Distributed and Collaborative Visualization

K.W. Brodlie†, D.A. Duce‡, J.R. Gallop§, J.P.R.B. Walton¶and J.D. Wood†

†School of Computing, University of Leeds, Leeds LS2 9JT, UK
‡Department of Computing, Oxford Brookes University, Oxford OX33 1HX, UK
§BITD, Rutherford Appleton Laboratory, Chilton, Didcot OX11 0QX, UK
¶The Numerical Algorithms Group Ltd, Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, UK

Abstract

Visualization is widely used in science, medicine and engineering. It can convey insight into phenomena that are well-understood, or display new data in order to uncover novel patterns of meaning. Visualization is a powerful tool in presentations (lectures, seminars, papers etc) and in discussions between colleagues. As such, it is an essentially collaborative activity. In this area, there is also a growth in the use of video conferencing to facilitate meetings between participants in geographically separate locations. This includes both specialized facilities (video conference rooms including Access Grid) and desktop video conferencing using the Internet and multicast communications.

Distributed visualization addresses a number of resource allocation problems, including the location of processing close to data for the minimisation of data traffic. The advent of the Grid Computing paradigm and the link to Web Services provides fresh challenges and opportunities for distributed visualization - including the close coupling of simulations and visualizations in a steering environment.

Distributed collaborative visualization aims to enhance the video conferencing environment (usually on the desktop) with access to visualization facilities. At the most basic level, pre-generated visualizations may be shared through a shared whiteboard tool. Richer approaches enable users to share control of the visualization method and its parameters. In one approach, a single visualization application is shared amongst a group of users; in another approach, the visualization dataflow paradigm is extended in order to allow sharing of visualization data between collaborators. Component middleware provides a framework for describing and assembling distributed collaborative visualization applications.

The AccessGrid allows group-group collaboration, rather than just person-person, and generally offers a rich environment for collaboration - we look at ways of integrating current visualization systems into this new type of environment.

XML has made a significant impact in many areas of computing, from e-business to mathematics. It is being increasingly used as the middle tier of client-server interfaces where its power and flexibility makes it ideal for middleware (for example, SOAP and related Web Services developments in W3C). Current developments in Grid middleware are based on an enhancement to Web Services (the Open Grid Services Architecture - OGSA).

This STAR reviews the state of the art in these areas, draws out common threads in these diverse approaches and looks at strengths, weaknesses and opportunities for further development in this field.

1. Motivation

Rogowitz[92] has written "Visualization is the process of mapping numerical values into perceptual dimensions". The use of visual imagery to convey scientific insight and truth is not a new phenomenon. Descartes (quoted by Collins[26]) wrote "imagination or visualization, and in particular the use of diagrams, has a crucial role to play in scientific investigation". More recently interest in visualization was focused by the NSF Panel Report on Visualization in Scientific Computing[75]. Their definition of visualization is interesting:

> "Visualization is a method of computing. It transforms the symbolic into the geometric, enabling

researchers to *observe* their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. ... Richard Hamming observed many years ago that 'the purpose of [scientific] computing is insight, not numbers'. ... The goal of visualization is to leverage existing scientific methods by providing new scientific insight through visual methods."

The report highlights the need for scientists to learn to visually communicate with one another. "Much of modern science cannot be expressed in print. DNA sequences, molecular models, medical imaging scans, brain maps, simulated flight through a terrain, simulations of fluid flow, and so on, all need to be communicated visually." Visualization is a medium of communication.

Much of modern science and engineering involves more than one person. Much (one is tempted to say the overwhelming majority) of design, research and development is not the work of one individual in isolation. It is the work of small groups of people, to large teams of people, each with their characteristic skills and expertise, making their contribution to the overall endeavour.

During the 1980s computer networking became widespread in many organisations, and led to the new discipline of Computer Supported Cooperative Working (CSCW) which gathers together researchers interested in how people work together, and how computers and related technologies affect the behaviour of groups of people. CSCW systems started to emerge. Such systems aim to provide support for group working. CSCW systems typically provide audio and video communication channels between participants in a cooperative session with the addition of groupware tools such as shared text editors, shared whiteboard, shared drawing tools, etc.[51] Given the significance of visualization as a medium of communication in a wide range of contexts, the question naturally arises, how is visualization used within group working and how can this be supported in the CSCW system?

The use of visualization in collaborative working might involve a group of people sitting around a meeting table discussing hardcopy output, or viewing a video. It might involve a group of people clustered around a workstation, with one person in the 'driving seat', discussing a visualization, perhaps making suggestions as to how the visualization could be changed in order to draw out other features in the data, for example by changing a colour map or using a different technique to present the data. Participants might take it in turns to 'drive' the visualization system, each working with their own particular data sets. It might involve exchanging visualizations by email, followed by email discussion, or dissemination to a wider community through the World Wide Web. It might also involve the cooperative use of a visualization system as a tool within a CSCW session.

The AccessGrid[2] is providing a rich environment for large scale distributed meetings. remote lectures and collaborative working across an increasing range of institutions. AccessGrid facilities include large format projection displays, high quality video and audio channels, and the possibility for group interaction with applications. A commercial version of the AcessGrid called the inSORS Grid has recently appeared on the market from inSORS Integrated Communications [62].

Is visualization any different to other media that may be used in cooperative working? In some senses the answer to this question is no. There are many issues that visualization has in common with other media, for example, text. Control of a visualization system raises similar issues to shared control of a text editor. Who has control? How do participants know who has control? How is control passed between participants? There are some senses in which visualization is different to other media. Visualizations are typically generated by a pipeline or network of processing steps. This raises the possibility of sharing data between participants at different points in the processing, which may lead to a useful tradeoff between data volume and local processing capability.

Bergeron[11] in his introduction to a panel session at Visualization '93 argued that the goals of visualization can be divided into three categories *descriptive visualization*, *analytical visualization* and *exploratory visualization*. Descriptive visualization is used when the phenomenon represented in the data is known, but the user needs to present a clear visual verification of this phenomenon (usually to others). Analytical visualization (directed search) is the process we follow when we know what we are looking for in the data; visualization helps to determine whether it is there. Exploratory visualization (undirected search) is necessary when we do not know what we are looking for; visualization may help us to understand the nature of the data by demonstrating patterns in that data.

Comparison between data from different sources is often a fundamental ingredient of collaboration. In the geosciences, for example, it is now realized that much insight is to be gained by sharing data, comparing different kinds of data gathered from different instruments, for example sea height measurements, surface temperature, ocean depth, whereas in the past researchers would concentrate on data from a single instrument which they would almost jealously guard. A researcher's scientific capital was in both the data and the methods used to analyse it. Nowadays the trend is towards sharing and comparison. There seems to be relatively little attention paid to the use of visualization to enable comparison, see for example Pagendarm and Post[82] and more recently Zhou, Chen and Webster[142], but this is nevertheless a

topic to which attention needs to be paid in systems aiming to support cooperative working.

In the late 1980s, Modular Visualization Environments (MVEs) started to appear, the earliest examples being apE and the first version of AVS. MVEs provide a set of building blocks which perform functions such as reading data, generation of visualizations such as contouring and rendering. MVEs typically provide a visual editor with which to construct applications, by linking together a set of building blocks, and it is perhaps the power of this visualization programming metaphor that has made MVEs so popular today. The modular building blocks may be (but are not necessarily) implemented as separate processes. When this is done, this provides a natural way in which such systems may become distributed systems. Examples of systems of this kind include AVS[72], Khoros[141], IBM Data Explorer[1] (now OpenDX[80]), and IRIS Explorer[44, 131].

In the early 1990s, the advent of the World Wide Web led to another approach to visualization applications, a client-server approach in which the visualization required is defined through the client, computed, and returned to the client for presentation. The concept of applets provides a mechanism by which part of the visualization computation may be down-loaded to the client and executed client-side. Such approaches are termed *web-based visualization*.

As well as involving more than one person, much of modern science also involves more than one machine. Grid computing is currently attracting much attention and funding. The essence of Grid computing is "the large scale integration of computer systems (via high speed networks) to provide on-demand access to data-crunching capabilities and functions not available to an individual or group of machines[42]". Shalf and Bethel[98] write "the promise of Grid computing, paricularly Grid-enabled visualization, is a transparent, interconnected fabric to link data sources, computing (visualization) resources, and users into widely distributed virtual organizations". The challenges they identify are familiar: how to support distributed heterogeneous components (as they point out, visualization system users want to use the best tool for the job, regardless of source), dynamic partitioning of visualization components between resources, and algorithms that maintain interactive performance in the presence of latency. The Grid infrastructure is evolving to a service-based architecture in which functional capabilities of services are represented by interfaces which can be discovered along with semantic descriptions. Approaches based on the Grid architecture are termed *Grid-based visualization*.

The aim of *distributed collaborative visualization* is to harness the processing power of many humans and many machines.

It is useful to distinguish at this stage between three terms.

1. *Distributed visualization*. This involves collaboration at the system level. It is interesting that the current batch of MVEs have all been designed with this aim: it is possible to place modules on different computers. Of course this is most useful when it is a computationally intense visualization task, when some modules may usefully be located on a supercomputer, others locally on a workstation. One can also see simple web-based visualization in this class. Although several computers may be involved in the computation, such a distributed visualization system is still a single-user system. Working in a distributed environment does not by itself imply working in cooperation with other users. It is useful to distinguish *parallel* and *distributed* visualization. Parallel visualization involves the use of parallel processing resources to execute a visualization algorithm. The term parallel processing is normally used to describe the situation in which more than one processor among a group of processors is active at any one instant in executing the algorithm. The boundary between distributed and parallel processing can be fuzzy, but we view the difference in terms of the granularity of the task being performed and the inherent notion that more than one processor is necesarily active at any one time in parallel execution.
2. *Collaborative visualization*. Visualization is often a cooperative activity; several people may work together to interpret a visualization. This is collaboration at the human level. It is interesting that the current MVEs did *not* have this as a design requirement, and until recently were all single-user systems. Similarly web-based visualization systems have been single-user. Cooperation is achieved by humans clustering around a single workstation, around a Responsive Workbench device or in a CAVE, for discussion, or through some means outside the visualization system (for example, sending visualization output to a collaborator for comment).
3. *Distributed collaborative visualization*. This brings the two concepts together, allowing collaboration at both the system and human level. We shall be able in this STAR to give examples to show that both MVEs and web-based visualization can be extended to this class. A distributed collaborative visualization toolset should allow its users, geographically distributed, not only to run remote resources, but to share images, and possibly also to interact and cooperate, across a network, in the intermediate steps which lead to the creation of the final output.

Section 2 of this paper presents a structure for the field of distributed and collaborative visualization and introduces a 3-layer model. This model is explored further in Section 3. Section 4 describes a framework in which systems can be compared and contrasted. Section 5 considers data sources and presentation environments, then section 6 discusses a range of visualization systems and frameworks for collaboration. Section 7 describes some current research projects and directions and Section 8 draws some conclusions.

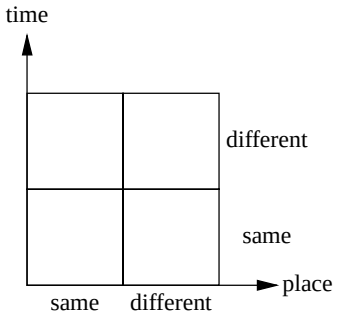

Figure 1: *Applegate's place-time matrix.*

2. A Structure for the Field

2.1. People view - types of CSCW

CSCW involves both people and machines. In the CSCW literature, Applegate's place-time matrix is a widely cited classification scheme for cooperative working[7]. This scheme, shown in Figure 1, focuses on the people involved in CSCW and the way in which the types of involvement may be classified.

Two dimensions are used: time and place. Members of a CSCW group may be located at the same place or a different place, and be present in a CSCW session at the same time or at different times. There is thus the notion that a session may extend in time, and not all participants need be present at the same time.

The same time, same place box corresponds to all members of the group being present in the same place at the same time. Examples of systems to support such forms of working include conference rooms equipped with individual workstations to support decision making processes. In visualization a group of users clustered around a single workstation, around a Responsive Workbench or in a CAVE, would fall into this box.

