DSVOL II - A Distributed Visualization and Sonification Application Communicating via an XML-Based Protocol

VERIDIANA CHRISTIE LUCAS SALVADOR[1], ROSANE MINGHIM[1], HAIM LEVKOWITZ[2]

[1] Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, Brasil
{veri, rminghim}@icmc.sc.usp.br

[2] Institute for Visualization and Perception Research, Department of Computer Science, University of Massachusetts Lowell, Lowell, MA, USA
haim@cs.uml.edu

Abstract: Distribution of the visualization process over the World-Wide Web allows remote access to expensive resources, and the cooperation amongst teams of experts that are located in different places. This paper presents an architecture and a corresponding prototype implementation of a distributed system for visualization and sonification of scientific data. The distribution is accomplished by the addition of visualization and sonification servers, accessible over the Internet, and implemented using an XML-based SOAP (Simple Object Access Protocol) protocol. The system is an evolution of a previously-developed prototype, with improvement in the visualization and in the distribution aspects of the architecture. Former distribution scheme employed CORBA (Common Object Request Broker). A discussion on the use of both distribution tools is given in the light of visualization tasks.

1 Introduction

Some classical problems in scientific visualization such as support for collaborative work, handling of large data sets, and taking better advantage of remote hardware resources can be tackled by using distributed architectures.

With the currently available technology it is possible to apply distributed computation concepts to many applications. That is the case for visualization. In the field of visualization many times systems do not reach a good number of potential end-users as fast or as effectively as they should. Part of the reason for that has to do with most systems being difficult to access, use, and install. Additionally, they are demanding in computational resources, and very rarely adapted to user's needs. Distribution of system components as well as providing access to the system on the Internet may come to contribute to the relief of some of the impairing aspects of visualization.

Although a research platform, the DSVol (Distributed Sound for Volumes) system intends to, among other things, work on ways to make visualization and sonification functions available to end users of real applications, in consonance with the goals of the project PowerVis, in development at ICMC – USP, Brazil, with cooperation with UMass Lowell, USA.

With that motivation, DSVol II is a prototype, evolved from a previous prototype of a system (DSVol), which implements an architecture for distributed visualization and sonification on the Internet, in a flexible and inexpensive way. The use of sound to convey information is called sonification. It can be a valuable tool in some visualization applications. In DSVol, a visualization server is responsible for executing demanding mapping tasks while a sonification server is responsible for calculating and processing sound streams, to be presented at the user end (the client), with rendering and display processed on the client side. The system is accessible on the Internet.

The implementation of this architecture, presented here, uses XML (eXtensible Markup Language) as a means to achieve data and parameter passing between system modules. This is compared with the previous version, which used CORBA as the distribution platform.

Interaction and sonification tasks have also evolved from a previous version, and are presented here. They are implemented in C++, as extensions to the visualization system. At the core of the system is VTK (The Visualization Toolkit), also developed in C++.

DSVol has a JAVA interface, which is the gateway to Internet access, communicating with the C++ code via JNI (Java Native Interface).

Section 2 presents an overview of previous work on distribution for visualization purposes. Section 3 describes the architecture of DSVol. Section 4 presents the interaction tasks in DSVol, responsible for triggering visual and aural mappings. Section 5 describes the implementation of the communication between modules using SOAP, a tool for communication using XML, and Section 6 outlines some of the conclusions drawn from this work.

2 Distributed Visualization

Some typical problems of visualization may be suitable for use of distributed architectures. Data sets are becoming too large and, sometimes, unbearable for conventional visualization systems and local disks. Many times these data sets are remotely located causing problems of bandwidth and latency of data transfer. An important feature of the research in scientific visualization due to its multidisciplinary nature is the collaborative work between researches in different areas, sometimes located at different places. These cases demand that data sets and results be shared. Visualization tasks are also demanding in computational resources that many times cannot be supported at the user's desktop. One way to improve resource access is by processing part of the visualization pipeline on a remote hardware.

One of the initial works on distributed architecture for visualization was presented by Ang, Martin, and Doyle [10]. They developed VIS, a visualization tool integrated into the Mosaic browser as a visualization service. VIS distributes the generation of 3D models from volume data amongst available hardware.

