Ein Natural User Interface zur Interaktion in einem Cave Automatic Virtual Environment basierend auf optischem Tracking

Thomas Jung, Stephan Krohn, Peter Schmidt

HTW Berlin,
Wilhelminenhofstr. 75A, D-12459 Berlin
eMail: t.jung@htw-berlin.de
URL: http://www.f4.htw-berlin.de/~tj

Zusammenfassung. Im Rahmen dieser Arbeit wird ein Natural User Interface (NUI) vorgestellt, das die Interaktion in einem Cave Automatic Virtual Environment (CAVE) ermöglicht. Im Unterschied zu anderen CAVE-Systemen wird der hier beschriebene CAVE durch ein NUI gesteuert. Das NUI basiert zurzeit auf einer Tiefensensor-Kamera der Firma Microsoft („Kinect"). Mit Hilfe dieses NUI wird die betrachterabhängige Darstellung der 3D-Welt (Head Tracking) realisiert. Die marker- und kabellose Interaktion mit unterschiedlichen Anwendungen (3D-Modellierung, 3D-Navigation, Computerspiele) wird diskutiert.

1 Einleitung

CAVE-Systeme [1] werden schon seit den 1990er Jahren zur Erzeugung immersiver Virtueller Umgebungen verwendet, die Interaktion erfolgte dabei zunächst über magnetische Tracking-Systeme (z. B. Polhemus Fastrak [2]). Magnetische Tracking-Systeme haben den Nachteil, dass der Benutzer durch Sensorkabel beeinträchtigt wird. Später wurden vermehrt optische Tracking Systeme in CAVEs eingesetzt (z. B. [3], [4], ArtTrack2 der Firma A. R. T.), dabei müssen die Benutzer jedoch in der Regel mit aktiven oder passiven Markern instrumentalisiert werden.
Natural User Interfaces vermeiden weitestgehend sichtbare Bedienelemente, um die Bedienung damit natürlicher zu gestalten. Beispiele für zweidimensionale NUIs sind die immer weiter verbreiteten Multitouch-Interfaces, dreidimensionale NUIs setzen auf die Erkennung von Gesten. Time-Of-Flight-Kameras (ZCams), die die Entfernungen zu Bildpunkten bestimmen, ermöglichen die markerlose Erkennung von Gesten. Waren ZCams früher aufgrund ihrer hohen Kosten nur einem kleinen Anwenderkreis vorbehalten, gewinnt die Entwicklung neuer Konzepte im Bereich User Interfaces seit der Vermarktung der Kinect-Spielsteuerung durch die Firma Microsoft an Dynamik.
Im Rahmen dieser Arbeit wird untersucht, ob und ggf. wie die Kinect-Spielsteuerung in der CAVE der HTW Berlin für die Steuerung unterschiedlicher Anwendungen eingesetzt werden kann.[1]

[1] Diese Arbeit ist gefördert durch die HTW Berlin

2 Bildgenerierung

In einer CAVE werden auf die den Benutzer umgebenden Projektionsflächen in der Regel Stereobilder projiziert. Die Projektion der 3D-Szene kann jedoch nur für einen Punkt innerhalb der CAVE korrekt erfolgen, von anderen Betrachterstandpunkten aus müssen sich an den Kanten der Projektionsflächen geknickte Linien für den Betrachter ergeben. Aus diesem Grund war schon die erste CAVE [1] mit einer Head-Tracking-Komponente ausgestattet, so dass sich ein Benutzer frei innerhalb der CAVE bewegen konnte, und die Bildgebung passend zu der Position seines Kopfes erfolgte. Es stellt sich nun die Frage, ob die Kinect-Spielsteuerung für ein Head-Tracking eingesetzt werden kann.
Die Kinect-Spielsteuerung basiert auf einer von der Firma Primesense, Ltd. patentierten Technologie Light Coding™, die Tiefenwerte ohne Time-Of-Flight-Messungen bestimmen kann [5]. Da nur Infrarot-Licht in die Szene gesendet wird, ist das System ideal für die besonderen Lichtverhältnisse in einer CAVE geeignet.
Laut Microsoft beträgt der horizontale Öffnungswinkel des Tiefensensors 58,5 Grad bei einer Auflösung von 640 x 480 Pixeln, Tiefenwerte können in Entfernungen von 80cm bis 4m erfasst werden[6]. Obere Schranken für die Genauigkeit der Messwerte lassen sich aus den Dateien zum Microsoft Kinect SDK rekonstruieren[7]. Zurzeit können zusammen mit den Skelettdaten nur Tiefenbilder in einer Auflösung von 320 x 240 übertragen werden. Ein Bit entspricht dabei einer Tiefe von einem Millimeter. Ein 3D-Punkt in Metern lässt sich dann aus den Pixeln des Tiefenbilds wie folgt berechnen:

(1) $z_m = z / 1000$
(2) $x_m = (x - 160) * k * z_m$
(3) $y_m = (120 – y) * k * z_m$
(4) $k = 0.003501$

Die Konstante k hängt dabei von der inversen Brennweite der Kamera und der Auflösung ab. Damit repräsentiert ein Pixel im minimalen Abstand von 80cm in der Breite knapp 3mm und im maximalen Abstand von 4m ca. 14mm. Die reale Auflösung liegt jedoch bedingt durch das Verfahren darunter, Microsoft gibt die Genauigkeit mit wenigen Millimetern im Abstand von 80cm bis zu 5cm im Abstand von 4m an [6].
Die Abtastrate beträgt 30Hz, die durch das Gerät bedingten Latenzen sind unklar. Während die Auflösung und Genauigkeit der Tiefenwerte im Vergleich zu anderen Geräten recht hoch ist, war praktisch zu untersuchen, ob Blickfeld und Abtastrate der Kamera für den Einsatz in einer CAVE genügen.

3 CAVE der HTW Berlin

Die CAVE der HTW hat eine Grundfläche von 3m x 3m, die Projektionswände sind ca. 2,3m hoch. Es wird auf drei Seitenwände in stereo projiziert, zusätzlich gibt es eine Bodenaufprojektion in mono. Für die Positionierung der Kinect kommen deshalb nur Standpunkte oberhalb von 2,3m in Frage. In Abbildung 1 ist der begehbare Raum der CAVE aus Sicht der Kinect dargestellt, wenn sie in der Mitte oberhalb der vorderen Wand mit einem Neigungswinkel von ca. 30 Grad nach

unten ausgerichtet ist. In Abbildung 2 ist das vom Gerät erzeugte passende Tiefenbild dargestellt.

Abb. 1. Begehbarer Raum aus Sicht der Kinect

Die Kinect kann über unterschiedliche Software-Bibliotheken von PCs aus angesteuert werden. Bereits eine Woche nach der kommerziellen Veröffentlichung der Kinect in den USA wurde am 11. 11. 2010 die Veröffentlichung eines Open Source Treibers bekanntgegeben, der in Kombination mit dem kurze Zeit darauf freigegebenen Open Source Framework OpenNI [8] das Tracking von Bewegungen ermöglicht. Microsoft reagierte am 16. 6. 2011 mit dem Release des Microsoft Kinect SDKs. Die Middleware der an OpenNI beteiligte Firma Prime Sense ähnelt vom Funktionsumfang her dem Microsoft Kinect SDK. Bei beiden Varianten ist das Tracken von menschlichen Skeletten integraler Bestandteil des jeweiligen Frameworks. Dabei gibt es jedoch leichte Unterschiede hinsichtlich der getrackten Joints: Während das Microsoft Kinect SDK die Position für 20 Joints pro Bild berechnet [9], liefert die OpenNI-Bibliothek nur 15 Gelenkpositionen und erfordert darüber hinaus eine initiale benutzerbezogene Kalibrierung.
Benutzertests haben ergeben, dass die Kopfposition bei beiden Frameworks relativ stabil erkannt wird, auch wenn sich nicht alle Teile des Skeletts im Blickfeld der Kamera befinden. Die Position des Kopfes ist Bestandteil des Skeletts und kann deshalb direkt für das Tracken verwendet werden. Für die Erkennung des Gelenkpunkts gilt annähernd die Genauigkeit, die durch die Auflösung der Kinect grundsätzlich erzielt werden kann, also zwischen wenigen Millimeter bis hin zu 5cm je nach Entfernung des getrackten Objekts.
Benutzertests haben gezeigt, dass die Auflösung für das Headtracking völlig ausreichend ist. Dies ist auch leicht nachvollziehbar, da einige CAVE-Systeme sogar ohne Head-Tracking betrieben werden, bzw. Benutzer, die sich nahe an der getrackten Person befinden, meist „subjektiv zufrieden“ mit der Darstellung sind.

