildern notwendig ist, um eine ruckelfreie, stetige Bewegung zu vermitteln. Da dies erst ab einer Rate von 24 Bildern pro Sekunde erreicht wird, werden für eine einminütige Sequenz

<sup>1</sup>Der sogenannte Stroboskopische Effekt, entdeckt durch M.Faraday



Abbildung 1: Phenakistoskop (1832) von Joseph Plateau

bereits über 1000 Einzelbilder benötigt. An dieser Stelle setzt die computergestützte Animationsentwicklung an, um die Erstellung der einzelnen Bilder möglichst einfach und flexibel zu gestalten. Dabei sind vor allem bei der Generierung von 3D-Animationen aufgrund der erhöhten Komplexität der Berechnungen schnelle und effiziente Algorithmen unbedingt erforderlich.

### 1.2 Wichtige Begriffe

Der Begriff **Animation** leitet sich vom lateinischen Begriff "animare" ab, was soviel bedeutet wie Atem einhauchen oder mit Leben füllen. Die Aufgabe der Animation ist es nun, die Illusion der Bewegung bzw. Lebendigkeit von normalerweise leblosen Objekten, vor allem von Zeichnungen, zu vermitteln, was eigentlich ausschließlich durch oben genanntes Prinzip erreicht wird. Das **Frame** bezeichnet dabei das einzelne Bild einer Animation. Meist wird dieser Begriff jedoch noch etwas weiter gefasst, und auch der gesamte Berechnungs- und Renderingprozess miteinbezogen. Im Kontext der Animation wird häufig von der Einhaltung von **Constraints** gesprochen. Als Constraints werden in diesem Zusammenhang alle relevanten Beschränkungen und Bedingungen bezeichnet, die von der Animation zu berücksichtigen sind. Diese Beschränkungen können dabei zeitlicher Natur sein oder sich ganz allgemein auf die Konsistenz der Animation beziehen. Es soll zum Beispiel sichergestellt werden,

dass Objekte nicht durch andere “hindurchfliegen” können, und sich keine unnatürlichen Bewegungsabläufe ergeben.

### 1.3 Anwendungsgebiete

Ein wichtiges Anwendungsgebiet für 3D-Animationen stellt die **Wissenschaft** dar. Hierbei steht vor allem die Simulation und Analyse komplizierter Prozesse im Vordergrund. Da Wissenschaftler eher an der exakten Beschreibung der Verhaltensweisen und Reaktivität verschiedener Systeme als an schöner Grafik interessiert sind, wird hier das Rendering meist in den Hintergrund gedrängt. Der Versuch der Wissenschaft, das Verhalten und die Eigenschaften der Objekte eines Systems in mathematische Formeln zu fassen, erfordert meist den Einsatz von Partikelsystemen, deren Berechnung sich als sehr komplex und zeitaufwendig erweist.

In ähnlicher Weise kommen 3D-Animationen auch im Bereich der **Technik** zum Einsatz. Auch hier leisten sie einen wesentlichen Beitrag zur Simulation und Analyse der Eigenschaften neuer Entwicklungen. So z. B. werden 3D-Animationen eingesetzt, um die Aerodynamik bestimmter Bauteile von Flugzeugen bzw. Autos zu visualisieren.

Ein weiteres Gebiet, in dem 3D-Animationen immer mehr an Bedeutung gewinnen, stellt die **Unterhaltungsindustrie** dar. Hierbei sind vor allem Computerspiele zu nennen, für deren Erfolg aufwendige 3D-Echtzeit-Grafik bereits unabdingbar ist. Zwar sollen hier nicht komplexe Modelle und Abläufe exakt berechnet werden, jedoch ergibt sich eine besondere Herausforderung aus der Tatsache, dass der Anwender die Animation interaktiv beeinflussen können soll und somit Echtzeitberechnungen benötigt werden, die stabil und außerdem noch schön anzusehen sind. Ein sehr ähnliches Konzept stellen die sogenannten Virtuellen Welten dar, in der sich der Benutzer ebenfalls völlig frei bewegen können soll. Auch in Filmen kommen immer häufiger 3D-Animationen zum Einsatz. Entweder werden nur Teile des Films, wie z. B. ausgestorbene Wesen oder Kreaturen aus einer Phantasiewelt, als Animation berechnet und in den eigentlichen Film eingebunden, oder aber der gesamte Film besteht aus einer einzigen 3D-Animation, wie es in letzter Zeit bei Trickfilmen zu beobachten war (siehe Abbildung (2)). Der Vorteil gegenüber Computerspielen liegt hierbei in



Abbildung 2: Bild aus dem Trickfilm *Final Fantasy*

der Tatsache, dass die einzelnen Bilder vorausberechnet werden können, und dadurch der Berechnungsprozess prinzipiell fast beliebig kompliziert sein kann.

Da es im allgemeinen kompliziert ist, sehr realistische 3D-Animationen zu erzeugen, vor allem im Bereich der Wissenschaft, und deshalb immer bessere und effizientere Beschreibungsmodelle und Algorithmen benötigt werden, hält die 3D-Animation auch Einzug in den Bereich der **Forschung**.

## 2. ENTWICKLUNGSPHASEN EINER ANIMATION

### 2.1 Unterschied Echtzeitanimation und vorberechnete Animation

Bei der Konzeption einer Animation muss vor allem berücksichtigt werden, ob es sich um eine Echtzeitanimation handelt, wie sie vor allem im Bereich der Computerspiele und im Bereich der Virtuellen Welten anzutreffen ist, oder ob die Einzelbilder der Animation vorausberechnet werden können, wie dies bei Filmen oder in wissenschaftlichen Simulationen der Fall ist. Sind Echtzeitanimationen erwünscht, so muss sichergestellt werden, dass die Komplexität der Berechnungen so begrenzt ist, dass mindestens 24 Frames pro Sekunde berechnet werden können, was wesentliche Einschränkungen in der Exaktheit der Beschreibungsmodelle und auch Qualitätsverluste mit sich bringt. Können jedoch die Bilder im Voraus berechnet werden, so sind hinsichtlich der Komplexität der Berechnungen und der Aufwendigkeit der Darstellungen kaum Grenzen gesetzt.

