An Open and Extensible Framework for Visualization

Christoph Bastuck, Timo Hambuerger, Thorsten Hof, Maik Keller,
Peter Kohlmann, John Mehnert, Stefanie Nowak

christoph@jeremys-fate.com, timo.hambuerger@gmx.de, thof@cityweb.de,
kellermaik@gmx.de, Peter.Kohlmann@gmx.de,
john_mehnert@web.de, Stefanie.Nowak@t-online.de

Supervisor: Dr. Chistof Rezk-Salama, Prof. Dr. Andreas Kolb
University: University of Siegen
Departement: Computer Graphics and Multimedia Systems
Area of operation: Practical term, Student research project

Abstract: The paper presents a uniform framework for scientific visualization, that comprises a variety of different filtering, mapping and rendering techniques. We offer architectural concepts that implement a novel approach to extensibility on different levels. We have developed an open and expandable model for intuitive user interaction, which can be adapted to meet the demands of different scientific communities.

1 Motivation

The visualization of scientific data is of great importance in many scientific disciplines, such as medicine, natural and computational science and engineering. In recent years a huge variety of sophisticated visualization techniques have been developed by numerous research groups all over the world. Simultaneously, computer graphics applications have moved from expensive supercomputers to low-cost desktop computers, due to an increase in computational power of graphics processor which outperforms Moore's law.

To account for the high complexity inherent in scientific data fields, visualization has become a creative and investigative task, which requires a high degree of user interaction. In an interactive process, the user analyzes the data field, reveals and examines different aspects and the interrelationships which are hidden within the large amount of data.

2 Architectural considerations

Our aim was to create a uniform visualization framework which does not restricted itself to a designated application area or a specific data origin. After the general framework is implemented, specific applications can be build on top of it. In consequence, we focus on three main aspects, the visualization framework must fulfill:

1. Open architectural design with easily expandable functional range
2. Extensibility to arbitrary multidimensional and multivariate data
3. Intuitive usability for non-expert users

The task of creating images from abstract data fields can be described using several levels of abstraction. On the highest level, the process of image generation is considered a three-tier approach, frequently quoted as the *visualization pipeline*.

The first tier is a *preprocessing* step, where filters are applied that manipulate the data. In the second tier, the *mapping* step, the output of the preceding step is mapped to geometric shapes and visual attributes. In the third tier, the *rendering* step, images are created as the result of the visualization.

The first step contains several pipeline elements which can be arranged in a *chain*. The user must have full control over the configuration chain at runtime. Simultaneously, the system must assure consistency among its elements. Additionally, the architecture allows the user to implement new elements such as filters and converters. The same idea was applied in the second tier. The user can write his own mapping procedures, which extends the functional range of the framework. In order to meet all the requirements of current and future applications, the framework has to support multidimensional datasets (e.g. 2D and 3D) with store multiple data fields (e.g scalar, vector or multivariate data) that are discretized on a common grid (e.g. uniform, curvilinear or unstructured).

As a result of the open architecture, arbitrary visualization tasks can be achieved with respect to usability. The configuration of the visualization pipeline is decisive for the results of the visualization process. Hence, its graphical representation is the interface for user interaction, element insertion and deletion as well as configuration of individual elements. Its handling should be intuitive and mostly independent of the user's previous knowledge. Therefore, it is important to provide sufficient abstraction from the pure technical view of the pipeline [BKM91]. In addition to the flexible pipeline configuration, a multi-view concept allows the user to choose at runtime from a set of possible views onto the pipeline. This set can be expanded by implementing custom views. As a result, we can adapt the usability to different approaches of user interaction.

3 Implementation

We used the concepts of object oriented programming to achieve openness. Abstract base classes and polymorphism are used to define basic interfaces, that allow the framework to be extended at any time. An abstract class is built for all pipeline elements that includes basic pipeline functionality, a runtime type recognition system and functionality concerning data flow and data manipulation. Especially efficient operations for execution of linear filters and an effective mechanism for allocation and deallocation of system memory, during traversal of the pipeline are implemented. An abstract base class for data conversion extends the set of pipeline elements. Converters change the type of data by executing a characteristic function. As an example, we implemented 3D line integral convolution [RSHTE99] as a converter that generates a 3D scalar field out of a

3D vector field. Currently implemented mapping procedures comprise particle traces, 2D and 3D line integral convolution [CL93], stream surfaces and ribbons for vector field visualization [PVH+02] as well as texture based volume rendering [RS02] and *Marching Cubes* [LC87] for 3D scalar fields, only to mention a few.

We decided to implement manager classes, which assure compatibility among pipeline elements with respect to data type and the output of the preceding unit. Factory classes are used to achieve an open set of pipeline elements and mapping procedures, for an extensible framework that supports new pipeline elements and mapping procedures. To manage the variety of different data, we developed separate abstract classes for the grid and the variables, which provide base functionality and define a programming interface. Factory classes are used for instantiation. Since *multivariate* datasets can have multiple values at each grid point, we decided to create an (optionally compressed) archive that contains a raw data file for each variable. The grid structure is stored in a separate file. For type information, we added a meta file that holds information of the data using XML. The file format can be extended for future needs.

