Dokumentation

KP Graphische Datenverarbeitung

Projekt: KP Graphische Datenverarbeitung 1.0
letzte Änderung: 24. Juli 2008

Inhaltsverzeichnis

1 Zielbestimmungen **3**
1.1 Musskriterien . . . 4
1.2 Wunschkriterien . . . 4
1.3 Abgrenzungskriterien . . . 4

2 Vorüberlegungen **5**
2.1 Ausgangssituation . . . 5
2.2 Inspirationsquellen . . . 6
2.2.1 Darstellung . . . 6
2.2.2 Immersion . . . 7
2.2.3 Intuitivität . . . 7
2.2.4 Fazit . . . 8

3 Graphische Darstellung des semantischen Netzes **9**
3.1 Mindestanforderungen der Darstellung . . . 9
3.2 Grundideen zum strukturellen Aufbau der Szene . . . 9
3.3 Aufbau und Navigation innerhalb des erstellten Programmes . . . 10
3.4 Einbeziehung der Mindestanforderungen . . . 13
3.5 Evaluation von Vrml/X3D und des Instantplayers . . . 14
3.5.1 Kurzbeschreibung . . . 14
3.5.2 Funktionsweise . . . 14
3.5.3 Möglichkeiten der Nutzung von Funktionalitäten . . . 15
3.5.4 Fazit . . . 16

4 Headtracking **17**

5 Gestenerkennung **18**
5.1 Evaluierung verschiedener Möglichkeiten zum LED Tracking . . . 18
5.1.1 Visionlib . . . 18
5.1.2 OpenCV . . . 18
5.1.3 WiiUse/WiiUseJ . . . 18
5.2 Implementation . . . 18

A Fehlerliste des InstantPlayers **20**
A.1 Schwere Fehler . . . 20
A.2 Mittelschwere Fehler . . . 23
A.3 Leichte Fehler . . . 25
A.4 Fazit . . . 26

1 Zielbestimmungen

Übliche PC Monitore oder Fernseher projizieren eine aufgenommene oder generierte dreidimensionale Szenerie auf eine flache zweidimensionale Fläche. Die Mensch-Maschine-Interaktion ist aus diesem Grund ebenfalls für Aktionen auf zweidimensionalen Flächen ausgelegt [Maus, Tastatur, Touchscreen, Stiftbasierte Interaktion, Matten auf welchen durch Tritte Aktionen ausgelöst werden, Tracking von Gegenständen mit einer Kamera]. Einige Geräte vermitteln den Eindrück in die Tiefe navigieren zu können wie z.B. Joystick, Lenkrad mit Pedalerie, wobei diese aktiven Interationstechniken ebenfalls für die Benutzung vor zweidimensionalen Bildschirmen angelegt sind. In echten dreidimensionalen Umgebungen wie vor der Powerwall des Lehrstuhl Computergrafik und Visualisierung, oder in einer Cave [5 Seiten Würfel in welchem sich der Benutzer / Betrachter steht und auf jede Seite ein Bild projiziert wird] sind umfangreichere Techniken gefragt, da hier mehr Freiheitsgrade existieren und die Notwendigkeit besteht, den Betrachterstandpunkt zu kennen. Am Lehrstuhl CGV existieren aktuell verschiedenste Geräte zur Interaktion in 3D Räumen wie beispielsweise: Spacepilots, Gyromaus, ein Datenhandschuh, ein Tracker sowie Kamerabasierte Eingabemethoden. In der CAVE des Zentrum für Virtuellen Maschienenbau wird aktuell ein sogenannter Flystick eingesetzt.
Im Zuge dieses Projektes sollen Interaktionsmethoden entwickelt, geschaffen sowie evaluiert werden, welche an einem zu erstellenden Medienbrowser prototypisch umgesetzt werden. Im Einzelnen müssen hierfür:

- Metaphern für Mediendaten gefunden,
- eine intelligente Datenhaltung/Aufbereitung geschaffen,
- ein ansprechendes Oberflächendesign erstellt,
- verschiedene Interaktionstechniken und -methoden implementiert und evaluiert,
- sowie eine Ergebnisbetrachtung in Form einer Dokumentation erstellt werden.

1.1 Musskriterien

- Entwicklung eines lauffähigen Programmes zur übersichtlichen Darstellung von polygonalen Netzen in vom Instant-Player unterstützten Formaten
- Einbindung von Kameras zum Tracken der Hände, was eine entsprechende Navigation ermöglicht
- Einbindung einer Navigationsmöglichkeit mit der Maus
- Einbindung verschiedener Interaktionsmethoden, im Speziellen Gestenerkennung und Headtracking
- Erstellen einer intuitiven Steuerung zur Navigation zwischen den Medien
- Erstellen einer Beispielszenerie mit verschiedenen Modellen
- Dokumentation des Endergebnisses

1.2 Wunschkriterien

- Erweiterung der Navigationsmöglichkeiten um weitere Schnittstellen
- Erweiterung um die Navigation mittels Spacepilot
- Aufbau eines semantischen Netzes zwischen verschiedenen Modellen zur Anzeige in dem zu schreibenden Programm

1.3 Abgrenzungskriterien

- Der Hauptaugenmerk des Projektes liegt auf der Entwicklung der Interaktion zwischen Nutzer und Medien
- Das Endprodukt versteht sich als prototypische Umsetzung
- Ein Editor zur Erstellung des der Visualisierung zugrunde liegenden semantische Netzes ist nicht zwingender Bestandteil des Projektes

2 Vorüberlegungen

Um diese Aufgabenstellung Schritt-für-Schritt zu verfeinern, mussten zuerst bereits vorhandene Ideen gefunden sowie evaluiert werden. Deren Vor- und Nachteile mussten herausgearbeitet und zu einem konsistenten Gesamtkonzept verwoben werden.
Im Zuge dessen wird dieses Kapitel zuerst vorhandene Möglichkeiten vorstellen, Projekte, welche in dieses einflossen, herausarbeiten und schließlich die Grundziele benennen.

2.1 Ausgangssituation

Der erste Schritt unserer Vorüberlegungen führte uns auf bereits vorhandene 3D-Medien-Browser. Die Suche danach führte relativ schnell zu Erfolgen, da das Katalogisieren großer Mengen von Medien natürlich eine bereits bekannte Aufgabe war.
Exemplarisch betrachten wir hier den Medienbrowser "Browz3D".