### 2.2 Phasen

#### 2.2.1 Konzeption und Planung

Die erste Phase bei der Erstellung einer Animation umfasst die allgemeine Planung und Konzeption. Hierbei sind einige grundlegende Fragen zu beantworten. Zunächst sollte man sich darüber im Klaren sein, ob es sich um eine Echtzeitanimation oder um eine vorberechnete Animation handelt. Die grundlegenden Auswirkungen dieser Entscheidung auf die Animation wurden bereits oben erläutert. Außerdem sollte festgelegt werden, was eigentlich dargestellt werden soll und was mit der Animation erreicht werden soll. Es besteht nämlich ein wesentlicher Unterschied zwischen einer wissenschaftlichen Simulation und einer Animation im Bereich der Unterhaltung, bei der möglichst eindrucksvolle Effekte erwünscht sind. Weiterhin sollte man sich auch über den Detaillierungsgrad der Darstellung Gedanken machen. Vor allem bei Echtzeitanimationen, bei der die Frames schnell und effizient berechnet werden müssen, ist dies eine sehr entscheidende Fragestellung.

#### 2.2.2 Entwurf

Nachdem die obigen grundlegenden Fragen beantwortet wurden, kann nun mit dem eigentlichen Entwurf der Animation begonnen werden. Zunächst einmal sollten diejenigen Objekte der Szene identifiziert werden, die einer Bewegung bzw. einer Zustandsänderung unterworfen werden sollen. Anschließend wird für diese Objekte die Art der Bewegung oder Zustandsänderung festgelegt und dafür eine geeignete Animationsmethode ausgewählt. So zum Beispiel besteht die Möglichkeit, die Bewegung oder Zustandsänderung durch parametrisierte Kurven, durch Key-Framing oder auch mit Hilfe von Partikelsystemen darzustellen. All diese Animationsmethoden werden später noch näher erläutert. Vielfältige Möglichkeiten beim Entwurf der Animation ergeben sich durch die Tatsache, dass die Dynamik der Sze-

ne auch durch die Veränderung des Blickwinkels, der Perspektive und der Position des Betrachters erreicht werden kann. Deshalb sollte die Planung der "Kameraführung" beim Entwurf der Animation nicht zu kurz kommen. Gerade durch den Wechsel zwischen Nah- und Fernbetrachtung von Objekten können eindrucksvolle Effekte hervorgerufen werden. Hier sei aber nochmals auf den Unterschied von Echtzeitanimationen und vorberechneten Animationen hingewiesen. Bei Echtzeitanimationen, wie sie zum Beispiel in 3D-Shootern üblich sind, kann die "Kameraführung" kaum vorausgeplant werden, da der Blickwinkel und die Perspektive der Interaktion des Benutzers unterworfen sind.

### 2.2.3 Modellierung

Der erste Schritt bei der Modellierung der an der Animation beteiligten Objekte besteht darin, eine geeignete Modellierungsmethode auszuwählen. Dabei muss je nach Anwendungsgebiet abgewogen werden, ob Exaktheit oder schöne Grafik im Vordergrund stehen soll. In wissenschaftlichen Anwendungen wird meistens mehr Wert auf eine exakte Darstellung der Formen und Verhaltensweisen gelegt. Deshalb werden die Objekte meist sehr genau nachgebildet, während die Texturierung etwas vernachlässigt wird. In Computerspielen hingegen werden die Szenen häufig aus relativ einfachen Grundobjekten aufgebaut, und eine gewisse Detailtreue wird durch gezielten Einsatz von Texturen erreicht. Manchmal werden sogar ganze Objekte durch eine einzige Textur dargestellt, die gemäß dem aktuellen Blickwinkel ausgerichtet wird. Auch bei der Modellierung muss wieder die Berechnungszeit der Szene berücksichtigt werden. Bei Echtzeitgrafik muss weiterhin sichergestellt sein, dass die Modellierung nur so aufwendig ist, dass die für eine ruckelfreie Bewegung notwendige Framerate eingehalten werden kann.

### 2.2.4 Aufzeichnung und Nachbearbeitung

Nachdem die obigen Phasen abgeschlossen sind, kann mit der Aufzeichnung der Animation begonnen werden. Ist das Ergebnis nicht zufriedenstellend, so können in einem Nachbearbeitungsschritt verschiedene Parameter weiter angepasst bzw. ganze Modellaspekte umgeändert werden. Klar ist aber auch, dass bei interaktiven Animationen die Aufzeichnung entfällt. Eine Nachbearbeitung ist auch nur in der Form möglich, dass die Anwendung getestet wird, und bei Mängeln Anpassungen bzw. Verbesserungen im Modell vorgenommen werden. Eine zusätzliche Schwierigkeit ergibt sich auch aus der Vielfalt der Interaktionsmöglichkeiten, deren Anzahl meist so gross ist, dass nicht alle simuliert werden können.

## 3. ANIMATIONSMETHODEN

Im folgenden soll ein kurzer Überblick über die verschiedenen Animationsmethoden gegeben werden. Darüber hinaus sollen auch Vor- und Nachteile der jeweiligen Methoden diskutiert werden.

### 3.1 Separate Erzeugung der Einzelbilder

Dies ist wohl die einfachste Methode zur Erzeugung einer Animation. Es werden nacheinander alle für die Animation benötigten Einzelbilder manuell erzeugt. Der Vorteil dieser Methode ist ganz offensichtlich. Es wird weder ein exaktes Modell noch eine mathematische Beschreibungsstruktur für die bewegten Objekte benötigt. Allerdings kann das einfach

zu verstehende und anzuwendende Grundprinzip der Methode die Nachteile kaum aufwiegen. Zunächst einmal sei hier auf den enormen Aufwand bei der Erstellung der Animation hingewiesen, da, wie bereits oben erwähnt wurde, für eine einminütige Animation bereits mehr als 1000 Einzelbilder benötigt werden. Außerdem erweist sich diese Methode als äußerst unflexibel. Soll eine Änderung vorgenommen werden, so ist eine manuelle Anpassung aller betroffenen Frames notwendig. Für Echtzeitanimationen ist dieses Verfahren so wieso absolut unbrauchbar.

### 3.2 Beschreibung der Bewegungsabläufe durch Zustände

Dies ist ebenfalls eine sehr einfache Animationsmethode. Jedoch werden im Gegensatz zu der oben vorgestellten Methode die Bewegungsabläufe und Eigenschaftsänderungen bereits explizit durch die Einführung von Objektzuständen festgelegt. Jeder Zustand kann dabei Informationen über die Position, die Orientierung und weitere Eigenschaften des Objekts enthalten. Während der gesamten Animation, d. h. in jedem Einzelbild, befindet sich das Objekt in genau einem dieser Zustände. Die eigentliche Bewegung bzw. Eigenschaftsänderung wird nun dadurch erreicht, dass das Objekt von einem Frame auf das nächste in einen anderen Zustand wechselt. Ein wesentlicher Vorteil dieser Methode gegenüber dem oben vorgestellten Verfahren liegt nun darin, dass die Zustände meist mit Hilfe geeigneter Prozeduren initialisiert bzw. festgelegt werden können und die Einzelbilder anhand der Zustandsinformationen automatisch berechnet werden können. Dies stellt eine enorme Verringerung des Aufwands bei der Erstellung der Animation dar. Außerdem ist dieses Verfahren in begrenztem Maße auch für interaktive Animationen geeignet, da für jedes sich bewegende Objekt eigene Zustände definiert werden können und anhand von Benutzereingaben die Art und Geschwindigkeit der Zustandswechsel bestimmt werden kann. Auch ist eine Kombination dieser Methode mit anderen Verfahren, wie z. B. prozedurale Methoden oder Bewegungspfade denkbar. Für Echtzeitanimationen, wie sie in Computerspielen üblich sind, ist dieses Verfahren allerdings etwas problematisch, da die Zustandswechsel prinzipiell auf jede Rechengeschwindigkeit abgestimmt werden müssen, um ein realistisches Zeitverhalten gewährleisten zu können.

