Ausarbeitung zum Vortrag

Simulation von Festkörpern mit orientierten Partikeln

Michael Wolff

5. Februar 2014

basierend auf dem Paper
„Solid Simulation with Oriented Particles"
von Matthias Müller & Nuttapong Chentanez (2012)

Inhaltsverzeichnis

1 Einführung

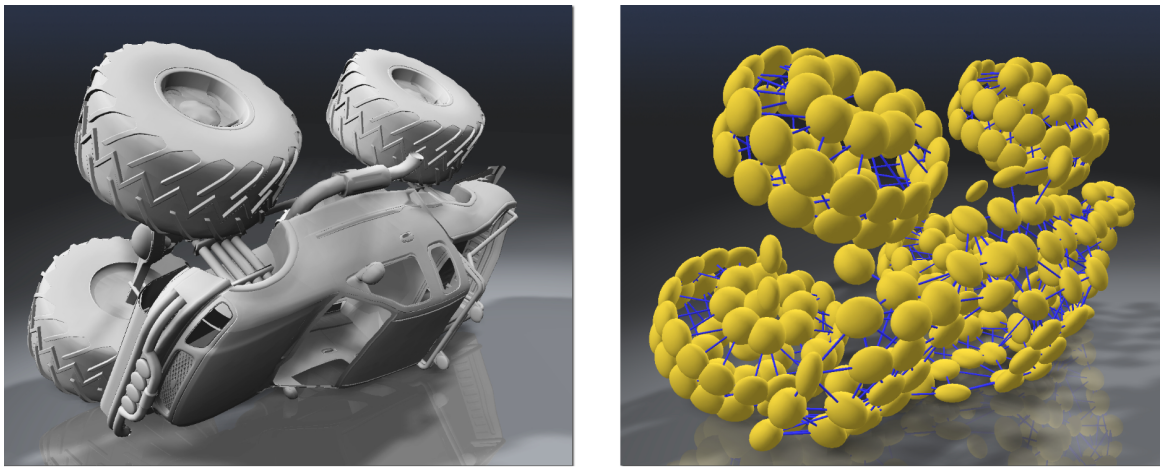

Abb. 1.1: Simulation eines Monstertrucks
Links: Visuelle Repräsentation
Rechts: Physikalische Repräsentation mit orientierten Partikeln

Bei der computerbasierten Modellierung und Simulation von Objekten wird ein möglichst hoher Grad an Realismus verlangt. Wenn es um medizinische Anwendungen geht, ist die maximale Detailliertheit der Darstellung wichtig. In Computerspielen hingegen ist vor allem der Gesamteindruck und die gute Berechenbarkeit entscheidend. Zudem sollte es möglich sein, Eigenschaften und Verhalten der Objekte leicht durch den Entwickler zu manipulieren. Mit dem Verzicht auf eine besonders hohe Genauigkeit der Modelle, welche zwangsläufig mit einer hohen Komplexität einherginge, ist es möglich, eine relativ dünne physikalische Repräsentation von Objekten zu schaffen (siehe Abb. 1.1). Ein solches Modell birgt klare Vorteile in Sachen Laufzeiteffizienz und bietet dennoch einen ausreichend guten visuellen Gesamteindruck. Es ist für den Entwickler außerdem gut möglich, physikalische Eigenschaften der Modelle wie Festigkeit oder Biegsamkeit einfach zu beeinflussen.

Die vorgestellte Methode basiert auf Verallgemeinerungen des *Shape Matching* (SM) [2] und *Position Based Dynamics* (PBD) [3]. Die Idee ist die Verwendung orientierter Partikel in Form von Ellipsoiden. Die zusätzliche Information der Partikel ermöglicht nicht nur eine sehr dünne physikalische Repräsentation von Objekten, sondern bietet noch andere

Vorteile wie die Tatsache, dass Partikel anisotropischer Form, wie Ellipsoide, Oberflächen besser approximieren als Sphären.

2 Verallgemeinertes Shape Matching (SM)

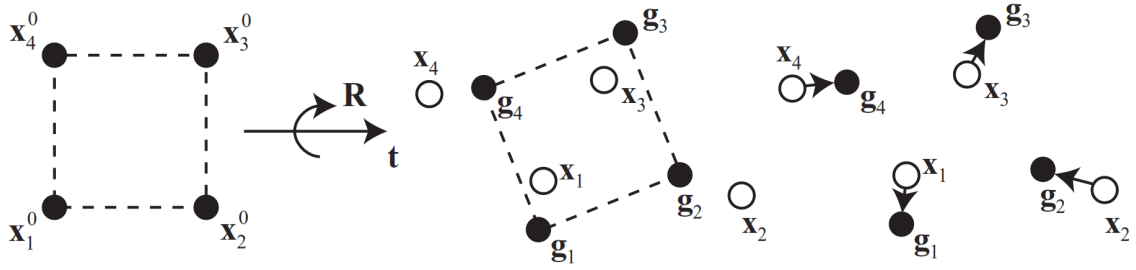

Abb. 2.1: Visualisierung des Shape Matching-Algorithmus
1. Passe Originalform $\mathbf{x}_i^0$ an deformierte Form $\mathbf{x}_i$ an
2. Verschiebe deformierte Punkte $\mathbf{x}_i$ in Richtung der angepassten Form $\mathbf{g}_i$

