Published in 1st International Conference on Distributed Frameworks for Multimedia Applications (DFMA), 176-182, 2005 1
which should be used for any reference to this work

A Collaborative Extension of a Visualization System

Steve Casera Hans-Heinrich Nägeli Peter Kropf
Computer Science Department - University of Neuchâtel - Switzerland
steve.casera@unine.ch hans-heinrich.naegeli@unine.ch peter.kropf@unine.ch

Abstract

This paper presents an extension of the single-user scientific visualization system ZoomIn to a collaborative system. We discuss the principal challenges of remote collaboration in the case of scientific visualization and present the solutions realized for ZoomIn. The main issues identified include the support of slow/fast network connections, portability, ease-of-use, privacy, configuration of the working mode and permissions. We describe why the main concepts of the original ZoomIn are well suited for a collaborative extension and how collaboration is introduced.

In particular, the volume of data for scientific visualization may be very large. Therefore, the transfer of data has to be optimized. We present three different methods of data transfer for collaborative visualization and compare their efficiency with respect to particular application scenarios.

1. Introduction

Collaborative scientific visualization may be defined as the situation where visualization users wish to share their computation or simulation results and their visualizations in order to help each other in discovering scientific phenomena. These users may be located at dispersed geographic sites, using diverse equipments, having access to different data and having distinct knowledge of the scientific domain of the visualized results. Therefore there is a need to develop a collaborative visualization system that allows multiple users at different locations to discuss, create, manipulate, share, analyse and visualize complex datasets over a heterogeneous network. In order to be effective, such a system requires to be ease-to-use, efficient and extensible.

Together with the recent growth of network bandwidth and computing power, a multitude of computer-supported cooperative work (CSCW) systems have emerged [1]. Examples of such systems include shared workspaces, instant messaging services, online games, peer-to-peer file sharing and more specialized software. Visualization for computational science applications, such as biomedical applications, or grid computing [2] are undoubtedly domains where there is a need for collaboration.

CSCW systems, also known as groupware, share some typical advantages and disadvantages. Grudin [3] enumerates eight challenges that need to be met in order to build a usable system: 1) Disparity in work and benefit 2) Critical mass 3) Disruption of social processes 4) Exception handling 5) Unobtrusive accessibility 6) Difficulty of evaluation 7) Failure of intuitivity 8) Adoption processes. Ellis [4] gives pros and cons of such systems.

Collaborative visualization allows users to interact while studying phenomena regardless of the distance between them. This allows the users to share their knowledge with others. Another advantage is the possibility to use remote computational resources, so that users can take advantage of the power of another user's equipments.

The second section describes some existing collaborative visualization systems, explains their main features and introduces basic concepts used in our system. The third section proposes some solutions to the main problems arising when developing collaborative systems for scientific visualization. The fourth section describes in more detail the main features and the conceptual design of the collaborative extension of ZoomIn. In the last section, some possible extensions of our software are proposed.

2. Previous work

Collaborative visualization systems can be divided in two categories: *specialized systems*, i.e. software for visualization only, and *generic collaborative frameworks* extended with visualization capabilities. The systems of the second category are generally less

sophisticated with respect to visualization and require therefore some effort of adaptation to meet the specific needs of scientific visualization.

In the following we describe the main features of some collaborative visualization systems.

COVISE [5] is a modular visualization environment (MVE) allowing a user to run modules locally and remotely. It uses a data manager process on each host to control the flow of data between modules.

COVISA[6] is a collaborative module that extend IRIS Explorer. It allows connecting to a collaborative session and provides a tool to share the map construction processes. In addition, data and control can be shared among separate instances of the system being run by each collaborator.

CSpray [7] is a collaborative visualization system based on the spray metaphor. It is designed for small groups and uses avatars to represent users in the virtual world.

The Shastra architecture [8] provides an application-independent substrate as demonstrated by several scientific and engineering design applications. A specific implementation has been completed for scientific visualization. The design is based on the data-simulation-analysis-visualization (DSAV) loop.

VisAD (Visualization for Algorithm Development) [9] is a Java component library for interactive and collaborative visualization and analysis of numerical data. It uses a general mathematical data model that can be adapted to virtually any numerical data and that supports data sharing among different users.

