Design and Implementation of an Immersive Geoscience Toolkit

Christophe Winkler, Fabien Bosquet, Xavier Cavin, Jean-Claude Paul

To cite this version:

Christophe Winkler, Fabien Bosquet, Xavier Cavin, Jean-Claude Paul. Design and Implementation of an Immersive Geoscience Toolkit. IEEE Visualisation'99, 1999, San Fransisco, CA, 5 p. inria-00098771

HAL Id: inria-00098771

https://hal.inria.fr/inria-00098771

Submitted on 26 Sep 2006

Design and Implementation of an Immersive Geoscience Toolkit

Christophe Winkler* Fabien Bosquet† Xavier Cavin* Jean-Claude Paul*

Abstract

Having a better way to represent and to interact with large geological models are topics of high interest in geoscience, and especially for oil and gas companies. We present in this paper the design and implementation of a visualization program that involves two main features. It is based on the central data model, in order to display in real time the modifications caused by the modeler. Furthermore, it benefits from the different immersive environments which give the user a much more accurate insight of the model than a regular computer screen. Then, we focus on the difficulties that come in the way of performance.

Keywords: visualization, geosciences, immersive environments, multi-pipe rendering.

1 Introduction

Integrating information coming from different sources, getting a better understanding of 3D geological objects' relationships, having a better way to represent and to interact with large geological models are topics of high interest in geoscience, and especially for oil and gas companies.

Geoscience toolkits must have extended modeling capabilities, in order to ensure the model's consistency. These capabilities are grouped in several different kinds of modeling tools: structural, geophysical (velocity modeling) or reservoir-specific (geostatistics, well planning...) tools. All these features are based on the interpolation capabilities of the modeler, and interact directly with a central data model.

Feedback to the user is mostly provided through the visualization of geological structures like horizons, faults, well paths or seismic data. This involves two main features of the visualisation tool: it must be based on the central data model, in order to display in real time the modifications caused by the different modeling tools; furthermore, it should benefit from the different immersive environments which give the user a much more accurate insight of the model than a regular computer screen.

We chose to extend the capacities of a standard geoscience program, GOCAD[1] [9], to enable the visualization of its model within different immersive environments, like CAVES™, REALITYCENTERS™ or workbenches.

In this paper, we will briefly describe the main points of the design and implementation of the program. Then, we will focus on the solutions chosen to adapt the software to immersive environments. This will lead us to present the main feedback and the difficulties that showed up during the experiments. Finally, we will conclude and present our future work.

*INRIA, Nancy, France, {winkler, cavin, paul}@loria.fr

†T-Surf Corporation, Houston, TX, bosquet@t-surf.com

[1]The GOCAD (Geological Object Computer Aided Design) software is maintained and distributed by T-Surf `http://www.t-surf.com`

2 System overview

We based the immersive toolkit on GOCAD's library which implements a whole set of modeling functionalities.

The displayed geological structures are organised into a scene graph referring to the corresponding data in the central data model. This enables data duplication to be minimized, and facilitates the visual update when the model is changed.

2.1 Modeling process

The purpose of 3D modeling in the geosciences is to create a realistic representation of the underground. It may be thought of as a partition of the 3D space into regions, bounded by interfaces between different rock layers, and discontinuities corresponding to faults. An accurate representation of the geometry is required to enable the user to have a global view of the structures to be modeled.

The main difficulty raised by geological modeling is the diversity of the initial data available, both in terms of nature and in terms of reliability.

- Wells provide exact data, in the form of well markers, indicating which geological layer is traversed at a given depth. But wells are very expensive to dig, and only a small number of them are available, irregularly scattered all over the domain of interest.
- Seismic acquisitions provide data by causing an explosion and by measuring the vibrations reflected and refracted by the underground structures. Combining the data given by a large number of captors results in a 3D seismic cube, similar to a radiography of the underground. One should keep in mind that due to the different causes of precision loss, induced by the captors and by the computations involved in the reconstruction of the seismic cube, this information is quite inaccurate.
- The last kind of information to be taken into account is the *a-priori* knowledge of geologists concerning the domain to be studied. For this reason, the modeling process should be *slightly interactive*, which means that a solution should be automatically provided to the user, who then has the ability to edit it and to inject more information into the system.