Abb. 2.1 visualisiert den normalen SM-Algorithmus. Den Input stellt eine Menge von Partikeln und deren Startposition $\mathbf{x}_i^0$ dar. Die gestrichelten Linien skizzieren die Form der Partikelwolke, allerdings dient diese Form nur der Veranschaulichung. Tatsächlich werden keinerlei echte Verbindungen benötigt. Nach jedem Zeitschritt wird jeder Partikel in Richtung seiner Zielposition $\mathbf{g}_i$ verschoben. Diese Zielpositionen resultieren aus der Anpassung der initialen Form, spezifiziert durch $\mathbf{x}_i^0$, an die aktuelle Konfiguration der Partikelwolke, gegeben durch $\mathbf{x}_i$.

SM beschreibt also, wie sich Objekte deformieren und bewegen. Für die Beschreibung der Transformationen wird zum einen die globale Translation $\mathbf{t}$ und die Rotation $\mathbf{R}$, welche die aktuellen Positionen der Partikel optimal annähern, benötigt. Gegeben n Partikel mit ihren Ruhepositionen $\overline{\mathbf{x}}_i$, aktuellen Positionen $\mathbf{x}_i$ sowie Masse m_i muss zunächst Matrix $\mathbf{A}$ berechnet werden ...

$$\mathbf{A} = \sum_i m_i(\mathbf{x}_i - \mathbf{c})(\overline{\mathbf{x}}_i - \overline{\mathbf{c}})^T \in \mathbb{R}^{3\times 3} \tag{2.1}$$

... wobei die Massezentren wie folgt definiert sind:

$$\mathbf{c} = \sum_i m_i \mathbf{x}_i / \sum_i m_i \tag{2.2}$$

$$\overline{\mathbf{c}} = \sum_i m_i \overline{\mathbf{x}}_i / \sum_i m_i \tag{2.3}$$

Die Polarzerlegung $\mathbf{A} = \mathbf{RS}$ liefert die optimale Rotation und für die Translation gilt $\mathbf{t} = \mathbf{c} - \bar{\mathbf{c}}$, woraus sich die Zielposition für jeden Partikel berechnet:

$$\mathbf{g}_i = \mathbf{R}(\bar{\mathbf{x}}_i - \bar{\mathbf{c}}) + \mathbf{c} \tag{2.4}$$

Problematisch wird es allerdings, wenn die Partikel nahezu koplanar bzw. kolinear sind. In dem Fall ist $\mathbf{A}$ schlecht konditioniert oder sogar singulär, was dazu führt, dass $\mathbf{R}$ nicht wohldefiniert ist. Bei diesem Problem helfen die neuen Orientierungsinformationen der Partikel.

Angenommen man hat zwei Mengen von Partikeln mit ihren Matrizen $\mathbf{A}_1$, $\mathbf{A}_2$. Für die Vereinigung der beiden Mengen können die Matrizen nicht einfach addiert werden, da jede Partikelmenge sein eigenes Massezentrum besitzt. Entsprechend muss Gleichung 2.1 umformuliert werden:

$$\begin{aligned} \mathbf{A} &= \sum_i m_i \mathbf{x}_i \bar{\mathbf{x}}_i^T - M\mathbf{c}\bar{\mathbf{c}}^T \\ &\text{mit } M = \sum_i m_i \end{aligned} \tag{2.5}$$

Im nächsten Schritt wird für einen einzelnen Partikel die Matrix $\mathbf{A}_\mathrm{i}$ definiert. Mittels Gleichung 2.5 wird nun das Massezentrum jedes $\mathbf{A}_\mathrm{i}$ zu einem Globalen verschoben ...

$$\mathbf{A} = \sum_i (\mathbf{A}_i + m_i \mathbf{x}_i \bar{\mathbf{x}}_i^T - m_i \mathbf{c}\bar{\mathbf{c}}^T) \tag{2.6}$$

... und Gleichung 2.1 verallgemeinert zu:

$$\boxed{\mathbf{A} = \sum_i (\mathbf{A}_i + m_i \mathbf{x}_i \bar{\mathbf{x}}_i^T) - M\mathbf{c}\bar{\mathbf{c}}^T} \tag{2.7}$$

Für die Berechnung der Matrizen einzelner Partikel $\mathbf{A}_\mathrm{i}$, wie oben erwähnt, wird Gleichung 2.1 über das Volumen des Partikel integriert ...

$$\begin{aligned} \mathbf{A}_\text{sphere} = ... &= \frac{1}{5} m r^2 \mathbf{R} \\ &\text{mit Radius } r \end{aligned} \tag{2.8}$$

$$\begin{aligned} \mathbf{A}_\text{ellipsoid} &= \frac{1}{5} m \begin{bmatrix} a^2 & 0 & 0 \\ 0 & b^2 & 0 \\ 0 & 0 & c^2 \end{bmatrix} \mathbf{R} \\ &\text{mit Radien } a, b, c \end{aligned} \tag{2.9}$$

Dadurch ist Matrix $\mathbf{A}$ stets vollen Ranges, auch für einzelne Partikel.