Since visualization requires creative and investigative skills, we designed the *TreeView*,

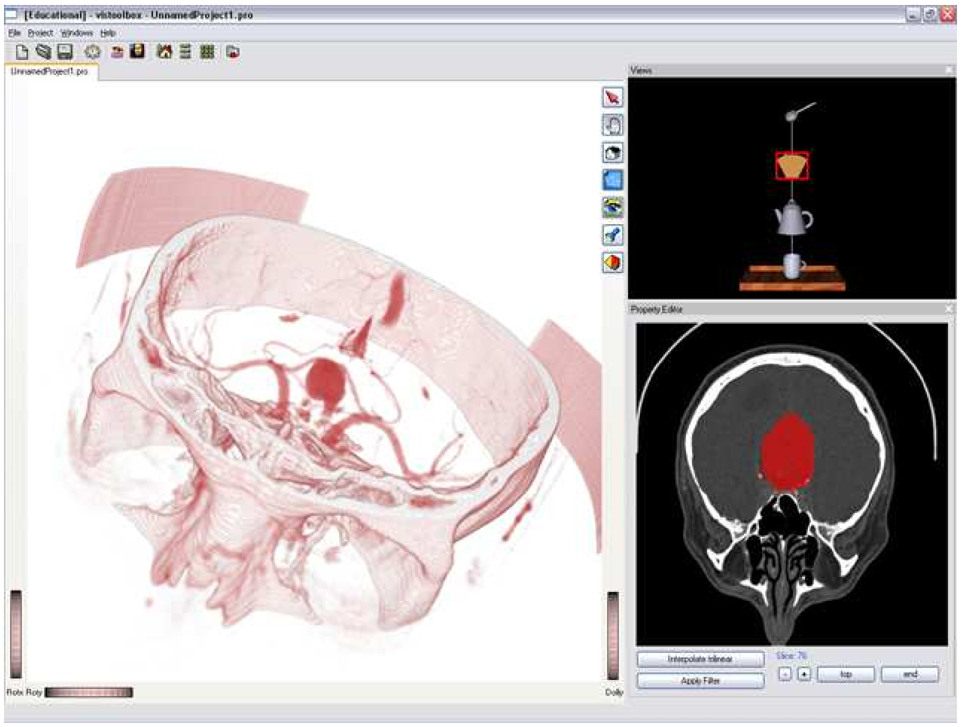

Figure 1: Example of the visualization framework: Main window with CT angiography of the human brain visualized by direct volume rendering, tree view *(upper right)* and the selection filter for volumetric data *(lower right)*

which represents an abstraction of the visualization pipeline. Icons, analogous to entities in everyday life, describe its elements and enable the user to configure the pipeline. Analogies are used to create a common understanding of the functionality by choosing entities we all can refer to. This approach offers a playful access and contributes to an intuitive handling. As different graphical representations of the visualization pipeline can be intuitive for different people, we designed a separate interface for user interaction to configure the pipeline. Through inheritance, new views can be implemented and easily integrated in the framework. At runtime, the user can select the view that meets his requirements.

4 Conclusion and Future Work

We presented a visualization framework that focuses on an open and extensible architecture, which we implemented in a one year project at the University of Siegen. The result of our work is available as *vistoolbox* under GPL open source license. By providing access to everyone we emphasize the extensible character of the framework. The open architecture and detailed documentation (doxygen) empowers users to contribute new functionality to the framework. The software is based on Trolltech Qt and SIM Coin 2.0 and runs on different platforms (Linux, WindowsXP, MacOS).

For future work we are planning to add new mapping procedures and other pipeline elements, especially converters that help to find coherences and thus create understanding of natural phenomena.

References

[BKM91] N. Bevana, J. Kirakowskib, and J. Maissela. What is Usability? In *Proceedings of the 4th International Conference on HCI*, Stuttgart, 1991.

[CL93] B. Cabral and L. Leedom. Imaging Vector Fields Using Line Integral Convolution. In *Proc. SIGGRAPH*, 1993.

[LC87] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Comp. Graphics*, 21(4):163–169, 1987.

[PVH$^+$02] F. Post, B. Vrolijk, H. Hauser, R. Laramee, and H. Doleisch. Feature Extraction and Visualization of Flow Fields. In *State-of-the-Art Proceedings of EUROGRAPHICS*, pages 69–100, Saarbrücken, Germany, September 2002.

[RS02] C. Rezk-Salama. *Volume Rendering Techniques for General Purpose Graphics Hardware*. Dissertation, University of Erlangen-Nuremberg, Germany, September 2002. Arbeitsberichte des Instituts für Informatik, Band 35, Nummer 5.

[RSHTE99] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive Exploration of Volume Line Integral Convolution Based on 3D–Texture Mapping. In *Proc. IEEE Visualization*, 1999.