



Fachhochschule Bingen



DIPLOMARBEIT

VON

ANDREAS DAHM

Kollisionserkennung und Behandlung für komplexe Kleidungsstücke

**Betreuer: Prof. Dr.-Ing. Volker Luckas
Dr.-Ing. Arnulph Fuhrmann**

München, März 2010

Diplomaufgabe

Thema: Kollisionserkennung und Behandlung für komplexe Kleidungsstücke

Die Kollisionserkennung zwischen einzelnen Kleidungsstücken, sowie innerhalb der einzelnen Schnitteile, ist für eine realistische Simulation von Kleidung unabdingbar.

Gleichzeitig aber auch aufgrund der Flexibilität des Material sehr aufwändig. Für die Kollisionserkennung zwischen unterschiedlichen Kleidungsstücken steht das sehr robuste Layering zu Verfügung, das Durchdringungen von Schnitteilen auf unterschiedlichen Lagen effektiv verhindert.

Allerdings treten aktuell innerhalb eines Schnitteiles teilweise Selbstdurchdringungen auf. Dies tritt bei komplexen Kleidungsstücken wie z.B. Faltenröcken auf, bei denen gerade das Simulieren von Falten problematisch ist.

Aufgabe dieser Diplomarbeit ist es, neben der Erkennung solcher Selbstdurchdringungen, vor allem das Auflösen der Kollisionen zur Laufzeit. Diese Auflösung soll sich möglichst nahtlos in die bisherige Simulation eingliedern.

Dabei ist zu beachten, dass die Behandlung der Kollisionen die Qualität der Simulation verbessern soll. Dazu ist es auf jeden Fall von Nöten, die Kollisionsantwort physikalisch plausibel zu gestalten.

Vorwort

Mein besonderer Dank gilt in erster Linie meinen Eltern, ohne deren uneingeschränkte Unterstützung und Vertrauen in mich, es deutlich schwieriger geworden wäre, mein Studium in der Art und Weise zu bestreiten, wie ich es getan habe. Auch wäre ohne ihre Unterstützung die Diplomarbeit in München nur sehr schwer realisierbar gewesen.

Die vorliegende Arbeit wurde bei der Firma assyst GmbH in Aschheim-Dornach geschrieben. Ich danke der assyst GmbH für das Bereitstellen des Themas und die Möglichkeit, diese umzusetzen.

Für die Unterstützung und Betreuung innerhalb der assyst GmbH möchte ich mich bei Herrn Dr.-Ing. Arnulph Fuhrmann und Herrn Harald Preusser für die vielen Ratschläge, konstruktive Kritik und Testdaten bedanken.

Großer Dank gilt auch Herrn Prof. Dr.-Ing. Volker Luckas, der die Betreuung der Diplomarbeit seitens der Fachhochschule Bingen übernommen hat.

Last but not least möchte ich mich bei meinen Korrekturlesern bedanken. Vor allem bei meinem Onkel Herrn Michael Hüntel und meinem Kommilitonen Herrn Lothar Herzog.

München, März 2010

Andreas Dahm

Inhaltsverzeichnis

Erklärung zur Diplomarbeit	I
Vorwort	II
Inhaltsverzeichnis	III
1 Einleitung	1
1.1 Motivation	1
1.2 Aufgabenstellung und Zielsetzung	1
1.3 Aufbau der Arbeit.....	2
1.4 Abgrenzung	3
2 Bisherige Erkennung und Vermeidung von Kollisionen.....	4
2.1 Einleitung.....	4
2.2 Kollisionserkennung und Behandlung zwischen Kleidung und Avatar.....	5
2.2.1 Einleitung	5
2.2.2 Funktionsweise	5
2.2.3 Nachteil	5
2.3 Kollisionsvermeidung zwischen der Kleidung	6
2.3.1 Einleitung	6
2.3.2 Funktionsweise	6
2.3.2.1 SelfCollisionAvoider.....	6
2.3.2.2 TriangleSelfCollisionDetector.....	6
2.3.3 Nachteile	7
2.4 Kollisionsbehandlung zwischen verschiedenen Kleidungslagen	8
2.4.1 Einleitung	8
2.4.2 Funktionsweise	8
2.4.3 Nachteil	8
2.5 Nutzen in Vidya	9
3 Grundlagen – Theoretische Betrachtung.....	11
3.1 Allgemein.....	11
3.2 Problem der Durchdringung.....	11
3.3 Einarbeitung in die Thematik	13
3.4 Mögliche Ansätze	13
3.5 Resolving Surface Collisions through Intersection Contour Minimization ..	17
3.5.1 Einleitung	17
3.5.2 Identifikation der Kollisionskonturen	17
3.5.3 Minimierung der Kollisionskonturen.....	19

3.5.4	Skalierung des Gradientenvektors	21
3.5.5	Lokales und globales Schema.....	21
3.5.6	Besonderheiten.....	23
3.5.7	Limitationen.....	23
3.6	Conservation of Momentum.....	24
4	Umsetzung	27
4.1	Algorithmen	27
4.1.1	Einleitung	27
4.1.2	Kompletter Ablauf.....	28
4.1.3	Erstellen von Intersection-Contour-Segmenten	29
4.1.4	Erstellung der Intersection-Contour	30
4.1.5	Einfluss der Massen und Positionen	31
4.1.6	Spatial Hashtable	35
4.1.7	Bounding Volume Hierachy (BVH)	35
4.2	Implementierung.....	37
4.2.1	Flussdiagramm	37
4.2.2	Klassendiagramm	38
4.2.3	Datenstrukturen	39
4.2.3.1	IntersectionContourSegmentList.....	39
4.2.3.2	SpatialHashEntryIntersectionContourSegment.....	40
4.2.3.3	IntersectionContour	40
4.2.4	Schwerpunkt bei der Umsetzung.....	41
5	Erweiterung des Verfahrens	42
5.1	Globales Verfahren auch bei einem Mesh	42
5.2	Physikalisch plausible Kollisionsantwort	44
5.3	Kombination der Verfahren zur Verbesserung Laufzeit und Ergebnisse ...	44
6	Ergebnisse	46
6.1	Übersicht	46
6.2	Testszenarien.....	46
6.2.1	Selbstdurchdringung Arminnenseite.....	46
6.2.2	Selbstdurchdringung Armbeuge	49
6.2.3	Extremtest rotierende Kugel mit starken Durchdringungen.....	51
6.2.4	Durchdringung verschiedene Stofflagen.....	54
6.2.5	Auflösung von Durchdringungen in mehreren Schritten	56
6.3	Zeitliche Kohärenz.....	59
6.3.1	Vergleich zu den anderen Kollisionsdetektoren.....	59
6.3.2	Optimierung der Laufzeit durch Kombination der Verfahren	60
6.4	Aufgetretene Probleme	60
6.4.1	Performance	60
6.4.2	Bestimmen der Parameter	61
6.5	Limitationen	62

6.5.1	Fast planare Ebenen.....	62
6.5.2	Planare Ebenen	63
6.5.3	Zuständigkeitsfreier Bereich	63
7	Konklusion.....	64
8	Zusammenfassung und Ausblick.....	66
	Abbildungsverzeichnis	68
	Diagrammverzeichnis	70
	Literatur- und Quellenverzeichnis.....	71
	Verwendete Software	73
	Anhang	74

1 Einleitung

1.1 Motivation

Im Bereich der Bekleidungsindustrie hat die Simulation von realistischer Kleidung in den letzten Jahren immer mehr an Relevanz gewonnen. Aus diesem Grund bietet das Gebiet der Bekleidungssimulation viele interessante Forschungsthemen, in denen auch in früher Vergangenheit sehr intensiv geforscht und entwickelt wurde. Auch die immer besser werdenden numerischen Verfahren und die immer schneller werdenden Rechner erlauben es, komplexe dynamische Szenen sowohl physikalisch korrekt, als auch in vertretbarer Zeit zu berechnen. Eines der Kernthemen der meisten solcher Simulationen ist die Behandlung, Erkennung und Vermeidung von Kollisionen. Die Relevanz begründet sich darauf, dass die Durchdringung zweier fester Körper in der Realität nicht möglich ist. Gerade unmögliche Zustände fallen sehr stark ins Auge, da sie nicht Erwartungskonform für den Nutzer sind und hinterlassen damit einen negativen Eindruck. Ein Kollisionserkennungssystem hat einige wichtige Eigenschaften, welche bei der Entwicklung berücksichtigt werden müssen. Die eingesetzten Verfahren müssen mit bestehenden Verfahren zusammenarbeiten. Sie dürfen die Rechenzeit nicht zu stark negativ beeinflussen, müssen physikalisch zumindest plausibel und vor allem robust arbeiten. Unter robust ist zu verstehen, dass es keine oder nur wenige Situationen geben darf in denen das System nicht arbeitet oder es die Situation noch weiter verschlimmert. Die Verfahren die für diese Anforderungen eingesetzt werden, sind meist sehr komplex und nicht selten ein Schwachpunkt von vielen Simulationen.

Die Hauptmotivation ist es nun in dieser Arbeit eine Möglichkeit zu erarbeiten, um auftretende Selbstdurchdringungen aufzulösen. Unter Selbstdurchdringungen ist zu verstehen, dass ein zusammenhängendes Stoffstück mit sich selber kollidiert. Diese Art von Kollisionserkennung ist zur Zeit noch nicht in Vidya möglich. Warum dies noch nicht implementiert ist und wo genau die Schwierigkeiten dabei liegen zeigt Kapitel 3. Da aber solche Durchdringungen immer wieder auftreten, müssen sie gelöst werden. Durch die Lösung dieses Problems gewinnt die Simulation nochmals deutlich an Qualität und vereinfacht das Arbeiten mit Vidya extrem.

1.2 Aufgabenstellung und Zielsetzung

Das Thema dieser Diplomarbeit ist es, eine robuste Kollisionserkennung und Behandlung zu entwickeln, welche an die Anforderungen der Simulationssoftware Vidya angepasst ist. Dabei sind drei Kernbedingungen definiert, die auf jeden Fall zu lösen sind. Erstens sollen Selbstkollisionen erkannt und automatisch aufgelöst wer-

den können. Zweitens soll das Verfahren auch mit starken initialen Durchdringungen umgehen und diese zuverlässig automatisch auflösen können und drittens sollen auch kleinere zur Laufzeit auftretende Durchdringungen erkannt und aufgelöst werden. Dies bedeutet, dass vor allem hochaufgelöste Polygongitter, die sich stark deformieren und selbst durchdringen können, verarbeitet werden müssen. Es müssen demzufolge Methoden gefunden werden, welche auch mit vielen Polygonen schnell arbeiten und auch bei starken Selbstdurchdringungen zuverlässig zu einem durchdringungsfreien Zustand konvergieren.

1.3 Aufbau der Arbeit

Zu Beginn der Arbeit wird in Kapitel 2 der Ist-Zustand in Vidya festgestellt, um sich einen Überblick zu verschaffen, welche Methoden es bereits gibt, welche Strukturen genutzt werden können und auf was bei der späteren Reintegration zu achten ist. Anschließend wird in Kapitel 3 analysiert, welche anderen Verfahren es für das konkrete Problem gibt und es wird über die Vor- und Nachteile der einzelnen Verfahren diskutiert. Nachdem alle Schwerpunkte gegeneinander abgewogen sind, wird sich für das Verfahren entschieden, welches am viel versprechendsten für die Lösung des Problems ist. In Kapitel 4 wird die konkrete Umsetzung beschrieben, welche speziellen Datenstrukturen nötig sind, wie der grundsätzliche Ablauf des Verfahrens ist und wie die einzelnen Komponenten zusammenarbeiten. In diesem Kapitel wird auch auf die Implementierung des Verfahrens eingegangen. An welchen Stellen das Verfahren wie und aus welchem Grund verändert oder erweitert wurde, wird in Kapitel 5 beschrieben. Anschließend wird in Kapitel 6 genau untersucht, ob die Ergebnisse des Verfahrens allen Anforderungen gerecht werden, wo es Probleme gegeben hat, wie diese gelöst wurden und welche weiteren Probleme weiterhin bestehen. Zudem wird auch die Laufzeitveränderung durch den Einsatz der Kollisionserkennung analysiert. In Kapitel 7 wird ein Fazit über die Arbeit gezogen. Einige Entscheidungen werden diskutiert und es wird auch erläutert, ob und wie gut das eingesetzte Verfahren ist. Abschließend erfolgt in Kapitel 8 eine Zusammenfassung der Arbeit. Zudem wird auch betrachtet, an welchen Stellen das Verfahren noch verbessert werden kann oder welche Möglichkeiten es gibt, dieses in Zukunft weiterzuentwickeln.

1.4 Abgrenzung

Für das vollständige Verständnis dieser Arbeit sind Kenntnisse in folgenden Bereichen nötig:

- Java
- Computergrafik
- Mathematisches Verständnis über Vektoren
- Grundlegende Informatikkenntnisse
- Grundkenntnisse im Versionisierungswerkzeug SVN
- Softwareengineering

2 Bisherige Erkennung und Vermeidung von Kollisionen

Im folgenden Kapitel wird beschrieben, wie der aktuelle Zustand in Vidya ist. Dazu werden die bestehenden Verfahren kurz erläutert, da diese mit dem neuen Verfahren reibungslos zusammenarbeiten müssen.

2.1 Einleitung

In Vidya sind einige Kollisionsdetektoren zur Vermeidung von Kleidungskollisionen untereinander, zur Auflösung von Kollisionen zwischen Avatar und Kleidung oder verschiedenen Kleidungslagen bereits aktiv. Das neue Verfahren muss sich in den bisherigen Ablauf (siehe Diagramm 2-1) nahtlos eingliedern und darf die anderen Detektoren nicht negativ beeinflussen oder das Ergebnis der bereits durchgeführten Tests verfälschen. Damit dies gewährleistet ist, müssen die vorhandenen Verfahren analysiert werden um etwaige Seiteneffekte zu erkennen.

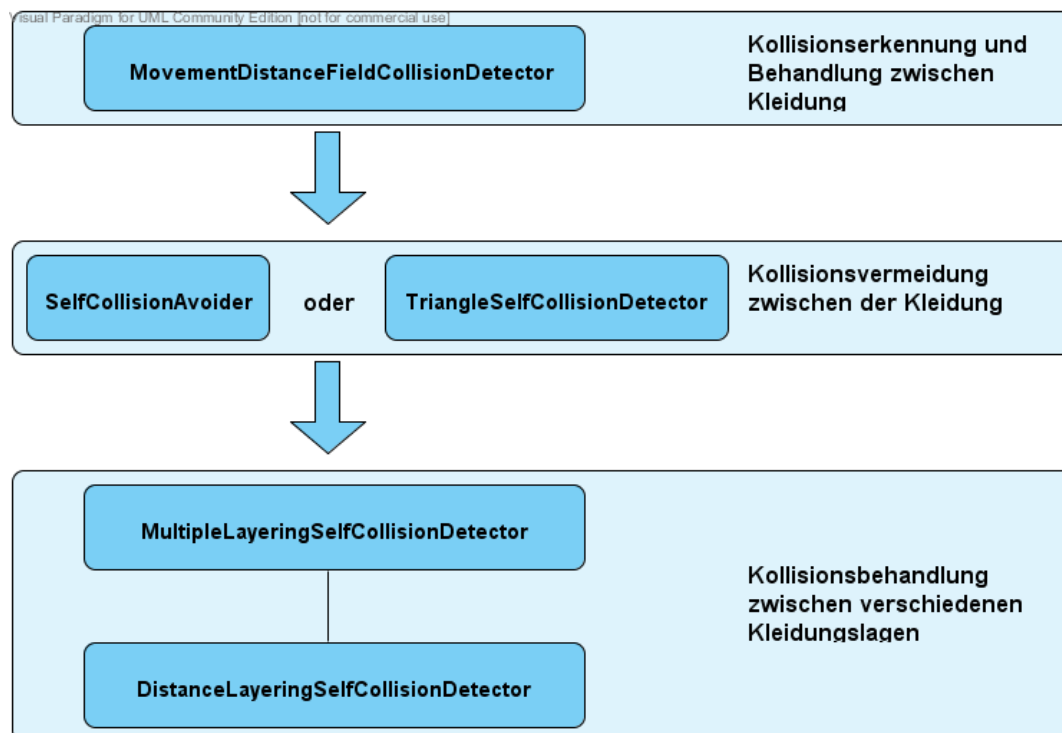


Diagramm 2-1 Bisherige Ablaufreihenfolge der Kollisionsdetektoren in Vidya.

2.2 Kollisionserkennung und Behandlung zwischen Kleidung und Avatar

2.2.1 Einleitung

Ein anderes zentrales Anwendungsgebiet für ein Kollisionserkennungssystem in Vidya ist die Behandlung von Kollisionen zwischen Kleidung und Avatar. Dazu gibt es in Vidya schon eine sehr robuste und schnelle Lösung. Dabei muss darauf geachtet werden, dass der Stoff beim Simulieren nicht in den Avatar eindringt, er muss sich aber der Form des Avatars im Rahmen seiner physikalischen Parameter anschmiegen. Da dabei ebenfalls beim Eindringen Partikel bewegt und deren Positionen bzw. Geschwindigkeiten verändert werden, muss dieses Verfahren mit in Betracht gezogen werden, auch wenn es nichts direkt mit der Problematik dieser Diplomarbeit zu tun hat.

2.2.2 Funktionsweise

Bei dieser Kollisionserkennung wird anders vorgegangen als bei klassischen Kollisionserkennungssystemen. Zu Beginn wird anhand des Volumens vom Avatar ein sogenanntes Distanzfeld berechnet. Dies hat den Vorteil, dass es für die Laufzeit unerheblich ist, aus wie vielen Polygonen der Avatar aufgebaut ist. Die Kollisionserkennung und Behandlung ist auf der räumlichen Ausdehnung des Avatars aufgebaut. Anschließend interagieren die Polygone des Stoffes mit dem erstellten Distanzfeld. Dadurch ist eine sehr robuste und schnelle Kollisionsbehandlung möglich, ohne auf die Auflösung des Avatars Rücksicht nehmen zu müssen.

2.2.3 Nachteil

Das Verfahren an sich hat keinen relevanten Nachteil, außer dass einmalig das Distanzfeld für den Avatar berechnet werden muss. Da dies aber nach einmaliger Berechnung persistent abgespeichert werden kann, ist es kein wirklicher Nachteil. Das eigentliche Problem ist, dass sich diese Funktionsweise nicht für die in dieser Diplomarbeit gestellte Problematik anwenden lässt. Der entscheidende Grund ist dabei, dass es sich bei dieser Diplomarbeit um ein Problem zwischen deformierbaren Körpern und nicht zwischen einem starren und einem deformierbaren Körper handelt. Aus diesem Grund müsste zu jedem Zeitschritt das Distanzfeld neu berechnet werden, was aus Laufzeitsicht nicht vertretbar ist.

2.3 Kollisionsvermeidung zwischen der Kleidung

2.3.1 Einleitung

Eine gute und übliche Weise mit Kollisionen umzugehen ist es, sie im Vorfeld zu vermeiden. Dazu gibt es in Vidya auch bereits implementierte Verfahren. Diese Verfahren helfen, die Simulation ruhiger und das Ergebnis besser aussehen zu lassen. Gerade mit den in diesem Abschnitt beschriebenen Verfahren muss eng zusammengearbeitet werden. Ohne diese Verfahren würden zwar die Kollisionen aufgelöst werden, da aber nichts den Stoff auf Abstand halten würde, würden bei jedem Zeitschritt durch Benutzerinteraktion, Schwerkraft oder Bewegung neue Durchdringungen auftreten. Das Ergebnis wäre eine sehr unruhige Simulation mit welcher der Benutzer nicht arbeiten könnte.

2.3.2 Funktionsweise

2.3.2.1 SelfCollisionAvoider

Der SelfCollisionAvoider arbeitet direkt auf den Eckpunkten des Polygongitters der Kleidungsstücke. Dabei wird versucht, zwischen den Eckpunkten der einzelnen Dreiecke einen minimalen Abstand einzuhalten. Sprich: wenn ein minimaler Abstand unterschritten wurde, werden die beiden beteiligten Eckpunkte von einander weggeschoben und so der Abstand eingehalten. Dies funktioniert bei großen Abständen sehr gut und ist auch nicht zu Rechenintensiv. Problematisch wird es erst, wenn durch eine initiale Durchdringung oder durch Benutzerinteraktion eine Durchdringung passiert.

2.3.2.2 TriangleSelfCollisionDetector

Der TriangleSelfCollisionDetector arbeitet ähnlich wie der SelfCollisionAvoider, nur betrachtet er nicht die einzelnen Partikel, sondern zusammengehörige Dreiecke und hält dadurch verschiedene Stoffe auf Abstand, wie in Abbildung 2-1 zu sehen. Dadurch ist er Rechenintensiver als der SelfCollisionAvoider, funktioniert aber bei kleinen Distanzen zuverlässiger. Der TriangleSelfCollisionDetector hat aber dieselben Probleme wie der SelfCollisionAvoider, denn er ist nicht zum Auflösen von Kollisionen gedacht. Das heißt, bei initialen oder durch Benutzerinteraktion hervorgerufenen Durchdringungen ist es diesem Verfahren nicht mehr möglich, die Kleidungsstücke auf Abstand zu halten.



Abbildung 2-1 Zwei Textilstücke, die nur durch den TriangleSelfCollisionDetector auf Abstand gehalten werden.

2.3.3 Nachteile

Beide Kollisionsdetektoren haben allerdings einen entscheidenden Nachteil. Solange der Stoff Durchdringungsfrei ist arbeiten sie sehr robust und gut. Sollte allerdings mal eine Durchdringung auftreten, wird diese durch die Detektoren sogar noch konserviert. Dies liegt daran, dass hier nur Eckpunkte bzw. Dreiecke auf Abstand gehalten werden. Doch auf welcher Seite sie auf Abstand gehalten werden, ist den Detektoren egal. Sprich: dringt Dreieck A in Dreieck B ein, werden beide Dreiecke zwar durch den TriangleSelfCollisionDetector auf Abstand gehalten. Da die Durchdringung aber schon statt gefunden hat und Partikel von Dreieck A schon auf der falschen Seite sind, wird der Abstand auf der falschen Seite versucht aufrecht zu halten. Das führt zur Fixierung des Problems. Dies ist wichtig zu wissen, da das neue Verfahren gegen diese Kräfte ankämpfen muss, um trotzdem die Kollision aufzulösen, auch wenn die beiden Detektoren dagegen arbeiten.

2.4 Kollisionsbehandlung zwischen verschiedenen Kleidungsstücken

2.4.1 Einleitung

Zum auseinanderhalten verschiedener Stofflagen, wird zurzeit in Vidya das sogenannte Layering eingesetzt. Dadurch lassen sich solche speziellen Anforderungen wie Bluse über Rock oder Rock über Bluse (siehe Abbildung 2-2) sehr robust simulieren. Wichtig ist noch, dass das Layering aktuell bei der Simulation, auch Durchdringungen zwischen verschiedenen Stofflagen behebt und daher das neue Verfahren mit dem Layering zusammenarbeiten muss.

2.4.2 Funktionsweise

In Vidya funktioniert das Layering eigentlich sehr trivial. Und zwar wird der Abstand eines Partikels zum Avatar gemessen. Anschließend der Abstand eines anderen Partikels auf einem anderen Layer. Sollten die Partikel aus dem höheren Layer einen niedrigeren Abstand zum Avatar haben als die aus dem niedrigeren Layer, tauschen diese beiden Partikel ihre Positionen. Dadurch ist dann wieder die korrekte Reihenfolge der Layer garantiert. Durch diesen Mechanismus ist das Layering sehr robust.

2.4.3 Nachteil

Das Layering hat allerdings mehrere Nachteile. Es ist relativ aufwändig, da jedes mal die Distanzen aller Partikel geprüft und verglichen werden müssen. Weiterhin funktioniert es nur mit einem Avatar, da dieser als Referenzobjekt genommen wird. Ist jetzt kein Avatar verfügbar, funktioniert das Layering nicht. Der dritte Nachteil ist, dass nur Durchdringungen aufgelöst werden, wenn die Kleidungsstücke verschiedene Lagen haben. Haben die Kleidungsstücke dieselbe Lage, wird das Layering nicht aktiv. Dadurch, dass es im Grundsatz darauf aufbaut, dass es mehrere Stofflagen gibt, ist es unglücklicherweise auch nicht zu Behandlung von Selbstkollisionen geeignet.

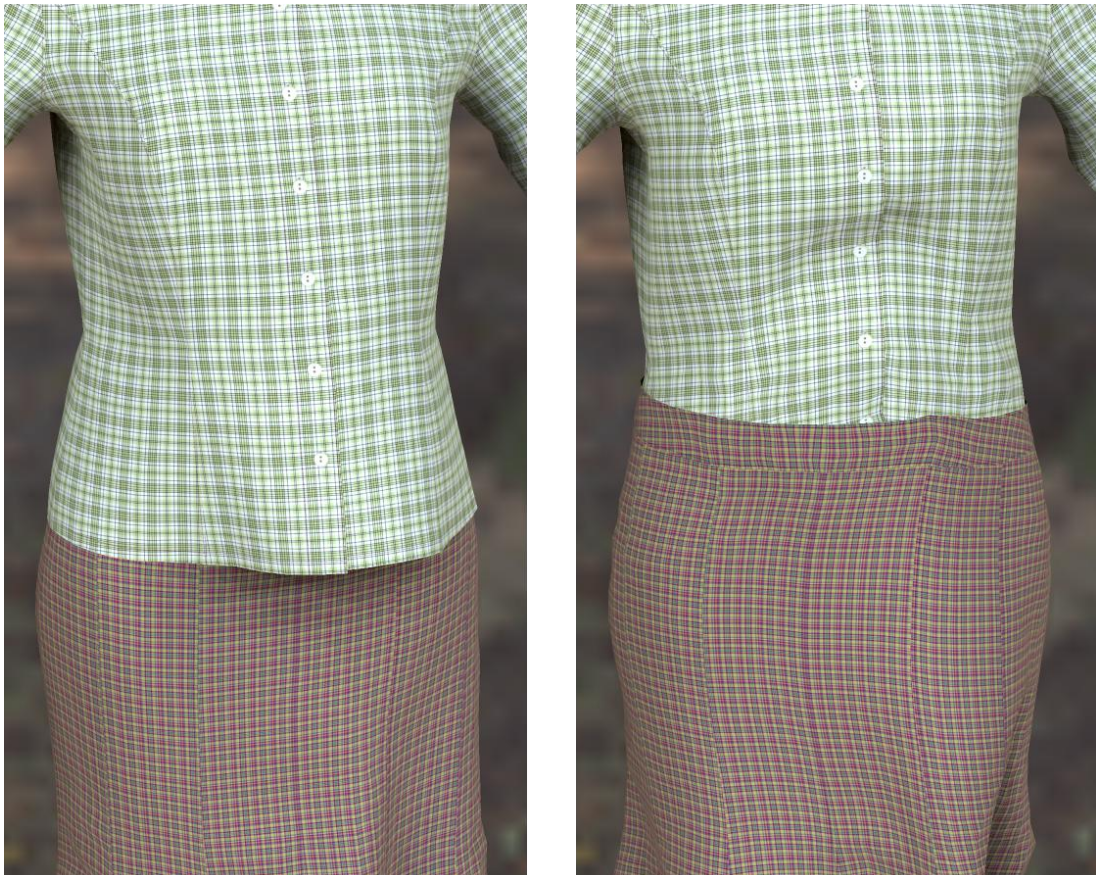


Abbildung 2-2 Bluse über Rock (l.) oder Rock über Bluse (r.)

2.5 Nutzen in Vidya

Die bisherigen Abschnitte zeigten eine Zusammenfassung über alle bereits vorhandenen Verfahren in Vidya. Diese Verfahren werden zur Kollisionsvermeidung oder Kollisionsbehandlung genutzt. Dadurch, dass es schon so viele aktive Verfahren gibt und dennoch die Notwendigkeit besteht ein neues Verfahren zu entwickeln, da die vorhandenen nicht alle Probleme abfangen, zeigt sich auch die Relevanz und Komplexität von Kollisionserkennungssystemen.

Zwar ist es möglich, mit den bestehenden Verfahren Kollisionen zu vermeiden, allerdings bieten diese fast keine Möglichkeiten, aufgetretene Kollisionen robust und vor allem automatisch zu behandeln. Bisher war bei solchen Durchdringungen wie in Abbildung 2-3, immer Benutzerinteraktion nötig, um die Durchdringung aufzulösen.

Das Layering aus Abschnitt 2.3 bietet zwar einen gewissen Automatismus um robust Kollisionen zwischen verschiedenen Stofflagen aufzulösen, jedoch ist das Layering an sich sehr rechenintensiv und versagt bei Durchdringungen bei nur einer Stofflage.