There are many CSCW frameworks that claim to be suited for application in different fields including scientific visualization [10][11]. Unfortunately many of them have deficiencies [12] at a conceptual, architectural or technological level that make very difficult to embed an implementation of powerful visualization tools into them. Given such drawbacks, we decided to add collaborative features to an existing scientific visualization platform: ZoomIn developed at the University of Neuchâtel.

ZoomIn [13] was developed with the objective to create an easy to use, extensible and customisable system offering an intuitive user interface.

ZoomIn uses a conceptual model that is based on the visualization hierarchy introduced by Haber and McNabb [14].

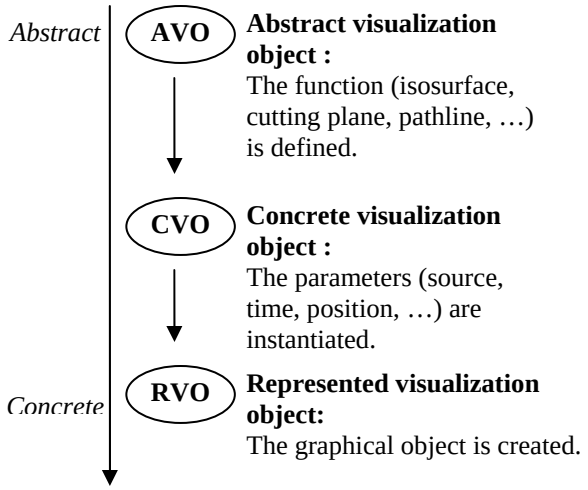

Figure 1. Visualization object hierarchy

This hierarchy is composed of three levels (Fig. 1). The following example illustrates the essence of this concept:

- An AVO (Abstract Visualization Object) is a function that maps a set of data onto a geometric object with certain properties. The cutting plane is an example of such an AVO, the source being a scalar field, the region of interest being a plane, and the colormap being an additional parameter. Other examples of AVO are an isosurface, arrows or probe. Hereafter an AVO is called a tool.
- A CVO (Concrete Visualization Object) is a 3D object that exists only in the conceptual world of the user. The semi-transparent cutting plane of the temperature field in the plane *p* and at time *t* (with the corresponding color and texture) is an example of a CVO. In the mentioned example, the source is the temperature field, the time is *t*, the region of interest is the plane *p*, the first additional parameter is the colormap. A CVO thus encompasses *parameters* including color, transparency, and so on.
- An RVO (Represented Visualization Object) is the approximation of the (ideal) CVO that is manipulated by the graphical system. Whereas in our example the CVO's semi-transparent cutting plane of the temperature field in the plane *p* and at time *t* has a smooth contour, the corresponding RVO is a *graphical object* composed of a set of triangles that is projected onto the screen.

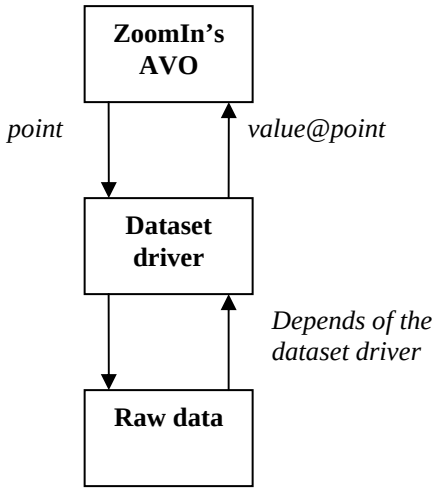

Figure 2. Data access in ZoomIn

Given a CVO, an AVO produces an RVO by accessing some part of the data. In ZoomIn, data access is done using dataset drivers (Fig. 2):

- An AVO accesses the data from a given dataset by requesting the value of the field at a given point.
- The dataset driver is used to access a particular piece of raw data. Thus, given a point coordinate, it will compute the field value at this point, by accessing raw data. Some generic dataset drivers are provided with ZoomIn. If a dataset driver is not available for some particular raw data, developers can implement new dataset drivers.

This conceptual framework has several advantages that will be shown in the next section.

3. Collaborative features

This section gives an overview of how the main collaborative problems that Grudin [2] derives from the challenges presented in the first section are solved in ZoomIn. Note that these problems are related with each other.

