Ausarbeitung zum Hauptseminar:

Virtual- und Augmented Reality

Thema:

Architektur und Entwicklung von VR-Anwendnungen (VR-Frameworks)

Name:	**Helge Groß**
Fachsemester:	**6**
Professor:	**Prof. Heinrich Hußmann**
Betreuer:	**Dipl.-Inf. Arnd Vitzthum**
Semester:	**Sommersemester 2004**
Datum:	**25.07.2004**

Ludwig-Maximilians-Universität München
Lehr- und Forschungseinheit Medieninformatik

Überblick

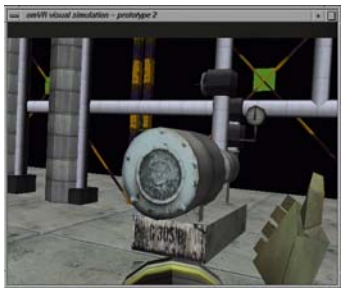
In dieser Seminararbeit wird die Architektur und Entwicklung von VR-Anwendungen betrachtet. Die Grundlage beim Entwickeln von Virtual Environments (VE) sind VR-Frameworks. Zunächst wird geklärt in welchen Bereichen bzw. Situationen VR eine Hilfe sein kann. Die Betrachtung der Architektur einer VR-Anwendung geschieht anhand eines Beispiels. Des Weiteren erfolgt eine kurze Einführung in den Bereich der VR-Autorenwerkzeuge.

1 Einleitung

VR-Anwendungen eignen sich für viele Entwicklungsbereiche in der Industrie. Die durch VR ermöglichte schnelle Präsentation und Untersuchung von Produkten ohne deren physisches Modell spart oft hohe Kosten. Diese Technik wird häufig beim Produkt-Design, für Material-Tests oder in der Automobilindustrie bei Crash-Tests eingesetzt. An den virtuellen Modellen können auch kinematische, ergonomische und aerodynamische Analysen vorgenommen werden. Dieser Vorteil wird beispielsweise in der Luft- und Raumfahrt genutzt. Im Lern- und Entertainmentbereich werden realistische Anwendungen für Simulationen, Training und Unterhaltung benötigt. Dabei handelt es sich nicht nur um Echtzeit-3D Computerspiele sondern z.B. Anwendungen wie ein VR-Sicherheitstraining für gefährliche Arbeitsplätze, das später näher betrachtet wird. [1]

2 Architektur von VR-Anwendungen

2.1 SAVE (Safety Virtual Environment)

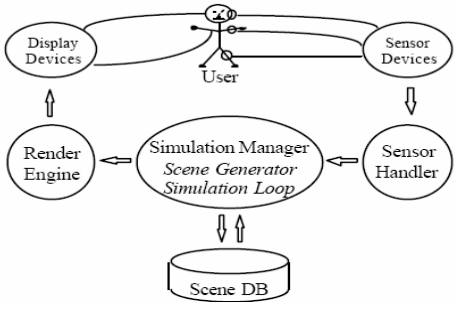


VR-Sicht in SAVE [4]

Die VR-Anwendung SAVE wurde für den österreichischen Mineralölkonzern OMV entwickelt. Kerngebiet von SAVE ist die Ausbildung von Angestellten für die Arbeit in Raffinerien. Gefährliche- und schwierige Situationen können so ohne Risiko trainiert werden, wie z.B. das Anlegen von Sicherheitskleidung im Notfall (Gasmaske), das Ablesen und Überprüfen von Instrumenten unter Stress und evtl. eingeschränktem Sichtfeld usw. Zur Interaktion mit der virtuellen Welt steht dem Trainee eine virtuelle Hand-Metapher zur Verfügung. [3] [4] [5]

2.2 Architektur von SAVE

Auf Hardwareebene besteht SAVE aus einer Grafik-Workstation auf der der Szene-Simulator läuft, einem Trainer-PC auf dem die Trainer-Applikation läuft, einem Head-Mounted-Display (HMD) und einem magnetischen Trackersystem für eine 3D-Maus und das HMD. Der Szene-Simulator liefert die VR-Szene und handhabt die Benutzerinteraktion (Trackingdaten und Mausklicks). Über die Trainer-Applikation steuert