Der Nutzen der Diplomarbeit soll in Vidya sein, eine Methode zu implementieren, welche die Spezialfälle behandelt, bei denen die anderen Methoden versagen. Dazu gehört in erster Linie die Selbstkollisionen aufzulösen. Dieses Problem tritt sehr häufig in der Praxis auf. Vor allem bei komplexen Kleidungsstücken wie beispielsweise Faltenröcken.

Der Grund, weshalb dies in der Praxis recht häufig auftritt liegt daran, dass Vidya keine kontinuierliche Kollisionserkennung hat. Das heißt, es wird nicht zwischen jedem Zeitschritt interpoliert und geprüft, ob der neue berechnete Zustand kollisionsbehaftet ist, sondern der Zustand für Zeitschritt A und anschließend der für Zeitschritt B berechnet. Dadurch können Kollisionen beim Berechnen des neuen Zeitschritts auftauchen, welche im Nachhinein behandelt werden müssen.

Zusammenfassend bedeutet dies, dass das neue Verfahren sowohl mit kleinen zur Laufzeit auftretenden Durchdringungen, als auch mit großen initialen Durchdringungen zurechtkommen muss.



Abbildung 2-3 Kapuze und Sportjacke mit starken Durchdringungen.

3 Grundlagen – Theoretische Betrachtung

3.1 Allgemein

Wie eingangs erwähnt, ist die Kollisionserkennung und deren Behandlung eine der häufigsten Anforderungen in einer Simulationssoftware. Dabei ist es wichtig sowohl Selbstkollisionen, als auch Kollisionen verschiedener Kleidungsstücke untereinander zu erkennen und zu behandeln. In diesem Bereich wurde sehr viel in den letzten Jahren entwickelt. Bei Teschner, et al. [2005] findet sich ein State of the Art Report, der einen Überblick über den aktuellen Stand der Dinge zeigt.

Gerade weil es nicht immer möglich ist Kollisionen zu vermeiden, da sie entweder Initial oder durch Benutzerinteraktion aufgetreten sind, ist es sehr wichtig Methoden zu entwickeln, die auftretende Durchdringungen zur Laufzeit auflösen und dabei robust funktionieren.

3.2 Problem der Durchdringung

Das große Problem bei Durchdringungen ist es, wie damit umgegangen werden soll. Dazu gibt es seit Jahren schon die verschiedensten Ansätze. Die entsprechenden Arbeiten wurden in Diagramm 3-1 anhand ihrer Veröffentlichungen aufgeführt. Es ist zu erkennen, dass die Behandlung von Selbstkollisionen kein einfaches Unterfangen ist und intensiv daran geforscht wird. Dabei wurden in diesem Diagramm nur die wichtigsten Arbeiten der letzten 20 Jahre aufgeführt. Für die konkrete Betrachtung in Abschnitt 3.4 wurde sich auf die Arbeiten der letzten 10 Jahre konzentriert, da die meisten der aktuelleren Arbeiten auf die früheren aufbauen. Allerdings muss sich erst einmal genau überlegt werden, wo die Problematik dabei liegt. Deformierbare Körper wie Kleidungsstücke, können in den verworrensten Arten und Weisen miteinander kollidieren. Und schon alleine so ein „triviales“ Beispiel wie in Abbildung 3-1 zeigt, dass eine Auflösung der Situation nicht ganz so einfach ist. Es tauchen Fragen auf wie „Möglichst schnell auflösen?“, „Möglichst realistisch auflösen?“, „Alle Kollisionen in einem Schritt oder in mehreren auflösen?“ oder „Wie vermeide ich es, die Durchdringung noch schlimmer zu machen?“ und schnell zeigt sich, dass die grundlegende Problematik sehr komplex ist.

Dennoch ist es ein Problem welches irgendwie gelöst werden kann und auch gelöst werden muss, gerade in realistischen Simulationen. Dies hat den Grund, da in der Realität keine Kollisionen auftreten und daher in Simulationen unangenehm auffallen. Dass heißt, die Auflösung der Durchdringung muss physikalisch plausibel sein. Demzufolge muss aber auch die Behandlung robust sein, so dass sie, wenn möglich, alle Durchdringungen erkennt und korrekt auflöst. Sie soll jedoch auch schnell

sein, da die meisten Simulationen Echtzeitsysteme sind, mit denen Benutzer arbeiten.

Das erklärt auch, weshalb es so viele Möglichkeiten gibt, Kollisionen zu erkennen und aufzulösen. Diese Ansätze sind oft in der Theorie sehr gut, versagen aber in der Praxis. Entweder haben sie so viele Sonderfälle und Limitationen, dass sie nur für ganz bestimmte Simulationstypen geeignet sind. Oder sie sind schlicht und einfach zu langsam zum Einsatz in einer Simulation. In Abschnitt 3.4 wird näher auf die einzelnen Verfahren eingegangen, welche zur Lösung des Problems in dieser Diplomarbeit in Betracht gezogen worden sind.

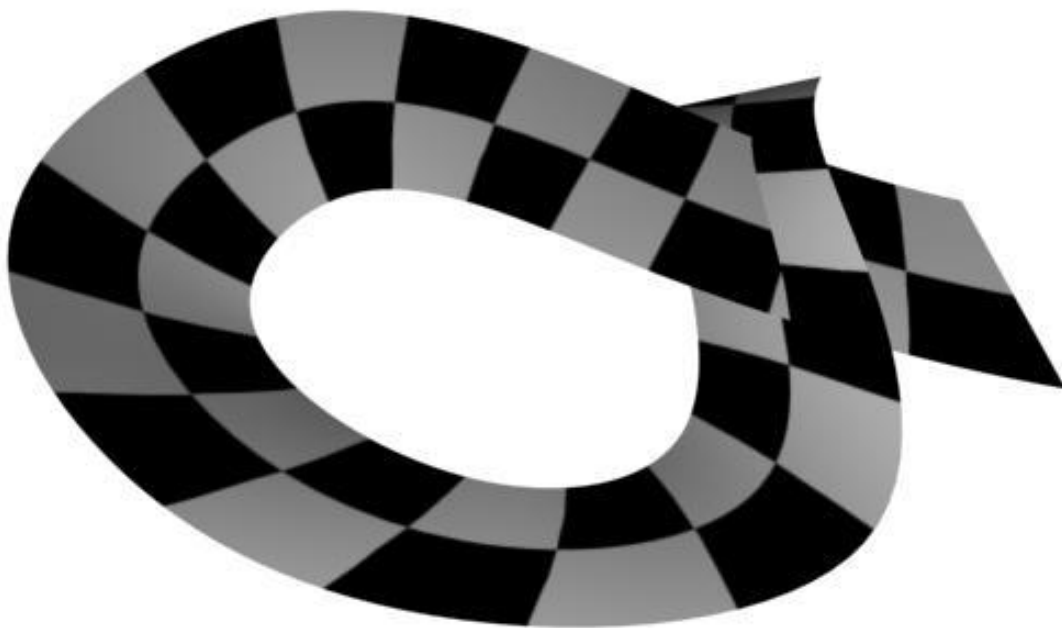


Abbildung 3-1 Schal mit Selbstdurchdringung. – Abbildung aus Metzger [2001, S. 41]

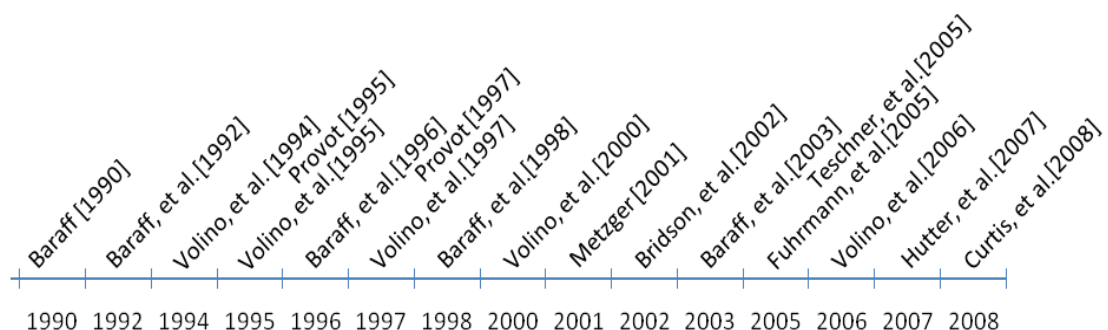


Diagramm 3-1 Historie über die Entwicklung der Kollisionserkennungssysteme für deformierbare Körper.

3.3 Einarbeitung in die Thematik

Anfangs war es essentiell, sich erst einmal einen Überblick über den Bereich zu verschaffen, welche Ansätze es schon gibt, wo die Limitationen dieser Ansätze sind und wie aktuell diese Ansätze sind, respektive ob es schon darauf aufbauende Veröffentlichungen gibt (siehe Abschnitt 3.4).

Um einmal einen Ansatzpunkt zu haben, wurde Anfangs im ersten Schritt analysiert, welches die aktuellen Kollisionsmechanismen in Vidya sind und wie diese funktionieren. Dafür war das State of the Art Paper „Collision Detection for Deformable Objects“ von Teschner, et al. [2005] und „Ontologies for Virtual Garments“ von Fuhrmann, et al. [2005] sehr hilfreich und haben einen ersten Einblick in die Thematik erlaubt, worauf anschließend aufgebaut werden konnte.

Ein weiterer wichtiger Punkt war die Erkennung von Kollisionen zur Laufzeit. Dafür wurde das bestehende BVH Verfahren (siehe Abschnitt 4.1.7) genutzt, das zurzeit schon in Vidya für andere Kollisionen genutzt wird und welches auch in „Optimized Continuous Collision Detection for Deformable Triangle Meshes“ von Hutter, et al. [2007] und „Effiziente Kollisionsdetektion in der Simulation von Textilien“ von Metzger [2001] beschrieben wird.

Während der gesamten Diplomarbeit mussten manche Themen, unter anderem bei der Berechnung einiger geometrischer Probleme, in der freien Enzyklopädie Wikipedia [2009/2010] nachgelesen werden.

3.4 Mögliche Ansätze

Da die Problematik der Selbstdurchdringung schon sehr oft Thema von wissenschaftlichen Veröffentlichungen war und untersucht wurde wie diese Problematik lösen ist, musste sich erst einmal ein Überblick über aktuelle und vielversprechende Verfahren verschafft werden. Dabei waren die Schwierigkeit a priori abzuschätzen, welches der Verfahren am viel versprechendsten ist und somit die größten Chancen auf Erfolg bietet. In Tabelle 3-1 ist eine Übersicht über die in Frage gekommenen Verfahren zu sehen und wo die Vorteile bzw. die Schwächen der einzelnen Verfahren liegen.

Ein interessanter Ansatz zur Lösung des Problems ist in Baraff, et al. [2003] beschrieben und nennt sich Global Intersection Analysis (GIA). Es soll in Verbindung mit dem sogenannten Flypapern von Partikeln dafür sorgen, dass bei Durchdringungen in besonders kritischen Situationen, wie in Abbildung 3-2 zu sehen, der Stoff nicht anfängt zu wabern. Die Situation ist per se sehr komplex, da es sein kann, dass der Avatar nicht genügend Spielraum für die Kleidung lässt. Die Idee des GIA ist dabei, die Region, welche durchdrungen wurde, durch eine Kollisionskontur zu markieren und anschließend durch geometrische Betrachtung die Innenseite dieser

Kontur zu identifizieren. Eine Kollisionskontur beschreibt einen zusammenhängenden Zug aus Kanten-Dreiecksdurchdringungen (siehe Abbildung 3-3).

Im letzten Schritt wird dann auf die beteiligten Partikel eine Interaktionskraft angewendet, welche die Durchdringung auflösen soll. Allerdings gibt es hier auch ein paar Probleme. Und zwar muss bei jedem Zeitschritt geprüft werden, ob die zuvor fixierten Partikel durch das Flypapern wieder gelöst werden können, oder ob neue Partikel fixiert werden müssen. Das ist mit der Zeit sehr aufwändig und auch anfällig für Fehler. Auch müssen beim GIA die Kollisionskonturen geschlossen sein. Es kann aber in der Simulation durchaus vorkommen, dass bei besonderen Stellen, wie z.B. ein Kragen der in ein Hemd eindringt, eine nicht geschlossene Kontur auftritt. Das Positive an diesem Verfahren ist, dass durch das Flypapern und - je nach Größe der Interaktionskraft - die Konvergenzgeschwindigkeit sehr hoch ist.

Das in Bridson, et al. [2002] beschriebene Verfahren hat sich durch den Titel im ersten Moment sehr interessant angehört, hat aber bei näherer Betrachtung einige essentielle Nachteile, weswegen es sich schnell als ungeeignet für den Einsatz in Vidya herausstellte. Zum einen ist die Laufzeit dieses Verfahrens sehr hoch. Was aber noch deutlich negativer ist, ist die Tatsache, dass es auf einer Historie basiert. Es wird bei der Berechnung des neuen Zeitschritts geprüft, ob eine Durchdringung aufgetreten ist und anschließend für die betroffenen Stellen der durchdringungsfreie Zustand t^{-1} fixiert. Das Problem dabei ist, dass es nur solange gut funktioniert solange es einen durchdringungsfreien Vorzustand gibt. Da es aber durchaus passieren kann, dass es in Vidya initiale Durchdringungen gibt, kann ein durchdringungsfreier Zustand nicht garantiert werden. Auch scheinen die nötige Datenhaltung und der Vergleich von den beiden aufeinanderfolgenden Zeitschritten sehr fehleranfällig und aufwändig zu implementieren zu sein.

Das dritte Verfahren nennt sich Intersection Contour Minimization und ist in Volino, et al. [2006] beschrieben. Dieses Verfahren greift die Idee von Baraff, et al. [2003] auf und identifiziert bei aufgetretenen Durchdringungen Kollisionskonturen. Im zweiten Schritt wird dann versucht, diese Konturen geometrisch zu verkleinern. Dabei ist es ein großer Vorteil, dass die Konturen nicht unbedingt geschlossen sein müssen. Ein weiterer Pluspunkt hierfür ist der geringe Rechenaufwand und die hohe Robustheit. So kann das Verfahren, wie erwähnt, auch mit offenen Konturen umgehen. In dem Paper von Volino, et al. [2006] werden zwei verschiedene Ansätze vorgestellt. Neben der lokalen wird auch eine globale Ausprägung des Verfahrens beschrieben. Das globale Verfahren hat zwar eine etwas höhere Laufzeit als das lokale, konvergiert aber gerade bei großen Kollisionskonturen deutlich besser und schneller als das lokale Verfahren.

Volino hat auch schon in früheren Arbeiten sich mit dem Thema auseinandergesetzt und in Volino, et al. [2000] ein Verfahren mit Namen Accurate Collision Response on Polygonal Meshes beschrieben. Dabei werden sowohl Proximities, also Annäherungen der Kleidung, als auch die Durchdringungen von deformierbaren Körpern

beschrieben. Das Paper ist jedoch schon relativ alt und es gibt aktuellere Arbeiten von Volino. Trotzdem bietet es auf den ersten Blick einige interessante Ansätze. Jedoch sind die Nachteile beim genaueren Lesen zu groß. So ist das Verfahren an sich relativ langsam. Was jedoch ein stark negatives Kriterium ist, ist die Tatsache, dass bei einer aufgetretenen Kollision der Kollisionsweg zurück-verfolgt und dann durch schrittweises Integrieren versucht wird, diese Kollision zu verhindern. Gibt es also einen Kollisionsweg der nicht zurückverfolgt werden kann oder eine initiale Durchdringung, schlägt das Verfahren fehl.

Durch das Betrachten der verschiedenen Vor- und Nachteile der einzelnen in Frage kommenden Verfahren, wurde sich für das Intersection Contour Minimization Verfahren von Volino, et al. [2006] entschiede, da es den vielversprechendsten Ansatz bietet und auch darauf hingewiesen wurde, dass es sich für komplexe Stoffsimulationen eignet in dem noch weitere Kollisionsvermeidungsverfahren aktiv sind.

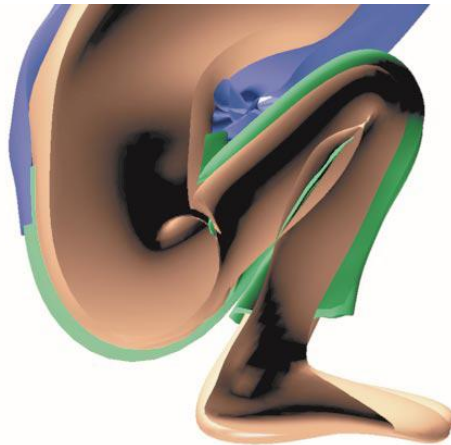


Abbildung 3-2 Besonders kritische Situation beim Auflösen von Durchdringungen, da dort viele Selbstdurchdringungen auftreten und der Avatar rel. Wenige Platz zum interagieren bietet. – Abbildung aus Baraff, et al. [2003, S. 3]

	Laufzeit	Robustheit	Limitationen	Konvergenz- geschwindig- keit	Nur durch- dringungsfreie Zustände
Intersection- Contour Minimization Global Volino, et al. [2006]	+	++	-	++	/
Intersection- Contour Minimization Lokal Volino, et al. [2006]	++	++	-	+	/
GIA and Fly- papering Baraff, et al. [2003]	-	-	-- Es müssen immer ge- schlossene Konturen vorhanden sein.	+++ Selbstkollisio- nen werden fixiert. Andere durch Kräfte in einem Zeit- schritt aufge- löst	/
Robust Treat- ment of Collisions Bridson, et al. [2002]	---	- Ist der Zeit- schritt t^{-1} ungültig schlägt das Verfahren fehl	--	/ Jeder gültige Zustand ist Durchdrin- gungsfrei	+
Accurate Col- lision Respon- se on Polygo- nal Meshes Volino, et al. [2000]	--	- Kollision wird zurückverfolgt und dann integriert	-	/ Bei starken Durchdringung mehrere Subf- rame iteratio- nen nötig	+

Tabelle 3-1 Übersicht über Vor- und Nachteile von Verfahren die näher betrachtet wurden.

3.5 Resolving Surface Collisions through Intersection Contour Minimization

3.5.1 Einleitung

Die grundlegende Idee dieses Verfahrens ist es, sich durchgedrungene Oberflächenregionen durch das berechnen relativer Verschiebevektoren durchdringungsfrei aufzulösen. Dabei wird auf die Arbeit von Baraff, et al. [2003] aufgebaut und versucht, die in diesem Paper beschriebenen Kollisionskonturen geometrisch zu verkleinern.

Es ist an sich eine robuste und intuitive Annahme, dass, wenn sukzessive alle auftretenden Kollisionskonturen versucht werden in der Länge zu minimieren, die Simulation gegen einen kollisionsfreien Zustand konvergiert.

3.5.2 Identifikation der Kollisionskonturen

Um die Kollisionskonturen überhaupt minimieren zu können, müssen im ersten Schritt aus allen aufgetretenen Kollisionen zwischen Kanten und Dreiecken zusammenhängende Kollisionskonturen erstellt werden (siehe Abbildung 3-3). Dabei gilt im Allgemeinen, je größer eine Kontur ist, desto schneller und besser kann sie aufgelöst werden. Es führt zu keinerlei Problemen, wenn Konturen unterbrochen werden. Dies hat nur zur Folge, dass anstatt einer großen Kontur zwei kleinere Konturen vorhanden sind. Doch das hat bei dem Verfahren kaum Nachteile. Lediglich die Konvergenzgeschwindigkeit bis die Durchdringung aufgelöst ist, verringert sich.

Wichtig dabei ist es, die Konturen korrekt zu identifizieren. In Abschnitt 4.1 wird auf das Erstellen der Konturen genauer eingegangen.

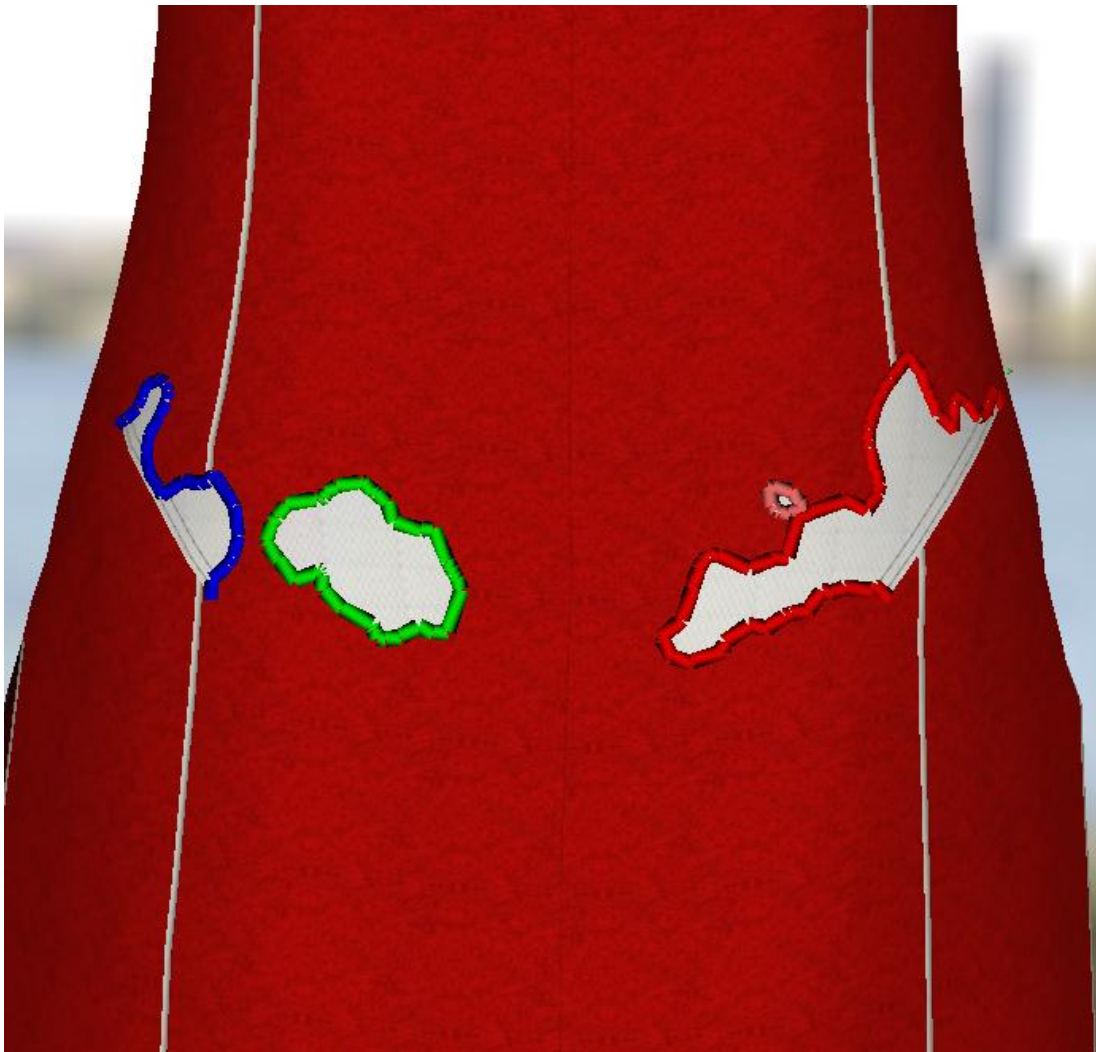


Abbildung 3-3 Starke Durchdringung zweier übereinanderliegender Kleidungsstücke. Die Kollisionskonturen markieren die einzelnen durchdrungenen Regionen.

3.5.3 Minimierung der Kollisionskonturen

Nachdem die Kollisionskonturen identifiziert sind, muss der Algorithmus für jede zusammenhängende Kontur angewendet werden. Dabei wird das bewährte Divide and Conquer Vorgehen angewendet. Es wird jede eingedrungene Kante E der Kontur erst einmal für sich alleine betrachtet. Im Normalfall gehören zu jeder Kante mehrere adjazente Polygone B_i . Befindet man sich an der Kante eines Stoffstückes, so gibt es mindestens ein Polygon welches adjazent zu E ist.

Wenn jetzt E in ein anderes Polygon A eindringt, so entstehen zwischen jedem B_i und A eine Kollisionslinie (siehe Abbildung 3-4), diese wird dann durch einen Verschiebevektor D versucht zu verkleinern. Dazu verschiebt D die Kante E relativ zum Polygon A . Dies verändert die Länge der Kollisionslinie zwischen A und B_i (siehe Abbildung 3-5).

D wird durch einen linearen Gradientenvektor G ausgedrückt.

$$G = \sum_i \left(R_i - 2 * \frac{E * R_i}{E * N} * N \right)$$

Damit G bestimmt werden kann, sind neben der Kante E noch die Normale N und der Vektor R_i nötig. N ist dabei die Normale des Polygons A und R_i repräsentiert die Richtung der Kollisionskontur. Um R_i korrekt zu bestimmen, wird das Kreuzprodukt aus den Normalen M_i , der Polygone B_i und N berechnet und anschließend normalisiert. Danach ist es wichtig R_i so auszurichten, dass der Vektor von der Kante E in Richtung des Polygons B_i zeigt. Wenn dies nicht geschieht, zeigt der resultierende Gradient genau in die falsche Richtung und die Durchdringung wird vergrößert statt verkleinern.

Damit gibt G die Richtung an in die sich E bewegen muss um sich von A zu entfernen. Hierfür sind die Richtungen von E und N irrelevant.

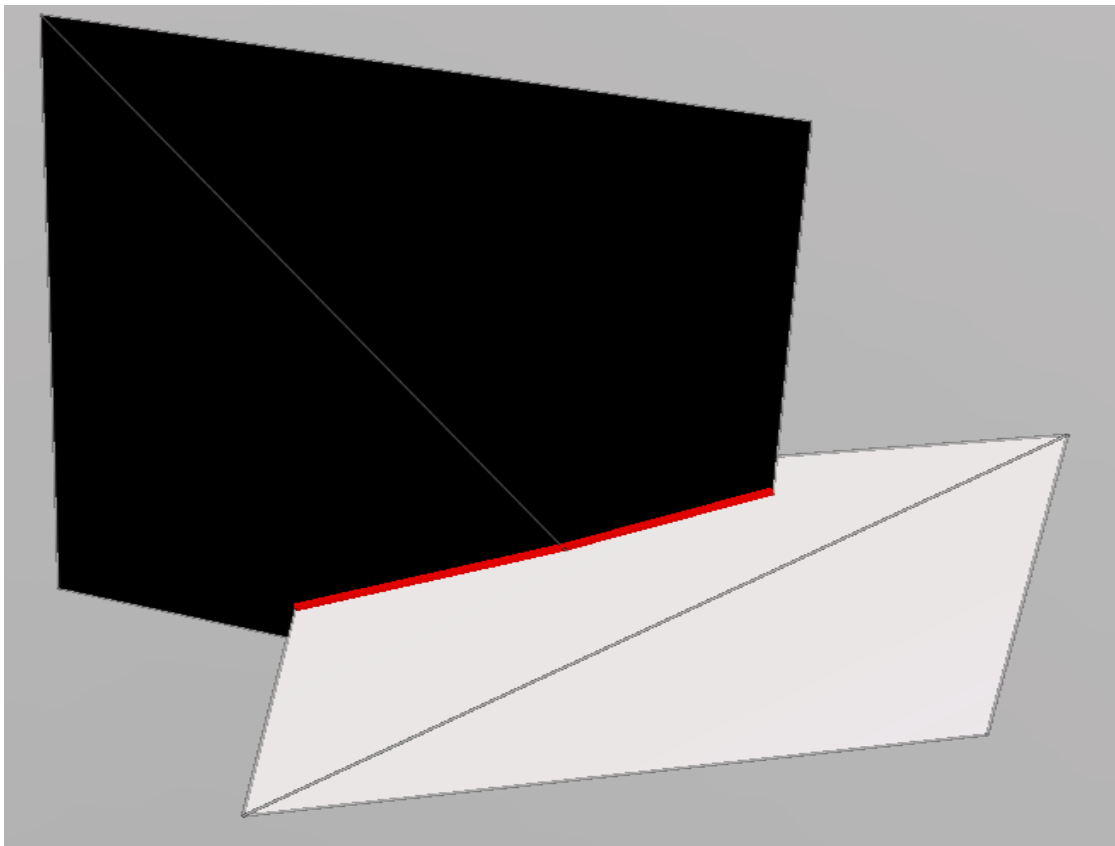


Abbildung 3-4 Schnittkante der beiden Polygone (rot).