- *Slow/fast network support*: A major need in collaborative environment is the efficient exchange of data. This is especially true for collaborative scientific visualization domain because datasets may be very large. Depending on the situation, the data to be sent represent raw data, a visualization result or some intermediate form to be transferred. This can result in a huge amount of data and the access to them must be optimized. We describe these optimizations in section 4.
- *Heterogeneous platform support*: Ideally the software should be portable, and should even be executable from a web browser. ZoomIn uses Java, RMI and Java3D, thus it is likely that it will work on many platforms.
- *Ease-of-use*: A collaborative version of a system has to resemble the single user version. In ZoomIn, creating a shared session and working with it is similar to what happens on the one-user version. Only one supplementary menu has been added to create a session.
- *Privacy*: Privacy in ZoomIn is flexible. Once one creates a shared session, two worlds appear: the private world and the shared world. Everybody has read-access to the shared world; the creator of the session can grant write access to other users depending on the desired working mode. The private world is similar to the shared world except that, by default, other users have no permission to access it. The private world is mainly designed for making tests without polluting or disturbing the shared world, e.g. for exploring an interesting phenomenon in a visualization. Once an interesting object is found, the user can make a copy of it into the shared world so that everybody can see it. But a private world may also be shared to allow other users to see the investigations made by one of the participants.
- *Representation of session users*: In order to collaborate with others, a user needs to know where other users are located in the visualized space and what they are looking at. Avatars (Fig. 3) indicate in ZoomIn where the users are and what they are looking at. Clicking on another user's avatar will transfer his view to one's local view. A virtual pencil allows to indicate interesting positions in the world; a pencil can easily be positioned and saved for further retrieval.

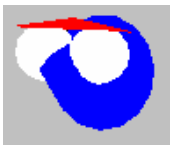

Figure 3. ZoomIn avatar

- *Working mode*: As in groupware applications [15], the working mode may be different depending on the user's type: it may be a teacher that makes a presentation of an interesting dataset to his students or a scientist collaborating with colleagues to find complex phenomena. Clearly a teacher does not want to give write access to the students but two scientists will probably grant it each other. ZoomIn allows for a complete customization of the working mode through permissions. There are two main modes:

- A presentation mode: an instructor handles a presentation to which other participants are remotely attending. But some more flexibility is needed: an instructor may grant a participant the permission to interact with the world, thus allowing him to ask questions or to show interesting phenomena in the world. A mode with more than one instructor can also be easily realized.
- A free mode: everybody can interact with the shared world, e.g. by adding graphical objects.

By setting permissions, the operating mode can be customized and other intermediate modes can be created. The user that created the session has the right to set permissions and to give this right to other users.

- *Flexibility*: Commonly used configurations are predefined and are directly available:
  - Light client. In this configuration, it is assumed that the dataset resides on a server. Clients can connect to the server; which will perform the computations and send the results back to the client. This is useful if one has a powerful computer with limited graphic capability. The server makes all the heavy computations and the clients only have to render the world.
  - Data server: One can start a session and only share a dataset. This will lead to a data server that will only compute data points. Thus the clients will only have to build the graphical object.
  - Remote visualization: With a data server and a client, one has a powerful remote visualization system.

  Note that in these examples, the software will remain the same, so users will keep the same interface.
- *Technical details*: The ZoomIn system relies on a replicated architecture. This help to avoid problems of contradicting and inconsistent data input and manipulation. After a user has made a modification of the workspace, the result is sent to all participants. No other modifications are allowed until all participants will have been notified and will have locally included them. A client-server protocol is applied to efficiently transmit graphical object data. Furthermore, a multicast network protocol is used to broadcast parameters and control messages, as well as to implement the replicated architecture and the users' exchanges (e.g. chat). Functional tests have demonstrated the soundness of this approach.

4. System design

This section explains why the main concepts of ZoomIn are well-suited to a collaborative extension and describes some of the main design aspects of ZoomIn, i.e. the various ways to exchange information with the host.

From the above section, an abstraction that describes at least three characteristics is required:

1) A description of the graphical object, e.g. a set of triangles that composes an isosurface.

2) A description of the object before it has been built, i.e. the parameters that the user has chosen for the object, e.g. an isovalue, a dataset name and the box in which the isosurface has to be computed.

3) A description of the way to access data. Data access should be independent and clearly separated from the construction of the graphical objects.

