UNIVERSITY OF APPLIED SCIENCES

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Cornelius Malerczyk (Hrsg.)

Neue Anwendungen der Mensch-Computer-Schnittstelle

Friedberger Hochschulschriften Band 34

Cornelius Malerczyk (Hrsg.)

Neue Anwendungen der Mensch-Computer-Schnittstelle

Friedberger Hochschulschriften Band 34

Friedberger Hochschulschriften Band 34

© 2013 Cornelius Malerczyk und die Autoren, Friedberg

Herausgeber der Friedberger Hochschulschriften:
Die Dekaninnen und Dekane des Campus Friedberg
der Technischen Hochschule Mittelhessen

Alle Rechte vorbehalten, Nachdruck, auch auszugsweise, nur mit schriftlicher Genehmigung und Quellenangabe.

Einzelne Hochschulschriften auch online abrufbar:
www.thm.de/bibliothek/hochschulschriften

ISSN 1439-1112

VORWORT

Seit einiger Zeit rückt die Schnittstelle zwischen Mensch und Computer wieder deutlich stärker in den Fokus von Forschung und Entwicklung. Das liegt zu einem großen Teil an der Verbreitung von Smartphones, die heute mit über 30 Millionen Geräten bereits einen Großteil der Mobiltelefone in Deutschland ausmachen. Immer öfter werden Funktionalitäten vom klassischen PC auf das mobile Kleinstgerät übertragen. Kalenderfunktionen, Emails, Routenplanung oder das Surfen im Internet - immer mehr Anwendungen werden zunehmenden mobil und unterwegs ausgeführt. Und mit den Smartphones kommen insbesondere auch neue Herausforderungen an die Bedienung der Geräte. Als neues Eingabemedium hat sich bereits die Touch-Technologie, also die Bedienung eines Gerätes mit kombinierter Ein- und Ausgabeeinheit durch Fingergesten, durchgesetzt. Das Smartphone ist zwar vielleicht das am weitesten verbreitete und damit auch das prominenteste Gerät, das von neuen Technologien und Ideen der Schnittstelle zwischen Mensch und Computer profitiert, die Liste an Hardware und Software-Anwendungen, die sich von den klassischen Eingabemedien wie Computermaus und Tastatur löst, wird aber jährlich länger. Spielekonsolen, die sich bereits in vielen Haushalten zu finden, sind ein weiteres Beispiel. Nintendos Wii mit der Fernsteuerung Wiimote oder die Microsoft Kinect als Steuereinheit für die Spielekonsole Xbox 360 zeigen, dass mit geeigneter Kombination der Vielzahl an heute preisgünstig verfügbaren Sensoren Eingabemedien geschaffen werden können, die völlig neue Wege in der Bedienung eines technischen Systems ermöglichen. Insbesondere die Tatsache, dass recht schnell nach der Markteinführung der Kinect der Hersteller Microsoft bereits eine Programmierschnittstelle kostenfrei zur Verfügung gestellt hat, lässt die Anzahl an neuen Anwendungen und Ideen zu und macht die Technologie auch für Forschungsanwendungen interessant.

Dem Trend, die neuen technologischen Möglichkeiten für die Schnittstelle zwischen Mensch und Maschine zu nutzen, trägt auch der Friedberger Masterstudiengang Medieninformatik Rechnung. Mit der Reakkreditierung des Studiengangs 2010 wurde das Wahlpflichtfach “Fortgeschrittene Mensch-Maschine-Interaktion” zu einem Pflichtfach umgewandelt und ermöglicht es den Studierenden, sich hier intensiv mit den Neuerungen und Herausforderungen in Theorie und Praxis auf diesem Wissenschaftsgebiet auseinanderzusetzen. In den beiden letzten Veranstaltungssemestern wurde die Lehrveranstaltung durch Team-orientierte Projektarbeiten mit der Ausarbeitung von Publikationen, wie sie für wissenschaftliche Konferenzen typisch sind, geprägt. Die Ergebnisse wurden am Ende des jeweiligen Semesters in Form einer kleinen öffentlichen Konferenz zum Thema “Neues aus der Fortgeschrittenen Mensch-Maschine-Interaktion” dem interessierten Publikum vorgestellt. Ausgewählte Arbeiten der beiden Veranstaltung aus den vorangegangenen Sommersemestern werden nun hier veröffentlicht. Dieser Band der Friedberger Hochschulschriften umfasst somit sechs einzelne Publikationen von insgesamt elf Autoren, die sich mit den unterschiedlichsten Themengebieten und Anwendungsmöglichkeiten beschäftigt haben. Zwei Publikationen verwenden die Microsoft Kinect und Wiimote als Eingabegerät zur Steuerung einer Anwendung. Andere beschäftigen sich mit neuen Anwendungen für das klassische Smartphone mit seiner Möglichkeit, unterschiedlichste Sensoren zu verwenden oder nutzen eine einfache Webcam zur Steuerung eines 3D-Modellierungs-Werkzeugs.

Mir als Dozenten der Veranstaltung bleibt zu wünschen, dass auch in den kommenden Semestern ähnlich interessante und hochwertige Entwicklungen und damit auch Publikationen entstehen, die dann vielleicht in einem weiteren Band veröffentlicht werden können.

Prof. Dr. Cornelius Malerczyk

INHALT

Arlt, Hans Christian und Krause, Kai …..... 07
Entwicklung und Analyse von Ego- und Thirdperson Steuerungsmethoden eines 3D Charakter mit Hilfe der Kinect Hardware und Unity3d

Becker, Dennis und Rühl, Benjamin …..... 11
Analyse der Fernsehsteuerung mit Hilfe eines Smartphones zur Ablösung der klassischen Fernbedienung

Beckmann, Daniel …..... 15
Entwicklung hybrider Steuerungskonzepte für First-Person-Shooter unter Verwendung von Kinect mit Wiimote- und Nunchuk-Controller

Ceh, Claudia, Kilb, Andreas und Brahms, Sebastian …..... 19
Entwicklung eines mobilen Rauminformations-Systems auf Basis von QR-Codes

Fiedler, Andreas und Krause, Lars-Olof …..... 23
Evaluation and Analysis of Natural Gestures for Multi-Touch Enables Geographic Information Systems

Salziger, Peter …..... 27
Einsatz von Head Tracking in Autodesk Maya zur erweiterten 3D-Wahrnehmung

ENTWICKLUNG UND ANALYSE VON EGO- UND THIRDPERSON STEUERUNGSMETHODEN EINES 3D CHARAKTERS MIT HILFE DER KINECT HARDWARE UND UNITY3D

Hans Christian Arlt, Kai Krause

Technische Hochschule Mittelhessen
Wilhelm-Leuschner-Str. 13
61169 Friedberg

KURZFASSUNG

Ziel dieser Arbeit ist die Entwicklung von Steuerungsansätzen für einen 3D Charakter mit Hilfe der *Kinect Hardware* und *Unity3D*. Hierbei sollen die einzelnen Ansätze jeweils für die *Ego*- bzw. die *Thirdperson*-Perspektive betrachtet werden. Der Schwerpunkt liegt in der Erkennung und Übertragung der Bewegungsabläufe (z.B. das Laufen, Drehen und Springen). Eine der großen Herausforderungen ist die Erkennung der Drehung des Charakters im 3D-Raum. Eine weitere wichtige Fragestellung beschäftigt sich damit, ob es sinnvoll ist einzelne Bewegungen von Gliedmaßen auf den Spielcharakter in der *Thirdperson*-Perspektive zu übertragen.

Index Terms— CG, Spiele Design, Virtueller Charakter, 3D, Echtzeit Rendering, Kinect Sensor

1. EINLEITUNG

In traditionellen *Ego*- und *Thirdperson Shootern* kommen zur Steuerung des Spielcharakters entweder Tastatur und Maus oder Gamepads zum Einsatz. Hierbei erfolgt die Steuerung durch das Drücken von Knöpfen oder bestenfalls durch die Erkennung der Neigungen des Gamepads. Moderne Spielsteuerungen, wie die im November 2010 von Microsoft veröffentlichte *Kinect*, erlauben hingegen eine direkte Kontrolle des Spielcharakters durch Körperbewegungen. Hierbei wird der Körper des Spielers mit Hilfe der Bilderkennung analysiert und als einfaches Skelett in einer Tiefenkarte räumlich abgebildet. Alle Bewegungen des Spielers können so eins-zu-eins auf den Spielcharakter übertragen werden. Die *Kinect Hardware* erkennt den Spieler optimal, wenn dieser sich im Bereich von 1,2m bis 3,5m (vgl. Abb. 1) vor den Sensoren befindet.[1] Der Bewegungsspielraum des Spielers wird dadurch auf eine Fläche von etwa $4m^2$ begrenzt, wodurch seine translativen Bewegungen stark eingeschränkt sind. Ein weiteres Problem entsteht durch die Drehung des Spielcharakters. Soll dieser sich um $180°$ drehen, muss sich der Spieler ebenfalls um $180°$ und würde sich dadurch vom Bildschirm abwenden. Um den Spielcharakter dennoch uneingeschränkt im virtuellen Raum bewegen zu können, muss ein Verfahren entwickelt werden, dass die translativen Bewegungen und die Drehung um die Achse des Spielcharakters von den Bewegungen des Spielers entkoppelt. Hierfür gibt es unterschiedliche Ansätze, die in diesem Artikel näher erläutert werden sollen.

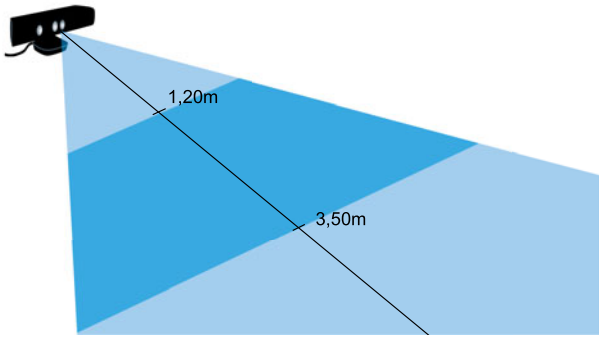


Abbildung 1: Erkennungsbereich der *Kinect* - Die beste Erkennung des Spielers erfolgt in einem Abstand von ca. 1,20 m bis 3,50 m. Wie die Zeichnung veranschaulicht, ergibt sich so auch eine Beschränkung an den Seiten.

2. STAND DER TECHNIK

2.1. Kinect Sports

Das Xbox 360 Spiel *Kinect Sports* ist eine Sammlung von kleineren Sportspielen, die mit Hilfe von Körperbewegungen gesteuert werden. Eines dieser Spiele heißt “110 Meter Hürdenlauf”. Hierbei muss der Spieler auf der Stelle rennen, um seinen Spielcharakter entsprechend vorwärts laufen zu lassen und springen, wenn der Charakter eine Hürde überwinden soll. Die Laufrichtung des Spielcharakters ist jedoch durch die Spur auf der Aschenbahn fest vorgegeben. Der Spieler beeinflusst daher lediglich die Vorwärtsbewegung und deren Geschwindigkeit, hat jedoch keinen Einfluss auf die Laufrichtung des Spielcharakters.

Abbildung 2: Kinect Sports - Spielen mit vollem Körpereinsatz. Der Spieler muss springen und laufen, um das Spielziel zu erreichen.[2]

2.2. Ghost Recon : Future Soldier

Ein weiteres Spiel für die *Xbox Kinect* ist der bisher noch nicht veröffentlichte *Egoshooter* "Ghost Recon : Future Soldier" von Ubisoft. Hierbei wird mit Hilfe von Körperbewegungen das Menü gesteuert. So lassen sich beispielsweise mit einfachen Gesten die Waffen des Spielcharakters modifizieren. In einem Schießstand können diese Waffen dann, ebenfalls mit Hilfe von Gesten, testweise abgefeuert werden. Ob und wie sich der Spielcharakter mit Hilfe der *Kinect*-Steuerung durch die 3D-Szenerie bewegen lässt, wurde von Ubisoft bisher nicht bekannt gegeben.

Abbildung 3: Ghost Recon : Future Soldier - Durch Fingererkennung ist der Spieler in der Lage Aktionen auszuführen oder wie hier seine Waffe zu modifizieren.[3]

3. KONZEPTION UND ANSÄTZE

Zur Ansteuerung eines 3D Charakters in einer 3D Szene (3D Welt, Labyrinth etc.) mit Hilfe der *Microsoft Xbox Kinect* gibt es verschiedene Ansätze. Es muss überlegt werden, ob bestimmte Gesten des Spielers als Auslöser für Aktionen des Charakters genutzt werden, oder ob die gesamten Bewegungen des Spielers auf den Charakter übertragen werden sollen. Es soll herausgefunden werden, welche der verschiedenen Ansätze eher für die *Ego-* und welche für die *Thirdperson-Perspektive* geeignet sind. Weiterhin soll geklärt werden, worin die Schwachstellen der verschiedenen Vorgehensweisen liegen.

3.1. Mouse- and Keyboardmapping

Der erste Lösungsansatz orientiert sich an einer Spielsteuerung mit Tastatur und Maus. Die Fußstellungen des Spielers simulieren das Drücken der Steuerungstasten "A,W,S,D" bzw. der Pfeiltasten (vgl. Abb. 4). Durch Setzten von Schranken können durch den Abstand der Füße verschiedenen Aktionen ausgeführt werden. Macht der Spieler beispielsweise einen Schritt nach vorne, und überschreitet dabei die Schranke, wird dem Charakter der Befehl zum Vorwärtslaufen gegeben. Das gleiche Verhalten wäre für eine Seitwärts- bzw. Rückwärtsbewegung denkbar. Zur Ansteuerung der Kamera und somit der Veränderung des Sichtfeldes und der Fortbewegungsrichtung, soll die rechte Hand genutzt werden. Durch vertikale und horizontale Bewegungen kann so der Blickwinkel auf jeder Achse um bis zu +/-90° geändert werden (vgl. Abb. 5, links). Mit Hilfe des Höhenwerts der Spielerhüfte, können Zusatzbewegungen wie Springen oder Ducken umgesetzt werden. Verändert sich dieser Wert in positiver Richtung, soll der Charakter im Spiel springen. Verändert sich der Wert in negativer Richtung, soll der Charakter sich ducken. Weitere Spielfunktionen, wie das Schießen mit einer Waffe oder das Nachladen der selben, könnten durch Gesten mit der linken Hand umgesetzt werden. Hierbei bietet es sich an, die Position auf der Z-Achse zu nutzen. Wenn der Spieler seinen linken Arm ausstreckt, wird ein Schuss abgefeuert. Das Nachladen könnte durch eine nach hinten ziehende Bewegung ausgelöst werden.

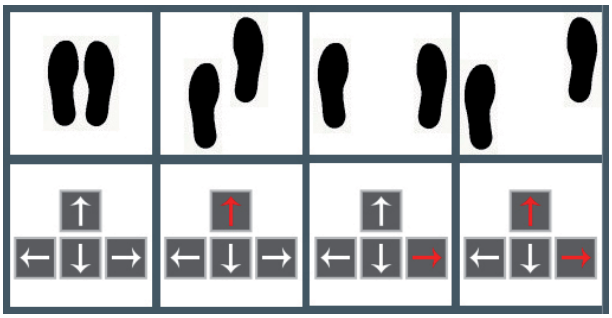


Abbildung 4: Steuerungsansatz Mouse- and Keyboardmapping - Mit den Positionen der Füße wird das Laufen des Charakters gesteuert. So sind Vorwärts-, Rückwärts- sowie Seitwärtsbewegungen möglich.

