Volume Rendering in a Virtual Environment

Jürgen Schulze-Döbold, Uwe Wössner, Steffen P. Walz, Ulrich Lang

High Performance Computing Center (RUS/HLRS), University of Stuttgart, Germany
{schulze,woessner,walz,lang}@hlrs.de

Abstract. This paper describes a flexible rendering system for scalar volume data which has been integrated into the visualization system COVISE. It allows direct volume rendering based on texture mapping hardware on both the desktop and in projection based virtual environments. Special care has been taken for the design and usability of the virtual reality user interface and for interactive frame rates. It features a new slider-less interaction window for the manipulation of the transfer function and methods for direct interaction with the volume data, such as a sub-volume probing mode. The interface has been evaluated in collaboration with a group of potential end users.

1 Introduction

Volume rendering has become an important visualization method, predominantly in the fields of scientific and medical visualization. Due to the usually large amount of data it is difficult to analyze it in a suitable way. Our approach is to display the volume graphics in a virtual environment, and to provide the user with suitable interaction techniques. The color and opacity transfer functions can be edited interactively from within the virtual world by a 3D user interface. It is designed for CAVE-like environments such as the CUBE at HLRS, but it could also be used with other VR systems after slight modifications.

The volume data is displayed using graphics hardware accelerated texture mapping, which allows the concurrent display of volume and surface graphics in a straightforward way. The most challenging issue was to provide the user with both an intuitive and easy to use interface for the manipulation of the transfer functions, and additional methods to display only those sections of the data which are of interest to the user.

2 Previous Work

Several developments were required to create our volume rendering system in a virtual environment. The underlying visualization framework COVISE is described in [Ran95], details about the integrated VR renderer COVER can be found in [Ran98].

The volume rendering capabilities were added by developing a volume rendering library, which uses 3D texturing hardware for direct volume rendering, as described in [Cul93]. This development was necessary because none of the publicly available volume rendering packages met our requirements, which are multipipe support and availability of an API. For VolPack, 3DDataMaster, Amira, TeleInVivo, VoxelView and VTK this was investigated in [Woh00], SGI's OpenGL Volumizer could not be used because it lacks multipipe support when used with Performer.

The most interesting developments of user interfaces for volume rendering include the Crumbs tool [Bra95] and the StudyDesk interface [Woh00]. None of them allow modification and display of both the color and the opacity transfer function in one window, and they are not designed to be used in conjunction with a traditional visualization system.

3 COVISE

COVISE is a modular visualization system, designed to support collaborative visualization of data in virtual environments as well as on the desktop. The architecture of COVISE allows developers to extend the existing functionality by integrating new code as modules. In a visual application builder, these modules are connected to form a dataflow network (see Fig. 1).

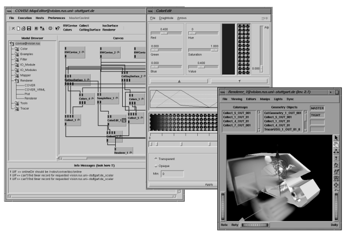


Fig. 1. COVISE desktop user interface and OpenInventor renderer

At the end of the pipeline there is a render module, which can either be a desktop renderer, based on Open Inventor, or the VR oriented render module COVER, which is Performer based, see section 5. Existing modules can be used to visualize geometry, cutting surfaces, iso surfaces, streamlines, etc. New modules have been introduced to sample data from unstructured or non-uniform structured grids to uniform grids. Unlike in the medical field, in technical environments simulation domains are usually modeled using unstructured grids. Existing modules have been extended to support value blanking, which is useful for representing regions containing no data in uniform grids. Finally, both render modules have been extended to support direct volume rendering of uniform scalar fields by integrating VIRVO.

4 Volume Renderer

The volume rendering library VIRVO (Virtual Reality Volume Rendering) provides an API for texture-based volume rendering. The API manages the volume data, provides file I/O and conversion routines, and renders the volumes using OpenGL routines. The renderer supports texturing using 2D or 3D textures. For interactive frame rates the volume data sets have to fit into texture memory entirely, bricking is not supported. The volume data is displayed by drawing equidistant planar texture slices.