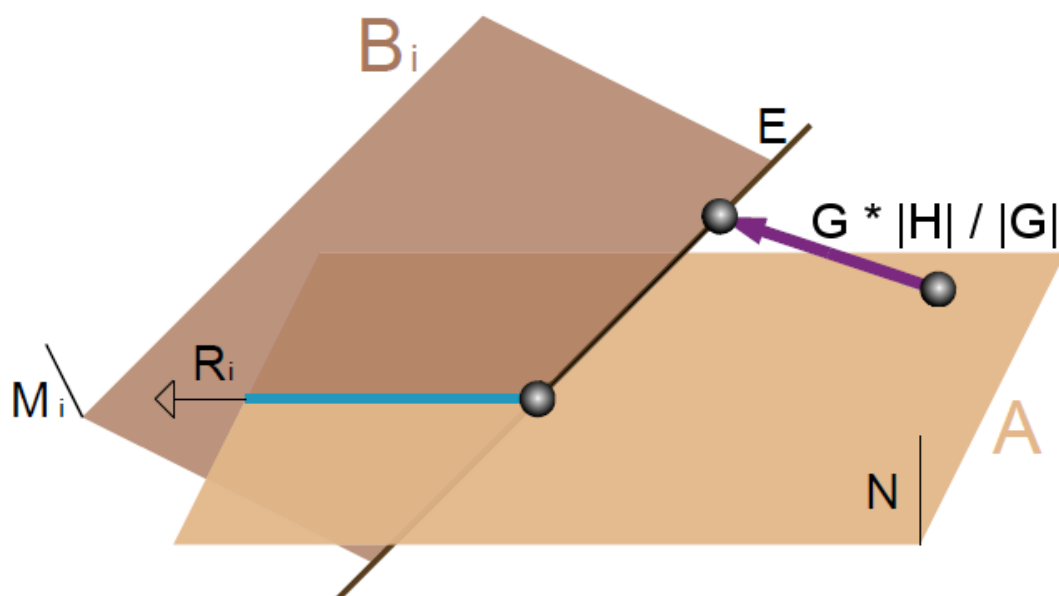


Abbildung 3-5 Skalierter Vektor G , welcher E in Richtung R_i relativ zu A verschiebt.

3.5.4 Skalierung des Gradientenvektors

Nach dem der Gradientenvektor G wie in Abschnitt 3.5.3 beschrieben berechnet wurde, muss dieser noch auf die richtige Länge skaliert werden. Wenn dies nicht geschieht, kann, je nach Koordinatensystem, die Verschiebung zu groß oder zu klein sein. Das Anpassen des Vektors ist entscheidend für die spätere Qualität. Ist der resultierende Vektor am Ende zu groß, fängt die Simulation stark an zu springen. Doch ist sie zu niedrig, dauert das Auflösen evtl. zu lange oder kann nicht gegen andere Kräfte in der Simulation ankämpfen.

Gerade weil das Bestimmen der Skalierung nicht so einfach ist, ist in Volino, et al. [2006] eine Möglichkeit beschrieben, wie G mit Hilfe eines Interaktionsvektors H skaliert werden kann.

$$|H| = h_0 * \frac{|G|}{\sqrt{|G|^2 + g_0^2}}$$

Hierbei gibt h_0 den maximalen Betrag von G an. Durch g_0 wird die Möglichkeit geboten, die Korrektur zusätzlich zu stabilisieren und es kann auch gesteuert werden, wie schnell sich $|H|$ an h_0 annähert. Am Ende wird G dann mit $|H|$ skaliert bzw. es wird zuerst $|H|$ und $|G|$ in Relation zueinander gesetzt und anschließend wird G mit diesem Skalar multipliziert um ihn anzupassen (siehe Abbildung 3-5).

3.5.5 Lokales und globales Schema

In den vorherigen Abschnitten wurde die Behandlung von einer Kanten-Polygon-Durchdringung beschrieben. Der Nachteil dabei ist, dass die Konvergenzgeschwindigkeit sehr niedrig ist und zusammenhängende Dreiecke teilweise sehr unterschiedliche Vektoren G zugewiesen bekommen. Dies wird auch als das lokale Schema bezeichnet und ist in Abbildung 3-6 gut zu sehen. Für kleine Konturen ist das lokale Schema durchaus ausreichend und obendrein noch weniger rechenintensiv als das globale Schema.

Die Idee des globalen Schemas ist hingegen, alle Vektoren einer zusammenhängenden Kontur zu addieren. Demzufolge werden alle Partikel einer zusammenhängenden Kontur in dieselbe Richtung verschoben (siehe Abbildung 3-7). Dabei ist es wichtig darauf zu achten, dass das Vorzeichen vom entsprechenden Vektoren G_i ggf. negiert werden muss, je nachdem von welcher der beiden Oberflächen ausgegangen wird. Dadurch ist gerade bei großen Konturen eine sehr schnelle Konvergenzgeschwindigkeit vorhanden und der komplette Stoff bewegt sich in dieselbe Richtung. Der einzige Nachteil ist die etwas höhere Rechenzeit, welche beim globalen Verfahren benötigt wird. Diese resultiert in erster Linie daraus, dass zusammenhängende Konturen erstellt werden müssen.

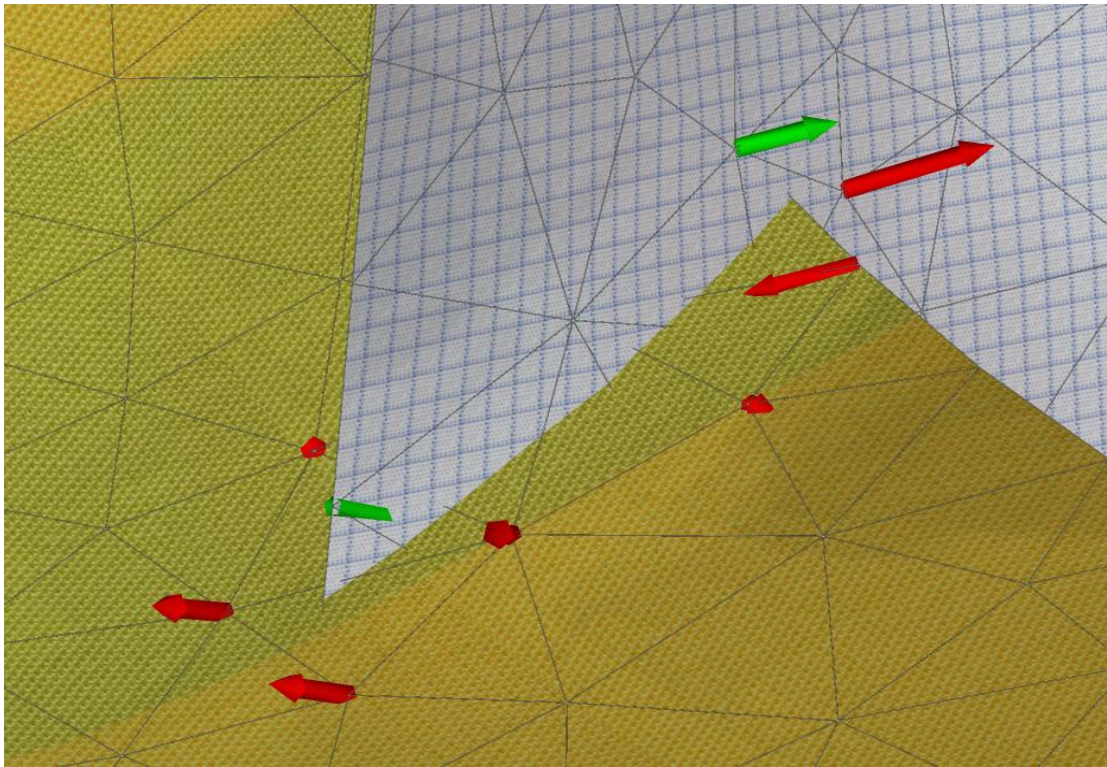


Abbildung 3-6 Verschiebevektoren nach dem lokalem Verfahren.

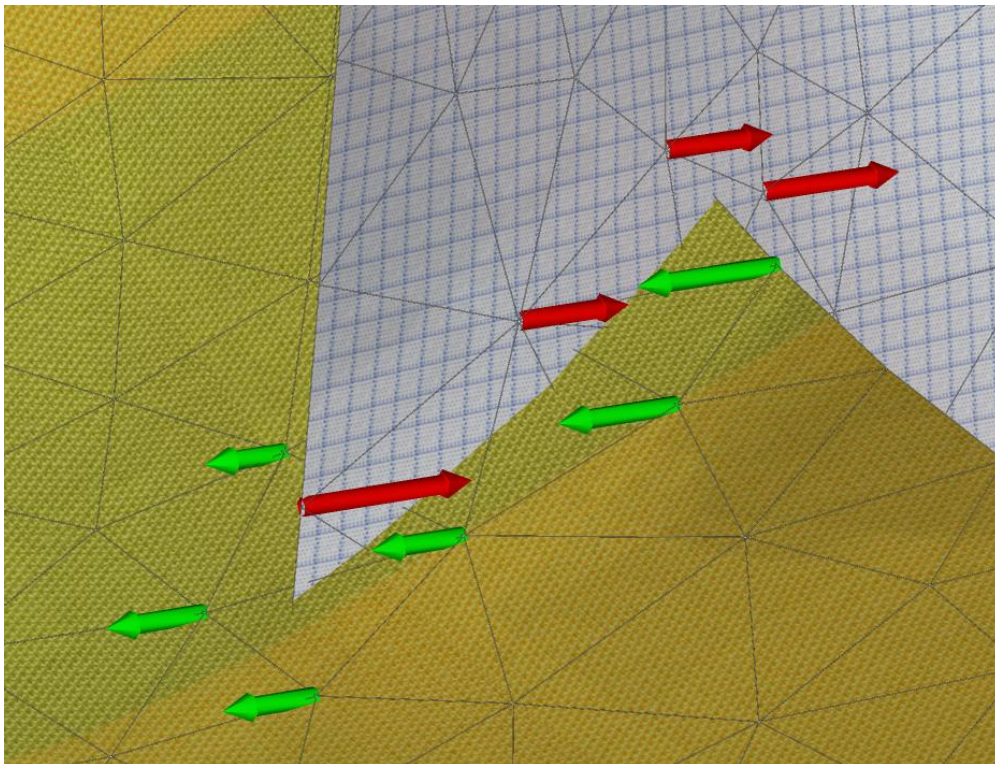


Abbildung 3-7 Verschiebevektoren nach dem globalen Verfahren.

3.5.6 Besonderheiten

Das Verfahren ist als sehr robust beschrieben und hat sich auch als solches erwiesen. So wurden bisher in Tests alle aufgetretenen Kollisionen vollständig und korrekt aufgelöst, egal wie komplex der Stoff oder das Szenario war. Zwar braucht es manchmal einige Iterationen bis alle Durchdringungen aufgelöst sind, aber im Regelfall konvergiert das Verfahren sehr schnell und löst, je nach Parametereinstellung, Durchdringungen innerhalb weniger (<100) Iterationsschritten auf. Dabei ist es egal, ob eine Kontur durch Berechnungsungenauigkeiten oder geometrische Sonderfälle unterbrochen ist. Es verringert sich lediglich die Konvergenzgeschwindigkeit.

Ein weiterer Vorteil ist, dass die eigentliche Lage im Koordinatensystem keine Rolle spielt. Genauso ist die Ausrichtung der Normalen der Dreiecke unerheblich für das Verfahren, so dass auch in sich verdrehte Stoffe korrekt aufgelöst werden können.

Bei der Geschwindigkeit ist das Verfahren auch sehr vorteilhaft für eine „echtzeitfähige“ Anwendung, da der Berechnungsaufwand sehr moderat ist. Es gibt keine komplizierten Wurzel- oder Winkelfunktionen und ebenfalls keine aufwändigen Integrale oder ähnliches. Einzig die Identifikation der Dreieckskollisionen ist bei einer unpassenden Hierarchie ein Problem.

Die Eingliederung in ein bestehendes System, gerade wenn andere Kollisionsdetektoren bereits am Laufen sind, ist auch sehr gut möglich, so dass das Zusammenspiel mit anderen Kollisionsdetektoren (siehe Kapitel 2) ohne große Anpassungen möglich ist.

3.5.7 Limitationen

Unglücklicherweise gibt es bei dem Verfahren auch ein paar Kollisionsanordnungen bei denen die Konturminimierung fehlschlägt. Einige dieser Konfigurationen sind auf Abbildung 3-8 zu sehen, diese Konfigurationen führen dazu, dass eine Verkleinerung der Kontur zu einem lokalen Minimum führt. In diesen Fällen führt eine sukzessive Minimierung der Kontur zu einem Dead End Zustand und kann nicht mehr aufgelöst werden.

Da es sich bei Vidya um ein hoch dynamisches Simulationstool handelt, sind solche speziellen Anordnungen aber eher die Ausnahme und in den Tests noch nie aufgetreten, sofern die besonderen Situationen nicht extra nachgestellt wurden. Selbst wenn versucht wurde diese Situationen nachzustellen, reicht meistens schon eine minimale Bewegung des Stoffes um aus diesem Zustand rauszukommen und das Verfahren kann ganz normal weiterarbeiten.

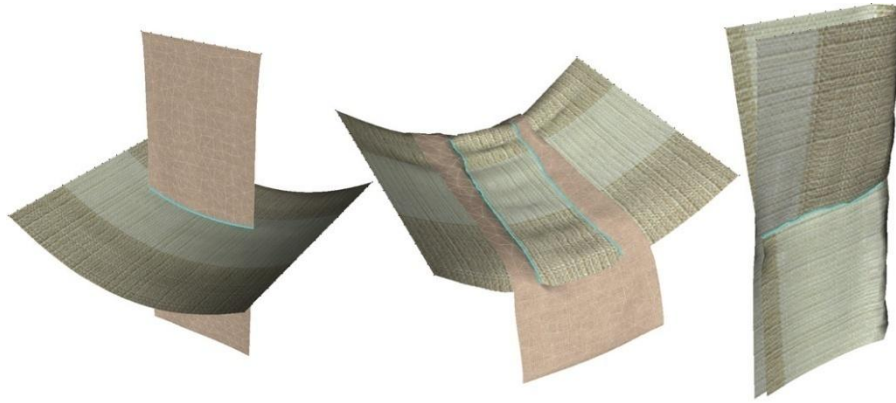


Abbildung 3-8 Limitationen der Konturminimierung - Abbildung aus Volino, et al. [2006, S. 6]

3.6 Conservation of Momentum

Die physikalische Korrektheit der Simulation soll nicht durch die Diplomarbeit verfälscht werden. Es ist zwar nicht möglich, die Kollisionsbehandlung absolut physikalisch korrekt arbeiten zu lassen, da es in der Realität gar nicht passieren kann, dass sich Stoffe durchdringen. Trotzdem musste bei der Korrektur der entsprechenden Dreiecke deren Masse und Geschwindigkeit berücksichtigt werden. Genauso spielt es auch eine Rolle, ob der Stoff in einen schweren starren Stoff eingedrungen ist oder in einen biegsamen leichten Stoff. Das heißt, es mussten nicht nur die eigenen physikalischen Merkmale berücksichtigt werden, sondern auch die des Stoffstücks, mit dem kollidiert wurde.

Das Grundproblem ist, dass sich die Szene ohnehin schon in einem physikalisch illegalen Zustand befindet, da Durchdringungen in der realen makroskopischen Welt nicht zwischen festen Körpern möglich sind. Daher muss sich grundsätzlich die Frage gestellt werden, ob es sinnvoll ist einen physikalisch illegalen Zustand physikalisch korrekt aufzulösen. Bei der Kollisionsbehandlung wird ein Interaktionsvektor, welcher den Kraftstoß darstellt, erzeugt und wirkt auf das bestehende System. Eine Idee wäre es, den Kraftstoß einfach anzuwenden und alle Partikel gleich zu bewegen. Das hätte allerdings zur Folge, dass z.B. das Stoffgewicht nicht berücksichtigt würde. Alleine dieser Effekt wäre so unplausibel für den Benutzer, dass es negativ auffallen würde.

Um den Kraftstoß korrekt auf die beteiligten Partikel aufzuteilen, mussten auch Sonderfälle wie fixierte Partikel berücksichtigt werden. Zudem hat das Anwenden von Impulserhaltung und relativen Geschwindigkeiten einen „beruhigenden“ Effekt auf die Simulation, so dass es kein flattern oder springen von Partikeln gibt, wodurch die Simulationsqualität insgesamt weiterhin auf einem sehr hohen Level bleibt. Die Grundlagen zu dieser Thematik konnten sich aus Eberly [2004] angeeignet werden.

Bei dem Verfahren von Volino, et al. [2006] sind die beiden interagierenden Primitiven Kante und Dreieck. Zwischen diesen Primitiven muss der Kraftstoß wie nachfolgend beschrieben aufgeteilt werden (siehe Abbildung 3-9).

Im ersten Schritt wird der Kraftstoß anhand der Massen der beiden Primitiven aufgeteilt. Sprich: sind die Massen der beiden Partikel der Kante genauso groß wie die Masse der drei Partikel des Dreiecks, wird der Kraftstoß zu 50% auf die Kante und zu 50% auf das Dreieck projiziert. Sind die Massen unterschiedlich, werden diese in Relation zur Gesamtmasse gesetzt und der Kraftstoß entsprechend aufgeteilt. Dabei gibt es eine Besonderheit, die berücksichtigt werden musste. Und zwar können in Vidya einzelne Partikel fixiert sein. Dadurch haben diese Partikel eine unendliche Masse. Dies würde bedeuten, dass sobald ein Partikel fixiert ist, der Kraftstoß vollständig auf das Objekt ohne fixierte Partikel angewendet wird. Das ist aber nicht gewünscht. Denn wenn in einem Dreieck ein Partikel fixiert ist, sollten sich die anderen Partikel des Dreiecks dennoch bewegen und somit auch einen Teilkraftstoß abbekommen. Dadurch wird beim Berechnen der Gesamtmassen beider Primitive kein fixierter Partikel mehr berücksichtigt, außer es sind alle Partikel eines Objektes fixiert. Denn dann geht der gesamte Kraftstoß an das andere Objekt. Wenn alle fünf beteiligten Partikel (zwei für die Kante und drei für das Dreieck) fixiert sein sollten, wird der Kraftstoß nicht angewendet und es entsteht keine Bewegung.

Im zweiten Schritt wird der jeweilige Teilkraftstoß innerhalb der Primitiven auf die nicht fixierten Partikel aufgeteilt. Dabei wird auch hier der Teilkraftstoß anhand der Massen der Partikel aufgeteilt. Jedoch kommt noch ein weiterer Faktor hinzu. Und zwar die relative Entfernung der Partikel zum Kollisionspunkt. Damit wird erreicht, dass ein Partikel der sich relativ zu den anderen Partikeln nah an einem Kollisionspunkt befindet mehr von dem Teilkraftstoß abbekommt, als weit entfernte Partikel.

Dass sich der Teilkraftstoß anhand des relativen Abstandes zum Kollisionspunkt auf die entsprechenden Partikel aufteilt, war zunächst nur eine plausible Annahme. Erhärtet wurde die Annahme dadurch, dass in Vidya bei Kante-Kante- oder Dreieck-Partikel-Kollisionen genau derselbe Mechanismus angewandt wird. Weiterhin ist in Bridson, et al. [2002, S. 4] ebenfalls eine Methode beschrieben, wie der Drehmoment die Geschwindigkeiten der beteiligten Partikel anhand der relativen Position zum Kollisionspunkt verändert.

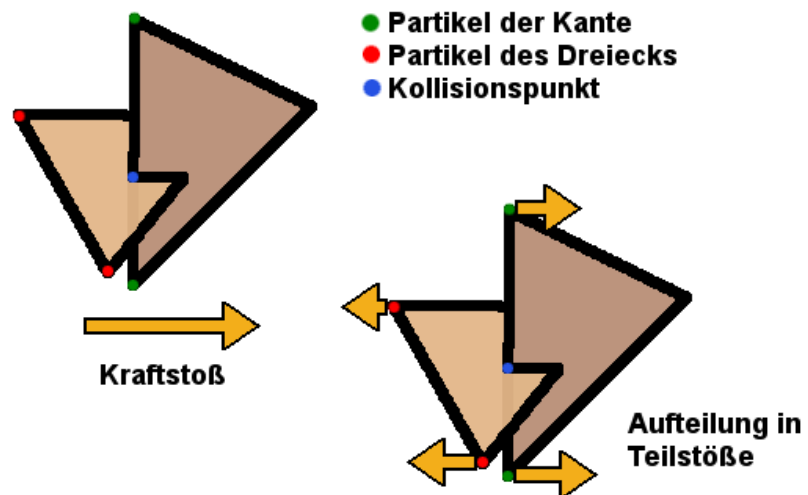


Abbildung 3-9 Aufteilung des Kraftstoßes auf die Partikel der Kante und des Dreiecks.

4 Umsetzung

Im folgenden Kapitel wird es primär um die spezielle Umsetzung des Verfahrens in Java und um die Integration in Vidya gehen. Dabei wird zuerst in Abschnitt 4.1 ein Überblick über die geschriebenen Algorithmen gegeben und anschließend in Abschnitt 4.2 der strukturelle Aufbau erklärt.

4.1 Algorithmen

4.1.1 Einleitung

In diesem Abschnitt geht es um die verwendeten und entwickelten Algorithmen, sowie einer abstrakten Darstellung wie die einzelnen Teile der Diplomarbeit ineinander greifen. Der komplette Ablauf ist schematisch auf Diagramm 4-1 zu sehen und anschließend näher unter Abschnitt 4.1.2 erläutert.

Visual Paradigm for UML Community Edition [not for commercial use]

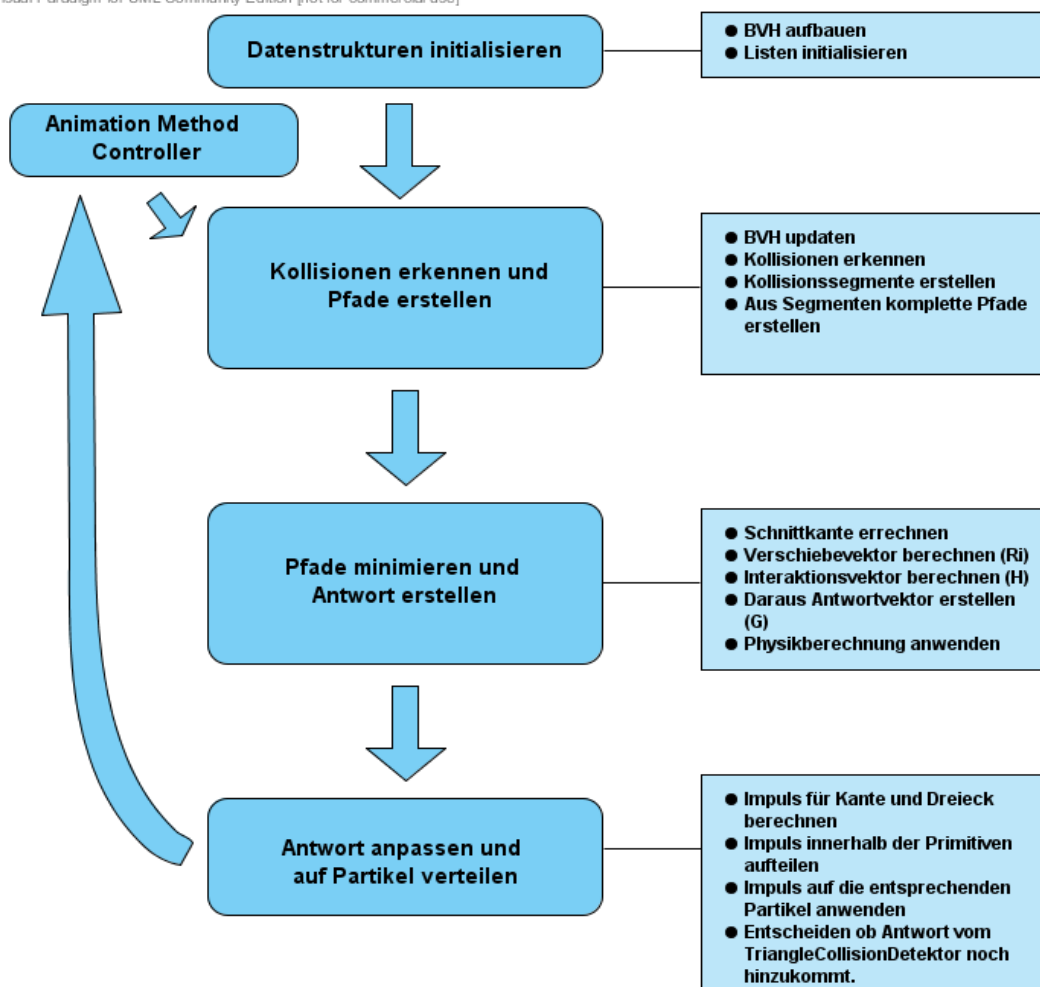


Diagramm 4-1 Schematische Darstellung des kompletten Ablaufs des implementierten Verfahrens.

4.1.2 Kompletter Ablauf

Im ersten Schritt wird beim laden von neuer Kleidung für jedes Kleidungsstück eine BoundingVolumeHierarchy (siehe Abschnitt 4.1.7) angelegt und diese korrekt mit der Geometrie der Kleidung in der Szene initialisiert.

Da das Verfahren komplett in Vidya integriert ist, wird es im aktivierten Zustand bei jedem Zeitschritt vom AnimationMethodController aufgerufen. Bei jedem Zeitschritt werden die Bounding Volumes¹ der Geometrie und die Struktur der BVHs aktualisiert. Dies ist performanter, als immer komplett neue BVHs aufzubauen. Anschließend werden die verschiedenen BVHs gegeneinander getestet und dabei auf mögliche Kollisionen von Primitiven² getestet. Um auch die Selbstdurchdringung zu erfassen wird auch jeder BVH gegen sich selber getestet. Da dies relativ aufwändig ist, gibt es Optimierungen, die unnötige Knotentests beim traversieren der Hierarchie ausschließen.

Im Falle einer wahrscheinlichen Kollision zwischen Primitiven, werden diese beiden Primitive dann nochmals in einem genauen Kollisionstest gegeneinander getestet. Sollte sich dabei herausstellen, dass sich die Primitiven wirklich durchdrungen haben, wird ein Intersection-Contour-Segment (IC-Segmente) daraus erzeugt (siehe Abschnitt 4.1.3). Für den Fall, dass keine Kollision aufgetreten ist, wird das betroffene Dreieckspaar an den TriangleSelfCollisionDetector gegeben.

Nachdem alle möglichen Kollisionen erkannt und daraus IC-Segmente erstellt wurden, wird mit Hilfe einer Spatial³ Hashtable die Rechenungenauigkeit von Float-Werten kompensiert (siehe Abschnitt 4.1.6). Denn es ist besonders wichtig, dass die Eckpunkte der Segmente absolut gleich sind. Denn nur so ist es möglich, effizient eine geschlossene Kontur aus den Segmenten zu erstellen. Für die Konturminimierung ist es nicht weiter relevant, dass die Pfade zusammenhängend sind. Denn es ist nur ein kleiner Optimierungsschritt notwendig, um möglichst große zusammenhängende Konturen zu erstellen. Dies erhöht die Konvergenzgeschwindigkeit. Es musste also abgewogen werden, ob die Rechengeschwindigkeit im Vordergrund steht und dafür Floatungenauigkeiten in Kauf genommen werden, was die Konvergenzgeschwindigkeit verringert oder umgekehrt.

Im nächsten Schritt werden aus den einzelnen IC-Segmenten komplette zusammenhängende Konturen erstellt. Dabei wird versucht die Konturen so groß wie möglich zu erstellen (siehe Abschnitt 4.1.4).

¹ **Bounding Volumes** ist in der algorithmischen Geometrie ein einfacher geometrischer Körper, der ein komplexes dreidimensionales Objekt oder einen komplexen Körper umschließt.

² **Primitive** sind in der Computergrafik elementare geometrische Formen. Aus diesen einfachen geometrischen Formen lassen sich komplexe Objekte erstellen. Im Kontext dieser Diplomarbeit sind Primitive meistens Dreiecke.

³ **Spatial** engl. für räumlich.

Nachdem aus allen IC-Segmenten Konturen erstellt wurden, werden diese Konturen einzeln nach dem Verfahren von Volino, et al. [2006] sukzessive verkleinert. Dadurch werden dann Schrittweise die Durchdringungen aufgelöst.

Nach der Berechnung des Verschiebevektors wird auf alle Kanten-Dreieckspaare die physikalische Korrektur, wie in Abschnitt 4.1.5 beschrieben, angewandt. Die Ergebnisse werden in den jeweiligen Partikeln gespeichert, da später noch Anpassungen durch die Kombination mit dem TriangleSelfCollisionDetector geschehen müssen.