3.2. Körper- und Bewegungsmapping

Der zweite Lösungsansatz soll sich von der ursprünglichen Ansteuerung von Computerspielen abheben. Hierbei steht die Bewegungsübertragung des Spielers auf den Charakter im Vordergrund. Hebt der Spieler einen Arm, hebt der Spielcharakter ihn auch. Läuft der Spieler auf der Stelle, läuft auch der Charakter und bekommt über die Geschwindigkeit der Fußbewegung die Bewegungsgeschwindigkeit mitgeteilt (vgl. Abb. 5, rechts). Hierbei könnte die Laufrichtung durch die Drehung des Oberkörpers des Spielers realisiert werden. Auch hier ist die Drehmöglichkeit auf ca. +/- 90° beschränkt (vgl. Abbildung 5, rechts). Der Spieler kann sich zwar um 180° drehen, und der Charakter tut es in der Szene ebenfalls, in der Regel sieht der Spieler dann aber den Bildschirm nicht mehr. Um mit diesem Ansatz ein uneingeschränktes Laufen zu ermöglichen, wären mehrere Bildausgabegeräte erforderlich. Die Aktionen Springen und Ducken könnten wie beim *Mouse- und Keyboardmapping* durch den Höhenwert der Hüfte realisiert werden. Das Auslösen eines Schusses könnte wie in der ersten Variante beschrieben umgesetzt werden (vgl. Abs. 3.1). Beim Nachladen könnte es zu Problemen kommen, da der Spieler beim Laufen gegebenenfalls die Arme zu stark bewegt. In diesem Fall ist die Verwendung der *Kinect* Spracherkennung vor zu ziehen, um die Aktion über einen akustischen Befehl auszulösen.[4]

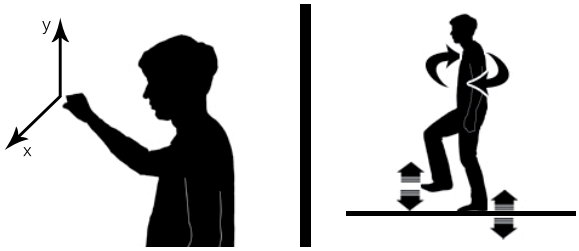


Abbildung 5: Steuerungsansätze für die Kamera und Richtungswechsel des Spielers

4. IMPLEMENTIERUNG

Ursprünglich wurde die *Kinect Hardware* für die Spielekonsole Xbox 360 von Microsoft konzipiert. Schon nach kurzer Zeit entwickelten findige Entwickler eigene Treibersoftware für die *Kinect*. Mittlerweile existieren Treiber für Windows, Linux und Mac OS. Sie wurden größtenteils unter einer *Open Source* Lizenz veröffentlicht. Hierdurch ist es möglich die Microsoft *Kinect* lizenzfrei in selbst geschriebenen Anwendungen zu verwenden.[5] Für dieses Projekt wurden die folgenden Treiber und Komponenten genutzt:

- OpenNI Binaries[6]
- OpenNI Compliant Middleware Binaries[6]
- OpenNI Compliant Hardware Binaries[6]
- SensorKinect (Avin)[7]

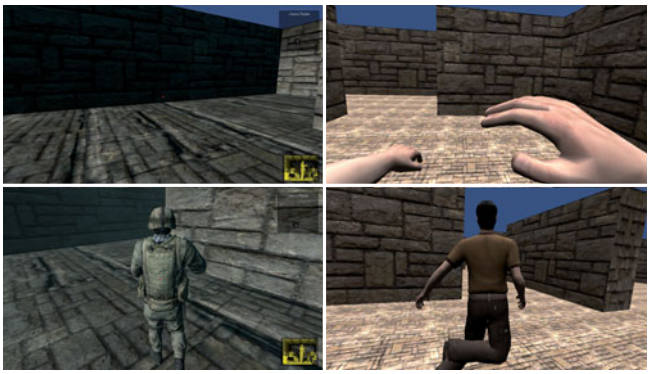


Abbildung 6: Mouse- und Keyboardmapping in Egoperspektive (o. l.) und Thirdperson-Perspektive (u. l.) mit Userradar und Depthmapviewer für besseres Userfeedback. Die Laufbewegungen entstehen durch vorgefertigte Animationen. Körper- und Bewegungsmapping. Die Bewegungen des Spielers werden hierbei eins-zu-eins auf den Spielcharakter übertragen. Hierbei können Fehlstellungen entstehen, die jedoch nur in der Thirdperson-Perspektive (u. r.) sichtbar werden.

Die *Middleware* stellt Bibliotheken zur Verfügung, die das *Usertracken*, also das Erkennen und Verfolgen von Personen über die Tiefenbildkamera, übernehmen. Weiterhin enthält die *Middleware* Methoden zur Skelettberechnung, wenn sich der Spieler kalibriert. Diese Bibliotheken können genutzt werden, um die *Kinect* aus einer Anwendung heraus anzusteuern und auszulesen. Der Entwickler Amir Hirsch schrieb Skripte für *Unity3D* und stellte diese den Entwicklern in einem Testprojekt zur Verfügung[8]. Mittlerweile wurde dieses Projekt verfeinert, optimiert und wird nun standardmäßig mit der *Middelware* ausgeliefert. Dieses Beispielprojekt stellt den *Unity3D* Entwicklern die Schnittstellen der *Kinect* zur Verfügung. In der aktuellen Version beinhaltet es auch praktische Skripte, die ein bessere Nutzerrückmeldungen erlauben. Als Beispiel ist das *Userradar* zu nennen. Es zeigt dem Spieler an, wo er sich gerade im Erkennungsbereich der *Kinect* befindet (vgl. Abb. 6). Der Hauptunterschied zwischen *Ego-* und *Thirdperson-Perspektive* ist die Position der Kamera. Bei der Egoperspektive sitzt die Kamera an der Position des Charakterkopfes, sodass der Charakter gar nicht oder nur zum Teil sichtbar ist (vgl. Abb. 6 o.). Bei der *Thirdperson-*Perspektive wird die Kamera hinter den Charakter platziert, sodass dieser vollständig zu sehen ist. Die Umsetzung der einzelnen Varianten ist grundsätzlich identisch, da in beiden Fällen das im Testprojekt enthaltene, *Skelletonskript* verwendet wird um die Bewegungsinformationen zu erhalten. Der große Unterschied liegt hierbei darin, dass bei dem *Mouse- und Keyboardmapping* die einzelnen Skelettpunkte an ein Kugelskelett gebunden werden. Hierbei ist jeder Gelenkpunkt ein eigenes *GameObject* in *Unity3D*). Diese Zwischeninstanz dient nur zur Verarbeitung der Bewegungen und Gesten des

Spielers. Bei der Variante Körper- und Bewegungsmapping wird keine solche Zwischeninstanz benutzt. Stattdessen werden die Skelettpunkte, welche die *Kinect* ermittelt direkt auf die zugehörigen Skelettpunkte des Charakters gemappt. Um den Charakter von den translativen Bewegungen des Spielers loszulösen, werden nur dessen Rotationsbewegungen berücksichtigt. Lediglich der *Root* des Charakterskelettes kann auf der Y-Achse transliert werden, um das Ducken und Springen zu ermöglichen. Es ist unerheblich, ob die Bewegung des Charakters durch die Geschwindigkeit der Füße beim Laufen auf der Stelle, oder durch eine bestimmte Fußstellung, ausgelöst wird. Der entscheidende Punkt liegt eher darin, dass bei der ersteren Variante der Spieler auf der Stelle laufen muss um seine Bewegungen auf den Charakter zu übertragen. Zwar wird der Charakter durch den virtuellen Raum bewegt, aufgrund der eins-zu-eins Übertragung der Beinbewegung, sieht dies jedoch nur wenig realistisch aus. Dieses Problem wird umgangen, indem die Laufbewegung, wie in der ersten Variante, durch vorgefertigten Animationen simuliert wird. Weitere Probleme, die bei der Implementierung des zweiten Ansatzes auftraten, entstanden durch den Verbund des Charakterskeletts. Einzelne Punkte übertrugen Zitterbewegungen auf das ganze Skelett. Darüber hinaus können unschöne Verschränkungen von einzelnen Gliedmaßen entstehen. Diese konnten auf das fehlerhafte Erkennen einzelner Skelettpunkte durch die *Kinect Hardware* zurück geführt werden (vgl. Abbildung 6 u. r.).

Abbildung 7. Vor- und Nachteile der einzelnen Verfahren.

5. FAZIT

Mit diesem Artikel wurden zwei unterschiedliche Konzepte vorgestellt, mit denen sich *Ego-* bzw. *Thirdperson-Shooter* Charaktere unter Verwendung der *Kinect Hardware* von Microsoft durch eine 3D Spielszenerie steuern lassen. Diese Konzepte lösen dabei die Problematik des begrenzten Bewegungsraums des Spielers. Die beschriebenen Ansätze sind nicht marktreif, zeigen aber prinzipiell in welche Richtung sich die *Natural-User-Interfaces* (kurz: *NUI*) im Anwendungsgebiet der *Ego-* bzw. *Thirdperson-Shooter* Spiele bewegen könnten. Bei der Implementierung sind teilweise gravierende Fehler beim Körper- und Bewegungsmapping aufgetreten. Sobald einzelne Körperteile nicht richtig erkannt werden, kommt es zu unschönen Ergebnissen bei der Darstellung des Charakters. Auch das Übertragen von Störungen (zitternde Punkte) durch den Verbund der Charaktergelenke ist ein Problem, das gelöst werden muss (vgl. Abb. 7). Die auftretenden Fehlstellungen beim Charakter könnten durch *Full Body Inverse Kinematic* (kurz: *FBIK*) behoben werden. Mit einem *FBIK* können sich die Gelenke nicht mehr in beliebige Positionen verdrehen. Das Zittern könnte durch Glättungsalgorithmen minimiert werden. Die genannten Probleme sind jedoch vernachlässigbar wenn, wie bei der Egoperspektive, lediglich die Arme des Charakters zu sehen sind. Es hat sich herausgestellt, dass die Unterschiede zwischen *Ego-* und *Thirdperson-Perspektive* einzig bei dem Körper- und Bewegungsmapping relevant sind. Nach der Befragung von Probanden ergab sich ein Mittelweg, welcher Lösungsansätze aus beiden Varianten enthalten sollte. Hierbei sollen die Beine des Charakters animiert werden, der Oberkörper jedoch gemappt. Hierdurch behält der Spieler den Bezug zum Charakter und die Laufanimationen wirken realistischer. Als Auslöser für die Vorwärtsbewegung könnte eine Schranke genutzt werden, die beim Laufen auf der Stelle ausgelöst wird.

6. REFERENZEN

[1] "Kinect: Die technischen spezifikationen," GamersGlobal, Juni 2010. [Online]. Available: http://www.gamersglobal.de/news/24332

[2] "Endlich da: Kinect für xbox 360," August 2010. [Online]. Available: http://computer.de.msn.com/spiele/kinect/article.aspx?cp-documentid=154666699

[3] "Kinect gunsmith advert," Juni 2011. [Online]. Available: http://www.ghostrecon.net/files2/index.php?act=view&-id=1299

[4] J. Walz, "Minority report im fernsehsessel - windows-programme mit gesten steuern," 2011.

[5] D. Tajeddini, "Minority report im fernsehsessel - windows-programme mit gesten steuern," pp. 168–172, November 2011.

[6] "Openni downloads," OpenNI, Juli 2011. [Online]. Available: - http://www.openni.org/Downloads/OpenNIModules.aspx

[7] "Sensorkinect," GitHub, Juli 2011. [Online]. Available: https://github.com/avin2/SensorKinect

[8] "More kinect hacking," Januar 2011. [Online]. Available: http://fpgacomputing.blogspot.com/2011/01/more-kinect-hacking.html

ANALYSE DER FERNSEHER-STEUERUNG MIT HILFE EINES SMARTPHONES ZUR ABLÖSUNG DER KLASSISCHEN FERNBEDIENUNG

Dennis Becker, Benjamin Rühl

Technische Hochschule Mittelhessen
Friedberg, Hessen

ABSTRACT

Seit über 60 Jahren gibt es die Fernbedienung als Eingabegerät zur Steuerung des eigenen Fernsehers. Durch neue Technologien wie Smartphones ergeben sich neue Ansätze und Konzepte. Diese Arbeit untersucht, ob die Bedienung mit der klassischen Fernbedienung überholt ist und durch neue Gesten und Funktionen auf Smartphones ersetzt werden kann. Es soll im Ansatz überprüft werden, ob diese neuen Technologien massentauglich und nutzerfreundlich gestaltet werden können - so dass jedermann intuitiv umsteigen kann.

Es werden neue Konzepte zur Bedienung per Smartphone untersucht und kritisch bewertet. Es stellt sich heraus, dass die fehlende Haptik eines flachen Smartphone-Bildschirms ein Problem darstellt. Aufgrund dieses fehlenden taktilen Feedbacks eignet sich ein Smartphone noch nicht als Ersatz für eine klassische Fernbedienung, denn diese kann für die Hauptfunktionen von den meisten Nutzern fast blind benutzt werden und lenkt nicht vom aktuellen Fernsehgeschehen ab.

Index Terms— Fernbedienung, Fernseher, Eingabegeräte, Gestensteuerung, Smartphone

1. DIE FERNBEDIENUNG - EIN ZEITLOSES INSTRUMENT?

Smartphones nehmen ihren Nutzern in der heutigen Zeit immer mehr Aufgaben ab. Mussten sie noch vor ein paar Jahren Routenplanungen ausdrucken, erkennt heute fast jedes Smartphone die aktuelle Position und kann den Benutzer mit Ansagen sicher ans Ziel führen. Doch wieso drückt im 20. Jahrhundert noch jeder auf einer analogen Plastikfernbedienung mit unsäglich vielen Knöpfen herum? Diese Arbeit soll untersuchen ob es dem Smartphone gelingen kann ein Instrument zu ersetzen, welches seit mehr als 60 Jahren, fast unverändert, existiert. Dabei wird der Fokus auf die grundlegenden Funktionen einer Fernbedienung gelegt und darauf, welche Probleme sich bei einem Smartphone als Ersatz zeigen.

2. AKTUELLE MARKTSITUATION

Fast alle großen Firmen aus der Unterhaltungselektronik-Branche versuchen sich daran, die klassische Fernbedienung zu ersetzen und Alternativen zu bieten. Dabei gibt es zwei grundlegende Ansätze. Zum Einen versuchen die Hersteller, eigenständige Hardware zu entwickeln, die speziell den modernen Anforderungen einer Fernbedienung gewachsen sind. Zum Anderen sollen vorhandene Smartphones mit reiner Software oder einer Kombination aus Adapter (z. B. Infrarot-Sender) und Software zur Allzweckwaffe werden. Diese Arbeit konzentriert sich auf letzteres, fasst aber relevante Entwicklungen neuer Eingabegeräte zusammen.

2.1. Spezielle Hardware

Gesture Remote Ein Zusammenschluss von drei Innovationsfirmen zeigt eine komplett neues Eingabegerät (Abbildung 2). Das Gerät liegt in der Hand und wird durch die Bewegung des Daumens oberhalb der Oberfläche gesteuert. Dabei kann nicht nur von links nach rechts oder von oben nach unten gewischt werden, sondern auch die Höhe des Daumens über dem Gerät dient als Eingabegeste. [1] Um die Bedienung des Fernsehergerätes zu ermöglichen, muss eine speziell darauf angepasste Software auf dem Fernseher laufen, die für diese Bewegungen optimiert ist. Alleine das Einführungsvideo zeigt deutlich, dass man sich bei diesem Gerät einarbeiten muss. Die variable Höhe des Daumens, die nicht fassbar ist, wird speziell ältere Menschen Schwierigkeiten bereiten.