The VTK library was extended to allow the visualization of extremely large, time-varying data sets using parallel and distributed computing resources in a work developed by Ahrens et al. This work distributes processes via MPI and is not available over the Web [11]. Wood, Brodlie, and Wright discussed different distributed scenarios for implementing Web-based visualization platforms [12]. Engel *et al.* demonstrated how to utilize local low-end desktop and remote high-end graphics hardware for interactive visualization of tomographic image data combining local, remote, and hybrid rendering techniques [21]. Frisch and Ertl addressed a CORBA based connection between a finite element solver PAM-CRASH (from ESI Group, France) and the visualization tool CrashViewer (developed at Stuttgart University, Germany) [22]. Aeschlimann *et al.* used active frames, mobile objects containing data and programs to manipulate the data, to describe a framework for building interactive distributed visualizations (through a Ethernet network) of large scientific data sets, using the VTK library to perform the visualization tasks [13]. Zunino *et al.* presented a platform-independent visualization system for visualizing particle tracing in computational fluid dynamics, using a distributed architecture based on Sun Microsystems technology Jini to perform distribution tasks, and Java 3D library for the visualization phase [19]. Liere *et al.* presented a distributed blackboard architecture for simulation and scientific visualization using the TCP/IP protocol for communication between modules. The visualization module uses the VTK library to perform visualization tasks [20].

Alternative approaches to distribute visualization processes were investigated by several research efforts, such as collaborative environmental data analysis [12], remote surgical training [14], and remote access to volume visualization algorithms that operate on local user data [15].

Like some of the work quoted above, the DSVol prototype uses (and extends) the VTK library to perform visualization tasks. Unlike these, DSVol proposes a visualization system that provides visualization pipeline distribution over the Web. To perform distribution tasks, we adopt the use of SOAP [7], since DSVol is accessible over the Web and SOAP is an XML-based protocol developed to distribute objects over the Internet. XML [5, 6] is a format for structured documents and data on the Web.

The idea behind SOAP is to use well-formed XML as a wire protocol that can be transported to a remote system. With the development of specific technologies for remote exchange of information such as SOAP, it brings up questions on how they can affect the access of information in visualization, particularly when alternative channels of data mapping are available, as it is the case with sonification. The presence of different forms of representation (aural and visual in our case) poses an additional challenge to the visualization pipeline, in that various demanding tasks are done at the same time on the same data to result in sometimes different entities (geometric objects and sound streams) to be displayed together in the same scene.

The architecture presented here takes as a basis a pipeline distribution scheme and includes the sonification element in it, proposing an implementation via XML. It uses JAVA as a tool for Web programming, and VTK (with added extensions) as the core visualization software. We also compare the results with the previous version of the system, which used CORBA as the communication platform amongst remote modules, and comprised a more restricted number of interaction processes.

3 DSVol – Distributed Sound for Volumes

DSVol is a prototype of a distributed visualization system that uses sonification to support volume visualization tasks [1]. This prototype architecture comprises three modules integrated by a common Java interface: a client, which is responsible for interaction and sonification tasks; a visualization server, responsible for visualization calculation; and a sound server, responsible for the calculations involved in the sonification tasks (see Figure 1). The client invokes methods from the servers

defined in WSDL (Web Services Description Language) files. Each server describes its form of access through one of these files. DSVol modules are written in C++, and communicate with the Java interface via JNI (Java Native Interface) [2]. The Visualization Toolkit (VTK) [3], a visualization class library written in C++, is used to provide basic visualization functionality to client and servers. The communication amongst the client module and the visualization and sonification servers uses the XML-based protocol SOAP (Simple Object Access Protocol) [5, 6, 7]. These communication elements are described in detail in section 5.

