Fachseminar
Graphische Datenverarbeitung SS98
Prof. M. Gross

The Haptic Display of Complex Graphical Environments

Seminararbeit von Thomas Ammann

auf der Grundlage des gleichnamigen Papers von

Diego C. Ruspini, Krasimir Kolarov und Oussama Khatib
Stanford University

Inhalt

1 Motivation

Je länger je mehr werden Computer dazu verwendet, reale Situationen in virtuellen Umgebungen nachzubilden, um so Kosten zu sparen und Gefahren zu vermeiden. Grosser Aufwand wurde bei der Entwicklung von Flugsimulatoren betrieben, um so Piloten ausbilden zu können, ohne sie den Gefahren eines wirklichen Fluges aussetzen zu müssen. Mit den stetigen Fortschritten in der Computerhardware entstand auch ein zunehmendes Verlangen nach Virtual Reality Lösungen. Nebst Video- und Audiodaten, wie sie schon längst kombiniert eingesetzt werden, versucht man mit einer haptischen Schnittstelle, die Illusion zu perfektionieren. Virtual Reality Anwendungen verlangen typischerweise Interaktion und Manipulation, die ohne haptische Schnittstelle nur unbefriedigend verwirklicht werden können, wodurch der Benutzer zum schlichten Betrachter wird.

2 Penalty Methode

Bei Penalty Methoden erzeugt das haptische Device eine Kraft proportional zur Penetration des Objektes im virtuellen Raum. Für einfache geometrische Objekte sind die Richtung und Grösse der Kraft leicht zu bestimmen. Bei komplexeren Objekten ist es jedoch sehr schwierig, die nächste Oberfläche zu finden, und erfordert im schlimmsten Fall eine globale Suche (Abbildung 1(a)). Massie und Salisbury haben diese Technik erweitert, indem sie das Volumen eines Objektes unterteilt und jedem Volumenelement eine Oberfläche zugeordnet haben. Diese Methode kann jedoch nicht verhindern, dass Kraftdiskontinuitäten auftreten (Abbildung 1(b)). Der Benutzer stösst von oben her in ein Objekt und bewegt sich schliesslich in ein zweites Volumenelement, welchem die Oberfläche rechts zugeordnet ist, wodurch der Benutzer eine Kraft in diese Richtung verspürt. Bei sehr kleinen oder dünnen Objekten besteht die Gefahr, dass die Kraft nicht ausreicht, den Benutzer daran zu hindern, das Objekt zu durchdringen (Abbildung 1(c)).

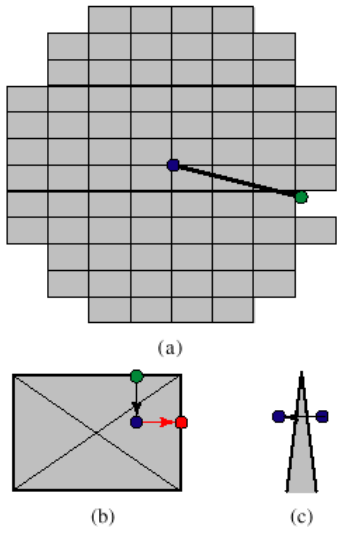

Abbildung 1[1]

[1] Sämtliche Abbildungen stammen aus [1]