Collaboration that involves exchange of letters, faxes, emails, between members of a group, falls into the different time, different place box. There is a notion of a group session, the group working on a common problem over an extended period of time. There is a shared history of working contained in the trace of letters, faxes, emails, etc. exchanged between group members. The World Wide Web typically falls into this box, though developments such as cooperative web browsing[63, 71, 87, 96] are extending the web to other forms of working.

Video conferencing falls into the different place, same time box. Members of the group are co-present in time, but at different physical locations. Video conferencing may involve a specially equipped video conferencing suite, with multiple cameras etc. at one extreme, or may be based on a suitably equipped workstation per participant. The latter approach can be characterised as collaboration as an extension of the normal working environment. This notion is taken further in the work of Fuch's group at UNC Chapel Hill[45]. which is investigating the use of cameras and sophisticated display technology to assign a region of each office to collaboration, so that one's collaborators are brought into the office in an even more direct way than through workstation-based video conferencing. However, collaborative working is not just video conferencing. There is a need to share information other than through audio and video channels, and a need to share applications used in the creation, analysis and presentation of information. "Groupware" falls into this category. Much work in distributed cooperative visualization aims to support this type of collaboration.

In any one collaboration, it is likely that several different types of collaboration will be used, for example, formal face-to-face meetings, coupled with different time - different place styles of working between meetings, coupled with informal same time, different place sessions. This raises issues about seamless transitions between different types of working, and the organization of group memory so that it is equally accessible from different types of meeting. Xerox PARC, for example, have worked extensively in this area [14].

2.2. Model

In this section we present a layered model of distributed and collaborative visualization that encompasses the major approaches found in current practice. "Ordinary", i.e. non-collaborative, non-distributed visualization is a special case of distributed and collaborative visualization.

The model is described in terms of three layers:

- the *conceptual layer* which describes the visualization to be performed, independently of the visualization software with which it is to be realized.
- the *logical layer* in which the visualization is expressed as a particular configuration of software entities, but independently of the physical resources with which the configuration will be realized.
- the *physical layer* in which software entities are associated with physical resources.

The conceptual layer abstracts away the details of visualization software and physical resources. It captures the intent of the designers of the visualization. A description in the conceptual layer will capture the nature of the data sources, the visualization itself and the control and viewing environment. Typical data sources would be data repositories or simulation enabled for computational steering. A visualization might be to display an isosurface of a particular field in a data set from one source and an isosurface of a field from a second source, overlaid with coastline outlines. The control and viewing environment might be a remote teaching environment in which the lecturer alone can control the data sources and visualization parameters and students can view the results without control of the view. The conceptual layer thus captures the

visualization design independent of a realization. The process by which a conceptual visualization design is created, might itself involve collaboration.

The logical layer introduces the software entities in which the conceptual description is to be realized. It is convenient to see this in two parts, the *logical visualization design* and the *core software*. In the traditional library context, the logical visualization design is the user program and the core software is the library of subprograms. In a Modular Visualization Environment (MVE) context the logical visualization design is a description of the composition of modules into a network and the core software is the set of modules provided by the MVE. The logical layer thus involves a binding of the conceptual visualization design to a particular software architecture. Examples of bindings to particular software architectures are described later in this paper (see Section 3.2).

The relationship between the conceptual layer and the logical layer also involves refinement of the human user interface and a binding of these in the chosen software architecture and core software.

The logical layer also includes constraints on the resources required by the logical visualization design. A computation might, for example, require a processor with particular characteristics. Constraints on the resources required to realise the chosen human user interface may also be included.

The physical layer is a binding of the logical visualization design and core software to particular physical resources. For example, a visualization realised using a parallel visualization library might be bound to a particular parallel processing machine; a visualization realized using an MVE might include modules bound to processing resources remote from the main controlloing executive. Interface devices are also included in the binding, for example, a particular graphical workbench or devices linked to an AccessGrid. The binding also includes binding of links between entities to particular communication patterns, for example, shared memory, web service and Grid FTP.

This model thus treats the realization of a visualization as a refinement process in which the abstract conceptual design is refined into a logical design using particular choices of core software and then into a physical design in which physical resources are associated with the core software and other software entities.

Figure 2 illustrates the twin concepts of logical and physical environments for a common conceptual scenario. The logical layer has been reduced to a description in which there are two visualization sessions in progress, each associated with an individual researcher. The sessions are shown as dataflow networks, but the concept is wider than the MVE-type systems—as explained earlier, the sessions might be user programs making library calls, or RMI calls; or they might be command-driven packages. Collaboration is shown

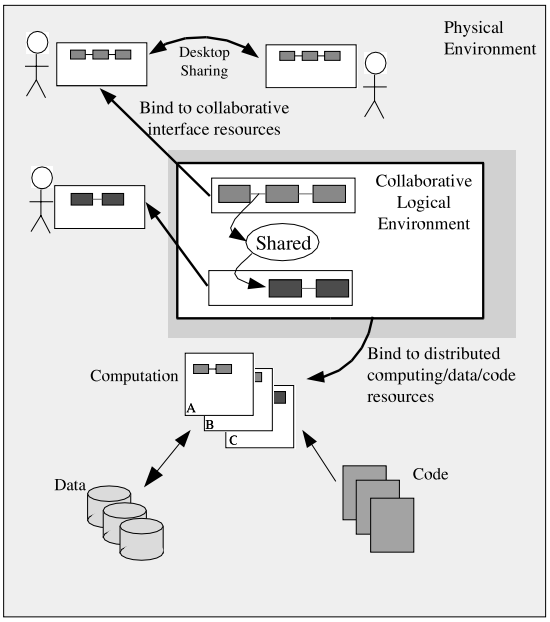

Figure 2: *Example of the logical and physical layers in the model.*

in the logical description as a shared area into which data may be placed by one session, and retrieved by another. This again is meant as a general concept: it covers the interlinked network approach of COVISA/MANICORAL; the shared object approach of CSpray; and the shared parameter approach of COVISE. Indeed it covers both synchronous and asynchronous collaboration—in the synchronous collaboration we would see the shared area as transient, and in the asynchronous case, as persistent.

The physical layer describes the real-world realisation of the logical layer. Here we see that the two logical sessions have been constructed by researchers, through an interface. (These are the people on the left of the figure.) Indeed this is a continual process which exists throughout a session, whereby controls are operated, commands are given and views are displayed. A researcher can place material in the shared area, for the other to retrieve.

An alternative, rather simple, form of collaboration is also shown in the figure. This is the desktop sharing concept, sometimes known as Remote Frame Buffer (RFB), whereby the interface of one researcher is distributed to another researcher (see Section 6.2). If permission is given, the other researcher can input command and control operations through the RFB. This person is shown top right in the figure.

The figure also shows the binding of the logical description to physical resources—as well as the binding of the interface, we have a binding of the data to a file descriptor

(either local or remote); a binding of software components to a file descriptor (again local or remote); and a finally a binding of the execution of the components to a machine descriptor. (At present we do not distinguish the display of the visualization from the interface; this would be possible.)

This concept allows us to describe within a single framework many different approaches to visualization, and we expand on these in the next section.

3. Exploring the 3-Layer Model

The 3-layer model introduced in Section 2.2 distinguishes between the overall description or intent of the visualization application (conceptual layer), the software entities (logical layer) and allocation of physical resources (physical resources). In this section, we examine all the layers in more detail.

3.1. Conceptual layer

To describe the possibilities in this layer, we begin by outlining a possible collaboration, increasing the sophistication as it develops. We then go on to define a set of possibilities.

3.1.1. A possible collaboration

A user begins by using a visualization package to analyse some data on their desktop system. For our purpose here, we do not need to say which visualization algorithm or which package.

The user then decides to make use of a large dataset held in a remote managed archive. The archive hosts a simple visualization package which is accessed via a web page, so the user visualizes the data remotely, receiving images back to the web browser. This helps decide which subset of the data is of most interest.

The user now needs a visual comparison between the local data and the remote archive, so downloads a data subset and plots both it and the local data on the same picture. More data is needed from the remote archive, so a new download each time is cumbersome to use. It so happens that the visualization package has a file import mechanism, which allows the user to specify a web URL and the subset to be extracted. Reliance can be placed on the local web client to cache the remote data for repeated use.

The visual comparisons show up some discrepancies and help is needed from a colleague on another continent, who has the necessary further skills. So we have user U_1 who uses visualization package V_1 to analyse local data D_1 and remote data D_R. The resulting image is sent to the new colleague U_2. To try and understand the problem further, U_2 examines the data with a visualization package V_2 with which they are more familiar. U_1 makes D_1 available via the web and tells U_2 how to access it and D_R.

They then agree that user U_2 starts using package V_1. This is sufficiently flexible that U_1 is able to provide a partially computed result from V_1 and supplies it to U_2, who sets up V_1 to read, compute and display it. Ideally, since U_2 is not so familiar with V_1, U_1 assists by passing across the necessary scripts to implement that part of the visualization design. U_2 manipulates a control parameter (for instance, an isosurface threshold) and has an idea about what the original discrepancy might be due to. U_1 agrees to take the opportunity of controlling the threshold parameter from U_2 and manipulates it too. U_2 offers access to further data D_2. They then agree that they should share the insight they have gained and arrange to call U_3 and U_4. They plan to show these other colleagues how they reach their conclusion by showing what they did step by step.

Naturally U_1 and U_2 also use tools to handle audio and video information between themselves and later on with U_3 and U_4 also.

This sequence of events is not the only one that could have been chosen. What is notable is that the two participants had a number of choices available to them and their use of collaborative software was able to take different forms, depending on what was required at any given time. In describing it, we made assumptions about what an actual visualization package is capable of doing.

3.1.2. Conceptual view of service

Visualization may be viewed as a service, a relationship or contract between a service provider and service user. Based on a scheme of Brodlie[17], we consider the different players involved in a service view; they take different roles and may be individuals or organisations. The original scheme was for web-based, single user visualization, but we extend the scope of the idea here.

3.1.2.1. The user This player—the user—may be a specialist scientist or engineer or may be a member of the public. The range of skill and experience can be quite varied, but in all cases we can assume that they are familiar with at least a web browser. In certain cases, such as the experienced researcher, they may also be familiar with an existing visualization core system and be prepared to tailor its use for themselves or others.

The computing power available to the user is diverse. Desk top PCs are nowadays powerful enough to run a web browser with an VRML or X3D plugin and a Java interpreter. In most cases the desktop system is also capable of running most visualization packages and higher performing versions are used to process larger amounts of data or execute more complex algorithms. A user out of the office or laboratory may be using a PDA on which the software and display is extremely limited. However such devices can use web browsers, in some cases Java run time systems too, and their portability is sufficiently attractive to promote their use.

The method and speed of connecting the user's local system or device may also vary. The user may be in an institution that is well connected or one that is permanently connected, but on a lesser bandwidth, or the user may rely at times on a modem or a wireless connection.

Collaborative visualization implies two or more users. If no more than two, they have the opportunity to share information with each other in some detail. With a large number of users, they are likely to take up a limited set of distinct roles, such as lecturer and audience or peer to peer. Large participation is also liable to cause bottlenecks and methods of scaling up have to be considered.

Participants may be clustered at particular institutions and the growth of this requirement has led to the establishing and use of the Access Grid, which is discussed in Sections 1 and 5.2.

Collaborating participants may also bring a variety of skills and resources. One may be responsible for hosting an application whereas others may make use of a web browser to link to the application. Where participation is on a more equal basis, anyone may run an application and offer results to others.

3.1.2.2. The visualization service provider and designer This player is responsible for the visualization service. It may be accessed via conventional Internet client/server techniques or via Java RMI or via a web page or, increasingly likely, via Web or Grid services. Three distinct levels of service can be described.

Full service in which the visualization result is entirely created by the service provider and is returned to the user as an image or a picture description (2D) or a 3D scene. Some dataset holders provide basic visualization services so that users can browse visually before deciding what to download. In a collaboration, this is a centralized approach.

Software delivery in which the software to create the visualization result is downloaded to the user to be executed by the local processor. This method, which could be achieved by a Java applet, allows the users to work locally without having the responsibility of installing the software.