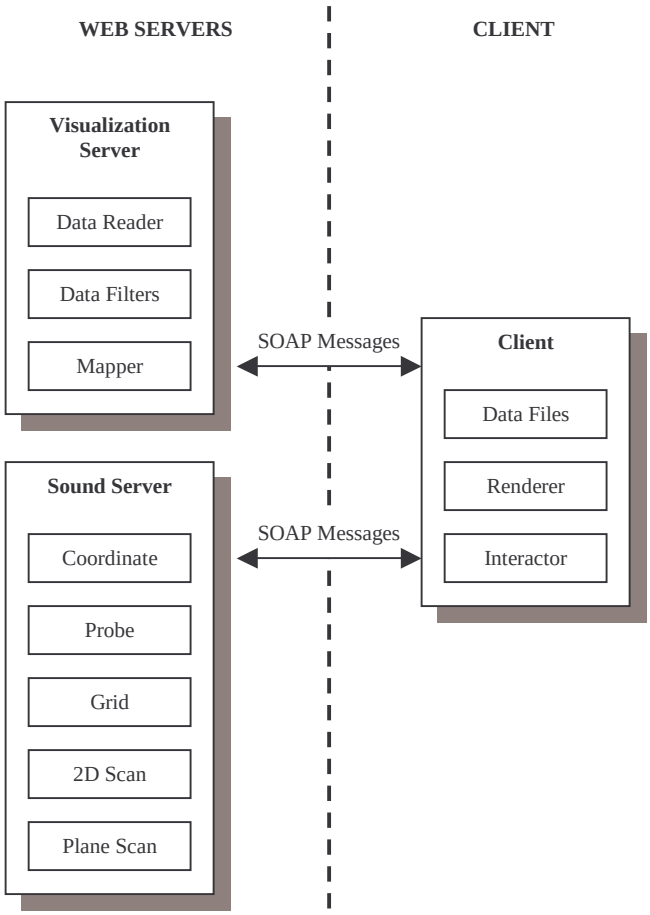

Figure 1 – DSVol Structure.

In the architecture, the visualization server is responsible for processing part of the visualization pipeline. It adopts the reference model proposed by Upson [4] for client-server visualization. This model considers the visualization process as a pipeline where, starting from the data set, a number of steps are followed: load, filter, map, and render (see Figure 2). Adopting this basic pipeline, the visualization server in DSVol is responsible for loading, filtering and mapping of the data set. Rendering and display are performed at the client. The vertical line on Figure 2 indicates the distribution of pipeline tasks between server and client.

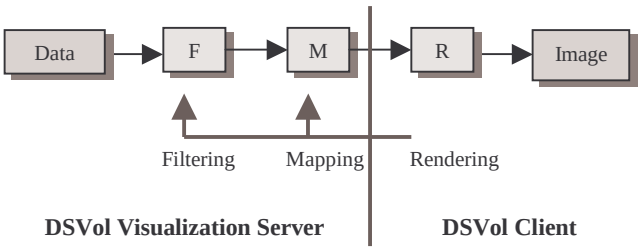

Figure 2 – DSVol pipeline.

The mapping realized on the server side is sent to the DSVol client module, where the transformed data are rendered and presented.

The DSVol sound server is responsible for calculating the sound to be played on the user hardware (DSVol client module), based on information sent by the requiring program (in this case, the DSVol client). This module provides the implementation of methods that calculate the sound properties of different sounds that are necessary for the various sonification tasks associated with the graphical processes of DSVol. The methods of the sound server receive specific parameters depending on the kind of sonification task to be realized. In the current version of the server, it calculates specific frequency values that will be played on the client side, and sends it back to the client. The server is evolving to accomplish more complex sonification mappings. On the client side, the calculated frequency values, as well as volume and timbre, defined by the user through the Java interface, are transformed in MIDI messages and played back. In DSVol, the sonification server and visualization servers are accessed on demand by the different graphical processes available at the client. Those processes are described in the next section.

4 DSVol Graphical Processes

The DSVol client module is composed of five graphical processes: Progressive Display, 2-D Scan, Sound Probe, Grid Display, and Plane Scan. Each graphical process is responsible for allowing the user to interact with the volumetric scene. They are also responsible for one or several distinct sonification tasks.

In the *Progressive Display* the volume is presented slice by slice and the sound of the desired feature is played as each slice is shown (see Figure 3). It is used to frame

sonification tasks that vary with the change in position over a particular axis. During this process, the sonification tasks help users understand the progress or structure of some scalar value or feature in their data as the presentation evolves. It adds information without increasing visual clutter.

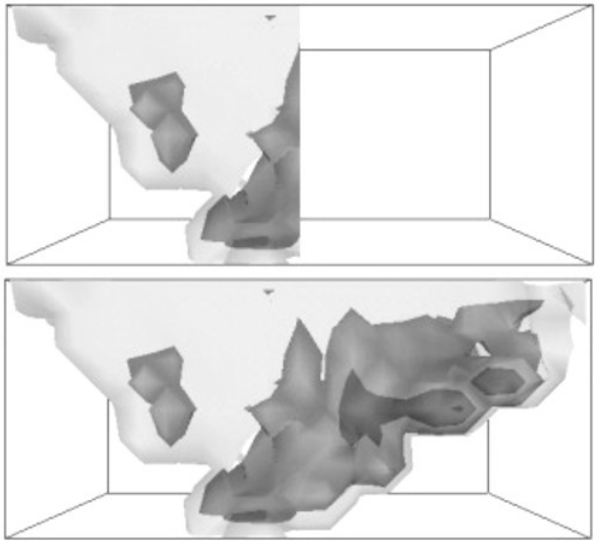

Figure 3 – Progressive Display Process.