3 Verallgemeinerte positionsbasierte Dynamiken (PBD)

Nachdem SM die räumliche Entwicklung der Partikel beschreibt, geht es bei PBD um die zeitliche Entwicklung, also die zeitliche Abfolge der Transformationen.

Der Ablauf der normalen PBD gliedert sich in drei Phasen pro Zeitschritt. In der Prognosephase werden die voraussichtlichen Positionen $\mathbf{x}_\mathrm{p}$ aller Partikel mittels explizitem Euler-Verfahren berechnet:

$$\mathbf{x}_p \leftarrow \mathbf{x} + \mathbf{v}\Delta t \tag{3.1}$$

Die Korrekturphase dient der Korrektur der prognostizierten Positionen durch den *Solver*, sodass diese bestimmte Restriktionen, wie eine maximale Distanz innerhalb der Partikelkonfiguration, erfüllen. In der Integrationsphase werden schließlich die Zustandsvariablen der Partikel aktualisiert, die tatsächliche Transformation also ausgeführt:

$$\begin{aligned} \mathbf{v} &\leftarrow (\mathbf{x}_p - \mathbf{x})/\Delta t \\ \mathbf{x} &\leftarrow \mathbf{x}_p \end{aligned} \tag{3.2}$$

$\mathbf{v}$ ist die Geschwindigkeit eines Partikels.

3.1 Integration

Aufgrund der zusätzlichen Informationen durch die Orientiertheit der Partikel kommen zwei neue Variablen hinzu: das Orientierungsquaternion $\mathbf{q}$ und die Rotationsgeschwindigkeit ω. Ein Quaternion ist aus einem Zahlenbereich, der die reellen Zahlen, ähnlich wie die komplexen Zahlen, erweitert und der Beschreibung von Rotationen dient:

$$\mathbf{q} = \mathbf{x}_0 + \mathbf{x}_1 i + \mathbf{x}_2 j + \mathbf{x}_3 k \tag{3.3}$$

Der Imaginärteil kann als 3-dimensionaler Vektor gesehen werden und beschreibt zusammen mit der Position $\mathbf{x}$ eine Rotationsachse. $\mathbf{x}_0$ entspricht dem Rotationswinkel.

Durch diese neuen Variablen wird sowohl die Prognosephase ...

$$\begin{aligned} \mathbf{x}_p &\leftarrow \mathbf{x} + \mathbf{v}\Delta t \\ \boxed{\mathbf{q}_p \leftarrow \left[\frac{\omega}{|\omega|}\sin(\tfrac{|\omega|\Delta t}{2}), \cos(\tfrac{|\omega|\Delta t}{2})\right]\mathbf{q}} & \\ \text{mit} & \\ \mathbf{q}_p \leftarrow \mathbf{q} \text{ für } |\omega| < \epsilon & \end{aligned} \tag{3.4}$$

... als auch die Integrationsphase erweitert ...

$$\begin{aligned} \mathbf{v} &\leftarrow (\mathbf{x}_p - \mathbf{x})/\Delta t \\ \mathbf{x} &\leftarrow \mathbf{x}_p \\ \boxed{\omega \leftarrow \text{axis}(\mathbf{q}_p\mathbf{q}^{-1}) \cdot \text{angle}(\mathbf{q}_p\mathbf{q}^{-1})/\Delta t} & \\ \boxed{\mathbf{q} \leftarrow \mathbf{q}_p} & \\ \text{mit} & \\ \text{axis()} \rightarrow \text{normierte Richtung eines Quaternion} & \\ \text{angle()} \rightarrow \text{Winkel eines Quaternion} & \end{aligned} \tag{3.5}$$

3.2 Reibung

Kollidiert ein Partikel mit einer Wand oder einem anderen Partikel, entsteht Reibung. Sie wird verrechnet, indem nach der Integrationsphase die lineare Geschwindigkeit und die Rotationsgeschwindigkeit um einen konstanten Faktor s runterskaliert wird:

$$\begin{aligned} \mathbf{v} &\leftarrow \mathbf{v} + (\mathbf{v}_s - \mathbf{v})\perp_{\mathbf{n}} \cdot s_{lin} \\ \omega &\leftarrow \omega + \frac{\mathbf{r}}{|\mathbf{r}|^2} \times (\mathbf{v}_s - \mathbf{v} - \omega \times \mathbf{r}) \cdot s_{rot} \\ \text{mit} & \\ \mathbf{v}_s \rightarrow \text{Geschwindigkeit des Objekts am Kollisionspunkt} & \\ \mathbf{n} \rightarrow \text{Normale des Objekts am Kollisionspunkt} & \\ s_{lin} \in [0...1], s_{rot} \in [0...1] & \\ \mathbf{r} = r\mathbf{n} & \end{aligned} \tag{3.6}$$