Auf dieser Beobachtung basieren im Übrigen auch die kuppelförmigen Projektionsumgebungen, die im Grundsatz ja auch nur für einen Punkt im Kuppelzentrum ein korrektes Bild generieren können, dabei jedoch im Unterschied zu CAVEs vermeiden, dass die Betrachter visuelle Artefakte, wie Knicke an Leinwandkanten beobachten können.

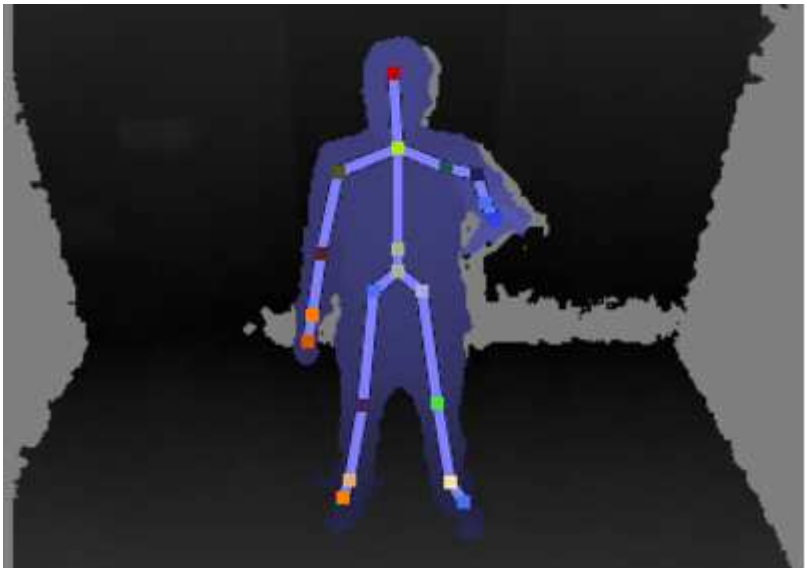

Abb. 2. Von der Kinect generiertes Tiefenbild (mit Skelett, 20 Joints)

Während eine CAVE, die nur Monobilder projizieren würde, mit der Position des Kopfes auskommen würde, muss für die Stereoprojektion die Ausrichtung des Kopfes bekannt sein, da hier ja zwei Bilder für die Position der jeweiligen Augen des Betrachters pro Leinwand generiert werden müssen.
Die Quellentrennung in der hier beschriebenen CAVE erfolgt über horizontale/vertikale Polarisation, womit implizit angenommen wird, dass der Benutzer sich ohnehin nur um seine vertikale Achse drehen wird, da sonst der Stereoeindruck verloren gehen würde.
Die Berechnung der Orientierungsmatrix zur Kopfposition erfolgt mit Hilfe der beiden Schultergelenke, wobei die Verbindungslinie die x-Achse des lokalen Koordinatensystems bildet und die y-Achse aus dem Mittelpunkt zwischen beiden Schultern und der erkannten Kopfposition errechnet wird. Statt des Kopfschwerpunktes wird für das Headtracking der Punkt in der Mitte zwischen beiden Augen benötigt, so dass hier ggf. noch ein experimentell bestimmter Offset berücksichtigt werden muss.
Tests mit unterschiedlichen Benutzern haben ergeben, dass die Genauigkeit der ermittelten Messdaten für Kopfposition und –ausrichtung völlig ausreicht, kritischer ist die Zuverlässigkeit der Erkennung und die Latenz. Wird der Kopf des Benutzers nicht mehr von der Kinect erkannt, bricht das Headtracking ab, befindet sich der Kopf dann wieder im getrackten Bereich, setzt das Headtracking nahtlos fort. Weist

man Benutzer auf diesen Umstand hin, wird diese Eigenschaft getestet, im normalen Betrieb bewegen sich Benutzer jedoch eher im Zentrum der CAVE.
Die Latenz bedingt durch die notwendige Bildverarbeitung fällt naturgemäß stärker bei schnellen Bewegungen ins Gewicht; auch hier ist zu beobachten, dass sich Benutzer aber eher vorsichtig und damit langsam bewegen. Letztlich scheint die Kinect für das Headtracking in einer CAVE völlig ausreichend, so dass die Instrumentierung des Benutzers durch Marker im Sinne eines Natural Userinterfaces vermieden werden kann.