3 Constraint basierte Methode

Eine Constraint basierte Methode wurde zuerst von Zilles und Salisbury in [2] vorgestellt und später von Ruspini übernommen und erweitert. Ein Schlüsselelement dieser Methode ist die virtuelle "Proxy", welche den Finger des Benutzers in der virtuellen Umgebung repräsentiert. Die Oberflächen der virtuellen Objekte bilden dabei die Constraints, welche die Bewegung der Proxy einschränken. Abbildung 2 zeigt, wie sich die Position der Proxy ändert, wenn sich der Finger des Benutzers bewegt. Die Bewegung der Proxy erinnert an diejenige eines Roboters, der versucht, sein Ziel zu erreichen. Solange der Roboter nicht behindert wird, bewegt er sich auf sein Ziel zu. Stösst er jedoch auf ein Hindernis, wodurch eine direkte Bewegung verunmöglicht wird, kann er entlang einer Oberfläche seine Distanz zum Ziel doch noch weiter verringern. Der Roboter stoppt, sobald die Distanz lokal minimal ist. Da zwischen der Bewegung des Roboters und derjenigen der Proxy eine enge Beziehung besteht, wurden viele der hier verwendeten Algorithmen in der Robotik entwickelt.
Die auf den Benutzer ausgeübte Kraft ist proportional zur Distanz zwischen Proxy und haptischem Interface. Dadurch soll bewirkt werden, dass sich der Benutzer der Proxy annähert, sobald diese ihm aufgrund eines Hindernisses nicht folgen kann.

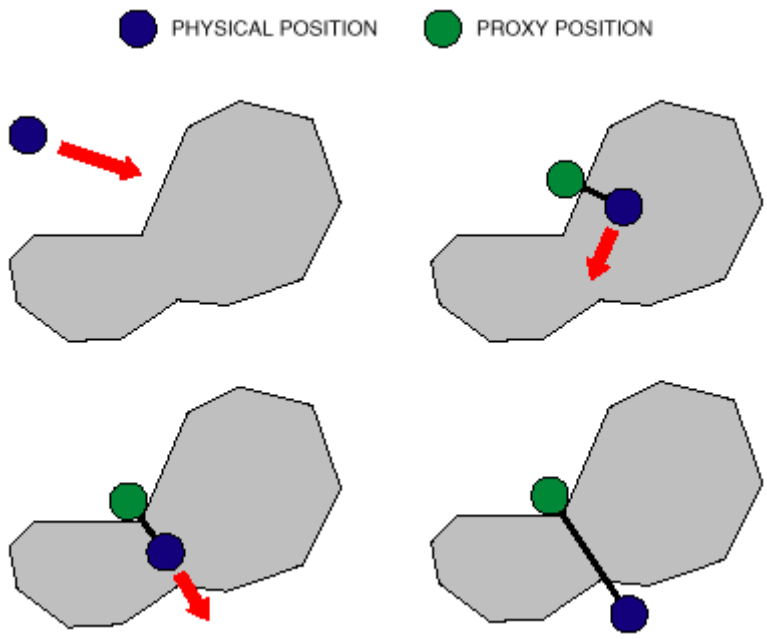

Abbildung 2

4 Collision Detection

Da ein Objekt normalerweise aus einer grossen Zahl von Primitiven besteht, ist es äusserst rechenintensiv, für jede dieser Primitiven abzuklären, ob sie den Weg der Proxy kreuzt. In den meisten Situationen wird die Proxy nur mit einem sehr kleinen Teil dieser Primitiven Kontakt haben. Mit hierarchisch angeordneten Kugeln lässt sich gemäss Quinlan[5] die Komplexität des Problems jedoch reduzieren.

