Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 69

3D Pointing Toolkit

Dennis Root

Studiengang:	Informatik
Prüfer/in:	Prof. Albrecht Schmidt
Betreuer/in:	M.Sc. Thomas Kubitza
Beginn am:	2013-07-01
Beendet am:	2013-07-09
CR-Nummer:	D.2.2, D.2.13, E.1, H.5.2, I.3.8

Kurzfassung

Der Mensch verbringt einen großen Teil seines Lebens in seinem zu Hause und die sogenannten „Smart-Homes“ von heute werden immer intelligenter. Deshalb besteht Bedarf nach neuen Interaktions- und Feedbackmöglichkeiten für elektronische Geräte, die in unsere modernen Haushalte integriert sind. Mit jedem neuen Feature, das dem Anwender geboten wird, muss nämlich erforscht werden, wie der Mensch auf gewisse Feedbacks reagiert und wie gut er mit neuen Interaktionstechniken umgehen kann. Um Daten über einen Anwender, seine Bewegungen und Aktionen sammeln zu können, werden häufig Trackingsysteme eingesetzt. Diese wiederum benötigen Softwarelösungen, die die gesammelten Daten auswerten und in sinnvoller Form an andere Anwendungen weitergeben.

Eine solche Softwarelösung ist das in dieser Bachelorarbeit vorgestellte Toolkit, das präzise Positions- und Orientierungsdaten eines optischen Trackingsystems verarbeiten und an andere Anwendungen weitergeben soll. Mit den verarbeiteten Daten ist es dann möglich eine große Bandbreite von Studien durchzuführen. Das Toolkit kann mit Hilfe der vom Trackingsystem gelieferten Daten Zeigegesten des Anwenders im dreidimensionalen Raum erkennen und damit Schnittberechnungen mit virtuellen Objekten durchführen. Die genannten Objekte wurden zuvor im virtuellen 3D-Raum des Toolkits modelliert. Das Toolkit ist durch seinen integrierten Editor sehr flexibel, da die virtuelle Umgebung an alle realen Gegebenheiten angepasst werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Trackingsysteme im Allgemeinen haben der Forschung schon in der Vergangenheit viele Möglichkeiten geboten die reale Umwelt in eine virtuelle Realität zu integrieren, um damit neue Erkenntnisse über gewisse Gegebenheiten unserer Umwelt zu gewinnen. Heute sind Trackingsysteme weiterhin unersetzbare Hilfsmittel für Studien und besonders optische Trackingsysteme ermöglichen es auf relativ einfache Weise Daten über unsere Umgebung zu sammeln. Mit optischen Trackingsystemen ist es z. B. möglich Bewegungsabläufe von Menschen oder Tieren zu erfassen, um mehr über die biologischen Hintergründe zu erfahren und diese Erkenntnisse auf unsere Technik zu übertragen. Mit der Erfassung von menschlichen Bewegungen wird es z. B. möglich neue Interaktionsmöglichkeiten zu bieten, mit denen man Fernseher oder Displays mit der Größe einer Kinoleinwand durch Gesten steuern kann. Eine weitere Trackingtechnologie, die jeder von uns schon verwendet hat, ist unser heutiges GPS-System, das Unmengen an Möglichkeiten mit sich gebracht hat. Ein Beispiel hierfür ist die Navigation auf der ganzen Erde. Das Tracking allein ist aber insoweit nutzlos, da es zwar interessante Daten liefert, diese Daten aber ohne die Auswertung eines entsprechenden Programms nicht von anderen Anwendungen verwertet werden können. Es werden Tools benötigt, die die erzeugten Daten auswerten und dem Menschen verständlich darlegen. Diese Tools haben nämlich die Aufgabe Realweltdaten auf eine modellierbare virtuelle Umwelt zu projizieren und neue verwertbare Daten zu erzeugen.

Solche Umgebungen, aus denen die Daten gewonnen werden, sind die modernen Haushalte, in denen wir einen Großteil unserer Lebenszeit verbringen. Die Anforderungen an unsere Haushalte werden immer größer und mit aktuellen technischen Neuerungen wird es möglich immer mehr Features in das sogenannte „Smart-Home“zu integrieren. Diese neuen Features verlangen aber immer neuere Konzepte in der Mensch-Computer-Interaktion, welche die Interaktion an sich, sowie auch das Feedback für den Anwender betreffen. Um Studien für die zukünftige Interaktion mit dem „Smart-Home“durchführen zu können, benötigt man Hilfsmittel, mit denen man präzise und schnell Daten sammeln kann und die sich preislich im erschwinglichen Bereich befinden. Ein solches Hilfsmittel sind die beschriebenen optischen Trackingsysteme, die mit einer hohen Genauigkeit die Positions- und Orientierungsdaten von speziellen Markern erfassen können. Um die erfassten Daten für Forschungs- und Studienzwecke verwenden zu können, wird eine virtuelle dreidimensionale Repräsentation des Raums und der darin enthaltenen Marker benötigt. Die Repräsentation ermöglicht es einerseits Objekte zu modellieren, die sich in einem durchschnittlichen Haushalt befinden und andererseits mathematische Berechnungen durchzuführen.

Im Rahmen der Bachelorarbeit soll ein Toolkit entwickelt werden, das es im 3D-Raum ermöglicht Zeigegesten, die mit Hilfe des Trackingsystems ermittelt werden, zu erkennen. Diese Zeigegesten werden im Toolkit dazu verwendet in Echtzeit Schnittberechnungen, mit

den im virtuellen Raum modellierten Gegenständen, durchzuführen. Die Ergebnisse der Berechnungen sollen mit Hilfe einer entsprechenden Netzwerk-API für andere Anwendungen verfügbar gemacht werden. Das Toolkit soll außerdem über einen Editor verfügen, mit dessen Hilfe man die Objekte im virtuellen dreidimensionalen Raum bearbeiten kann.

Gliederung

Der Aufbau dieser Arbeit ist so gegliedert, dass anfangs in einer **Einleitung** einerseits geschildert wird welche Motivation hinter dieser Ausarbeitung steckt und andererseits wird ein Überblick über die Aufgabenstellung und die Ziele der Arbeit gegeben.

Im drauf folgenden **Kapitel: Grundlagen und Verwandte Arbeiten** werden im ersten *Abschnitt 2.1* die technischen Grundlagen, nämlich die Trackingtechnologien, erläutert. Im anschließenden *Abschnitt 2.2* werden die Interaktionsmöglichkeiten erwähnt, die in Verbindung mit optischen Trackingsystemen möglich sind. Im *Abschnitt 2.3* werden dann die nötigen Grundlagen im Bereich der Computergrafik vermittelt, die den Kern des entwickelten Toolkits darstellen.

Im **Kapitel: Konzept und Use Cases** werden dem Leser das Konzept und zwei Use Cases vorgestellt, die anhand von zwei konkreten Beispielen die möglichen Einsatzgebiete für das 3D Pointing Toolkit deutlich machen sollen. Es wird außerdem im *Abschnitt 3.2* die Bedeutung des Toolkits für die Forschung und Benutzerstudien erläutert, das eines der wichtigsten Ziele der entwickelten Software ist.

Das eigentliche System mit den wichtigsten Bestandteilen wird im **Kapitel: 3D Pointing Toolkit** *Abschnitt 4.1* vorgestellt und im *Abschnitt 4.2* wird im Detail auf die Kommunikation des entwickelten Toolkits eingegangen. Der *Abschnitt 4.3* erläutert das Echtzeitrendering mit allen dazugehörenden Grundlagen und der letzte *Abschnitt 4.4* bewertet die Performance des vorgestellten Toolkits.

Das letzte **Kapitel: Zusammenfassung und Ausblick** fast die Ergebnisse der Arbeit zusammen und betrachtet sie aus einem kritischen Blickwinkel. Zusätzlich wird ein Ausblick auf die Zukunft in diesem Bereich gegeben und es werden mögliche Anknüpfungspunkte aufgezeigt.

2. Grundlagen und Verwandte Arbeiten

2.1. Technische Grundlagen

2.1.1. Trackingsysteme

Um Informationen über die Position und Orientierung eines Objekts in einem Raum zu erhalten setzt man Trackingsysteme ein, die auf unterschiedliche Weisen arbeiten. Die von den Trackingsystemen gelieferten Realweltdaten werden dann in einer virtuellen dreidimensionalen Umgebung dargestellt. Mit Hilfe der virtuellen Umgebung kann der User in einem computergenerierten Modell eines 3D-Raums Interaktionen durchführen. Hierbei spricht man vom sogenannten „Motion Caputring“. Im Folgenden werden die grundlegenden Trackingtechnologien beschrieben, die Vor- und Nachteile aufgezeigt und ihre Einsatzgebiete erwähnt. Es wird im Allgemeinen zwischen Trackingsystemen unterschieden, die mit Hilfe von akustischen Signalen, elektromagnetischen Feldern, Radiowellen, Beschleunigungssensoren und mit optischen Sensoren arbeiten. Am Ende soll deutlich werden, wieso in dem hier präsentierten Fall ein optisches Trackingsystem die sinnvollste Alternative ist. Tiefgreifendere Informationen findet man bei [BWA01] und [RBG01].

Definition: Degree of Freedom

Degree of Freedom bezeichnet die Anzahl an Parametern in einem System, die unabhängig voneinander variieren können. Bei einem Trackingsystem sind es die möglichen Bewegungen, die durchgeführt bzw. erfasst werden können. Die ersten drei DOFs sind Translationen in X-, Y-, Z-Richtung und die weiteren drei Parameter sind die möglichen Rotationen um die einzelnen Achsen. Diese Rotationen werden häufig als Pitch, Roll und Pan bezeichnet.

2.1.2. Tracking Technologien mit 3DOF und 6DOF

Akustisches Tracking

Das akustische Tracking benutzt den Schall um eine Position eines Objektes im 3D Raum zu bestimmen. Für das Tracking werden für gewöhnlich Schallwellen im für den Menschen nicht hörbaren Bereich verwendet, damit der User keine unangenehmen Töne bzw. Geräusche wahrnehmen kann. Um akustisches Tracking im dreidimensionalen Raum verwirklichen zu können, benötigt man entweder ein System-Setup mit einem Empfänger und drei Sendern

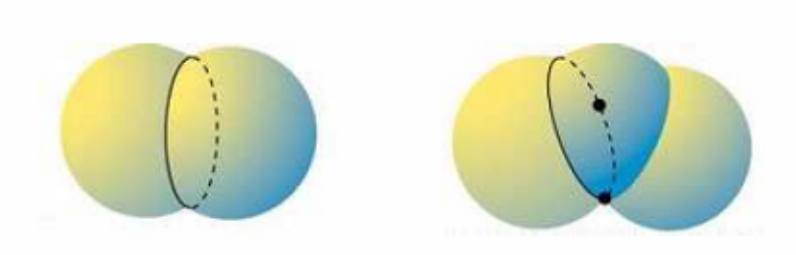

Abbildung 2.1.: (links) ein Schnitt zwischen zwei Kugeln bildet einen Schnittkreis (rechts) ein Schnitt mit drei Kugeln liefert zwei Schnittpunkte. *Quelle: [BWA01]*

oder drei Empfängern und einem Sender. Dies ist der Fall, da mit Hilfe eines Empfängers und Senders nur die Entfernung zu einem Objekt ermittelt werden kann. Dadurch kann sich das Objekt auf der Oberfläche einer Kugel befinden 2.1, was für eine genaue dreidimensionale Positionsbestimmung noch nicht ausreicht. Durch einen zweiten Sender bzw. Empfänger entsteht eine weitere Kugel, die die erste Kugel schneidet und wodurch sich die mögliche Position des Objekts auf den Schnittkreis der beiden Kugeln begrenzt 2.1. Mit Hilfe eines dritten Senders bzw. Empfängers entstehen zwei Schnittpunkte mit allen drei Kugeln, von denen einer durch die vorherigen Positionen verworfen werden kann 2.1.

Um die Entfernung zu einem Objekt zu bestimmen, werden beim akustischen Tracking generell die zwei folgenden Techniken angewendet:

1. TOF (Time of Flight) In diesem Fall wird die Zeit gemessen, die der Schall vom Sender zum Empfänger benötigt. Mit Hilfe der gemessenen Zeit und der Geschwindigkeit des Schalls kann die Entfernung berechnet werden. Es gilt nämlich:

$$d[m] = v[\frac{m}{s}] * t[s] \quad (2.1)$$

2. Phasen Kohärenz Bei dieser Methode wird die Distanz zum Objekt mit Hilfe der Phasendifferenz zwischen der Phase des Schalls am Sender und der am Empfänger ermittelt.

Um beim akustischen Tracking 6DOF zu erreichen werden insgesamt drei Sender und drei Empfänger benötigt. Aus den gewonnen Informationen kann zusätzlich zur Position des Objekts auch die Orientierung berechnet werden. Dies kann mit genau dieser Anzahl an Sendern und Empfängern bewerkstelligt werden, da aus den ermittelten Positionen der einzelnen Sender die Orientierung des Objekts, der die drei Sender an sich trägt, ermittelt werden kann. Denn die festgelegte räumliche Stellung der drei Sender lässt einen Rückschluss auf die Orientierung zu.

Vorteile Diese Trackingtechnologie ist im Vergleich zu den anderen Techniken sehr kostengünstig und es werden keine magnetischen Felder erzeugt.

Nachteile Es wird eine direkte Sichtlinie von Empfänger zu Sender benötigt, was die Bewegungsfreiheit des Users einschränkt. Ein weiterer Nachteil ist die Eigenschaft des

Systems, dass es temperaturabhängig ist. Schallwellen breiten sich bei verschiedenen Temperaturen unterschiedlich schnell aus und diese Unterschiede müssen beachtet und für genaue Messergebnisse durch das System ausgeglichen werden. Das Medium Schall hat als weiteren Nachteil die Eigenschaft, dass die Reichweite ziemlich begrenzt ist. Akustische Trackingsysteme haben im Allgemeinen auch eine geringe Genauigkeit und Abtastrate.

Einsatzgebiete Akustisches Tracking wird durch seine Nachteile eher in Bereichen eingesetzt, die nicht auf extrem genaue Messergebnisse angewiesen sind. Ein Beispiel hierfür ist die Unterhaltungsindustrie.

Inertiales Tracking

Inertiales Tracking kann in der 3DOF Variante die X-, Y-, Z-Position eines Objektes mittels Beschleunigungssensoren bestimmen. Für jede Richtung kann jeweils die Beschleunigung mit Hilfe der Trägheit einer Masse ermittelt werden und durch den Zusammenhang von Beschleunigung über eine bestimmte Zeit kann die neue Position bestimmt werden. Dieser Zusammenhang wird durch diese Formel dargestellt:

$$pos = a \int \int dt^2 \tag{2.2}$$

Um ein System mit 6DOF zu erreichen, werden beim inertialen Tracking Gyroskope verwendet. Damit lassen sich im System Drehimpulse um die jeweiligen Achsen messen, mit denen es dann möglich ist die Drehung um einen bestimmten Winkel um eine Achse zu bestimmen. Gyroskope arbeiten mit einer sich konstant rotierenden Masse, die durch die Drehimpulserhaltung Lageänderungen entgegenwirkt. Ein Beispiel hierfür ist ein Kreisel, dessen Rotationsachse der Gravitationsachse entspricht. Wenn der Kreisel sich schnell genug dreht, kippt er nicht um, sondern wirkt der Gravitationskraft und damit dem Umkippen entgegen. Mechanisch arbeitende Gyroskope sind zwar sehr präzise, werden aber wegen der Größe und Masse selten zum Tracking von Menschen eingesetzt. Die Alternative zu mechanischen Gyroskopen bilden micromechanische Sensoren, die viel kleiner sind und mit sogenannten Vibrationskreiseln arbeiten.

Vorteile Ein großer Vorteil dieser Trackingtechnologie ist die Tatsache, dass kein bestimmtes „Medium“ benötigt wird, über das irgendwelche Signale übermittelt werden müssen. Daher ist keine zusätzliche Hardware wie Sender oder Empfänger nötig. Durch die kleinen Ausmaße des ganzen Systems, wird der Benutzer auch in kaum einer Weise in seiner Bewegungsfreiheit eingeschränkt und kann sich komplett frei im Raum bewegen. Außerdem gibt es keine äußeren Störfaktoren, die das System beeinflussen könnten.

Nachteile Da das inertiale Tracking eine Startkonfiguration hat, von der aus mit Hilfe der Beschleunigungsmessungen die folgenden Positionen und Orientierungen berechnet werden, muss das System von Zeit zu Zeit rekalibriert werden. Damit wird vermieden, dass sich eventuelle Messungenauigkeiten zu einem großen Fehler aufaddieren.

Einsatzgebiete Inertiales Tracking benötigte früher große und schwere Messinstrumente, wohingegen heute kleine elektronische Bauteile diese Aufgabe übernommen haben. Mit diesen kleinen Elementen findet inertiales Tracking häufig Einsatz in hybriden Trackingtechnologien, da sie alleine zu starke Messungenauigkeiten über die Zeit aufweisen.

Magnetisches Tracking

Magnetisches Tracking kann unter Verwendung von Magnetfeldern die Postion und Orientierung eines Objekts im Raum ermitteln. Dies geschieht entweder durch Magnetfelder, die durch Wechselstrom mit niedriger Frequenz erzeugt werden, oder durch pulsierende gleichstromgenerierte Magnetfelder. Um ein 6DOF System zu erhalten, werden auf der Senderseite drei separate Spulen benötigt, die orthogonal um einen Kern gewickelt sind und auf der Empfängerseite drei „Antennen“, welche auch Spulen sind. Eine Messung der Position und Orientierung im Raum erfolgt durch drei separate Sende- und Messvorgänge, bei denen eine Spule nach der anderen ein Signal aussendet bzw. empfängt. Wenn eine Sendespule eine Magnetfeld aufbaut, wird bei der Empfängerspule eine messbare Spannung induziert, die von der Entfernung zum Sender und der Ausrichtung zum Magnetfeld des Senders abhängig ist. Die induzierten Spannungen können dann mit bekannten Werten verglichen werden, um die Entfernung zum Objekt zu ermitteln. Um zusätzlich noch die Orientierung im Raum berechnen zu können, werden die drei induzierten Spannungen der jeweiligen Empfängerspulen untereinander verglichen. Wichtig ist noch zu erwähnen, dass Magnetfelder, die mit Wechselstrom generiert werden, den Nachteil haben, dass ferromagnetische Objekte, die in der Reichweite des Empfängers liegen, das erzeugte Magnetfeld beeinflussen können. Dadurch, dass Wechselstrom seine Richtung andauernd ändert, entstehen ständig Wirbelströme, die schwer zu korrigieren sind. Magnetfelder, die durch Gleichstrom generiert werden, erzeugen hingegen keine Wirbelströme. Messungenauigkeiten bleiben jedoch auch in der Gleichstromvariante erhalten.

Vorteile Magnetische Trackingsysteme benötigen, durch ihre Arbeitsweise bedingt, keine direkte Sichtlinie zwischen Empfänger und Sender, wodurch der Anwender frei in seinen Bewegungen ist. Die eingesetzten Sensoren sind meist leicht und klein, was auch wiederum eine bessere Handhabung ermöglicht. Mit magnetischen Systemen ist es möglich mehrere Personen gleichzeitig zu erfassen.

Nachteile Ein großes Problem, mit dem magnetische Trackingsysteme konfrontiert werden, sind durch metallische Objekte verursachte Verzerrungen und Beeinflussungen der Messergebnisse. Außerdem ist diese Technologie nur in kleinen Bereichen des erzeugten Magnetfelds präzise und es können zusätzlich Probleme beim Erfassen von schnellen Bewegungen entstehen. Dadurch, dass alle benötigten Bauteile eine eigene Stromversorgung brauchen, müssen viele Kabel verwendet werden.