Für die Kollision zweier Partikel sind die Gleichungen analog:

$$\begin{aligned}
\mathbf{v}_1 &\leftarrow \mathbf{v}_1 + \left(\frac{\mathbf{v}_1 + \mathbf{v}_2}{2} - \mathbf{v}_1\right)\perp_{\mathbf{n}} \cdot s_{lin} \\
\mathbf{v}_2 &\leftarrow \mathbf{v}_2 + \left(\frac{\mathbf{v}_1 + \mathbf{v}_2}{2} - \mathbf{v}_2\right)\perp_{\mathbf{n}} \cdot s_{lin} \\
\omega_1 &\leftarrow \omega_1 + \frac{\mathbf{r}_1}{|\mathbf{r}_1|^2} \times \left(\mathbf{v}_{avg} - \mathbf{v}_1 - \omega_1 \times \mathbf{r}_1\right) \cdot s_{rot} \\
\omega_2 &\leftarrow \omega_2 + \frac{\mathbf{r}_2}{|\mathbf{r}_2|^2} \times \left(\mathbf{v}_{avg} - \mathbf{v}_2 - \omega_2 \times \mathbf{r}_2\right) \cdot s_{rot} \\
&\text{mit} \\
\mathbf{n} &= (\mathbf{x}_2 - \mathbf{x}_1)/|\mathbf{x}_2 - \mathbf{x}_1| \\
\mathbf{r}_1 &= r\mathbf{n}, \mathbf{r}_2 = -r\mathbf{n} \\
\mathbf{v}_{avg} &= (\mathbf{v}_1 + \omega_1 \times \mathbf{r}_1 + \mathbf{v}_2 + \omega_2 \times \mathbf{r}_2)/2
\end{aligned} \tag{3.7}$$

4 Simulationsmodell

4.1 Implizites Shape Matching

Das implizite SM ist eine Datenstruktur, welche eine SM-Gruppe pro Partikel definiert. Eine Gruppe besteht aus dem korrespondierenden Partikel und allen Partikeln, die über eine Kante mit diesem verbunden sind. Nach der Prognosephase iteriert der *Solver* mehrfach über alle SM-Bedingungen und berechnet die Zielpositionen der Partikel. Anschließend werden alle Partikel der Gruppe in Richtung ihrer Zielpositionen um einen Anteil abhängig von $s_{\text{stiffness}}$ bewegt. $s_{\text{stiffness}}$ ist ein Faktor zur Bestimmung der Festigkeit/Starrheit der Konfiguration und kann für jeden Partikel separat spezifiziert werden. Zuletzt wird noch die Orientierung des Hauptpartikels, die sich aus der optimalen Rotation des SM ergibt, aktualisiert. Da jeder Partikel Hauptpartikel einer impliziten Gruppe ist, werden letztlich die Orientierungen aller Partikel aktualisiert. Die Tatsache, dass jeder Partikel mehreren impliziter Gruppen angehört, die Gruppen sich also gegenseitig überlappen, sorgt dafür, dass Transformationen der Gesamtkonfiguration gleichmäßig über das Objekt propagieren und ein Arm beispielsweise keine plötzlichen, unnatürlichen Knicke aufweist.

4.2 Streckung & Biegung

SM modelliert Streckung und Biegung zugleich. Bei Bedarf sind sie auch separat über Entfernungsrestriktionen von PBD spezifizierbar. Beispielsweise könnte eine Erhöhung der Biegsamkeit durch Verringerung von $s_{\text{stiffness}}$ und bestimmter Entfernungsbedingungen herbeigeführt werden.

4.3 Explizites Shape Matching

Zusätzlich zu den impliziten SM-Gruppen können durch den Nutzer auch explizite spezifiziert werden. Diese expliziten Gruppen können beliebig viele Partikel umfassen. Das implizite SM, wie zuvor beschrieben, wird für diese Gruppen nicht ausgeführt. Im Gegensatz zu den impliziten Gruppen erfahren alle Partikel, die Teil einer solchen Gruppe sind, die gleiche SM-Rotation. Dies führt zu starren Komponenten, die unabhängig vom Rest des Objekte auf sie einwirkende Kräfte reagieren wie die Räder eines Autos. Partikel kön-

nen Teil mehrerer expliziter Gruppen sein; sie gelten dann als nicht orientiert und erlauben die Modellierung von Gelenken.

4.4 Plastische Deformation

Im Kontext der expliziten Gruppen bedeutet „starre Komponente" nicht, dass diese nicht deformierbar sind. Eine Deformation einer expliziten Gruppe wird durch die Kollision einer der Gruppenpartikel mit einer Geschwindigkeit, die einen festgelegten Grenzwert überschreitet, herbeigeführt. In diesem Fall wird die explizite Gruppe kurzzeitig deaktiviert und implizites SM findet statt.

4.5 Torsion

Torsion verhält sich ähnlich wie Reibung. Für diesen Fall wird über alle Kanten iteriert und die Rotation benachbarter Partikel angepasst:

$$\begin{aligned} \mathbf{q}_1 &\leftarrow \text{slerp}(\mathbf{q}_1, \mathbf{q}_2, \frac{1}{2} s_{torsion}) \\ \mathbf{q}_2 &\leftarrow \text{slerp}(\mathbf{q}_2, \mathbf{q}_1, \frac{1}{2} s_{torsion}) \end{aligned} \tag{4.1}$$

und $\text{slerp}(\mathbf{q}_1, \mathbf{q}_2, s)$ liefert $\mathbf{q}_1$ wenn $s = 0$, $\mathbf{q}_2$ wenn $s = 1$
sonst sphärische Interpolation der beiden

5 Kollisionsbewältigung

Die Verwendung von Sphären zur Repräsentation der Partikel resultiert in einer holprigen, unebenen Beschreibung von Oberflächen, was eine unnatürliche Reibung und falsche Kollisionsbewältigung zur Folge hat. Die Orientierungsinformationen der Partikel ermöglichen die Darstellung in Form von Ellipsoiden, welche Oberflächen sehr viel besser approximieren als Sphären.