4.1 Slice Orientation

The VIRVO system only allows for arbitrary texture slice orientation when the hardware supports 3D texturing. Then the following strategy for the slice orientation is used: Without head tracking, the texture slices are drawn parallel to the screen. For volume rendering in virtual environments, VIRVO uses two different methods to find the texture slice normals. Whenever the center of the user's eyes is outside the volume boundaries, the normals are set parallel to the line from the volume center to the center of the eyes. If the user's eyes are inside the volume, the normals are set parallel to the current viewing direction, which yields the least amount of artifacts.

The reason for using two different methods is that although the least amount of artifacts occurs in the latter case, the constantly changing slice orientation while looking around the scene turned out to be irritating.

4.2 Interactive Classification

For an interactive classification, the mapping from scalar values to RGBA (red, green, blue, alpha) values has to be done using look-up tables, because otherwise the volume data would have to be reloaded into texture memory for every change. Depending on the volume size, this would take up to a few seconds, due to the limited throughput rate from main memory to texture memory (see measurements in [Vol00]).

Our SGI Onyx2 system provides the texture color table extension, which allows interactive changes of the look-up table. This gives instant feedback to the user on transfer function modifications.

4.3 Slice Distance and Opacity Correction

Current graphics hardware is limited in the pixel fill rate. Thus the number of texture slices used to draw the volume is the critical factor for maintaining a certain frame rate, which in turn defines the slice distance. Since we need to achieve interactive frame rates to sustain the effect of immersion, linking the slice distance to the frame rate is a fast method to keep the rendering time low.

In order to retain the correct overall volume opacity when changing the number of slices, VIRVO adapts the opacity values in all slices accordingly by using look-up tables for the opacity values (see section 4.2) and by re-computing the tables whenever the number of slices changes. The required opacity correction formula is derived in [Wei00].

5 COVER

The virtual environment renderer COVER was originally developed as a VR renderer for COVISE, but it can also be used as a standalone VR renderer with full VRML97 capabilities. It supports all major tracking devices and arbitrary projection-screen configurations. Additionally, it is extensible through a flexible plugin system. It is based on the multiprocessing graphics API OpenGL Performer, which introduces several difficulties when trying to include custom rendering code. The plugin interface provides access to all three Performer processes (Application, Cull and Draw) as well as to the COVISE module interface. A plugin was developed which uses VIRVO to render volume data and which provides the 3D user interface described in section 6.

The actual rendering of the volume has to be done in the Draw process, but the user interaction and the interface to the COVISE pipeline is done in the Application process, thus various information has to be passed between these processes. This is done through a shared data structure which includes queues and shared states for the data objects, the user's position and orientation, the probe position, etc.

The plugin uses a custom Performer node to represent the volume objects in the Performer scenegraph. Thus the view frustum culling features of Performer can be exploited, and draw buckets can be used to ensure the correct order of rendering as for a combination of volume and polygonal objects the correct drawing order has to be maintained.

The volume plugin is based on OpenGL routines instead of Performer commands in order to be system independent and to take advantage of the latest OpenGL extensions. Thus, special care had to be taken to ensure that the Performer state is not changed after the execution of OpenGL commands.

6 Transfer Function Editor

The transfer function editor was designed specifically for use in immersive virtual environments such as the CAVE, see [Cru93]. Currently a three button input device is required, we are using a three button 3D mouse. Starting at the mouse position, a “laser” beam of infinite length is cast into the pointed direction. The current interaction element is determined by its intersection with this beam.

The editor and all input elements are placed on a floating menu, see Fig. 2. In the entire editor, we explicitly do not use any sliders to modify scalar values. Instead, we created rotary knobs, which are manipulated by clicking on them and then twisting the hand, similar to real knobs. There are three advantages to this technique. First, the human hand can accommodate twists much more precisely than translations. Secondly, rotary knobs do not have a limited value range, they can be rotated multiple times for manipulation of unconstrained values. Thirdly, the accuracy of the adjustment is independent of the user's distance from the knob.

