InVIS - Interactive Visualization of Medical Data Sets

Jörg Meyer, Steffen Gelder, Timna E. Schneider, Hans Hagen
University of Kaiserslautern
Department of Computer Science
P. O. Box 3049
D-67653 Kaiserslautern
Germany

E-Mail: jmeyer@informatik.uni-kl.de

June 9, 1997

Abstract

Interactive rendering of large data sets requires fast algorithms and rapid hardware acceleration. Both can be improved, but none of this ensures interactive response times. If a scene is too complex, performance decreases, and neither faster algorithms nor speeding up hardware can guarantee interactive behavior. Certain timing characteristics should be incorporated in order to support such properties.

In our new approach we propose an interactive rendering pipeline with special timing predicates. We apply this technique to medical imaging, where large data sets derived from CT or MRI scans and CAD designs must be rendered in real-time with immediate monitoring feedback. Interactive behavior enables the user to manipulate and adjust the visualization straight on demand.

Keywords: Interactive Rendering, Time-Based Rendering, Pipelining, Volume Visualization, Surface Rendering, Hybrid Rendering, Medical Imaging, Scientific Visualization

1 Introduction

Current algorithms try to improve rendering times by speeding up computation routines or by making use of hardware features for accelerating image output. But neither faster algorithms nor faster hardware can guarantee interactive response times. If a system is slow, it is hard to achieve interactive behavior of the software. Advanced technologies in scientific visualization and medical imaging helped to improve image quality in diagnostics, therapy planning, and other medical applications. But if image quality increases, performance usually goes down rapidly.

To solve these problems, the algorithms should adapt automatically to system performance and computing power of a system, and they should also automatically adapt to complexity of a given scene. If these goals can be achieved, the system will be able to guarantee interactive response times, no matter how powerful the machine is. Of course, image quality will decrease on a less powerful machine, when there is not enough time between each display cycle to render the image in full quality, but the algorithm should make sure that at least a preview to the final image is provided, and when there is more time available (e. g. if no user input must be processed and if the scene does not change any more), the image will be rendered in full detail.

We use a pipelining concept for our interactive rendering system [11]. Each stage of the pipeline is designed to be scaled by control parameters. These parameters determine how much time can be spent at each stage in order to obtain a certain performance rate. To simplify matters, each stage should be scalable by a single parameter only. So we have a simple means of control for the whole pipeline. Initially, parameter settings are taken from a lookup table, which contains some startup values determined by heuristics and measurements for each step. After some iterations, the system is automatically adjusted to a pre-defined timing value for each display cycle.