The AVO and dataset abstraction levels of ZoomIn (see Fig.1 and 2) fully comply with these three points: an RVO is the representation of a graphical object; a CVO encapsulates the parameters of an object before it has been built and the data is accessed through a dataset driver that is clearly separated from the AVO. Moreover the separation of the function achieved by the visualization object (the definition of what it has to compute and how it does it) from the parameters and from the obtained result (a set of properties and a graphical object) satisfies the above requirements.

There are three possibilities for building an RVO from a given CVO:

"Local dataserver" case: The dataset is available on the local host (Fig. 4). The RVO is built directly from the raw data. This is the default situation and the fastest way to build the RVO in a single-user environment. In a collaborative environment however, the dataset quite often is not available on the local host.

"Remote dataserver" case: The dataset is not available on the local host but requested from a remote host (Fig. 5). The CVO is built on the local host, and the data are fetched from one of the remote hosts that own the dataset. The client has only to build the RVO from the CVO and the data. Note that many data transfers may be needed for certain AVO's.

"Remote graphical object" case: The dataset is not available on the local host and the RVO is fetched from a remote host (Fig. 6). In this case, the CVO is transmitted to a remote host that owns the dataset; the RVO is built there and transmitted back. This implies that it is necessary to transmit the RVO through the network as the RVO is the graphical object.

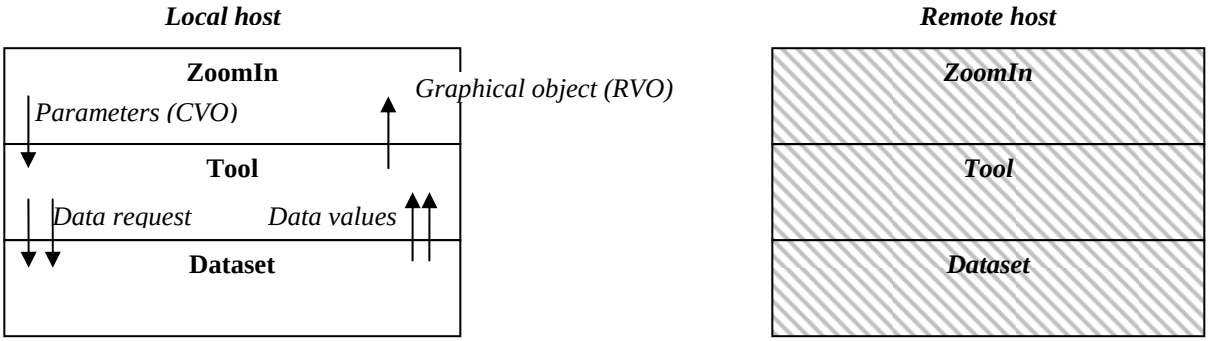

Figure 4. "Local dataserver": Dataset available on the local host

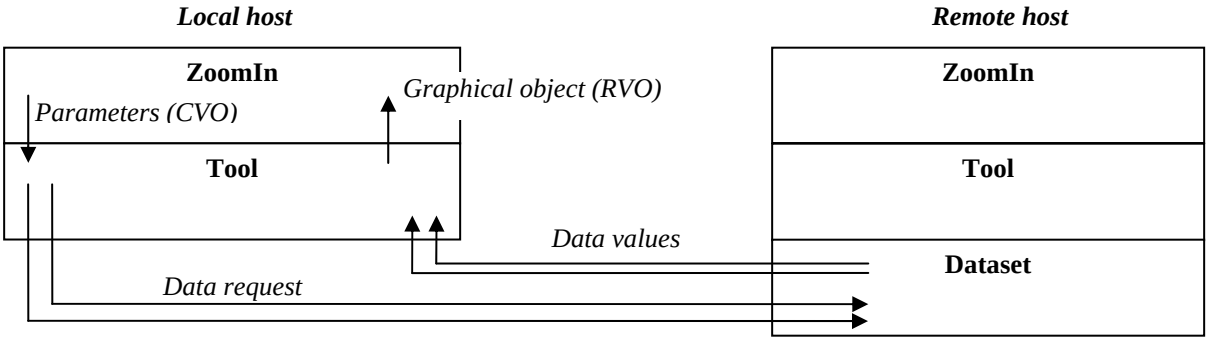

Figure 5. "Remote dataserver": Dataset not available on the local host and computation of the dataset on a remote host