Für jedes starre, d.h. nicht in sich bewegliche Objekt, kann die folgende Struktur aufgebaut werden. Jedes Polygon wird zuerst mit kleinen Kugeln vollständig überdeckt. Diese entsprechen den Blättern eines nahezu balancierten binären Baumes. Jeder der Knoten im Baum entspricht einer Kugel, die selbst alle Kugeln in den Blättern, nicht aber gezwungenermassen diejenigen in den Knoten des entsprechenden Teilbaums enthält. Mit einer Divide and Conquer Strategie, ähnlich dem Quick Sort Algorithmus, werden die Knoten des Baumes gebildet. Es wird eine Box um sämtliche Kugeln in den Blättern gelegt. Die Kugeln werden dann durch eine Ebene, die durch die Mittelpunkte der vier längsten Seiten der Box verläuft, in zwei idealerweise gleich grosse Mengen aufgespaltet. Der Baum wird gebildet, indem dieser Vorgang rekursiv für die entstandenen Teilmengen von Kugeln wiederholt wird. Um die Kugel für einen bestimmten Knoten zu berechnen, werden zwei Heuristiken verwendet. Die erste bestimmt die kleinste Kugel, welche beide Kugeln der Kindknoten beinhaltet. Die zweite untersucht direkt die Blattknoten. Bei dieser Variante liegt das Zentrum der Kugel im Mittelpunkt der entsprechenden Box, und der Radius wird so gewählt, dass alle Blattknoten in der Kugel enthalten sind. Für einen gegebenen Knoten wird diejenige Variante gewählt, aus welcher die Kugel mit dem kleineren Radius resultiert. Die erste Heuristik scheint in der Nähe der Blattknoten bessere Ergebnisse zu erzielen, die zweite bei der Wurzel. Der Algorithmus zum Aufbau dieser Struktur besitzt eine Komplexität $O(n \lg n)$, wobei n der Anzahl der Blattkugeln entspricht. Der Algorithmus zum Detektieren einer Kollision ist allerdings lediglich noch von der Komplexität $O(\lg n)$. Abbildung 3 zeigt die Kugelhierarchie eines typischen Modells, in welcher die Blätter gelb gefärbt sind.

Abbildung 3

5 Proxy Position

Für jeden Zeitschritt wird ein Ziel der Proxy definiert, welches sie durch eine lineare Bewegung zu erreichen versucht. Anfänglich entspricht dieses Ziel der Position des haptischen Interfaces, es wird jedoch geändert, sobald die Proxy auf ein Hindernis stösst. Beim Kontakt mit einer Constraint Plane wird die weitere Bewegung der Proxy sofort auf den Halbraum über der Ebene beschränkt. Alle sich im unteren Halbraum befindlichen Oberflächenelemente können für die weitere Collision Detection in diesem Zeitschritt eliminiert werden. Der Schnitt aller dieser Halbräume, die im Laufe eines Zeitschrittes durch Iteration entstehen, bildet einen konvexen, nicht beschränkten Polyeder. Die gesuchte Position der Proxy entspricht dem Punkt innerhalb dieses Polyeders, der die Distanz zum Benutzer minimiert. Da dieses Problem unabhängig von Koordinatentranslation ist und die aktuelle Proxy Position auf allen Constraint Planes liegt, kann das Problem kompakt wie folgt geschrieben werden

$$\begin{aligned} &\text{minimiere } \|x - p\| \text{ unter folgenden Nebenbedingungen} \\ &\hat{n}_1^T x \geq 0, \\ &\hat{n}_2^T x \geq 0, \\ &\vdots \\ &\hat{n}_m^T x \geq 0. \end{aligned} \qquad (1)$$

p ist dabei der Vektor von der aktuellen Proxy Position zum Benutzer, x ist das neue Zwischenziel, und $\hat{n}_i$, $0 \leq i \leq m$, sind die nach aussen gerichteten Einheitsnormalen der Constraint Planes.