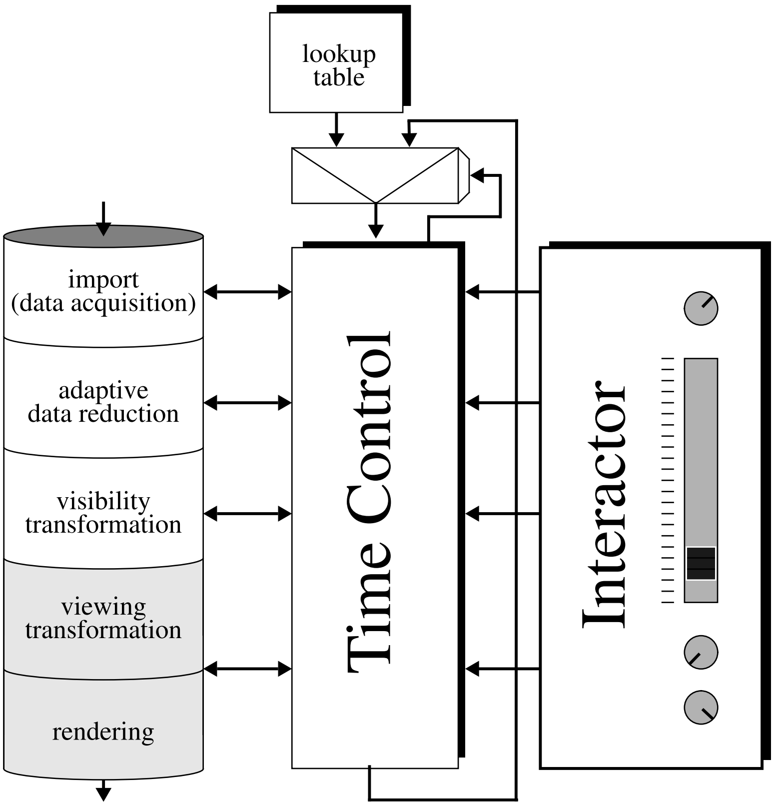

Fig. 1: rendering pipeline

Each step of the rendering pipeline is connected to the interactive control unit (ICU) via the time control unit (TCU), see figure 1. The time control unit makes sure that under all circumstances the data flow through the pipeline is guaranteed.

During data acquisition, which is the initial step of the rendering pipeline, selected features of the data set can be read or omitted from the original data set, thus dividing it into a visible and a non-visible portion.

In order to scale down complexity of a data set, adaptive data reduction can be applied for choosing a desired level of detail in an image. Visibility transformation also requires interactive control, because it involves selection of colors, transparency and material properties. For fast rendering, hidden surfaces and volumes can be removed or postponed for further image refinement. The viewing transformation, which is part of the graphics engine and which precedes the rendering stage, is located at the final step of the rendering pipeline. It determines perspective, and sure enough also requires interactive control and immediate feedback.

Even so our system runs on different platforms with different architectures and variable performance rates and nevertheless always ensures interactive behaviour, we still try to improve overall performance by optimizing each step of the pipeline. Each module can deal with original data as well as reduced data, because all modules are designed to be scalable and automatically adapt to scene complexity and pre-defined performance rates. In the following chapters we will introduce some of the techniques we used to accelerate the pipeline, and we explain the usage of the control parameter for each stage.

2 Data acquisition and processing

Data acquisition is the initial step of the rendering pipeline. For medical applications, it is often required to visualize real data, e. g. 3-D reconstructions from radiological scans, together with artificial objects, either structuring elements, such as rulers, planes and simulated rays, or hand-modelled items, such as prosthetic implants, surgery tools etc. [23]

Our rendering pipeline is designed to be capable of visualizing volumetric data from different origins (CT, MRI scans, etc.) as well as geometric data in different representations (polygon data, surface data, CSG objects, etc.).

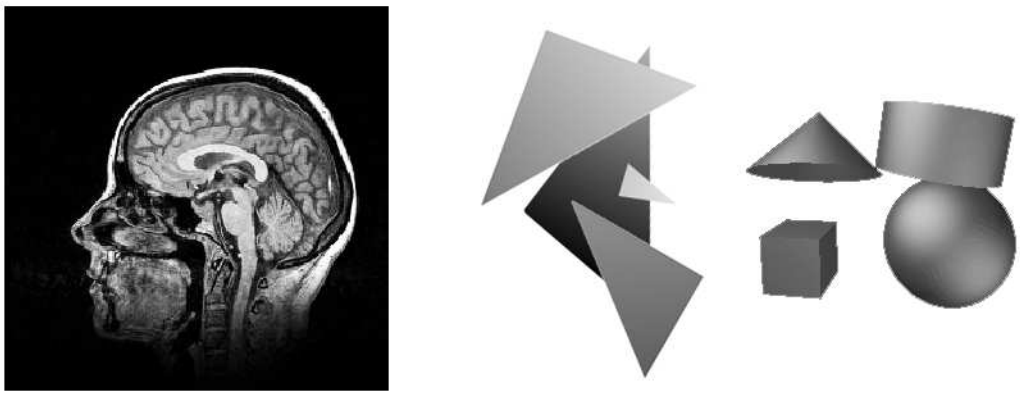

Figure 2: data representation
volumetric: MRI, geometric: polygons, CSG

In order to maintain precision and accuracy of data when combining different kinds of data sets, we try to avoid converting one data set into the representation of another in order to achieve a common data structure [12]. Assets and drawbacks of some rendering techniques and their application to interactive visualization will be discussed in the following chapters.