3 Interaktion

Natürliche Userinterfaces haben den Anspruch ohne Interaktionsgeräte auszukommen, deren Funktion zunächst erlernt werden muss. Dies impliziert, dass am besten ganz auf Eingabegeräte verzichtet werden sollte und der Benutzer in natürlicher –ihm bereits vertrauter- Weise, z. B. mit seiner Hand oder seinen Fingern mit dem System interagieren können soll.
Voraussetzung dafür ist jedoch beim beschriebenen System die Erkennung von Teilen des menschlichen Skeletts, hier insbesondere der Hände. Aus der im letzten Abschnitt dargestellten relativ geringen horizontalen Auflösung lässt sich direkt folgern, dass einzelne Finger zurzeit nicht erkannt werden können. Genaue Zeigeoperationen über die Position der Hand sind aus dem gleichen Grunde nicht möglich, da die visuelle Auflösung der dargestellten 3D-Inhalte die möglichen 320 (bzw. 640) Bildpunkte in der Horizontalen deutlich überschreitet.
Andererseits haben (Multi-) Touch Interfaces grundsätzlich ein ähnliches Problem, da es nur schwer möglich ist, mit dem Finger auf einem Bildschirm pixelgenau zu selektieren. Und dennoch sind Touch Interfaces z. B. auf Smartphones oder Tablet Computern sehr stark verbreitet. Im Folgenden wird die Steuerung durch die Kinect deshalb am Beispiel zweier unterschiedliche Anwendungen untersucht.

4 Navigation Interface

3D Interaktion schließt drei elementare 3D-Benutzungsschnittstellen ein, Auswahl/ Bearbeitung (englisch *selection/ manipulation*), Systemsteuerung (englisch *system control*) und Navigation (englisch *navigation*) [10]. In einer großen 3D-Umgebung ist die Navigation die am weitesten verbreitete Möglichkeit zur Interaktion [11].
Kinect bietet durch die Erfassung von 20 Gelenkpunkten (Skelett) im Raum umfangreiche Möglichkeiten zur Umsetzung eines NUI. Abbildung 3 zeigt eine einfache steuerungsbasierte Technik zur Fortbewegung.
Eine solche Steuerungstechnik wird unabhängig von der Position und der Ausrichtung des Anwenders im Raum ausgeführt. Dadurch ist die Präsenz des Anwenders, also das Gefühl, sich innerhalb der Szene zu befinden, gering. Bei einer Technik wie *grabbing the air* [12] wird eine höhere Präsenz des Anwenders erreicht. Dabei bewegt sich ein Anwender fort, als würde er sich an einem Seil nach vorne ziehen. Durch die direkte Übertragung der in eine beliebige Richtung ausgeführten Handbewegungen in eine Verschiebung der 3D-Umgebung findet eine Auge-Hand-Koordination statt. Aufgrund der bereits erwähnten Ungenauigkeit kann das "Greifen" (englisch *grabbing*) durch Kinect nicht zuverlässig erkannt werden. Derzeit scheint die Verwendung eines weiteren Gerätes unverzichtbar zu

sein. Das "Greifen" wird daher durch Betätigung einer Taste einer kabellosen Wiimote ausgelöst, die der Benutzer in der Hand halten muss.

