Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Philipp Roßberger

Entwicklung einer Anwendung zur physikbasierten Manipulation von Objekten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einleitung

In aktuellen Forschungsarbeiten von Kruger u. a. (2005) und Agarawala und Balakrishnan (2006) aus dem Umfeld der Human Computer Interaction (HCI) werden neuartige Interaktionstechniken vorgestellt, die ein simultanes Verschieben und Drehen von Objekten auf Bedienoberflächen von Computern erlauben. Dies wird durch den Einsatz spezieller Algorithmen bzw. Software-Bibliotheken ermöglicht, mit deren Hilfe physikalische Gesetze und Charakteristiken von Gegenständen nachgebildet werden können. Auf diese Weise ist eine physikbasierte Interaktion mit Objekten in Computerprogrammen möglich.

Wie Tests mit Versuchspersonen erwiesen haben, sind physikbasierte Interaktionstechniken schnell erlernbar (Agarawala und Balakrishnan, 2006) und herkömmlichen Techniken zur Translation und Rotation im Hinblick auf den benötigten Interaktionsaufwand überlegen (Kruger u. a., 2005).

Auf Basis dieser Erkenntnisse wurde vom Autor im Rahmen des Semesterprojekts ein Prototyp entwickelt, der eine physikbasierte Interaktion erlaubt. Die vorliegende Ausarbeitung präsentiert die zur Realisierung des Prototypen ausgeführten Arbeiten und dabei gesammelten Erkenntnisse.

Der Prototyp konzentriert sich auf eine simultane Rotation und Translation von Objekten. Dabei werden Aspekte aus den Arbeiten von Kruger u. a. (2005) und Agarawala und Balakrishnan (2006) in einem Programm vereint. Eine nähere Beschreibung dieser beiden und weiterer verwandter Arbeiten findet sich in Roßberger (2007a). Die hier beschriebenen Erkenntnisse aus dem Projekt werden in die Masterarbeit des Autors einfließen, deren Inhalte in Roßberger (2007b) vorgestellt werden.

Im folgenden Kapitel 2 dieser Ausarbeitung wird das konkrete Projektvorhaben erläutert. Danach werden in Kapitel 3 Technologien beschrieben, die zur Implementation des Projektvorhabens erforderlich waren. Die Bearbeitung des Projekts beinhaltete eine Reihe von Aufgabenfeldern, die in Kapitel 4 vorgestellt werden. Abschließend wird in Kapitel 5 ein Zusammenfassung über die Inhalte dieser Arbeit und ein Ausblick hinsichtlich der Weiterverwendung der gesammelten Ergebnisse gegeben.

2 Projektvorhaben

Ziel des Projekts war die Entwicklung einer Applikation mit der Objekte auf einer Ebene gleichzeitig verschoben und gedreht werden können. Dazu wird zunächst in einer dreidimensionalen Darstellung ein Punkt auf der Oberfläche eines Objekts mit dem Cursor ausgewählt. Nach Selektion des Punktes soll das Objekt so auf Cursorbewegungen reagieren, wie man es in der Realität bei der Bewegung eines Fingers in Berührung mit einem Gegenstand auf einer Tischfläche erwarten würde.