The challenge of geologic modeling is to take into account various kind of information, such as well markers, sets of points corresponding to interfaces and faults picked from seismic cubes, while letting the user interactively introduce additional information at each step of the process (fig. 3). The resulting model is named a structural model, since it represents the geometric structure of the underground.

Except for a few specific objects (the wells, channels and lenses), the geological objects' representations in GOCAD are based on regular or irregular meshes (point sets, polygonal lines, triangulated surfaces, tetraedralized solids and regular or stratigraphic grids). These meshed based objects are well suited for a *Boundary Representation* (B-Rep) modelisation, first represented by *winged edge* structures [1, 10]. Lately, B-Rep models were generalised by *cellular models* represented by *Generalized Map* (G-Map) structures [6]. These representations allow for a topological representation of the model (with neighbour relationships), and the geometry can be considered as one specific property.

The model can be provided with values representing properties such as porosity or permeability. Any property defined on mesh nodes can be interpolated across the model with an original method, the *Discrete Smooth Interpolation* (DSI) [7].

In a geomodeling project, several teams cooperate and exchange information in a form specific to each team. Structural geologists manipulate surfaces, geophysicists use tetraedralized solids, and reservoir engineers prefer hexaedral grids. GOCAD's library federates the modeling process into a coherent formalism and a common set of tools working on a single database. The main concerns are:

- the generation and interactive edition of geological objects constrained by well markers and seismic data;
- the generation of structured and unstructured meshes;
- the statistical analysis of properties;
- geostatistic and stochastic simulations (generation of equiprobable models).

In any case, one of the most common way to exploit the model is clearly to visualize it.

2.2 Data visualization

The standard visualization process is the following: first, the user selects the different geological objects to be displayed; then, an internal scene graph is built according to these objects and their properties (color, shading, transparency...); finally, this graph is traversed for the rendering stage.

In order to minimize the duplication of data, every shape node of the scene graph points to the internal data model. This means that the geometrical representation of the shape is built out of the topological model. For example, for a triangulated surface, the rendering algorithm has access to a list of triangles. But for efficiency considerations, a list of triangle strips will be generated: beginning with one triangle of the surface, the algorithm will add one of its neighbours that can be retrieved through topological relationships. The triangle strip ends when no other neighbour can be found for the last triangle. Triangle strips are generated until all the triangles are rendered.

Flags are used to avoid displaying twice the same triangle. They indicate if the triangle has already been taken into account. Unfortunately, these flags are actually directly stored in each triangle. As will be shown later, this leads to nasty side effects when two threads have to use simultaneously these flags.

A home-made implementation has been developed to visualize volumetric data. Only parallelepipeds can be visualized as volumes. The core of the volume visualization algorithm proceeds by slicing the box by a set of planes [2]. They are either planes, axis-aligned according to the box's coordinate system or perpendicular to the view direction. The number of slicing planes is controlled by the user. Increasing this number enhances the quality of the volume rendering, at the cost of lowering the frame-rate.

The volumetric data is a 3D texture image which is decomposed into a set of bricks. Each brick is mainly a small box of volumetric texture, so that the set of bricks covers the cube. The bricks are then sorted from back to front depending on their distance to the camera. Then, depending on the graphics board's capability, they are sent down to the board as a 2D or 3D texture.

3 Multi-pipe visualization process

In order to give the user a better insight into the geological model, we decided to enhance the visualization process. Two major enhancements have been implemented. First, we extended the existing software to run in immersive environments, which represents a considerable task. Indeed, there are multiple hardware choices, and some important implementation considerations had to be taken into account. Second, we improved the visualization process in order either to have a better frame-rate or to display larger databases. Both improvements help to increase the interaction with the model.