Einsatzgebiete Magnetisches Tracking wird auf Grund seiner kleinen Sensoren in vielen Bereichen eingesetzt. Hier muss jedoch darauf geachtet werden, dass keine ferro-

magnetischen Stoffe die Messungen stören. Häufig setzen Forschungs- aber auch Militäreinrichtungen magnetisches Tracking ein.

Mechanisches Tracking

Diese Art von Tracking ist für den Einsatz in „Smart-Homes“nicht geeignet, da mechanisches Tracking typischerweise Winkel von Gelenken und die Abstände zwischen einzelnen Gelenken misst. Es ist zwar möglich mit einer Anfangsposition alle weiteren Positionen der gemessenen Gelenke durch die gemessenen Winkeländerungen zu bestimmen. Um aber solche Messungen durchführen zu können, muss der User z. B. ein Exoskelett anlegen. Dadurch ist der Einsatz aufwändig und mit zunehmender Präzision steigt auch die Komplexität des „Messinstruments“.

Vorteile Ein sehr geschätzter Vorteil des mechanischen Trackings ist die extrem hohe Genauigkeit, die erzielt werden kann. Deswegen finden diese Systeme häufig Einsatz in der Medizin, um präzise Informationen über Position und Orientierung, von z. B. Operationswerkszeugen zu erhalten. Diese Systeme benötigen außerdem keine direkte Sichtlinie und sie arbeiten auch nicht mit magnetischen Feldern, Schallwellen oder Licht, was in manchen Fällen störend sein kann.

Nachteile Mechanisches Tracking hat zwar einzelne enorme Vorteile, diese Art von Tracking birgt aber auch extreme Nachteile. Einer davon ist die Eigenschaft, dass diese Systeme häufig als Exoskelett am User angebracht werden und dadurch die Bewegungsfreiheit sehr stark beeinträchtigt wird. Die Exoskelette können nämlich schwer oder schwer beweglich sein. Ein weiterer Nachteil ist der, dass diese Systeme nicht auf große Räume angewendet werden können, da es technisch kaum möglich ist.

Einsatzgebiete Die bekanntesten Einsatzgebiete des mechanischen Tracking sind einerseits sogenannte „Boom-Type Displays“und andererseits Operationswerkzeuge. Ein Boom-Type Display ist ein kleines Display, das am Kopf und direkt vor den Augen eines Menschen angebracht ist und das an Stäben befestigt ist. Diese Stäbe bilden einen Arm, der das Display hält und der die Position und Orientierung des menschlichen Kopfes erfasst. Die erwähnten Operationswerkzeuge mit mechanischem Tracking können die menschlichen Bewegungen erfassen und auf einen Roboterarm übertragen, der bei einer Operation viel ruhigere und genauere Bewegungen ausführen kann.

Optisches Tracking

Allgemein In diesem Abschnitt wird das optische Tracking etwas genauer unter die Lupe genommen, da es eines der wichtigsten und am häufigsten eingesetzten Trackingtechnologien ist. Der interessierte Leser kann in dem Survey zum Thema Trackingtechnologien von Welch et al. [WF02] mehr erfahren. Optische Trackingsysteme werden in folgende zwei grundlegende Systemkonfigurationen eingeteilt:

Outside-In Es gibt optische Sensoren, die in der Umgebung an definierten Stellen angebracht sind. Diese erfassen die Position und Orientierung von sogenannten Targets, die am zu erfassenden Ziel angebracht sind.

Inside-Out Beim inside-out Ansatz sind die optischen Sensoren am Ziel angebracht und diese ermitteln die Position und Orientierung im Raum mit Hilfe von definierten Punkten, die in der Umgebung platziert sind.

Das Toolkit, das in dieser Arbeit präsentiert wird, basiert auf dem outside-in Ansatz. Optisches Tracking, auch bildbasiertes Tracking genannt, arbeitet generell mit einem sogenannten Target, welches als Lichtquelle dient und Sensoren, die lichtempfindlich sind **Abbildung: 2.3**. Durch das Analysieren, der von den Sensoren gewonnen Bilder, wird die Position und Orientierung der Targets im Raum ermittelt. Hierbei wird zwischen passiven und aktiven Lichtquellen unterschieden. Aktive Lichtquellen strahlen mit Hilfe von z. B. LEDs Licht in unterschiedlichen Lichtspektren aus und benötigen daher eine Stromquelle. Sie haben zwar den Vorteil, dass sie ein optisches Trackingsystem robuster gegen Licht aus der Umgebung machen, aber sie sind wegen der benötigen Stromquelle wartungs- und kostenintensiver. Da die meisten optischen Trackingsysteme für den Inneneinsatz gedacht sind, werden häufig passive Targets verwendet, die meist eine sphärische Form besitzen und deren Oberfläche reflektierend ist **Abbildung: 2.2**, oder einen starken Kontrast zur Umgebung bilden. Ein solches System beschreiben Pintaric et al. in dem Paper [PK07]. Es basiert auf dem outside-in Prinzip, arbeitet mit passiven Targets und zeigt sehr deutlich die Vor- und Nachteile eines solchen Trackingsystems auf.

Zusätzlich zu den beschriebenen Targets werden optische Sensoren für das Tracking benötigt. Jeder optische Sensor arbeitet auf eine leicht unterschiedliche Weise. Es werden die folgenden Typen von Sensoren unterschieden:

- Charge Coupled Devices (CCDs)

  CCDs bestehen aus 1D oder 2D Photozellen-Arrays, bei denen die einzelnen Photozellen je nach Lichteinstrahlung eine gewisse Spannung produzieren. Dadurch entsteht ein digitales Bild der Umgebung mit den relativen Lichtintensitäten, die auf die einzelnen Zellen einwirken. Durch die vorhin beschriebenen Gegebenheiten der Targets und der Umgebung, sind es die Targets, die den hellsten Punkt in dem Array erzeugen. Dadurch bekommt man in dem Photozellen-Array eine 2DOF Position des Targets. Eine weitere Eigenschaft der CCDs ist, dass sie nur Einzelaufnahmen machen und bedingt durch die Technik und Verarbeitungsdauer der eingehenden Signale eine begrenzte Anzahl an Bildern pro Sekunde liefern können.

- Photosensoren

  Photosensoren verändern ihren Widerstand je nach dem wie viel Licht einfällt. Mehrere zusammengesetzte Photosensoren liefern damit ein analoges Signal, das genau wie CCDs ein 2DOF Abbild der Umgebung und dem Target darstellt. Sie haben den Vorteil, dass sie simpel und schnell sind, da sie ein kontinuierliches Signal liefern.

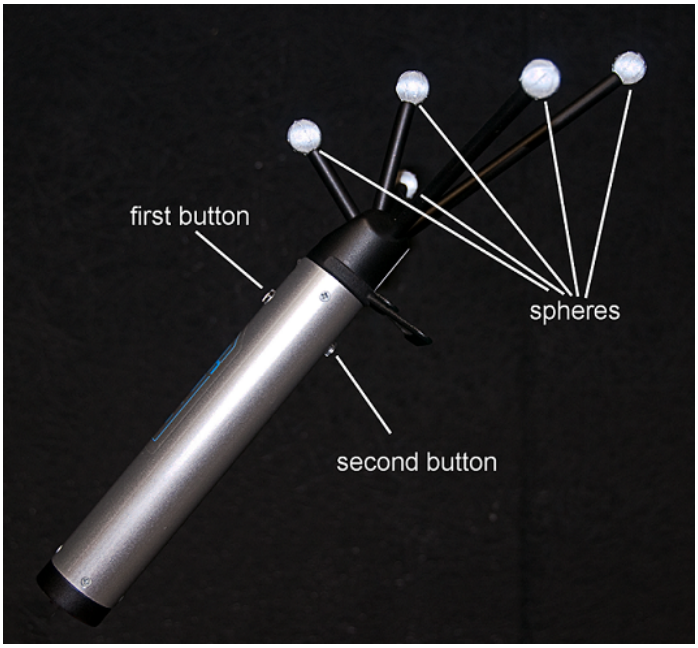

Abbildung 2.2.: Ein Beispiel für einen sogenannten „Wand“mit passiven Targets, wie er bei optischem Tracking eingesetzt wird.

Abbildung 2.3.: Optischer Sensor mit im Kreis angeordneten Infrarotlicht emmitierenden Einheiten. *Quelle: [SEN]*

- Position Sensing Detectors (PSDs)

  PSDs arbeiten auf eine ähnliche Weise wie Photosensoren. Sie bestehen aus einem 1D oder 2D Halbleiterbauelement, das eine Menge an Strömen liefert, die der Menge an einstrahlendem Licht von einem bestimmtem Photosensor entsprechen. Daraus wird, wie bei den anderen Sensoren, eine 2DOF Position des Targets ermittelt.

Um nun aus den ermittelten 2DOF Daten die Position im Raum, die 3DOF entspricht, zu berechnen, werden mehrere Sensoren an unterschiedlichen Postionen angebracht. Mit dem Wissen über die jeweiligen Postionen der einzelnen optischen Sensoren, lässt sich mathematisch die 3D Position des Targets ermitteln. Um mit einem optischen Trackingsystem 6DOF zu erreichen benötigt man lediglich mehrere Targets. Denn aus der räumlichen Stellung der jeweiligen Targets zueinander, lässt sich die Orientierung des Objekts tracken. Siehe hierfür **Abbildung 2.4**.

Vorteile Optische Trackingsysteme haben den ganz großen Vorteil, dass sie durch ihre Schnelligkeit sehr gut für Echtzeitanwendungen geeignet sind. Dadurch, dass Licht als Medium verwendet wird und die elektronischen Bauteile die Signale meist sehr schnell verarbeiten können, sind Update Raten von z. B. 100Hz möglich, was 100 Bildern in der Sekunde entspricht. Außerdem sind die heutigen optischen Trackingsysteme sehr präzise. Pintaric et al. [PK07] stellen ein System vor, dass im Bereich von $\pm 5mm$ arbeitet und Winkel im Bereich von 0.05° erfassen kann. Optische Trackingsysteme sind überdies unempfindlich gegenüber magnetischen und akustischen Störfaktoren.

Nachteile Einer der größten Nachteile von optischen Trackingsystemen ist die Eigenschaft, dass zwischen Sensor und Target eine direkte Sichtlinie bestehen muss. Dies kann den User unter Umständen in seiner Bewegungsfreiheit einschränken und ungenaue Daten zur Folge haben. Da in einem Testraum aber je nach System mehrere Dutzend Sensoren an unterschiedlichen Positionen platziert werden können, ist dieses Problem meist unter Kontrolle zu bekommen. Optische Trackingsysteme können zusätzlich durch äußere Lichteinwirkung in ihrer Arbeitsweise behindert werden. Dies lässt sich aber auch wieder durch einen Testraum, in dem die Umwelteinflüsse kontrolliert werden können, unterbinden.

Einsatzgebiete Optische Trackingsysteme werden in vielen Bereichen eingesetzt, da sie kostengünstig sind, ihren Anwender kaum in seiner Bewegungsfreiheit einschränken und zusätzlich sehr genau und schnell arbeiten. Ein Einsatzgebiet, mit dem sich jeder von uns fast täglich beschäftigt, ist die Filmindustrie. Moderne Filme beinhalten fast immer Szenen, die fast vollständig im Computer entstanden sind, da sie so aufwändig sind und viel zu hohe Kosten verursachen würden. Um die Bewegungen der Schauspieler in die virtuelle Szene integrieren zu können, bedienen sich die Filmemacher des sogenannten „Motion Caputring“. Hierbei werden am Menschen reflektierende, sphärische Kügelchen angebracht, die von Kameras erfasst werden. Mit diesen Daten über die Positionen und Bewegungen der Schauspieler, lassen sich animierte Figuren in der virtuellen Szene erstellen, die nach jedem beliebigen Wunsch gestaltet werden können. Beispiele hierfür sind sehr viele aktuell in Hollywood produzierten Filme, z. B. Iron Man, Avatar usw.

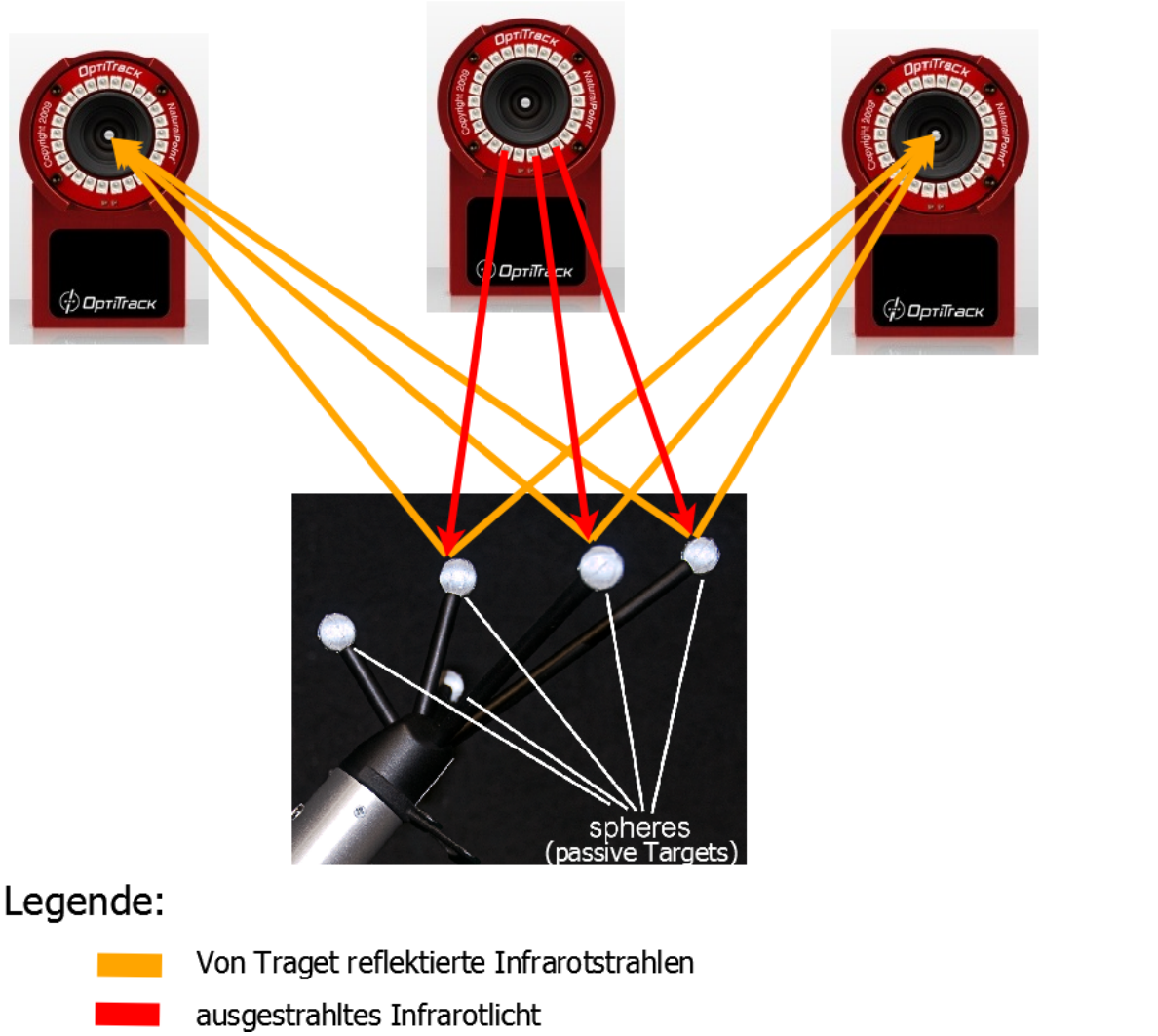

Abbildung 2.4.: Zwei optische Sensoren erfassen die von den drei passiven Targets reflektierten Infrarotstrahlen. Es werden mindestens zwei Sensoren und drei Targets benötigt um ein 6DOF System zu erhalten, da ein Sensor nur die 2DOF Position des Targets erfassen kann. Es wird erst mit zwei Sensoren und mit Hilfe der bekannten Positionen der Sensoren möglich, die 3DOF Position eines Targets im Raum zu berechnen. Um nun noch die Orientierung des Objekts im Raum zu berechnen werden drei Targets benötigt, denn mit Hilfe ihrer Positionen im Raum zueinander, wird es erst möglich die Orientierung zu erhalten.

Optisches Tracking kann aber auch dafür eingesetzt werden, intuitive Interaktionsmöglichkeiten für Menschen zu bieten. Ein Beispiel hierfür ist das Paper von Shizuki et al. [SHTT06], in dem ein Laserpointer verwendet wird, um Interaktionen mit einer großen Leinwand zu ermöglichen. Hierbei wird der rote Punkt, den ein Laserpointer wirft, von einer Kamera erfasst und mit Hilfe bestimmter Zeigegesten, die die Außenflächen der Leinwand verwenden, kann der Anwender einfach durch Präsentationsfolien navigieren oder ein Bild rotieren. Die Idee dahinter ist, dass beim Überqueren einer Randfläche das System anfängt auf einen Befehl, z. B. in Form einer Geste, zu warten. Da die Randflächen einer Leinwand so groß sind, ist diese Interaktion gemäß Fitts' Law für den Anwender sehr angenehm und leicht umzusetzen.

2.2. Optische Trackingsysteme - Interaktionsmöglichkeiten

Optisches Tracking in Verbindung mit einer großen virtuellen Umgebung bietet dem Anwender viele Möglichkeiten mit der realen Umwelt zu interagieren. Eine Möglichkeit sind sogenannte Laserpointer Interaktionstechniken. Hierbei verwendet man meist einen echten Laserpointer, der einen Strahl auf eine Fläche wirft und dann von Kameras erfasst wird. Es werden wahlweise auch Devices verwendet, die nur als virtuelle Laserpointer dienen. In beiden Fällen wir von der Spitze des Devices aus ein Strahl ausgesendet, der mit der Interaktionsfläche geschnitten wird und z. B. als Cursor dienen kann, um Objekte auszuwählen oder Gesten auszuführen. Ein gutes Beispiel hierfür ist das im vorherigen Kapitel erwähnte Paper [SHTT06], bei dem ein echter Laserpointer verwendet wird, um durch Gesten mit einer großen Leinwand zu interagieren. Hierbei erfasst das System eine Überquerung einer definierten Grenzlinie, die die Innenfläche und die Außenfläche der Interaktionsfläche trennt. Wenn eine solche Überquerung z. B. der rechten Grenzlinie festgestellt wird, kann in einer Präsentation die nächste Folie angezeigt werden. Ein weiteres Beispiel ist die von Kirstein et al. [KM98] vorgestellte Arbeit. Es wird vorgestellt wie ein Punkt eines Laserpointers durch Kameras erfasst werden kann und anschließend direkt als Cursor verwendet werden kann, um Eingaben vorzunehmen. Dabei wird die Position des erfassten Punkts des Laserpointers an den Eingangkanal der Maus am Computer gesendet und als direktes Eingabesignal verwendet. Der größte Vorteil dieser Laserpointer Interaktionstechniken ist der, dass sehr schnelle und intuitive Interaktionen ermöglicht werden. Die Genauigkeit dieser Methode wird jedoch durch das Zittern der menschlichen Hand in Mitleidenschaft gezogen. Cao et al. [CB03] stellen in ihrer Arbeit weitere Interaktionstechniken vor, die einen Stab benutzen, um mit einem großen Display zu interagieren. In ihrer Arbeit erfassen Kameras einen speziellen Stab und dieser wird dazu verwendet mit Gesten Aktionen auszuführen. Weitere Interaktionsmöglichkeiten, die einen Laser Pointer verwenden, werden von Olsen et al. [ON01] vorgestellt und bewertet. Bei allen präsentierten Interaktionstechniken, die einen Laser Pointer in Verbindung mit einer großen Interaktionsfläche verwenden, stellt sich die grundlegende Frage, wie gut ein Laser Pointer für solche Zwecke überhaupt geeignet ist. Dieser Frage gehen Oh et al. [OS02] in ihrer Arbeit nach, indem sie den Laser Pointer mit einer Maus in Bezug auf Fitts' Law vergleichen. Eine weitere Interaktionstechnologie, die sich bei einem optischen Trackingsystem anbietet, ist das Tracking des gesamten Körpers

des Anwenders oder nur einzelner Körperteile. Krüger [Kru91] stellte eine Möglichkeit vor, bei der spielerische Dinge, die auf einer großen Leinwand dargestellt wurden, mit dem Körper bedient wurden. Die Fortschritte in der Technik ermöglichen es mittlerweile einzelne Körperteile und sogar einzelne Finger und Fingerbewegungen zu erfassen. Ein aktuelles Beispiel hierfür ist das von LeapMotion [LEA] vorgestellte kleine Device, das im dreidimensionalen Raum die Hände und sogar Finger eines Menschen tracken kann, um damit kleine Spiele oder hoch komplexe CAD Anwendungen bedienen zu können. Hierbei werden Genauigkeiten von 0,01mm erreicht, mit denen es möglich wird sehr präzise Tasks zu meistern.