Dieses Problem kann in zwei Schritten gelöst werden. Zuerst wird eine minimale Zahl von aktiven Constraint Planes definiert, die dann verwendet werden, um ein neues Zwischenziel zu berechnen. Falls die gesuchte Lösung auf einer Seite des Polyeders liegt, wird eine Constraint Plane benötigt. Liegt sie auf einer Kante, werden zwei benötigt, in einem Knoten drei.
Die minimale Anzahl der benötigten Constraint Planes kann mit Hilfe eines zum Polyeder dualen Körpers ermittelt werden. Dieser Körper wird gebildet durch die Einheitsnormalen der Constraint Planes und den Ursprung (Ebene im Unendlichen). Zur Bestimmung der aktiven Constraint Planes wird ein Einheitsvektor $\hat{p}$ eingeführt, der parallel zu p verläuft. Die Knoten der zu $\hat{p}$ nächstgelegenen Seite, Kante oder Knoten, respektive der Constraint Planes, bilden die Lösung. Liegt der nächste Punkt zu $\hat{p}$ auf einer Seite, existieren zwei aktive Constraint Planes. Liegt der nächste Punkt auf einer Kante zwischen einem der Knoten x und dem Knoten o, so wird eine Constraint Plane benötigt. Befindet sich der Vektor $\hat{p}$ jedoch unterhalb des Körpers, sind drei Constraints Planes aktiv, so dass die Position der Proxy zum Benutzer nicht weiter verringert werden kann. In Abbildung 4 wird die Proxy durch die Schnittgerade der beiden aktiven Constraint Planes a und b in ihrer Bewegung eingeschränkt.
Ersetzt man die Ungleichungen in (1) durch Gleichungen, kann das Problem mit Lagrange Multiplikatoren, wie es Zilles in [3] vorschlägt, gelöst werden. Da nie mehr als drei Constraint Planes aktiv sein können, kann die Lösung in $\boldsymbol{O}(m)$ Zeit gefunden werden, wobei m die Zahl der ursprünglichen Constraint Planes ist.
Nachdem auf diese Weise ein neues Zwischenziel gefunden worden ist, kann die Iteration fortgeführt werden.

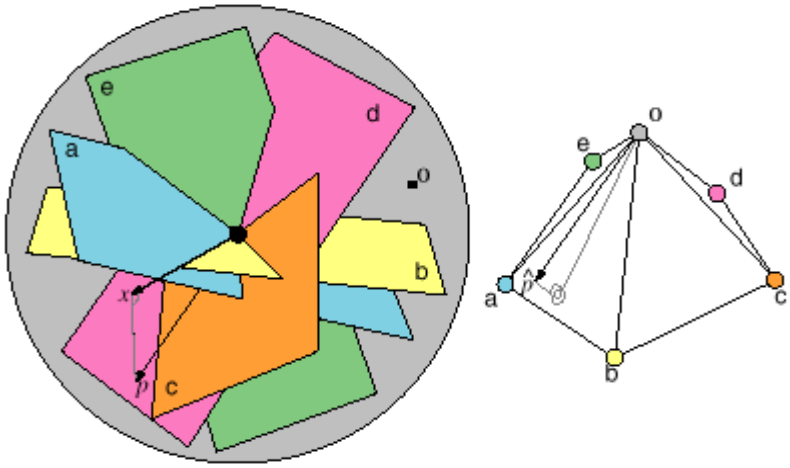

Abbildung 4

6 Force Shading

Die Objekte der realen Welt werden im haptischen System durch Polyeder nachgebildet, was jedoch für die wenigsten dieser Objekte genau möglich ist. Das Modell könnte zwar ausreichend verfeinert werden, so dass für den Menschen kein spürbarer Unterschied zur Realität mehr besteht, dies würde allerdings einen sehr grossen Aufwand an Speicher und Rechenleistung bedeuten. Eine Möglichkeit, Oberflächen zu gestalten, sind Non Uniform Rational B-Splines(NURBS), bei denen allerdings die Collision Detection sehr rechenintensiv ist. Da keine einfache Transformation zwischen globalen Koordinaten und den Koordinaten der Fläche existiert, muss ein Punkt auf der Oberfläche iterativ gesucht werden.
Aufgrund der Schwierigkeiten mit Freiformflächen werden stattdessen ebene Oberflächen verwendet, deren Normale aber zwischen den Knoten ähnlich dem Phong Shading interpoliert wird. In der Realität ist die Oberflächennormale definiert durch den Gradienten, welcher gegeben ist durch die Ableitungen an der Oberfläche. Bei virtuellen Objekten existiert diese Bedingung nicht, wodurch die Oberflächennormale von der Form eines Objektes entkoppelt werden kann.