LG Magic Motion Control Die Firma LG hat für ihre SMART-TV Geräte eine neue Fernbedienung entwickelt (Abbildung 2), die für Hauptfunktionen (Lautstärke, Kanal wechseln, Ein/Ausschalten) herkömmliche Tasten hat, sonst aber nur mit dem Zeigen auf den Bildschirm das Gerät steuert. Kern ist also die Software die auf dem TV-Gerät läuft - die Fernbedienung dient als Zeiger auf den Bildschirm (ähnlich der Wii-Spielekonsole). Durch anvisieren und klicken (Taste auf der Fernbedienung) können Aktionen ausgeführt werden. Immer nötig ist ein direkter Sichtkontakt auf den Fernseher, sowohl für das Gerät als auch für den Nutzer. [2] Damit geht der klassische Charakter einer Fernbedienung verloren, denn die Funktionen des

TVs werden nicht direkt von der Fernbedienung aus gesteuert, sondern auf dem Gerät selbst.

Samsung RMC30D Die Smartphone-ähnliche Fernbedienung von Samsung steht zwischen den eingangs erwähnten Kategorien (Abbildung 2). Denn dieses vergleichbar teure Gerät hat einen herkömmlichen Touchscreen und damit keine Vorteile gegenüber einem Smartphone. Im Gegenteil, die Funktionen sind alleine auf das TV-Gerät beschränkt und der Nutzer muss sich dieses Gerät zusätzlich anschaffen. [3]

Abb. 1. V.l.n.r: Gesture Remote, LG Magic Motion Control und Samsung RMC30D [1] [2] [3]

2.2. Smartphone-Apps

Im Folgenden sollen besondere Fernbedienung Apps für Android und iOS vorgestellt und deren Unterschiede herausgestellt werden. Auch hier gibt es wieder zwei Kategorien, in welche die Apps eingeordnet werden können. Auf der einen Seite reine Fernseher-Steuerungen, die nur die Funktionalität des TV-Gerätes beeinflussen und auf der anderen Seite sogenannte Universal-Fernbedienung-Apps. Durch die technologische Ausbreitung im Wohnzimmer gibt es immer mehr Geräte die gleichzeitig bedient werden wollen (Verstärker, DVD-Player, SAT-Receiver und viele andere). Um die Untersuchungen dieser Arbeit und deren Erkenntnisse möglichst einfach zu halten, wird sich die spätere Auswertung auf die klassische Fernbedienung und deren mögliches Smartphone-Pendant beziehen. Durch den Bildschirm, den Smartphones besitzen, lassen sich kontextbezogene Informationen und Schaltflächen anzeigen. Multitoucheingaben ermöglichen die Steuerung über verschiedene Gesten.

Voomote Zapper Die Firma zero1.tv GmbH vertreibt ab Mitte Oktober 2011 einen Adapter für das iPhone und iPad, der Befehle in Infrarot-Signale umsetzen kann und damit jegliche Geräte steuert. [4] Diese Technik ist nicht neu und es gibt bereits Anbieter, die die gleiche Funktionalität bieten. Allerdings ist der Hauptbestandteil die App, mit der das TV-Gerät gesteuert werden soll. Besonders ist hier die Einbindung der Fernsehzeitschrift in die Applikation. So kann der Nutzer auf seiner Couch sitzen, das aktuelle Programm studieren, Zusatzinformationen wie Trailer und Beschreibung abrufen, darüber twittern oder sich an eine zukünftige Sendung erinnern lassen. Gefällt dem Nutzer das Programm, kann er mit einem Klick das Bild auf sein TV-Gerät bringen. Die restliche Steuerung, beispielsweise der Lautstärke, passiert über virtuelle Tasten, die eingeblendet werden können.

Panasonic Remote-App Panasonic hat für die eigene Viera-Produktserie eine App für Android und iOS entwickelt, in der alle Funktionen der Fernbedienung übernommen wurden. Neben den üblichen Schaltflächen für die Tasten, die es auch auf der echten Fernbedienung gibt, soll der Nutzer per Gestensteuerung jederzeit auf die wichtigsten Funktionen zugreifen können. Schüttelt er beispielsweise sein Smartphone, schaltet sich der Fernseher aus. Um die Lautstärke zu regeln, kann man mit zwei Fingern eine Kreisbewegung machen, als würde man einen echten Lautstärkeknopf drehen. Mit zwei Fingern hoch und runter wischen schaltet zwischen den Kanälen um. [5]

Logitech Harmony® Link Ein Komplettsystem mit externem Infrarot-Sender und eigener Smartphone-App bietet Logitech seit kurzer Zeit in den USA an. Dort enthalten ist unter anderem ein Programmguide und die Steuerung von bis zu acht Home-Entertainment Geräten. Sobald man einen sogenannten "Gesture-Modeäktiviert, kann man mit einer Bewegung von links nach rechts Vor- und Zurückspulen, bei einer Wischbewegung von oben nach unten wird die Lautstärke reguliert. [6]

2.3. Wo ist das Problem?

Die Ansätze und Entwicklungen der verschiedenen Hersteller sind vielversprechend. Für Technologiefans wird auch das ein oder andere Produkt jetzt schon vielen Ansprüchen gerecht. Allerdings ist noch kein Produkt in der Lage jede Benutzerschicht anzusprechen. Auch wenn Logitech schon einen sehr umfassenden Ansatz nimmt, der Nutzer wird immer vom aktuellen Geschehen auf dem Fernseher abgelenkt und muss sich damit beschäftigen, seine Fernbedienung zu bedienen. Das kann nicht das Ziel sein. Die Frage ist, was fehlt, damit der Nutzer die wichtigsten Funktionen ohne Nachdenken und vielleicht sogar ohne hinzuschauen am Fernseher bedienen kann? Dabei geht es gar nicht um jede Funktion, die ein modernes Fernsehergerät unterstützt, sondern um die wesentlichen Funktionen, auf die der Nutzer schnell zugreifen möchte.

Lautstärke In den vorgestellten Applikationen wird die Lautstärke entweder wie bei Panasonic über eine Geste

gesteuert oder aber über virtuelle Schaltflächen. Das heißt, der Nutzer muss seinen Blick vom Fernseher abwenden, das Smartphone wohlmöglich aktivieren, die richtige Schaltfläche finden und diese betätigen. Die Autoren empfehlen, die bei fast allen Smartphones vorhandenen Hardwaretasten für das Einstellen der Lautstärke zu nutzen. Der Anwender kennt sein Smartphone und weiß, wo sich diese Tasten befinden. Er kann deshalb das Fernsehbild im Auge behalten und gleichzeitig die Lautstärke einstellen. Einige Smartphones (u. a. das iPhone) bieten auch eine spezielle Taste an, um den Ton komplett an-, bzw. auszuschalten.

Kanal wechseln Als Erweiterung zu Schaltflächen auf dem Bildschirm nutzt u.a. Logitech die Möglichkeit, deutlich mehr Informationen zu dem aktuellen Programm als nur den Sendernamen anzuzeigen. So wird dann nicht auf dem Fernsehergerät "gezappt", sondern in der Applikation. Die komplette Aufmerksamkeit des Nutzers liegt dann auf dem Smartphone und das könnte sogar bei ausgeschaltetem TV-Gerät der Fall sein. Ein Faktor der hierbei rausfällt, ist das die Aktion mit dem Fernseher von einer einzigen Person gesteuert wird. Beim herkömmlichen Kanal umschalten können auch andere Personen im Raum entscheiden und sehen was passiert und was kommt. Im Laufe dieser Arbeit wurden verschiedene Konzepte erarbeitet, die einen intuitiven Kanalwechsel bieten sollen.

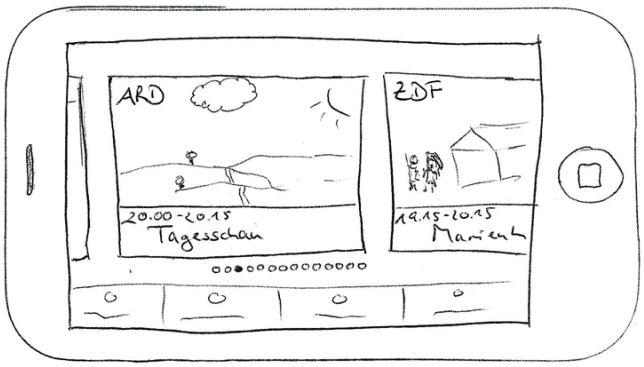


Abb. 2. Konzept einer bebilderten Kanalauswahl

Das in Abbildung 2 gezeigte Konzept bietet dem Nutzer Standbilder aus dem aktuellen Fernsehprogramm, die in regelmäßigen Abständen aktualisiert werden. Wenn der Nutzer das Vorschaubild nach oben (also in Richtung TV-Gerät) wischt, schaltet das Gerät automatisch auf den ausgewählten Kanal. Zwar ist dies eine moderne und bessere Alternative zu der sogenannten Picture-in-Picture-Funktion aktueller TV-Geräte, aber der Nutzer muss sich komplett vom Fernseher abwenden um sich mit der Anwedungen auf dem Smartphone zu beschäftigen. Also gilt auch dieses Konzept nicht als Lösung zum Ersatz der klassischen Fernsehfernbedienung. Zwar könnte eine solche Funktion von Technikliebhabern gewünscht sein, das beschriebene Problem für den Normalnutzer ist aber das Kleinere im Wandel von der Fernbedienung zum Smartphone.

3. PROBLEM DES FEHLENDEN FÜHLBAREN FEEDBACKS

Smartphones mit all ihren technischen Raffinessen haben einen entscheidenden Nachteil: Sie erfordern unsere vollständige Aufmerksamkeit. Jede Interaktionsmöglichkeit muss mit den Augen erfasst und ausgewertet werden. Erst dann bewegt sich der Daumen auf den gewünschten Button und drückt ihn. Bei einer Fernbedienung ist die Vorangehensweise identisch. Jedoch lernt man mit der Zeit wo sich welche Tasten befinden und man bedient die Fernbedienung immer öfter blind. Dieser Lerneffekt bleibt bei einer Applikation auf einem Smartphone aus. Warum? Das Navigieren unserer Finger auf einer Fernbedienung kann sowohl durch den Gesichtssinn, als auch durch den Tastsinn erfolgen. Nach einer Einarbeitungszeit sind die wichtigsten Tasten und ihre Platzierung auf der Fernbedienung bekannt und man erreicht die gewünschten Funktionen ohne auf die Fernbedienung zu schauen. Dieses Navigieren durch tasten ist bei einem ebenen Touchscreen nicht möglich. Merchel et al. untersuchten die Hypothese, dass taktiles Feedback die Fehlerrate bei Interaktionen mit einem Touchscreen verringert. Hierzu ließen sie Probanten auf einer OnScreen-Tastatur verschiedene Nummerfolgen eintippen und verglichen die eingegebenen Zahlen mit der vorgeschriebenen Nummernfolge. Das Ergebnis bezeugte: Vibrierte das Smartphone bei der Berührung der Nummernfelder verringerte sich die Fehlerrate enorm. Ausserdem stellten sie fest, dass Benutzer nur zwischen einer begrenzten Anzahl an Vibrations-Stimuli unterscheiden können. Obwohl die Probanten 6 verschiedene Stimuli fühlten ordneten Sie diese nur in 2 Gruppierungen ein: weiche und harte Vibrationen. [7]

Mit der gleiche Hypothese, jedoch auf die Eingabe von Text mit einer Software-Tastatur bezogen, beschäftigten sich Hoggan et al. im Jahr 2008. [8] Er vergleichte die Fehlerrate bei Texteingaben über Hardware-Tastaturen, Software-Tastaturen mit und Software-Tastaturen ohne taktilem Feedback. Dabei stellte er fest: Eingaben über Software-Tastaturen ohne taktilem Feedback weisen eine erhöhte Fehlerrate im Vergleich zu Eingaben mit taktilem Feedback auf. Software-Tastaturen mit haptischem Feedback hingegen haben eine ähnlich niedrige Fehlerrate wir Hardware-Tastaturen.

Doch ist das Vibrieren des kompletten Smartphones als Reaktion auf das Drücken eines Teilbereichs des Touchscreens eine logische Konsequenz? Koskinen et al. untersuchten aktuelle Technologien für haptisches Feedback. Ihre Studie beschäftigte sich mit einfachen Vibrationsmotoren, wie sie in vielen aktuellen Smartphones verbaut sind und

Piezo-Aktoren. Letztere Technologie ermöglicht das Vibrieren von Teilbereichen des Touchscreens. Sie belegten, dass Piezo-Aktoren, statistisch gesehen, keinen signifikanten Vorteil gegenüber den Vibrations-Motoren aufweisen. [9]

Betrachtet man diese Studien sollte es möglich sein die Bedienung von mobilen Applikation auf aktuellen Smartphones mit haptischem Feedback zu unterstützen.

4. LÖSUNGEN FÜR HAPTISCHES FEEDBACK

Das Betriebssystem Android bietet per API die Möglichkeit den eingebauten Vibrationsmotor anzusprechen. Dabei kann ein Vibrationsmuster und dessen zeitliche Dauer festgelegt werden. Die Firma immersion. bietet für Entwickler, auf Basis dieser API-Schnittstelle, ein erweitertes SDK an. Das MOTIV SDK ermöglicht die Definition von Vibrationsmustern anhand eines grafischen Editors. [10]

Auf Grundlage der oben genannten Studien und mit Hilfe des MOTIV SDK's wurde nun ein Prototyp einer minimalistischen Fernbedienung entwickelt. Dieser Prototyp unterstützt den Benutzer durch haptisches Feedback bei der "blinden Bedienung". Dabei werden unterschiedliche Interaktionselemente mit unterschiedlichem Feedback versehen, um die tastende Navigation so einfach wie möglich zu gestalten.

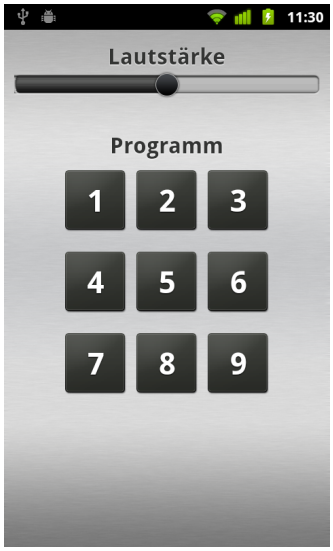


Abb. 3. Oberfläche des Haptik-Prototypen

5. FAZIT

Es gibt viele unterschiedliche aber dennoch vielversprechende Fernbedienungen auf dem Markt. Sie sind ein guter Ansatz um die Fähigkeiten des Smartphones als intuitive Fernbedienung auszunutzen. Jedoch gibt es noch viel Potential zur Verbesserung. Bisher erfordern alle Applikationen die volle Aufmerksamkeit des Benutzers und sind somit kein Ersatz für eine übliche Fernbedienung. Dabei kann man mittlerweile den Vibrationsmotor in Smartphone als Standard ansehen. Dies ermöglicht Entwicklern ihre Applikationen um eine rudimentäre haptische Komponente zu erweitern. Das dies die Bedienung von Touchscreens vereinfacht, wurde durch verschiedensten Studien belegt. Nun ist es an Hardware-Herstellern gelegen, bessere Umsetzungen für taktiles Feedback zu entwickeln. Erst wenn sich Interaktionselemente anfühlen wie ihr Pendant aus der Realität lassen sie sich blind bedienen.