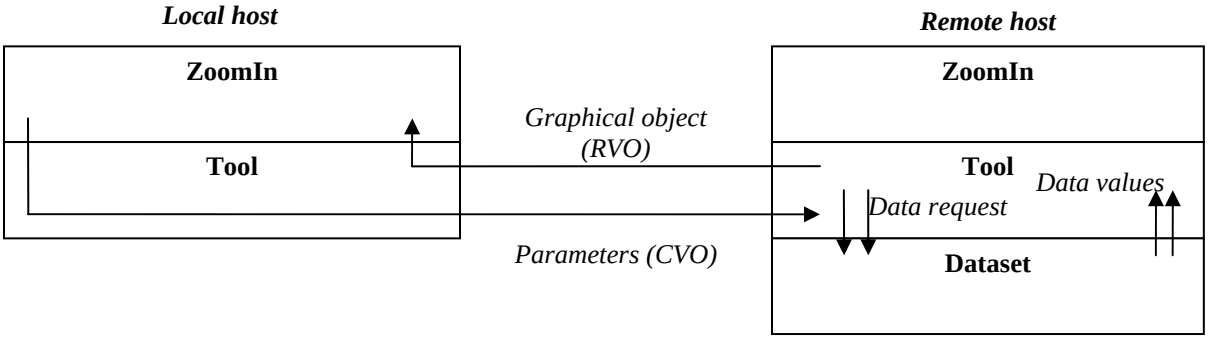

Figure 6. "Remote graphical object": Dataset not available on the local host and computation of the graphical object on a remote host

The main difficulty consists in choosing the most suitable and effective method in a given situation. To do this, one has to optimize the network transmission time and reduce the computational cost. One can compare the three cases discussed above:

1) The "local dataserver" case requires the host to store the entire dataset. If it is not available, it may be remotely transmitted by any method (ftp, CD-Rom, …) before the beginning of a visualization session. Transmission of other information during the session is needed to collaborate, but this only requires little bandwidth and can be easily provided by a modem connection for instance. This method is clearly the best in terms of network usage. Collaboration is, however, limited here to the order of creation of graphical objects, representation of the avatars and communications e.g. "chat".

2) The "remote dataserver" case. If the network latency is high and if many data requests are made (some AVOs, e.g. the streamline, require asking many data item one after the other instead of asking all data at once), this method may become slow. The computational cost depends on the datasets. Depending of the AVO, the amount of data may be huge; this is for instance the case for an isosurface.

3) The “remote graphical object” case requires the building of the RVO on a host. The computation of the RVO may require much time, e.g. in the case of an isosurface and quite often the computational cost is higher than in the “remote dataserver” case. The generated RVO may represent a large amount of data, which has to be transmitted.

The choice of the most appropriate transmission method depends on the situation: one has to take into account the AVO type, the Region Of Interest (in fact, the number of points that are sampled), the available network bandwidth between users and the dataset type. The following examples illustrate two solutions and the most appropriate choices for collaboration:

- A user wants to create a probe, i.e. a representation of a vector field in one point [16]. Clearly the best method is to fetch the information about the point of interest from the remote dataserver. The probe will be computed on the local host containing this data. Only a small amount of data has to be transmitted. If the RVO were transmitted, the graphical data amount would be large as all the triangles that compose the probe will be transmitted.
- A user wants to create a streamline. In order to compute the streamline, one point after another needs to be asked from the dataserver. This is not very efficient as one has to fetch *n* data item one after the other. It is most efficient to compute the streamline on a remote host and then transmit the graphical representation.

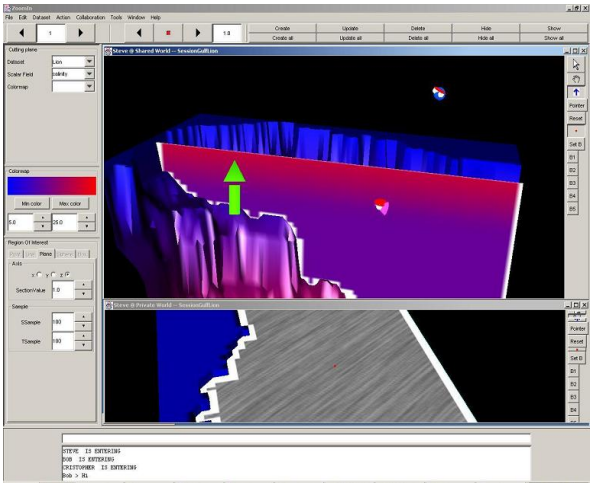

Figure 7. Screenshot of the GUI of ZoomIn

Figure 7 describes a typical ZoomIn window configuration: users set parameters in the left part, the upper part shows the shared world with the avatars and the pointers, the part below reflects the private world and finally the chat system is shown at the bottom.