Eine weitere Interaktionstechnologie, die durch optisches Tracking ermöglicht wird, ist das Hand-Raycasting. Dabei sind an der Hand des Users Tragets angebracht, die vom Trackingsystem erfasst werden und mit deren Hilfe Informationen über die Position und Orientierung der Hand ermittelt werden können.

2.2.1. Hand-Raycasting

Wie bei Interaktionen mit Laserpointern, wird auch beim Hand-Raycasting aus den gewonnen Positions- und Orientierungsdaten der Hand ein imaginärer Strahl ermittelt. Dieser Strahl entspricht der Zeigerichtung der Hand und wird mit der Interaktionsfläche geschnitten. Hierbei kann entweder nur die Hand, aber auch die einzelnen Finger des Anwenders erfasst werden, um Interaktionen zu ermöglichen. Die Hand kann dazu verwendet werden, Zeigegesten in Richtung ausgewählter Objekte oder auch ganze Gesten zu erkennen. Die Hand allein bietet aber wenig Möglichkeiten eine Auswahl zu bestätigen oder zusätzliche Aktionen umzusetzen. Wenn zusätzlich noch die Finger erfasst werden, kann zum Beispiel das Zusammenführen von Daumen und Zeigefinger als Geste erkannt werden, um ein angezeigtes Objekt zu bestätigen oder eine weitere Funktionalität des angezeigten Objekts zu aktivieren. Vogel et al. [VB05] stellen eine Interaktionsmethode vor, die mit unter das Hand-Raycasting verwendet. Es werden zwei Auswahlmöglichkeiten vorgestellt, von denen eine die Berührung von Zeigefinger und Daumen verwendet, um Eingaben vornehmen zu können. Die andere erkennt es, wenn ein imaginärer Button in der „Luft geklickt“wird. Dabei wird erkannt, wie sich der Zeigefinger im Vergleich zu der ganzen Hand und der Umgebung nach unten bewegt und diese Aktion signalisiert einen Klick. Zusätzlich zu den „Klick-Techniken“ werden zwei Zeigetechniken vorgestellt. Eine davon ist das direkte Hand-Raycasting, mit dem man sich durch Zeigegesten sehr schnell durch eine virtuelle Umgebung bewegen kann. Diese Technik ermöglicht es dem Anwender zwar sehr schnelle Eingaben zu tätigen, diese sind jedoch relativ ungenau und können nicht dafür genutzt werden um zielsicher durch ein User Interface, wie wir es von den heutigen Computersystemen kennen, zu navigieren. Für die Eingaben, die eine höhere Genauigkeit verlangen, wird eine weitere Zeigegeste präsentiert, die ein relatives Mapping der Hand des Anwenders verwendet um einen Mauszeiger zu bewegen. Hierbei wird kein Eins-zu-Eins-Mapping benutzt, da ein Anwender bei langsamen Bewegungen meist eine genauere Eingabe vornehmen will und bei schnelleren sich auch schneller durch das Interface bewegen will. Als letzte Interaktionsmethode wird ein Hybrid aus den beiden Zeigetechniken präsentiert. Hierbei

kann der Anwender mit einer offenen Hand das langsame Bewegen der Maus verwenden und wenn die Hand zu einer Zeigegeste geformt wird, kann sich der Anwender schnell durch das Interface bewegen um schnell und relativ genau auf eine spezielle Stelle zeigen zu können. Diese hybride Technik hat den Vorteil, dass das direkte Hand-Raycasting eine sehr natürliche, intuitive und schnelle Form des Menschen ist, das Interesse auf ein Objekt oder eine spezielle Stelle zu richten. Mit dem relativen Mapping der Hand auf einen Mauszeiger kann der Anwender genauere Eingaben vornehmen.

2.3. Computergrafik

3D Anwendungen sind in ihrer Berechnung meist sehr aufwendig und benötigen leistungsstarke Computerhardware. Um bei Echtzeitanwendungen ein flüssiges Bild zu erhalten, benötigt man eine minimale Anzahl an Bildern in der Sekunde. Das sind meist 25 FPS. Hierfür verfügt heutzutage fast jeder Computer über eine eigene Einheit, die nur für Grafikberechnungen zuständig ist. Da es damit trotzdem nicht möglich ist bis ins kleinste Detail realgetreue Bilder und Modelle zu erzeugen, bedient sich die Computergrafik einiger Tricks um Bilder zu generieren, die nach Außen hin nur so wirken als ob sie realistisch wären. Mit diesen Modellen ist es dann möglich physikalische Gesetze zu simulieren, um unsere Umwelt visualisieren zu können. Die Visualisierung wird heutzutage in vielen Bereichen eingesetzt, da es durch die heutige Computergrafik ermöglicht wird komplexe Sachverhalte für den Menschen in Echtzeit sinnvoll darzustellen, um daraus neue Erkenntnisse gewinnen zu können. In diesem Abschnitt werden die grundlegenden Konzepte, die derzeit in der Computergrafik verwendet werden, aufgegriffen und erklärt.

2.3.1. Repräsentation von Objekten

Da eine detailgetreue Repräsentation mit der heute zur Verfügung stehenden Hardware nicht möglich ist, werden Objekte aus Polygonen zusammengesetzt. Dies ist zwar nur eine Approximation, sie bietet aber in Verbindung mit kleinen Tricks ein sehr gutes Ergebnis.

Der Grundbaustein, auf dem die ganze Computergrafik basiert, sind die sogenannten **Vertices**. Dies sind festgelegte Punkte im 3D Raum, mit deren Hilfe es dann möglich ist sogenannte **Edges** zu bilden, die jeweils zwei Vertices als Eckpunkte besitzen. Diese Kanten sind die Berandung von Flächen, die **Faces** heißen. Faces können ein Dreieck mit drei Eckpunkten oder ein **Quad** mit vier Eckpunkten sein. Die Grafikhardware ist so konzipiert, dass sie Faces extrem schnell und sogar parallelisiert verarbeiten kann. Dies ist notwendig, da schon simple Objekte, wie z. B. ein Zylinder aus mehreren Dutzend Faces besteht. Ein **Polygon** wiederum besteht aus mehreren Faces. Für komplexe Objekte, die in der Computergrafik modelliert werden, verwendet man in vielen Bereichen sogenannte **Meshes**, die eine effiziente Speicherung von Polygonen ermöglichen. Ein Mesh wird durch Tabellen von z. B. Vertices beschrieben, die die Zusammensetzung eines Polygons beschreiben. Es wird zwischen folgenden drei Mesh-Datenstrukturen unterschieden [Smio6]:

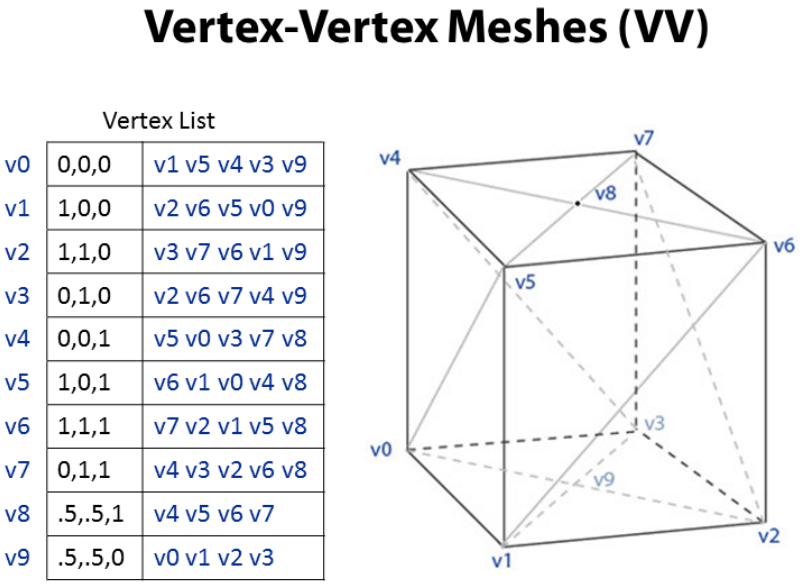

Abbildung 2.5.: Vertex-Vertex-Mesh: für jeden Vertex werden die X-,Y-,Z-Koordinaten und alle Nachbarn in geordneter Reihenfolge gespeichert. *Quelle: [MES13]*

- **Vertex-Vertex Meshes** speichern für jeden Vertex alle seine Nachbarn in geordneter Reihenfolge und die jeweiligen X-,Y-,Z-Koordinaten. **Abbildung 2.5** zeigt eine solche Datenstruktur.
- **Face-Vertex Meshes** führen eine Liste aller existierenden Faces und Vertices. Jeder Vertexeintrag enthält alle angrenzenden Faces. Siehe **Abbildung 2.6**.
- **Winged-Edge Meshes** ist eine weit verbreitete Datenstruktur, die explizit alle Faces, Vertices und Edges speichert. Diese Methode bietet eine hohe Flexibilität, benötigt aber auch am meisten Speicherplatz. Die Vertex-Liste speichert alle anliegenden Egdes, die Face-Liste speichert alle umrandenden Edges und die Edge-Liste speichert die Eckpunkte, anliegenden Faces und die vier am nächsten an den Eckpunkten anliegenden Edges. Diese Datenstruktur ist die komplexeste und ist in **Abbildung 2.7** zu sehen.

Mit Hilfe solcher Meshes können extrem detaillierte Objekte in diversen Programmen, wie z. B. 3dsMax oder der Freeware Google ScetchUp erstellt werden und in verschiedensten Anwendungen verwendet werden. Dazu gehören 3D Graphics Engines, Modellierprogramme usw. Diese Objekte haben dann eine definierte Position und Orientierung im Raum, wobei die Orientierung in der Computergrafik häufig durch sogenannte Quanternions ausgedrückt wird.

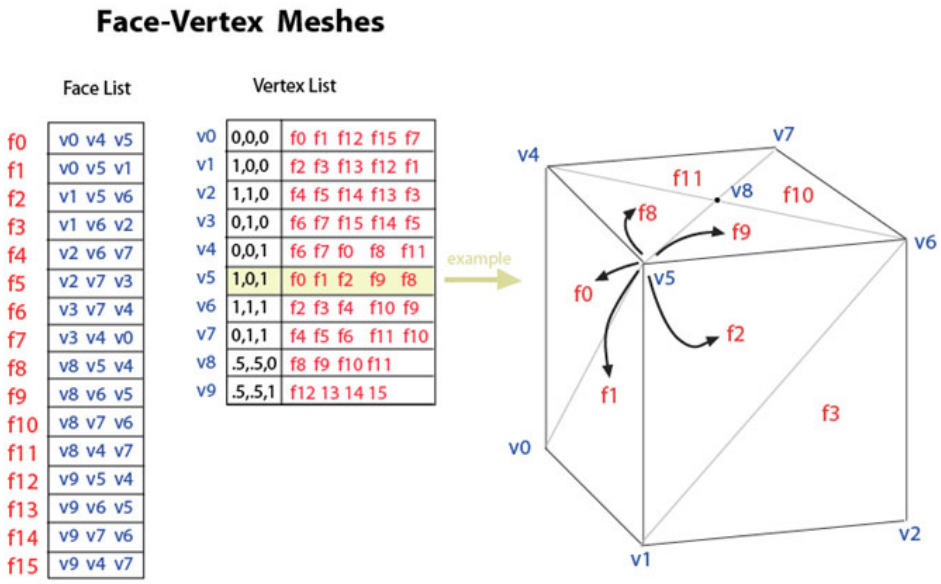

Abbildung 2.6.: Face-Vertex-Mesh: Liste mit allen Faces und Vertices. Jeder Vertexeintrag enthält alle angrenzenden Faces. *Quelle: [MES13]*

Definiton: Quanternions

In der Computergrafik haben sich zur Darstellung von Rotationen und Orientierungen von Objekten im dreidimensionalen Raum Quanternions durchgesetzt. Ein Quaternion ist ein Viertupel mit den drei Komponenten x, y und z, die beschreiben um welche der X-, Y- und Z-Achse die Rotation auftritt und eine w-Komponente, die aussagt um welchen Wert rotiert werden soll. Die mathematische Definition eines Quaternions ist $Q = w + ix + jy + kz$, wobei i, j und k eine imaginäre Komponente darstellen und w, x, y und z reelle Werte sind. Quaternions haben sich wegen folgenden Vorteilen durchgesetzt:

- Der sogenannte „Gimbal Lock“wird vermieden. Grob umschrieben ist ein „Gimbal Lock“das Wegfallen eines Freiheitsgrades der Rotation, wenn bei Euler Winkeln bei der Hintereinanderausführung der Rotationen die erste und dritte Rotationsachse zusammenfallen. Dies hat zur Folge, dass jede Rotation von der vorhergehenden beeinflusst wird.
- Eine bestimmte Rotation zu verändern ist einfach.
- Es sind wenige mathematische Rechenoperationen notwendig um ein Quaternion zu normalisieren. Dies wird beim Korrigieren von Rechenungenauigkeiten, die durch die Computerhardware verursacht werden, benötigt.

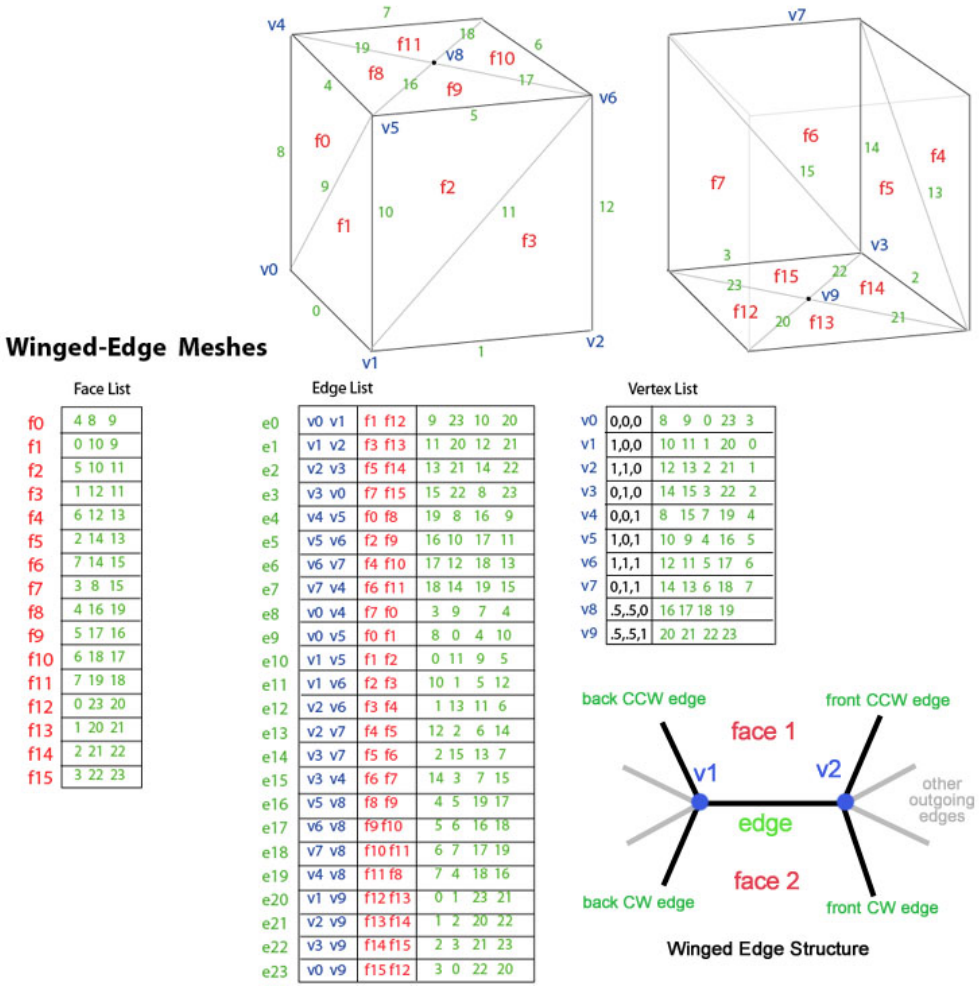

Abbildung 2.7.: Winged-Edge-Mesh: explizite Speicherung aller Faces, Edges und Vertices. Vertex-Liste speichert alle anliegenden Egdes, die Face-Liste speichert alle umrandenden Edges und die Edge-Liste speichert die Eckpunkte, anliegenden Faces und die vier am nächsten an den Eckpunkten anliegenden Edges. *Quelle: [MES13]*

2.3.2. Schnittberechnungen

Der Vorteil von 3D Anwendungen ist der, dass alle Objekte in einer virtuellen Szene existieren und sie in dieser leicht mit Translationen, Rotationen, Skalierungen oder Scherungen manipuliert werden können. Für das entwickelte Toolkit waren im Speziellen die Schnittberechnungen von unterschiedlichen virtuellen Objekten, oder von Objekten und Strahlen sehr interessant. Damit ist es nämlich möglich Zeigegesten des Users in die virtuelle Umgebung zu übertragen und mit einer Szene zu schneiden. Schnittberechnungen sind im Grunde simple mathematische Rechnungen, die vom Toolkit oder einer Graphics Engine durchgeführt werden.

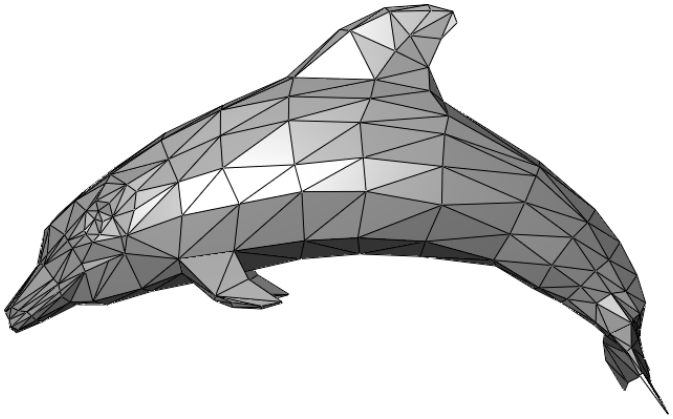

Abbildung 2.8.: Beispiel für ein Polygonnetz. *Quelle: [DOL]*

Grundlagen: Ray-Mesh Intersection

Typischerweise wird die Schnittberechnung zwischen einem Strahl und einem beliebig komplexen Polygon für jedes einzelne Dreieck im Polygon durchgeführt [KC95]. Das grobe Vorgehen hierfür ist das folgende:

1. Führe eine Schnittberechnung mit dem erzeugten Strahl und der Ebene, auf der das zu prüfende Dreieck liegt, durch. Hierbei wird der Abstand t zum Schnittpunkt berechnet.
2. Prüfe für den Schnittpunkt mit dem Abstand t, ob er sich vor dem Strahlursprung und nicht dahinter befindet. Die Bedingung hierfür ist $(t > 0)$. Zusätzlich muss geprüft werden, ob es einen weiteren Schnittpunkt gibt, der sich vor dem Berechneten befindet. Wenn einer dieser beiden Test negativ ausfällt wird der Schnittpunkt verworfen, da er sich entweder hinter dem Strahlursprung befindet, oder ein anderes Objekt die Sicht verdeckt, was in beiden Fällen bedeutet, dass der berechnete Schnittpunkt nicht sichtbar ist.
3. Fallen hingegen beide Tests aus 2. positiv aus, so kann mit Hilfe der Strahlgleichung $\vec{s} = \vec{pos}_0 + t * \vec{v}$, wobei $\vec{pos}_0$ der Ursprung des Strahls, t der berechnete Abstand und $\vec{v}$ der Richtungsvektor des Strahls sind, der Schnittpunkt berechnet werden.
4. Im letzten Schritt muss dann noch geprüft werden, ob sich der berechnete Schnittpunkt im zu prüfenden Dreieck befindet. Dieser Schritt ist mit Abstand der aufwendigste und teuerste im gesamten Ablauf. Mit Hilfe von Baryzentrischen Koordinaten kann der Rechenaufwand jedoch reduziert werden.