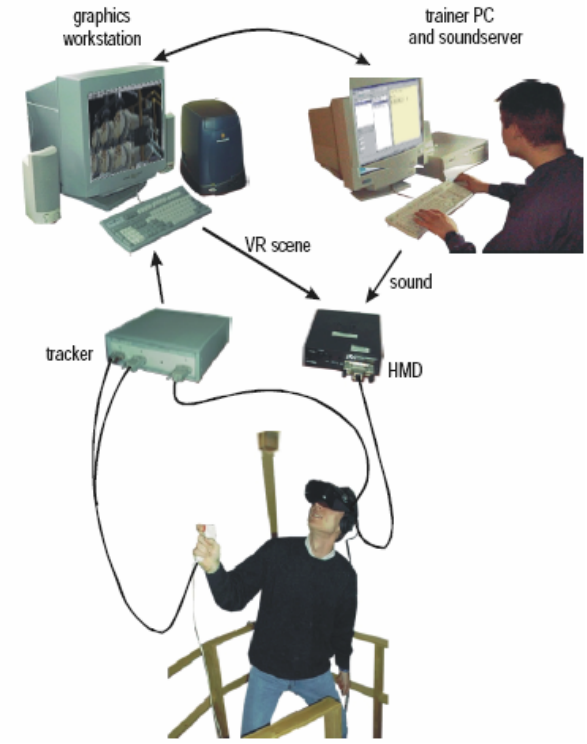


Allgemeine VR-Architektur [25]

der Trainer die zu trainierenden Variablen der Szene wie z.B. die Werte von Instrumenten. Neue Szenen können über ein Userinterface erstellt werden. Durch das HMD, die Hand-Metapher und das Trackingsystem entsteht ein hoher Immersionsgrad. [3] [4] [5]

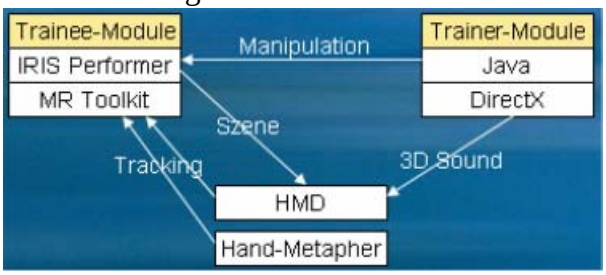
Auf Softwareebene gliedert sich die Architektur von SAVE in zwei Module. Das Trainee-Modul verwendet das IRIS Performer VR-Framework zur Erzeugung der Szene und das MR Toolkit VR-Framework zum Handhaben der Eingabegeräte. Das Trainer-Modul ist eine Java-Anwendung die mit dem Trainee-Modul kommuniziert. Gleichzeitig dient es auch als Soundserver, der mittels DirectX den 3D-Sound an das HMD liefert (akustische Warnsignale). [4]

SAVE ist als Objektorientiertes Design entwickelt worden. Der Vorteil ist bei VR-Anwendungen derselbe wie bei „normalen“ Anwendungen: Flexibilität, Verständlichkeit- und Übersichtlichkeit des Programmcodes und Erweiterbarkeit. Die Nutzung der Objektorientierung erlaubt es VR-Probleme auf einer sehr hohen Abstraktionsebene zu spezifizieren. [4]



Topologie von SAVE [4]

SAVE soll als Beispiel für eine einfache VR-Architektur gesehen werden. Weitere Möglichkeiten für die Architektur von VR-Anwendungen sind z.B. verteilte VR-Systeme, die mittels DIVE (Distributed Interactive Virtual Environment) umgesetzt werden können oder die Bindung an die Architektur bzw. Dateistruktur einer VR-Engine.



Modul-Architektur von SAVE [4]

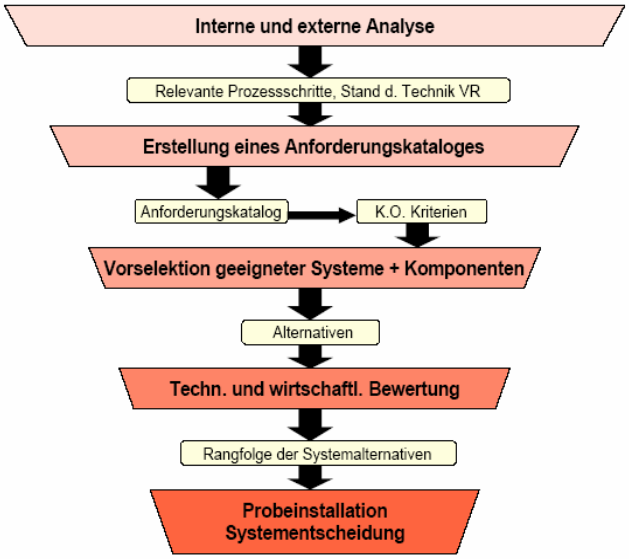
3 Entwicklungsphasen und Workflow von VR-Anwendungen

3.1 Vorgehensweise bei der Entwicklung einer VR-Anwendung

An erster Stelle steht die Analyse des durch VR zu lösenden bzw. zu vereinfachenden Problems. Dabei sollte sich die eigentliche Aufgabenstellung an die zukünftige VR-Anwendung herauskristallisieren. An dieser Stelle sind auch die wirtschaftlichen Rahmenbedingungen des Projekts festzuhalten (Budget). Evtl. kann der Entwicklungszeitraum schon grob überschlagen werden. [6]