2.1 Geometric and Volumetric Data

Data sets in medical applications potentially consist of different data types. When it comes to visualizing both data sets simultaneously, a common rendering algorithm is desired, which can handle volumetric data as well as geometric data. There are two different ways for using the same rendering algorithms for both data types. On the one hand, we can fit a geometric structure into a volumetric grid. Such a conversion would be advantageous, because the different stages of the pipeline could be used in the same manner for all different types of input data.

But the application of a non-preserving conversion algorithm always implies the introduction of inaccuracies and artifacts. For example, while positioning geometric data into a volumetric grid, the algorithm applies a discrete scan conversion to the original object. This results in a loss of detail due to discrete sampling, and requires approximations if data does not exactly fit into the regular volumetric grid.

On the other hand, if we try to convert volumetric data into a geometric representation, a surface reconstruction algorithm, such as “Marching Cubes” [9], will be applied. This causes problems when there is no distinctive or no real surface information inherent in the data set, or if weak contrasts prevent the system from detecting a valid surface by thresholding [10]. Nevertheless, initially we use an adaptive version of a surface reconstruction algorithm in order to obtain a rapid preview to the data set, and we use direct volume rendering with pre-segmented data, i. e. labelled data, for the final image [16].

Therefore we make use of a new pipelining approach for rendering volumetric and geometric data simultaneously. This method is called *pipelined hybrid rendering*. In 1990, Levoy [7] introduced a raytracing approach for hybrid rendering. For each pixel, a ray is cast into the scene, scanning both data sets at the same time. Volumetric data is sampled at discrete positions along the ray, while geometric data is interleaved whenever the ray hits a surface. The main idea is to maintain the original representation for both data sets while passing them through all preprocessing stages of our pipeline. The same transformation and projection is applied to different data sets. All available data sets can be processed simultaneously, and, as a side effect, all stages and alternative pathways of the pipeline can run in parallel mode, thus increasing overall performance and decreasing response time for the user interface. This is one of the major advantages of the concept.

2.2 Hybrid Data

We use different data structures for each kind of data set. As already mentioned in the previous section, we try to maintain the original representation throughout the pipeline. Although the data set might be reduced or approximated, it always keeps its original data structure.

Volume data, as provided by CT or MRI scanners, is represented as a three-dimensional grid of scalar values, usually arranged in a set of sequential slices [6]. It is fed into the pipeline through a multipurpose import module, which can read density (grey level) data and label data for segmentation in any resolution and in any file or byte order. It allows interactive removal and selection of ROIs (*regions of interest*). This is the initial access point for the interactor and the time-control module.

Geometric data appears in various representations. Polygon data (triangles) and CSG objects (*constructive solid geometry*, [2]) are imported through special description files. Similar to volumetric data, the data set can be clipped in order to select ROIs.

Figure 3 shows a CT scan of a human head (volumetric data set), merged with a geometric data set, showing a surgical knife (CSG object) as well as some semi-transparent textured planes (original data mapped on some triangles). The image has been rendered on a “Distributed Volume Priority Z-Buffer” (*DVPZB*, [6]), which is a preliminary system that helps us to determine which algorithms can be optimized for speed in order to achieve interactive frame rates.