### 3.3 Prozedurale Animation und Bewegungspfade

Diese Methoden sollen das oben erwähnte realistische Zeitverhalten garantieren. Positions-, Orientierungs- und Eigenschaftsveränderungen werden hier mit Hilfe von mathematischen Funktionen bzw. Prozeduren dargestellt, die als Parameter die Zeit erhalten. Soll nun ein Frame berechnet werden, so werden einfach die entsprechenden Funktionen bzw. Prozeduren für den Zeitpunkt ausgewertet, zu dem das Frame dargestellt werden soll. Das Verfahren der prozeduralen Animation tritt dabei in mehreren Varianten auf. Am häufigsten werden parametrisierte Funktionen zur Formulierung der Positionsveränderung von Objekten verwendet. Man spricht dann von der Festlegung von **Bewegungspfaden**. Beispiele für einfache Bewegungspfade sind Strecken,

Kreis- (1) oder Ellipsenbahnen (2).

$$\vec{x}(t) = \vec{m} + r \cdot \begin{pmatrix} \cos t \\ \sin t \\ 0 \end{pmatrix} \quad (1)$$

$$\vec{x}(t) = \vec{m} + r \cdot \begin{pmatrix} a \cdot \cos t \\ b \cdot \sin t \\ 0 \end{pmatrix} \quad (2)$$

Meist sind allerdings kompliziertere Bewegungspfade als Kreise und Ellipsen erwünscht bzw. erforderlich. Abhilfe schafft hier die Verwendung von sogenannten Splines. Unter Splines oder Splinefunktionen versteht man in der Numerik Funktionen, welche stückweise Polynome sind und an den Nahtstellen, die durch Kontrollpunkte festgelegt werden können, gewissen Differenzierbarkeitsbedingungen genügen, um einen glatten Kurvenverlauf gewährleisten zu können. Auch für Splines gibt es eine Reihe von Varianten. So z. B. bestehen Bezier-Splines aus aneinandergefügten Bezier-Kurven<sup>2</sup>, die an den Verbindungsstellen ebenfalls gewissen Differenzierbarkeitskriterien genügen müssen.

Parametrisierte Funktionen lassen sich aber nicht nur zur Positionsbeschreibung von Objekten heranziehen, sondern können auch zur Beschreibung dynamischer Veränderungen der Ausdehnung und Position von Texturen verwendet werden. Wird z. B. die Position der Textur auf dem Objekt abhängig von der Zeit verändert und zusätzlich noch eine Streckung bzw. ein Zusammenziehen der Textur durch eine Funktion beschrieben, so lassen sich interessante Effekte, wie z. B. ein Pulsieren oder eine Strömung, simulieren. Darüber hinaus kann auch die Veränderung der Form von Objekten mit Hilfe von parametrisierten Funktionen mit zur prozeduralen Animation gerechnet werden. So zum Beispiel lässt sich eine Wasseroberfläche mit Wellen durch einfache Funktionen beschreiben, wie Abbildung (3) verdeutlichen soll. Es wurde lediglich eine Sinus-Funktion und eine einhüllende Exponentialfunktion zur Modellierung herangezogen.

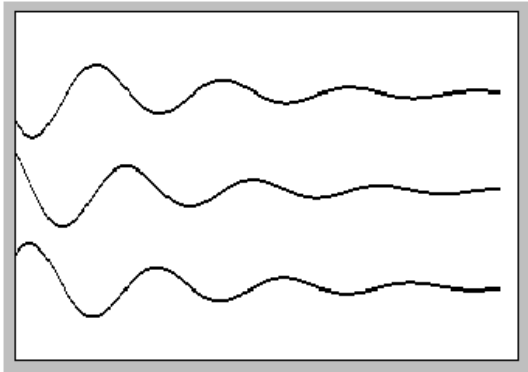


Abbildung 3: Wasseroberfläche mit Wellen

### 3.4 Key-Frames

Eine weit verbreitete Animationsmethode ist das Key-Framing. Die grundsätzliche Idee stammt dabei aus der klassischen Zeichentrick-Animation. Bei diesem Verfahren werden die Grundzüge der Animation durch Schlüsselbilder vorgegeben, in denen die Position und die Orientierung der

<sup>2</sup>Bezier-Kurven sind ebenfalls spezielle Polynomkurven über dem verallgemeinerten Bernsteinpolynom

sich bewegenden bzw. sich verändernden Objekte festgelegt sind. Die Zwischenbilder werden anschließend durch einen entsprechenden Algorithmus berechnet. Dazu ist es allerdings notwendig, dass die gesamte Szene durch eine Reihe zeitabhängiger Parameter bestimmt werden kann. Diese Parameter werden dann interpoliert, um zu jedem beliebigem Zwischenzeitpunkt die korrekten Darstellungen zu erhalten. Grundsätzlich wird zwischen Positionsparametern und Orientierungsparametern unterschieden, für deren Interpolation auch verschiedene Verfahren zum Einsatz kommen. Allgemein spricht man jedoch von Animationsparametern. Zu berücksichtigen ist hier, dass die Parameter je nach Komplexität der zu erstellenden Animation gewissen Bedingungen, sogenannten Constraints, unterworfen sind. Außerdem können funktionale Abhängigkeiten zwischen den Parametern auftreten. Zum Beispiel hängt die Position der Schulter in Laufrichtung eines Charakters gegenläufig von der Fuß- und Hüftposition ab. Zudem dürfen sich keine inkonsistenten Winkel bzw. Positionen von Einzelgliedern ergeben. Auf diese Problematik wird später im Abschnitt über Inverse Kinematik noch näher eingegangen.