3.1 Immersive environments

An immersive toolkit should be flexible enough to accommodate several environments, ranging from the simple user display (the common computer screen) to immersive VR environments. In these high end configurations, the global representation of the scene is displayed to the user by means of multiple sub-images projected on screens.

At first, we developed our own internal library allowing for an external hardware configuration file, and implementing multiple pipes management facilities.

The possibility of configuring the software without having to recompile anything is essential. As with the CAVE LIBRARY [8], this goal is achieved by having the hardware part of the virtual environment defined in an external configuration file (fig. 1). This file defines the number of displays, the screen geometries and their relative positions (given by the position of three corners of the screen), as well as the different devices and their specific parametrisations used to drive the visualization. Thus it is possible to adapt at runtime to any hardware environment.

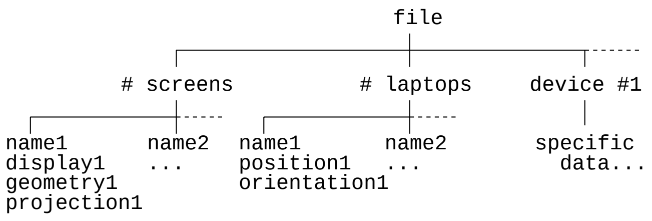


Figure 1: Configuration file.

Depending on the characteristics of each screen, a specific sub-image must be computed. These images are then displayed together, and are seen by the user as one large image. Synchronizing the display of the sub-images must be specially considered. Our implementation has a barrier that synchronizes the rendering threads just before they swap the frame-buffers. This must be applied throughout the whole rendering process, since nothing guarantees that the sub-images are computed equally fast. This solution avoids some visual artefacts which can occur when the different sub-images are not displayed exactly simultaneously. In this case, the global image would be distorted.

For efficiency, it is better if all the images are computed simultaneously. This is possible with computers having multiple graphics pipes or graphics cards, and implies an obvious parallelism of the visualization process. In this case, the scene graph traversal must be thread-safe.

Immersed in new hardware environments, the users expect an improved visualization, allowing a faster display of larger scenes.

3.2 Multi-pipe acceleration

In order to accelerate the visualization process, H. Igehy *et al.* [4] designed a parallel graphics interface. This API increases the graphics performance by lowering the communication overhead between the host system and the graphics system. Nevertheless, the presented solution does not yet address the pure graphics hardware limitations.

Another solution leading to a more efficient rendering is to allow multiple pipes to collaborate on one single image. The goal is to benefit from a graphics hardware parallelism. We ported our toolkit on SGI's MPU[2] library.

Different acceleration designs can be used:

2D composition : the image is divided into a number of sub-images computed by separate graphics pipes. Only a part of the scene is sent to each pipe, thus accelerating the rendering time. All the resulting images are then merged into one single frame buffer[3].

3D decomposition (fig. 2): the scene is divided in ordered sub-scenes, sorted in back to front order. Each sub-scene (A, B and C) is assigned to one pipe (respectively #1, #2 and #3)). The first pipe computes the image corresponding to its sub-scene (stage a). The resulting frame buffer is then copied to the next pipe (α), which *adds* the image corresponding to its sub-scene (stage b). This process is repeated for all the pipes to arrive at the final image. Furthermore, once one pipe has passed its frame buffer to the next one, it can simultaneously get the buffer from the preceding pipe and start its local rendering with new viewing parameters before the total image is finished.

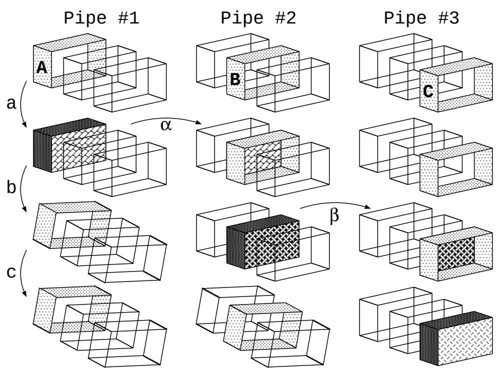


Figure 2: 3D composition.