   Definition Baryzentrische Koordinaten

   Gegeben sind k Punkte $P_1, P_2, ..., P_k \in \mathbb{R}$. Wenn Punkt Q in der Form $Q = \lambda_1 P_1 + \lambda_2 P_2 + ... + \lambda_k P_k$ mit $\lambda_1 + \lambda_2 + ... + \lambda_k = 1$ dargestellt werden kann, so bezeichnet man $(\lambda_1, \lambda_2, ..., \lambda_k)$ als Baryzentrische Koordinaten von Q bezüglich der Basispunkte $P_1, P_2, ..., P_k$ mit $(k \leq n + 1)$. Siehe hierfür die **Abbildung 2.9**.

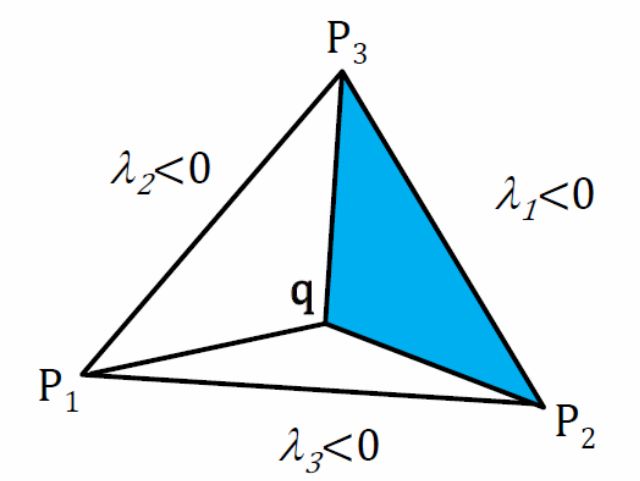

Abbildung 2.9.: Baryzentrische Koordinaten $\lambda_1, \lambda_2, \lambda_3$ von q bezüglich der drei Basispunkte P_1, P_2, P_3

Wie vorhin gezeigt liegen in den meisten Mesh-Datenstrukturen die Vertices der einzelnen Dreiecke explizit vor. Mit Hilfe der Eckpunkte P_1, P_2, P_3 müssen λ_1, λ_2 und λ_3 mit $\lambda_1 = \frac{A_\triangle(Q,P_2,P_3)}{A_\triangle(P_1,P_2,P_3)}$, $\lambda_2 = \frac{A_\triangle(P_1,Q,P_3)}{A_\triangle(P_1,P_2,P_3)}$ und $\lambda_3 = \frac{A_\triangle(P_1,P_2,Q)}{A_\triangle(P_1,P_2,P_3)}$ bestimmt werden. Der Schnittpunkt liegt im Dreieck, wenn λ_1, λ_2 und λ_3 jeweils $> null$ sind.

Der so errechnete Schnittpunkt bezieht sich auf die Weltkoordinaten. Für manche Anwendungsfälle ist es jedoch auch interessant die Koordinaten des Schnittpunkts in Bezug auf die Objektkoordinaten zu kennen. Hierbei ist zwischen 2D- und 3D Objekten zu unterscheiden:

relativer Schnittpunkt in 2D

Bei 2D Objekten ist die Berechnung des Schnittpunkts relativ simpel, da keine Sonderfälle auftreten. Es muss ein Koordinatensystem definiert werden, das sich auf das Objekt bezieht. Hierbei bietet es sich bei 2D Objekten an, die beliebige Formen aufweisen können, einen Punkt in der Mitte des Objekts als Nullpunkt zu definieren. Um nun den Schnittpunkt, der in Weltkoordinaten berechnet wurde, auf die Objektkoordinaten zu übertragen, muss eine Transformationsmatrix von dem Welt- auf das Objektkoordinatensystem ermittelt werden, mit der der Schnittpunkt anschließend multipliziert wird. Der errechnete Punkt ist der relative Schnittpunkt bezogen auf das 2D Objekt.

relativer Schnittpunkt in 3D

Bei 3D Objekten ist die Vorgehensweise zur Berechnung eines relativen Schnittpunkts ähnlich. Es muss auch eine Transformationsmatrix ermittelt werden, mit der der Weltkoordinaten-Schnittpunkt multipliziert wird. Hierbei können aber Sonderfälle auftreten, die beachtet

werden müssen. Bei 3D Objekten kann der errechnete relative Schnittpunkt nämlich auch auf der Rückseite des Objekts liegen und somit für den Betrachter nicht sichtbar sein. D.h. wenn sich noch ein anderer Schnittpunkt finden lässt, der auf der Sichtlinie vor dem errechneten Punkt liegt, so muss er verworfen werden. Andere Sonderfälle sind die seitlichen Flächen von dreidimensionalen Objekten, bei denen es interessant sein könnte zu wissen, in welchem Winkel der Sichtstrahl die Oberfläche trifft. Denn je flacher der Einfallswinkel, desto größer sind die Ungenauigkeiten in Bezug auf die tatsächliche Position wohin der Anwender zeigt. Bei sehr flachen Winkeln können schon kleinste Abweichungen darüber entscheiden, ob ein Objekt angezeigt wird oder nicht.

2.4. EI-Toolkit

In dieser Arbeit wird ein Netzwerktoolkit namens EI Toolkit verwendet [Holo5]. Netzwerkkommunikationen benötigen nämlich für unterschiedliche Ressourcen meist verschiedene Protokolle. Um das Managen der einzelnen Ressourcen der Anwendung abzunehmen, kann das EI-Toolkit eingesetzt werden. Es kümmert sich nämlich um die unterschiedlichen Ressourcen und verwendet zur internen Kommunikation ein einheitliches Protokoll. Dies vereinfacht die programminterne Kommunikation und ermöglicht es dem Entwickler seine Aufmerksamkeit anderen Dingen zuzuwenden.

3. Konzept und Use Cases

Das entwickelte 3D Pointing Toolkit basiert auf einem optischen Trackingsystem und auf einer Echtzeitrendering 3D Graphics Engine. Das Trackingsystem arbeitet mit Licht im Infrarotbereich und baut auf dem „outside-in“ Prinzip mit passiven Targets auf. Da der Versuchsraum isoliert von äußeren Lichtverhältnissen betrieben werden kann und die Größe des zu trackenden Volumens begrenzt ist, ist diese Trackingtechnologie eine der besten Varianten um ein Objekt zu erfassen. Das Ziel des 3D Pointing Toolkits besteht darin, diese erfassten Daten in eine virtuelle Umgebung zu übertragen, um dann im virtuellen dreidimensionalen Raum Schnittberechnungen für unterschiedliche Anwendungsfälle durchführen zu können. Das Toolkit ist so konzipiert, dass Gegenstände aus der realen Umgebung leicht in die virtuelle Umgebung integriert und während der Laufzeit daran Anpassungen vorgenommen werden können. Der User bewegt sich dann in einer virtuellen Umgebung, kann auf Objekte zeigen und weitere Tasks, wie Objekte auswählen oder manipulieren, durchführen. Es ist außerdem so entwickelt, dass zeitlich und räumlich präzise erfasste Events per definierter Netzwerk-API über das UDP-Protokoll verbreitet werden, damit andere Anwendungen auf die Daten zugreifen und sie verarbeiten können. Damit ist die Verwendung des Toolkits sehr vielseitig und es kann auf unterschiedlichste Weisen verwendet werden.

Anwendungsbeispiel Ein simples Anwendungsbeispiel kann so aussehen, dass eine Lampe im Toolkit modelliert wird, damit der Anwender dann per Pointing-Gesten auf das Objekt zeigen und es damit auswählen kann. Ein weiteres Programm, das auf die Netzwerknachrichten des Toolkits hört, kann die Lampe dann z.B. direkt ansteuern und die Helligkeit dieser Lampe regulieren.

Die Einsatzmöglichkeiten des Toolkits sind durch die präzise und schnelle Arbeitsweise sehr vielseitig. Es bietet einen globalen Schnittpunkt und einen zu einem Objekt relativen Schnittpunkt, wodurch auch komplexere Interaktionen ermöglicht werden. Im Folgenden werden einige Anwendungsbeispiele für das vorgestellt 3D Pointing Toolkit präsentiert.

3.1. Use Cases

3.1.1. Steuerung von Haushaltsgeräten

Der User kann direkt auf Objekte, wie z.B. eine Heizung oder die Raumfenster zeigen und auf eine beliebige Art und Weise seine Auswahl bestätigen. Solange der User auf das Objekt zeigt, wird es z. B. durch einen in den Raum integrierten Beamer angestrahlt oder es wird

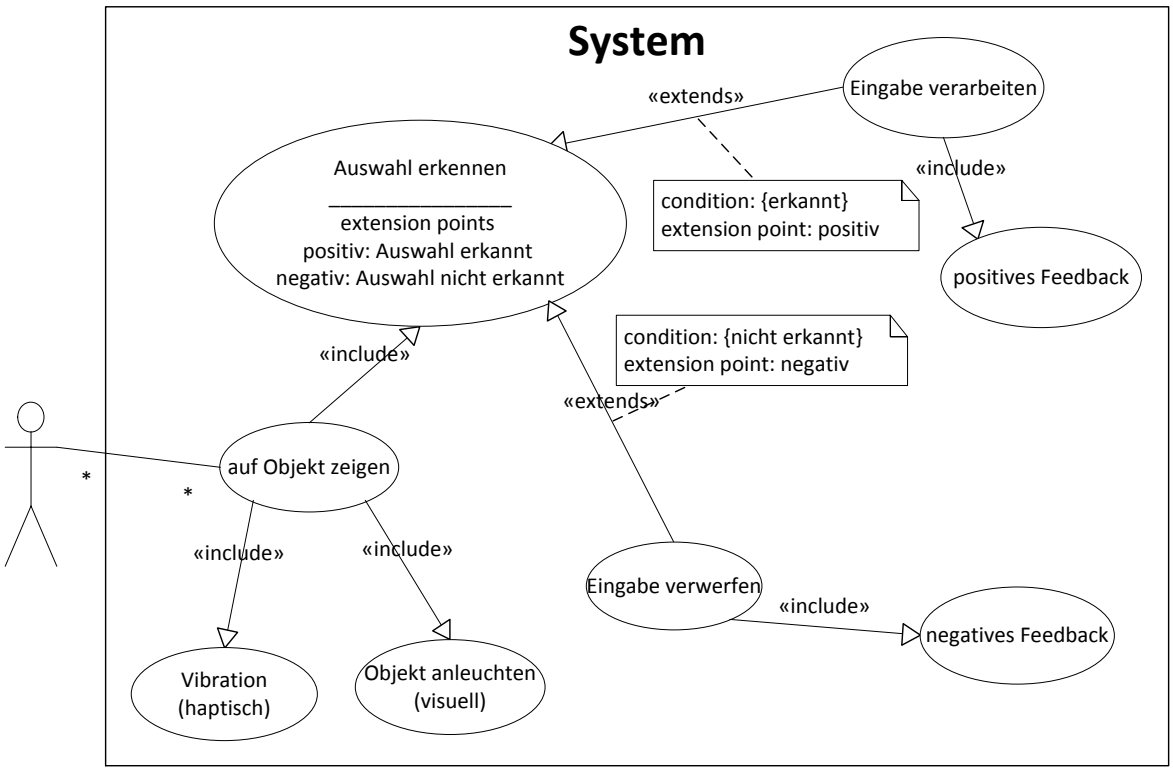

Abbildung 3.1.: Use Case: Steuerung von Haushaltsgeräten mit Hilfe des 3D Pointing Toolkits

ein haptisches oder akustisches Feedback gegeben. Der User hat dann die Möglichkeit durch eine Geste oder eine Spracheingabe das angezeigte Objekt auszuwählen. Wenn das Objekt erfolgreich ausgewählt wurde, kann der User durch Gesten- oder Spracheingaben Befehle an das System übermitteln, die dann verarbeitet werden. So kann z. B. die Leistung einer Heizung durch die Befehle „heißer“ oder „kälter“ angepasst werden, oder die Fenster durch die Befehle „auf“ oder „zu“ geöffnet oder geschlossen werden. Die Heizung liese sich auch durch eine nach oben oder nach unten hin gehende Bewegung der Hand heißer oder kälter machen.

3.1.2. TV-/PC-Steuerung

Ein anderer Use Case für das Toolkit in Verbindung mit anderen Anwendungen, könnte die Steuerung von modernen elektronischen Geräten sein. Der User kann z. B. auf einen Fernseher zeigen und durch eine zusätzliche Geste oder Spracheingabe den Fernseher auswählen. Solange der User nur auf den Fernseher zeigt, kann er durch einen Beamer farblich umrandet werden und wenn der User seine Auswahl bestätigt, kann sich die Umrandungsfarbe verändern um zu signalisieren, dass die Auswahl erfolgreich war. Im gleichen Zug

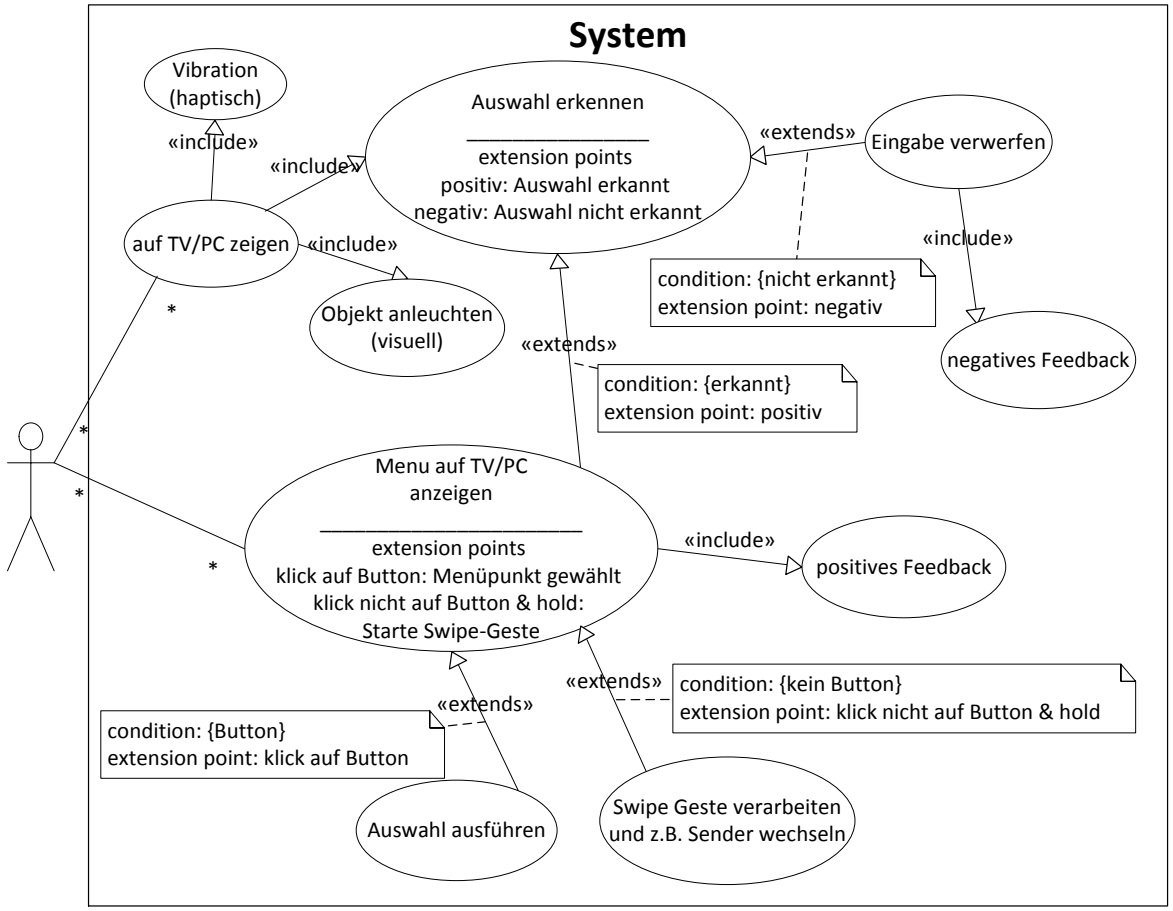

Abbildung 3.2.: Use Case: TV-/PC-Steuerung mit Hilfe des 3D Pointing Toolkits

wird im Fernseher ein Menü angezeigt. Da das Toolkit neben den Weltkoordinaten des Schnittpunkts, auch die relativen Koordinaten in Bezug auf das ausgewählte Gerät ausgeben kann, können die Schnittpunktkoordinaten auf 2D gemapped werden, damit ein kleiner Zeiger auf dem Fernseher angezeigt wird. Das heißt, dass der User genau weiß auf welche Stelle des Fernsehers er genau zeigt und er kann mit diesen Informationen auf ein zuvor eingeblendetes Menü zeigen und damit durch zusätzliche Eingaben mit dem Fernseher interagieren. Dies ist bei den modernen „Smart-TVs“, im Vergleich zu den derzeit verwendeten TV-Fernbedienungen, eine sehr gute und vielseitige Eingabealternative .

Die gezeigten Use Cases, für die das vorgestellte 3D Pointing Toolkit verwendet werden kann, sind nur eine kleine Auswahl. Durch den implementierten Editor ist es nämlich möglich, beliebige virtuelle Umgebungen zu schaffen und immer an die Gegebenheiten der realen Umwelt anzupassen.

3.2. Bedeutung für Nutzerstudien

Die Mensch-Computer-Interaktion wird früher oder später auch in den modernen Haushalt integriert werden. Es gibt jetzt schon Möglichkeiten mit seinem Smartphone von Unterwegs seine Heizung oder Klimaanlage zu steuern, um sich beim Eintritt in sein „Home-Sweet-Home“ komplett wohlzufühlen. Kühlschränke können ihren Inhalt automatisch überwachen, um selbstständig eine Einkaufsliste für den Besitzer zu erstellen, damit dieser beim Einkauf auch nichts Wichtiges vergisst. Die heutigen Fernseher bieten mittlerweile auch ein eigenes Interface, das man dazu verwenden kann sich online Filme anzuschauen, oder sogar im Internet zu surfen. Es wird derzeit jedoch keine uneingeschränkte Interaktion mit solchen Haushaltsgeräten ermöglicht und es ist noch nicht absehbar, wann es erschwingliche Systeme geben wird, die eine intuitive Interaktion ermöglichen. An diesem Punkt greift das entwickelte 3D Pointing Toolkit, das zwar auf einer sehr kostspieligen Hardware aufbaut, aber es ermöglicht Erkenntnisse über die zukünftigen Interaktionsmöglichkeiten in modernen Haushalten zu gewinnen. Es können Daten für verschiedenste Forschungsgebiete gesammelt und ausgewertet werden um dementsprechend angepasste Systeme zu entwickeln.