The Progressive Display process does not allow user interaction during presentation. To allow user interaction between presentations of neighboring volume slices, another graphical process was implemented: the *Plane Scan*. Here, the user can move a plane forward and backward and listen to features related to the slice as it is presented (see Figure 4). This visual process allows users to explore features of the volume, slice by slice, on any desired axis. Particular features on a slice can be sonified in this process, while its movement is controlled by the user.

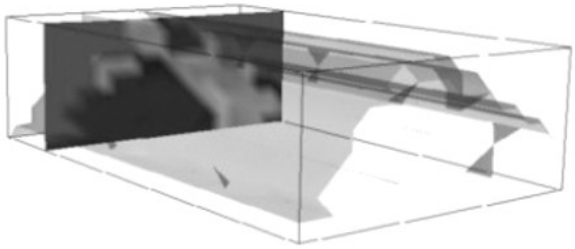

Figure 4 – Plane Scan process.

Figure 5 presents the fourth graphical process implemented by DSVol, the *2-D Scan*. In this process, the user chooses a volume slice using the previous tool, and then scans this slice listening for information about the volume. The sonification tasks of the *2-D Scan* is meant to help users analyze the contents of a specific volume slice, scanning or tracing paths on this slice. It adds an additional dimension to the presentation, which can be compared to, or combined with the dimension already presented by color. The position being sonified is presented by a little square under control via the cursor.

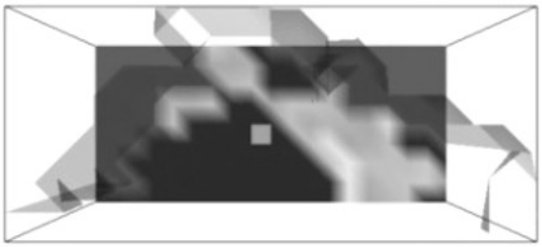

Figure 5 – 2-D Scan process.

The *Sound Probe* graphical process allows the analysis of volumetric features inside the whole volume. In this process, a tetrahedral probe is presented inside the volume (see Figure 6). This probe can be re-dimensioned and moved by the user. It can also be selected as the etire volume. The sound produced during the process represents a feature or value of scalar nature inside the probe, helping the user understand data located inside the probe, which may be hidden by graphical presentation or not represented there.

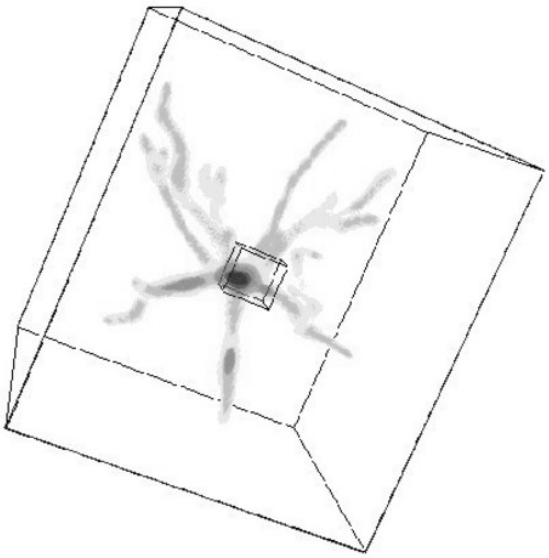

Figure 6 – Sound Probe process.

The *Grid Display* graphical process allows the direct analysis of values or features on the data grid that may be lost during the mapping process, or hidden in the rendered image. This process projects the data mesh onto the screen, and the user can listen to data values pointing the mouse on a desired position of the grid (see Figure 7). Because of the relation between the data on the grid (being sonified) and the values on the display (being visualized) it is possible for the user to make mental associations between the two elements.