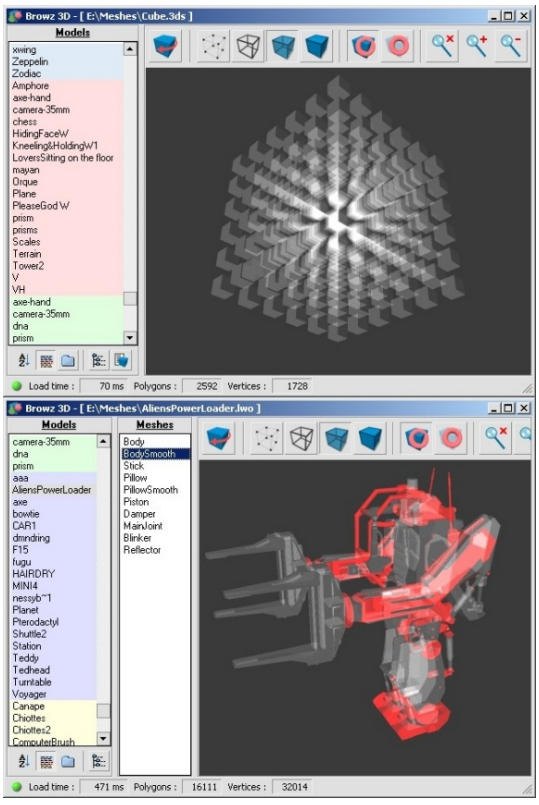

Abbildung 1: 3D-Medien-Browser "Browz3D"

Der Vorteil der gängigen Medienbrowser ist, dass bereits viele verschiedene Medientypen unterstützt werden.
Allerdings wird bereits bei diesem kleinen Beispiel ein gravierender Nachteil dieser Art ersichtlich - es werden zwar 3D-Modelle angezeigt, aber dennoch ist die Darstellung der Struktur rein zweidimensional. Auf der linken Seite sieht man eine Art Dateibaum als Auswahl

und rechts ein Ansichtsfenster des Mediums. Diese Eigenschaft reicht bereits, um die vorhandenen Möglichkeiten für unsere Zwecke als untauglich einzustufen, zerstören sie doch augenblicklich den Eindruck einer dreidimensionalen Welt und disqualifizieren sich somit von vornherein für den Einsatz an der Powerwall, in der Cave oder mit der Anaglyph-Brille.
Um den Umstieg auf eine komplett dreidimensionale Darstellung zu bringen, mussten also komplett neue Metaphern gefunden werden.

2.2 Inspirationsquellen

Für die für unser Projekt zu nutzenden Metaphern suchten wir nun also bereits bestehende Projekte, die für das Arbeiten mit 3D-Medien bereits gute Techniken zur Verfügung hatten. Aus diesen Ansätzen sollten wir Ziele erarbeiten und diese in unserem Medienbrowser verwirklichen.

2.2.1 Darstellung

Die erste Art von 3D-Anwendungen, die in unser Projekt einflossen, waren Projekte wie "BumpTop" [htta] oder "3D-Tabbing" [hndp].

Abbildung 2: BumpTop

Abbildung 3: 3D-Tabbing

Beide Anwendung haben gemeinsam, dass sie einen schwer zu überblickenden, zweidimensionalen Sachverhalt durch eine dreidimensionale Darstellung vereinfachen. Dieser Umstand passte wie perfekt für unseren Sachverhalt. Die menge an verschiedenartigen Medien sollte möglichst übersichtlich dargestellt werden.
Das erste große Ziel unseres Projektes war demnach die **Übersichtlichkeit der Darstellung**.

2.2.2 Immersion

Um dem Nutzer die Möglichkeit zu geben, ohne visuelle Störungen in der Szene zu navigieren, musste er direkt in die Szene eintauchen. Dies sollte dem schnellen Auffinden des gewünschten Mediums dienlich sein und brachte uns auf die Projekte von Johnny Chung Lee [hj] und Trackmania [httc].
Die Idee Trackmanias, einen Anaglyph-Modus in das Programm einzubauen, konnten wir Eins-zu-Eins in unser Projekt übernehmen, um so auch einen immersiven Effekt am "Rechner zu Hause" zu ermöglichen.

Abbildung 4: Johnny Chung Lees Headtracking

Johnny Lees Headtracking bewirkt einen viel tieferen Immersionseffekt beim Nutzer, da der Nutzer wesentlich direkter mit der Szene interagieren kann. Die Szene reagiert bereits auf seine Position. Auch mit diesem Effekt konnten wir die immersive Wirkung unseres Browsers weit über das bloße Benutzen der Powerwall oder der Cave hinaus erhöhen! Das zweite große Ziel bestand also in der **Immersion** des Nutzers.

2.2.3 Intuitivität

Bei einer rein dreidimensionalen Darstellung der Szene stellte sich uns schon bald die Frage, wie man sich innerhalb dieser bewegen sollte. Wir mussten also Wege und Möglichkeiten finden, in einer dreidimensionalen Struktur zu navigieren.
Bereits verwirklichte Möglichkeiten dazu bestanden in speziellen Eingabegeräten wie der Gyromaus und dem Spacepilot oder der Gestenerkennung wie zum Beispiel bei OpenCV [httb].
Alle Eingabemöglichkeiten sollen hierbei eine möglichst einfache Handhabung garantieren. Die **Intuitivität** der Eingabe ist das Entscheidende.

Abbildung 5: Spacepilot

2.2.4 Fazit

Insgesamt sollte unser Projekt also die drei großen, oben genannten Ziele vereinen und auf einen 3D-Medienbrowser anwenden:

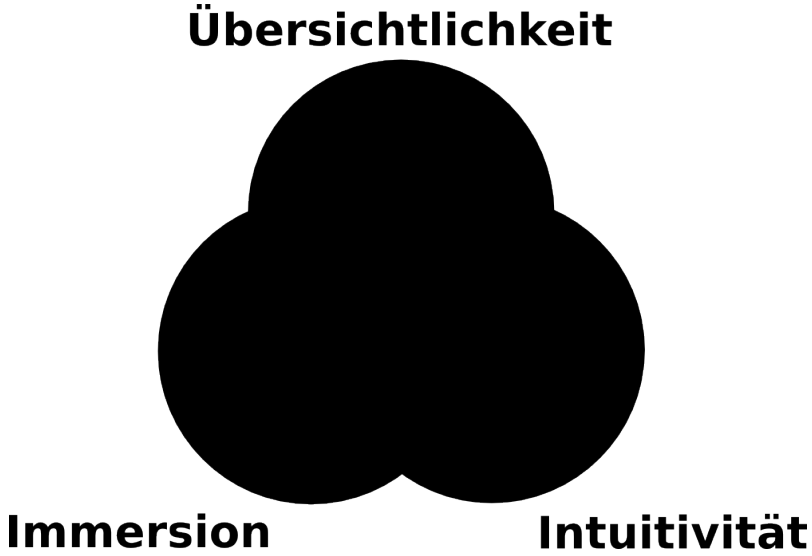

Abbildung 6: Übersicht

3 Graphische Darstellung des semantischen Netzes

3.1 Mindestanforderungen der Darstellung

Um ein Netz übersichtlich und intuitiv präsentieren zu können muss die Darstellung unter anderem folgenden naheliegenden Kriterien genügen:

1. Die Anzahl der dargestellten Elemente muss so gering wie möglich sein. Jedoch darf sie nicht zu gering sein, da sonst komplexere Netze nicht mehr zusammenhängend und unter großem Navigationsaufwand angezeigt werden könnten
2. In jedem Netzelement muss die ungefähre Orientierung innerhalb des Netzes erkannt werden können
3. Schritte zwischen verschiedenen Ebenen innerhalb des Netzes müssen nachvollziehbar sein
4. Blattelemente sollten sich so wenig wie möglich überlagern, gleichzeitig aber deutlich zu erkennen und ausreichend groß dargestellt sein

