WP 2.5 Project Deliverable

Review of existing VR systems, Adaptability to VIRTUALFIRES

Project Number	IST-2000-29266
Project Title	Virtual Real Time Fire Emergency Simulator
Deliverable Type	Report
Deliverable Class	Internal

Deliverable Number	D2.5
Title of Deliverable	Review of existing VR systems, Adaptability to VIRTUALFIRES [WP2.5]
Nature of the Deliverable	Report
Contributing WPs	WP2.5
Contractual Date of Delivery	15. January 2002
Actual Date of Delivery	28. January 2002
URL	www.virtualfires.org
Authors	Sascha Schneider (FIGD), Michael Schmidt (FIGD), Volker Luckas (FIGD), Gunther Lenz (SiTu), Thomas Reichl (SiTu), Gert Svenson (KTH), Kai-Mikael Jää-Aro (KTH)
Contact Details	Institute for Structural Analysis / SiTu Research Univ. Prof. Dipl.-Ing. Dr. techn. Gernot Beer Lessingstrasse 25/II 8010 Graz / Austria Tel.: +43 316 8736180 Fax: +43 316 8736185 Email: gernot.beer@ifb.tu-graz.ac.at

Abstract	The summarized result of WP2.5 is presented in this report. VR systems available at the project partners institutes and their features are presented and compared to serve as a basis for WP 2.6 ("Specification of selected VR system & required extensions").
Keywords	Developer Tools, Specification, Software, Simulation, Visualization

Contents

Review of existing VR systems, Adaptability to VIRTUALFIRES

1 FIGD

1.1 Virtual Design 2

Virtual Design 2 [FIGD1] is constructed modularly. The software is arranged in basic modules and application packages, which are combined according to the customer's needs. Each of the basic module supports certain infrastructures. The application packages contain a number of functionality for specific operational areas, to be configured by a graphical user interface. Many applications require the customizing of software and customer-specific functionality. Therefore, vrcom provides a developer license which allows to functionally enhance the software by dynamically loaded modules. With the product rollout of Virtual Design 2 in June 1999, developers not only have the whole range of functions of the offered user-oriented modules at their disposal, but also a application programmers interface (API). The developer license allows the customer to deploy Virtual Reality in completely new areas and, as a result, secures competitive advantages.

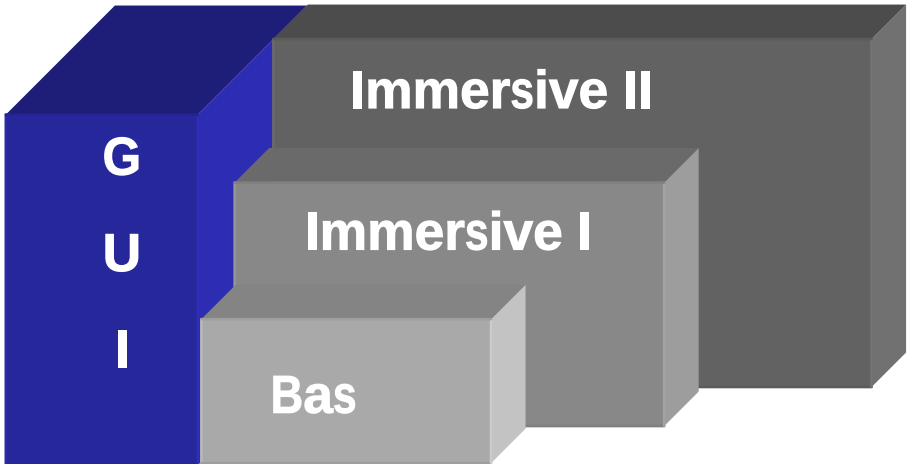

Figure 1: Basic software modules of Virtual Design 2

Virtual Design 2 was developed to enable the user fulfilling the same tasks on digital mockups as on physical ones. The basic module **Basis** provides the core functions of visualizing and interacting with computer-generated data on a workstation. Computer-generated models can easily be loaded, actions can be defined in order to make certain changes in the scenario. Examples for those actions are the recording and interactive playback of camera movements in realtime, the definition of constraints for models or parts of models or the scale and rotation of individual object components. The scale and the rotation of objects can be graphically and interactively stipulated in discrete steps. **Immersive I** extends the basic module **Basis** with a more exact collision detection and further options for interaction. The user can intuitively grab and alter any depicted objects, collisions are highlighted optically. **Immersive II** goes even further, providing **Immersive I** with the possibility of a flexible configuration of the output. This module not only supports the multi-screen projection in the so-called CAVE or on a horizontal output device, but also the control of projections supporting more than four projectors. Furthermore, models can be visualized in dependency on the user's position and orientation.

Standard application packages have been developed through various industrial pilot projects. Each of these are implemented in connection with a basic module. Since each project opens up new areas of operation, further solutions are to follow.

Functionality	Basis	Immersive I	Immersive II
GUI	X	X	x
Tools for data compression and optimization	X	X	x
Reloading, deleting and copying of objects	X	X	x
Transformation, rotation and scaling of objects	X	X	x
Definition of constraints	X	X	x
Real time camera tracking	X	X	x
Configurable navigation	X	X	x
Events by collision, I/O devices, sensors or iteration	X	X	x
Selection of objects	X	X	x
Defining and playing back animations	X	X	x
Point, direction and spot lights	X	X	x
Two sided lighting	X	X	X
Transparent objects	X	X	X
Backface Culling	X	X	X
Time dependent attribute animations	X	X	X
Level of detail	X	X	X
Environment maps	X	X	X
ClearCoat	X	X	X
Sound	X	X	X
Enhanced collision detection		X	X
Grabbing and moving objects		X	X
Multi processor collision detection		X	X

Figure 2: Funcitonalities of Virtual Design 2

Input/Interaction	Basis	Immersive I	Immersive II
Keyboard	x	x	x
Mouse	x	x	x
6D-Spacemouse	x	x	x
Flying Joystick		x	x
Virtex CyberGlove		x	x
Polhemus FASTRAK		x	x
Ascension Flock of Birds		x	x

Figure 3: Input / interaction possibilities for Virtual Design 2

Supported Output Devices	Basis	Immersive I	Immersive II
Monitor	x	x	x
Virtual Research VR4/VR5		x	x
n-Vision HMD		x	x
Projection desk			x
Holobench			x
N sided CAVE			x

Figure 4: Supported output devices of Virtual Design 2

Output Modes	Basis	Immersive I	Immersive II
Mono projection	x	x	x
Active stereo projection	x	x	x
Passive stereo projection		x	x
Single pipe	x	x	x
Multi pipe		x	x

Figure 5: Output modes of Virtual Design 2

Accessories	Basis	Immersive I	Immersive II
3D scene editor	x	x	x
Data import and processing tools	x	x	x

Figure 6: Accessories of Virtual Design 2

1.2 OpenSG (Cave Version)

OpenSG [FIGD2] is a portable scenegraph system to create realtime graphics programs, e.g. for virtual reality applications. It is developed following Open Source principles and can be used freely. It runs on IRIX, Windows and Linux and is based on OpenGL.

The current version is 1.0. Some of the supported features are:

- Multithreaded Asynchronous Scenegraph Manipulation: One of the central parts of the OpenSG design. The OpenSG data structures are set up in a way that allows multiple independent threads to manipulate the scenegraph independently without interfering with each other.

- View Volume Culling: Only render what's in the field of view. This feature is very useful in large scenes.

- Portability: The whole system works on Irix, Linux and Windows, and is compiled daily on each to ensure that changes that break portability are not detected when it's too late to fix them.

- State Sorting: To efficiently use graphics hardware, the OpenGL state has to be changed as rarely as possible. Sorting the visible objects by state is the main way to do that.

- Striper: Turning independent triangles and quads into connected strips significantly increase performance.

- Loader: VRML97, OBJ, OFF, RAW: These are the loaders that are integrated in OpenSG.

- Standard Nodeset: The standard scenegraph nodes like Groups, Transformations (Matrix and Component), Level of Detail and geometry are supported.

- Flexible Geometry: The Geometry node can handle all OpenGL primitives, and can handle all of them mixed in one node. Thus there's no need to create multiple objects just to mix triangle strips and fans.

- Runtime Changeable Type System: All objects are created from prototypes that can be exchanged at runtime. This allows easy adaption of global defaults, as well as exchanging system objects with newer versions better suited to the application at hand, at runtime.

- Reflective Structures: A flexible type system like the one mentioned above demands flexible tools. To allow building those, all structures can give out information about themselves and their data. This allows writing very generic tools.

2 KTH

This section will go through the available resources at the Center for Parallel Computers (PDC) at the Royal Institute of Technology (KTH). We will start by listing the existing hardware that can be used for simulation and visualisation, after this we will list the software packages that we consider suitable for use and the important properties of these.

2.1 Software

We have studied a number of software packages from the standpoint of their suitability for visualisation within VIRTUALFIRES. The software can be grouped into three categories:

1. Packages for scientific visualisation. COVISE, AVS/Express MPE and VTK belong to this group. They contain pre-prepared functions for the common visualisation methods and thus would not require us reimplement streamlines, isosurfaces etc. However, all the packages mentioned are extensible and allow the easy addition of new modules both for computation and visualisation. All three software packages can be used in CAVEs.

2. Applications specifically created for CFD visualisation. EnVis is currently the only representative of this group. The advantages with domain-specific applications are that they are optimised for the subject task, thus resulting in higher performance, in addition one can immediately start visualisation, but extension may be difficult. In the case of EnVis, though, we do have access to the source code.

3. VR platforms. We have here included CAVELib, VR Juggler and GNU/Maverik. These do not in themselves contain neither visualisation functions, nor even graphics functions, instead these have to be supplied in a standard graphics API, such as OpenGL or OpenGL Performer. They do however, enable the interaction and presentation of said graphics in CAVE displays or on head-mounted displays.

Some important points with relevance to all these software packages: as it may turn out to be useful to do distributed visualisations, with participants at different sites, so it has been noted which packages easily allow distribution of the graphics. Furthermore, for the commercial packages, the list prices for the various modules have been noted, as indicators for the future. Still it should be noted that in some cases is the software already available at PDC, and in all cases the prices are negotiable.

We have also mentioned RasDaMan, which while not being a graphics package, may still be useful as a database engine for the foreseen large data sets.

2.1.1 COVISE

COVISE [KTH1] is visualisation software developed by VirCinity IT-Consulting GmbH [KTH2]. Main points:

- Graphical dataflow language to define the visualisation elements.

- Supports concurrent computation and visualisation, allowing interactive control of simulation parameters.

- Post-processing modules for CFD data.
- Reads data from Fluent, FIRE, PATRAN, Nastran, and others. Customised data input modules can be added.
- Visualisation modules include isosurfaces, particle traces, vector fields and others. Modules can be extended. There is a separate module for volume visualisation.
- Runs on SGI, Linux and HP, has computation modules for Cray T3E and NEC SX.
- Has modules allowing remote collaboration on visualisation.
- COVER is a module for using COVISE with CAVE-type displays. It supports a number of trackers and interaction devices. An HMD version does not yet exist but can be developed.
- The list prices for COVISE is 13 kEUR for the base version + 63 kEUR for a three-pipe immersive module + 15 kEUR for the volume rendering module. Educational institutions pay 3250 EUR for the base version, 15750 EUR for the three-pipe immersive module and 3000 EUR for the volume rendering module.
- PDC has good contacts with VirCinity and can (within reason) affect the development of new modules.

COVISE is available at PDC and has been used to some extent.

2.1.2 AVS/Express MPE

AVS/Express [KTH3] is a well-established visualisation system, developed by Advanced Visual Systems, Inc. [KTH4]. Its main points consist of:

- Graphical dataflow language to define the visualisation elements.
- Modules can be written to allow communication with concurrently executing simulation software.
- Reads data from NetCDF, CGNS, FLD, UCD. Data import formats can be interactively described.
- Visualisation modules include stream ribbons, isosurfaces, glyphs and others.
- Runs on Irix, AIX, Linux, Solaris, HP-UX, Windows 98, 2000 and NT and others.
- The Multipipe Edition allows the use of AVS with CAVE-type displays on the SGI platform.
- The list price for a single license for AVS/Express is about 2 kUSD, the MPE is 25 kUSD.

AVS/Express is available at PDC. AVS has been used quite a lot at PDC. MPE is currently not available at PDC.

2.1.3 VTK

The Visualization Toolkit [KTH5] is an open-source visualisation library, developed by Kitware, Inc. [KTH6]. Its main points consist of:

- Object-oriented library, using dataflow model for visualisation.
- Easily extensible.
- API for C++, Tcl, Java and Python.
- Reads data from VTK, PLOT3D and POP. Easily extended for other data formats.
- Visualisation modules include streamlines, isosurfaces, volume rendering and many others.
- Runs on Unix and Windows.
- VtkActorToPF class written by Paul Rajlich allows graphics output to OpenGL Performer and thus use of VTK in e g CAVELib applications.
- Source code is freely available.

VTK and VtkActorToPF are available and have been used extensively at PDC.

2.1.4 EnVis

EnVis [KTH10] is a fluid dynamics visualisation program developed by Jonas Engström, then at PDC, for Volvo Aero Corporation. Main points:

- Reads data from VOLSOL, PLOT3D and EnSight.
- Visualisation methods include particle traces, streamlines, isosurfaces and colour mapping.
- Runs on Irix.
- Supports remote collaboration.
- Requires CAVElib, and thereby should be possible to use on HMDs, even though this has not been tested.
- Source code is freely available.

EnVis is available and has been used at PDC.

2.1.5 CAVElib

CAVELib [KTH7] is a function library to write applications for CAVE displays, originally developed by the Electronic Visualization Laboratory [KTH8] at the University of Illinois, now marketed by VRCO, Inc. [KTH9] Main points:

- Simple programming model, easy to port applications written for other displays.
- API for Open GL, Open GL Performer and Open Inventor.
- Support for multi-user applications.
- Runs on Irix, Solaris, Linux, HP-UX and Windows 2000.
- Supports CAVE-type displays as well as head-mounted displays. Programs can also be run in 'simulator' mode on ordinary workstations.

CAVELib is available and has been used extensively at PDC.

2.1.6 VR Juggler

VR Juggler [KTH11] is a virtual reality application framework, developed at the Virtual Reality Applications Center [KTH12] at Iowa State University. Main points:

- Collection of C++ classes.
- Supports OpenGL and Iris Performer.
- Runs on Unix and Windows.
- Supports CAVE- and HMD-type displays in addition to ordinary workstations.
- Source freely available.

VR Juggler is available at KTH, but has been used very little.

2.1.7 GNU/Maverik

GNU/Maverik [KTH13] is a virtual reality system which has been developed by the Advanced Interfaces Group [KTH14] at the University of Manchester. Main points:

- Micro-kernel and framework for developing applications, specifically intended for large and complex environments.
- Support for multi-user applications by the Deva software.
- Runs on Unix, MacOS and Windows.
- Requires some amount of configuration to work on CAVE- and HMD-type displays.
- Source freely available.

Maverik has not been used at PDC up to now.

2.1.8 RasDaMan

The Raster Data Management System [KTH15] is a multidimensional database system for raster data developed by Active Knowledge GmbH. [KTH16] Main points:

- Handles matrix data of (almost) arbitrary dimensions and type.
- Efficient handling of subsets of large matrices.
- Compression of data.
- API for C++ and Java.
- Available for Solaris, HP-UX, Linux and Windows NT.

RasDaMan is available and has been used at PDC.

2.2 Recommendations

Of the listed systems, we would recommend Strindberg as a computation engine, as it is the most powerful machine we have available and the necessary software is either already available or easily portable to that machine. For visualisation we recommend Boye, as allowing the use of the Cube, which in our experience is a very suitable visualisation environment.

With regards to software, the matter can be argued further, but our recommendation would be to use COVISE, since it is already available, contains the necessary functionality, is extensible, we have close contact with the developers and it can be used in a distributed setting. Admittedly it does not currently contain support for head-mounted displays, but that is easily added, according to the developers.

3 SiTu

At the moment there is only one VR-System available, the TVS (Tunnelling Visualisation System). It was developed by Dr. Gernot Oppriessnig on the institute of structural analysis.

3.1 Description of TVS – Tunnelling Visualisation System

3.1.1 Requirements

3.1.1.1 OS

The currently available version runs on IRIX 6.4 and above. This version will not be further developed.

A version running on Windows 2000 is under development and it is scheduled for release by the end of march. It will be entirely based on OpenGL 1.1 and Microsofts MFC framework.

3.1.1.2 Hardware

A sgi-Workstation with Hardware-Texturing, like the Octane-MXI, is the preferred environment. While execution on an O2 is possible, the framerate of the visualization drops to an unacceptable low level of a few frames per second.

Support of a spaceball from 3Dconnexions for navigation is built in. This spaceball is connected via a RS-232 interface to the computer.

3.1.1.3 Data Input

Currently there is only support for reading data from BEFE.

The import is performed in two steps:

First the BEFE-data is converted to ASCII-data by a filter-program. This filter also removes all geometry data of internal Finite-Elements, which are not necessary for visualization, i.e. all Elements, that do not belong to a boundary of a geometry or to geological structures.

Also all FE data values are recomputed to according values of points on a definable regular grid. These gridvalues are the basis for the visualization of the scalar FE-values by means of fog or vectors.

In the second step this ASCII-file is read into the TVS-System. All Data must be read by TVS before it can be visualised. The currently developed Version for Windows should be able to load data separately for each timestep.

3.1.2 Display-options

3.1.2.1 Geometry

The BE-surface and the FE-bounding box can be displayed in the following ways:

- Wireframe

- Shaded surfaces with/without texture

This does also apply for the deformed geometry, which can be animated via timesteps additionally.

The mapping of textures onto surface-patches is supported in two ways:

The texture image is put onto a group of patches, where the image is stretched over the whole group of patches as shown in figure 1.

The texture image is repeated on every single patch, where the image is tiled.

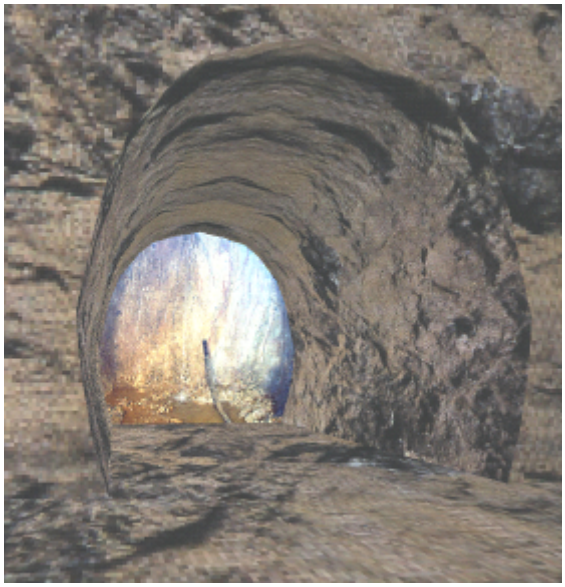

Figure 1: Virtual Tunnel with photorealistic Textures

3.1.2.2 Data

Surface data can be displayed in colour plots for scalar values and in arrows for vectorial data

Grid data can be displayed in Fog for scalar values and in arrows for vectorial data

It is possible to display data on a cutting-plane, which can be positioned freely in space. The display of scalar data on this plane is done by colour plots and for vectorial data the projection of the vectors onto the plane is shown by means of arrows.

Contour plots:

For the visualization of scalar values of BE-data there are 2 options for the interpolation of the gaussian values to the values at the corners of the elements:

- Element wise extrapolation: This yields to discontinuities at the borders
- Smoothing: Local weighted averaging on neighbouring cell-values is performed to get a smooth and continuous image of the values

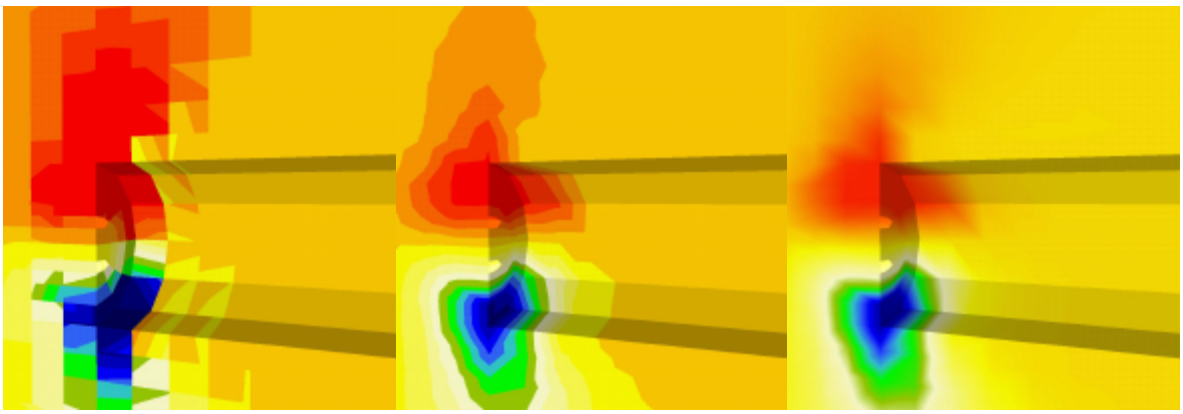

Figure 2: Two steps of global smoothing
(left: not smoothed, middle: smoothed with sharp contours, left: smoothed with blured contours)

Vector plots:

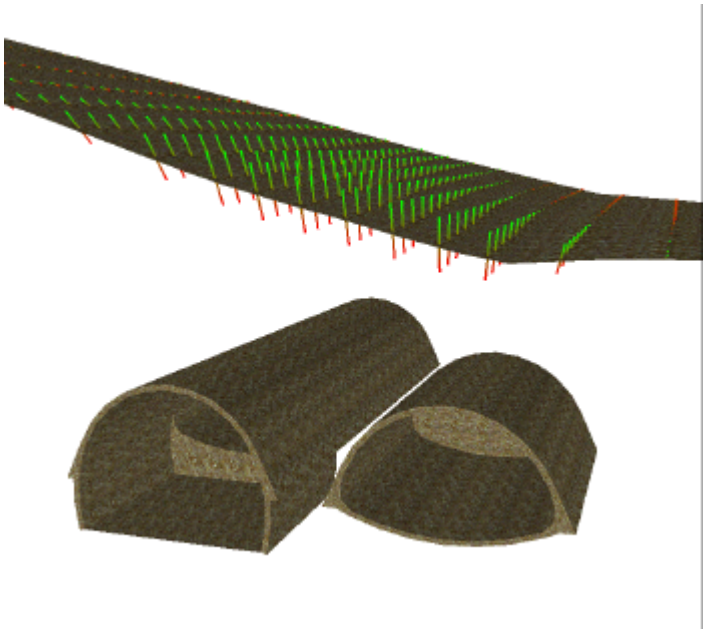

Figure 3: Display of surface-displacements as arrows

Figure 3 shows an example of displaying simulation results as vector plots. Displayed are the settlements on the surface above two tunnels.

3.1.2.3 Realisation of the fogging system

In a 3D model of the tunnel many properties are a function of three dimensions and should be pictured in a way, that takes account of these special three-dimensional properties. The method implemented in TVS is to display the data as fog [SiTu1] with variable density depending on the value of the scalar being displayed. This method yields to good visual impression. During the virtual walk through the model a high stress-value becomes visible as a dense cloud. The structures behind the fogged zone are still visible. Figure 4 shows the amount of stress in the rock pillar between two tunnels, displayed as fog.

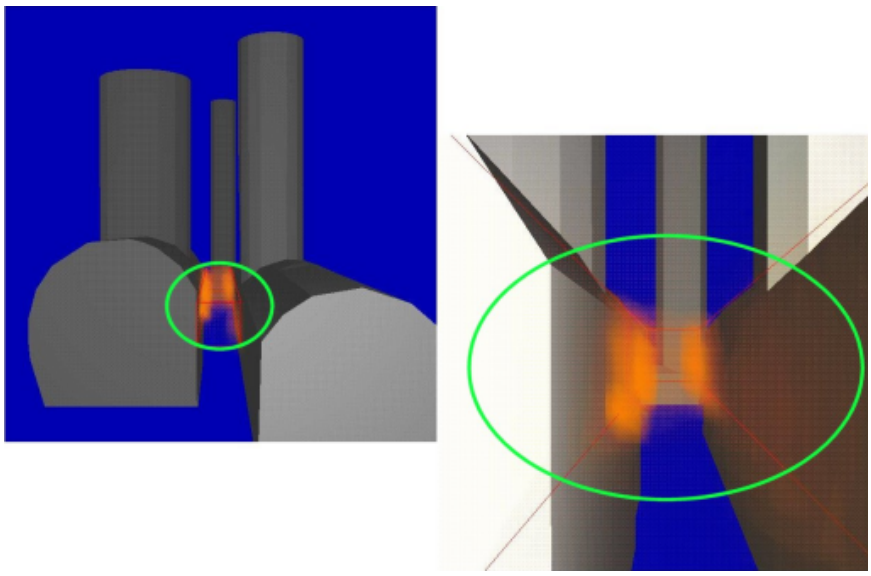

Figure 4: Display of 3D-scalar field as fog

The user can define different scaling-curves [SiTu1, SiTu2], e.g. to highlight maximum / minimum values. With this method it is also possible to draw "fuzzy" geological features also known as "shear bands". These are described as points clustered around a surface.

The three-dimensional visualisation of spatially distributed values as fog makes great demands on the hardware used. If all pixels of the cloud have to be projected into the view level, a lot of calculation effort is necessary. The density values have to be integrated along all lines to the viewing point. In most cases, other objects will be visible and the calculation must take this into account. The projection has to be redone after each change of position or viewing direction. This would take too long for visualisation in real-time. If a fog scene is created as a cluster of points, the graphics system will be stretched to the limit. With a special procedure it is possible to create the fog in real-time.

The method is based on the following idea: Three-dimensional space is divided into planes, where the values are known. In these planes, the scalar values are represented by the density. If the planes are drawn at right angles to the viewing-direction and the borders are extrapolated to zero, the cloud appears to be three-dimensional.

It is shown experimentally, that instead of at 90°, planes may be viewed at an angle between 45° to 135°. This means that it is enough to place the planes perpendicular to one of the three coordinate-directions.

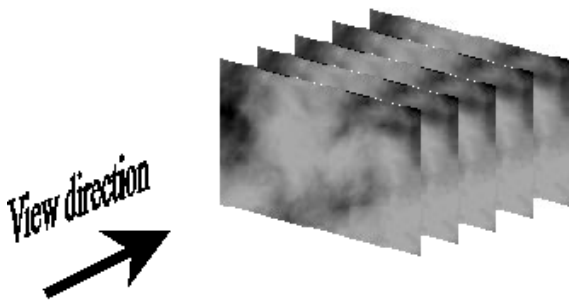

Figure 5: Drawing of fog as parallel planes

The values are known on fixed raster-points, so no calculation is necessary to define the planes. The addition of the density values is done by the graphics hardware. The depth-buffer ensures that only visible parts of the cloud are drawn. If the measured or calculated values are not regular it is necessary to interpolate. A good method to do this is to find a weighted average-value in each grid-point.

3.1.3 User Interface

TVS allows the user to perform a virtual walk through a tunnel which exists in computer memory only. During the virtual walk through the model the user may observe different results of numerical simulations and geological features. With the use of electronic shutter-glasses or a headmounted display (HMD) the model becomes 3D. Tunnelling engineers usually are not very familiar with the use of complicated computer-software, and they should not be required to study huge user-manuals, to be able to walk through the virtual tunnel. The tunnel can be displayed with photo-realistic textures. This textures are derived from pictures taken by a stereoscopic camera .Figure 1 shows an example for the usage of such textures.

Another way to display geological structures and groundwater-level, is, to paint surfaces in colours. The user is able to select, which part of the model should be visible, an in which way it should be displayed. Making the model translucent gives good results, if it is necessary, to see more layers of the model. During the virtual walk-through the position and the viewing-direction are displayed in the horizontal and vertical section of the model. Figure 6 shows the graphical user-interface of TVS.

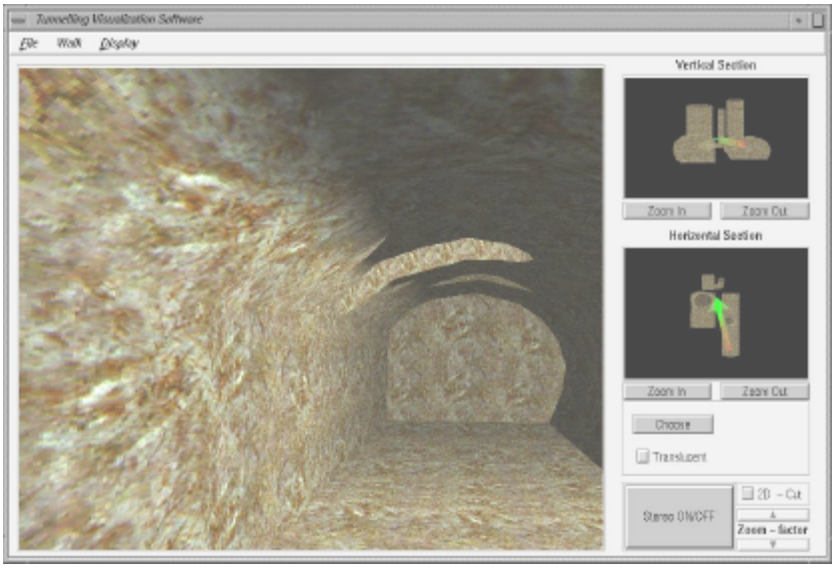

Figure 6: Graphical User-Interface of the Tunnelling Visualisation Software

To make the navigation easy a SpaceBall is used. Moving the ball of this three-dimensional input device will change the position, turning it changes the viewing-direction. A simple, linear transformation SpaceBall–movement to viewer – movement is not very efficient. The user should be able to determine his position very precisely, and, without moving his hand from the SpaceBall, move very fast. For this purpose a progressive scaling is used. To improve the user-friendliness of the navigation, the user can decide, that only the dominant two directions of SpaceBall movement will be processed. It is also possible, to choose these directions manually [SiTu3].

3.1.4 Implementation Details

The current design of TVS does not support any kind of parallel data processing. The whole dataset, except for the textureimages, is contained in one single file. This file is read at once and the full dataset is kept in memory.

3.1.5 Availability to Virtual Fires

In case of selecting TVS Dr. Gernot Opriessnig will provide the sourcecode and support. There is no additional documentation available.

Literature/Links

[FIGD1] Virtual Design 2 - http://www.vrcom.de/

[FIGD2] OpenSG - http://www.opensg.org

[KTH1] http://www.vircinity.de/d+e/data0801/englisch/e_produkte/xe_fs_prod_cov.html

[KTH2] http://www.vircinity.de/

[KTH3] http://www.avs.com/software/soft_t/avsxps.html

[KTH4] http://www.avs.com/

[KTH5] http://public.kitware.com/VTK/

[KTH6] http://www.kitware.com/

[KTH7] http://www.vrco.com/products/cavelib/cavelib.html

[KTH8] http://www.evl.uiuc.edu/home.html

[KTH9] http://www.vrco.com/

[KTH10] http://www.pdc.kth.se/projects/envis/

[KTH11] http://www.vrjuggler.org/

[KTH12] http://www.vrac.iastate.edu/

[KTH13] http://aig.cs.man.ac.uk/maverik/

[KTH14] http://aig.cs.man.ac.uk/

[KTH15] http://www.active-knowledge.de/Products/index.en.html

[KTH16] http://www.active-knowledge.de/

[SiTu1] Beer, Watson „Introduction to Finite and Boundary Element Methods for Engineers“
John Wiley & Sons, 1992.

[SiTu2] Gernot Opriessnig, Gernot Beer “Visualization in Tunnelling” Proceedings 1998 IEEE Conference on Information Visualization 33-38

[SiTu3] Robert A. Noble, Gordon J. Clapworthy “Improving Interactivity within a Virtual Sculpting Enviroment” Proceedings IEEE IV´98