Auf die Analyse folgt die Erstellung des Anforderungskatalogs bzw. Pflichtenhefts. Zunächst sollte der Immersionsgrad bestimmt werden, der unter anderem die Komplexität der

VR-Anwendung vorgibt. Es gilt zu klären welche Interaktionsmöglichkeiten dem Nutzer bereitgestellt werden müssen. Interaktion und Immersion benötigen dann je nach dem VR-Hardware in Form von Eingabe- (Space-Maus), Ausgabe- (HMD) und Trackinggeräten. Weitere wichtige Bestandteile des Anforderungskatalogs sind die Performanz (Schnelligkeit) und Flexibilität (Portabilität) der Applikation. Die Performanz spielt insbesondere in Verbindung mit dem (Head-)Tracking eine große Rolle, da hierbei das Bild ständig neu berechnet werden muss. Bezüglich der Ein- und Ausgabegeräte muss feststehen ob es sich um eine Desktopanwendung für eine Person handelt, die z.B. auch mit einem 3D-TFT zu lösen wäre oder ob die Präsentation der VR für eine Gruppe zur Verfügung gestellt wird und deshalb mit einer CAVE oder Holobench interagieren muss.



Entwicklungsprozess [1]

Dies setzt für die Anwendung Multiple-Pipe-Support voraus, da auf mehreren Projektionsebenen dargestellt werden muss. Stereofähigkeit ist eine weitere Grafische Voraussetzung die die Anwendung evtl. beherrschen muss. [1] [7] [6]

Nach diesen Vorgaben lassen sich nun die geeigneten Komponenten (Frameworks, APIs) bestimmen, die verwendet werden müssen um die Funktionalitäten darauf aufbauend zu entwickeln. [1]

Im Idealfall werden unterschiedliche Prototypen entwickelt von denen durch Evaluierung anhand von erfüllten Anforderungen und Kostengünstigkeit letztendlich einer in den eigentlichen Entwicklungsprozess übernommen wird. [1]

3.2 Aufgabenverteilung im Workflow einer VR-Anwendung

Bei der Entwicklung großer VR-Anwendungen und besonders bei der Entwicklung von Echtzeit-3D Computerspielen, werden die verschiedenen Aufgaben auf verschiedene Entwickler-Rollen verteilt. Es wird unterschieden zwischen Programmierer, Modellierer, Texturierer und Sound-Designer. [8]

Der Programmierer ist für die Logik, Algorithmen, programmierte Animation und Interaktion mit der Anwendung zuständig. VR ist immer Echtzeit-3D, kann folglich ohne Programmcode nicht funktionieren. Handwerkszeug des Programmierers ist meist die Programmiersprache C++, diverse VR-Frameworks bzw. APIs und evtl. die Scriptsprache einer VR-Engine. [8]

Der Modellierer erstellt mittels 3D-Authorenwerkzeugen wie z.B. 3D Studio MAX oder Maya die benötigten 3D-Objekte. Hierbei ist eine präzise Vorgehensweise nötig, da ungenau modellierte Objekte oder Szenen später zu Grafikfehlern führen können. Bei Verwendung einer VR-Engine legt er die Texturkoordinaten (UVW) fest, andernfalls ist dies Aufgabe des Programmierers bei der Anzeige eines 3D-Objekts. Der Modellierer ist für kleine Animationen zuständig, die direkt am 3D-Modell designt werden können. [8]

Der Texturierer erzeugt die Bitmaps die auf die 3D-Modelle gelegt werden sollen. Dabei handelt es sich meist um die Erstellung möglichst real wirkender Materialien und Oberflächen. Als einzige Vorgabe damit am Ende alles zusammenpasst benötigt er das Seitenverhältnis, denn auf die 3D-Objekte gemappt wird von den beiden anderen Entwicklern. [8]

Der Sound-Designer kreiert möglichst realistische Geräusche und Klänge. Dazu dienen Audioeditoren wie Adobe Audition, OpenAL oder PortAudio. [8]

4 VR-Autorenwerkzeuge (VR-Entwicklungsumgebungen)

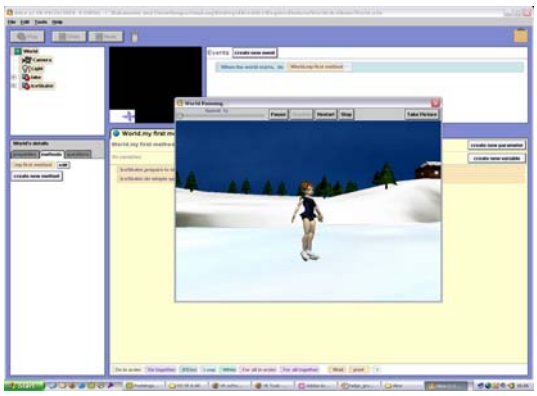
4.1 Funktionsweise von VR-Autorenwerkzeugen