5.1 Ellipsoidkollision

Eine Partikelkollision unter der Verwendung von Sphären findet statt, wenn ein Partikel näher an einer Ebene oder einem anderen Partikel ist als sein Radius r. In diesem Fall wird der Partikel entlang der Ebenen- bzw. Sphärennormale $\mathbf{n}$ verschoben (Abb. 5.1 Links). Für Ellipsoide ist eine Kollision nicht ganz so einfach zu bewältigen. Wird der Radius des Ellipsoid in Richtung der Normalen verwendet, dann wird die Kollision nicht richtig aufgelöst (Abb. 5.1 Mitte). Lediglich für den Fall, dass die Hauptachse des Ellipsoids an der Normalen ausgerichtet ist, wäre die Kollision korrekt bewältigt. Um die Kollision für Ellipsoide stets korrekt zu bewältigen, wird die richtige Distanz d benötigt (Abb. 5.1 Rechts). Die Berechnungsvorschrift für d findet sich im Paper [1].

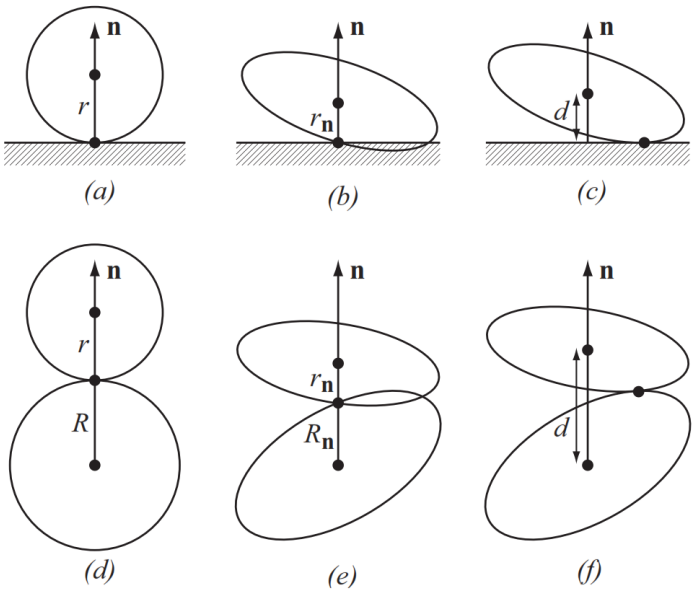

Abb. 5.1: Partikelkollision
Links: Sphären
Mitte: Ellipsoide approximiert
Rechts: Ellipsoide korrekt

5.2 Ellipsoidrepräsentation von Objekten

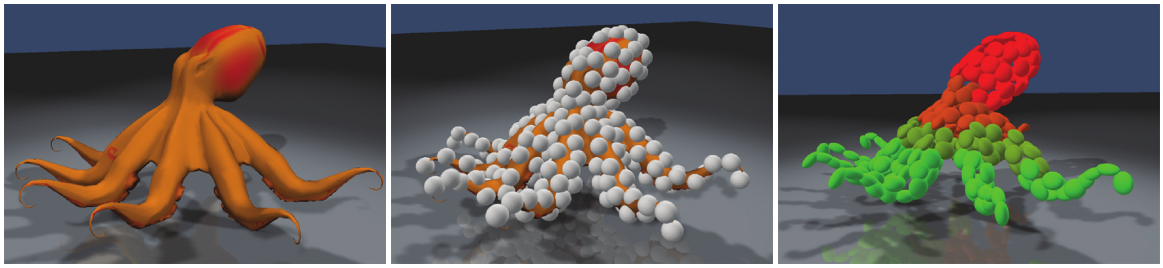

Abb. 5.2: Vom visuellen Gitternetz zur Repräsentation durch Ellipsoide
Links: Visuelles Netz
Mitte: Simulationspartikel
Rechts: Ellipsoidrepräsentation mit farblich markierter Festigkeit

Um Objekte durch Ellipsoide zu repräsentieren, müssen zunächst Simulationspartikel auf dem visuellen Gitternetz (Polygonnetz) platziert werden (Abb. 5.2 Mitte). Diese Partikel besitzen alle den gleichen initialen Radius. Im nächsten Schritt wird die Richtung der Hauptachse berechnet. Dafür werden alle Vertices, die innerhalb des Partikelradius liegen, betrachtet. Ein Vertex ist ein Knoten des visuellen Gitternetzes. Von einer solchen Vertexwolke wird die Kovarianzmatrix berechnet und mittels Polarzerlegung die Orientierung bestimmt. Zur Bestimmung der Radien werden Oriented Bounding Boxes (OBB) verwendet. Dies ist ein Rahmen, der alle betrachteten Vertices einschließt und entsprechend der zuvor bestimmten Orientierung ausgerichtet ist. Die Größe einer solchen Box ergibt dann den Radius eines Ellipsoids. Zusätzlich zur Bestimmung der Orientierungen können Partikel auch in Richtung des Massezentrums umliegender Vertices verschoben werden. Werden diese zwei Schritte mehrfach vollzogen, so bilden die Ellipsoide ein geschlossenes und stimmiges Skelett (Abb. 5.2 Rechts).