Nachdem die physikalische Korrektur abgeschlossen ist, iteriert der Intersection-ContourDetector (siehe Abschnitt 4.2.2) über alle Partikel und prüft, ob diese von der Konturminimierung beeinflusst wurden. Für den Fall dass sie nicht beeinflusst wurden, wird die Antwort vom TriangleSelfCollisionDetector auf den Partikel angewandt. Sollte der Partikel allerdings beeinflusst worden sein, wird geprüft, ob die Antworten vom TriangleSelfCollisionDetector und von der Konturminimierung zusammen oder gegeneinander arbeiten. Arbeiten sie gegeneinander, wird nur die Antwort der Konturminimierung angewandt. Arbeiten sie allerdings zusammen werden beide Korrekturen angewandt.

4.1.3 Erstellen von Intersection-Contour-Segmenten

Beim Erstellen der Intersection-Contour-Segmente (IC-Segmente) wird beim durchdringen zweier Dreiecke die Schnittkante als IC-Segment identifiziert. Dabei gab es mehrere Fälle zu beachten. Unter anderem die Spezialfälle, wenn Dreieck A nur mit einem Partikel Dreieck B berührt, es also keine Kante sondern nur einen Schnittpunkt gibt. Oder eine Kante von Dreieck A genau auf einer Kante von Dreieck B liegt und wenn beide Dreiecke planar sind (siehe Abbildung 4-1).

Neben den Spezialfällen gibt es aber auch noch bei den normalen Fällen Unterscheidungen (siehe Abbildung 4-2). Es muss herausgefunden werden, ob Dreieck A mit zwei oder mit nur einer Kante Dreieck B durchdringt. Aus diesem Grund werden im ersten Schritt alle drei Kanten von Dreieck A gegen Dreieck B geprüft. Sollte hierbei nur eine oder gar keine Kollision aufgetreten sein, werden im zweiten Schritt alle Kanten von Dreieck B gegen Dreieck A geprüft. Dadurch ist sichergestellt, dass, wenn es eine Kollision gibt, auch ein gültiges IC-Segment aus zwei Kollisionspunkten erstellt werden kann.

Zusätzlich muss darauf geachtet werden, dass redundante Tests vermieden werden, da dadurch unnötiger Rechenaufwand entsteht, was gerade beim Echtzeitbetrieb zu vermeiden ist. Auch muss abgefangen werden, dass adjazente Dreiecke nicht als „kollidiert“ identifiziert werden. Da Dreiecke nicht biegsam sind, kann dieser Fall ohne Probleme ausgeschlossen werden. Denn zwei adjazente Dreiecke können sich nicht durchdringen. Im schlimmsten Fall müssen hier sechs Kanten-Dreieckstests durchgeführt werden bis feststeht, ob es tatsächlich zu einer Kollision

gekommen ist, daher ist es besonders wichtig die BVH (siehe Abschnitt 4.1.7) zu nutzen und auch redundante Tests im Vorfeld auszuschließen.

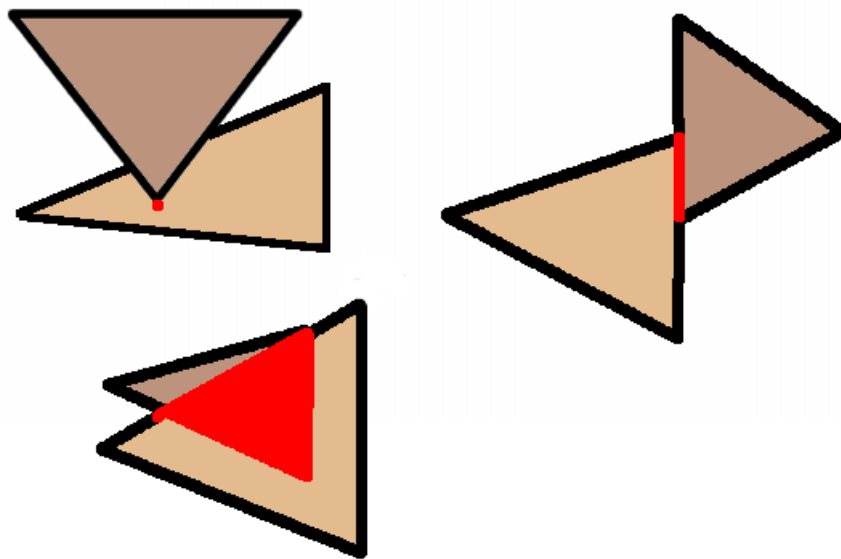


Abbildung 4-1 Sonderfälle bei der Kollision zweier Dreiecke. Oben links Berührung eines Partikels mit einem anderen Dreieck. Unten links zwei planare Dreiecke. Oben Rechts Schnittgerade genau an einer Kante.

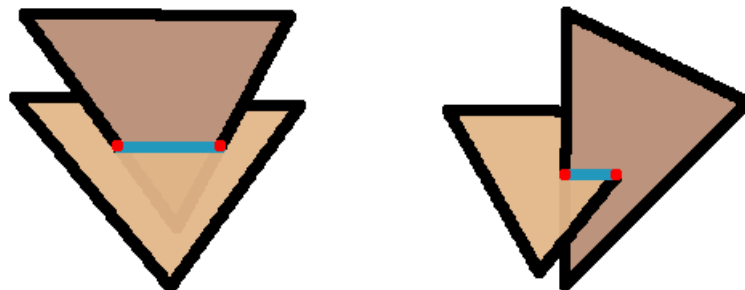


Abbildung 4-2 Normalfälle bei der Kollision zweier Dreiecke.

4.1.4 Erstellung der Intersection-Contour

Nachdem aus allen Dreieckskollisionen IC-Segmente wie in Abschnitt 4.1.3 beschrieben erstellt wurden, wird aus den einzelnen IC-Segmenten zusammenhängende Konturen erstellt. Dazu wird das erste IC-Segment aus der Liste genommen und daraus der erste der beiden Kollisionspunkte betrachtet. Anschließend wird das nächste Segment gesucht, welches diesen Kollisionspunkt ebenfalls besitzt. Das wird jetzt solange gemacht, bis kein passendes Segment mehr auffindbar ist. Anschließend wird der andere Kollisionspunkt des Startsegments genommen und die Kontur in die andere Richtung weitergeführt. Alle zugeordneten Segmente werden dabei aus der globalen Liste direkt entfernt, wodurch es keine Doppelverwendungen geben kann.

Dies wird solange gemacht, bis alle IC-Segmente zugeordnet sind. Dabei können natürlich auch kleine Konturen entstehen die nur aus einem IC-Segment bestehen.

Zur Darstellung einer Kontur gibt es zwei Listen. Die eine repräsentiert die Kontur auf der eindringenden Oberfläche. Die andere Liste die Oberfläche in die eingedrungen wurde. Hier kommt die Struktur der `IntersectionContourSegmentList` zur Geltung, da die Segmente anhand ihrer Kollisionspunkte sortiert sind. Da zum identifizieren einer kompletten Kontur ebendiese Kollisionspunkt für Kollisionspunkt entlangehandelt werden muss, also IC-Segment für IC-Segment, kann einfach bei jedem Einfügen eines neuen IC-Segments geschaut werden, ob eine der beiden Kanten in dem Segment adjazent zu einer bereits eingetragenen Kante in einer der beiden Listen ist. Dieser Ansatz kann sowohl für die Selbstkollision, als auch für die Kollision zwischen mehreren Kleidungsstücken verwendet werden.

Interessant ist, dass die Konturen nur vom globalen Verfahren benötigt werden. Und dort ist es auch nicht so wichtig, dass die Konturen wirklich ohne Lücken identifiziert wurden. Das Verfahren an sich ist so robust, dass es auch gute Ergebnisse liefert, wenn die Konturen unterbrochen sind. Es ist zwar Wert darauf gelegt worden, Anomalien zu beseitigen, dennoch ist es gut zu wissen, dass nicht zu viel Aufwand investieren werden muss, um diese Anomalien zu beachten. Lediglich die Konvergenzgeschwindigkeit sinkt, wenn Konturen unterbrochen sind. Im schlimmsten Fall würde es sich so wie das lokale Verfahren verhalten, welches bei kleinen Durchdringungen kaum schlechter arbeitet als das globale Verfahren.

4.1.5 Einfluss der Massen und Positionen

Die physikalische Behandlung wurde genau so umgesetzt wie in Abschnitt 3.6 beschrieben. Zwar sind in Vidya Berechnungen für Kraftstöße zwischen zwei Primitiven enthalten, allerdings beschränken diese sich auf die Fälle „Partikel gegen Dreieck“ und „Kante gegen Kante,..“. Der Sonderfall Kante-Dreieck musste daher neu implementiert werden.

Dabei sind die Massen und die relative Position des Kollisionspunktes zu den einzelnen Partikeln ausschlaggebend und wurden folgendermaßen berechnet.

Im ersten Schritt wurde der gesamte Kraftstoß anhand der Masse der beiden Primitiven gesplittet. Dabei wurde mit inversen Massen gearbeitet. Dieses hat den Vorteil, dass, falls ein Partikel fixiert ist, dieser die invertierte Masse 0 hat und nicht in die Berechnung mit einfließt.

Zur Aufteilung des Kraftstoßes auf die beiden beteiligten Primitiven, wurde im ersten Schritt die gesamte inertierte Masse berechnet:

$$iM_{gesamt} = iM_{edge} + iM_{triangle}$$

Anschließend wird ein Skalierungsfaktor berechnet, mit dem sich der Kraftstoß aufteilen lässt:

$$sE = \frac{iM_{edge}}{iM_{gesamt}}$$

$$sT = \frac{iM_{triangle}}{iM_{gesamt}}$$

Mit diesen Faktoren wird dann der Kraftstoß auf die jeweiligen Primitive aufgeteilt. Wenn einer der beiden Primitiven komplett fixiert ist, geht der gesamte Kraftstoß an das andere Primitiv. Für den Fall, dass beide fixiert sind, wird der Kraftstoß nicht berücksichtigt.

Nachdem der Kraftstoß auf die beiden Primitive aufgeteilt wurde, wird der Kraftstoß noch innerhalb des Primitiv auf die entsprechenden Partikel aufgeteilt. Die Massen werden dort analog zur Verteilung zwischen den Primitiven berechnet.

Das Besondere beim Verteilen des Kraftstoßes innerhalb eines Primitives ist, dass hier noch die relative Position der Partikel zum Schnittpunkt (iP) mit berücksichtigt wird. Hierzu wird im ersten Schritt der quadratische Abstand zwischen Partikel und Kollisionspunkt berechnet:

$$dP_x = \text{sqrtDist}(P_x - iP)$$

Es wird mit dem quadratischen Abstand gerechnet um Rechenzeit zu sparen, da keine Wurzel gezogen werden muss. Wenn alle Distanzen berechnet wurden, wird im nächsten Schritt die Gesamtdistanz berechnet:

$$dE_{gesamt} = dP_0 + dP_1$$

$$dT_{gesamt} = dP_0 + dP_1 + dP_2$$

Nachdem die Gesamtdistanz und die Einzeldistanzen berechnet wurden, werden diese noch in Relation gesetzt, um einen entsprechenden Skalierungsfaktor zu berechnen:

$$sD_{edge} P_0 = \frac{dP_1}{dE_{gesamt}}$$

$$sD_{edge} P_1 = \frac{dP_2}{dE_{gesamt}}$$

$$sD_{triangle} P_0 = \frac{dP_1 + dP_2}{dT_{gesamt}} * 0.5$$

$$sD_{triangle} P_1 = \frac{dP_0 + dP_2}{dT_{gesamt}} * 0.5$$

$$sD_{triangle} P_2 = \frac{dP_0 + dP_0}{dT_{gesamt}} * 0.5$$

Nachdem die Faktoren für die Masse und die relative Distanz der einzelnen Partikel berechnet wurden, werden diese Faktoren addiert und der jeweilige Teilstoß innerhalb des Primitives damit skaliert:

$$E_{momentum} P_x = E_{momentum} * (sM_{edge} P_x + sD_{edge} P_x) * 0.5$$

$$T_{momentum} P_x = T_{momentum} * (sM_{triangle} P_x + sD_{triangle} P_x) * 0.5$$

Bei der Berechnung des Teilstoßes innerhalb der Primitiven werden auch fixierte Partikel berücksichtigt. Sollte ein, oder im Falle eines Dreiecks zwei, Partikel fixiert sein, wird der Kraftstoß zwischen den nicht fixierten Partikeln entsprechend aufgeteilt.

Durch diese Aufteilung des gesamten Kraftstoßes auf die beiden beteiligten Primitiven, lässt sich ein einfaches Modell von der Impulserhaltung anwenden, was die Simulationsqualität deutlich steigert. Abbildung 4-3 sind die die Antwortvektoren ohne physikalische Korrektur zu sehen. Jeder beteiligte Partikel bekommt dieselbe Antwort, unabhängig von Masse und Abstand zum Kollisionspunkt. In Abbildung 4-4 ist dieselbe Szene mit physikalischer Korrektur. Hier wird auch sichtbar, dass das karierte Stoffstück deutlich schwerer ist als das andere. Aus diesem Grund wirkt die Kollisionsantwort deutlich schwächer auf das karierte Stoffstück.

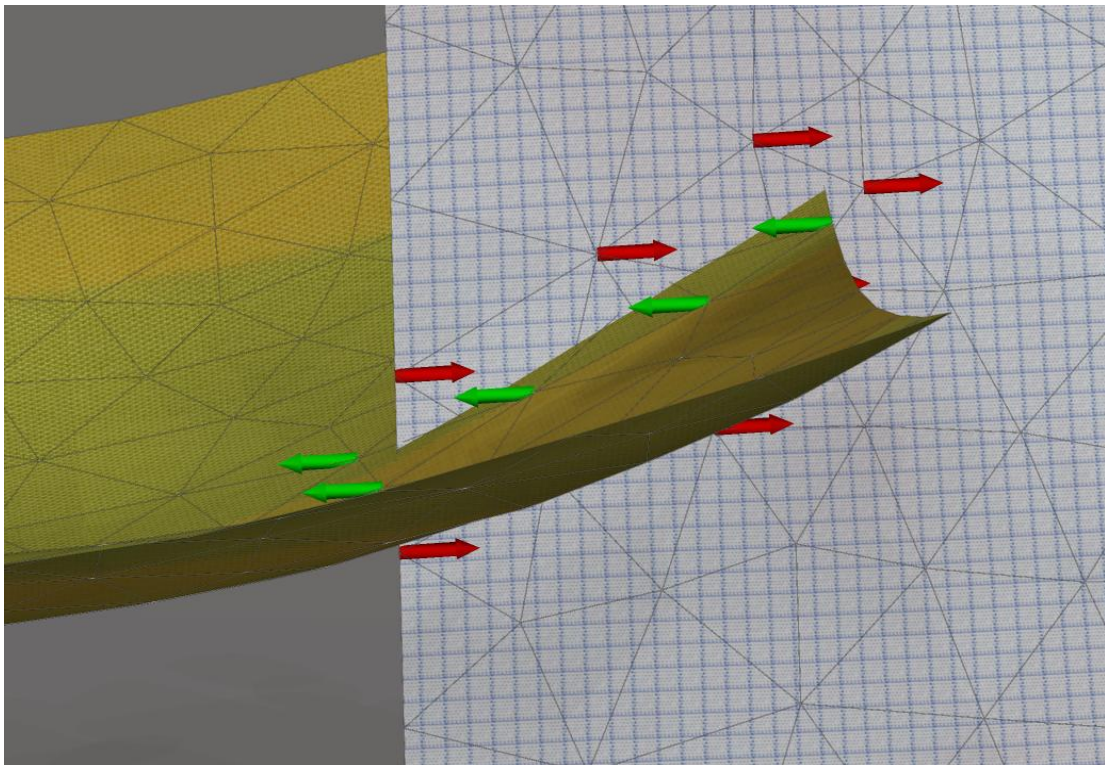


Abbildung 4-3 Antwortvektoren ohne physikalische Korrektur. Die Antwort für jeden beteiligten Partikel ist gleich obwohl das karierte Stoffstück (rechts) etwa 6-mal schwerer ist.

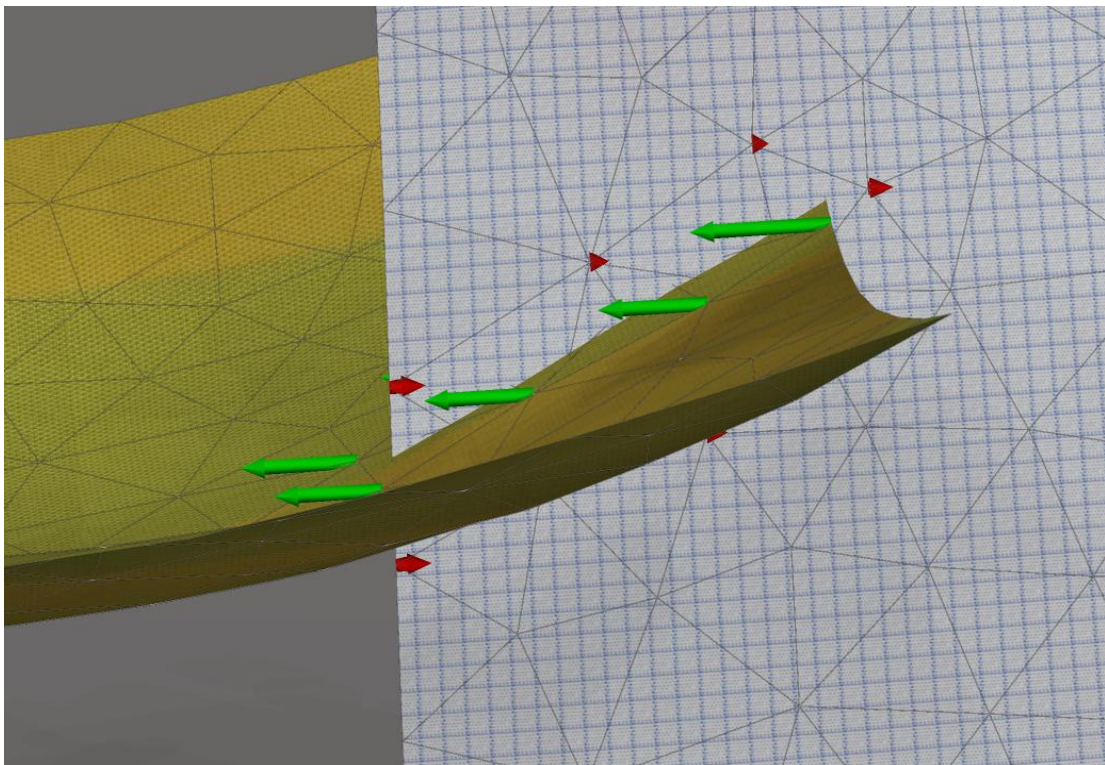


Abbildung 4-4 Antwortvektoren mit physikalischer Korrektur. Das karierte Stoffstück (rechts) bekommt eine deutlich schwächere Kollisionsantwort, da etwa 6-mal schwerer ist.

4.1.6 Spatial Hashtable

Jedes IC-Segment besteht aus zwei Kollisionspunkten. Durch diese Punkte lässt sich eine komplette zusammenhängende Kontur identifizieren. Das liegt einfach daran, dass zwei adjazente Dreiecke eine gemeinsame Kante haben und dadurch auch den exakt gleichen Kollisionspunkt auf dem gegenüberliegenden Dreieck.

Zumindest sollten sie exakt gleich sein, wenn es keine Rechenungenauigkeit geben würde. Da diese aber nicht auszuschließen ist und die Kollisionspunkte als Schlüssel fungieren, muss sichergestellt sein, dass es exakt dieselben Punkte sind.

Um das zu gewährleisten, werden alle Kollisionspunkte in eine Spatial Hashtable, also räumliche Hashtabelle, hinzugefügt und anschließend alle Kollisionspunkte, die sehr nah beieinander sind, in einer speziellen Epsilonumgebung, im Rahmen der Floatungenauigkeit, miteinander gemittelt (siehe Abbildung 4-5). Dann wird dieser Kollisionspunkt in den betroffenen IC-Segmenten aktualisiert. Dadurch ist sichergestellt, dass zwei angrenzende Dreiecke den exakt selben Kollisionspunkt haben und dieser als Schlüssel genommen werden kann.

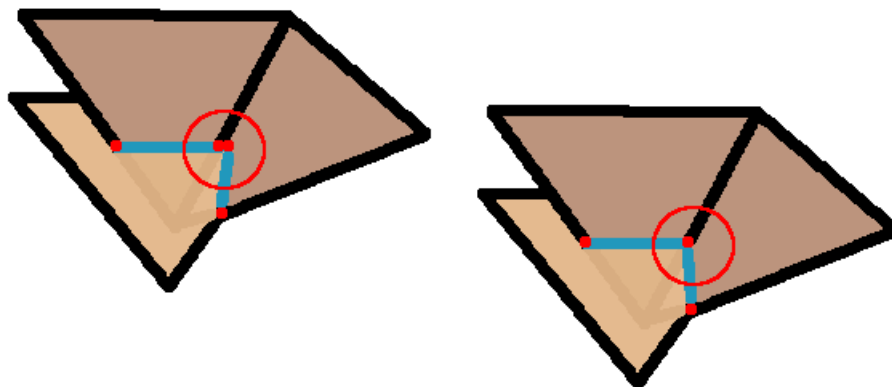


Abbildung 4-5 Anpassung der Kollisionspunkte durch die SpatialHashMap um Floatungenauigkeit auszugleichen.

4.1.7 Bounding Volume Hierachy (BVH)

Gerade bei der Prüfung von auftretenden Kollisionen ist es sehr aufwändig, jedes Dreieck gegen jedes andere zu testen. Vor allem, da in Vidya Kleidungsstücke schon mal mehrere 1000 Dreiecke haben können und es somit mehrere Millionen Tests wären, die für jeden Zeitschritt ausgeführt werden müssten. Das würde das Arbeiten mit Vidya unmöglich machen. So ist es nötig, das Dreiecksgitter in einer geeigneten Hierarchie zu unterteilen, so dass sehr schnell sehr viele unnötige Dreieckstests ausgeschlossen werden können und am Ende nur noch wenige Dreiecke gegeneinander testen werden müssen.

Jetzt gibt es mehrere Möglichkeiten solche Strukturen aufzubauen. Eine davon ist die BVH. Diese unterteilt das Dreiecksmesh in mehrere Bounding Volumes und sortiert diese dann in eine Hierarchie ein, um anschließend sehr effizient unnötige Dreieckstests auszuschließen.

Dabei wird als Bounding Volume der sogenannte k-Dop verwendet, siehe Metzger [2001, S. 25]. Hierbei wird das Objekt in mehrere konvexe Polyeder unterteilt. Dies hat den Vorteil, dass die Objekte sehr gut abgebildet werden können und „k-Dop-gegen-k-Dop“-Tests relativ wenig Rechenzeit benötigen. Allerdings müssen die k-Dops und auch die Hierarchie bei jedem Zeitschritt aktualisiert werden, damit diese immer auf den aktuellen Daten arbeiten.

Durch die BVH ist es möglich, sehr effizient alle möglichen Kollisionen zu identifizieren, welche dann anschließend genau gegeneinander geprüft werden. Unglücklicherweise bringt die BVH bei der Identifikation von Selbstkollisionen kaum einen Vorteil, da dabei eine BVH gegen sich selber getestet wird. Und dies führt dazu, dass so gut wie alle Dreieckstests durchgeführt werden müssen. Aus diesem Grunde wurde die BVH in Vidya soweit angepasst, dass bei einer Selbstkollision zumindest die inneren Knotentests wegfallen, da diese bei einem Selbsttest eines BVHs unnötig sind.

Mit dieser Struktur war es möglich, Selbstkollisionen auch zur Laufzeit zu behandeln, ohne dass die Geschwindigkeit der Simulation zu stark einbricht.

4.2 Implementierung

4.2.1 Flussdiagramm

In diesem Abschnitt ist in Diagramm 4-2 der Ablauf des kompletten Verfahrens vereinfacht grafisch dargestellt. Hierbei sind auch die beiden Programnteile gut zu sehen, welche sich bei komplexeren Szenen negativ auf die Laufzeit auswirken. Zum einen ist die Schleife im oberen Abschnitt, in der durch die BVH traversiert wird. Zum anderen ist es am Ende des Ablaufs die Schleife über alle Partikel, um deren Position zu aktualisieren.

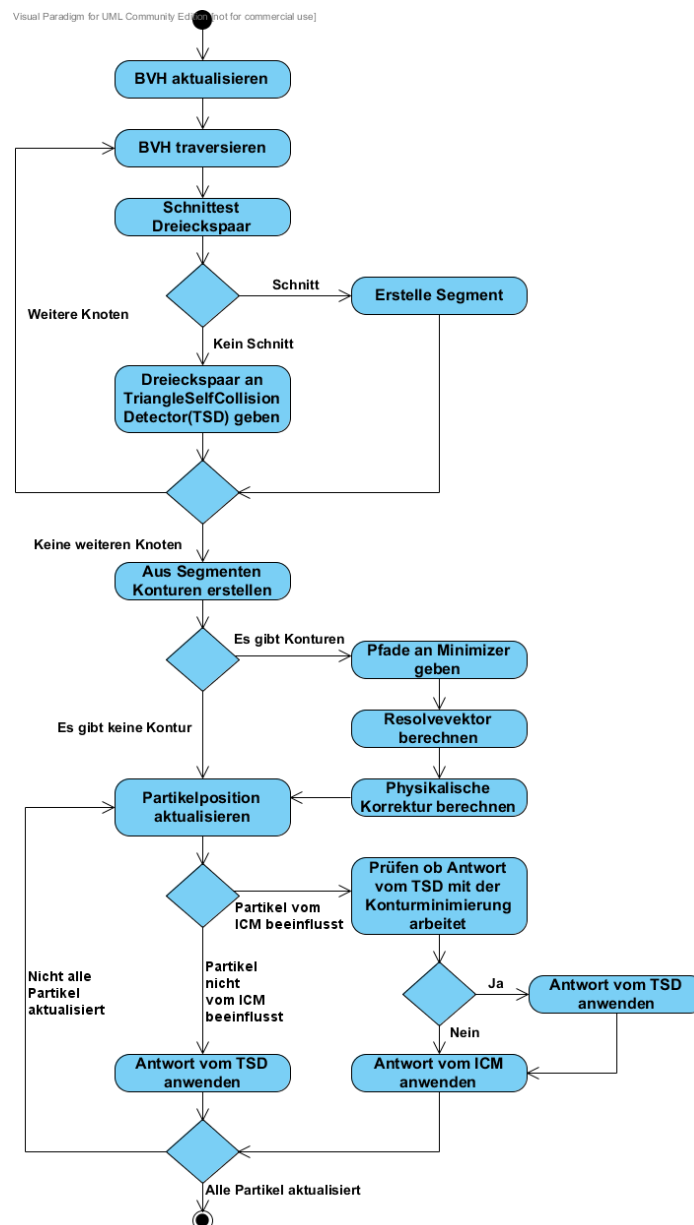


Diagramm 4-2 Flussdiagramm über den kompletten Ablauf der Konturminimierung.

4.2.2 Klassendiagramm

Diagramm 4-3 zeigt den Aufbau und die Verbindung der einzelnen Klassen untereinander. Wie auf dem Klassendiagramm zu erkennen ist, ist die komplette Diplomarbeit ein eigenständiges Modul, welches nur über eine Schnittstelle vom AnimationMethodController angesprochen wird. Aus diesem Grund ist das aktivieren oder deaktivieren der Konturminimierung selbst zur Laufzeit ohne Probleme möglich und könnte auch ohne Probleme aus Vidya genommen oder ersetzt werden.