4D decomposition: this time decomposition assigns to each pipe a different time-frame to be computed. While one pipe is computing the n^{th} frame, the following pipe computes the $(n+1)^{\text{th}}$ frame.

The use of the Monster MPU allows for a nearly linear speed-up on the SGI ONYX2.

[2]Actually, only MPU v2.0 (or Monster MPU) supports the presented acceleration capabilities. Contact P. Bouchaud for more information.

[3]The difference with immersive environments is that these environments rely on a hardware combination of the sub-images (for example with projectors), whereas the 2D decomposition results in one single image that can be displayed on one computer screen.

Another software improvement has been implemented in order to accelerate the rendering. The key point is that a single scene is displayed on multiple sub-images. When one object is completely displayed on one sub-image, it can not be simultaneously viewed on another sub-image. Thus, special care must be accorded for the view frustum clipping. This technique allows objects to be discarded before they are sent to the graphics pipeline, reducing the graphical work. Hierarchical methods, based on octrees or bsp-trees [3], may accelerate significantly the visualization process.

Actually, some considerations must be taken into account: the achieved software acceleration depends on the performance of the graphics hardware, the size of the displayed model and the ratio of the data which is seen. Depending on these factors, the obtained speed-ups range from one to about fifty.

4 Results and discussion

The immersive extension of the geoscience tool has already been successfully tested in various VR environments: in a CAVE™ (fig. 4) driven by four pipes, in various REALITYCENTERS™ with one or three pipes (fig. 5), or on workbenches... The simple ASCII configuration file provides efficient run-time portability of the software for any hardware environment. The user simply has to define the number of screens, their resolution and their relative position. This suffices to define at run-time the projections to be done to compute the different sub-images.

4.1 Interaction and interface considerations

The immersive visualization gives the users more freedom to interact with the model. Thus, they are tempted by manipulations they would not try with a window-based interface.

For example, a common request is to implement the possibility to pick an object in the scene, to drag it around to have a better view of this particular object (without having the whole scene moving around), and then to drop it so that it automaticly *snaps* to it's original position.

Meanwhile, the main difficulty for the users is to adapt to the new environment. They have to switch from the standard screen/keyboard/mouse scheme, with the Graphical User Interface they are used to, to a new immersive environment without the concept of a frame or a window defining the work area. Users have the most difficulty adapting to the new interface, which gives access to the numerous functionalities of the modeler (about 700 functions).

We began to develop a time consuming solution, which consists in redesigning a brand new 3D interface from scratch. Unfortunately, no standard API exists to help building such interfaces.

A better alternative is to use video mapping techniques to display the standard interface on a panel (3D rectangle called *laptop*) embedded in the 3D world. The implementation is quite easy, and the user is again faced with the interface he is used to. Furthermore, any existing interface can be displayed on the laptop, without writing any new lines of code.

4.2 Parallelism considerations

Different solutions exist to parallelize an algorithm, either using a process-level parallelism based on the C function `fork`, or using threads like the posix threads[4] [5]. The main difference between both approaches concerns the way the memory is shared. When using processes, each process runs in a separate address space. Sharing data requires allocating special structures called *shared*

[4]The standard for UNIX threads is defined in ANSI/IEEE POSIX 1003.1-1995

memory objects. In contrast, the thread approach enables several tasks to run in the same address space. Thus, all the data structures and variables are shared between the threads.

In the case of a geoscience tool including modeling functionalities, it is essential to share the geological data model between the rendering and the modeling processes. Indeed, the functionalities based on interpolation algorithms can deeply change the whole database. Thus, the different rendering processes have to access these data in order to guarantee the consistency of the displayed scene and the model. This is one of the aspects that makes the thread solution best suited for this application, since the memory can not be easily shared with a `fork`-based approach.