#### 3.4.1 Interpolation der Position

Die Interpolation der Positionsparameter wird auf der Basis geeigneter Kurven-Klassen durchgeführt. Das eigentliche Problem der Positionsinterpolation kann auf das Auffinden einer Beschreibungsform derjenigen Kurve reduziert werden, die durch die Zeit-Positions-Wertepaare festgelegt ist, welche durch die Schlüsselbilder gegeben sind. Da Splines, Bezier-Splines oder auch andere Spline-Varianten, wie bereits oben erwähnt wurde, gerade durch die Angabe solcher Kontrollpunkte konstruiert werden, sind diese Kurven-Klassen für die Interpolation der Positionsparameter sehr gut geeignet. Es existieren eine Reihe von Algorithmen zur Konstruktion solcher Kurven. Erwähnt hierbei seien der **de-Casteljau-Algorithmus** zur Konstruktion von Bezier-Kurven und der **Catmull-Rom-Ansatz**<sup>3</sup> zur Interpolation von Bezier-Splines. Das Verfahren nach Catmull-Rom sucht dabei (z. B. kubische) Bezier-Kurven zwischen je zwei benachbarten Punkten und verbindet diese dann zu einer zusammenhängenden Kurve.

#### 3.4.2 Interpolation der Orientierung

Die Interpolation der Orientierungsparameter erweist sich manchmal als etwas problematisch, da einige straightforward-Ansätze auf den ersten Blick zwar richtig erscheinen, jedoch bei genauerer Betrachtung nicht funktionieren bzw. unsinnige Ergebnisse liefern. Man kann davon ausgehen, dass eine Orientierung durch eine bestimmte Rotation bezüglich einer Ausgangslage beschrieben wird. Für diese Rotation gibt es nun mehrere Beschreibungsformen, die sich allerdings völlig unterschiedlich für die Interpolation der Orientierungsparameter eignen. Üblicherweise ist eine Rotation im dreidimensionalen Raum durch die Angabe einer Matrix gekennzeichnet. Matrix (3) stellt z. B. eine Rotation um den Winkel  $\phi$  um die z-Achse dar.

$$R(\phi) = \begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

<sup>3</sup>siehe auch E. Catmull, R. Rom, A Class of interpolating Splines 1974

Sind nun zwei Orientierungen durch zwei Rotationsmatrizen  $R_1$  und  $R_2$  gegeben, wobei  $R_1$  eine Drehung um die z-Achse um  $-90^\circ$  und  $R_2$  eine Drehung um  $+90^\circ$  darstellt, so erscheint die direkte Interpolation der Rotationsmatrizen ein durchaus sinnvoller Ansatz zu sein. Der Ansatz:

$$R_0 = \frac{1}{2} (R_1 + R_2)$$

liefert jedoch statt der erwarteten Einheitsmatrix die völlig unsinnige Rotationsmatrix

$$R_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Die direkte Interpolation der Rotationsmatrizen führt also nicht zu korrekten Ergebnissen und ist daher für die Bestimmung der Orientierungsparameter beim Key-Framing nicht geeignet.

Eine weitere Beschreibungsform für Orientierungen stellen die sogenannten **Euler-Winkel** dar. Die Orientierung wird dabei durch die Hintereinanderausführung von Drehungen um die x-, y- und z-Achse im lokalen Koordinatensystem des Objekts erzeugt. Prinzipiell ist die Abarbeitungsreihenfolge zwar beliebig, muss aber in einer Anwendung fest gewählt werden, um zu konsistenten Zuständen zu gelangen. Die Gesamttrotation ergibt sich gemäß Gleichung (6).

$$R(\phi_x, \phi_y, \phi_z) = R_3(\phi_z) \circ R_2(\phi_y) \circ R_1(\phi_x) \quad (6)$$

Dabei treten allerdings mehrere Probleme auf. Zum einen ist dies der sogenannte **Gimbal-Lock**. Aufgrund der gegenseitigen Beeinflussung der Euler-Winkel reduzieren sich  $\phi_x$  und  $\phi_z$  bei der Wahl von  $\phi_y$  zu  $\frac{\pi}{2}$  zu einem Freiheitsgrad, da die Wahl des Winkels  $\phi_x$  durch eine entsprechende Wahl von  $\phi_z$  kompensiert werden kann. Dies soll durch folgende Gleichung verdeutlicht werden.

$$R\left(\phi_x, \frac{\pi}{2}, \phi_z\right) = \begin{pmatrix} 0 & \sin(\phi_x - \phi_z) & \cos(\phi_x - \phi_z) \\ 0 & \cos(\phi_x - \phi_z) & -\sin(\phi_x - \phi_z) \\ -1 & 0 & 0 \end{pmatrix}$$

Ein weiteres Problem sind die Mehrdeutigkeiten, die in Verbindung mit den Euler-Winkeln auftreten. Zum Beispiel beschreiben  $R(\pi, 0, 0)$  (siehe Abbildung (4)) und  $R(0, \pi, \pi)$  (siehe Abbildung (5)), wie sich leicht nachvollziehen lässt, dieselbe Orientierung. Die Interpolationspfade sind jedoch völlig unterschiedlich. Dies wird in Gleichung (8) deutlich.

$$R(t\pi, 0, 0) \neq R(0, t\pi, t\pi), t \in ]0, 1[ \quad (8)$$

Aus diesen Gründen sind Euler-Winkel für die Beschreibung der Orientierung bei Key-Frame-Animationen ungeeignet.

Abhilfe schafft die Verwendung von **Quaternionen**. Diese stellen eine Erweiterung der komplexen Zahlen dar. Schon mit "normalen" komplexen Zahlen konnten Drehungen um den Winkel  $\phi$  im Zweidimensionalen durch eine einfache Multiplikation mit der Zahl  $a = \cos \phi + i \sin \phi$  dargestellt

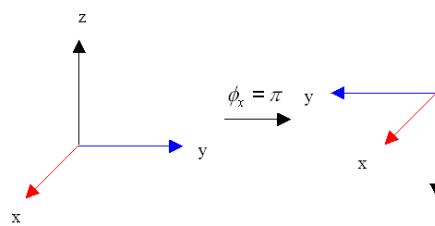


Abbildung 4: Orientierung bei Gesamttrotation mit  $R(\pi, 0, 0)$

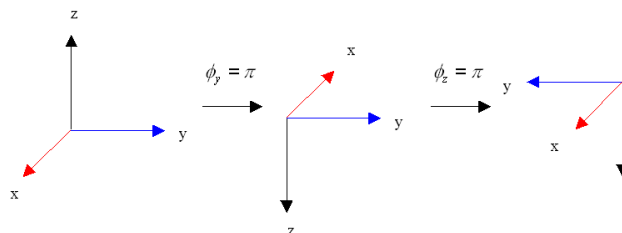


Abbildung 5: Orientierung bei Gesamttrotation mit  $R(0, \pi, \pi)$

werden. Eine ähnliche Eigenschaft weisen auch Quaternionen auf. Quaternionen werden als komplexe Zahl mit einem Realteil  $s$  und drei Imaginärteilen  $x, y$  und  $z$  definiert.

