Computerunterstützte Animation
Objektorientierte Methoden und Konzepte zur Simulation von Multikörpersystemen

Jörg Reiberg

14.07.1995

1 Einleitung

Die Darstellung von Bewegung und Verhalten eines komplexen Objekts im dreidimensionalen Raum stellt häufig ein großes Problem dar. Das liegt zum einen an der Schwierigkeit, diese Objekte interaktiv zu erstellen und zu verändern, zum anderen an den hohen Anforderungen an die Rechenleistung bei der Implementierung. Ein Konzept für die Simulation solcher Objekte ist die Aufstellung von teilweise recht großen Systemen (partieller) Differentialgleichungen und deren numerische Berechnung, sowie allen damit verbundenen Schwierigkeiten wie Stabilität, Konvergenz und Rechengeschwindigkeit.

Ein neueres Konzept, mit dem wir uns im folgenden beschäftigen, ist das *Particle-based-Modelling*. Dabei wird das Verhalten eines komplexen Objekts durch das Verhalten seiner Bausteine beschrieben. Die Reaktionen zwischen diesen Bausteinen werden durch einfache mathematische Terme (z.B. Newton-Mechanik) gegeben. Die Berechnung des komplexen Objekts erfolgt durch die Ermittlung des Zusammenspiels seiner Partikel, also der Auswertung vieler einfacher Terme, statt der Lösung einiger recht komplexer Gleichungen. Dies ermöglicht die Implementierung auf Parallel-Rechnern; bei den erforderlichen hohen Rechenleistungen sicherlich äußerst interessant. Für die Implementierung bietet sich die objekt-orientierte Programmierung an, die durch Datenkapselung und Vererbung die Wiederverwendung von Code und die Vereinfachung von Kontrollmechanismen ermöglicht.

2 Particle-based-Modelling

Diese Technik dient zur Simulation unregelmäßiger Objekte, wie Wasser, Wolken, Dampf oder Feuer. Auch zur Animation komplexer Systeme, wie Vogel- und Fischschwärme kann sie verwendet werden [REY87]. Unregelmäßige Körper oder Substanzen werden aus diskreten kleinen Teilchen, den *Partikeln*, zusammengesetzt. Bei oben erwähnten Schwärmen spricht man von *Subobject Systems*, da jeder Partikel dabei einen ganzen Vogel, Fisch o.ä repräsentiert. Eine Partikel-Klasse enthält einige Parameter, die das Verhalten eines Teilchens sowie mögliche Wechselwirkungen mit anderen Teilchen, beruhend auf Potentialen, Entropie, Spin oder ähnlichem, angeben.

Die meisten Attribute der Partikel sind in der Regel dynamisch, d.h. ein Partikel kann seine Farbe, Form oder Geschwindigkeit während der Simulation in gewissen Grenzen ändern. Ein weiteres wichtiges Attribut eines Partikels ist seine Lebensdauer. Sie verleiht dem Partikelsystem eine große Dynamik. Sie wird bei der Erstellung von Filmsequenzen (Clips) in Einzelbildern, den *Frames*, angegeben. Mit jedem neu zu berechnenden Frame wird die Lebenszeit aller beteiligten Partikel um eins dekrementiert. Desweiteren können auch andere Vorgänge, wie z.B. das Verlassen eines gewissen Sichtbarkeitsbereichs, die Lebenszeit eines Partikels vorzeitig beenden.

Eine oft benutzte Methode zur Initialisierung eines Partikelsystems, ist die Angabe von Grundwerten (mean values) und der maximalen Abweichungen (variations) von Attributen. So läßt sich z.B. die Anfangsgeschwindigkeit eines Partikels berechnen durch

$$\text{IntitialSpeed} = \text{MeanSpeed} + \text{Rand}() * \text{VarSpeed},$$

wobei Rand() einen "zufälligen" Wert aus $[-1, 1]$ liefert.