Zur Entwicklung einer VR-Anwendung kann eine VR-Entwicklungsumgebung verwendet werden. Diese so genannten IDEs (Integrated Development Environments) stellen die einfachste und intuitivste Form der VR-Entwicklung dar. Dabei wird das Virtual Environment vollständig über ein GUI (Grafical User Interface) erstellt. Um wirklich schnell entwickeln zu können, muss das GUI eines VR-Autorenwerkzeugs intuitiv verständlich sein. Damit ermöglicht es Rapid Application Prototyping, eine Entwicklungsweise bei der schnellstmöglich Ergebnisse erzielt werden. Auch das Testen der Anwendung mit Eingabegeräten ist dabei ohne das komplette VR-Setup z.B. über die Tastatur simuliert möglich. Die Manipulation der Szene erfolgt mittels einer Skriptsprache, über die den 3D-Objekten unkompliziert ein Verhalten zugewiesen werden kann. [2] [6]

Vorteile von VR-Autorenwerkzeugen sind: Es sind 3D-Modelle aller gängigen Formate importierbar. Sie unterstützen sowohl verschiedene Plattformen als auch VR-Geräte. Sie ermöglichen die rapide Entwicklung navigierbarer 3D-Szenen. Es stehen Bibliotheken mit Objektverhalten zur Verfügung und auch Mehrbenutzerfähigkeit gehört zum verwendbaren Funktionsumfang. In Verbindung mit kommerzieller Hardware wird eine hohe Performanz erreicht. [6] [11]

Nachteile von VR-Entwicklungsumgebungen sind: Spezielle Interaktionen sind schwierig oder sogar unmöglich zu implementieren, da nicht in die dafür notwendige Tiefe der 3D-Grafik eingegriffen werden kann. [6] [11]

4.2 Alice



User Interface von Alice

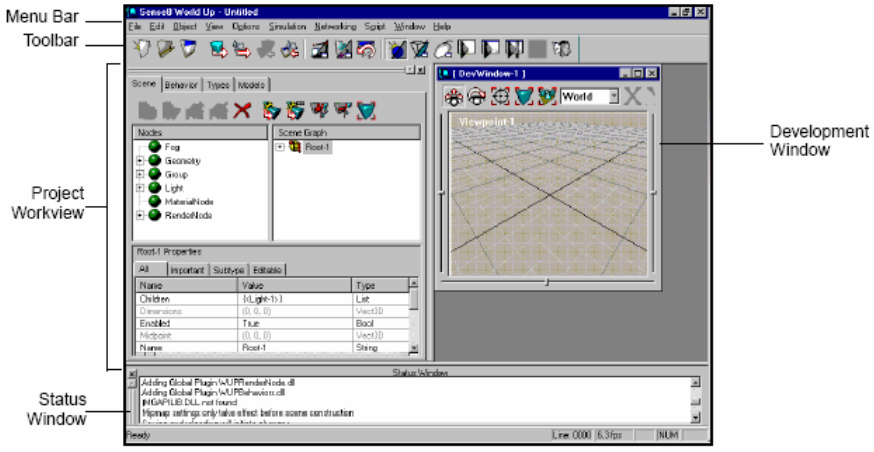
Alice ist ein Rapid Prototyping System für VR, das seit 1998 an der Carnegie Mellon University in Pittsburgh (PA) entwickelt wird. Alice unterstützt ausschließlich die Windows-Plattform und ist frei erhältlich. Die frei erhältliche Version unterstützt keine VR-Hardware, sondern nur Maus und Tastatur als Eingabegeräte. Sie ist auf Anfrage jedoch erweiterbar für HMDs, Datagloves usw. Alice ist ein High-Level VR-Autorenwerkzeug, da das VE über ein GUI zusammengestellt wird. Das Ziel bei der Entwicklung war es, ein für den User einfach zu handhabendes Werkzeug bereitzustellen, das ohne jeglichen technischen Hintergrund bedient werden kann. Als Manipulationssprache

verwendet Alice Python, eine objektorientierte Skriptsprache, die es Unerfahrenen erlaubt durch ihren gut lesbaren Programmcode einfach Alice-Scripte zu schreiben. [6] [10]

Einschränkungen bei Alice sind: Zur Verwendung von VR-Hardware wird die Developer-Version benötigt. Außerdem bietet Alice keine volle Kontrolle über die Polygone und Vertices von Geometrien, weshalb bei großem Datenvolumen in Verbindung mit Updates für jeden Frame mit Performanzproblemen zu rechnen ist. [6] [10]