3.2 Grundideen zum strukturellen Aufbau der Szene

Um die verschiedenen Medienobjekte in der gewünschten Übersichtlichkeit anzeigen zu können, müssen diese in eine geeignete Datenstruktur eingefügt werden. Die wohl einfachste Möglichkeit hierfür ist eine Liste. Allerdings bieten die unterschiedlichen Medien keine Ordnungsrelation, was einer sinnvolle Reihenfolge entgegen steht.
Für die Kategorisierung, also Gliederung in Oberbegriffe und Spezialisierungen, bot sich die Baumstruktur an. Diese bringt allerdings eine gravierende Einschränkung mit sich: ein Medium kann nur genau einer Kategorie/einem Oberbegriff zugeordnet werden.
Um auch Assoziationen/Kanten zwischen Medien unterschiedlicher Kategorien zuzulassen wurde die Baum- zur Netzstruktur erweitert. Eben diese ermöglicht eine bessere Abbildung zwischen angebotener visualisierten Struktur und ebenfalls assoziativ angelegtem Gedächtnis.

3.3 Aufbau und Navigation innerhalb des erstellten Programmes

Nach Programmstart präsentiert sich die erste Ebene des semantischen Netzes. Dabei befindet sich das Wurzelelement im Vordergrund und die direkten Kinder durch Linien verbunden kreisförmig angeordnet im Hintergrund. Eine Bodenebene (gekachelt dargestellt) markiert die aktuelle Ebene.

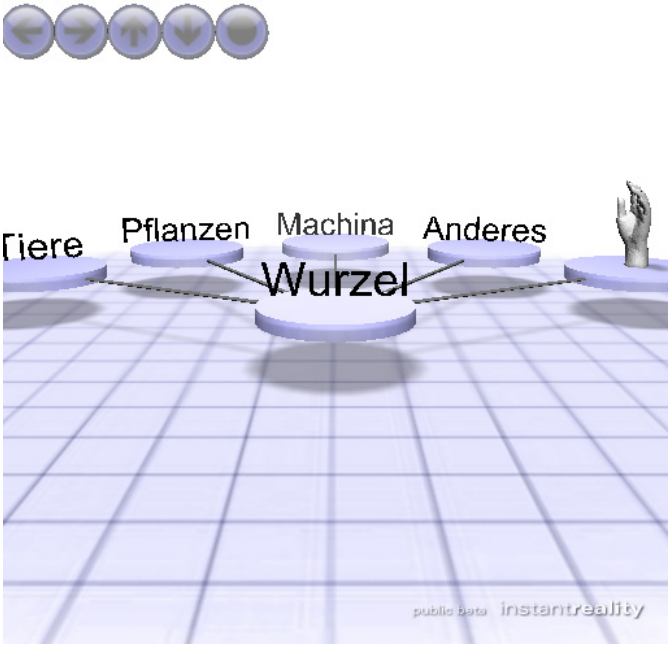

Abbildung 7: Programmstart

Bei Bewegung nach links oder rechts können bestimmte Kindelemente ausgewählt, die dann zentral hinter dem Wurzelelement angeordnet werden.

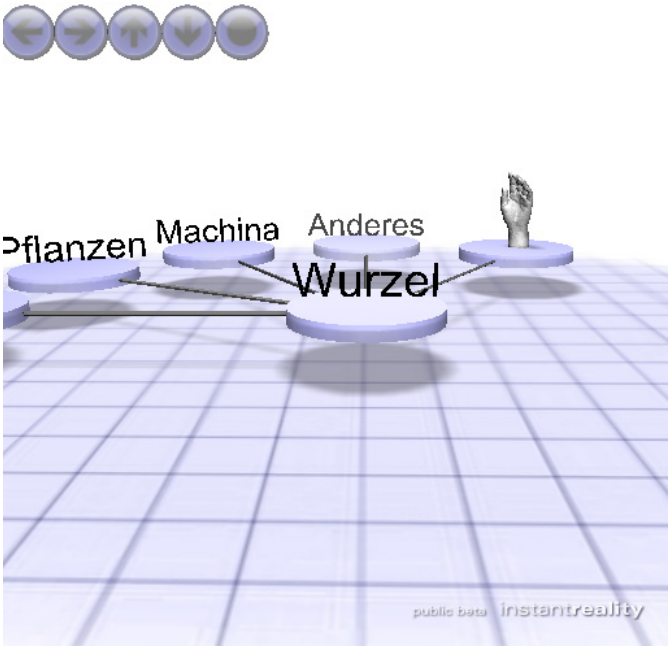

Abbildung 8: Bewegung nach rechts

Nach Bestätigung der Auswahl rückt die aktuelle Ebene unter den Boden, das gewählte Element schiebt sich auf die Position, an der vorher das Wurzelelement lag und stellt nun seinerseits das Wurzelelement seiner Kindelemente dar.

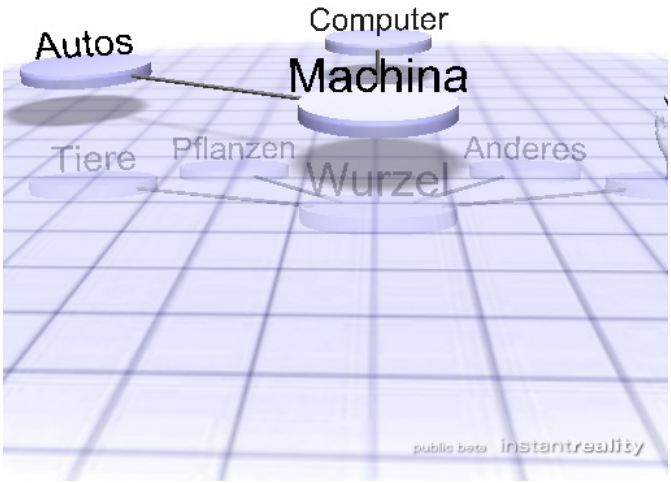

Abbildung 9: Ebene Zwei

Ist ein Blattknoten erreicht stellt sich dieser nicht durch ein Verzweigungstag, sondern durch das Objekt selbst dar. Wird es ausgewählt zoomt die Kamera darauf und es wird in den Betrachtungsmodus geschaltet, indem der Benutzer durch einfache bewegte Gesten affine Transformationen an dem Objekt durchführen kann.