$$\underline{q} := s + ix + jy + kz = (s, \vec{v}), \vec{v} = (x, y, z)$$

wobei  $i, j$  und  $k$  die drei imaginären Einheiten darstellen. Die Rechenregeln für Quaternionen sind sehr ähnlich zu denen der komplexen Zahlen. Besonderheiten ergeben sich allerdings im Bezug auf Kommutativität und die Verrechnung der Imaginärteile untereinander. Eine Rotation um eine Achse  $\vec{v}$  mit Winkel  $\phi$  kann mit Hilfe eines Einheitsquaternions

$$\underline{q} = \left( \cos\left(\frac{\phi}{2}\right), \sin\left(\frac{\phi}{2}\right) \cdot \vec{v} \right)$$

beschrieben werden durch:

$$R_{\underline{q}}(\vec{p}) = \underline{q} \underline{q}(\vec{p}) \bar{\underline{q}}$$

wobei  $\underline{q}(\vec{p}) = (0, \vec{p})$  und  $\vec{p}$  Ortsvektor von P

Eine Hintereinanderausführung von Drehungen wird durch einfache Multiplikation von Einheitsquaternionen erreicht. Mehrdeutigkeiten wie bei Eulerwinkeln treten hierbei nicht auf. Die Interpolation der Orientierung erfolgt dann mit der **linearen sphärischen Interpolation** oder der **Sphärischen-Spline-Interpolation**.

Die obigen Ausführungen verdeutlichen, dass das Key-Framing bereits zu den etwas komplizierteren Animationsmethoden zu zählen ist. Dafür bietet diese Methode auch eine relativ große Flexibilität. Soll die Animation umgeändert werden, so sind lediglich die Schlüsselbilder anzupassen. Offensichtlich ist auch, dass das Key-Framing für interaktive Animationen bzw. für Echtzeitanimationen nur sehr bedingt geeignet ist.

### 3.5 Partikelsysteme

Diese Animationsmethode ist sehr stark an exakten physikalischen Grundmodellen orientiert und wird deshalb häufig in wissenschaftlichen Simulationen eingesetzt. Ein Partikelsystem besteht dabei aus einer Menge von einzeln berechneten und modellierten Teilchen, den sogenannten Partikeln. Jedem dieser Teilchen sind gemäß dem zugrundeliegenden physikalischen Modell bestimmte Attribute wie z. B. Position, Orientierung, Geschwindigkeit, Größe, Farbe und Lebensdauer zugeordnet. Das Verhalten der Partikel wird durch prozedurale Beschreibungen, Gleichungssysteme und durch zusätzliche Attribute festgelegt. Dadurch ist man in der Lage mit Partikelsystemen ganz unterschiedliche Phänomene wie Rauch, Feuer, Schneestürme, Wolken, fließendes Wasser und viele weitere Naturgegebenheiten zu modellieren und zu animieren. Natürlich werden die Partikel je nach System unterschiedlich definiert, z. B. als Rauchteilchen, Schneeflocken, Wassertropfen usw., und mit an das System angepassten Attributen versehen (Dichte, Transparenz usw.). Ganz offensichtlich ist auch die Tatsache, dass die zugrundeliegenden Systeme beliebig kompliziert werden können, eben bis alle Aspekte des realen physikalischen Modells einbezogen sind. Jedoch liefern auch schon vereinfachte Modelle annehmbare Ergebnisse, wie in Abbildung (6) veranschaulicht werden soll.

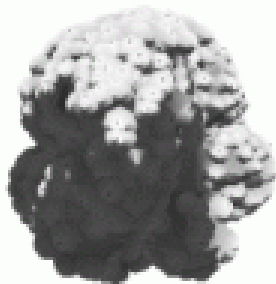


Abbildung 6: Simulation von Wolken mit Partikelsystemen

#### 3.5.1 Basismodelle

Zu diesen einfacheren Modellen gehören die Modelle von W. T. Reeves und Karl Sims, deren Grundideen im folgenden erläutert werden sollen.

##### Das Modell von Reeves

Nach Reeves bieten sich Partikelsysteme für die Modellierung und Animation einer Vielzahl von dynamischen Objekten an. Dazu sind beispielsweise Wasser, Wolken, Feuer und Rauch zu zählen. All diesen Objekten ist gemeinsam, dass sie keine klar definierte, durch Polygone beschreibbare Oberfläche besitzen. Außerdem sind sie betreffend Form

und Bewegung sehr variabel und von stochastischen Prozessen beeinflussbar. Partikelsysteme nach dem Modell von Reeves besitzen dabei mehrere Vorteile. Zum einen ist es möglich, relativ komplexe Systeme mit relativ geringem Aufwand nachzubilden. Außerdem kann die Detailliertheit der Darstellung durch die verwendete Anzahl von Partikeln angepasst werden.

Die eigentliche Animation der Partikel wird dabei in mehrere Phasen unterteilt. Meist wird ein Modell mit fünf Phasen angenommen, die nun vorgestellt werden sollen.

In **Phase 1** werden neue Partikel erzeugt. Dabei werden stochastische Annahmen miteinbezogen und die Durchschnittszahl der bereits vorhandenen Partikel berücksichtigt. Dies kann bezüglich des gesamten Animationsbereichs oder auch nur bezüglich eines Ausschnitts der Animation geschehen. **Phase 2** ist durch die Attributierung der gerade erzeugten Partikel gekennzeichnet. Den Teilchen werden nun Eigenschaften wie Position, Beschleunigung, Farbe, Transparenz, Form und Lebenszeit zugewiesen. Auch hier werden wieder stochastische Annahmen (z. B. Mittelwerte von Attributwerten) einbezogen. Da in dieser Phase die realen physikalischen Eigenschaften des zu betrachtenden Systems auf Attributwerte der künstlichen Partikel umgesetzt werden, ist diese Phase als entscheidend für den Realismus der Animation anzusehen. In **Phase 3** sterben diejenigen Partikel ab, deren maximale Lebensdauer abgelaufen ist bzw. deren Farbe bzw. Transparenz bestimmte Werte überschritten hat, die deshalb nicht mehr sichtbar und so für die Animation irrelevant sind. Dies kann auch der Fall sein, wenn sich das Teilchen außerhalb des Animationsbereichs bewegt oder aufhält. Die Bewegung der Partikel wird in **Phase 4** generiert. Dabei sind nicht nur die anfangs gesetzten Attribute von Bedeutung, sondern es müssen noch zusätzlich äußere Einflüsse, wie die auf die Partikel wirkenden Kräfte (z. B. Gravitation), aber auch andere Aspekte wie die Energieentwicklung oder gegenseitige Beeinflussung der Teilchen berücksichtigt werden. **Phase 5** dient der eigentlichen Animation. Dabei bedient man sich der üblichen Verfahren.