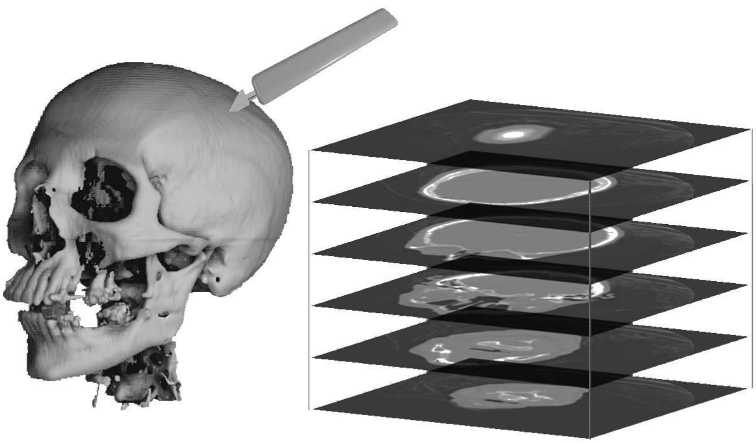

Figure 3: human skull (hybrid rendering)

3 Scalable Data Reduction

Different types of data sets require different techniques of data reduction. In our system, we try to apply algorithms which maintain data in the original representation. The topology of a data set should be unchanged, and the approximation should represent the data set with the best accuracy that can be achieved at a specified level of detail. We use several standard algorithms, which we have adapted in a way that they can be controlled by a single parameter. The following section describes the control mechanism.

3.1 Decimation of Volumetric Data

Several concepts have been published for rendering and decimation of volumetric data sets [22]. Levoy [8] uses a binary octree for space subdivision. Using this structure, transparent regions can be skipped in order to speed up visualization. Wilhelms and van Gelder [20] propose a space-efficient subdivision algorithm, called *branch on need* (BON).

Some algorithms use hierarchical data models and special rendering techniques, such as volume splatting [5][19], or exhaustive compression techniques, suitable for efficient storage and transmission of data [3]. Putting much effort in preprocessing for faster and easier access in the rendering stage seems to be appropriate for interactive systems [1].

An algorithm, which is one of the best suited to achieve the goals which have been pointed out previously, is a multiresolution wavelet approach [17]. This technique requires allows us to reduce complexity of a data set at different levels of detail. It requires fairly extensive preprocessing, but yields fast evaluation at rendering time. The algorithm is scalable, and thus provides a means for visualization of a data set at different resolutions. The transformation preserves most of the features of a data set, neglecting for example high-frequency components in the image only, i. e. fine details, without grouping or clustering the data set into blocks. In contrast to simple node removal algorithms, it preserves gradient information and relevant properties of a data set, without the risk that some important features are hazardously omitted from the grid. A single control parameter is used, which gives us immediate control over the number of basis function coefficients.

A 3-D version of the algorithm, which works on a grid (instead of a 2-D slice) and therefore achieves even better compression rates, is currently under development.

3.2 Decimation of Geometric Data

The geometric data set is subdivided into two groups: polygon data (triangles) and CSG objects. For polygon data, standard algorithms, as described by Schroeder et al. [14], Turk, or Hoppe, can be applied.

Turk [18] tries to improve the effectiveness of his algorithm by spreading a set of new vertices onto the surface of the model. A *mutual tessellation* between the original data set and the new vertices is used as an intermediate polygonal model, which is then optimized by locally removing most of the original points and creating a local retriangulation of the neighborhood. If each edge of the data set belongs to exactly one or two triangles only, the algorithm produces a new model of the object without changing its topology.

Hoppe, DeRose et al. [4] use the same technique of spreading new points onto the surface. They use an energy function to minimize the original data set. This function takes into account the deviation of the mesh geometry from the original mesh, the number of vertices, and a regulating term with a spring constant that helps the optimization process to find local minima. The algorithm reduces the number of vertices in a mesh in a significant way. Vertices are concentrated in regions with high curvature, and long edges are aligned in directions of low curvature. This method guarantees accuracy of data, since sharp features of the original mesh are retained.

We use an adaptive version of the latter algorithm. A single parameter controls the density of the new mesh. A mesh with low density can be rendered much faster than a detailed mesh with high density. Previously calculated meshes can be stored in order to provide fast access to different levels of resolution. If there is not enough time to calculate a new mesh, a pre-calculated mesh which matches timing restrictions best is used.