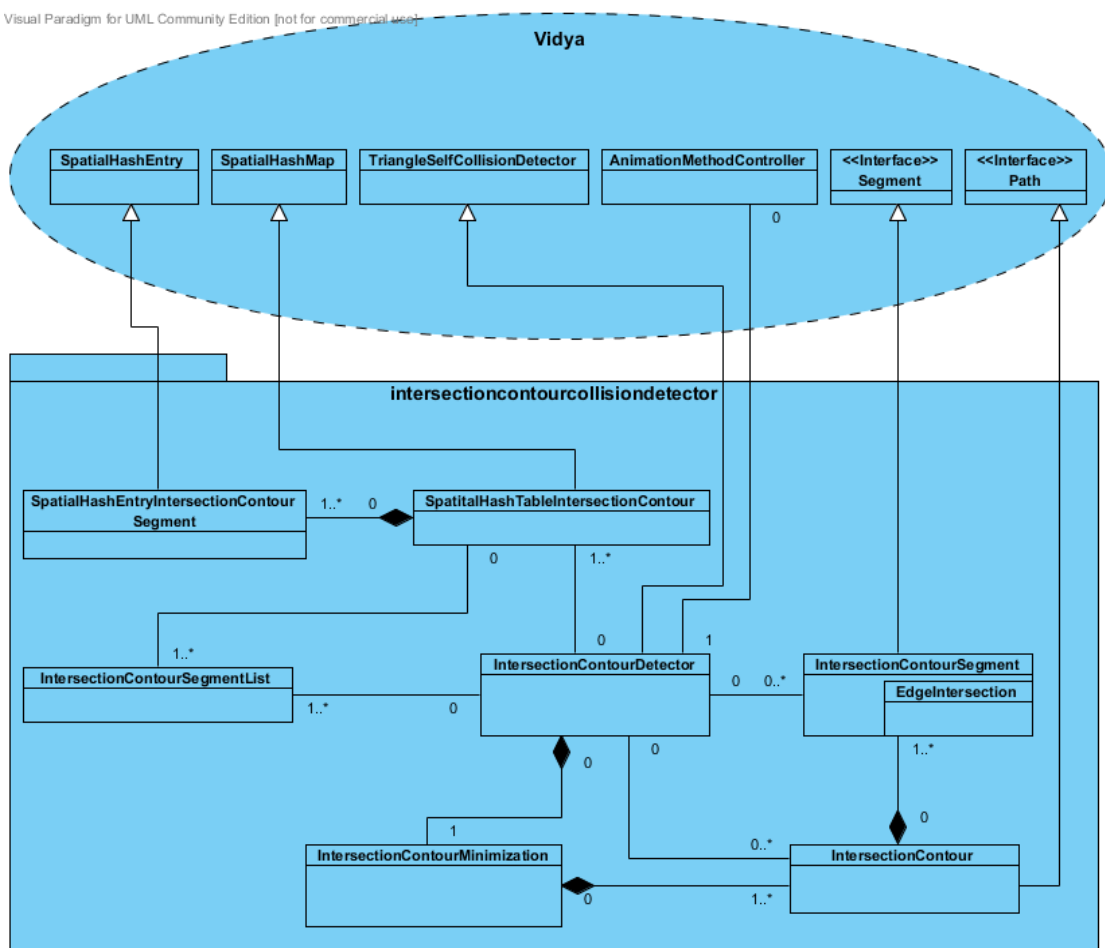
Der zentrale Einstiegspunkt ist der IntersectionContourCreator, welcher zum aktualisieren der Datenstrukturen und zum Erkennen von Kollisionen zuständig ist. Anschließend werden in den erkannten Kollisionssegmenten die Schnittpunkte mit Hilfe der SpatialHashTableIntersectionContour und dem SpatialHashEntryIntersectionContourSegment angeglichen, um evtl. Floatungenauigkeiten zu beseitigen. Mit der daraus erstellten IntersectionContourSegmentList, werden dann im letzten Schritt die kompletten Konturen erstellt.

Anschließend werden die Konturen an den IntersectionContourMinimization weitergegeben. Dort geschieht dann die Kernaufgabe: das Minimieren der Konturen anhand des Verfahrens von Volino, et al. [2006]. Anschließend wird der resultierende Vektor noch skaliert, physikalisch angepasst und die Antwort innerhalb des jeweiligen Partikels gespeichert.

Danach iteriert der IntersectionContourDetector über alle Partikel, um die neue Position zu setzen. Dabei wird geprüft ob die Antwort vom TriangleSelfCollisionDetector gegen oder mit der Antwort von der Konturminimierung arbeitet. Falls sie gegeneinander arbeiten, wird die Antwort vom TriangleSelfCollisionDetector nicht angewandt.

Anschließend geht es wieder zurück zum AnimationMethodController.

Visual Paradigm for UML Community Edition [not for commercial use]



Legende



:Zeigt eine notwendige Abhängigkeit.



:Beschreibt eine einfache Beziehung.

0

:Diese Klasse ist von der verbundenen Klasse aus nicht sichtbar.

1

:Es gibt genau eine Instanz der Klasse, die von der verbundenen Klasse sichtbar ist.

0..*

:Es gibt keine oder beliebig viele Instanzen, die von der verbundenen Klasse sichtbar sind.

1..*

:Es gibt mindestens eine oder beliebig viele Instanzen, welche von der verbundenen Klasse sichtbar sind.

Diagramm 4-3 Klassendiagramm aller Klassen die für die Diplomarbeit verwendet wurden.

4.2.3 Datenstrukturen

4.2.3.1 IntersectionContourSegmentList

Bei dieser speziellen Liste handelt es sich um eine Relation zwischen einem Punkt im 3D, in diesem Fall dem Kollisionspunkt, und den zugeordneten IC-Segmenten. Diese Zuordnung ist zum identifizieren zusammenhängender Konturen sehr hilfreich und wichtig. Anhand der Kollisionspunkte wird dann das nächste passende IC-Segment gesucht, welches dann logischerweise an das aktuelle Segment angrenzt. Dabei wird jedes gefundene Element aus der Liste entfernt. Und zwar solange, bis

die Liste leer ist. Hierdurch können sehr schnell und robust die Kollisionskonturen erstellt werden.

Der reine Ablauf ist verhältnismäßig trivial. Es wird geschaut, welcher der beiden Kollisionspunkte des aktuellen IC-Segments dem entsprechenden Schlüssel entspricht. Anschließend wird der zweite Kollisionspunkt genommen. Nach diesem wird dann in der Liste gesucht. Das passiert solange, bis kein passender Eintrag mehr zum zweiten Kollisionspunkt gefunden wird. Das bedeutet entweder, dass es sich um eine offene Kontur handelt und der Algorithmus an einem Eckpunkt angelangt ist oder es sich um eine geschlossene Kontur handelt und der Algorithmus einmal rund ist. In diesem Fall findet der Algorithmus sein Startsegment nicht mehr, da dieses schon aus der Liste entfernt wurde.

4.2.3.2 SpatialHashEntryIntersectionContourSegment

Diese Struktur ist eine Spezialisierung der SpatialHashEntry Klasse aus Vidya. Im Normalfall werden dort einfach nur die enthaltenen Koordinaten aller nah beieinander liegenden Einträge betrachtet und gemittelt. Im aktuellen Fall allerdings fungieren diese Koordinaten im IntersectionContourDetector als Schlüssel, um zusammenhängende Konturen zu identifizieren. Dies muss berücksichtigt werden, da ein Verändern des Schlüssels dazu führt, dass im Anschluss nicht mehr von einem IC-Segment auf das nächste per Kollisionspunktkoordinaten geschlossen werden kann.

Aus diesem Grund beinhaltet diese Ableitung nicht nur die Koordinaten, sondern auch alle zugehörigen IC-Segmente. Dies ist wichtig, da diese nach dem Mitteln von den Koordinaten angepasst werden müssen. Die Methode, um die Koordinaten zu mitteln, wird dabei aus der Vaterklasse genutzt und in der Ableitung noch um einen Nachbearbeitungsschritt erweitert. So wird anschließend zu jedem der beiden Kollisionspunkte in einem IC-Segment eine quadratische Differenz gezogen und geprüft, zu welchem der beiden ursprünglichen Kollisionspunkte die Differenz kleiner ist. Dabei gibt die räumliche Differenz einen guten Indikator ab, welcher ursprüngliche Kollisionspunkt angepasst wurde.

Im letzten Schritt wird dann der Kollisionspunkt mit der geringsten Differenz durch den neuen gemittelten Kollisionspunkt ersetzt. Da dies in allen zugeordneten IC-Segmenten geschieht, sind diese danach eindeutig identifizierbar. Dadurch werden die zusammenhängenden Konturen identifiziert.

4.2.3.3 IntersectionContour

Bei der IntersectionContour handelt es sich um einen Container, der eine zusammenhängende Kontur repräsentiert. Dabei enthält dieser Container alle einzelnen Kanten-Kollisionen, die zu dieser Kontur gehören. Es ist wichtig, dass er die übergebenen IC-Segmente aufbricht und dort die Kanten-Kollisionen extrahiert, da beim hinzufügen von einem IC-Segment direkt überprüft wird, zu welchem Kleidungsstück oder, im Falle einer Selbstkollision, zu welchem Polygonzug diese Kante gehört.

Diese Auftrennung ist später für die Anwendung des globalen Verfahrens essentiell, da über die Zuordnung entschieden wird, ob das Vorzeichen beim Addieren der einzelnen Verschiebevektoren negiert werden muss oder nicht. Auch wird durch die Zuordnung entschieden, ob der globale Verschiebevektor nach der Aufsummierung negiert oder nicht negiert auf die Kanten-Dreieck-Paare angewendet werden muss.

Zusätzlich bietet diese Containerklasse noch viele nützliche Komfortfunktionen, die gerade beim weiteren verarbeiten der Daten den Code lesbarer und auch einfacher machen.

4.2.4 Schwerpunkt bei der Umsetzung

Der Schwerpunkt während der ganzen Umsetzung lag darauf, die Diplomarbeit nach der erfolgreichen Fertigstellung in Vidya zu integrieren. Das Ziel war also nicht, einen reinen Prototypen zu entwickeln, sondern ein fertiges Modul. Dieses Modul hat als Ziel, mit möglichst wenigen Anpassungen in die aktuelle Version von Vidya integriert zu werden. Dazu war es nötig, während der ganzen Entwicklung durch das Versionisierungstool SVN, immer wieder sogenannte MergeBacks auf die aktuelle Vidya Version im Trunk zu machen. Dadurch wurden alle Änderungen an der aktuellen Vidya Version in den jeweiligen Branch übernommen und es konnten im Vorfeld Konflikte zwischen den verschiedenen Versionen behoben werden. Weiterhin konnte dadurch auch direkt geprüft werden, ob die Konturminimierung irgendwelche Seiteneffekte auf andere Funktionen in Vidya hat oder ob irgendwelche neuen Entwicklungen in der produktiven Version Einfluss auf die Konturminimierung haben.

Zwar wurde die Konturminimierung anfangs prototypenhaft umgesetzt, um zu prüfen ob der Ansatz grundsätzlich funktioniert. Nachdem aber feststand, dass mit der Konturminimierung weiterhin gearbeitet werden kann und der Ansatz erfolgsversprechend war, wurde über mehrere Iterationen der Quellcode vereinfacht, restrukturiert und refaktoriert.

Auch die ausführliche Kommentierung des Codes und das Einhalten vom Codestyle waren wichtige Punkte, die zum späteren Integrieren und Warten des Moduls unabdingbar sind.

5 Erweiterung des Verfahrens

In diesem Kapitel wird beschrieben, an welchen Stellen das Verfahren von Volino, et al. [2006] angepasst bzw. erweitert wurde, um es den entsprechenden Gegebenheiten in Vidya anzupassen.

5.1 Globales Verfahren auch bei einem Mesh

Die Besonderheit beim globalen gegenüber dem lokalen Verfahren ist, dass im Normalfall unterschieden werden muss, von welchem Mesh⁴ ausgegangen wird. Sprich, beim lokalen Verfahren wird jedes Kanten-Polygon paar unabhängig betrachtet und der resultierende Vektor G auf die Kante E und negiert auf das Polygon A angewendet. Beim globalen Verfahren ist es jetzt so, dass alle Vektoren G_i zusammen zu einem Vektor addiert werden. Dazu muss unterschieden werden, von welchem Mesh ausgegangen wurde und auch das Vorzeichen des Vektors G_i muss ggf. umgedreht werden.

Die Schwierigkeit ist bei der Selbstkollision, dass es keine zwei getrennten Meshes gibt. Aus diesem Grund werden anstatt zwei verschiedene Meshes, zwei getrennte Polygonzüge auf dem Mesh als Entscheidungsmittel genommen (siehe Abbildung 5-1 und Abbildung 5-2). Diese beiden getrennten Polygonzüge lassen sich dann ganz normal weiterverarbeiten, wie es auch der Fall bei zwei getrennten Meshes wäre. In Abschnitt 4.1.4 wird genauer beschrieben, wie die Unterscheidung der beiden Polygonzüge geschieht.

Zwar hat Volino in seinem Verfahren nie ausdrücklich gesagt, dass das globale Verfahren nicht auf ein Mesh anwendbar ist, es wurde aber auch nie beschrieben dass es funktioniert. In allen Testfällen waren mindestens zwei getrennte Meshes vorhanden. Außer in „The Ribbon Simulation“ Volino, et al. [2006, S. 5], denn dort wird ein Test mit einem einzigen langen Mesh beschrieben. Allerdings geschieht dort die Kollisionsbehandlung lediglich mit dem lokalen Verfahren.

⁴ **Mesh** (engl. für Polygongitter) bezeichnet ein Netz welches aus Polygonen (Vielecke) aufgebaut ist.

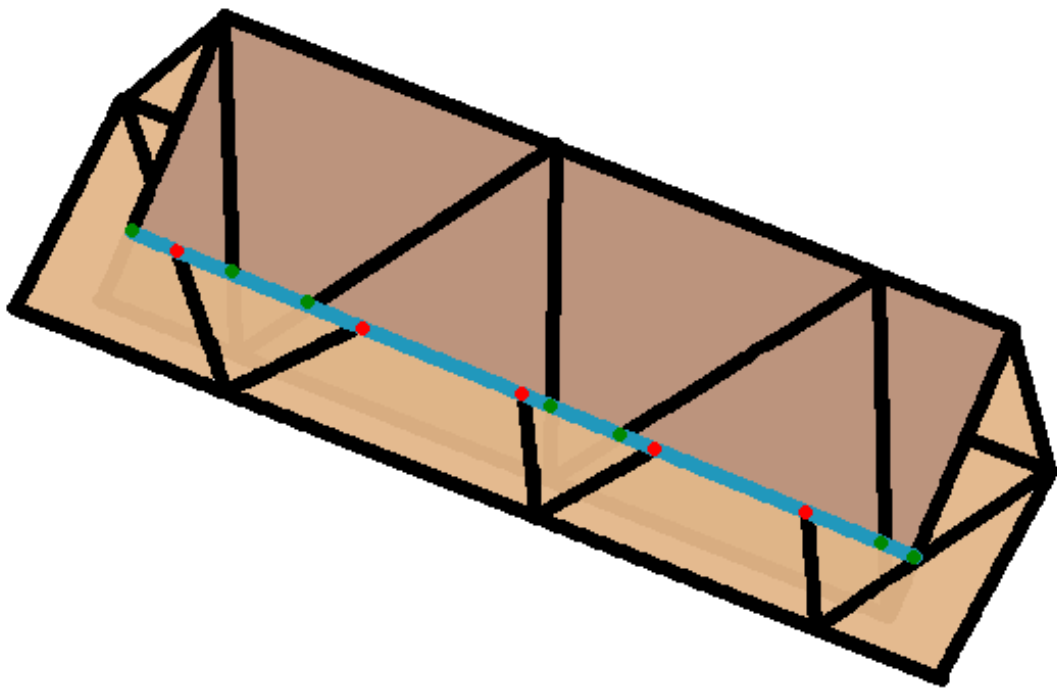


Abbildung 5-1 Selbstdurchdringung bei dem zwei Polygonzüge anhand zusammenhängender Dreiecke unterschieden werden. Ein Kanten-Polygonzug mit roten Kollisionspunkten und ein Kanten-Polygonzug mit grünen Kollisionspunkten.

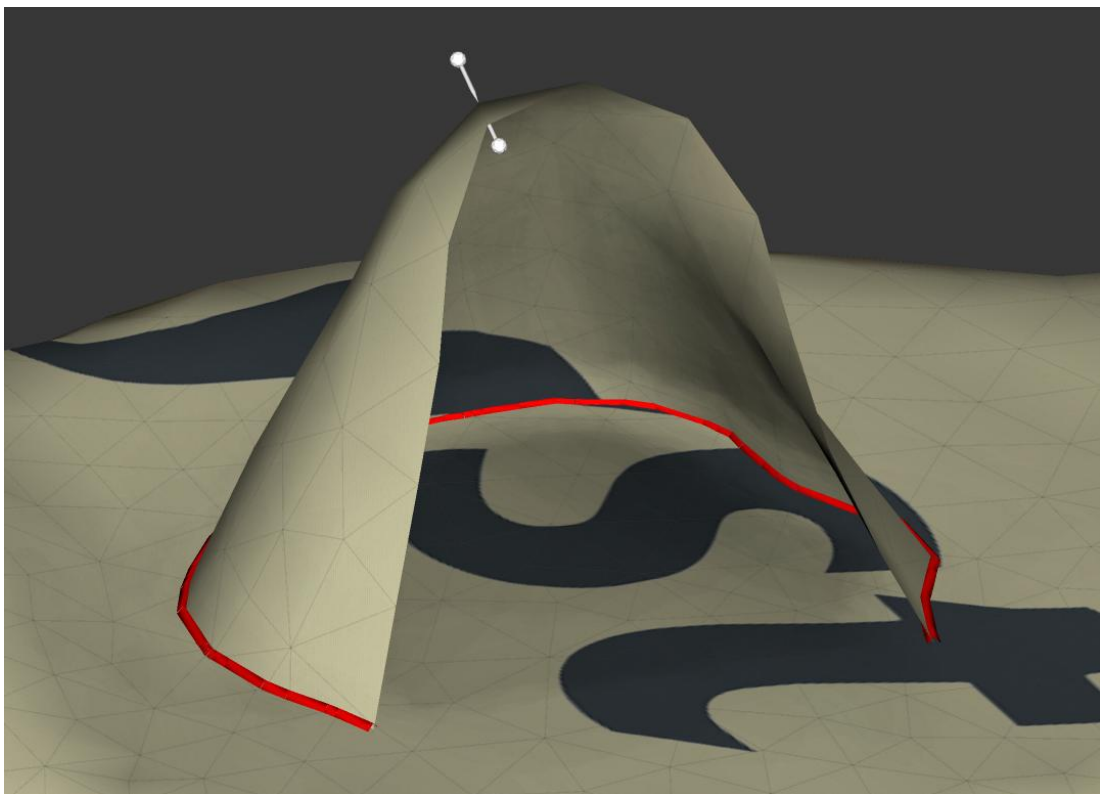


Abbildung 5-2 Darstellung eine Kollisionskontur auf nur einem Kleidungsstück, durch die Unterscheidung zweier getrennter Polygonzüge.

5.2 Physikalisch plausible Kollisionsantwort

Eine weitere Anpassung des Verfahrens von Volino, et al. [2006] ist das anschließende anpassen des Vektors G anhand von physikalischen Parametern.

Es ist essentiell, dass bei einer Bekleidungssimulation physikalische Gesichtspunkte mit eine Rolle spielen. Darunter fällt die Berücksichtigung der Massen der beteiligten Primitiven und der Abstand zum Kollisionspunkt. Diese Größen werden bei der Kollisionsantwort berücksichtigt, denn schließlich soll ein steifer und schwerer Stoff nicht sehr stark von einem leichten und weichen Stoff beeinflusst werden. Dazu wird der resultierende Verschiebevektor nochmals anhand der Kollisionsteilnehmer skaliert und auf die bereits vorhandene Geschwindigkeit und Position der einzelnen Partikel hinzuaddiert. Genauer ist dies in Abschnitt 4.1.5 beschrieben.

Die Anpassung des Vektors führt aber nicht zu einer geringeren Konvergenzgeschwindigkeit, sondern kann unter Umständen sogar eine beschleunigende Wirkung haben. So wird ein Stoff evtl. aus der Kollision in wenigen Iterationen hinaus gedreht, anstatt ihn komplett raus zu schieben.

Der Grund, weshalb die Konvergenzgeschwindigkeit nicht abnimmt liegt daran, dass der Vektor G den gesamten Kraftstoß darstellt. Anstatt diesen Kraftstoß gleichmäßig auf alle beteiligten Partikel zu verteilen, wird er einfach anhand der oben beschriebenen Parameter aufgeteilt. Wichtig dabei ist, dass der gesamte Kraftstoß weiterhin bestehen und so auch die maximale Verschiebung weiterhin erhalten bleibt.

5.3 Kombination der Verfahren zur Verbesserung Laufzeit und Ergebnisse

Bei der Entwicklung hat sich bestätigt, was seit Anfang an vermutet wurde. Und zwar, dass der `TriangleSelfCollisionDetector`, siehe Abschnitt 2.3.2.2 bei einer aufgetretenen Durchdringung gegen die Konturminimierung arbeitet. Ein weiteres Problem der beiden Kollisionsdetektoren war, dass sie auf exakt derselben BVH gearbeitet haben, sie diese aber nicht gemeinsam genutzt haben, sondern jeder seine eigene BVH erstellt und aktualisiert hat. Das hat den Rechenaufwand stark negativ beeinflusst, siehe Abschnitt 6.3. Allerdings ist es auch nötig, die BVH getrennt zu aktualisieren und zu traversieren, da am Anfang die Verfahren disjunkt waren und somit der `TriangleSelfCollisionDetector` die Position der Objekte verändert hat. Damit es nicht zu falschen Ergebnissen kommt, musste die Konturminimierung die BVH nochmals aktualisieren und danach neu traversieren.

Um die Problematik der Performance und der gegeneinander arbeitenden Verfahren zu lösen wurden beide Verfahren kombiniert (siehe Diagramm 5-1), so dass sie auf derselben BVH arbeiten. Dadurch wurde der Rechenaufwand zum durchlaufen der BVH und zum Aktualisieren um 50% verringert. Dies alleine hat schon zu einer deutlichen Verbesserung der Simulation geführt. Da die beiden Verfahren jetzt aber

schon miteinander kombiniert waren, konnten auch die Kollisionsantworten kombiniert werden, so dass der `TriangleSelfCollisionDetector` nur die Partikelpositionen ändert mit denen die Konturminimierung nichts zu tun hat.

Hierbei stellte sich aber ein großer Nachteil heraus. Wenn in einer Animation ein Stoffstück mit einer hohen Geschwindigkeit in ein anderes eindringt, wird die Durchdringung anfangs nicht merklich aufgehalten und der Stoff weht ohne Gegenkraft immer weiter in den anderen Stoff ein. Das lag daran, dass der Korrekturvektor von der Konturminimierung zu schwach war und die betroffenen Partikel nicht mehr vom `TriangleSelfCollisionDetector` beeinflusst wurden.

Die Lösung des Problems war, sich die Kollisionsantwort des `TriangleSelfCollisionDetector` anzuschauen und zu analysieren, ob die Antwort mit oder gegen die Konturminimierung arbeitet. Arbeit sie dagegen, wird die Antwort ignoriert. Arbeitet sie allerdings in dieselbe Richtung wie der Verschiebevektor der Konturminimierung, wird die Antwort vom `TriangleSelfCollisionDetector` zusätzlich angewandt. Diese Lösung stellte sich als in der Praxis als sehr Vorteilhaft heraus.

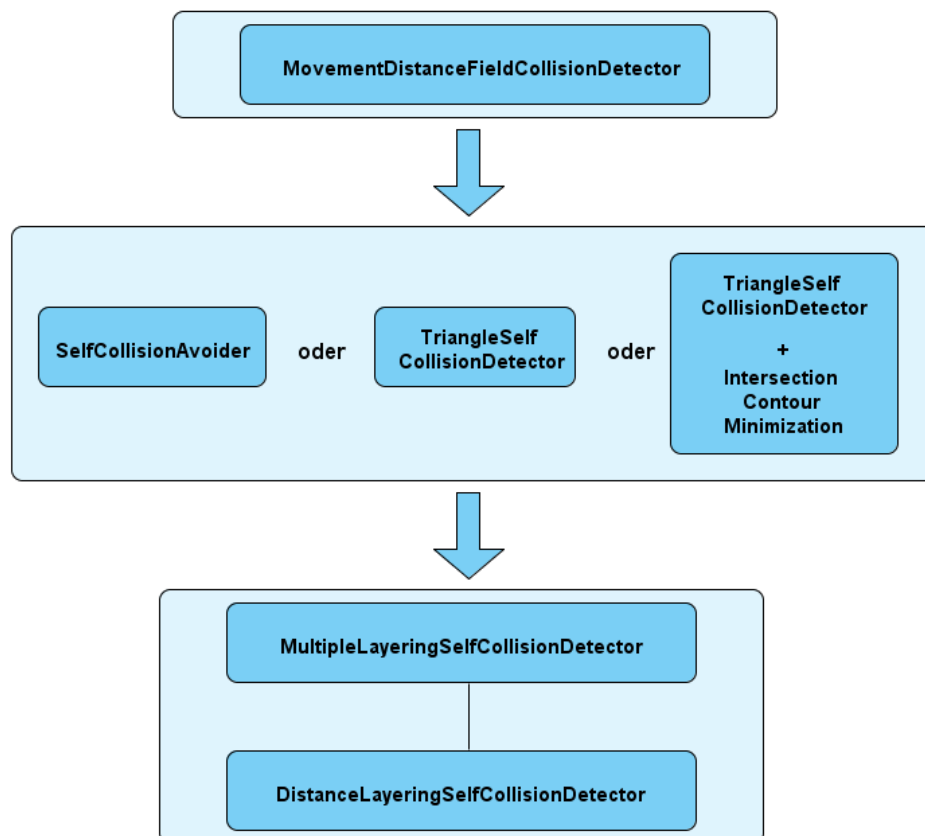


Diagramm 5-1 Ablauf der Kollisionsdetektoren nach der Kombination mit dem `TriangleSelfCollisionDetector`.

6 Ergebnisse

6.1 Übersicht

Die Konturminimierung hat sich während der gesamten Tests als sehr robust und rechenextensiv erwiesen.

Eine Übersicht über eine repräsentative Menge der Testszenarien ist in Abschnitt 6.2 beschrieben und ausführlich mit Screenshots dokumentiert.

In Abschnitt 6.3 ist dokumentiert, was die Laufzeitanalyse ergeben hat, in wie fern die Konturminimierung die Laufzeit der kompletten Simulation beeinflusst und wie sich die Kombination der Konturminimierung mit dem `TriangleSelfCollisionDetector` ausgewirkt hat.

Bei der Entwicklung sind einige Probleme aufgetreten, welche in Abschnitt 6.4 beschrieben sind. Die dort beschriebenen Probleme wurden erfolgreich gelöst, was auch zwingend nötig war, da sie ansonsten die Einsetzbarkeit von der Konturminimierung negativ beeinflusst hätten.

Als letztes beschreibt Abschnitt 6.5 die Limitationen der Konturminimierung. Diese Limitationen konnten oder mussten im Laufe der Diplomarbeit nicht behoben werden. Sie sind entweder so selten aufgetreten, dass es den Aufwand nicht gerechtfertigt hätte oder wären so komplex zu lösen gewesen, dass die Lösung zu aufwändig geworden wäre.

6.2 Testszenarien

In diesem Abschnitt sind einige Testszenarien dargestellt und beschrieben. Insbesondere sollen diese Szenarien zeigen, dass das Verfahren sehr robust und praxistauglich arbeitet und auch bei besonders kritischen Stellen ohne Anpassung einsetzbar ist.

6.2.1 Selbstdurchdringung Arminnenseite

Dieses Testszenario ist eines der Hauptprobleme, das immer wieder beim Simulieren aufgetreten ist. Es handelt sich um ein ganz normales T-Shirt, wie in Abbildung 6-1 zu sehen ist. Abbildung 6-2 zeigt das konkrete Problem. Es kann beim Vernähen passieren, dass sich der Stoff von der Körperaußenseite (blau) und der Stoff der Arminnenseite (rot) durchdringen. Dies passiert häufig aus dem Grund, dass beim Vernähen nur der `SelfCollisionAvoider` (siehe Abschnitt 2.3.2.1) und das Layering (siehe Abschnitt 2.4) aktiv sind. Bei diesen kleinen Abständen versagt der Self-

CollisionAvoider allerdings. Und da es sich nur um eine Stofflage handelt, kann das Layering das Problem auch nicht lösen.

Der Konturminimierung gelingt es, wenn die Geometrie des Avatars ausreichend Spielraum bietet, in wenigen Iterationen die Durchdringung korrekt, wie in Abbildung 6-3, aufzulösen.



Abbildung 6-1 Ausgangsposition eines T-Shirts mit Selbstdurchdringungen auf der Arminnenseite.