Das Toolkit basiert außerdem darauf eine Auswahl durch Pointing-Gesten zu treffen, was eine intuitive und einfache Möglichkeit für den Menschen darstellt einen Gegenstand zu erfassen. Ein Mensch lernt nämlich schon als Kleinkind sich durch „Zeigen“ auf gewisse Sachen mit seiner Umwelt zu verständigen. Dies ist ein Teil der sogenannten Deixis, einem Fachbegriff aus der Sprachwissenschaft. Gesten, die dazu verwendet werden eine Nachricht zu übermitteln, werden von Kleinkindern im vorsprachlichem Alter von Natur aus erlernt [Gro11]. Hierzu gehört auch die Zeigegeste, die nach Bates et al. [BE79] im Speziellen zu den deiktischen Gesten zählt und fast immer einen Bezug auf ein spezielles Objekt hat. Es wurde durch Pizzuto und Copabianco in einer Feldstudie festgestellt, dass Kleinkinder in ihrer Kommunikation sehr häufig deiktische Gesten, insbesondere die Zeigegeste, verwenden um sich ihrer Umwelt mitzuteilen [PE05]. Diese Art sich zu Verständigen behält jeder Mensch auch im Alter bei und er kann damit besser verdeutlichen, was er ausdrücken will und in vielen Fällen das Gesagte auch einfach unterstreichen. Hier zeigt sich, dass es sinnvoll ist Zeigegesten zur Interaktion mit „Smart-Homes“ zu verwenden und dass es einem User leicht fallen könnte, diese Interaktionstechnik anzuwenden. Damit ist es sehr interessant in Studien noch folgende offene Fragen zu untersuchen und zu klären:

Genauigkeit Eine Zeigegeste setzt sich meist aus den Wahrnehmungen, die man mit seinen Augen macht und der entsprechend angepassten Koordination seiner Hand zusammen. Hierbei ist die Hand-Auge-Koordination von Mensch zu Mensch etwas unterschiedlich und die Zeigerichtung wird von jedem etwas anders wahrgenommen. Dementsprechend muss geprüft werden bei welcher Haltung der Hand welche Abweichung stattfindet. Es muss außerdem geprüft werden, wie groß die Objekte im Toolkit modelliert werden müssen, damit ein User sie leicht und zielsicher treffen kann.

Feedback Ein weiterer interessanter Punkt, der untersucht werden muss, ist das Feedback, das dem User gegeben wird. Ein Mensch benötigt nämlich für seine Aktionen eine entsprechende Reaktion um sagen zu können, ob das was er gemacht hat richtig

war. Hierbei können alle vom Menschen vorhandenen Wahrnehmungskanäle genutzt werden. Es gibt die Möglichkeiten ein Feedback über die visuelle, auditive, olfaktorische oder gustatorische Wahrnehmung oder auch über die Sensibilität, also dem Tastsinn zu geben, bei denen jeder einzelne Feedback-Typ seine Vor- und Nachteile hat.

Benutzbarkeit Das ganze System ist auch mit mehreren Targets zu betreiben. In diesem Fall ist es interessant zu wissen, ob ein User die gewünschten Interaktionen mit einer Hand ausführen kann, oder ob es sinnvoll ist beide Arme und vielleicht sogar die Beine zu tracken. Dies gibt dem Anwender nämlich eine größere Bewegungsfreiheit. Dadurch, dass mehr als nur eine Hand getracked wird, kann es jedoch sein, dass die Interaktion zu komplex wird oder es für das System zu schwierig wird die Gesten richtig zu interpretieren.

4. 3D Pointing Toolkit

Im nun folgenden Hauptteil werden die Grundbausteine des entwickelten Toolkits vorgestellt. Es werden die Zusammenhänge zwischen den einzelnen Teilen erläutert und in Betracht auf Skalierbarkeit, Benutzerfreundlichkeit, Performance und Robustheit untersucht. Als Grundlage für das Toolkit wird das optische Trackingsystem „OptiTrack“verwendet, das mit der 3D Graphics Engine Ogre3D über UDP Pakete kommuniziert. Am Anfang wird das verwendete Trackingsystem und das System-Setup beschrieben, wonach die Kommunikation zwischen den zwei System beleuchtet wird. Das eigentliche Verarbeiten der generierten Daten, nämlich das Echtzeitrendering, wird im darauf folgenden Teil im Detail beschrieben. Beim Echtzeitrendering werden dann Fragen geklärt, die die Repräsentation der dreidimensionalen Szene, die interne Objektrepräsentation, die Schnittberechnung und die definierte Netzwerk-API betreffen. Für die Bedienung des 3D Pointing Toolkits wurde ein GUI verwendet, das Möglichkeiten bietet den integrierten Editor zu bedienen und im „Live-Modus“ z. B. die Kameraperspektive sinnvoll zu steuern.

4.1. System-Setup

Das verwendete Trackingsystem basiert auf dem optischen Tracking. Hierfür wird Infrarotlicht von neun Motion-Capture Kameras 2.3, die rechteckig an der Decke angebracht sind, ausgestrahlt und das von den Targets reflektierte Restlicht erfasst. Die Targets sind mindestens drei kleine sphärische Kugeln, die an einem Marker in einer genau definierten Ausrichtung angebracht sind 2.2. Mit dem System ist eine Frame-Rate von 100 FPS möglich und es wird ein Trackingvolumen von mehr als $12m^2$ erfasst. Der Marker selbst wird dann wiederum an der Hand des Users angebracht. Das Trackingsystem verarbeitet die erfassten Trackingdaten und übergibt die X-,Y- und Z-Position und die Orientierung, die eine Winkelabweichung der einzelnen Achsen darstellt, an die Kommunikations-Einheit. Die verarbeiteten Daten bekommt die Hauptanwendung, die sie weiterverarbeitet und auf dem Ausgabemonitor darstellt.

4.2. Kommunikation

Die zwei Kernelemente, auf denen die Kommunikation basiert, ist das vom Hersteller des Trackingsystems gelieferte NatNet SDK [NAT13] und das an der Universität Stuttgart entwickelte EI Toolkit [Holo5]. Das NatNet SDK ist eine Client/Server-Architektur, die es

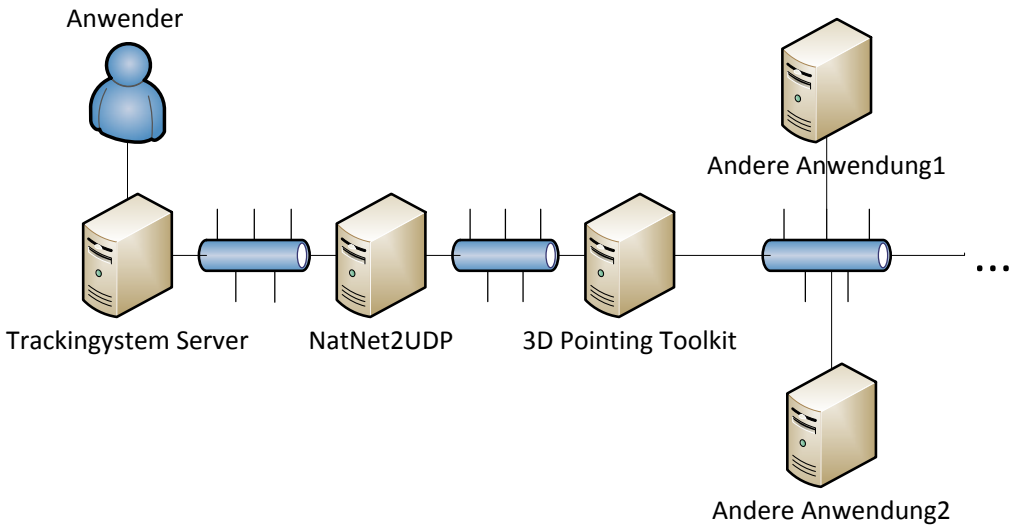

Abbildung 4.1.: Kommunikationstruktur und Dataflow des gesamten Systems. Die Daten werden vom Anwender am Trackingserver eingegeben, wonach diese vom NatNet2UDP Tool zu Nachrichten im UDP Format umgewandelt werden. Die umgewandelten Daten werden dann schlussendlich vom 3D Pointing Toolkit verarbeitet und anderen Anwendungen zur Verfügung gestellt.

ermöglicht, die vom Trackingsystem generierten Daten, auf unterschiedliche Weisen den entsprechenden Applikationen mitzuteilen, wobei das Trackingsystem gleichzeitig den Server bildet. Um die generierten Daten anderen Anwendungen zur Verfügung zu stellen, wird in diesem Fall das UDP-Protokoll verwendet, da die verwendete Netzwerkinfrastruktur auf Ethernet basiert und prinzipiell keine Pakete verloren gehen und die Paketreihenfolge eingehalten wird. Um noch eine höhere Abstraktion in der Kommunikation zu erreichen, wird das sogenannte EI Toolkit verwendet. Das EI Toolkit ist dafür zuständig unterschiedlichste Eingangssignale und Protokolle zu verarbeiten und ein einheitliches Protokoll für die Kommunikation mit einer anderen Applikation zu bieten.

4.2.1. NatNet SDK/NatNet2UDP

Wie in **Abbilung 4.2** zu sehen ist, bekommt das NatNet SDK die Daten vom Tracking Tools Server über das Netzwerk per Multicast bzw. Unicast zugesendet. Multicast ist die leichtere Variante, da keine spezifische IP oder kein definierter Port benötigt werden. Die empfangenen Daten werden dann über UDP weiter verschickt.

4.2.2. EI-Toolkit

Abbildung 4.3 zeigt die Architektur des EI Toolkits. Externe wie auch interne Ressourcen werden durch sogenannte „Stubs“ repräsentiert, die Daten von den einzelnen Ressourcen erhalten aber auch an sie verschicken können. Der große Vorteil des EI-Toolkits ist der, dass die Anwendung sich nicht mit der Steuerung und Organisation der einzelnen Ressourcen beschäftigen muss und es sehr einfach ist Ressourcen hinzuzufügen oder zu entfernen. Für das vorgestellte 3D Pointing Toolkit bedeutet dies, dass die eingehenden Signale des NatNet SDKs vom EI Toolkit gemanagt werden und die relevanten Daten mit einem allgemeinen Protokoll der Anwendung zur Verfügung gestellt werden.

4.3. Echtzeitrendering

Eine der größeren Anforderung an das entwickelte Toolkit besteht darin, die vom Trackingsystem generierten Daten, die mit 100Hz erstellt werden, in Echtzeit zu verarbeiten und eine ähnlich gute Wiederholungsrate bei der Verarbeitung und Ausgabe zu bieten. Hierbei bietet

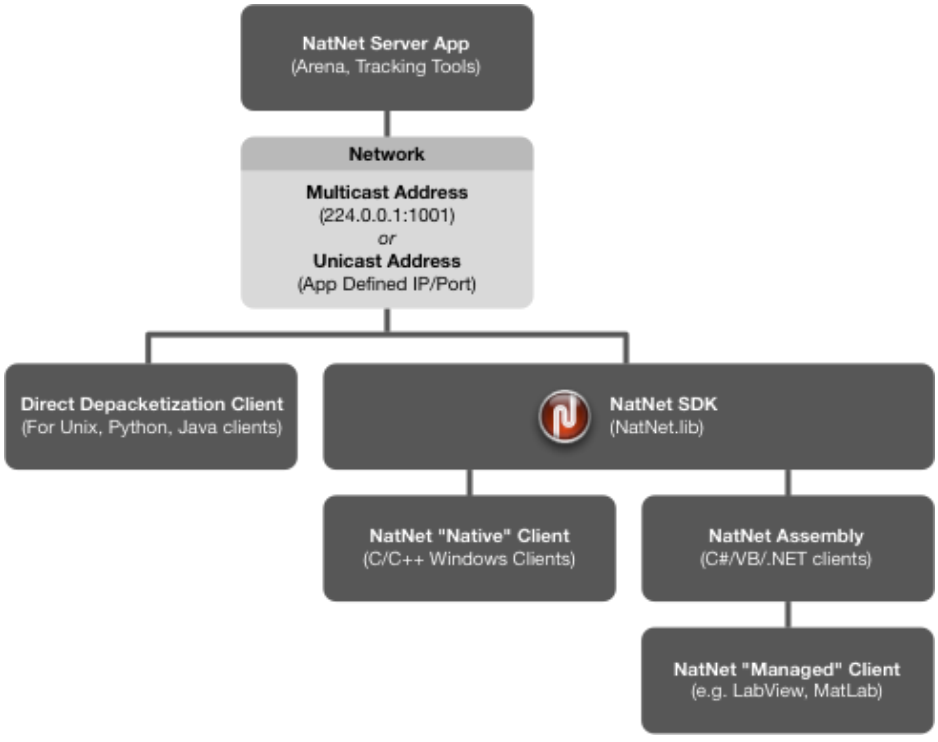

Abbildung 4.2.: Dataflow des NatNet SDKs. Der Tracking Tool Sever sendet per Netzwerk Daten an das SDK, wonach das SDK die Daten über das UDP Protokoll weiterschickt. *Quelle: [NAT13]*

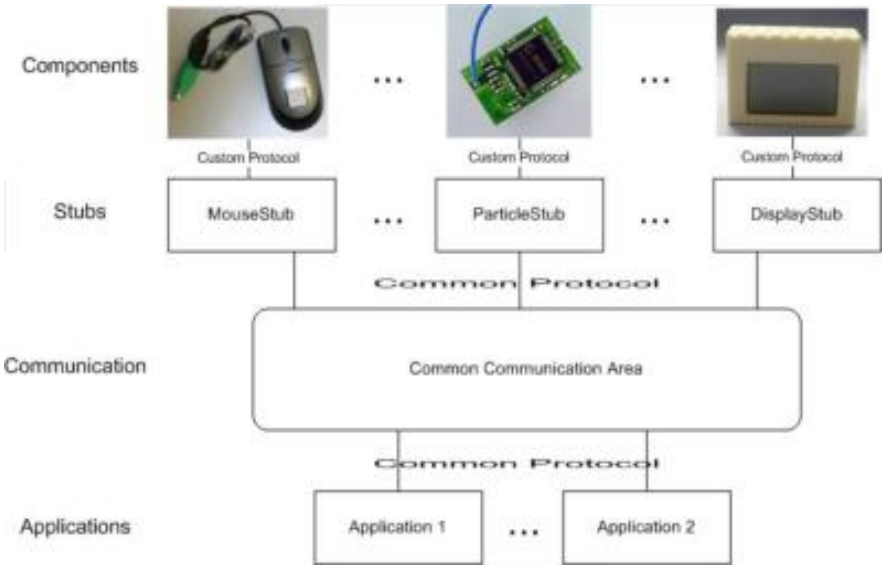

Abbildung 4.3.: Funktionsweise des EI Toolkits. Jede angeschlossene Komponente wird durch einen Stub gemanagt. Für die interne Kommunikation wird ein einheitliches Protokoll verwendet, das unabhängig vom angeschlossenen Gerät immer gleich ist. *Quelle: [Holo5]*

sich der Einsatz der hardwarenahen Programmiersprache C++ und den zwei bekannten Programmierschnittstellen OpenGL und DirectX an. Sie ermöglichen es nämlich komplexe 3D Szenen schnell darzustellen. Diese zwei 3D-APIs sind sehr vielseitig und können an viele Anforderungen angepasst werden, sie sind jedoch in ihrer Bedienung bei grundlegenden Aufgaben sehr aufwendig. Hierfür gibt es 3D Grafik Engines, die den Einsatz von vielen Grundfunktionalitäten der 3D Computergrafik vereinfachen und dem Anwender leichter zugänglich machen. Ein Beispiel hierfür ist die Open Source 3D Grafik Engine Ogre3D [OGR09], die objektorientiert ist und von dem vorgestellten 3D Pointing Toolkit verwendet wird. Ogre3D verwendet einen Szenegraphen und bietet eine mitgelieferte Library zur Verarbeitung von Benutzereingaben. Zusätzlich können noch weitere Bibliotheken leicht integriert werden. Durch die Verwendung der zwei erwähnten 3D-APIs und der Programmiersprache C++ kann eine sehr hohe Performance des Tools erreicht werden, da alle Bauteile der Computerhardware für die jeweils richtige Aufgabe verwendet werden. Am Anfang wird auf die Fragestellung eingegangen, wie man ganze Szenen aus der Realwelt modellieren, im Editor des Toolkits verändern und am Ende speichern kann. Im Anschluss werden die wichtigsten Einzelteile des Echtzeitrendering aufgegriffen und genauer betrachtet.

4.3.1. Szenen Repräsentation

Wie in **Abbildung 4.4** zu sehen ist, verwendet Ogre3D einen SceneManager, der für alle SceneNodes verantwortlich ist. Die einzelnen SceneNodes sind für die Objekte verantwortlich,

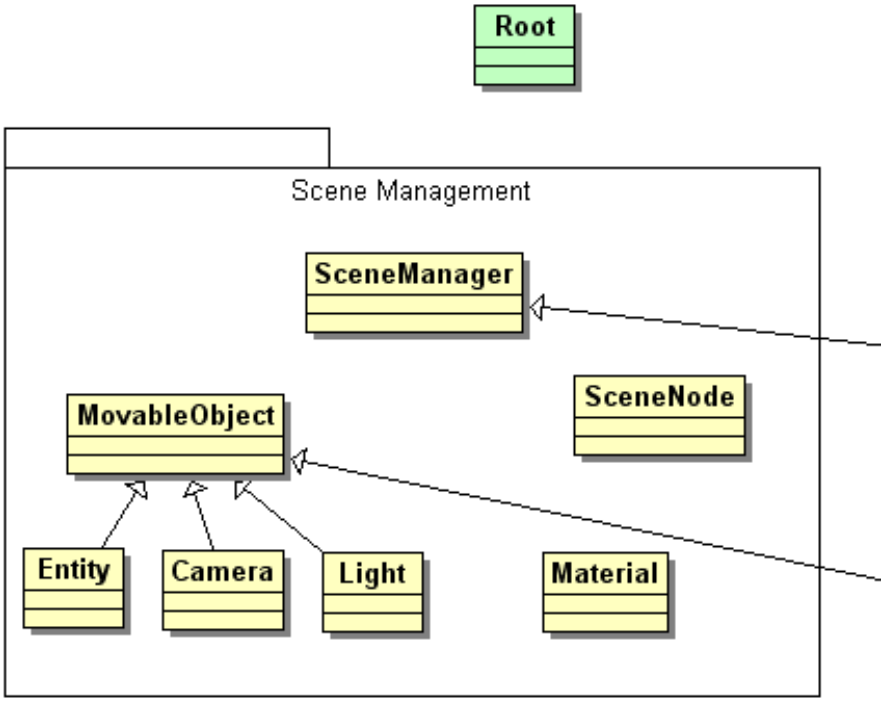

Abbildung 4.4.: Das SceneManagement von Ogre3D. Ein SceneManager ist für die ganze Szene zuständig. Ihm werden einzelne SzeneNodes zugewiesen, denen MovableObjects zugewordnet werden können. *Quelle: [STR]*

die an sie angehängt werden. Das bedeutet, dass der SceneManager ein eigenes System hat, in dem er die Positionen, Orientierungen und weitere Eigenschaften der einzelnen SceneNodes verwaltet. Die SceneNodes wiederum haben ein eigenes System, in dem sie die Informationen der an sie angehängten Objekte verwalten. Eine Szene die Ogre3D verwendet, kann programmatisch beschrieben werden, was jedoch den Nachteil hat, dass Veränderungen, die während der Laufzeit an der Szene vorgenommen werden, nach Beenden des Programms nicht gespeichert werden. Hierfür verwendet das entwickelte Toolkit stattdessen eine XML-Repräsentation der Szene, die bei jedem Start der Anwendung eingelesen und vor dem Beenden wieder abgespeichert wird. Die einzelnen Objekte, die für die Interaktion und somit für die Schnittberechnung verwendet werden, werden zusammen mit ihrem Namen und ihren vier Eckpunkten in der XML-Datei abgespeichert. Für jedes Objekt wird beim Programmstart ein neuer SceneNode erstellt, der nur für dieses Objekt verantwortlich ist. Der Name wird jedem Objekt intern zugewiesen und alle Operationen, die für ein Objekt durchgeführt werden, laufen auf dem zugewiesenen Namen ab. Dies trägt zur Skalierbarkeit des Toolkits bei, da beliebig viele Objekte in die Berechnungen eingebunden werden können und das Toolkit alles unabhängig von der Anzahl, der in der Szene existierenden Objekte und nur anhand der Objektnamen verarbeiten kann. Im Anhang findet sich eine beispielhafte XML-Repräsentation einer kleinen **Szene A.5 auf Seite 58**.