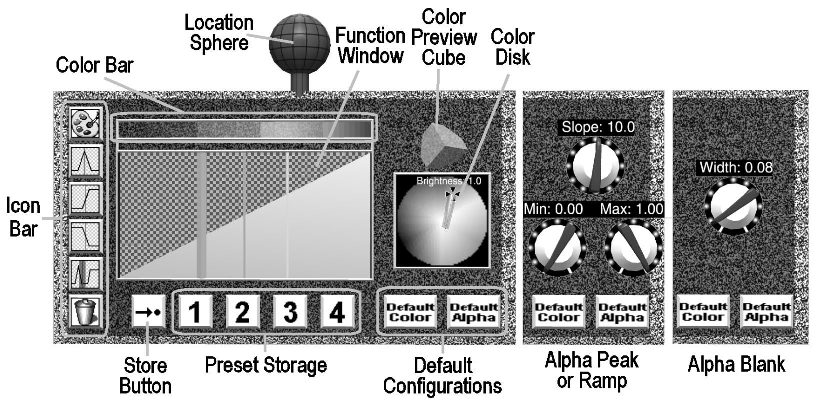


Fig. 2. Transfer function editor

6.1 Editor Position

A sphere on top of the floating menu can be used to set the location of the editor within the virtual world. Dragging it with the left mouse button down, the menu always points to the user; using the right button additionally modifies the orientation. The middle button is used to resize the menu.

6.2 Transfer Functions

The color and opacity transfer functions are modified in a two-dimensional window. The scalar values are assumed to be located horizontally with the smallest value on the left. The transfer functions assign both a color and an opacity to each scalar value. The resulting RGBA values, as applied to the volume data, are displayed in the color bar above the function window. Opacity, which equals the alpha value, is perceived by a varying visibility of the window background texture.

Color Selection. Changing the color value (R, G, and B components) requires setting a control point (pin) in the function window, which is done by clicking on the color palette icon and then clicking on the desired scalar value position in the function graph. The colored disk, which appears to the right of the function window, is designed according to the HSV color model. The hue and saturation components are selected by clicking the left button on the desired location in the colored disk, the brightness is modified by pressing the middle button in the colored disk and twisting the input device.

Both color and alpha pins can be selected by clicking next to them with the left mouse button, they are shifted by dragging with the right mouse button. If more than one pin is located closely together or on top of each other, repeated left mouse button clicks cycle through them. Removing a pin is done by first selecting it and then clicking the trash can icon.

Alpha Selection. Alpha pins, which describe the opacity function, can be set independently of color pins. Setting an alpha pin requires selecting a pin of the desired type in the icon bar and dragging it to a scalar position in the function window. There are three basic types of alpha pins: peaks, ramps, and blanks. Peaks and ramps have a slope, a minimum, and a maximum value. They are used to gradually fade in or out scalar ranges. If peaks or ramps overlap, the maximum value determines the resulting alpha value. Alpha blanks only have a width, they force a scalar range to be transparent. Blanks dominate peaks and ramps.

The properties of alpha pins can either be adjusted by the rotary knobs to the right of the function window (see Fig. 2, middle panel for peaks or ramps, right panel for blanks), or they can be adjusted by twisting the mouse while pressing the middle mouse button with the pointer in one of the three sensitive areas around the pin position. The extent of these areas is displayed in a horizontal bar, located between the function window and the color bar.

Preset Functions. Up to four pairs of transfer functions can be stored for later use. This can be done by first clicking the store icon and then clicking the desired storage number icon. Stored functions can be retrieved by clicking the respective storage number icon directly.

Some commonly used default color and default alpha pin configurations can be applied by clicking the Default Color and Default Alpha buttons. Multiple clicks on these buttons cycle through a number of default configurations.

7 Other Features

The transfer function editor only allows editing the transfer function related parameters. More features can be addressed by the Volume Menu, which is a conventional COVER style menu, and which does not provide rotary knobs instead of sliders. This menu is needed to select the probe mode, the clipping plane, and the frame rate, and it provides an interface to set the animation parameters for time-dependent data sets. The following sub-sections describe these features more detailed.