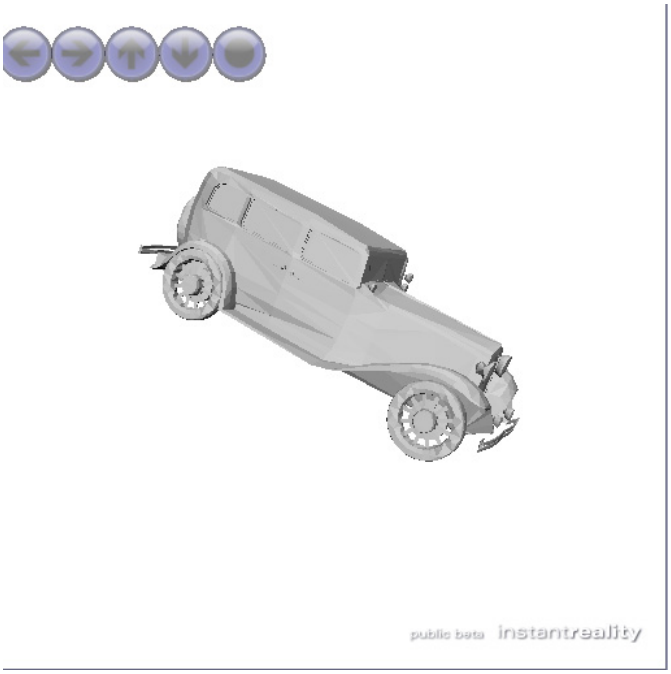

Abbildung 10: Objektanzeige

Für eine genauere Erklärung der Funktionen und der Steuerung ist ein separates Handbuch verfügbar.

3.4 Einbeziehung der Mindestanforderungen

Die umgesetzte Gestaltung und Darstellung des Netzes spiegelt die im Punkt „Mindestanforderungen“ genannten Punkte in folgender Form wider:

1. Es wird jeweils nur eine Ebene, also ein Wurzelelement mit seinen Kindelementen gezeigt, was für ein intuitives Verständnis der Eltern-Kind-Beziehungen ausreichend ist. Weiterhin sollten nur maximal fünf Kindelemente sichtbar sein. Sind mehr Elemente vorhanden wird dem Benutzer lediglich ein Fenster von fünf Elementen präsentiert, das er durch Links- und Rechtsbewegungen verschieben kann

2. Damit die Orientierung innerhalb des Netzes gewahrt bleibt rücken vorhergehende Ebenen in den Boden. Zu jedem beliebigen Zeitpunkt ist es damit möglich, den Weg über mindestens vier Ebenen zurückzuverfolgen. Außerdem ist diese Art der Darstellung mit Punkt 1 und (siehe unten) Punkt 4 optimal vereinbar, was bei anderen Präsentationen wie dem „immer weiter Eindringen in den Raum“ nicht der Fall wäre

3. Um die Schritte innerhalb des Netzes nachvollziehbar zu machen, bietet es sich an, auf plötzliche Änderungen der Szene zu verzichten und stattdessen flüssige Animationen einzubauen. Dieses Verhalten kommt der wirklichen Welt näher, da dort normalerweise auch keine Bewegungen von einem Ort zum anderen ohne Übergänge nachvollzogen werden können. Es wurde daher im Programm komplett auf Sprünge verzichtet und jede Bewegung durch eine flüssige lineare Animation dargestellt

4. Die Darstellung von Blattelementen wurde durch die Verlagerung der Szene in die Tiefe des virtuellen Raumes erreicht. Dadurch entsteht die Möglichket, Objekte wie Bilder und polygonale Netze aufrecht gestellt innerhalb des semantischen Netzes zu präsentieren, was eine optimale Sichtbarkeit ermöglicht. Weiterhin sind die Blattelemente automatisch so skaliert, dass sie nicht über den Rand der Scheibe, die ihren Sichtbarkeitsbereich repräsentieren, herausgehen können, wodurch eine Verdeckung minimiert wird

3.5 Evaluation von Vrml/X3D und des Instantplayers

3.5.1 Kurzbeschreibung

Der Instantplayer vom Frauenhofer Institut für Graphische Datenverarbeitung in Darmstadt ist ein Programm zur Darstellung von Mixed-Reality-Szenen, die auf dem Syntax von Vrml/X3D basieren. Konkret handelt es sich dabei um eine Implementierung des ebenfalls vom Frauenhofer-Institut entwickelten Avalon-Frameworks, das einen skalierbaren Ansatz zur Erstellung und Kommunikation zwischen Mixed-Reality-Komponenten ermöglichen soll. [Beh]
Der Instantplayer wird so beschrieben:
"*The instantreality-framework is a high-performance Mixed Reality (MR) system, which combines various components to provide a single and consistent interface for AR/VR developers. [...]*
The framework provides a comprehensive set of features to support classic Virtual Reality (VR) and advanced Augmented Reality (AR) equally well. The goal was to provide a very simple application interface while still including the latest research results in the fields of high-realistic rendering, 3D user interaction and total-immersive display technology. The system design includes various industry standards, like VRML and X3D, to easy the application development and deployment." [IGDa]

3.5.2 Funktionsweise

Der Instantplayer verarbeitet Vrml/X3D-Dateien. Die Szene wird dabei als beliebig tief verschachtelter Szenengraph beschrieben, deren Knoten entweder Transformations-, Verzweigungs- oder Formknoten zur Darstellung von Primitiven oder polygonalen Netzen, sind. Zusätzlich besteht die Möglichkeit, Dynamik in die Szene einzubauen. Dafür existieren folgende Möglichkeiten:

- Sensoren: Hierbei handelt es sich um Knoten, die Datenströme lesen, verarbeiten und als Ergebnis in einen sogenannten OutSlot bereitstellen können

- Timer: Ein Timer stellt einen Knoten dar, der zu einem gewählten Zeitpunkt über eine gewählte Dauer kontinuierlich Ereignisse erzeugen und sie über OutSlots zur Verfügung stellen kann
- Interpolatoren: Diese Knoten erhalten als Parameter (InSlots) Schlüsselwerte sowie eine Liste, wie diese Schlüsselwerte über die Zeit verteilt sind und generieren daraus in beliebiger Genauigkeit interpolierte Werte. Sie werden normalerweise mit Timern zusammengeschaltet.
- Skript-Knoten: Mit Hilfe von ECMAScript, bzw. Java können Skripte in die Szene eingebaut werden, die Parameter über InSlots erhalten und Ergebnisse in Form von Ereignissen über OutSlots bereit stellen können.

Weiterhin enthält fast jedes andere Knotenelement eine Reihe von In- und OutSlots. Die zentrale Anweisung innerhalb von Vrml/X3D ist der befehl „ROUTE“, der den OutSlot eines Knotens mit dem InSlot eines anderen verbindet. Dadurch wird es möglich komplexe Verbindungen zwischen Knoten herzustellen.
Die Besonderheit des Instantplayers besteht darin, dass er [IGDb]:

- für alle gängigen Betriebssysteme verfügbar ist
- den Satz an vorhandenen Knoten um moderne Elemente wie neue Eingabegeräte, prozedurale Texturen, Echtzeitschatten und Fellsimulation erweitert
- Es zusätzliche Bibliotheken zur Verarbeitung von Problemen aus dem Bereich Computer-Vision gibt
- Szenen parallel berechnen kann
- Szenen mit niedrigem Arbeitsaufwand an stereoskopische Anzeigegeräte wie Powerwall, CAVE, Interlaced-Displays oder Anaglyph-Displays angepasst werden können.