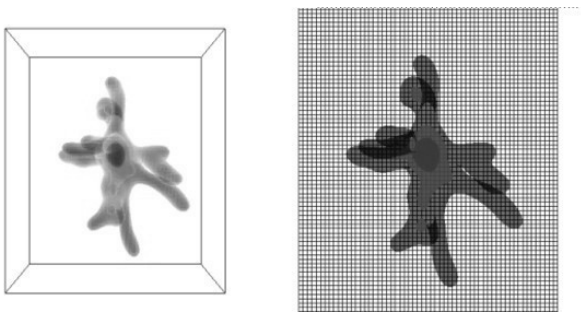

Figure 7 – Grid Display process.

These five processes offer a sonification framework to assist visualization tasks. They were added to the interaction already existing in VTK, and are meant to support access to the sonification functions. Of these interaction functions, two (the 2D scan and the plane scan) were added also to the first version of DSVol.

A Java interface gives access to these interface processes on the Web (see figure 8).

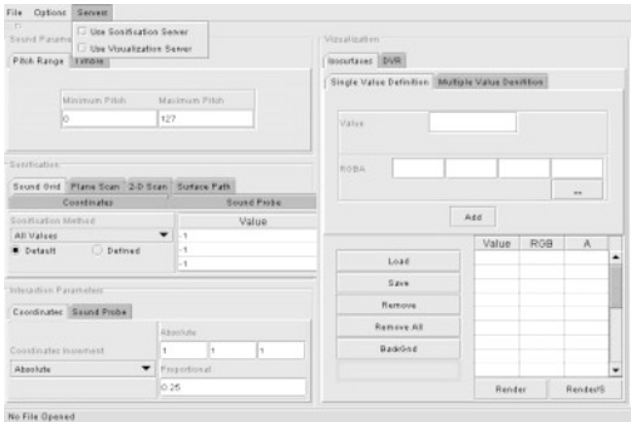

Figure 8 – DSVol Java Interface.

In this framework it was our intention to allow remote access to costly processing, such as visualization mappings (in the visualization server) and sound calculation and processing (in the sonification server). Meanwhile interaction and rendering would continue to be done at the user's end, that is, in the client module of the system. Additionally, Web access to the system is an important feature of the environment.

The main difference between the first version of DSVol system and the current one is the way that elements of the system communicate. The first version used communication between modules via CORBA. In the current version, DSVol modules communicate via SOAP. The next section presents details of the communication between the elements of DSVol implemented by the current version.

5 Communicating on the Web via SOAP

XML is a meta-language that was designed to describe data. It was created as a way to structure, store, and send information. XML is becoming a common tool for data manipulation and data transmission over the Web [6].

Using XML to handle data sets can reduce incompatibility amongst visualization systems, and help cooperation amongst researches, since XML is stored in plain text format, and is independent of hardware, software, and application.

One visualization application that uses XML as a way to structure the data set is MoDL (Molecular Dynamics Language) [8]. MoDL is an XML application that provides visualization of chemical simulation data over the Web, where the simulation data is marked up,

transmitted, converted to a VRML world, and visualized using a VRML plug-in inside a Web browser.

The DICE (Distributed Interactive Computing Environment) is a toolkit that uses XML as a way to structure data [16]. It is designed to allow interactive, runtime visualization of codes running on a high performance platform from the users desktop.

SOAP (Simple Object Access Protocol) uses XML to allow applications to communicate over the Web [7]. The current version of DSVol uses SOAP as a protocol for exchanging information amongst client and servers, as opposed to the first prototype, which employed CORBA.

Web-access to DSVol capabilities, as well as ease of setup, are required features. SOAP improves inter-platform implementation where these two features are fundamental [9].

Using this protocol, the DSVol client connects to the visualization and sonification servers by specifying the server URL, where the WSDL documents are located (see Figure 9). The WSDL is an XML format for describing the methods offered by the servers.

```
…
// Connect to the service
ISoapConnectorPtr Con;
Con.CreateInstance(__uuidof(HttpConnector30));

// Specify the server URL
Con->Property["EndPointURL"] =
 "http://DSVolServer/DSVolSOAP/VisualServer/
  VisualServer.wsdl";

// Connect to the Visual Server
Con->Connect();
…
```

Figure 9 – Connection to the DSVol Visual Server.