7.1 Animation

Time dependent data sets can either be displayed in single frame mode or as an animation with adjustable speed. The volume data for the time steps are entirely loaded into texture memory whenever they are to be displayed. If the texture memory is insufficient to accommodate all time steps, OpenGL automatically caches the data in main memory, thus slowing down animation speed considerably. Fig. 3 shows seven time steps of a statistical FE simulation.

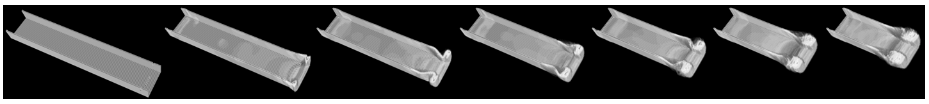


Fig. 3. Time steps of FE statistics simulation

7.2 Image Quality

The default image quality mode is the adaptive mode (see section 4.3) in which the number of volume slices is automatically adjusted, so that the user selected frame rate is maintained. Whenever the user wants to see the image in a better quality, he can click a mouse button while holding the mouse above his eye level to trigger the High Quality Mode (see Fig. 4). In this mode no downsampling is performed on the volume data. Since in High Quality Mode usually no interactive frame rate can be achieved, the user must press a mouse button again to return to the adaptive mode.

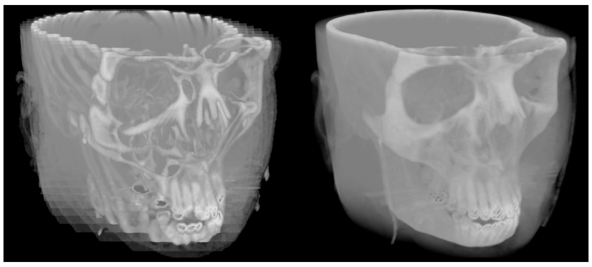


Fig. 4. Adaptive Mode (left) compared to High Quality Mode (right)

7.3 Clipping Planes

OpenGL clipping planes cut off all geometry on one side of the plane. Thus when applied on volume data displayed with the slicing technique, artifacts occur at the clipping plane, whenever the plane is not parallel to the slices. Therefore our clipping approach does not use an OpenGL clipping plane. Instead, in clipping plane mode, we match the volume slice orientation with the clipping plane orientation, and we clip the volume by not drawing the slices on one side of the plane. Furthermore, the volume slices are shifted so that there is always a slice drawn at the clipping plane location. Multiple OpenGL clipping planes can be used additionally.

7.4 Probe Mode

Motivated by the clear boxes in [Bra95], we developed the probe mode (see Fig. 5). When it is selected, the volume data boundaries are displayed and only a cubic subset of the volume is rendered. The user can drag-and-drop the cube with the left mouse button, its size is changed by twisting the mouse while pressing the middle button. Interactive speed is obtained because the volume data in texture memory is not changed but only the texture coordinates are set according to the current position.

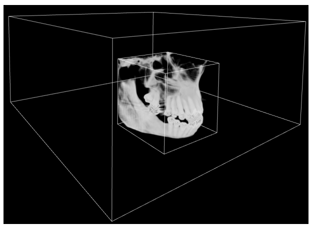


Fig. 5. Probe Mode

8 Evaluation

Derived from [Swa99], our evaluation process consisted of two main phases: a usability inspection and a scenario-based evaluation. Both phases were slightly modified in order to better suit our needs.

8.1 Phase 1: Usability Inspection

In the usability inspection, a specialist from HLRS, who was not involved in the system's development process, reported obvious flaws in the design of the user interface and made redesign suggestions to the developers, who decided on their implementation. The above described volume rendering system already includes these changes.

8.2 Phase 2: Scenario-Based Evaluation

Before the external users were invited, a pilot study of the system was performed by an in-house expert. We detected several problems with e.g. the chronology of the evaluation procedure, the number of tasks, and the understanding of the questionnaire. Then we optimized these factors.