Der hier beschriebene Ansatz erzeugt die Shading Effekte lediglich durch Veränderung der Proxy Position und trägt so zur Stabilität des Systems bei (Abbildung 5).
Stösst die Proxy auf eine Constraint Plane, wird die Oberflächennormale an dieser Stelle mit Hilfe der an den Knoten definierten Normalen interpoliert. Durch diese Normale wird eine zusätzliche Constraint Plane definiert, die ebenfalls durch den Kontaktpunkt verläuft. Im ersten Schritt wird ein neues Zwischenziel mit Hilfe der interpolierten anstelle der eigentlichen Constraint Plane berechnet. Dieses Zwischenziel wird nun als Position des Benutzers betrachtet. Im zweiten Schritt wird mit der ursprünglichen Constraint Plane und dem berechneten Zwischenziel das Ziel für diese Iteration evaluiert. Während diese Methode etwas rechenintensiver ist als frühere Ansätze, berücksichtigt sie alle Constraint Planes und erzeugt das richtige Resultat, auch wenn mehrere schattierte Oberflächen existieren.

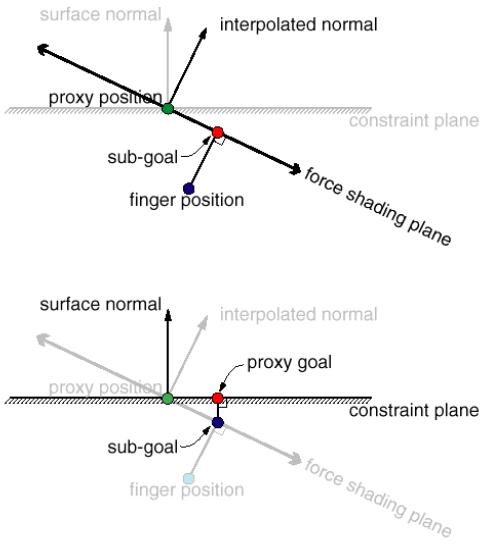

Abbildung 5

In Abbildung 6 ist der Unterschied zwischen einer schattierten und einer ebenen Oberfläche gezeigt. In beiden Grafiken ist die Distanz zwischen der Position des Benutzers und derjenigen der Proxy dargestellt, während der Benutzer sich rund um das Objekt bewegt. Wie man in Abbildung 6(a) erkennen kann, tritt an den Kanten eine Kraftdiskontinuität auf. In Abbildung 6(b) ist dies nicht der Fall, da die Kraft immer parallel zur interpolierten Oberflächennormale ist.

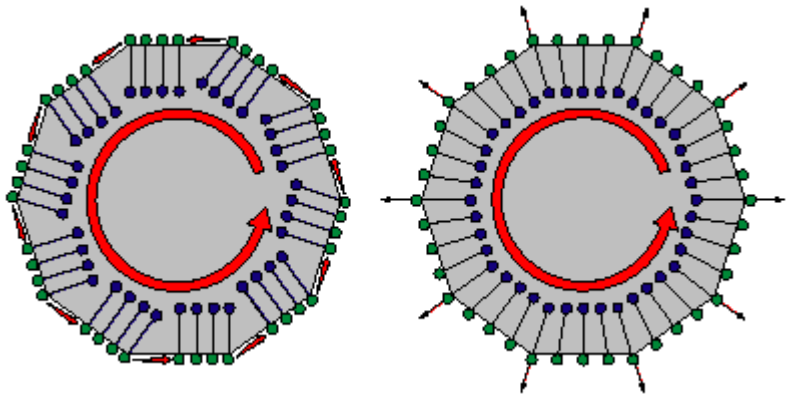

Abbildung 6 Unterschied zwischen Flat- und Force-Shading

7 Reibung

Wirkt auf einer Oberfläche ausschliesslich eine senkrechte Kraft, so fühlt sich ein Objekt spiegelglatt an, und der Benutzer neigt dazu, von konvexen Gebieten in konkave wegzugleiten. Die Reibung macht es erst möglich, eine Oberfläche zu untersuchen, Objekte zu manipulieren oder festzuhalten und zu bewegen. Dabei wird unterschieden zwischen statischer Reibung, welche die Proxy an einem bestimmten Punkt festhält, und dynamischer Reibung, welche die Geschwindigkeit der Proxy verringert, sobald diese sich bewegt.