3.5.3 Möglichkeiten der Nutzung von Funktionalitäten

Während der Arbeit zeigte sich, dass der InstantPlayer zu dieser Zeit (BETA4, April – Juli 2008) lediglich rudimentäre Erweiterungen nur teilweise unterstützt und sich viele Elemente als sehr fehlerhaft herausstellten. Eine unvollständige Liste der größten Probleme, auf die gestoßen wurde befindet sich in Anhang A („Gesammelte Probleme des InstantPlayers“).
Die Visionlib als Instant-Player eigenes Tracking-System bietet die Möglichkeit einfaches markerbasiertes Tracking durchzuführen. In späteren Releases soll auch Tracking mittels LEDs anwendbar sein, welches sich derzeit noch in einem Alphastadium befindet (Information vom 29.Mai 2008). Eine Verwendung des LED-Trackings oder Verwendung mehrerer formbasierter Marker war hier, vor allem aufgrund mangelnder Dokumentation und Unvollständigkeit des InstantVision Tools, nicht möglich. Wegen der mangelnden Genauigkeit und Robustheit des markerbasierten Trackings, war die Visionlib zur Durchführung des Komplexpraktikum nicht geeignet.

Es wurde außerdem deutlich, dass die Möglichkeiten zum Debuggen von Skripten / X3D-Beschreibungen nicht oder nur sehr grundlegend vorhanden sind. So startet der InstantPlayer bei falsch geformten X3D-Texten beispielsweise nicht ohne eine Fehlermeldung zu hinterlassen. Die Debugconsole ist nur mäßig nutzbar, da sie keine Filterfunktionen besitzt und in der Windows-Version auch nicht geleert werden kann.
Schließlich zeigte sich, dass das Erstellen von komplexen Projekten mit Vrml/X3D einen weitaus höheren Aufwand an Planung erfordert.

- Durch die Fähigkeit in sich geschlossene Elemente über eine abstrakte Schnittstelle zu verbinden („ROUTE“) kann auf der einen Seite eine sehr starke Modularisierung erreicht werden. Andererseits muss das Interface, das jedes dieser Module in Form von Out- / InSlots bereitstellt jedoch genau auf die anderen Module passen. Passt beispielsweise ein InSlot nicht mit dem verbundenen OutSlot über den Datentyp zusammen kommt keine Verbindung zustande und eine potentielle Ereigniskette wird nicht ausgeführt.
- Weiterhin bietet das ROUTE-System vielfältige Möglichkeiten zur Parallelisierung von Programmteilen.
- Erwähnenswert ist außerdem, dass durch den Syntax von X3D viel Redundanz erzeugt wird und dadurch Quelltexte schnell mit sich wiederholenden Informationen anwachsen.

All diese Elemente erfordern ausgereifte Entwicklungs- und Debugwerkzeuge, die momentan nicht in ausreichender Qualität vorhanden sind.

3.5.4 Fazit

VRML ansich stellt eine recht elegante und einfache Möglichkeit zur Beschreibung von Szenen dar. Mithilfe von Sensor- /Timer- /Interpolationsknoten und Skriptknoten kann einfach Dynamik eingefügt werden. Mit gründlicher Planung ist so die Entwicklung komplexer Szenen in verhältnismäßig kurzer Zeit möglich.
Der InstantPlayer allerdings ist noch nicht für den produktiven Einsatz geeignet. Obwohl er offiziell bereits einen Beta-Status hat, ist die Fehler-/Absturzhäufigkeit sowie die Fülle an fehlender Funktionalität eher für die Alpha-Phase eines Programmes typisch. Einige schwere Fehler deuten auf strukturelle Schwachstellen hin, aufgrund dessen der Zeitraum bis zur wirklichen Nutzbarkeit in Jahren angegeben werden muss. Weiterhin fehlt es an geeigneten Möglichkeiten Programme für den InstantPlayer zu untersuchen und zu debuggen.
Über die Möglichkeiten der VisionLib können keine weiteren Aussagen außer ihrer momentanen Untauglichkeit getroffen werden, da Funtionalität nicht dokumentiert ist.

4 Headtracking

Um die Immersion über den 3D-Eindruck durch die Stereoprojektion hinaus zu steigern, wurde sich als Ziel gesetzt die Kopfposition zu verfolgen und die Szene an diese perspektivisch anzupassen. Hierfür gibt es mehrere technische Ansätze:

1. Einerseits besteht die Möglichkeit unter Nutzung einer Webcam und der OpenCV-Bibliothek aus der Folge der Einzelbilder der Webcam die Position des Kopfes im Raum über einen Gesichterkennungsalgorithmus zu berechnen. Dies erwies sich allerdings aus mehreren Gründen als wenig robust. Zum einen kam es zu 1-2 Einzelbildern in der Folge auf denen das Gesicht nicht erkannt wurde, andererseits gab es Schwierigkeiten bei der Zuordnung, wenn mehrere Gesichter im Sichtbereich der Webcam waren

2. Um also den Nutzer von eventuellen Zuschauern unterscheiden zu können, wurden LEDs an der Polarisationsbrille des Nutzers angebracht. Diese, ebenfalls erst per Webcam und Ecthfarb-LEDs per OpenCV und für zu empfindlich gegenüber äußeren Einflüssen erachtet (siehe Gestenerkennung), wurden durch Infrarot-LEDs ersetzt und per Wiimote wesentlich zuverlässiger verfolgt

Die Errechnung der perspektivischen Anpassung der Szenenrepräsentation erwies sich auf einfachem Wege als schwierig, da das Setzen des *Viewfrustum* im *InstantPlayer* leider noch nicht funktioniert. Deshalb wurde die Metapher des "um die Ecke schauen" gewählt und die Szene entsprechend der Kopfbewegung des Nutzers rotiert, z.b. wenn er sich nach links bewegt, sieht er mehr vom rechten Teil der Szene. Außerdem entsteht dadurch eine große Immersion im ObjectViewer, da der Nutzer das Gefühlt hat, um das Medienobjekt herumgehen zu können.

5 Gestenerkennung

5.1 Evaluierung verschiedener Möglichkeiten zum LED Tracking

5.1.1 Visionlib

Die Visionlib als Instant Player eigenes Tracking System bietet die Möglichkeit einfaches markerbasiertes Tracking durchzuführen. In späteren Releases soll auch Tracking mittels LEDs anwendbar sein, welches sich derzeit noch in einem Alphastadium befindet (Information vom 29.Mai 2008). Eine Verwendung des LED-Trackings oder Verwendung mehrerer formbasierter Marker war hier, vor allem aufgrund mangelnder Dokumentation und Unvollständigkeit des InstantVision Tools, nicht möglich. Wegen der mangelnden Genauigkeit und Robustheit des markerbasierten Trackings, war die Visionlib zur Durchführung des Komplexpraktikum nicht geeignet.

5.1.2 OpenCV

Durch die Verwendung der OpenCV library können LEDs mit Hilfe helligkeits- und formbasierter Annahmen sicher über einfache Webcams erkannt werden. Um hierzu möglichen externen Einflüssen durch Beleuchtungen zu vermeiden, wird ein IR-Passfilter benötigt und die Verwendung von IR-LEDs.