Anzumerken ist außerdem, dass dieses Verfahren es erlaubt, Partikelsysteme einer gewissen Rekursion zu unterwerfen. Partikelsysteme können nämlich aus Partikeln aufgebaut sein, die ihrerseits wiederum Partikelsysteme bilden.

##### Das Modell von Sims

In einem Artikel geht Sims davon aus, dass die Partikel mit Hilfe eines parallelen Supercomputers bearbeitet und berechnet werden können, der mehrere tausend Prozessoren besitzt, die sich ihrerseits wieder in mehrere virtuelle Prozessoren untergliedern lassen. Der Rechner ist somit in der Lage, jedes Teilchen mit einem eigenen (virtuellen) Prozessor zu bearbeiten. Da jedes Partikel in seinem Verhalten gleich und den selben physikalischen Modelleigenschaften unterworfen ist, ist laut Sims eine parallele Verarbeitung mit vielen Prozessoren für Partikelsysteme optimal geeignet. Problematisch jedoch könnte sich die Tatsache auswirken, dass sich die Partikel in ihrer Entwicklung gegenseitig beeinflussen können. Zur Programmierung wird Starlisp herangezogen, das eine parallele Erweiterung von Lisp darstellt. Auch bei Sims ist eine möglichst exakte Abbildung des zugrun-

deliegenden physikalischen Modells Voraussetzung für eine realistische Simulation. Jedoch sollte man sich auch nicht zu stark an das physikalische Grundmodell klammern, vor allem wenn ein bestimmtes Verhalten des Partikelsystems erwünscht ist, das so nicht exakt aus dem Grundmodell abzuleiten ist. In seinem Modell stellt Sims deshalb eine Reihe einfacher Methoden vor, mit denen sich korrekt erscheinende Effekte erzielen lassen. Bewegungsgleichungen löst Sims dabei mit Hilfe Eulers Integrationsmethoden mit einer Näherung durch ein kleines Zeitintervall:

$$x = x_0 + \int v dt$$

$$v = v_0 + \int a dt$$

bzw. mit der oben angesprochenen Näherung:

$$x = x_0 + v \cdot \Delta t$$

$$v = v_0 + a \cdot \Delta t$$

Bei der Erzeugung eines neuen Partikels wird diesem ein virtueller Prozessor zugewiesen. Die Eigenschaften der Partikel (im Wesentlichen sind dies die gleichen wie bei Reeves) werden in einer Datenstruktur abgelegt, die dann an den virtuellen Prozessor übergeben wird. Sims beschreibt außerdem noch ein Verfahren, um die Bewegung der Teilchen flüssiger erscheinen zu lassen. Er gibt für ein Partikel zwei Positionswerte an, einen für den Kopf, den anderen für den Schwanz des Teilchens. Bei der Bewegung erhält nun der Schwanz als neue Position die alte Position des Kopfes. Die eigentliche Bewegung wird durch mehrere unterschiedliche Ansätze erzeugt. Beispiele hierfür sind explizite Positionsänderungen und Geschwindigkeitsänderungen bedingt durch Beschleunigungen. Außerdem beschreibt Sims spezielle Ansätze wie Strudel und Dämpfungen.

Die eigentliche Darstellung der Partikel ist relativ kompliziert. Im wesentlichen beruht sie darauf, dass jeder Prozessor einen Teil der Pixel eines Partikels berechnet. Abschließend werden die Ergebnisse aller Prozessoren miteinander verglichen und nach bestimmten Kriterien eine korrekte Darstellung ermittelt.

### 3.5.2 Weitere Modelle

Ein für den Endbenutzer relativ einfacher Ansatz wurde von **J. Weichert und D. Hartmann** vorgestellt. Er beruht auf der Kombination von vorberechneten Animationsprimitiven, wie z. B. Quellen, Senken von Objekten oder Wirbel. Diese müssen nur noch entsprechend zusammengefügt werden, um die fertige Animation zu erhalten. Komplikationen ergeben sich allerdings, wenn sich die Partikel in ihrem Verhalten gegenseitig beeinflussen, was zum Beispiel bei Zusammenstößen der Fall ist. Die Kombination der Primitive verschiedener Objekte geschieht dabei über die numerische Lösung der **Navier-Stokes-Gleichungen**. Diese Gleichungen beschreiben mathematisch die Bewegung von Flüssigkeiten, wobei jedes Partikel wiederum einzeln berechnet und verschoben wird und als Vektoren dargestellt ist. Das gesamte Modell beschreibt demnach ein Vektorfeld. Auswirkungen der Viskosität auf die Bewegung kann ebenso berücksichtigt werden wie der Einfluss der Dichte.

Ein weiteres Modell, das auf Partikelsystemen beruht und

zur Beschreibung von Wassermengen dient, ist das sogenannte **Shallow-Water**. Hier wird die zu animierende Wassermasse in einzelne Zellen aufgeteilt. Zwar beruht dieses Modell ebenfalls auf den Gleichungen von Navier-Stokes, jedoch werden verschiedene Vereinfachungen gemacht. Die Wasseroberfläche wird nur als Höhenfeld angegeben, die Wasserteilchen besitzen keine vertikale Beschleunigung und die horizontale Beschleunigung innerhalb einer Zelle wird als konstant angenommen. Aufgrund dieser Vereinfachungen lassen sich allerdings einige reale Aspekte wie z. B. eine Brandung mit brechenden Wellen nicht simulieren.