6 Skinning des visuellen Gitternetzes

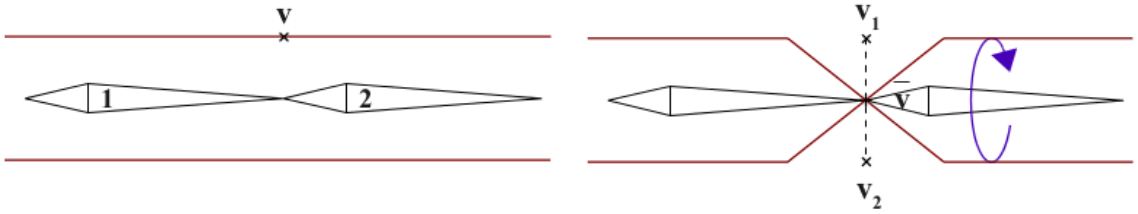

Abb. 6.1: Candywrapper-Artefakt schematisch; Lineare Interpolation von $\mathbf{v}_1$ *und* $\mathbf{v}_2$ *führt zu schlechter Transformation und Volumenverlust*

Skinning ist der Prozess, welcher beschreibt, wie das visuelle Gitternetz an das zugrundeliegende physikalische Skelett (die Ellipsoidrepräsentation) angepasst wird. Es definiert also auch sichtbare Verformungen eines Objekts für den Fall, dass sich das physikalische Skelett transformiert. Die vorgestellte Methode verwendet dafür Linear Blend Skinning (LBS). Hierbei wird jedem Vertex gewichtete Verbindungen zu umliegenden Partikeln zugewiesen. In der Regel besitzt ein Vertex lediglich ein bis zwei Verbindungen zu Partikeln. Bei einer Transformation des Skeletts wird nun jeder Vertex anhand des gewichteten Einflusses seiner verbundenen Partikel transformiert.

Die Verwendung von LBS kann allerdings in bestimmten Fällen zu visuellen Artefakten führen (siehe Abb. 6.1). Solche Artefakte kommen zustande, da beim LBS die lineare Interpolation der Positionen berechnet wird:

$$\overline{\mathbf{v}} = \sum_{j=1}^{n} w_j T_j \mathbf{v} = w_1(T_1.\mathbf{v}) + w_2(T_2.\mathbf{v}) = 0.5\mathbf{v}_1 + 0.5\mathbf{v}_2 \tag{6.1}$$

Alternativ könnte für das Skinning auch Dual Quaternion Skinning (DQS) verwendet werden. Wie bereits erwähnt, lässt sich mit einem einzelnen Quaternion eine Rotation beschreiben. Verwendet man nun Paare von Quaternionen, so lassen sich damit auch Translationen beschreiben, wodurch visuelle Artefakte, wie Candywrapper, vermieden werden. Auch hinsichtlich der Laufzeit wäre DQS überlegen.

7 Simulationsframework

Das Simulationsframework beinhaltet ein Designtool, welches dazu verwendet wird, die physikalische Repräsentation von Objekten zu erzeugen. Ausgangspunkt stellt ein visuelles Gitternetz dar, auf dem der Nutzer Simulationspartikel mit einem Spraytool aufträgt. Partikel können für größere Kontrolle auch einzeln positioniert werden. Kanten, welche die Partikel verbinden, werden ebenfalls mit einem Spraytool platziert, können aber auch wieder einzeln definiert werden. Kanten werden nur gesetzt, wenn sie eine festgelegte Distanz nicht überschreiten. Skinninggewichte, also jene Gewichte, die bei Transformationen den Einfluss der Partikel auf die Vertices bestimmen, werden automatisch festgelegt. Des Weiteren gibt es noch Tools zur Bestimmung der Torsions- und Partikelfestigkeit, zur Definierung expliziter SM-Gruppen, zur Anpassung der Skinninggewichte und für das automatische Einfügen der Ellipsoide.