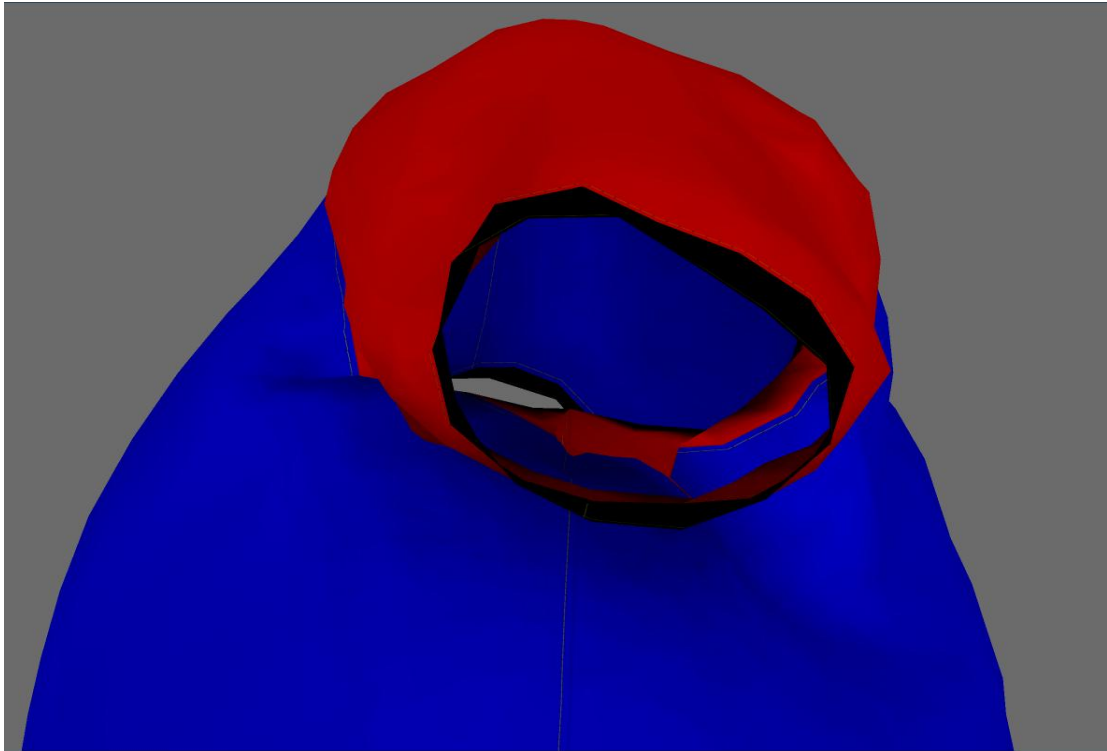


Abbildung 6-2 Der blaue Stoff durchdringt den roten Armstoff auf der Innenseite.

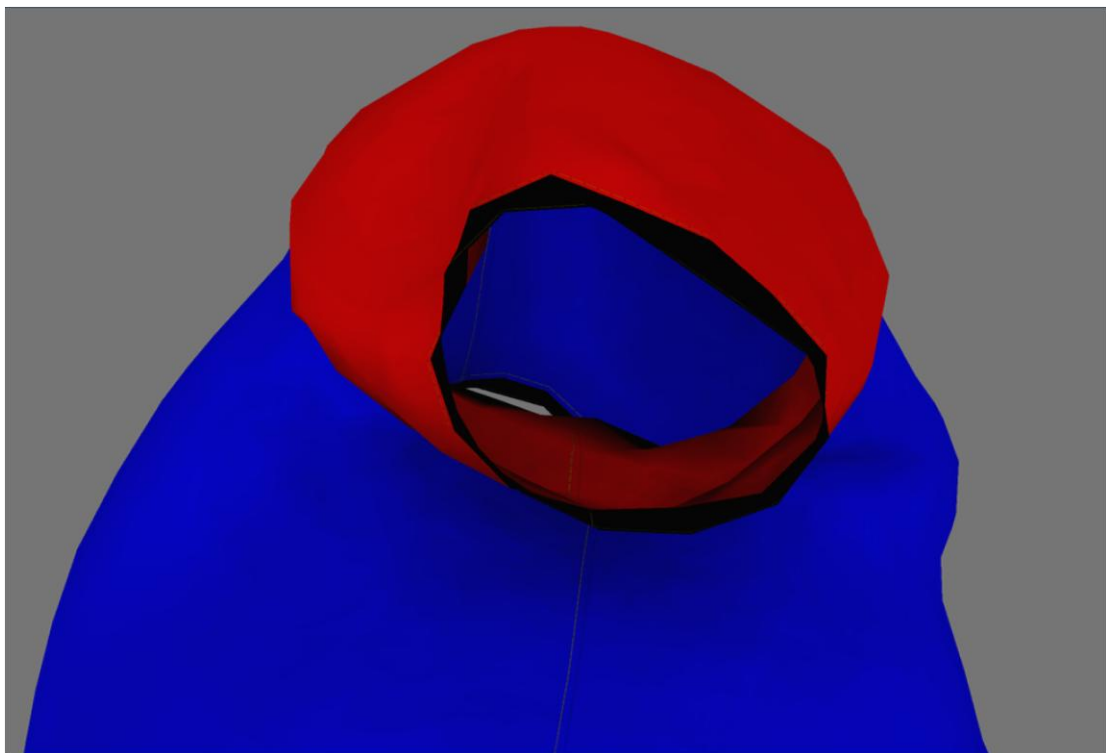


Abbildung 6-3 Nach Anwendung des globalen Verfahrens ist die Durchdringung korrekt aufgelöst worden.

6.2.2 Selbstdurchdringung Armbeuge

Eine weitere typische Situation sind Probleme in den Kniekehlen oder in der Armbeuge. Diese Probleme entstehen sehr oft beim animieren von Avataren. Auch hier bieten der SelfCollisionAvoider (siehe Abschnitt 2.3.2.1) und das Layering (siehe Abschnitt 2.4) keinen ausreichenden Schutz.

Auf Abbildung 6-4 ist das grundsätzliche Problem zu sehen und wie es sich in der Simulation zeigt. Dort ist schon zu erkennen, dass eine Selbstdurchdringung die Qualität stark mindert, da dieser Zustand einfach unnatürlich aussieht. Abbildung 6-5 zeigt dieselbe Situation, allerdings im Querschnitt. Dort wird das eigentliche Problem klar dargestellt und es zeigt sich, wie sich die einzelnen Dreiecke überschneiden. Wichtig in dem Zusammenhang ist die Spalte in der Geometrie der Armbeuge. Diese ist aus dem Grunde wichtig, damit ein gewisser Spielraum zum agieren des Verfahrens vorhanden ist.

Anschließend ist auf Abbildung 6-6 zu sehen, dass das globale Verfahren die Durchdringungen korrekt aufgelöst hat und es keine Durchdringungen an der Armbeuge mehr gibt. Abschließend zeigt Abbildung 6-7 die Szene wie sie komplett Durchdringungsfrei aussieht.

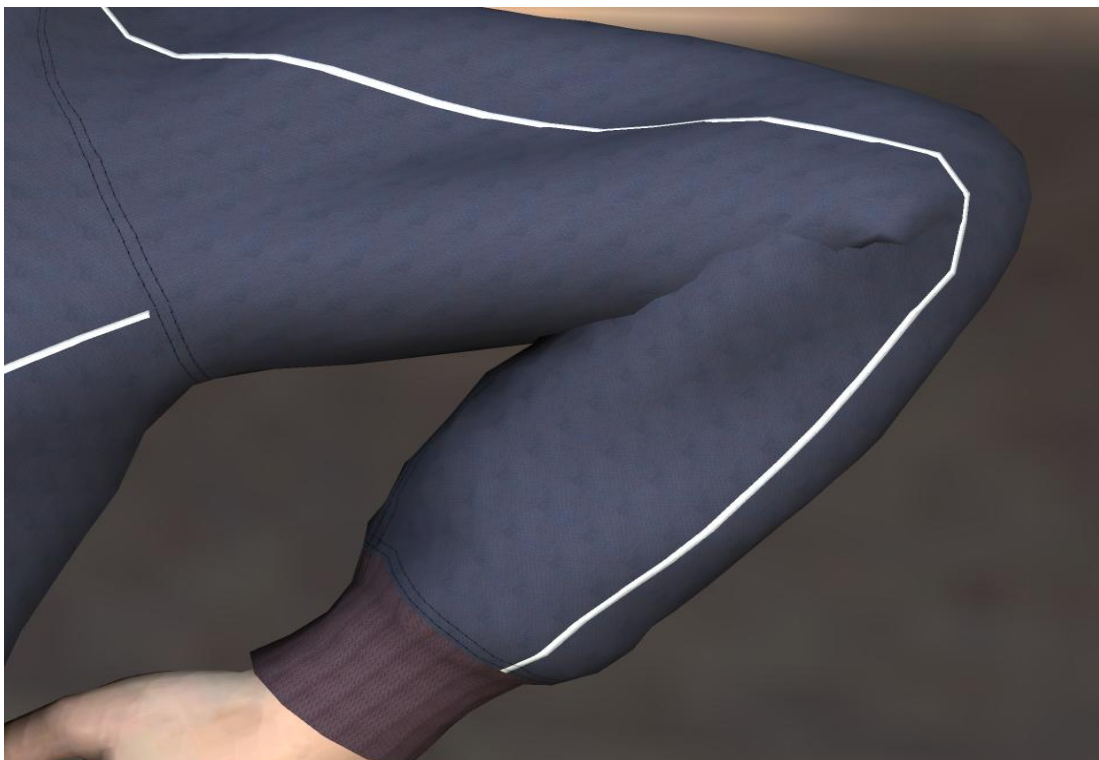


Abbildung 6-4 Problembereich Armbeuge, dabei durchdringt sich der Ärmel stark selber.

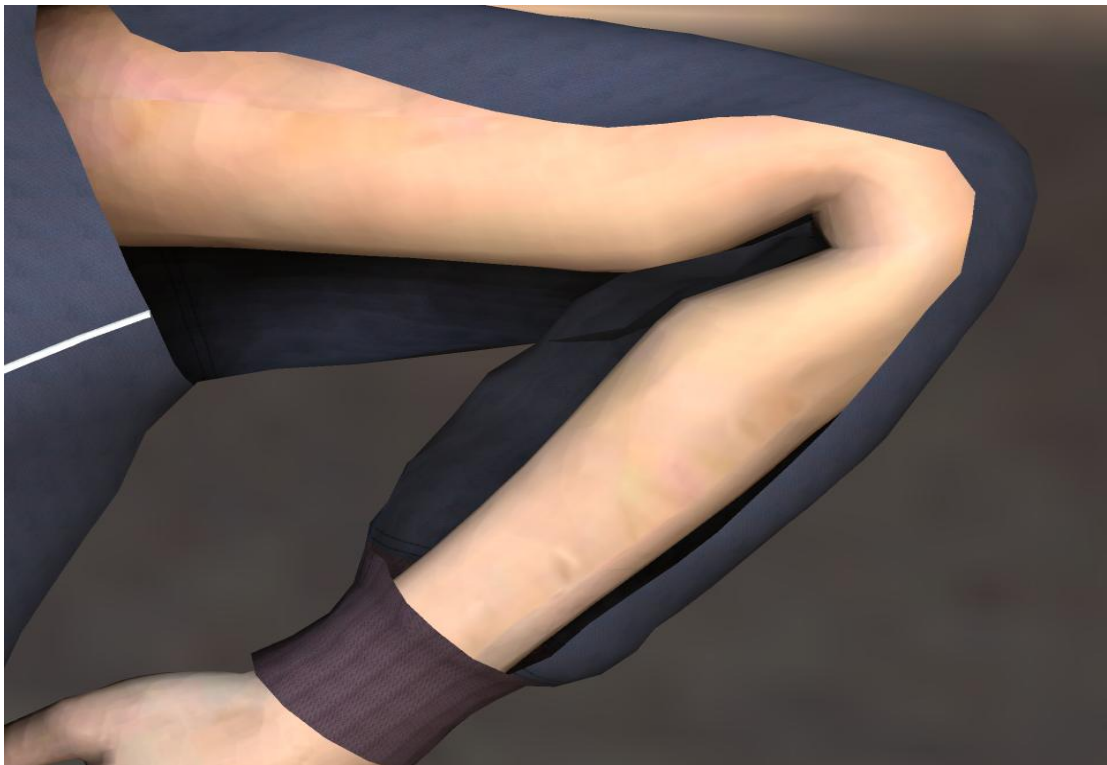


Abbildung 6-5 Der Querschnitt zeigt die einzelnen Dreiecke welche sich überschneiden.

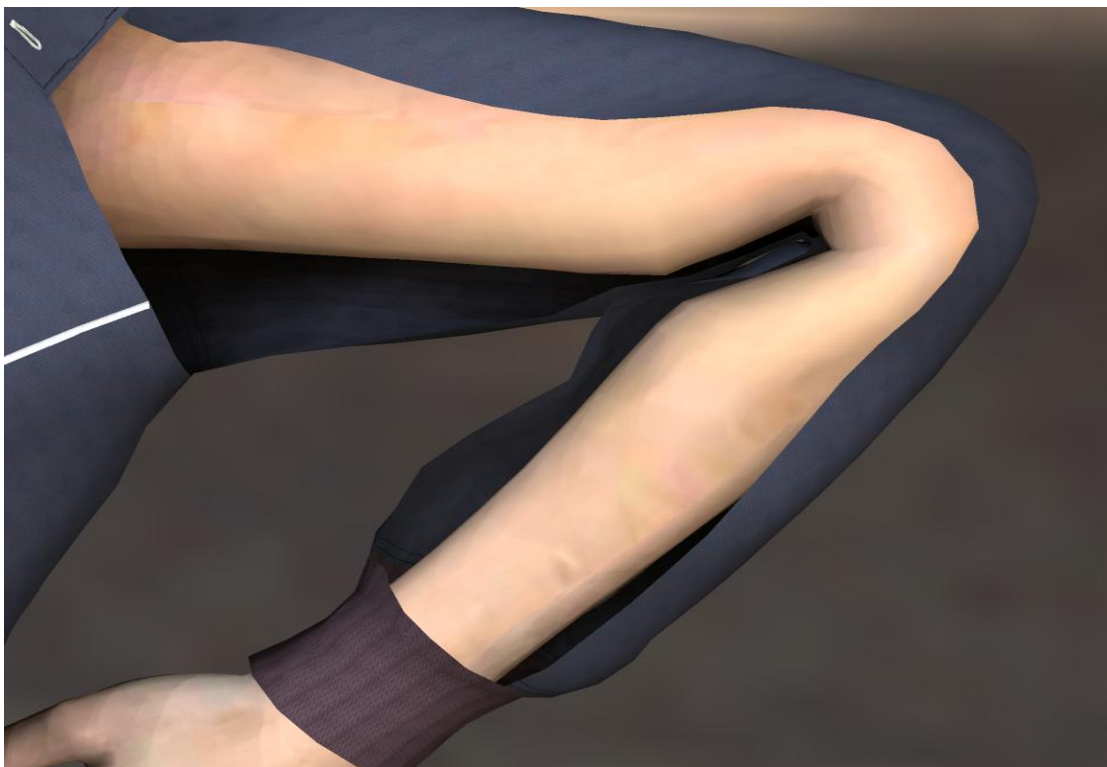


Abbildung 6-6 Auch hier führt das globale Verfahren zu einer deutlichen Verbesserung der Situation und zu einem überschneidungsfreien Zustand. Voraussetzung ist allerdings dass der Avatar genügend Spielraum für den Stoff zur Verfügung stellt.

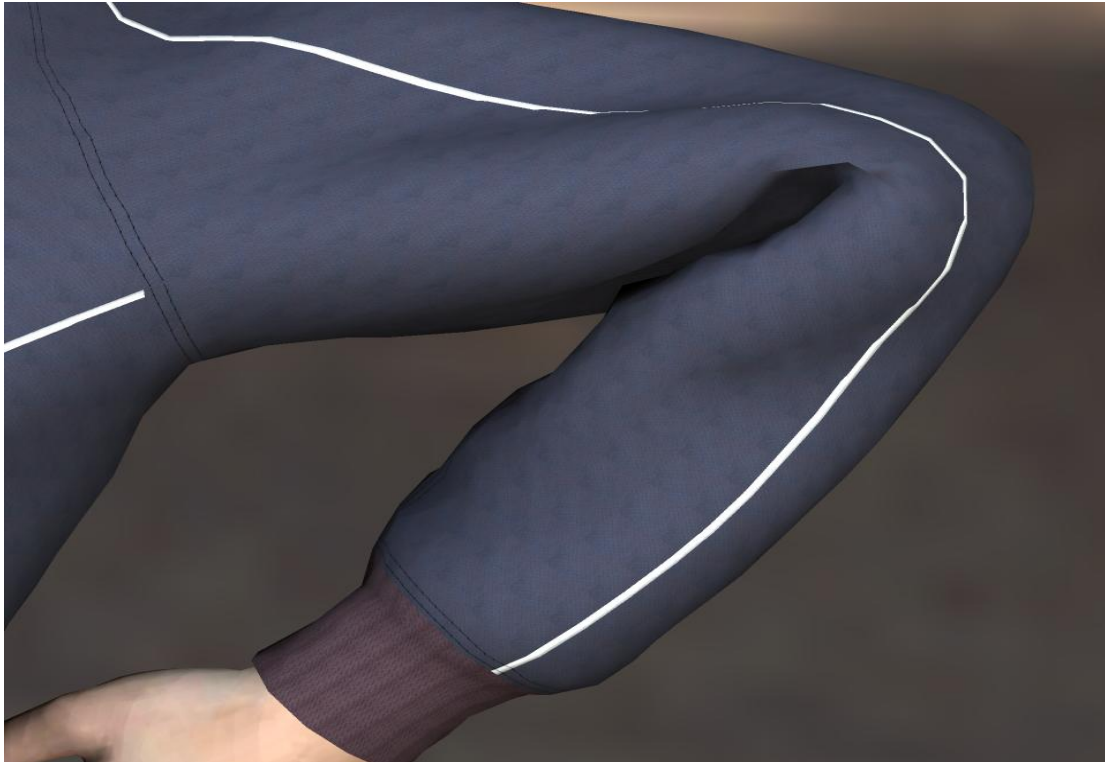


Abbildung 6-7 Totale der Szene ohne Selbstdurchdringungen in der Armbeuge.

6.2.3 Extremtest rotierende Kugel mit starken Durchdringungen

Bei diesem Testszenario wurde getestet, wie sich die Konturminimierung bei einer laufenden Simulation verhält. Dabei wurde ein Stoffstück, wie in Abbildung 6-8 zu sehen, über eine rotierende Kugel fallen gelassen. Ohne Konturminimierung entstehen dabei starke Selbstdurchdringungen, welche durch die farbigen Konturen markiert wurden. Bereits nach ~15 Zeitschritten ist auf Abbildung 6-9 eine deutliche Minimierung der Konturen zu sehen, obwohl sich die Kugel die ganze Zeit über weiter gedreht hat. Dabei konnte auch überprüft werden ob die Kollisionsantwort korrekt mit den bestehenden Geschwindigkeiten zusammenarbeitet. Sollten die Geschwindigkeiten falsch berücksichtigt worden sein, wäre das bei einer laufenden Simulation sofort stark aufgefallen. Auf Abbildung 6-10 ist die Szene nach ~50 Zeitschritten zu sehen. Es wurden alle Selbstdurchdringungen erfolgreich aufgelöst, denn ansonsten wären noch Konturen zu sehen. Anschließend wurde die Animation einige Zeitschritte laufen gelassen, um in Abbildung 6-11 zu zeigen, dass auch während der Animation die Szene frei von Durchdringungen bleibt. Wenn Durchdringungen auftreten, werden diese, da es in der Regel kleine Durchdringungen sind, in 1-2 Zeitschritten aufgelöst.

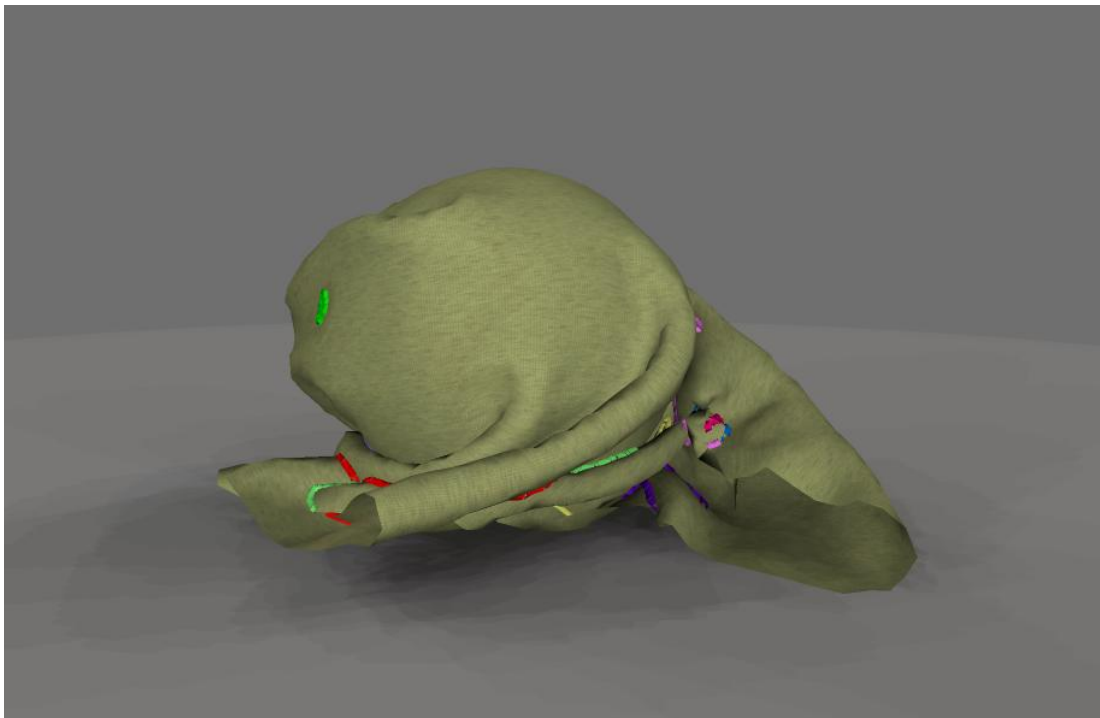


Abbildung 6-8 Starke Selbstdurchdringungen bei der Rotation einer Kugel unter dem Stoff.



Abbildung 6-9 Trotz laufender Animation sind nach ~15 Iterationen die Durchdringungen stark reduziert.

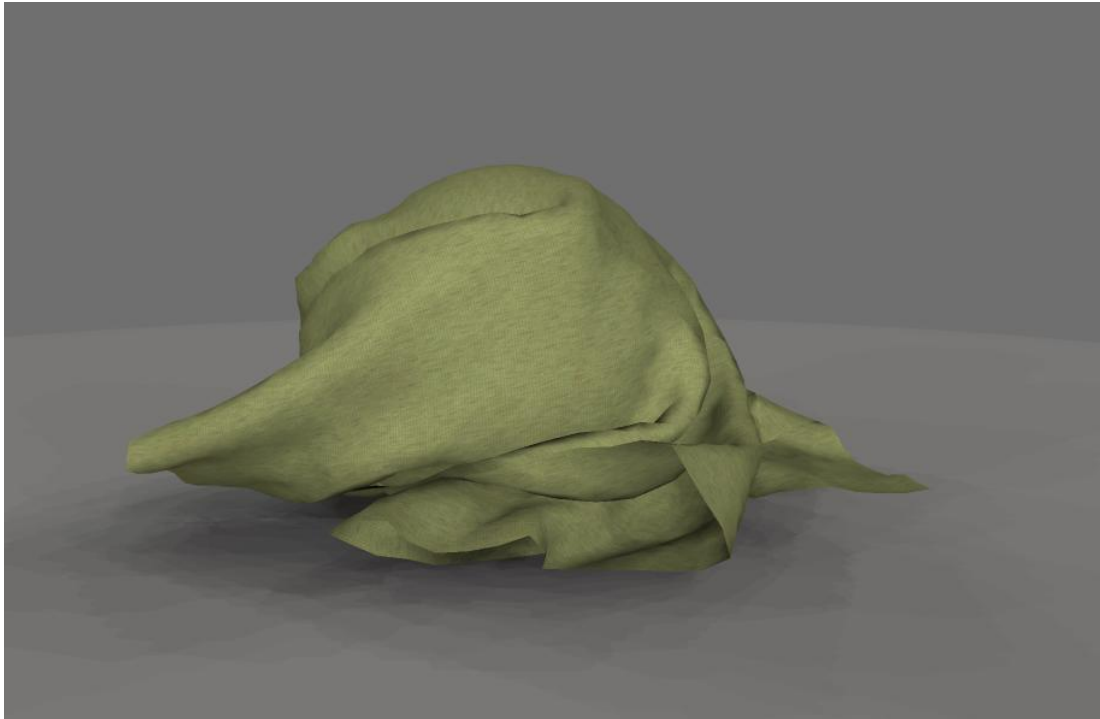


Abbildung 6-10 Nach insgesamt ~50 Iterationen ist die Szene frei von allen Durchdringungen.

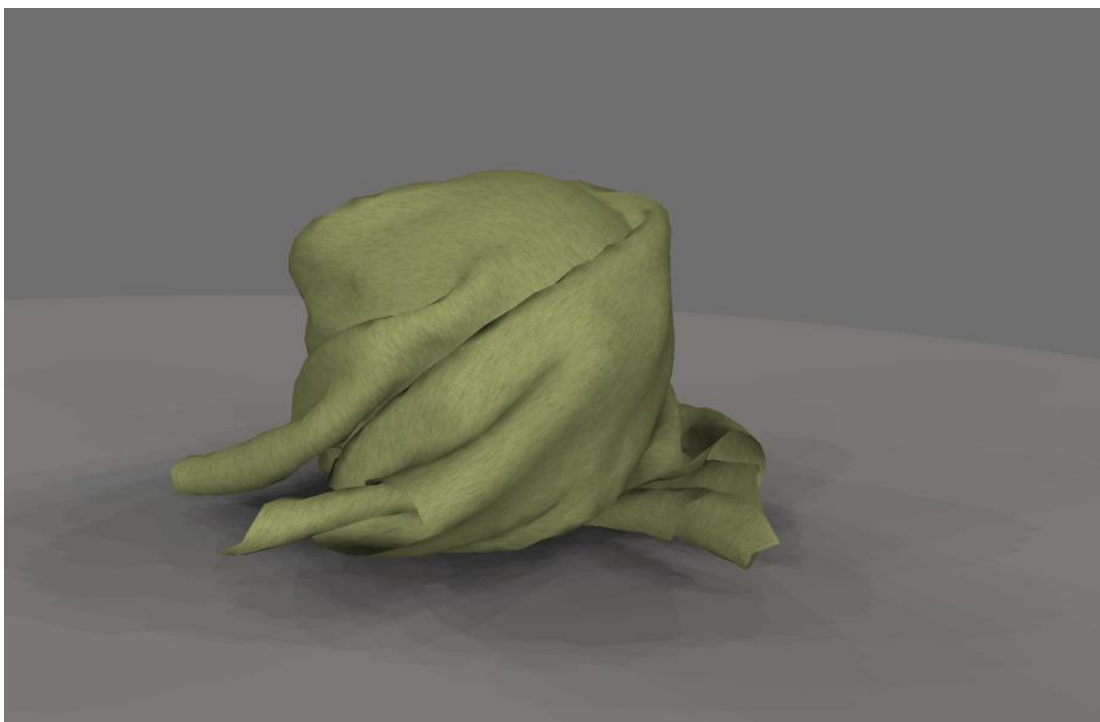


Abbildung 6-11 Auch weiterhin während der Animation frei von Durchdringungen.

6.2.4 Durchdringung verschiedene Stofflagen

Hier soll gezeigt werden, dass die Konturminimierung auch als Ersatz für das Layering eingesetzt werden kann. Das Layering, wie in Abschnitt 2.4 beschrieben ist dafür zuständig, verschiedene Stofflagen auseinander zu halten. Da das Layering aber auch einen signifikanten Rechenaufwand hat, siehe Abschnitt 6.3, ist es von Vorteil, wenn in manchen Szenen, bei denen einfache Durchdringungen zweier oder mehrerer Kleidungsstücke auftreten, auf das Layering verzichtet werden kann. In Abbildung 6-12 ist eine solche Szene zu sehen, wo sich die beiden übereinanderliegenden Kleidungsstücke durchdringen. In Abbildung 6-13 ist der Problembereich genauer zu sehen und es sind auch die einzelnen identifizierten Konturen deutlich zu erkennen. Für diese kleinen Durchdringungen benötigt das globale Verfahren sehr wenige Iterationen (<5) um in den Zustand auf Abbildung 6-14 zu kommen. Somit lässt sich diese Szene auch ohne eingeschaltetes Layering sehr gut simulieren.



Abbildung 6-12 Durchdringung zweier verschiedener Kleidungsstücke in der Totalen.