Local operation in which the visualization software is assumed to be already resident with the user and no service is required from an external provider. In a collaboration, this would imply each participant running largely independently, but exchanging results when useful.

3.1.2.3. The data provider This player supplies the data. Here too three distinct possibilities can be described.

Specialist data agency. Many organisations are responsible for dataset management which includes collecting, publishing, portal provision, metadata management and long term curation. This includes NASA and Unidata in the USA, the ESA and the EU in Europe and—at a national level—the Atomic Energy Authority (UKAEA) and Natural Environmental Research Council (NERC) in the UK. Data extraction is likely to need to be more sophisticated in the long term, for instance: "return me the data for the 24 hour periods in which the temperature was above t degrees C for more than 4 hours at any place in region R". The dataset may be sufficiently large that the only practical location for performing the extraction is (in networking terms) close to the data. Having reduced the data, the subsequent visualization processes could be the responsibility of the user.

The user. The user and data provider may be the same person. This would typically be the case in the traditional use of visualization by a scientist, analysing their own data. In a collaboration, one of the users may be the data provider—or indeed several of the users may be providing data.

The Visualization Service Provider. Here the roles of data and visualization service are combined—for instance when a data provider sees visualization as an essential means of interpreting the data.

We also need to consider the origin of the data (the data source).

Archive. The data may be managed in a long term archive. Collaborating participants may all be interested in the same data archive and exchange information about subsets of common interest.

Computation. The data may be produced by a computation. This may still be running when a user is analysing the data and this offers the possibility of computational steering. It is potentially useful for a collaborating team to be able to access the same computation while it is still executing.

Experiment. The data may originate from observations, taken from remote experimental apparatus. In some cases, it is possible for the user to supply control information, which is usually required in advance for efficient use of the apparatus.

3.2. Logical layer

In this layer, we introduce some common software architectures for distributed and collaborative visualization. We first consider a service-oriented architectural view and then look at more general architectures used by Modular Visualization Environments (MVEs).

3.2.1. Service model for distributed and collaborative visualization

3.2.1.1. Motivation The World Wide Web grew from its initial focus to have a much wider impact as a distributed computing environment. One of the trends in current web development is the Web Services Architecture and its extension to Grid computing, the Open Grid Services Architecture

(OGSA). A Web Service is roughly a software component that can be *described* using a service description language, *published* to a registry of services, *discovered* through standard mechanisms, *invoked* through a declared API usually over a network, and *composed* with other services. A service can also be thought of as a contract between a provider (a server) and a user (a client).

The earliest applications of web technology in visualization were not service based, but were simply for descriptive visualization (in Bergeron's categorisation[11]) where a prepared visualization of scientific data was placed as an image, or perhaps a VRML model (and in the future will be a X3D model) within a web page. It is still often used in this way of course, providing a very useful means of publishing results. Most visualization systems now allow VRML as an output option—see Walton[132] for example.

However it was realised that it is also possible to carry out analytical or exploratory visualization, where the investigative process itself is carried out on the Web environment. Use of web technology allows distributed visualization for a single user, but it also provides a means of asynchronous collaborative working, where the participants use the system in turn. Furthermore with no additional software other than web tools, participants can simultaneously access a web page containing original data or partial results.

In a distributed collaborative visualization session (same time, different place), we need to identify which components are shared.

Shared. In a collaborative application, some components exist just once and perform their tasks on behalf of all participants. The resulting data is distributed to all participants.

Individual. Other components are under the control of each participant which allows individuals to inspect results in the way most suited to their experience and environment.

In the client/server model, the shared components can be associated with the server and there are multiple clients, in principle one for each participant. In practice the shared components are initiated by one of the participants.

Current research is concerned with identifying appropriate Web Services or OGSA-based Services and components for distributed and collaborative visualization.

A visualization service provider is faced with a range of possible ways in which to provide a service. For example, the service might be provided by software running on a server belonging to the service provider, delivering results to clients. An alternative is to provide an application to the client that the client then runs. In general this might be a distributed application where some of the resources used might belong to the service provider.

A key issue is the location of the visualization processing which, at this point, we need to define more closely:

- It is convenient to see visualization software in two parts: the visualization design and the core software. In a traditional procedure library context, the visualization design is the user program, the core software the procedure library. In the context of an MVE, the visualization design is the visual program connecting modules into a pipeline, the core software is the set of modules provided with or added to the MVE. The design and core software together form a visualization executable.
- The visualization executable is conventionally described as the chain—Filter, Map and Render—preceded by a Data Source.

Another key issue is the location of the data that are to be visualized. Provision of data can be viewed as a service.

We can distinguish a number of general cases which we explore in the next sections.

3.2.1.2. Client-based visualization We distinguish client-based from server-based systems by determining where the visualization executable is located. Here we consider the case where the visualization is executed on the client.

Visualization design and core software both present on the client In this first approach, the distribution involves only the data, which is located remotely at a URL. The visualization software is held locally, and executes locally—and so this is a client-based approach. Typically the data is fetched from the URL as a particular MIME-type, causing the appropriate browser plug-in or helper application—in other words the visualization application—to be launched. A sensible application of this approach would be where data is held centrally, say data collected by a government agency, and can then be examined by any interested party who has the appropriate client software. Equally this mode supports a form of collaboration where a researcher places data at a URL for download and visual analysis by others.

It is also possible to launch the local visualization software directly, as the file import mechanism in many packages allows access to a URL.

This client/service provider split is illustrated in Figure 3.

Since large remote files are inconvenient to handle, especially if only a small amount of data is required, a further development, DODS, allows a subset of the remote dataset to be extracted. DODS, Distributed Oceanographic Data System[97], consists of a data server, which can provide a front end for several data formats, and a client library which can be relinked with any application that already supports the netCDF library. A DODS-enabled application is capable of selecting a subset of the variables in the archive and a selected set of values of any base variable. The DODS server can also return CSV (Comma Separated Values) which the DODS data archive to be read by an Excel client.

We can extend this model to handle collaboration as shown in Figure 4.

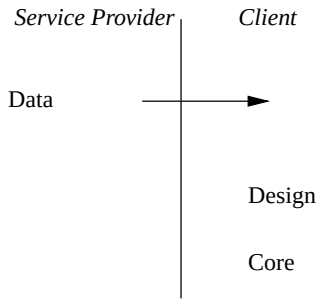

Figure 3: *Client-based: Design and Core on client.*

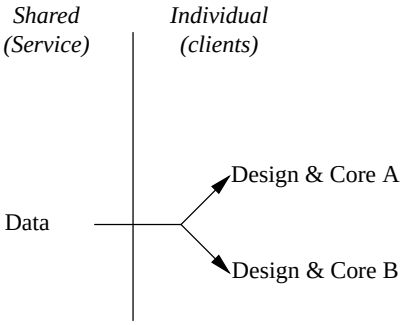

Figure 4: *Client-based collaboration: design and core on the clients.*

This diagram shows that the data is retrieved by two participants. Each can use their familiar, individually chosen visualization core software: for example, one using IRIS Explorer, another using Matlab. Although the variety of images that may result from using different visualization systems could lead to problems of interpretation, this variety could be fruitful so long as the participants are able to share their results.

In this case, none of the software needs tailoring for collaborative use, but the approach does rely on all the chosen systems being able to read directly from a Web URL, which is now quite common.

Visualization design downloaded from server: core software present on client As mentioned above, we can distinguish between the visualization design and the core software. For an MVE, the design is in the form of a pipework of modules which can be held remotely and the core software is the library of modules themselves and the basic MVE infrastructure. Thus one can see the MVE as an empty workspace into which a visualization program can be transferred and run. Thus a set of example or demonstration pipelines could be held on a server and has significant potential for education and training as studied by Yeo[140].

Note that the data could be local or remote (via a URL)

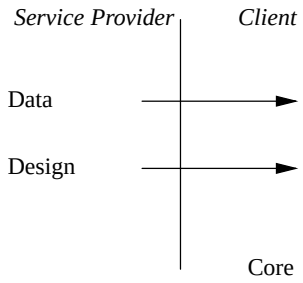

Figure 5: *Client-based: core software present on the client.*

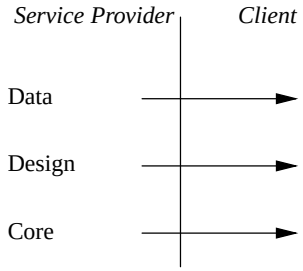

Figure 6: *Client-based: visualization design and core software downloaded to the the client.*

thus enabling both data and design to be delivered as shown in Figure 5.

Visualization design and core software both downloaded to the client In this approach, the distribution involves both the design and the core software, which is located remotely at a URL. This is fetched to the local machine where it is executed to create the visualization. Typically the software is fetched as a Java applet, to run within the framework of a Java Virtual Machine.

This case is shown in Figure 6.

3.2.1.3. Server-based visualization Here we consider architectures where the visualization execution takes place on a server.

Image Display on client One extreme is that as much as possible takes place on the server. This is suitable if a remote data archive allows plots to be specified, for instance the Space Physics and Aeronomy Research Collaboratory[100] at the University of Michigan, USA.

If we apply this to a collaborative situation, the application is centralised and each participant receives the results in terms of images or movie files. This is probably the most commonly used approach in distributed collaborative visualization. Any visualization system can be used and generic tools can be used to disseminate the results.

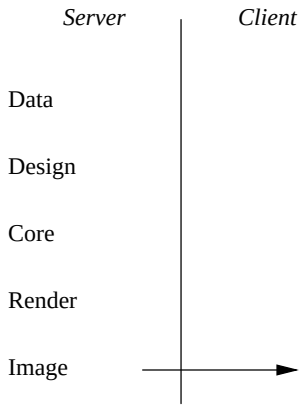

Figure 7: *Server-based: image display on the client.*

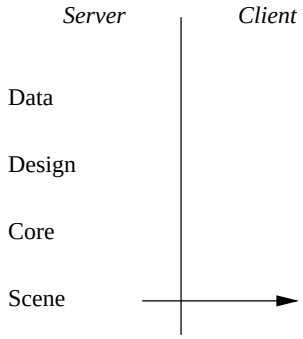

Figure 8: *Server-based: rendering on the client.*

For institutions which are in a position to host such a facility, the Access Grid (see Sections 1 and 5.2) offers an infrastructure to disseminate images and movie files.

If the application is centralised, it is nonetheless possible to provide any participant with some control, but—again—it relies on generic tools.

This is illustrated in Figure 7.

Model rendering on the client In this approach, the distribution involves carrying out part of the visualization task remotely, and transfering an intermediate representation to the local machine for eventual display (see Figure 8). Web standards provide us with a range of formats that can be used for the intermediate representation, depending on the nature of the data, for example:

- 3D Geometry: X3D (previously VRML)
- 2D Geometry: SVG
- 2D Images: PNG and JPEG
- 2D Movies: MPEG

3.2.1.4. Generalized MVE architectures An MVE offers intermediate possibilities for sharing control and distributing output and these arise because modules may be introduced which provide support for collaborative working. Thus it becomes possible to pass data and control information between instances of the visualization systems run by different users in the collaborative session.

In this way it is also possible to see collaborative visualization as an extension of ordinary single-user visualization. Collaborators may be introduced into a visualization session seamlessly. Collaboration becomes an extension to, not a replacement of the normal working environment.

Wood, Wright and Brodlie[139] have described reference models for this type of collaborative working. Their model is an extension of the familiar Haber and McNabb visualization model. Haber and McNabb[53] describe visualization in terms of the sequential composition of three types of processes, originally termed data enrichment, visualization mapping and rendering, but now (using terminology due to Upson[111]) referred to as Filter, Map and Render.

The extension of the model to encompass collaboration is accomplished by introducing, potentially anywhere in the pipeline, intermediate import and export points for data and control information. The model in its most general form is shown in Figure 9. For simplicity, the Data Source is omitted from the diagram.

F denotes the filter transformation; M the visualization mapping; and R the rendering. The horizontal arrows represent the progression of raw data through the transformation pipeline, emerging as an image. Control information can be imported from or exported to another pipeline at any stage. This is represented by the process parameters symbol. Similarly, data can be imported from or exported to another pipeline at any stage. This is denoted by the vertical arrows branching from the horizontal arrows between each processing stage.