6. REFERENCES

[1] "Gesture Remote – the future of TV control," Abgerufen von `http://gesture-remote.com/` am 01.10.2011.

[2] "LG SmartTV - Mit Home Dashboard und Magic-Motion-Fernbedienung," Abgerufen von `http://www.lg.com/de/microsite/SmartTV.jsp` am 20.09.2011.

[3] "Eigenschaften von RMC30D | Fernseher (Zubehör) von Samsung," Abgerufen von `http://www.samsung.de/de/Privatkunden/TVHeimkino/Fernseher/Zubehoer/rmc30d/RMC30D1P2ZG/accdetail.aspx` am 10.09.2011.

[4] "VooMote Zapper | Universalfernbedienung und TV-Programmzeitschrift," Abgerufen von `http://www.voomote.tv/zapper/de/` am 03.10.2011.

[5] "Produkte - Flachbildfernseher - Willkommen - VIERA Remote App [...]," Abgerufen von `http://www.panasonic.de/html/de_DE/Produkte/Flachbildfernseher/Willkommen/VIERA+Remote+App/7565923/index.html` am 28.09.2011.

[6] "Logitech Harmony® Link," Abgerufen von `http://www.logitech.com/en-us/remotes/universal-remotes/devices/harmony-link` am 28.09.2011.

[7] Sebastian Merchel and Ercan Altinsoy, *Audiotactile Feedback for Touch Screens*, 2009.

[8] E Hoggan, S A Brewster, and J Johnston, *Investigating the effectiveness of tactile feedback for mobile touchscreens*, p. 1573, ACM Press Addison Wesley, 2008.

[9] Emilia Koskinen, Topi Kaaresoja, and Pauli Laitinen, *Feel-good touch: finding the most pleasant tactile feedback for a mobile touch screen button*, vol. 0, pp. 297–304, ACM, 2008.

[10] "MOTIV Haptic Development Platform for Android," Abgerufen von `http://www.immersion.com/products/motiv/` am 28.09.2011.

ENTWICKLUNG HYBRIDER STEUERUNGSKONZEPTE FÜR FIRST-PERSON-SHOOTER UNTER VERWENDUNG VON KINECT MIT WIIMOTE- UND NUNCHUK-CONTROLLER

Daniel Beckmann

Technische Hochschule Mittelhessen,
Wilhelm-Leuschner-Str. 13, 61169 Friedberg

ABSTRACT

In dieser Publikation werden Steuerungskonzepte für First-Person-Shooter (FPS) vorgestellt, die Kinect und Wiimote-Controller als Eingabesysteme kombinieren. Durch die Kombination beider Systeme sollen bisherige Steuerungsmethoden verbessert werden, um eine höhere Benutzerakzeptanz im (Hard-)Core-Gaming Bereich zu erzielen. Die entwickelten Konzepte zielen darauf ab, den Nunchuk-Controller der Wiimote für die Fortbewegung des Charakters zu verwenden und mittels Kinect eine Abbildung des Spielerskeletts in der virtuellen Umgebung zu erzeugen. Dadurch kann der Spieler verschiedene Bewegungsabläufe wie Ducken oder Ausweichen durch Körperbewegungen steuern.

Index Terms— Human computer interaction, gestures, game controls, first person shooter, player experience

1. EINLEITUNG

Die Steuerung von Videospielen durch Körperbewegungen und Gesten wird immer beliebter. Zu den bekanntesten und beliebtesten Systemen zählen unter anderem die Nintendo Wii und die Microsoft Xbox Kinect. Diese ermöglichen eine einfache und intuitive Steuerung von Videospielen, die vor allem die Zielgruppe der Casual Gamer (Gelegenheitsspieler) anspricht.

Die Kinect ermöglicht es das Skelett des Spielers auf einen virtuellen Charakter räumlich abbilden zu können (siehe Abb. 2). Hierbei können alle Bewegungen des Spielers in Echtzeit umgesetzt werden [1, 2]. Der Ansatz der Kinect, Eingaben nur über Körperbewegungen ohne zusätzlichen Controller tätigen zu können, macht jedoch komplexere Spiele wie First-Person-Shooter (FPS) schwer umsetzbar und die Zielgruppe der (Hard-)Core-Gamer lässt sich somit nur kaum ansprechen. Ein Beispiel eines Problems ist die Fortbewegung im Spiel. Diese kann zwar über die Spielerposition im Raum umgesetzt werden, jedoch nur innerhalb des Erfassungsbereichs der Kinect. Demnach muss diese Interaktion über Umwege, wie beispielsweise das Laufen auf der Stelle oder eine virtuelle Tastatur auf dem Boden realisiert werden [3], die jedoch umständlich wirken und das Spielgefühl wesentlich verringern.

Die Wii mit seinem Wiimote- und Nunchuk-Controller bietet hingegen die Möglichkeit, Eingaben über einen Analogstick zu tätigen. Auch Zeigebewegungen können durch den Gyrosensor im Motion Plus Adapter oder mittels der Infrarot-Sensorleiste, die als Orientierungspunkt im Raum dient, umgesetzt werden. Dieses System bietet somit zusätzliche Eingabemöglichkeiten, die die Steuerung der Kinect erweitern können. Es sollen deshalb Steuerungskonzepte entwickelt werden, die die Stärken beider Systeme ausnutzen, um eine möglichst intuitive Steuerung zu ermöglichen. Diese sollen vor allem in komplexeren Spielen wie FPS eingesetzt werden können und im (Hard-)Core-Gaming-Bereich auf eine größtmögliche Benutzerakzeptanz abzielen.

Abbildung 1. Microsoft Xbox Kinect (links) und Nintendo Wiimote mit Nunchuk-Controller (rechts). (Quellen: Microsoft, Nintendo)

2. STAND DER TECHNIK

Die Steuerungs- und Interaktionsmöglichkeiten von Kinect und Wiimote in Spielen wurden bereits oft getrennt voneinander betrachtet. Arlt und Krause entwickelten bereits Ansätze um einen 3D-Charakter über die Kinect steuern zu können [3]. Es zeigte sich, dass verschiedene Umsetzungen zwar möglich sind, jedoch mit wesentlichen Einschränkungen einherkommen. In einer Untersuchung von Controllern für FPS zeigte sich ebenfalls, dass neue Eingabegeräte oft zu Problemen in der Benutzbarkeit führen. Jedoch ergab sich durchaus ein Mehrwert bei intuitiven Umsetzungen [4]. Somit lassen

sich bisher vor allem in Casual Games oder Lernspielen durch den gezielten Einsatz von beispielsweise einer Wiimote sehr gute Ergebnisse mit einer hohen Nutzerakzeptanz erzielen [5]. Zum Teil wurden auch bereits beide Systeme in Kombination betrachtet. Lasorne und Guyo stellten in ihrem Projekt "Gundam Kinect" einen Prototyp eines FPS vor, der mittels Kinect und Wiimote gesteuert wird [6]. Der Schwerpunkt wurde hier jedoch lediglich auf die Möglichkeit der Umsetzung gelegt. Ähnliche Projekte wurden von verschiedenen Hobbyentwicklern umgesetzt. Hierzu wurden bereits bestehende Computerspiele über Wiimote und Kinect angesteuert. Die Kinect wurde hier mit dem "Flexible and Articulated Skeleton Toolkit" (FAAST)[1] angebunden, das verschiedene Gesten in Tastatureingaben umsetzen kann [7]. Die Bewegungen verhalten sich somit nicht für jeden Spieler individuell, sondern wegen des Auslösens einer Aktion bei jedem Spieler identisch. Hierbei handelt es sich auch um Machbarkeits-Prototypen, die zeigen sollen dass die Kombination beider Systeme möglich ist und in welche Richtung sich die Spielebranche entwickeln könnte.

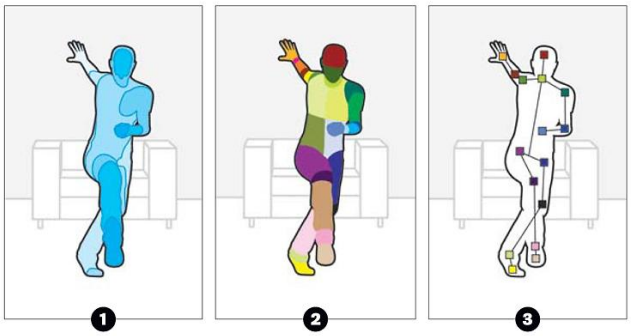


Abbildung 2. Die Kinect ermöglicht es den Spieler durch Bildverarbeitungsmethoden zu erkennen und auf ein virtuelles Skelett abzubilden. (Quelle: fastcompany.com)

3. ENTWICKLUNG VON STEUERUNSKONZEPTEN

Komplexe Spiele wie FPS erfordern verschiedenste Eingaben vom Spieler. Diese werden im Nachfolgenden betrachtet und parallel dazu erörtert, welches der beiden Eingabesysteme jeweils besser geeignet ist und wie die Steuerung umgesetzt werden könnte.

3.1. Fortbewegung

Die Fortbewegung eines Spielcharakters über die Kinect stellt sich als relativ problematisch heraus [3], denn der Erfassungsbereich, in dem der Spieler erkannt werden kann ist begrenzt. Auch alternative Bewegungsabläufe oder Gesten, wie beispielsweise das Laufen auf der Stelle oder eine virtuelle Tastatur auf dem Boden sind zu fehleranfällig und auf Dauer nicht praktikabel. Deshalb sollte das Fortbewegen über den Analog-Stick des Nunchuk-Controllers der Wiimote realisiert werden (siehe Abb. 1 Nunchuk-Controller). Dieser hat sich in der Spieleszene bereits durchgesetzt und wird in allen gängigen Konsolenspielen zur Fortbewegung genutzt.

Ein neuer Ansatz bei FPS ist ein Skelettmapping durch die Kinect. Hierbei wird der Spieler eins zu eins in der Spielumgebung abgebildet. Der Spieler kann somit durch seine Körperbewegungen mit dem Charakter in Deckung gehen, sich zur Seite lehnen oder auch Springen. Bei herkömmlichen Bewegungsaktionen in Spielen werden solche Interaktionen durch einen Tastendruck ausgelöst und immer nach dem gleichen Muster ausgeführt. Mittels eines Skelettmappings können solche Aktionen für jeden Spieler individuell ausgeführt werden. Dadurch sind alle denkbaren Bewegungsabläufe für den Spielecharakter möglich. Andere typische Aktionen, wie ein Lehnen zur Seite sind also auch einfach verwendbar und müssen nicht zusätzlich implementiert werden. Die Alternative hierzu ist das ledigliche Auslösen von vordefinierten Aktionen, wenn verschiedene Körperbewegungen erkannt werden. In diesem Fall sind die letztendlich im Spiel umgesetzten Bewegungen für jeden Spieler gleich. Hierdurch geht jedoch die Spieldynamik verloren, auf die ein System wie die Kinect abzielt. In manchen Fällen ist das aber auch unumgänglich, weil beispielsweise ein Sprung über einen Zaun oder eine Mauer für den Spieler schwer umzusetzen ist.

Abbildung 3. Der Spieler wird über die Kinect erfasst. Dadurch sind beliebige Bewegungen möglich, die auch im Spiel umgesetzt werden.

3.2. Drehung und Zielbewegung

In herkömmlichen FPS wird eine Drehung des Charakters gleichzeitig mit der Zielbewegung umgesetzt. Das Fadenkreuz befindet hierbei immer in der Mitte des Bildschirms. Für die Umsetzung der Zielbewegung bieten sich sowohl die Hand als auch die Wiimote an. Für ein bestmögliches Ergebnis sollten beide Eingabemethoden kombiniert werden. Als primäres Gerät eignet sich die Wiimote besser, da auch ein Zielen möglich ist, wenn die Hand nicht erkannt wird. Für die Drehung zu den Seiten ist ein relatives System üblich, das heißt der Spielcharakter dreht sich,

[1] http://projects.ict.usc.edu/mxr/faast/

wenn der Controller nach links oder rechts gerichtet wird. Der Winkel des Controllers zur Mitte bestimmt hierbei die Drehgeschwindigkeit. Hier zeigen sich jedoch schnell die Probleme, dass der Controller nie genau in der Mitte gehalten werden kann und auch durch die Abtastung Ungenauigkeiten entstehen. Dadurch ist das Fadenkreuz und Blickfeld des Spielers ständig am Wackeln. Zur Lösung dieses Problems bieten sich zwei Vorgehensweisen an:

- Fadenkreuz von der Bildschirmmitte entkoppeln
- Fadenkreuz fixieren – Drehbewegung durch gleichzeitig gedrückt gehaltene Taste

In vielen Spielen für die Wii ist das Fadenkreuz bereits von der Bildschirmmitte entkoppelt und kann frei im Bild bewegt werden. Eine Drehbewegung wird nur ausgeführt, wenn mit dem Fadenkreuz in einen Randbereich des Bildes gezeigt wird (vgl. Abb. 4). Diese Variante stellt sich aus Erfahrungswerten als etwas unpraktikabel heraus, da sie für (Hard-)Core-Gamer sehr ungewohnt wirkt.

Für den zweiten Ansatz muss vor der Drehbewegung eine Taste des Controllers gedrückt gehalten werden. Erst danach wird eine Bewegung der Wiimote direkt in eine Ziel- und Drehbewegung umgesetzt. Das Fadenkreuz verweilt wie bei üblichen FPS dauerhaft in der Mitte des Bildschirms. Eine Neupositionierung der Wiimote ist durch einfaches Loslassen der Taste möglich. Das Drücken der Taste zum Drehen wirkt im ersten Moment etwas ungewohnt, ist jedoch auf Dauer durchaus praktikabel. Bei dieser Methode können der Wiimote auch Bewegungsgesten zugewiesen werden, die sich somit auch nicht störend auf die Steuerung des Charakters auswirken, wie es beim ersten Ansatz der Fall wäre.

3.3. Primäre Aktionen

Primäre Aktionen wie das Schießen sind essentiell wichtig für den Spieler. Somit sollten sie auf möglichst leicht erreichbare Tasten der Wiimote oder des Nunchuks gelegt werden. Ein Ausführen solcher Aktionen durch Gesten ist gegebenenfalls etwas unzuverlässig und wirkt teilweise nicht allzu intuitiv, wenn man sie mit den realen Bewegungsabläufen vergleicht. Sekundäre Aktionen bieten sich hierfür eher an, da sie in der Realität auch mit größeren Bewegungsabläufen verbunden sind.

In vielen FPS entspricht der Rechtsklick einem Anvisieren mit leichtem Zoomeffekt, der auch zu den primären Aktionen zählt. Für diese Aktion kann die Steuerung zusätzlich vereinfacht werden, indem eine Bewegung der Wiimote bei gedrückter Taste fürs Anvisieren direkt in einer Drehbewegung resultiert. Das zusätzliche Drücken der Taste für die Drehbewegungen kann deshalb weggelassen werden, weil man davon ausgehen kann, das ein Spieler beim anvisieren auch gleichzeitig zielen möchte.