8 Ergebnisse

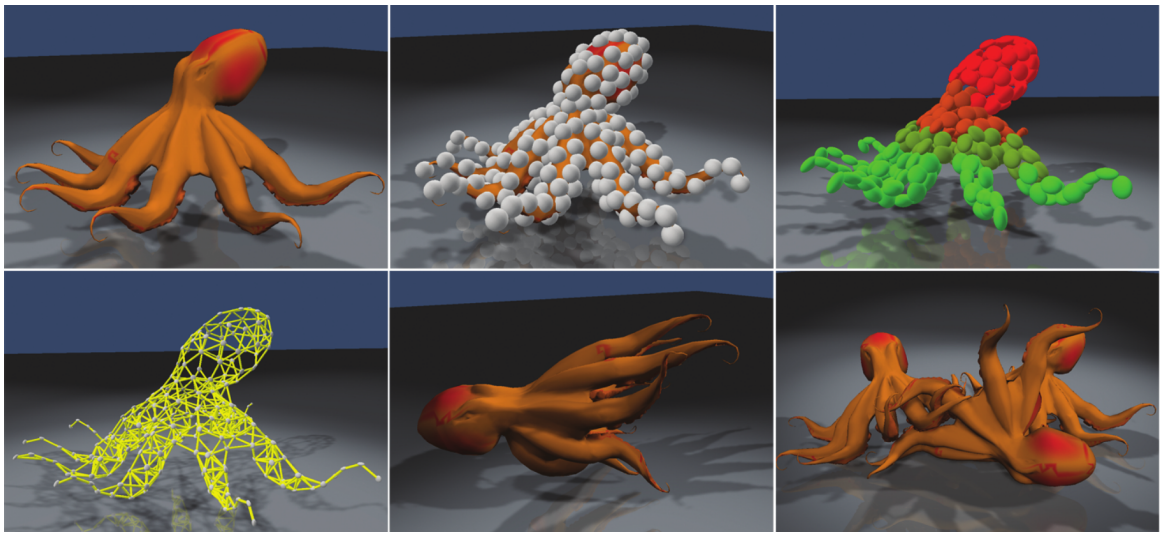

Abb. 8.1: Entstehung des physikalischen Modells
↖ Ausgangspunkt: Visuelles Netz; ↑ Platzierung der Simulationspartikel; ↗ Einfügung der Ellipsoide, Festigkeit farblich markiert
↙ Generierung der Kanten; ↓ Weiche Deformation; ↘ Interaktion & Kollision

Um die vorgestellte Methode zu präsentieren und dessen Vorteile herauszustellen, wurden einige Simulationen auf einem Kern eines Intel Core2@2.4GHz und einer NVIDIA Quadro FX 5800 ausgeführt. Alle gleich beschriebenen Simulationen können in einem Demonstrationsvideo [6] nachvollzogen werden.

Das erste Beispiel (Abb. 8.1) zeigt den Prozess der Entstehung eines physikalischen Modells unter Verwendung des vorgestellten Frameworks. Das physikalische Netz setzt sich aus 300 Partikeln und 750 Kanten zusammen. Das visuelle Netz besteht aus 5.000 Vertices und 10.000 Dreiecken. Die Simulation des Modells mit Selbstkollision braucht 4ms, das Skinning und Rendering zusammen 6ms. Die komplexe Interaktion und Kollision, welche durch den *Solver* berechnet werden, dreier solcher Objekte benötigt 10ms.

Abb. 8.2: Simulation eines Feuerfischs und 6 Planzen sowie deren Interaktion und Kollision

Das nächste Beispiel ist eine Unterwasserszene, in der ein Feuerfisch an 6 Pflanzen vorbeischwimmt und diese berührt (Abb. 8.2). Dabei ist lediglich die Bewegung des Kopfs gescriptet. Die Simulation setzt sich aus insgesamt 1.000 Partikeln und 3.500 Kanten für das physikalische Netz sowie 30.000 Vertices und 48.000 Dreiecken zusammen. Für die Interaktion und Kollisionen benötigt der *Solver* 10ms; das Skinning braucht 9ms. Diese Szene wurde für den Vergleich mit einer anderen Methode erstellt, bei der ebenfalls ein schwimmender Feuerfisch simuliert wurde. Die andere Methode basiert auf dem *Elaston*-Ansatz [7] und verwendet für die Simulation 5.000 Elastons, ein Fünffaches mehr als diese Methode Partikel verwendet. Entsprechend weist die Simulation des *Elaston*-Ansatzes eine höhere Berechnungszeit auf. Rein optisch nehmen sich beide Methoden für diese Simulation nicht viel.

Das dritte Beispiel zeigt die Kollision zweier Monstertrucks (Abb. 8.3). Dabei besteht ein Monstertruck aus 300 Partikeln und 800 Kanten sowie 63.000 Vertices und 100.000 Dreiecken. Die Komplexität des visuellen Netzes übersteigt die des physikalischen Netzes weit mehr als bei den anderen Beispielen. Entsprechend benötigt das Skinning 30ms; die Berechnung der Interaktion und Kollision 8ms. Der Effekt der drehenden Räder wird durch drei explizite SM-Gruppen erzielt: Der Fahrzeugkörper sowie zwei Achsen. Dadurch befinden sich je zwei Partikel pro Achse in zwei expliziten Gruppen und gelten somit als nicht-orientiert, was zu einem Freiheitsgrad führt.