Nevertheless, this solution still has some drawbacks: the whole application must be made thread-safe. The programmer has to ensure that the data will not be displayed while the modeler thread is performing any changes to the data. This can happen for example during an interpolation stage, that also relies on the above-mentioned flags. During a time-consuming interpolation, the scene graph is automatically informed that the model changed, thus the triangle strips are recomputed. This resets all the flags currently used for the interpolation...

5 Conclusion and future work

Various problems arose from extending the window-based sequential software to a multi-threaded program running in immersive environments with multiple pipes:

- the ability of the visualization tool to handle very large data models which are common in geoscience;
- the flexibility to deal with different multi-pipe environments, which implies a parallel rendering process;
- the side effects inherent to this parallelism must be fixed, in particular when both the modeling and rendering processes access the model's data.

Different improvements in the visualization process are undertaken:

- replacing the actual internal scene graph by a supported API like OPTIMIZER or even FAHRENHEIT. This will reduce the maintenance costs, and allow us to benefit from the latest hardware and OPENGL enhancements;
- exploiting the specific geometry of the superposed geological layers: techniques based on occlusion culling, implemented in OPTIMIZER or presented in [11], should prove to be very effective.

6 Acknowledgements

We want to thank the *Centre Charles Hermite* research center, for allowing us to access to their Origin2000 and Onyx2.

This work has also been supported by the SILICON GRAPHICS's center in Cortaillod, where we performed our tests on an Onyx2 with 4 pipes. A special thanks to Chris Henn for inviting us, to Jean-Jacques Pittet for managing their REALITYCENTER™, and to Patrick Bouchaud for working on the Monster MPU utility.

References

[1] B. Baumgart. A Polyhedron Representation for Computer Vision. In *AFIPS Nat. Conf. Proc.*, volume 44, pages 589–596, June 1975.

[2] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction using Texture Mapping Hardware. In *1994 Symposium on Volume Visualization*, pages 91–98, October 1994.

[3] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley, Reading, Massachusetts, 1990.

[4] H. Igehy, G. Stoll, and P. Hanrahan. The Design of a Parallel Graphics Interface. In *SIGGRAPH'98 Conference Proceedings*, Annual Conference Series, pages 141–150. ACM SIGGRAPH, Addison Wesley, July 1998.

[5] S. Kleiman, D. Shah, and B. Smaalders. *Programming with Threads*. SunSoft Press, 1996.

[6] P. Lienhardt. N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds. *Journal on Coumputational Geometry and Applications*, 4(3):275–324, 1994.

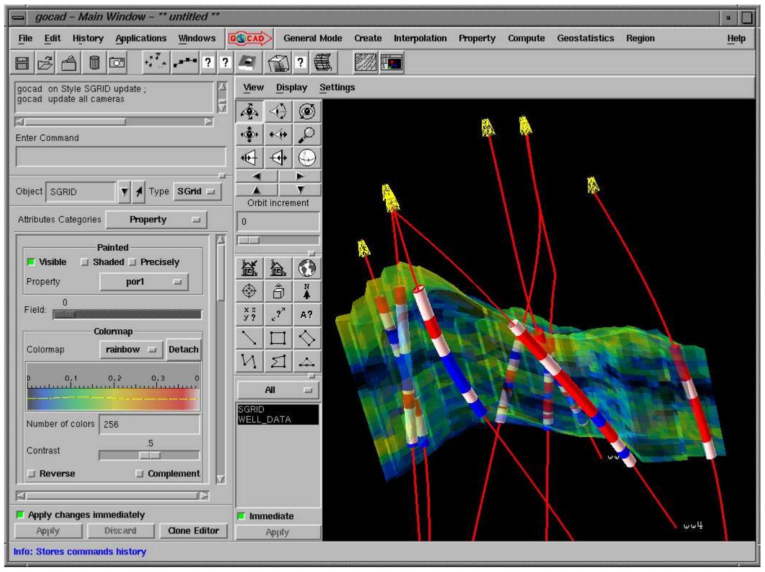
[7] J.L. Mallet. Discrete Smooth Interpolation. *Computer Aided Design Journal*, 24(4):263–270, 1992.