The key concepts captured by the notation are:

- The generation of a visualization design may be described by a three-stage processing pipeline, following the Haber and McNabb model.
- Each processing stage is controlled by a set of parameters.
- A distributed collaborative visualization system can be modelled by a collection of pipelines, each of which represents the processing stages "owned" by a particular participant. These pipelines may be complete (contain all stages) or partial (contain only some stages, e.g. only the rendering stage). The stages visible to all participants can be indicated by a surrounding box.
- Control information may be exported from one pipeline to others in order to synchronise parameter values between pipeline stages.

Duce *et al.*[33] developed a similar model—the MANICORAL model. One additional feature in that is the explicit rep-

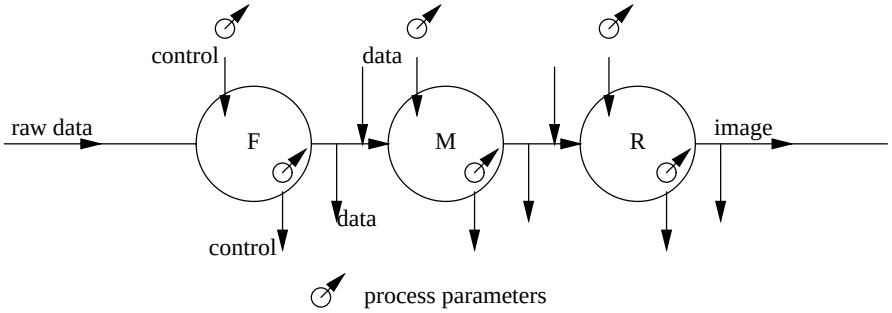

Figure 9: *Wood's model for collaborative visualization.*

resentation of the interaction mechanism that allows a user to control the parameters of a module, through the introduction of an associated control module. This encapsulates and simplifies issues related to arbitration between different input sources and dynamic changes in control and arbitration.

We note in passing that these MVE models deal with concepts and abstractions that might also arise when composing visualizations from collections of components that are Web Services or OGSA-based and that the models described here might be applicable in that context also.

For a visualization system to be a suitable basis for the kind of collaborative working described in these models, there need to be well-defined points at which data can be identified as passing between components of the system. Plausible types of system include: AVS and AVS/Express which have been adapted in this way in projects and the work done in the COVISA project which has become commercially available as part of the IRIS Explorer system.

The MVE approach offers a number of advantages over a simple client/server model.

- A simple client/server model assumes that the visualization core is all in one place.
- The simple model envisages the responsibility for the shared components being in one place. As the participants' expertise increases, responsibility for shared components may be divided. Experimental Internet applications such as collaborative dance, music or theatre could be a model here. In principle any collaborative visualization participant can initiate some modules and offer output or the possibility of sharing control.
- Not all participants will have the same degree of individual control.
- The simple flow of data from shared components to individually controlled ones may break down. Suppose we start with one participant controlling some shared modules. All the other participants take the output and look at the resulting 3D scene. Someone notices an interesting feature and wishes to communicate it back to everyone else.

3.3. Physical layer

The physical layer binds the logical design and core software to particular physical resources.

We outline the kind of decisions that can be made.

- An archived data source could be the master copy real data archive or a mirror copy available over the Grid or a locally cached subset.
- In the logical layer, decisions were made about the placement of subsets of the visualization application according to the roles played by individual participants. Here in the physical layer, a user's visualization application could be placed on a compute host according to computing and networking resources and may also depend on access rights. The software could be placed: on the user's local desktop computer; on a specific high performance computing engine, such as a cluster; or on a computer on the Grid with the best available fit. In principle, all the visualization processing functions could be placed elsewhere on the Grid, with the local desktop carrying out no more than a coordinating role.
- Rendering could take place on a local PC; or on a nearby computer with a powerful graphics accelerator; or a ray-traced movie sequence could be calculated on a render farm.
- The input and output environment needs to be chosen and this is discussed more fully in Section 5.
- In addition to using shared application software, collaborating participants need a means of communicating with each other and need conferencing and whiteboard tools.

In Figure 10, a possible collaborative logical environment is shown and also its binding to physical resources.

4. Analysis Factors

In this section, we describe a framework within which different collaborative visualization systems may be compared and contrasted. It sets out a range of factors that might distinguish one system from another. For potential developers of new collaborative visualization systems, it gives a checklist

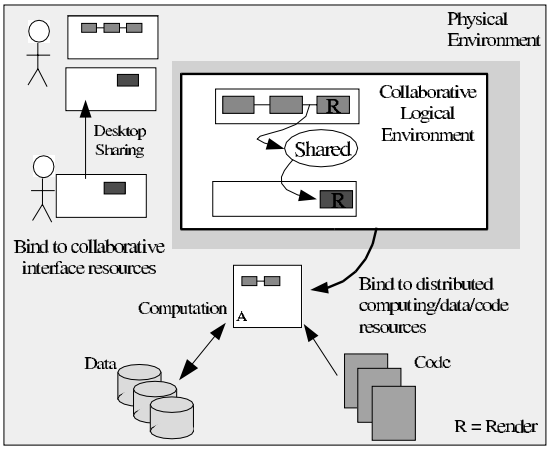

Figure 10: *Logical and physical layer for collaboration.*

against which a new design can be evaluated. The presentation here is based on the work of Wood in his PhD thesis[137], and his experiences in developing the COVISA extension of IRIS Explorer[139], but is also influenced by Duce *et al.*[32] and the earlier work by Pang[85]. Some of the description is specific to MVEs, where we talk of functionality involving modules, but other parts apply more widely.

- Base Visualization System.
  - In an ideal world, one would like to be able to collaborate between different visualization systems, since different researchers in a team project may have preference for using different systems. Such a goal is very difficult to achieve, since there is no standard functionality, no common architectural model, nor any standard for data exchange between systems. Thus in what follows, we shall assume that collaboration is between users of a specified base visualization system.
- Multiple Platforms.
  - In a collaborative session, it is more natural for each individual user to be able to use their own desktop system. There is therefore a requirement to support collaborative systems across a heterogeneous set of workstations, rather than tie a solution to a single platform. This goal, of interoperability between platforms (hardware and operating system), is definitely achievable, in contrast to the interoperability between visualization systems.
- Functionality.
  We can view collaboration functionality in terms of what information can be exchanged between users in a session:
  - *Exchange of data*. The system should be able to share data from any point within a pipeline (rather than simply being able to share the raw data, or the end-product of the visualization). This allows different collaborators to look at the data in different ways, and allows some data to be kept private from other users (see *privacy* later).
  - *Exchange of parameters*. The system should allow sharing of parameter input to modules; this enables, for example, two collaborators to jointly steer the visualization, by sharing control of modules in their (probably) common pipeline.
  - *Exchange of modules*. Tthe system should allow one user to automatically launch a module into the environment of their collaborators. This should also allow the automatic copying of the parameter set.
  - *Exchange of networks*. This extends the point above to cover a number of modules along with their interconnections. This would allow, for example, a more experienced user to set up a network for a novice, or allow a collaborator to send a fragment of a network to other collaborators complete with parameter settings and connections. This is potentially useful to help collaborators who join late catch up with the current state of the system.
- Participation.
  These aspects relate to how a user interfaces to the collaborative system:
  - *Setting up*. The initial setting up of a collaborative session should be as simple as possible, requiring the least effort on the part of the participant. Ideally, all elements required for setting up should be put in place by an administrator if possible.
  - *Joining/leaving*. It may not be possible for all collaborators to be available at the start of a conference, or to remain until the end. Facilities should exist for users to join and leave at any time.
  - *Automatic launch/connection*. As the collaborative system extends the modular paradigm of dataflow, where modules can be added to the system at any time, it is important to aid users by automating the external connections of the shared elements.
  - *Floor control*. Users should be able to set the type of conference control that they require. This can be used to offer different levels of access to a session to individual users - for example to create a 'See What I'm Showing' style of conference.
  - *Privacy*. In addition to participating in a conference where all elements are shared, users need to work privately while still remaining part of the conference. This is required to support conferences that contain parties with different skill backgrounds. For example, consider the design of an aircraft wing: a materials specialist may wish to look at tensile strength of a cross section, while a flow analyst will be interested in the air flow over that section. Both, however, need to be aware of how a single design change will affect their area of interest and hence need to collaborate.

This type of group needs the flexibility to share some parts of the pipeline while having other, domain specific, parts under their own control.
  - *Global View*. The ability to view the network editor of other collaborators is useful to reassure users that they understand what each user is doing. It also improves the collaborative map building process since an expert user can more easily aid a novice. However, this will be more than a simple view of any collaborator's entire pipeline since the ability to have private work contexts may mean that some pipeline elements are not shared. Note that this starts to place severe demands on the screen real estate, and so recent developments in large screen projection displays (such as AccessGrid, see section 1) become very useful here.

- System.
  The following factors relate to the behaviour of the implemented system:
  - *Performance*. The addition of collaborative elements to the single user system should not lower the overall performance of the tool. Also, shared data objects should be routed as quickly as possible between collaborators. Of increasing importance in this regard is the use of compression technologies, as the size of datasets continues to grow.
  - *Reliability*. Data objects passed into the collaborative session should be guaranteed to arrive at the correct output points intact. All participants should also be guaranteed that the data objects they are sharing are identical for all collaborators.
  - *Robustness*. The system should be able to survive the failure of any one part without the entire session collapsing. For example, if one user is suddenly unavailable the rest should be able to continue.
  - *Scalability*. Where the system is used for demonstration by one person to a group, there is a need for the system to be scalable. However, in situations where all participants are actively engaged in the visualization process, scalability of the system beyond four or five users is less important - since the cognitive load on the participants will become a greater issue.

- Target User
  The tools need to be applicable to a broad range of users, with different skill levels in their use of visualization systems. In particular, we need to address two distinct classes of user:
  - *Visualization Programmers*. These would be considered an expert user of the base visualization system. They would be comfortable with dynamically constructing visualization pipelines determined by the current direction of their investigation.
  - *Visualization End User*. These are not expert users of any particular visualization system, yet derive benefit from using tailored visualization applications.

5. Visualization Input and Output

In this section we consider data sources and presentation environments. The presentation environment is in effect the binding of the user interface to the physical layer in the reference model described in Section 2.2.

5.1. Input

The source of the data to be visualized can take a variety of forms: data stored in some kind of repository (e.g. local filestore or data portal), data generated by a program (e.g. some kind of simulation) and data streamed from an instrument of some kind. The Grid is raising interesting challenges about resource discovery and the possibilities for using semantic markup as introduced by the Semantic Web to enable higher level forms of resource discovery. Atkinson *et al.*[8] is a good survey of the issues of data access and integration.

The issue of which data formats should visualization systems support is a complex one which is beyond the scope of this paper. We note that there is increasing interest in the use of XML to markup scientific data in some communities. The paper by Brodlie *et al.*[19] gives a review of some pertinent activities.

Large scale scientific computations often take the form of simulations: mathematical models that may run for very protracted periods of time tracking the evolution of some physical phenomenon. An example is black hole simulations[12] carried out using the Cactus Grid-enabled problem solving environment[20]. An important question for such simulations is how to interact with the simulation whilst it is in progress, perhaps modifying the parameters in response to observed phenomena in the evolving simulation. The research questions here are how to provide a computational steering interface to a simulation, such that the user can view the state of the simulation through visualization and change the parameter settings, without disrupting the simulation and to do this in a Grid environment. Projects such as RealityGrid[88] and gViz[52] are addressing these issues.

5.2. Presentation environments

Nowadays there are many alternatives to the pervasive desktop CRT. Mayer[73] divides the experience of viewing visual media into four categories: the "postage stamp" experience in which the field of view is constrained by technology and bandwidth, the ubiquitous "television experience", the "theatre experience" which provides richer emotional involvement and the "immersive experience" where as Mayer puts it "mere eye scan motion is not enough and the viewer begins to engage head scan motion ... the viewer can leave the centre of focus and turn both their head and attention to discover and study details of the screen and contextual environment".