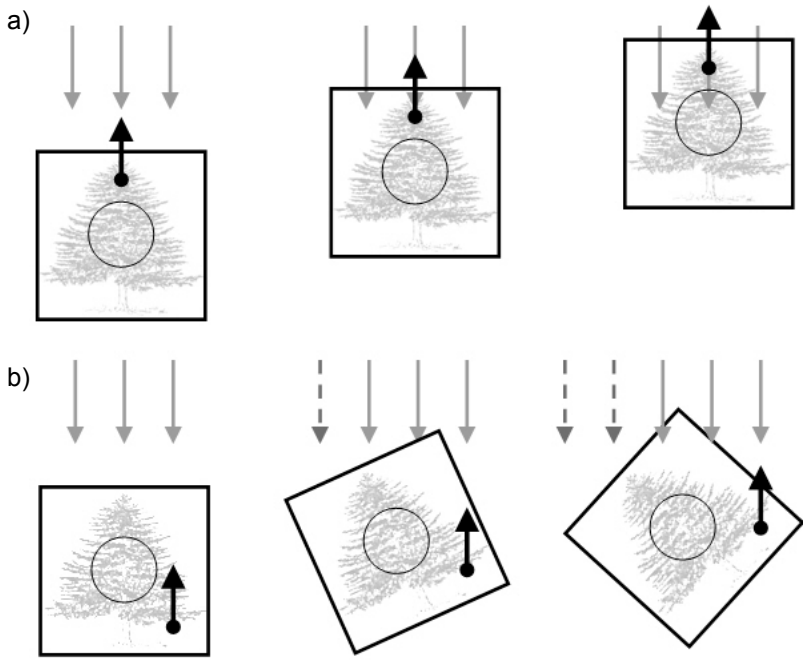

Abb. 2.1: Während sich das Objekt in Reihe a) nur verschiebt, wird bei b) aufgrund des veränderten Berührungspunktes gleichzeitig eine Drehung ausgelöst. Abbildungen aus Kruger u. a. (2005).

Wie in Abbildung 2.1 zu sehen, spielt die Position des Berührungspunkts eine entscheidende Rolle für das Verhalten des gehaltenen Objekts. Aufgrund von Reibungskräften, die entgegengesetzt zur Bewegungsrichtung wirken, kommt es beim Verschieben des Objekts zu einer Drehbewegung, die der Translationsbewegung überlagert ist.

Die Reaktionen des Objekts auf Reibungskräfte sollte mit Hilfe der Physik-Engine ODE[1] (Open Dynamics Engine) simuliert werden. ODE erlaubt es, Objekte mit physikalischen Charakteristiken wie Masse und Oberflächenbeschaffenheit zu versehen. Diese führen in Kombination mit den Eigenschaften des Untergrunds zu einem Reibungskoeffizienten, der die Reaktionen des Objekts maßgeblich beeinflusst.

Zur Realisierung des Projekvorhabens wurden mehrere Softwarekomponenten ausgewählt, auf die im folgenden Kapitel eingegangen wird.

[1] http://www.ode.org/

3 Eingesetzte Technologien

Wie in Kapitel 2 erwähnt, wurde zur Modellierung der physikalischen Objektcharakteristiken und der Simulation der Objektreaktionen die Physik-Engine ODE ausgewählt. Die Wahl fiel auf ODE weil es sich dabei um eine Softwarebibliothek handelt, die als Open Source verfügbar ist. Ferner ist ODE mittlerweile relativ stabil, gut dokumentiert und wird von einer aktiven Entwicklergemeinde betreut. Ausserdem wurden vom Autor bereits bei früheren Projekten Erfahrungen mit ODE gesammelt. Bei den ausgeführten Versuchen wurde ODE in der Version 0.6 benutzt.

Zur dreidimensionalen Visualisierung der Simulationsergebnisse wurde das ebenfalls als Open-Source verfügbare OpenSG[1] (Open Scene Graph) in der Version 1.6 eingesetzt. OpenSG erlaubt es, mit Hilfe eines Szenengraphen dreimdimensionale Szenen aufzubauen und in Echtzeit zu manipulieren. Zur Darstellung der Inhalte setzt OpenSG auf OpenGL. Wie mit ODE wurden vom Autor mit OpenSG bereits bei vorangegangenen Projekten positive Erfahrungen gesammelt.