5.1.3 WiiUse/WiiUseJ

WiiUse ist eine cross platform library (Windows & Linux) zur kommunikation mit der Wii-Remote via Bluetooth. Die Wii-Remote enthält eine monochrome Kamera mit vorgeschaltetem IR-Passfilter. Bis zu 4 IR-LEDs können von der Wii-Remote mit 100Hz hardwareseitig getrackt werden bei einer effektiven Auflösung von 1024*768 (mittels 8x Subpixelgenauigkeit). Die Positionsdaten der LEDs können direkt über WiiUse ausgelesen werden. WiiUseJ ist eine Java-Portierung von WiiUse. Allerdings konnte diese aufgrund von Problemen des Instant Players beim Aufruf der zugehörigen Klassendateien nicht verwendet werden. Deshalb wurde für die Verwirklichung des Projektes die WiiUse library verwendet.

5.2 Implementation

Zur Gestenerkennung wurden 2 an den Händen (eine pro Hand) befestigte LEDs verwendet. Hierzu wurde deren relative Bewegung im Bezug auf die Position im vorhergehenden Frame betrachtet.
Bei der Gestenerkennung wird zwischen der Ansicht im SzeneView-Modus und im ObjectView-Modus unterschieden.
Befindet sich de Anwender im SceneView-Modus erfolgt eine einhändige Navigation mittels einer einzelnen Hand-LED. Durch horizontale Bewegung der Hand-LED kann der Anwender, um den aktuellen Knotenpunkt des Netzes rotieren und dabei den gewünschten Knotenpunkt auswählen. Ist der gewünschte Knotenpunkt ausgewählt, so kann der Nutzer durch Bewegung der Hand-LED nach unten den gewählten Knotenpunkt aufrufen und zu diesem

wechseln. Bewegt er die Hand-LED nach oben, so kann der Nutzer auf den vorhergehenden Knotenpunkt zurückwechseln.
Sobald der Anwender in den ObjectView-Modus durch auswählen eines Objektes im SzeneView-Modus wechselt, wird zu einer zweihändigen Gestenerkennung gewechselt. Hierzu wird die Geste anhand der relativen Bewegung der LEDs zueinander, als auch der Änderung der Position der LEDs im Vergleich zum vorhergehenden Frame bewertet. Es wird hierbei zwischen Gesten zur Rotation, Skalierung, Translation und Wechsel in den SzeneView-Mode unterschieden. Wurde eine Geste erkannt, so bleibt diese erhalten bis mindestens eine Hand-LED für einen bestimmten Timeout von der Wii-Remote nicht erkannt / gesehen wurde. Um zu vermeiden, dass eine Geste auch dann zurückgesetzt wird, wenn eine LED nur für einen oder wenige Frames nicht sichtbar ist, wird die Gestenerkennung für die Zeit der Unterbrechung angehalten, bis beide LEDs wieder sichtbar sind oder ein bestimmtes Timeout erreicht wurde. Wurde das Timeout erreicht, wird die aktuelle Geste zurückgesetzt und eine neue Geste kann erfolgen. Translation: Diese Geste wird gewählt wenn die beiden Hand-LEDs um eine bestimmte Mindestdistanz in die gleiche Richtung bewegt werden. Wird die Geste erkannt, so wird das betrachtet Objekt in die entsprechende Richtung verschoben.
Skalierung: Diese Geste ist wie für die Translation aufgebaut. Sie wird erkannt, sobald beide Hand-LEDs um eine Mindestlänge in die entgegengesetzte Richtung bewegt werden. Bewegen sich die LEDs aufeinander zu, so wird das Objekt verkleinert, bis es ein definiertes Minimum der Skalierung erreicht hat oder die LEDs sich nicht weiter aufeinander zubewegen. In gleicher Weise wird das Objekt entsprechend vergrößert wenn sich die beiden Hand-LEDs voneinander wegbewegen.
Rotation: Zur Ausführung dieser Geste wird eine beliebige Hand-LED an der aktuellen Position gehalten, während die andere LED senkrecht zur jeweiligen Rotationsachse bewegt wird. Die Geste wird erkannt, sobald eine bestimmte Mindestbewegung der einen LED erfolgt ist, während sich die Position der anderen LED nicht/kaum geändert hat. Eine bestimmte Abweichung der Position der „statischen“ LED von der „Ursprungsposition“ für die Geste muss hierbei mit einbezogen werden, da nicht davon ausgegangen werden kann, dass der Anwender die LED auf den Pixel genau positioniert halten kann. Auch hier bleibt die Geste solange Aktiv, bis mindestens eine der beiden Hand-LEDs für ein bestimmtes Timeout verdeckt wurde. Dadurch ist es möglich beide Hände für die Rotation zu verwenden sobald die Geste initialisiert wurde.
Wechsel zu SceneView-Mode: Um vom ObjectView-Mode zurückzuwechseln bewegt der Benutzer beide Hand-LEDs aufeinander zu bis sich diese fast berühren aber noch von der Wii-Remote als einzelne LEDs interpretiert werden können. Sobald sich die LEDs für eine bestimmte Framezahl in dieser Maximaldistanz befinden wird die Geste erkannt und es erfolgt der Wechsel zurück in den SceneView-Mode.

A Fehlerliste des InstantPlayers

A.1 Schwere Fehler

Beim Erstellen des Prototypen zeigten sich mehrere schwere Fehler, die entweder ein unvorhergesehenes Verhalten verursachten oder den Instantplayer (IP) komplett zum Absturz brachten. Dabei muss gesagt werden, dass das nur die Fehler sind, die während der Erstellung unseres noch sehr frühen Prototypen entstanden sind. Es ist davon auszigehen, dass es noch viel mehr Fehler gibt.

- Der IP stürzt ab, sobald man innerhalb eines Tags auf selbiges referenziert. Folgendes Beispiel reproduziert diesen Bug:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://
    www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile='Immersive' version='3.0'>
  <Scene>