Chonology. The evaluation was put into the following order:

1. *Self-Rating Questionnaire.* In a conference room, the users filled out a questionnaire asking them to self-rate personal skills and knowledge, based on a Likert scale with a value range from 1 to 5, see [Lin99]. We also asked users about their gender, age, and physical constraints such as glasses.
2. *Introductory Presentation (Briefing).* On several slides we gave a brief introduction to volume rendering and the CUBE. We explained the functionality of the transfer function editor in detail and finally presented two application scenarios.
3. *Passive Introduction.* The users were brought to the CUBE, where three evaluators were present: M, V, and P. V and P sat outside the VE. Evaluator M explained the functionalities of both the CUBE and the volume rendering system. The passive introduction averaged around 5 minutes. Evaluator V videotaped Steps 3 to 5, P conducted Participant Observation during the same time.
4. *Active Introduction.* The users were handed the 3D mouse and started exploring scenario #1 with M still explaining. This step seamlessly faded into Step 5, it averaged to 5 minutes.
5. *Exploring the Scenarios.* M left the CUBE and served as an expert mediator, but only helped users when asked to. About 10 minutes before the end, it was switched to scenario #2. This step was aborted when the total elapsed time of steps 3 to 5 reached 30 minutes.
6. *Follow-up Questionnaire.* Back in the conference room, users filled out a questionnaire. Ten questions asked for the overall usability of the user interface, 24 dealt with single components of the volume rendering system. These questions were based on Likert scales. Five additional questions were posed as multiple-choice questions.
7. *Focused Expert Interview (Debriefing).* Finally, a focused expert interview was conducted by P. It mainly addressed the system's functionalities and direct interaction. This interview took between 20 minutes and 1.5 hours. The users' comments were hand-noted.

Scenarios. We chose the following two scenarios in which the users could explore the volume rendering system:

- Scenario #1 was the skull of the male Visible Human CT Data Set [Spi96] (see Appendix, Plate A). It was proposed to emboss the ivories, the auditory canal, or the spine.
- Scenario #2 was a volume data set representing the temperature distribution in the interior of a passenger car (see Appendix, Plate B). The car body, the seats, and the driver were rendered with polygons. It was proposed to visualize the air conditioning output or particularly hot areas.

8.3 Evaluation Results

In this section we present the outcome of the evaluation analysis.

Self-Rating. The evaluation proved that there was the expected range of users' skills (see Table 1). Twelve volunteer users participated in the evaluation. They stemmed from either related departments at the University of Stuttgart, or they worked with visualization software professionally.

Table 1. Self-Rated experience on a Likert-Scale from 1 (low) to 5 (high)

Type of Experience	Average Score
CAVE or CUBE	2.83
Volume Rendering	1.90
Scenarios	1.13
Scientific Visualization	2.33

Follow-Up Questionnaire. We found the following significant answers from the questionnaire:

- For 2 users the image quality was not detailed enough, 7 users did not mind the poor image quality with large volumes.
- 11 users found the High Quality mode useful, 1 found it awkward to use.
- 11 users thought the automatic alignment of the editor menu was useful.
- 10 users liked the layout of the user interface components.
- 1 user had trouble interacting with the rotary knobs.
- For 9 users, defining the alpha function with pins made sense.
- All users found the probe mode to be helpful, 10 users thought it was easy to use.
- 9 users found the clipping plane beneficial, 8 users thought it was easy to use.

Overall usability satisfaction with the system scored from 55 % to 90 %. The numbers are based on a composite measure derived from a System Usability Scale [Bev91].

Table 2. Overall Usability Issues

Issue: RS:	Alpha and color pins are hard to differentiate when both are gray. Use striped alpha pins.
Issue: RS:	Mouse button assignment is confusing when working with pins. Assign selection and dragging to the same button, with an additional undo-function.
Issue: RS:	Imprecise pin positioning in the transfer function field. Introduce a magnification mode and a numeric display of scalar values.
Issue: RS:	When entering the probe mode, users were unaware of the active clipping plane mode. Give feedback on clipping plane position.
Issue: RS:	Users wished for a rectangular or trapezoidal alpha pin. Introduce an additional alpha pin type.
Issue: RS:	Invisible colors in color bar resulting from transparent regions in the alpha function made it difficult to set the color of color pins. Separate the color bar into two regions with and without alpha representation.
Issue: RS:	After deletion of all color pins the volume turns black and thus becomes invisible on a black background, even with opaque regions in the alpha function. Use white as the default color instead of black.