[8] D. Pape, C. Cruz-Neira, and M. Czernuszenko. CAVE Library. `http://www.evl.uic.edu/pape/CAVE/prog/CAVEGuide.html`. Electronic Visualization Laboratory, University of Illinois at Chicago.

[9] `http://www.ensg.u-nancy.fr/GOCAD`.

[10] K. Weiler. Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. *Computer Graphics and Applications*, 5(1):21–40, 1985.

[11] H. Zhang, D. Manocha, T. Hudson, and K. E. Hoff III. Visibility Culling using Hierarchical Occlusion Maps. In *SIGGRAPH'97 Conference Proceedings*, Annual Conference Series, pages 77–88. ACM SIGGRAPH, Addison Wesley, August 1997.


```
gocad  on Style SGRID update ;
gocad  update all cameras
```

Info: Stores commands history

Figure 3: GOCAD’s standard GUI, displaying wells and seismic data rendered with a given transparency.

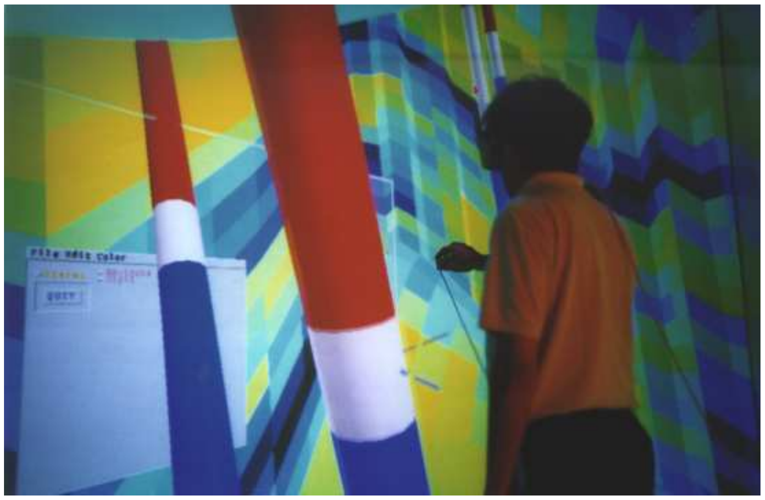


Figure 4: Visualisation of the same scene in a CAVE^{TM}-like environment with a four pipes configuration (courtesy of Arco).

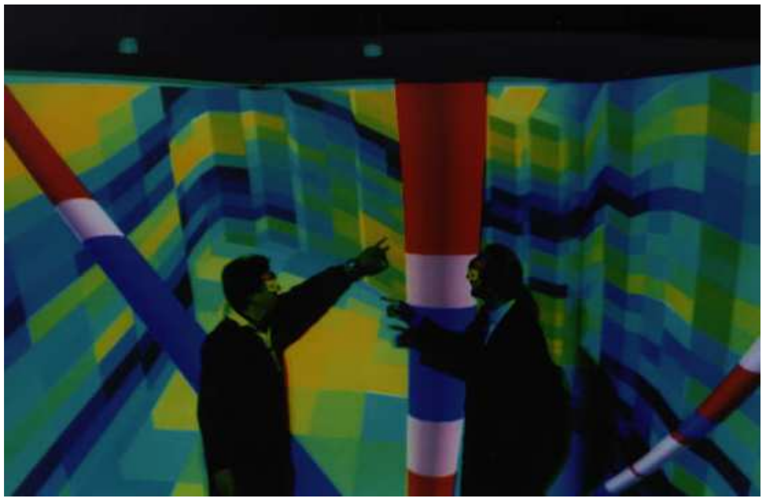


Figure 5: Visualisation of the same scene in a 3 pipes REALITYCENTER^{TM} (courtesy of SILICON GRAPHICS’s Cortaillod’s office).