CSG objects are usually less complex than polygon objects. Since they have been introduced in order to speed up rendering of complex object surfaces, it is generally not necessary to apply data reduction algorithms in order to increase performance.

4 Transformations

Visibility transformations include coloring and transparency as well as hidden surface or hidden volume removal. Since coloring depends on a look-up table for material labels, there is not much to do for a time-dependent scalable algorithm. This stage can only be skipped in order to speed up the visualization. In this case, a standard color and transparency will be applied.

A hidden object removal algorithm can be applied, if the view point does not change much during a rendering sequence. When there is more time left, missing objects, previously hidden by other objects, can be added to the visibility transformation and passed on to the rendering stage. A single, time-dependent parameter controls whether hidden object removal is necessary or not.

5 Interactive Visualization

Interactive visualization of large medical data sets is a trade-off between quality and speed. Fast rendering can be achieved by data reduction, preprocessing of the data set, and a powerful graphics engine. But image quality must not be neglected. For medical applications, scalable data reduction algorithms should preserve accuracy of data. They should provide different levels of detail.

Interaction can be applied to all stages of the pipeline, as suggested in the first chapter. We have already discussed data acquisition, adaptive data reduction and visibility transformations. The final stage of the pipeline, i. e. the display stage, consists of a viewing transformation and a final rendering stage. The viewing transformation, which is influenced by user interaction, can also be controlled by a single parameter, which determines the *event resolution* in time steps, i. e. the period in which user events are passed through to the user interface. Note that this parameter is independent of the frame refresh rate.

The rendering stage makes use of an Open GL interface to the graphics engine, which makes the algorithm independent from hardware specifications and enables us to run our software on a high speed workstation as well as on a regular personal computer. The algorithm automatically adapts to system performance.

The system is designed for various platforms, namely UNIX (IRIX 5.3 with OpenGL and HP-UX with MESA, an OpenGL emulator for the X Window system), and Microsoft™ Windows NT. Figure 4 shows the graphical user interface (*GUI*) for the Windows NT version of our InVIS (*Interactive Visualization*) system. On the left, some controls for the different stages of the pipeline are shown, such as data selection, display representation and material selection. The second column provides some 2-D preview tools, i. e. browsers for the slices in each direction. On the right, a large drawing area for 3-D reconstructions is located, which enables the user to manipulate and evaluate the data set.