3.4. Sekundäre Aktionen

Für sekundäre Aktionen im Spiel können entweder Tasten der Controller verwendet oder auch Gesten eingesetzt werden. Diese sollten jedoch intuitiv und möglichst vergleichbar mit den realen Bewegungsabläufen sein. Je nach Typ können die Gesten über die Wiimote oder Kinect umgesetzt werden. Wird eine Geste erkannt, so wird eine Aktion im Spiel ausgelöst. Im nachfolgenden Absatz ein Auszug verschiedener Aktionen, für die sich Gesten anbieten:

Zum Nachladen bietet sich eine Bewegung der linken zur rechten Hand an. Diese Bewegung kann über die Kinect durch die Positionierung der Hände abgefragt werden. Eine Aktion wie das Werfen einer Granate kann beispielsweise über eine Wurfbewegung ausgelöst werden, die auch von Wiimote oder Nunchuk erkannt werden kann. Zum Wechseln der Waffe kann die Hand mit der Wiimote nach unten geschwungen werden, dadurch entsteht der Eindruck des Greifens eines Objekts von unten.

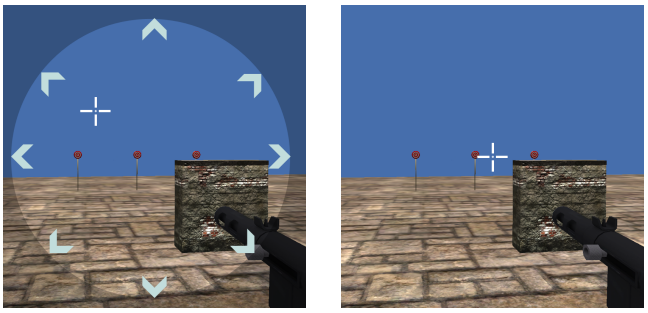


Abbildung 4. Zwei verschiedene Ansätze für das Zielen und Drehen im Spiel - Von der Bildschirmmitte entkoppeltes Fadenkreuz mit Drehbewegungen, die in den Randbereichen ausgelöst werden (links) und fixiertes Fadenkreuz mit unmittelbaren Drehbewegungen (rechts).

4. IMPLEMENTIERUNG

Dieser Abschnitt beschreibt die prototypische Umsetzung eines Spiels, in dem die entwickelten Steuerungskonzepte umgesetzt wurden. Für die Implementierung des Prototyps wurde Unity 3D von Unity Technologies ausgewählt. Unity ermöglicht eine schnelle Umsetzung interaktiver 3D-Anwendungen in kürzester Zeit [8]. Zur Ansteuerung der Kinect unter Windows kann das Software Development Kit (SDK) des Herstellers Microsoft verwendet werden. Dieses ist für nicht-kommerzielle Zwecke frei verfügbar. Die Einbindung der Kinect in Unity erfolgt daraufhin mit dem Zigfu Development Kit (ZDK) von Zigfu[2]. Dieses stellt verschiedene Objekte zur Verfügung, um eine Gestenerkennung oder Skelett-Tracking über die Kinect nutzen zu können.

[2] http://zigfu.com

Die Wiimote wird über die Software GlovePIE angesprochen. Mit GlovePIE können verschiedenste Eingabegeräte an einem Windows-PC verwendet werden. Mittels einer Skriptsprache lassen sich Controlleraktionen auf beliebige Tasten oder auch Mausbewegungen umsetzen.

Im ersten Schritt wurde ein Skript für GlovePIE geschrieben, welches die verschiedenen Eingabetasten der Wiimote auf diverse Tasten der Tastatur umsetzt. Für eine genaue Umsetzung der Bewegung wird ein Wii Motion Plus Adapter für die Wiimote benötigt. Dieses Gerät erweitert die Wiimote um einen Gyrosensor, der die Bewegungsdaten genau abtasten kann und ohne zusätzliche Infrarot-Leiste auskommt. Die Ausgabedaten des Gyrosensors werden ebenfalls im GlovePIE-Skript ausgewertet und dann in eine Mausbewegung umgesetzt. Dadurch kann in Unity ohne zusätzliches Plugin gearbeitet werden, da nur auf gängige Eingabesignale wie Maus und Tastatur reagiert werden muss.

Für die Umsetzung der Charaktersteuerung wurde eine Szene erstellt, welches die Kinect in Unity einbindet. Das Spielerskelett wird hierbei auf eine Menge von Kugeln und Quadern abgebildet. Für die Umsetzung der Steuerung muss ein eigener Controller entwickelt werden, um den so entstehenden Spielcharakter bewegen zu können. Hierzu wurde ein Controller für die Fortbewegung und für die Steuerung des Fadenkreuzes und somit der Blickbewegung entwickelt. Zum Testen des Zielens wurden verschiedene Objekte in der Szene platziert.

5. ZUSAMMENFASSUNG UND ERGEBNISSE

In diesem Artikel wurden verschiedene Konzepte vorgestellt, die dazu verhelfen sollen die Steuerung von FPS über Gesten-Interaktionssysteme zu verbessern. Diese Konzepte lösen einige der beschriebenen Probleme, wie beispielsweise die Drehung oder Bewegung eines First-Person-Charakters im virtuellen Raum. Generell wurde durch die Beobachtungen deutlich, dass eine Erweiterung bestehender Systeme mittels eines zusätzlichen Controllers als durchaus sinnvoll zu erachten ist. Dadurch kann die Steuerung deutlich erweitert und dahingehend verbessert werden, dass auch (Hard-)Core-Gamer als Zielgruppe angesprochen werden können.

Insgesamt sollte die Bewegung des Spielcharakters über einen Analogstick (hier der Nunchuk-Controllers der Wiimote) umgesetzt werden. Auch für Ziel- und Drehbewegungen bietet sich die Wiimote eher an, jedoch kann die Kinect ebenfalls dafür verwendet werden. Das beste Ergebnis wird sich durch Kombination beider Eingabesysteme erhofft. Für die Drehbewegung bietet sich an, das Fadenkreuz wie bei herkömmlichen FPS in der Mitte des Bildschirms zu fixieren und durch gedrückt halten einer bestimmten Taste freizugeben. Die Kinect kann als Eingabegerät für ein Skelettmapping des Spielers verwendet werden, um Bewegungsabläufe wie das Ducken oder Lehnen zur Seite direkt im Spiel umzusetzen.

Der mit diesen Konzepten umgesetzte Prototyp stellt eine Basis für zukünftige Entwicklungen dar, die in der Praxis durchaus Potenzial haben könnten. Im nächsten Schritt sollten die Ergebnisse evaluiert werden. Besonders interessant ist hierbei, welche Rolle die Kinect in Hinsicht auf Bewegungsabläufe des Spielers einnehmen soll. Hier muss entschieden werden, ob ein Skelettmapping mit direkter eins zu eins Umsetzung des Spielers genutzt werden soll oder ob die Bewegungen des Spielers nur zum Auslösen vordefinierter Aktionen (wie Ducken, Springen oder Lehnen) verwendet werden sollen. Im letzteren Fall würde sich der Einsatz des FAAST Frameworks anbieten. Eine weitere Fragestellung ist, inwiefern sich der Bruch in der Bedienung beim Wechseln der Bewegungssteuerung zwischen Körper und Analogstick auf das Spielgefühl auswirkt. Es könnte verwirrend sein, den Charakter über verschiedene Bewegungsabläufe steuern zu können, aber zur Fortbewegung den Analogstick nutzen zu müssen.

6. REFERENZEN

[1] J. Sebastian, Z. Ai, and J.W. Decker, "Performance measurements for the Microsoft Kinect skeleton," IEEE Virtual Reality, 2012, pp. 119–120.

[2] M. Raptis, D. Kirovski, and H. Hoppe, "Real-Time Classification of Dance Gestures from Skeleton Animation," ACM SIGGRAPH/Eurographics symposium on Computer animation, 2011.

[3] H.C Arlt and K. Krause, "Entwicklung und Analyse von Ego- und Thirdperson Steuerungsmethoden eines 3D Charakters mit Hilfe der Kinect Hardware und Unity 3D," in *Fortgeschrittene-Mensch-Maschine Interaktion*. Technische Hochschule Mittelhessen, 2011.

[4] K.M. Gerling, M. Klauser, and J. Niesenhaus, "Measuring the impact of game controllers on player experience in fps games," in *MindTrek 2011*, Tampere, Finland, 2011.

[5] R.A. Sale, M. Dimaiwat, M.R. Solamo, and R. Feria, "KineSpell 2:Acceptability Test of Wii Remote as aController of an Educational Game," in *Human-Centric Computing (HumanCom), 2010*. HumanCom, 2010.

[6] F. Lasorne, N. Guyon, and S. Wan, "Gundam Kinect," 2011.

[7] E.A. Suma, B. Lange, A. Rizzo, D.M. Krum, and M. Bolas, "FAAST: The Flexible Action and Articulated Skeleton Toolkit," IEEE Virtual Reality Conference, 2011, pp. 247–248.

[8] W. Goldstone, *Unity Game Development Essentials*, Packt Publishing, Birmingham, UK, 2009.

ENTWICKLUNG EINES MOBILEN RAUMINFORMATIONS-SYSTEMS AUF BASIS VON QR-CODES

Claudia Ceh, Andreas Kilb, Sebastian Brams

Technische Hochschule Mittelhessen
Fachbereich 11: Informationstechnik - Elektrotechnik - Mechatronik
Wilhelm-Leuschner-Straße 13 61169 Friedberg
E-Mail:{Claudia.Ceh, Anderas.Kilb, Sebastian.Brams}@iem.thm.de

ABSTRACT

Heutzutage können ortsbezogene Informationen nicht nur über Hinweissschilder, sondern auch über Smartphones bereitgestellt werden. Hierfür ist Kenntnis des aktuellen Standorts nötig. Möglichkeiten zur Ortung bieten W-LAN, GPS oder das Einscannen eines Quick Response (QR)-Codes. Die Verwendung eines QR-Codes für diesen Zweck ist Gegenstand dieses Papers. Wir haben am Beispiel einer Hochschule eine Software entwickelt, die durch Einscannen eines QR-Codes verschiedene Rauminformationen darstellt. Dabei unterscheidet die Applikation zwischen Vorlesungsraum, Labor und Büro. Darüber hinaus soll ein Performance Test zeigen, ob das gleichzeitige Parsen aller Daten sinnvoll ist.

***Index Terms*—** QR-Code, NFC, GPS, W-LAN, fingerprint, geoinformation, mobile-tagging

1. EINLEITUNG

Der Campus einer Hochschule oder Universität kann für Besucher oder Studenten im ersten Semester sehr verwirrend sein. Es gibt nur wenige Informationstafeln und Pläne um herauszufinden, wo man sich gerade befindet. Viele Informationen stehen über Umwege zur Verfügung. Gerade wenn man wissen möchte, wann ein Raum belegt ist oder wie die Sprechzeiten eines Professors sind. Diese Informationen stehen zwar im Internet, allerdings kostet die Suche Zeit.
Wesentlich schneller und benutzerfreundlicher wäre es, diese Informationen direkt am Raum abrufen zu können. Zusätzlich wäre es sinnvoll, sich anzeigen zu lassen, wo auf dem Campus man sich aktuell befindet. Mobile internetfähige Geräte wie Notebooks, Mobiltelefone oder Tablets machen dies möglich. In den vergangenen Jahren hat sich die Mobilfunkindustrie gewandelt. Immer mehr Mobiltelefone verfügen über einen mobilen Internetzugang. Dabei kann es sich um General Packet Radio Service (GPRS), Universal Mobile Telecommunications System (UMTS) oder Wireless Local Area Network (W-LAN) handeln. Eine Studie des Statistischen Bundesamtes [1] zeigt, dass im Jahre 2010 16 % der Internetnutzer das mobile Internet ihres Mobiltelefons verwenden. Dies sind 78 % mehr als im Vorjahr. Durch eine bessere Netzabdeckung und den Zuwachs an internetfähigen Mobiltelefonen wird diese Zahl in den nächsten Jahren weiter steigen. Es ist zu erwarten, dass auch die Nachfrage nach mobil abrufbaren Informationen steigt.
Im kommenden Abschnitt gehen wir kurz auf ähnliche Forschungsprojekte ein und zeigen wie diese auf ortsbezogene Informationen zugreifen. Der Abschnitt 3 beschäftigt sich mit den aktuellen Techniken, die für eine solche Ortsbestimmung eingesetzt werden. Den Prototyp unserer Applikation beschreiben wir im vorletzten Abschnitt. Dort gehen wir auf die Implementierung und den Performance Test ein. Mit diesem Test zeigen wir, ob es sinnvoll ist alle Daten auf einmal abzurufen und in die Datenbank zu parsen. Im Fazit fassen wir unsere Erkenntnisse kurz zusammen und geben Anstöße für zukünftige Forschungsarbeiten.

2. VERWANDTE ARBEITEN

Die Idee ein ortsbezogenes Informationssystem für den Innenraum zu entwickeln ist nicht neu. Es gibt mittlerweile einige Ansätze dies umzusetzen:
Gallagher et al. [2] stellen in ihrer Arbeit ein Positionierungssystem für den Innen- und Außenbereich vor. Das System verwendet W-LAN zur Positionsbestimmung im Innenbereich. Zum Einsatz kommt eine Technik mit dem Namen *Fingerprint*. Auf dieses Verfahren gehen wir im Abschnitt 3 des Papers näher ein. Für den Außenbereich verwendet Gallagher et al. GPS.
Ein ähnliches Projekt ist zur Zeit am Campus Gießen der Technischen Hochschule Mittelhessen (THM) in der Entwicklung. Auch hier verwenden Weber et al. [3] W-LAN mit *Fingerprint*, allerdings liegt der Fokus auf einer Verbesserung der Genauigkeit. Dies soll durch die Verwendung von Schlitzkabel ermöglicht werden.
Strauß et al. [4] haben sich in ihrer Veröffentlichung mit dem Einsatz von QR-Codes für ortsbezogene Dienstleistungen befasst. Sie zeigen unterschiedliche Möglichkeiten auf, bei denen QR-Codes den Standort bestimmen. Eine der Ein-

satzmöglichkeiten ist das *Mobile Campus Informationssystem*. Es dient zur Orientierung an der TU Graz und enthält Informationen über Dekanate, Institute und Hörsäle. Zusätzlich ist eine Suche von Adressen und Gebäuden möglich. Wir verwenden für unser System keine W-LAN Positionierung, sondern eine Ortsbestimmung mittels QR-Code. Das System der TU Graz verwendet zusätzliche Parameter, die beim Aufrufen der Informations-Webseite übergeben werden. Unsere Applikation verwendet eine interne Datenbank, in der die Rauminformationen gespeichert sind. Der QR-Code enthält die Raumnummer, nach der in der Datenbank gesucht wird.

3. TECHNIKEN ZUR ORTSBESTIMMUNG

W-LAN, QR-Code und Near Field Communication (NFC) sind neben GPS weitere Techniken die einen Zugriff auf standortbezogene Informationen ermöglichen. Diese Techniken werden nun kurz erläutert.

3.1. Global Position System

GPS eignet sich nur für den Außenbereich, da eine metergenaue Bestimmung in Gebäuden nicht möglich ist. Dies liegt an dem stark reduzierten GPS-Empfang. Die Ursache ist das Dämfpungsverhalten der verwendeten Baustoffe. In der Nähe von Fenstern erhöht sich der GPS-Empfang, doch die Genauigkeit ist nicht mit der auf einem freien Gelände vergleichbar [5]. Um die aktuelle Position zu berechnen benötigt der Empfänger die Signallaufzeit von mindestens vier Satelliten. Ist dies gegeben, kann eine metergenaue Positionierung stattfinden.