4.3 WorldUp

WorldUp ist ein 3D Real-Time GUI der Firma Sense8, das aktuell bei Version 5 steht. Es handelt sich um eine kommerzielle Software die sowohl auf Windows-, UNIX- und Grafikplattformen wie SGI läuft. WorldUp bietet echte VR-Hardware Unterstützung. Das Werkzeug kommt sowohl mit diversen Eingabegeräten wie Space-Mouse, Space-Ball, Data-Gloves, Ausgabegeräten wie HMDs, Stereo Displays, Sound als auch mit High-End- und Low-End-Trackern zurecht. WorldUp ist ebenfalls wie Alice eine High-Level-Anwendung. [6] [9]



User Interface von WorldUp

Zusätzlich zum Import von 3D-Daten der Formate VRML, 3DS, DXF und FLT ist aber auch noch ein Modellierer integriert. Das Prinzip der Erlangung schneller Ergebnisse wird insbesondere durch die Möglichkeit der Änderung von Parametern des Designs in Echtzeit realisiert. Zur Verbreitung über das World Wide Web stehen Browser Plugins zur Verfügung.

Die Manipulation der Szene erfolgt in WorldUp über eine Visual Basic ähnliche Skriptsprache, aber auch OpenGL und C können stattdessen benützt werden. [9]

Resultierend aus allen Features ist WorldUp die zurzeit verbreitetste VR-Simulations Software. [9]

5 VR-Frameworks

5.1 Unterschied zwischen Framework und API

Eine API (Application Programming Interface) ist die Schnittstelle eines Betriebssystems oder einer Software über die standardisierte Programmbausteine wie z.B. Methoden zur Verfügung gestellt werden. Dieses Prinzip garantiert die Portabilität des Programmcodes, wenn die API für verschiedene Plattformen implementiert wurde. Da der Programmierer sich

nur um die Programmbausteine der Schnittstelle kümmern muss, erleichtert ihm dieses Verbergen der unterliegenden Komplexität die Arbeit. [21]

Ein Framework ist ein Programmgerüst in das an bestimmten Stellen Anwendungsspezifischer Programmcode eingehängt wird. Es verhält sich folglich wie eine Art Standard-Softwarearchitektur. Der Hauptkontrollfluss einer mittels eines Frameworks entwickelten Anwendung, steckt im Framework. Dadurch ist ein gemeinsames Verhalten aller auf Grundlage desselben Frameworks implementierter Anwendungen garantiert. Die eigentliche Applikation wird vom umgebenden Framework aus aufgerufen. [20]

5.2 Klassifizierung von VR-Frameworks

VR-Frameworks, VR-APIs und Grafikprogramme allgemein lassen sich in ein Schichtenmodell gliedern. Dieses Schichtenmodell besteht aus dem High-Level, Low-Level, Hardwarenahem-Level und dem Hardware-Level. [24]

VR-Grafiksoftware der High-Level Schicht unterteilt sich in Anwendungen wie z.B. WorldUp und Frameworks wie z.B. WorldToolKit. Beide Kategorien bieten Funktionalität zur Unterstützung beim Modellieren von 3D-Objekten, Interagieren mit diesen und Rendern von Szenen. Dazu stehen bei VR-Autorenwerkzeugen ein GUI mit Drag&Drop-Funktionalitäten und evtl. eine einfache Skriptsprache und ein integrierter Modellierer zur Verfügung. Für alle High-Level Systeme gilt: Sie übernehmen die Berechnungen und Verwaltung der Geometrien. [9] [24]

Bei VR-Grafiksoftware der Low-Level Schicht handelt es sich ausschließlich um APIs und Frameworks wie z.B. das MR Toolkit. Sie bieten beste Performanz und Flexibilität zur Lösung eines speziellen grafiknahen Problems. Dem Programmierer stehen viel mehr Operationen zur Verfügung so dass größerer Einfluss auf die Grafik und die Möglichkeit mit den Geometrien zu arbeiten besteht. Dies macht die Handhabung gegenüber High-Level Vertretern selbstverständlich viel komplexer, da z.B. Kenntnisse in Linearer Algebra zur Transformation der Geometrien nötig sind. [2] [24]

Auf der hardwarenahen Schicht wird direkt mit der Hardware kommuniziert. Wichtigster Vertreter hier ist Microsofts DirectDraw aus DirectX. [24]

Auf Hardware-Ebene finden sich die Grafikkarten wie z.B. nVidia's GeForce und ATI's Radeon, die heutzutage einen Grossteil der Grafikberechnung übernehmen. [24]

Schicht	Beispiel
Anwendungen (High-Level)	3D Studio MAX, Maya, WorldUp, Alice, ...
High-Level (API/Framework)	Java3D, OpenInventor, WorldToolKit, VR Juggler, dVISE, IRIS Performer
Low-Level (API/Framework)	OpenGL, Direct3D, CAVE, MR Toolkit
Hardwarenah	DirectDraw (Windows)
Hardware	nVidia, GeForce, ATI, ...

Schichtenmodell der Grafiksoftware [24]

5.3 Direct3D

Direct3D ist eine kommerzielle 3D-Grafik API von Microsoft für Windows. Aktuell ist die Version 9.0 aus DirectX 9.0. Direct3D ist wie DirectDraw und DirectShow eine Komponente aus Microsoft's DirectX Multimedia Toolkit. Die 3D-API baut auf die Windows 2D-API DirectDraw auf, weshalb auch Kenntnisse darüber zur Entwicklung nötig sind. Direct3D ist

keine richtige VR-API sondern eben eine 3D-API, da z.B. keine direkte VR-Hardware Unterstützung verfügbar ist. Hierzu ist die Hinzunahme weiterer Komponenten erforderlich.

Direct3D hat zwei verschiedene Modi die zum Entwickeln verwendet werden können. Der Immediate Mode (Direct3D IM) bietet Low-Level Funktionalität, also komplexe Operationsmöglichkeiten und größere Freiheit im Umgang mit den 3D-Grafiken. Der Retained Mode (Direct3D RM) bietet High-Level Funktionalität, die auf dem darunter liegenden Immediate Mode aufsetzt. In diesem Modus besteht die Möglichkeit mit einem hierarchischen Szenegraph Interface zu arbeiten. Direct3D übernimmt im Retained Mode das Geometrie-Management. Leider ist der RM sehr langsam und ein veraltetes DirectX-Beiwerk, das von Microsoft schon lange nicht mehr weiterentwickelt wird. [13] [14]

5.4 MR Toolkit

Das MR Toolkit (Minimal Reality Toolkit) wurde von der Computer Research Group der Universität von Alberta (CA) entwickelt. Es ist ein vollständiges, für Forschung und Wissenschaft frei verfügbares VR-Framework. Konzipiert wurde es ursprünglich für die Plattformen UNIX, HP, SGI und IBM. Um auch für Windows präsent zu sein, entwickelte man MR Objects, ein in C++ geschriebenes Pendant. MR Toolkit dient als Grundlage vieler High-Level Anwendungen bzw. Toolkits. [6] [22]

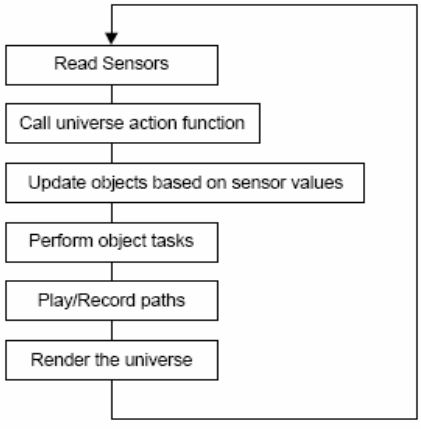
Dieses Low-Level Framework kann im Quellcode eines C, C++ oder FORTRAN Programms verwendet werden. Es steht dabei keine nutzbare Kollisionserkennung und keine Mehrbenutzerfähigkeit zur Verfügung. Beide Funktionalitäten müssen selber programmiert werden. Bei der Implementierung an der Universität von Alberta war es das Ziel hardwarenahe Virtual Environment Entwicklung zu ermöglichen. [6] [12]

Das MR Toolkit ist VR-Hardwarefähig. Neben Ein- und Ausgabegeräten wie sie schon angesprochen wurden, können auch Trackingsysteme von Ascension Technologies und Polhemus eingesetzt werden. Ein Nachteil ist, dass die VR-Ausgabe schwerpunktmäßig auf HMDs ausgerichtet ist. Es besteht keine Möglichkeit projektionsbasierte Geräte wie z.B. eine CAVE oder eine Holobench zu bedienen. [6][22]

5.5 WorldToolKit (WTK)

Das WorldToolKit ist ein kommerzielles VR-Framework der Firma Sense8. WTK ist mit Version 10 das derzeit wohl meistbenutzte VR-Entwicklungstool auf dem Markt. Es ist nicht in jeder Hinsicht das beste Tool, aber das insgesamt umfassendste und es ist auch crossplattform-portabel (SGI, Sun, Linux und Windows). [9]