7.1 Statische Reibung

Sind f_n und f_t die Kräfte vertikal und tangential zu einer gegebenen Constraint Plane, welche den statischen Reibungsparameter μ besitzt, so befindet sich die Proxy in statischem Kontakt mit der Oberfläche, falls $f_t \leq \mu f_n$. Solange sich also der Benutzer innerhalb des Kegels aufhält, wird die Proxy daran gehindert, sich zu bewegen, indem das Zwischenziel gleich der Position der Proxy gesetzt wird (Abbildung 7).

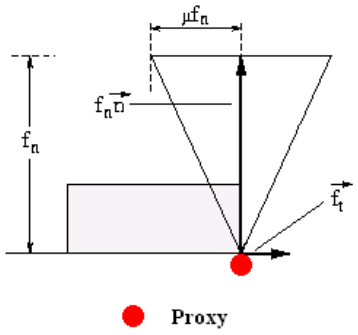

Abbildung 7

7.2 Dynamische Reibung

Die dynamische Reibungskraft*f* ist gegeben durch

$$f = -b\dot{x} \qquad (2)$$

wobei *b* dem dynamischen Reibungskoeffizienten entspricht. Der Proxy wirkt demzufolge eine Kraft proportional zu ihrer Geschwindigkeit entgegen.

7.3 Statische und dynamische Reibung

Um Oberflächen realistisch modellieren zu können, sind sowohl statische als auch dynamische Reibung nötig. Die Kraft *f* auf die Proxy ist gegeben durch

$$\begin{aligned} f &= f_t - \mu f_n - b\dot{x} \quad &&\text{,falls } \mu f_n < f_t - b\dot{x} \\ f &= 0 &&\text{sonst} \end{aligned} \qquad (3)$$

wobei μ wiederum dem statischen und *b* dem dynamischen Reibungskoeffizienten entspricht. Im dynamischen Gleichgewicht, d.h. wenn die Kraft auf die Proxy verschwindet, ist die Geschwindigkeit der Proxy gegeben durch

$$\begin{aligned} \dot{x} &= \frac{f_t - \mu f_n}{b} \quad &&\text{,falls } \mu f_n < f_t \\ \dot{x} &= 0 &&\text{sonst} \end{aligned} \qquad (4)$$

Im Falle mehrerer Constraint Planes wird das Minimum der Geschwindigkeiten verwendet.

8 Oberflächenfestigkeit

Einer Oberfläche wird ein Parameter s, $0 \leq s \leq 1$, für deren Oberflächenfestigkeit zugewiesen. Es wird ein Punkt *p'* bestimmt, welcher gegeben ist durch

$$p' = v + s(p - v) \tag{5}$$

wobei *p* der Position der Proxy auf der Constraint Plane und *v* der Position des Benutzers entspricht (Abbildung 8). Um die auf den Benutzer ausgeübte Kraft zu bestimmen, wird nun nicht mehr die Position der Proxy verwendet, sondern der Punkt *p'*. Die Position der Proxy wird aber weiterhin benötigt, um der Oberfläche eines Objektes folgen zu können. Bewegt sich die Proxy von einer Oberfläche zu einer benachbarten mit grösserer Oberflächenfestigkeit, hat dies den unangenehmen Effekt zur Folge, dass der Benutzer eine plötzliche Kraft verspürt, da der Punkt *p'* eine nicht stetige Bewegung macht. Es ist deshalb wünschenswert, die Oberflächenfestigkeit durch eine Veränderung der Oberflächen selbst zu realisieren, um so einen realistischeren Effekt zu erzeugen.