[1] http://www.opensg.org/

4 Arbeitsfelder während des Projekts

Neben der Modellierung der physikalischen Parameter beinhaltete die Umsetzung des Projektvorhabens eine Reihe weiterer Arbeitsfelder, die in den folgenden Abschnitten erläutert werden. Dazu zählt die Koppelung von ODE mit OpenSG (siehe 4.2) und die Evaluation alternativer Physik-Engines, wie in 4.3 beschrieben. Ferner war eine Einarbeitung in die Entwicklungsumgebung Visual Studio 2003, die Programmiersprache C++ und eine Vorbereitung der Hardware-Plattform von Apple erforderlich. Die dabei gesammelten Erfahrungen werden in 4.4 beschrieben.

4.1 Realisierungsansätze zur physikbasierten Interaktion

Es wurden mehrere Ansätze getestet um das, in Kapitel 2 beschriebene, gewünschte Verhalten der Objekte zu erreichen. Dies war erforderlich, da im Vorfeld nicht klar war, welche Einschränkungen ODE im Hinblick auf die Modellierung der physikalischen Zusammenhänge mit sich bringt und mit welchem Ansatz sich realitätsnahe Simulationsergebnisse erreichen lassen.

4.1.1 Fingerobjekt mit Ball-Joint

Verbindungen zwischen Objekten lassen sich mit ODE durch so genannte Joints modellieren. Bei einem Joint handelt es sich um ein Gelenk, das die Bewegungsfreiheit von Objekten zueinander einschränkt. ODE bietet unterschiedliche Joint-Typen an. Dazu zählen die in Abbildung 4.1 dargestellten Joint-Typen.

Wie in Kapitel 2 beschrieben, sollte das Verhalten eines Objekts simuliert werden, das mit Hilfe eines Fingers bewegt wird. Zwischen Objekt und Fingerspitze existiert dabei ein Kontaktpunkt. Aufgrund der verfügbaren Freiheitsgrade, die sich zwischen dem Finger und dem berührten Objekt ergeben, kann der Berührungspunkt als Kugelgelenk interpretiert werden. Auf dieser Erkenntnis basiert der in diesem Abschnitt vorgestellte Modellierungsansatz.

Bei der Umsetzung wurden zwei Objekte erzeugt, die über ein Kugelgelenk miteinander verbunden sind. Das eine Objekt repräsentiert dabei den Gegenstand auf der Tischfläche,

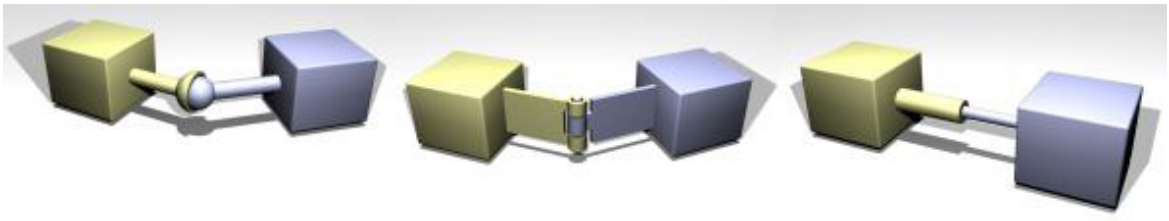

Abb. 4.1: Drei Joint-Typen, die bei ODE verfügbar sind. Zu sehen sind von links nach rechts Ball-, Hinge- und Slider-Joint. Abb. aus Smith (2006).

das andere den Finger. Das Kugelgelenk wurde mit einem Ball-Joint modelliert, dessen Teile in Abb. 4.3 gezeigt sind. Schematisch betrachtet (siehe Abb. 4.2 befindet sich der Drehpunkt des Ball-Joint dort, wo die Fingerspitze das Objekt berührt.

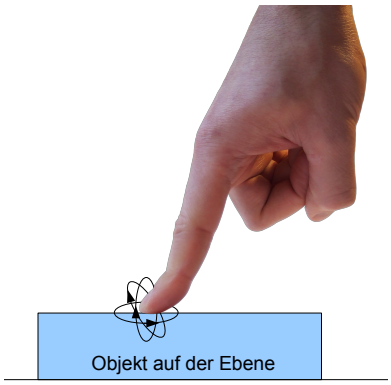

Abb. 4.2: Die drei Freiheitsgrade der Fingerspitze bei einem Kontaktpunkt.