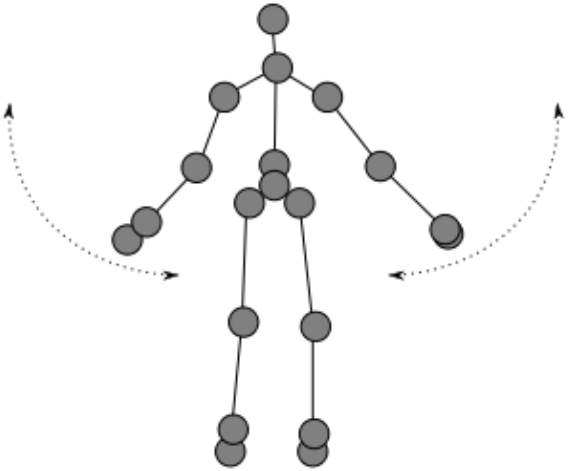

Abb. 3: NUI auf der Basis von Microsoft Kinect. Das Heben des linken oder rechten Arms dreht die Ansicht (Rotation), das Heben beider Arme führt zu einer Vorwärtsbewegung.

Ausgelöst durch einen Tastendruck wird kontinuierlich die Distanz zwischen aktueller und vorheriger Position der rechten Hand bestimmt. Die gesamte Szene wird um ein der tatsächlich zurückgelegten Entfernung entsprechendes Maß verschoben. Durch Betätigung einer anderen Taste wird die Szene analog

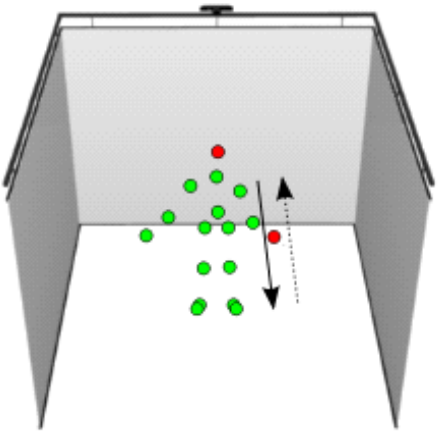

Abb. 4: Ausgelöst durch einen Tastendruck wird kontinuierlich die Distanz zwischen aktueller und vorheriger Position der rechten Hand bestimmt.

zur Fortbewegung um einen festen Bezugspunkt, etwa dem Kopf des Anwenders, gedreht. Dazu wird kontinuierlich der Winkel zwischen zwei Positionen und dem Bezugspunkt bestimmt. Bei einer Drehung um beispielsweise 180 Grad wird der Vorgang mehrmals wiederholt, indem die Hand ohne Betätigung der Taste nach vorn und mit Betätigung der Taste zurück bewegt wird (Abb. 4).