        <Transform DEF='testTransform'>

              <Script>
                    <field name='testTransform' type='
                       SFNode' accessType='inputOutput'
                       ><Transform USE='testTransform'/
                       ></field>
                    <![CDATA[ecmascript:
                          function initialize() {
                                println("do
                                   something");
                          }
                    ]]>
              </Script>
        </Transform>
  </Scene>
</X3D>
```

  Ein Backtrace zeigt, dass das Programm in eine Endlosschleife läuft, bis der Speicher voll ist. Der Fehler wurde im Forum bereits gepostet mit der Reaktion, dass ein neues Ticket eröffnet wurde. Wann der Fehler allerdings beseitigt sein wird, ist so nicht zu sagen. Prinzipiell kann das Auftreten allerdings umgangen werden, indem man außerhalb referenziert. Besonders für Skripte, wo eine solche Referenzierung sinnvoll ist, erhöht das allerdings die Unordnung und verringert die Intuitivität.

- Der IP stürzt manchmal ab, sobald ein ROUTE nicht richtig gesetzt wird. Um in einem Skript / in einer Prototypendefinition auf eine Eigenschaft eines vorhandenen

Elements zuzugreifen ist es oft nötig, den Befehl ROUTE zu verwenden, der eine Variable auf eine andere automatisch mappt. Schnell kann es vorkommen, dass man Routes falschherum setzt, beziehungsweise den Rahmen des Routings größer setzt als notwendig (anstelle eines inputOnly-Parameters kann man ihn beispielsweise auch als inputOutput setzen). Hier passiert es manchmal, das der IP abstürzt. Dass diese Erklärung der Grund ist, konnte dem Forum als Antwort auf den Bugreport entnommen werden. Inzwischen konnte der Bug allerdings auch mit (vermutlich) richtig gesetzen ROUTEs reproduziert werden.

- Der IP stürzt *manchmal* aus noch unklarer Ursache bei komplexeren Szenen ab. Der Bug scheint beim Auflösen eines Skript-Tags aufzutreten. Ein Backtrace kann im Wiki nachgelesen werden.

- Die Reihenfolge von durch ROUTE gesetzten Parametern ist verwirrend. Um zwischen Skripten kommunizieren zu können ist es oft notwendig, gemeinsame Variablen zu setzen. Dabei setzt ein Skript eine Variable und das andere Skript reagiert auf diese Änderung. Müssen jetzt mehrere Variablen gesetzt werden, ist das Verhalten des IP seltsam, welche Änderung zuerst wahrgenommen wird. Ein Beispiel wäre ein Prototyp für ein Auto, das den Parameter „RichtungSetzen“ und „Losfahren(Geschwindigkeit)“ besitzt. Dieser Prototyp müsste beide Variablen exportieren können, während ein anderes Skript diese Variablen importiert. Jetzt ist es jedoch notwendig, die Variablen in der richtigen Reihenfolge zu setzen. Folgendes klinisches Skript verdeutlicht das:

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile='Immersive'>
  <Scene>
        <Script DEF='script1'>
                <field name='testFeld1' accessType='
                   inputOnly' type='SFInt32' />
                <field name='testFeld2' accessType='
                   inputOnly' type='SFInt32' />
                <![CDATA[ecmascript:
                        function testFeld1(v) {
                                println("Testfeld1 gesetzt
                                   !");
                        }

                        function testFeld2(v) {
                                println("Testfeld2 gesetzt
                                   !");
                        }
                ]]>
        </Script>
```

```
<Script DEF='script2'>
        <field name='setTestFeld1' accessType='
            outputOnly' type='SFInt32' />
        <field name='setTestFeld2' accessType='
            outputOnly' type='SFInt32' />

        <ROUTE fromNode='script2' fromField='
            setTestFeld1' toNode='script1' toField='
            testFeld1' />
        <ROUTE fromNode='script2' fromField='
            setTestFeld2' toNode='script1' toField='
            testFeld2' />

        <![CDATA[ecmascript:
                function initialize() {
                        setTestFeld2 = 1;
                        setTestFeld1 = 1;
                }
         ]]>
</Script>


  </Scene>
</X3D>
```

Das intuitive Verhalten wäre: „Setze erst testFeld2, dann testFeld1“. Eben je nach Reihenfolge des Aufrufs in der initialize-Methode von Skript2. Es zeigt sich jedoch, dass erst testFeld1, dann testFeld2 gesetzt wird. Würde „testFeld1“ jetzt dem Parameter „Losfahren“ entsprechen gäbe es ein Problem. Tests, wann was gilt haben gezeigt, dass die Reihenfolge der Feldbeschreibung des Skriptes, das Daten verändert, zählt. In dem Fall sorgt also die Tatsache, dass in script2 die Zeile “¡field name='setTestFeld1' ... ¿“ über der Zeile “¡field name='setTestFeld2' ... ¿“ steht dafür, dass testFeld1 immer vorher gesetzt wird. Das entspricht in keiner Form irgendeinem erwarteten Verhalten, sorgt für zusätzlichen Arbeitsaufwand und macht die Anwendung instabil und fehleranfällig. Es ist uns allerdings nicht klar, wie das vorgeschriebene Verhalten in VRML sein soll - es mag sein, dass dem so sein muss, was allerdings suboptimal wäre.

- Das Marker-Tracking mittels Visionlib ist funktionsfähig für einen einzelnen Marker, für mehrere Marker muss die entsprechende Config-File “.pm” mittels des Instant Vision Tools angepasst werden. Allerdings ist für dieses Tool keine Dokumentation vorhanden. Eine Abwandlung des Markers, um verschiedene Marker zu verwenden, war nach langer intensiver Suche möglich. Unter Zuhilfenahme der Beispieldatei “visionlib.pm” aus einem Tutorial konnten einzelne Marker erfolgreich getrackt werden.

Das Tracking war dabei allerdings nur sehr mäßig → hohe Fehleranfälligkeit des resultierenden Trackings. Nur sehr langsame Bewegungen konnten erfolgreich getrackt werden. Bei raschen Bewegungen konnte der Marker nicht verfolgt werden und die Position des Markers musste neu initialisiert werden. Die Beispieldatei visionlib.pm ist zudem auch nicht für das Projekt geeignet, da über diese nicht die Position des Markers, sondern die Position der Kamera relativ zum Marker übergeben wird. Auch ist es nicht möglich mehrere „IOSensor" Elemente vom Typ „VisionLib" mit verschiedenen ConfigFiles zu verwenden. In diesem Falle wird nur der erste Marker durch den ersten IOSensor erfolgreich getrackt. Für den zweiten Marker versucht die Visionlib ein zweites mal die angeschlossene Kamera anzusprechen und stopt hierbei, da diese bereits in Verwendung ist. Zur Lösung dieses Problems wäre eine größere Anpassung der Config-File notwendig. Allerdings ist hierzu keinerlei Dokumentation erhältlich, weder für die VisionLib selbst, noch für das Instant Vision Tool. Bezüglich dem LED Tracking haben wir eine Anfrage im Forum des Instant Players geschrieben, hierzu die Antwort:

```
Hello,

LED Tracking is considered alpha-state, so there is no doc yet.
If you like to experiment, here is a config file for ridget body tracking,
which was one good enouth for http://a4www.igd.fraunhofer.de/projects/45/
The LEDModel describes the positions of the LED on some object,
note the  x,y,z values there.

You might also try triangulation with IRLEDDetectAction+LEDTriangulationAction,
but I have no config for this at hand.

cheers, mb
```

- Ein weiteres Problem ist, dass die Verwendung der Visionlib in den Daily-Builds des Instant Players nicht vollständig möglich ist. So wird hierbei das Kamerabild nicht ordnungsgemäß getrackt, bzw. gar nicht ausgegeben. In der offiziellen Beta 4 des Instant Players kann die Visionlib zwar verwendet werden, allerdings mit großen Einschränkungen. So erhält am beim Aufruf von „LEDActionTracking" im Instant Vision Tool die Meldung:

  ```
  "no license for feature Vision, VPLED Tracking"
  ```

  Insofern ist LED Tracking in der offiziellen Beta 4 nicht anwendbar.

A.2 Mittelschwere Fehler

- Um Bewegungen von Objekten zu realisieren ist es notwendig ein Timer-Knoten zu setzen. Dieser Knoten erzeugt offenbar eine Referenz auf alle übergeordneten Elemente.