The WSDL document, defined for the DSVol visualization server, provides two methods that can be accessed by the client. The first method, *SetFile*, receives a VTK file, as an attachment, and saves it on the server end. The second method, *GetMapper*, receives the iso-surface values to be used as parameters for filtering, followed by graphical mapping. After mapping, the method serializes the result (using and instance of the *vktStructuredPointsWriter* object) and sends it back to the client. On the client side this result is rendered and displayed.

The WSDL document defined for the DSVol sonification server provides several methods to calculate the sound properties for different sonification tasks related with the graphical processes of DSVol, such as *CoordSonification*, *MaxSonification* and *MinSonification*, used on the Progressive Display process; *PopDensSonification*, used on the Sound Probe process; and *GridSonification*, used on the Grid Display process.

To handle SOAP requests on the server side, an ASP listener is specified in the WSDL document, by identifying an ASP file (see Figure 10). The ASP listener provides additional processing, such as parsing and verifying input and security, before processing the contents of a SOAP message.

```
<definitions>
…
  <service name='VisualServer' >
    <port name='VisualServerPortType'
          binding='tns:VisualServerBinding' >
      <soap:address location=
          'http://DSVolServer/DSVolSOAP/
           VisualServer/VisualServer.asp' />
    </port>
  </service>
…
</definitions>
```

Figure 10 – Visual Server WSDL file fragment.

On the client side, the parameters of each method invoked are encoded in a SOAP request message (see Figure 11). This SOAP request message is sent to the server, where it is decoded and processed. The result of the requested operation is encoded to a SOAP response message, wich is sent back to the client.

```
…
// Build the SOAP Message
Serializer->StartEnvelope("","","");
Serializer->StartBody("");
Serializer-> StartElement("GetMapper",
                   "uri:VisualServer,"","m");
Serializer->StartElement("I","","","");
Serializer->WriteString(_bstr_t(I));
Serializer->EndElement();
Serializer->StartElement("SURFACE","","","");
Serializer->WriteString(_bstr_t(SURFACE));
Serializer->EndElement();
Serializer->EndElement();
Serializer->EndBody();
Serializer->EndEnvelope();

// Send the message to the web service
Connector->EndMessage();
…
```

Figure 11 – Creating the SOAP request message.

6 Conclusions

The previous version of DSVol used CORBA to distribute visualization and sonification tasks. CORBA is a vendor-independent open architecture for implementing network based applications, developed by the Object

Management Group (OMG) [17, 18]. CORBA is a good choice for projects that demand reliability and managing of complexity [9].

On the other hand, CORBA is considerably difficult to develop and employ, and sometimes the required costs and effort for code development do not make up for performance benefits. CORBA also can present potential firewall security issues because it requires open network connections to transmit data.

The current version of DSVol substituted CORBA by SOAP. In contrast to the problems presented by CORBA, SOAP provides the benefits of simplicity, ease of learning and ease of implementation. It makes data more easily understandable, and offers a simple solution for making remote procedure calls via HTTP, which solves the firewall security issues [9]. One disadvantage of SOAP is the consumption of bandwidth in transit because of the size of the text-based messages, and their demand on time due to the need for extracting and parsing XML data. This limitation will become less important as processor and network speeds increase.

DSVol II, this second prototype of DSVol presents an alternative solution for visualization on the Web using distributed resources that, due to the use of this current technology and inexpensive software is effective, low cost, and flexible, both in adding visualization and interaction tasks, and in providing pipeline distribution, with potential for solving some of the problems encountered in everyday visualization. Consecutive versions of the system will be kept available at the site www.lcad.icmc.usp.br/~DSVol.

Next steps in the project are the creation of an XML structure for VTK files; the extension of the VTK library to allow writing and reading of this structure; and the extension of the sonification server to perform sound synthesis by software. The goal is to have a complete interaction model to work in a distributed manner over the Internet, which encompasses the aspect of sonification as a support tool.

7 Acknowledgments

We want to acknowledge FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo) for funding this project.

8 References

[1] R. Minghim, V. C. L. Salvador, B. S. Freitas, M. C. F. Oliveira, L. G. Nonato – "Distributed Sound For Volumes – Data Analysis Using Distributed Visualization and Sonification", in *Proceedings of SPIE - Visualization and Data Analysis 2002*, vol. 4665, pp. 379-390, January 2002, San Jose, CA, USA, 2002.