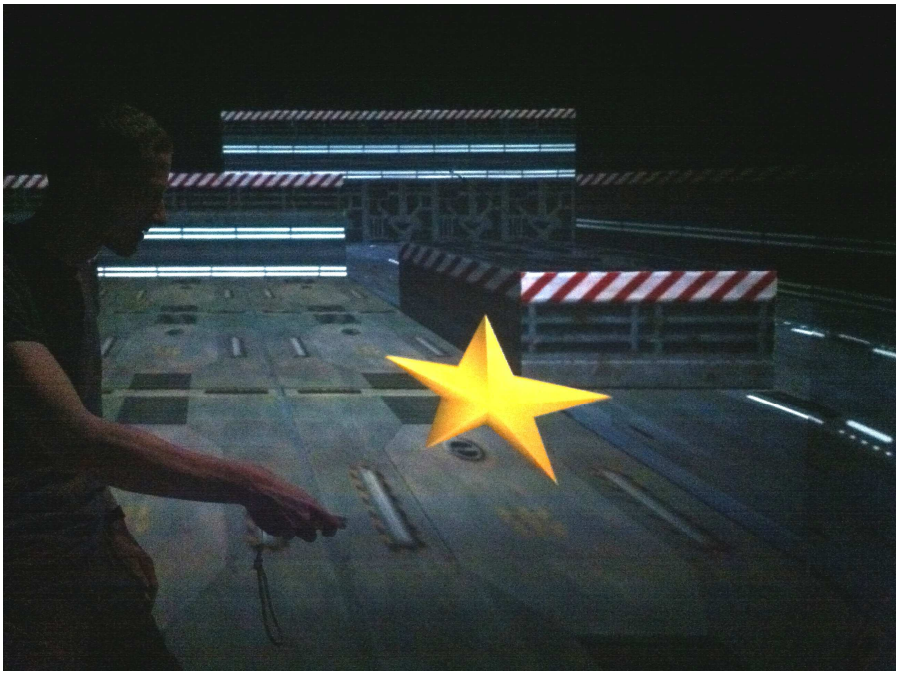

Abb. 5: Navigation durch Grabbing in the air in der CAVE der HTW Berlin

Das von Kinect gelieferte Tiefenbild unterliegt vor allem an den Rändern von Objekten starken Schwankungen, die zu einem "zittern" (englisch *jitter*) bei der Erkennung der Gelenkpunkte führen. Bei einer Bewegung der Szene wird dieses "Zittern" direkt auf die Darstellung übertragen. Das Microsoft SDK bietet in diesem Zusammenhang eine Funktion zur Glättung durch Double Exponential Smoothing [13] an, die für das Navigation Interface ausreichende Ergebnisse erzielt. Die Latenz bei der Umsetzung der Bewegung und die Wiederholrate zur Erkennung der Gelenkpunkte ist hier ebenfalls unproblematisch, da bei der Fortbewegung insgesamt keine besonders schnellen Bewegungen notwendig sind.
Die Abbildung der von Kinect generierten 3D-Daten auf den Interaktionsraum der CAVE wurde durch eine grobe Vermessung der Position und Ausrichtung des Geräts realisiert. In unseren bisherigen Tests erwiesen sich die so generierten Daten als ausreichend. Zukünftig ist eine genauere Kalibrierung durch optische Verfahren geplant. Ungenauigkeiten wirken sich in dem Fall vor allem bei der Rotation aus, da der Anwender seine Aktion auf einen bestimmten Punkt im Raum

bezieht, dessen Position möglichst exakt mit den von Kinect gelieferten Daten übereinstimmen sollte.
Die synchrone Darstellung der unterschiedlichen Abbildungen der CAVE der HTW Berlin erfolgt durch das komponentenbasierte Framework VR Juggler [14]. Gerätedaten werden in einem innerhalb des PC-Cluster als Master festgelegten Knoten empfangen und an die anderen Knoten verteilt. Für jedes Gerät existiert ein eigener Server, der von der entsprechenden Klasse des de facto Standards *Virtual Reality Peripheral Network* (VRPN) abgeleitet wurde [15]. Die Daten für die Position und die Orientierung des Kopfes und der Hand werden in der VR Juggler Konfiguration auf die bereits vorhandenen Proxies VRHead und VRWand abgebildet. Die Orientierung der Hand entspricht dabei zurzeit der Einheitsmatrix. Für eine zeigerbasierte Steuerung wäre aber auch die Nutzung des Vektors zwischen Ellbogen und Handgelenk zur Bestimmung der Orientierung im Raum denkbar.

5 3D – Modellierung

Anfang der 90er Jahre wurde der vermutlich erste immersive 3D-Modellierer „3DM“ von Butterworth und Davidson an der University of North Carolina entwickelt [16]. Weitere immersive Modelliersysteme aus dieser Zeit sind u.a. CHIMP und IM-Designer. Im Jahr 2000 untersuchten Deisinger und andere zusammen mit 36 professionellen Designern, die Anforderungen an ein immersives Modelliersystem am Beispiel von drei bestehenden CAVE-basierten Systemen [17]. Bis heute hat sich jedoch noch immer keine bevorzugte Technik für die immersive 3D-Modellierung herauskristallisiert.

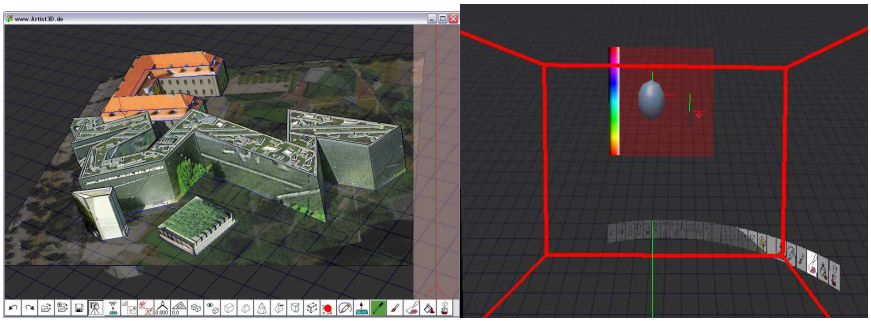

Abb. 6: links: 2D-Userinterface von Artist3D (Modell des Jüdischen Museums) rechts: Immersives User Interface in der CAVE