Das Resultat ist, dass, wenn ein Knoten aus der Szene entfernt wird, der Reference-Counter gesenkt wird, jedoch dann noch nicht Null ist. Der Knoten verschwindet (optisch), ist intern jedoch noch gespeichert und der Timer läuft weiter. Wird der Timer auf „loop“ gestellt, läuft er unendlich lange weiter und das Objekt wird also nicht aus dem Speicher entfernt. Weiterhin wird nie die „shutdown()“-Methode eines im entfernten Knoten enhaltenen Skriptes aufgerufen. Eine Anfrage im Forum zeigte, dass man sich gedulden muss, bis der Garbage-Collector wieder durchgelaufen ist (den man nicht selbst triggern kann). Da dieses Verhalten allerdings keine schwere Einschränkung ist sondern nur unverständlich, wird er als mittel eingestuft. Laut IP-Entwicklern ist ein korrektes Verhalten in solchen Situationen im VRML-Standard seltsamerweise nicht definiert.

- Variablen im Skript werden ohne zusätzliche Spezifikation grundsätzlich global angelegt. Erstellt man innerhalb eines Skriptes in einer Methode eine Variable ohne das „var“-Stichwort werden diese Variablen immer global angelegt. Folgendes Beispiel verdeutlicht das:

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile='Immersive'>
  <Scene>
        <Script DEF='script1'>
                <![CDATA[ecmascript:
                        function test1() {
                                i = 2;
                        }

                        function test2() {
                                println(i);
                        }

                        function initialize() {
                                test1();
                                test2();
                        }
                ]]>
        </Script>
  </Scene>
</X3D>
```

Beim Aufruf von „test2()“ wird „2“ ausgegeben. Stellt man in „test1()“ das Stichwort „var“ vor die Zeile „i = 2“ wird die Variable lokal angelegt. Nun ist es so. dass man häufig dazu neigt, genau das zu vergessen, was dazu führt, dass z.B. rekursive Methoden, die eine Variable als Zähler benutzen, nicht funktionieren und das Programm wiederum in eine Endlosschleife bringen. Es muss allerdings erwähnt werden, dass dieses Verhalten nicht unbedingt einen Fehler darstellt und als Duct-Taping einer

Eigenschaft an das „this“-Objekt gedeutet werden könnte.

- Die Kameraanbindung der visionLib ist in GNU/Linux nur teilweise implementiert. Momentan funktioniert sie nur für „uEye“-Kameras, die offenbar ein spezielles Framework in GNU/Linux benutzt. Es existiert jedoch die einheitliche Schnittstelle „Video for Linux“ (V4L). Das Plugin dazu ist allerdings offenbar lediglich ein Dummy. Laut Entwicklern existiert die Funktionalität noch nicht.

- Einige Komponenten innerhalb des Scriptings sind nicht oder teilweise implementiert. Ein wichtiger Befehl, der in der ECMAScript-Umsetzung für VRML vorhanden sein sollte ist „replaceWorld(Knoten)“. Damit kann die gesamte Szene entfernt und durch „Knoten“ ersetzt werden. Beim Aufruf dieses Kommandos erscheint der Fehler „ECMA Script FATAL: replaceWorld() not implemented“. Ob weitere Befehle nicht implementiert sind, ist nicht klar, da auf der Internetseite keinerlei Informationen dazu vorhanden sind.

- Es ist - anders als auf der IP-Seite behauptet - nicht möglich, den Ladestatus von polygonalen Netzen, die mit dem ¡INLINE ... /¿-Tag eingefügt wurden, zu tracken. Normalerweise kann man einen Sensor an solche Elemente hängen, der z.B. ein Skript startet, wenn etwas geladen wurde. Offenbar ist das INLINE-Tag, das gerade für unsere Zwecke von zentraler Bedeutung ist, nicht in der Lage, Informationen an den Sensor zu senden.

A.3 Leichte Fehler

- Es ist nicht klar, was genau die ECMAScript-Implementierung im IP nun leisten kann. Offiziell soll der VRML97-Standard EAI unterstützt werden. Hier sind einige Methoden offenbar jedoch nicht oder nur teilweise implementiert. Auf der anderen Seite existiert das Objekt „currentScene“, das nicht zu EAI, sondern zum neueren X3D-scripting-Standard SAI gehört. Auch hier werden nur einige Methoden teilweise oder gar nicht unterstützt.

- Es scheint nicht möglich zu sein, die Bounding-Box eines geladenen Polynetzes zu lesen. Man kann sie beim Erstellen eines INLINE-Tags angeben, jedoch nicht lesen. Dies ist allerdings notwendig zum korrekten Ausrichten der geladenen Modelle. Der VRML-Standard sieht allerdings anscheinend für dieses Problem auch keine Lösung vor.

- Quelltext zu debuggen ist sehr kompliziert, da die Konsole nur minimalistische Meldungen über die Fehler ausgibt und jedesmal den kompletten Skriptcode mit in das Fehlerfenster schreibt.

- Wird die visionLib verwendet ohne dass eine Kamera vorhanden oder zugriffsbereit ist, stürzt der IP manchmal ab.

A.4 Fazit

Die obengenannten Fehler sind entweder schwere Bugs, bei denen zu hoffen ist, dass sie möglichst schnell gefixt werden oder welche, die man nur mit Mühe und viel zusätzlicher Zeit beheben kann. Alles in allem scheint die ECMAScript-Anbindung daher noch nicht ausgereift genug um in Projekten zum Einsatz zu kommen. Viele Elemente, wie die VisionLib, befinden sich außerdem noch im Alpha-Stadium, sind fehlerhaft und nicht dokumentiert. Es ist generell nicht klar, was bereits funktioniert und was nicht. Der Instantplayer hat allerdings den unschlagbaren Vorteil, dass stereoskopisches Anzeigen ohne Probleme funktioniert. Getestet wurden bisher Farbfilter und Polarisationsfilter. Es soll außerdem problemlos möglich sein, das Bild in einer CAVE anzuzeigen. Außerdem ist er portabel und für die größeren Betriebssysteme in ähnlichem Entwicklungsstand vorhanden.

Abbildungsverzeichnis

1 3D-Medien-Browser "Browz3D" 5
2 BumpTop 6
3 3D-Tabbing 6
4 Johnny Chung Lees Headtracking 7
5 Spacepilot 8
6 Übersicht 8
7 Programmstart 10
8 Bewegung nach rechts 11
9 Ebene Zwei 12
10 Objektanzeige 13

Literatur

[Beh] J. Behr. *Avalon: Ein skalierbares Rahmensystem für dynamische Mixed-Reality Anwendungen*. PhD thesis, Technische Universität Darmstadt.

[hj] http://www.cs.cmu.edu/ johnny/projects/wii/. Head tracking for desktop vr displays using the wii remote.

[hndp] http://f6design.com/journal/2007/06/06/microsofts-new-design philosophy/. Microsoft's new design philosophy.

[htta] http://www.bumptop.com/. Bumptop, rethink your desktop.

[httb] http://www.intel.com/technology/computing/opencv/index.htm. Open source computer vision library.

[httc] http://www.trackmania.com. Trackmania.

[IGDa] Frauenhofer IGD. *instantreality 1.0 home*.

[IGDb] Frauenhofer IGD. Instantreality 1.0 story – what is it. Technical report, Frauenhofer IGD.