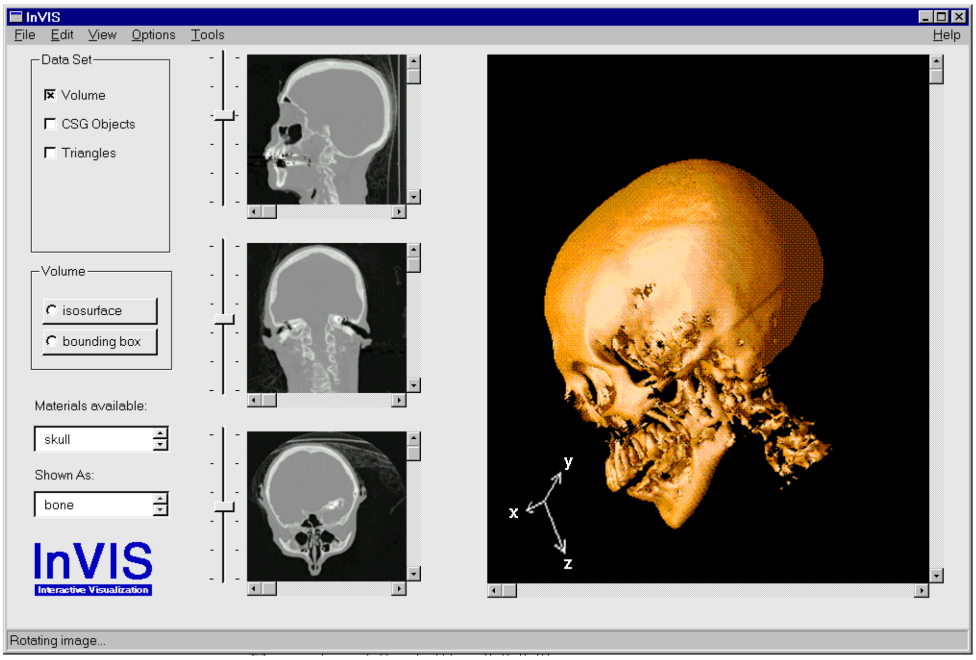

Fig. 4: InVIS user interface for MS Windows NT

6 Conclusion

The InVIS system is a 2-D / 3-D rendering tool for hybrid medical data sets. It provides interactive frame rates on any platform. Overall performance is increased by putting as much effort as possible in the preprocessing, i. e. the initial stages of the pipeline. Most of the final rendering is executed by a fast graphics engine, which allows interactive manipulation on perspective and size, and immediately responds to user interactions.

Some operations, which would require more time than is available for interactive rates, are subdivided into hierarchical steps of execution. First, a coarse visualization is created, which represents the data set in less detail. Later, if no user action happens, a refined image is calculated in hierarchical time steps.

Interactive behavior is maintained throughout the visualization process. A temporary lack of image quality is compensated by post-processing, i. e. refining, the final image. The best results were obtained on a Silicon Graphics Indigo2 Extreme workstation and on a Silicon Graphics O2. Final charts (including a dual-processor Pentium 200 MHz system) will be given during the demonstration.

A multi-user operating system, such as UNIX (without real-time extensions), can not always guarantee fixed frame rates and reproducible response times. But in general, interactive behavior and the user's desire for immediate response and feedback is fulfilled much better with our time-based approach than with time-independent models.

References

[1] Challinger, Judy: "Scalable Parallel Volume Raycasting for Nonrectilinear Computational Grids"; *IEEE Parallel Visualization Workshop*; October 1993

[2] Foley, J. D.; van Dam, A.; Feiner, S. K., Hughes, J. F.: *Computer Graphics*; pp. 557-560, 672, 901; Addison Wesley, July 1995

[3] Fowler, James E.; Yagel, Roni: "Lossless Compression of Volume Data"; Ohio State University, *preprint*

[4] Hoppe, Hugues; DeRose, Tony D.; Duchamp, Tom; McDonald. John; Stuetzle, Werner: "Mesh Optimization"; *Computer Graphics Proceedings, Annual Conference Series*; pp. 19-26; 1993

[5] Laur, David; Hanrahan, Pat: "Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering"; *ACM Proceedings, Computer Graphics, Vol. 25, No. 4*; pp. 285 - 288; July 1991

[6] Lengen, Rolf H. van: "The Volume Priority Z-Buffer"; *Focus on Scientific Visualization*; Hagen, H.; Müller, H.; Nielson, G. M. (publ.), Springer Verlag, New York; October 1992

[7] Levoy, Marc: “A Hybrid Ray Tracer for Rendering Polygon and Volume Data”; *IEEE Computer Graphics and Applications, Vol. 10, No. 3*; 1990

[8] Levoy, Marc: “Efficient ray tracing of volume data”; *ACM Transactions on Graphics, Vol. 9, No. 3;* pp. 245 - 261; July 1991

[9] Lorensen, William E.; Cline, Harvey E.: “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”; *ACM Proceedings, Computer Graphics, Vol. 21*; SIGGRAPH ‘87, Anaheim; July 1987

[10] Meyer, Jörg: *Hybrider Raytracer mit Anti-Aliasing*, University of Kaiserslautern, Germany; 1993

[11] Meyer, Jörg; Gelder, Steffen; Kretschmer, Kay; Silkenbäumer, Karsten; Hagen, Hans: “Interactive Visualization of Hybrid Medical Data Sets”; *Proc. of WSCG ‘97, Plzeň, Czech Republic*; pp. 371 - 380; February 1997

[12] Mosher Jr., C. E.; Johnson, E. R.: “Integration of Volume Rendering and Geometric Graphics”; *Proceedings of the Chapel Hill Workshop on Volume Visualization*; Chapel Hill, HC; May 1989

[13] Rossignac, Jarek: “Representations, Design and Visualization of Solids and of Geometric Structures”, Tutorial; *Eurographics ‘93*, Barcelona; pp. 16-26; September 6-7, 1993

[14] Schroeder, William J.; Zarge, Jonathan A.; Lorensen, William E.: “Decimation of Triangle Meshes”; *ACM Proceedings, Computer Graphics, Vol. 26, No. 2*; pp. 65-70; July 1992

[15] Silkenbäumer, Karsten; “Implementierung von Constructive Solid Geometry (CSG)-Objekten in ein hybrides Visualisierungssystem”; University of Kaiserslautern, Germany; December 1996

[16] Sobierajski, Lisa; Cohen, Daniel; Kaufman, Arie; Yagel, Roni; Acker, David E.: “A fast display method for volumetric data”; *The Visual Computer, Vol. 10*; pp. 116-124; 1993

[17] Stollnitz, Eric J.; DeRose, Tony D.; Salesin, David H.: “Wavelets for Computer Graphics: A Primer, Part 1 & 2”; *IEEE Computer Graphics and Applications;* pp. 76 - 84 / pp. 75 - 96 resp.; May / July 1995

[18] Turk, Greg; “Re-Tiling Polygonal Surfaces”; *ACM Proceedings, Computer Graphics, Vol. 26, No. 2*; pp. 55-64; July 1992

[19] Westover, Lee: “Footprint Evaluation for Volume Rendering”; *ACM Proceedings, Computer Graphics, Vol. 24, No. 4*; pp. 367 - 376; August 1990

[20] Wilhelms, Jane; Van Gelder, Allen: “Octrees for Faster Isosurface Generation”; *ACM Transactions on Graphics, Vol. 11, No. 3*; pp. 201 - 227; July 1992

[21] Wyvill, Geoff; Trotman, Andrew: “Exact Ray Tracing of CSG Models by Preserving Boundary Information”; *Department of Computer Science*; University of Otago; Dunedin; New Zealand; pp. 411-428

[22] Yagel, Roni; Ebert, David S.; Scott, James N.; Kurzion, Yair: “Grouping Volume Renderers for Enhanced Visualization in Computational Fluid Dynamics”; *IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 2*; June 1995

[23] Yagel, Roni: “Realistic Display of Volumes”; *Image Capture, Formatting, and Display, Vol. 1653*; pp. 470-476; 1992

About the Authors

JÖRG MEYER is Research Assistant and Ph. D. student at the University of Kaiserslautern, Germany. He received his M. S. in 1995 from the University of Kaiserslautern and specialized in the field of volume visualization, hybrid rendering, and medical imaging. His current research focusses on new rendering and display technologies and interactive volume visualization.

STEFFEN GELDER is currently working on his master thesis in computer science at the University of Kaiserslautern, Germany. His research centers on Scientific Visualization and Medical Imaging.

HANS HAGEN is full professor of computer science at the University of Kaiserslautern since 1988. He received his M. A. in 1979 at the University of Freiburg, Germany, and his Ph. D. in 1982 from Mathematics Department at University of Dortmund, Germany. His research interests include Geometric Modeling, Scientific Visualization, Computer Aided Geometric Design (CAGD), and Computer Graphics.

TIMNA E. SCHNEIDER is currently preparing her master thesis in computer science at the University of Kaiserslautern, Germany. She is an expert in wavelet compression and data reduction.