Modern PDA devices equipped with colour display and wireless network capability (such as the Compaq iPaq range)

fall into the "postage stamp" category. These devices are being used in DCV applications, see for example the work of Stegmaier *et al.*[104] and there appears to be growing interest in the use of this kind of device in portable wireless video conferencing, including AccessGrid conferencing[108]. Mayer's paper hints that the postage stamp experience can be enhanced to the level of the higher categories by careful design of the presentation. To realise this in a cooperative working setting where some participants are in the postage stamp category whilst others enjoy a theatre setting is a challenging research topic. The minimal graphics ideas of Herman and Duke may well find application here[55].

Large format displays may be considered to be in the theatre category. A recent collection of papers edited by Funkhouser and Li[46] covered this area. Topics included multiprojector display systems, research challenges in system architecture, remote visualization by distribution of compressed high-resolution images and case studies in using large format displays.

The "Office of the Future" project at the University of North Carolina at Chapel Hill[134] takes the view that the office will one day have ceiling mounted digital projectors and cameras that work together to support high-resolution projected imagery, human-computer interaction and dynamic image-based modelling. A related project at UNC, the "office of real soon now" (see Bishop and Welch[13]) describes experiments with large screen projection as the only computer display, using readily available equipment. The authors write enthusiastically of the experience and note the impact it has on local collaboration where a group of collaborators can share the viewing experience. Uselton[112] has reported experience with use of this kind of technology at Lawrence Livermore National Laboratory.

The AccessGrid fits into the theatre experience category also. Brodlie *et al.*[18] reported on experimental use of the COVISA collaborative visualization system within an AccessGrid context to a UK e-Science meeting in September 2002. This technology featured in an impressive demonstration of collaborative visualization in the AccessGrid, involving 14 UK AccessGrid sites, led by Lakshmi Sastry at Rutherford Appleton Laboratory, in May 2003.

For many visualization problems VR is a promising technology. Gallop[47] provides an introduction to the application of VR in this field. The CAVE[29], is extensively used for scientific visualization. The need to cooperate at a distance is no less when VR is involved.

VRML on the Web provides a way to cooperate using 3D visualization with the participants at different places, and working at different times. This approach has the advantage of relying on a formal (ISO/IEC) and accepted standard. Some examples of the use of VRML are discussed in Section 7.1.

Several projects have experimented with cooperative VR. This allows people at different locations to set up representations of themselves (avatars), explore the same world together and communicate with each other about what they observe. For example DIVE[21, 22, 31, 77] allows several networked participants to interact in a virtual world over the Internet. This kind of technology can be applied to visualization results if the geometric output can be input to the cooperative VR system. The multiple users in the virtual world would together explore the abstract scene created by the visualization system.

However this is a passive way of approaching visualization. Experience shows that users want shared control over the process not just shared exploration of the output. The experience of the VIVRE project[124] also shows that users studying large problems with VR expect to use the virtual environment to exercise control over the visualization process[15]. It is therefore a natural step to allow dynamic cooperative control of the visualization process from the virtual environment. However this is not common as yet. The University of Stuttgart are extending COVISE (discussed in Section 6.1.4) by adding VR capability to the system[27, 28]. The user works in a CAVE and some of the user controls in COVISE are available in the virtual environment. The user can access further controls outside the CAVE.

Augmented reality techniques also find application in collaborative visualization. The Studierstube project[105] allows multiple users in a "study room" to cooperate studying 3-dimensional scientific visualization. Each participant wears an individually head-tracked see-through head mounted display and thus has their own personal view. It is plain that each person's view is individually rendered thus increasing the computational load. New visualization calculations can be triggered from any participant's virtual environment. Subsequent work in this project has included the use of augmented reality in mobile collaborative working.

There is a growing interest in expanding the scope of immersive technology to include aural and haptic senses in addition to the visual sense. The Visual Haptic Workbench is an example of this trend[16].

6. Systems and Frameworks

In this section, we describe some standalone visualization systems and packages (see Section 6.1), concentrating on their distributed or collaborative functionality. We then discuss a few enabling technologies for distributed and collaborative visualization (see Section 6.2) which could be used alongside existing systems (even those which do not have these features built-in).

6.1. Visualization systems

For each of the systems below, we first describe some aspects of their architecture and features, before commenting on

their use in a distributed or collaborative environment, illustrated with examples where possible. Our selection (which is not intended to be exhaustive) is a mixture of commercial and public-domain systems; information on their availability can be obtained from the references (usually web-based) in this subsection.

6.1.1. Amira

Amira[5] is an object-oriented visualization system, based on Open Inventor[135]. Unlike many of the other visualization systems discussed here (see Sections 6.1.9, 6.1.2, 6.1.10), it is not based on a dataflow model. Instead, data objects are persistent in memory and accessed by Amira modules using the C++ interfaces of the data classes. Because data are passed between modules as C++ objects (as in a normal C++ program), there is no overhead for module communication. Users can extend Amira (adding new modules, data classes, editors and I/O methods) by deriving from an existing C++ class. Amira users create applications via a visual programming interface by connecting Amira modules together.

Using Amira to display the results of a simulation could be done[4] by standard methods such as connecting to the simulation via file transfer, or by embedding the simulation into an Amira module, or by writing a new module which communicates with the simulation via sockets or shared memory. We have been unable to find any references to distributed or collaborative visualization using Amira.

6.1.2. AVS5 and AVS/Express

AVS/Express[10] provides an application development environment for visualization using a visual network editor. Although in many respects an MVE, it is based on an object-oriented model which is accessible via the visual editor (and a script language V) that allows successively deeper levels of the module structure to be accessed. An application can be built making use of the higher-level modules resulting in an obvious AVS/Express look-and-feel or alternatively can be built using the many lower-level facilities—including a GUI toolkit—resulting in a more highly tailored interface. It is possible to create packaged run-time applications which has helped the product to gain acceptance in commercial markets, as well as the science, research and educational market.

Although the idea of a visually specified network of modules was already known, Express's predecessor, AVS5[9], is believed to be the first commercial system to apply the concept to computer-assisted visualization, having been in active use since around 1989[111]. The name AVS (without qualification) was originally associated with this product but has gradually come to be associated with the newer product, AVS/Express. The digit 5 refers to the version that original AVS had reached when AVS/Express was introduced. AVS5 is still updated with minor releases and platform updates, while AVS/Express continues to be under active development.

Part of the AVS/Express application can be executed on another host machine, by means of the Remote Module Support facility, which is controlled from a V file. AVS5 also supports the use of remote modules.

The MANICORAL project[32] used AVS/Express as the basis of an experimental distributed collaborative system. The Collaborative AVS project[66] used AVS5 to build a distributed collaborative system (called cAVS), which is still available for use[23]. The project was originally based at the San Diego Supercomputing Center before moving to the University of Texas.

6.1.3. Cactus

Cactus[20] consists of a central core component, called the *flesh*, and a set of modules called *thorns*. The thorns implement a range of computational codes which can run on distributed computing resources while being connected to, and orchestrated by, the flesh. The flesh controls when thorns will execute and how data is routed between them. Cactus comes with a set of thorns such as mesh generators etc. and also provides a low level API to allow users to integrate their own code as thorns. The systems appears to be controlled by scripts that determine the choice and execution order of thorns, rather than the graphical interfaces of other systems such as SCIRun (see Section 6.1.12).

Cactus builds on the Globus Toolkit[49] to provide secure access to remote resources, together with secure communications and job scheduling on remote resources. It also uses a number of other standard libraries and toolkits such as PETSc for scientific computation and HDF5 for data output.

Visualization is provided via standard products such as OpenDX (see Section 6.1.10), Amira (see Section 6.1.1) and IRIS Explorer (see Section 6.1.9). These systems effectively operate as thorns connected to the Cactus system via special modules written for each system which are able to read the data formats exported by Cactus (for example, HDF5) using the Cactus API. No specific mention is made of its use for collaborative visualization, though since the documentation says it links to IRIS Explorer, which offers collaborative tools (see Section 6.1.9), then collaboration could in principle be achieved in this fashion.

6.1.4. COVISE

COVISE[136] falls into the modular visualization environment category and was originally developed by Wierse *et al*. It allows a user to run modules both locally and remotely, employing a data manager process on each host to manage the flow of data beween modules. Like other distributed MVEs, such as IRIS Explorer (see Section 6.1.9), modules connected together on the same host communicate data through shared memory, while modules connected beween hosts pass data via the local data managers. The user interface is connected to the modules through a controller process.

Collaborative working was originally achieved by replicating the user interface on the desktop of each collaborator and connecting it to the single controller process started by the first user. Control is managed on a master/slave basis where the master can drive the system and the slaves get to see the results. As changes are made and the output updated, the new 3D scene is passed from the controller process to each user interface for local rendering.

More recently, a company named Vircinity[115] is marketing several new versions of COVISE, one of which offers collaboration. This operates differently from the original in that now the whole system is replicated at each site and the user interfaces are synchronised together. This can yield better performance since only small user interface changes need be shared across the network rather than potentially large 3D scene descriptions. Data is assumed to have been shared before the start of the session.

6.1.5. CSpray / Spray

CSpray[84] is the collaborative version of the Spray[83] visualization system developed at UCSC. It takes a novel approach to the visualization process by using a spray can metaphor and smart particles (or *sparts*). The user takes a spray can and manipulates it in 3D to aim into the data space. Pressing the top of the can sprays the sparts into the volume where they interact with the data generating abstract visualization objects (AVOs). AVOs can take the form of lines, spheres, polygons and can implement visualizations such as coloured "dust" particles, streamlines and contours.

The collaborative extension works through individual users being able to create spray cans which they can use to generate AVOs. These cans may be private and used only by their creator, or public and used by all. The AVOs created may be shared by all collaborators or kept private to allow a user to independently of other collaborators. Viewpoints may be shared, and the current position of collaborators within the scene may be viewed.

We have been unable to find much information on the current status of CSpray, although we note that it is listed as one of the visualization systems being used in the REINAS project[93] on the visualization of environmental data.

6.1.6. Ensight

Ensight[38] is a standalone application aimed at the visual analysis and postprocessing of engineering data. It offers a standard set of visualization and plotting algorithms with interfaces to several engineering solvers for CFD and FEA problems but, being aimed at end-users rather than developers, is not extensible (no new algorithms can be added, although users can create their own readers to import data in custom formats).

A more advanced version of the package (called Ensight Gold) incorporates extra features for handling large data sets including parallel processing, multipipe rendering for output to immersive environments and collaboration. The collaboration architecture[37] is a master/slave one: it allows the display from a *pilot* user's Ensight session to be sent to that of a *copilot* user, so that all actions of the pilot are reproduced on the copilot's machine. At any point in the collaboration, the copilot can take over control by requesting to be the pilot; if this request is successful, the users swap roles. Collaboration is currently limited to two participants.

6.1.7. FAST / FASTexpeditions

FASTexpeditions[41] is the collaborative extension to the Flow Analysis Software Toolkit (FAST[40]) developed at NASA Ames. FAST is a toolkit of programs that run simulataneously allowing the user to visualize the result of numerical simulations. It is designed primarily to visualize unstructured data from CFD type applications. The FAST system allows the generation of an audit trail of interactions that can be saved so as to allow a session to be replayed. This audit trail is used in both the web component and the collaborative component of FASTexpeditions.

In the web environment the audit trail is used as the trigger to start FAST as a helper application to the web browser. Once the audit trail file is downloaded and read into the localy executing copy of FAST it takes control of the system, guiding the user through the visualization process. In the collaborative setting, multiple distributed copies of FAST are running (one per user) and the audit trail file is streamed in real time from one user's copy (the *pilot*, in FASTexpedition's terminology) to the rest (the *passenger*s). Thus the pilot is able to control all of the visualization systems and guide the passengers through the visualization process. The pilot is distinguished from the other users by a token; passing this to one of the passengers causes them to become the pilot.