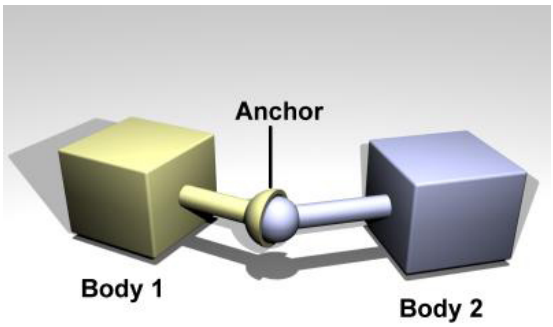

Abb. 4.3: Bestandteile eines Ball-Joint. Abb. aus Smith (2006)

Bei einer Bewegung des Cursors wird das „Fingerobjekt“ verschoben. Aufgrund der Verbindung über den Ball-Joint, sollte das Objekt auf der Ebene den Bewegungen des Fingerobjekts nun entsprechend folgen. Dies war aber nicht der Fall. Bewegungen des Cursors resultierten vielmehr in einem willkürlich wirkenden „Zappeln“ des Objekts auf der Ebene. Nach ausführlicher Recherche wurde klar, dass ODE mit der „manuellen“ Repositionierung des Fingerobjekts anhand der Cursor-Position nicht umgehen kann. Dieser direkte Eingriff widerspricht der kontinuierlichen Physik-Simulation auf der ODE basiert. Aus diesem Grund war die Realisierung des Vorhabens wie eben beschrieben nicht möglich. Wie im folgenden Abschnitt beschrieben, wurde stattdessen ein alternativer Ansatz gefunden, der mit Hilfe von Kräften funktioniert.

4.1.2 Objektmanipulation durch einen Contact-Joint

Neben den bereits in 4.1.1 beschriebenen Joint-Typen, bietet ODE auch so genannte Contact-Joints. Diese lassen sich laut Smith (2006) einsetzen, um zu verhindern, dass zwei Objekte einander durchdringen, die sich an einem Punkt berühren. Dies wird sichergestellt indem bei beiden Objekten eine Kraft am Berührungspunkt wirkt, welche die Objekte auseinander hält. Die Richtung der Kraft wird über eine Normale definiert. Diese zeigt in der Regel in die entgegengesetzte Richtung zur Bewegungsrichtung des Körpers.

Bei der Implementation wurde das Objekt auf der Ebene nicht mit einem anderen Objekt über den Contact-Joint verbunden, sondern mit *null*. Die Verbindung mit *null* bedeutet, dass sich der Joint an einer festen Stelle im Raum befindet und damit nicht den Kräften unterliegt, die auf die anderen Körper im Simulationsraum wirken.

Der soeben beschriebene Ansatz liefert zufrieden stellende Ergebnisse: das Objekt auf der Ebene reagiert realitätsnah auf Bewegungen des Cursors und die Simulation läuft stabil. Die Implementation des Verfahrens und die Reihenfolge der Berechnungsschritte wird im folgenden Absatz näher beschrieben.

Implementationsdetails

Die physikalischen Berechnungen durch ODE erfolgen vor jedem Durchlauf des Rendering-Prozesses, der Neuberechnung des auf dem Monitor gezeigten Bildes. Dabei werden bei jedem Zyklus folgende Schritte durchlaufen:

1. Zuerst wird der Punkt bestimmt, an dem das Objekt gehalten wird. Dies geschieht mit Hilfe eines Ray-Casting-Verfahrens, das in 4.2.1 näher beschrieben wird.
2. Der Greifpunkt liegt jetzt in globalen Koordinaten vor. Diese müssen nun in das relative Koordinatensystem des Körpers umgerechnet werden, damit die Kraftnormale für den Contact-Joint mit den richtigen Werten erzeugt werden kann.
3. Danach wird ein Contact-Joint erzeugt, der mit dem Objekt auf der Ebene und *null* verbunden wird. Die Kraftnormale des Contact-Joint zeigt vom Greifpunkt an der Oberfläche in Richtung des Betrachters. Dadurch „klebt“ das Objekt unterhalb des Cursors.
4. Im Anschluss werden die Positionen aller Körper berechnet, die mit ODE modelliert wurden. Die berechneten Werten bilden den Endzustand des physikalischen Systems für diesen Durchlauf.
5. Zuletzt wird der erzeugte Contact-Joint zerstört, damit der Contact-Joint beim nächsten Zyklus an der aktuellen Position des Cursors erzeugt werden kann.

ODE ist lediglich für die Berechnung des modellierten physikalischen Modells zuständig. Die Darstellung der dreidimensionalen Szene und der darin enthaltenen Körper wird von OpenSG übernommen, wie in 3 bereits erläutert wurde. Der folgende Abschnitt beschreibt Schnittstellen zwischen ODE und OpenSG und wie Werte zwischen diesen beiden Software-Bibliotheken im entwickelten Prototyp ausgetauscht werden.

4.2 Koppelung von OpenSG und ODE

ODE und OpenSG tauschen im hier entwickelten Prototypen an zwei Stellen im Programmcode Informationen miteinander aus. Diese werden in den beiden folgenden Abschnitten beschrieben.

4.2.1 Ray-Casting

Wie bereits in 4.1.2 beim ersten Aufzählungspunkt erwähnt, wird der Greifpunkt am Objekt mittels Ray-Casting bestimmt.

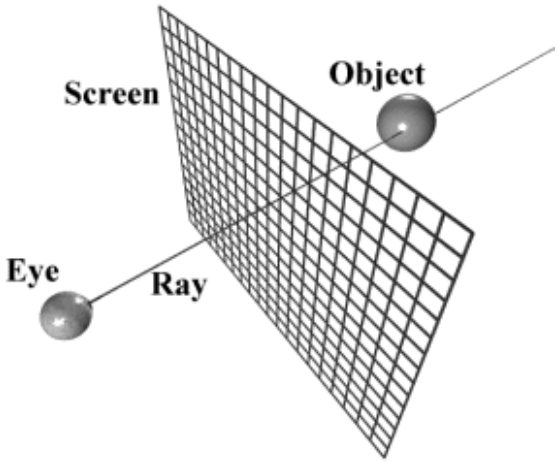

Abb. 4.4: Prinzip des Ray-Casting. Abb. von Buck (1999).

Abbildung 4.4 zeigt die Funktionsweise des Ray-Tracing-Verfahrens. Dabei wird ein Strahl (Ray) von einem Beobachtungspunkt (Eye) orthogonal zur Projektionsfläche des Bildes (Screen) in die dreidimensionale Szene geschickt. Der Strahl trifft nun unter Umständen auf ein Objekt (Object) in der Szene. Das erste Objekt auf das der Strahl trifft ist das nächstgelegene zum Betrachter.

Im Prototypen wird Ray-Casting benutzt um festzustellen ob sich ein Objekt unter dem Cursor befindet. Dazu wird von der Position des Cursors ein Strahl in die dreidimensionale Szene geschickt. Sowohl OpenSG als auch ODE bieten Ray-Casting-Funktionalität. Es wurden jedoch die Methoden von OpenSG benutzt, da diese erlauben die Cursor-Position zu bestimmen.

Um den Contact-Joint zu erzeugen werden die von OpenSG berechneten Koordinaten des Auftreffpunkts im ODE-Programmteil übernommen. Dies ist möglich weil das Koordinatensystem von OpenSG unverändert auf ODE übertragen werden kann.

4.2.2 Visualisierung des physikalischen Modells

Nach Berechnung der Position des Objekts auf der Ebene durch ODE (siehe 4.1.2, Aufzählungspunkt 4), muss dieses auf dem Bildschirm dargestellt werden. Dazu ist eine Übertragung der Objektposition und -translation von ODE zu OpenSG nötig.

