Ein leistungsfähiges System zur Online-Präsentation von Sequenzen komplexer virtueller 3D-Szenen

Dem Fachbereich Elektrotechnik und Informationstechnik

der Universität Hannover

zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte

Dissertation

von

Dipl.-Ing. Stephan Olbrich

geboren am 31. März 1961 in Braunschweig

2000

Referent: Prof. Dr.-Ing. H. Pralle
Korreferent: Prof. Dr.-Ing. C.-E. Liedtke
Tag der Promotion: 02.05.2000

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Regionalen Rechenzentrum für Niedersachsen (RRZN) sowie am Lehrgebiet Rechnernetze und Verteilte Systeme (RVS) der Universität Hannover.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. H. Pralle für die wissenschaftliche Betreuung dieser Arbeit sowie die Bereitstellung exzellenter Arbeitsmöglichkeiten am RRZN/RVS.

Herrn Prof. Dr.-Ing. C.-E. Liedtke danke ich für die Übernahme des Korreferats und für sein Interesse an meiner Arbeit.

Besonders danken möchte ich jedoch meiner Frau Susanne und meinen Kindern Lutz und Anton für ihr Verständnis und ihre Unterstützung, ohne die die Erstellung der vorliegenden Arbeit nicht möglich gewesen wäre.

Hannover, im Januar 2000

Stephan Olbrich

Kurzfassung

Virtual-Reality-Präsentations- und Interaktionstechnologien auf der Basis von 3D-Modellen sind inzwischen für verschiedenartige Anwendungsfelder leistungsfähig nutzbar. Als Beispiele für die klassischen 3D/VR-Anwendungen sind dreidimensionale, intuitive Nutzeroberflächen oder die Visualisierung statischer geometrischer Objekte (z. B. für Architekturentwurf, Produktdesign) zu nennen. Während diese vorwiegend für den Standalone-Betrieb ausgelegt sind, besteht jedoch zunehmender Bedarf für netzverteilte Szenarien. Derartige Anforderungen bestehen insbesondere für die Visualisierung komplexer, mehrdimensionaler und zeitabhängiger Ergebnisse aus aufwendigen Simulationsrechnungen. Dies ist ein Bereich der „Grand Challenge"-Probleme sowohl des Hochleistungsrechnens als auch bezüglich der visuellen Aufbereitung.

Eine hochqualitative Ergebnispräsentation sollte in multimediale, auf Internet-Standards basierende Informationssysteme (World Wide Web – WWW) integriert werden, um letztlich „3D/VR-Movies" zu unterstützen. Diese sollen während der ausgespielten 3D-Animation eine interaktive räumliche Navigation, Tracking sowie synchrone Exploration durch geographisch verteilte Nutzergruppen – „Tele-Immersion" – ermöglichen. In derartigen Client-Server-Konfigurationen bestehen besondere Anforderungen, die verschiedene Entwurfs- und Realisierungsaspekte der Repräsentation, Kommunikation und Präsentation betreffen, also eine sorgfältige Betrachtung und Gestaltung der gesamten Prozeßkette erfordern.

In der vorliegenden Arbeit wurden die dafür notwendigen Grundlagen und Anforderungen diskutiert. Wegen der Unzulänglichkeiten der derzeit verfügbaren, im wesentlichen auf Internet-Standards basierenden Werkzeuge wurden einige neuartige Ansätze zur leistungsfähigen Online-Präsentation komplexer virtueller 3D-Szenen in eigene Entwicklungen integriert:

- Eine Vorverarbeitung polygonaler 3D-Modelle, die in Standard-Datenformaten (Virtual Reality Modeling Language – VRML, ISO/IEC 14772:1997) vorliegen, resultiert in einer binären Repräsentation zur effizienteren Übertragung und Darstellung – einem eigenen „DVR"-Datenstromformat. Dadurch können wesentlich komplexere 3D-Szenen interaktiv verarbeitet werden als mit herkömmlichen, VRML-basierten Werkzeugen.
- Zur effizienten Präsentation wurde ein 3D-Viewer auf der Basis von OpenGL (Industrie-Standard-API für 3D-Graphik) als Plugin für WWW-Browser implementiert. Dabei werden stereoskopische Displays sowie dreidimensionale Tracking- und Interaktionsgeräte unterstützt und hochqualitative Renderingverfahren (Antialiasing, Colormanagement) angewandt.
- Unter Anwendung eines standardisierten Streaming-Steuerprotokolls (Real Time Streaming Protocol – RTSP, RFC 2326) wurde ein Client-Server-System zur Ausspielung zeitlich veränderlicher Szenenbestandteile im DVR-Format realisiert.

Hiermit wurden erstmalig Streaming- und medienspezifische Skalierungsmethoden, die ursprünglich für herkömmliche kontinuierliche Audio/Video-Medien entwickelt wurden, auf teleimmersive Szenarien adaptiert.

Auf der Basis dieser Komponenten wurde ein leistungsfähiges System zur Unterstützung der 3D-Visualisierung zeitvarianter Phänomene konfiguriert. Vorbereitete 3D-Animationen können damit übertragen und „on the fly“ stereoskopisch dargestellt und währenddessen im virtuellen Raum sowie in der Zeitachse interaktiv navigiert werden. Diese Systemarchitektur wurde in praktischen Online-*Präsentationsszenarien* erprobt, in denen VRML-Szenensequenzen vorverarbeitet, im DVR-Format auf einem 3D-Streamingserver bereitgestellt und von einer 3D-Graphikworkstation abgerufen wurden.

In einer Fallstudie wurden Messungen durchgeführt, die zeigen, daß mit dem 3D-Streamingsystem durch die optimierte Ausnutzung der verwendeten Ressourcen (Plattenspeicher, Server, Netz, Client, Graphiksubsystem) bei einer mittleren Datenrate von ca. 300 Mbit/s eine Bildrate von ca. 5 Szenen/s mit ca. 100.000 attributierten Polygonen je Szene erzielt werden kann.

Möglichkeiten für weitere Entwicklungen bieten sich durch Ergänzungen für weitere Szenarien, die z. B. auf innovativen Weitverkehrsnetz-Infrastrukturen zur verteilten Visualisierung von Ergebnissen des Höchstleistungsrechnens angewandt werden können:

Explorationsszenario

Es handelt sich hier um eine Arbeitsumgebung mit interaktiver, bidirektionaler Kopplung von Simulations-, Visualisierungs-, Präsentations- und Interaktionsmoduln, die z. B. zur Durchführung virtueller Experimente geeignet ist. Zur Erzielung eines hohen Interaktionsgrades ist die Generierung der 3D-Geometrien in den Datenfluß so zu integrieren, daß der 3D-Streamingserver einen zusätzlichen 3D-Dateneingang erhält, welcher z. B. direkt aus den Visualisierungsergebnissen eines Simulationsrechenprozesses gespeist werden kann. Simulations- und Visualisierungsparameter sollten dabei mit Hilfe einer erweiterbaren Nutzeroberfläche des Clients über einen Kontrolldatenfluß (z. B. über RTSP) interaktiv gesteuert werden können.

Diskussionsszenario

Hier soll eine kooperative Nutzung von Präsentations- oder Explorationsszenarien durch geographisch verteilte Gruppen (Computer-Supported Collaborative Work – CSCW) ermöglicht werden. Dazu müssen Synchronisationsmechanismen (z. B. Navigation, Telepointer und Streamingsteuerung) sowie Conferencing-Komponenten (z. B. Einblendung des Videobildes als animierte 2D-Textur in der 3D-Szene) in das 3D-Streamingsystem integriert werden.

Schlagworte: Animation, Real Time Streaming, Scientific Visualization, Tele-Immersion, Virtual Reality.

Abstract

Virtual Reality (VR) presentation and interaction techniques based on 3D models are now efficiently useful for various applications. Beyond the classical 3D/VR applications, such as three-dimensional, intuitive graphical user interfaces or visualization of static geometric objects, which are originally constructed as stand-alone systems – for example for architecture or product design purposes – there is an increasing need for networked scenarios. Under particular consideration are tools for visualization of complex, multi-dimensional, and multi-variable (e. g. time-dependant) simulation computations, as results of expensive super-computer calculations. This is a field of „Grand Challenges“ not only regarding High-Performance Computing (HPC), but also concerning the process chains for interactive, visual exploration.

High-quality presentation of results should be integrated in multimedia information systems, based on Internet standards (World Wide Web – WWW), in order to support „3D/VR Movies“. While playing out such a 3D animation, it should be allowed to spatially navigate through the scene by taking advantage of 3D interaction and tracking devices and stereoscopic displays, offering a method for remote exploration in a distributed system – „Tele-Immersion“. These client-server configurations require particular consideration of different design and implementation aspects with respect to representation, communication, and presentation instances.

In this thesis the basics and requirements, classified to performance, quality, and functionality issues, are discussed. Because of insufficiencies in currently available tools, which are essentially based on Internet standards, several innovative approaches for efficient online presentation of complex 3D scenes were integrated in new developments:

- A preprocessing step converts polygonal 3D models, which exist in the standard file format VRML (Virtual Reality Modeling Language, ISO/IEC 14772), into a special binary representation: the „DVR“ 3D file format. This allows more efficient transport and presentation processes, and enables to productively handle more complex scenes, compared with VRML-based tools.
- A 3D viewer, based on OpenGL (Industry standard API for 3D graphics), was implemented as a WWW browser plugin to efficiently display 3D scenes, which are coded in the DVR format. Stereoscopic displays, three-dimensional tracking and interaction devices are supported, and high-quality rendering capabilities (such as antialiasing, color management) are applied.
- A client-server system was developed, which provides server-based, synchronized play-out of time dependent scene components as DVR streams, applying the standardized streaming protocol RTSP (Real Time Streaming Protocol, RFC 2326).

As such, streaming and media-specific scaling techniques, which were originally designed for conventional continuous audio/video media, were for the first time adopted to tele-immersive scenarios.

Based on these components, an efficient system to support 3D visualization of time-dependant phenomena was configured. Using this system, prepared 3D animations are transfered and presented stereoscopically „on-the-fly“, while interactive spatial navigation as well as in the time axis are supported. This system architecture was evaluated in several applications of an online *presentation scenario,* where VRML scene sequences were proprocessed, stored in DVR format on a 3D streaming server, and accessed from a 3D graphics workstation.

In a case study, measurements have shown that the 3D streaming system allows a sustained throughput of ca. 300 Mbit/s, and an frame rate of ca. 5 scenes/s with a complexity of ca. 100.000 attributed polygons per scene. This high performance is a result of optimized utilization of the ressources in the processing pipeline: disk file system, server system, network infrastructure, client system, and graphics subsystem.

Future prospects are given in the supplement of the developed mechanisms to support further scenarios, which could be applied on innovative wide area networking infrastructure, to support distributed visualization of HPC results:

Exploration scenario

The working environment should be extended by interactive, bidirectional coupling of simulation, visualization, presentation, and interaction moduln, in order to handle a virtual experiment paradigm. To achieve a high level of interaction, the generation of 3D geometries has to be integrated into the data flow in a way that the 3D streaming server will get an additional 3D data input, which could be driven by visualized results of a simulation process. Simulation and visualization parameters should then be controlled interactively by an extensible user interface at the client, by using a control data flow (e. g. over RTSP).

Discussion scenario

The presentation and exploration scenarios as described above should be used cooperatively by geographically distributed virtual environments (Computer-Supported Collaborative Work – CSCW). Required for this purpose are extensions to realize synchronization mechanisms (e. g. navigation, telepointer, streaming control) and audio-visual conferencing components (e. g. video as part of the virtual 3D scene by an animated 2D texture).

Keywords: Animation, Real Time Streaming, Scientific Visualization, Tele-Immersion, Virtual Reality.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API – Application Programming Interface
ATM – Asynchronous Transfer Mode (www.atmforum.com)
AVS – Application Visualization System (www.avs.com)
BIFS – Binary Format for Scenes (MPEG-4)
BOOM – Binocular Omni-Orientation Monitor (Fa. Fakespace)
BSD – Berkeley Software Distribution
B-WiN – Breitband-Wissenschaftsnetz des DFN
CAVE – Cave Automatic Virtual Environment (Fa. Pyramid)
CAVERNUS – CAVE Research Network Users Society (www.ncsa.uiuc.edu/VR/cavernus/)
CD-ROM – Compact Disk – Read Only Memory
CD-RW – Compact Disk – Read/Write
CGI – Common Gateway Interface (WWW-Server-Schnittstelle zur Generierung dynamischer Dokumente) oder Computer Graphics Interface (ISO/IEC 9636)
CGM – Computer Graphics Metafile (ISO/IEC 8632)
CGRM – Computer Graphics Reference Model (ISO/IEC 11072)
CIE – Commission Internationale de l'Éclairage (International Commission on Illumination)
CIEXYZ – Farbmodell auf der Basis geräteneutraler, imaginärer Primärfarben
CMS – Color Management System
CMYK – Farbmodell Cyan/Magenta/Yellow/Key (Black)
COM – Component Object Model
CORBA – Common Request Broker Architecture
CSCW – Computer-Supported Collaborative Work
CSMA/CD – Carrier Sense Multiple Access with Collision Detection
DAT – Digital Audio Tape
DCOM – Distributed Component Object Model (Fa. Microsoft) [115]
DFN – Verein zur Förderung eines Deutschen Forschungsnetzes e. V.
DGM – Digitales Geländemodell
DLL – Dynamic Link Library
DSO – Dynamic Shared Object

DVD – Digital Versatile Disk
DVR – DocShow-VR; auch: Distributed Virtual Reality
EEPROM – Electrically Erasable Programmable Read Only Memory
EPROM – (UV-) Erazable Programmable Read Only Memory
FTP – File Transfer Protocol (RFC 959)
GIF – Compuserve Graphics Interchange Format
GKS – Graphical Kernel System
(ISO/IEC 7942, auch GKS-3D: ISO/IEC 8805)
GLR – GL Render Server Facility [94]
GLS – OpenGL Stream [36]
GUI – Graphical User Interface
HMD – Head-Mounted Display
HMMI – Human Multimedia Interaction
HTML – Hypertext Markup Language (RFC 1866)
HTTP – Hypertext Transfer Protocol (RFC 2068)
ICC – International Color Consortium (www.color.org)
ICM – Image Color Matching (Microsoft)
IEC – International Electrotechnical Commission (www.iec.ch)
IETF – Internet Engineering Task Force (www.ietf.org)
IFIP – International Federation for Information Processing (www.ifip.or.at)
IID – Initiative Informationsgesellschaft Deutschland (www.iid.de)
IP – Internet Protocol
ISO – International Organization for Standardization (www.iso.ch)
ITU – International Telecommunication Union (www.itu.int)
JPEG – Joint Photographic Experts Group
(JPEG-Kompression: ISO/IEC 19018)
KCMS – Kodak Color Management System
LCD – Lyquid Crystal Device
LLC – Logical Link Control
LWP – Light-Weight Process (auch: Light-Weight Protocol)
LZW – Kompressionsverfahren nach Lempel-Ziv & Welch,
z. B. in GIF, TIFF verwendet
MAC – Media Access Control
MHEG – Multimedia and Hypermedia Information Coding Expert Group
(www.mheg.org; MHEG-1: ISO/IEC 13522)
MIME – Multipurpose Internet Mail Extensions (RFC 2045)
MJPEG – Motion JPEG
MM – Multimedia

MPEG – Moving Pictures Experts Group (www.mpeg.org) (MPEG-1: ISO/IEC 11172, MPEG-2: ISO/IEC 13818, MPEG-4, MPEG-7)
MPI – Message-Passing Interface
MPP – Massiv-Parallelrechner
NURBS – Non-Uniform Rational B-Splines
OMG – Object Management Group (www.omg.org)
OSI – Open Systems Interconnection (ISO/IEC 7498)
PAC – Proxy Automatic Configuration
PCMCIA – Personal Computer Memory International Association (www.pcmcia.org)
PDU – Protocol Data Unit
PHIGS – Programmer's Hierarchical Interactive Graphics System (ISO/IEC 9592)
PVM – Parallel Virtual Machine
PVP – Paralleler Vektorrechner
QoS – Quality of Service
RAID – Redundant Array of Inexpensive Disks
RAM – Ramdom Access Memory
RFC – Request for Comments (Internet-Standard) (www.rfc-editor.org)
RGBA – Farbmodell Rot/Grün/Blau/Alpha
RPC – Remote Procedure Call
RRZN – Regionales Rechenzentrum für Niedersachsen, Universität Hannover (www.rrzn.uni-hannover.de)
RSVP – Resource Reservation Protocol
RTP – Transport Protocol for Real-Time Applications (RFC 1889)
RTSP – Real-Time Streaming Protocol (RFC 2326)
RVS – Lehrgebiet Rechnernetze und Verteilte Systeme, Universität Hannover (www.rvs.uni-hannover.de)
SDK – Software Development Kit
SGI – Silicon Graphics, Inc. (www.sgi.com)
SGML – Standard Generalized Markup Language (ISO/IEC 8879)
SMIL – Synchronized Multimedia Integration Language [197]
sRGB – Standard default RGB color space (www.srgb.com)
TCP – Transmission Control Protocol
TLI – Transport Level Interface
TIFF – Tagged Image File Format (Fa. Adobe) [1]
UDP – User Datagram Protocol
URL – Uniform Resource Locator (RFC 1737)
VTK – Visualization Toolkit [161]

VR – Virtual Reality
VRML – Virtual Reality Modeling Language (ISO/IEC 14772)
YUV – Farbmodell der Videotechnik: Luminanz (Y) / Chrominanz (U, V)
W3C – World Wide Web Consortium (www.w3.org)
WDM – Wave Division Multiplex
Web3D – Web 3D Consortium (www.web3d.org)
WWW – World Wide Web
XML – Extensible Markup Language [196]

1 Einleitung

1.1 Einführung und Motivation

In Wissenschaft und Industrie stehen heute leistungsfähige computergestützte Arbeitsumgebungen für Forschungs- und Entwicklungszwecke zur Verfügung. Dies betrifft insbesondere die Ausstattung an Arbeitsplatzrechnern und leistungsstarken Servern sowie Kommunikationsnetzen. Zunehmend ist durch qualitativ hochwertige Video- und 3D-Graphik-Unterstützung anwendungsreife Multimediatechnologie, bis hin zu Methoden der Virtuellen Realität, gegeben. Internet-Technologien haben sich hier durch Standardisierung, breite Verfügbarkeit und Zuverlässigkeit nicht nur im Weitverkehrsbereich, sondern auch in lokalen Netzanwendungen – „Intranets“ – durchgesetzt. Das World Wide Web (WWW) bildet dafür einen generischen Rahmen, in dem verschiedenartige Netzdienste gekapselt und unterstützt durch multimediale Elemente komfortabel genutzt werden können.

Das hohe Leistungspotential der Komponenten einer solchen Infrastruktur wird allerdings häufig nicht ausgeschöpft, wie sich im Vergleich mit den jeweiligen Spezifikationen der beteiligten Geräte zeigt. Die Ursachen sind vielfältig, z. B. wurden die verfügbaren Werkzeuge zum Teil ursprünglich für die Anwendung in Einzelplatzsystemen konzipiert, oder die jeweilige Auslegung wurde für den Einsatz in Kommunikationsnetzen mit relativ niedrigen Transferraten optimiert. Auf der anderen Seite bestehen in bestimmten Szenarien erhebliche Anforderungen in bezug auf Qualitäts- und Funktionalitätsaspekte, so daß produktive Anwendungen wegen vorhandener Leistungsdefizite zum Teil verhindert werden. Daher besteht Bedarf, die potentiellen Engpässe zu identifizieren, um letztlich zu einer Erhöhung der Leistungsfähigkeit entsprechender Produktionsketten beizutragen.

1.2 Generelle Ziele der Arbeit

Die vorliegende Arbeit soll einen Beitrag zu einer Verbesserung der Online-Präsentation virtueller 3D-Szenen im Kontext wissenschaftlicher Visualisierungsanwendungen leisten. Dies geschieht unter besonderer Berücksichtigung der Anforderung, hochkomplexe, zeitabhängige, dreidimensionale, geometrische Repräsentationen, wie sie z. B. bei der visuellen Aufbereitung aufwendiger Simulationsrechnungen auf Supercomputern entstehen können, mit Hilfe leistungsfähiger Verfahren als hochqualitative, multimediale Online-Dokumente zu präsentieren.

Dabei ist auf eine akkurate Darstellung (z. B. verlustfreie Codierung), auf möglichst kurze Latenzzeiten (z. B. Startup beim Zugriff auf ein Dokument oder Reaktionszeit bei der Navigation im virtellen Raum) sowie eine hohe Aktualisierungsfrequenz bei der Darstellung von Szenenänderungen (z. B. Update-Rate für Sequenzen virtueller 3D-Szenen) zu achten.

In einem derartigen Anwendungsszenario, welches zur Unterstützung der in wissenschaftlichen Kontexten üblichen Exploration, Präsentation und Kooperation dienen kann, muß eine möglichst korrekte Ergebnisinterpretation gewährleistet werden. Aufgrund dieser fundamentalen Anforderung werden verlustbehaftete Kompressionsverfahren in den hier behandelten Produktionsketten generell als nicht zulässig betrachtet.

Unter Ausnutzung der in diesem Anwendungsumfeld üblicherweise vorhandenen performanten Ressourcen

- Kommunikationsnetze: z. B. lokales Hochschulnetz, Breitband-Wissenschaftsnetz, Gigabit-Pilotnetze;
- Arbeitsplatzrechner: z. B. 3D-Graphiksupercomputer mit Präsentations- und Interaktionsgeräten zur Unterstützung von Methoden der Virtuellen Realität;
- Server für verschiedenartige Aufgaben: z. B. WWW, Multimedia-Streaming, Compute-Server

wurde ein effizientes, verteiltes System entwickelt, welches zur qualitativ hochwertigen Betrachtung komplexer 3D-Szenen dient, die von entfernten Servern abgerufen werden.

Dabei wurden folgende innovative Ansätze eingebracht:

- Entwicklung eines speziellen Datenstromformats – hier genannt: „DVR“ – zur Beschreibung von 3D-Szenen, welches auf hohe Transport- und Präsentationseffizienz optimiert sowie für die Unterstützung qualitativ hochwertiger Virtual-Reality-Präsentationsmethoden ausgelegt ist;
- Entwicklung von Verfahren zur Erzeugung des DVR-Formates:
  - Methoden zur direkten Generierung des DVR-Formats zwecks Integration in Simulations- und Visualisierungsumgebungen;
  - Prozesse zur Vorverarbeitung von VRML-Dateien in das DVR-Format, um eine Interoperabilität mit dem VRML-Standard zu erzielen;
- Entwicklung eines effizienten 3D-Viewers, welcher als Erweiterung von WWW-Browsern konstruiert wurde und damit die Einbettung von 3D-Szenen in WWW-Seiten gestattet sowie Hypermedia-Fähigkeiten aufweist;
- Entwicklung eines 3D-Streamingverfahrens zur Realisierung von zeitlich und örtlich während der Abspielung frei navigierbaren „3D-Filmen“. Dabei wurden Methoden angewandt, welche üblicherweise im Zusammenhang mit der Ausspielung von Video-/Audio-Medienströmen eine Rolle spielen:
  - Streaming-Steuerprotokoll: „Real Time Streaming Protocol“ (RTSP);
  - Medienspezifische Skalierungsoptionen:
    z. B. Detaillierungsgrad, Bildrate.

In der Arbeit wurden nach analytischer Betrachtung die zur Erreichung der oben erläuterten Ziele notwendigen Designentscheidungen getroffen, Systeme entworfen und schließlich die praktische Umsetzung verfolgt. Eine Erprobung und Evaluierung der Implementierung wurde anhand von Fallstudien durchgeführt.

1.3 Aufbau der Arbeit

In *Kap. 2* werden die zur Durchführung und zum Verständnis dieser Arbeit benötigten Basistechnologien – *Multimedia* und *Virtual Reality im Netz*, *wissenschaftliche Visualisierung* – grundlegend diskutiert.

Kap. 3 stellt eine Problemanalyse dar. Nach einer Charakterisierung des gegebenen Anwendungsszenarios wird zunächst auf Leistungspotentiale sowie Unzulänglichkeiten der in diesem Kontext existierenden Systeme eingegangen. Für die kritischen Instanzen *Präsentation*, *Kommunikation* und *Repräsentation* der Verarbeitungskette werden dann jeweils die Aspekte *Leistungsfähigkeit*, *Qualität* und *Funktionalität* im Vergleich zum Stand der Technik erörtert.

Auf das Design und die Implementierung der entstandenen Systemarchitektur wird in *Kap. 4* eingegangen. Hier wird die Auslegung und praktische Realisierung der Grundbausteine – 3D-Dateiformat „DVR“, Vorverarbeitung, Generierung, DVR-Viewer, DVR-Streamingverfahren – behandelt.

Im Rahmen von Fallstudien werden die entwickelten Werkzeuge in *Kap. 5* anwendungsnah bewertet. Auf der Basis verschiedener 3D-Szenen-Datensätze wurden Leistungsvergleiche zwischen einem Standard-VRML-Viewer und dem DVR-Viewer durchgeführt. Die Leistungsfähigkeit des DVR-Streamingkonzepts wird anhand der Erfahrungen in einem lokalen Testbed illustriert.

Die Arbeit endet in *Kap. 6* mit einer Zusammenfassung der erzielten Ergebnisse sowie einem Ausblick auf die vom erreichten Stand ausgehenden zukünftigen Anwendungs-, Forschungs- und Entwicklungspotentiale.

2 Grundlagen

2.1 Multimedia-Technologie und Virtuelle Realität

2.1.1 Begriffsumfeld

Techniken zur Multimedia-Kommunikation im weiteren Sinn nutzen Menschen bereits seit Jahrtausenden. Von den frühen visuellen und akustischen Formen – wie Felsmalereien (ca. 30.000 v. Chr.), Hieroglyphenschrift (ca. 3.000 v. Chr.) sowie Rauch-, Trommel-, Licht- und Flaggensignale – über die Einführung des Buchdrucks (Johannes Gutenberg, um 1450) bis hin zu den elektronischen Mitteln wie Schallplatte, Telefon, Radio und Fernsehen handelte es sich zunächst um die Anwendung analoger Transport- und Speichermedien. Ein grundlegender Wandel ist seit der universellen Verfügbarkeit leistungsfähiger, miniaturisierter Computer, Speicher und Kommunikationsnetze zu beobachten, in denen unterschiedliche Darstellungsarten von Informationen auf digitaler Basis interaktiv behandelt werden können: z. B. Text, Sprache, Bild, Graphik, Bewegtbild, Videofilme und 3D-Szenarien. In der Geschichte der Medien sind drei Grundbestrebungen erkennbar [62]:

1. Überwindung von Zeit und Distanz – dies beschrieb Marshall McLuhan 1962 [112] wie folgt:

   „The new electronic interdependence recreates the world in the image of a global village."

2. Erschließung neuer Rezeptionskanäle – Ivan Sutherland entwickelte 1965 [185] die Vision:

   „The screen is a window through which one sees a virtual world. The challenge is to make that world look real, act real, sound real, feel real."

3. Erhöhung des Immersionsgrads durch größere Realitätstreue – hier sind Einflußfaktoren wie Interaktionsgrad, Szenenkomplexität, Bildauflösung, Bildrate, stereoskopisches Sehen, Sichtfeld, Tracking zu nennen.

Derzeit üblich sind zweidimensionale, fensterorientierte Nutzeroberflächen gemäß „Desktop-Metapher". Durch Anwendung dreidimensionaler, intuitiver Mensch-Maschine-Mensch-Schnittstellen entwickeln sich diese zu tele-immersiven Virtual-Reality-Systemen, die durch innovative Präsentations- und Interaktionstechnologien unterstützt werden. Insofern stellt *Virtual Reality* eine natürliche Weiterentwicklung der Multimedia-Technologie dar.

Ein Multimedia-System aus heutiger Sicht definiert Steinmetz [177] wie folgt:

„Ein Multimedia-System ist durch die rechnergestützte, integrierte Erzeugung, Manipulation, Darstellung, Speicherung und Kommunikation von unabhängigen Informationen gekennzeichnet, die in mindestens einem kontinuierlichen (zeitabhängigen) und einem diskreten (zeitunabhängigen) Medium kodiert sind.“[1]

Neben dieser technischen Sicht sei darauf hingewiesen, daß es sich hier um ein multi- und interdisziplinäres Gebiet handelt, in dem z. B. ökonomische, juristische, ergonomische, soziologische und kulturelle Gesichtspunkte stets mit berücksichtigt werden müssen. Die politischen Rahmenbedingungen begünstigten dabei in den letzten Jahren die Entwicklung, wie z. B. an verschiedenen Initiativen der G-7-Staaten [48], der EU [9], sowie auch in Deutschland [16][17] („Initiative Informationsgesellschaft Deutschland“: `www.iid.de`) festgestellt werden kann, die häufig ökonomisch („Globalisierung der Märkte“, „Information als Wirtschaftsfaktor“) oder sozialpolitisch motiviert sind.

Im weiteren steht jedoch die Betrachtung fortgeschrittener, anwendungsreifer Technologien zur Realisierung leistungsfähiger verteilter Systeme im Vordergrund:

- Hardware: Mikrocomputer, High Performance Computing (HPC)
- Software: Multimedia-Anwendungen, Client-Server-Modell
- Informationsverarbeitung und Kommunikationstechnik (IuK)

2.1.2 Grundbegriffe

In den folgenden Abschnitten werden in Anlehnung an den MHEG-Standard [85][177] die folgenden relevanten Medienbegriffe behandelt (siehe auch *Abb. 1*):

- Perzeptionsmedium – Menschliche Sinneswahrnehmung
- Präsentationsmedium – Gerätetechnik zur Darstellung, auch Akquisition
- Repräsentationsmedium – Digitale Codierung
- Speichermedium – Digitale Speicherung
- Kommunikationsmedium – Übertragungstechnik
- Informationsaustauschmedium – Online- oder Offline-Zwischenspeicher

Abschließend wird in *Abschn. 2.1.2.7* der zeitliche Aspekt erörtert, welcher in Multimedia-Systemen generelle Bedeutung besitzt.

---

[1] Die Begriffe *kontinuierlich* und *diskret* werden in diesem Zusammenhang etwas anders verwendet als im Sprachgebrauch der Nachrichtenübertragung.
In DIN 40146-1 [34] werden Zeitfunktionen hinsichtlich des Wertebereichs des Signalparameters sowie des Definitionsbereiches auf der Zeitachse in wertkontinuierliche und wertdiskrete sowie zeitkontinuierliche und zeitdiskrete Signale unterschieden. Für Nachrichtenspeicher wird darin eine Aufteilung in wertkontinuierlich oder wertdiskret bzw. ortskontinuierlich oder ortsdiskret gespeicherte Signale vorgenommen.

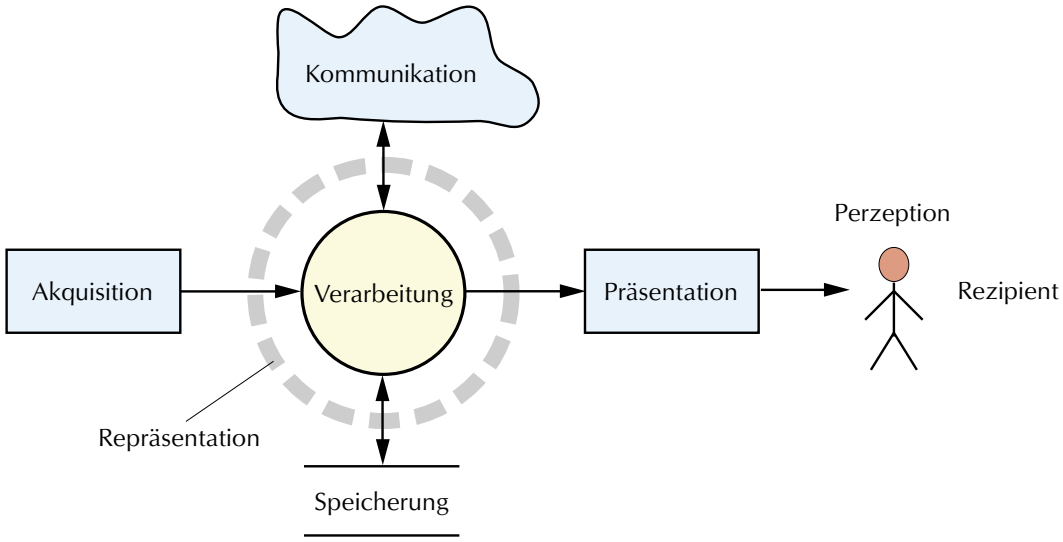


Abb. 1. Multimedia-Technologie – Abstraktes System-Modell

2.1.2.1 Perzeptionsmedien

Bei der Betrachtung eines Multimedia-Systems aus Nutzersicht wird deutlich, daß für die Maschine-Mensch-Kommunikation die verschiedenen Sinnesorgane des Menschen genutzt werden können. Entsprechend den verfügbaren Wahrnehmungskanälen kann eine Einteilung in

- visuell – z. B. Text, Einzelbild, Bewegtbild
- akustisch – z. B. Sprache, Musik, Geräusch
- haptisch – z. B. Kraftrückkopplung, taktile Interaktion
- olfaktorisch – Geruch, Geschmack

vorgenommen werden. Weiterhin spielt die Wahrnehmung des Zeitbezugs eine wichtige Rolle; dazu sei auf *Abschn. 2.1.2.7* verwiesen. Im folgenden wird im wesentlichen auf die visuellen Medien eingegangen, da die weiteren Medien nur vernachlässigbar an der in dieser Arbeit behandelten Problematik beteiligt sind.

2.1.2.2 Präsentationsmedien

Zur Darstellung visueller Medien können, je nach Zweck, verschiedenartige Technologien und Ausgabegeräte verwendet werden. Grob wird zwischen Hardcopy-Geräten – z. B. Drucker (2D-Pixelbild-Ausgabe), Stereolithographie (3D-Modell-Ausgabe) – und interaktiven Ausgabegeräten – z. B. Bildschirm, Stereoprojektion – unterschieden. Generell gelten gemäß der physiologischen Charakteristik des menschlichen Auges die folgenden Anforderungen bezüglich der Orts- und Intensitätsauflösung:

- Die Ortsauflösung des Auges liegt bei ca. 10 Zyklen je Winkelgrad [123], allerdings nur in der Mitte des Sichtbereichs.
- Die Wahrnehmbarkeit zeitlicher Änderungen nimmt oberhalb von ca. 10 Zyklen je Sekunde deutlich ab [123].
- Intensitätsunterschiede sind bei schrittweiser relativer Zunahme im Faktor 1,01 wahrnehmbar [46]. Um durch dementsprechend logarithmisch gleichmäßig zunehmende Intensitätsstufen einen Bereich normierter Intensitäten von I_0 bis 1 zu repräsentieren, werden

  $$n = \log_{1,01}(1/I_0) \tag{1}$$

  Intensitätsstufen benötigt. Je nach Gerät und Anwendungsszenario liegt der unterstützte Dynamikbereich $1/I_0$ zwischen 100 und 1000. Die Anzahl der dafür erforderlichen Intensitätsstufen beträgt gemäß *Gleichung 1*:

  $$n = 463, \ldots, 694 \ . \tag{2}$$

  Werden die Intensitäten linear abgebildet, so muß die Diskretisierung noch wesentlich feiner aufgelöst werden. Ein Intensitätsbereich von 1:100 erfordert dann beispielsweise 9900 Codes, d. h. 14 bit je Farbkomponente [151].

Die aktuell verfügbare Gerätetechnik ist jedoch, insbesondere für interaktive Anwendungen, weit von diesen Zielwerten entfernt (siehe auch [128][130]). Die tatsächliche Darstellungsqualität hängt außerdem noch von weiteren Parametern ab. Im folgenden sind einige weitere Aspekte und konkrete Realisierungen aufgeführt.

Ortsauflösung

Der Ausgabe auf Bildschirmen oder Druckern liegen Rasterbilder zugrunde, deren Auflösung gerätespezifisch ist. Üblich sind auf Computerbildschirmen – je nach Graphikkarte – 1024x768 bis 1600x1200 Pixel. Videobilder haben laut Digitalvideo-Standard ITU-R 601 [88] 720x576 Pixel, davon sind wegen ca. 10 % „Overscan“ nur ca. 640x512 Pixel sichtbar. Drucker bieten 300 bis 2540 dpi (dots per inch), je nach Druckprinzip und Bildgröße.

Bildrate

Zur Erzielung flüssiger Animationen sind mindestens 10 Bilder pro Sekunde wiederzugeben. Der PAL-Fernsehstandard schreibt 25 Bilder/s vor, die in jeweils zwei Halbbilder zerlegt werden, um eine möglichst flimmerfreie Darstellung zu bewirken. Computermonitore stellen – je nach Auflösung – 60–120 Bilder/s dar.

Farbwiedergabe

Für die rechnerinterne Abbildung von Farben sind verschiedene Modelle entwickelt worden, die in Farbräumen von jeweils 3–4 Komponenten arbeiten; z. B. RGB (rot, grün, blau – entsprechend den Primärfarben der Bildschirmgerätetechnik), CMYK (cyan, magenta, yellow, black – entsprechend der Drucktechnik), YUV (Helligkeitswert und Farbvektor – wie in der Digitalvideotechnik üblich [88]) oder CIEXYZ (empfindungsmäßig gleichabständi-

ges, geräteunabhängiges System auf der Basis imaginärer Primärfarben). Rechnerintern können Farbwerte genügend genau (gegebenenfalls mit erhöhter Wortlänge) und prinzipiell geräteunabhängig repräsentiert werden. Die Qualität der Wiedergabe hängt jedoch auch vom Prozeß der Farbraumkonvertierung und von der Genauigkeit der Darstellung auf Graphikkarte und Monitor ab. Zur Farbraumkonvertierung tragen Colormanagement-Systeme (CMS) [26][59] bei, welche sich auf Farbprofile gemäß ICC-Standard (International Color Consortium) stützen [70]. Entsprechende Realisierungen werden als C-Bibliotheken auf den verbreiteten Arbeitsplatzrechner-Plattformen angeboten, z. B. Microsoft Windows: ICM [113][116], Silicon Graphics Irix: Coloratura [168], Sun Solaris: KCMS [184] (siehe auch *Tabelle 8, S. 72*).
Graphikkarten bieten meist eine Auflösung von 8 bit je Primärfarbe (256 Stufen), gelegentlich auch bis zu 12 bit je Komponente (4096 Stufen). Zum Ausgleich der nichtlinearen Charakteristik von Bildschirmen – näherungsweise:

$$I = kU^{\gamma} \tag{3}$$

(erzeugte Intensität I, Geräte-Konstanten k und γ, normierte Steuerspannung U) mit γ im Bereich zwischen 2,2 und 2,5 für die meisten Bildröhren – wird eine sogenannte Gamma-Korrektur teilweise innerhalb der Graphikkarte hardwaremäßig durchgeführt. Zur weiteren Erläuterung der Gamma-Problematik sei auf die Literatur verwiesen [149][150][151].

Renderingverfahren

Zur perspektivischen Umsetzung einer polygonal beschriebenen 3D-Szene in 2D-Rasterbilder (Rendering) sind verschiedene Verfahren gebräuchlich [46]:

- Projektive Verfahren, z. B. Z-Buffer:
  Zunächst werden an den Eckpunkten der Polygone die 3D-Koordinaten in XY-Bildkoordinaten und Z-Tiefenwerte transformiert, und ein RGB-Farbwert wird aus den Objekt- und der Lichtquellen-Spezifikationen über ein Beleuchtungsmodell (z. B. Phong) berechnet. Anschließend werden die zu füllenden Pixel (Farb- und Z-Werte) in den Bildschirmspeicher geschrieben, sofern der jeweilige Z-Wert kleiner als der bereits im Bild vorhandene ist. Dieses Verfahren wird häufig durch 3D-Graphiksubsysteme beschleunigt, so daß auch komplexe Szenen in Echtzeit navigierbar sind. Die hardwarenahe Programmierung wird meist durch OpenGL [30][95][121][122], welches auf allen wesentlichen Rechnerplattformen verfügbar ist, unterstützt (siehe auch `www.opengl.org`).

- Strahlverfolgungsverfahren („Raytracing“):
  Je Pixel des zu erzeugenden Rasterbildes wird ein Sichtstrahl nach bestimmten Kriterien in der 3D-Szene verfolgt. Dabei können auch Spiegelungen und Schattenwürfe – im Gegensatz zum Z-Buffer-Verfahren – berücksichtigt werden. Dieses Verfahren ist relativ rechenaufwendig, kann aber für die Nutzung von Massiv-Parallelrechnern parallelisiert werden. Verbreitete Public-Domain-Software: Povray, Rayshade.

– Physikalisch motivierte Verfahren („Radiosity“):
Hier wird durch eine aufwendige Vorverarbeitung, in der die Energieverteilung auf den Szenen-Elementen berechnet wird, ein polygonales Modell erstellt, welches ausschließlich mit emittierenden Farben attributiert ist. Wird die Szene – incl. Lichtquellen – nun nicht mehr verändert, kann dieses Modell mit einem reinen Oberflächen-Renderer (z. B. Z-Buffer) dargestellt werden. Da der Renderer dann keine Lichtberechnung mehr durchführen muß, kann eine flüssigere Navigation erzielt werden.

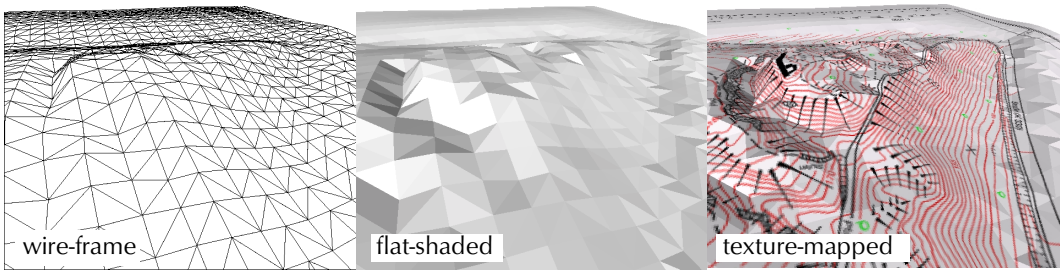


Abb. 2. Polygon-Rendering am Beispiel eines digitalen Geländemodells (DGM)
(DGM: Institut für Kartographie, Universität Hannover)

Daneben werden sogenannte „Volume-Rendering“-Verfahren verwendet, um von Volumen-Repräsentationen virtueller Objekte (Voxel) perspektivische Abbildungen zu generieren. Auf leistungsfähigen Graphiksystemen (z. B. Silicon Graphics Onyx2 Infinite Reality [169]) wird dreidimensionales Texture-Mapping als OpenGL-Extension angeboten. Dadurch wird z. B. die interaktive Visualisierung von 3D-Skalarfeldern ermöglicht, sofern die Größe des verfügbaren Texturspeichers dies zuläßt.

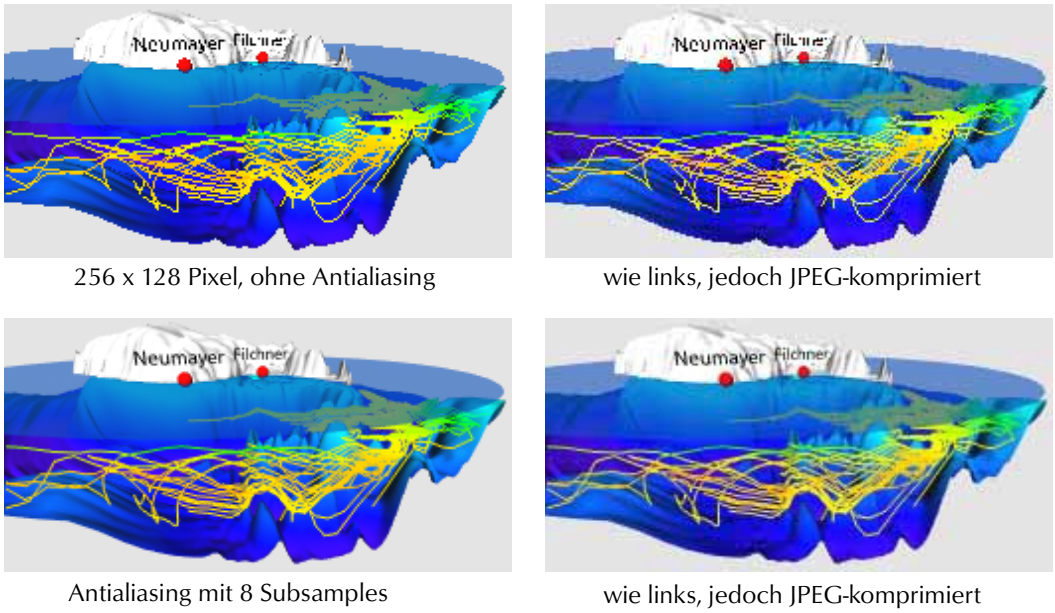


Abb. 3. Aliasing – Antialiasing, unkomprimiert – JPEG-Kompression (Zirkumpolarstrom, Alfred-Wegener-Institut für Meeres- und Polarforschung (AWI), Bremerhaven)

Anti-Aliasing
Die häufig geringe Ortsauflösung des Darstellungsgeräts kann zu Artefakten führen, wie z. B. Treppenstufen an Polygonkanten oder Moiré-Effekten bei feinen Strukturen (siehe auch *Abb. 3*). Dieses als Aliasing bezeichnete Phänomen, welches z. B. auch die Lesbarkeit von Text beeinträchtigt [15][109] [131], kann durch sogenannte Antialiasing-Verfahren reduziert werden. Dabei wird häufig die Renderingauflösung um ein Vielfaches erhöht, um anschließend ein gefiltertes Bild darzustellen, in dem dann – durch die Einbeziehung von Halbtönen – die Kanten geglättet erscheinen. Dieses Verfahren wird unter der Bezeichung „Global Scene Multisampling Antialiasing“ auf speziellen 3D-Graphik-Hochleistungsrechnern (z. B. Silicon Graphics Onyx2 Infinite Reality [169]) in Echtzeit durchgeführt und steht als gerätespezifische OpenGL-Extension zur Verfügung. Andere Methoden sind genereller anwendbar – z. B. Multipass-Rendering mit gezieltem Jitter in der perspektivischen Transformation, wobei die Filterung über den OpenGL-Accumulation-Buffer vorgenommen wird [57][121].

Artefakte durch verlustbehaftete Codierung
Kompressionsverfahren, wie sie üblicherweise für die kompakte Codierung von Bildern verwendet werden, können, wenn sie verlustbehaftet sind, zum Teil deutliche Auswirkungen auf die Bildqualität haben. Dies gilt vor allem dann, wenn das jeweilige Verfahren für Anwendungsszenarien ausgelegt ist, die sich vom konkreten Einsatzfall unterscheiden (Beispiel mit JPEG-Kompression: siehe *Abb. 3*).

2.1.2.3 Repräsentationsmedien

In diesem Abschnitt soll die Abbildung von Medien in binäre Datenströme oder Dateien erläutert werden. Dazu wird – im wesentlichen am Beispiel visueller Repräsentationen – zunächst die Einordnung in unterschiedliche Abstraktionsniveaus sowie die Dimensionalität diskutiert. Anschließend wird auf Codierung, Kompression sowie übliche Datei- und Datenstrom-Formate eingegangen.

Abstraktionsniveaus

Eine brauchbare Einordnung verschiedener Abstraktionsniveaus im Bereich der visuellen Medien läßt sich durch Anwendung des *Computer Graphics Reference Model* (CGRM) gemäß ISO/IEC 11072 [84] erreichen. Dieses Referenzmodell unterscheidet: Construction Environment, Virtual Environment, Viewing Environment, Logical Environment und Realization Environment – hier mit abnehmendem Abstraktionsniveau aufgezählt. Diese Schichten sind zwischen der Computergraphik-Anwendung und dem Anwender angeordnet. Zwischen diesen Environments findet innerhalb eines Graphiksystems eine bidirektionale Kommunikation statt. Außerdem können jeweils Import- und Export-Schnittstellen zum Austausch oder zur Kopplung zwischen verschiedenen Systemen vorgesehen werden. Je nach Abstraktionsebene kommen dafür Pixelbilder (Logical oder Realization Environment), 2D-Vektorgraphik (Virtual Environment) oder geometrische 3D-Graphik-Metafiles

(Construction Environment) in Frage (siehe auch [132]). Die zuletzt genannte Repräsentation dient als Schnittstelle für das im Rahmen dieser Arbeit entwickelte System.

Dimensionalität

Die räumliche Dimension verschiedener Medien kann in 1D, 2D und 3D eingeteilt werden. Diese wirkt sich letztlich in der Codierung aus. Beispielhaft seien genannt:

- 1D: Text (Plaintext, strukturierter Text, Hypertext)
- 2D: Pixelbild, Vektorgraphik
- 3D: Voxelgraphik (Volumen-Pixel), geometrische 3D-Graphik (Polygone)

Darüber hinaus spielt die zeitliche Dimension – z. B. „4D“ = 3D + T – eine wichtige Rolle. Darauf wird in *Abschn. 2.1.2.7* separat eingegangen. Auf dreidimensionalen, optional zeitabhängigen, geometrischen Szenenbeschreibungen basieren die im Rahmen dieser Arbeit entwickelten Mechanismen.

Codierung

Zur Transformation realer Medien in eine digitale, effiziente Repräsentation sind generell die folgenden vier Aufgaben zu erledigen:

1. Wandlung physikalischer Größen, z. B.
   - Optik – Bilderfassung mittels CCD-Chips (1D, 2D)
   - Akustik – dynamische, Kondensator-Mikrofone
2. Abtastung: Orts- bzw. Zeit-Diskretisierung

   Hier ist das Shannon’sche Theorem zu beachten, nach dem die Abtastrate mindestens der doppelten Bandbreite des Quellsignals entsprechen muß. Anderenfalls treten bei der Rekonstruktion auf der Decoder-Seite „Aliasing“-Artefakte auf. Gegebenenfalls muß eine Bandbegrenzung vorgenommen werden. Prinzipiell existiert das Aliasing-Problem auch bei der Erfassung von Farben, bei der das kontinierliche Wellenlängen-Spektrum meist durch drei Primärvalenzen repräsentiert wird, die an die menschliche Wahrnehmung angenähert werden.
3. Analog-Digital-Wandlung: Wert-Diskretisierung

   Abhängig vom tolerierbaren Fehler erfolgt hier die Quantisierung jedes Abtastwerts in eine Zahl, d. h. binäre Darstellung mit adäquater Bitanzahl je Wert. Es kommen sowohl Ganzzahl- als auch Festkomma- oder Fließkomma-Repräsentationen in Betracht.
4. Quellen-Codierung / Kompression

   Zur effizienten Informationsdarstellung wurden Kompressionsverfahren entwikkelt. Darauf wird in den folgenden Abschnitten eingegangen.

Die hier beschriebene digitale Repräsentation resultiert im Bereich visueller Medien in 2D-Pixel- bzw. 3D-Voxel-Darstellungen. Alternativ ist eine semantisch höherwertige objektorientierte Abbildung möglich, z. B. in Form von 2D-Vektorgra-

phik oder 3D-Szenenbeschreibungen. Das Problem der Akquisition derartiger Daten ist allerdings deutlich komplexer; für die 3D-Modellierung können sowohl weitgehend automatische Verfahren (z. B. 3D-Scanner, 3D-Oberflächenrekonstruktion aus Tomographie-Ergebnissen, Konvertierung aus CAD-Modellen) als auch manuelle Techniken (z. B. 3D-Szenen-Editoren) angewandt werden. Verfahren zur effizienten Codierung von 3D-Modellen unterscheiden sich grundlegend von denen zur Rasterbild-Codierung. In *Abschn. 2.2.2* (S. 26) wird darauf näher eingegangen.

Kompression – Hintergrund

Generell muß bei der Übertragung von Medien ein Optimierungsproblem gelöst werden, in dem für eine Verarbeitungskette – bestehend aus Codierung, Übertragung und Decodierung – unter Berücksichtigung gewisser Nebenbedingungen eine gegebene Kostenfunktion zu minimieren ist.

Variablen dieser *Kostenfunktion* sind z. B.:

- Monetäre Kosten für Bitrate, Datenvolumen, etc.
- Verzögerungs-, Rechen- und Übertragungszeiten für Codierung/Kompression, Transport, Decodierung/Dekompression
- Hardware-Realisierung versus Software-Implementierung

Nebenbedingungen können Begrenzungen der Wertebereiche der Variablen sein:

- Maximal vertretbare monetäre Kosten
- Minimale Darstellungsqualität
- Maximale Bitrate
- Maximale Reaktionszeit

Die verfügbare Kommunikationsdatenrate – insbesondere für Consumer-Anwendungen, verglichen mit dem Quellendatenstrom oder geforderten Reaktionszeiten – stellt oft eine äußerst begrenzte und/oder teuere Ressource dar. Da die aus den Codec-Prozessen resultierenden Verzögerungszeiten – z. B. durch Hardware-Realisierung – im allgemeinen als vernachlässigbar angesetzt werden, wird daher häufig ausschließlich auf einen möglichst hohen Kompressionsfaktor geachtet.

Kompression – Realisierung

Bei der Realisierung sind zunächst verlustbehaftete und verlustlose Verfahren zu unterscheiden. Signifikante Kompressionsfaktoren lassen sich oft nur durch verlustbehaftete Verfahren erzielen, die für die Charakteristik der zu verarbeitenden Medien ausgelegt sind. Die Quellen-Codierung dient hier zunächst zur Irrelevanz-Reduktion, deren Realisierung auch von den Nutzungsformen abhängt. Beispielsweise werden physiologisch begründete Redundanzen in der Intraframe-Codierung von Pixelbildern (JPEG-Kompression gemäß ISO/IEC 10918 [144]) bzw. Interframe-Codierung von Bewegtbildern (MPEG-Kompression) berücksichtigt. Durch Entropie-Codierung – z. B. Huffman-Verfahren [69] – kann die Redundanz der Daten weiter reduziert werden, sofern die Auftretenswahrscheinlichkeit der möglichen Codeworte unterschiedlich ist.

Kompression – (Gegen-)Beispiel

Unter bestimmten Voraussetzungen, wie z. B. relativ hohe verfügbare Bitraten, Dekompression in Software, ist der Einsatz von Kompressionsverfahren nicht mehr vorteilhaft.

In einer Studie zum „Online-Publishing“ in hochratigen Datennetzen wurden gescannte Abbildungen sowohl unkomprimiert (Bilddateiformat: TIFF, Farbmodell: RGB, Farbtiefe: 24 bit/Pixel) als auch JPEG-komprimiert gespeichert. Über das leistungsoptimierte Client-Server-System „DocShow/DocServ“ wurden diese schließlich als „Online-Dokumente“ abgerufen [130][131].

Die Variablen, Kostenfunktion sowie Nebenbedingung seien wie folgt:

- Variablen
  - Transferrate: r_T
  - Kompressionsverfahren (2 Varianten):
    (a) Ohne Kompression ($K = 0$)
    (b) JPEG: Software-Realisierung ($K = 1$)
- Kostenfunktion
  - Reaktionszeit T_{S_K} bei sequentiellem Auflauf von Transport und Decodierung: Summe aus Transportzeit T_{T_K} und Decodierzeit T_{D_K}

$$T_{S_K} = T_{T_K} + T_{D_K} \tag{4}$$

- Nebenbedingung
  - Limitierte verfügbare Transferrate: $r_T < 34$ Mbit/s

Bei der Ermittlung der Grenzrate r_G, bei der ein „Break-Even-Point“ vorliegt, werden Startup- und Display-Zeiten nicht berücksichtigt, da sie von den Variablen unabhängig sind.

Mit dem Datenvolumen N_K, der Transferrate r_T, der Decodier- und Konvertierrate r_{D_K} und der Kennzahl $K = 0$ (ohne Kompression) bzw. $K = 1$ (mit Kompression) ergibt sich aus *Gleichung 4*:

$$T_{S_K} = \frac{N_K}{r_T} + \frac{N_K}{r_{D_K}} \tag{5}$$

Für ein Beispielbild (898x912 Pixel) wurden auf einer leistungsfähigen Graphikworkstation (Silicon Graphics Reality Engine2 mit 2 CPUs MIPS R4400, 200 MHz, Betriebssystem Irix 5.3, X-Windows mit True-Color-Visual 32 bit/Pixel) einige Kenngrößen gemessen. Auf der Grundlage dieser in *Tabelle 1* aufgeführten Kenngrößen ist die Abhängigkeit zwischen Transferrate r_T und Gesamtzeit T_{S_K} für die beiden Kompressionsvarianten in *Abb. 4* graphisch dargestellt.

	Ohne Kompression: $K = 0$	Mit Kompression (JPEG): $K = 1$
N_K	1.650.360 byte	95.466 byte (Kompressionsfaktor: ca. 17,3:1)
T_{D_K}	0,097 s	0,616 s
$r_{D_K} = \frac{N_K}{T_{D_K}}$	136 Mbit/s	1,24 Mbit/s

Tabelle 1. Charakteristische Konstanten für das verwendete Farb-Rasterbild

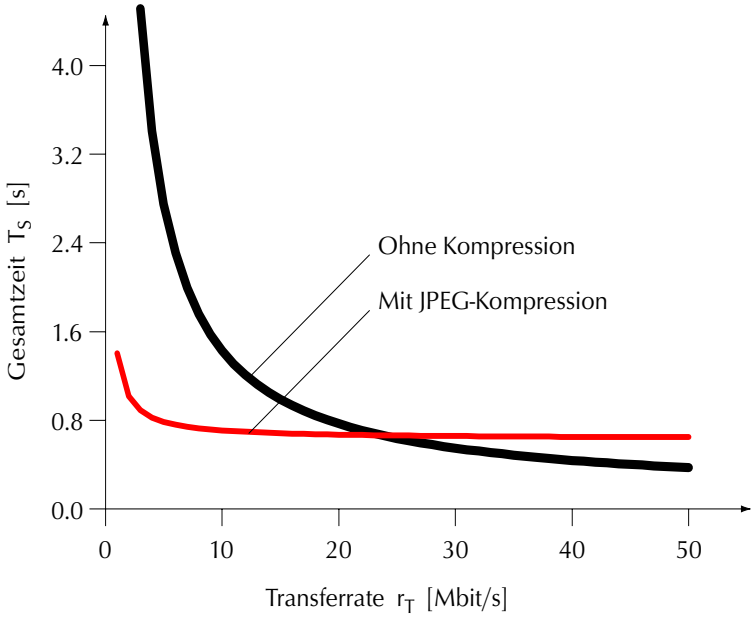


Abb. 4. Gesamtzeit (Transport, Decodierung, Konvertierung) in Abhängigkeit von der Transferrate

Die Grenzrate r_G, also der Schnittpunkt der beiden Kurven in *Abb. 4* (der sogenannte „Break Even Point“), kann durch den Ansatz

$$T_{S_0} = T_{S_1} \tag{6}$$

ermittelt werden.

Durch Gleichungsumformung ergibt sich:

$$r_G = \frac{N_0 - N_1}{\frac{N_1}{r_{D_1}} - \frac{N_0}{r_{D_0}}} \tag{7}$$

bzw. mit dem Kompressionsfaktor $\kappa = \frac{N_0}{N_1}$:

$$r_G = \frac{\kappa - 1}{\frac{1}{r_{D_1}} - \frac{\kappa}{r_{D_0}}} \tag{8}$$

Mit den Werten aus *Tabelle 1* beträgt die Grenzrate im vorliegenden Szenario $r_G = 24$ Mbit/s ; dabei ist die Reaktionszeit $T_S = 0{,}65$ s . Wird die verfügbare Bitrate $r_T = 34$ Mbit/s genutzt, so liegt das Optimum bei Verzicht auf Kompression vor.

Datei- und Datenstrom-Formate

Einige übliche Formate zur Repräsentation visueller Darstellungen sind in *Tabelle 2* aufgeführt. Zur Kompression von Pixelbildern werden verschiedene Verfahren eingesetzt, die zur Redundanz- bzw. Irrelevanz-Reduktion beitragen [38][157][177]. Dabei ist zu unterscheiden zwischen verlustlosen (z. B. LZW) und verlustbehafteten Verfahren (z. B. JPEG), abhängig von der Fähigkeit, die exakte Rekonstruktion des Bildes zu ermöglichen. Die Dateiformate zur Abbildung von vektororientierter 2D-Graphik wurden primär für Druck- bzw. Plottzwecke standardisiert. Zur Speicherung und Übertragung von 3D-Szenen existiert eine Vielzahl proprietärer Formate. Für Internet-Anwendungen wurde der ISO-Standard VRML97 [86] entwickelt.

	diskret (statisch)	kontinuierlich (animiert)
Pixel-Bild (2D)	• TIFF [1]: verschiedene Farbräume, meist unkomprimiert, oft verlustlos komprimiert (LZW), ICC-Farbprofile einbettbar • JPEG, komprimiert, bis ca. 10:1 • GIF: max. 256 Farben, verlustlos komprimiert (LZW) • PNG: Portable Network Graphics • SGI RGB, Wavefront RLA: RGB-Bilder, lauflängencodiert, optional mit Alpha-Kanal (Transparenz) • Abekas YUV: Digitalvideobild gemäß ITU-R 601 [88]	• MPEG-1: Video/Audio, bis ca. 1,5 Mbit/s, bis ca. 320x240 Pixel • MPEG-2: Video/Audio mit höherer Qualität • RealMedia (RealNetworks): ab 28,8 bit/s, SMIL-Support • AVI (Microsoft – Video for Windows): verschiedene Codecs • Quicktime (Apple): div. Codecs • Animated GIF • AVS Frame Sequence [7]: Uncompressed, Runlength-, Color-Cell-Compression • Sequenz von Einzelbildern, z. B. Motion-JPEG („MJPEG“)
Vektor-Graphik (2D)	• PostScript, PDF (Adobe) • CGM [77]	• Sequenz von Graphiken, ggf. auch in einer Datei speicherbar
3D-Szene	• Anwendungsspezifische Formate, z. B. für Animation: 3D Studio 3DS, Wavefront OBJ, etc.; CAD: DXF, IGES, VDAFS etc. • DVR (RRZN/RVS) [133][137] • GLS – OpenGL-Datenstrom [36] • Inventor (SGI) [200] • PHIGS Archive file format [81] • VRML 1.0 [199], VRML97 [86]	• MPEG-4 [35] • VRML97-Elemente • Sequenz von Einzelszenen

Tabelle 2. Datei- bzw. Datenstrom-Formate zur visuellen Repräsentation (Beispiele)

Darüber hinaus sind Metadaten gebräuchlich, um zusätzliche Attribute zu speichern (z. B. MPEG-7) oder um Layout und Zeitabläufe für Kompositionen mehrerer Medienströme zu spezifizieren (z. B. SGML [79], HTML [12], XML [196], HyTime [83], SMIL [197]). Durch eine Separierung der Attributierung, Layout- und Strukturbeschreibungen von den Mediendaten werden Mehrwertdienste ermöglicht, z. B. auf Datenbanken basierende Retrievalmethoden. Derartige Mechanismen sind im WWW essentiell, um den gezielten Zugriff auf die eigentlich gewünschten Informationen zu erleichtern und damit die Nutzungseffizienz des großen Angebots an Nachrichten zu erhöhen.

2.1.2.4 Speicherungsmedien

Zur rechnerinternen (Zwischen-)Speicherung, zur Archivierung, zur Sicherung sowie zum Informationsaustausch (siehe auch *Abschn. 2.1.2.6*) werden Speicherungsmedien verwendet. Aus den verschiedenen zur Verfügung stehenden Speicher-Technologien muß eine jeweils adäquate Auswahl getroffen werden, da z. B. Medien mit kürzerer Zugriffszeit und höherer Datenrate teurer sind und geringerer Kapazitäten aufweisen als solche mit größeren Latenzzeiten und Lese- bzw. Schreibraten [52].

Daher sind Speicher-Hierarchien zweckmäßig, die aus

- Primär-Speichermedien, z. B. Hauptspeicher des Rechners,
- Sekundär-Speichermedien, z. B. Festplatte, und
- Tertiär-Speichermedien, z. B. Magnetbänder,

bestehen.

Um eine größtmögliche Ausnutzung derartiger Speicherungssysteme und eine performante Realisierung von Anwendungen zu erzielen, sind verschiedene – zum Teil anwendungsspezifische – Strategien entwickelt worden. Dazu zählen z. B. Prefetching, Caching, etc., um z. B. in interaktiven Multimedia-Anwendungen Echtzeitfähigkeit zu erzielen [176].

2.1.2.5 Kommunikationsmedien

Zur Übertragung von Medien-Datenströmen kommen prinzipiell die aus der Datenkommunikation bekannten Infrastrukturen, Protokolle und Dienste in Frage. Die Teilaufgaben von Kommunikationssystemen werden gemäß ISO-OSI-(Open Systems Interconnection, Kommunikation offener Systeme)-Referenzmodell in 7 Schichten aufgeteilt [75][186]:

1. In der *Bitübertragungsschicht (Physical Layer)* wird das physikalische Verfahren zur Übertragung von Bitströmen festgelegt. Dabei können sowohl leitergebundene Systeme (Kupferkabel: Koaxial-Kabel, Twisted-Pair-Kabel; Lichtwellenleiter) als auch Funksysteme verwendet werden. Verbreitete Verfahren zur Modulation der Bit-Informationen auf dem jeweiligen physikalischen Medium sind Ethernet (für Rechner verfügbare Interfaces: 10 Mbit/s, 100 Mbit/s, 1000 Mbit/s) und ATM (Asynchronous Transfer Mode; für Rechner verfügbare Interfaces: 155 Mbit/s, 622 Mbit/s).

2. Die *Sicherungsschicht (Data Link Layer)* faßt eine bestimmte Anzahl Bits zu einem Datenübertragungsrahmen (Frame) zusammen, sorgt für den Datenträgerzugriff (MAC – Media Access Control) und gewährleistet die Konsistenz der übertragenen Nachrichtenblöcke (LLC – Logical Link Control), z. B. durch Fehlererkennung, Fehlerkorrektur und Flußsteuerung.

3. Die *Vermittlungsschicht (Network Layer)* beinhaltet Funktionen zur Übermittlung zwischen Knoten längs eines Weges.

4. Die *Transportschicht (Transport Layer)* realisiert die Schnittstelle zwischen den jeweiligen Prozessen in den beteiligten Endgeräten. Zu große Nachrichtenpakete werden hier den niedrigeren Schichten entsprechend fragmentiert und empfangsseitig wieder zusammengesetzt.

5. In der *Sitzungsschicht (Session Layer)* wird für einen verbindungsorientierten Dienst garantiert, daß die Verbindung erhalten bleibt.

6. Die *Darstellungsschicht (Presentation Layer)* dient zur Abstraktion verschiedener gerätespezifischer Datenformate, z. B. je nach CPU individueller Zahlendarstellungen.

7. Die *Anwendungsschicht (Application Layer)* behandelt die anwendernahen Dienste.

Für jede Schicht existieren ISO-Standards, die entsprechende Protokolle spezifizieren. Diese haben sich allerdings nicht durchgesetzt. Statt dessen werden weltweit im wesentlichen die IEEE- und IETF- bzw. Internet-Standards verwendet, z. B.

- Datenträgerzugriff: z. B. IEEE 802.3 (CSMA/CD – Carrier Sense Multiple Access with Collision Detection, Mehrfachzugriff mit Kollisionserkennung),
- Vermittlungsprotokoll: IP (Internet Protocol),
- Transportprotokoll: TCP (Transmission Control Protocol, RFC 1180 [174]),
- Anwendungsprotokolle: z. B. FTP (File Transfer Protocol, RFC 959 [148]).

Die anwendernahen Dienste können entsprechend ihrer Charakteristika, z. B. bezüglich Diensttyp, Kardinalität, Interaktivität, Verbindungsorientierung (Connection-Oriented Services, Connectionless Services) klassifiziert werden [60].

Einige Beispiele sind in *Tabelle 3* angegeben (siehe auch ITU-T I.211 [89]).

Bezeichnung	Kardinalität	Beispiele
Transport- und Auftragsdienste	1:1	Email (SMTP: Simple Mail Transfer Protocol), Entferntes Drucken (lpr), File Transfer Protocol (FTP),
Abrufdienste	1:1	World Wide Web (WWW): Hypertext Transport Protocol (HTTP)
Verteildienste	1:N	Digitales Fernsehen
Dialogdienste	M:N	Video-Conferencing (z. B. MBONE)

Tabelle 3. Klassifikation von Kommunikationsdiensten

In *Abb. 5* wird am Beispiel des WWW-Protokolls der Ablauf einer Client-Server-Kommunikation illustriert, welche auf HTTP 1.0 basiert.

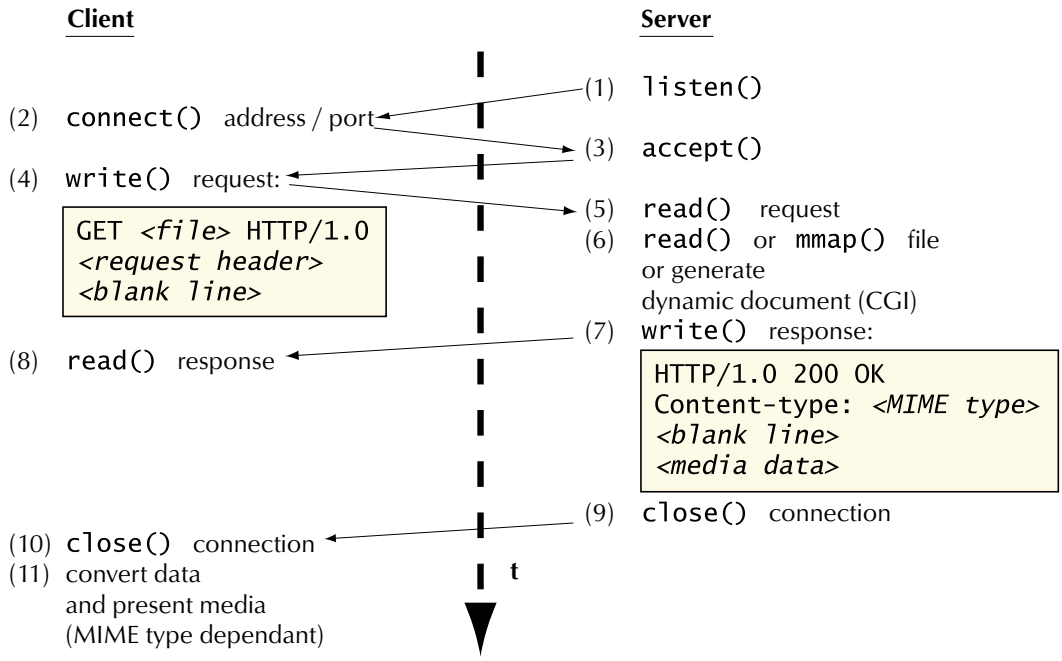


Abb. 5. Ablauf einer Client-Server-Kommunikation am Beispiel eines Abrufs über HTTP 1.0

Zum Abruf eines WWW-Dokuments wird zunächst seine Adresse benötigt, welche in Form einer URL (Uniform Resource Locator, RFC 1737 [13]) die Angaben

- Bezeichnung des verwendeten Protokolls, z. B. `http` (hier), `ftp`,
- Internet-Adresse des Netzinterfaces am WWW-Server, z. B. `www.rrzn.uni-hannover.de`,
- Port-Nummer des Server-Prozesses, z. B. `80` (default),
- Dateiname, z. B. `mml/index.html`

spezifiziert:

<protocol>`://`<address>[`:`<port>]/<filename>

Die Internet-Adresse sowie die Portnummer dienen dem WWW-Client zur Parametrisierung des Verbindungsaufbaus zu einem WWW-Server über TCP/IP. Der Dateiname wird innerhalb der ersten Anforderungszeile übermittelt. Der Server liefert daraufhin das von einer Datei eingelesene oder über die CGI-Schnittstelle (Common Gateway Interface) dynamisch generierte Multimedia-Dokument nach einem speziellen Vorspann, der unter anderem den Medientyp charakterisiert (MIME – Multipurpose Internet Mail Extensions, RFC 2045 [47]), an den Client aus. Dieser wiederum liest die Mediendaten ein und präsentiert sie dem MIME-Typ entsprechend, z. B. direkt innerhalb des WWW-Browsers oder in einem separaten Viewer. Schließlich wird die Verbindung auf beiden Seiten wieder geschlossen.

Breitbandkommunikation[1]

Leistungsfähige Kommunikationssysteme, wie sie z. B. zur Realisierung netzverteilter, hochqualitativer, multimedialer bzw. immersiver Virtual-Reality-Anwendungsszenarien notwendig sind, erfordern nicht nur eine breitbandige Netz-Infrastruktur (Schicht 1). Es ist dabei auch auf effiziente Implementierungen sämtlicher darauf aufsetzender Schichten, wie z. B. Transportprotokolle, Dienste und Anwendungen zu achten. Im Zusammenhang mit extrem hochratiger Datenübertragung fallen insbesondere vermeidbare Zeitaufwände ins Gewicht, die in den Endgeräten z. B. durch Kopieren von Speicherblöcken zu einem limitierenden Faktor werden können oder durch unangepaßtes Dienstedesign die Qualität der Anwendung signifikant reduzieren können [14].

Beispielsweise kann bereits das „Round Trip Delay“ t_D (die Laufzeit einer Nachricht über ein Übertragungssystem, bestehend aus den Signallaufzeiten für Hin- und Rückweg sowie gegebenenfalls serverseitiger Latenzzeit) die effektiv erzielbare mittlere Transferrate deutlich beeinflussen. Werden in einem Protokoll z. B. von einem Server einzelne Datenblöcke der Größe n jeweils explizit von einem Client durch Übermittlung eines vernachlässigbar kleinen Datenblocks angefordert, so beträgt die Gesamtzeit t_T von der Anforderung bis zur Beendigung der Übertragung mit einer Burstrate r_B:

$$t_T = t_D + \frac{n}{r_B} \quad . \tag{9}$$

Die effektive mittlere Übertragungsate r_T eines solchen Transportsystems – welches vereinfacht dem WWW-Protokoll HTTP (siehe auch *Abb. 5*) entspricht – beträgt dann:

---

[1] Obwohl dieser Begriff im Kontext digitaler Kommunikationsnetze üblicherweise verwendet wird (z. B. B-WiN: Breitband-Wissenschaftsnetz), ist anzumerken, daß es hier nicht um die Begrenzung von Frequenzbändern geht, sondern die Leistungsmerkmale in der Datenkommunikation gemeint sind, die z. B. durch Bitraten und Latenzzeiten beeinflußt werden. An dieser Stelle soll außerdem darauf hingewiesen werden, daß auch der häufig gebrauchte Ausdruck „Hochgeschwindigkeitsnetze“ mißverständlich ist und möglichst vermieden werden sollte, da meist nicht die Geschwindigkeit des einzelnen Bits gemeint ist (diese wird durch die Lichtgeschwindigkeit limitiert).

$$r_T = \frac{n}{t_D + \frac{n}{r_B}} \tag{10}$$

Eine gute Ausnutzung des Mediums kann dabei nur für relativ große Nachrichtenblöcke erzielt werden.

Soll z. B. $r_T \geq 0{,}9 \cdot r_B$ erreichen, so beträgt – wie sich durch Umformung aus *Gleichung 10* herleiten läßt – die Mindestnachrichtengröße:

$$n \geq 9 \cdot r_B \cdot t_D \,. \tag{11}$$

Mit $t_D = 10$ ms und $r_B = 100$ Mbit/s gilt dabei: $n \geq 9$ Mbit .

Benötigt eine Anwendung nach einem bestimmten Schema von einem Server eine hohe Anzahl von Nachrichtenblöcken, die klein sind im Vergleich zum Produkt aus Verzögerungszeit und Bitrate (siehe *Gleichung 11*), so ist ein serverseitig kontinuierlich ausgespielter Datenstrom einer Folge von Einzelanforderungen vorzuziehen. Dies ist z. B. in Video-/Audio-Streamingsystemen im Kontext WWW-basierter Abrufsysteme der Fall sowie auch beim „3D-Streamingsystem“, welches im Rahmen dieser Arbeit entwickelt wurde.

2.1.2.6 Informationsaustauschmedium

Im Gegensatz zu den Online-Transportmedien, die im vorigen Abschnitt behandelt wurden, sind hier die Offline-Medien betroffen. Dazu werden elektrische, magnetische oder optische Verfahren angewandt, die eine Persistenz der gespeicherten Daten während des physikalischen Transports des Mediums gewährleisten.

Je nach verfügbarem Platz, erforderlicher Kapazität, akzeptierter Speicherungs- bzw. Zugriffszeiten und gegebenenfalls weiterer Kriterien (z. B. Kosten, Zuverlässigkeit, Temperaturbereich, Größe, Gewicht, Wiederbeschreibbarkeit, etc.) werden diese dem Zweck entsprechend ausgewählt. Beispiele für transportable Speichermedien sind:

- elektrisch:
  EPROMs, EEPROMs, batteriegepufferte RAMs (z. B. PCMCIA-Karten),
- magnetisch:
  Harddisk-Moduln (z. B. PCMCIA-Karten), Magnetbänder (z. B. DAT),
- optisch:
  CD-ROM, CD-RW, DVD.

2.1.2.7 Zeitbezug

Bezüglich der zeitlichen Aspekte in Multimedia-Systemen ist zu unterscheiden zwischen diskreten, d. h. zeitunabhängigen (wie Text, Einzelbild) und kontinuierlichen, d. h. zeitabhängigen Medien (wie Bewegtbild, Audio). Für diskrete Medien spielen im wesentlichen die Latenz-, Übertragungs- und Darstellungszeiten eine Rolle.

Kontinuierliche Medien erfordern dagegen zusätzlich Maßnahmen zur Synchronisierung, und zwar sowohl intramedial – d. h. um z. B. eine konstante Bildrate für Bewegtbilder einzuhalten – als auch intermedial – d. h. um z. B. „Lippensynchronität“ bei einer Video/Audio-Präsentation sicherzustellen. Darüber hinaus entsteht für geographisch verteilte Client-Systeme gegebenenfalls die Notwendigkeit, eine Inter-Client-Synchronisierung durchzuführen. Im Fall der Nutzung separater Datenströme für synchron zu präsentierende Medien müssen Intra- bzw. Inter-Stream-Synchronisierungsverfahren bereitgestellt werden.

Sind kontinuierliche Medien über ein paketorientiertes Transportmedium zu übertragen, ist zunächst eine Paketierung durchzuführen, die eine von Paketgröße und Quelldatenrate abhängige Paketierungsverzögerung einführt. Weist das Transportverfahren eine Schwankung der Übertragungsverzögerung auf (Jitter), so ist eine Pufferung zur Glättung vorzusehen; diese führt zu einer zusätzlichen Ausspielverzögerung (siehe auch *Abb. 6*).

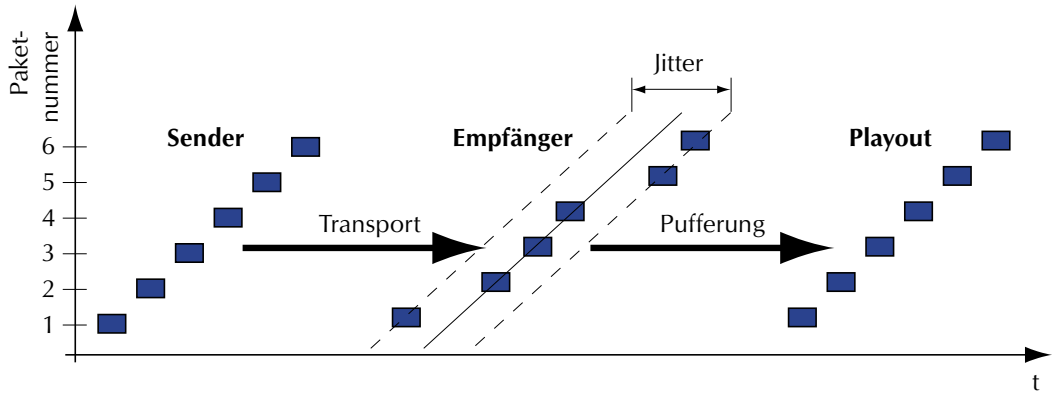


Abb. 6. Übertragung kontinuierlicher Medien – Paketierung, Transport, Pufferung, Ausspielung

Je nach Anwendungsszenario bestehen verschiedene Anforderungen, z. B.:

- Conferencing:
  Ende-zu-Ende-Verzögerung maximal ca. 150 ms,
- Abrufdienst:
  Reaktionszeit maximal ca. 2 s (siehe auch [166]),
- Interaktives Visualisierungssystem:
  Reaktionszeit maximal ca. 100 ms (siehe auch [100]),
  Update-Rate mindestens 10/s für eine flüssige Bewegung
  (siehe auch [123] bzw. *Abschn. 2.1.2.2*).

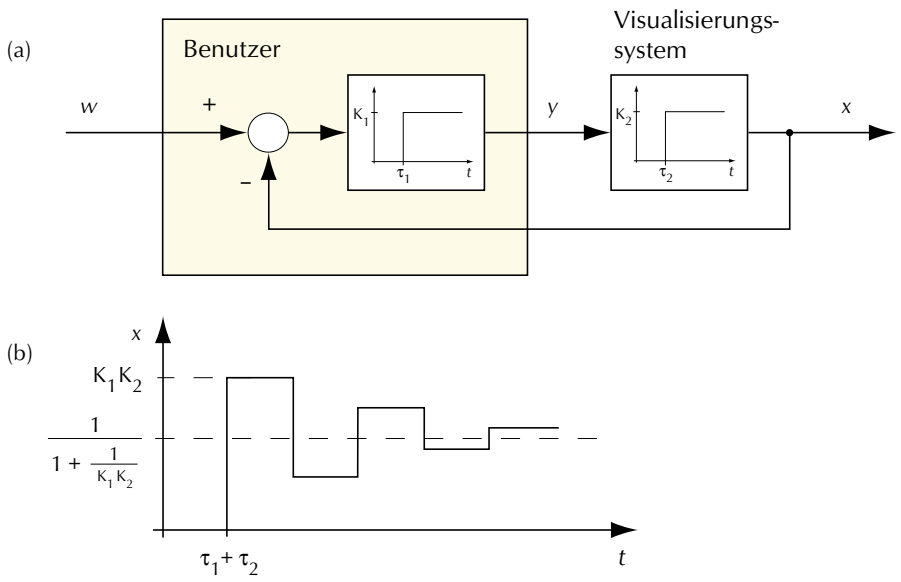


Abb. 7. Modell zum Einschwingverhalten eines interaktiven Visualisierungssystems [100].
(a) Blockschaltbild
(b) Sprungantwort bei Änderung des Sollwertes *w*
Benutzer und Visualisierungssystem sind als lineare Totzeitglieder modelliert, wobei:
τ_1 , τ_2 : Totzeiten, K_1 , K_2 : Verstärkungsfaktoren,
x: Systemantwort, *y*: Benutzerreaktion, *w*: Sollwert.

Wie in *Abb. 7* dargestellt, befindet sich der Benutzer mit dem gegebenenfalls netzverteilten Visualisierungssystem in einem Regelkreis, welcher je nach Charakteristik der Teilkomponenten zu Instabilitäten (Schwingungen) neigen kann. Dies kann z. B. das Orientierungsvermögen während der Navigation im virtuellen Raum beeinträchtigen. Außerdem können durch zu große Latenzzeiten ungünstige physiologische Effekte auftreten, wie z. B. Schwindelgefühl des Nutzers.

2.2 Informationssysteme – MM/VR-Anwendungen im Internet

Als Beispiel für verteilte Szenarien im Internet, die multimediale bzw. 3D/Virtual-Reality-Anwendungen unterstützen, wird im folgenden näher auf Informationssysteme eingegangen. Für die im Rahmen dieser Arbeit durchgeführten Entwicklungen hat der hier zugrunde liegende Kommunikationsdienst-Typ (in *Tabelle 3, S. 19* wurde dieser als Abrufdienst charakterisiert) eine besondere Bedeutung.

2.2.1 Multimedia im World Wide Web

Multimedia-Informationsdienste, die in Kommunikationsnetzen verteilt angeboten werden, beruhen heute im wesentlichen auf Protokollen, Adressierungsschemata und Diensten, die im Internet etabliert sind. Dazu gehören z. B. die von der IETF (Internet Engineering Task Force) in RFCs (Request for Comments) beschriebenen Internet-Standards TCP/IP [174], URL [13], HTTP [45], HTML [12], MIME [47]. Auf dieser Basis – im allgemeinen auch als „World Wide Web“ (WWW) bezeichnet – können Anwendungen entwickelt werden, welche Objekte beinhalten, die durch verschiedene Medientypen – wie z. B. Text, Hypertext, Bild, Graphik, Video, Audio, 3D-Szenen – und Codierungsverfahren repräsentiert werden und gegenseitig durch Hyperlinks referenziert werden können. Die heute am meisten verbreitete und benutzte WWW-Client-Software Communicator der Firma Netscape realisiert einen generischen Client. Dieser gestattet unter anderem die Einbettung von Multimedia-Viewern in HTML-Seiten mittels externer Viewer, durch Inline-Plugins sowie die Integration von Java-Applets. Dabei können Plugin- und Java- sowie JavaScript-Technologie auch kombiniert eingesetzt werden. Mit Hilfe dieser als *LiveConnect* bezeichneten, von Netscape spezifizierten Schnittstelle [124][125] ist es möglich, auf der HTML-Seite, die das Inline-Plugin beinhaltet, Bedienelemente zu plazieren, die vom Plugin bereitgestellte Funktionen auslösen können (siehe auch *Abb. 8*).

Bei den in eine WWW-Browser-Umgebung integrierbaren, MIME-Typ-spezifischen Viewern werden zwei Klassen unterschieden:

Externe Viewer

Dies sind separate Applikationen, welche – z. B. durch Anklicken eines Links auf ein Dokument mit entsprechendem MIME-Typ – in einem eigenen Fenster gestartet werden und dort eigenständig ablaufen.

Inline-Plugins

Diese werden durch *Dynamic Link Libraries* (DLL unter Microsoft Windows) bzw. *Dynamic Shared Objects* (DSO unter UNIX) realisiert und werden zur Laufzeit in den Browser-Prozeß eingebunden. Multimedia-Elemente können mittels eines `EMBED`-Tags, ähnlich wie mit dem `IMG`-Tag die Inline-GIFs oder -JPEGs, in das Layout von HTML-Seiten integriert werden. Auch der gewöhnliche Abruf per URL wird unterstützt. Die Darstellung erfolgt dann ebenfalls innerhalb des Browsers, wobei dann das gesamte Fenster ausgefüllt wird.

Auf weitere verfügbare Component-Technologien – wie z. B. Microsoft Active-X-Control, COM, DCOM [115], CORBA (siehe auch OMG Home Page: `www.omg.org`) – wird hier nicht weiter eingegangen.

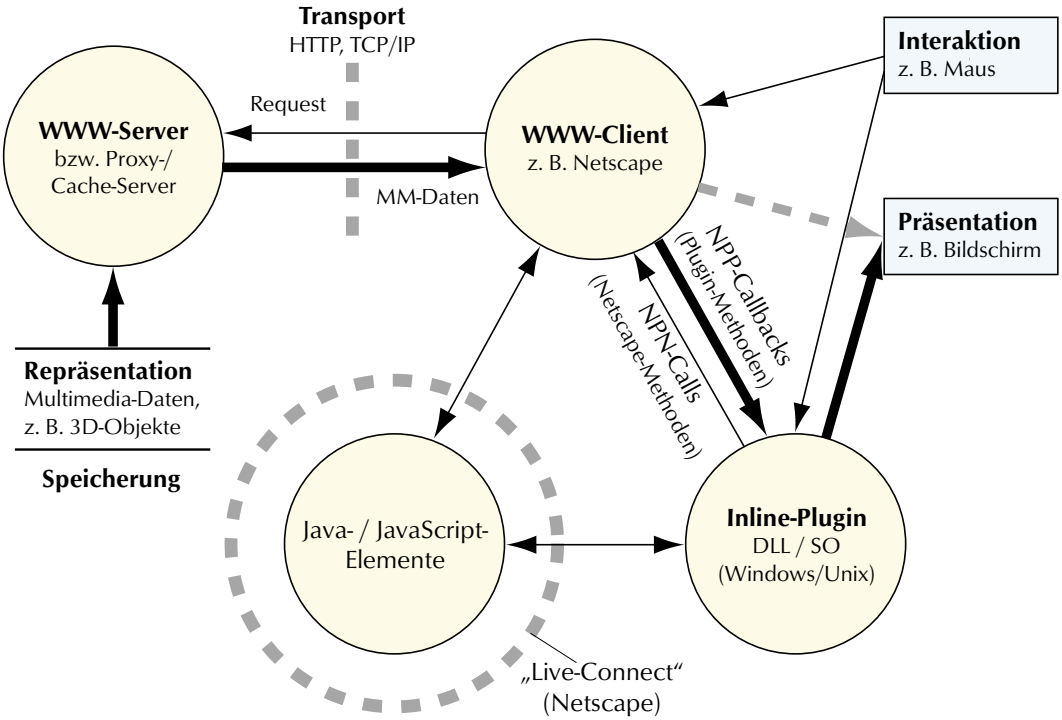


Abb. 8. Multimedia-Abrufdienst (World Wide Web) – Datenflußmodell

Zur Komposition von Hypertext mit Multimedia-Elementen – z. B. aus Gestaltungsgründen, für nutzerfreundlichere Oberflächen – ist insbesondere die zweite Klasse relevant.

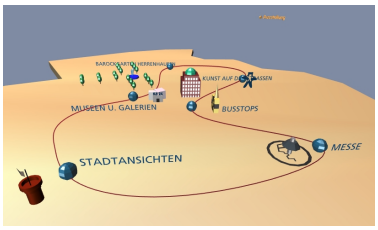
Die Einhaltung zeitlicher Vorgaben für Realtime-Multimedia (Video, Audio) wird häufig nach der burstartigen Übertragung einer vollständigen Datei am Client lokal bewirkt. Neuere Verfahren mit höherer Interaktivität werden von speziellen Viewern angewandt, welche nicht nur die Medien-Präsentation, sondern auch eine Kommunikationsverbindung mit speziellen Streaming-Servern realisieren. Dabei kommen Protokolle zum Einsatz, wie z. B. das *Real Time Streaming Protocol* (RTSP) [165] und das *Transport Protocol for Real-Time Applications* (RTP) [164], welche die für eine Intra-Stream-Synchronisierung erforderlichen Kontrollflüsse und Transportmechanismen bereitstellen. Entsprechend der Dienstgüte-Parameter der Kommunikationsverbindung – Bitrate, Latenz, Jitter, Fehlerrate – werden üblicherweise adäquate, qualitativ abgestufte Medienströme angeboten sowie Puffergrößen eingestellt. Eine Pufferung bewirkt eine Glättung von Schwankungen der Bitrate und Latenz des Transportmediums, um letztlich eine kontinuierliche On-the-fly-Präsentation zu ermöglichen. Die dadurch eingeführte Verzögerungszeit beträgt bei derzeit üblichen Verfahren, wie z. B. RealPlayer der Firma RealNetworks, einige Sekunden.

Zur Unterstützung von Kompositionen mehrerer kontinuierlicher Medien mit Inter-Stream-Synchronisation wurde der SMIL-Standard (Synchronized Multimedia Integration Language) entwickelt [197]. Zeitliche und örtliche Beziehungen mehrerer Text-, Bild-, Video- und Audio-Medienobjekte sowie Hyperlinks können hiermit in einer SGML-basierten Textdatei spezifiziert werden.

Um den gezielten Zugriff auf die eigentlich gewünschten Informationen zu erleichtern und damit die Nutzungseffizienz des im wesentlichen unstrukturierten Angebots an Nachrichten im WWW zu erhöhen, werden geeignete Mehrwertdienste entwickelt und angeboten. Diese basieren auf Metadaten, die explizit in speziellen Dateien (z. B. MPEG-7) oder Datenbanken gespeichert, in Dokumenten eingebettet bereitgestellt (z. B. `META`-Tags in HTML-Dokumenten) oder mittels Text- oder Bild-Analyseverfahren in Suchmaschinen gewonnen werden.

2.2.2 3D / Virtual Reality im World Wide Web

Die Integration von 3D/VR-Techniken in Informationssystemen kann auf folgenden Gebieten nützlich sein (siehe auch *Abb. 9*):



Informationsräume bzw. 3D-Nutzeroberflächen

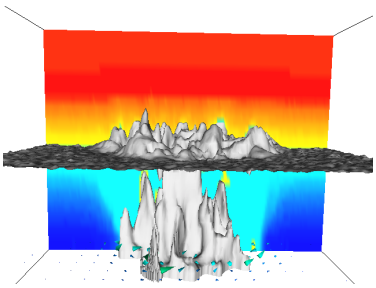
- Navigation durch räumliche Informationsanordnung
- Ergonomische Bedienoberflächen
- „Information Visualization“

Beispiel: „Hannover-Spaziergang“ [160], RRZN/RVS, 1996.

Darstellung geometrischer Objekte

- Modellierung, Rekonstruktion, Simulation
- z. B. Architektur, Landschaftsplanung, Fabrikplanung
- Design: „Virtual Prototyping“

Beispiel: „Golf IV“, Volkswagen AG, 1999.



Wissenschaftliche Datenvisualisierung

- z. B. im Kontext „High-Performance Computing“
  - Experiment / Theorie / Simulation
  - „Virtual Environment“ für Forschung und Lehre
- Exploration, Präsentation und Diskussion von Ergebnissen

Beispiel: „Oceanic Convection“ [153], Institut für Meteorologie und Klimatologie, Universität Hannover; Simulationsrechnung und Visualisierung am RRZN/RVS, 1997.

Abb. 9. Anwendungsszenarien für 3D-Informationsdienste im World Wide Web

1. zur Navigation durch real existierende oder abstrakte Informationsräume mittels dreidimensionaler, intuitiver Nutzer-Oberflächen.
2. zur Online-Präsentation von 3D-Szenen, in denen sich der Nutzer frei bewegen kann. Dazu zählen z. B.
   - Modelle realer Szenarien („Virtual Design“ [5][6]) oder
   - dreidimensionale Darstellungen von Ergebnissen aus Simulationsrechnungen oder Messungen („Wissenschaftliche Visualisierung“).

Die verwendeten, optional zeitabhängigen, geometrischen 3D-Repräsentationen bilden nicht nur die Basis für eigenständige, interaktive Visualisierungsanwendungen, sondern stellen auch einen Ausgangspunkt für abgeleitete Medien auf niedrigeren Abstraktionsniveaus dar. Dies können z. B. 2D-Pixelbilder oder Videos sein, die durch Rendering und zeitliches Sampling abgeleitet werden (*Abb. 10*).

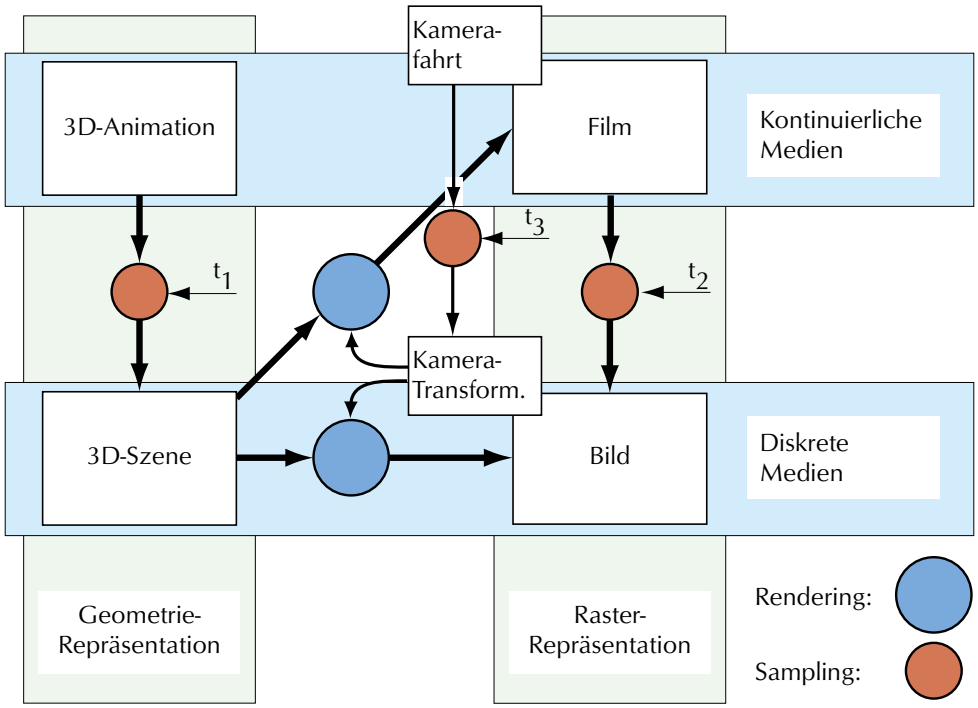


Abb. 10. Ableitungsketten visueller Medientypen:
Geometrie- / Raster-Repräsentation, kontinuierliche / diskrete Medien.
t_1, t_2, t_3: Abtastzeitpunkte (Sampling)

Repräsentation von 3D-Szenen im World Wide Web – VRML

Eine interaktive Betrachtung virtueller 3D-Objekte war urprünglich nur speziellen, als Einzelarbeitsplätzen konzipierten Virtual-Reality-Systemen vorbehalten, in denen die gesamten zur Laufzeit erforderlichen Daten und Methoden auf einem lokalen Dateisystem bzw. im Hauptspeicher des Rechners vorgehalten werden. In einer Startup-Phase, die relativ lange dauern kann, werden die Modelldaten – häufig von proprietären Dateiformaten – eingelesen. Die effiziente Anwendung dreidimensio-

naler Präsentations- und Interaktionstechniken basiert auf vorverarbeiteten Repräsentationen, welche im Hauptspeicher des Rechners liegen.

Als sich mit dem World Wide Web eine Plattform für Hypermedia-Dokumente bildete, welche allgemein verfügbar und in Wissenschaft und Industrie im Internet und in Intranets produktiv anwendbar wurde, bestand zunehmend der Bedarf, die Integration und Standardisierung von 3D-Szenenbeschreibungen voranzutreiben. Im Jahr 1994 wurde nach der Präsentation einer 3D-Nutzerschnittstelle für das Internet [145] eine öffentliche Diskussion, basierend auf einer Email-Liste, initiiert.

Eine Teilmenge des klartext-codierten Austauschformats der 3D-Graphikbibliothek Open Inventor [200] der Firma Silicon Graphics wurde schließlich als Basis für eine erste Spezifikation der „Virtual Reality Modeling Language“ (VRML 1.0) ausgewählt und um einige WWW-spezifische Sprach-Elemente erweitert, die zur Integration von Hyperlinks sowie zur Erstellung verteilter, aber statischer virtueller Welten dienen: `WWWAnchor` und `WWWInline` [10].

VRML ist in der Version 1.0 eine Sprache zur klartext-codierten Beschreibung statischer virtueller 3D-Szenen in geometrieorientierter Form. Aus den verfügbaren Grundelementen – wie einfache Graphikprimitive (Linien, Polygone, Quader, Kegel, Kugel, Zylinder), Attribute, Abbildungstransformationen, virtuelle Kameras und Lichtquellen – lassen sich damit hierarchische Szenengraphen modellieren. Auf WWW-Servern werden häufig mit *gzip* komprimierte VRML-Dateien bereitgestellt. Als Dateiendung für VRML-Dateien wird üblicherweise `.wrl` (für *world*) verwendet. Der MIME-Typ ist definiert als `x-world/x-vrml`.

In einer überarbeiteten Version 2.0 wurden neben einigen neuen Graphikprimitiven nun auch dynamische Elemente eingeführt, die eine flexible Steuerung der geometrischen Primitive, Attribute und Abbildungstransformationen erlauben, z. B. durch zeitliche Ereignisse oder gesteuert durch integrierbare Programme. Diese wurde schließlich als ISO-Standard „VRML97“ (ISO/IEC 14772) verabschiedet [86].

VRML 1.0 wird jedoch im WWW weiterhin häufig verwendet, da dieses Format zur Darstellung einzelner statischer Szenen vollkommen ausreicht. Außerdem erzeugen Werkzeuge zur Szenen-Generierung (z. B. VR-Autorensysteme, Visualisierungssoftware) zum Teil nur VRML-1.0-Texte. Daneben hat die Speicherung in diesem Format noch Vorteile, da von VRML 1.0 zu den aktuellen Dateiformaten (z. B. VRML97) konvertiert werden kann, jedoch für den umgekehrten Weg keine Werkzeuge zur Verfügung stehen und bestimmte nützliche Attribute – z. B. `focalDistance` in den VRML-Elementen zur Kameraspezifikation – zum Teil nicht mehr unterstützt werden. Der im Rahmen dieser Arbeit entwickelte Präprozessor, der VRML-Daten einliest, wurde daher zunächst auf der Basis frei erhältlicher Komponenten für VRML 1.0 entwickelt und steht als eigenständiger Modul zur Verfügung; später wurde ein Lesemodul für VRML97 hinzugefügt (siehe auch *Abschn. 4.2.1*).

Effiziente Codierung

Weitere Entwicklungen betreffen unter anderem die komprimierte, binäre Codierung [87]. Diese sollen in den VRML-Nachfolge-Standard „X3D“ (früher genannt: VRML-NG) integriert werden [104]. Ein Gerüst für Multimedia-Dokumente mit eigenen 3D-Repräsentationen wird auch im MPEG-4-Standard spezifiziert, wobei interaktive, audio-visuelle 3D-Szenen binär codiert werden (BIFS – *BI*nary *F*ormat for *S*cenes) [35][167].

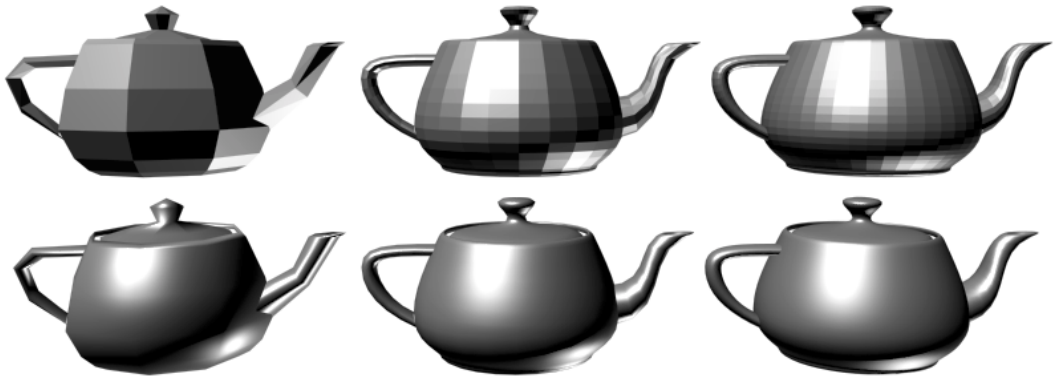


Abb. 11. Verschiedene Detaillierungsgrade eines „Teapots“. Polygone jeweils flach gerendert (oben) und durch interpolierendes Renderingverfahren geglättet (unten) dargestellt (3D Studio MAX)

In letzter Zeit wurden etliche Methoden entwickelt, um zur Verkürzung von Transport- und Rendering-Zeiten die Komplexität bzw. das Datenvolumen der Repräsentation polygonaler 3D-Oberflächenmodelle zu reduzieren. Die verschiedenen bekannten Ansätze können eingeteilt werden in:

1. Polygon-Reduktion und Glättung von Oberflächennetzen [97][98][161][162], auch zur Modellierung mehrerer Auflösungsstufen [27][96][99] (*Abb. 11*).
2. Topologie-orientierte geometrische Kompression [29][33][49][53][188][190], auch in Verbindung mit progressiver Codierung [41][51][65][67][106][108] [187].
3. Verdichtete Packung von Graphik-Primitiven,
   z. B. „Triangle Stripification“ [2][42][43][66][203] (*Abb. 12*).

Die Techniken der ersten beiden Klassen führen prinzipiell zu Abweichungen vom Original. Während in Consumer-Anwendungen verlustbehaftete Kompressionsverfahren toleriert werden können, müssen jedoch Veränderungen in wissenschaftlichen Visualisierungsanwendungen grundsätzlich ausgeschlossen werden. Akkurate Repräsentation und eine qualitativ möglichst hochwertige Präsentation sind essentiell für die Datenexploration, um Fehlinterpretationen der Ergebnisse von oft teuren, auf Supercomputern durchgeführten Simulationsrechnungen zu vermeiden. Die im Rahmen dieser Arbeit erarbeiteten Verfahren sollen das Szenario „Wissenschaftliche Datenvisualisierung“ besonders unterstützen (siehe auch *Abschn. 2.3*). Daher kamen für die Entwicklungen nur verlustfreie Verarbeitungsschritte in Betracht.

Darüber hinaus kann die Rekonstruktion aus einem komprimierten Code signifikante zusätzliche Rechenzeiten erfordern. Diese können in leistungsfähigen Netz- und Rendering-Systemumgebungen für den Durchsatz einen limitierenden Faktor darstellen. Beispielsweise sind für die Dekompressionsrate Werte von ca. 35.000 Dreiecken/s [188] bzw. ca. 12 Mbit/s [53] (rekonstruierte Polygondaten) bekannt. Diese Werte liegen um 1–2 Größenordnungen unter den Rendering- bzw. Übertragungsraten der derzeit verfügbaren Graphik- bzw. Datenkommunikationssysteme.

Das Prinzip der Codierung von Differenzen (z. B. Abweichungen von Schätzwerten bzw. Vorgängerwerten), wie es z. B. für Bilder und Bewegtbilder erfolgreich angewandt wird, ist grundsätzlich auch für eine Geometriekompression nutzbar. Dabei können sowohl innerhalb von statischen Szenen, also auch in kontinuierlichen 3D-Repräsentationen Kohärenzen ausgenutzt werden – vergleichbar mit der Intra- bzw. Inter-Frame-Codierung. Bei Verwendung von Codewörtern mit variabler Länge können dann beispielsweise die vermutlich häufiger vorkommenden kleinen Differenzwerte mit kürzeren Codes dargestellt werden (Entropie-Codierung, z. B. Huffman-Code). Dies ist dann vorteilhaft, wenn die Werte nicht gleichverteilt vorkommen. Die dafür erforderliche Wahrscheinlichkeitsverteilung läßt sich allerdings nur dann erzielen, wenn die Werte quantisiert werden. Die üblicherweise als float-Werte (z. B. 32 bit) vorliegenden 3D-Ortskoordinaten müßten dafür also zunächst in eine Festkomma-Darstellung (z. B. 8–16 bit) überführt werden. Aus einem derartigen Vorgehen resultieren folgende Probleme:

1. Je nach Quantisierungsverfahren (Bereichsdefinition, Bittiefe) wird die Genauigkeit der Repräsentation u. U. signifikant reduziert. Ist eine akkurate Darstellung jedoch zwingend erforderlich, z. B. um Interpretationsfehler bei der Exploration derartig repräsentierter Rechen- oder Meßergebnisse auszuschließen, so kann dieses Verfahren daher grundsätzlich nicht verwendet werden.
2. Im Fall der Ausspielung zeitabhängiger 3D-Szenen(-Sequenzen) wird bei Inter-Frame-Codierung eine direkte Positionierung auf einen bestimmten Zeitschritt, wie dies z. B. bei der Anwendung von Streamingverfahren gewünscht wird, deutlich erschwert.
3. Die Decodierung eines solchen Bitdatenstroms erfordert das Einlesen und Zusammensetzen von Einzelbits und damit einen erheblichen Rechenaufwand auf üblichen Rechnerarchitekturen. Diese relativ einfachen Instruktionen können zwar durch Hardware-Realisierung stark beschleunigt werden, jedoch ist im Fall der 3D-Codierung vorläufig nicht mit derartigen Entwicklungen zu rechnen.

Triangle Stripification ist dagegen eine hilfreiche Technik, die sowohl zur Reduktion des Datenvolumens, als auch zur Optimierung des Renderingprozesses dienen kann. „Triangle strips“ sind zusammenhängende Dreiecksnetze, in denen ein weiteres Dreieck aus jeweils zwei Eckpunkten eines vorangehenden Dreiecks und einem neu spezifizierten Eckpunkt gebildet wird. Daher kann für große verbundene Dreiecksnetze eine verlustfreie Kompression sowie eine Erhöhung der Renderingrate um einen Faktor von bis zu ca. 3 erzielt werden (siehe auch *Abb. 12, S. 31* und *Tabelle 12, S. 89*).

Die derzeit gültigen VRML-Dateiformate sehen keine speziellen Triangle-Strip-Elemente vor, obwohl diese in den generierenden Anwendungen häufig explizit erzeugt werden. Das Kompressionspotential ist daher für die Speicherung und Übertragung nicht nutzbar. Soll der Leistungsvorteil beim Rendering ausgenutzt werden, muß daher eine automatische Erkennung dieser Topologie aus den vorliegenden unabhängigen Polygonen (VRML: `IndexedFaceSet`) auf Viewer-Seite durchgeführt werden.

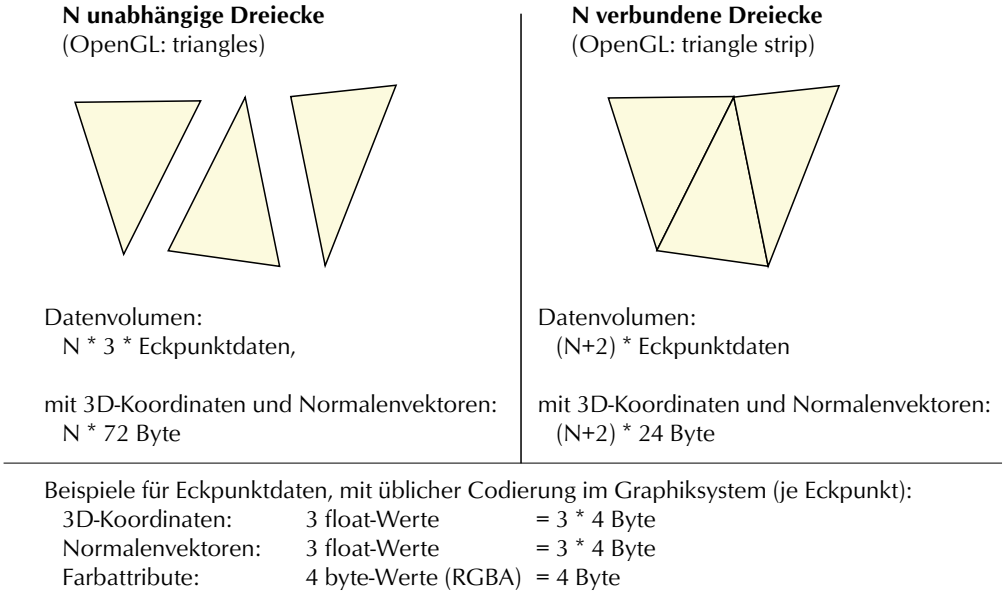


Abb. 12. Komplexität typischer Renderingprimitive:
Unabhängige Dreiecke versus verbundene Dreiecksnetze

Viewer bzw. Plugins für VRML

Bevor die heute verbreiteten VRML97-Viewer – wie z. B. CosmoPlayer (unter dem Arbeitstitel „PlatinumPlayer“ inzwischen „Open Source“-Freigabe angekündigt [142]) und VRwave [3] – entwickelt wurden, entstanden mit unterschiedlichen Zielsetzungen verschiedene andere 3D-Viewer für WWW-Anwendungen. In der Anfangszeit des WWW wurden die bereits vorhandenen 3D-Viewer in die damals verfügbaren WWW-Browser (Mosaic) integriert, sowohl für Inline-Darstellungen von 3D-Präsentationen auf WWW-Seiten (CLRMosaic) als auch zur Verwendung als separate Viewer-Applikation (WebOOGL). Später wurde auch auf effiziente Darstellung geachtet (z. B. i3D [8]), und es wurden neuartige Konzepte prototypisch erarbeitet bzw. in Produkte umgesetzt. Dazu gehören Ansätze zur Einbettung komplexer 3D-Objekte in digitale Dokumente, in denen hierarchische Oberflächennetz-Reduktionstechniken zur progressiven Übertragung und Darstellung eingesetzt werden [27][73].

Streamingverfahren in 3D/VR-Anwendungen

Der Begriff *Streaming* wird im Kontext der 3D/VR-Anwendungen mit mehreren unterschiedlichen Bedeutungen verwendet:

1. Audio-/Video-Streaming innerhalb einer 3D-Szene:
   Hier geht es um die Integration kontinuierlicher audio-visueller Medienströme, die von einem Streamingserver (siehe auch *Abschn. 2.2.1*) ausgespielt oder in einem interpersonellen Kommunikationsszenario übertragen werden. Dies kann z. B. über virtuelle Videobildschirme – d. h. ein mit Video-Textur versehenes Rechteck im virtuellen Raum – und räumlich angeordnete Audio-Quellen geschehen.

2. Progressiver Bildaufbau statischer 3D-Szenen:
   Hier sind zwei Varianten zu unterscheiden:

   (a) Durch eine zugrunde liegende hierarchische Codierung ist die Szene mit einem während der Übertragung zunehmenden Detaillierungsgrad darstellbar. Dies ist vergleichbar mit den Darstellungsverfahren für 2D-Pixelbilder, die in WWW-Browsern üblicherweise für Interlaced-GIF- oder Progressive-JPEG-Dateiformate angewandt werden, um nach relativ kurzer Ladezeit bereits einen groben Überblick zu gewinnen.
   (b) Es erfolgt eine inkrementelle Darstellung, bei der eine Szene mit festgelegter Topologie während der Übertragung bereits stückweise dargestellt wird. Für den Nutzer besteht nicht nur der Komfort, der durch den zu beobachtenden Fortschritt der Übertragung geboten wird. Darüber hinaus kann die potentielle Nebenläufigkeit von Transport- und Präsentationsprozessen prinzipiell zur Verkürzung der Gesamtzeit ausgenutzt werden.

3. Zeitabhängige Updates der 3D-Szene:
   Hier werden Veränderungen in der virtuellen Szene bzw. eines Teils daraus

   (a) durch inkrementelle Codes oder
   (b) durch Ersetzung durch eine jeweils vollständige, aktualisierte (Teil-)Szenenbeschreibung

   bewirkt.

   Als Anwendungsszenarien kommen z. B. 3D-Conferencing (geometrische Abbildung des bewegten Gesichts) bzw. die Visualisierung von Ergebnissen des Hochleistungsrechnens (3D-Exploration von Zeit- oder Parameter-Schritten mehrdimensionaler Datensätze) in Betracht.

Keine der hier beschriebenen Ausprägungen von Streamingverfahren wird im derzeit gültigen VRML-97-Standard unterstützt. Sie werden jedoch in Arbeitsgruppen zur Weiterentwicklung des VRML-Standards – z. B. in der „VRML Streaming (VS) Working Group“ (`www.web3d.org/WorkingGroups/vrml-streams`) – sowie im Kontext des MPEG-4-Standards diskutiert. Seit Mitte 1999 bestehen Bestrebungen, die bislang unabhängig vorangetriebenen Arbeiten der VRML- und

MPEG-Arbeitsgruppen konvergieren zu lassen. Realisierungen dieser Konzepte finden sich teilweise bereits in prototypischen Entwicklungen oder proprietären Produkten.

Für die im Rahmen dieser Arbeit realisierten Werkzeuge wurde die Integration von Verfahren gemäß 2b und 3b angestrebt und für die Präsentation statischer Einzelszenen bzw. zur Ausspielung von Szenensequenzen realisiert (siehe *Kap. 4*).

2.3 Wissenschaftliche Visualisierung

2.3.1 Motivation

Das Sinnesorgan Auge besitzt für die Kommunikation des Menschen mit seiner Umwelt eine überragende Bedeutung, die hauptsächlich auf der hohen Leistungsfähigkeit der visuellen Wahrnehmung beruht. Daher liegt es nahe, dies – neben weiteren Wahrnehmungskanälen und Lernhilfen [192] – für die Exploration wissenschaftlicher Ergebnisse bzw. zur Erschließung komplexer Zusammenhänge zu nutzen. Fortgeschrittene Präsentations- und Interaktionstechnologien werden in verteilten Szenarien mittels breitbandiger Kommunikationsnetze inzwischen leistungsfähig anwendungsreif. Dadurch werden neuartige Paradigmen ermöglicht: vom passiven Postprocessing über die interaktive Mensch-Maschine-Kommunikation bis hin zur „human multimedia interaction“ (HMMI) [40].

Auf dem Stand der Technik für die wissenschaftliche Visualisierung sind heute Arbeitsumgebungen, die der „Virtual-Reality“-Metapher [4] entsprechen, zum Teil als kombinierte Visualisierungs- und VR-Systeme [55][56][105][154] bereits verfügbar. Es muß dabei darauf hingewiesen werden, daß der Schritt von der herkömmlichen interaktiven 3D-Graphik zu einer produktiv einsetzbaren immersiven VR-basierten Visualisierungsumgebung erheblichen weiteren systemtechnischen Aufwand bedingt. So sind stereoskopische Displays, Headtrackingsysteme und dreidimensionale Zeigevorrichtungen zwingend erforderlich. Weiterhin müssen zur Erzielung genügend kurzer Latenzzeiten Höchstleistungssysteme für das Rendering – also die Konvertierung der vektororientierten 3D-Szenenbeschreibungen in die gerätespezifischen Rasterbilder – bereitgestellt werden [44][62][193].

Als Beispiel für ein Anwendungsszenario sei hier der „Virtual Windtunnel“ [25] genannt, in dem Strömungsvisualisierung anhand virtueller Modelle betrieben werden kann. Insbesondere die Visualisierung instationärer Strömungen stellt aufgrund der außerordentlich großen Ergebnisdatensätze ein „Grand Challenge“-Problem der wissenschaftlichen Visualisierung dar [103].

Einen Überblick über Techniken und Anwendungssysteme zur wissenschaftlichen Visualisierung bieten [20][23][37][127][158][161].

2.3.2 Ziel und Zweck

Für die wissenschaftliche Visualisierung hatte der NSF-Workshop „Visualization in Scientific Computing“ im Jahr 1987, bzw. die daraufhin veröffentlichte Studie [111], einen großen Einfluß. Daraus sei hier zitiert:

> „As a tool for applying computers to science, visualization offers a way to see the unseen. As a technology, Visualization in Scientific Computing promises radical improvements in the human/computer interface and may make human-in-the-loop problems approachable.“

Seit dieser Zeit ist die Rechenleistung erheblich erhöht worden, Rechner können größere Datenmengen handhaben, die Weitverkehrsnetze sind leistungsfähiger, und neuartige Benutzerschnittstellen ermöglichen die Steuerung von Simulationsrech-

nungen mit Hilfe immersiver virtueller Realität. Daneben können multimediale Informationsdienste (WWW) genutzt werden, und es werden kooperative Netzdienste, wie z. B. Videoconferencing (MBONE), zur Unterstützung der Gruppenkommunikation (Computer-Supported Collaborative Work – CSCW) – bereitgestellt.

Generell sind zwei wesentliche Ziele für die Visualisierung zu nennen (*Abb. 13*):

1. Erkenntnisgewinn in Forschung und Wissenschaft
2. Kenntnisvermittlung
   - zur Präsentation von Ergebnissen vor Fachpublikum,
   - zur Unterstützung der interdisziplinären Zusammenarbeit in der Wissenschaft,
   - für die fortgeschrittene Lehre und
   - für die Öffentlichkeitsarbeit.

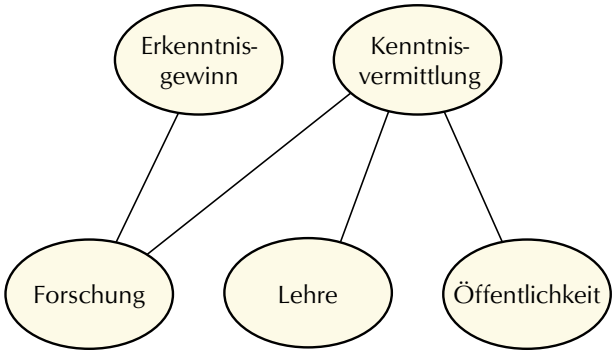


Abb. 13. Ziele der „Wissenschaftlichen Visualisierung“

2.3.3 Graphik- und Visualisierungssysteme

Modell

Zur Erörterung eines Gesamtkonzeptes zur Integration von Höchstleistungssystemen zur Exploration und Präsentation wissenschaftlicher Ergebnisse sei das in *Abb. 14* dargestellte Modell zugrunde gelegt.

Die im Rahmen von leistungsfähigen Visualisierungseinrichtungen zu betrachtenden Maßnahmen beinhalten im wesentlichen die Realisierung der Moduln

1. Filter – Ausschnitt der relevanten Datenbereiche,
2. Mapper – Umsetzung der auszuwertenden Größen in 3D-Objekte,
3. Rendering – Konvertierung der vektororientiert repräsentierten, beleuchteten, attributierten 3D-Objekte mittels virtueller Kamera in ein 2D-Rasterbild – und
4. Display – Ausgabegerät zur visuellen Präsentation.

Zur Verkürzung der Interaktionszyklen dienen zusätzlich

- leistungsfähige Kommunikationsnetze sowie
- fortgeschrittene Interaktionsgeräte, z. B. Trackingsysteme, etc.

Sollen Zwischenschritte gespeichert werden, um z. B. zu einem späteren Zeitpunkt neu aufsetzen zu können oder Rohdaten für Multimedia-Präsentationen bereitzustellen, sind weitere Transportmedien sowie Speicher- und Archivdienste erforderlich. Je nach Anwendung müssen den jeweiligen Anforderungen entsprechend, z. B. Austauschbarkeit, Effizient, Akkuratheit der Repräsentation, adäquate Dateiformate ausgewählt werden.

Das Modul *Datenquelle* generiert

(a) Rechenergebnisse, z. B. im Kontext des „High Performance Computing“ (HPC) Ergebnisse von Simulationsrechnungen typischerweise mehrdimensionaler, zeitabhängiger Phänomene auf Supercomputern. Dabei werden zwei Klassen von Rechnern unterschieden:

– MPP – massiv-parallele Rechner;

– PVP – moderat parallele Vektorrechner.

Darüber hinaus wird versucht, homogene oder heterogene Verbünde von Supercomputern über Weitverkehrsnetze zusammenarbeiten zu lassen, um einen weiteren Leistungsfortschritt zu erzielen: „Meta-Computing“.

(b) Meßergebnisse, z. B. aus der Akquisition realer Beobachtungen.

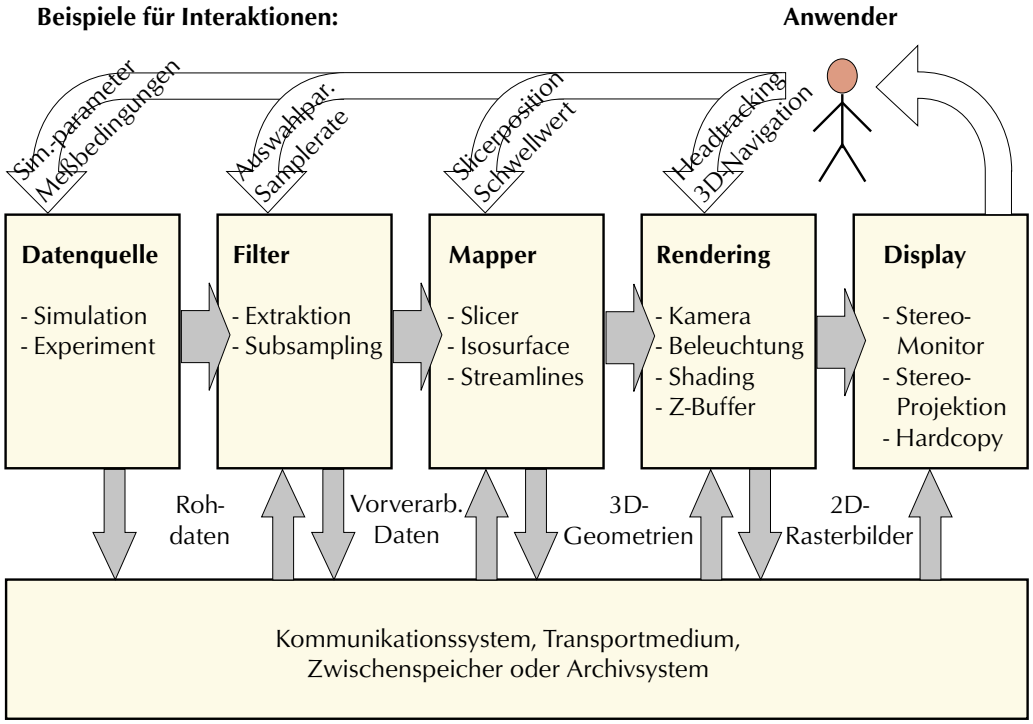


Abb. 14. Modell: „Wissenschaftliche Visualisierung“

Systemplattformen

Als Höchstleistungseinrichtung zur interaktiv nutzbaren Realisierung der *Filter-, Mapper- und Renderingprozesse* sind aus heutiger Sicht 3D-Graphik-Supercomputer der Firma Silicon Graphics (SGI Onyx2 Infinite Reality) als adäquat anzusehen. Diese stellen einen De-facto-Standard für Visualisierungs- und Virtual-Reality-Systeme dar. Softwareprodukte für diese Aufgabenbereiche unterstützen in jedem Fall diese Rechner- und Renderingplattform, zum Teil sogar ausschließlich.

Zunächst dient das auf Shared-Memory-Architektur [170] basierende Multiprozessorsystem zur z. T. äußerst rechenaufwendigen Aufbereitung von Rohdaten mit dem Ziel, geeignete symbolische Abbildungen – d. h. Geometrien, Schnittflächen, Isoflächen, Pfeile, Stromlinien, Partikelsysteme – zu produzieren. Aus den daraus generierbaren virtuellen 3D-Objekten werden schließlich räumlich betrachtbare Szenen zusammengesetzt, die zur anschaulichen Repräsentation der jeweiligen Ergebnisse – z. B. von Simulationsrechnungen – dienen. Die leistungsfähigen Graphik-Pipelines [169] ermöglichen eine interaktiv navigierbare, qualitativ hochwertige visuelle Präsentation dieser häufig hochkomplexen 3D-Szenen. Leistungsparameter sind hier z. B. die Polygonrate und Pixelfüllrate im Zusammenhang mit Multisampling-Antialiasing in Echtzeit. Als offene, hochperformante, hardwarenahe Renderingprogrammierschnittstelle auf der Basis einfacher Renderingprimitive dient OpenGL [122]. Systemspezifische Erweiterungen (OpenGL-Extensions) müssen verwendet werden, um spezielle Funktionalitäten oder Leistungsmerkmale ausnutzen zu können. Eine Programmierung auf höherem Abstraktionsniveau (Szenengraphen) ermöglichen die Optimizer-, OpenInventor- und Performer-Graphikbibliotheken – letztere dienen insbesondere auch der besseren Ressourcenausnutzung auf Multiprozessor- und Multi-Graphikpipe-Maschinen.

Dadurch können innovative Interaktionstechniken für die Visualisierung produktiv angewandt werden – hier sind Methoden der Virtuellen Realität bis hin zur Steuerung von entfernt ablaufenden, eng ins Gesamtsystem gekoppelten Simulationsrechenprozessen zu nennen.

Weitere Gerätetechnik ist für den *Display-Prozeß* erforderlich, also die immersive VR-Präsentation und Interaktion. Damit sind Bildschirme bzw. fortgeschrittene projektions- oder kopfgebundene Displays, wie z. B. Workbench-Systeme [101], Stereo-Großbildprojektion oder CAVE – ein begehbares Rundum-Stereodisplay, entwickelt am Electronic Visualization Laboratory, University of Illinois Chicago [22][32] mit bis zu 6 stereoskopischen Rückprojektionswänden [72] – in Verbindung mit Trackingsensoren (z. B. das magnetfeldbasierte Polhemus 3Space Fastrak [147] oder das ultraschallbasierte InterSense IS-600 Mark II [74][102]) bzw. dreidimensionalen Zeigevorrichtungen oder Eingabegeräten, wie z. B. SpaceMouse, SpaceBall, gemeint.

Daneben ist weiterhin Bedarf zur Produktion von hochqualitativen Hardcopy-Medien, wie z. B. auf Papier, Folie, Video und möglicherweise zukünftig auch auf 3D-Plottern, vorhanden.

Leistungsanforderungen an Rendering und Datentransport

Das Renderingsystem eines 3D-Graphikrechners verarbeitet einfache 3D-Primitive, wie z. B. im Raum angeordnete Dreiecke. Für jeden Eckpunkt sind dabei zumindest die 3D-Koordinaten (typ. 4 Byte je Komponente) zu spezifizieren; optional ist die Angabe von 3D-Normalenvektoren (typ. 4 Byte je Komponente), Farbwerten (typ. 4 Byte – RGBA), 2D-Texturkoordinaten (typ. 4 Byte je Komponente).

Für N Dreiecke mit Normalen je Eckpunkt beträgt das Datenvolumen n bei

unabhängigen Dreiecken:

$$n = 72 \cdot N \text{ byte} \tag{12}$$

miteinander verbundenen Dreiecken („Triangle Strips"):

$$n = 24 \cdot (N + 2) \text{ byte} \tag{13}$$

Die Renderingrate für Triangle Strips mit Normalenspezifikation und $N \gg 2$ beträgt auf einer SGI Infinite Reality (z. B. Modell DX-03 aus dem OpenGL-Benchmark *viewperf* [140], Rendering im „immediate mode" [137]; siehe auch *Abb. 33, S. 88*) ca. 4,3 Mio. Dreiecke pro Sekunde. Es entsteht somit ein interner Graphikdaten-Durchsatz von

24 x 4,3 Mio. byte/s = ***0,8 Gbit/s***.

Sollen 3D-Szenen von einem entfernten Server (z. B. WWW, DocShow-VR [133][135][137]) abgefragt und „on-the-fly" gerendert werden, so müssen Speicher- und Kommunikationssysteme diese Bitraten erzielen können, um nicht signifikante zusätzliche Latenzzeiten hervorzurufen.

Interaktiv – d. h. mit einer Renderingrate von 10 Frames pro Sekunde – navigierbare 3D-Szenen enthalten in diesem Fall max. ca. 0,4 Mio. Dreiecke mit einem Datenvolumen von

9,6 MByte je Szene,

d. h. ein „3D-Film" mit z. B. 1000 Einzelszenen benötigt in diesem Fall ein Datenvolumen von

9,6 GByte.

Die tatsächlich handzuhabenden Datenvolumina können allerdings ohne weiteres eine Größenordnung höher liegen – z. B. durch weicher ausgelegte Echtzeitanforderungen und dadurch erreichbare höhere Komplexitäten, durch größere Anzahl von Zeitschritten, durch ineffizientere Geometrie-Repräsentation, zusätzliche graphische Attribute oder die Notwendigkeit, Zwischenergebnisse vorhergehender Verarbeitungsschritte gleichzeitig im Speicher zu halten.

Um geometrische Repräsentationen auch auf weniger leistungsfähigen Rechnern handhaben zu können und eine verteilte Anwendung in Kommunikationsnetzen mit niedrigeren Datenraten zu ermöglichen, werden verschiedene Ansätze – zum Teil in Kombination – verfolgt. Dazu gehören: dichtere Packung von Graphikprimitiven (z. B. „Triangle Stripification", siehe *Abb. 12, S. 31*), Polygonreduktion von Oberflächennetzen (gegebenenfalls auch in mehreren, hierarchisch aufbauenden Auflösungsniveaus), topologieorientierte Geometriekompression sowie Kombinationen

dieser Verfahren (siehe auch *Abschn. 2.2.2*). In Anwendungen mit hohen Leistungsanforderungen – z. B. bezüglich der Darstellungsgenauigkeit, Interaktivität – und leistungsfähigen verfügbaren Ressourcen – z. B. VR-Graphikrechner, lokale Datennetze – können durch derartige Verfahren jedoch zusätzliche Qualitätsnachteile entstehen. Beispielsweise kann die Akkuratheit der Präsentation durch die Anwendung der meist verlustbehafteten Kompressionsverfahren leiden, oder die Bildrate bzw. Latenz kann sich durch den hohen erforderlichen Rechenaufwand verschlechtern.

Szenarien

Eine enge Kopplung aller in *Abb. 14* dargestellten Verarbeitungsprozesse stellt einen Idealfall dar. Der Anwender ist hier in einer interaktiven Weise in den Produktionsprozeß eingebunden, um in möglichst kurzen Iterationszyklen Ergebnisse erzielen bzw. Interpretationen ermöglichen und in direkter Weise auf Komponenten bzw. Parameter Einfluß nehmen zu können. Wenn dies mit Methoden der immersiven virtuellen Realität geschieht, in der nicht nur die Visualisierung, sondern auch die Bedienoberfläche dreidimensional präsentiert und gesteuert wird, entsteht dadurch eine Arbeitsumgebung, die zur Unterstützung „virtueller Experimente“ geeignet ist.

Werden die erforderlichen Verarbeitungsprozesse auf zwei Instanzen verteilt, sind mehrere Partitionierungsansätze denkbar (*Abb. 15*).

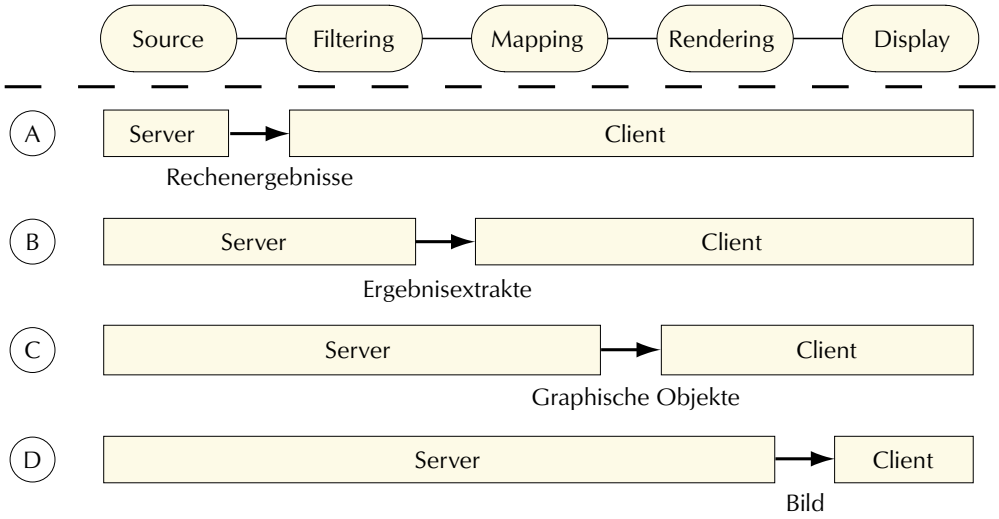


Abb. 15. Strategien für verschiedene Partitionierungsansätze

Technische und organisatorische Randbedingungen bei der Nutzung der angebotenen Ressourcen führen dazu, daß je nach Projekt individuelle Partitionierungs- und Anwendungsszenarien verfolgt werden. Insbesondere ist ein vollständig interaktiver Online-Produktionsprozeß nur ausnahmsweise realisierbar, da MPP bzw. PVP effizienter im Batch betrieben werden können und sich die Ergebnisse relevanter Problemgrößen häufig nicht in genügend kurzen Rechenzeiten ermitteln lassen. Es muß daher generell die Speicherung von Zwischenergebnissen betrachtet werden, nicht zuletzt um auch eine effiziente Reproduktion bestimmter Darstellungen zu ermöglichen.

Daher sind sowohl enge als auch lose Kopplungsformen zu betrachten, d. h.

- synchrone (z. B. RPC, PVM, MPI [54]) Kommunikation oder
- asynchrone (z. B. FTP, HTTP),
  d. h. über temporäre Dateien entkoppelte Kommunikation.

Darüber hinaus müssen Prozesse zur visuellen Aufbereitung durch räumlich entfernte Benutzer initiiert werden können. Je nach gewünschtem Ergebnis, d. h. extrahierte Rohdaten, 3D-Objekte, Bilder oder Videos, können dies Filter, Mapper oder Renderer sein. Ein leistungsfähiges Display-System muß allerdings beim Nutzer installiert sein, und die adäquate kommunikationstechnische Anbindung bedingt möglichst leistungsstarke Netze im Gbit/s-Bereich. Da das zur Realisierung von VR-Konzepten notwendige interaktive Rendering nach dem Stand der Technik direkt an das Display, d. h. an den Benutzerarbeitsplatz, gebunden ist (abgesehen von speziellen, in Entwicklung befindlichen Client-Server-Produkten, wie z. B. GLR [94] oder Übertragung komprimierter Videodaten [93]), werden für eine entfernte Nutzung vor Ort leistungsfähige 3D-Graphikrechner benötigt. Die vor Ort installierte Konfiguration kann dann z. B. die im Schema nach *Abb. 14* dargestellten Rendering- und Display-Prozesse übernehmen, also an der 3D-Geometrie-Schnittstelle ankoppeln. Zu diesem Zweck – aber auch zur Unterstützung künftiger kooperativer Konzepte bis hin zu verteilten VR-Szenarien (z. B. CAVE-to-CAVE [189]) – sind in den regionalen Rechenzentren 3D-Graphikrechner mittlerer Leistungsfähigkeit (derzeit „Deskside- bzw. Rack-Systeme“ von SGI, z. B. Onyx2 Infinite Reality) sowie stereoskopisches Display- und Interaktionsgerät einzuplanen.

Für Vorarbeiten sowie die Teilnahme an kooperativen 3D-Szenarien direkt am Projektarbeitsplatz wären auch kompatible Arbeitsplatzrechner wünschenswert („High-End-Desktop-Graphikworkstation“, z. B. SGI Octane). Dabei ist zu berücksichtigen, daß State-of-the-Art-Software für Visualisierung in immersiven VR-Szenarien, wie z. B. CAVE-Libraries und CAVE-Simulator zur Unterstützung von Vorbereitungsarbeiten am Arbeitsplatzrechner, derzeit ausschließlich für SGI-Systeme bereitgestellt wird.

Damit ergibt sich für Hochschulen und Forschungseinrichtungen ein mögliches Konzept dreier Ausstattungsebenen (*Abb. 16*).

Wird eine vollständige geschlossene Interaktionsschleife gemäß VR-Metapher nicht benötigt oder ist ein solches interaktives Szenario aufgrund zu großer Datensätze bzw. zu geringer Bitraten oder zu langer Verarbeitungszeiten nicht realisierbar, lassen sich z. B. Filter-, Mapper- und ggf. Renderer-Konfigurationen scriptmäßig vordefinieren. Auf dieser Grundlage kann die Produktion von 3D-Szenen, Bild-Sequenzen oder Videofilmen batchmäßig in Auftrag gegeben werden, um anschließend im Internet verteilt oder auf qualitativ hochwertige Videocassetten (z. B. Betacam-SP) aufgezeichnet zu werden.

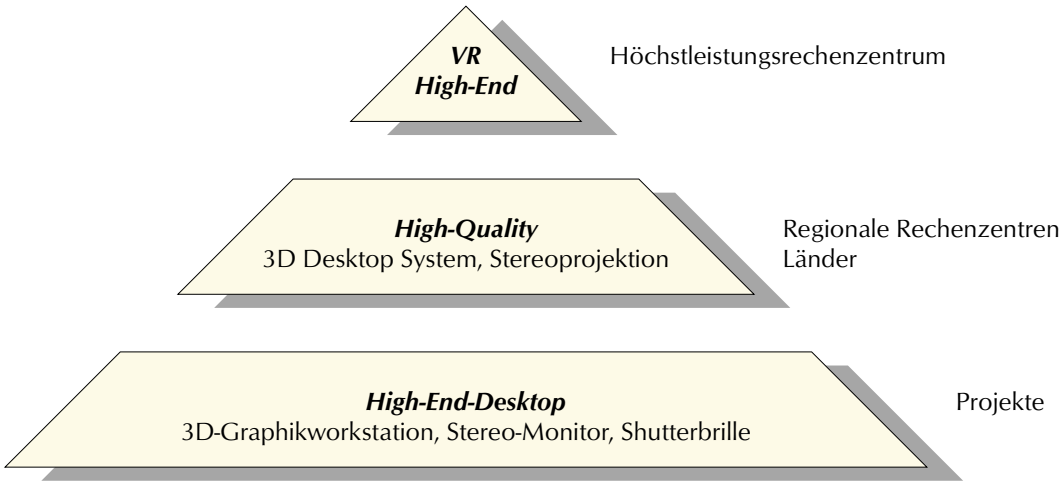


Abb. 16. Ausstattungsebenen zur Hochleistungsvisualisierung für Hochschulen und Forschungseinrichtungen

Einzelne oder Serien von statischen 3D-Szenen, die über Visualisierungssoftware – interaktiv oder im Batch – in Standardformaten (z. B. VRML) erzeugt werden, können über Informationsdienste, d. h. integriert in WWW-Szenarien, abgerufen und auf Clientseite interaktiv navigierbar dargestellt werden. Hiermit kann ein asynchroner Partitionierungsansatz nach der in *Abb. 15* dargestellten Strategie „C“ realisiert werden. Vorteilhaft an dieser Vorgehensweise ist, daß Filter- und Mapper-Verarbeitungsschritte effizient mit auf dem Host-Rechner (MPP/PVP) durchgeführt werden können und damit eine Reduktion der Komplexität der Daten an der Quelle bewirkt werden kann.

Für die effiziente Online-Präsentation der an dieser Schnittstelle generierten virtuellen 3D-Objekte ist das in dieser Arbeit entwickelte System geeignet.

Videoproduktionsdienste, die sowohl Digitalformate für Online- (WWW) oder Offline-Anwendungen (CD-ROM) als auch die Aufzeichnung auf Videocassetten unterstützen, werden z. B. im Multimedia-Labor des RRZN/RVS angeboten und weiterentwickelt [39][132][134][138].

Generell lassen sich die verschiedenen genannten Szenarien den folgenden Klassen von Graphik- bzw. Visualisierungssystemen zuordnen, die in dieser Reihenfolge auch die historische Entwicklung der Technologie widerspiegeln [44][100]:

1. *Passive Systeme*, charakterisiert durch:

   - rudimentäre Benutzersteuerung,
   - stapelorientierte Verarbeitung,
   - Anwendung in Verbindung mit

     – langsamen Ausgabegeräten (z. B. Drucker, Plotter) oder
     – zeitaufwendigen Prozessen, z. B. Generierung von graphischen Metafiles mittels Graphikbibliothek (z. B. UNIRAS, VTK [161]) direkt aus einem Batch-Simulationslauf oder scriptbasierte Animation mit qualitativ hoch-

wertigem Renderingverfahren wie Raytracing, welche nicht durch echtzeitfähige Graphik-Subsysteme unterstützt werden können.

2. *Interaktive Systeme*, charakterisiert durch:

   - fensterbasierte, zweidimensionale, mausgesteuerte Desktop-Nutzerinterface,
   - effiziente und zielgerichtete Interaktion,
   - z. T. bereits implementierte Unterstützung stereoskopischer Präsentation [64] – durch Einsatz von Trackingsystemen auf dem Weg zu einem immersiven System, mit der Einschränkung, daß zur Bedienung ein Kontextwechsel bezüglich der 2D-Präsentation und -Interaktion durchgeführt werden muß,
   - Anwendung in Visualisierungssoftware auf dem Stand der Technik, wobei unterschieden werden muß zwischen

     – universell einsetzbaren Baukastensystemen, die häufig nach dem Datenflußmodell arbeiten und deren Moduln z. T. auch auf verschiedenen, im Netz verteilten Rechnern ablaufen können (z. B. AVS [191], IBM Data Explorer, IRIS Explorer) und
     – monolithischen Postprocessingsystemen (z. B. Wavefront Data Visualizer, UNIRAS Gsharp, PV-WAVE, ISVAS, Ensight), bei denen der Benutzer von der z. T. aufwendigen Einarbeitungs- und Entwicklungsarbeit weitgehend entlastet ist, die allerdings häufig für spezielle Einsatzgebiete ausgelegt sind und eine Erweiterung nicht bzw. nur ungenügend unterstützen.

3. *Immersive Systeme*, charakterisiert durch:

   - intuitive dreidimensionale Präsentation und Interaktion („Virtual Reality“),
   - Unterstützung von stereoskopischen Darstellungssystemen:

     – Monitor in Zusammenhang mit Shutterbrille,
     – Großbildprojektion, ggf. auf mehreren Wänden (CAVE) mit alternierender Präsentation (ein Projektor je Fläche) – betrachtbar mittels Shutterbrille – oder kontinuierlicher Projektion mit unterschiedlicher Polarisationsrichtung (zwei Projektoren je Fläche) – betrachtbar mittels Polarisationsfilter-Brille – oder
     – kopfgebundene Bildschirme (z. B. Head-Mounted Display, BOOM),

   - Einsatz innovativer dreidimensionaler Interaktionstechniken bzw. Bedienoberflächen – Ergebnis aktueller Forschung auf diesem Gebiet,
   - harte Echtzeitanforderungen, d. h. Iterationszyklen mit max. 0,1 Sekunden.

2.3.4 Anwendungsgebiete immersiver Virtual-Reality-Systeme

Typische Anwendungsgebiete für immersive VR-Systeme sind Simulations- und Explorationsszenarien, die vom zeitlichen oder räumlichen Maßstab her oder aus anderen Gründen – z. B. Gefährlichkeit, Kosten – nicht für eine reale Durchführung geeignet sind.

Zu unterscheiden sind:

1. Exploration und Visualisierung wissenschaftlicher Rechenergebnisse, z. B. aus

   - Strömungssimulationen, wie beispielsweise in Strömungsmechanik, Strömungsmaschinen, Meteorologie, Klimatologie, Meeres- und Polarforschung,
   - Verformungsanalysen, wie beispielsweise in Umformtechnik und Mechanik sowie
   - Feldsimulationen, wie beispielsweise in Physik, Chemie.

   Dieses Anwendungsszenario entwickelt sich von der batch-orientierten Ergebnisaufbereitung über die interaktive Datenvisualisierung zunehmend in Richtung „virtueller Experimente“. Diese werden von einer engen Kopplung und kurzen Interaktionszyklen zwischen Simulationsrechnung, Visualisierungsfilter/-mapper und Renderingprozessen geprägt, wobei intuitive dreidimensionale Nutzer-Oberflächen in Virtual-Reality-Umgebungen genutzt werden können.

2. Rekonstruktion und Visualisierung mehrdimensionaler Meßdatensätze, z. B.

   - Computertomographie in Verfahrenstechnik oder Medizin und
   - Kartographie.

   Hier spielt die besonders anschauliche räumliche Darstellung und interaktive Wahl der Betrachterperspektive eine zunehmende Rolle. Teilweise sind hohe Rechenaufwände zur Umsetzung der Rohdaten in dreidimensionale Darstellungsformen erforderlich, bevor diese visualisiert werden können.

3. Darstellung und Manipulation komplexer geometrischer 3D-Objekte, z. B.

   - Simulation realer Umgebungen – Architektur und Landschaftsplanung,
   - Molekülvisualisierung in der Chemie,
   - Industriedesign im Automobil- oder Flugzeugbau,
   - Exploration dreidimensionaler Szenen aus den ersten beiden Szenarien – auch aus herkömmlichen passiven oder interaktiven Anwendungssystemen, d. h. statische oder dynamische 3D-Objekte aus der Datenvisualisierung oder Modellierung, die in einzelnen Dateien (z. B. VRML) bzw. Serien repräsentiert werden (z. B. Iso-Oberflächen, Stromlinien, Crash-Modelle, digitale Geländemodelle) sowie
   - Online-Präsentation virtueller 3D-Szenen im Internet mittels in WWW-Browser integrierbaren Viewern, z. B. VRML oder DocShow-VR (siehe *Kap. 4*).

Das zuletzt genannte Szenario (3) ist prinzipiell geeignet, Ergebnisse sämtlicher vorher genannten Anwendungen über Internet-Technologie der Öffentlichkeit darzustellen, sofern der WWW-Client über ausreichende Leistung in bezug auf Datennetzanschluß sowie Rechen- und Darstellungsleistung verfügt.

Durch eine zum Teil automatisierbare Konvertierung der hier noch auf relativ hohem Abstraktionsniveau vorliegenden 3D-Daten in Sequenzen von Einzelbildern oder Videos verschiedener Qualitäten kann jedoch grundsätzlich auch die Betrachtung für Internet-Clients mit geringerer Leistungsfähigkeit ermöglicht werden.

3 Problemanalyse

3.1 Ziel-Szenario

In Anwendungsumgebungen der wissenschaftlichen Visualisierung besteht zunächst der Bedarf, auf Rohdaten, Zwischenergebnisse der Filter- und Mapper-Prozesse oder aufbereitete 3D-, Video- oder Bild-Medien (siehe *Abb. 14, S. 36*) zuzugreifen, die auf entfernten Servern gespeichert sind. Darüber hinaus sollen – zur Verkürzung der Latenzzeiten und damit verbundener effizienterer Nutzung von Rechenkapazitäten – auch Ergebnisschritte einer Berechnung, die mit individuellen Startparametern initiiert wurde, im Pipeline-Verfahren „on the fly“ gesichtet werden, um die Rechnung interaktiv zu terminieren, mit geänderten Parametersätzen erneut zu starten oder während des Verlaufs zu steuern („Tracking“, „Computational Steering“ [201]). Perspektivische Zielsetzung ist die Bereitstellung einer möglichst intuitiven Arbeitsumgebung, eines „Virtual Reality Environment“, in welchem eine multimediale, immersive Interaktion mit dem Modell ermöglicht wird [31].

Zur computergestützten Zusammenarbeit – „Computer-Supported Collaborative Work“ (CSCW) – werden zusätzlich noch kooperative Kommunikationsdienste benötigt, z. B. Synchronisierung der Navigation, Audio-/Video-Conferencing, die letztlich eine Kopplung mehrerer Clientsysteme erfordern.

Aus diesem, in *Abb. 17* dargestellten, Gesamtkonzept wurde jedoch im Rahmen dieser Arbeit zunächst nur das Teilsystem unterhalb der „symbolischen Repräsentation“ betrachtet. Dieses ist durch Zwischenspeicherung der erzeugten 3D-Daten als entkoppelt anzusehen. Als Zukunftsperspektive wird die Integration von akustischen und haptischen Komponenten gesehen.

Als Gründe für die Einschränkung, den entfernten Zugriff auf vorbereitete, dreidimensionale, geometrische Graphikdaten aus dem Gesamtsystem besonders zu vertiefen, sind zu nennen:

- Simulationsrechnungen werden aufgrund der relativ langen Rechenzeiten und besseren Ausnutzung der Rechnerkapazitäten üblicherweise im Batchbetrieb durchgeführt. Leistungsfähigere Höchstleistungsrechner werden häufig nicht für interaktive Rechnungen, sondern vor allem zur Erhöhung der Problemgröße bzw. der Genauigkeit genutzt.
- Die zu visualisierenden Simulationsschritte werden meist nicht in der für eine flüssig darstellbare Online-Simulation erforderlichen Rate bereitgestellt.

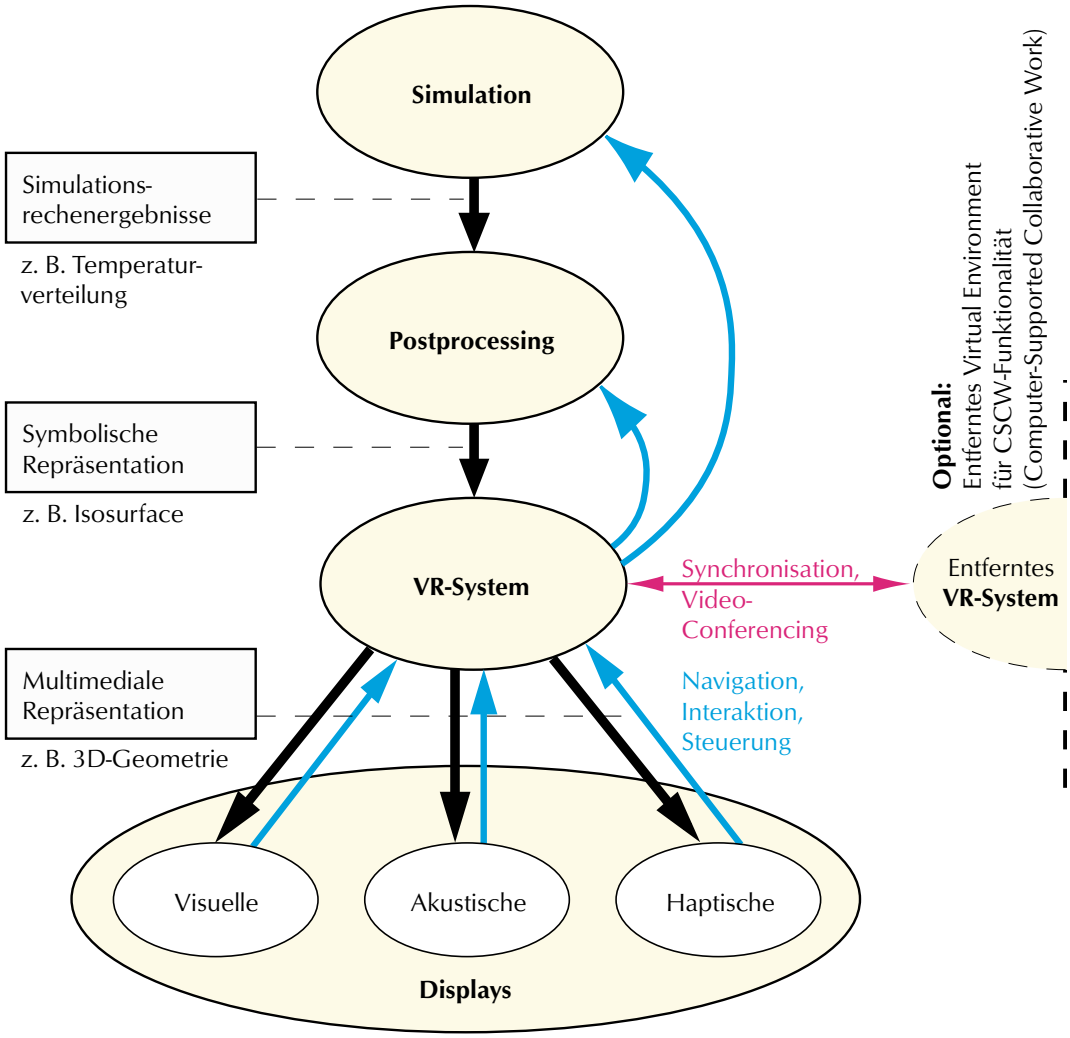


Abb. 17. „Virtual Reality Environment“ für die wissenschaftliche Visualisierung

Die Schnittstelle „symbolische Repräsentation“ wurde gewählt, weil an dieser Stelle durch die Aufbereitung bereits eine Datenextraktion erfolgt ist, die neben der anschaulichen Explorationsmöglichkeit auch zu einer Reduktion des Datenvolumens gegenüber dem Volumen der Rohdaten beiträgt. Dies wird an der Tatsache deutlich, daß in dreidimensionalen Problemen die Rohdaten mit zunehmender Auflösung (n Gitterpunkte je Dimension) eine Komplexität der Ordnung $O(n^3)$ aufweisen, die 3D-Objekte jedoch häufig nur mit $O(n^2)$ (Oberflächenobjekte wie z. B. Slicer, Isosurface) oder $O(n)$ (Linienobjekte, z. B. Streamline) zunehmen [136][152].

Eine Online-Aufbereitung zwischengelagerter oder im Pipelining zu übertragender Rohdaten verbietet sich in vielen Fällen aufgrund der außerordentlich hohen Rohdatenvolumina und Rechenanforderungen für das Postprocessing (siehe auch *Abschn. 2.3.3*).

Eine weitere Datenkompression in Repräsentationen auf niedrigerem Abstraktionsniveau, d. h. Pixelbilder, Videos, kommt nicht in Betracht, da stereoskopische Betrachtung und interaktive, dreidimensionale Navigation gefordert werden.

Die in dieser Arbeit entwickelten Systemkomponenten sollen eine hochqualitative Betrachtung von dreidimensional aufbereiteten Ergebnissen unterstützen, welche auf entfernten WWW- und speziellen Streaming-Servern zum Abruf bereitgestellt werden. Auf Client-Seite soll die Anwendung immersiver Präsentations- und Interaktionsmethoden ermöglicht werden.

Nach dem hier geleisteten Beitrag zur Unterstützung von

- *Präsentationsszenarien* – z. B. zur Präsentation wissenschaftlicher Ergebnisse –

können zukünftig auf dieser Basis tele-immersive, multimodale Erweiterungen für

- *Diskussionsszenarien* – z. B. für die gemeinsame Betrachtung von Ergebnissen in einem geographisch verteilten „Networked Virtual Environment“ [110][155][173], welches letztlich ein „Cooperative HyperMedia System“ [68] darstellt – sowie
- *Explorationsszenarien* – z. B. zur interaktiven Visualisierung und Steuerung von Simulations- und Postprocessing-Prozessen [31][92][105][201], bis hin zu „Virtuellen Experimentierumgebungen“ –

integriert werden.

Die notwendigen Speicher, Prozesse und Kommunikationsflüsse sind in *Abb. 18* vereinfacht dargestellt.

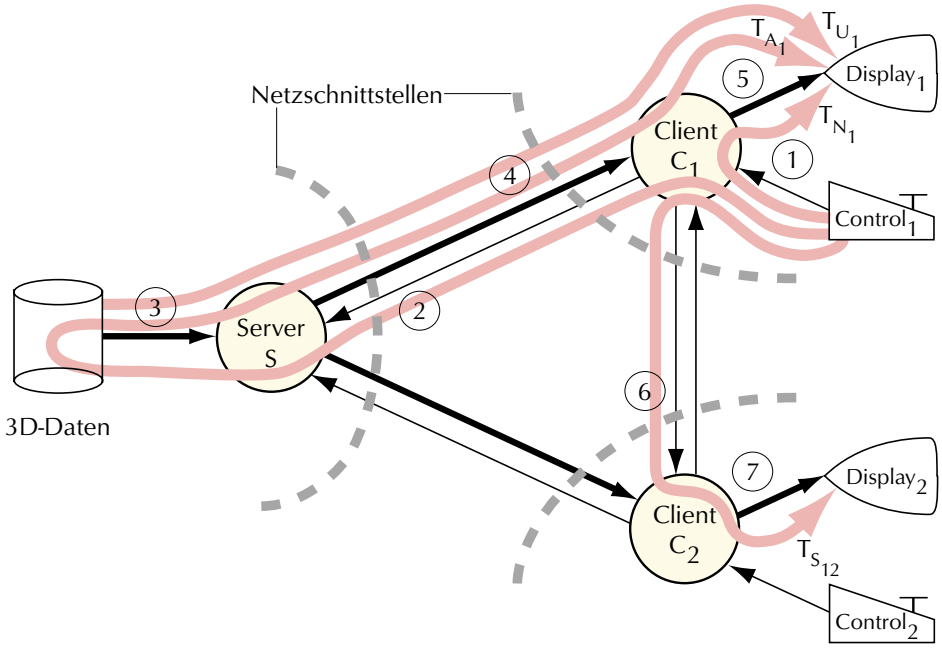


Abb. 18. Modell eines Systems zur verteilten Online-Präsentation von 3D-Szenen

Das Systemmodell entspricht prinzipiell einem Informationssystem. Dabei wird neben dem Abruf einzelner Szenen (Initiative geht stets vom Client aus, statusfrei) auch die Ausspielung von Sequenzen aus dem Server (Verbindung bleibt erhalten) betrachtet. Erweiterungen zur Kooperation mehrerer Clients (hier: 2) werden bereits vorgesehen.

Zur Illustration der Abläufe werden im folgenden die in drei wesentlichen Anforderungsszenarien auftretenden Latenz- bzw. Update-Zeiten diskutiert. Es wird dabei jeweils auf die in *Abb. 18* eingezeichneten Prozesse, Datenflüsse und Endgeräte verwiesen.

1. T_{A_i} : Latenzzeit bei Anforderung einer neuen 3D-Szene von Client C_i.

   Diese setzt sich wie folgt zusammen:
   - Anforderung durch Nutzer, z. B. mittels Mausklick
   - Übertragung des Steuerbefehls vom Interaktionsgerät zum Client C_i (1)
   - Verarbeitung in Client C_i, dabei gegebenenfalls Latenz durch Serialisierung der Abarbeitung weiterer Anforderungen (z. B. Navigation, s. u.)
   - Übertragung der Anforderung (2) zum Server S, ggf. zunächst noch Verbindungsaufbau erforderlich (HTTP)
   - Einlesen der Szenen-Repräsentation (3) von Datei mit `open()` und `read()`, alternativ: Einblendung der Datei in den virtuellen Speicher des Serverprozesses mit `mmap()`.
     Im Fall dynamischer Szenen, z. B. enger Kopplung mit einer Simulationsrechnung, ist zusätzliche Zeit zur Generierung bzw. Bereitstellung erforderlich.
   - Übertragung des Dateiinhalts zum Client C_i (4)
   - Verarbeitung der 3D-Daten, Rendering der 3D-Szene in C_i
   - Update des dargestellten Framebuffer-Bildes (5)

2. T_{N_i} : Latenzzeit bei lokaler Navigation in der 3D-Szene am Client C_i.

   Diese setzt sich wie folgt zusammen:
   - Anforderung durch Nutzer, z. B. mittels Trackinggerät
   - Übertragung der Navigationsparameter – z. B. Position, Orientierung – zum Client C_i (1)
   - Verarbeitung in Client C_i, dabei gegebenenfalls Latenz durch Serialisierung der Abarbeitung weiterer Anforderungen
   - Rendering der 3D-Szene in C_i
   - Update des dargestellten Framebuffer-Bildes (5)

3. T_{U_i} : Zeit zur Aktualisierung der auf Client C_i dargestellten Szene bei der Ausspielung aus dem Server S (Update-Zeit).
   Diese setzt sich wie folgt zusammen:
   - Einlesen der Szenen-Repräsentation (3) von Datei mit `open()` und `read()`, alternativ: Einblendung der Datei in den virtuellen Speicher des Serverprozesses mit `mmap()`.

Im Fall dynamischer Szenen, z. B. enger Kopplung mit einer Simulationsrechnung, ist zusätzliche Zeit zur Generierung erforderlich.
- Übertragung des Dateiinhalts (4) zum Client C_i
- Verarbeitung der 3D-Daten, Rendering der 3D-Szene in C_i
- Update des dargestellten Framebuffer-Bildes (5)

Wegen der potentiellen Nebenläufigkeit der Lese-, Übertragungs-, und Rendering-Prozesse kann die Update-Zeit, je nach Implementierung, geringer als die Summe der angegebenen einzelnen Zeiten sein. Im Idealfall, d. h. bei optimalem Pipelining, entspricht die Update-Zeit dem Maximum der in den genannten Prozessen jeweils benötigten Zeit, zuzüglich gegebenenfalls erforderlicher Startup-Zeiten. Eine derartige parallelisierte Ausführung sollte daher für den Entwurf und die Implementierung des 3D/VR-Streamingsystems in Betracht gezogen werden (siehe auch *Abschn. 4.2.3* und *Abschn. 5.3*).

4. $T_{S_{ij}}$: Latenzzeit bei Synchronisation der Navigation zwischen zwei Clients C_i und C_j.
   Diese setzt sich wie folgt zusammen:
   - Anforderung durch Nutzer, z. B. mittels Trackinggerät
   - Übertragung des Navigationsparameter zum Client C_i (1)
   - Verarbeitung in Client C_i
   - Übertragung der geänderten Navigationsparameter an Client C_j (6)
   - Verarbeitung in Client C_j, dabei gegebenenfalls Latenz durch Serialisierung der Abarbeitung weiterer Anforderungen
   - Rendering der 3D-Szene in C_j
   - Update des dargestellten Framebuffer-Bildes (7)

3.2 Klassifikation der Unzulänglichkeiten und Anforderungen

In den bisher üblichen, auf verbreiteten Internet-Werkzeugen beruhenden und auf VRML-Plugins basierenden Konfigurationen wurden diverse Unzulänglichkeiten und Engpässe identifiziert, die letztlich in einem Anforderungsprofil für die im Rahmen dieser Arbeit entwickelten Moduln mündeten. Dies betrifft die Aspekte *Leistungsfähigkeit, Qualität* und *Funktionalität* – unter Berücksichtigung relevanter Standards, wobei die jeweiligen Punkte noch den Instanzen *Präsentation* (Client, Rendering, Interaktion), *Kommunikation* (Datentransport, Steuerung) und *Repräsentation* (Server, Speicherung) zugeordnet werden können. In diesem Zusammenhang sei auch auf *Abschn. 2.1.2.2* (Präsentationsmedien), *Abschn. 2.1.2.5* (Kommunikationsmedien) und *Abschn. 2.1.2.3* (Repräsentationsmedien) verwiesen.

Die jeweilige Bewertung erfolgte unter besonderer Berücksichtigung der Anwendung zur Unterstützung der wissenschaftlichen Ergebnispräsentation, wie in *Abschn. 3.1* beschrieben.

Eine Übersicht der Anforderungen wird in *Tabelle 4* gegeben.

	Präsentation	Kommunikation	Repräsentation
Leistungsfähigkeit	• Kurze Latenz – Transport (Startup) – lokale / entfernte Interaktion • Hohe Bildrate – Navigation – Renderingleistung: Transformationen, Lichtrechnungen, Pixelfüllrate, effiziente Primitive	• Performante Protokolle und APIs • Effiziente Implementierung – Verwendung großer Blöcke – Vermeidung von Kopiervorgängen – Optimierung der Socket-Optionen	• Möglichst geringe Rechenaufwände – Decodierung – Dekomprimierung – Parsen des Szenengraphen – Berechnung von Normalenvektoren • Möglichst geringes Datenvolumen
Qualität	• Auflösung – Pixelanzahl – Farbtiefe – Zeit • Antialiasing – Multisampling • Color Management – Berücksichtigung eines ICC-Profils zur Ausgabegeräte-Charakterisierung	• Quality of Service – Hohe Bitraten – Kurze Latenz – Geringer Jitter – Niedrige Fehlerrate – Niedrige Verlustrate	• Auflösung – Geometrie (Quantisierung, Komplexität) – Normalenvektoren – Farben, Texturen – Timing-spezifikation – Lichtquellen – Virtuelle Kameras • Kompression – verlustfrei! • Color Management – geräteunabhänge Farbspezifikation gemäß ICC-Profil
Funktionalität	• Streaming – On-the-fly-Präs. – Dynamische Szenen • Stereoskopische Sichtgeräte • Navigationskomfort – Trackingsysteme – 3D-Eingabegeräte • Breite Plattformunterstützung – UNIX: SGI, Sun, HP – Microsoft: Windows 95/98/NT	• Streamingprotokolle – Steuerung – Mediendaten-Transport • Synchronisation – intrastream – interstream • Medienspezifische Skalierung – Bildrate – Auflösung – Bildqualität – Level of Detail	• Notwendige Attribute für Virtual-Reality-Präsentationstechnik, z. B. – focalDistance • Sequenzen einzelner 3D-Szenen: – „Virtual Reality Movie“

Tabelle 4. Klassifikation der Anforderungen

In *Tabelle 5* wird auf die zu berücksichtigende Standardisierung der Verfahren bzw. die verfügbare Gerätetechnik eingegangen.

	Präsentation	Kommunikation	Repräsentation
Standards	• Generische, erweiterbarer WWW-Browser – Netscape Communicator, Microsoft Internet Explorer – Plugin-Interface • Programmierschnittstellen (APIs) – 3D-Graphik: GKS-3D [78], Java-3D [183], OpenGL [122], PHIGS [80] – GUI: X-Windows, OSF/Motif, Win32 – Netscape-Plugin-Callbacks [125] • Stereoskopische Darstellung	• Netzinfrastruktur, z. B. Endgeräte-Interfaces – Ethernet: 10/100/1000 Mbit/s – ATM: 155/622 Mbit/s • Protokolle – IP – TCP, UDP – HTTP, RTSP, ... • APIs – BSD-Sockets – TLI-Sockets	• WWW-Server • Dynamische Dokumente – CGI • Cache- / Proxy-Techniken • Multimedia- und 3D/VR-Datenformate, z. B. – VRML97 [86] – Etc.: *Tabelle 2, S. 16*

Tabelle 5. Standards (Beispiele)

Die einzelnen in *Tabelle 4* und *Tabelle 5* aufgeführten Punkte werden in den folgenden Abschnitten näher erläutert.

3.2.1 Leistungsfähigkeit

Präsentation

Die *Startupzeit* auf Client-Seite ist für die produktive Arbeit im allgemeinen wesentlich zu lang. Sie liegt für typische Szenen mit ca. 100.000 Polygonen in der Größenordnung von Minuten, so daß die Interaktivität stark eingeschränkt ist. Dies wird hauptsächlich durch die Aufbereitung des meist in komprimierter Form (gzip) vorliegenden Klartextformats VRML in die rechnerinterne, zum Rendering geeignete Darstellung verursacht, welche äußerst rechenaufwendig ist. Dies betrifft Dekompression, Decodierung, Parsen, Aufbau und Traversierung des Szenengraphen, gegebenenfalls auch Berechnung von Normalenvektoren und Erzeugung einer Display-Liste auf Client-Seite.

Die *Framerate*, die sich letztlich auf die Navigationslatenz auswirkt, ist relativ gering. Dazu wurden in einer Studie Vergleiche mit der prinzipiell möglichen Rate, wie sie z. B. mit dem OpenGL-Benchmark *viewperf* [140] gemessen wurde, durchgeführt [146]. Zu den möglichen Ursachen zählen:

- Optimierte Renderingprimitive, z. B. *Triangle Strips*, wie sie in Visualisierungen häufig erzeugt werden und mit denen hohe Graphikdurchsätze erzielt werden können, werden in VRML nicht unterstützt. In der Darstellung gleichwertige VRML-Elemente, z. B. *IndexedFaceSet*, also eine Liste von Polygonen, könnten zwar auf Clientseite optimiert werden, indem z. B. *Independent Triangles* bzw. *Triangle Strips* automatisch erkannt werden. Dies würde aber zusätzlichen Rechenaufwand erfordern, d. h. zumindest in einer Erhöhung der Startupzeit resultieren.
- Die Implementierung des Traversierens und Renderns der aus VRML decodierten und geparsten Objektstruktur ist ineffizient. Die Struktur des VRML-Formats trägt zur Komplexität dieses Problems bei. Zur Ineffizienz des Renderingvorgangs führen unter anderem auch zu häufige Graphik-Statuswechsel (z. B. Materialänderungen etc.) und unzureichende Realisierung von Schleifen (z. B. zu kleine Schleifenkörper, kein Loop-Unrolling).

Kommunikation

Für den leistungsfähigen Datentransport müssen auf einer hochratigen Netzinfrastruktur möglichst *performante Protokolle und Programmierschnittstellen* (APIs) verwendet werden. Um die Endsysteme nicht zu einem Engpaß der Anwendung werden zu lassen, ist bei der effizienten Implementierung generell darauf zu achten, daß möglichst große Datenblöcke zusammenhängend verarbeitet werden können, Kopieroperationen möglichst vermieden werden und auf Anwendungsseite verfügbare Optionen zur Optimierung der Transportprotokolle (z. B. TCP/IP-Socket-Optionen) ausgenutzt werden [14][19][126]. Für die Übertragung mehrerer aufeinander folgender Medien-Elemente sollten möglichst persistente Verbindungen verwendet werden, um zusätzliche Performance-Nachteile zu vermeiden, die durch den Overhead beim Verbindungsaufbau entstehen, wie z. B. den Slow-Start-Algorithmus von TCP/IP [90][91][175][179].

Repräsentation

Um die oben erläuterten client-seitigen *Rechenaufwände möglichst gering* zu halten, sollte die Codierung möglichst an den Präsentationsalgorithmus angepaßt sein. Hier kommen zur Darstellung polygonaler 3D-Graphik insbesondere gepackte Listen aus Eckpunkten und Attributwerten, d. h. Normalenvektoren sowie gegebenenfalls Farbwerten, in binärer 32-Bit-Binärcodierung in Betracht. Diese lassen sich z. B. unmittelbar effizient mittels Renderingroutinen auf Basis des auf breiter Plattform verfügbaren 3D-Graphik-APIs OpenGL präsentieren.

Möglichst geringe Datenvolumina reduzieren den Kommunikationsaufwand. Jedoch können Kompressionsverfahren durch zusätzliche Rechenaufwände zur Kompression und Dekompression den Gesamtzeitaufwand deutlich erhöhen und je nach Verfahren Ungenauigkeiten in der Darstellung einführen (siehe auch *Abschn. 2.2.2*).

3.2.2 Qualität

Präsentation

Die *Auflösung* einer graphischen Darstellung wird durch verschiedene Parameter beeinflußt, z. B. Pixelanzahl bzw. Ortsauflösung und Farbtiefe des Wiedergabemediums sowie durch Diskretisierung und erzielbare Genauigkeit des zeitlichen Ablaufs, die z. B. durch die oftmals vorliegende Asynchronität von Update-Anforderungen und Bildwiederholrate des Bildschirms bzw. der Graphikkarte einen gewissen Jitter erzeugt.

Aliasing-Artefakte können besonders auf den niedrig-aufgelösten Computer-Bildschirmen beobachtet werden. Diese wirken sich oft störend als „Treppenstufen-Effekte“ an geneigten Kanten aus, gelegentlich auch als Moiré-Effekte. Auf der anderen Seite sind verschiedene Antialiasing-Verfahren bekannt und verfügbar, die zur Reduktion dieser Artefakte dienen können. Beispielsweise bieten die High-End-Graphikcomputer der Firma Silicon Graphics (z. B. SGI Onyx2 Infinite Reality) generell eine aufwendige Hardware-Unterstützung in Form eines Multisampling-Antialiasing, welches als OpenGL-Extension angesteuert werden kann.

Ein weiterer Gesichtspunkt zur Erzielung hoher Bildqualität ist die geräteunabhängige Reproduktion von Farben. Diese kann durch die Integration von *Color-Management-Systemen (CMS)* [26][59] erreicht werden, welche auf neutralen Farbspezifikationen beruhen. Durch die Verwendung von standardisierten Farbprofilen, sogenannten ICC-Profilen [70] – kann eine korrekte Farbwiedergabe prinzipiell unterstützt werden. Allerdings müßten dazu die in den Betriebssystemen zum Teil bereits verfügbaren CMS-Funktionen genutzt werden (z. B. in Microsoft Windows 98, SGI Irix und Sun Solaris). Außerdem muß die bisherige geräteabhängige RGB-Farbspezifikation im VRML-Standard durch die Festlegung eines ICC-Profils neutralisiert werden. Für unspezifische RGB-Farben wird generell das von den Firmen Hewlett-Packard und Microsoft entwickelte sRGB-Profil [71][181] favorisiert; dies wird auch für die Farbspezifikation auf WWW-Seiten vorgesehen [195]. Jedoch entspricht dies im Zusammenhang mit dem Rendering virtueller 3D-Objekte aufgrund der Nichtlinearität nicht dem häufig beabsichtigten linearen Verhalten, d. h. die Mitteltöne werden eventuell zu dunkel wiedergegeben.

Kommunikation

Die Dienstgüte – auch: *Quality of Service (QoS)* – der Kommunikationsverbindung sollte möglichst hoch sein. Eine Festlegung bestimmter Parameter – z. B. Bitrate, Latenz, Jitter, Fehler- und Verlustrate – ist wünschenswert. Allerdings sind zur Gewährleistung derartiger Verbindungsmerkmale bis auf Anwendungsebene sowohl deterministische Netzinfrastrukturen (z. B. Switched Ethernet statt Shared Media) als auch besondere Protokolle zur Reservierung von Ressourcen – gegebenenfalls über mehrere Netze hinweg – wie z. B. RSVP (Resource Reservation Protocol) oder virtuelle Ende-zu-Ende-Verbindungen über ATM-Netze erforderlich. Wegen der in den hier betrachteten Anwendungen extrem hohen erforderlichen Bitraten wurden

jedoch im Rahmen dieser Arbeit im wesentlichen dedizierte lokale Kommunikationsstrecken im „Best-Effort“-Betrieb genutzt.

Repräsentation

Für die Qualität der hier zu representierenden polygonalen 3D-Szenen sind insbesondere die folgenden *Auflösungsparameter* relevant:

- Geometrie: Quantisierung bzw. Genauigkeit der Ortskoordinaten- und Normalenspezifikationen, Komplexität der Topologie,
- Farben, Texturen: Quantisierung bzw. Genauigkeit der Materialspezifikationen sowie der gegebenenfalls angegebenen Texturkoordinaten,
- Lichtquellen: Genauigkeit der Ortskoordinaten und Lichteigenschaften,
- Virtuelle Kameras: Genauigkeit der Spezifikationen von Ort und Richtung,
- Timingspezifikation für zeitabhängige Szenarien.

Da davon auszugehen ist, daß für das Rendering Standardverfahren eingesetzt werden, denen meist eine 32-Bit-Gleitkomma-Arithmetik zugrunde liegt (z. B. OpenGL), wird eine derartige Genauigkeit und Zahlendarstellung für die Speicherung und Datenflüsse als adäquat angesehen.

Die Farbspezifikation stellt ein besonderes Problem dar, da diese – insbesondere für Rasterbilder, die hier als Texturen angewandt werden können – meist auf 8-Bit-Ganzzahlwerten je (unspezifischer) RGBA-Farbkomponente basieren. Hier sollte gefordert werden, daß die gewünschten Farbkomponenten, basierend auf einem *Colormanagement*-Standardverfahren (ICC-Profil), geräteunabhängig spezifiziert werden, um diese auf Präsentationsseite durch Farbraumkonvertierung möglichst hochwertig reproduzieren zu können. Dafür kommt z. B. das TIFF-Format mit eingebettetem ICC-Profil in Betracht.

Kompressionsverfahren sollten hier wegen des zu erwartenden hohen zusätzlichen Rechenaufwandes bzw. Reduktion der Qualität (z. B. durch Quantisierung) zunächst unberücksichtigt bleiben.

3.2.3 Funktionalität

Präsentation

Eine *On-the-fly-Darstellung*, d. h. inkrementelles, dem Übertragungsfortschritt entsprechendes Rendering der Szene, wird in bisherigen VRML-basierten Viewern nicht angeboten. Dies wird unter anderem durch die Anordnung der 3D-Daten im VRML-Format verhindert: Es werden zunächst Objekt-Koordinaten und -Attribute (z. B. Normalenvektoren, Farben, Texturkoordinaten) spezifiziert, und innerhalb der eigentlichen Objekte – z. B. *IndexedFaceSet* – wird durch Indizes darauf referenziert. Prinzipiell kann daher eine Darstellung erst begonnen werden, nachdem alle Koordinaten und Attribute übertragen wurden. In der Praxis wird die Latenz zusätzlich dadurch erhöht, daß erst nach dem Abschluß des Parsevorgangs mit dem Rendering begonnen wird, und zwar durch Traversieren der beim Parsen erzeugten Datenstruktur.

Real-Time-Streaming von dynamischen Szenen, d. h. die Ausspielung von Sequenzen komplett unterschiedlicher Einzelszenen, gewissermaßen als 3D-Film, ist nicht möglich. Diese Einschränkung wird durch fehlende Echtzeit-Merkmale der Viewer, ungenügende Übertragungs- und Bildaufbau-Geschwindigkeiten sowie in diesem Kontext unzureichende Datenstrom- und Steuerungsprotokolle verursacht. Das im Rahmen dieser Arbeit entwickelte System sollte für die Echtzeit-Anforderungen und Animationsfunktionalität sowohl Synchronisierungsverfahren, als auch fortgeschrittene Steuerprotokolle (z. B. Real-Time Streaming Protocol) anwenden. Ausreichende Übertragungs- und Darstellungsraten sollten durch den Einsatz hochratiger Kommunikationsinfrastrukturen und -protokolle sowie durch effiziente Codierung und Renderingroutinen erzielt werden.

Die *stereoskopische Bildausgabe*, ein wesentlicher Bestandteil eines immersiven VR-Systems, soll unterstützt werden.

Dreidimensionale Eingabegeräte, wie z. B. Head-Tracking-, Zeigevorrichtungen oder SpaceBall bzw. SpaceMouse, die 3 Ortskoordinaten und 3 Orientierungswinkel liefern, dienen ebenfalls zur Erhöhung des Immersionsgrades und unterstützen intuitive, komfortable Navigation, können aber durch verfügbare VRML-Plugins nicht genutzt werden. Dies sollte jedoch im hier entwickelten System der Fall sein.

Die Unterstützung einer *breiten Plattform* (verschiedene Rechnerarchitekturen, verschiedene WWW-Browser) erfordert nicht nur eine maschinenunabhängige Spezifikation der 3D-Datenflüsse (z. B. Zahlendarstellung gemäß IEEE-754-Standard), sondern auch einen generischen Software-Architekturansatz (z. B. Nutzung der offenen Plugin-Schnittstelle zur Integration in WWW-Browser sowie standardisierter Kommunikations- und 3D-Graphik-APIs).

Kommunikation

Eine kontinuierliche Online-Präsentation dreidimensionaler Medienströme bedingt den Einsatz eines *Streaming-Protokolls*, welches eine enge Verzahnung von Kommunikation und Präsentation ermöglicht sowie Maßnahmen zur Steuerung des Datenflusses bereitstellt. Dazu zählen zunächst Befehle wie Start, Stop und Positionierung.

Es werden aber auch Mechanismen zur *Synchronisation* von Datenströmen – intra- und inter-stream (siehe *Abschn. 2.1.2.7*) – benötigt, z. B. um eine konstante mittlere Bildrate zu erzielen [159].

Um auch für unterschiedliche Randbedingungen – z. B. Transferrate, Renderingrate – einen korrekten zeitlichen Ablauf zu ermöglichen, sind – je nach Medientyp – verschiedenartige *Skalierungsoptionen* erforderlich. In den bisherigen Video-Streamingsystemen werden dazu z. B. zeitliche, räumliche oder qualitative Kriterien entweder über eine Benutzerschnittstelle fest eingestellt oder adaptiv parametrisiert [28][63]. Für die Ausspielung von 3D-Szenensequenzen kommen z. B. Variationen in der Update-Rate oder der Szenenkomplexität in Frage. Zur Steuerung bzw. Regelung dieser Parameter sind entsprechende Kommunikationsprotokolle erforderlich.

Repräsentation

Die hier anzuwendende immersive Technik für die 3D-Darstellung erfordert nicht nur adäquate Beschreibungen der virtuellen Objekte, Materialien und Lichtquellen, sondern auch eine möglichst vollständige *Attributierung der Sichtbedingungen*. Dazu gehört auch die Angabe einer `focalDistance` (siehe *Abschn. 4.2.2*).

Für die letztlich beabsichtigte Ausspielung von *Sequenzen einzelner 3D-Szenen* müssen geeignete Dateiformate und Speicherungsorganisationen ausgewählt werden.

3.2.4 Standards

Präsentation

Als *Programmierschnittstellen (APIs)* bieten sich sowohl De-jure-Standards/Normen (z. B. ISO) als auch De-facto-/Industrie-Standards an. Normen im Bereich der graphischen Datenverarbeitung [50] haben sich nicht durchgesetzt (CGI: ISO/IEC 9636 [82], GKS: ISO/IEC 7942 [76], GKS-3D: ISO/IEC 8805 [78], PHIGS: ISO/IEC 9592 [80]). Statt dessen wird für die professionelle 3D-Graphikprogrammierung üblicherweise die offene Schnittstelle OpenGL [30][95][121][122][198][202] in C- bzw. C++-Programmen verwendet (siehe auch `www.opengl.org`). Die von der Firma Microsoft spezifizierte Direct-3D-Schnittstelle [114][163] wird zwar häufig für interaktive 3D-Graphik auf PCs angewandt (z. B. Spiele), steht aber nur auf Windows-Systemen zur Verfügung. Optimizer, OpenInventor und Performer sind objektorientierte C++-Klassenbibliotheken, die auf OpenGL aufsetzen und als Szenengraph-API dienen können. Diese werden jedoch auf vielen Plattformen nicht unterstützt. Für die Java-Programmierung wurde von der Firma Sun „Java 3D“ spezifiziert [183]. Zur C-Programmierung von Nutzeroberflächen werden auf UNIX-Systemen X-Windows- sowie darauf aufsetzende OSF/Motif-Bibliotheken [61] [141], auf Microsoft-Windows-Systemen die Win32-Bibliothek [172] sowie auch mehrere, z. T. generische, objektorientierte Bibliotheken (z. B. für C++ oder Java) angeboten.

Browser-Plugins werden über eine von Netscape offengelegte Callback-Schnittstelle [125][139] implementiert, die von den derzeit verbreiteten WWW-Browsern (Netscape Communicator, Microsoft Internet Explorer) unterstützt wird.

Der für *stereoskopische Präsentation* geeignete Quadbuffer-Modus [178], der in OpenGL standardisiert programmiert werden kann und zur Ansteuerung von LCD-Shutterbrillen dient, wird zunehmend auch von PC-Graphikadaptern angeboten. Im Bereich der UNIX-Systeme ist die Stereo-Unterstützung schon längere Zeit Stand der Technik. Beispielsweise bieten SGI-Workstations generell diese Möglichkeit, aber auch bestimmte Hochleistungs-Graphiksubsysteme der Firmen HP (z. B. fx6) und Sun (z. B. Elite 3D). Mehr Aufwand erfordert die Multi-Graphikpipe-Technik, bei der für jedes Bild ein eigenes Renderingsystem (Graphikkarte) verwendet wird. Dabei kann durch die Parallelisierung des Renderingvorgangs eine Erhöhung der Bildrate bzw. Reduktion der Navigationslatenz erzielt werden.

Kommunikation

Im Bereich der *Netz-Infrastruktur* ist der Fortschritt an verfügbarer Bitrate bemerkenswert. Für den Einsatz in lokalen Netzen stellt die aktuelle Ethernet-Technologie bis zu 1 Gbit/s bereit. In Weitverkehrsnetzen werden zunehmend optische Netze mit WDM-Technik (Wave Division Multiplex) verwendet, die für Backbone-Infrastrukturen Datenraten bis in den Tbit/s-Bereich je Faser bieten.

Als *Kommunikationsprotokolle* kommen insbesondere die Internet-Protokolle in Betracht, d. h. z. B. TCP/IP für verbindungsorientierte und UDP/IP für datagrammorientierte Dienste. Darauf aufsetzende Protokolle sind z. B. HTTP (Hypertext Transfer Protocol, RFC 2068) [45], RTSP (Real Time Streaming Protocol, RFC 2326) [165] über TCP/IP für die Streamingsteuerung oder RTP (Transport Protocol for Real-Time Applications, RFC 1889) [164] über UDP/IP für kontinuierliche Medien, wahlweise auch in Multicast-Szenarien.

Für die *Anwendungsprogrammierung* stehen auf allen Plattformen Socket-Bibliotheken zur Verfügung, die z. B. für die Kommunikation über TCP oder UDP verwendet werden können [156][180].

Repräsentation

Für den Abruf von Online-Dokumenten, im wesentlichen in Form von diskreten Medien, werden üblicherweise *WWW-Server* eingesetzt. Sollen multimediale Dokumente zum Abrufzeitpunkt dynamisch erzeugt werden, so kann dafür serverseitig eine CGI-Schnittstelle (Common Gateway Interface) verwendet werden. Im Zusammenhang mit Upload-Mechanismen [119] ist damit z. B. auch die Konvertierung von Client-Daten möglich.

Cache- / Proxy-Techniken dienen zur Reduktion des Datenverkehrs durch Zwischenspeicherung von Datenobjekten auf Servern, die aus Netzsicht näher am Client liegen als der eigentlich abgerufene Server.

Als grundlegendes *Standard-Dateiformat* für die hier beabsichtigte Anwendung kommt besonders VRML – z. B. VRML97 [86] – in Betracht. Eine Übersicht über VRML-bezogene Spezifikationen und Werkzeuge bietet der WWW-Server des Web3D-Konsortiums (siehe auch `www.web3d.org`) sowie das am San Diego Supercomputer Center (SDSC) entwickelte VRML-Repository [199]. Dieses Dateiformat eignet sich zwar aufgrund verschiedener Unzulänglichkeiten nicht direkt für die performante 3D-Transportschnittstelle, aber ein neu entworfenes Dateiformat sollte für eine Interoperabilität zumindest davon ableitbar sein.

4 Design und Implementierung

4.1 Systementwurf

Eine Übersicht über die wesentlichen Gesichtspunkte zur Systemauslegung wurde in *Tabelle 4* gegeben und in *Abschn. 3.2.1* bis *Abschn. 3.2.3* erläutert. Unter weiterer Berücksichtigung existierender Standards (siehe auch *Tabelle 5, S. 51* sowie *Abschn. 3.2.4*) – waren für das Systemdesign einige Entscheidungen zu treffen, die aufgrund der in den aufgestellten Forderungen vorliegenden Gegenläufigkeiten unter Umständen Kompromisse auf der Basis von sorgfältigen Abwägungen darstellten. Die im Rahmen dieser Arbeit durchgeführten Entwicklungen basieren auf dem Zusammenspiel der in *Abb. 8* als System-Modell graphisch dargestellten Komponenten, Protokolle, Dienste und Formate. Damit sollten für den Nutzer des WWW-Clienten leistungsmäßige, qualitative und funktionale Verbesserungen bei der Online-Präsentation virtueller 3D-Objekte auf der Basis geometrischer Beschreibungen erzielt werden.

4.1.1 Leistungsfähige Online-Präsentation statischer 3D-Szenen

Anhand der *Abb. 19* bis *Abb. 21* sollen die vollzogenen Entwicklungsschritte zur Hochleistungs-Online-Präsentation von statischen 3D-Szenen veranschaulicht werden:

1. Ursprünglich stellten VR-Systeme Standalone-Systeme dar. Die jeweils zu bearbeitenden Szenarien – bestehend aus den geometrie-orientierten 3D-Modelldaten sowie entsprechende Methoden – wurden zunächst aus einem Dateisystem eingelesen. Nach einer Startup-Phase, die wegen der aufwendigen Vorbereitungsschritte – z. B. Parsen, Aufbau eines Szenengraphen – relativ lange dauern konnte, wurde letztlich auf der Basis von im Speicher liegenden, optimierten Datenstrukturen gearbeitet (*Abb. 19*). Als Beispiel ist das am Fraunhofer-Institut für Graphische Datenverarbeitung entwickelte Virtual-Design-System zu nennen [5][6], welches im Rahmen einer vom Autor betreuten Studie inzwischen für CSCW-Anwendungen im VW-Konzern erweitert wurde [118].

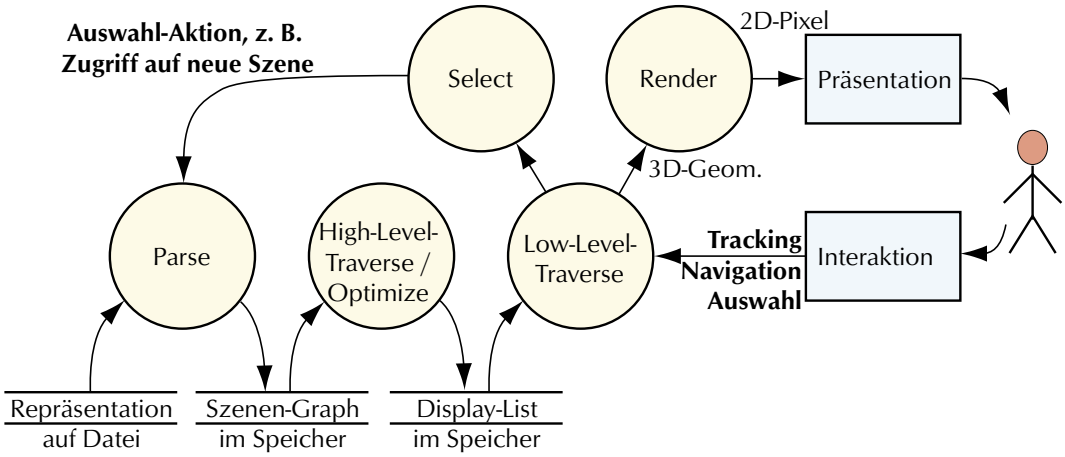


Abb. 19. Standalone-VR-System – Lokale Arbeitsweise

2. Nachdem Kommunikationsaspekte – z. B. Multimedia-/Virtual-Reality-Daten auf verteilten Servern, Zugriff von verteilten Client-Systemen, kooperative Nutzungsformen – eine zunehmende Rolle spielten, wurden derartige Systeme für die Anwendung im WWW-Kontext angepaßt. Nach der Standardisierung eines dafür zu verwendenden Dateiformats „VRML“ konnte ein gemäß *Abb. 19* realisiertes Viewer-System somit unmittelbar als „Helper Application“ konfiguriert und aufgerufen werden. Um auch WWW-Links zu unterstützen, die von Teilen einer 3D-Szenen ausgehen, wurden entsprechende VRML-Sprachelemente (`WWWAnchor`, siehe auch *Abschn. 2.2.2*) zur Aktivierung weiterer WWW-Dokumente genutzt. Externe VRML-Viewer rufen dazu üblicherweise den WWW-Browser mit einer speziellen Option auf, die ihn veranlaßt, die gewünschte URL zu laden (*Abb. 20*). Beispielsweise arbeitet der VRML-1.0-Viewer Webspace der Firma Silicon Graphics nach diesem Verfahren.

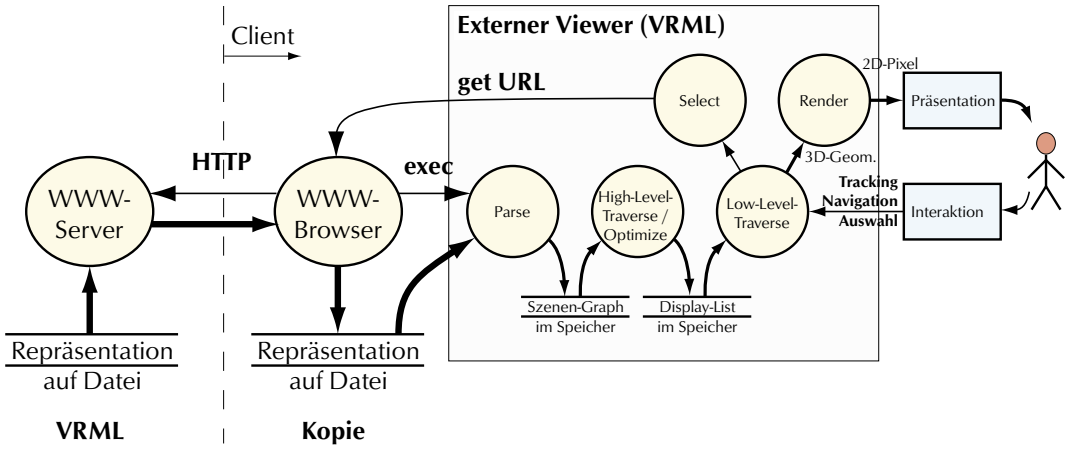


Abb. 20. Externer VRML-Viewer – „Helper Application“ für WWW-Browser im verteilten System: Zugriff auf entfernte Dokumente

3. Eine engere Integration des Viewers in den WWW-Browser wurde durch die Plugin-Architektur ermöglicht, die von der Firma Netscape für alle signifikanten Plattformen spezifiziert wurde [124][125] (siehe auch *Abb. 8, S. 25* in *Abschn. 2.2.1*) und auch vom Microsoft-Browser Internet Explorer unterstützt wird. Die gebräuchlichen VRML-Viewer, wie z. B. CosmoPlayer, liegen heute üblicherweise als Plugin-Software vor. Die potentiellen Leistungsengpässe, die durch Programmaufrufe sowie durch Schreiben und Lesen von Zwischendateien entstehen, werden hier dadurch vermieden, daß die Plugin-Software als Shared Library zur Laufzeit Bestandteil des Browser-Prozesses wird und mittels Callback-Routinen bidirektionale Schnittstellen bereitgestellt werden. Für die im Rahmen dieser Arbeit durchgeführte Viewer-Implementierung wurde daher diese Architektur ausgewählt (*Abb. 21*). Darüber hinaus wurde ein neues Dateiformat, genannt „DVR“, definiert, um die aufwendigen Vorverarbeitungsschritte („Parse“, „High-Level-Traverse“, „Optimize“ in *Abb. 20*) zu umgehen. Das DVR-Format ist prinzipiell auf dem Niveau der „Low-Level-Display-List“ in *Abb. 20* angesiedelt, d. h. die 3D-Rendering-Routinen (hier: OpenGL) können unmittelbar auf diese Datenstrukturen zugreifen.

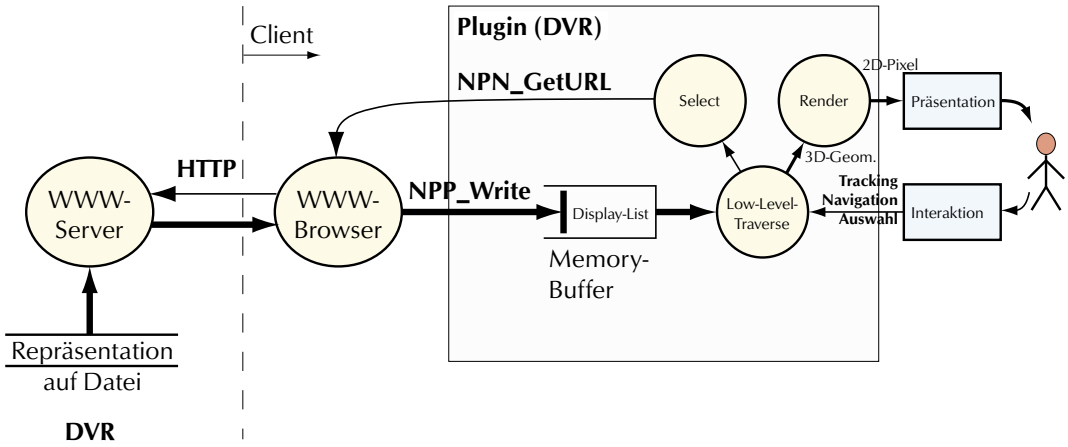


Abb. 21. Inline-Plugin für das DVR-Format – integriert in WWW-Browser

4.1.2 Eine Architektur zur Ausspielung von 3D-Szenensequenzen

Durch die Plugin-Architektur sowie die auf Effizienz ausgelegte Spezifikation des DVR-Formats und Implementierung des DVR-Viewers konnten die Updates von Szenen der geforderten Komplexität in relativ kurzer Zeit durchgeführt werden, d. h. in der Größenordnung von einer Sekunde. Dadurch wurde die Realisierung eines Streamingverfahrens ermöglicht, welches die Ausspielung von Szenen-Sequenzen ermöglicht, die, vergleichbar mit Audio-/Video-Streaming (z. B. RealMedia), auf einem separaten Streaming-Server bereitgestellt werden. Während der Abspielung eines solchen „3D-Films“ kann sich der Nutzer weiterhin frei in der virtuellen 3D-Szene bewegen, d. h. mittels Tracking- oder 3D-Interaktionsgerät kann interaktiv navigiert werden.

Da es bislang keinen flexiblen Mechanismus zur Erweiterung von WWW-Browsern um neue Protokolle gibt, wird dabei zunächst über HTTP ein kleiner Metadatensatz, genannt „DVRS“, geladen, der aufgrund einer separaten MIME-Kennzeichnung eine spezielle Plugin-Variante aktiviert. Die DVRS-Daten enthalten z. B. die Referenz auf die eigentlichen 3D-Daten, wodurch letztlich ein im DVRS-Plugin enthaltener Streaming-Client initiiert wird (*Abb. 22*). Dieser kommuniziert direkt mit dem Streaming-Server, wobei als Steuerprotokoll RTSP (Real Time Streaming Protocol, RFC 2326 [165]) und als 3D-Datentransportprotokoll das DVR-Format mit eingefügten Szenen-Trenndatenelementen (genannt „DVRP“), beide über TCP/IP, verwendet werden.

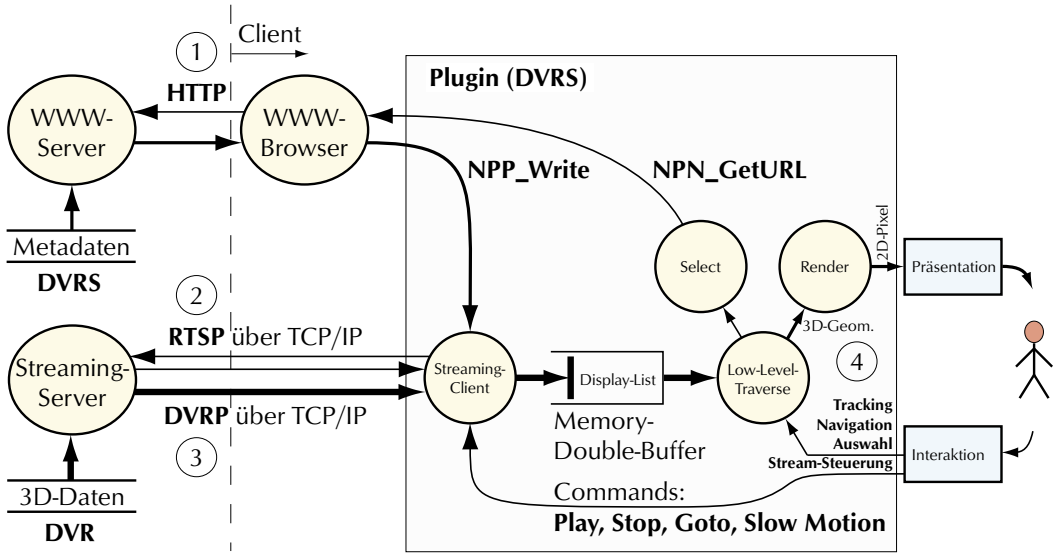


Abb. 22. Inline-Plugin für das DVRS-Format – 3D-Streaming-System

Der in *Abb. 22* dargestellte Ablauf ist durch die folgenden Schritte gekennzeichnet:

1. Der WWW-Client liest ein DVRS-Objekt (MIME-Typ: `application/x-doc-show-vrs`, Datei-Endung: `.dvrs`) über HTTP ein. Dieses wurde vom Browser z. B. durch ein `EMBED`-Tag angefordert, das in einer HTML-Webseite eingebettet ist.
2. Nachdem das dazu passende DVRS-Plugin (Erweiterung des DVR-Plugins, siehe auch *Abschn. 4.2.2*) in den WWW-Browser geladen wurde, baut dieses eine Verbindung zum 3D-Streaming-Server auf. Die dazu erforderlichen Startup-Informationen sind im DVRS-Datensatz spezifiziert, der dem Plugin im Callback `NPP_Write()` bereitgestellt wird. Zu den darin gespeicherten Metadaten gehören:
   - Referenz auf den 3D-Streaming-Server (Protokoll, IP-Adresse, Port-Nummer),
   - Ort der 3D-Daten (Pfad, Namenstemplate) und
   - Darstellungsattribute (z. B. Frame-Rate, Level of Detail).

3. Der 3D-Streaming-Server (siehe *Abschn. 4.2.3*) liest die 3D-Szenen aus Dateien im DVR-Format (siehe *Abschn. 4.2.1*) und liefert diese an den Client aus, wobei jeweils noch Szenen-Endmarkierungen eingefügt werden. Der Client liest diese in einen Wechsel-Puffer ein, um asynchron die folgenden Aktivitäten zu ermöglichen:

4. Rendering, Navigation, Auswahl und Stream-Steuerung:

   (a) Rendering:
   Sowohl durch das Eintreffen einer Szenen-Endmarkierung als auch durch Änderung der virtuellen Kameraperspektive wird eine Aktualisierung der Display-Ausgabe aktiviert.

   (b) Navigation:
   Entsprechend der Steuerung durch die Eingabegeräte, wie z. B. Maus, SpaceBall oder Headtrackingsystem, werden die Parameter der virtuellen Kamera modifiziert: Position, Orientierung und Blickwinkel. Kamerafahrten können als „Views“ bzw. „Cameras“ in VRML- bzw. DVR-Dateien auch vordefiniert werden, um dann wahlweise automatisch abzulaufen.

   (c) Auswahl:
   Eine Aktivierung von Hyperlinks in 3D-Szenen ist durch Anklicken der entsprechend attributierten Objekte im „Pick“-Modus des Plugins bzw. durch „Doppelklick“ möglich. Liefert die „Select“-Traversierung eine URL, so wird diese vom WWW-Browser mittels `NPN_GetURL()` angefordert.

   (d) Stream-Steuerung:
   Befehle, die die Streamingsteuerung betreffen, können vom Benutzer über einen fensterorientierten Dialog, ähnlich der Wiedergabesteuerung eines Videorecorders, gegeben werden. Dazu gehören: Anhalten, Positionierung, Ausspielung mit geänderter Framerate, Wahl eines alternativen Streams mit reduzierter Komplexität. Die Steuerung erfolgt über das RTSP-Protokoll.

4.1.3 Interoperabilität mit VRML-Generatoren

Um eine Kompatibilität zum VRML-Standard zu erzielen, wurden mehrere Mechanismen vorgeschlagen und realisiert [133]:

1. Eine Konvertierung mittels *batchorientiertem Kommandozeilen-Programm* (*Abschn. 4.2.1*).

2. Ein *interaktiver Online-Konvertierdienst* über eine Webseite mittels Upload einer VRML-Datei über ein WWW-Formular [119], Konvertierung in einem CGI-Programm auf einem WWW-Server und Auslieferung der Ergebnisdatei über HTTP; dabei erfolgt wahlweise eine Abspeicherung in Datei oder die direkte Anzeige im Plugin.

3. Ein *transparenter Konvertiervorgang* (siehe *Abb. 23)*, der als konvertierender Caching-Proxy realisiert ist und dazu die Proxy-Cache-Software `squid`, einen konfigurierbaren HTTP-Filter – `webfilt` [194] – sowie die oben genannten Konvertierprogramme anwendet. In einer Machbarkeitsstudie wurde auf Client-Seite in der „Proxy Automatic Configuration“ (PAC), einer flexiblen Spezifikation der zu nutzenden Proxy-Server, festgelegt, daß mit `.wrl` endende URLs über einen speziellen Konvertier-Proxy verarbeitet werden. Dieser holt die gewünschte VRML-Datei auf übliche Weise aus dem WWW, konvertiert sie jedoch in das DVR-Format und liefert diese dem Client mit dem passenden MIME-Type, so daß das 3D-Modell letztlich im DVR-Plugin angezeigt wird. Beim zweiten Abruf dieser URL kann die DVR-Datei direkt aus dem Cache ausgeliefert werden.

   Aus dieser Vorgehensweise ließe sich prinzipiell ein generelles *Konzept für universelle Multimedia-Konvertierungsdienste* entwickeln, welches zur Realisierung automatisiert ablaufender Ableitungsketten dienen könnte. Dieses wäre dann z. B. auch direkt auf WWW-Serverseite einsetzbar.

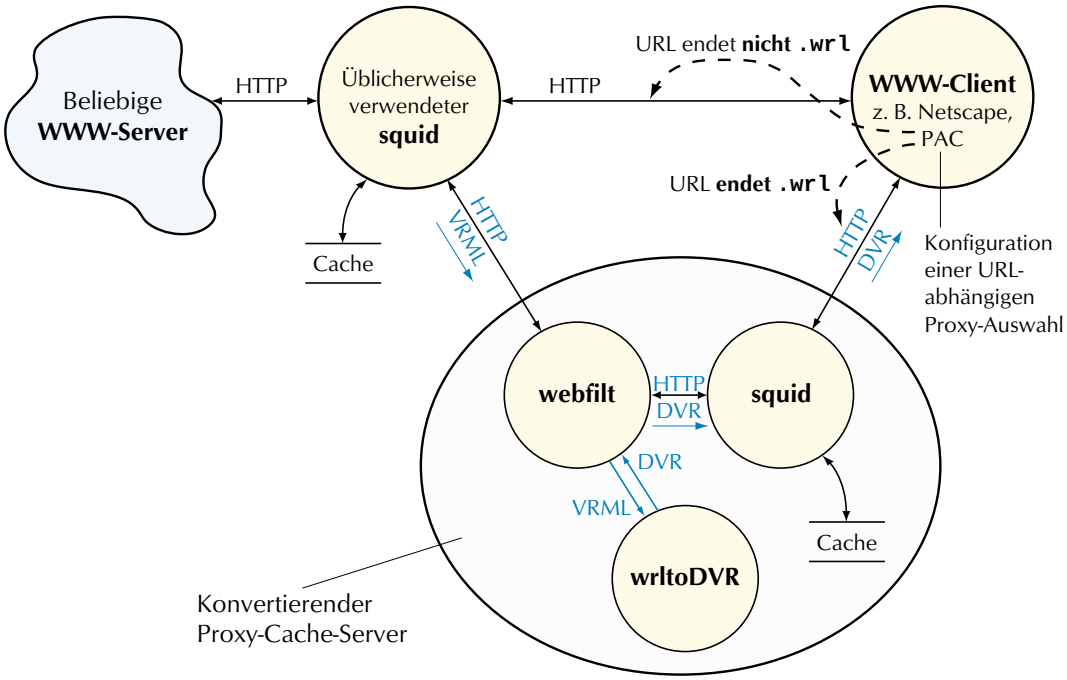


Abb. 23. Konvertierende Proxy-Cache-Server-Konfiguration

4.2 Implementierungsaspekte

Aus den vorgenannten Gründen wurden ein eigenes Format (*DVR*) einschließlich Generierungssoftware, ein leistungsfähiger Viewer (*DocShow-VR*) sowie ein Streamingserver (und zugehöriges Metadatenformat *DVRS*) entwickelt. Die bereits in *Abschn. 3.2* erörterten Anforderungen führten, unter Berücksichtigung der in *Abschn. 3.2.4* genannten relevanten Standards, zu einer adäquat ausgelegten, prototypischen Implementierung dieser Komponenten.

4.2.1 Einlesen von VRML, Vorverarbeitung, Aufbau und Generierung des 3D-Formats „DVR“

Es wurden zwei Programme implementiert, die VRML-Dateien der beiden derzeit relevanten VRML-Standards in das eigene DVR-Format konvertieren:

- `wrl1toDVR` konvertiert VRML-1.0-Dateien in DVR-Dateien,
- `wrl2toDVR` konvertiert VRML97-Dateien in DVR-Dateien.

Die Ausgangsbasis für die Implementierung des Konverters bildeten zunächst die frei verfügbare C++-Klassenbibliothek *QvLib* [182] zum Parsen von VRML 1.0, ein OpenGL-Programmfragment zum Traversieren und Rendern auf Basis von *QvLib* aus einer Email von Jan Hardenberg [58] und Informationen aus den Quellen des frei verfügbaren *read_vrml*-Moduls für die Visualisierungssoftware AVS.

Inzwischen wurde VRML 2.0 als ISO-Standard VRML97 [86] verabschiedet. Das hat dazu geführt, daß sich die Export-Möglichkeiten verschiedener Generierungswerkzeuge von VRML 1.0 nach 2.0 entwickelt haben und in neueren Versionen zum Teil keine Option für das VRML-1.0-Format mehr besteht.

Beispielhaft sind die folgenden Werkzeuge zu nennen:

- 3D-Modelliersoftware: CosmoWorlds
- Visualisierungssoftware: Ensight, VTK (Visualization Toolkit) [161]

Daher wurde eine Lösung erarbeitet, die auch eine Konvertierung aus dem VRML-2.0-Dateiformat gestattet. Eine Marktübersicht ergab, daß mehrere Toolkits angeboten werden, die zur offenen Unterstützung des Parsens von VRML-2.0-Dateien sowie zum Aufbau von Szenengraphen dienen können[1]. Zur Stabilität, Effizienz und strategischen Bedeutung der Produkte waren allerdings nur wenige Informationen verfügbar. Zum Teil handelte es sich um noch in Entwicklung befindliche Werkzeuge, bei denen auch die Abschätzung von Terminen zur Fertigstellung wesentlicher Funktionalitäten zu Schwierigkeiten führen könnte. Daher wurde die von der Firma TGS angebotene und um VRML-2.0-Ein- und Ausgabe-Funktionen („VRML-Master“) erweiterte Standard-Graphikbibliothek OpenInventor 2.5

[1] z. B.: Activate – VRML2 Toolkit for C/C++ Developers (`www.3dweb.com`), OpenWorlds – VRML 2.0 C++ Library (`www.openworlds.com`),VRaniML – VRML 2.0 C++ Class Library für Windows 95/NT (`www.greathill.com/vraniml/`), VRML-Master – VRML97 (2.0) Extensions für die Open Inventor C++ Library (`www.tgs.com/3DMS/index-c++.html`).

zunächst als befristete Demoversion getestet. Nachdem die Machbarkeit auf dieser Basis gezeigt war, wurden Entwicklungslizenzen für SGI Irix und Microsoft Windows 95/NT beschafft und die Software installiert. Durch den Einsatz dieses Standardprodukts war es auch möglich, anderweitig vorliegende Hilfsbibliotheken, z. B. zur Berechnung von Normalenvektoren, auf relativ einfache Weise in den Konvertierprozeß einzubinden[1].

Ein besonderes Problem stellte sich allerdings dadurch heraus, daß in den „Viewpoints“ (in VRML 1.0: `PerspectiveCamera`) keine `focalDistance` spezifiziert wird. Diese Angabe ist jedoch erforderlich, um eine korrekte stereoskopische Darstellung zu produzieren und einen Anhaltspunkt für die Near- und Far-Clipping-Ebenen zu bieten (siehe auch *Abschn. 4.2.2*). Daher wurde diese Angabe als weitere Kommandozeilen-Option vorgesehen. Auf die Berechnung eines Schätzwertes wurde verzichtet.

Zusätzlich wurde eine Option zum Aufsplitten von Teilszenen implementiert, die in einer „Switch“-Gruppe angeordnet sind. Dies war erforderlich, um die Visualisierung zeitabhängiger Phänomene mit Hilfe der Visualisierungssoftware Ensight zu unterstützen. Hier werden die Ergebnisse sämtlicher Zeitschritte in einer VRML-Datei abgelegt, die üblicherweise in einem VRML-Browser nacheinander „durchgeklickt“ werden können. Der im Rahmen dieser Arbeit realisierte Streaming-Mechanismus, dem Sequenzen von einzelnen Szenen im DVR-Format zugrunde liegen, stellt jedoch eine bessere Lösung des Problems der Präsentation von Zeitserien dar.

Aufbau des DVR-Formats

Das DVR-Format wurde anhand folgender Gesichtspunkte entworfen:

Decodierung mit geringstmöglichem Rechenaufwand und Bereitstellung für Immediate-Mode-OpenGL-Rendering

Das DVR-Format verwendet zur Zahlendarstellung das Binärformat gemäß IEEE 754 in Network-Byte-Ordering, d. h. entsprechend der rechnerinternen Zahlendarstellung der meisten UNIX-basierten Arbeitsplatzrechner. Ausnahmen sind Intel- und DEC-CPUs, auf welchen die Client-Software einmal nach der Übertragung eine Konvertierung durchführen muß. Dies wird durch das im Socket-API zur Netzwerkprogrammierung enthaltene Makro `ntohl()` erledigt, das auf 32-Bit-Integer- und – obwohl eher unüblich und nicht allgemein bekannt – auch auf -Float-Werte im 32-Bit-IEEE-Format angewandt werden kann. Exotischere Zahlendarstellungen sind inzwischen, wenn überhaupt, nur noch auf speziellen Hochleistungsrechnern anzutreffen, die aber als Client-Plattform hier ohnehin nicht in Frage kommen.

[1] Aufgrund der inzwischen gestarteten „VRML Open Source“-Initiative (`www.openvrml.org`) könnte für weitere Entwicklungen als Frontend die frei verfügbare, portable C++-Klassenbibliothek *LibVRML97* in Betracht gezogen werden (`www.vermontel.net/~cmorley/vrml.html`).

In VRML-Teilobjekten ohne explizite Normalenvektor-Spezifikationen werden bereits bei der Konvertierung die Normalenvektoren berechnet und abgespeichert. Besonders für geglättet darzustellende große polygonale Netze, in denen über die Spezifikation eines Grenzwinkels (VRML: `creaseAngle`) differenziert wird, ob über eine Kante zu interpolieren ist, können so erhebliche Rechenaufwände auf Client-Seite vermieden werden, wenn diese Vorberechnung auf Server-Seite bereits erfolgt ist.

Kürzestmöglicher Startup

Es werden direkte Koordinaten- und Attribut-Werte (im Gegensatz zu den indizierten Werten in VRML) in den DVR-Records verwendet, um unmittelbar mit dem Rendern beginnen zu können.

Unterstützung effizienten Renderings in OpenGL, d. h. Verwendung von Triangle-Strip- und Triangle- neben den allgemeineren Polygon-Primitiven

In einem Vorverarbeitungsschritt werden die im VRML-Format vorliegenden aufeinander folgenden unabhängigen Polygone (IndexedFaceSet) analysiert, Dreiecke miteinander verglichen und gegebenenfalls in die Triangle-Strip- oder Triangle-Primitive des DVR-Formats umgewandelt.

Optimierung der OpenGL-Statuswechsel

Bei der DVR-Generierung werden nur die notwendigen Statusänderungen (z. B. Materialeigenschaften, Transformationen) in entsprechenden Records ausgegeben.

Eine DVR-Datei besteht aus einer Sequenz von PDUs (Protocol Data Unit), die jeweils aus einen Header mit fester Länge (64 bit) und einem Datensatz variabler Länge (Vielfaches von 32 bit) bestehen (siehe auch *Abb. 24*).

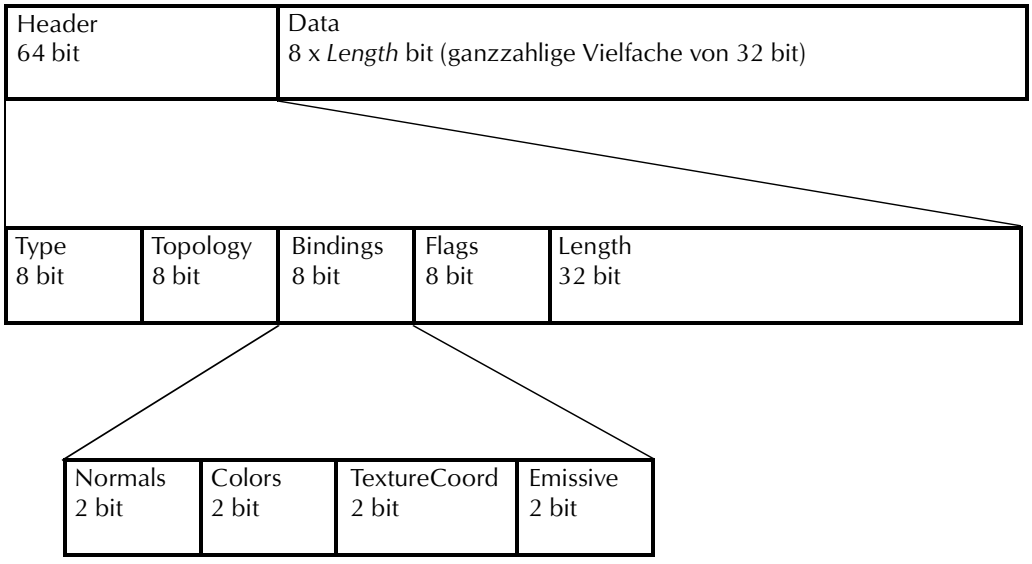


Abb. 24. Aufbau einer PDU im DVR-Format

Die Bedeutung der Felder in den DVR-PDUs ist wie folgt (siehe auch *Tabelle 6*):

1. Type (8 bit) – Funktion der PDU
   - Generator-Informationen („Header“: Version, Datum, Uhrzeit)
   - Graphik-Elemente („AsciiText“, „Cone“, „Cube“, „Cylinder“, „FaceSet“, „LineSet“, „PointSet“, „Sphere“)
   - Graphik-Attribute („Material“)
   - Text-Elemente („Info“: für AsciiText, Hyperlinks, Textur-URLs, Kameraname)
   - Transformationen („ProjectionMatrix“, „Texture2“, „Texture2Matrix“)
   - Virtuelle Kameras („Camera“)
   - Virtuelle Lichtquellen („Light“, „MaxLight“)
   - Steuerung („End“)
2. Topology (8 bit) – zur Unterscheidung verschiedener Topologie-Klassen in den Graphikelemente-PDUs FaceSet, LineSet und PointSet bzw. diverse Parameter in anderen PDUs.
3. Bindings (8 bit) – zur Spezifikation der im Graphikelement-Datensatz optional enthaltenen Attribute: Normalenvektoren, Farben, Texturkoordinaten, jeweils unterschieden nach
   - globaler Attributierung: 0
   - Attributspezifikation je Eckpunkt: 1
   - Attributspezifikation je Polygon: 2

   Zusätzlich wird unterschieden, welche Materialeigenschaft durch die Farbspezifikation beeinflußt werden soll:
   - Emissive = 0: diffuseColor
   - Emissive = 1: emissiveColor
4. Flags (8 bit) – für weitere PDU-spezifische Optionen, z. B.
   - Bit 7 = 1: Byte-Ordering des Datensatzes ist nicht systemspezifisch (Text),
     Bit 7 = 0: Datensatz enthält 32-bit-Wörter, die gegebenenfalls
     vor der ersten Benutzung mit `ntohl()` konvertiert werden müssen.
5. Length (32 bit) – Länge des an den Header folgenden Datensatzes, in Network-Byte-Order.
6. Data (8 x *Length* bit) – Parameter des jeweiligen Elements, abhängig von der Spezifikation im Header, ergänzt auf ganzzahliges Vielfaches einer Länge von 32 bit.

Type	Bedeutung	Topology	Inhalt des Data-Feldes
0	Header	„D“	2 int-Werte: version=2, timestamp
1	ASCII-Text	–	float- und int-Werte: spacing, justification, width, size, family, style
2	Cone	bit 0: Sides	2 floats: bottomRadius, height
		bit 1: Bottom	
3	Cube	–	3 floats: width, height, depth
4	Cylinder	bit 0: Sides	2 floats: radius, height
		bit 1: Top	
		bit 2: Bottom	
5	FaceSet	1: polygon	Koordinaten, Normalen, Farben (optional), Texturkoordinaten (optional)
		2: triangles	
		3: triangle strip	
		4: quad	
		5: quad strip	
6	LineSet		ähnlich FaceSet
7	PointSet		ähnlich FaceSet
8	Sphere	–	Radius; optional: *n* Mittelpunkte
9	Info	0: Attribute	2 strings, z.B. „BackgroundColor\01 1 1“
		1: CameraName	string
		2: TextString	
		3: TextureURL	
		4: AnchorURL	
10	Material	–	17 float-Werte: ambientColor, diffuseColor, specularColor, emissiveColor, shininess
11	ProjectionMatrix	–	4x4 matrix (16 floats)
12	Texture2	–	–
13	Texture2Matrix	–	3x3 matrix (9 floats)
14	Bindings=0: OrthogonalCamera Bindings=1: PerspectiveCamera	number	9 float-Werte: position (3), orientation (4), focalDistance (1), heightAngle (1)
15	Light	1: directional	18 float-Werte: diffuse (4), specular (4), position (4), spotDirection (4), linearAttenuation (1), spotCutoff (1)
		2: point	
		3: spot	
16	MaxLight	number	–
17	End of Scene	–	–

Tabelle 6. PDU-Typen im DVR-Format

Zur Illustration seien in *Abb. 25* und *Tabelle 7* VRML- und DVR-Formate am Beispiel des in *Abb. 12* dargestellten Triangle-Strips gegenübergestellt.

```
#VRML V1.0 ascii

DEF Default PerspectiveCamera {
  position 0 0 3
  focalDistance 3
}

NormalBinding { value PER_VERTEX }

Coordinate3 {
  point [   -1 0.95 0, -0.5 -1 0, 0 1 0,
            0.5 -1.2 0, 1 1.1 0 ]
}
Normal {
  vector [  0 0 1, 0 0 1, 0 0 1, 0 0 1,
            0 0 1 ]
}
IndexedFaceSet {
  coordIndex [ 0, 1, 2, -1, 2, 1, 3, -1,
               2, 3, 4, -1 ]
}
```

(a) Dateiformat: VRML 1.0

```
#VRML V2.0 utf8

DEF Default Viewpoint {
  position 0 0 3
}


Shape {
  geometry IndexedFaceSet {
    coord Coordinate {
      point [ -1 0.95 0, -0.5 -1 0,
              0 1 0, 0.5 -1.2 0, 1 1.1 0 ]
    }
    normal Normal {
      vector [0 0 1, 0 0 1, 0 0 1,
              0 0 1, 0 0 1 ]
    }
    coordIndex [0, 1, 2, -1, 2, 1, 3, -1,
                2, 3, 4, -1 ]
  }
}
```

(b) Dateiformat: VRML 97

Abb. 25. VRML-1.0- und VRML-97-Dateien eines „Triangle-Strips“ gemäß *Abb. 12, S. 31*, bestehend aus drei zusammenhängenden Dreiecken mit Normalenspezifikation je Eckpunkt

Lfd. Nr.	Header					Data
	Type	Topology	Bindings	Flags	Length	
1	0	„D“	„V“	„R“	8	2, timestamp
2	9	1	0	0x80	8	„Default“ (Kamera-Name)
3	14	1	1	0	36	0.0, 0.0, 3.0, 0.0, 0.0, 1.0, 0.0, 3.0, $\pi/4$
4	5	3	1	0	120	0.0, 0.0, 1.0, -1.0, 0.95, 0.0, 0.0, 0.0, 1.0, -0.5, -1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.5, -1.2, 0.0, 0.0, 0.0, 1.0, 1.0, 1.1, 0.0

Tabelle 7. DVR-Format des „Triangle-Strips“ aus *Abb. 12*, generiert aus VRML (siehe *Abb. 25*)

Generierung des DVR-Formats

Als Programmierschnittstelle zur Generierung von DVR-Dateien oder -Datenströmen wurde in C++ eine Klassenbibliothek „OutputDVR“ entwickelt. Diese dient in den Konvertierwerkzeugen `wrl1toDVR` und `wrl2toDVR` zur DVR-Ausgabe. Darüber hinaus können hiermit auch direkt DVR-PDUs erzeugt werden; z. B. wurde auf dieser Basis zum Visualization Toolkit (VTK) eine Klasse `vtkDVRExporter`, in Anlehnung an die bereits vorhandene vtkVRMLExporter-Klasse, hinzugefügt.

4.2.2 Ein Hochleistungs-Plugin als DVR-Viewer: „DocShow-VR“

Zur Plugin-Implementierung dienten zunächst einfache Beispiel-Plugins aus dem Plugin-Developmentkit von Netscape [124]. Für das 3D-Rendering wurde die Industriestandard-3D-Graphikprogrammierschnittstelle OpenGL [122] verwendet, da diese für sämtliche Plattformen verfügbar ist. Vorteile in der Verwendung höherer Schnittstellen wie OpenInventor, Performer oder Optimizer, die auf Szenengraphen basieren, waren nicht erkennbar, weil Szenengraphen nicht benötigt wurden. Außerdem werden diese APIs nur auf besonderen Plattformen unterstützt. Als OpenGL-Ersatz wurde die frei erhältliche Mesa-Library [143] auf Plattformen verwendet, auf denen keine OpenGL-Laufzeitumgebung installiert ist. Zur Vereinfachung des Installationsvorgangs wurden diese speziellen Plugin-Versionen mit der Mesa-Bibliothek statisch gelinkt.

Die GUI-Moduln, die zur Realisierung von Popup-Menüs, Dialogen und zum Eventhandling (Timer, Maus, Windows) dienen, wurden plattformabhängig implementiert. Um weitere Software-Installation und Lizenzierung zu umgehen und eine schlankere Realisierung zu ermöglichen, wurden zwei Varianten entwickelt: unter UNIX wurde auf X-Windows (Xlib) und OSF/Motif aufgesetzt und die OpenGL-Integration über die GLX-Bibliothek vorgenommen, für Microsoft Windows 95/98/NT wurden die Win32- und WGL-Bibliotheken verwendet (*Abb. 26*).

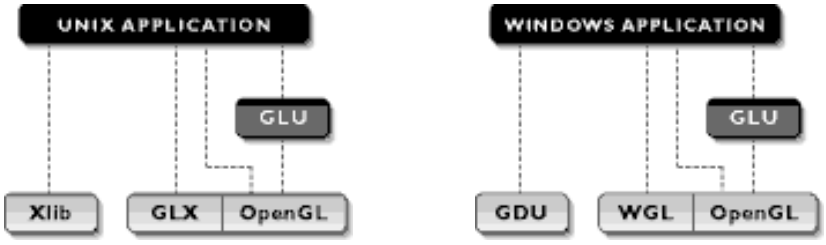


Abb. 26. Aufruf-Hierarchie von OpenGL-Anwendungen unter UNIX und Microsoft Windows.
GLU: OpenGL Utility Library (z. B. High-Level-Primitive, NURBS),
GDU: Graphics Device Utility Library (Win32).
(Quelle: http://www.opengl.org/About/Architecture.html)

Schleifenoptimierung

Um den CPU-Engpaß durch die rechenaufwendigen, ständig wiederkehrenden Abfragen im Schleifenkern beim Rendering attributierter Dreieckssequenzen mit OpenGL zu reduzieren, wurden die grundsätzlich möglichen Fälle klassifiziert und für die daraus resultierenden Klassen separate Funktionen implementiert. Da die Nummer der entsprechenden Fallgruppe im DVR-Recordheader codiert ist, kann der Renderingprozeß die jeweils passende Routine direkt ausführen. Die einzelnen Routinen wurden z. T. handoptimiert, z. B. durch Loop-Unrolling sowie optionale Verwendung optimierter Rendering-Primitive (`glVertexArray()` in OpenGL 1.1 bzw. `glVertexArrayEXT()` als OpenGL-1.0-Extension, z. B. in SGI Irix).

Colormanagement

Colormanagement [26][59] wurde integriert, um die Wiedergabe-Qualität von Bildern zu erhöhen, die als Textur auf Komponenten der 3D-Szene gemappt werden. Bilder, die im RGB- oder CMYK-Farbmodell als TIFF-Format [1] vorliegen, werden mit der im Quelltext frei verfügbaren C-Bibliothek „Sam Leffler's TIFF Library" (`ftp://ftp.sgi.com/graphics/`) decodiert. Diese wurde modifiziert, um die im TIFF-Format gegebenenfalls eingebetteten ICC-konformen Quell-Farbprofil-Informationen sowie ein konfigurierbares, monitor-spezifisches ICC-Zielprofil in die Farbraum-Konvertierung einzubeziehen. Die Implementierung erfolgte für SGI Irix 6.5, Sun Solaris und Microsoft Windows 98, indem die verschiedenen – da nicht standardisierten – APIs der auf der jeweilige Plattform verfügbaren Colormanagement-Systeme (CMS) Coloratura [168], KCMS [184] bzw. ICM 2.0 [116] angewandt wurden. Ältere Microsoft-Betriebssysteme, dazu zählen Windows 95 und Windows NT 4.0, konnten hier nicht berücksichtigt werden, da der Funktionsumfang der darin enthaltenen ICM-Version 1.0 nicht ausreichte [113].

Die verwendeten CMS-Funktionen sind in *Tabelle 8* aufgeführt. Auf diesen Plattformen unterstützt das Plugin auch die farbkorrigierte, direkte Betrachtung von Bildern im TIFF-Format (MIME-Typ: `image/tiff` oder `image/x-tiff`, Datei-Endung: `.tif`) im WWW-Browser.

	Microsoft Windows 98 ICM 2.0	SGI Irix 6.5 Coloratura	Sun Solaris KCMS
Initialization	–	cmsOpen	KcsAvailable
Open Profile	OpenColorProfile	cmsImportProfile	KcsLoadProfile
Close Profile	CloseColorProfile	cmsCloseProfile	KcsFreeProfile
Get Tag Value	GetColorProfileElement	cmsGetTag	KcsGetAttribute
Create Color Transformation	CreateMultiProfile-Transform	cmsCreateTfm	KcsConnectProfiles
Transform Colors	TranslateBitmapBits	cmsApplyTfm	KcsEvaluate

Tabelle 8. CMS-Funktionen in den APIs der verschiedenen Betriebssysteme

3D-Text

Zur Bereitstellung von perspektivisch innerhalb von 3D-Szenen darstellbaren Text-Elementen sind Pixelfonts und Strichgraphiken (z. B. Hershey-Fonts) aus qualitativen Gründen ungeeignet. Skalierbare Schriften werden derzeit als Outline-Fonts in verschiedenen Formaten angeboten, die z. B. von den Firmen Adobe (Type1) und Microsoft (TrueType [117]) spezifiziert wurden. Eine frei erhältliche Font-Triangulierungssoftware (Font3D Version 1.1 von Todd A. Prater) wurde weiterentwickelt, um mehrere TrueType-Fonts in Dreieckslisten zu konvertieren. Diese wurden zwecks effizienter Einbindungsmöglichkeit in Form von C-Quelltexten erzeugt.

Räumliche Navigation und Selektion

Auf der Basis von Quaternionen, die Rotationen mathematisch beschreiben und mit der Maus mittels „Virtual Trackball“-Bedienparadigma gesteuert werden können, ist es möglich, sowohl relativ zur virtuellen Kamera als auch relativ zum Objekt intuitiv zu navigieren (siehe auch [18][44][95]). Neben dem „Rotate Object“-Navigationsmodus wird ein „Rotate Camera“-Navigationsmodus mit separaten Quaternionen unterstützt. Die „Translate XY“- und „Translate Z“-Modi dienen der Verschiebung in den Bildschirmachsen bzw. senkrecht dazu. Eine „Walk“-Navigation ermöglicht eine intuitive automatische Vor- bzw. Rückwärtsbewegung jeweils in Richtung der Kamera-Orientierung. Über die Bewegung der Maus, bei gedrückter linker Maustaste, werden dabei Beschleunigung (Mausbewegung in Vertikalrichtung) und Richtung (Horizontalrichtung) bestimmt.

Darüber hinaus wurde eine Methode implementiert, die dazu dient, die Objekt- bzw. Kamera-Rotation auch nach dem Loslassen der Maustaste weiterzuführen. Damit lassen sich z. B. Objekte in eine Drehung versetzen, indem der Nutzer diese „in Schwung“ bringt. Als Kriterium wird der zeitliche Abstand der Mouse-Events herangezogen. Ist beim Loslassen der Maustaste seit dem letzten Move-Event weniger als eine Sekunde vergangen, wird die entsprechende Rotationstransformation wiederholt mit den zuletzt ermittelten Quaternionen beaufschlagt.

Im „Pick Hyperlinks“-Modus bzw. bei gleichzeitig gedrückten Ctrl- und Alt-Tasten (unter UNIX) oder durch Doppelklick (unter Windows 95/NT) können sensitive Bereiche in der 3D-Szene (Hyperlinks) aktiviert werden. Dadurch wird der Browser angewiesen, eine URL zu laden, die im entsprechenden `WWWAnchor`-Node der VRML-1.0-Szenenbeschreibung festgelegt wurde. Ein Target-Frame kann im `EMBED`-Tag spezifiziert werden.

Unterstützung von Trackingsystemen und 3D-Interaktionsgeräten

Es wurde die Unterstützung mehrerer Headtracking-Systeme und 3D-Interaktionsgeräte integriert, die über eine serielle Schnittstelle (RS-232) Koordinaten- und Winkelwerte zum jeweiligen Rechner übertragen. Dazu gehören:

- das ultraschallbasierte Headtracking-System Stereographics/Logitech CE-VR zur Unterstützung von „Desktop-VR“-Szenarien (*Abb. 31, S. 85*),
- das magnetfeldbasierte Tracking-System Polhemus Fastrak [147],
- das ultraschallbasierte Tracking-System InterSense IS-600 Mark II [74][102],
- SpaceBall, SpaceMouse.

In diesem Zusammenhang wurden die grundsätzlichen Zusammenhänge zwischen den Sichtbedingungen und notwendigen Transformationsvorschriften für eine korrekt angepaßte stereoskopische Wiedergabe analysiert und eine Parametrisierung eingeführt.

Stereoskopische Transformation aus VRML-Daten

Es wird zwar in der Fachliteratur zum Teil behauptet, daß VRML nicht zur Stereodarstellung sowie genereller VR-Anwendung geeignet ist [62]. Tatsächlich sind diesbezüglich jedoch nicht Unzulänglichkeiten des Dateiformats zu bemängeln, sondern der verfügbaren Viewer-Implementierungen. Zur korrekten Stereopräsentation in DocShow-VR war es nun erforderlich, die Spezifikation der Betrachtungsperspektive in adäquate OpenGL-Calls zu überführen. Dazu dienten zunächst die in VRML 1.0 im Node `PerspectiveCamera` spezifizierten Parameter `position`, `orientation`, `focalDistance` und `heightAngle`. In VRML97 steht das Sprachelement `Viewpoint` mit entsprechenden Angaben (außer `focalDistance`) zur Verfügung.

Die durch den tatsächlichen Betrachterstandpunkt und die dazu relative Anordnung der Darstellungsfläche (z. B. Monitor, Projektionsscheibe) des VR-Präsentationssystems vorgegebenen Verhältnisse stimmen allerdings im allgemeinen nicht mit den VRML-Vorgaben überein. Um virtuelle Objekte, die sich in einer Ebene im Abstand `focalDistance` vom Betrachter entfernt innerhalb des vertikalen Blickwinkelbereichs `heightAngle` befinden, auf eine gerätespezifische Darstellungsfläche der Höhe *projHeight* abzubilden, ist eine Maßstabstransformation im Verhältnis

$$2 \cdot focalDistance \cdot \tan\left(\frac{heightAngle}{2}\right) \quad : \quad projHeight \tag{14}$$

erforderlich. Da der tatsächliche Betrachterabstand allerdings generell von dem zu `focalDistance` maßstäblich entsprechenden Abstand abweicht, wird in der durchgeführten Implementierung die `heightAngle`-Vorgabe aus VRML ignoriert und entsprechend den gerätespezifischen Konfigurationswerten *projHeight* und *viewDistance* neu gesetzt (*heightAngle'*):

$$\tan\left(\frac{heightAngle'}{2}\right) = \frac{projHeight/2}{viewDistance} \tag{15}$$

bzw.

$$heightAngle' = 2 \cdot \operatorname{atan}\left(\frac{projHeight}{2 \cdot viewDistance}\right). \tag{16}$$

Letztlich ergibt sich eine Maßstabstransformation im Verhältnis

$$m = \frac{focalDistance}{viewDistance} \tag{17}$$

mit dem Problem, daß je nach Abweichung des Wertes *heightAngle'* von *heightAngle* gegebenenfalls ein Ausschnitt des beabsichtigten abgebildet wird, dafür aber die Perspektive erhalten bleibt. Die Alternative wäre, den virtuellen Betrachterstandort in der Szene zu variieren. Diese Vorgehensweise stellt jedoch einen gravierenden Eingriff in die Szenenkonfiguration dar, der in der vorliegenden Implementierung nur in Verbindung mit der Nutzung von Trackingsystemen realisiert wird.

Zur stereoskopischen Transformation wird noch eine vom Augabstand *eyeDistance* mit abhängende Szenen-Translation *eyeTranslation* sowie eine Verzerrung der Sichtpyramide um *eyeShift* berücksichtigt:

$$eyeTranslation = \frac{eyeDistance}{2} \cdot \frac{focalDistance}{viewDistance} \tag{18}$$

$$eyeShift = \frac{eyeDistance}{projWidth} \tag{19}$$

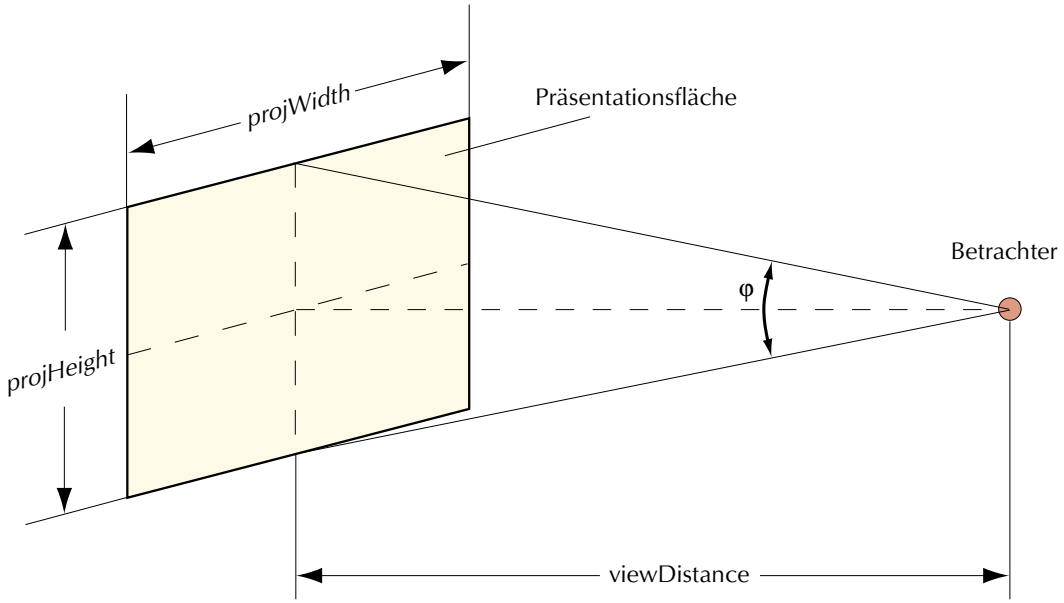


Abb. 27. Illustration der gerätespezifischen Sichtbedingungen.
$\varphi \equiv heightAngle'$ (siehe *Gleichung 15*)

Da der üblicherweise auf einem Z-Buffer-Verfahren basierende interaktive Renderingprozeß Hinweise über den Bereich der Tiefenwerte benötigt, wurde auch für diesen Zweck eine Annahme anhand des *focalDistance*-Wertes getroffen:

$$0{,}05 \cdot focalDistance \leq z \leq 50 \cdot focalDistance \tag{20}$$

Bei eingeschaltetem Trackingsystem werden sowohl die Szenen-Translation als auch der Sichtpyramidenstumpf entsprechend der Abweichung der Position des Betrachters vom Referenzort, der durch den Konfigurationsparameter *viewDistance* festgelegt ist, beeinflußt.

Problematisch für die perspektivisch korrekte Stereo-Präsentation sowie eine brauchbare Konfiguration der Near- und Far-Clipping-Ebenen (gemäß *Ungleichung 20*) ist die Tatsache, daß VRML-1.0-Dateien zum Teil keine oder unvollständige `PerspectiveCamera`-Elemente beinhalten und daß der VRML97-Standard überhaupt keine `focalDistance`-Spezifikation mehr zuläßt. Daher wurde der Vorverarbeitungsprozeß mit einer entsprechenden Kommandozeilen-Option versehen:

```
wrl2toDVR -focalDistance value datei.wrl > datei.dvr
```

Konfigurationsmöglichkeiten

Zur Anpassung an die individuelle Nutzungsumgebung sowie zur Konfiguration bzw. Auswahl von inhalts- bzw. darstellungsspezifischen Optionen wurden drei Konfigurationsmöglichkeiten implementiert:

1. Popup-Menüs, die aus dem Plugin heraus interaktiv bedient werden können.
2. Eine Konfigurationsdatei, die Bestandteil der Browser- bzw. Plugin-Installation ist und mit einem Texteditor bearbeitet werden kann (*Tabelle 9*).

Option	Wert	Erläuterung
Konfiguration der Sichtbedingungen		
projWidth	float (default: 35)	Breite der Darstellungsfläche in cm (Monitor, Projektion)
projHeight	float (default: 28)	Höhe der Darstellungsfläche in cm (Monitor, Projektion)
eyeDist	float (default: 6.5)	Augen-Abstand des Betrachters in cm (für Stereodarstellung)
viewDist	float (def.: 35)	Betrachterabstand in cm
Systemspezifische Einstellungen für Stereo-Ausgabe		
dualPipe	string (default: „“)	X-Server-Adresse der zweiten Graphik-Pipe für den Dual-Pipe-Stereomodus (z. B.: „:1“)
stereoEmbed	int (default: 0)	0: Stereowiedergabe nur im Fullscreen-Modus 1: Stereowiedergabe in sämtlichen Windows, auch im Inline-Plugin (nur für Quadbuffer-Stereomodus geeignet)
stereoOldStyle	int (default: 0)	0: Quadbuffer-Stereomodus 1: Oldstyle-Stereomodus
stereoHeight	int (def.: 492)	Zeilenanzahl im Oldstyle-Stereomodus
stereoOffset	int (def.: 532)	Zeilenoffset im Oldstyle-Stereomodus
setmonStereo	string	Befehl für Oldstyle-Stereomodus (default: /usr/gfx/setmon -n STR_RECT)
setmonDefault	string	Befehl zum Zurückschalten vom Oldstyle-Stereomodus (default: /usr/gfx/setmon -n 72HZ)
Beeinflussung des Renderings und Optimierungsoptionen		
antiAliasing	int (default: 1)	0: Antialiasing ausgeschaltet 1: Multisampling-Antialiasing aktiviert (soweit möglich) 2: Accumulation-Buffer-Antialiasing aktiviert
antiAliasingSamples	int (def.: 4)	Anzahl Subsamples für Multisampling-Antialiasing
quadric-ResolutionHQ	int (default: 6)	Hohe Auflösung der Quadric-Highlevel-Primitive (Cone, Cylinder, Sphere)

Tabelle 9. Globale Voreinstellungen in der DVR-Konfigurationsdatei `dvrconf.txt`

Option	Wert	Erläuterung
quadric-ResolutionLQ	int (default: 20)	Niedrige Auflösung der Quadric-Highlevel-Primitive (Cone, Cylinder, Sphere)
progressive-Foreground	int (default: 1)	0: Progressives Rendering im Backbuffer 1: Progressive Rendering im Frontbuffer
useOpengl-VertexArray	int (default: 1)	0: Keine Nutzung der OpenGL-1.1-VertexArrays 1: Ausnutzung der OpenGL-1.1-VertexArray
useExt-VertexArray	int (default: 1)	0: Keine Nutzung der VertexArray-Extension 1: Ausnutzung der OpenGL-VertexArray-Extension
useOpengl-TextureObject	int (default: 1)	0: Keine Nutzung der OpenGL-1.1-Texture-Objects 1: Ausnutzung der OpenGL-1.1-Texture-Objects
useExt-TextureObject	int (default: 1)	0: Keine Nutzung der Texture-Object-Extension 1: Ausnutzung der OpenGL-Texture-Object-Extension
useRendering-Optimization	int (default: 1)	0: Keine Nutzung der fallspezifisch optimierten Renderingroutinen in DocShow-VR 1: Ausnutzung der fallspezifisch optimierten Routinen
useDisplayList	int (default: 0)	0: Keine Nutzung der OpenGL-Displaylists 1: Ausnutzung von OpenGL-Displaylists
useMipMap	int (default: 1)	0: Texturemapping ohne Mipmaps 1: Texturemapping mit Mipmaps
textureMode	int (default: 0)	0: „Replace“; 1: „Decal“; 2: „Modulate“
textureQuality	int (default: 2)	0: Texturfilterung: „Nearest“ 1: Texturfilterung: „Linear“ 2: Texturfilterung: „Linear Mipmap“
Tracking- und 3D-Interaktionsgeräte		
headTracking	int (default: 0)	0: Head-Tracking ausgeschaltet 1: Head-Tracking eingeschaltet: Polhemus / Intersense 2: Head-Tracking eingeschaltet: Logitech / Crystal Eyes
fastrakStylus	int (default: 1)	Nummer des Eingangskanals für den Fastrak-Stylus
fastrakTracker	int (default: 2)	Nummer des Eingangskanals für den Fastrak-Tracking-sensor (Head-Tracking)
fastrakDevice	string (default: /dev/ttyd4)	Device-Name der seriellen Schnittstelle, an der das Trackinggerät Polhemus Fastrak oder InterSense IS-600 Mark II angeschlossen ist
logiDevice	string (default: /dev/ttyd2)	Device-Name der seriellen Schnittstelle, an der das Trackinggerät Stereographics/Logitech CE-VR angeschlossen ist
useSpaceBall	int (default: 0)	0: Keine Nutzung eines SpaceBall 1: Nutzung eines SpaceBall (über X-Input-Extension)
useSpaceMouse	int (default: 0)	0: Keine Nutzung einer SpaceMouse 1: Nutzung einer SpaceMouse (über Windows-Treiber)

Tabelle 9. Globale Voreinstellungen in der DVR-Konfigurationsdatei `dvrconf.txt`

Option	Wert	Erläuterung
CSCW-Konfiguration (Computer-Supported Collaborative Work)		
cscwDestination-Host	string (default: localhost)	Internet-Adresse des Kommunikationspartners
cscwDestination-Port	int (default: 9090)	Port-Nummer des Kommunikationspartners
cscwOwnPort	int (default: 9090)	Eigene Port-Nummer zur CSCW-Kommunikation

Tabelle 9. Globale Voreinstellungen in der DVR-Konfigurationsdatei `dvrconf.txt`

3. Plugin-Optionen, die im `EMBED`-Tag auf der HTML-Webseite mit angegeben werden können (*Tabelle 10*).

Option	Wert	Erläuterung
Generelle Optionen (Auswahl, siehe auch Netscape-Plugin-Dokumentation [125])		
SRC	string	URL
WIDTH	int	Breite des Plugin-Windows
HEIGHT	int	Höhe des Plugin-Windows
TYPE	string	MIME-Typ (z. B. applicaton/x-docshow-vr)
PLUGINSPAGE	string	URL (CGI) einer Plugin-Homepage
Plugin-spezifische globale Funktionseinstellungen		
ANTIALIAS	int	0: Antialiasing ausgeschaltet 1: Multisampling-Antialiasing aktiviert (OpenGL-Extension: Multisampling-Antialiasing) 2: Antialiasing mittes Accumulation-Buffer-Verfahren
BGCOLOR	#rrggbb	Hintergrundfarbe im Plugin-Window
DOUBLEBUFFER	int	0: Rendering-Window single-buffered 1: Rendering-Window double-buffered
PROGRESSIVE	int	0: Progressives Rendering ausgeschaltet 1: Progressives Rendering aktiviert (default)
TARGET	string	Frame-Name für im Plugin aktivierten Hyperlink
Beeinflussung des Renderings (Beleuchtung, Culling, Texturing, Transparenz, Normalisierung)		
AMBIENTLIGHT	int	0: Umgebungslicht ausgeschaltet 1: Umgebungslicht aktiviert (default)
AMBIENTLIGHT-INTENSITY	int	Intensität des Umgebungslichts in % (default: 20)
HEADLIGHT	int	0: Kamera-Licht ausgeschaltet 1: Kamera-Licht aktiviert (default)

Tabelle 10. Plugin-Optionen des `EMBED`-Tags

Option	Wert	Erläuterung
HEADLIGHT-INTENSITY	int	Intensität des Kamera-Lichts in % (default: 100)
SCENELIGHT	int	0: Szenen-Beleuchtung ausgeschaltet 1: Szenen-Beleuchtung aktiviert (default)
SCENELIGHT-INTENSITY	int	Intensität der Szenenbeleuchtg. in % (default: 100)
TWOSIDE	int	0: Polygone auf einer Seite beleuchtet 1: Polygone auf beiden Seiten beleuchtet (default)
CULLING	int	0: Culling ausgeschaltet 1: Culling aktiviert
CULLFACE	string	Culling-Orientierung: • FRONT: Culling der Polygon-Vorseite • BACK: Culling der Polygon-Rückseite
TEXTURING	int	0: Texture-Mapping ausgeschaltet 1: Texture-Mapping aktiviert (default)
TRANSPARENCY	int	0: Transparenz (Blending) ignoriert 1: Transparenz aktiviert (default)
SORT-TRANSPARENCY	int	0: Rendering in einem Durchgang (default) 1: Rendering in zwei Durchgängen: erst alle undurchsichtigen Szenenelemente, dann alle Objekte, die (halb-)transparente Materialeigenschaften haben
NORMALIZE	int	0: keine Normalisierung der Normalenvektoren 1: Normalisierung der Normalenvektoren (default)
Darstellung von Flächen und Linien		
LINEWIDTH	float	Linienbreite in Pixel (default: 1)
POLYGONMODE	string	Polygon-Renderingmodus • FILL: Flächenfüllung (default) • LINE: Liniendarstellung der Kanten • POINT: Punktdarstellung der Eckpunkte
ASPECTPIXEL	float	Seitenverhältnis eines Pixels (default: 1.0)
Sicht und Navigation		
CAMERA	int	Setzt Kamera-Nummer (default: erste Kamera)
NAVIGATION	string	Art der Navigation durch Maussteuerung: • ROTATE: Rotation des Objektes (default) • ROTATE_CAMERA: Rotation der Kamera • SCALE: Skalierung des Objekts • TRANSLATE_XY: Translation X/Y • TRANSLATE_Z: Translation Z • WALK: Vor- und Seitwärtsbewegung

Tabelle 10. Plugin-Optionen des EMBED-Tags

Option	Wert	Erläuterung
TRANSLATION	3 floats	Translation des Objekts (x, y, z)
QUATOBJECT	4 floats	Rotation des Objekts (Quaternionen)
QUATCAMERA	4 floats	Rotation der Kamera (Quaternionen)
MAXSUB-CAMERA	int	Anzahl interpolierter Positionen und Orientierungen zwischen zwei Viewpoints in der „Replay Cameras“-Funktion, sofern „Interpolate Cameras“ aktiviert ist
INTERPOLATE	int (default: 0)	0: Keine Interpolation von Viewpoints in der „Replay Cameras“-Funktion 1: Interpolation von Viewpoints aktiviert
LOOPCAMERAS	int (default: 0)	0: Kein Loop-Modus in der „Replay Cameras“-Funktion 1: Loop-Modus der „Replay Cameras“-Funktion aktiviert
PLAYCAMERAS	int (default: 0)	0: Kein Autostart der „Replay Cameras“-Funktion 1: Autostart der „Replay Cameras“-Funktion aktiviert
Szenen-Sequenzen (als Streaming-Ersatz für Offline- oder reine HTTP-Anwendungen)		
SEQUENCE	url;from–to,fps	url: URL der dynamischen Szenen mit Numerierung durch z. B. `%03d` entsprechend Formatierung in der Programmierspache C from: erste Frame-Nr. to: letzte Frame-Nr. fps: Frame-Rate (Frames je Sekunde)
LOOPSEQUENCE	int (default: 0)	0: Kein Loop-Modus in der „Replay Sequence“-Funktion 1: Loop-Modus der „Replay Sequence“-Funktion aktiv
PLAYSEQUENCE	int (default: 0)	0: Kein Autostart der „Replay Sequence“-Funktion 1: Autostart der „Replay Sequence“-Funktion aktiv

Tabelle 10. Plugin-Optionen des `EMBED`-Tags

4.2.3 DVR-Streaming und „DVRS“-Metadaten

Neben einem gewöhnlichen WWW-Server, welcher für die besprochenen Mechanismen einzelne 3D-Szenen im DVR-Format bzw. Meta-Informationen im DVRS-Format bereitstellt, wird ein spezieller 3D-Streaming-Server benötigt. Dieser wurde in einer Studie [107] auf einem UNIX-Betriebssystem (SGI Irix) prototypisch implementiert und anschließend weiterentwickelt. Dabei wurde das zunächst nur als Internet-Draft vorliegende, inzwischen in RFC 2326 standardisierte *Real Time Streaming Protocol* (RTSP) [165] für den Steuer-Datenfluß angewandt.

Es handelt sich dabei um ein Client-Server-Protokoll, welches für die Steuerung der Ausspielung oder Aufzeichnung kontinuierlicher Medienströme, wie Audio und Video, ausgelegt ist. Spezielle Protokolldatenelemente können über einen Steuerungskanal die an der Kommunikation beteiligten Server- und Client-Prozesse gegenseitig in bestimmte Zustände versetzen (siehe *Abb. 28*).

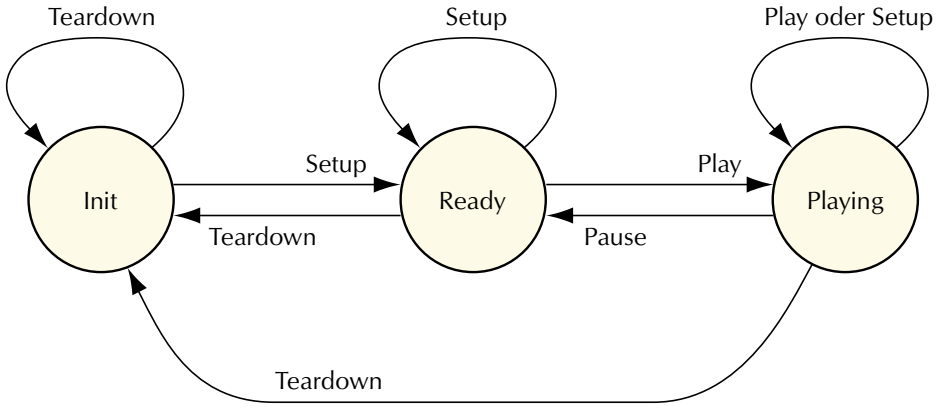


Abb. 28. Real Time Streaming Protocol (RTSP): Zustandsübergangsdiagramm für einen Ausspiel-Server

Über einen separaten Datenkanal werden dann, in Abhängigkeit vom jeweiligen Zustand, Medienströme transportiert. Im hier vorliegenden Anwendungsfall wurde dafür das DVR-Transport-Protokoll (DVRP) definiert, welches – im Gegensatz zu anderen Audio-/Video-Streaminganwendungen, die das *Transport Protocol for Real-Time Applications* (RTP) gemäß RFC 1889 [164] über UDP/IP verwenden – eine TCP/IP-basierte Übertragung festgelegt, da bei den hier betrachteten Anwendungen Verluste, Fehler oder Reihenfolge-Inkonsistenzen nicht toleriert werden können. Der DVRP-Datenstrom beinhaltet die im DVR-Dateiformat spezifizierten PDUs und fügt nach jeder Szene eine Szenen-Ende-Markierung ein (PDU-Typ = 17, siehe *Tabelle 6, S. 69*), um damit dem Client den Update-Befehl zu geben. Eine Intra-Stream-Synchronisierung wurde auf Basis der serverseitigen Auslieferungszeitpunkte realisiert. Die geforderte Framerate wird dabei im Mittel dadurch eingehalten, daß entweder entsprechende Wartephasen eingefügt werden oder gelegentlich Szenendateien ausgelassen werden.

Die Kommunikations- und Präsentationsvorgänge wurden in der DVRS-Plugin-Software in einer Weise implementiert, die ein effizientes Pipelining erlaubt. Dazu wurde ein Wechsel-Puffer – zum gleichzeitigen Rendern einer aktuellen 3D-Szene und Einlesen einer neuen 3D-Szene – mit dem Datenvolumen von zwei 3D-Szenen je Stream konfiguriert und dieser als „Shared Memory“ den beteiligten, parallel ablaufenden „Light-Weight Processes“ (LWP) – in SGI Irix: `sproc()` [171] – bereitgestellt:

- Der Hauptprozeß ist Bestandteil des WWW-Browsers und erledigt das Rendering einer jeweils vollständig übertragenen Szene auf dem primären Graphiksubsystem.
- Ein LWP realisiert den DVRP-Datentransport der jeweils aktuell neu übertragenen 3D-Szene.
- Ein weiterer LWP dient im Dual-Pipe-Stereobetrieb dazu, die jeweils aktuelle 3D-Szene auf dem zweiten Graphiksubsystem parallel zum primären zu rendern.

Die Interprozeß-Kommunikation zur Synchronisierung der Aufgaben wurde über bidirektionale Pipes realisiert, die zum Teil über den Eventmechanismus des X-Windows-Toolkits entsprechend registrierte Callback-Funktionen des Hauptprozesses auslösen. Beispielsweise erfolgt eine Mitteilung des DVRP-Prozesses über eine jeweils fertig übertragene 3D-Szene an den Hauptprozeß, der daraufhin einen neuen Renderingvorgang ausführt, während der DVRP-Prozess bereits die nächste 3D-Szene in den – inzwischen gewechselten – Lesepuffer einliest.

Zur Initialisierung des Clients werden Startup-Informationen benötigt, die in einem eigenen Klartext-Dateiformat abgelegt werden. In diesen „DVRS“-Metadaten werden die Eigenschaften eines oder mehrerer 3D-Streams spezifiziert, die z. B. durch Server-Adresse und Darstellungsattribute gekennzeichnet sind (siehe auch *Abb. 22, S. 62* in *Abschn. 4.1.2*). Die derart beschriebenen Streams werden dann synchron abgerufen und präsentiert. Dadurch ist es möglich, statische Szenenkomponenten zu definieren, die nur einmal am Beginn übertragen werden müssen – für diese wird dann die Framerate 0 spezifiziert.

Auf dem 3D-Streaming-Server liegen die „3D-Filme“ in Form von Serien einzelner DVR-Dateien. Dabei wird ein Teil des jeweiligen Dateinamens als Bildnummer, die standardmäßig mit 0 beginnt, durchnumeriert. Das jeweils benutzte Numerierungsschema wird im DVRS-Datensatz durch ein Namenstemplate spezifiziert, in dem an Stelle der Nummer eine bestimmte Anzahl „*“-Zeichen stehen. Dem Server wird dieses nach dem Verbindungsaufbau über das RTSP-Protokoll übermittelt. Dieser setzt die Bildnummer-Parameter in der jeweiligen „Play“-Anforderung dann in dieses Template ein, um die einzelnen 3D-Szenen der Reihe nach einzulesen und als DVRP-Datenstrom auszuspielen.

```
SCENES=2

MED=TCP
ADR=rtsp://origin-ge.rvs.uni-hannover.de/AWI/stations+etc2.dvr
MAXBYTES=2300000
FRAMES=1
FRAMES_P_SEC=0
SCENEEND

MED=TCP
ADR=rtsp://origin-ge.rvs.uni-hannover.de/AWI/float.****.dvr
MAXBYTES=300000
FRAMES=434
FRAMES_P_SEC=5
SCENEEND
```

Abb. 29. Beipiel für einen DVRS-Datensatz, der eine Überlagerung einer statischen Szene mit einer zeitlich veränderlichen Szene beschreibt (siehe auch *Abb. 3, S. 10*: die Stromlinien liegen hier als Szenensequenz vor, d. h. die 3D-Visualisierung jedes Zeitschritts der Strömungssimulation wurde jeweils in einer separaten DVR-Datei gespeichert)

5 Anwendung und Bewertung

5.1 Erprobung und Demonstration in Anwendungen

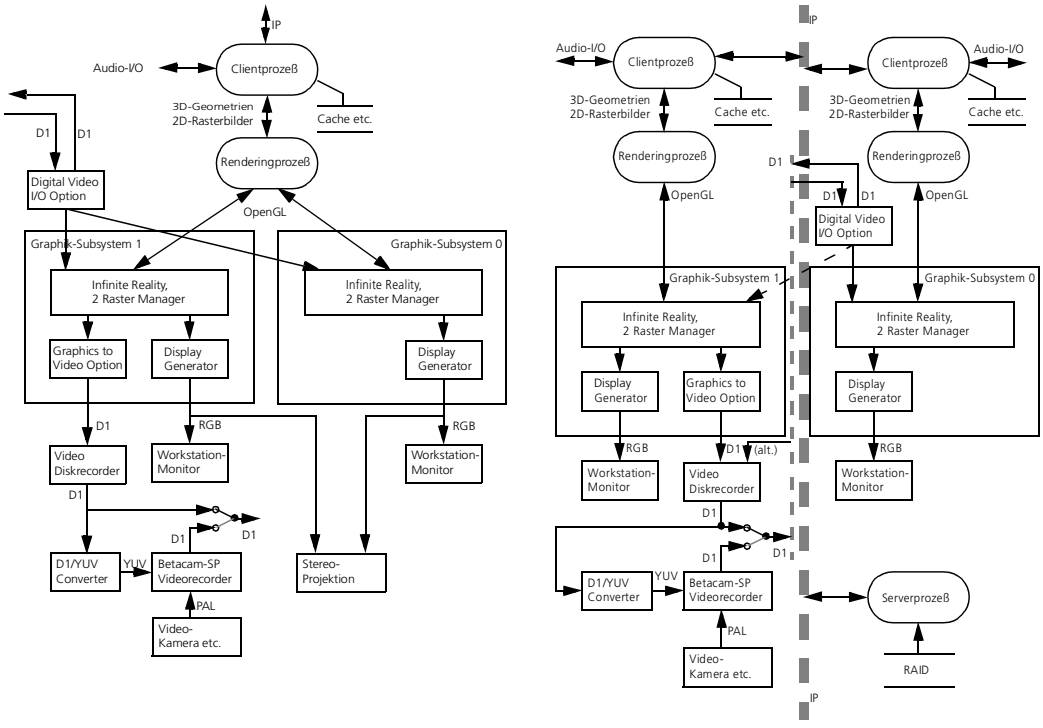
Unterstützung verschiedener Stereo-Präsentationssysteme

Grundsätzlich ist eine Reihe verschiedener Verfahren zur stereoskopischen Bildwiedergabe zu unterscheiden (siehe auch [64]), von denen am RRZN/RVS einige für den praktischen Anwendungsbetrieb installiert wurden (siehe auch *Abb. 30* und *Abb. 31*):

- *Ein einzelnes Display* – Bildschirm oder Projektion – ermöglicht die abwechselnde Präsentation der beiden Bilder, wobei der Betrachter eine aktive LCD-Shutterbrille trägt, die synchron mit der Darstellung jeweils ein Auge abdunkelt. Im Zusammenhang mit leistungsfähigen Arbeitsplatzrechnern (z. B. SGI Octane in *Abb. 31* (a)) sind dadurch auf dem Desktop bestimmte VR-Methoden anwendbar („Desktop-Virtual-Reality“). Dazu kann z. B. ein stereofähiger Monitor in Kombination mit einer LCD-Shutterbrille und einem integrierten ultraschallbasierten Trackingsystem (z. B. Stereographics/Logitech CE-VR in *Abb. 31* (a)) verwendet werden. Die jeweils eingesetzten Graphik-Subsysteme unterstützen unterschiedliche Stereo-Shutter-Modi:

  – Der *„Old Style“*-Modus ermöglicht „Full-Screen Stereo“ durch Halbierung der Zeilenanzahl bzw. Verdopplung der Vertikal-Synchronfrequenz (z. B. von 1280x1024 Pixel, 60 Hz auf 1280x492 Pixel, 120 Hz). Dieser Modus wird z. B. auf sämtlichen Workstations der Firma Silicon Graphics (SGI) unterstützt, z. B. Indy, Indigo2 Extreme, O2, Octane, Onyx Reality Engine2 (RE2) und Onyx2 Infinite Reality (IR).
  – Der *„Quad Buffer“*-Modus ermöglicht „Stereo in a Window“. Hier muß nicht zwischen zwei verschiedenen Modi umgeschaltet werden, da das entsprechende Window-System (X-Window bzw. Windows 95/98/NT) die Synchronisierung der Darstellung auf den beiden abwechselnd präsentierten Framebuffern bewirkt. Da 3D-Graphik standardmäßig im „Double-Buffer“-Modus produziert wird, muß die Graphikkarte letztlich vier Bildspeicher anbieten (LEFT_BACK, RIGHT_BACK, LEFT_FRONT, RIGHT_FRONT) – daher die Bezeichnung „Quad Buffer“. Dieser Modus wird z. B. auf den Graphiksystemen SGI Onyx RE2, SGI Onyx2 IR, SGI Octane, Sun Elite 3D, HP Kayak fx6 sowie Diamond Fire GL 1000 Pro angeboten.

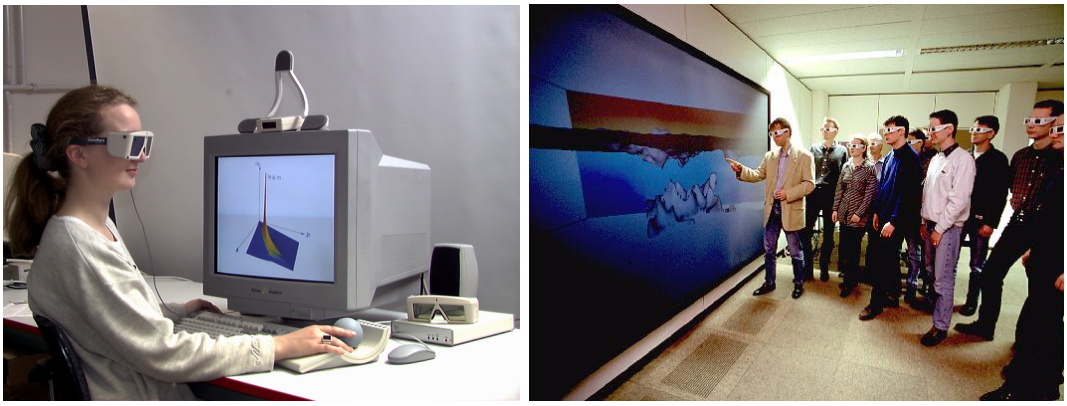
- *Zwei getrennte Displays* können entweder durch die Augen eines einzelnen Nutzers direkt betrachtet werden, wie HMD oder BOOM, oder durch Großbildprojektion mit vorgeschalteten Polarisationsfiltern einem größeren Zuschauerkreis die Betrachtung mit passiven Polfilterbrillen gestatten (*Abb. 31* (b)). Ein Großbild-Rückprojektionssystem entsprechend dieser Technik wurde Ende 1997 am RRZN/RVS installiert. Es handelt sich dabei um zwei Marquee-9500-Projektoren sowie eine Black-Matrix-Rückprojektionsscheibe mit 2,40 m Breite und 1,80 m Höhe. Der entsprechend ausgestattete 3D-Präsentationsraum bietet Platz für maximal 20 Zuschauer. Zunächst wurden die Projektoren mit zwei Videosignalen gespeist, die über einen „Dual Stereo Encoder“ aus der 3D-Graphikworkstation SGI Onyx RE2 erzeugt wurden. Dadurch wurde eine Kompatibilität zu dem für Bildschirm-Stereodarstellung adäquaten Shuttermodus der Workstation erreicht. Später, nach Inbetriebnahme eines leistungsfähigeren Graphikrechners SGI Onyx2 mit 2 IR-Graphikpipelines Ende 1998, konnte jeder Projektor an den Ausgang eines eigenen Graphiksubsystems angeschlossen werden (*Abb. 30* (a)). Durch diesen Parallelbetrieb können höhere Bildqualitäten und Frameraten erzielt werden.



(a) Stereoskopische Online-Präsentation komplexer 3D-Szenen, Video-Integration; Digitalvideosignal D1 lt. ITU-R 601 [88]:
- D1-Input z. B. für Videotexturen,
- D1-Output z. B. für Videomitschnitt.

(b) Demonstrator für verteilte Visualisierung

Abb. 30. Anwendungsbeispiele des 3D/VR-Hochleistungsrechners SGI Onyx2 Infinite Reality



(a) Stereo-Monitor, LCD-Shutterbrille, Head-Tracking und SpaceBall: „Desktop-VR“

(b) Stereo-Großbildprojektion im 3D-Präsentationsraum für ca. 20 Personen (RRZN/RVS)

Abb. 31. Beispiele nutzbarer Virtual-Reality-Gerätetechnik für 3D-Präsentation und -Interaktion

Unterstützung verschiedener Plattformen

Die entwickelte Software wurde auf mehrere Plattformen, d. h. Hardware-Architekturen / Betriebssysteme, portiert:

- UNIX
  - Hewlett-Packard (PA-RISC): HP/UX
  - Silicon Graphics (MIPS): IRIX
  - Sun (SPARC): Solaris
- Microsoft Windows 95/98/NT
  - Intel-PCs (Pentium)

Anwendungsbeispiele

Auf der Basis der entwickelten Systemkomponenten wurden mehrere praktische Anwendungen aus verschiedenen Fachgebieten unterstützt. Zum Teil wurden diese im Rahmen des DFN-Projekts „DFN-Expo – Entwicklung und Demonstration innovativer Online-Präsentationstechnologien“ als virtuelle Exponate veröffentlicht.

Beispielhaft seien hier einige genannt:

Statische 3D-Objekte
- Virtuelle Bronchoskopie,
- Rekonstruktion eines Regenwurmgangsystems,
- Fertigungstechnik,
- Thermodynamik,
- Architektur.

3D-Streaming
- Ozeanische Konvektion (siehe auch *Abb. 32*) [152][153],
- Molekulardynamik,
- Verfahrenstechnik,
- Meeres- und Polarforschung.

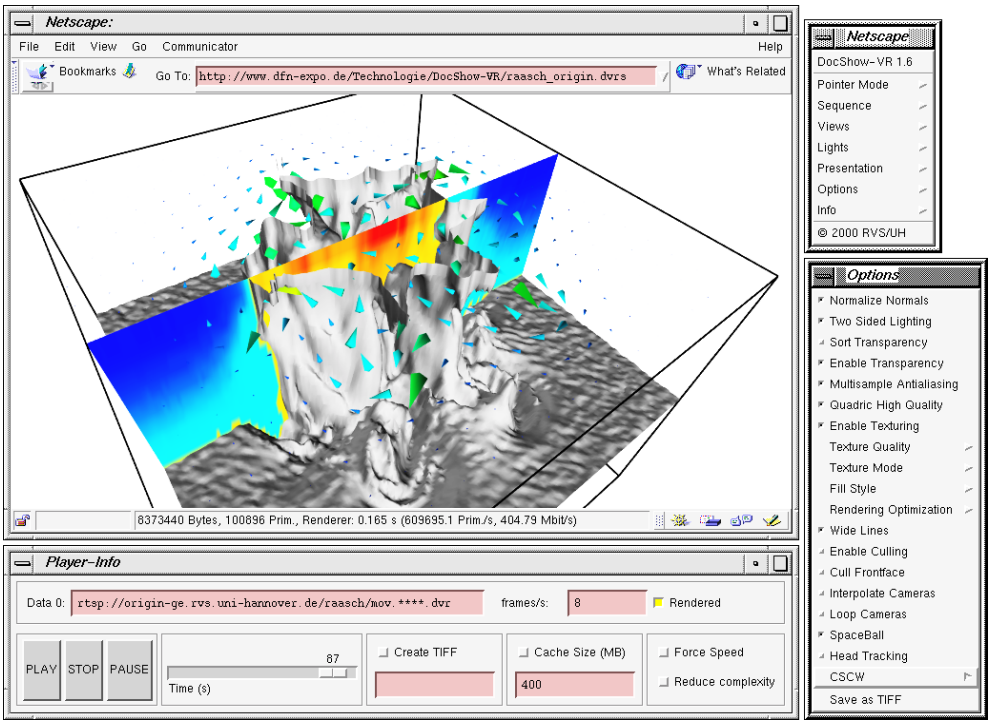


Abb. 32. Anwendung „Ozeanische Konvektion“ in der Nutzeroberfläche „DocShow-VR“

5.2 Online-Präsentation statischer 3D-Objekte

Für die Evaluierung der Implementierung wurden die Datenvolumina, Startup-Zeiten und Rendering-Raten in einigen typischen Anwendungen gemessen. Dazu wurde das Plugin DocShow-VR an geeigneten Stellen instrumentiert, d. h. Zeiten gemessen. In der UNIX-Implementierung diente dazu der System-Call `gettimeofday()`, unter Windows wurde die Win32-Funktion `GetCurrentTime()` verwendet. Am Plugin CosmoPlayer konnten die Renderingzeiten über eine eingebaute Funktionalität ermittelt werden, die durch Drücken der Taste „+“ aktiviert wird. Die Startup-Zeit wurde „von Hand“ gestoppt.

Es wurde die 3D-Szene „DX-03“, welche Bestandteil des OpenGL-Benchmarks *viewperf* [140] ist, verwendet. „DX-03“ repräsentiert das Ergebnis einer Visualisierungsanwendung. Es wurde mit dem IBM Data Explorer erzeugt und enthält 91.584 Dreiecke, die als 976 „Triangle-Strips“ mit je ca. 100 Dreiecken repräsentiert werden (siehe auch *Abb. 33, S. 88*); an den Eckpunkten liegen Normalenvektoren vor. Dieses Objekt wurde zunächst aus dem viewperf-spezifischen MSH-Dateiformat („triangle mesh“) mit Hilfe eigener Software, die im Rahmen einer Studie entwikkelt wurde [146], in VRML-Formate (Version 1.0 und 2.0) konvertiert. Die VRML-1.0-Datei wurde schließlich in das DVR-Format konvertiert, wobei eine Variante mit Triangle-Strip-Optimierung und eine Variante mit separaten Dreiecken erzeugt wurde.

5.2.1 DocShow-VR versus CosmoPlayer

In *Tabelle 11* werden Ergebnisse eines Vergleichs bei der Nutzung der Inline-Plug-ins CosmoPlayer (für VRML-Formate, zum Testzeitpunkt: Version 1.02) und DocShow-VR (DVR-Format, zum Testzeitpunkt: Version 0.9) zur Online-Präsentation der Beispielszene „DX-03“ gegenübergestellt.

	DocShow-VR 0.9 / DVR-Format		CosmoPlayer 1.02 / VRML 2.0	
	Triangle Strips	Independent Triangles	un-compressed	compressed (gzip)
Datenvolumen [byte]	2.252.744	6.601.928	9.768.093	2.266.695
Startup-Zeit; „progressive aus“: ohne Renderzeit	0,795 s, progressive: 0,911 s	1,709 s, progressive: 1,894 s	ca. 25 s	ca. 25 s
Äquivalente Bitrate (Startup)	22,7 Mbit/s, progr.: 19,8 Mbit/s	30,9 Mbit/s, progr.: 27,9 Mbit/s	3,1 Mbit/s	0,73 Mbit/s
Rendering-Zeit	0,155 s, optimized: 0,084 s	0,151 s, optimized: 0,144 s	ca. 0,36 s	
Renderingrate [Dreiecke/s]	590.864, optimized: 1.090.286	606.517, optimized: 636.000	254.400	
Äquivalente Bitrate (Rendering)	116 Mbit/s optimized: 214 Mbit/s	350 Mbit/s optimized: 367 Mbit/s		

Tabelle 11. Leistungsvergleich des Szenarios „DX-03“ – DocShow-VR versus CosmoPlayer. Fenstergröße 512 x 512 Pixel auf einer Silicon Graphics (SGI) Onyx Reality Engine2 (2 x R4400, 200 MHz, 2 RMs); Zugriff auf einen WWW-Server Apache auf einer SGI Challenge L (2 x R4400, 200 MHz), Verbindung: TCP/IP über ATM, 155 Mbit/s. Varianten in der DVR-Messung:
(a) „Progressive“ – mit inkrementeller Darstellung während des Transports
(b) „Optimized“ – ohne „Normalize Normals“, „Two-sided Lighting“, „Transpar.“

In der beschriebenen Testkonfiguration wurden die folgenden Beobachtungen gemacht:

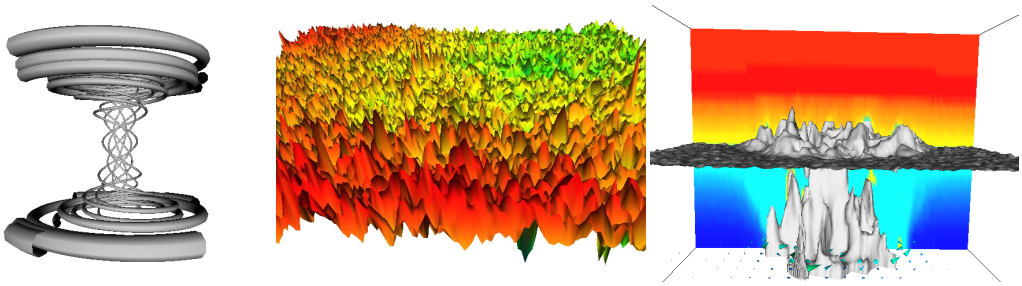
- Das Datenvolumen der optimierten DVR-Datei ist ähnlich wie das der gzip-komprimierten VRML-2.0-Datei.
- Die Startup-Zeit wurde gegenüber der Anwendung mit dem CosmoPlayer auf ca. 1/28 reduziert.
- Die Renderingrate ist in DocShow-VR 2- bis 4-mal so hoch wie im CosmoPlayer. Es wurde die im SGI-Datenblatt des Reality-Engine2-Graphiksubsystems spezifizierte Peak-Polygonrate – „3D triangle meshes, smooth shaded, z-buffered, phong lighting: 1,07 Mio. triangles/s“ – erreicht. Diese sind auch vergleichbar mit entsprechenden Meßergebnissen aus dem OpenGL-Benchmark *viewperf*.

Allerdings mußte dazu in der Vorverarbeitung die Triangle-Strip-Optimierung durchgeführt werden, und die Plugin-Optionen „Normalize Normals“, „Two-sided Lighting“ und „Transparency“ mußten abgeschaltet werden („optimized“ in *Tabelle 11*). Werden diese Optimierungsmöglichkeiten nicht beachtet, so sind Nachteile bezüglich der Datenvolumina, Startup-Zeiten und der Rendering-Performance die Folge.

Jedoch können die genannten erheblichen Vorteile nicht verallgemeinert werden, da die Optimierung der Modell-Repräsentation sowie der Rendering-Optionen nicht generell anwendbar sind, z. B. weil andere 3D-Objekte nicht in diesem Maße automatisch optimierbar sind oder je nach Anwendungskontext ineffizientere Graphik-Attribute verwendet werden müssen.

5.2.2 Optimierungen in Vorverarbeitung und Renderingvorgang

Zur Beurteilung der Auswirkungen verschiedener implementierter Optimierungsoptionen wurden drei verschiedene, anwendungsnahe Beispielszenen verwendet, die jeweils eine Komplexität in der Größenordnung von ca. 100.000 Polygonen aufwiesen. In *Abb. 33* sind Beispielansichten sowie Charakteristika dargestellt.



(a) **„DX-03“**
Anwendung des IBM Data Explorer (aus dem OpenGL-Benchmark *viewperf* [140])

91.584 Dreiecke mit Normalen / Eckpunkt

(b) **„Messung der Rauhheit einer Oberfläche“**
AVS-Ergebnis (Institut für Fertigungstechnik, Universität Hannover)

130.050 Dreiecke; Normalen und Farben / Eckp.

(c) **„Ozeanische Konvektion, Zeitschritt Nr. 492“**
AVS/Express-Ergebnis (Institut für Meteorologie und Klimatologie, Univ. Hannover)

96.412 Primitive; verschiedene Attribute

Abb. 33. Anwendungsszenarien der durchgeführten Messungen

Auswirkung der Optimierungen bei der Vorverarbeitung

Die in *Abb. 33* dargestellten 3D-Szenen lagen als VRML-Dateien vor und wurden zunächst in das DVR-Format konvertiert. Dabei wurden jeweils zwei Varianten erzeugt: eine mit unabhängigen Dreiecken und eine weitere, die, soweit möglich, automatisch erkannte Triangle-Strips repräsentiert. Wie die Ergebnisse in *Tabelle 12* zeigen, sind die Vorteile dieser Optimierung sowohl in bezug auf das Datenvolumen als auch auf die Renderingzeit deutlich.

	Datenvolumen der DVR-Dateien	Renderingleistung	HP Kayak fx6+	SGI Onyx2 IR
(a) „DX-03“	Triangle strips: 2.252.744 byte	Renderingzeit: -polygonrate: -bitrate:	15 ms 6,11 Mio./s 1,20 Gbit/s	21 ms 4,34 Mio./s 0,85 Gbit/s
	Independent triangles: 6.601.928 byte	Renderingzeit: -polygonrate: -bitrate:	46 ms 1,95 Mio./s 1,15 Gbit/s	53 ms 1,72 Mio./s 0,99 Gbit/s
(b) „Messung der Rauhheit einer Oberfläche“	Triangle strips: 3.658.168 byte	Renderingzeit: -polygonrate: -bitrate:	31 ms 4,195 Mio./s 0,944 Gbit/s	62 ms 2,095 Mio./s 0,471 Gbit/s
	Independent triangles: 10.924.656 byte	Renderingzeit: -polygonrate: -bitrate:	125 ms 1,040 Mio./s 0,699 Gbit/s	193 ms 0,675 Mio./s 0,453 Gbit/s
(c) „Ozeanische Konvektion, Zeitschritt Nr. 492“	Triangle strips: 3.117.596 byte	Renderingzeit: -polygonrate: -bitrate:	47 ms 2,051 Mio./s 0,531 Gbit/s	86 ms 1,118 Mio./s 0,289 Gbit/s
	Independent triangles: 8.001.604 byte	Renderingzeit: -polygonrate: -bitrate:	109 ms 1,002 Mio./s 0,666 Gbit/s	141 ms 0,684 Mio./s 0,454 Gbit/s

Tabelle 12. Datenvolumina, Renderingzeiten, -polygonraten und -bitraten in den drei Anwendungsszenarien gemäß *Abb. 33*.
Plugin: DocShow-VR 1.4. Fenstergrößen: (a) 512x512, (b) und (c) 640x480 Pixel. Rendering: OpenGL immediate mode, vertex arrays, 1 light, one-sided lighting (außer c: two-sided), disable transparency. Plattformen:
– HP Kayak XW fx6+, 2 x Pentium III Xeon, 550 MHz, Windows NT 4.0 SP4
– SGI Onyx2 Infinite Reality, 4 x MIPS R10000, 195 MHz, Irix 6.5.5

Das Datenvolumen wird durch die eingeschaltete Optimierung bei der Vorverarbeitung der Szenen (a) und (b) nahezu um den prinzipiell erreichbaren Faktor 3 (siehe auch *Abb. 12, S. 31*) reduziert. Die VRML-Polygondaten wurden nahezu vollständig in die Triangle-Strip-Primitive des DVR-Formats umgewandelt.

Die Renderingleistung wurde auf verschiedenen Plattformen gemessen:

- 3D-Graphikworkstation der Firma Hewlett-Packard (Pentium, Windows NT),
- 3D-Graphiksupercomputer der Firma Silicon Graphics (MIPS, Irix).

Die Renderingzeit beträgt für die Triangle-Strip-Repräsentation in einigen Fällen ca. 1/3 gegenüber der mit unabhängigen Dreiecken. Die absolute Zeit liegt je nach Modell zum Teil deutlich unter 100 ms, der für eine interaktive Navigation maximal tolerierbaren Update-Zeit. Die getesteten Graphiksysteme sind für Anwendungen mit 3D-Szenen der hier vorliegenden Komplexität von ca. 100.000 Polygonen also geeignet.

Die Renderingbitraten, die den Datendurchsatz zwischen dem Hauptspeicher und dem Graphiksubsystem charakterisieren, liegen zwischen 0,3 und 1,2 Gbit/s.

Die Renderingrate eines aktuellen Hochleistungs-PCs übertrifft sogar die des für VR-Anwendungen üblicherweise eingesetzten High-End-Graphikrechners. Allerdings bietet das PC-Graphiksystem derzeit keinen Hardware-Support für Global-Scene-Multisampling-Antialiasing und ist daher aufgrund der relativ geringen Bildqualität für immersive VR-Anwendungen in dieser Form ungeeignet.

Auswirkungen der Optimierungsoptionen des Renderingvorgangs

Um zu zeigen, welche Vor- oder Nachteile die in *Abschn. 3.2.1* und *Abschn. 4.2.2* diskutierten Maßnahmen bringen, wurden die Rendering- und Display-List-Compilationszeiten für die Szene „DX-03“ unter verschiedenen Bedingungen gemessen:

- „No Optimization“: Rendering mit einer Universal-Routine.
- „Loop Unrolling“: Rendering mittels spezialisierter Routine, die der jeweiligen Topologie (hier: Triangle-Strips) besonders angepaßt ist; je 4 Normalen- und Koordinatenvektoren werden blockweise verarbeitet.
- „Vertex Array“: Rendering mittels OpenGL-Erweiterung „Vertex Array“.
- „Display List“: Rendering als Display-List; dabei wurde auch der Compile-Vorgang separat gemessen.

Die auf verschiedenen Plattformen ermittelten Ergebnisse sind in *Tabelle 13* aufgeführt.

Rechner / Graphikkarte	Immediate Mode			Display List	
	No Optimiz.	Loop Unroll.	Vertex Array	Compilation	Rendering
Onyx 2 / IR	37 ms	29 ms	21 ms	100 ms	15 ms
HP Kayak XW / fx6+	15 ms	15 ms	15 ms	46 ms	15 ms
PC / Elsa Erazor II	85 ms	80 ms	*135 ms*	130 ms	60 ms
PC / Elsa Synergy II	65 ms	60 ms	50 ms	195 ms	50 ms

Tabelle 13. Renderingzeiten für Szene „DX-03“ mit verschiedenen Optimierungsoptionen. Plugin: DocShow 1.4. Fenstergröße: 512x512. Plattformen wie in *Tabelle 12* und
– PC, Elsa Erazor II (Treiber 4.11.01.0301-0023), Pentium II, 400 MHz, Windows 98
– PC, Elsa Synergy II (Treiber 4.11.01.0300-0016), Pentium III, 500 MHz, Win. 98

Es wurde beobachtet, daß die Nutzung der OpenGL-Vertex-Array-Erweiterung nicht in jedem Fall den erhofften Vorteil bringt, sondern je nach Plattform auch zu einem deutlichen Performance-Verlust führen kann. Dieser beruht vermutlich auf einer unzulänglichen Treiber-Implementierung der jeweiligen Graphikkarte[1].

Die Anwendung von OpenGL-Display-Listen ist generell mit einer erheblichen Startup-Zeit für die „Compile“-Aktion, in der OpenGL eine Optimierung durchführt und die Graphikdaten gegebenenfalls in einem hardwarespezifischen Cache ablegt, verbunden. Für den Fall statischer Szenenpräsentation und -Navigation liegt die

[1] Diese These stützt sich unter anderem auf Messungen, die auf dem „PC / Elsa Erazor II“ mit einem neueren Graphiktreiber durchgeführt wurden. Mit diesem Treiber (der aber andere Probleme aufwies) wurde für „Vertex Arrays“ ca. eine Halbierung der Renderingzeit erzielt.

„Compile“-Zeit zwar in einem tolerierbaren Bereich, und die Renderingrate wird zum Teil deutlich erhöht. Im Zusammenhang mit der hier betrachteten Ziel-Anwendung „3D-Streaming“, bei der häufig nach einem Szenen-Update nur ein einziger Renderingvorgang erfolgt, ist jedoch die Summe aus Compilation und Rendering zu betrachten – diese ist wesentlich höher als die Renderingzeit im OpenGL-„Immediate Mode“. Daher sind Display-Listen in diesem Anwendungsfall ungeeignet.

5.3 3D-Streaming – „Virtual Reality Movies“

Zur Evaluierung des Produktionsprozesses in einer typischen Anwendung aus der „Wissenschaftlichen Visualisierung“ wurde ein Testbed aufgebaut, welches aus den folgenden Komponenten bestand:

1. DVR-Datensatz:

   Aus einer *Simulationsrechnung* eines instationären Strömungsphänomens (Ozeanische Konvektion [153]), welche auf einem Supercomputer Siemens/Fujitsu VPP 300 durchgeführt wurde, resultierten Rohdaten von 940 Zeitschritten. Diese beinhalteten jeweils Temperaturwerte und Strömungsgeschwindigkeiten an 161x161x31 orthogonalen Gitterpunkten, gespeichert als float-Werte in 32 bit Genauigkeit. Das Datenvolumen dieser Ergebnisdaten betrug demnach:

   940 x 161 x 161 x 31 x (1 + 3) x 4 byte = 12.085.407.040 byte

   Die Ergebnisdaten jedes einzelnen Zeitschritts wurden dann mit Hilfe von Moduln einer mit AVS/Express implementierten *Visualisierungsanwendung* in eine dreidimensionale Szene aufbereitet. Dabei wurde das Temperaturfeld durch einen gemäß Farbtabelle eingefärbten vertikalen „Slicer“ sowie eine „Isosurface“ eines festgelegten Temperaturwertes und das Geschwindigkeitsfeld an äquidistanten Positionen auf einer festgelegten, horizontalen Ebene als „Arrows“ visualisiert. Aus AVS/Express heraus wurde mit dem Modul „OutputVRML“ jede einzelne 3D-Szene im Format VRML 1.0 gespeichert. Aus diesen VRML-Dateien wurde schließlich durch *Konvertierung* mit dem Werkzeug „wrl1toDVR“ eine Szenensequenz im DVR-Format erzeugt.

   940 *DVR-Dateien* in einer Größe von jeweils ca. 5–9 Mbyte standen letztlich für eine Leistungsbewertung zur Verfügung, wobei jede ca. 100.000 Primitive (Linien, Polygone) repräsentierte.

   Ein Beispiel (Szene Nr. 492) ist in *Abb. 33* (c) dargestellt.

2. Client-Server-Konfiguration für 3D-Streaming:

   Als *Client-System*, bei dem es hauptsächlich auf 3D-Renderingleistung ankommt, wurde ein Silicon Graphics (SGI) Onyx2 Racksystem (4 Prozessoren R10000/195 MHz, 2 Gbyte Hauptspeicher, 9 Gbyte SCSI-Disk, Betriebssystem Irix 6.5.4) verwendet, das mit zwei Graphiksubsystemen vom Typ „Infinite Reality“ (jeweils mit zwei 64 MB Raster Managern) ausgestattet war.

Ein *3D-Streaming-Server* wurde ebenfalls auf dieser Maschine installiert, und zwar für Entwicklungszwecke und Testmöglichkeiten eines High-Performance-TCP/IP-Szenarios über die Loopback-Kommunikation. Die DVR-Dateien wurden auf einem 140-Gbyte-RAID-System – bestehend aus 10 SCSI-Disks mit je 18 Gbyte – gespeichert, welches über Fibre Channel (800 Mbit/s) an die Onyx2 angeschlossen war.

Ein *separater 3D-Streaming-Server* wurde auf einem Serverrechner vom Typ SGI Origin200 (R10000/225 MHz, 512 Mbytes Hauptspeicher, 9 + 18 Gbyte SCSI-Disk, Betriebssystem Irix 6.5.4) installiert. In diesem Fall wurden einige DVR-Dateien auf einer eingebauten SCSI-Disk (18 Gbyte) gespeichert.

Als *Datennetz* zwischen der Onyx2 und der Origin200 diente eine Back-to-back-Verbindung auf der Basis von Gigabit-Ethernet. Zur Optimierung der Leistungsfähigkeit wurde auf beiden Maschinen die Unterstützung von „Alteon Jumbo-Frames“ durch eine Konfiguration im Betriebssystemkern aktiviert. Wie in anderen Untersuchungen gezeigt wurde, kann durch Nutzung dieser Option, die allerdings bislang proprietär ist, das hohe Leistungspotential dieser Netzinfrastruktur erheblich besser ausgenutzt werden, da wegen der größeren MTU – 9000 byte statt der standardisierten Größe von 1500 byte – die Endgeräte (hier: Onyx2, Origin200) wesentlich weniger durch Interrupts belastet werden. In der hier verwendeten Konfiguration konnten bis zu ca. 720 Mbit/s (Memory-to-memory, von der Origin200 zur Onyx2) umgesetzt werden [11].

		Client SGI Onyx2	Server 1 SGI Onyx2	Server 2 SGI Origin200
Lesen von DVR-Dateien, ein Leseblock je Datei (in Klammern zweiter Leseversuch: aus Betriebssystem-Cache)		–	250–350 Mbit/s (980–1020 Mbit/s)	109–110 Mbit/s (1090–1170 Mbit/s)
TCP/IP-Übertragung vom Server zum Client mit dem Meßprogramm *ttcp* (ttcp -l1048576 -n100 -b262144)		–	790–950 Mbit/s (TCP/IP-Loopback)	560–670 Mbit/s (TCP/IP über Gigabit-Ethernet)
3D-Rendering: OpenGL im „immediate mode“ (DocShow-VR)		380–420 Mbit/s	–	–
Client-Server-Streaming-Pipeline (DocShow-VR)	Maximale Framerate	–	280–320 Mbit/s	324–326 Mbit/s
	4 frames / Sekunde	–	251 Mbit/s	251 Mbit/s
	2 frames / Sekunde	–	126 Mbit/s	126 Mbit/s

Tabelle 14. Ergebnisse der Messungen des Durchsatzes (typische Werte) in einer lokalen (Client und Server auf einem Rechner, TCP/IP-Verbindung über Loopback-Interface) und in einer verteilten Konfiguration (Client und Server über Gigabit-Ethernet verbunden) im Szenario „Ozeanische Konvektion“ (Szenen Nr. 720–739, 165.353.764 byte)

Zunächst wurde der Durchsatz jeweils an den kritischen Pfaden (Lesen von Datei, Übertragung über TCP/IP, 3D-Rendering über OpenGL) separat gemessen. Schließlich wurde noch die „Über-alles“-Performance der gesamten Streaming-Verarbeitungskette bewertet, wobei die vorgegebene Framerate variiert wurde.

In *Tabelle 14* sind einige charakteristische Größen zusammengestellt. Die wesentlichen Beobachtungen aus diesem Szenario werden im folgenden zusammengefaßt:

- Der Durchsatz in der verteilten Konfiguration ist höher als im Fall der Ausführung von Client und Server auf derselben Maschine. Die Ursachen hierfür sind:
  - die geringere Taktrate auf der Client-Maschine (195 MHz gegenüber 225 MHz),
  - der Overhead, der dadurch entsteht, daß auf einer Maschine 3 Prozesse bzw. Threads gleichzeitig laufen und dabei um gemeinsame Betriebsmittel konkurrieren: der Serverprozeß (Lesen von Datei, Senden über TCP/IP-Socket), der Clientprozeß (im wesentlichen Rendering) und ein Client-Thread, der den Empfang der 3D-Daten über einen TCP/IP-Socket durchführt.
- Die Dauerrate der Streaming-Pipeline ist deutlich geringer als die separat gemessene Renderingrate. Sofern die abgerufenen Szenensequenzen klein genug sind, um im Cache des Betriebssystems gehalten zu werden (was hier der Fall war), stellt die Renderingrate in der vorliegenden Konfiguration jedoch den limitierenden Faktor dar. Die weitere Verringerung des Durchsatzes kann durch folgende Ursachen begründet werden:
  - Im „Double-Buffer“-Betrieb des Graphiksystems entsteht nach der Fertigstellung jedes Bildes durch Abwarten der Vertikalsynchronisierung eine „Swap-Buffer“-Latenz. Diese entspricht maximal dem Kehrwert der Bildrate des Displaysystems:

    $$t_{SwapBuffer} \leq \frac{1}{DisplayFrameRate} \ . \tag{21}$$

    Im vorliegenden Fall mit 72 Hz Bildwiederholrate beträgt diese also:

    $$t_{SwapBuffer} \leq \frac{1}{72} \ \mathrm{s} \ . \tag{22}$$

  - Innerhalb der Plugin-Software signalisiert der Transport-Thread dem Hauptprozeß jeweils die Beendigung der Übertragung einer Szene über eine Pipe. In X-Windows/Motif-Anwendungen, wie im hier verwendeten WWW-Browser Netscape Communicator, werden sämtliche Ein-/Ausgabe-, Timer- und X-Windows-Events in einer zentralen Funktion aus dem X-Toolkit der X-Windows-Library `XtAppMainLoop()` behandelt. Diese dient als Dispatcher und arbeitet die auftretenden Events sequentiell durch den Aufruf anwendungsspezifisch zur Laufzeit registrierter Callback-Rou-

tinen ab. Zur Auffrischung des Inhalts eines Fensters wird üblicherweise der Xlib-Aufruf `XClearArea()` verwendet, der zusätzlich noch einen Round-Trip-Vorgang über den X-Server nach sich zieht. Darüber hinaus wurde dafür ein Timer verwendet, um der Anwendung die Gelegenheit zu geben, auf weitere Events reagieren zu können, z. B. von der Maus.
Die Reaktionszeit bis zum Start des nächsten Renderingvorgangs liegt nach eigenen Messungen in der Größenordnung von 10–20 ms.

– Die Initialisierung des OpenGL-Status zu Beginn jedes Renderingvorgangs benötigt ebenfalls Zeit, die allerdings im Vergleich zu den bislang diskutierten Einflüssen nach eigener Einschätzung vernachlässigbar ist.

In der folgenden *Abb. 34* ist der zeitliche Ablauf der auf Client-Seite auftretenden Aktivitäten veranschaulicht, auch um die korrekte Funktion der implementierten Moduln qualitativ zu demonstrieren. Dazu wurden die Zeitpunkte relevanter Ereignisse sowie Datenvolumina innerhalb der Client-Software gemessen und gespeichert. Diese Daten wurden nach Ablauf des Anwendungsszenarios auf Datei geschrieben. Schließlich wurden die Ergebnisdaten des Meßprotokolls visualisiert.

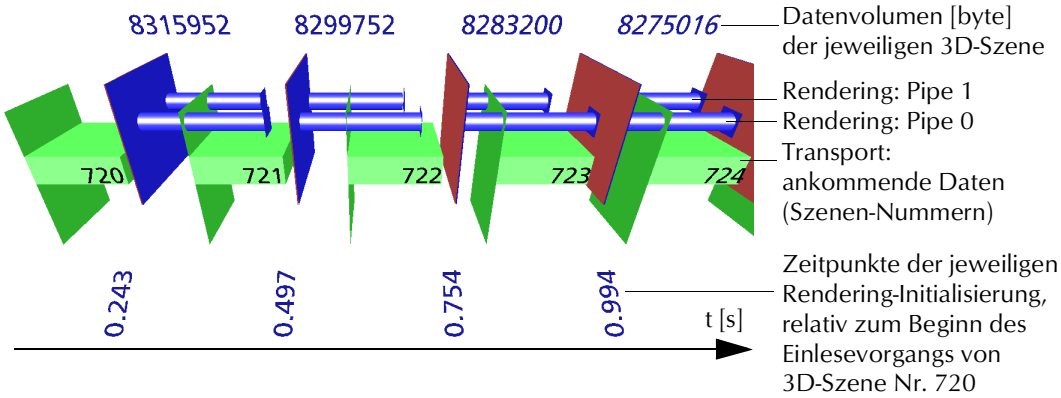


Abb. 34. Zeitablauf der Transport- und Renderingvorgänge im 3D-Streaming-Client mit 4 frames/s. Balken: Übertragung einer 3D-Szene, Röhren: Dual-Pipe-Rendering.

Zunächst wurde der 3D-Streaming-Server durch den „Sequence“-Dialog der Nutzeroberfläche des „DocShow-VR“-Plugins veranlaßt, die Szenen 720 bis 739 mit 4 Frames je Sekunde auszuspielen. Da die verfügbare Datenrate höher ist als für diese Anforderung nötig, legt der Server zur Synchronisierung nach der Übertragung einer Szene jeweils eine Pause ein, um die Übertragung der nächsten Szene jeweils im vorgegebenen Zeitraster zu beginnen. Dies ist an den unteren Balken, die die vom Client eingelesenen Datenpakete illustrieren sollen, zu erkennen. Der Client wurde hier im Dual-Pipe-Modus konfiguriert, d. h. die Szenen werden nicht nur im Hauptprozeß des Plugins auf dem ersten Display gerendert, sondern zusätzlich wird in einem separaten Thread eine Aktualisierung des Bildes auf dem zweiten Display vorgenommen. Dieser Parallelbetrieb wird mit den oben angeordneten Röhren dargestellt, die sich zeitlich wiederum mit dem Datentransport der nächsten Szene überlappen.

Kritisch anzumerken ist, daß die Synchronisierung derzeit nur grob erfolgt, da diese nur serverseitig bewirkt wird. Daher wirken sich Jitter der Kommunikationsstrecke sowie variable Renderingzeiten unmittelbar auf die aktuelle Update-Rate des Client aus. Im Mittel wird die Rate jedoch eingehalten, da der Beginn der Ausspielung als Bezugspunkt dient und der Server gegebenenfalls auch Frames ausläßt, wenn die Daten aufgrund der in TCP/IP integrierten Flußkontrolle nicht genügend schnell gesendet werden können.

Zur Glättung wäre als weitere Maßnahme eine clientseitige Pufferung – eventuell in Kombination mit Regelungsmechanismen (wie z. B. in [28][63] für Videostreaming) – vorteilhaft, die im Rahmen weiterer Forschungs- und Entwicklungsarbeiten hinzugefügt werden sollte.

6 Zusammenfassung und Ausblick

Es wurden technische Aspekte aus dem Bereich der auf Internet-Standards basierenden, multimedialen Informationsdienste (World Wide Web – WWW) behandelt. Zunächst wurden fortgeschrittene Multimedia-Technologien bis hin zur immersiven Virtuellen Realität (VR) im Kontext der Präsentation wissenschaftlicher Ergebnisse in netzverteilten Arbeitsumgebungen analysiert. Unter besonderer Berücksichtigung eines ausgewählten Ziel-Szenarios – die netzverteilte, multimediale Unterstützung der wissenschaftlichen Visualisierung – wurden verschiedenartige Unzulänglichkeiten in den derzeit verfügbaren, auf Internet-Standards basierenden Konfigurationen (z. B. WWW-Browser-/Viewer-Software) identifiziert, die eine produktive Anwendung beeinträchtigen bzw. verhindern können. Die daraus abgeleiteten Anforderungen an die zu entwickelnden Moduln wurden in bezug auf die Aspekte *Leistungsfähigkeit* (z. B. Startupzeit, Framerate), *Qualität* (z. B. Renderingverfahren) und *Funktionalität* (z. B. Streaming) charakterisiert. Unter Berücksichtigung relevanter Standards wurden diese den Instanzen *Präsentation* (z. B. Client, Display), *Kommunikation* (z. B. Datentransport, Steuerung) und *Repräsentation* (z. B. Codierung, Server) zugeordnet.

Die betrachteten Prozeßketten beschränken sich dabei auf einen visuell wahrnehmbaren Medientyp, nämlich statische, polygonal repräsentierte 3D-Objekte. Diese sind zur immersiven Präsentation und Interaktion in einem Virtual-Reality-System besonders geeignet. Zur Erweiterung hinsichtlich der zeitlichen Dimension wurden Sequenzen von Einzelszenen verwendet.

Zur Lösung der aufgezeigten Probleme wurde eine Reihe innovativer Ansätze erarbeitet. Diese wurden zu Demonstrations- und Evaluierungszwecken in eigene Entwicklungen integriert:

- Eine Vorverarbeitung polygonaler 3D-Modelle, die in Standard-Datenformaten (Virtual Reality Modeling Language – VRML, ISO/IEC 14772:1997) vorliegen, resultiert in einer binären Repräsentation zur effizienteren Übertragung und Darstellung – einem eigenen „DVR“-Datenstromformat. Dadurch können wesentlich komplexere 3D-Szenen interaktiv verarbeitet werden als mit herkömmlichen, VRML-basierten Werkzeugen.
- Zur effizienten Präsentation wurde ein 3D-Viewer – „DocShow-VR“ – auf der Basis von OpenGL (Industrie-Standard-API für 3D-Graphik) implementiert. Dabei werden stereoskopische Displays sowie dreidimensionale Tracking- und Interaktionsgeräte unterstützt und hochqualitative Renderingverfahren (Antialiasing, Colormanagement) angewandt.

Die Realisierung erfolgte als Inline-Plugin für WWW-Browser auf UNIX- und Microsoft-Windows-95/98/NT-Plattformen. Dadurch wird eine komfortable Einbettung von 3D-Abbildungen in WWW-Seiten, z. B. nach gestalterischen Gesichtspunkten, ermöglicht.

- Unter Anwendung eines Streaming-Steuerprotokolls (Real Time Streaming Protocol – RTSP, Internet-Standard: RFC 2326) wurde ein leistungsfähiges Client-Server-System zur Ausspielung zeitlich veränderlicher Szenenbestandteile realisiert. Animationen virtueller 3D-Szenen, die als Sequenzen von DVR-Dateien vorbereitet und auf einem 3D-Streamingserver bereitgestellt werden, können damit von einer 3D-Graphikworkstation interaktiv abgerufen werden. Der bereits erwähnte 3D-Viewer wurde dazu entsprechend erweitert. Während der Ausspielung kann im virtuellen Raum sowie in der Zeitachse interaktiv navigiert werden.

Erstmalig wurden Streaming- und medienspezifische Skalierungsmethoden, die ursprünglich für herkömmliche kontinuierliche Audio/Video-Medien entwickelt wurden, auf teleimmersive Szenarien adaptiert. Als Skalierungsverfahren wurden hier Optionen sowohl in zeitlicher Hinsicht (Bildrate) als auch bezüglich der Objektkomplexität (Level of Detail) verwendet.

Abschließend wurden die in der praktischen Anwendung gewonnenen Erfahrungen dargestellt und eine Bewertung anhand von Messungen durchgeführt. Im lokalen Netz (IP über ATM, 155 Mbit/s) wurde auf einem leistungsfähigen Arbeitsplatzrechner (Silicon Graphics Onyx Reality Engine2) eine anwendungsnahe, statische 3D-Szene mit einer Komplexität von ca. 100.000 Polygonen („DX-03“ aus dem OpenGL-Benchmark *viewperf*) von einem WWW-Server (Silicon Graphics Challenge L) abgerufen. Der Zeitaufwand für Startup und Bildaufbau wurde dabei von ca. 25 Sekunden (VRML-Format, Cosmoplayer) auf ca. 0,9 Sekunden (DVR-Format, DocShow-VR), also auf 1/28, reduziert. Während der Übertragung der 3D-Daten bietet DocShow-VR außerdem einen progressiven Bildaufbau („On-the-fly-Rendering“). Durch die Verkürzung der Latenz beim Abruf virtueller 3D-Szenen relativ hoher Komplexität sowie die Unterstützung von VR-Gerätetechnik (Stereoskopische Displays, 3D-Interaktion, Headtracking) wurde die Interaktivität und damit Komfort und Produktivitätspotential bei der 3D/VR-Anwendung in Informationssystemen erheblich erhöht. Darüber hinaus wurde durch die Verschiebung des Engpasses vom Client- zum Kommunikationssystem und den Einsatz breitbandiger Transportnetze ein Streaming von 3D-Szenen mit genügend kurzen Zeitintervallen zwischen den jeweiligen Szenen-Aktualisierungen ermöglicht.

Dies wurde mit der Ausspielung interaktiv räumlich und zeitlich navigierbarer „3D-Filme“ mittels leistungsfähigem Server (Silicon Graphics Origin200), Kommunikationsnetz (IP über Gigabit-Ethernet) und 3D-Graphikworkstation (Silicon Graphics Onyx2 Infinite Reality) demonstriert. Sequenzen von Beispielszenen mit je ca. 100.000 attributierten Polygonen konnten bei einer mittleren Datenrate von ca. 300 Mbit/s mit ca. 5 Szenenwechseln pro Sekunde dargestellt werden.

Somit wurde ein deutlicher Beitrag zur hochqualitativen Online-Präsentation diskreter und kontinuierlicher dreidimensionaler visueller Medien geleistet.

Das entwickelte System eignet sich z. B. hervorragend zur Unterstützung der 3D-Visualisierung komplexer, mehrdimensionaler und zeitabhängiger Ergebnisse von Simulationsrechnungen.

Weitere Fortschritte können durch die Bereitstellung leistungsfähigerer Ressourcen erzielt werden. Die derzeit im Pilotstadium befindlichen Gigabit-Netze sowie High-End-Visualisierungssupercomputer sind dazu einsetzbar. Außerdem kann der Transport von 3D-Medien durch den Einsatz fortgeschrittener Protokolle verbessert werden. Hier sind insbesondere Verfahren zur Reservierung von Dienstgüten (ATM bzw. RSVP) sowie „Reliable Muticast"-Transportprotokollen [129] zur Unterstützung der Gruppenkommunikation zu nennen. Aufgrund der bislang nur serverseitig bewirkten Synchronisierung führen Schwankungen der Dienstgüteparameter des Transportsystems (z. B. Latenz, Datenrate) zu variierenden Bildraten. Zur Glättung ist eine clientseitige Pufferung – eventuell in Kombination mit Regelungsmechanismen – als Erweiterungsmaßnahme anzustreben.

Darüber hinaus lassen sich durch Ergänzungen des im Rahmen dieser Arbeit primär unterstützten *Präsentationsszenarios* weitere Anwendungsszenarien entwikkeln. Diese könnten z. B. auf innovativen Weitverkehrsnetz-Infrastrukturen zur verteilten Visualisierung von Ergebnissen des Höchstleistungsrechnens angewandt werden:

Explorationsszenario

Es handelt sich hier um eine Arbeitsumgebung mit interaktiver, bidirektionaler Kopplung von Simulations-, Visualisierungs-, Präsentations- und Interaktionsmoduln, die z. B. zur Durchführung virtueller Experimente geeignet ist. Zur Erzielung eines hohen Interaktionsgrades ist die Generierung der 3D-Geometrien in den Datenfluß so zu integrieren, daß der 3D-Streamingserver einen zusätzlichen 3D-Dateneingang erhält, welcher z. B. direkt aus den Visualisierungsergebnissen eines Simulationsrechenprozesses gespeist werden kann. Simulations- und Visualisierungsparameter sollten dabei mit Hilfe einer erweiterbaren Nutzeroberfläche des Clients über einen Kontrolldatenfluß (z. B. über RTSP) interaktiv gesteuert werden können.

Diskussionsszenario

Hier soll eine kooperative Nutzung von Präsentations- oder Explorationsszenarien durch geographisch verteilte Gruppen (Computer-Supported Collaborative Work – CSCW) ermöglicht werden. Dazu müssen Synchronisationsmechanismen (z. B. Navigation, Telepointer und Streamingsteuerung) sowie Conferencing-Komponenten (z. B. Einblendung des Videobildes als animierte 2D-Textur in der 3D-Szene) in das 3D-Streamingsystem integriert werden.

Literaturverzeichnis

1. Adobe Systems Inc.: *Tagged Image File Format (TIFF) Specification, Revision 6.0*, 1993.
2. Akeley, K., Haeberli, P., Burns, D.: *tomesh.c.* C Program on SGI Developer Toolbox CD, 1990.
3. Andrews, K., Pesendorfer, A., Pichler, M., Wagenbrunn, K. H., Wolte, J.: *Looking inside VRwave: The Architecture and Interface of the VRwave VRML97 Browser.* Proceedings of VRML '98 Symposium, Monterey, California, 1998. (http://www2.iicm.edu/keith/publications.html)
4. Astheimer, P, Böhm, K., Felger, W., Göbel, M., Müller, S.: *Die Virtuelle Umgebung – Eine neue Epoche in der Mensch-Maschine-Kommunikation*. Teil 1: Einordnung, Begriffe und Geräte, Teil 2: Interaktions- und Präsentationstechniken, Systeme, Anwendungen. Informatik-Spektrum, Springer, Band 17, Hefte 5 und 6, Oktober und Dezember 1994.
5. Astheimer, P., Felger, W., Müller, S.: *Virtual Design – A Generic VR System for Industrial Applications*. Computers & Graphics, Vol. 17, No. 6, 1993.
6. Astheimer, P., Dai, F. Felger, W., Göbel, M., Haase, H., Ziegler: *Virtual Design II – An Advanced VR System for Industrial Applications.* Virtual Reality World 95, Februrar 1995.
7. AVS: *Animating AVS Data Visualizations*. CONVEX Press, 1992.
8. Balaguer, J.-F., Gobbetti, E.: *i3D. A High-Speed 3D Web Browser.* Proceedings of VRML '95 Symposium, San Diego, 1995. (http://www.crs4.it/~gobetti/)
9. Bangemann, M., et al: *Bericht der EU-Kommission zur Informationsgesellschaft*. 1994.
10. Bell, G., Parisi, A., Pesce, M.: *The Virtual Reality Markup Language – Version 1.0 Specification*. 09.11.1995.
11. von Berg, A.: *Gigabit Ethernet Performance Tests on SGI Machines – TCP Bulk Data Transfer with and without Jumbo Frame Option.* Technical Report. Universität Hannover, Regionales Rechenzentrum für Niedersachsen (RRZN) / Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), 1999.
12. Berners-Lee, T., Connolly, D.: *Hypertext Markup Language – HTML 2.0*. RFC 1866, November 1995.
13. Berners-Lee, T., Masinter, L., McCahill, M.: *Uniform Resource Locators (URL)*. RFC 1737, December 1994.

14. Biersack, E. W.: *Hochleistungs-Transportprotokolle*. Informationstechnik und Technische Informatik, Vol. 35, Nr. 4, 1993.
15. Black, A., Boag, A.: *Choosing Binary or Greyscale Bitmaps – Some Consequences for Users*. Proceedings of Electronics Publishing (EP 92), 1992.
16. BMWi (Bundesministerium für Wirtschaft): *Die Informationsgesellschaft – Fakten, Analysen, Trends*. BMWi Report, 1995.
17. BMWi (Bundesministerium für Wirtschaft): *Info 2000 – Deutschlands Weg in die Informationsgesellschaft*. Bericht der Bundesregierung, 1996.
18. Bobick, N.: *Rotating Objects Using Quaternions*. Game Developer, Vol. 2, No. 26, July 1998.
(http://www.gamasutra.com/features/programming/19980703/quaternions_01.htm)
19. Brockners, F., Zier, L.: *Ab in die Wüste – TCP-Tuning für schnelle Netze*.
iX Magazin für professionelle Informationstechnik, Juli 1997.
20. Brodlie, K. W., et al (Eds.): *Scientific Visualization – Techniques and Applications*. Springer, 1992.
21. Broll, W.: *DWTP – An Internet Protocol for Shared Virtual Environments*. Proc. of VRML '98 Symposium, Monterey, California, 1998.
(http://ece.uwaterloo.ca/vrml98/papers/content.html)
22. Brown, M.: *CAVERN: The CAVE Research Network*. TERENA-NORDUnet Networking Conference, Lund, Sweden, 1999.
(http://www.terena.nl/tnnc/2A/2A2/2A2.ppt)
23. Brown, J. R., Earnshaw, R., Jern, M., Vince, J.: *Visualization – Using Computer Graphics to Explore Data and Present Information*. John Wiley & Sons, 1995.
24. Brutzman, D., Zyda, M., Watsen, K., Macedonia, M.: *Virtual Reality Transfer Protocol (vrtp) Design Rationale.* Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises: Sharing a Distributed Virtual Reality, MIT, 1997. (http://www.stl.nps.navy.mil/~brutzman/vrtp/vrtp_design.ps)
25. Bryson, S., Levit, C.: *The Virtual Windtunnel*. IEEE Computer Graphics and Applications, Vol. 12, No. 4, 1992.
26. Burger, R. E.: *Colormanagement*. Springer, 1997.
27. Campagna, S., Kobbelt, L., Seidel, H.: *Enhancing Digital Documents by including 3D-Models.* Computers & Graphics, Vol. 22, No. 6, 1998.
28. Chen, S., Pu, C., Staehli, R., Cowan, C., Walpole, J.: *A Distributed Real-Time MPEG Video Audio Player.* Proceedings of NOSSDAV (International Workshop on Network and Operating System Support for Digital Audio and Video) 1995. Lecture Notes in Computer Science, Vol. 1018, Springer, 1995.
(http://www.cse.ogi.edu/~scen/)
29. Chow, M.: *Optimized geometry compression for real-time rendering*. Proceedings of IEEE Visualization 1997.
(http://home.earthlink.net/~mmchow/gcompiler/gcompiler.html)

30. Claussen, U.: *Programmieren mit OpenGL*. Springer, 1997.
31. Corrie, B., Sitsky, D., Mackerras, P.: *Integrating High Performance Computing and Virtual Environments*. Proceedings of the Seventh Parallel Computing Workshop, Canberra, Australia, September 1997. (http://cap.anu.edu.au/cap/projects/parvis/bibliography/BCDSPM97.ps.gz)
32. Cruz-Neira, C., Sandin, D. J., DeFanti, T. A.: *Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE*. Proceedings of ACM SIGGRAPH 1993.
33. Deering, M.: *Geometry compression*. Proceedings of ACM SIGGRAPH 1999.
34. DIN 40146-1: *Begriffe der Nachrichtenübertragung – Grundbegriffe*. Beuth, 1994.
35. Doenges, P. K., Capin, T., K., Lavagetto, F., Ostermann, J., Pandzic, I. S., Petajan, E., E.: *MPEG-4: Audio/Video & Synthetic Graphics/Audio for Mixed Media*. Image Communication Journal, Vol. 5, No. 4, May 1997.
36. Dunwoody, C.: *The OpenGL Stream Codec: A Specification, Version 0.0*. SGI, 1996. (http://reality.sgi.com/employees/dunwoody/glsspec.txt)
37. Earnshaw, R. A., Wiseman, N.: *An Introductory Guide to Scientific Visualization*. Springer, 1992.
38. Effelsberg, W., Steinmetz, R.: *Video Compression Techniques*. dpunkt-Verlag, 1998.
39. Einhorn, R.: *Entwicklung einer Online-Preview-Komponente für die Videoproduktion in Breitbandnetzen*. Studienarbeit, Lehrgebiet Rechnernetze und Verteilte Systeme, Universität Hannover, 1996.
40. Encarnacao, J., Frühauf, M.: *Global information visualization: the visualization challenge for the 21st century.* In [158].
41. Engel, K., Gross, R., Ertl, T.: *Progressive Iso-surfaces on the Web.* Proceedings of IEEE Visualization 1998.
42. Evans, F., Skiena, S. S., Varshney, A.: *Optimizing Triangle Strips For Fast Rendering*. Proceedings of IEEE Visualization 1996. (http://www.cs.sunysb.edu/~stripe/stripeVIS.pdf)
43. Evans, F., Skiena, S. S., Varshney, A.: *STRIPE: A Software Tool for Efficient Triangle Strips.* Proceedings of ACM SIGGRAPH 1996. (http://www.cs.sunysb.edu/~stripe/stripeSIGGRAPH.html)
44. Felger, W.: *Innovative Interaktionstechniken in der Visualisierung*. Springer, 1995.
45. Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2068, January 1997.
46. Foley, J. D., van Dam, A., Feiner, S. K.: *Computer Graphics – Principles and Practice.* Addison-Wesley, 2nd edition, 1990.

47. Freed, N., Borenstein, N.: *Multipurpose Internet Mail Extensions (MIME)*. RFC 2045, November 1996.
48. G7 Business Round Table: *Building a Global Information Society.* 1995.
49. Garland, M., Heckbert, P. S.: *Surface Simplification Using Quadric Error Metrics*. Proceedings of ACM SIGGRAPH 1997. (http://www.cs.cmu.edu/~garland/quadrics.html)
50. Göbel, M., Mehl, M.: *Standards der graphischen Datenverarbeitung*. Expert Verlag, 1989.
51. Guéziec, A., Taubin, G., Horn, B., Lazarus, F.: *A Framework for Streaming Geometry in VRML*. IEEE Computer Graphics and Applications, Vol. 19, No. 2, 1999.
52. Gulbins, J., Seyfried, M., Strack-Zimmermann, H.: *Dokumenten-Management.* Springer, 1999.
53. Gumhold, S., Straßer, W.: *Real Time Compression of Triangle Mesh Connectivity.* Proceedings of ACM SIGGRAPH 1998.
54. Gropp, W., Lusk, E., Skjellum, A.: *Using MPI – Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1994.
55. Haase, H.: *Symbiosis of Virtual Reality and Scientific Visualization System*. Computer Graphics Forum, Vol. 15, No. 3, 1996.
56. Haase, H., Dai, F., Strassner, J., Göbel, M.: *Immersive Investigation of Scientific Data*. In [127].
57. Haeberli, P., Akeley, K.: *The Accumulation Buffer – Hardware Support for High-Quality Rendering.* Proceedings of ACM SIGGRAPH 1990.
58. Hardenberg, J.: *RE: QvLib questions*. VRML Hypermail Archive, 27.03.1995. (http://vag.vrml.org/www-vrml/arch/1107.html)
59. Has, M., Newman, T.: *Color Management – Current Practice and the Adoption of a New Standard*. 1996. (http://www.color.org/overview.html)
60. Heinrichs, B.: *Multimedia im Netz*. Springer, 1996.
61. Heller, D., Ferguson, P. M.: *Motif Programming Manual for OSF/Motif Release 1.2*. O'Reilly, 1994.
62. Hennig, A.: *Die andere Wirklichkeit – Virtual Reality – Konzepte, Standards, Lösungen*. Addison-Wesley, 1997.
63. Hess, R., Hutschenreuther, T., Schill, A.: *Video Communication and Media Scaling System „Xnetvideo": Design and Implementation*. In: Butscher, B., Moeller, E., Pusch, H. (Ed.): Interactive Distributed Multimedia Systems and Services. Proceedings of European Workshop IDMS '96, March 1996. Lecture Notes in Computer Science, Vol. 1045, Springer, 1996. (http://mephisto.inf.tu-dresden.de/RESEARCH/SFB358/paper-berlin.ps)
64. Hodges, L. F.: *Time-multiplexed Stereoscopic Computer Graphics*. IEEE Computer Graphics and Applications, Vol. 12, No. 2, 1992.

65. Hoppe, H.: *Efficient implementation of progressive meshes*. Computers and Graphics, Vol. 22, No. 1, 1998.
66. Hoppe, H.: *Optimization of Mesh Locality for Transparent Vertex Caching*. ACM Computer Graphics, Proceedings of ACM SIGGRAPH 1999.
67. Hoppe, H.: *Progressive Meshes*. Proceedings of ACM SIGGRAPH 1995.
68. Hornung, C., Santos, A.: *A Proposal for a Reference Model for Cooperative HyperMedia Systems*. In: Kjelldahl, L. (Ed.): *Multimedia*, Springer, 1992.
69. Huffman: *A Method for the Construction of Minimum Redundancy Codes*. Proceedings IRE, No. 40, 1952.
70. ICC.1: *ICC Profile Specification*. International Color Consortium, 1998. (http://www.color.org/ICC-1_1998-09.PDF)
71. IEC 61966-2.1: *Default RGB colour space – sRGB*. 3rd Working Draft, 1998. (http://www.srgb.com/sRGBstandard.pdf)
72. Ihrén, J., Frisch, K. J.: *The Fully-Immersive CAVE*. Proceedings of 3. International Immersive Projection Technology Workshop, 1999.
73. Intel Corp., MetaCreations Corp.: *MetaStream 3-D File Format*. File Format Version 2.0, Document Version 1.0, 1999. (www.metacreations.com)
74. InterSense Inc.: *IS-600 Mark II*. User Manual, 1999.
75. ISO/IEC 7498-1: *Open Systems Interconnection – Basic Reference Model: The Basic Model*, 1994.
76. ISO/IEC 7942: *Graphical Kernel System (GKS)*, 1985.
77. ISO/IEC 8632: *Metafile for the storage and transfer of picture description information* (8632-1: Functional specification, 8632-2: Character encoding, 8632-3: Binary encoding, 8632-4: Clear text encoding), 1992.
78. ISO/IEC 8805: *Graphical Kernel System for Three Dimensions (GKS-3D)*, 1988.
79. ISO/IEC 8879: *Standard Generalized Markup Language (SGML)*, 1979.
80. ISO/IEC 9592: *Programmer's Hierarchical Interactive Graphics System (PHIGS)*, 1989.
81. ISO/IEC 9592-2: *Programmer's Hierarchical Interactive Graphics System (PHIGS) – Part 2: Archive file format*, 1989.
82. ISO/IEC 9636: *Interfacing Techniques for Dialogues with Graphical Devices (Computer Graphics Interface, CGI)*, 1991.
83. ISO/IEC 10744: *Hypermedia/Time-based Structuring Language (HyTime)*, 1992.
84. ISO/IEC 11072: *Computer Graphics Reference Model*, 1992.
85. ISO/IEC 13522-1: *Information Technology – Coded Representation of Multimedia and Hypermedia Information Objects (MHEG)*, 1997.
86. ISO/IEC 14772-1: *The Virtual Reality Modeling Language (VRML97) – Part 1: Functional specification and UTF-8 encoding*, 1997.

87. ISO/IEC 14772-3: *The Virtual Reality Modeling Language (VRML97) – Part 3: Compressed Binary Format Specification.* Editor's Draft 5, 1997.
88. ITU: *Recommendation 601-2: Encoding Parameters of Digital Television for Studios*. ITU Genf, November 1991.
89. ITU: *Recommendation I.211: Integrated Services Digital Network (ISDN) Service Capabilities – B-ISDN Service Aspects*. ITU Genf, März 1993.
90. Jacobson, V.: *Congestion avoidance and control*. Proceedings of ACM SIGCOMM 1998.
91. Jacobson, V., Braden, R., Borman, D.: *TCP Extensions for High Performance*. RFC 1323, May 1992.
92. Johnson, C., Parker, S. G., Hansen, C., Kindlmann, G. L., Livnat, Y.: *Interactive Simulation and Visualization.* IEEE Computer, Vol. 32, No. 12, December 1999.
93. Jung, T.: *Remote Control of Virtual Environments via Low-Bandwidth Connections*. Proceedings of High-Performance Computing and Networking Conference (HPCN) 1998, Springer, 1998.
94. Kilgard, M. J.: *GLR, an OpenGL render server facility.* 1996 (www.sgi.com).
95. Kilgard, M. J.: *OpenGL Programming for the X Window System*. Addison-Wesley, 1996.
96. Klein, R.: *Multiresolution representations for surface meshes*. Proceedings of SCCG, 1997.
97. Klein, R., Straßer, W.: *Handling of Very Large 3D-Surface-Datasets Using Mesh Simplification and Multiresolution Modeling*. Tutorial, Computer Graphics International 1998, Hannover, Germany, June 22–26, 1998.
98. Kobbelt, L., Campagna, S., Seidel, H.: *A General Framework for Mesh Decimation*. Graphics Interface Proceedings, Vancouver, BC, 18–20 June 1998.
99. Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.: *Interactive Multi-Resolution Modeling on Arbitrary Meshes.* Proceedings of ACM SIGGRAPH 1998.
100. Krömker, D.: *Visualisierungssysteme*. Springer, 1992.
101. Krüger, W., Bohn, C.-A., Fröhlich, B., Schüth, H., Strauß, W., Wesche, G.: *The Responsive Workbench: A Virtual Work Environment*. IEEE Computer, Vol. 28, No. 7, July 1995.
102. Kuhlen, T. et al: *Ziel erfaßt! Tracking-System-Test*. In: 3D Live, IWT, 2/99.
103. Lane, D. A.: *Scientific Visualization of Large-Scale Unsteady Fluid Flows*. In [127].
104. Leavitt, N.: *Online 3D: Still Waiting After All These Days*. IEEE Computer, Vol. 32, No. 7, July 1999.
105. Leigh, J., Johnson, A. E., Brown, M., Sandin, D. J., DeFanti, T. A.: *Visualization in Teleimmersive Environments.* IEEE Computer, Vol. 32, No. 12, December 1999.

106. Lengyel, J.: *Compression of time-dependent geometry.* Proceedings of ACM Symposium on Interactive 3D Graphics 1999. (http://research.microsoft.com/~jedl/)

107. Leonhardt, C.: *Zur Anwendung von Streamingverfahren auf die Online-Präsentation von Sequenzen virtueller 3D-Objekte.* Diplomarbeit, Universität Hannover, Regionales Rechenzentrum für Niedersachsen (RRZN) / Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), 1998.

108. Li, J., Li, J., Kuo, C. C.: *Progressive compression of 3D graphics models*. Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS) 1997. (http://church.computer.org/proceedings/icmcs/7819/78190135abs.htm)

109. Lie, H., Bender, W.: *The Electronic Broadsheet: all the News that fit the Display.* Proceedings of Electronics Publishing (EP 92), 1992.

110. Macedonia, M., Zyda, M.: *A Taxonomy for Networked Virtual Environments*. IEEE Multimedia, Vol. 4, No. 1, January–March 1997.

111. McCormick, B. H., DeFanti, T. A., Brown, M. D. (Eds.): *Visualization in Scientific Computing*. ACM Computer Graphics, Vol. 21, No. 6, 1987.

112. McLuhan, M.: *The Gutenberg Galaxy.* 1962.

113. Microsoft: *Color Management in Microsoft Windows Operating Systems – An Overview of Microsoft Color Management Technology.* White Paper, April 1997. (http://www.microsoft.com/Windows/platform/icmwp.htm)

114. Microsoft: *DirectX Web site*. (http://www.microsoft.com/DirectX)

115. Microsoft: *DCOM Technical Overview.* White Paper, November 1996. (http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomtec.htm)

116. Microsoft: *ICM 2.0 SDK Web site*. (http://www.microsoft.com/msdn/sdk/icm20.htm)

117. Microsoft: *TrueType 1.0 Font Files*. Technical Specification, Revision 1.66, November 1995.

118. Mooshake, S.: *Entwicklung einer Netzplattform für ein VR-System.* Diplomarbeit, Universität Hannover, FG Graphische Datenverarbeitung des Instituts für Informatik in Kooperation mit: Regionales Rechenzentrum für Niedersachsen (RRZN) / Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), VR-Labor der Volkswagen AG, 1999.

119. Nebel, E., Masinter, L.: *Form-based File Upload in HTML*. RFC 1867, 07.11.1995.

120. Neidecker-Lutz, B. K., Ulichney, R.: *Software Motion Pictures*. Digital Technical Journal, Vol. 5, No. 2, Spring 1993.

121. Neider, J., Davis, T., Woo, M.: *OpenGL Programming Manual: The official guide to learning OpenGL, Release 1*. Addison-Wesley, 1993.

122. Neider, J., Davis, T., Woo, M.: *OpenGL Reference Manual: The official reference document for OpenGL, Release 1*. Addison-Wesley, 1992.

123. Netravali, A. N., Haskell, B. G.: *Digital Pictures*. Plenum Press, 1988.

124. Netscape: *Netscape Navigator LiveConnect/Plug-In Software Development Kit*. 1998.
(http://home.netscape.com/comprod/development_partners/plugin_api/)

125. Netscape: *Plug-In Guide – Communicator 4.0*. January 1998.
(http://developer.netscape.com/docs/manuals/communicator/plugins/)

126. Niederberger, R., Grund, H., Pless, E., Hommes, F.: *High speed Supercomputer communications in Broadband Networks.* TERENA-NORDUnet Networking Conference, Lund, Sweden, 1999.
(http://www.terena.nl/tnnc/9B/9B3/9B3.html)

127. Nielson, G. M., Hagen, H., Müller, H.: *Scientific Visualization. Overviews, Methodologies, Techniques.* IEEE Computer Society, 1997.

128. Nocula, C., Olbrich, S.: *Zur Diskrepanz der Auflösung verschiedener Präsentationsmedien.* Studie, Universität Hannover, Regionales Rechenzentrum für Niedersachen (RRZN) / Lehrgebiet Rechnernetze und Verteilte Systeme (RVS). In: [131].

129. Obraczka, K.: *Multicast Transport Mechanisms: A Survey and Taxonomy.* IEEE Communications, January 1998.

130. Olbrich, S., Pralle, H., Grüneberg, L.: *Anforderungen an eine Client/Server-Konfiguration für das „Online-Publishing“*. In: Müller-Schloer, C. (Hrsg.): Arbeitsplatz-Rechensysteme, 3. GI/ITG-Fachtagung, 1995.

131. Olbrich, S., Pralle, H., et al: *Regionales Testbed Nord, Projekt P5.1 – Online-Dokumente, Abschlußbericht.* Universität Hannover, Regionales Rechenzentrum für Niedersachen (RRZN) / Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), 04.04.1997.
(http://www.rtb-nord.uni-hannover.de/onlinedokumente/)

132. Olbrich, S., Pralle, H.: *Verfilmung von Multimediadaten in einem verteilten System*. Informationstechnik und Technische Informatik, Vol. 39, Nr. 1, 1997.

133. Olbrich, S., Pralle, H.: *High-Performance Online Presentation of Complex 3D Scenes*. In: van As, H. R. (Ed.): High Performance Networking – IFIP TC-6 Eighth International Conference on High Performance Networking (HPN ’98), Vienna, Austria, September 1998. Kluwer Academic Publishers, 1998.
(http://www.dfn-expo.de/Technologie/DocShow-VR/Vortraege/hpn98/)

134. Olbrich, S., Pralle, H.: *Multimedia- und Virtual-Reality-Technologie für die wissenschaftliche Visualisierung.* In: Buziek, G. et al (Hrsg.): Kartographische Animationen – Ein Leitfaden für die Praxis. Springer-Verlag, 2000.
(in Vorbereitung)

135. Olbrich, S.: *Hochwertige Online-Präsentation virtueller 3D-Exponate aus dem Wissenschaftsbereich – Anforderungen und Lösungsansätze*, Vortrag auf dem DFN-Arbeitkreis Informationsdienste, Hannover, 24.09.1997.

136. Olbrich, S.: *Verteilte Nutzung von 3D/VR-Streamingverfahren zur Präsentation und Diskussion komplexer Ergebnisse*. Workshop „Wissenschaftliche Anwendungen auf Höchstleistungsrechnern“, RRZN, Universität Hannover, 29.09.1999.

137. Olbrich, S., Pralle, H.: *Virtual Reality Movies – Real-Time Streaming of 3D Objects*. TERENA-NORDUnet Networking Conference, Lund, Sweden, 1999. (http://www.terena.nl/tnnc/2A/2A1/2A1.pdf)
(wird erscheinen in: Computer Networks, Elsevier Science, 2000)

138. Olbrich, S., Reumann, K.: *Video Animation of Plot Sequences in a Computer Center Environment*. Proceedings of COMPUGRAPHICS 1991.

139. Oliphant, Z.: *Programming Netcape Plug-Ins*. Sams.net Publishing, 1996.

140. OPC – The OpenGL Performance Characterization Projekt: *Viewperf Information and Results*.
(http://www.specbench.org/gpc/opc.static/viewin~1.html)

141. Open Software Foundation: *OSF/Motif Programmer's Guide, Programmer's Reference, Style Guide, Revision 1.2*. Prentice Hall, 1993.

142. Parisi, T.: *Corrections: Keeping Web 3D on Course*. Keynote address at the VRML '99 Symposium, Paderborn, 1999.
(http://www.c-lab.de/vrml99/keynotes.htm#keynote1,
http://www.nyvrmlsig.org/news/messages/97.html)

143. Paul, B.: *The Mesa 3-D graphics library.*
(http://www.ssec.wisc.edu/~brianp/Mesa.html)

144. Pennebaker, W. B., Mitchell, J. L.: *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.

145. Pesce, M., Kennard, P., Parisi, A.: *Cyberspace*. First WWW Conference, May 1994. (http://www.hyperreal.org/~mpesce/www.html)

146. Pfuhl, W.: *Zur Bewertung des Datenfluß- und Online-Präsentations-Verhaltens digitaler Repräsentationen von 3D-Objekten*. Diplomarbeit, Universität Hannover, Regionales Rechenzentrum für Niedersachsen (RRZN) / Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), 1997.

147. Polhemus Inc.: *3Space Fastrak User's Manual*. Revision F, November 1993.

148. Postel, J., Reynolds, J.: *File Transfer Protocol (FTP)*. RFC 959, October 1985.

149. Poynton, C.: *A Technical Introduction to Digital Video – Chapter 6: Gamma*. Wiley, 1996.
(http://www.inforamp.net/~poynton/PDFs/TIDV/Gamma.pdf)

150. Poynton, C.: *Gamma FAQ – Frequently-Asked Questions about Gamma.*
(http://www.inforamp.net/~poynton/GammaFAQ.html)

151. Poynton, C.: *The Rehabilitation of Gamma*. In: Rogowitz, R. E., Pappas, T. N. (Eds.), Human Vision and Electronic Imaging III, Proceedings of SPIE 3299, Bellingham, 1998.
(http://Home.InfoRamp.Net/~poynton/papers/IST_SPIE_9801/)

152. Raasch, S., Schröter, M., Ketelsen, K., Olbrich, S.: *A Large-Eddy Simulation Model for Massively Parallel Computers – Model Design and Scalability.* Fifth European SGI/Cray MPP Workshop, Bologna (Italy), September 9–10, 1999. (http://www.cineca.it/mpp-workshop/abstract/sraasch.htm)

153. Raasch, S., Etling, D.: *Modeling Deep Ocean Convection: Large Eddy Simulation in Comparison with Laboratory Experiments*. Journal of Physical Oceanography, Vol. 28, 1998.

154. Rantzau, D., Rühle, R.: *Architektur und Anwendungen einer modularen und skalierbaren VR-Umgebung für technisch-wissenschaftliche Simulation und Visualisierung*. Proceedings GI-Workshop „Visualisierung im Engineering“ (VisEng ’97), Stuttgart, 10.–11. Mai 1997.
(http://www.hlrs.de/people/rantzau/viseng97_rantzau.ps)

155. Reitmayr, G., Carroll, S., Reitemeyer, A., Wagner, M. G.: *DeepMatrix – An open technology based virtual environment system*. The Visual Computer, Vol. 15, No. 7/8, 1999. (http://link.springer.de/link/service/journals/00371/)

156. Rieken, B., Weiman, L.: *Adventures in UNIX Networking Applications Programming.* Wiley, 1992.

157. Riempp, R., Schlotterbeck, A.: *Digitales Video in interaktiven Medien*. Springer, 1995.

158. Rosenblum, L., Earnshaw, R. A., Encarnacao, J., Hagen, H., Kaufmann, A., Klimenko, S., Nielson, G., Post, F., Thalmann, D.: *Scientific Visualization – Advances and Challenges.* Academic Press, 1994.

159. Rust, L. F., de Saqui-Sannes, P., Courtiat, J.-P.: *Basic Synchronisation Concepts in Multimedia Systems*. Proceedings of NOSSDAV (International Workshop on Network and Operating System Support for Digital Audio and Video) 1992. Lecture Notes in Computer Science, Vol. 712, Springer, 1993.

160. Scherzer, Y.: *Virtuelle Welten – Navigation in virtuellen Räumen.* DFN-Mitteilungen, Nr. 41, Juni 1996.

161. Schroeder, W., Martin, K., Lorensen, B.: *The Visualization Toolkit.* 2nd Edition. Prentice Hall, 1997. (siehe auch http://www.kitware.com/vtk.html)

162. Schroeder, W.: *Polygon Reduction Techniques*. IEEE Visualization 1995.

163. Schulze, S.: *Durchblick in 3D – Die 3D-Bibliotheken Direkt3D und OpenGL*. c’t Magazin für Computer-Technik, Heft 19 / 1999.

164. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: *RTP: A Transport Protocol for Real-Time Applications.* RFC 1889, January 1996.

165. Schulzrinne, H., Rao, A., Lanphier, R.: *Real Time Streaming Protocol (RTSP)*. RFC 2326, 14.04.1998.

166. Shneiderman, B.: *Response Time and Display Rate in Human Performance with Computers.* ACM Computing Surveys, Vol. 16, Nr. 3, 1984.
167. Signès, J., Cazoulat, R., Pelé, D., Le Mestre, G.: *An MPEG-4 Player integrating 3D, 2D graphics, video and audio*. International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging 1997.
168. Silicon Graphics, Inc.: *Coloratura Programmer's Guide*. 1998.
169. Silicon Graphics, Inc.: *Onyx2 Reality and Onyx2 Infinite Reality.* Technical Report, 1996.
170. Silicon Graphics, Inc.: *Origin Servers*. Technical Report, 1997.
171. Silicon Graphics, Inc.: *Topics in IRIX Programming*. 1999.
172. Simon, R. J.: *Windows 95 Win32 Programming API Bible*. Waite, 1996.
173. Singhal, S., Zyda, M.: *Networked Virtual Environments – Design and Implementation*. Addison-Wesley, 1999.
174. Socolofsky, T., Kale, C.: *A TCP/IP Tutorial*. RFC 1180, January 1991.
175. Spero, S. E.: *Analysis of HTTP Performance problems*. July 1994. (http://www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html)
176. Staehli, R., Walpole, J.: *Constraint-Latency Storage Access.* IEEE Computer, Vol. 26, Nr. 3, March 1993.
177. Steinmetz, R.: *Multimedia-Technologie: Einführung und Grundlagen*. Springer, 1993.
178. StereoGraphics Corp.: *CrystalEyes Software Development Kit. A StereoGraphics Product*. Updated: 01.12.1997. (http://www.stereographics.com/support/developers/ce_sdk.pdf)
179. Stevens, W.: *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*. RFC 2001, January 1997.
180. Stevens, W. R.: *UNIX Network Programming, Volume 1 – Networking APIs*. Prentice Hall, 1998.
181. Stokes, M., Anderson, M., Chandrasekar, S., Motta, R.: *A Standard Default Color Space for the Internet – sRGB.* Version 1.10, November 5, 1996. (http://www.color.org/sRGB.html)
182. Strauss, P., Bell, G.: *The VRML Programming Library – QvLib, Version 1.0 beta 1*. 1995. (http://vag.vrml.org/www-vrml/vrml.tech/qv.html)
183. Sun Microsystems, Inc.: *The Java 3D API*. Technical White Paper. July 1997. (http://java.sun.com/products/java-media/3D/collateral/j3d_api/j3d_api_1.html)
184. Sun Microsystems, Inc.: *KCMS Application Developer's Guide*. 1997. (http://docs.sun.com/)
185. Sutherland, I. E.: *The Ultimate Display.* IFIP Congress, 1965.
186. Tanenbaum, A. S.: *Computer-Netzwerke*. Wolfram's Fachverlag, 1990.
187. Taubin, G., Gueziec, A., Horn, W., Lazarus, F.: *Progressive Forest Split Compression*. Proceedings of ACM SIGGRAPH 1998.

188. Taubin, G., Rossignac, J.: *Geometric Compression Through Topologic Surgery.* ACM Transactions on Graphics, Vol. 17, No. 2, April 1998.

189. Terrence, L., Papka, M. E., Pellegrino, M., Stevens, R.: *Sharing Visualization Experiences among Remote Virtual Environments*. In: Chen, M., Townsend, P., Vince, J. A. (Eds.): *High Performance Computing for Computer Graphics and Visualization*, Springer, 1996.

190. Touma, C., Gotsman, C.: *Triangle Mesh Compression*. Proceedings of Graphics Interface 1998.

191. Upson, C., Faulhaber, T., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R., van Dam, A.: *The Application Visualization System: A Computational Environment for Scientific Visualization*. IEEE Computer Graphics and Applications, Vol. 9, No. 4, July 1989.

192. Vester, F.: *Denken, Lernen, Vergessen*. Deutscher Taschenbuch Verlag (dtv), 25. Auflage, November 1998.

193. Vince, J.: *Virtual Reality Systems*. Addison-Wesley, 1995.

194. Vöckler, J.-S.: *A quick glance at webfilt*. Universität Hannover, Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), 03.09.1997.

195. W3C: *Cascaded Style Sheets, level 1*. W3C Recommendation, 11.01.1999.
(Latest version: http://www.w3.org/WWW/TR/REC-CSS1)

196. W3C: *Extensible Markup Language (XML) 1.0 Specification*. W3C Recommendation, 10.02.1998.
(Latest version: http://www.w3.org/TR/REC-xml)

197. W3C: *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification.* W3C Recommendation, 15.06.1998.
(Latest version: http://www.w3.org/TR/REC-smil/)

198. Walnum, C.: *3-D Graphics Programming with OpenGL*. Que, 1995.

199. Web 3D Consortium: *The VRML Repository.* (http://www.web3d.org/vrml/).

200. Wernecke, J.: *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor.* Release 2, Addison-Wesley, 1994.

201. Wijk, van J., Liere, van R.: *An Environment for Computational Steering*. In [127].

202. Wright, R. S., Sweet, M.: *OpenGL superbible – The complete guide to OpenGL programming for Windows NT and Windows 95*. Waite, 1996.

203. Xiang, X., Held, M., Mitchell, J.: *Fast and effective stripification of polygonal surface models.* Proceedings of ACM Interactive 3D Graphics Symposium 1999.
(http://www.ams.sunysb.edu/~jsbm/publications.html)

Lebenslauf

Stephan Olbrich

Persönliche Daten

31.03.1961	geboren in Braunschweig Eltern: Helga Olbrich (geb. Kropp), Johannes Olbrich
seit 1988	verheiratet mit Susanne Olbrich (geb. Wittkopp)
März 1989	Geburt des Sohnes Lutz
März 1992	Geburt des Sohnes Anton

Ausbildung und Studium

1967–1971	Grundschule Haltenhoffstraße, Hannover
1971–1980	Gymnasium Goetheschule, Hannover Abschluß: Abitur
1980	Grundpraktikum: Siemens AG, Hannover
1980–1987	Universität Hannover Studium Elektrotechnik / Nachrichtentechnik mit Vertiefungsrichtung Nachrichtenverarbeitung Abschluß: Dipl.-Ing.

Berufstätigkeit

1981–1987	Wissenschaftliche Hilfskraft Betreuung der Vorlesungen von Prof. Dr.-Ing. H. Pralle Regionales Rechenzentrum für Niedersachsen (RRZN), Universität Hannover
1985–1987	Fachpraktikum, Hard- und Software-Entwicklung altec electronic GmbH, Hannover
1987–1989	Entwicklungsingenieur altec electronic GmbH, Hannover
seit 1989	Wissenschaftlicher Mitarbeiter in der Gruppe Graphische Datenverarbeitung Regionales Rechenzentrum für Niedersachsen (RRZN), Universität Hannover
seit 1993	Mitarbeit im Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), Universität Hannover Leitung von DFN-Projekten mit Förderung des BMBF: • Regionales Testbed (RTB) Nord (1994–1996) – „Videoproduktion im Netz“ – „Online-Dokumente“ • „DFN-Expo – Entwicklung und Demonstration innovativer Online-Präsentationstechnologien“ (1997–1999)
seit 1998	Leitung des Multimedia-Labors am RRZN/RVS

Sonstiges

1995–1999	Leitung des DFN-Arbeitskreises „Informationsdienste“
1999	Multimedia-Innovationspreis der Hannover-Region (14.10.1999): „Leistungsfähige Integration komplexer virtueller 3D-Szenen in Informationssystemen“. Zukunftsfabrik Kommunikation, Multimedia-Initiative der Hannover-Region, Technologie-Centrum Hannover