6.1.8. IDL and PV-Wave

IDL[61] and PV-Wave[89] are array-oriented languages for the creation of visualization and analysis applications. We discuss them together because they share a common heritage in design and architecture (PV-Wave was an offshoot of the original IDL development tree), and still have much in common, even at the level of specific commands. Users of these systems can enter commands from the keyboard, where they are immediately executed, or combine them into scripts which are compiled and executed. Both systems contain a large number of high-level graphics, image processing and numerical analysis functions, along with routines for the control of an application's interface. In the past, IDL and PV-Wave have been viewed[129] as being stronger for the display of data in one and two dimensions than in three; this has been improved in later releases. Thus, IDL now includes Object Graphics, which provides an object-oriented interface to graphics commands which are based on OpenGL,

and PV-Wave now includes new functions which allow access to some of the VTK (see Section 6.1.14) algorithms.

Although IDL and PV-Wave are both proprietary languages, they incorporate interfaces to standard languages such as C and Fortran, and also support communication with other applications using Remote Procedure Calls. There is also a Java-based interface to PV-Wave[67] through which a Java client can send data and graphics commands to PV-Wave running on the server. PV-Wave then generates a graphics file which is returned to the Java client and displayed. Apart from this, we have been unable to find any references to distributed or collaborative applications using IDL or PV-Wave.

6.1.9. IRIS Explorer

IRIS Explorer[44, 131] is a modular visualization system based on the data flow model. The system provides a large selection of modules, listed in the *Librarian*, which the user launches into the workspace (*Map Editor*) and connects together with wires to form a dataflow pipeline, or map. The system can be extended by users adding their own code as modules and integrating them into IRIS Explorer using the provided API. These modules could be anything from visualization algorithms generating geometry to simulations producing data.

This system provides a mechanism to allow modules within a pipeline to be run on a number of remote computing resources. The user opens a new Librarian on each remote resource with the authentication being managed by `rsh`. Each remote Librarian lists local modules which can be dragged into the Map Editor and connected into the Map in the same manner as locally running modules. Remote modules are simply differentiated by having the name of the host on which they are executing on their control panel. IRIS Explorer transparently manages the data transfer between resources as the data passes along the pipeline, using shared memory for modules connected together on the same host and through sockets for modules connected across host boundaries.

Collaborative working can be achieved using a set of provided modules originally developed as part of the COVISA[139] project. These modules allow connection to a collaborative session and provide a tool to aid sharing of the map construction process. In addition, data and control can be shared between the separate instances of IRIS Explorer being run by each collaborator. The modules can be used in the exploration phase of visualization where the map may be changing as new modules are tried and different data is shared, and also in pre-defined maps packed as applications for end-users.

6.1.10. OpenDX

OpenDX[80] is a modular visualization system based on the dataflow model. It originally began as the IBM commercial product Visualization Data Explorer, but was offered by IBM as an Open Source project in 1999. There is an active mailing list and improvements continue to be made to the system.

Many example applications of its use are available within the distribution. Others may be obtained from Treinish[79] and via the project website[80].

Like other MVEs (see Sections 6.1.2 and 6.1.9), OpenDX can access modules on multiple hosts, but we are not aware of any work to develop a collaborative visualization system based on it.

6.1.11. pV3

pV3[90] is a library for the real-time visualization of large-scale transient (unsteady) systems. Based on an earlier system called Visual3[123] (pV3 stands for *p*arallel *V*isual3), it has been re-designed specifically for the co-processing visualization of data generated in a distributed compute arena. One of its design goals is to allow the compute solver to run as independently as possible–thus for example, pV3 can be configured to plug into the simulation, display the transient data, and unplug from the calculation.

pV3 provides a centralized interface to a distributed computation. The pV3 user inserts calls to the pV3 API into the simulation code which extract visualization data from the simulation data structures and passes it to the pV3 server. pV3 uses PVM for passing data around the network (though it can also use MPI). Without a pV3 server running, the number of members of the PVM group `pV3Server` is checked at each simulation timestep. If no servers are found, no action is taken. When a pV3 server starts, it enrols in the `pV3Server` group and, when the solution is next updated, the visualization session begins. At each subsequent timestep, the appropriate visualization data is calculated, collected together and sent to the active server(s), which display the data.

Since the number of pV3 servers is unrestricted, a collaborative application could be constructed by starting up a set of them at different locations and connecting them to the simulation. The view which each server provides to their user is independent, but one server can pass its 3D viewpoint to the other servers in the group so that they're all looking at the same part of the data. pV3 also supports a soft cursor (for each additional server) so that one can see where another user is pointing. Computational steering (which is also supported by pV3) is controlled in these settings so that only one user can modify the state of the simulation at any time. The control can be handed off to another user, if desired.

6.1.12. SCIRun

SCIRun[86] is an MVE developed at the SCI institute at Utah university. It allows the user to connect a set of pre-written routines together in a workspace to form a dataflow network.

These routines execute as independent threads within a single executable, in contrast to other MVEs such as IRIS Explorer (see Section 6.1.9) which run modules as individual processes. Despite SCIRun operating as a single executable, the opportunity still exists for users to extend it by adding their own code. In this case, it is compiled directly into the system itself.

Advantages of using threads are that there is marginally less overhead during startup, when compared to forking new processes, and all threads may access data directly without having to be connected through shared memory. A disadvantage of threads is that they must exist on the same machine, and in that respect SCIRun has been targeted at shared memory parallel systems. To extend beyond this limitation, more recent implementations of SCIRun have employed "Bridging"[65] which essentially allows SCIRun to access features of commonly available libraries. One level of bridging supports socket communication to external processes.

SCIRun has been used for computational steering within the Uintah[30] which is designed to run on massively parallel distributed memory architectures. At present, no collaborative components built directly into SCIRun have been reported, but it has been used with VNC[95] to provide collaboration through screen sharing.

6.1.13. VisAD

VisAD[57] is a Java component library for interactive and collaborative visualization and analysis of numerical data. It makes use of Java's Remote Method Invocation (RMI) technology, which allows methods of remote Java objects to be invoked from other Java virtual machines, possibly on different hosts (or on different Java interpreters on the same host).

VisAD applications can run in any of three networked modes: *standalone*, *server* or *client*. Standalone mode is the ordinary non-networked application mode. Server mode is almost identical to standalone mode, except that, once the data setup is complete, the application listens for (and responds to) clients in the background. In client mode, the application connects to the server application and has it send the information necessary to construct copies of any (or all) of the server's `Display`s. (`Display` is the top-level object in the VisAD visualization architecture. It contains a window into which data objects are rendered, along with some associated controls.) Once that is done, the server and client remain connected and forward any significant changes to each other, so that the `Display`(s) remain synchronized. A client's `Display` information comes from the server. All data for the copied `Display` comes from the server as well, so if the server application is terminated, the client's `Display` will no longer work.

The VisAD home page[120] contains a number of examples of collaborative applications. For example, the *Milky Way Galaxy Designer* is an interactive application which allows the user to adjust model parameters in order to match a sky map to earth observation data. In collaborative mode, all users see the same set of parameter values, and all can adjust them. Documentation on how to make a VisAD application collaborative is also available[118]. Another example (the *2D Shallow Fluid Model*) demonstrates the use of VisAD in computational steering. Here, users can set the initial configuration of the simulation and change its parameters (both physical and numerical) during the run.

The construction of distributed applications in VisAD is facilitated by its event-driven design[119]. Typically, a VisAD object will implement the methods of a listener interface before registering itself as a listener with another object. Whenever something interesting changes within this second object, it passes an event to all its listeners which, through RMI, could be on other machines. Thus, VisAD could be used in the development of an application which is distributed across a number of machines in a network.

6.1.14. VTK

VTK[128, 94] is an object-oriented software system for 3D computer graphics, image processing and visualization. It consists of a C++ class library, together with several interpreted interface layers including Tcl/Tk, Java and Python which can be used to access the classes and build applications (applications can also be written in C++, of course). VTK is based on the dataflow model; application builders use it to create a VTK *pipeline* whose elements correspond to the elementary steps in the Haber-McNabb visualization reference model[53]: a *Source* for data, a *Filter* to convert the data into geometry and a *Mapper* to map the geometry to graphics.

VTK has been used as part of a tool for the development of collaborative visualization applications for CAVE-type environments[127]. These types of systems require the rendering of the same geometry multiple times simultaneously to different graphics pipes, which VTK does not support (since its geometry structure cannot be traversed by multiple processes simultaneously). Accordingly, the geometry, once created in VTK, was then passed to IRIS Performer for (multi-pipe) rendering[126]. Other elements of the tool[127] included an application framework to assist with the construction of tele-immersive applications.

VTK has also been used as the visualization component of the ICENI toolkit (see Section 7.2.4).

6.2. Enabling technologies for distributed collaborative visualization

Here, we briefly discuss a few examples of enabling technologies, including remote display and rendering (see Sections 6.2.1 and 6.2.2) and some of the more general work on peer-to-peer technology (see Section 6.2.3).

6.2.1. Shared desktop display

A very simple yet powerful approach to collaborative visualization is to make use of a tool for remote display. Probably the two best known of these are VNC, which is multiplatform, and Microsoft NetMeeting, which is Microsoft-specific. These allow the desktop of one machine (the 'server') to be displayed on a number of clients (the 'viewers'). The success of these methods stems from efficient compression of the frame buffer so that it becomes feasible to transmit it to a number of viewers, even when there are frequent screen changes (although too frequent changes reduce the performance considerably).

6.2.1.1. VNC VNC stands for 'Virtual Network Computing' and was first developed at AT&T. It is now being developed in two strands: RealVNC[91] is being evolved by the original developers as an open source project; tightVNC[109] is a derivative of VNC, offering different compression algorithms. VNC has been successfully used in AccessGrid visualization experiments[18], and Stegmaier *et al.*[104] use VNC for efficient image compression (having rendered OpenGL remotely). A nice feature of VNC is that input control can be passed between server and viewers.

6.2.1.2. Microsoft technologies Microsoft have included remote display with their NetMeeting product for several years, and more recently have included it with MSN Messenger[78] (version 4.7). The 'passport' service is used to handle authentication.

6.2.2. Remote rendering

6.2.2.1. SGI Vizserver Vizserver[125] is an SGI product designed to allow remote access to the high performance graphics offered by its hardware. Traditionaly, to benefit from the graphics performance of machines like an SGI Onyx with Infinite Reality graphics, users had to be able to view a display connected directly to the machine. Vizserver allows users to have access to that performance by running the server component of the software on the Onyx and connecting to it using client software provided by SGI. Once connected, the client software opens a local window and displays an SGI desktop from which users may run any normal SGI application. The OpenGL output from these programs is rendered to a buffer on the server and the pixels transferred across the network to the client. The user is able to select from a number of compression schemes to trade off visual quality with framerate.

Collaboration is achieved by allowing multiple (up to 5) clients to attach to Vizserver and each recieve the same image. Clients are available for WindowsNT/2000/XP, Solaris and Linux. Each distinct session attached to the server uses the graphics resources of a whole graphics pipe.

Vizserver has been used by Nigel John at Manchester University to help surgeons at Manchester Royal Infirmary access 3D views of patient data whilst in the operating theatre[113, 64, 74]. Vizserver is used to give access to the 32-processor SGI Onyx at Manchester University from a laptop in the operating theatre. This display is projected onto the theatre wall so surgeons can manipulate three-dimensional reconstructions of the patient's organs whilst operating on them in real life.

6.2.2.2. Chromium Chromium[58], is an open source graphics library[24] which allows distributed network rendering for OpenGL applications. It does this by intercepting OpenGL API calls, and then modifying, deleting, replacing or augmenting them. Thus, for distributed rendering, the commands are split across a collection of graphics cards. A particular feature of Chromium is that it is non-invasive—no modification (or even recompiliation) of the application is required.

Chromium has been used in the ICENI project (see Section 7.2.4) for distributed rendering across the network.

6.2.3. Peer-to-peer technologies

A powerful model for distributed computing has been developed in the JXTA Project for Peer-to-Peer (P2P) computing[68]. JXTA provides a networking framework to support P2P programming between a group of peers. Existing single-user applications written in Java can become collaborative, by replicating the application at each peer, and using the concept of JXTA Pipes to share any interface controls between the peers.

Commercially, JXTA has been used in the VistaBoard charting system[122], and in the context of Grid computing it has been used in the Triana system[106].