### 3.5.3 Animation von Haaren

Die Animation von Haaren (siehe Abbildung (7)) erweist sich als äußerst schwierig, vor allem wenn man realistische Effekte erzielen will. Am ehesten gelingt dies noch bei Verwendung von Partikelsystemen. Jedes Haar wird dabei durch mehrere miteinander verbundene Partikel gebildet, woraus sich eine wesentlich höhere Komplexität im Vergleich zu den weiter oben besprochenen Modellen ergibt. Es müssen nämlich zusätzlich noch Aspekte wie Elastizität und maximale Biegung von Haaren berücksichtigt werden. Diese sind dann durch zusätzliche Attribute oder Constraints (z. B. max. Winkel zwischen den Segmentpartikeln) in das Modell aufzunehmen. Durch die Tatsache, dass das Haar an einer Stelle fixiert, nämlich angewachsen ist, und dadurch die Position und Bewegung der anderen Segmentpartikel wesentlich eingeschränkt ist, müssen grundsätzlich sogar noch Prinzipien von Direkter und Inverser Kinematik<sup>4</sup> einbezogen werden. Neben diesen Aspekten gilt es auch noch die Wechselwirkungen und gegenseitige Beeinflussung von Haaren zu beschreiben und zu modellieren. Vor allem die Vielzahl der Kollisionen stellt hierbei ein wesentliches Problem dar. Effiziente Algorithmen zur Collision-Detection sind deshalb unerlässlich. Da die Form der Haare durch die Einteilung in Segmentpartikel häufig etwas unrealistisch ist, werden häufig Splines dazu verwendet, um die Haarsilhouette etwas zu glätten.

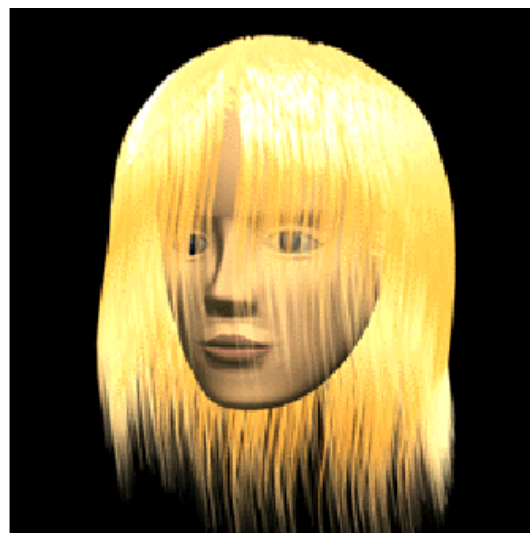


Abbildung 7: Haare modelliert durch ein Partikelsystem

<sup>4</sup>siehe auch Kapitel 4

## 4. EINHALTUNG VON KONSISTENZ-BEDINGUNGEN

Im Vergleich zu statischen Bildern sollen Animationen Bewegungen und dynamische Aspekte simulieren. Dadurch treten allerdings neue Problemstellungen auf, die sich vor allem auf die Konsistenz der Szenen und Darstellungen beziehen. Insbesondere ist dies bei interaktiven Animationen der Fall, deren Steuerung dem Benutzer obliegt. Zum Beispiel soll ein Flugzeug nicht durch einen Berg hindurchfliegen können, und ein Roboterarm muss in zusammenhängender Form mit realistischen Winkeln zwischen den Einzelgliedern verbleiben, falls der Greifer an eine bestimmte Position bewegt wird. Lösungsmöglichkeiten für diese Probleme sollen im folgenden vorgestellt werden.

### 4.1 Inverse Kinematik

Die Inverse Kinematik wird vor allem dazu verwendet, korrekte Konfigurationen von Gelenkkörpern zu berechnen. Dadurch kann z. B. das oben erwähnte Problem des Roboterarms gelöst werden. Um ein Verständnis für Inverse Kinematik zu bekommen, muss man sich allerdings zuerst mit den Prinzipien von Hierarchischer Verknüpfung und Direkter Kinematik auseinandersetzen.

#### 4.1.1 Hierarchische Verknüpfung

Die Grundidee der hierarchischen Verknüpfung soll am Beispiel eines menschlichen Arms erläutert werden. Der menschliche Arm besteht aus einem Oberarm, einem Unterarm und einer Hand mit den zugehörigen Fingern. All diese Einzelglieder sind durch Gelenke verknüpft, die bestimmte Winkel zwischen diesen Gliedern zulassen. Wird nun der Oberarm um einen bestimmten Winkel bewegt, so müssen entsprechend der Unterarm und die Hand mit den Fingern um eben diesen Winkel mitbewegt werden. Bei einer Veränderung des Winkels zwischen Unterarm und Hand bleiben jedoch der Ober- und Unterarm in Ruhe. Eine Hierarchie wird in diesem Modell also so umgesetzt, dass bei einer Bewegung eines Teils die mit ihm verbundenen Objekte, welche in der Hierarchie tiefer angesiedelt sind, mitbewegt werden müssen, während hierarchisch höherstehende Objekte in ihrer Position verbleiben.

#### 4.1.2 Direkte Kinematik

Soll ein Roboterarm oder allgemein ein Gelenkkörper gesteuert werden, so sind vor allem die Position und die Orientierung des Endeffektors (der Teil des Roboters, der mit einem bestimmten Werkzeug, z. B. einem Greifer oder einem Schrauber, versehen ist) entscheidend. üblicherweise ist der Endeffektor in der Hierarchischen Verknüpfung ganz unten angesiedelt. Bei Anwendung von Direkter Kinematik werden seine Position und Orientierung nun dadurch bestimmt, dass schrittweise die Winkel zwischen den Einzelgliedern angegeben werden. Die Reihenfolge der Angaben ist dabei durch die Hierarchische Verknüpfung vorgegeben. Winkel zwischen hierarchisch höherstehenden Objekten werden zuerst mit Werten versehen.

#### 4.1.3 Inverse Kinematik

Die Inverse Kinematik geht genau den entgegengesetzten Weg. Hier werden die Gelenkwinkel in Abhängigkeit der Position und Orientierung des Endeffektors bestimmt. Die Transformation ist ungleich schwerer als die der Direkten

Kinematik. Allerdings ist dies nicht nur in Animationen der weitaus sinnvollere Ansatz, denn, wie bereits oben angedeutet wurde, ist es meist die Position oder die Bewegung des Endeffektors, die bekannt bzw. bestimmten Bedingungen unterworfen ist. Betrachtet man die Inverse Kinematik unter rein mathematischen Aspekten, so ist sie gleichzusetzen mit der Lösung eines nichtlinearen Gleichungssystems, wobei natürlich die Frage nach der Existenz und Geschlossenheit von Lösungen beantwortet werden muss. Es gibt durchaus Achsenanordnungen, für die geschlossene Lösungen existieren, doch in den meisten Fällen muss auf eine numerische Lösung zurückgegriffen werden. Für diese gilt: Für alle Systeme mit Rotations- und Translationsgelenken, die insgesamt höchstens 6 Freiheitsgrade in einer einzelnen Kette haben, ist das Problem der Inversen Kinematik numerisch lösbar.