3.2. Wireless Local Area Network

Eine Positionsbestimmung mittels W-LAN ist beispielsweise mit der Received Signal Strenght (RSS) möglich. Hierzu wird die Signalstärke der Access Points verwendet. Das *Radio Frequency (RF) Fingerprinting* Verfahren basiert auf RSS und ist in zwei Phasen aufgeteilt:

Offline-Phase: An unterschiedlichen Positionen wird die RSS von den verfügbaren Access Points ermittelt und in eine Datenbank gespeichert. Daraus ergibt sich eine *Radio-Map*. Bei der Messung müssen verschiedene Einflussfaktoren berücksichtigt werden. Um die Einflussfaktoren zu minimieren, ist es sinnvoll Mehrfachmessungen durchzuführen und diese zu mitteln [6].

Online-Phase: Die aktuelle Signalstärke wird mit den Werten aus der Radio-Map verglichen. Die Position lässt sich aus den RSS-Werten bestimmen, deren Entfernung am geringsten zu der angegeben Signalstärke ist.

Die Genauigkeit schwankt und ist unter anderem von den folgenden Faktoren abhängig:

- Menge der sichtbaren Access Points
- Qualität der *Radio-Maps*
- Störungen des W-LAN Signals (Wände, Menschen)

An der THM im Fachbereich Elektro- und Informationstechnik wird zur Zeit an einem Projekt mit dem Namen *Wirless Indoor Positioning System (WIPoS)* gearbeitet. Das Ziel ist es, zu verhindern, dass durch eine Umgebungsänderung ein erneutes kalibrieren der Radio-Map nötig ist. Hierzu verwendet das WIPoS-Team sogenannte Schlitzkabel um für eine gleichmäßige Signalstärke zu sorgen [3].

3.3. Near Field Communication

NFC ist eine drahtlose Übertragungstechnik zum Austausch von Daten. NFC-Geräte können sowohl aktiv-passiv als auch aktiv-aktiv kommunizieren. Im passiven Modus (aktiv-passiv) erzeugt nur der Sender ein Radiofrequenz (RF)-Magnetfeld und der passive Empfänger zieht aus diesem Feld seine Energie. Im aktiven Modus (aktiv-aktiv) erzeugen sowohl Sender als auch Empfänger ein RF-Magnetfeld. Die Reichweite beträgt ca. 10 cm [7].

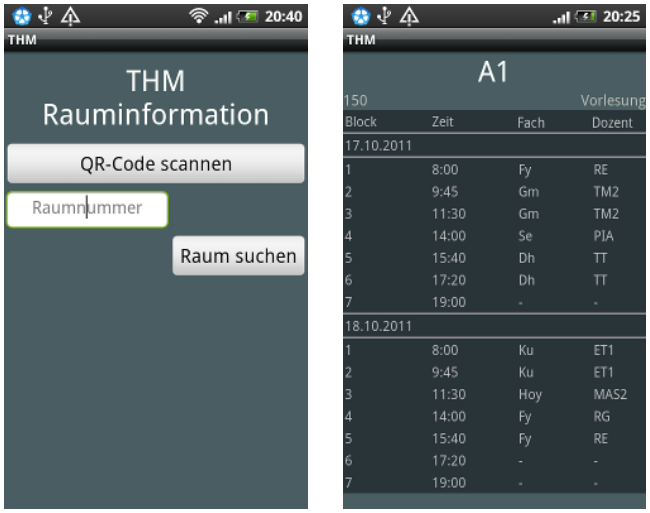
3.4. QR-Code

Ein QR-Code ist ein zweidimensionaler Strichcode, der aus einer gepunkteten quadratischen Matrix besteht. Die kodierten Daten werden binär dargestellt und können auch dann noch ausgelesen werden, wenn der Code bis zu 30 % zerstört wurde. In drei Ecken enthält der QR-Code ein Muster, das zur Orientierung dient [8].
Es können bis zu 250 Zeichen verschlüsselt gespeichert werden. Der QR-Code kann mit einem Lesegerät (zum Beispiel Mobiltelefon) fotografiert und mittels Software dekodiert werden.

4. ENTWICKLUNG DER APPLIKATION

Für die Implementierung unserer Applikation verwenden wir das Smartphone-Betriebssystem Android in der Version 2.3.3. Diese Version ist derzeit aktuell und mit 38.2 % Marktanteil weit verbreitet [9].
Die Abbildung 1 zeigt einen Screenshot des Startbildschirms. Wie in Abbildung 1 (a) zu sehen ist, gibt es zwei Möglichkeiten auf die Rauminformationen zuzugreifen. Die erste Möglichkeit ist das Scannen eines QR-Codes. Der zweite Weg ist die manuelle Eingabe der Raumnummer. Anschließend wird die Raumnummer in der Datenbank gesucht. Je nach Raumart wird ein anderer Bildschirm aufgerufen. Handelt es sich um einen normalen Vorlesungsraum, zeigt die Applikation neben dem Raumnamen und der Anzahl der Plätze auch die Belegung des Raumes an (siehe Abbildung 1 (b)). Hier erfolgt eine geordnete Darstellung nach Datum. Der Anwender kann

auf einen Blick sehen, in welchem Block der Raum von einer Veranstaltung belegt wird.



(a) Startbildschirm (b) Raumdetails

Fig. 1. Screenshots der Applikation.

Für des Erzeugen des QR-Codes haben wir die Webseite qurify [2] verwendet. Diese Seite ermöglicht das Einstellen der Qualität und Größe des zu erzeugenden QR-Codes.

4.1. Implementierung

Startet der Anwender die Applikation, wird der aktuelle Raumplan des Campus Friedberg geladen. Die Applikation parst den Raumplan und speichert diesen in eine SQLite Datenbank. Hierfür wird der Parser der freien Java-Bibliothek jsoup [1] verwendet. Dieser Parser ist einfach aufgebaut, leicht zu verwenden und verfügt über eine gute Dokumentation. Um sicher zu gehen, dass die Rauminformationen schnell abgerufen werden, haben wir die Geschwindigkeit gemessen. Wir wollten herausfinden ob, es sinnvoll ist, den kompletten Raumplan zu parsen oder ob dies die Applikation unnötig verlangsamt. Die Ergebnisse sind in Abschnitt 4.3 aufgeführt. Zu Beginn hatten wir mit der Datenbank *Couchbase Mobile* experimentiert. Dies ist eine mobile Version der NoSQL-Datenbank CouchDB. Zur Zeit liegt Couchbase Mobile nur als Developer Preview vor. Couchbase ermöglicht eine Synchronisation zwischen lokaler und externer Datenbank, diese Funktion wollten wir für unsere Applikation nutzen. Das Einbinden der Bibliothek wurde allerdings durch eine mangelhafte Dokumentation dieses Prozesses undurchführbar. Couchbase Mobile ist in Zukunft sicherlich eine gute Alternative zu den üblichen SQL-Datenbanken.
Durch die Probleme während der Entwicklung, haben wir uns gegen Couchbase und für SQLite entschieden. SQLite ist eine Datenbank, die sehr häufig auf mobilen Geräten eingesetzt wird. Dies liegt neben der geringen Größe auch an der schnellen Implementierung. Android stellt für die Implementierung von SQLite eine Hilfsklasse zur Verfügung. Die *SQLiteOpenHelper* Klasse enthält Methoden, in denen das Erstellen und Aktualisieren einer Datenbank implementiert ist.

[2] http://www.qurify.com/de/
[1] http://jsoup.org/

	Galaxy S2	Desire Z
CPU	Dualcore ARM Cortex A9	ARM Cortex A8
Taktrate	1200 MHz	800 MHz
Arbeitsspeicher	1024 MB	512 MB
Speicher (intern)	16 GB	1,5 GB

Fig. 2. Spezifikation der Testgeräte.

4.2. Testumgebung

Um sicher zu gehen, dass die Performance bei Geräten mit unterschiedlicher Leistung ähnlich ist, haben wir uns bei den Tests für zwei verschiedene Smartphones entschieden. Die Testgeräte waren das Samsung Galaxy S2 und das HTC Desire Z. Beide Geräte verwenden Android 2.3.3. In der Tabelle 2 sind technische Spezifikationen zu finden. Mit der Applikation TrafficStats Lite [1] haben wir den Datentraffic gemessen. Die Tests erfolgten mit W-LAN Netz und mit UMTS.

4.3. Ergebnisse

Zwischen den beiden Smartphones konnte ein Unterschied von bis zu 10 Sekunden festgestellt werden. Wie in Abbildung 3 zu sehen ist, dauert das Parsen der Rauminformationen auf dem HTC Desire Z mit W-LAN Verbindung 49,5 Sekunden. Mit einer UMTS Verbindung wartete der Anwender 44,5 Sekunden. Das Samsung Galaxy S2 brauchte mit einer W-LAN Verbindung 58,1 Sekunden für das Parsen und mit UMTS 53,3 Sekunden.

5. FAZIT

Die gemessenen Ergebnisse zeigen, dass das Parsen mit über 40 Sekunden viel zu lange dauert. Somit ist ein schneller Zugriff auf die Rauminformationen nicht gewährleistet. Diese starke Verzögerung entsteht durch die stark verschachtelte Tabelle des Raumplanes. Trotz der langen Zugriffszeit konnte man anschließend problemlos auf die Rauminformationen zugreifen. Der Anwender konnte sich informieren, ob und mit welchem Kurs der Raum belegt ist.
Solch eine Applikation hat sehr viel Potential, nicht nur den Inhalt betreffend. Es könnten neben dem Stundenplan, einer

[1] http://goo.gl/tIInt

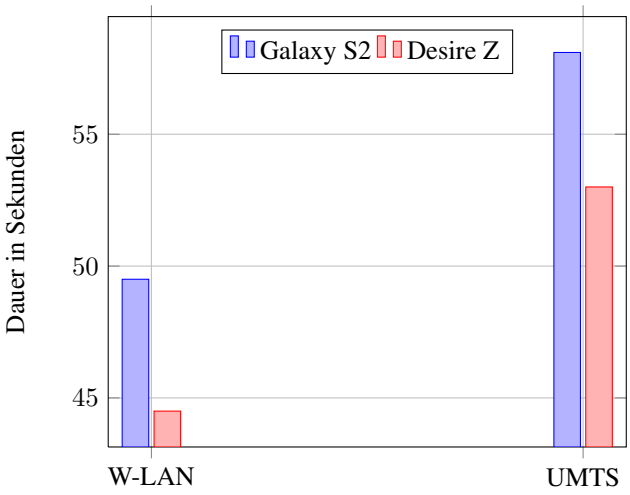


Fig. 3. Dauer des Parsing-Vorgangs.

virtuellen Karte und dem Speiseplan der Mensa noch viele andere Informationen angezeigt werden. Doch auch im technischen Bereich ist solch eine Applikation ausbaufähig. Verbessert sich die Situation des W-LAN Netzes, indem ein flächendeckender stabiler Empfang gewährleistet wird, könnte man dies als Ortungssystem einsetzen. Hierfür muss eine gewisse Vorarbeit geleistet werden. Denn für ein Ortungssystem via *Fingerprint* sind viele Referenzmessungen nötig, um eine gute Radio-Map zu erstellen.
Zur Zeit verfügt NFC noch über einen sehr geringen Marktanteil. Im Jahre 2010 waren nur 4,1 % der Mobiltelefone NFC-fähig [10]. Laut dieser Studie von iSuppli wird der Marktanteil in den nächsten Jahren rapide ansteigen und somit wäre NFC eine gute Alternative zu QR-Codes.

6. REFERENCES

[1] Statistisches Bundesamt Deutschland, "Mobile Internetnutzung über das Handy 2010 stark gestiegen," `http://goo.gl/KLKWz`, Website, Pressemitteilung, 2011, [Aufgerufen am 28.07.2011].

[2] Thomas J. Gallagher, Binghao Li, Andew G. Dempster, and Chris Rizos, "A sector-based campus-wide indoor positioning system," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2010.

[3] Mark Weber, Ulrich Birkel, Ralf Collmann, and Julia Engelbrecht, "Comparison of various methods for indoor RF fingerprinting using leaky feeder cable," in *Positioning Navigation and Communication (WPNC)*, 2010, pp. 291 – 298.

[4] Clemens Strauß, Johannes Scholz, Martin Ebner, and Paul Schmidmayr, "Einsatz von Quick Response Codes für ortsbezogene Dienstleistungen," in *Konferenzband Geoinformatik*, 2009, pp. 57–63.

[5] Wikipedia, "Global Positioning System," `http://goo.gl/e43C2`, Webseite, Aufgerufen am 08.08.2011.

[6] Gunnar Jehl, "Deterministic WLAN Position Determination," `http://goo.gl/9gkJW`, PDF, 2007, Aufgerufen am 20.08.2011.

[7] areamobile.de, "NFC: Near Field Communication," `http://goo.gl/WBJzz`, Webseite, Aufgerufen am 21.08.2011.

[8] Wikipedia, "QR-Code," `http://goo.gl/lAKxY`, Webseite, Aufgerufen am 22.08.2011.

[9] Android Developer, "Platform Versions," `http://goo.gl/OdZfb`,Webseite, 14. 10. 2011.

[10] iSuppli Corporation, "Cell Phone Mobile Payment Market Set for Take Off," `http://goo.gl/ufrCj`, Webseite, Pressemitteilung, 2010, Aufgerufen am 28.08.2011.

EVALUATION AND ANALYSIS OF NATURAL GESTURES FOR MULTI-TOUCH ENABLED GEOGRAPHIC INFORMATION SYSTEMS

Andreas Fiedler, Lars-Olof Krause

Department IEM - Media Informatics
Technische Hochschule Mittelhessen - University of Applied Sciences,
Wilhelm-Leuschner-Str. 13, 61169 Friedberg

ABSTRACT

In this paper we face the question "What are intuitive gestures?" for using geographic information systems on a multi-touch enabled device. We took a look at how others went at designing multi-touch gestures and found most gesture sets to be defined by developers and not by observing the future users. We conducted an online survey to get to know which geographic information systems are used and what they are used for. With this data we developed and performed a Wizard of Oz like Hands on Test to identify gestures for basic and advanced tasks. The gestures most used were compiled into a gesture set of our own.

Index Terms— Human computer interaction, Geographic Information Systems

1. INTRODUCTION

Multi-touch enabled systems are on the rise, they are used in more and more areas of daily life. Modern smartphones with multi-touch displays are available to a growing number of people. But the used gestures to control the system are still mainly created by developers and designers and need to be taught to the average users. The resulting gestures were designed for recognizability on technical bases and not around how an unlearned user would try to accomplish a specific purpose. There are relatively few scientific papers dealing with the actual development of intuitive gestures for touch surfaces by observation of users. Most of the time they are just defined and called natural without any proof. In this paper, we will try to find commonly used gestures for controlling geographic information systems on an multi-touch table.

2. RELATED WORK

There are different approaches on dealing with multi-touch systems, more precisely defining of gestures for them.

2.1. Wizard of Oz studies

Beringer used a Wizard of Oz test to collect gesture data for the training of a gesture analyzer. The evaluation of these tests showed that over 1/3 of the gestures where pointing gestures with an arbitrary number of fingers [1]. In the Wizard of Oz study of Voida et al. the subjects were asked to define gestures and voice commands to manipulate 2D-Objects in an augmented reality environment. The preferred gestures were pointing gestures with minimal body movement, similar to using common pointing devices like the PC mouse [2]. The result of the mentioned studies is that users mainly point at elements of the multi-touch system. This behavior seems to be a result of the widespread usage of WIMP-Systems (window, icon, menu, pointing devices) like Microsoft Windows, Linux with GUI, Mac OS, etc.