Im Prototyp wurde das Objekt auf der Ebene zweimal erzeugt: im physikalischen Modell von ODE und im Szenengraphen von OpenSG. Die Position des Objekts, also dessen X-, Y- und Z-Koordinaten lassen sich direkt vom ODE-Objekt zum OpenSG-Objekt übertragen. Die Translation wird mittels Quaternionen von ODE zu OpenSG übernommen. Quaternionen erlauben es Körperrotationen anhand eines Vektors und eines Rotationswinkel anzugeben. Diese werden in der Computergrafik häufig angewandt, da bestimmte Berechnungen mit Quaternionen leichter durchzuführen sind als mit Rotationsmatrizen.

4.3 Evaluation anderer Physik-Engines

Aufgrund der Probleme beim ersten Implementationsansatz (siehe 4.1.1), wurde neben ODE die Physik-Engine Newton[1] getestet. Dabei stellte sich heraus, dass im Newton-SDK[2] ein Beispiel-Programm enthält, in dem das gewünschte Verhalten zum Greifen eines Objekts an einer bestimmten Stelle bereits implementiert ist.

Da Newton jedoch erst am Ende des Projekts getestet wurde, gab es zu diesem Zeitpunkt mehr Erfahrung mit der Bedienung und Funktionsweise von ODE. Aus diesem Grund wurde die Arbeit mit ODE fortgesetzt. Das Resultat ist der in Abschnitt 4.1.2 beschriebene Implementationsansatz.

[1] http://www.newtondynamics.com/
[2] Software Development Kit

Zu Beginn der Arbeiten an der Master-Thesis muss daher eine Entscheidung zwischen ODE und Newton getroffen werden. Da das gewünschte Verhalten inzwischen mit ODE realisiert wurde, ist es nahe liegend weitere Arbeiten mit dieser Engine durchzuführen. Bei den Tests erwies sich Newton allerdings beim ersten Eindruck als besser dokumentiert als ODE. Ferner kann mit Newton die Oberflächenbeschaffenheit der Objekt einfacher konfiguriert werden. Aus diesen Gründen ist es vermutlich sinnvoll zu testen, wie gut sich Newton mit OpenSG koppeln lässt und aufgrund der dabei gesammelten Erfahrungen eine Entscheidung zu treffen.

4.4 Entwicklungsumgebung und Hardware-Plattform

Die Entwicklung des Protoypen erfolgte mit C++ und Visual Studio 2003. Trotz geringer Erfahrung mit diesen Technologien konnte die Implementation relativ problemlos durchgeführt werden. Dies lag vor allem daran, dass der Prototyp in ein von OpenSG vorgegebenes Beispielprogramm eingebettet wurde und damit ein Großteil des Programmcodes wieder verwendet werden konnte. Auch das Hinzufügen weiterer Programmteile konnte teilweise durch die Übernahme von Code aus anderen Beispielprogrammen von ODE und OpenSG vereinfacht werden.

Auf Seiten der Hardware wurde die Entwicklung zu Beginn des Projekt auf einem Mac Mini mit einem Intel DuoCore-Prozessor durchgeführt, auf dem mit Hilfe von Apple Bootcamp[3] Windows XP Professional installiert wurde. Der Intel GMA 945 Chipsatz im Mac Mini unterstützt leider nicht alle OpenGL-Befehle und weist im Vergleich zu aktuellen 3D-Grafikkarten eine relativ schlechte Leistung bei der Darstellung von dreidimensionalen Inhalten auf. Da OpenSG aber auf OpenGL basiert, wurde die Implementierung auf einem Notebook-PC mit einer ATI Radeon X700 fortgesetzt. Auf diesem Gerät konnte der Prototyp mit voller OpenGL-Unterstützung und deutlich besseren Frameraten ausgeführt werden.

[3] http://www.apple.com/de/macosx/bootcamp/

5 Zusammenfassung und Ausblick

Die vorliegende Ausarbeitung beschreibt die Arbeiten des Autors bei der Entwicklung eines Prototypen zur physikbasierten Manipulation von Objekten in einer dreidimensionalen Darstellung.

In Kapitel 2 wird zunächst das Projektvorhaben vorgestellt und der beabsichtigte Funktionsumfang des Prototypen erklärt. Dieser ermöglicht es durch Einsatz der Physik-Engine ODE ein Objekt auf einer Ebene mit Hilfe eines Kontaktpunktes gleichzeitig zu verschieben und zu drehen.

Zur Umsetzung des Vorhabens wurden zwei Software-Bibliotheken verwendet, die in Kapitel 3 beschrieben werden. Dabei handelt es sich neben ODE um das Visualisierungsframework OpenSG. Dieses ermöglicht es, die von ODE berechneten Simulationsergebnisse dreidimensional darzustellen.

Die Entwicklung des Prototyps erforderte die Auseinandersetzung mit verschiedenen Arbeitsfeldern, die in Kapitel 4 erläutert werden. Ein Schwerpunkt dabei war, wie in 4.1 dargelegt, die Entwicklung eines physikalischen Modells, das eine physikbasierte Interaktion nach den in 2 formulierten Zielen erlaubt. Ferner musste ODE mit OpenSG gekoppelt werden. Die dazu erforderlichen Schnittstellen werden in 4.2 ausgeführt.

Da zu Beginn des Projekts die gewünschte physikbasierte Interaktion nicht mit ODE realisiert werden konnte, wurde eine weitere Physik-Engine namens Newton getestet. In Abschnitt 4.3 findet sich u.a. ein Vergleich dieser Engine mit ODE.

Am Ende der Ausarbeitung berichtet Teil 4.4 über Erfahrungen mit der Entwicklungsumgebung Visual Studio 2003, der Programmiersprache C++ und der eingesetzten Hardware-Plattform.

Die über das Projekt gesammelten Erfahrungen, werden in die Masterarbeit des Autor einfließen, die in Roßberger (2007b) ausführlich vorgestellt wird. Trotz anfänglicher Schwierigkeiten, konnte während des Projekts letztendlich ein Prototyp entwickelt werden, der eine physikbasierte Interaktion zur simultanen Translation und Rotation von Objekten ermöglicht. Welchen Nutzen die physikbasierte Interaktion dem Anwender tatsächlich bringt, muss über Untersuchungen mit Versuchspersonen ermittelt werden, die ebenfalls Bestandteil der Masterarbeit des Autors sein werden.

Literaturverzeichnis

[Agarawala und Balakrishnan 2006] AGARAWALA, Anand ; BALAKRISHNAN, Ravin: Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM Press, 2006, S. 1283–1292. – ISBN 1-59593-372-7

[Buck 1999] BUCK, Jamis: *The Recursive Ray Tracing Algorithm*. Webseite. Dezember 1999. – URL http://www.geocities.com/jamisbuck/raytracing.html. – Letzter Aufruf am 28. Februar 2007.

[Kruger u. a. 2005] KRUGER, Russell ; CARPENDALE, Sheelagh ; SCOTT, Stacey D. ; TANG, Anthony: Fluid integration of rotation and translation. In: *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 2005, S. 601–610. – ISBN 1-58113-998-5

[Roßberger 2007a] ROSSBERGER, Philipp: *Flüssige Interaktionstechniken für kollaboratives Arbeiten*. Seminararbeit. Februar 2007. – unveröffentlichte Studienarbeit

[Roßberger 2007b] ROSSBERGER, Philipp: *Physikbasierte Interaktion im Collaborative Workspace*. Seminararbeit. Februar 2007. – unveröffentlichte Studienarbeit

[Smith 2006] SMITH, Russell: *Open Dynamics Engine v0.5 User Guide*, February 2006