7. Current Projects and Future Work

7.1. Web-based Visualization Projects

An important class of distributed visualization applications are based on Web technologies, and in particular on the client-server concept. In this subsection we describe a number of projects in which different aspects of web-based visualization have been explored. We follow the classification introduced in Section 3.1.2.2: *Full service* where the visualization is created remotely by the service provider - here the visualization design and core software are both supplied by the visualization service provider, and the service provider also executes the software for the user; *Software delivery* where the service provider transfers over the network to the user, the software needed to create the visualization—here the design at least is transferred, plus perhaps some core software, but the execution is the responsibility of the user; and *Local operation* where the visualization software is assumed to be already available locally, and is just triggered by download of data over the web—here the design and core is with the user, who also has responsibility for the execution.

7.1.1. Full Service

In this category, the main visualization processing is carried out on a server and graphical data is transmitted to the client, for viewing within the browser. There are different styles, determined by the extent of the rendering delegated to the client: the graphics data may be in the form of a rendered image, requiring only image display software; or it may be in the form of a 3D VRML (or X3D) model, requiring a VRML browser on the client.

This approach makes modest demands on the client side—in terms of processing power (only rendering is required); in terms of software availability (only image or 3D graphics software needed, and commonplace in browsers anyway, with VRML/X3D an ISO standard for 3D Internet graphics); and in terms of training (only ability to use image or VRML tools required). Nothing comes free of course, and this approach requires the presence of a server with the following ingredients: processing power to create the visualization as an image or 3D model; visualization software itself; and human involvement to create suitable visualization applications. Moreover, if several clients connect at once, then the processing requirement on the server can be quite severe. In early versions of this approach, the user interaction was through an HTML form, but in later versions a Java applet is usually used in order to provide a richer interaction environment.

An early example of this approach is described by Wood *et al.*[138], based on IRIS Explorer as the visualization system. The basic principle of the method is as follows. The user enters details on an HTML form to specify the location of the dataset (as a URL say), and a 'recipe' for the style of visualization required. The form is processed by a CGI script on the server, and a visualization server is executed. This invokes IRIS Explorer, using its scripting language (Skm) to define an appropriate pipeline. The output of the pipeline is a VRML file, which the visualization server returns to the browser for viewing.

A demonstrator[3] was created, to view air quality information in the UK—and this can still be accessed today. The data is collected on an hourly basis at a number of locations throughout the UK, and the Atomic Energy Authority (AEA) publish the data on a Web site. The demonstrator allows a user to select the time and location of data, and the pollutant to be analysed. The approach can in principle be extended to the presentation of many different types of public service information—for example, road traffic data, weather forecast details, and so on.

Wood[139] also extended his system to allow a number of collaborators to investigate data over a period of time, giving a form of asynchronous collaboration. It works as follows. A first user creates a visualization using the form front-end to the IRIS Explorer system running on the server. If an interesting visualization results, they may choose to store the parameters which defined that visualization. Note that these parameters completely define the visualization, and of course require much less storage than saving, for example, the VRML output.

Now consider another user in the collaborating group. On fetching the HTML page, they can also gain access to the data saved by the earlier user. They can use this as a starting point for their own study of the data: they can add their own textual comment on the visualization, or they may choose to change some of the defining parameters, creating a new visualization. If this new visualization is of interest, its defining parameters may be saved for later use by other collaborators. This sequence of stored definitions effectively forms a tree of exploratory visualizations, built up over a period of time by the collaborators.

Note that this approach can even be used by a group working in the same place, and so it could also be placed in the same-place, different-time quadrant of the Applegate matrix.

As mentioned earlier, more recent examples of server-side web-based visualization tend to use Java applets to provide a more flexible GUI than the forms interface used by Wood and colleagues. In the Vis-a-Web system, Pagendarm and Trapp[110, 81] provide an applet front-end to allow users to access the HighEnd visualization sysytem running on a server; a novel feature is the fact that software from different sources can be used to make a *single* visualization—VRML files from different sources can be combined before transfer to the browser. Lefer[70] builds a web-based system using the VTK toolkit of Shroeder *et al.*[94] as underlying software. Treinish[79], in a paper on visualization design for operational weather forecasting, describes how IBM Data Explorer can be used in a Web environment. He envisages a partitioning of Data Explorer into a client-server system, with a Java-based applet on the client interacting with Data Explorer on a server. Finally, Walton[130] envisaged the publication of visualizations as VRML scenes in which the visualization algorithm, containing instructions for modifying the geometry, is directly incorporated into the scene as a scripting node.

An important thread of activity was sparked by the work of Hendin *et al.*[54]. They describe a web-based approach to volume rendering which exploits 2D texture mapping hardware. On the server, three stacks of rectangular faces are constructed, each stack orthogonal to one of the co-ordinate axes. These are stored as a VRML scenegraph. On the client-side, the current viewpoint is used to select the 'most orthogonal' stack and this is rendered using texture hardware.

This work has been developed further in a series of papers by the graphics group at Erlangen. Engel and Ertl[34] improve the clipping operation, and introduce a collaborative aspect. The server keeps a record of participating users, and allows the broadcast of viewpoint, clipping region or transfer function from one user to the other collaborators. In later work, Engel *et al.*[35] describe an approach which uses 3D texture hardware on a remote server, transmitting the resulting images to the client for display within an applet.

Engel *et al.*[36] have studied isosurface extraction as a server-based application. The creation of the surface using a Marching Cubes algorithm is carried out on the server, and the resulting polygonal data structure is streamed to the client. They study two variants: one in which VRML is used, the other in which an OpenGL 'immediate rendering' approach is used. The OpenGL approach is substantially faster, for a large number of triangles.

A Visible Human Web Service is offered by EPFL, Switzerland[56, 121]. This uses a Java applet front-end to select the slice of interest through the Visible Human (this can be at any orientation, and at any position); the request is passed to a server process which extracts the relevant slice and returns to the applet for display.

7.1.2. Software Delivery

In the 'Full service' examples above, the use of Java applets is essentially limited to the provision of a good user interface, and the actual visualization process is handled remotely by a server. However there is an important group of examples where the Java applet carries out the visualization itself, as well as provide the user interface.

An early example is described by Michaels and Bailey[76]. VizWiz is a Java applet which creates isosurface, cutting plane or elevation grid visualizations, from data supplied by the user. A difficulty with this general approach is that the data must be held locally on the server which delivers the applet, for security reasons. This is not ideal! It is likely that the data will either be held locally by the user on the client, or in the case of shared public data, at some URL (not necessarily where the applet is). Michaels and Bailey solve the problem for local data by using the Netscape file upload option, which transfers a local file to the server. A major snag is that the data is transferred to the server and back again, just to circumvent security restrictions!

An alternative solution has been proposed by Kee[69] and Stanton[103], allowing data from any URL. The data are fetched temporarily to the server by a Java application (distinct from the visualization applet) which executes on the server and, being an application, is not subject to the security restrictions of applets. However, again there is large data transfer involved.

Another example of this approach is given by Wegenkittl and Groeller, in their FROLIC[133] system. This creates visualizations of dynamical systems, using a variant of Line Integral Convolution and other methods. The systems are defined analytically, and so there is no problem with uploading data.

There are two fundamental limitations to this approach: first, the user is limited by the power of their desktop machine; second, the upload of data seems against the spirit of Java applets. However neither of these limitations apply in applets which are aimed at teaching or demonstration: here the examples can be simple and need not be computationally demanding, and the data can be provided from the server,as part of the demonstration. Many excellent examples of visualization applets for teaching can be found on the web: for instance, Falstad[39] has some applets for maths and physics teaching; the Vestac project[114] has some statistics applets.

In terms of the 'design' and 'core software' distinction introduced in Section 2.2, one can view the applet as the design and the Java Virtual Machine as the core software. If we follow this idea through to MVEs, we get a similar idea where the 'design'—that is, the description of the dataflow pipeline—is supplied by the visualization service provider, while the 'core software'—that is, the MVE operating environment and module code—is the responsibility of the user. Thus one can see the MVE as an empty workspace into which a visualization program, that is, a specific pipeline of modules, can be transferred across the Web. Thus a set of 'example', or 'demonstration' pipelines can be held on a server, and associated with a particular MIME type; on being fetched by the browser, the pipeline can be automatically loaded into the empty workspace as the application is launched. This has significant potential for education and training, and was prototyped by Yeo[140] using IRIS Explorer as the MVE. The pipeline of modules is then available for use in the normal manner by the user. An interesting variation would use the distributed computing capability of MVEs, where modules can run remotely—allowing the visualization service provider to at least share some of the responsibility for execution.

7.1.3. Local operation

Although we are describing this last, historically it was the original approach. The first instance we are aware of is the system described by Ang *et al.*[6]. Their work focussed on volume visualization, and they defined the MIME-type 'hdf/volume' to identify volumetric data in the NCSA Hierarchical Data Format (HDF) representation. On receiving a file of this MIME-type, they were able to invoke the inter-client communication facilities of the Mosaic browser, to fire up the user interface of their volume visualization system, VIS.

Another early example of this is the Web version of Vis-5D (now evolving as Vis-5D+[117]). Vis-5D data files retrieved over the web can be set to launch the Vis-5D application on the client. The original Vis-5D web pages[116] give examples of its use in this way, for instance to provide daily weather forecast visualizations from data delivered by the US National Weather Service.

This is the most limited of the three approaches and is probably mainly of interest now for historical reasons. Most MVEs will allow data to be read in from a URL directly, and so this functionality becomes available in a different way.

7.2. Collaborative Visualization Projects

In this subsection, we discuss a few projects currently active in the UK e-Science programme[107] which are using or developing methods of collaborative visualization.

7.2.1. climate*prediction*.net

The climate*prediction*.net[25] public-resource computing project aims to harness the spare CPU cycles of a million individual users' PCs to run a massive ensemble of climate simulations using an up-to-date, full-scale, three-dimensional atmosphere-ocean climate model. The ensemble of runs is required in order to establish uncertainty limits on model predictions, leading to better-informed policy decisions regarding climate change. Participants in the project download the model and carry out a series of runs, each of which is characterized by a unique set of parameter values that are obtained from a cental server. Results are uploaded to the server at the end of each run for later analysis.

Visualization plays an important role in this project, both on the desktop, and in the analysis phase. Unlike some other public-resource projects, a high degree of long-term participant interest is needed because of the resource requirements of each run. Accordingly, desktop visualization has been added[101] to the model in order to show its real-time evolution. An important design consideration for the visualization is the need to take the background of the participants into account—the interface must be simple enough to be run by inexperienced PC users with a limited scientific background, and yet compelling enough to be readily grasped, since one of the broader aims of the project is to raise awareness of the issues surrounding climate change.

The desktop visualization has been implemented using a public-domain implementation of Open Inventor[135]. The analysis phase of the project, during which data from the individual runs will be collated and compared, will provide opportunities for the application of distributed collaborative visualization systems (see Section 6.1).

7.2.2. GAPtk

The Grid-Aware Portals toolkit (GAPtk) project[48] aims to provide scientists with a set of visualisation utilities based on the Web and Grid services model along with appropriate APIs that will enable them to exploit the power of the Grid for their data analysis. One of the primary goals of the project is to retain existing familiarity with domain-specific application environments. The layered architecture of the toolkit is designed to hide the Grid computing structure from the portal user. It consists of a set of interfaces on the client side, which talks to the GAPtk server using SOAP[99]; the server communicates with the Grid using Globus[49].

The main results from the project will be a set of software modules and services that can be embedded within portals for problem solving environments. The toolkit is being used to provide visualization capabilities within the GODIVA[50] project, which is investigating ocean circulation and its effect on climate change.

7.2.3. gViz - Visualization Middleware for e-Science

gViz[52] is a project funded by the UK e-Science Core Programme, and has two main targets: first, to grid-enable two existing visualization systems, IRIS Explorer (see Section 6.1.9) and pV3 (see Section 6.1.11), so that visualization tools are available as early as possible for users of computational grids; second, to develop some longer term thinking on distributed and collaborative visualization, where XML languages are used to describe visualization data, and visualization programs themselves.