2.2. Defining Gestures

Many developers just define some multi-touch gestures without proper research on how the user would naturally act to solve a task. As example we refer to the following papers: Kim et al. designed a multi-touch system for google earth with different one- and two-hand gestures. These gestures were predefined by the developers. They didn't state how the gestures were chosen at all [3]. Yan et al. defined "four simple and natural interactive gestures" for their multi-touch system Mimosa [4]. It is not stated why the gestures are called natural in this paper. Other developers define the multi-touch gesture but also perform usability evaluations e.g.: Wu et al. designed an multi-touch gesture set and did usability studies to evaluate the ability of the subjects to understand, learn and execute the example gestures [5].

2.3. Observation

From our point of view the most suited way to develop natural multi-touch gestures is to observe the future user solving tasks common in the system to be. The observation could be in form of a Wizard of Oz test, but also just solving tasks with real objects which will have digital representations in the

system later on. Liu et al. observed papers been passed on a table between several persons and found the movement to be a combination of translation and rotation. They then mimicked this behavior in the gestures they developed [6]. Artinger et al. created five different gesture sets and compared them by usability test with common users. The tested functions were pan, zoom, rotate and select. The Free and the Default gesture set were rated highest. In the Default set the pan action is done by one finger flick / sticky finger, the zoom action by pinching two fingers, the rotate action by rotation two fingers around their common center. In the Free set the actions are the same as in the Default set but they can be used with up to five fingers on each hand [7]. Wobbrock et al. showed different effects to the test subjects through animation of simple objects and asked them how these effects could be accomplished with a gesture. The subjects were asked to do a gesture with one and two hands separately. As result they created a gesture set from the most common gestured used by the test subjects [8]. In the referred paper you can find the exact description and visualization of the developed gesture set.

3. METHODS OF EVALUATION

To evaluate or find natural multi-touch gestures for using an geographic information system we decided to create an online survey as first step. With the data we received we created an Hands on Test similar to an Wizard of Oz test to elicit gestures for the most common actions.

3.1. Online Survey

The online survey based on the open source survey system LimeSurvey [9] which was installed on our own server. The participants were found by asking contacts of our social networks to take part in the survey and refer others to it. The survey was into three parts dealing with the knowledge and usage of geographic information system, multi-touch systems and a part about the persons affinity towards computer and new technologies. The part about geographic information systems contained questions about which systems are known, which are used and how often and how intuitive they seemed. The part about multi-touch systems contained questions about how often these systems are used, how intuitive the usage is and if geographic information systems are used on them. The personal questions contained general questions about gender and age, a set of questions about the computer knowlege and usage, and a block of questions about how they feel about and deal with new technologies.

3.2. Hands on Test

For the Hands on Test we used the multi-touch table of our university. The table is based on the Frustrated Total Internal Reflection (FTIR) technology as described by Han [10].

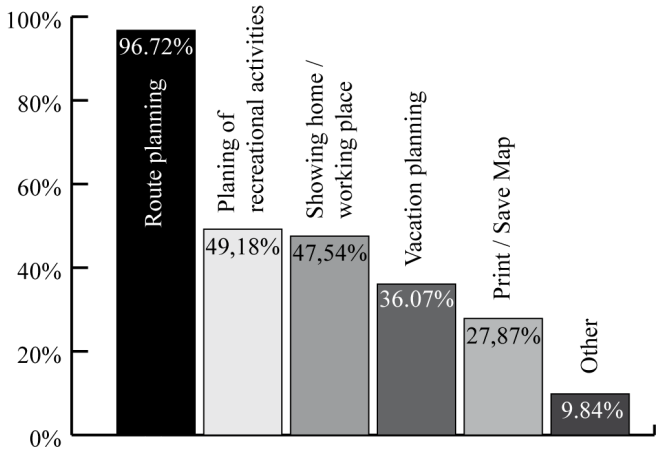


Fig. 1: Usage of geographic information systems.

The test system was realized using the Google Earth browser plugin with custom javascript controls using the Google Earth API [11], the npTuioClient [12] and the TUIOjs [13] implementation in an installation of FireFox 3.6. The user actions at the multi-touch table itself caused no actions in the system except displaying the touch-points as dots on the screen. Similar to a Wizard of Oz test all actions in Google Earth were simulated by us using our custom controls. The difference of the test to a Wizard of Oz test is that the user was told that we will simulate the actions. Through the online survey we got to know what most of the participants do with geographic information systems. From this data we created tasks for the Hands on Test. The users were asked to perform these tasks thinking aloud and using any methods they thought would yield the tasked result. The participants were divided into three groups with slightly different test setups. One group had just the map to work with for all tasks except if they asked for a specific tool. Another group had tool icons like a compass rose and a measuring scale displayed from the start. The last group was asked to perform basic tasks on a globe first. Some of the general tasks where to navigate to and show the subjects home, to find different places, to find out what roads they would take from one city to another, to rotate the view, to measure distances and finally to orient the view northwards. These tasks consists of several elemental gestures namely move, zoom, rotate and activating special functions.

4. RESULTS

In this section we will present the major results of the online survey and Hands on Test.

4.1. Online Survey

After advertisement on various social networks our online survey was completed by 61 participants. The survey showed the group mainly consisted of advanced and professional

computer users and 67.21% had prior experience with multi-touch enabled systems. The geographic information system second most used is Google Earth, it is used by 24.59% of the participants and 100% have heard about it or tried it. The geographic information system most of the participants use is Google Maps, it is used by 86.89% and 77.05% find its use fully intuitive. Our survey showed that the main usage of geographic information systems lies in planning of routes, showing where you live and work and planning of leisure time activities as shown in the diagram (Fig. 1).

4.2. Hands on Test

For the Hands on Test we had 15 average to advanced computer users as participants, 5 per group, although the results show that there were no significant differences between the groups. The tests showed that for basic action all participants used similar gesture. For the more complex functionalities there were no explicit gestures. The gesture to move the earth/map is a slide (touch the screen and move the finger over the surface) of one or more fingers in the direction the earth was meant to move (Fig. 2a) The action to zoom-in is subdivided into two gestures, they are Double-Tab for a jump-zoom (Fig. 2g) and pinch apart for a normal zoom (Fig. 2b). The pinch gesture was prevalent for zooming out (Fig. 2c). Both pinch gestures were performed with two or more fingers. To rotate the participants used two or more fingers to make a circular movement around the center of the fingers (Fig. 2d) or a circular movement of one finger around the other (Fig. 2e). In every instance the circular movement was in the direction of the intended rotation of the map. As special function we asked for route planning and distance measurement. Most participants were clueless how to do route planing without context menus, text masks or wizards. Some just traced their route with the finger, others asked for icons to use. Of the users that were given route planing icons from the start only a few used them by placing them on start and target. The users who used the rout icons dragged or clicked (multiple single taps) the icons in position (Fig. 2f). To measure a distance most users requested a scale if not present. The way they wanted to use or create it ranged from context menus to dragging and rescaling the scale icon (Fig. 2i). A general way of opening a context menu was pressing the screen for a longer duration (Fig. 2h).

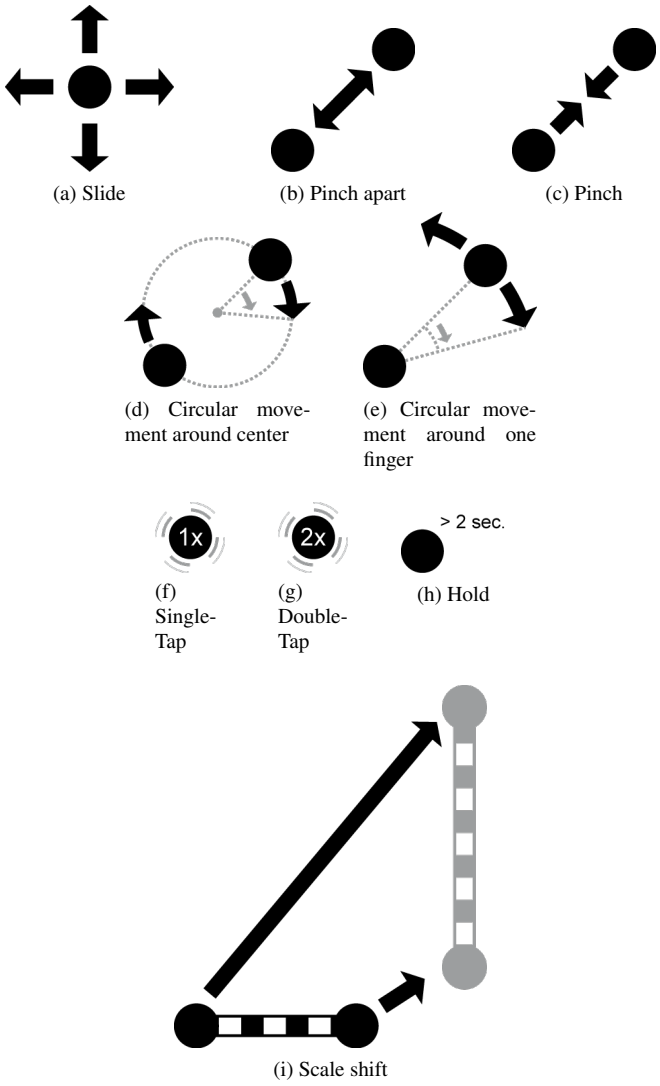


Fig. 2: Result gesture set with the gestures most used.

5. CONCLUSION

The result gesture set for the simple tasks was quite clear among most of the participants. The gestures used and how the participants used them gave us the feeling that they had at least seen them used somewhere before e.g. in advertisements. The variations of the result gestures only differed in the number of fingers that are used. The variations of the found gesture set should be easy to implement in a gesture recognition software. For the advanced features the result was not that clear, we recommend the usage of a so called touch-ghost [14], so the user is guided through the usage of special functions, when he performs actions which can not be recognized. As further research an implementation of the gesture set should be created and evaluated in a bigger group to analyze if they are accepted or have to be slightly modified. Another course of action could be to create a big testing scenario with subjects with virtually no experience with computers and multi-touch systems to eliminate these influences.

6. REFERENCES

[1] N. Beringer, "Evoking gestures in smartkom - design of the graphical user interface," in *Revised Papers from the International Gesture Workshop on Gesture and Sign Languages in Human-Computer Interaction*, ser. GW '01. London, UK: Springer-Verlag, 2002, pp. 228–240. [Online]. Available: http://dl.acm.org/citation.cfm?id=647592.728738

[2] S. Voida, M. Podlaseck, R. Kjeldsen, and C. Pinhanez, "A study on the manipulation of 2d objects in a projector/camera-based augmented reality environment," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '05. New York, NY, USA: ACM, 2005, pp. 611–620. [Online]. Available: http://doi.acm.org/10.1145/1054972.1055056

[3] J. Kim, J. Park, H. Kim, and C. Lee, "Hci(human computer interaction) using multi-touch tabletop display," in *Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on*, aug. 2007, pp. 391 –394.

[4] Q. Yan, Y. Wu, Z. Liu, and H. Zhang, "Mimosa: A multi-touch system used for virtual exhibition in public spaces," in *Industrial and Information Systems (IIS), 2010 2nd International Conference on*, vol. 1, july 2010, pp. 491 –494.

[5] M. Wu, C. Shen, K. Ryall, C. Forlines, and R. Balakrishnan, "Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces," in *Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on*, jan. 2006, p. 8 pp.

[6] J. Liu, D. Pinelle, S. Sallam, S. Subramanian, and C. Gutwin, "Tnt: improved rotation and translation on digital tables," in *Proceedings of Graphics Interface 2006*, ser. GI '06. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2006, pp. 25–32. [Online]. Available: http://dl.acm.org/citation.cfm?id=1143079.1143084

[7] E. Artinger, M. Schanzenbach, F. Echtler, S. Nestler, T. Coskun, and G. Klinker, "Beyond pinch-to-zoom: Exploring alternative multi-touch gestures for map interaction," München, 2010, technische Universität München, Report TUM-I-08-19.

[8] J. O. Wobbrock, M. R. Morris, and A. D. Wilson, "User-defined gestures for surface computing," in *Proceedings of the 27th international conference on Human factors in computing systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1083–1092. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518866

[9] Limesurvey. Version 1.91 Build 10030. [Online]. Available: http://www.limesurvey.org

[10] J. Y. Han, "Low-cost multi-touch sensing through frustrated total internal reflection," in *Proceedings of the 18th annual ACM symposium on User interface software and technology*, ser. UIST '05. New York, NY, USA: ACM, 2005, pp. 115–118. [Online]. Available: http://doi.acm.org/10.1145/1095034.1095054

[11] Google earth api. Google Inc. 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States. Version: 1. [Online]. Available: http://code.google.com/intl/en/apis/earth/

[12] nptuioclient. Amsterdam. Version: 0.2-win. [Online]. Available: https://github.com/fajran/npTuioClient

[13] (2009, Jun.) Tuiojs. Amsterdam. [Online]. Available: https://github.com/fajran/tuiojs/

[14] D. Vanacken, A. Demeure, K. Luyten, and K. Coninx, "Ghosts in the interface: Meta-user interface visualizations as guides for multi-touch interaction," in *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, oct. 2008, pp. 81 –84.

EINSATZ VON HEAD TRACKING IN AUTODESK MAYA ZUR ERWEITERTEN 3D-WAHRNEHMUNG

Peter Salziger

Technische Hochschule Mittelhessen, University of Applied Sciences, Friedberg

ABSTRACT

Dieses Paper beschäftigt sich mit der Integration und dem Einsatz von Head Tracking in der 3D Visualisierungssoftware Autodesk Maya. Mittels einer handelsüblichen Webcam kann der Nutzer den Viewport durch seine Kopfpositionierung und Rotation steuern. Dabei erhält er nicht nur eine dreidimensionale Sicht auf den 3D-Raum, sondern genießt u.a. Vorteile beim Modellieren hinsichtlich der Usability. Entscheidend ist, dass er für den so erlangten räumlichen Eindruck keine zusätzliche Hardware benötigt, wie z.B. spezielle 3D Brillen, Monitore und Softwareanpassungen.

Index Terms— head tracking, webcam, user interface, stereoscopy, autodesk maya, FaceAPI

1. EINLEITUNG

Räumliches Sehen, vor allem Stereoskopie, hat in den letzten Jahren eine Renaissance erlebt. Neben dem Einzug der Technologie in die Kinos, nutzen auch immer mehr Privatanwender die neuen Möglichkeiten zu Hause, vor allem vor den neuen 3D-Fernsehern. Das Hauptproblem ist jedoch immer noch, dass man für die korrekte Darstellung zusätzliche Hardware braucht. Zur Standardausrüstung gehört hier die 3D-Brille mit deren Polarisationsfiltern zwei verschiedene Bilder auf die Netzhäute projiziert werden, vorausgesetzt das Wiedergabemedium unterstützt diese Technologie. Neben den hohen Anschaffungskosten wird die zu tragende Brille meist als störend empfunden.

Auch in die 3D-Entwicklungssoftware Autodesk Maya[1] hat die Technologie der Stereoskopie Einzug gefunden. Primär wird sie jedoch dazu verwendet, um stereoskopische Ausgabebilder zu erzeugen. Der Entwickler, welcher die 3D Szenen generiert, genießt während der Produktion keine Vorteile in Sachen Usability. Zwar kann er sich während der Produktionsphase eine stereoskopische Ausgabe anzeigen lassen, um die Tiefenwirkung mit Hilfe einer 3D-Brille abschätzen zu können bzw. zu justieren, jedoch unterstützt ihn der räumliche Tiefeneindruck nicht bei seiner eigentlichen Arbeit, dem Gestalten der Szene (z.B. Modellierung). Des Weiteren stört auch hier die Verwendung einer 3D-Brille beim Arbeiten.