[2] Sun Microsystems, Inc – "Java Native Interface Specification", available at http://java.sun.com/products/jdk/1.2/docs/guide/jni/spec/jniTOC.doc.html (April/16/2002), March 1997.

[3] W. J. Schroeder, K. Martin, W. Lorensen – The Visualization Toolkit – An Object-Oriented Approach to 3D Grraphics, Prentice-Hall, 2nd Edition, 1998.

[4] C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. van Dam – "The Application Visualization System: A Computational Environment for Scientific Visualization", in IEEE Computer Graphics and Applications, vol. 9, no. 4, pp. 30-42, 1989.

[5] F. Arciniegas – C++ XML, New Riders Publishing, First Edition, August, 2001.

[6] W3Schools – XML Tutorial, available at http://www.w3schools.com/xml/default.asp.

[7] K. Scribner, M. C. Stiver – Understanding SOAP – The Authoritative Solution, Sams Publishing, 2000.

[8] B. Arun, V. Chandru, A.D. Ganguly, S. Manohar – "Molecular Dynamics Visualization with XML and VRML", in Proceedings of Computer Graphics International 2000 (CGI'00), Geneva, June, 2000.

[9] J. R. Borck – "Future of Networked Apps", InfoWorld Magazine, July 2001, available at http://www.infoworld.com/articles/tc/xml/01/07/16/010716tcsoap.xml (April, 19th 2002).

[10] C. S. Ang, D. C. Martin, M. D. Doyle – "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web", in Proceedings of IEEE Visualization'94, 1994, pp. 13-20.

[11] J. Ahrens, C. Law, W. Schroeder, K. Martin, M. Papka – "A Parallel Approach for Efficiently Visualizing Extremely Large, Time-Varying Datasets", available at http://public.kitware.com/VTK/pdf/pvtk.pdf, April, 20th 2002.

[12] J. Wood, K. Brodlie, H. Wright – "Visualization Over The World Wide Web And Its Application To Environmental Data", in Proceedings IEEE Visualization'96, October/November, San Francisco, Califórnia, pp. 81-86, 1996.

[13] M. Aeschlimann, P. Dinda, J. Lopez, B. Lowekamp, L. Kallivokas, D. O'Hallaron, "Preliminary Report on the Design of a Framework for Distributed Visualization", in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (Las Vegas, Nevada), invited paper, pages 1833-1839, CSREA Press, June 1999.

[14] K. Brodlie, N. El-Khalili, Y. Li – "Using web-based computer graphics to teach surgery", in Computer & Graphics, n. 24, 2000, pp 157-161.

[15] A. D. Alves, M. C. F. Oliveira, R. Minghim, L. G. Nonato – "Interactive Visualization over the Web", in Proceedings of SIBGRAPI'2000, IEEE Computer Society Press, October/2000, pp. 259-268.

[16] J. J. Hare, J. A. Clarke, C. E. Schmitt – "The Distributed Interactive Computing Environment", Proceedings of 21st Army Science Conference, available at www-lil.univ-littoral.fr/~lefer/Web3Dgv/IEEEWorkshop98/Hare.abstract.html (April, 20th 2002), 1998.

[17] M. Henning, S. Vinoski – Advanced CORBA Programming with C++, Addison-Wesley Professional Computing Series, 1999.

[18] Object Management Group – "*The Common Object Request Broker: Architecture and Specification*", available at http://www.infosys.tuwien.ac.at/Research/Corba/OMG/cover.htm,(April , 20th 2002), 1995.

[19] Zunino, C.; Montrucchio, B.; Sanna, A.; Demartini, C. – "A distributed visualization environment for scientific visualization based on Jini technology"', in IEEE Proceedings of SCCG'2001, April 2001, pp. 95-101.

[20] Liere, R.; Harkes, J.; Leeuw, W. – "A Distributed Blackboard architecture for Interactive Data Visualization", in Proceedings of IEEE Visualization'98, Outubro/1998, North Carolina, USA, pp. 225-231.

[21] Engel, K.; Hastreiter, P.; Tomandl, B.; Eberhardt, K.; Ertl, T. – "Combining Local and Remote Visualization Techniques for Interactive Volume Rendering in Medical Applications", in Proceedings of IEEE Visualization'00, 2000, pp. 449-452.

[22] Frisch, N.; Ertl, T. – "Embedding Visualization Software Into a Simulation Environment", in Proceedings of SCCG 2000, Bratislava, 2000.