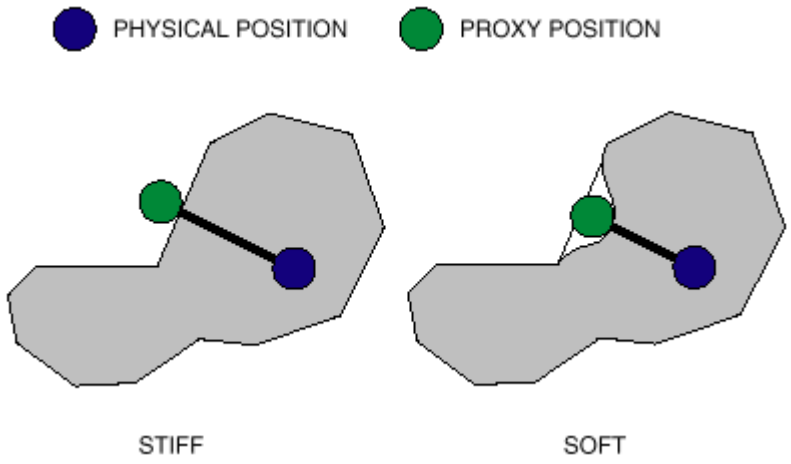

Abbildung 8

9 Texturen

Eigenschaften wie statische und dynamische Reibung oder Festigkeit werden auf die Oberfläche eines Objektes abgebildet, wodurch die verschiedenen haptischen Effekte erzeugt werden. Die Textur wird einmal pro Zyklus an der Stelle der Proxy ausgewertet, was dazu führen kann, dass Oberflächenmerkmale, die zwischen zwei Clocks liegen, nicht gerendert werden. Die Auswertung der Textur entlang des Weges der Proxy ist vor allem dann anzustreben, wenn die Textur sehr feine Linien im Verhältnis zu den Abständen aufweist.
Mit Hilfe einer weiteren Constraint Plane kann das Force Shading modifiziert werden, so dass der Oberfläche eine dreidimensionale Struktur gegeben werden kann (Bump Mapping). Mit Hilfe von weiteren Constraint Planes wird es möglich, anstelle einer stetig differenzierbaren eine Oberfläche mit Ecken und Kanten zu modellieren.

10 Ausblick

Die Bewegung von Objekten wird erzeugt, indem das Objekt wie in graphischen Anwendungen an seiner neuen Position gerendert wird. Dies resultiert in einer unstetigen Bewegung der Objekte. In einigen Fällen ist es sogar möglich, dass die Proxy im einen Zeitschritt ausserhalb und im nächsten innerhalb eines Objektes liegt.
Interessant ist eine Erweiterung um implizit gegebene Freiformflächen wie B-Splines oder die Verwendung von Volumendaten, ohne diese zuerst in eine Oberflächenvermaschung transformieren zu müssen.

Literaturverzeichnis

[1] Ruspini D., Kolarov K., Khatib O., "**The Haptic Display of Complex Graphical Environments**", 1997. Stanford University, Interval Research Corporation, 1997, ACM SIGGRAPH.

[2] Zilles C., Salisbury K., "**A Constraint-based God-object Method for Haptic Display**", 1994. Dynamic Systems and Control 1994, vol. 1, pp 146 - 150.

[3] Zilles C., "**Haptic Rendering with the Toolhandle Haptic Interface**", 1995. M.Eng. thesis, MIT.

[4] Mark W., Randolph S., Finch M., Van Verth J., Taylor R., "**Adding Force Feedback to Graphics Systems: Issues and Solutions**", 1996. Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp 447 - 452.

[5] Quinlan S., "**Efficient Distance Computation between Non-Convex Objects**", Int. Conference on Robotics and Automation, 1994