Im folgenden soll nun eine Technik zur Lösung des Problems für Gelenkkörper kurz erläutert werden.  $X$  stehe hierbei für die Stellung des Endeffektors und  $\Theta$  für die Gelenkwinkel. Anstatt nun

$$X = f(\Theta)$$

zu lösen, beschränkt man sich auf die Lösung der Gleichung

$$\dot{X} = J(\Theta) \cdot \dot{\Theta}$$

wobei die Berechnung der Jacobi-Matrix<sup>5</sup>  $J(\Theta)$  speziell für Gelenkkörper angepasst wurde. Durch dieses Verfahren erhält man eine Näherung für das gesuchte  $\Theta$ , die durch Iteration weiter verbessert werden kann.

### 4.2 Collision Detection

Wie bereits weiter oben angedeutet wurde, ist eine Kollisionserkennung für die meisten interaktiven Animationen unerlässlich. Die verwendeten Algorithmen müssen dabei sehr effizient sein, da bei komplexen Szenenaufbauten sehr viele Überprüfungen vorgenommen werden müssen. Da diese Algorithmen sehr schnell relativ komplex werden, beschränken wir uns im folgenden auf zwei Spezialfälle. Die Kollisionserkennung zwischen zwei Kugeln und zwischen zwei Quadern. Die Kollisionserkennung zwischen zwei Kugeln kann dabei straightforward erfolgen. Es muss lediglich überprüft werden, ob der Abstand der Mittelpunkte größer oder kleiner als die Summe der Radien ist (siehe Abbildung (8)).

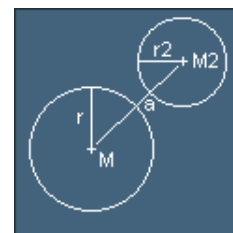
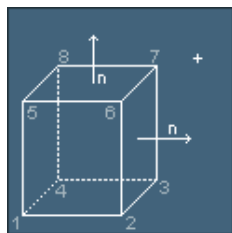


Abbildung 8: Kollisionserkennung zwischen Kugeln

Als etwas komplizierter, aber immer noch relativ einfach erweist sich die Kollisionserkennung zwischen zwei Quadern (siehe Abbildung (9)). Zunächst einmal muss ein Verfahren entwickelt werden, das eine Überprüfung erlaubt, ob ein

<sup>5</sup>Die Jacobi-Matrix ergibt sich als Zusammenfassung der partiellen Ableitungen von Funktionen  $R^n \rightarrow R^m$

Punkt innerhalb oder außerhalb eines Quaders liegt. Dies kann in zwei Schritten erfolgen. Zuerst werden die Normalenvektoren aller Seiten so bestimmt, dass sie nach außen gerichtet sind.



**Abbildung 9: Kollisionserkennung zwischen Quadern**

Gilt dann die Bedingung (14) für den Normalenvektor von mindestens einer Seite, so liegt der Punkt außerhalb des Quaders.

$$\vec{n} \cdot (\vec{p} - \vec{b}) > 0 \quad (14)$$

Wobei  $\vec{p}$  für den Ortsvektor des betrachteten Punktes steht, und  $\vec{b}$  für den Ortsvektor eines Punktes auf der gerade betrachteten Quaderseite. Es ist nun einfach festzustellen, ob sich zwei Quader überschneiden. Die obige Überprüfung wird einfach für alle Eckpunkte eines Quaders durchgeführt.

Bei der Kollisionserkennung zwischen Kugel und Quader kann ähnlich vorgegangen werden. Hier muss allerdings noch der Abstand des Kugelmittelpunktes von den Ebenen, die durch die Seitenflächen bestimmt sind, berechnet werden. Dies kann z. B. über die Hesse-Normalen-Form erreicht werden.

Interessant ist auch ein Ansatz zur Überprüfung, ob ein Punkt innerhalb oder außerhalb eines Ellipsoids liegt. Mit Hilfe einer geeigneten Transformationsmatrix lässt sich der Ellipsoid in die Form einer Kugel bringen. Die oben genannte Überprüfung wird dadurch trivial.

Es ist allerdings noch anzumerken, dass hier nur relativ spezielle Fälle betrachtet wurden. Berücksichtigt man allgemeinere Objektformen und Situationen, so wird die Kollisionserkennung recht schnell sehr kompliziert.

## 5. ZUSAMMENFASSUNG UND AUSBLICK

Die obigen Ausführungen verdeutlichen, dass sich die verschiedenen Animationsmethoden wesentlich in ihrer Komplexität unterscheiden. Entscheidend bei der Erstellung einer Animation ist deshalb die Phase der Konzeption und Planung. Man sollte bereits frühzeitig im Klaren darüber sein, was man darstellen will, zu welchem Zweck und mit welcher Qualität bzw. Realismusstufe, damit die geeignete Methode ausgewählt werden kann und unnötige Komplexität vermieden werden kann. Wichtig zu beachten ist auch der Unterschied zwischen interaktiven Echtzeitanimationen, wie sie in Computerspielen auftreten, und vorberechneten Animationen. Bei Echtzeitanimationen muss die Darstellungsmethode und Qualität dahingehend angepasst

sein, dass immer eine Framerate von 24 Bildern pro Sekunde erreicht werden kann. Bei vorberechneten Animationen hingegen sind der Komplexität und Aufwendigkeit der Algorithmen und Berechnungen kaum Grenzen gesetzt.

Weitere Herausforderungen im Bereich der Animation ergeben sich auch im Bezug auf die Darstellung der Mimik und Muskelbewegungen im Gesichtsbereich, zusammengefasst unter dem Schlagwort *facial animation*. Das wesentliche Problem hierbei liegt in der Tatsache, dass man nur sehr schwer, falls überhaupt zu geschlossenen Modellen gelangen kann, die Gesichtsausdrücke ädaquat zu beschreiben vermögen. Für künstliche Charaktere in Filmen interessant ist auch das Problem der Synchronisation der Lippenbewegungen mit der Sprache (lip synchronisation). Auch hier ist es schwierig ein geeignetes Modell zur Darstellung zu finden.

## 6. REFERENCES

- [1] Patrick Klein, Ernst-Joachim-Preussler, Animation von Rauch und Flüssigkeit, <http://www.informatik.uni-frankfurt.de/~erps/animation/welcome.html>
- [2] Professor Dr. Andreas Kolb, Keyframe-Animation
- [3] Friedrich Wagner, Die Steuerung von Gelenkkörpern <http://www.informatik.uni-rostock.de/~wagner/Texte/preprint/node4.html>
- [4] W. T. Reeves, Particle Systems - A Technique for Modeling a Class of Fuzzy Objects
- [5] <http://www.neobrothers.de/tutorials/3dcollision.html>
- [6] <http://courses.ncssm.edu/gallery/collections/toys/html/exhibit07.htm>
- [7] [http://www.iemag.com/news/daily/e3\\_preview/square.asp](http://www.iemag.com/news/daily/e3_preview/square.asp)