Zu den Anforderungen dieser Arbeit gehört schließlich die Entwicklung eines Prototypen, welches den Nutzer ohne die Verwendung von teurer Zusatzhardware einen räumlichen Eindruck vermitteln lässt. Ferner soll der Nutzer mit seiner Kopfbewegung auch den Blickwinkel auf den 3D-Raum steuern können, so als würde er sich selber in diesem Raum befinden. Als positiven Nebeneffekt hat er mehr Freiraum beim Modellieren, da er die Positionierung der virtuellen Kamera nicht mehr über Tastatur und Maus steuern muss, sondern über eine intuitive Kopfbewegung. Seine eigene Kopfbewegung und -Rotation soll hierbei von einer handelsüblichen Webcam erfasst werden ("Head Tracking").

2. VERWANDTE ARBEITEN

Es gibt bereits mehrere Projekte, welche eine Head Tracking Technologie verwenden, um in ihrer Applikation dem Nutzer eine Steuerung über seine Kopfbewegungen ermöglichen. Dabei verwenden alle Projekte die sogenannte FaceAPI[2] von Seeing Machines. Mit Ihr ist es möglich die Koordinaten eines von der Webcam erfassten Kopfes auszulesen. Um die Erkennungsalgorithmen braucht man sich nicht zu kümmern.

2.1. Head Tracking in First-Person Games[3]

Torben Skol et al. haben eine Head Tracking Technologie in einen First Person Egoshooter eingebunden. Dazu haben sie die Source Engine von Valve modifiziert. In der Evaluation empfanden die Probanden die "Handy-Cam"-Umsetzung am interessantesten. Hierbei wird die Kopfbewegung direkt auf die virtuelle Kamera gemappt und somit ein realistisches Bewegungsgefühl vermittelt.

2.2. Human Machine Interaction[4]

Simon-Alexander Zerawa et al. steuern einen virtuellen Fahrstuhl über vordefinierte Kopfgesten. Mit einer Kopfbewegung nach oben bzw. nach unten wird das gewünschte Stockwerk ausgewählt und mit einer Bewegung nach links wird das gewünschte Stockwerk angefahren.

2.3. Maya Stereoskopie

Autodesk Maya verwendet im Viewport zur Darstellung eines räumlichen Eindruckes die farbanaglyphische Technik. Hierbei werden die zwei perspektivischen Halbbilder gleichzeitig dargestellt und in komplementären Farben eingefärbt. Zur Trennung der beiden Bilder werden Brillen mit Rot/Grün-Filter verwendet. Der Nutzer muss weiterhin die Tastatur und Maus verwenden, um z.B. um ein Objekt herum zu schauen.

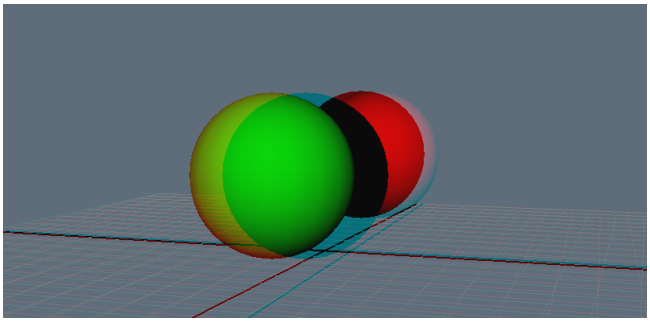


Fig. 1. Maya Stereo Anaglyphen Modus

3. DER HEAD TRACKER

3.1. Anforderungen

Ziel war es, einen Head Tracker zu finden, welcher akkurat und ressourcensparend den Livestream einer Webcam verarbeitet. Zudem muss er eine API auf C++ Basis zur Verfügung stellen, um ihn in eine eigene Applikation einbinden zu können. Positionskoordinaten und Rotationswerte des zu erfassenden Kopfes müssen während dem Betrieb ausgelesen werden können. Diese Anforderungen erfüllt die FaceAPI von Seeing Machines und wurde bereits auch in den verwandten Projekten erfolgreich eingesetzt. Fig. 2. veranschaulicht den Algorithmus.

Fig. 2. FaceAPI Tracker

Laut der Dokumentation von Seeing Machines erfasst der Head Tracker eine Kopfrotation auf der Y und Z-Achse zwischen 90 und 90 Grad, sowieso eine Rotation auf der X-Achse zwischen -30 und 60 Grad.

3.2. API Zugriff

Die FaceAPI gibt dem Entwickler komfortablen Zugriff auf alle wichtigen Parameter. Wie in Fig. 3. zu sehen, erhält man Zugriff auf eine Variable, welche neben den Positions- und Rotationswerten auch ein Konfidenzintervall enthält. Dieses kann Werte zwischen 0 und 1 annehmen und ist ein wichtiges Indiz für die Qualität der erhaltenen Koordinaten.

```
struct smEngineHeadPoseData{
    float head_pos.x;
    float head_pos.y;
    float head_pos.z;
    float head_rot.y_rads;
    float head_rot.x_rads;
    float head_rot.z_rads;
    float confidence;
}
```

Fig. 3. C++ Übergabedeklaration

Würde dieser Wert die Zahl 0 annehmen, sind weitere Verarbeitungsschritte zu stoppen, da der Tracker derzeit keinen Kopf erkennt.

4. IMPLEMENTIERUNG

Um die Technologie in Autodesk Maya in Form eines Plugins zu integrieren, müssen zwei verschiedene Komponenten programmiert werden. Die eine Komponente stellt den Kern des Systems dar, integriert die FaceAPI und beeinflusst eine virtuelle Kamera im Maya Viewport. Die zweite Komponente äußert sich in einer grafischen Oberfläche, welche in Maya angedockt wird. Mit ihrer Hilfe, kann der User die zu beeinflussende Kamera auswählen und die Verarbeitung starten und stoppen.

4.1. Maya Plugin - Die Engine in kompilierter Form

Da die FaceAPI nur eine Implementierung über C/C++ zulässt, bedarf es hier die Entwicklung eines nativen Plugins für Maya. Dies wurde in Microsoft Visual Studio 2008[5] entwickelt und ist hauptsächlich dafür zuständig, die von der FaceAPI übertragenden Positionswerte auf eine zuvor definierte Kamera im Maya Viewport zu mappen. Hierbei wurde das Offset-Mapping verwendet. Die Positionswerte werden nicht absolut auf die Maya Kamera übertragen, sondern relativ zu den bestehenden Positionswerten hinzu addiert. Dies ermöglicht dem Maya Benutzer, sich weiterhin wie gewohnt per Maus und Tastatur durch die Szene zu navigieren. Seine Kopfbewegungen werden zusätzlich auf die manuelle Bewegung addiert.

4.2. Grafisches Userinterface

Der Benutzer soll die Kontrolle darüber haben, welche Kamera in der Szene sich seiner Kopfbewegung anpasst. Zudem soll er die Verarbeitung manuell starten und stoppen

können. Eine grafische Benutzeroberfläche wird in Maya mit der Scriptsprache MEL realisiert. Ein bereits fertiges Script lässt sich an die übliche Menüstruktur andocken, um es später leicht finden zu können. Das MEL-Script ruft dann das eigentlich verarbeitende Plugin mit den entsprechenden Parametern auf (Kamera, Start/Stop).

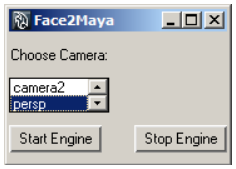


Fig. 4. Maya Mel GUI

5. ERGEBNIS

Durch die Implementierung der FaceAPI ist es gelungen, die Illusion eines dreidimensionalen Raumes hinter dem 2D Bildschirm zu erschaffen. Fig. 5. verdeutlicht noch einmal das Ergebnis, in dem der Nutzer nun in der Lage ist, durch eine intuitive Kopfbewegung hinter Objekte zu schauen.

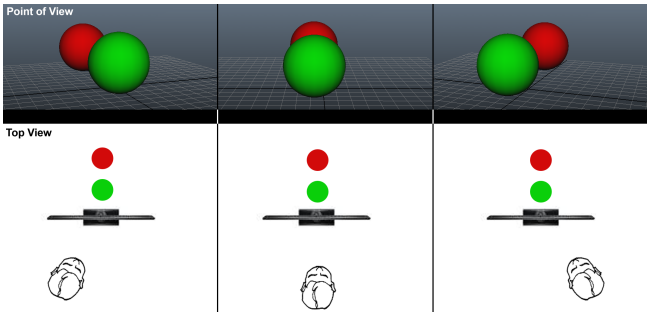


Fig. 5. Demonstration der Head Tracking Implementierung

Im Folgenden wird ein Szenario vorgestellt, in welchem die Technologie in Bezug auf die Usability-Aspekte Vorteile erbringen kann. Es gibt beim Modellieren Szenarien, in welchen man die Perspektive nicht per Maus und Tastatur verschieben kann. Modelliert man z.B. eine Bézierkurve[6] und ist gerade dabei einen Punkt zu setzen, welcher bereits sehr nahe am Bildschirmrand und sogar über den Bildschirmrand hinaus platziert werden soll, muss man ihn zwangsläufig kurz vor dem Bildschirmrand platzieren, da man in diesem Moment die Perspektive nicht verstellen kann. Mit aktivem Plugin reicht es jedoch aus, seinen Kopf leicht zu drehen und sich somit mehr Raum zu verschaffen (vgl. Fig. 6.).

Generell gibt es während dem Modellieren viele denkbare Szenarien, in welchen man sich mit einer kurzen Kopfbewegung mehr Überblick verschaffen kann. Typische Use-Cases wären hier noch weiterhin zu erforschen.

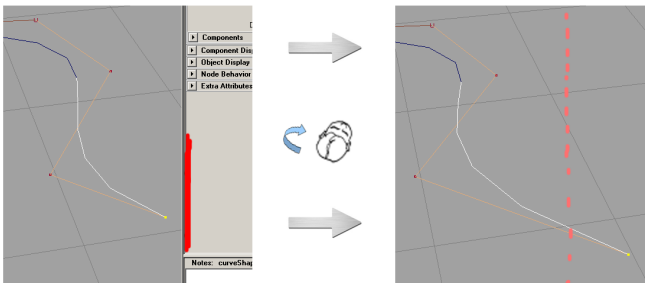


Fig. 6. Usability Vorteile

6. AUSBLICK

Die FaceAPI von Seeing Machines erfreut sich einer immer größeren Beliebtheit und wird in immer mehr Applikationen eingebunden. Die Einbindung in Autodesk Maya auf prototypischer Basis konnte in dieser Ausarbeitung verwirklicht werden. Im Moment ist von der FaceAPI jedoch nur eine 32 Bit Version verfügbar, was zur Folge hat, dass sie letztendlich als 32Bit Plugin kompiliert und nur in einer 32 Bit Maya Version ausgeführt werden kann. Vermutlich wird Seeing Machines in naher Zukunft auch eine 64 Bit API zur Verfügung stellen, welches lediglich ein erneutes Kompilieren unter Visual Studio zur Folge hätte, um das Plugin unter Maya 64 Bit zum Ausführen zu bringen.

Ein weiterer Aspekt, den es zu optimieren gilt, ist das Grundrauschen, welches die API in den übergebenden Koordinaten erzeugt. Sicherlich ist dies abhängig von der Qualität der Webcam und der Umgebungsbeleuchtung, dennoch zeigt Fig. 7. die Koordinatenänderung im zeitlichen Verlauf beim Versuch, den Kopf ruhig zu halten.

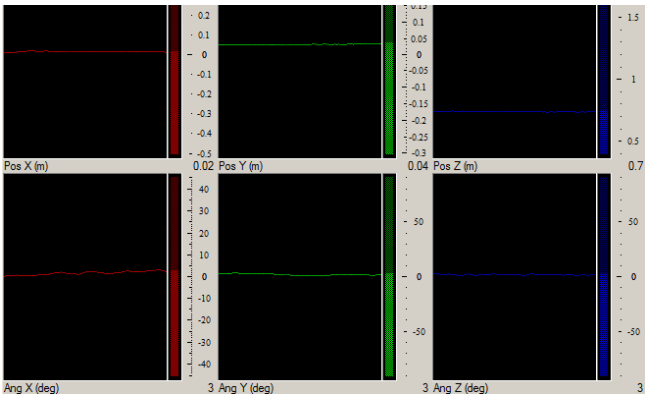


Fig. 7. Rauschen in den Koordinaten

Folglich überträgt sich dieses Rauschen auf die virtuelle Kamera in Maya. In dieser Ausarbeitung wurden die Koordinaten vor der Verarbeitung gerundet, dennoch Bedarf es hier die Implementierung von besseren Algorithmen.

Ein weiterer interessanter Effekt wäre denkbar, wenn man die Tracking-Technologie mit der Maya internen Stereoskopie-

Funktion kombiniert. Dies optimiert zusätzlich den Tiefeneindruck.

Aus den bisherigen Ergebnissen entwickeln sich vor allem folgende Fragestellungen:

- Welche Szenarien (Use Cases) beim Arbeiten in Autodesk Maya erfordern bzw. genießen die Unterstützung von Head Tracking?
- Macht es Sinn, die Erkennung von Gesichtsgesten einzubauen, um damit bestimmte Commands aufzurufen? In der kommerziellen FaceAPI Version lassen sich auch Mundwinkel und Augenbrauen tracken.
- Lässt sich das Tracking Verfahren auch als Motion Capturing einsetzen? Man bedenke hierbei die Möglichkeiten bezüglich "lip sync animations".

Durch das qualitativ recht hochwertige Tracking mit Hilfe von alltäglichem Hardwareequiptment (Consumer-Webcams), eröffnen sich schließlich große Möglichkeiten der Mensch-Maschine-Interaktion in Bezug auf die 3D-Software Autodesk Maya. Es können Verfahren entwickelt werden, welche den Umgang mit der Software effizienter und intuitiver gestalten.

7. REFERENCES

[1] "Autodesk Maya," `http://usa.autodesk.com/maya/`, aufgerufen am 01.10.2011.

[2] "Seeing Machines," `http://www.seeingmachines.com/product/faceapi/`, aufgerufen am 08.04.2011.

[3] Torben Sko and Henry J. Gardner, "Head tracking in first-person games: Interaction using a web-camera," *Lecture Notes in Computer Science*, vol. 5726, pp. 342–355, 2009.

[4] Simon-Alexander Zerawa, Stefan Kohlhauser, Charlotte Roesener, and Andreas Perner, "Human machine interaction using head pose estimation," in *Proceedings of 2nd International Conference on Robotics in Education (RiE 2011)*, Roland Stelzer and Karim Jafarmadar, Eds. 2011, pp. 103–108, INNOC - Austrian Society for Innovative Computer Sciences.

[5] "Visual Studio," `http://http://www.microsoft.com/germany/visualstudio/`, aufgerufen am 19.05.2011.

[6] "Bezierkurve," `http://de.wikipedia.org/wiki/Bezierkurve/`, aufgerufen am 23.08.2011.

Technische Hochschule Mittelhessen

Campus Friedberg
Wilhelm-Leuschner-Str. 13
61169 Friedberg

www.thm.de