4.3.2. Echtzeitrendering

Der Kern des 3D Pointing Toolkits ist die 3D Repräsentation der „Realwelt-Szene“ und die Schnittberechnung zwischen Zeigestrahl des Anwenders und der sich im Raum befindenden Objekte. Da das Toolkit so konzipiert wurde, dass es skalierbar ist muss natürlich auch der Schnittberechnungsalgorithmus unabhängig von der Anzahl der zu testenden Objekte ablaufen. Dies geschieht mit Hilfe des Szenegraphen von Ogre3D, der alle vorhandenen Objekte mit Hilfe von eindeutig zuzuordnenden String-Namen verwaltet und für unterschiedlichste Berechnungen bereit hält. Wie schon erwähnt enthält das vorgestellte Toolkit einen Editor, der es dem Anwender gestattet während der Laufzeit kleine Änderungen an der Szene vorzunehmen. So können noch kleine Ungenauigkeiten korrigiert werden, ohne die ganze Anwendungen beenden zu müssen. Um dem Anwender eine angenehme Bedienung zu ermöglichen, wurde ein GUI implementiert, das die Verwendung des Toolkit vereinfachen soll. In diesem Abschnitt wird im ersten Teil das GUI erläutert. Im darauf folgenden Teil wird der Kern, nämlich der Algorithmus zur Schnittberechnung, vorgestellt und am Ende wird die Netzwerk-API präsentiert, die von anderen Anwendungen verwendet werden kann.

Toolkit-GUI

Die implementierte grafische Benutzoberfläche soll dem Anwender erleichtern, alle wichtigen Funktionalitäten auf einfach Art und Weise zu bedienen, um damit die Benutzerfreundlichkeit des Toolkits zu steigern. Folgende Hauptfunktionalitäten stehen zur Verfügung:

Live-Mode Das Toolkit bietet einen „Live-Mode Button“, der dem Benutzer eine freie Sicht auf die gerenderte Szene bietet und noch verschiedene vordefinierte Kamerapositionen zur Auswahl stellt. **Abbildung 4.5** zeigt die Anwendung im Live-Mode.

Editor-Mode Es steht außerdem ein „Editor Button“ zur Verfügung. Hier hat der Benutzer die Möglichkeit, die zwei wichtigen Funktionalitäten „Add Rectangle“ und „Move Object“ auszuwählen. Wie in **Abbildung 4.6** und **Abbildung 4.7** zu sehen ist, erscheint nach der Auswahl einer der zwei Funktionen jeweils ein neues Fenster. Bei „Add Rectangle“ werden vom Benutzer alle vier Eckpunkte des zu zeichnenden Rechtecks erfragt. „Move Object“ bietet dem User ein Objekt aus der Szene auszuwählen und die Position bzw. die Orientierung des Objekts durch Rotation zu verändern.

Schnittberechnung

Ogre3D bietet eine integrierte Möglichkeit sogenannte „Rays“ mit einem definierten Startpunkt und einer Richtung zu generieren, um dann festzustellen, ob Kollisionen mit Objekten in der Szene auftreten. Wenn Kollisionen zwischen Strahl und Objekt auftreten, werden alle Kollisionen mit der entsprechenden Entfernung vom Startpunkt bis zum Kollisionspunkt in einem Buffer gespeichert. Problematiken wie die Frage, welches Objekt in einer Szene von welchem anderen Objekt verdeckt wird oder auch andersherum, sowie die Frage welche Objekte nicht in die Schnittberechnung eingehen sollen, müssen vom Algorithmus zur

Schnittberechnung beachtet werden. Da Ogre3D alle Objekte in einem Szenegraphen mit einem Rootknoten verwaltet, geht der in **Abbildung A.6 auf Seite 59** gezeigte Algorithmus zur Ermittlung des relevanten Schnittpunkts folgendermaßen vor:

1. Der Buffer mit den Ergebnissen für Kollisionen enthält alle geschnittenen Objekte sortiert in der Reihenfolge, in der die Objekte geschnitten wurden. Da für Pointing-Gesten meist nur das vom User aus erste Objekt relevant ist und alle anderen davon verdeckt werden, wird im gezeigten Algorithmus nur auf das erste Objekt geachtet. Im ersten Schritt fragt der Algorithmus beim Wurzelknoten nach seinem ersten Kindknoten.
2. Für alle Kindknoten vergleicht der Algorithmus, ob der Name des zuerst geschnittenen Objekts mit dem Namen des Objekts, der am Kindknoten angehängt ist, übereinstimmt.

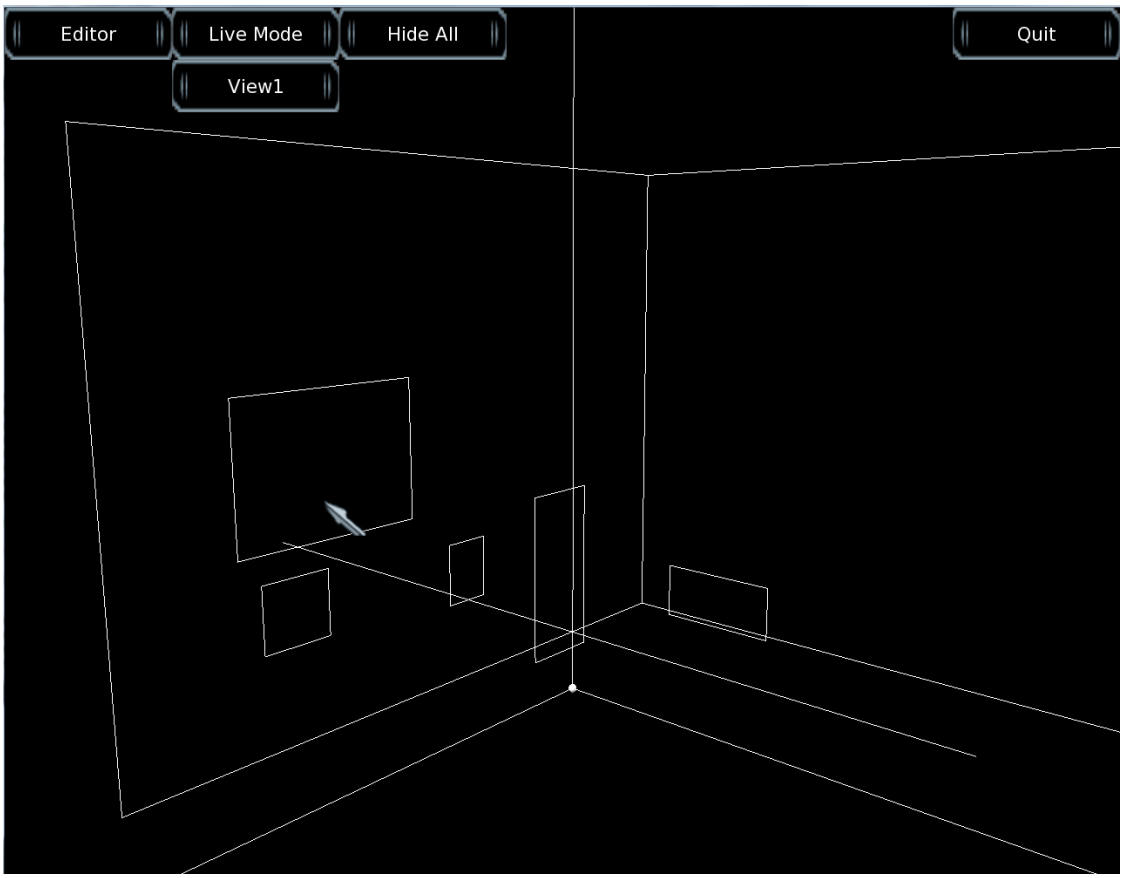

Abbildung 4.5.: Das System im Live-Mode: Uneingeschränkte Sicht auf die 3D Szene, die Schnittberechnungen und Ereignisse. Es stehen zusätzlich ein oder zwei vordefinierte Viewpoints zur Auswahl, um dem Anwender die Interaktion zu erleichtern.

Abbildung 4.6.: Das System im Editor-Mode: Der Anwender hat die drei Auswahlmöglichkeiten „Add Rectangle“, „Move Object“ und „Console“. Hier sieht man das eingeblendete Fenster, mit dem man zur Szene ein neues Rechteck hinzufügen kann. Nachdem die vier Randpunkte und ein Name eingegeben sind, kann die Auswahl mit „Add Rectangle“ bestätigt werden, wonach gleich das neue Objekt in der Szene erscheint.

3. Wenn das der Fall ist hat der Algorithmus den richtigen Kindknoten, der für das relevante Objekt zuständig ist, gefunden und es können alle nötigen Operationen auf dem Objekt bzw. auf seinem zuständigen Knoten durchgeführt werden.
4. Im letzten Schritt ermittelt der Algorithmus den Schnittpunkt in Bezug auf das Weltkoordinatensystem durch einsetzen der von Ogre3D mitgelieferten Distanz in die Strahlgleichung $\vec{s} = \vec{pos}_0 + t * \vec{v}$, wobei $t = Distanz$.

Netzwerk-API

Das 3D Pointing Toolkit verwendet UDP-Packete um die gewonnen Informationen an Anwendungen zu übermitteln, die sie weiterverarbeiten. Die API sieht folgendermaßen aus:

$$OT : typ : x : y : z : object$$

Die Netzwerk-API soll es einer anderen Anwendung ermöglichen, die Daten sinnvoll für ihre Zwecke verwenden zu können. Sinnvoll bedeutet in dem Fall, dass die andere Anwendung erkennt, dass es sich um Pakete vom 3D Pointing Toolkit handelt. Hierfür werden an den Anfang der übermittelten Pakete die Buchstaben *OT* gestellt, die für „OptiTrack“stehen. Außerdem muss übermittelt werden, wie die X-,Y- und Z-Koordinaten des berechneten

Abbildung 4.7.: Das System im Editor-Mode: Hier wurde die Auswahl „Move Objekt“ getroffen. Der Anwender kann einen Objektnamen eingeben, mit den vier Pfeilen bewegen und im oder gegen den Uhrzeigersinn drehen.

Schnittpunkts lauten, was mit $x : y : z$ erledigt wird. Am Ende des Pakets steht der Name des Objekts, welches gerade geschnitten wurde. Außerdem muss unterschieden werden um welche Sorte von Schnitt es sich handelt. Dies passiert an der Stelle *typ* im Paket. Es wird zwischen drei verschiedenen Schnittpunkttypen unterschieden, da es interessant und für den Einsatzbereich des Toolkits auch wichtig zu wissen ist, ob in dem Moment nur ein Übergang von einem auf ein anderes Objekt stattgefunden hat, oder ob sich nur der Punkt, auf den der Anwender in ein und dem selben Objekt zeigt, verändert hat. Der erste Fall wird benötigt, um unterschiedliche Feedbacktypen beim Ändern des angezeigten Objekts zu testen. Dafür muss die Anwendung, die auf das Toolkit hört, nämlich wissen, ob es eine Veränderung in Bezug auf das derzeit angezeigte Objekt gab. Wenn sich das Objekt verändert, muss z. B. der Lichtspot vom alten auf das neue Objekt gerichtet werden. Hierfür werden die zwei Signale „in“ und „out“ verwendet. Hierbei signalisiert *in*, dass ein neues Objekt angezeigt wird und *out* signalisiert, dass ein Objekt nicht mehr angezeigt wird. Der zweite Fall wird mit dem Signal „on“ belegt und bedeutet, dass der Anwender noch auf das selbe Objekt zeigt. Es ist noch hinzuzufügen, dass das Toolkit *in*- und *out*-Signale nur bei einer tatsächlichen Veränderung verschickt und das *on*-Signal kontinuierlich mit 10Hz übermittelt wird.

4.4. Evaluierung des Systems

Das vorgestellte Toolkit ist darauf ausgelegt 3D Positions- und Orientierungsdaten eines optischen Trackingsystems in Echtzeit auszuwerten und anderen Anwendungen zur Verfügung zu stellen. Hierbei ist besonders die Bildwiederholungsrate wichtig, da dadurch eine Aussage über die Verarbeitungsgeschwindigkeit des Gesamtsystems getroffen werden kann. Das Echtzeitrendering beinhaltet nämlich die Schnittberechnungen im dreidimensionalen Raum und es verwaltet auch die gesamte virtuelle Szene. Damit ist das Rendering der Flaschenhals des gesamten 3D Pointing Toolkits. Das System wurde so implementiert, dass die Framerate von Anfang an auf 100FPS begrenzt ist, da das Trackingsystem Daten mit 100Hz verschickt. Diese maximale Framerate wird mit zunehmender Komplexität der virtuellen Szene niedriger.

4.4.1. Testhardware und Ergebnis

Das entwickelte Toolkit wurde auf einem Testsystem mit einem Intel CPU mit vier Kernen und jeweils einer Taktrate von 2.83GHz und einer NVIDIA Quadro NVS290 Grafikkarte evaluiert. Dieses System ermöglicht bei einer komplexen Szene mit über 250 eigenständigen Objekten und über 1,5 Mio gerenderten Dreiecken eine maximal Framerate von 87FPS **Abbildung 4.8**. Dieser Maximalwert kommt dann zustande, wenn kein einziges Dreieck im sichtbaren Bereich liegt, im Hintergrund aber trotzdem alle einzelnen Objekte vom SceneManager gemanagt werden müssen. Die Abbildung zeigt, dass die Framerate mit zunehmender Anzahl an gerenderten Dreiecken abnimmt und ihr Minimum bei 23FPS liegt. Dies ist bei einer solch komplexen Szene noch eine hinnehmbare Bildwiederholungsrate. Als Vergleichswert für ein angenehmes und ruckelfreies Bild können 25FPS angenommen

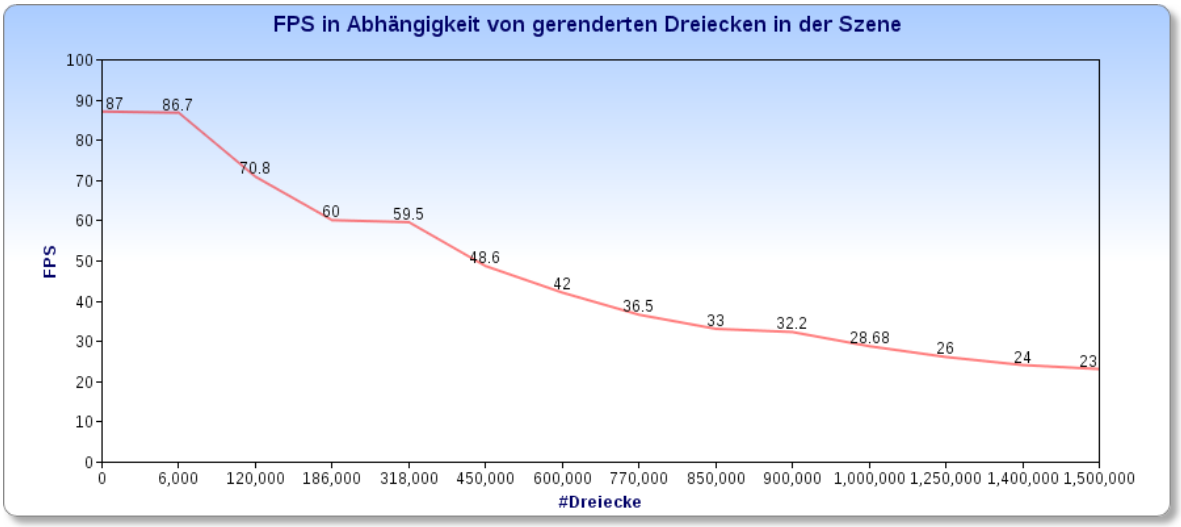

Abbildung 4.8.: In dieser Abbildung ist zu sehen wie ein Testsystem, auf dem das 3D Pointing Toolkit ausgeführt wird, mit einer komplexen virtuellen Szene zurechtkommt. Hierbei wurde eine Szene mit mehr als 250 eigenständigen komplexen Objekten verwendet, die zusammen über 1,5 Mio Dreiecke beinhalten. Das Diagramm lässt erkennen, dass ein durchschnittliches Testsystem auch bei einer sehr komplexen Szene ca. 25FPS liefern kann, was einer akzeptablen Bildwiederholungsrate entspricht.

werden, wie sie bei jedem normalen Film verwendet werden. Damit ist zu sehen, dass bei einem durchschnittlichen Computersystem sehr komplexe Szenen mit ordentlichen Frameraten verarbeitet werden können.

4.4.2. Interaktion mit dem Toolkit

Um Schnittberechnungen mit Hilfe der Zeigegesten durchführen zu können, werden die Gegenstände aus der realen Umgebung in die virtuelle Szene des Toolkits integriert. Hierbei hat sich bei der Entwicklung des Toolkits die Frage gestellt, ob die virtuellen Gegenstände die identischen Ausmaße haben sollen, wie die Objekte aus dem realen Testraum. Anfangs wurden die Größen der Objekte eins zu eins übernommen. Nach den ersten Testläufen wurde jedoch festgestellt, dass das optische Trackingsystem zwar zentimetergenaue Ergebnisse liefert, es dem Anwender aber trotzdem schwer fällt ein bestimmtes Objekt anzuvisieren. Es wurden drei Faktoren gefunden, die einen Einfluss auf die Interaktion mit dem Toolkit haben:

Auge-Hand-Koordination Das 3D Pointing Toolkit in Verbindung mit dem optischen Trackingsystem „OptiTrack“ wurde von verschiedenen Personen getestet und bewertet. Bei diesen Tests wurde die Feststellung gemacht, dass jeder Mensch eine individuelle

Empfindung in Bezug auf die Zeigerichtung seiner Hand hat. Hierbei hat sich gezeigt, dass die Anwender besonders bei kleinen Gegenständen Probleme hatten einzuschätzen, ob sie nun auf das mit den Augen anvisierte Objekt zeigen oder nicht. Es wurde darauf verzichtet, dem Anwender ein optisches Feedback zu geben, wohin er genau zeigt. Dieser Umstand war sehr förderlich dabei festzustellen, wie die individuelle Wahrnehmung der Zeigerichtung von der Realität abweicht, da der Anwender keine Möglichkeit hatte seine Bewegungen durch optische Wahrnehmungen zu korrigieren. Um diese Abweichungen, die vom Anwender abhängig sind, zu korrigieren, bieten sich folgende zwei Möglichkeiten an:

- 1. In einer aufwendigen Kalibrierungsphase, können die individuellen Abweichungen der Auge-Hand-Koordination in Abhängigkeit von der Stellung des Kopfes und der Hand korrigiert werden. Hierbei muss jedoch jeweils die Hand und der Kopf vom Trackingsystem erfasst werden, um feststellen zu können bei welcher Auge-Hand-Konstellation welche Abweichungen auftreten.
- 2. Die wesentlich einfachere Möglichkeit die auftretenden Fehler korrigieren zu können, ist es dem Benutzer ein optisches Feedback in Form eines Punktes zu geben. Bei dieser Variante kann der Anwender mögliche Abweichungen durch seine Wahrnehmungen ausgleichen.