5. Conclusion

We have described the extension of existing scientific visualization software, ZoomIn, to a collaborative version.

Compared to other systems, the benefits for the users of using ZoomIn are the ease of use of the system as the collaborative part only adds few items to the user interface, the possibility to define flexible working modes, the working with slow network connections, the data transfer optimization, the users representation with avatars and much more.

We have tried to solve the eight challenges of Grudin [3], for example to address the problem of unobtrusive accessibility; we have added groupware features to an already successful application. In order to further evaluate our system we are now collecting feedbacks from users.

Currently, ZoomIn is fully functional and is freely available on our website. The current state of our research, more documentation and the software can be found at http://iiun.unine.ch/paral/zoomin/

References

[1] Brodlie K.W., Duce D.A., Gallop J.R., Walton J.P.R.B. and Wood J.D., Distributed and Collaborative Visualization, Computer Graphics Forum, *The Eurographics Association*, Vol. 23, No. 2, 2004, pp. 223-251

[2] Foster I., The grid: computing without bounds, *Scientific American*, Vol. 288, No. 4, 2003, pp. 62-67

[3] Grudin J. , Eight Challenges for Developers, *Communications of the ACM*, Vol. 37, No. 1, 1994, pp. 93-105

[4] Ellis C.A., Gibbs S.J. and G.L. Rein, Groupware: Some issues and experiences, *Communications of the ACM*, Vol. 34, No. 1, 1991, pp. 38-58

[5] Wierse A., Lang U. and Rühle R., Architectures of Distributed Visualization Systems and their Enhancements, *4th Eurographics Workshop on Visualization in Scientific Computing*, Abingdon, U.K., 1993.

[6] Wood J. D., Wright H. and K. W. Brodlie, Collaborative Visualization, *Proceedings of Visualization ’97, IEEE Computer Society Press*, 1997, pp. 253–259

[7] Pang A. and Wittenbrink C.M., Collaborative 3D Visualization with CSpray *Computer Graphics*, Vol. 17, No. 2, 1997, pp. 32-41

[8] Bajaj C. and Cutchin S., Web based collaborative visualization of distributed and parallel simulation, *Proceedings of IEEE Parallel Visualization and Graphics Symposium* 1999, pp. 47-54

[9] Hibbard W., VisAD: Connecting people to computations and people to people, *Computer Graphics* 32, No. 3, 1998, pp. 10-12, http://www.ssec.wisc.edu/~billh/visad.html

[10] Chabert A., Grossman E., Jackson L. S., Pietrowiz S. and Seguin C., Java object-sharing in Habanero, *Communications of the ACM*, Vol. 41, No. 6, June 1998, pp. 69-76 http://www.isrl.uiuc.edu/isaac/Habanero/

[11] Roseman, M. and Greenberg S., Building Groupware with GroupKit, M. Harrison (Ed.) Tcl/Tk Tools, O'Reilly Press. , 2003, pp. 535-564, http://www.groupkit.org/

[12] Lopez P. and Skarmeta A, ANTS Framework for cooperative work environments, *IEEE Computer*, (ISSN 0018-9162), March 2003, pp. 56-62

[13] Sanglard H., Toward an easy-to-learn and extensible platform for scientific visualization, *PhD thesis*, University of Neuchâtel, Switzerland, 2001

[14] Haber R.B. and McHabb A., Visualization idioms: A conceptual model for scientific visualization systems, *Visualization in scientific computing, IEEE Computer Society Press* 1990, pp. 74-93

[15] Brinck T., Groupware Design issue, 2002, http://www.usabilityfirst.com/groupware/design-issues.txl

[16] De Leeuw W. and Van Wijk J., A probe for local flow field visualization, *Proceedings of Visualization '93, IEEE Computer Society Press*, 1993, pp. 39-45