Im Rahmen dieser Arbeit wurde ein einfach bedienbares 2D-Modellierwerkzeug (Artist3D [18], siehe auch Abb. 6 links) zunächst mit Hilfe einer Imageplane-Interaktionstechnik [19] in die immersive Umgebung übertragen. Die Bildebene steht dabei jeweils senkrecht zur Blickrichtung. Die Bedienelemente aus der 2D-Variante wurden als 3D-Menu in die dreidimensionale Welt integriert, wobei die einzelnen Menüpunkte abhängig von der Blickrichtung in Bodennähe der CAVE angeordnet sind, so dass sie leicht mit der rechten Hand per Raycasting ausgelöst

werden können. Später sollen dann freie 3D-interaktionstechniken integriert werden, von denen bisher nur eine Kamerasteuerung implementiert wurde. In Abbildung 6 rechts symbolisieren die roten Linien die Verbindungskanten der CAVE-Seiten, der Ellipsoid zeigt den Kopf, der Quader in der Nähe die rechte Hand des Benutzers an. Gut zu sehen sind in der Abbildung die Imageplane mit einem Farbauswahlbalken sowie die Menuobjekte.
Raycasting-Techniken sind sehr viel störanfälliger gegenüber Jitter als zum Beispiel Head tracking. Da zwei Punkte benötigt werden, um einen Strahl zu generieren (z. B. Kopf und Hand oder Ellenbogen und Hand) verdoppelt sich die durch Jitter verursachte Ungenauigkeit. Da die Entfernung zu den Zielobjekten im Verhältnis zur Entfernung zwischen den beiden Referenzpunkten meist deutlich größer ist, verstärkt sich die Ungenauigkeit am Zielobjekt darüber hinaus gemäß Strahlensatz. Dies führt in der Praxis dazu, dass auch bei eingestellter Glättung die Präzision zur Selektion von entfernten Details fehlt. Der Cursor zittert dabei oft um das auszuwählende Detail.

6 Content – Generierung

Eine interaktive 3D-Anwendung besteht üblicherweise aus mehreren Komponenten. Typische Komponenten sind hierbei die graphische Darstellung (Rendering), die Verarbeitung von Benutzereingaben, Soundausgabe sowie eventuell zusätzliche Komponenten wie z.B. Physik oder KI. Da die Entwicklung und Integration dieser Komponenten einigen Entwicklungsaufwand bedeuten lässt sich die Entwicklungszeit einer Anwendung enorm verkürzen, wenn hierfür auf entsprechende Frameworks zurückgegriffen werden kann, die einen möglichst hohen Abstraktionsgrad bieten.
Bei der Erstellung von 3D-Anwendungen ist es außerdem üblich, dass die so genannten Assets (3D-Modelle, Texturen, Sounds etc.) von Künstlern mit wenig oder gar keinem Wissen über Programmierung erstellt werden.
Deshalb sollen neben VRJuggler auch Game-Engines für die Darstellung in der CAVE genutzt werden. Die meisten Game-Engines nutzen eine stark spezialisierte Render-Pipeline, die in der Regel nicht verändert werden kann. Für die Anpassung an die CAVE ist allerdings ein Eingriff in die Render-Pipeline notwendig um die entsprechenden Projektionsmatrizen zu erstellen.
Eine frei verfügbare Software, die alle diese Möglichkeiten bietet ist die Unity3D™-Engine[20]. Auf deren Basis konnte ein Framework erstellt werden, dass es erlaubt, mit wenigen Handgriffen, und ohne jegliche Programmierung, eine mit Unity3D entwickelte Anwendung “CAVE-fähig” zu mache. Das System generiert automatisch die benötigte Anzahl an Projektionen. Dabei wird jede Projektion ähnlich wie bei VRJuggler auf einem eigenen Rechner berechnet. Das System ist also einfach zu skalieren. Die Synchronisation erfolgt dabei wie bei herkömmlichen Netzwerkspielen üblich und nutzt die internen Netzwerkfunktionen der Unity3D™-Engine. Damit kann zwar keine framegenaue Synchronisation erreicht werden, durch den hier verwendeten passiven Ansatz zur Quellentrennung (über Polarisationsfilter) ist dies in der Praxis jedoch kein Problem.