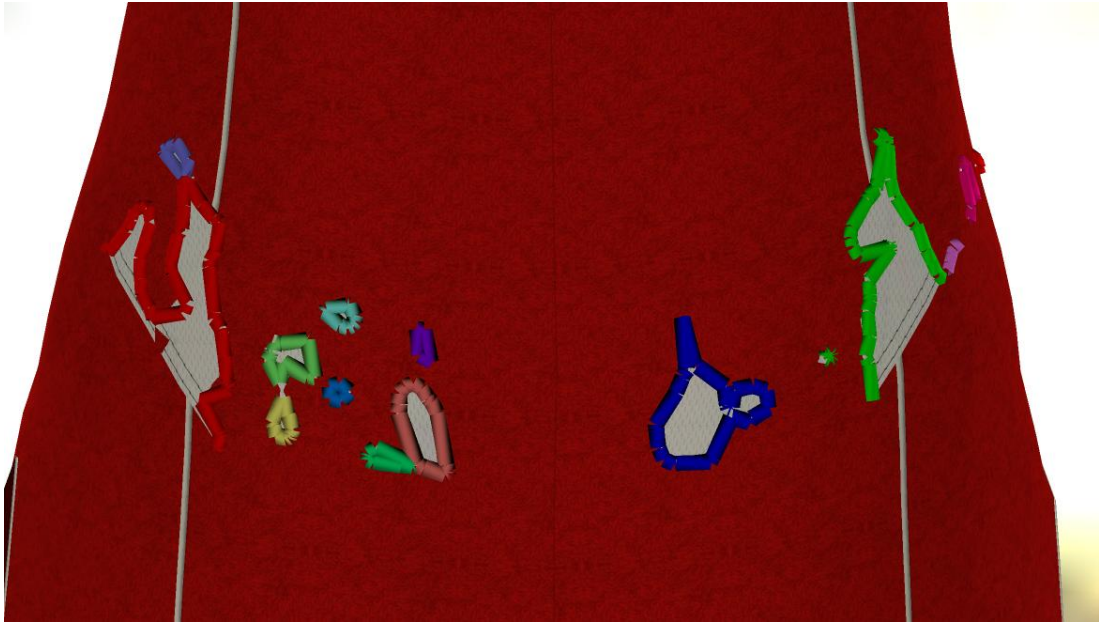


Abbildung 6-13 Durchdringung zweier verschiedener Kleidungsstücke mit eingezeichneten Kollisionskonturen.

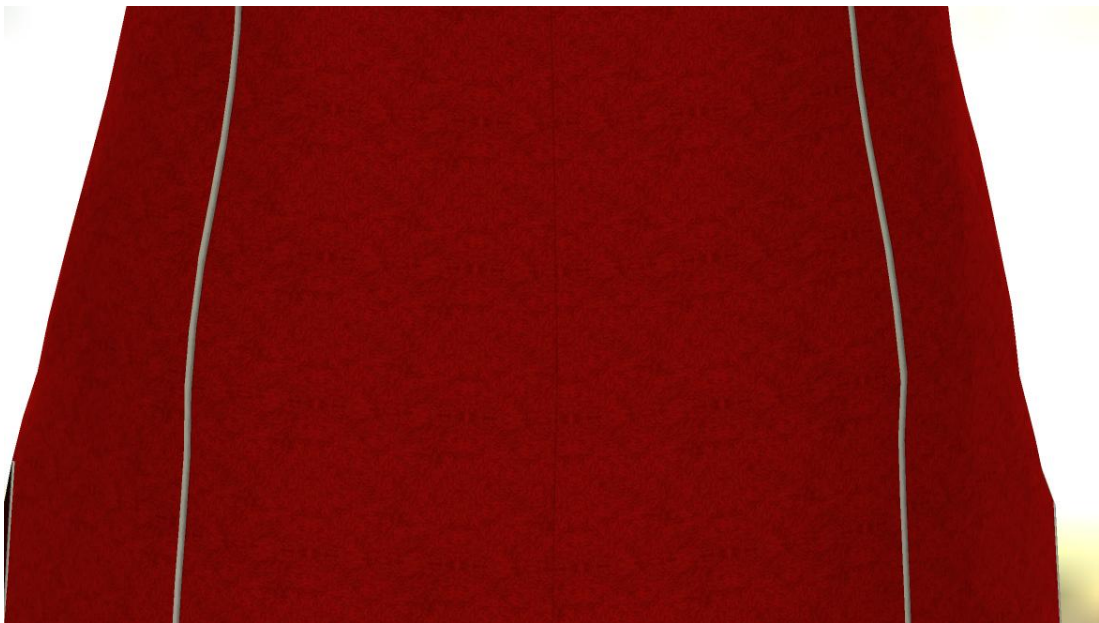


Abbildung 6-14 Durchdringener Bereich nach erfolgreicher Auflösung der Durchdringungen.

6.2.5 Auflösung von Durchdringungen in mehreren Schritten

Dieses Szenario soll verdeutlichen, wie die globale Konturminimierung genau arbeitet. Dabei entscheidet der Parameter h_0 wie schnell die Szene konvergiert. Es muss dabei gegeneinander abgewogen werden, ob eine schnelle Konvergenzgeschwindigkeit gewünscht ist, dafür aber starke Kraftstöße in der Szene, welche zu einem Nachschwingen führen können. Oder ob es besser ist langsam zu konvergieren und dadurch schwächere Kraftstöße bekommt. Abbildung 6-15 zeigt die Ausgangssituation, bei dem das karierte Kleidungsstück das gestreifte durchdringt. Die Kollisionskontur ist mit rot deutlich markiert. Bereits nach ~15 Zeitschritten ist auf Abbildung 6-16 eine deutliche Minimierung der Kollisionskontur zu sehen. Nach weiteren ~30 Zeitschritten, siehe Abbildung 6-17, ist von der ursprünglichen Durchdringung nur noch ein kleiner Rest übrig. Auf Abbildung 6-18 ist die Szene nach nur ~45 Iterationen zu sehen, dort ist die Durchdringung schon beinahe komplett aufgelöst. Nach insgesamt ~55 Iterationen ist, auf Abbildung 6-19, die Szene frei von Durchdringungen zu sehen. Ein Zeitschritt hat in der dargestellten Szene ca.1-2ms benötigt. Damit ist die Szene nach nur ~55-110ms. durchdringungsfrei. Es wurde dabei ein h_0 fest gewählt, mit dem sowohl eine annehmbare Konvergenzgeschwindigkeit erreicht wird, aber auch die Kraftstöße nicht zu hoch sind.

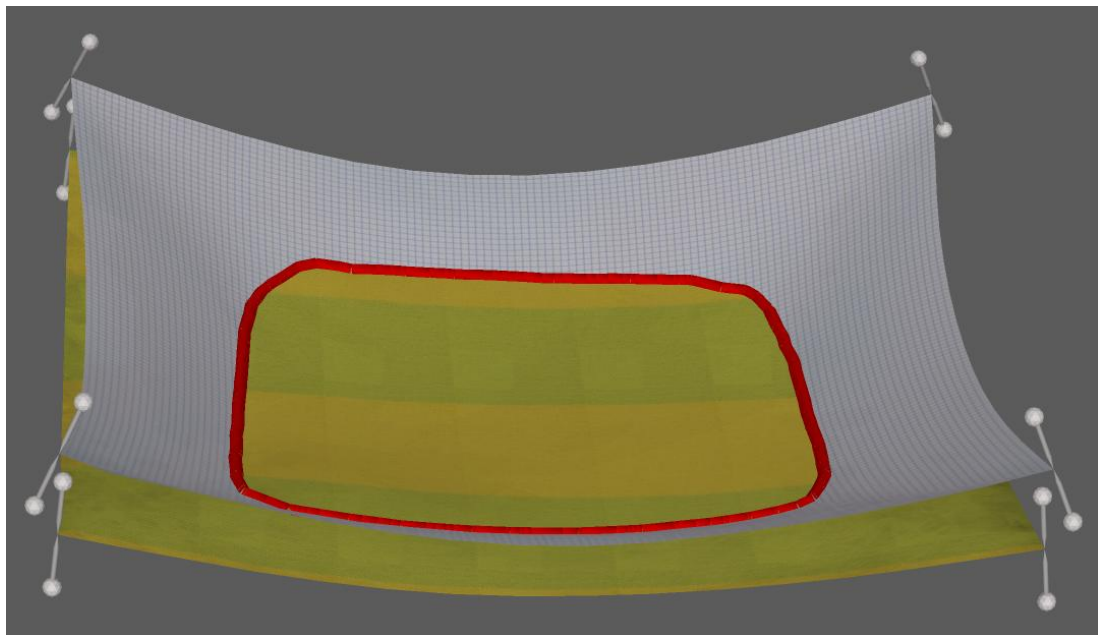


Abbildung 6-15 Testszene mit zwei durchdrungenen Stoffteilen und eingezeichnete Kollisionskontur.

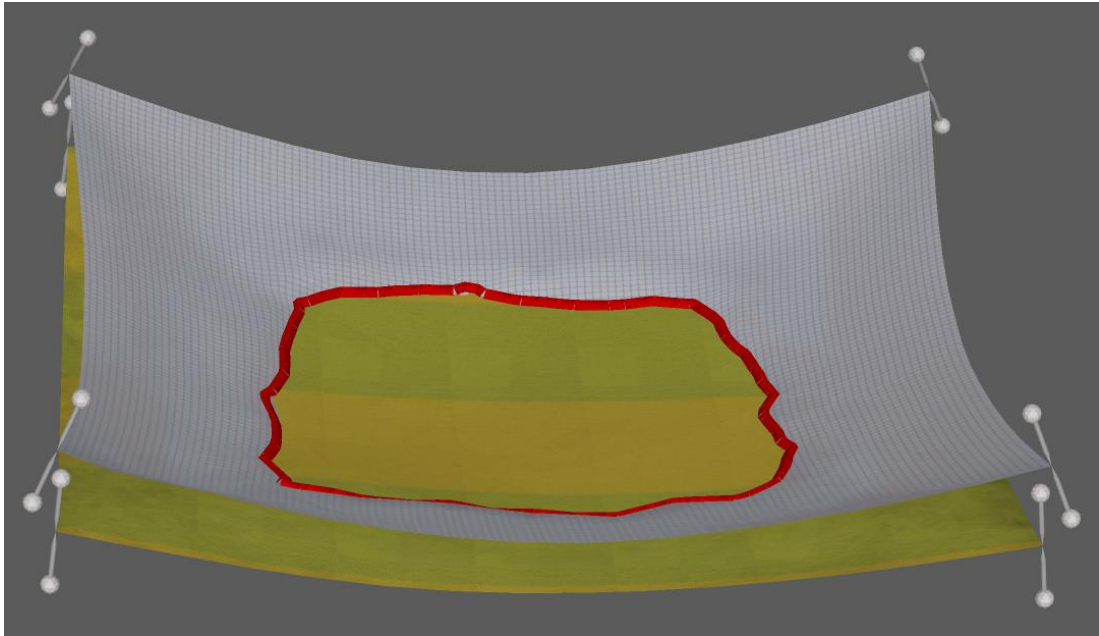


Abbildung 6-16 Reduktion der Kollisionskontur nach ~15 Iterationen.

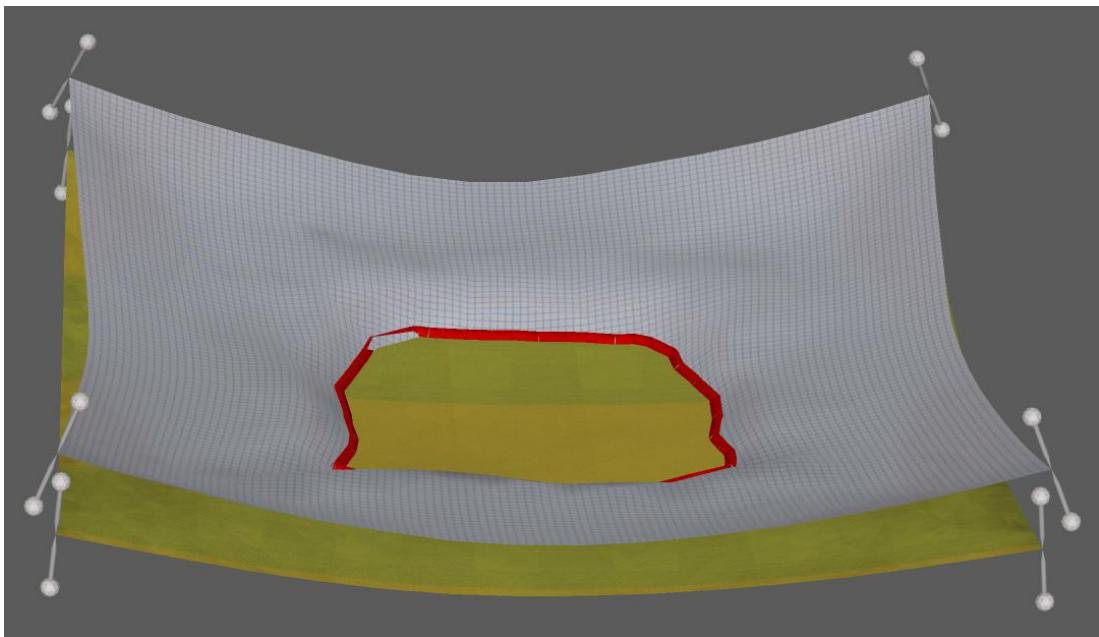


Abbildung 6-17 Reduktion der Kollisionskontur nach ~30 Iterationen.

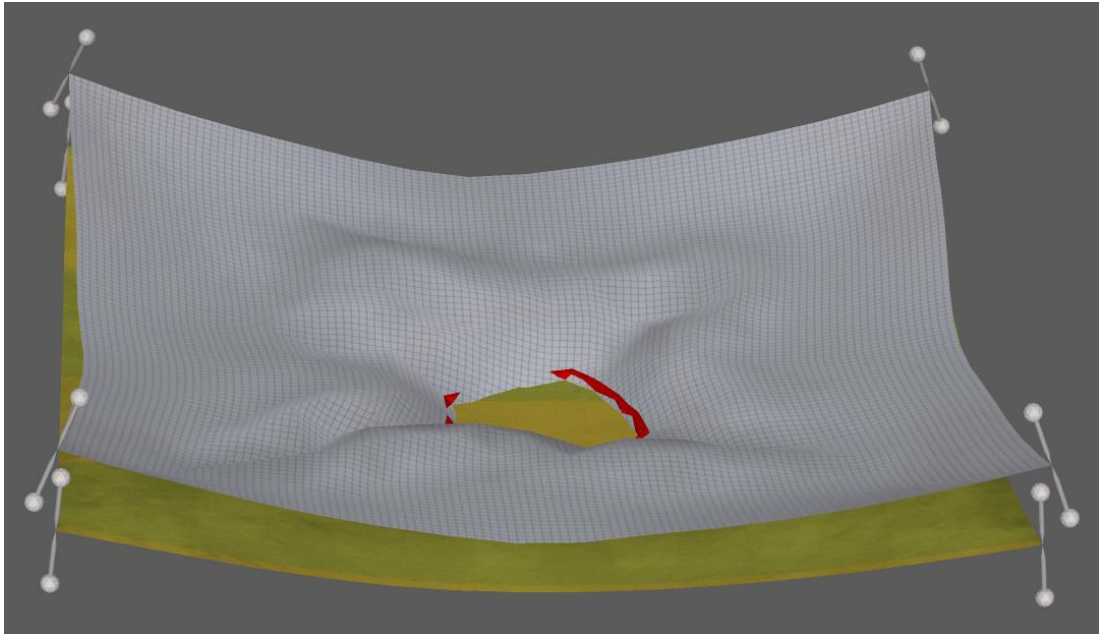


Abbildung 6-18 Reduktion der Kollisionskontur nach ~45 Iterationen.

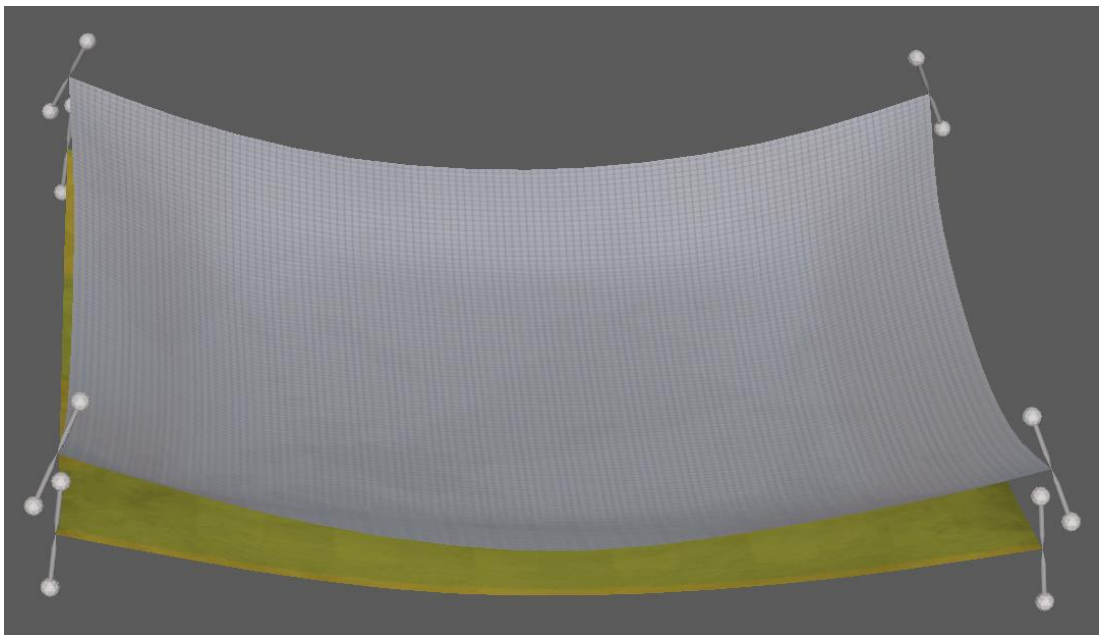


Abbildung 6-19 Komplette Auflösung der Durchdringung nach ca. ~55 Iterationen.

6.3 Zeitliche Kohärenz

6.3.1 Vergleich zu den anderen Kollisionsdetektoren

Im bisherigen Simulationsablauf wurde meistens mit aktiviertem Avider und, je nach Szene, mit aktiviertem Layering gearbeitet. Von daher bezieht sich der prozentuale Zuwachs im Diagramm 6-1, relativ zu den Zeiten der kompletten Simulation mit aktiviertem Avider und Layering. In der aktuellen Version benötigt die Berechnung eines Zeitschrittes bei der Testszene, mit ~16000 Partikeln und zwei Kleidungsstücken, ~430ms auf einem Intel C2D mit 2 GHz Taktrate. Dies ist die Zeit auf die sich die 100% des Aviders + Layering beziehen.

In den Szenen, welche auf das Layering verzichten können, ist die Konturminimierung (ICM) ~20% schneller als der TriangleSelfCollisionDetector (TSD) mit aktiviertem Layering und dabei nur ~10% langsamer als der Avider mit aktiviertem Layering. Der Unterschied zwischen TSD und Layering, sowie TSD kombiniert mit ICM und Layering, beträgt ebenfalls lediglich nur 5%. Daran sieht man, dass der zusätzliche Overhead durch die Konturminimierung sehr minimal ist.

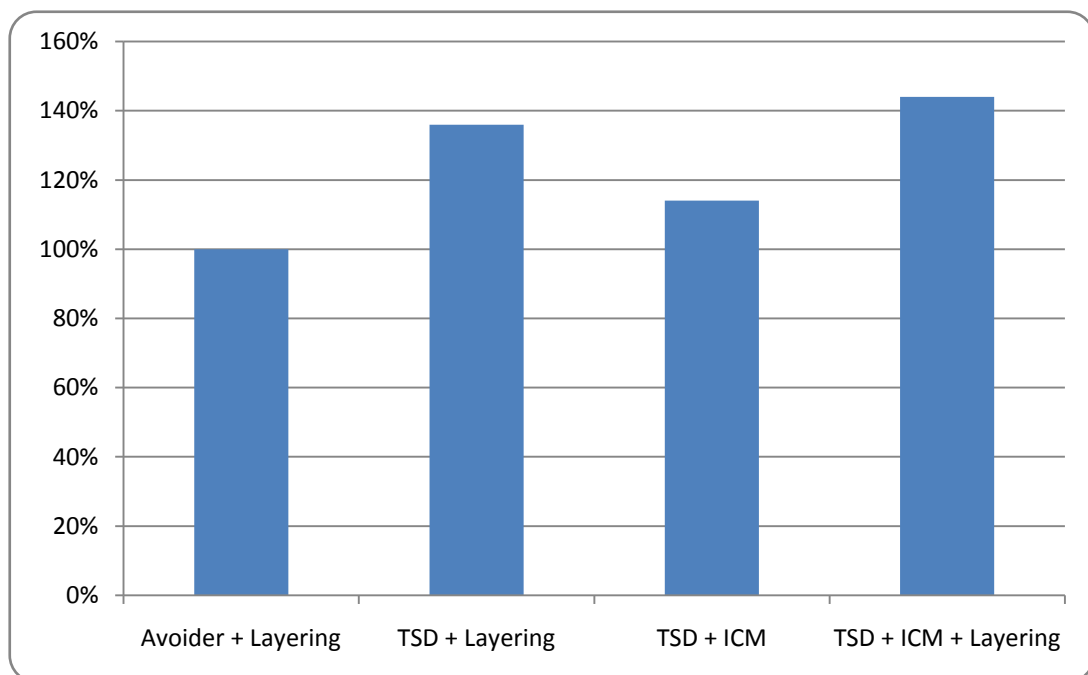


Diagramm 6-1 Prozentuale Darstellung des Overheads durch die Konturminimierung.

6.3.2 Optimierung der Laufzeit durch Kombination der Verfahren

Durch die Kombination vom TriangleSelfCollisionDetector (TSD) und der Konturminimierung (ICM), wurde ein beträchtlicher Performancezuwachs festgestellt. Diagramm 6-2 zeigt den relativen Zuwachs bezogen auf den TriangleSelfCollisionDetector. Hier wurde nur die Laufzeit der Kollisionsbehandlung und nicht der gesamten Simulation gegenübergestellt. Aus diesem Grund sind die Werte von den Daten in Abschnitt 6.3.1 abweichend. Es handelt sich um dieselbe Testszene und dieselbe Hardware wie in Abschnitt 6.3.1.

Durch die Kombination der beiden Verfahren wurde der Overhead in der Laufzeit um ~80% reduziert.

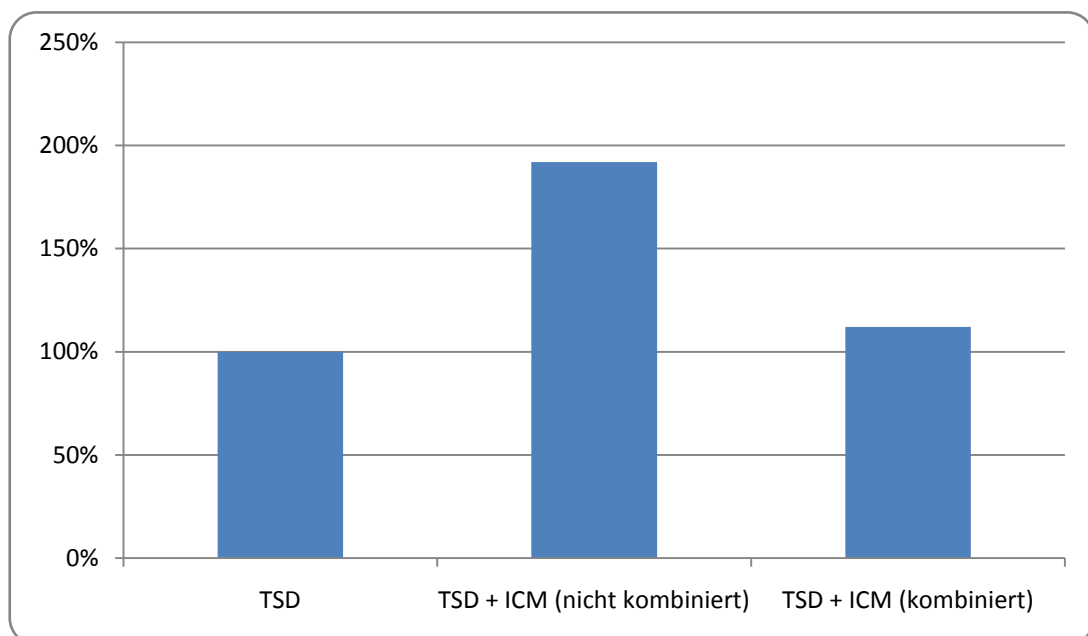


Diagramm 6-2 Prozentuale Darstellung des Geschwindigkeitszuwachs durch Kombination des TriangleSelfCollisionDetectors mit der Konturminimierung.

6.4 Aufgetretene Probleme

6.4.1 Performance

Gegen Ende der Entwicklung sind Probleme mit der Performance aufgetreten, die aber in erster Linie darin begründet waren, dass der TriangleSelfCollisionDetector und die Konturminimierung jeweils eine eigene BVH hatten und diese updaten und traversieren mussten. Dies war anfangs auch zwingend notwendig, da die Verfahren nacheinander abliefen und sich somit die Position der Primitiven nach dem TSD veränderten. Wenn die Konturminimierung keine eigene BVH gehabt hätte, wäre auf

veralteten Daten gearbeitet worden, wodurch eine falsche Kollisionsantwort entstanden wäre.

Der negative Einfluss der Rechenzeit, siehe Abschnitt 6.3.2, war aber so dramatisch dass dringend an der Performance gearbeitet werden musste. Aus diesem Grund entstand die Idee, den TSD mit der Konturminimierung zu kombinieren. Dies führte zu einer drastischen Verbesserung der Performance.

Neben der Kombination mit dem TSD wurden auch noch andere Möglichkeiten zur Steigerung der Performance der gesamten Kollisionserkennung diskutiert. Eine Idee wäre das Einführen von sogenannten Normalenkegeln, wie in Metzger [2001] beschrieben. Dadurch könnte nochmals drastisch der Aufwand von unnötigen Primitivtests reduziert werden. Eine weitere Idee wäre es, zusätzlich auch noch das Layering zu kombinieren, so dass alle drei Verfahren auf denselben Datenstrukturen arbeiten. Diese Kombination würde wahrscheinlich noch mal zu einer ähnlichen Performancesteigerung führen, wie die Kombination mit dem TSD.

Es war auch eine Überlegung, die komplette Kollisionserkennung zu überarbeiten. In Curtis, et al. [2008], gibt es einige interessante Ansätze zur Verwendung von sogenannten Representative-Triangles. Hierbei muss aber im ersten Schritt das dort beschriebene Verfahren nochmals genau geprüft werden, ob es praktisch anwendbar ist und wo die Limitationen liegen.

6.4.2 Bestimmen der Parameter

Frei definierbare Parameter in Kollisionserkennungssystemen sind immer sehr problematisch. Eigentlich möchte der Benutzer auf einen Knopf drücken und dann passiert alles automatisch, ohne dass der Benutzer eingreifen muss. Unglücklicherweise ist es oft nötig, dass manche Parameter, je nach Situation, angepasst werden müssen, da z.B. die Stärke der Kollisionsantwort an die jeweilige Situation angepasst werden muss.

So hat auch die Konturminimierung zwei Parameter, die zum Beeinflussen der Kollisionsantwort gedacht sind. Die Parameter h_0 und g_0 sind in Abschnitt 3.5.4 näher beschrieben. Das Problem hierbei ist, dass g_0 zum Stabilisieren der Antwort gedacht ist. Und solange die Länge der Kontur auch deutlich größer ist als g_0 , hat g_0 keinen nennenswerten Effekt. Verringert sich aber die Länge der Kontur, weil sie fast komplett aufgelöst wurde, es keine großen zusammenhängenden Konturen gibt oder das lokale Verfahren verwendet wird, spielt dieser Faktor eine immer größere Rolle und verringert den Einfluss von h_0 .

Die Interaktion soll für den Benutzer so gering wie möglich gehalten werden. Es ist zwar möglich, die Parameter h_0 und g_0 vom Benutzer anpassen zu lassen, jedoch müssen vernünftige initiale Werte für beide Parameter bestimmt werden. Denn wenn die Werte unpassend gewählt sind, kann es sein dass z.B. h_0 zu schwach ist

und sich so gut wie nichts bewegt. Es kann aber auch sein, dass der Wert zu stark ist und es den Stoff somit zerreit.