Schwankungen im Randbereich Ein weiterer Faktor, der die Interaktion mit der virtuellen Szene beeinflusst, ist die Genauigkeit und Empfindlichkeit des optischen Trackingsystems. Es werden nämlich schon sehr feine Abweichungen erfasst. Das kann besonders im Randbereich von Objekten dazu führen, dass ein Anwender sie eigentlich anvisiert, aber schon eine kleine unbeabsichtigte Bewegung diesen Umstand verändern kann. Dieses Phänomen wurde von den meisten Anwendern als unerwünscht empfunden und kann durch die Vergrößerung der virtuellen Objekte behoben werden.

Entfernung und Winkel zum Objekt Eine weitere Rolle spielt die Entfernung und Ausrichtung zu dem Objekt, das anvisiert werden soll. Abhängig davon wie weit der Anwender vom ausgewählten Gegenstand entfernt ist und in welchem Winkel er zu dem Objekt steht, verändern sich die Parameter der Zeigegeste. Wenn ein Objekt aus einem sehr flachen Winkel anvisiert wird, so sind schon kleinste Abweichungen in der Zeigerichtung entscheidend darüber, ob ein Objekt ausgewählt wird oder nicht. Diesen Sachverhalt veranschaulicht **Abbildung 4.9**. Wenn der Anwender einen Gegenstand jedoch direkt von vorne anzielt, so hat er mehr Spielraum und kleinere Abweichungen sind nicht so entscheidend. Ähnlich verhält es sich mit der Entfernung zu dem Objekt, das der Anwender auswählen will. So ist es aus der Nähe einfacher genaue Zeigegesten durchzuführen, da Veränderungen in der Zeigerichtung eine kleine Auswirkung haben. Wenn der Anwender jedoch weiter vom Objekt entfernt ist, so können schon kleine Änderungen des Zeigevektors darüber entscheiden, ob ein Objekt angezeigt wird oder nicht. **Abbildung 4.9** soll dies verdeutlichen. Dieses Problem kann wiederum durch eine Vergrößerung der virtuellen Objekte beseitigt werden, da damit Auswirkungen der Entfernung und des Winkels reduziert werden.

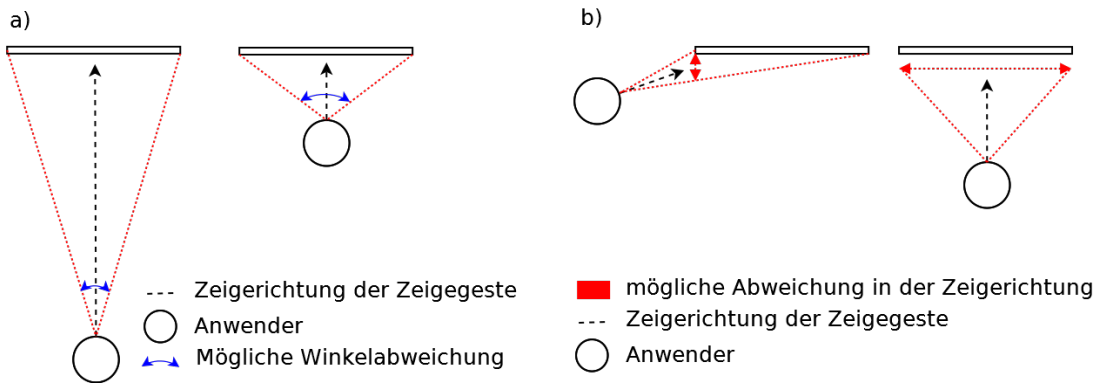

Abbildung 4.9.: a) Die Entfernung des Anwenders zum Objekt beeinflusst die Winkelabweichung, die möglich ist, sodass der Anwender immer noch auf ein anvisiertes Ziel zeigt. Je näher er am Objekt ist, desto größere Winkelabweichungen sind möglich. b) Der Winkel des Anwenders zum Objekt beeinflusst die Abweichung der Zeigerichtung, die möglich ist, sodass der Anwender immer noch auf das anvisierte Ziel zeigt. Je direkter er vor dem Objekt steht, desto größere Abweichungen der Zeigerichtung sind möglich.

Fazit

Die genannten Faktoren, die sich auf die Interaktion mit dem 3D Pointing Toolkit auswirken, können durch die zwei genannten Lösungsansätze zwar nicht beseitigt aber abgeschwächt werden. In dem Fall des vorgestellten Pointing Toolkits wurde eine Vergrößerung der virtuellen Objekte um einen festen Wert von 15cm an jeder Seite gewählt. Das bedeutet für das größte Objekt eine prozentuale Steigerung der Seitenlängen um 10% und beim kleinsten Objekt eine Steigerung um 100%. Es wurde eine feste Größe von 15cm gewählt, da es einem Anwender immer leichter fällt ein großes Objekt auszuwählen und anzuvisieren. Problematisch hingegen sind die kleinen Objekte, da die oben genannten Faktoren besonders bei kleinen Objekten eine große Rolle spielen. Somit ist es sinnvoll dem Anwender den Umgang mit kleinen Gegenständen zu erleichtern, wohingegen bei den größeren nur eine kleine Anpassung vorgenommen werden muss. Um dem Benutzer die Interaktion mit dem Toolkit noch weiter zu erleichtern, kann zusätzlich der zweite Lösungsansatz eingesetzt werden, da er ihm eine Hilfestellung bei der Auge-Hand-Koordination geben kann. Mit dem gegebenen Feedback können mögliche Fehleinschätzungen korrigiert und an die realen Gegebenheiten angepasst werden.

5. Zusammenfassung und Ausblick

Das Resümee aus der vorgestellten Arbeit, ist die Erkenntnis, dass einerseits Trackingsysteme nicht aus unserer Umwelt und auch Forschung wegzudenken sind und andererseits Anwendungen, die die von Trackingsystemen gewonnen Daten auswerten, immer wichtiger werden. Denn schon bald werden die kommenden Trackingtechnologien Einzug in unser alltägliches Leben halten. Die anfangs vorgestellten Trackingtechnologien, die sich extrem in ihren Funktionsweisen unterscheiden, haben alle ihre Vor- und Nachteile, die sie nur für spezielle Einsatzgebiete verwendbar machen. Hier kann die Forschung jedoch ansetzen und hybride Technologien entwickeln, die die Nachteile einzelner Systeme reduzieren und stattdessen die jeweiligen Vorteile mehr zur Geltung bringen. Es gibt schon einige Trackingsysteme, die mehrere Technologien verbinden und damit die Stärken der einzelnen Systeme hervorheben. Damit werden neue Möglichkeiten im Bereich des Tracking geschaffen, die in machen Einsatzgebieten dringend benötigt werden. Beispiele für solche Hybride sind Trackingsysteme, die eine Technologie mit dem inertialen Tracking kombinieren. Dies bietet sich an, da inertiales Tracking eine hohe Genauigkeit und Wiederholungsrate liefert, es jedoch ab und zu eine Rekalibrierung benötigt, damit sich keine großen Fehler zusammenaddieren [BWA01]. In diesem Gebiet kann jedoch noch viel Forschung betrieben werden, um Systeme zu schaffen, die klein und handlich sind und die gleichzeitig hohe Wiederholungsraten und Genauigkeiten bieten.

In dieser Arbeit wurden auch Interaktionsmöglichkeiten vorgestellt, die es mit Hilfe von einem optischem Trackingsystem und Motion Capturing ermöglichen mit einer virtuellen Umgebung zu interagieren. Wie in der Arbeit von Shizuki et al. [SHTT06] vorgestellt, gibt es schon Ansätze, die versuchen einem Anwender eine möglichst intuitive und trotzdem mächtige Interaktionstechnik an die Hand zu geben, um ein Computersystem über Laserpointer-Eingaben zu steuern. Bei Eingaben über Laserpointer oder das Hand-Raycasting, wie sie auch bei dem vorgestellten Toolkit verwendet wurden, entstehen jedoch Unzulänglichkeiten, die durch die Natur der Eingabetechnologien hervorgerufen werden. Hierbei handelt es sich einerseits um das natürliche „Zittern“ der menschlichen Hand, was sich durch spezielle Algorithmen teilweise beheben lässt. Das „Zittern“ ist bei Aufgaben, die eine hohe Präzision benötigen jedoch trotzdem noch zu spüren. Andererseits geht es um den Mangel an Eingabemöglichkeiten, wie z.B. Buttons, die die Interaktion mit komplexen Menüs sehr aufwendig machen. So sind die beiden Interaktionsmöglichkeiten zwar perfekt dafür geeignet mit sehr großen Flächen zu interagieren, wenn man jedoch im Zentimeterbereich arbeiten muss, entstehen Fehler, die manchmal nicht hinnehmbar sind. Wenn zusätzlich noch komplexe Anwendungsmenüs hinzukommen, dann muss sich der Entwickler sehr viele Gedanken darüber machen, wie er Eingabemöglichkeiten, wie z.B. Buttons oder Gesten, dazu verwendet dem Anwender eine angenehme Interaktion zu ermöglichen. Er muss dann

aber auch darauf achten den Anwender nicht durch zu aufwendige Gesten zu überfordern. Diese Interaktiontechniken müssen von Entwicklern noch weiter verbessert und so gestaltet werden, dass sich eine breite Masse der Bevölkerung damit zurechtfinden kann. Denn durch die vorgestellte Idee der „Smart Homes“ werden sich in der Zukunft immer mehr Menschen mit diesen Interaktionstechnologien beschäftigen. Wenn noch zusätzlich die Trackingsysteme immer kleiner und erschwinglicher werden, wird jeder durchschnittliche Haushalt die nötige Hardware besitzen, solche Interaktionen auch tatsächlich nutzen zu können. Ein perfektes Beispiel hierfür ist der erst kürzlich vorgestellte Kinect 2 Sensor, der in Verbindung mit der XBOX One präsentiert wurde. Der Sensor kann Personen, deren Gesten, Handbewegungen und Rotationen und scheinbar sogar deren Mimik erfassen und somit eine Bandbreite von neuen Interaktionsmöglichkeiten bieten [Car13]. Der Sensor verwendet nach Angaben der Hersteller die TOF-Technik in Verbindung mit Infrarotlicht, um eine Tiefenkarte der Umgebung zu erstellen.

Ohne, wie die in dieser Arbeit vorgestellten, Toolkits sind diese technischen Neuerungen im Bereich der Trackingsysteme jedoch nutzlos. Wenn man das Beispiel der Kinect 2 nimmt, so sieht man, dass bei einer generierten dreidimensionalen Tiefenkarte eine virtuelle 3D Repräsentation des Raums benötigt wird, um weitere Berechnungen durchführen zu können. So ist zum Beispiel die Gestenerkennung darauf angewiesen, dass im Hintergrund auf den erfassten Daten Berechnungen zur Erkennung von Gestenmustern durchgeführt werden. In dieser Arbeit wurde das Konzept des Hand-Raycasting zur Interaktion mit dem System benutzt, das sehr einfach und für diesen Zweck gut geeignet ist. Diese Eingabemethode ermöglicht es dem Benutzer schnell eine Auswahl auf großem Raum zu treffen. Es gibt jedoch Defizite bei der Genauigkeit, wenn man mit sehr kleinen Objekten arbeitet. Dieses Interaktionskonzept bedarf auch mehr Forschung im Bereich der Bedienbarkeit. Hier stellen sich die Fragen, wie sich die Haltung des Arms, die individuellen Gegebenheiten des Körperbaus und der Wahrnehmung auf die Zeigegesten auswirken. Dies sind nämlich Faktoren, die von Mensch zu Mensch unterschiedlich sein können und die Interakion beeinflussen.

Das hier verwendete System-Setup wurde so gewählt, dass hohe Abtastraten und hohe Frameraten in der Verarbeitung erreicht werden konnten, womit eine kleine Latenzzeiten und hohe Wiederholungsraten erreicht werden können. Das Toolkit an sich bietet viele Möglichkeiten die virtuelle Repräsentation der Umgebung während der Laufzeit sowie auch vor dem Start zu bearbeiten und neue Objekte, die mit in die Berechnungen aufgenommen werden sollen, hinzuzufügen oder auch zu entfernen. An dieser Stelle kann noch angeknüpft werden um zusätzlich komplexere Mesh-Modelle von Objekten einzubinden, damit auch vielschichtigere Interaktionskonzepte möglich werden. Außerdem wurde ein GUI implementiert, das den Umgang mit dem erwähnten Editor erleichtern soll. Ein weiterer wichtiger Teil des vorgestellten Toolkits ist die Kommunikation der einzelnen Bestandteile. Hier wurde das erwähnte EI Toolkit eingesetzt, das es dem Entwickler leichter machen soll viele unterschiedliche Ressourcen, die mit der Anwendung kommunizieren, mit einem einheitlichen Protokoll zu verwenden. Dieser Ansatz nimmt viel Arbeit ab und gestattet es, sich auf andere Fragestellungen zu konzentrieren. Im letzten Kapitel wurde die Performance des Toolkits unter einem festgelegten Testsystem getestet und ausgewertet. Dabei wurde festgestellt, dass das Toolkit mit sehr komplexen virtuellen Szenen umgehen kann und skalierbar ist. Es wurde

außerdem die Interaktion mit dem Toolkit und dem optischen Trackingsystem bewertet, wobei festgestellt wurde, dass drei Faktoren die Interaktion stark beeinflussen. Diese drei Parameter wurden beschrieben und es wurden jeweils Lösungsansätze vorgeschlagen.

Schlussendlich ist das entwickelte Toolkit ein Werkzeug, das anderen Anwendungen verarbeitete Trackingdaten in sinnvoller Form zur Verfügung stellen soll. Somit ist es sehr vielseitig einsetzbar und die vorgestellten Use Cases sind nur eine kleine Auswahl von Einsatzmöglichkeiten des Toolkits und es können noch viele andere hinzukommen.

Ausblick

Leap Motion [LEA] und der neue Kinect 2 Sensor [Car13] sind nur zwei Beispiele, die zeigen, dass schon heute die Trackingtechnologien immer kleiner und leistungsstärker werden. Es wurde auch eine Trackingtechnologie vorgestellt, die einzig und allein auf W-LAN Signalen basiert und die keine besonderen Targets braucht, die am zu erfassenden Objekt angebracht werden müssen [QP13]. Die von Pu et al. vorgestellte Technologie basiert auf standard W-LAN Geräten, wie Smartphones oder W-LAN Router, von denen mindestens zwei benötigt werden. Die Funktionsweise baut auf einem W-LAN Gerät auf, das als Sender W-LAN Signale aussendet, die vom zu trackenden Objekt reflektiert und von einem zweiten Gerät erfasst werden. Durch die Schwankungen im W-LAN Signal können bestimmte Gesten eines Menschen erfasst und zum Bedienen von z. B. Multimediageräten verwendet werden. Diese Technologie ist ein gutes Beispiel dafür, was in der Zukunft auf uns zukommen wird und was wir für Möglichkeiten in der Mensch-Computer-Interaktion haben werden. Mit dieser Erkenntnis wird auch klar, dass immer höhere Anforderungen an die Software, die im Hintergrund die erfassten Daten auswertet, gestellt werden und dass sie eine immer wichtigere Rolle spielen werden. Da die derzeit verwendeten Interaktionskonzepte nicht ausreichen um die modernen Multimediageräte zu bedienen, wird auch die Forschung auf neue Trackingtechnologien und die dazugehörende Software angewiesen sein, um neue Erkenntnisse gewinnen und neue Konzepte vorstellen zu können.

A. Anhang

Abbildung A.6 stellt den Pseudecode des Algorithmus zur Schnittberechnung dar und **Abbildung A.5** präsentiert ein Beispiel für eine Szene die im erläuterten XML-Format gespeichert wurde. Es sind noch die **Abbildung A.3** und **Abbildung A.4** angehängt, von denen die erste das GUI im Ausgangszustand und das zweite die Console zeigen. Zur besseren Sichtbarkeit sind noch **Abbildung A.1** und **Abbildung A.2** angefügt, die die beiden vorgestellten Use Cases darstellen.