Literatur

1. Cruz-Neira C., Sandin D. and DeFanti T., “Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE”, Computer Graphics, SIGGRAPH Annual Conference Proceedings, 1993.
2. Polhemus, Inc., Colchester, VT, 3SPACE FASTRAK User's Manual: Revision F, November 1993.
3. Sharlin E., Green M., Watson B., and Figueroa P., "A Wireless, Inexpensive Optical Tracker for the CAVE", in Proceedings of the IEEE Virtual Reality 2000. New Brunswick, New Jersey, March, 2000, pp. 271-278
4. Murgia A., Wolff R., Sharkey P. M. and Clark B.: Low-cost optical tracking for immersive collaboration in the CAVE using the Wii Remote, Proc. 7th ICDVRAT with ArtAbilitation, Maia, Portugal, 2008
5. http://www.primesense.com/?p=535, Abrufdatum 30.05.2011
6. Microsoft Research, “Kinect™ Services for Microsoft® RDS 2008 R3 Using a Kinect Sensor with Robotics Developer Studio, Preliminary Version – July 2011”, http://research.microsoft.com/en-us/downloads/f8cda115-e9ec-44d1-abcd-3dfdd09d2e77/, Abrufdatum 21.10.2011
7. Microsoft Kinect SDK. Datei „MSR_NuiSkeleton.h“, Funktion “NuiTransformDepthImageToSkeletonF“
8.http://www.openni.org, Abrufdatum 30.05.2011
9. Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, Kipman A, and Blake A, “Real-Time Human Pose Recognition in Parts from a Single Depth Image”, in IEEE Conference on Computer Vision and Pattern Recognition, June 2011
10. Bowman D, Kruijff E., LaViola J. and Poupyrev I.. An Introduction to 3D User Interface Design, *Presence: Teleoperators and Virtual Environments*, vol. 10, pp. 96-108, 2001
11. Bowman D., Kruijff E., LaViola J. and Poupyrev I. *3D User Interfaces: Theory and Practice*. Boston: Addison-Wesley, p. 184, 2005
12. Ware C. and Osborne S.. Exploration and virtual camera control in virtual three dimensional environments. In Proceedings of the 1990 symposium on Interactive 3D graphics, pages 175{183, New York, USA, 1990.
13. Holt C. "Forecasting Trends and Seasonal by Exponentially Weighted Averages". International Journal of Forecasting 20 (1): 5–10, (January–March 2004).
14. Bierbaum A., Just C., Hartling P., and Cruz Neira C. 2001. Flexible application design using VR Juggler. Proceedings of SIGGRAPH 2000, New Orleans, July 2000.
15. Taylor II R., Hudson T, Seeger A. Weber H., Juliano, J., Helser, A..: VRPN: a device-independent, networktransparent VR peripheral system. In: Proceedings of the 2001 ACM Symposium on Virtual Reality Software and Technology, pp. 55-61, 2001
16. Butterworth J. et al, "3DM: A three dimensional modeler using a head mounted display", Proc. SIGGRAPH 1992
17. Deisinger J. et al: Towards Immersive Modeling - Challenges and Recommendations: A Workshop Analyzing the Needs of Designers, Proc. of the 6th Eurographics Workshop on Virtual Environments, 2000
18. Jung T., Artist3D- Ein einfach bedienbares Werkzeug zur 3D-Modellierung von Oberflächen, In Proc Workshop 3D-NordOst 2006, Berlin, Germany, December 2006, pp 13-20
19. Pierce J., Forsberg A., Conway M., Hong S., Zeleznik R., Mine M.: Image Plane Interaction Techniques In 3D Immersive Environments, Proceedings of the 1997 Symposium on Interactive 3D graphics, pp 39 ff
20. http://unity3d.com/, Abrufdatum 31.10.2011