Eine komplett automatische Anpassung von den Parametern anhand der Konturlnge oder der Anzahl der Segmente innerhalb einer Kontur wurde kurz ausprobiert, hat aber nicht zu dem gewnschten Ergebnis gefhrt. Aus diesem Grund ist fr g_0 ein, durch Tests ermittelter, adquater Wert bestimmt worden. Fr h_0 wird die durchschnittliche Dreieckflche des aktuellen Meshes verwendet. Beide Werte haben sich in mehreren Tests als sehr robust erwiesen und funktionieren in den meisten Fllen sehr gut. In den wenigen Extremfllen, in denen die Pfade so klein werden dass h_0 durch g_0 zu schwach wird, hat der Benutzer die Mglichkeiten, h_0 manuell zu erhhen oder g_0 zu verringern.

Der Grund, weshalb h_0 auf die durchschnittliche Dreieckflche gesetzt wird, liegt daran, dass h_0 die maximale Lnge der Kollisionsantwort angibt. Da ein Springen der Dreiecke durch eine zu starke Kollisionsantwort versucht wird zu vermeiden, ist es ein guter Richtwert, die Kollisionsantwort mit der Dreieckflche zu initialisieren. Dabei ist sichergestellt, dass die Kollisionsantwort, sofern das Mesh nicht komplett entartet ist, nicht zu stark ist.

Sollte sich herausstellen, dass der Benutzer doch zu viel an den Parametern anpassen muss, so kann noch mal die Idee mit dem dynamischen Anpassen der Werte aufgegriffen werden. Es kann auch sein, dass eine andere Funktion als die die unter Abschnitt 3.5.4 beschrieben ist, ein besseres Resultat liefert.

6.5 Limitationen

6.5.1 Fast planare Ebenen

Ein Problem, welches auch beim Testen ab und zu aufgetreten ist, ist dass das Verfahren bei zwei fast planaren Flchen leicht instabil wird. Der Effekt, der dabei beobachtet wurde ist, dass sich die Richtung des Interaktionsvektors, welcher angibt welche Partikel in welche Richtung geschoben werden sollen, sich fr ein bis zwei Zeitschritte umdreht. Anschließend ist die Richtung wieder korrekt und die Durchdringung wird korrekt aufgelst.

Dieser Effekt ist nur in bestimmten synthetischen Testszenarien aufgetreten. Besonders hufig ist das Problem erschienen, als die Konturminimierung und der TriangleSelfCollisionDetector noch getrennt arbeiteten. Dabei hielt der TriangleSelfCollisionDetector die Stoffstcke immer auf Spannung zueinander, wodurch diese beinahe planar zueinander lagen. Das Problem trat auch nur kurz vor der endgltigen Auflsung von geschlossenen Konturen auf, da auch in dieser Situation die Stoffstcke besonders nah beieinander liegen.

Durch die Kombination beider Verfahren, konnte der Effekt nicht mehr nachgestellt oder beobachtet werden. Für den Fall, dass er doch noch mal auftreten sollte und dann so stark stört, dass er behoben werden muss, wäre es eine Möglichkeit, die Pfade von einem zum nächsten Zeitschritt irgendwie zu identifizieren. Somit wäre es möglich zu überprüfen, falls die Interaktionsrichtung sich auf einmal stark zum vorherigen Zeitschritt unterscheidet. Für den Fall könnte die Interaktionsrichtung des letzten Zeitschritts genommen werden oder keine Verschiebung vorgenommen werden, bis wieder ein plausibles Ergebnis geliefert wird.

6.5.2 Planare Ebenen

Wie auch schon in Volino, et al. [2006] beschrieben, hat die Konturminimierung neben den in Abschnitt 3.5.7 beschriebenen Limitationen auch Probleme mit planaren Ebenen.

In dem Fall, dass z.B. zwei Stoffstücke absolut identisch aufeinander liegen, können keine Konturen erstellt werden und es kann auch nichts korrigiert werden. Dieser Fall tritt allerdings in der normalen Benutzung von Vidya nahezu gar nicht auf. In einigen extra erstellten Testszenarien hat sich gezeigt, dass, selbst wenn Kleidungsstücke genau aufeinander liegen, entweder die Floatungenauigkeit oder der TriangleSelfCollisionDetector dafür sorgen, dass manche Dreiecke über und manche unter den anderen liegen. Dadurch können dann wieder Pfade erstellt werden und die Konturminimierung kann wie vorgesehen arbeiten.

6.5.3 Zuständigkeitsfreier Bereich

Ein Problemfall, welcher allerdings ebenfalls äußerst selten auftritt ist, wenn ein Eckpunkt eines Dreiecks genau auf der Ebene eines Dreiecks liegt. Hier kann kein Kollisionspfad berechnet werden. Das liegt daran, dass Start- und Endpunkt des Segments auf genau derselben Stelle liegen. Daher gibt es keine Schnittkante sondern höchstens einen Schnittpunkt mit der Länge 0. In diesem Fall wird die Konturminimierung keine Antwort generieren und es wird sich nichts an dem aktuellen Zustand ändern.

Das Problematische dabei ist, dass in diesem Zustand auch der TriangleSelfCollisionDetector keine Antwort generiert, da es für ihn keinen Anhaltspunkt gibt in welche Richtung er den Partikel schieben soll. Vermutlich wird beim nächsten Zeitschritt das Dreieck in das andere eindringen und von der Konturminimierung behandelt oder beide Primitive entfernen sich voneinander und es greift der TriangleSelfCollisionDetector.

Dieses Problem ist eher theoretischer Natur und konnte bei den Tests nie beobachtet werden. Wenn dieses Problem aber irgendwann einmal konkret werden sollte, könnte es evtl. dadurch behoben werden, wenn die Konturminimierung auf den generierten Dreiecken der Stoffdicke arbeitet.

7 Konklusion

In dieser Diplomarbeit wurde ein Verfahren vorgestellt, welches zum Erkennen und Auflösen von Kollisionen geeignet ist. Die spezielle Anforderung war, dass es sowohl Kollisionen zwischen disjunkten Körpern, aber primär Selbstkollisionen, korrekt erkennt und auflöst. Dabei war anfangs zu überlegen, ob eine physikalische Behandlung der Kollisionsantwort sinnvoll ist, da es sich um einen physikalisch illegalen Zustand handelt. Abbildung 4-3 und Abbildung 4-4 In dieser Diplomarbeit wurde ein Verfahren vorgestellt, welches zum Erkennen und Auflösen von Kollisionen geeignet ist. Die spezielle Anforderung war, dass es sowohl Kollisionen zwischen disjunkten Körpern, aber primär Selbstkollisionen, korrekt erkennt und auflöst. Dabei war anfangs zu überlegen, ob eine physikalische Behandlung der Kollisionsantwort sinnvoll ist, da es sich um einen physikalisch illegalen Zustand handelt.

In den Testszenarien in den Abschnitten 6.2.1, 6.2.2 und 6.2.3 wird sehr gut deutlich, dass das Verfahren sehr robust und praxistauglich mit Selbstkollisionen umgeht. Speziell in Kapitel 6.2.3 ist zu sehen, dass die Konturminimierung nicht nur für statische Simulationen zu gebrauchen ist, sondern auch in Animationen sehr gut einsetzbar ist. Ein weiterer Vorteil des Verfahrens ist, dass es auch mit Kollisionen zwischen verschiedenen Kleidungsstücken sehr gut zurecht kommt. Der Vorteil dabei ist, dass dadurch in manchen Fällen auf das rechenintensive Layering verzichtet werden kann. In Abschnitt 6.2.4 und 6.2.5 wird dies nochmals genau gezeigt.

Zwar gibt es einige wenige Limitationen in dem Verfahren, diese sind aber teilweise theoretischer Natur und treten nur in sehr unwahrscheinlichen Fällen auf. Die Limitationen wurden in Abschnitt 6.5 beschrieben.

Wie Abschnitt 6.3 zeigt, ist die Konturminimierung zusätzlich noch sehr rechenextensiv. Der zusätzliche Rechenaufwand fällt kaum ins Gewicht. Aus diesem Grund kann der Vorteil der Konturminimierung ohne Nachteile genutzt werden, was gerade den Benutzern von Vidya sehr entgegen kommt. Da das Verfahren an sich sehr wenig Rechenzeit benötigt, wurde auch darauf verzichtet, dynamisch oder durch Benutzerinteraktion zwischen dem globalen und dem lokalen Verfahren umzuschalten. Den einzigen Vorteil, den das lokale Verfahren gegenüber dem globalen hat ist, dass es keine zusammenhängenden Konturen erstellen muss. Da aber im normalen Anwendungsfall sehr wenige und kleine Durchdringungen auftreten, mal abgesehen von evtl. initialen Durchdringungen, erhöht das Erstellen der Konturen die Rechenzeit so gut wie kaum. Aus diesem Grund wird die bessere Kollisionsantwort des globalen Verfahrens auch bei kleinen Durchdringungen genutzt. Trotzdem ist der Quellcode für das lokale Verfahren jederzeit aktivierbar, für den Fall, dass es doch irgendwann einmal noch genutzt werden sollte.

Dank der doch recht einfachen Implementierung und Funktionsweise der Konturminimierung, konnte sie auch fast problemlos mit dem TriangleSelfCollisionDetector kombiniert werden und arbeitet mit den anderen Kollisionsdetektoren sehr gut zusammen. Dies war auch, neben dem Performancevorteil, der Hauptgrund, weshalb die beiden Verfahren kombiniert wurden.

Trotzdem kann durch die modulare Entwicklung die Konturminimierung jederzeit ein- oder ausgeschaltet werden, sollte es doch zu unvorhersehbaren Problemen kommen.

Der Gewinn für Vidya ist dabei sehr hoch, da jetzt auch Kleidungsstücke simuliert werden können, welche vorher sehr gerne zu Selbstkollisionen geneigt haben. Es können auch Stoffe die sehr weich sind, nun sehr gut simuliert werden. Gerade dabei hat die bestehende Kollisionsbehandlung bisher starke Probleme gehabt. Die einzige Möglichkeit bisher war es, den Stoff per Hand mühsam zu entwirren.

8 Zusammenfassung und Ausblick

In dieser Diplomarbeit wurde ein Verfahren erarbeitet, mit dem das Problem der Selbstkollision in Vidya gelöst werden konnte. Als Grundlage wurde die Konturminimierung von Volino, et al. [2006] verwendet. Dieses Verfahren wurde auch näher analysiert und die Funktionsweise, sowie die Vorteile, als auch die Limitationen dokumentiert.

Im ersten Schritt wurde sich ein Überblick über die vorhandenen Verfahren in Vidya verschafft und anschließend analysiert, welche Ansätze es zur Lösung der Selbstkollision schon gibt. Dabei wurde dann untersucht, welche Vor- und Nachteile diese Verfahren haben und welches die größten Erfolgsaussichten hat. Anschließend wurde dieses Verfahren prototypenhaft implementiert, um die ersten Ergebnisse begutachten zu können. Dabei stellte sich die Konturminimierung als sehr robust heraus und es konnte weiter darauf aufgebaut werden.

Weiterhin wurde eine physikalische Komponente entwickelt, mit der sich die Kollisionsantwort physikalisch plausibel anpassen lässt, um die Qualität der Simulation zu steigern. Für die nötigen Designparameter des Verfahrens wurde eine Möglichkeit entwickelt, diese dynamisch anhand der Dreiecksflächen zu bestimmen. Dadurch funktioniert die Auflösung von Selbstkollisionen so gut wie ohne Benutzerinteraktion. Für den Fall, dass die Kollisionsantwort dennoch zu stark oder schwach sein sollte, hat der Benutzer die Möglichkeit, die Antwort zu verstärken oder abzuschwächen.

Da das Verfahren im Anschluss der Diplomarbeit produktiv eingesetzt wird, ist auch die komplette Implementierung dokumentiert und einige Datenstrukturen und Hierarchien beschrieben. Dies dient auch gleichzeitig als Entwicklerdokumentation, um sich einen Überblick über das Verfahren zu verschaffen.

Anschließend wurden durch ausführliche Testszenen und Zeitmessungen die Effizienz und die Stabilität der Implementierung unter Beweis gestellt. Es wurden aber auch Probleme beschrieben, die gelöst wurden oder noch weiterhin bestehen.

Des weiteren sind bei der Entwicklung viele grundsätzliche Entscheidungen getroffen worden. Die Vor- und Nachteile, sowie die Gründe für die Entscheidungen und deren Alternativen wurden ausführlich dokumentiert

Um die Konturminimierung noch besser in Vidya einsetzbar zu machen, kann sie evtl. mit dem Layering kombiniert werden. Wenn es machbar ist, dass beide auf denselben Datenstrukturen arbeiten, wird die Performance noch mal drastisch gesteigert, da das Layering dann auf derselben BVH wie die Konturminimierung arbeiten kann und damit die Traversierung und das Aktualisieren der BVH wegfällt. Dies hat schon bei der Kombination der Konturminimierung mit dem TriangleSelfCollisionDetector zu einer drastischen Steigerung der Performance geführt. Zudem

könnte es zu einer Verbesserung der Simulationsqualität führen, da es aktuell sein kann, dass die Konturminimierung mit der integrierten Antwort vom TriangleSelfCollisionDetector und das Layering an einigen Extremstellen, z.B. Hemdtaschen, gegeneinander Arbeiten und so eine unruhige Simulation entsteht.

Ein anderer Ansatz die Konturminimierung zu verbessern wäre es, eine bessere Möglichkeit zu finden, den Skalierungsfaktor h_0 besser automatisch anzupassen, so dass der Benutzer noch weniger oder sogar nie manuell in die Korrektur eingreifen muss, sondern alles automatisch passiert. Dazu wäre eine Idee, sich Pfade von einem Zeitschritt zum nächsten zu merken und zu schauen, ob diese nicht kleiner oder gar größer geworden sind. Sollte dies korrekt identifiziert werden, könnten entsprechende Gegenmaßnahmen automatisch greifen.

Auch bleibt mit der Konturminimierung noch ein Problem. Wenn der Avatar geometrisch z.B. in der Achselhöhle oder beim Animieren in den Kniekehlen nicht genügend Spielraum für den Stoff lässt, fangen diese Bereiche stark an zu flattern, da die Kollision mit dem Avatar, die Konturminimierungen und das Layering stark gegeneinander arbeiten. Hier wäre es interessant den Ansatz von Baraff, et al. [2003] genauer zu untersuchen und zu schauen, ob das dort beschriebene Flypapern doch praktisch einsetzbar ist.

Abbildungsverzeichnis

ABBILDUNG 2-1 ZWEI TEXTILSTÜCKE, DIE NUR DURCH DEN TRIANGLESELFCOLLISIONDETECTOR AUF ABSTAND GEHALTEN WERDEN.....	7
ABBILDUNG 2-2 BLUSE ÜBER ROCK (L.) ODER ROCK ÜBER BLUSE (R.)	9
ABBILDUNG 2-3 KAPUZE UND SPORTJACKE MIT STARKEN DURCHDRINGUNGEN.....	10
ABBILDUNG 3-1 SCHAL MIT SELBSTDURCHDRINGUNG. – ABBILDUNG AUS METZGER [2001, S. 41]	12
ABBILDUNG 3-2 BESONDERS KRITISCHE SITUATION BEIM AUFLÖSEN VON DURCHDRINGUNGEN, DA DORT VIELE SELBSTDURCHDRINGUNGEN AUFTRETEN UND DER AVATAR REL. WENIGE PLATZ ZUM INTERAGIEREN BIETET. – ABBILDUNG AUS BARAFF, ET AL. [2003, S. 3]	15
ABBILDUNG 3-3 STARKE DURCHDRINGUNG ZWEIER ÜBEREINANDERLIEGENDER KLEIDUNGSSTÜCKE. DIE KOLLISIONSKONTUREN MARKIEREN DIE EINZELNEN DURCHDRINGENEN REGIONEN.	18
ABBILDUNG 3-4 SCHNITTKANTE DER BEIDEN POLYGONE (ROT).	20
ABBILDUNG 3-5 SKALIERTER VEKTOR G , WELCHER E IN RICHTUNG R_i RELATIV ZU A VERSCHIEBT.	20
ABBILDUNG 3-6 VERSCHIEBEVEKTOREN NACH DEM LOKALEM VERFAHREN.	22
ABBILDUNG 3-7 VERSCHIEBEVEKTOREN NACH DEM GLOBALEM VERFAHREN.	22
ABBILDUNG 3-8 LIMITATIONEN DER KONTURMINIMIERUNG - ABBILDUNG AUS VOLINO, ET AL. [2006, S. 6].....	24
ABBILDUNG 3-9 AUFTEILUNG DES KRAFTSTOßES AUF DIE PARTIKEL DER KANTE UND DES DREIECKS.....	26
ABBILDUNG 4-1 SONDERFÄLLE BEI DER KOLLISION ZWEIER DREIECKE. OBEN LINKS BERÜHRUNG EINES PARTIKELS MIT EINEM ANDEREN DREIECK. UNTEN LINKS ZWEI PLANARE DREIECKE. OBEN RECHTS SCHNITTGERADE GENAU AN EINER KANTE.	30
ABBILDUNG 4-2 NORMALFÄLLE BEI DER KOLLISION ZWEIER DREIECKE.	30
ABBILDUNG 4-3 ANTWORTVEKTOREN OHNE PHYSIKALISCHE KORREKTUR. DIE ANTWORT FÜR JEDEN BETEILIGTEN PARTIKEL IST GLEICH OBWOHL DAS KARIERTE STOFFSTÜCK (RECHTS) ETWA 6-MAL SCHWERER IST.....	34
ABBILDUNG 4-4 ANTWORTVEKTOREN MIT PHYSIKALISCHER KORREKTUR. DAS KARIERTE STOFFSTÜCK (RECHTS) BEKOMMT EINE DEUTLICH SCHWÄCHERE KOLLISIONSANTWORT, DA ETWA 6-MAL SCHWERER IST.	34
ABBILDUNG 4-5 ANPASSUNG DER KOLLISIONSPUNKTE DURCH DIE SPATIALHASHMAP UM FLOATUNGENAUGKEIT AUSZUGLEICHEN.	35
ABBILDUNG 5-1 SELBSTDURCHDRINGUNG BEI DEM ZWEI POLYGONZÜGE ANHAND ZUSAMMENHÄNGENDER DREIECKE UNTERSCHIEDEN WERDEN. EIN KANTEN-POLYGONZUG MIT ROTEN KOLLISIONSPUNKTEN UND EIN KANTEN- POLYGONZUG MIT GRÜNEN KOLLISIONSPUNKTEN.	43
ABBILDUNG 5-2 DARSTELLUNG EINE KOLLISIONSKONTUR AUF NUR EINEM KLEIDUNGSSTÜCK, DURCH DIE UNTERSCHIEDUNG ZWEIER GETRENNTER POLYGONZÜGE.	43
ABBILDUNG 6-1 AUSGANGSPOSITION EINES T-SHIRTS MIT SELBSTDURCHDRINGUNGEN AUF DER ARMINNENSEITE.....	47
ABBILDUNG 6-2 DER BLAUE STOFF DURCHDRINGT DEN ROTEN ARMSTOFF AUF DER INNENSEITE.....	48
ABBILDUNG 6-3 NACH ANWENDUNG DES GLOBALEM VERFAHRENS IST DIE DURCHDRINGUNG KORREKT AUFGELOST WORDEN.	48
ABBILDUNG 6-4 PROBLEMBEREICH ARMBEUGE, DABEI DURCHDRINGT SICH DER ÄRMEL STARK SELBER.	49
ABBILDUNG 6-5 DER QUERSCHNITT ZEIGT DIE EINZELNEN DREIECKE WELCHE SICH ÜBERSCHNEIDEN.	50
ABBILDUNG 6-6 AUCH HIER FÜHRT DAS GLOBALE VERFAHREN ZU EINER DEUTLICHEN VERBESSERUNG DER SITUATION UND ZU EINEM ÜBERSCHNEIDUNGSFREIEN ZUSTAND. VORAUSSETZUNG IST ALLERDINGS DASS DER AVATAR GENÜGENDE SPIELRAUM FÜR DEN STOFF ZUR VERFÜGUNG STELLT.	50
ABBILDUNG 6-7 TOTALE DER SZENE OHNE SELBSTDURCHDRINGUNGEN IN DER ARMBEUGE.	51
ABBILDUNG 6-8 STARKE SELBSTDURCHDRINGUNGEN BEI DER ROTATION EINER KUGEL UNTER DEM STOFF.	52
ABBILDUNG 6-9 TROTZT LAUFENDER ANIMATION SIND NACH ~15 ITERATIONEN DIE DURCHDRINGUNGEN STARK REDUZIERT.	52
ABBILDUNG 6-10 NACH INSGESAMT ~50 ITERATIONEN IST DIE SZENE FREI VON ALLEN DURCHDRINGUNGEN.	53
ABBILDUNG 6-11 AUCH WEITERHIN WÄHREND DER ANIMATION FREI VON DURCHDRINGUNGEN.	53
ABBILDUNG 6-12 DURCHDRINGUNG ZWEIER VERSCHIEDENER KLEIDUNGSSTÜCKE IN DER TOTALEN.	54
ABBILDUNG 6-13 DURCHDRINGUNG ZWEIER VERSCHIEDENER KLEIDUNGSSTÜCKEN MIT EINGEZEICHNETEN KOLLISIONSKONTUREN.	55
ABBILDUNG 6-14 DURCHDRINGENEN BEREICH NACH ERFOLGREICHER AUFLÖSUNG DER DURCHDRINGUNGEN.	55
ABBILDUNG 6-15 TESTSZENE MIT ZWEI DURCHDRINGENEN STOFFTEILEN UND EINGEZEICHNETER KOLLISIONSKONTUR... ..	56
ABBILDUNG 6-16 REDUKTION DER KOLLISIONSKONTUR NACH ~15 ITERATIONEN.....	57

ABBILDUNG 6-17 REDUKTION DER KOLLISIONSKONTUR NACH ~30 ITERATIONEN.....	57
ABBILDUNG 6-18 REDUKTION DER KOLLISIONSKONTUR NACH ~45 ITERATIONEN.....	58
ABBILDUNG 6-19 KOMPLETTE AUFLÖSUNG DER DURCHDRINGUNG NACH CA. ~55 ITERATIONEN.....	58

Diagrammverzeichnis

DIAGRAMM 2-1 BISHERIGE ABLAUFREIHEFOLGE DER KOLLISIONSDETEKTOREN IN VIDYA.	4
DIAGRAMM 3-1 HISTORIE ÜBER DIE ENTWICKLUNG DER KOLLISIONSERKENNUNGSSYSTEME FÜR DEFORMIERBARE KÖRPER.	12
DIAGRAMM 4-1 SCHEMATISCHE DARSTELLUNG DES KOMPLETTEN ABLAUFS DES IMPLEMENTIERTEN VERFAHRENS.	27
DIAGRAMM 4-2 FLUSSDIAGRAMM ÜBER DEN KOMPLETTEN ABLAUF DER KONTURMINIMIERUNG.	37
DIAGRAMM 4-3 KLASSENDIAGRAMM ALLER KLASSEN DIE FÜR DIE DIPLOMARBEIT VERWENDET WURDEN.....	39
DIAGRAMM 5-1 ABLAUF DER KOLLISIONSDETEKTOREN NACH DER KOMBINATION MIT DEM TRIANGLESELFCOLLISIONDETECTOR.	45
DIAGRAMM 6-1 PROZENTUALE DARSTELLUNG DES OVERHEADS DURCH DIE KONTURMINIMIERUNG.	59
DIAGRAMM 6-2 PROZENTUALE DARSTELLUNG DES GESCHWINDIGKEITZUWACHS DURCH KOMBINATION DES TRIANGLESELFCOLLISIONDETECTORS MIT DER KONTURMINIMIERUNG.	60

Literatur- und Quellenverzeichnis

Baraff, D. „Curved Surfaces and Coherence for Non-Penetrating.“ *ACM Computer Graphics*, 24. 1990. 19-28.

Baraff, D., A. Witkin, und M. Kass. „Untangling Cloth.“ *ACM SIGGRAPH 2003 Papers*. San Diego, California, 2003. 862-870.

Baraff, D., und A. Witkin. „Dynamic Simulation of Non-Penetrating Flexible Bodies.“ *Computer Graphics (SIGGRAPH'92 proceedings)* . 1992. 303-308.

—. „Large Steps in Cloth Simulation.“ *Computer Graphics (SIGGRAPH'98 proceedings)*. 1998. 106-117.

—. „Linear Time Dynamics Using Lagrange Multipliers.“ *Computer Graphics (SIGGRAPH'96 proceedings)*. 1996. 137-146.

Bridson, R., R. Fedkiw, und J. Anderson. „Robust Treatment of Collisions, Contact and Friction for Cloth Animation.“ *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. San Antonio, Texas, 2002. 594-603.

Curtis, S., R. Tamstorf, und D. Manocha. „Fast Collision Detection for Deformable Models using Representative-Triangles.“ *Symposium on Interactive 3D Graphics*. California, 2008. 61-69.

Eberly, David H. *Game Physics*. Morgan Kaufmann, 2004.

Fuhrmann, A., C. Groß, und A. Weber. „Ontologies for Virtual Garments.“ *Fraunhofer Publica List*. Darmstadt: Fraunhofer, 2005.

Hutter, M., und A. Fuhrmann. „Optimized Continuous Collision Detection for Deformable Triangle Meshed.“ *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. Redwood City, California, 2007. 61-69.

Metzger, J. „Effiziente Kollisionsdetektion in der Simulation von Textilien.“ *Diplomarbeit*. Tübingen: Universität Tübingen, 2001.

Provot, X. „Collision and self-collision handling in cloth model dedicated to design garment.“ *Graphics Interface*. 1997. 177–189.

—. „Deformation constraints in a mass-spring model Deformation constraints in a mass-spring model.“ *In Graphics Interface*. 1995. 147–154.

Teschner, M., et al. „Collision Detection for Deformable Objects.“ *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. Redwood City, California, 2005. 61-69.

Volino, P., und N. Magneta-Thalmann. „Accurate Collision Response on Polygonal Meshes.“ *Proceedings of the Computer Animation*. 2000. 154.

- . „Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces.“ *In Comp. Anim. and Simulation*. Springer-Verlag, 1995. 55–65.
- . „Developing simulation techniques for an interactive clothing system.“ *In Proc. of the 1997 International Conf. on Virtual Systems and MultiMedia, IEEE*. 1997. 109–118.
- . „Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity.“ *Proc. of Eurographics, vol. 13 of Computer Graphics Forum*. 1994. 155–166.
- . „Resolving Surface Collisions through Intersection Contour Minimization.“ *ACM SIGGRAPH 2006 Papers*. Boston, Massachusetts, 2006. 1154-1159.
- Wikipedia. *Wikipedia - Die freie Enzyklopädie*. 2009/2010. <http://www.wikipedia.de>.

Verwendete Software

In der unten stehenden Tabelle findet sich sämtliche Software, welche bei der Umsetzung der Diplomarbeit genutzt wurde.

Einsatzgebiet	Bezeichnung	Version	Homepage
Dokumentation	Microsoft Office	2009	http://office.microsoft.com
Entwicklungsumgebung	Eclipse	3.5	http://www.eclipse.org
Programmiersprache	Java	1.6	http://www.java.com
Bildbearbeitung	Gimp	2.6	http://www.gimp.org
Diagramme	Visual Paradigm	7.2	http://www.visual-paradigm.com
Bekleidungssimulation	Vidya	2.0.1	http://www.assyst.de/
Versionisierungstool (Eclipse)	Subclipse	1.6	http://subclipse.tigris.org
Versionisierungstool (Windows)	Tortoise	1.6	http://tortoisesvn.tigris.org/

Anhang

Auf der beiliegenden CD befinden sich alle verwendeten Grafiken, Diagramme, sowie alle genutzten Papers. Auch sind die Visual Paradigm Projektdateien auf der CD vorhanden. Die Diplomarbeit selbst ist als PDF-Datei auf der CD zu finden. Nicht auf der CD ist ein Quellcode, da die Arbeit von der assyst GmbH gesperrt wurde.

Verzeichnis	Bemerkung
/Diplomarbeit	Diplomarbeit als PDF-Datei.
/Grafiken	Alle in der Diplomarbeit genutzten Grafiken.
/Diagramme	Alle in der Diplomarbeit genutzten Diagramme.
/Papers	Alle während der Diplomarbeit genutzten Papers.
/VP Projekte	Visual Paradigm Projekte in denen die Diagramme erstellt wurden: • Diplomarbeit.vpp – Alle Diagramme außer dem Flussdiagramm • Flussdiagramm.vpp – Nur das Flussdiagramm Alle Diagramme wurden mit der Community Edition von Visual Paradigm in der Version 7.2 erstellt.