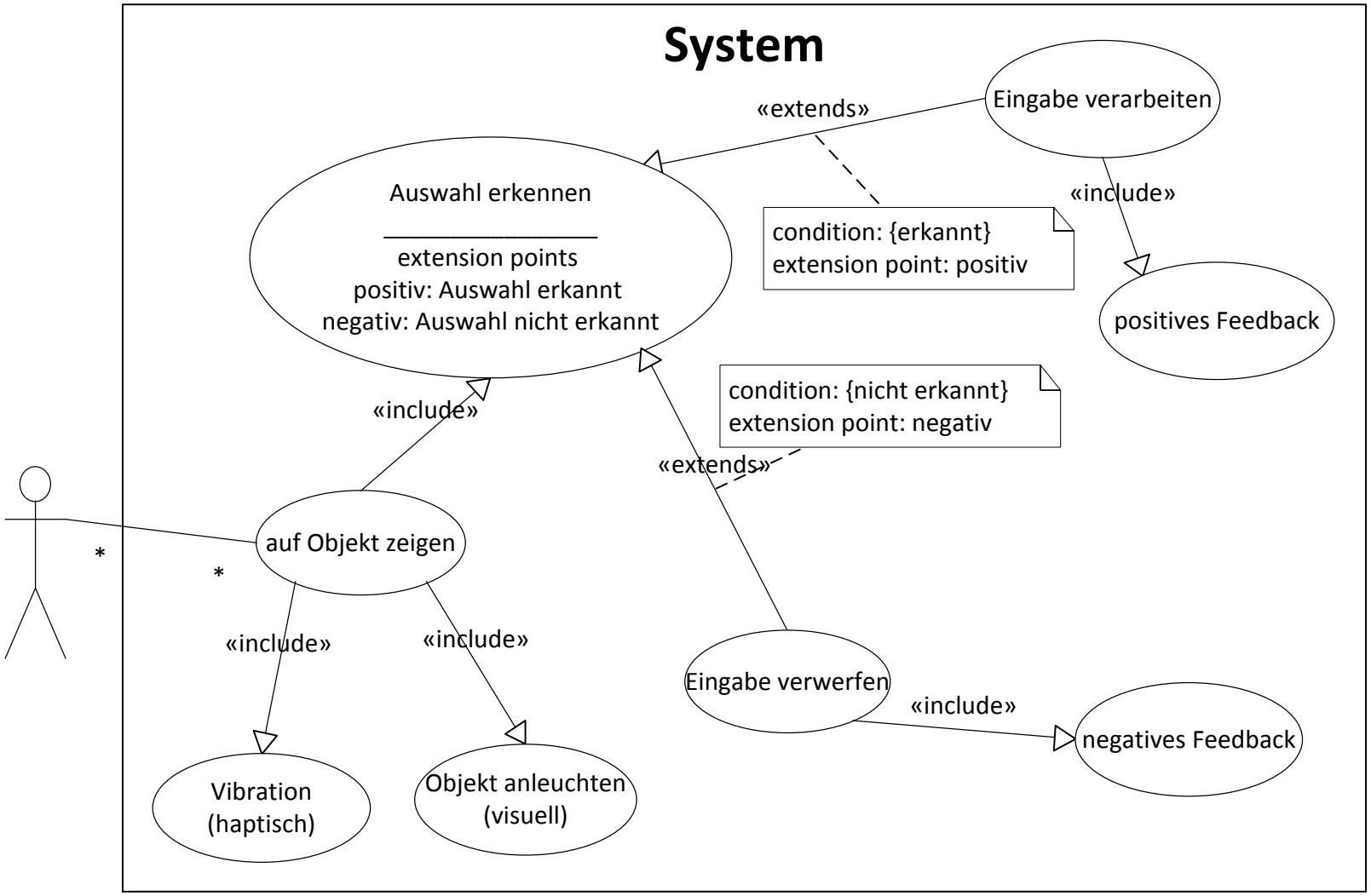

Abbildung A.1.: Use Case: Steuerung von Haushaltsgeräten

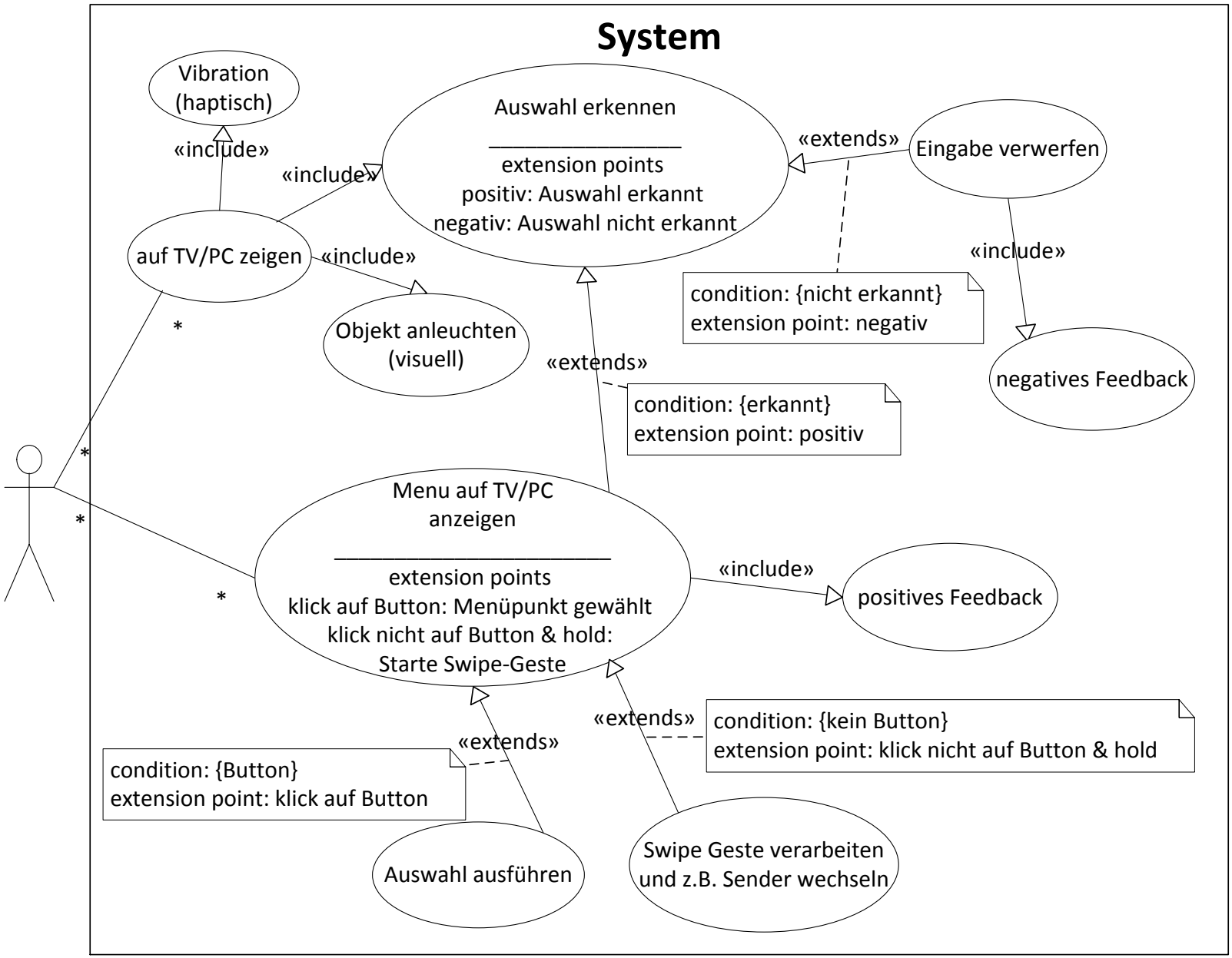

Abbildung A.2.: Use Case: TV-/PC-Steuerung

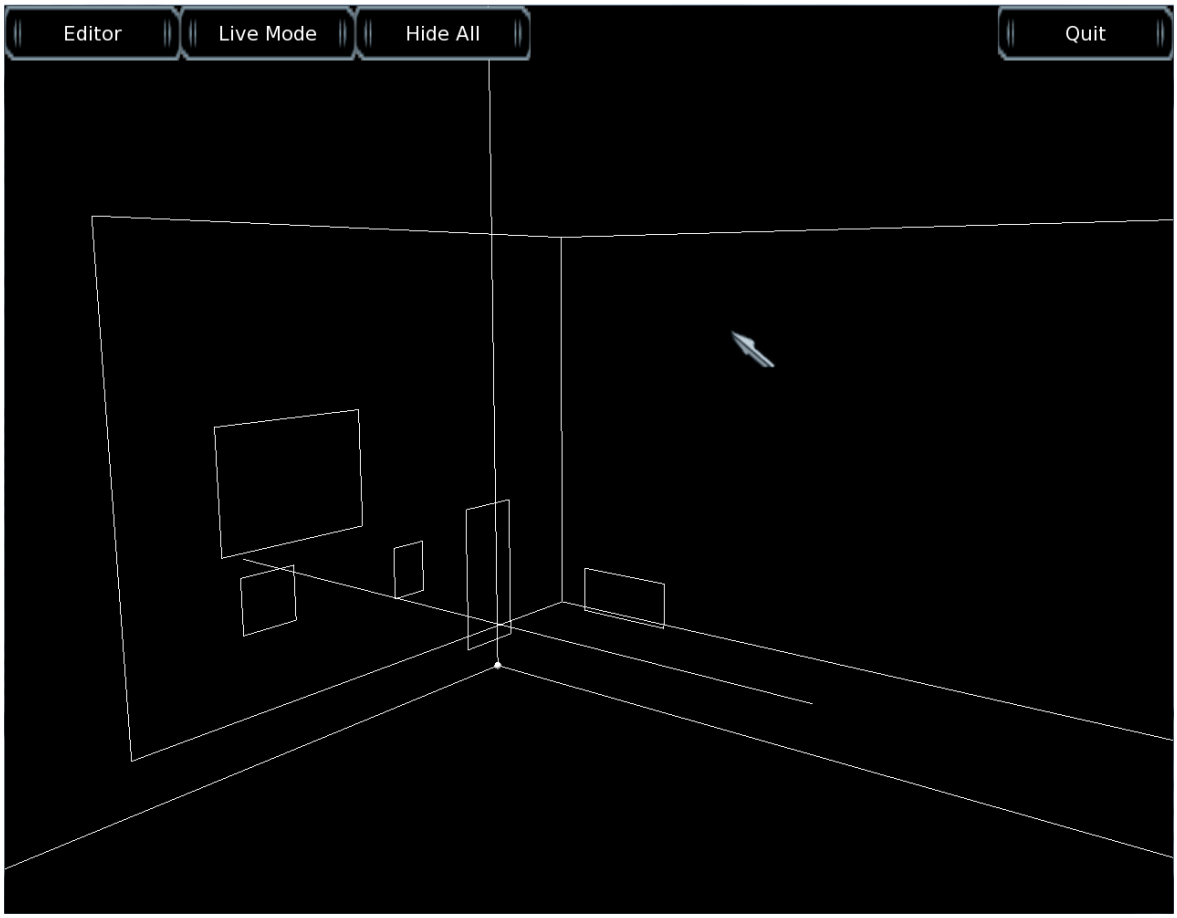

Abbildung A.3.: Das vorgestellte GUI im Ausgangszustand mit den Auswahlmöglichkeiten „Editor“, „Live Mode“, Hide All und „Quit“.

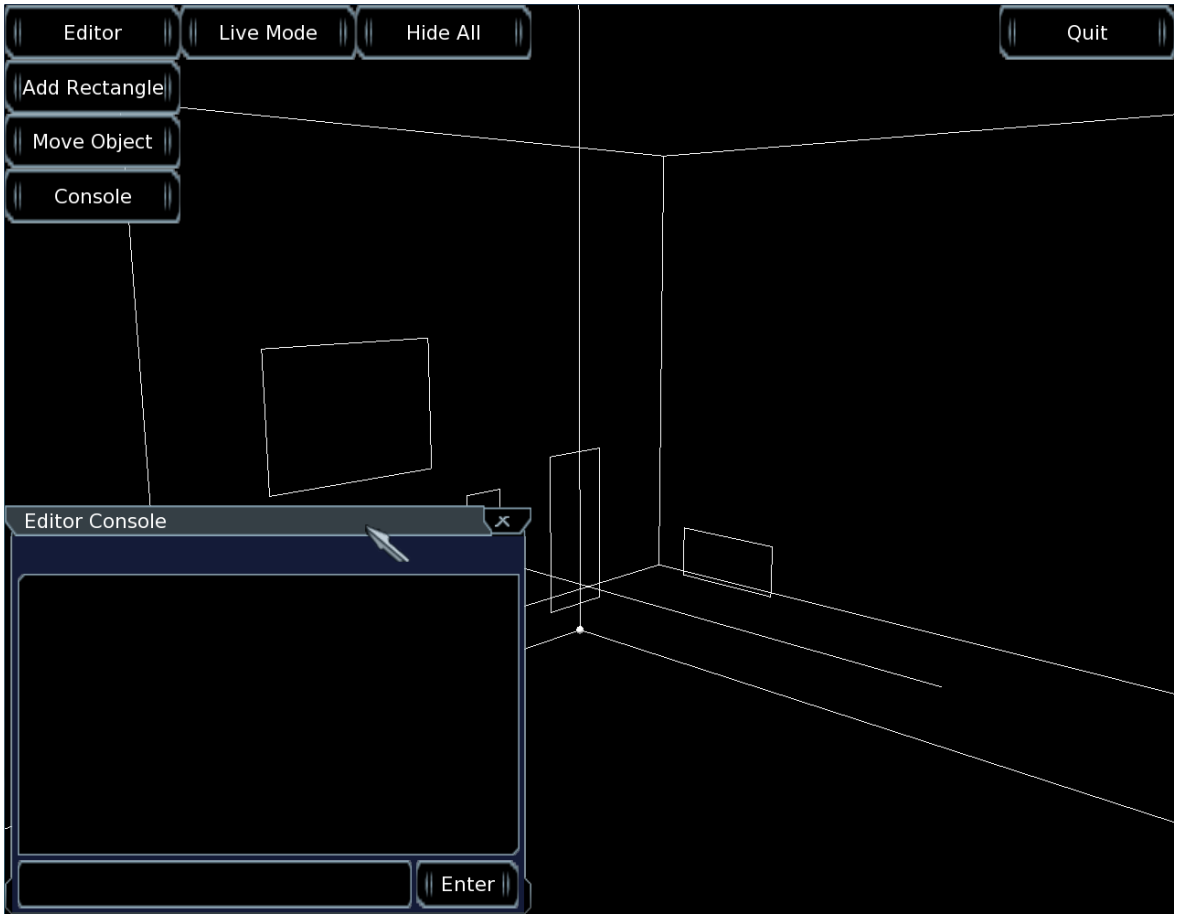

Abbildung A.4.: Das präsentierte GUI mit der Console, die es erleichtern soll neue Befehle zum System hinzuzufügen.

```
<?xml version="1.0" ?>
<SceneRep>
  <Rectangles>
    <rect name="TV1">
      <Vertex x="-95.0" y="190.0" z="130.0" />
      <Vertex x="-95.0" y="190.0" z="280.0" />
      <Vertex x="-95.0" y="290.0" z="280.0" />
      <Vertex x="-95.0" y="290.0" z="130.0" />
    </rect>
    <rect name="Tischlicht1">
      <Vertex x="-95.0" y="95.0" z="40.0" />
      <Vertex x="-95.0" y="95.0" z="55.0" />
      <Vertex x="-95.0" y="125.0" z="55.0" />
      <Vertex x="-95.0" y="125.0" z="40.0" />
    </rect>
    <rect name="Lampe1">
      <Vertex x="-95.0" y="0.0" z="-75.0" />
      <Vertex x="-95.0" y="0.0" z="-105.0" />
      <Vertex x="-95.0" y="150.0" z="-105.0" />
      <Vertex x="-95.0" y="150.0" z="-75.0" />
    </rect>
    <rect name="Heizung1">
      <Vertex x="-45.0" y="10.0" z="-245.0" />
      <Vertex x="55.0" y="10.0" z="-245.0" />
      <Vertex x="55.0" y="40.0" z="-245.0" />
      <Vertex x="-45.0" y="40.0" z="-245.0" />
    </rect>
    <rect name="Radio1">
      <Vertex x="-95.0" y="105.0" z="260.0" />
      <Vertex x="-95.0" y="105.0" z="225.0" />
      <Vertex x="-95.0" y="135.0" z="225.0" />
      <Vertex x="-95.0" y="135.0" z="260.0" />
    </rect>
  </Rectangles>
</SceneRep>
```

Abbildung A.5.: Ein Beispiel für eine XML-Scene die vom Toolkit verwendet wird

```
for (allIntersetionResults){
    //check for intersections != 0 and >=3
    //because of useless Objects "sphere" and two walls
    if (moreThanThreeIntersections){
        //only take the first intersection
        if (firstIntersection){
            //creater iterator that holds all
            //the nodes that have an attached object
            childItr = rootSceneNode->getChildIterator();
            //loop in all SceneNodes
            while (thereAreMoreSceneNodes && loopNode){
                childItr.getNext();
                //get iterator for all objects
                //that are attached to SceneNode
                attachedObjects = currentSceneNode
                                ->getAttachedObjectIterator();
                //reject all intersections with needless objects
                if (!needleesObject){
                    while (thereAreMoreAttachedObjects && loopObject){
                        attachedObjects.getNext();
                        //check if AttachedObject is same as "intersectionObject"
                        if (IntersectionObject == attachedObject){
                            //end loops since correct object of scene was found
                            loopNode = false;
                            loopObject = false;
                            //check for cases:
                            //  1) newObject intersected
                            //  2) intersection still inside of oldObject
                            //1)
                            if (intersectionWithNewObject){
                                calculateIntersectionPoint();
                                sendIntersectionPoint();
                            //2)
                            } else if (intersectionInOldObject){
                                calculateRelativeIntersection();
                                sendIntersectionPoint();
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Abbildung A.6.: Algorithmus zur Schnittberechnung Ray-SceneObject

Literaturverzeichnis

[BE79] B. I. C. L. V. V. Bates E., Benigni L. The emergence of symbols: Cognition and communication in infancy, 1979. (Zitiert auf Seite 32)

[BWA01] G. Bishop, G. Welch, B. D. Allen. Tracking: Beyond 15 minutes of thought. *SIGGRAPH Course Pack*, 2001. (Zitiert auf den Seiten 6, 11, 12 und 49)

[Car13] D. Cardinal. Kinect for the Xbox One: Sensor revolution or marketing hype? Technischer Bericht, Ziff Davis, Inc, 2013. URL http://www.extremetech.com/gaming/156436-kinect-for-the-xbox-one-sensor-revolution-or-marketing-hype. (Zitiert auf den Seiten 50 und 51)

[CB03] X. Cao, R. Balakrishnan. VisionWand: interaction techniques for large displays using a passive wand tracked in 3D. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, UIST '03, S. 173–182. ACM, New York, NY, USA, 2003. doi:10.1145/964696.964716. URL http://doi.acm.org/10.1145/964696.964716. (Zitiert auf Seite 20)

[DOL] URL http://en.wikipedia.org/wiki/File:Dolphin_triangle_mesh.png. (Zitiert auf den Seiten 6 und 26)

[Gro11] F. Groß. *Die gestische Entwicklung praeverbaler Kleinkinder unter dem Einfluss des Babysigning und dem Erwerb einer Gebaerdensprache*. Diplomarbeit, Universitaet Trier, 2011. (Zitiert auf Seite 32)

[Hol05] P. Holleis. EI Toolkit, 2005. URL https://wiki.medien.ifi.lmu.de/HCILab/EiToolkit. (Zitiert auf den Seiten 7, 28, 35 und 38)

[KC95] J. A. Kin, K. Choi. Ray Tracing Triangular Meshes. Technischer Bericht, In Western Computer Graphics Symposium, 1995. (Zitiert auf Seite 26)

[KM98] C. Kirstein, H. Mueller. Interaction with a Projection Screen Using a Camera-Tracked Laser Pointer. In *PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MULTIMEDIA MODELING. IEEE COMPUTER*, S. 191–192. Society Press, 1998. (Zitiert auf Seite 20)

[Kru91] M. Krueger. Videoplace and the interface of the future. 1991. (Zitiert auf Seite 21)

[LEA] Leap Motion. URL https://www.leapmotion.com/product. (Zitiert auf den Seiten 21 und 51)

[MES13] Polygon mesh, 2013. URL http://en.wikipedia.org/wiki/Polygon_mesh. (Zitiert auf den Seiten 6, 23, 24 und 25)

[NAT13] NatNet SDK, 2013. URL http://www.naturalpoint.com/optitrack/products/natnet-sdk/. (Zitiert auf den Seiten 7, 35 und 37)

[OGR09] Ogre 3D, 2000-2009. URL http://www.ogre3d.org/. (Zitiert auf Seite 38)

[ON01] D. R. Olsen, Jr., T. Nielsen. Laser pointer interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, S. 17–22. ACM, New York, NY, USA, 2001. doi:10.1145/365024.365030. URL http://doi.acm.org/10.1145/365024.365030. (Zitiert auf Seite 20)

[OS02] J.-Y. Oh, W. Stuerzlinger. Laser Pointers as Collaborative Pointing Devices, 2002. (Zitiert auf Seite 20)

[PE05] C. M. Pizzuto E. *The link and differences between deixis and symbols in children's early gestural-vocal system*, S. 179–199. 2005. (Zitiert auf Seite 32)

[PK07] T. Pintaric, H. Kaufmann. Affordable Infrared-Optical Pose Tracking for Virtual and Augmented Reality. In G. Zachmann, Herausgeber, *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, S. 44–51. Shaker Verlag, Aachen, 2007. URL http://publik.tuwien.ac.at/files/pub-inf_5236.pdf. Vortrag: IEEE Virtual Reality 2007, Charlotte, NC (USA); 2007-03-14 – 2007-03-17. (Zitiert auf den Seiten 16 und 18)

[QP13] S. G. S. P. Qifan Pu, Sidhant Gupta. Whole-Home Gesture Recognition Using-Wireless Signals, 2013. URL http://wisee.cs.washington.edu/. (Zitiert auf Seite 51)

[RBG01] J. P. Rolland, Y. Baillot, A. A. Goon. A SURVEY OF TRACKING TECHNOLOGY FOR VIRTUAL ENVIRONMENTS, 2001. (Zitiert auf Seite 11)

[SEN] URL http://www.naturalpoint.com/optitrack/products/v100-r2/buy.html. (Zitiert auf den Seiten 6 und 17)

[SHTT06] B. Shizuki, T. Hisamatsu, S. Takahashi, J. Tanaka. Laser pointer interaction techniques using peripheral areas of screens. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, S. 95–98. ACM, New York, NY, USA, 2006. doi:10.1145/1133265.1133284. URL http://doi.acm.org/10.1145/1133265.1133284. (Zitiert auf den Seiten 20 und 49)

[Smi06] C. Smith. *On vertex-vertex systems and their use in geometric and biological modelling*. Dissertation, Calgary, Alta., Canada, Canada, 2006. AAINR19574. (Zitiert auf Seite 22)

[STR] URL http://www.ogre3d.org/docs/manual/manual_4.html#The-Core-Objects. (Zitiert auf den Seiten 7 und 39)

[VB05] D. Vogel, R. Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, S. 33–42. ACM, New York, NY, USA, 2005. doi:10.1145/1095034.1095041. URL http://doi.acm.org/10.1145/1095034.1095041. (Zitiert auf Seite 21)

[WF02] G. Welch, E. Foxlin. Motion tracking: no silver bullet, but a respectable arsenal. *Computer Graphics and Applications, IEEE*, 22(6):24–38, 2002. doi:10.1109/MCG.2002.1046626. (Zitiert auf Seite 15)

Alle URLs wurden zuletzt am 30. 06. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift