# High-Performance Online Presentation of Complex 3D Scenes

*S. Olbrich, H. Pralle*
*Lehrgebiet Rechnernetze und Verteilte Systeme (RVS),*
*Universität Hannover*
*Schloßwender Str. 5, D-30159 Hannover, Germany*
*Tel.: +49 511 762 3078, Fax: +49 511 762 3003*
*email: olbrich@rvs.uni-hannover.de*

**Abstract**

Online presentation of virtual 3D objects in the Web, based on Internet standards, is limited due to several performance bottlenecks, quality restrictions and missing functionality. Particularly in the scientific context, where high-performance client, server, and network equipment exists, and requirements for high complexity of represented 3D geometry are given – e. g. in the case of scientific visualization of large datasets – the potential performance of such scenario is not utilized. This paper describes the concept, implementation and evaluation of an optimized viewer, based on a new 3D stream format. The design of this 3D representation and the strategies realized in the viewer were tuned for efficiency, especially to take advantage of high bitrates to obtain short latency. This leaded to the feasibility of streaming sequences of 3D objects, applying the „Real Time Streaming Protocol“ (RTSP) to enable on-the-fly presentation as a 3D movie, freely navigatable at the client side, using virtual reality methods, such as stereoscopic presentation in conjunction with tracking systems.

**Keywords**

Virtual Reality, VRML, Scientific Visualization, Hypermedia, Browser, Plugin.

## 1 INTRODUCTION

Distributed multimedia information services in the actual discussion are based on protocols, addressing schemes, and services that are established in the Internet. These are described in Internet Standards called RFCs (Request for Comments),

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35388-3_42

H. R. van As (ed.), *High Performance Networking*

e. g.: TCP/IP, URL [3], HTTP [5], MIME [6], HTML [2]. The combination of these Internet techniques – well-known as WWW (World Wide Web) – allows the development of interactive, distributed applications that take advantage of the integration of different media types: text, hypertext, image, graphics, video, audio, and 3D objects – or compositions of these, timely or spatially synchronized, as hypermedia documents. On the client side, a generic browser – such as Netscape – serves as the user interface with extensible presentation and interaction capabilities.

3D technology is useful in different application areas of information systems to take advantage of three-dimensional, ergonomic user interfaces, adapted to virtual reality metaphers, for example:

- as a method to navigate in an information space, or
- for online presentation of virtual 3D scenes, e. g.:
  - reproductions of real objects or
  - artificial scenes, such as results from scientific visualization.

In the application of 3D techniques that are established in the WWW, such as VRML (Virtual Reality Modeling Language) – applied in Version 1.0 [1] and 2.0, now called VRML97 [9] – in conjunction with appropriate viewers (see also [19]), several limits have been observed. For certain scenarios, especially in the context of

- high quality application requirements, e. g. handling objects with high complexity,
- network infrastructure offering high bitrates, such as local networks,
- high performance server and client systems,

which exist in scientific and industrial research environments, several constraints regarding performance, quality, and functionality aspects prohibit any useful application, in particular:

- unacceptable delays from request to presentation,
- no progressive presentation capability,
- low frame rates while navigating in the scene,
- low quality of rendering – e. g. no support of antialiasing,
- little support of immersive virtual reality methods, such as stereo presentation and tracking systems.

In the following, the requirements in our focussed application scenario and reasons for the disadvantagous characteristics of current implementations are analysed, resulting in the introduction of an innovative approach and the evaluation of a prototypic realization for the accelerated online-presentation of virtual 3D objects: *DocShow-VR*.

## 2 SCENARIO: „SCIENTIFIC VISUALIZATION“

We focus on applications of 3D information services to support visualization applications, in order to contribute advanced methods for high-quality online presentation of scientific results. The typical considered working environment in science and research consists of

- high-performance graphics workstations – „Clients“, and
- data sources, such as compute or information servers – „Servers“,
- connected to high-speed local and/or wide area network – „Intranet/Internet“.

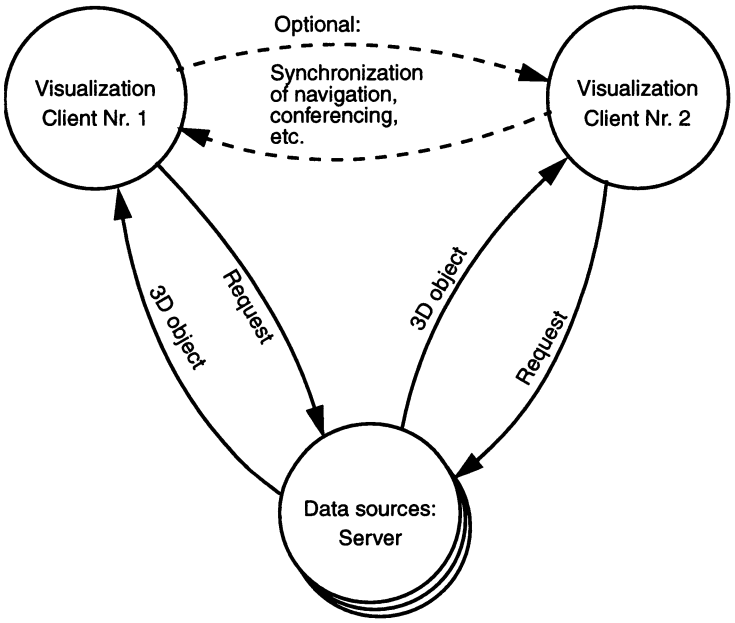


**Figure 1** Visualization clients in a collaborative, distributed system.

In the scientific visualization context, high-quality services and applications are required because of the constraint not to allow lossy processing steps, that could result in misinterpretation of the representation. Involved are several data and media types, in particular vector-oriented 3D objects. Compression algorithms – if considered at all – have to be lossless in order to preserve the accurary of the representation. Besides that, they have to be implemented efficiently in software, since hardware solutions for such special problems are not realistic for the small number of products on the market.

This is in contrast to consumer applications, where lossy compression techniques – such as JPEG and MPEG for 2D raster images and videos, respectively – are widely used.

For data exploration of high-volume, multidimensional simulation or measurement results, interactive, three-dimensional computer graphics systems are appropriate, and immersive, virtual reality systems are increasingly required – often supported by specialized visualization software. To take advantage of the underlying 3D representation for platform-independent, navigatable high-quality online presentation, the requirements are:

- standardized 3D file format – such as VRML (Virtual Reality Modeling Language) [1][9][19],
- open export interfaces for 3D file format in the visualization software – e. g. „write VRML" module for AVS [22],
- generic WWW browser – such as Netscape,
- 3D viewer as external application or as inline-plugin – e. g. Cosmoplayer,
- presentation and interaction devices.

With this prerequisites it is, in prinicipal, possible to construct a system for publishing visualization results by storing VRML files on a WWW server and accessing and presenting them via Internet protocols, as shown in Figure 2.

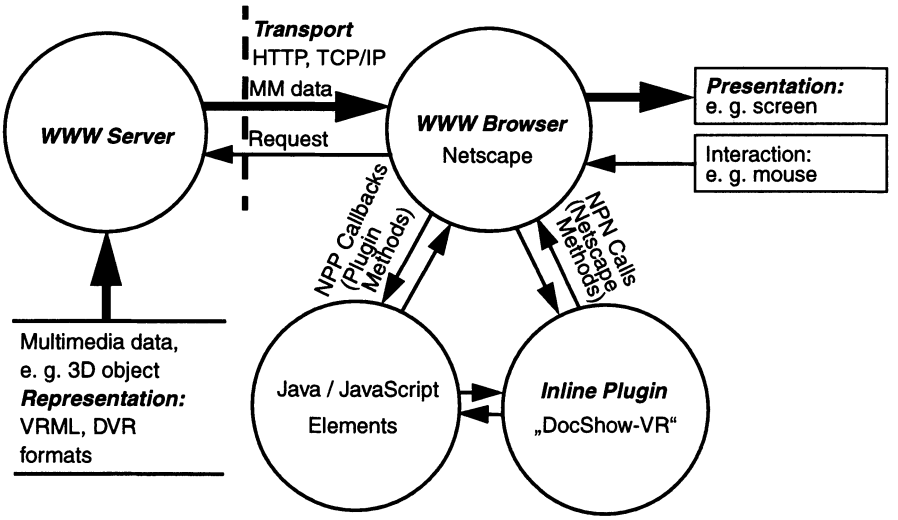


**Figure 2** Multimedia online publishing in the World Wide Web: client/server model.

In Figure 2, the viewer is integrated as an *Inline-Plugin*, which is realized as a shared library, according to the conventions specified by Netscape [15]. If a URL is called whose content matches the MIME type that is supported by this plugin, a presentation window is opened inside the Navigator or Communicator application, and the data is streamed directly into the plugin via callback routines provided by the plugin software. As such, this interface works significantly more efficient than the external viewer application call mechanism using a local file copy, in particular allowing smaller latency and higher throughput.

An Inline-Plugin can also be called by using an `EMBED`-Tag, similar to the `IMG`-Tag for presenting GIF or JPEG images, embedding such a presentation in an HTML page. Besides that layout-supporting capability, Netscape has defined a programming interface that allows to communicate with Java and JavaScript – this is called *LiveConnect*. By applying this mechanism it is, for example, possible to offer plugin functions that can be controlled by user-created buttons on the same HTML page.

## 3 PROBLEMS

It has been shown that this VRML-based configuration, consisting of available internet tools, is insufficient in the considered scenario, regarding several aspects:

1. Performance,
2. Quality of Presentation,
3. Functionality.

### 3.1 Performance

The **load times** for complex objects with polygon count in the order of 100.000 or more – which is typical for scientific visualization results – are in the order of minutes. These startup times are much too long for the intended productive working environment, and the interactivity is very restricted. It is mainly caused by processing expense at the client side – available communication networks, such as local networks, allow much higher bitrates. The preparation of the cleartext-encoded VRML format, mostly gzip-compressed, into the binary representation that is suitable for graphics rendering, is very computational expensive. Involved are decompression, decoding, and parsing at the client side.

The **navigation speed** is very slow, often resulting in several seconds latency – where the rendering performance of the considered graphics workstations would allow interactive frame-rates. Possible reasons are:

- Special rendering primitives – such as „triangle strips“ – that are frequently generated by visualization tools and that are optimized for high graphics throuphput, are not supported in VRML. The universal VRML polygon lists could on principal be optimized at the client side by automatic recognition of

*independent triangles* and *triangle-strips* and appropriate conversion. But this would be computational expensive, and the startup time would increase.

- The traversing and rendering steps of the previously decoded and parsed 3D object structure are implemented inefficiently. It should be taken into account that frequent changes in the graphics state and insufficiently designed loops can significantly contribute to inefficiency.

## 3.2 Quality of Presentation

**Aliasing artifacts** are observed on low-resolution displays as „staircase-stepping". These could be reduced by antialiasing techniques, which are partially available in hardware, without performance degradation, such as multisampling antialiasing on SGI high-end workstations – but this is not supported by currently existing VRML viewers.

A further important issue is to achieve high image quality, that means to **reproduce correct colours**, independently on the presentation device. This could be done by integration of a colour management system in conjunction with object colour specification using device-independent colour space and colour profiles [8], respectively.

Missing capabilities of an immersive virtual reality system:

- **Stereoscopic presentation** is not supported. This is a partially offered mode of 3D graphics adapters, e. g. all SGI workstations are prepared to drive the Crystal Eyes LCD shutter glasses in order to „stereo-view" interlaced presented stereo images.
- **Three-dimensional** input devices, such as 6-degree-of-freedom head-tracking or spaceball devices, are not supported.

## 3.3 Functionality

**On-the-fly presentation** of 3D objects – that means: incrementally loading and progressively rendering – is not supported. This is caused by the structure of the VRML file format, in which object coordinates, attributes (such as normal vectors and colours), and topology data (as indexes, refering to the first data elements) are separated. In principal, the presentation can begin only after all coordinates and attributes are transfered. In practise, the latency is further increased, because the rendering typically starts after completely loading and parsing the file, by traversing the represented object structure.

Streaming of **dynamic scenes** – that means: sequences of completely different geometries, presented in real-time, as a 3D-film – is not supported. This limitation is particularly caused by missing real-time capabilities of the browsers and the absense of a 3D streaming protocol.

## 4 SOLUTION: 3D FILE FORMAT „DVR", INLINE-PLUGIN „DOCSHOW-VR"

To overcome the efficiency, quality, and functionality problems described in the previous section, we have developed a new 3D file format (DVR), an optimized viewer (DocShow-VR), and a VRML-to-DVR converter (wrl1toDVR).

Several innovative approaches are implemented by now:

1. Minimal startup time and „on-the-fly" rendering, progressively while transfer.
   - Decoding and providing graphics data for rendering with minimal computational cost.
     - The DVR format uses the IEEE format for binary representation of float and integer values in network-byte-ordering, similar to the binary encoding of CGM (Computer Graphics Metafile, ISO 8632). This corresponds to the internal representation of most workstations, where these values can be used directly as arguments for the graphics rendering, which is based on OpenGL [13], the high performance, low-level 3D graphics API with the currently widest platform support. Excepted are workstations and PCs based on Intel and DEC CPUs, where one conversion step has to be executed after the transfer. This is done by the macro *ntohl()*, which is applied on 32-bit integer and float values. DVR records consisting of such byte-order-sensitive data are recognized by a flag in the appropriate record header, which is reset after conversion, in order to detect pending conversions, too.
     - For parts of 3D objects where no explicit normal vectors are specified, a conversion process computes the required normals and stores them in the DVR file.
   - Storage and transmission of direct coordinate and attribute values, instead of indexing them in a topology section. This allows starting of rendering as soon as possible, and allowing progressive rendering and efficient pipeling of the transport and rendering processes.
   - To take advantage of this short-latency, incremental, streaming process strategy, the viewer was implemented as a Netscape inline-plugin, using immediate-mode graphics instead of display lists in OpenGL.
   - Usage of network infrastructure providing high bitrates, such as IP over ATM (155 Mbps) in our institute.
   - Compression techniques to reduce volume of transmitted data are not yet involved. This is an area for future development, since geometry-based, lossless compression is available [21], and lossy techniques could be used for integration of different levels of detail in the representation [11]. But in the moment we explicitely avoided any computational overhead at the client side that could introduce additional latency.

2. Efficient OpenGL rendering support: application of *triangle-strip* and *triangle* primitives besides the universal *polygon* primitives, optimization of graphics status changes, loop optimizations.

   - In a preprocessing step, the independent polygons of the VRML format are analysed and consecutive polygons compared, in order to recogize more efficient rendering primitives which are then appropriately written into the DVR file.

   - VRML-based information regarding the graphics state, such as material properties or transformation matrices, are converted to DVR records which are output only when necessary – for example, in case of a status change before a new graphics primitive.

   - To reduce a CPU bottleneck that could be caused by compute-intensive, frequently executed `if`-statements in sensitive loop kernels (while rendering of attributed sequences of graphics primitives), the possible cases were classified, and for each class separate, optimized routines were implemented. The number of the detected class is coded in the respective DVR record header, so the rendering process can directly execute the appropriate routine. These routines are individually optimized, e. g. by loop unrolling and optional usage of optimized rendering calls, such as *glVertexArray()* in OpenGL 1.1.

3. Compatibility to standards and transparent application.

   - A converter has been implemented, providing the preprocessing features as explained above, that takes VRML 1.0 files as input and writes *DVR* files. At the beginning of our work, VRML 2.0 was under development, but we started to support VRML 1.0, based on the publicly available VRML 1.0 parser library from SGI. In this way we were operating on top of an internet standard for the representation of 3D scenes.
     Our acceleration mechanism is on principal similarly applicable on VRML 2.0, which became an ISO standard (VRML97 [9]), and we will eventually upgrade in this way. But question is if our results would influence the specification of an optional binary coding of a future, revised VRML standard [10] or a virtual reality transport protocol [4].

   - Our VRML-to-DVR converter – `wrl1toDVR` – was built into a proxy cache, based on `squid` and a universal `webfilt` tool, which is able to analyse and optionally modify HTTP protocol elements [23]. In a prototypic configuration we provided a Netscape specific PAC (proxy automatic configuration) file in order to access our converting proxy cache for VRML files requested by the user. In case of a cache miss, the proxy automatically converts them using `wrl1toDVR`, called by `webfilt`, and delivers the resulting DVR format to the client, while storing it in the `squid` cache (see Figure 3).

In this way a transparent VRML access and acceleration mechanism was demonstrated. Further developments in this direction could also be suitable for automatic creation of derived media types, such as images or videos or general application for converting services.

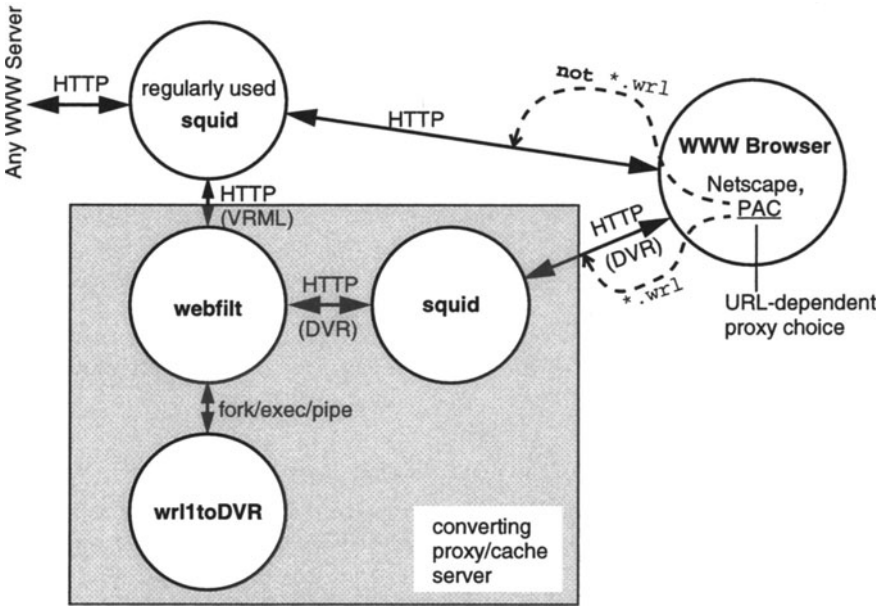


**Figure 3** Proxy/cache configuration consisting of squid, webfilt, and wrl1toDVR.

4. High-quality presentation and interaction, contributing to immersive virtual reality.
   - Aliasing artefacts, such as stair-case stepping on low-resolution displays, can be reduced by wellknown antialiasing techniques. Our viewer supports a hardware-accelerated method called *multisampling antialiasing*, which is offered on SGI high-end graphics workstations as an OpenGL extension, without significant performance degradation.
   - Stereoscopic viewing, available on all SGI workstations, is supported by our plugin. The viewing transformation is controlled by the specifications of the `PerspectiveCamera` node in the original VRML format. The `focalDistance` now gets particular significance for the appropriate observer-to-display distance, but the `heightAngle` must be overridden according to the height of the display and the distance of the observer to the display. The actual constellation has to be calibrated by measuring the width and height of the display, the distance of the observer to it and the eye distance of the

observer and writing these values into a configuration file, which is read by the plugin software at startup. This mechanism has been successfully applied to monitor and large-screen stereo projection devices, using active LCD shutter technique and passive polarizing glasses, respectively.

- Head-tracking systems serve to measure the current position and orientation of the observer. These values are used to control the viewing transformation in order to get a holography-similar presentation. By now, our plugin supports one such device, the ultrasound-based Logitech/Stereographics CE-VR which is integrated in LCD shutter glasses.

5. Streaming sequences of 3D scenes.

- In analogy to techniques for streaming video and audio media, we have thought about such a mechanism to present time-dependent 3D objects, as „stereoscopic movies", viewable and navigatable similar to holography. In order to keep synchronization requirements, we can take advantage of
  - our efficient transmission and presentation capabilities,
  - scaling in quality (fine or coarse representation with appropriate level of complexity) and in time (variable time steps), allowing to vary bitrates or rendering rates,
  - and a control protocol already developed for video and audio.

  We have extended our plugin to support the *Real Time Streaming Protocol* (RTSP [18]), in conjunction with a special streaming server that delivers *3D streams*. These consist of sequences of 3D scene descriptions in DVR format, terminated by appropriately inserted „end-of-scene records". Since in this streaming case we had to apply and implement a protocol that is not supported in the Netscape browser, we are requesting first a description file via HTTP, representing a special MIME type that is detected by our DocShow-VR plugin as a script. This is then interpreted in order to request one – or more, then presented as composition – 3D streams. The protocol handling is realized as a separate thread, so that the – eventually moving – scene can be asynchronously navigated, e. g. using mouse-control or head-tracking devices.

The implementation of the VRML-to-DVR converter and the viewer was based on the freely available C++ library for parsing VRML 1.0 from SGI – QvLib [20], an email message from Jan Hardenberg, consisting of fragments of an implementation of a rudimentary, OpenGL-based VRML-1.0 viewer [7], and the Netscape Navigator Plug-In Software Development Kit [14]. The current release of the converter runs on several UNIX workstations – additional converting support is given by a web page, working with a form-based file upload [12], and the DocShow-VR plugin viewer software actually supports UNIX (HP/UX, SGI Irix, SUN Solaris) and Windows 95/NT. For UNIX platforms not providing an OpenGL runtime environment, plugin versions linked with the OpenGL emulation library Mesa [17] were created.

The binaries, the web-based VRML-to-DVR converter service, and several application examples are publicly available:

http://www.dfn-expo.de/Technologie/DocShow-VR/

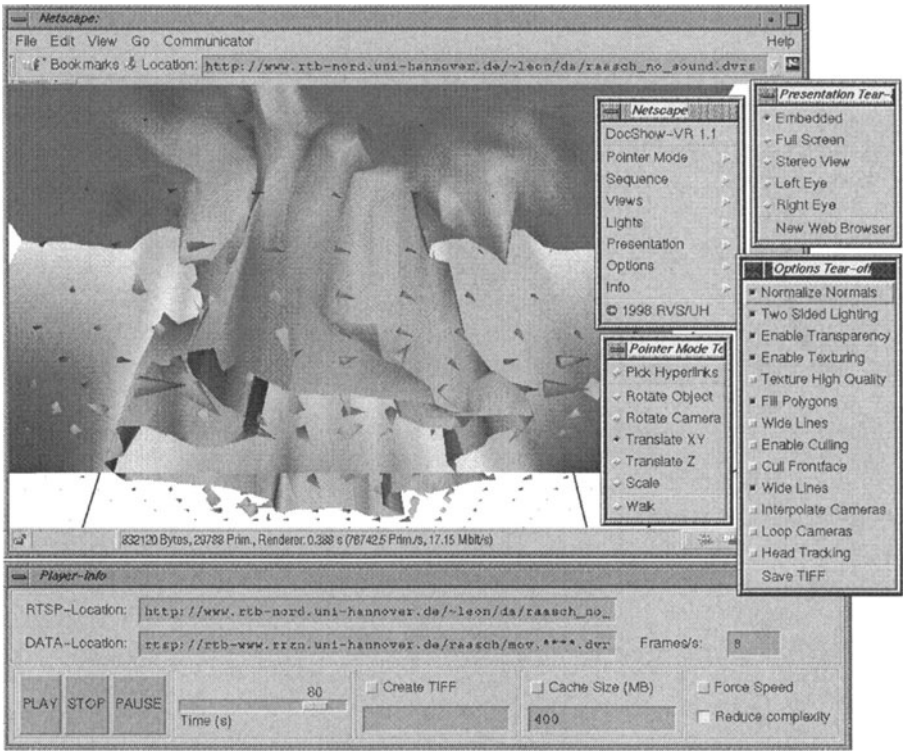


**Figure 4** Application example: „Atmospheric convection“,
also showing some popup menus of „DocShow-VR“.
Data by courtesy of the Institute for Meteorology, University Hannover.

## 5 EVALUATION: „SGI COSMOPLAYER“ VERSUS „DOCSHOW-VR“

In order to evaluate our implementation, we compared the data volumes, startup times, and rendering rates in typical applications, using the Inline-Plugins *CosmoPlayer 1.02* (VRML 1.0/2.0) and *DocShow-VR 0.9*. Here we show results from measurements with the 3D scene „DX-03“ that is distributed with the OpenGL benchmark *viewperf* [16]. The object was first converted from the viewperf-specific MSH (triangle-mesh) format to VRML 1.0/2.0 and DVR formats – DVR optionally with or without triangle-strip optimization. „DX-03“ represents a model that was

generated by IBM Data Explorer, a scientific visualization software, and consists of 91584 triangles as triangle-strips with ever ca. 100 triangles. The tests can be reproduced via this web page:

http://www.dfn-expo.de/Technologie/DocShow-VR/dx.html

It could be shown that *DocShow-VR* produces rendering rates similar to those of *viewperf*. But the triangle-strip optimization of the VRML-to-DVR converter had to be processed, and the plugin options *Normalize Normals*, *Two-sided Lighting*, and *Transparency* had to be disabled to reach this. The plugin options can switched on or off by using popup menues, or by specifying them in the EMBED-tag that produces the 3D picture in the HTML page. Are these optimization facilities not noticed, disadvantages regarding data volumes, startup times and rendering performance are established (see Table 1).

**Table 1** Performance comparison: DX-03 (512 x 512) on an SGI Onyx Reality Engine$^2$ (2 x R4400, 200 MHz, 2 RMs), accessing an Apache WWW server on an SGI Challenge L (2 x R4400, 200 MHz), connected via ATM (155 Mbps)

| | DocShow-VR 0.9 / DVR Format | | CosmoPlayer 1.02 / VRML 2.0 | |
|---|---|---|---|---|
| | Using triangle strips | Without triangle strips | uncompressed | compressed (gzip) |
| Data volume [bytes] | 2.252.744 | 6.601.928 | 9.768.093 | 2.266.695 |
| Startup time (not progressive: without rendering) | 0,795 s, progressive: 0,911 s | 1,709 s, progressive: 1,894 s | ca. 25 s | ca. 25 s |
| Equivalent bitrate | 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 time | 0,155 s, optimized: 0,084 s | 0,151 s, optimized: 0,144 s | ca. 0,36 s | |
| Rendering rate [triangles/s] | 590.864, opt.: 1.090.286 | 606.517, optimized: 636.000 | 254.400 | |

*Results*

The data volume of the optimized DVR file is similar to the gzip-compressed VRML file.

The startup times in this test case were reduced to 1/28.

The rendering rate differs in this test case in DocShow-VR around the factor 2 and is at best 4 times better than the CosmoPlayer. The peak polygon rate of the applied Reality Engine$^2$ graphics subsystem, as specified in the data sheet from SGI – „3D triangle meshes, smooth shaded, z-buffered, phong lighting: 1,07 Mio. triangles/s" – is achieved. Nevertheless, these data and rendering optimizations are not generelly applicable, since other 3D objects are partly not automatically optimizable, or in certain applications inefficient graphics attributes have to be used.

## 6 ACKNOWLEDGEMENTS

This work is partly funded by the DFN-Verein (German Research Network), with means of the BMBF (German Ministry for Education, Science, Research, and Technology) under the project „DFN-Expo". The authors would like to thank C. Grimm and J.-S. Vöckler (RVS) for the discussion about and configuration of a converting proxy/cache-server prototype.

## 7 REFERENCES

1. Bell. G., Parisi, A., Pesce, M.: *The Virtual Reality Modeling Language – Version 1.0 Specification.* 09.11.1995. (http://www.vrml.org/Specifications/VRML1.0/)
2. Berners-Lee, T., Connolly, D.: *Hypertext Markup Language – HTML 2.0.* RFC 1866, 03.11.1995. (ftp://nis.nsf.net/documents/rfc/)
3. Berners-Lee, T., Masinter, L., McCahill, M.: *Uniform Resource Locators (URL).* RFC 1737, 20.12.1994. (ftp://nis.nsf.net/documents/rfc/)
4. Brutzman, D., Zyda, M., Watsen, K., Macedonia, M.: *Virtual Reality Transfer Protocol (vrtp) Design Rationale.* Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises (WET ICE): Sharing a Distributed Virtual Reality, MIT, 18.–20.06.1997. (http://www.stl.nps.navy.mil/~brutzman/vrtp/vrtp_design.ps)
5. Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1.* RFC 2068, 03.01.1997. (ftp://nis.nsf.net/documents/rfc/)
6. Freed, N., Borenstein, N.: *Multipurpose Internet Mail Extensions (MIME).* RFC 2049, 02.12.1996. (ftp://nis.nsf.net/documents/rfc/)
7. Hardenberg, J.: *RE: QvLib questions.* VRML Hypermail Archive, 27.03.1995. (http://vag.vrml.org/www-vrml/arch/1107.html)
8. Has, M., Newman, T.: *Color Management: Current Practice and The Adoption of a New Standard,* 1996. (http://www.color.org/overview.html)
9. ISO/IEC 14772-1: *The Virtual Reality Modeling Language (VRML97) – Part 1: Functional specification and UTF-8 encoding.* International Standard, 1997. (http://www.vrml.org/Specifications/VRML97/)

10. ISO/IEC 14772-3: *The Virtual Reality Modeling Language (VRML97) – Part 3: Compressed Binary Format Specification.* Editor's Draft 5, 1997.
11. Klein, R.: *Multiresolution representations for surface meshes.* In: Proceedings of the SCCG, 1997. (http://www.gris.uni-tuebingen.de/people/staff/reinhard/mai97.ps.gz)
12. Nebel, E., Masinter, L.: *Form-based File Upload in HTML.* RFC 1867, 07.11.1995. (ftp://nis.nsf.net/documents/rfc/)
13. Neider, J., Davis, T., Woo, M.: *OpenGL Programming Guide – The Official Guide to Learning OpenGL, Release 1.* Addison-Wesley, 1993.
14. Netscape: *Netscape Navigator LiveConnect/Plug-In Software Development Kit,* 1998. (http://home.netscape.com/comprod/development_partners/plugin_api/index.html)
15. Netscape: *Plug-In Guide – Communicator 4.0.* January 1998. (http://developer.netscape.com/docs/manuals/communicator/plugin/)
16. OPC – The OpenGL Performance Characterization Projekt: *Viewperf Information and Results.* (http://www.specbench.org/gpc/opc.static/viewin~1.html)
17. Paul, B.: *The Mesa 3-D graphics library.* (http://www.ssec.wisc.edu/~brianp/Mesa.html)
18. Schulzrinne, H., Rao, A., Lanphier, R.: *Real Time Streaming Protocol (RTSP).* RFC 2326, 14.04.1998. (ftp://nis.nsf.net/documents/rfc/)
19. SDSC: *VRML Repository.* (http://www.sdsc.edu/vrml/)
20. 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)
21. Taubin, G., Rossignac, J.: *Geometric Compression Through Topology Surgery.* IBM Research technical report RC-20340, 16.01.1996. (http://www.research.ibm.com/vrml/binary/pdfs/ibm20340.pdf)
22. 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.* In: IEEE Computer Graphics and Applications, July 1989.
23. Vöckler, J.-S.: *A quick glance at webfilt.* RVS, University Hannover, 03.09.1997. (voeckler@rvs.uni-hannover.de)