Overall Usability Issues. After having consulted both quantitative and qualitative data, we filtered the most important key findings occurring across all investigative methods. Issues are separated into a short description and a redesign strategy (RS), see Table 2.

9 Performance Numbers

Table 3 shows the slice numbers which are achieved on our SGI Onyx2 system with two Infinite Reality2 pipes. Full size means the object fills one wall entirely, half size means that the object fills only a quarter of a wall. The test object is the 256x256x120 voxels skull, which was also used in the evaluation's scenario #1.

Table 3. Rendering Performance

Frame Rate [fps]	**#Slices, Full Size**	**#Slices, Half Size**
1	381	381
2	172	381
5	65	255
10	32	93
15	16	53
20	9	19

10 Conclusion and Future Work

We have presented a volume rendering system for virtual environments which includes interaction elements for adjusting the transfer function, a probe mode, and a clipping plane. The evaluation of the system shows its general usability, but it points out a number of issues to be solved in future versions. We were surprised at how little the users were disturbed by the relatively poor image quality of large volumes, but this may in part be due to the existence of the high quality mode.

Beyond solving the usability issues, future work on the system should concentrate on two fields:

First of all the rendering quality must be improved. This can either be accomplished by using faster texturing hardware, so that more texture slices can be drawn for each frame. Or the rendering algorithm can be improved, for example by applying level-of-detail strategies.

Secondly, a more diverse set of direct interaction techniques with the volume data and more functionality for the transfer function editor needs to be developed. The evaluation showed that different kinds of user groups have entirely different requirements, resulting from their profession, which cannot be solved with one single implementation.

11 Acknowledgements

This research is supported by the Deutsche Forschungsgemeinschaft within the SFBs 382 and 374. The car cabin data set is courtesy of the DaimlerChrysler AG. The Visible Human Data Set is courtesy of the National Library of Medicine and the Visible Human Project. The statistical FE simulation data was used with the kind permission of J. Marczyk, EASi, and SGI. The COVISE visualization system is a common development of the HLRS Visualization Department and its spin-off company Vircinity, which offers COVISE commercially.

12 References

[Bev91] N. Bevan, J. Kirakowski, J. Maissel, “What is usability?”, in H.-J. Bullinger (ed.), Human Aspects in Computing: Design and use of interactive systems and work with terminals, Amsterdam, 1991

[Bra95] R. Brady, J. Pixton, G. Baxter, P. Moran, C.S. Potter, B. Carragher, A. Belmont, “Crumbs: A Virtual Environment Tracking Tool for Biological Imaging”, IEEE Symposium on Frontiers in Biomedical Visualization Proceedings, 1995

[Cru93] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, “Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE”, SIGGRAPH 1993 Proceedings

[Cul93] T.J. Cullip, U. Neumann, “Accelerating Volume Reconstruction With 3D Texture Hardware”, Technical Report TR93-027, University of North Carolina, Chapel Hill, 1993

[Lin99] R.W. Lindeman, J.L. Sibert, J.K. Hahn, “Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments”, CHI 99 Proceedings, pp. 64-71

[Ran95] D. Rantzau, et.al., “Collaborative and Interactive Visualization in a Distributed High Performance Software Environment”, Proceedings of the International Workshop on High Performance Computing for Graphics and Visualization, Swansea, Wales, 1995

[Ran98] D. Rantzau, K. Frank, U. Lang, D. Rainer, U. Woessner, “COVISE in the CUBE: An Environment for Analyzing Large and Complex Simulation Data”, Proceedings of the IPTW 1998

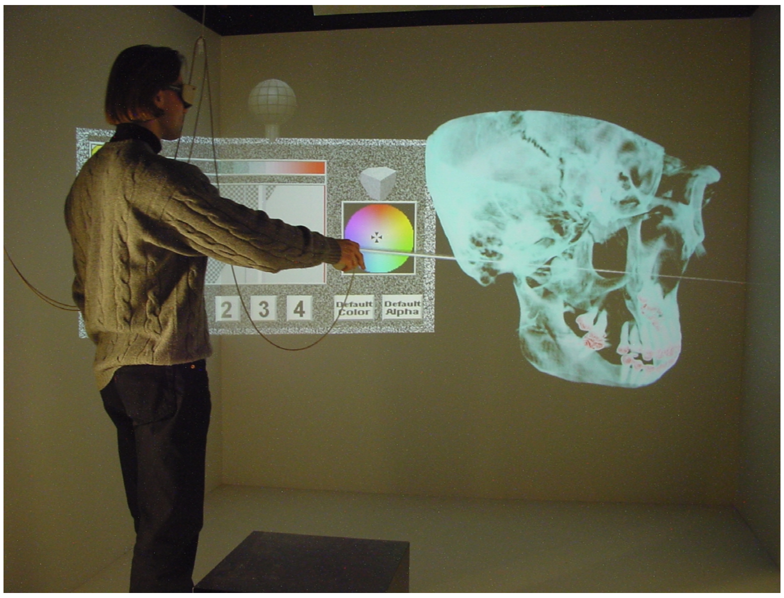
[Spi96] S. Spitzer, M.J. Ackerman, A.L. Scherzinger, D. Whitlock, “The Visible Human Male: A Technical Report”, Journal of the American Medical Informatics Association, 1996

[Swa99] K. Swartz, U. Thakkar, D. Hix, R. Brady, “Evaluating the Usability of Crumbs: A Case Study of VE Usability Engineering”, Proceedings of the IPTW 1999

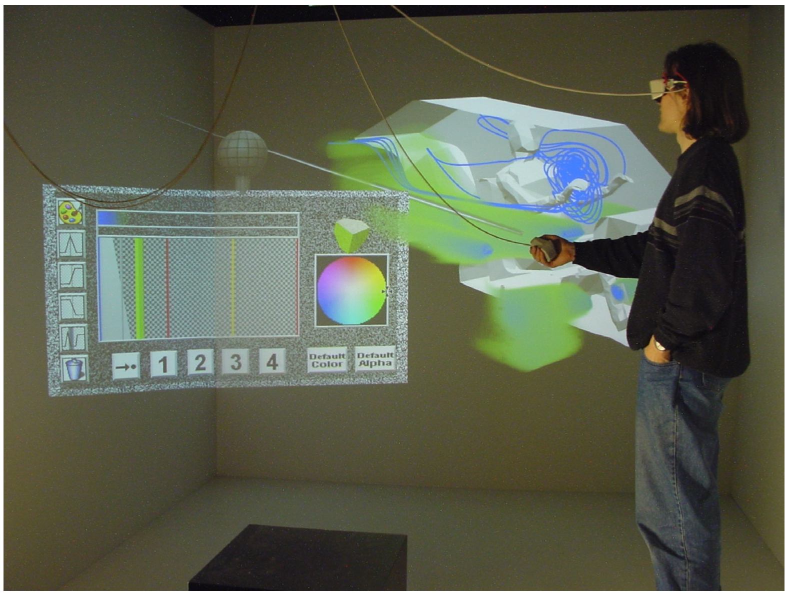
[Vol00] W.R. Volz, “Gigabyte Volume Viewing Using Split Software/Hardware Interpolation”, IEEE VolViz 2000 Proceedings

[Wei00] M. Weiler, R. Westermann, C. Hansen, K. Zimmermann, T. Ertl, “Level-Of-Detail Volume Rendering via 3D Textures”, IEEE VolViz 2000 Proceedings

[Woh00] W. Wohlfahrter, L.M. Encarnacao, D. Schmalstieg, “Interactive Volume Exploration on the StudyDesk”, Proceedings of the IPTW 2000



Scenario 1: Visible Human Skull (Schulze-Döbold et al., Plate A)



Scenario 2: Air Conditioning Simulation (Schulze-Döbold et al., Plate B)