Die Form (Geometry) eines Partikels ist in der Regel sehr einfach (Punkt), kann aber auch komplexer (Vogel im Schwarm) sein. Ist sie einfach, wird nicht nur die Berechnung eines Objekts, sondern auch die Anwendung von Spezialeffekten beschleunigt. Ein wichtiger Spezialeffekt in der Animation ist das *Motion-Blurring*, das bei Objekten, die sich schnell bewegen, auftritt. Erzeugt man z.B. einen Clip mit 24 Bildern/s, so beträgt die Linsenverschlußzeit (Shutter) bei einer herkömmlichen Filmkamera ca.1/50s. Das bedeutet, daß die gesamte Bewegung, die das Objekt während dieser Zeit durchführt in *einem* Bild festgehalten wird - das Objekt verschmiert (blur). Das Blurring eines Punktes kann man als Flugkurve beschreiben. Bei normalem Shutter reicht dabei in der Regel die Approximation durch die Strecke zwischen Anfangs- und Endpunkt vollkommen aus. [REE83]

3 Steuerungsmethoden

Nachdem das Prinzip eines Partikelsystems erläutert worden ist, beschäftigen wir uns in diesem Abschnitt mit der Frage, wie man die Partikel in einem System steuern kann. Vorgestellt werden die *Fast Multipole Methode* und das *Simulated Annealing.*

3.1 Die Fast Multipole Methode (FMM)

Kraftfelder in Partikelsystemen, die elektromagnetischen, gravitationsbedingten o.ä Ursprungs sind, lassen sich mit der Fast Multipole Methode berechnen. Diese basiert auf einem $O(n)$ Algorithmus, der die Wechselwirkungen in einem Multikörpersystem berechnet und kontrolliert. Idee ist dabei, die Partikel durch eine Unterteilung des Simulationsraumes zu gruppieren.

3.1.1 Potentialfelder

Ist eine Ladung im Ursprung gegeben, dann läßt sich für jeden Punkt $p = (x, y, z) \in \mathbb{R}^3$ mit $\|P\| = r \neq 0$ das Potential bestimmen als $\Phi = \frac{1}{r}$. Befindet sich die Ladung nicht im Ursprung, sondern im Punkt q, dann ist das Potential natürlich der Kehrwert des Abstands $d(p, q) =: r$. Um eine Reihenentwicklung für das Potential in der Abstandsvariablen zu erhalten betrachtet man im folgenden Kugelkoordinaten, wobei $p = (r, \theta, \phi)$ und $q = (\rho, \alpha, \beta)$.

Bezeichne γ den Winkel zwischen den Vektoren p und q, dann ergibt sich aus dem Cosinus-Satz

$$r'^2 = r^2 + \rho^2 - 2r\rho\cos\gamma \tag{1}$$

Das Potential ist dann

$$\frac{1}{r'} = \frac{1}{r\sqrt{1 - 2u\mu + \mu^2}} \tag{2}$$

wobei

$$\mu = \frac{\rho}{r} \quad \text{und} \quad u = \cos\gamma \tag{3}$$

Für $\mu < 1 \Leftrightarrow r > \rho$ kann kann der Wurzelkehrwert in eine Potenzreihe in μ entwickelt werden:

$$\frac{1}{\sqrt{1 - 2u\mu + \mu^2}} = \sum_{n=0}^{\infty} P_n(u)\mu^n \tag{4}$$

$P_n(u)$ bezeichne dabei das Legendre-Polynom vom Grad n. Daraus ergibt sich folgender Ausdruck für das Potential

$$\frac{1}{r'} = \sum_{n=0}^{\infty} \frac{\rho^n}{r^{n+1}} P_n(u) \tag{5}$$

Gleichung 5 wird häufig als *Multipole Erweiterung* bezeichnet und beschreibt das *Far Field* bzgl. der Ladung q, da sie nur für $r > \rho$ gilt. Vertauscht man q und p, so wird das Potential in p auch durch $1/r'$ beschrieben:

$$\frac{1}{r'} = \frac{1}{\rho\sqrt{1 - 2\frac{r}{\rho}\cos\gamma + \frac{r^2}{\rho^2}}} = \sum_{n=0}^{\infty} \frac{r^n}{\rho^{n+1}} P_n(u) \tag{6}$$

Gleichung 6 gilt nur in der offenen Kugel mit Mittelpunkt im Ursprung und Radius ρ. Daher bezeichnet man sie als *Lokale Erweiterung*.

3.1.2 Kugelfunktionen

Um Restriktionen bzgl. der Verteilung der Ladungen zu umgehen, wählt man eine allgemeine Erweiterung, die das Potentialfeld im 3D beschreibt, die Laplace-Gleichung

$$\nabla^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = 0. \tag{7}$$

deren Lösung durch folgende Reihenentwicklung gegeben ist:

$$\Phi = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left(L_n^m r^n + \frac{M_n^m}{r^{n+1}} \right) Y_n^m(\theta, \phi) \tag{8}$$

Die Terme $Y_n^m(\theta,\phi)r^n$ in Gleichung 8 sind *Kugelfunktionen* oder auch *Spherical Harmonics* vom Grad n, die Koeffizienten L_n^m und M_n^m bezeichnet man als *Momente*.

Die Multipole Erweiterung einer Gruppe von Teilchen mit gegebenen Koordinaten $Q_i = (\rho_i, \alpha_i, \beta_i)$ und Ladungen q_i läßt sich nun folgendermaßen schreiben:

$$\mathcal{M}_u^v = \sum_{j=0}^{\infty} q_i \rho_i^u Y_u^{-v}(\alpha_i, \beta_i), \qquad u, v = 0, 1, 2, \ldots$$

Der far effect der Gruppe berechnet sich folgendermaßen:

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \frac{\mathcal{M}_j^k}{r^{j+1}} Y_j^k(\theta, \phi)$$

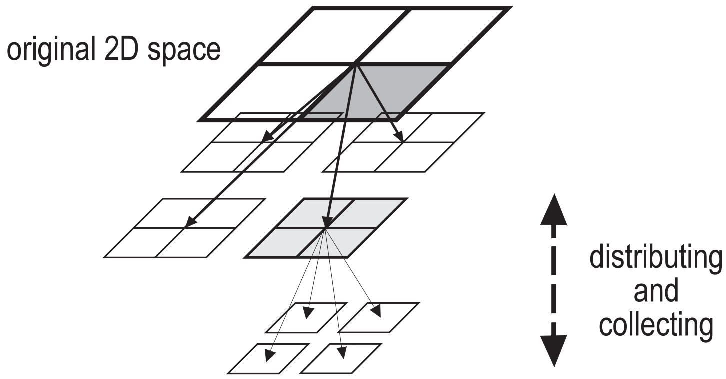

Abbildung 1: Die zwei Phasen der FMM

Analog berechnet man mit den lokalen Erweiterungen $\mathcal{L}$ den near effect von Partikeln einer Gruppe:

$$\Phi(P) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} \mathcal{L}^k Y_j^k(\theta, \phi) r^j$$

wobei

$$\mathcal{L}_u^v = \sum_{n=0}^{u} \sum_{m=-n}^{n} \frac{\mathcal{M}_n^m \mathcal{G}_{u,v}^m \mathcal{A}_n^m \mathcal{A}_u^v Y_{u+n}^{m-v}(\alpha, \beta)}{\mathcal{A}_{u+n}^{m-v} \rho^{u+n+1}}$$

dabei seien

$$\mathcal{G}_{i,h}^k = \begin{cases} (-1)^i (-1)^{\min(|h|,|k|} & hk > 0 \\ (-1)^i & \text{sonst} \end{cases}$$

und

$$\mathcal{A}_j^k = \frac{(-1)^j}{\sqrt{(j-k)!(j+k)!}}$$

3.1.3 Der Greengard-Algorithmus

Wie bereits erwähnt besteht die zentrale Idee des Algorithmus in der Gruppierung der Partikel. Die dazu verwendete Datenstruktur ist die *Greengard Hierarchie*, welche aus einem Octree zur Gruppierung der Partikel, und den jeweiligen Multipole- und lokalen Erweiterungen, die in den Knoten gespeichert werden, besteht. Zunächst wird der Simulationsraum so weit unterteilt, bis sich in jedem Teil nur noch eine gewisse Anzahl von Partikeln befindet

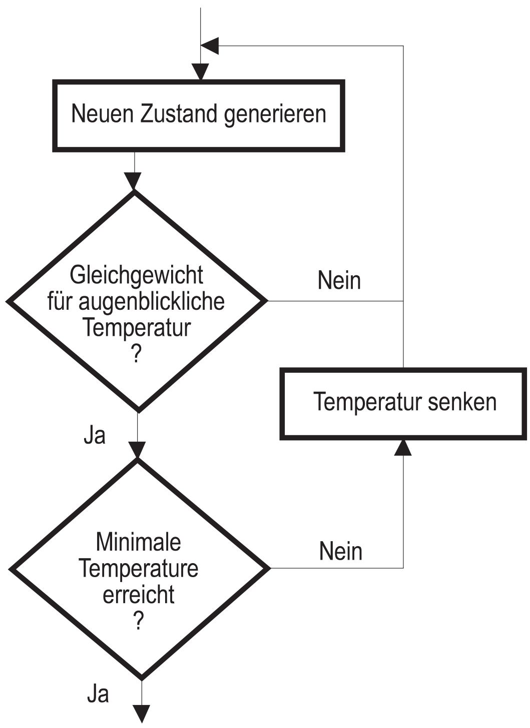

Abbildung 2: Ablauf des Simulated Annealing

(s. Abb.1). Diese Unterteilung wird durch den Octree repräsentiert. In der 1.Phase der FMM (Distributing) traversiert man dann den Octree von den Blättern zur Wurzel und berechnet in jedem Knoten die Multipole Erweiterungen. In der 2.Phase (Collecting) durchläuft man den Baum von der Wurzel bis zu den Blättern und berechnet dabei in jedem Knoten dessen lokale Erweiterung; dabei werden alle Multipole Erweiterungen mit einer gewissen Entfernung in lokale Erweiterungen transformiert. Detailliertere Informationen zur Transformation von Erweiterungen und zum Greengard-Algorithmus findet man in [G87]

3.2 Simulated Annealing

Diese Methode simuliert den Prozeß, der nach dem Schmelzen eines Festkörpers beim darauffolgenden Auskühlen stattfindet. Das flüssige Material hat eine bestimmte Partikelstruktur. Wird nun die Umgebungstemperatur gesenkt, versuchen die Teilchen, sich im energieärmsten Zustand anzuordnen. Dieser Ablauf läßt sich mit Partikelsystemen simulieren, wobei das Material nicht als Solid, sondern als Menge seiner Partikel dargestellt wird. Jeder

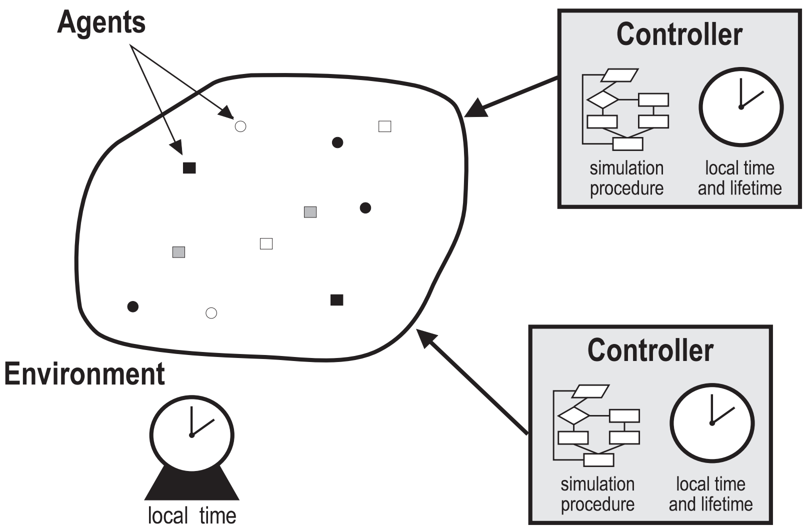

Abbildung 3: Agents, Environment und Controller

dieser Partikel hat eine bestimmte, temperaturabhängige, Bindung an die anderen Partikel. Zur Simulation des Abkühlprozesses werden nun die Effekte dieser Bindungen ausgewertet. Treten keine, oder nur Änderungen kleiner als eine gewisse Schranke im Partikelsystem auf, so ist der energieärmste Zustand für diese Temperatur erreicht; die Temperatur kann bis zur angestrebten Minimaltemperatur schrittweise weiter abgesenkt werden (s.Abb.2).

Simulated Annealing eignet sicht grundsätzlich gut zur Simulation von Optimierungsprozessen, da bei diesem Verfahren nicht die Gefahr besteht, in lokalen Extremstellen stecken zu bleiben.

4 Simulationskonzept

In diesem Abschnitt wird ein Konzept zur Simulation von Partikelsystemen vorgestellt, das aus drei wichtigen Komponenten (s.Abb.3) besteht:

- Simulationsobjekte, auch *Agents* genannt; in unserem Fall also die Repäsentanten der Partikel.
- Umgebungen, oder auch *Environments*, die Objekte gruppieren und beeinflussen.
- Steuerungen, auch *Controller* genannt, die Bewegung, Verhalten der Simulationsobjekte bestimmen.

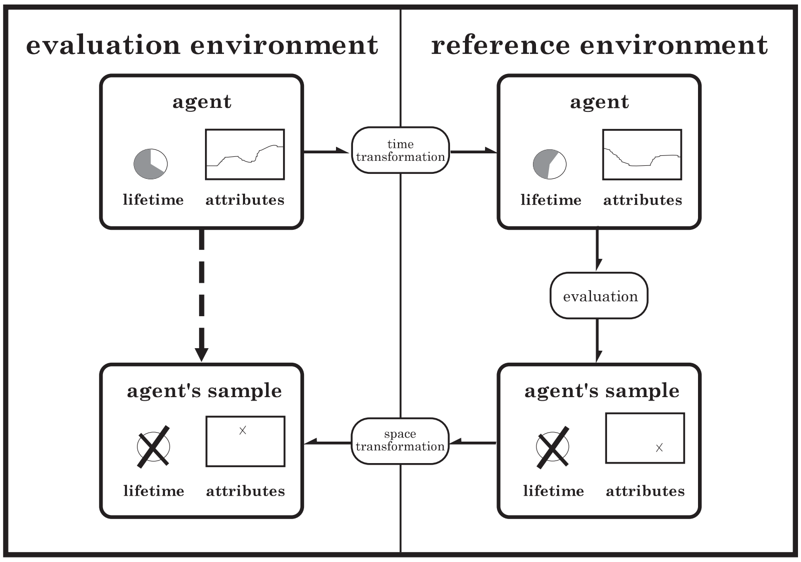

Abbildung 4: Berechnungsprozess eines Agents

Ein Agent repräsentiert einen diskreten Simulationskörper. Dabei beinhaltet eine Agent-Klasse das Aussehen (Geometry) und das Verhalten des Körpers sowie einige auf der Applikation basierende Attribute. Beschreibt eine Agent Klasse beispielsweise einen Wasserpartikel in einer Wolke, so kann man als Aussehen eine einfache Form, wie den Punkt, benutzen.

In Environments werden ein oder mehrere Agents zusammengefaßt. Damit unterliegen sie alle den gleichen externen Einflüssen und dem selben Timing, denn ein Environment besitzt immer eine eigene Zeituhr (local time). Ein Agent, oder besser dessen Attribute wie Masse, Geschwindigkeit, Ladung oder Positon, werden durch ein environmentspezifisches Verhalten, welches von der aktuellen local time abhängt, berechnet. (s. Abb.4). Der Vorteil, den ein Environment bietet, liegt also zum einen in der Vereinfachung der Strukturierung von Agents, zum anderen bleibt der Agent unabhängig von der Applikation, was eine hohe Flexibilität bezüglich der Wiederverwertbarkeit bedeutet. Die Knoten des Octrees im Greengard-Algorithmus (s.3.1.3) kann man z.B als Environments implementieren.

Der Controller schließlich bestimmt, wie die Berechnungen durchgeführt werden. Ob also durch FMM, Simulated Annealing, Finite-Element-Methode oder Applikations-Skripts. In einer Applikation können durchaus mehrere Controller ein und dasselbe Environment steuern. Das bedeutet, daß Teilchen unterschiedlichen Einflüssen ausgesetzt sind und auch unterschiedliche Einflüsse ausüben können.

Um komplexere Partikel, z.B. Vögel in einem Schwarm, zu simulieren, setzt man einen Vogel wiederum aus seinen Grundbausteinen (Rumpf, Kopf, Flügel und Schwanz) in einem Environment zusammen und behandelt diesen als einen speziellen Agent. Man erreicht damit, daß ein Controller nicht die Agents des Environments selbst steuert, sondern das gesamte Environment. Dies ist sinnvoll, da im Beispiel »Vogelschwarm« der Vogel dem Schwarmverhalten unterliegt, nicht aber sein Flügel. Diese Kapselung ist also dann vorteilhaft, wenn man weiß, daß alle Agents in einem Environment von einem Simulationsschritt eines Controllers nicht direkt beeinflußt oder verändert werden. Oft kann man so unnötigen Rechenaufwand vermeiden.

Ein anderes, ebenfalls objektorientiertes Konzept wird in [REY87] vorgestellt. Es handelt sich dabei um das *Actor Model*. Jedes "Individuum" in einem Vogelschwarm wird von einer weiteren Instanz des Vogel-Objekts repräsentiert. Dabei werden Ablauf, Berechnung und Status eines Individuums in einem Objekt, dem sogenannten Actor, zusammengefaßt. Der anfallende Speicherbedarf ist wesentlich höher als beim Agent-Environment-Controller Modell, bietet sich bei Schwärmen und Herden jedoch deswegen an, weil zum einen die Zahl der Objekte viel geringer ist und zum zweiten jedem Individuum eine gewisse Autonomie erlaubt wird. Jeder Actor ist eine Art virtueller Rechner, der mit den anderen Actors Nachrichten (Messages) austauscht.

5 Anwendungsbeispiel: Air Quality Control

Ein Gebiet, in dem Multikörper-Systeme zur Anwendung kommen, ist die Simulation von Schadstoffemissionen durch Fabriken, Kraftwerke und Verkehrsmittel und der Ausbreitung dieser Schadstoffe in der Luft. Die Schadstoffe werden als diskrete Partikel in einem dreidimensionalen Raum dargestellt und ihre Bewegung durch lokale (micro-scaled) und globale (mesoscaled) Windmodelle gesteuert. Durch die Windmodelle werden die jeweiligen *Windfelder* berechnet:

5.1 Lokales Windfeld

Ein lokales Windfeld ist ein diskretes Vektorfeld. Es wird für Volumina mit einer Grundläche von $100 - 5000m^2$ und einer Höhe von $50 - 300m$ berechnet. Dabei liegt der Gitterabstand bei ca.$5m$. Die Berechnung des lokalen *Windfeldes* benötigt zwei Eingabemengen: 1. Die metereologischen Daten, wie die globale Windgeschwindigkeit und -richtung sowie Turbulenzzonen in der Umgebung von Gebäuden. 2. Die aerodynamischen Daten, wie Anordnung, Form, Oberflächenstruktur, Material etc. von Objekten, Häusern o.ä.

5.2 Globales Windfeld

Die Berechnung des globalen Windfeldes benötigt ebenfalls zwei Eingabemengen: 1. Die durchschnittliche Höhe des Gebiets. 2. Der prozentuale Anteil der fünf Bodenarten: Argrar, Wald, Wasser, sowie schwache und starke Bebauung, wobei jeder Bodenart ein gewisser Rauhheitswert zukommt, der für eine Verminderung der Windgeschwindigkeit sorgt. Weitere benötigte Eingabedaten sind der geostrophische Wind (Luftdruckunterschiede in großen Höhen unabhängig von der Corioliskraft), das Temperaturprofil über dem Simulationsgebiet und die Temperaturänderung während der Simulation am Boden des Simulationsgebietes.

Beide Windfelder, das lokale und das globale werden durch Environments repräsentiert. Wie die Partikel sich nun in solchen Windfeldern verhalten, bestimmen die verwendeten Controller.

5.3 Ausblick

In einer Simulationsanwendung wie der Air Quality Control wird zur Zeit mit ca. 10^5 Partikeln gerechnet, eine Erweiterung auf bis zu 10^{10} Partikel wird angestrebt. Außerdem sollen in Zukunft zusätzlich auch chemische Prozesse zwischen Partikeln simuliert werden. Die ständige Erweiterung der Komplexität von solchen Simulationssystemen, die eine immer realistischere Wiedergabe natürlicher Vorgänge ermöglicht, ist mit immer größer werdenden Anforderungen an die Rechenleistung verbunden. Hier setzt man auf den massiven Einsatz von großen Parallelrechnern. Die Ausgabe erfolgt auf Low-Cost Workstations, die über schnelle Datenleitungen mit den Großrechnern verbunden sind.

Literatur

[KM93] V.Kühn, W.Müller, *Advanced Object-oriented Methods and Concepts for Simulations of Multi-Body Systems*, Journal of Visualization and Computer Animation, Vol.4: 95-111, 1993.

[G87] L.Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, 1987.

[REE83] W.T.Reeves, *Particle Systems - A Technique for Modeling a Class of Fuzzy Objects*, Computer Graphics, 17, (3), 359-376, 1983.

[REY87] C.W.Reynolds, *Flocks, Herds, and Schools: A Distributed Behavioral Model*, Computer Graphics, 21, (4), 25-34, 1987.