Es bietet Unterstützung für ein breites Spektrum an VR-Geräten (ca. 10 Eingabe-, Ausgabe- und Trackinggeräte). Das WorldToolKit ist in C implementiert und kann in C/C++ Quellcode verwendet werden. Aufgabe des Entwicklers ist die Manipulation der Szene, während das System die Geometrien managet. Die Vielzahl an Funktionalitäten wie z.B. Kollisionserkennung, ist auf insgesamt 20 Klassen verteilt. Der Import vieler gängiger 3D-File-Formate ist



WTK Grafikdarstellung [23]

gewährleistet und eine eigenen Sound API rundet das Angebot ab. [6] [12]

Durch die Hinzunahme von World2World (ebenfalls Sense8) als WTK-Server, ist es möglich Multiuserfähige VEs zu erzeugen. Die SGI-Version von WTK unterstützt Multi-Pipe und ist somit zur Darstellung in CAVEs oder auf Holobenches geeignet. Ein mittels WTK erzeugtes VE wird vom so genannten WTK-Universe umfasst, auf dem ständig der in der Abbildung dargestellte WTK Control Loop zum Rendering der Geometrien ausgeführt wird. [23]

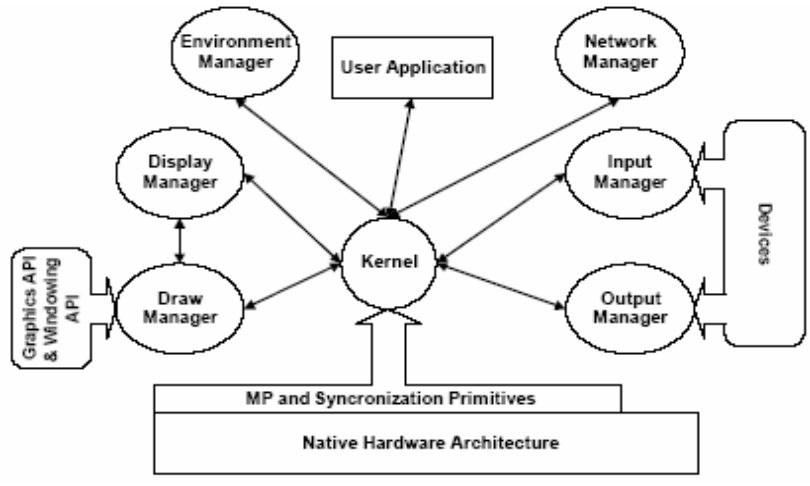
5.6 VR Juggler

Der VR Juggler wurde an der Iowa State University am ICEMT (Iowa Center for Emerging Manufacturing Technology) entwickelt. Es handelt sich dabei ebenfalls um ein VR-Framework, welches aber für Wissenschaft und Forschung frei erhältlich ist. Der VR Juggler läuft auf SGI-, HP-UX- und Windows-Plattformen. [6] [15]

Das Framework bietet VR-Hardwareunterstützung für 3D-Eingabe und Tracking. Die Grafikausgabe ist auf Projektionsgeräte ausgelegt, da am ICEMT ein solches Gerät namens C2 entwickelt wurde. Des Weiteren ist VR Juggler zusätzlich HMD-fähig. [16]

Vorgabe bei der Implementierung des VR Juggler's in C++ war, dass sich die Entwickler die dieses Tool später nutzen werden voll auf die Programmierung der VR-Welten konzentrieren können, anstatt sich Gedanken über das System zu machen auf dem diese später laufen werden. Deshalb wurde VR Juggler plattformunhabhängig konzipiert. Ein weiteres Ziel war es, ein skalierbares VR-Framework zur Lösung in C++ geschriebener VR-Anwendungen aller Arten zu entwickeln. Zur Erleichterung der Entwicklung wurde ein Performance Monitoring mittels eines Time-Stamp Verfahrens eingebaut, so dass Latenzzeiten wie z.B. die Dauer vom Signaleingang eines Eingabegerätes bis zum Update der Grafik gemessen werden kann. [17]

VR Juggler besteht intern aus Komponenten für die einzelnen Aufgaben, die so genannten Managers. Der Kernel kontrolliert alle diese Manager und übernimmt die Kommunikation zwischen ihnen. Ausschließlich vom Kernel werden Informationen an die eigentliche Applikation geliefert. Der Input-Manager handhabt die Eingabe- und Positionsgeräte (Tracker). Er unterscheidet digitale Signale wie einen gedrückten Mausbutton und analoge Signale wie sie z.B. ein Joystick liefert. Der Output-Manager spricht die Grafikanzeigegeräte über ein abstraktes Interface an, wodurch es irrelevant ist um was für eins es sich speziell handelt. Der Draw-Manager kapselt die Grafik-API, ruft die Zeichenfunktionen auf und kontrolliert die Stereosicht. Der Display-Manager hält die Informationen der Grafikfenster wie z.B. deren Größe und Position. Außerdem steuert er die Grafik-Pipeline. Der Network-Manager kann jede beliebige Netzwerk-Datenübermittlungs-Methode implementieren (CORBA, Sockets, ...). Der



Architektur des VR Juggler Frameworks [16]

Environment-Manager bietet ein Laufzeit-Kontrollsystem, das die Beobachtung der Parameter einer laufenden Anwendung ermöglicht. [16] [18] [19]

6 Fazit

Es wurden einige Möglichkeiten zur Entwicklung einer VR-Anwendung betrachtet. Sowohl VR-Autorenwerkzeuge als auch VR-Frameworks bieten zur Entwicklung gewisse Vor- und Nachteile. Die Wahl des besten Weges bzw. der Best Practice für ein VR-Projekt bedeutet die richtige Wahl der Komponenten zu treffen. Zukünftig sollte die Entwicklung bei größerem Funktionsangebot trotzdem einfacher werden. Es bleibt abzuwarten ob sich umfassendere GUI-Tools gegenüber evtl. transparenteren und unkompliziert zu erlernenden Frameworks bzw. APIs durchsetzen. Möglich wäre auch die Erzeugung von Virtual Environments über Tangible User Interfaces. Was auch immer die Zukunft bringt, es sollte ein weniger undurchdringlicher Dschungel von VR-Tools sein, als er es gegenwärtig ist. [6]

7 Quellenverzeichnis

[1] Virtuelle Produktentwicklung:
Institut für Rechneranwendung in Planung und Konstruktion (RPK) (TU Karlsruhe)

[2] Principles of VR software:
http://www.cs.umu.se/kurser/TDBD12/HT00/lectures/software.pdf

[3] Components for a virtual environment:
http://www.acmc.uq.edu.au/pdfs/paperpaderborn.pdf

[4] A VR based safety training in a petroleum refinery:
http://webster.fh-hagenberg.at/staff/haller/research/publications/1999/eg99.pdf

[5] SAVE (Safety Virtual Environment):
http://webster.fh-hagenberg.at/staff/haller/research/publications/2001/haller@web3d2001.pdf

[6] Software Tools for Virtual Reality Application Development:
http://www.vrjuggler.org/pub/vr.dev.tools.1998.SIGGRAPH98.pdf

[7] VR-Software im LRZ:
http://www.lrz-muenchen.de/services/peripherie/vr/vrsoftware.html

[8] Vorlesung „Interaktive virtuelle 3D-Welten“:
http://www.medien.informatik.uni-muenchen.de/lehre/ws0304/iv3d/material.html

[9] VR-Software der Firma Sense8:
http://www.sense8.com/products/index.html

[10] Alice – A 3D-Authoring-System:
http://www.alice.org

[11] VR software classification:
http://www.cs.unt.edu/~steiner/5330-VR/lecture2.html

[12] VR-Software Liste:
http://www.diverse.vt.edu/VRSoftwareList.html

[13] DirectX 9.0:
http://msdn.microsoft.com/msdnmag/issues/03/07/DirectX90/default.aspx

[14] Einführung in DirectDraw und Direct3D:
http://www.cs.wpi.edu/~matt/courses/cs563/talks/cbyrd/pres3.html

[15] VR Juggler – Open Source Virtual Reality:
http://www.vrjuggler.org/about.php

[16] VR Juggler – A Framework for Virtual Reality Developement:
http://www.vrjuggler.org/pub/vr.juggler.framework.for.vr.dev.1998.IPT98.pdf

[17] VR Juggler – The Programmer´s Guide:
http://www.cvmt.auc.dk/education/teaching/e03/cvg9/vr/vr.juggler.programmer.guide_pages.pdf

[18] VR Juggler – An Open Source Platform for Virtual Reality Applications:
http://www.vrjuggler.org/pub/vrjuggler-aiaa2002.pdf

[19] VR Juggler – A Virtual Platform for Virtual Reality Application Development:
http://www.vrjuggler.org/pub/ieeeVR.2001.vrjuggler.slides.pdf

[20] Was ist ein Framework:
http://de.wikipedia.org/wiki/Framework

[21] Was ist eine API:
http://de.wikipedia.org/wiki/API

[22] MR Toolkit – A Virtual Reality Framework:
http://www.cs.ualberta.ca/~graphics/MRToolkit/

[23] Introduction to WTK (WorldToolKit):
http://cggmwww.csie.nctu.edu.tw/courses/vr/2001/doc/relative/Intro_to_WTK.ppt

[24] Vorlesung „Computergrafik”:
http://www.ifi.unizh.ch/~noser/COURSES/cgVorlesung4.pdf

[25] Introduction into Virtual Reality Environments:
http://www.it.uu.se/edu/course/homepage/igs/ht03/lectures/igs_01_intro_vr.pdf