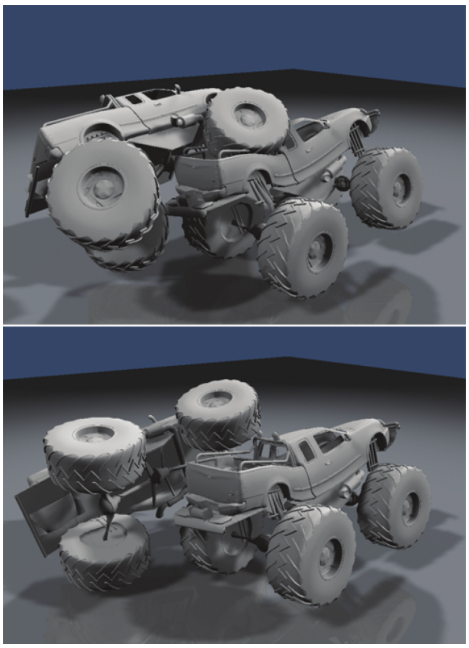

Abb. 8.3: Kollision und plastische Deformation von Monstertrucks

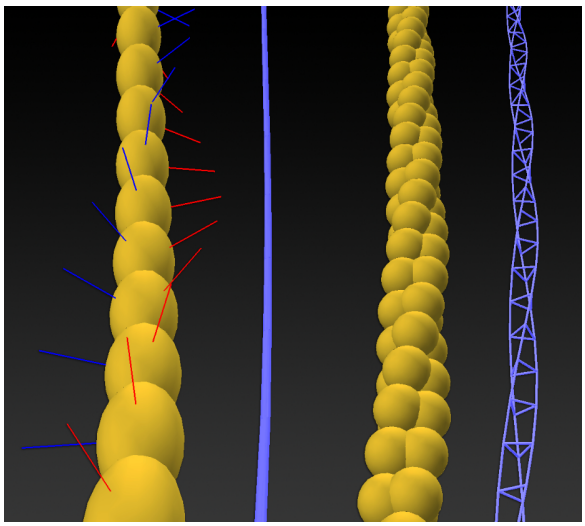

Abb. 8.4: Vergleich für Modellierung eines Seils mit orientierten und nicht-orientierten Partikeln;
Die Kantenstruktur ist in hellblau, die Orientierung in rot darstellt

Das letzte Beispiel zeigt einen Vergleich orientierter und nicht-orientierter Partikel bei der Modellierung eines Seils (Abb. 8.4). Mit orientierten Partikeln lässt sich ein Seil als eine einfache Kette bzw. Aneinanderreihung von Ellipsoiden darstellen. Eine einfache Kette mit Orientierungsinformationen ist ausreichend, um sowohl das Kollisionsvolumen als auch die Verdrehungsdynamiken zu simulieren. Für dieselbe Darstellung mit einem identischen Informationsgehalt mittels nicht-orientierter Partikel werden sechsmal mehr von genau diesen benötigt.

9 Fazit

Die vorgestellte Methode benötigt für die Simulation von Festkörpern relativ wenige Simulationspartikel und hinterlässt dennoch einen guten visuellen Eindruck. Diese dünne Repräsentation von Objekten geht zugunsten der Rechenzeit. Auch lassen sich die physikalischen Eigenschaften wie Biegsamkeit oder Festigkeit leicht durch den Benutzer manipulieren. Durch die zusätzlichen Orientierungsinformationen der Partikel ist Shape Matching gerade in dünnen Regionen des Netzes stabil. Mit dem Simulationsframework lassen sich vielseitige physikalische Modelle in kürzester Zeit erzeugen.

Ein Nachteil ist, dass das physikalische Verhalten direkt abhängig von der Netzgeometrie ist. Bei feinen Anpassungen konvergiert das Verhalten nicht zu einem bestimmten. Diese Tatsache führt zu Problemen, wollte man automatische Modelle unterschiedlicher Auflösung zur Optimierung der Laufzeit generieren.

Die entscheidenden Schritte dieser Methode sind das Shape Matching und das Skinning. Um diese Prozesse zu beschleunigen arbeiten die Autoren des Originalpaper derzeit an einer GPU-beschleunigten Version des *Solver*. Bisherige Simulation befassen sich lediglich mit der Interaktion und Kollision von Festkörpern miteinander. Der nächste Schritt wäre, Interaktionen mit Flüssigkeiten in einem gemeinsamen Framework zu untersuchen.

10 Referenzen

1 M. Müller, N. Chentanez:
Solid Simulation with Oriented Particles (2012)
http://www.matthiasmueller.info/publications/orientedparticles.pdf[02.02.14]

2 M. Müller, B. Heidelberger, M. Teschner, M. Gross:
Meshless Deformations Based on Shape Matching (2004)
http://www.beosil.com/download/MeshlessDeformations_SIG05.pdf[02.02.14]

3 M. Müller, B. Heidelberger, M. Hennix, J. Ratcliff:
Position Based Dynamics (2006)
http://www.matthiasmueller.info/publications/posbaseddyn.pdf[02.02.14]

4 L. Kavan, S. Collins, J. Zara, C. O'Sullivan:
Geometric Skinning with Approximate Dual Quaternion Blending (2008)
http://www.seas.upenn.edu/~ladislav/papers/sdq-tog08/sdq-tog08.pdf[02.02.14]

5 http://irit.fr/~Rodolphe.Vaillant/?e=29[02.02.14]

6 http://www.youtube.com/watch?v=LRHqs4GJuCA[02.02.14]

7 S. Martin, P. Kaufmann, M. Botsch, E. Grinspun, M. Gross:
Unified Simulation of Elastic Rods, Shells, and Solids (2010)
http://graphics.ethz.ch/~peterkau/download/publication_Mar10.pdf[02.02.14]