In terms of IRIS Explorer, a demonstrator has already been built in which the modules in a network can be distributed across a set of Grid resources, but controlled from the desktop. This is achieved in a secure manner using Globus middleware to replace the existing insecure `rsh` mechanism. The COVISA collaborative facilities (see Section 6.1.9) are immediately available, and so we have a framework for Grid-enabled distributed collaborative visualization. An important application of this is in computational steering, where the simulation model runs on a remote server, but is controlled from the desktop. Indeed, a separate computational steering API is being developed: this will lead to a more flexible approach where the simulation runs externally to the visualization system, and the visualization system will act as a front-end monitoring tool which can connect to the simulation to check its progress, change parameters, or receive results for display.

The use of an XML language to describe visualization data promises to lead to better inter- operability of visualization systems, and the more effective development of new visualization software (which can be written to operate on these standard datatypes). Using XML to describe visualization programs—for example the way in which pipelines are constructed in an MVE application—will pave the way for the exchange of visualizations between users of different systems. In fact, it could almost be seen as a realisation of the conceptual layer discussed in Section 2.2.

7.2.4. ICENI

The goal of the ICENI[60] project is the provision of high-level abstractions for scientific computing which will allow users to construct and define their own applications through a graphical composition tool that is integrated with distributed component repositories. The project aims to deliver this environment across a range of platforms and devices on the Grid using a scheduling service. ICENI is being implemented in Java and JINI, but is able to interoperate with the Open Grid Services Architecture (OGSA)[43].

One of the applications of ICENI—at least, for

demonstration purposes—is computational steering and visualization[102]. The ICENI framework is used to link together a visualization *client* and *server*, and to pass data to the server from a running application. The visualization server hands the data off to a *renderer* (current demos are based on VTK (see Section 6.1.14) which can then send the graphical output back to the visualization client. This can either be done using standard OpenGL remote rendering, or using Chromium (see Section 6.2.2.2).

Collaborative visualization can be supported either by connecting multiple renderers to the server (for the display of different datasets from the same application) or by connecting multiple clients to a renderer (for the display of the same dataset at multiple locations).

7.2.5. RealityGrid

RealityGrid[88] is a project which aims to examine how scientists in the condensed matter, materials and biological sciences communities can make more effective use of the distributed computing and visualization resources provided by the Grid. Development of RealityGrid is being run along twin tracks: a "fast track" uses currently-available Grid middleware to construct a working grid, while a "deep track" is harnessing computer science research to creating a flexible problem-solving environment in which to embed the RealityGrid.

ICENI (see Section 7.2.4) is being used within the "deep track" to enable collaborative visualization and computational steering within RealityGrid[59].

8. Conclusion

As networks become more pervasive and collaboration becomes more and more important in many areas of life, it is natural to ask what computing can contribute. In many areas of science and engineering and increasingly in other areas too, visualization plays a key mediating role in communications between humans.

As the area of distributed and collaborative visualization matures, we can expect the research ideas discussed in this report to translate into products in the market place. We have already seen this in the area of MVEs, where the COVISA extension of IRIS Explorer is now an integral part of the commercial release. The usage of these systems in the field is likely to stimulate a further round of research, as strengths and weaknesses emerge.

We may also look forward to a future where visualization service providers start to exploit the service-oriented architectures described in this STAR. We can expect them to harness web service and Grid service technologies, in order to deliver visualization to the desktop of the scientist, the engineer, or the clinician - indeed the market must be very large.

We have attempted in this STAR to give a review of approaches to supporting the use of visualization in collaborative situations, through what we have termed distributed and collaborative visualization. We have examined approaches based on a service-oriented model, looking at both MVEs and web-based approaches. This is a field in which we may expect to see continued growth over the coming years, as Grid computing becomes more pervasive and as high quality networking becomes more available and more affordable. Recent research in Grid-based visualization has just been published in a special issue of IEEE Computer Graphics and Applications[98].

It is interesting to note that a distributed and collaborative approach was used in the preparation of this STAR by the five authors. The text was developed individually, but with regular review meetings to discuss, critique and re-organise, held using AccessGrid nodes at Leeds, RAL and Oxford: the same time, different place quadrant of the Applegate matrix. This excellent technology enabled us to meet much more often than we would have done otherwise, yet we never met as a group in the same location. The reader is left to judge the efficacy of this approach from the point of view of the end results! For the authors, this was a highly effective use of time, though it was very helpful that the authors knew each other well and have built up a common understanding of at least some of the material over a period of years.

Acknowledgements

DAD and JRG gratefully acknowledge fruitful discussions with colleagues at the University of Lancaster in the Visual Beans project and colleagues in the gViz project. KWB and JDW likewise thank all those who have contributed to our understanding of the subject—notably Helen Wright (now at the University of Hull) and many students at the University of Leeds. JPRBW thanks Bob Haimes for helpful discussions. We appreciate funding from EPSRC, in particular recent funding from the e-Science Core Programme towards the gViz project.

References

1. G. Abram and L. Treinish. An Extended Dataflow Architecture for Data Analysis and Visualization. *Computer Graphics*, 29(2):17–21, 1995. 3
2. Access Grid website. `http://www.accessgrid.org`. 2
3. Air quality data. `http://www.visualization.leeds.ac.uk/`. 20
4. Amira and simulation output. `http://www.amiravis.com/usersguide30/faq.html#A193`. 15
5. Amira website. `http://www.amiravis.com/`. 15

6. C. S. Ang, D. C. Martin, and M. D. Doyle. Integrated Control of Distributed Volume Visualization Through the World Wide Web. In D. Bergeron and A. E. Kaufman, editors, *Proceedings of IEEE Visualization '94*. IEEE Computer Society Press, 1994. 21

7. L. M. Applegate. Technology Support for Cooperative Work: A Framework for Studying Introduction and Assimilation in Organizations. *Journal Organizational Computing*, 1:11–39, 1991. 4

8. M. Atkinson, P. Kunszt, I. Narang, N. W. Paton, D. Pearson, and P. Watson. Database Access and Integration. In *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 2003. A preprint is available at `http://www.ogsadai.org.uk/docs/docs.php`. 13

9. AVS5 website. `http://help.avs.com/AVS5/`. 15

10. AVS/Express website. `http://www.avs.com/software/soft_t/avsxps.html`. 15

11. D. Bergeron. Visualization Reference Models (panel session position statement). In G. M. Nielson and D. Bergeron, editors, *Proceedings of IEEE Visualization '93*. IEEE Computer Society Press, 1993. 2, 8

12. E. W. Bethel and J. Shalf. Grid-Distributed Visualizations Using Connectionless Protocols. *IEEE Computer Graphics and Applications*, 23(2):51–59, March/April 2003. 13

13. G. Bishop and G. Welch. Working in the office of 'real soon now'. *IEEE Computer Graphics and Applications*, 20(4):76–78, July/August 2000. 14

14. S. A. Bly, S. R. Harrison, and S. Irwin. MediaSpaces: Bringing People Together in a Video, Audio and Computing Environment. *Communications of the ACM*, 36(1):28–47, 1993. 4

15. D. R. S. Boyd, J. R. Gallop, and J. P. R. B. Walton. Putting You In The Picture: Enhancing Visualization With A Virtual Environment. In *IEEE Visualization '99—Late Breaking Hot Topics*, 1999. Available from `http://www.nag.co.uk/doc/TechRep/PDF/tr1_03.pdf`. 14

16. J.D. Brederson, M. Ikits, C. R. Johnson, and C. D. Hansen. The Visual Haptic Workbench. In *The Fifth PHANToM Users Group Workshop (PUG 00)*, pages 46–49, 2000. 14

17. K. W. Brodlie. Visualization over the World Wide Web. In *Proceedings of the 1997 Scientific Visualization Conference (Dagstuhl '97)*, pages 23–29. IEEE Computer Society Press, 2000. 6

18. K. W. Brodlie, J. D. Wood, D. R. S. Boyd, L. Sastry, J. R. Gallop, C. Osland, and S. E. Bunn. Collaborative Visualisation Using Access Grid. `http://www.comp.leeds.ac.uk/kwb/all_hands/BOF.pdf`, 2002. 14, 19

19. K. W. Brodlie, J. D. Wood, D. A. Duce, J. R. Gallop, D. Gavaghan, M. Giles, S. J. Hague, J. P. R. B. Walton, M. Rudgyard, B. Collins, J. Ibbotson, and A. Knox. XML for Visualization. In *Proceedings of the EuroWeb 2002 conference*. British Computer Society, Electronic Workshops in Computing(eWiC), 2002. 13

20. Cactus project website. `http://www.cactuscode.org/`. 13, 15

21. C. Carlsson and O. Hagsand. DIVE–A Multi User Virtual Reality System. In *Proceedings of VRAIS '93*. IEEE Computer Society Press, September 1993. 14

22. C. Carlsson and O. Hagsand. DIVE–A Platform for Multi-User Virtual Environments. *Computers and Graphics*, 17(6):663–669, 1993. 14

23. cAVS project website. `http://www.tacc.utexas.edu/cavs/`. 15

24. Chromium source. `http://sourceforge.net/projects/chromium/`. 19

25. climate*prediction*.net website. `http://climateprediction.net/`. 22

26. B. M. Collins. Data Visualisation–Has it all been seen before? In R. A. Earnshaw and D. Watson, editors, *Animation And Scientific Visualization: Tools & Applications*, pages 3–28. Academic Press, 1993. 1

27. COVISE and VR. `http://www.hlrs.de/structure/organisation/vis/covise/features/cover/`. 14

28. COVISE website. `http://www.hlrs.de/structure/organisation/vis/covise/`. 14

29. C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surrond-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 135–142. ACM Press, 1993. 14

30. D. de St. Germain, J. McCorquodale, S. Parker, and C. R. Johnson. Uintah: A massively parallel problem solving environment. In *Ninth IEEE International Symposium on High Performance and Distributed Computing*, 2000. 18

31. DIVE website. `http://www.sics.se/dce/dive/online/diveinfo.html`. 14

32. D. A. Duce, J. R. Gallop, I. J. Johnson, K. Robinson, C. D. Seelig, and C. S. Cooper. Distributed Cooperative Visualization - The MANICORAL Approach. In *Eurographics UK Chapter Conference 1998*. University of Leeds, 1998. 12, 15

33. D. A. Duce, D. Giorgetti, C. S. Cooper, J. R. Gallop, I. J. Johnson, K. Robinson, and C. D. Seelig. Reference Models for Distributed Cooperative Visualization. *Computer Graphics Forum*, 17(4):219–233, 1998. 10

34. K. Engel and T. Ertl. Texture-based Volume Visualization for Multiple Users on the World Wide Web. In M. Gervaut, D. Schmalstieg, and A. Hildebrand, editors, *Virtual Environments '99. Proceedings of the Eurographics Workshop in Vienna, Austria*, pages 115–124, 1999. 20

35. K. Engel, P. Hastreiter, B. Tomandi, K. Eberhardt, and T. Ertl. Medical Volume Rendering over the WWW. In T. Ertl, B. Hamann, and A. Varshney, editors, *Proceedings of IEEE Visualization 2000*, pages 449–452. IEEE Computer Society, 2000. 20

36. K. Engel, R. Westermann, and T. Ertl. Isosurface Extraction Techniques for Web-based Volume Visualization. In D. Ebert, M. Gross, and B. Hamann, editors, *Proceedings of IEEE Visualization '99*, pages 139–146. IEEE Computer Society, 1999. 21

37. Ensight Gold—Collaboration. `http://www.ceintl.com/products/collab.html`. 16

38. Ensight website. `http://www.ceintl.com/`. 16

39. Falstad website. `http://www.falstad.com/mathphysics.html`. 21

40. FAST website. `http://www.nas.nasa.gov/Software/FAST/`. 16

41. FASTexpeditions website. `http://www.nas.nasa.gov/Software/FAST/FASTexpeditions/home.html`. 16

42. I. Foster. The Grid: computing without bounds. *Scientific American*, April:62 – 67, 2003. 3

43. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002*. Available at `http://www.globus.org/research/papers/ogsa.pdf`. 22

44. D. Foulser. IRIS Explorer: A Framework for Investigation. *Computer Graphics*, 29(2):13–16, 1995. 3, 17

45. H. Fuchs. Building Telepresence Systems: Translating Science Fiction Ideas into Reality. *Computer Graphics Forum*, 16(3):C–189, 1997. 4

46. T. Funkhouser and K. Li. Large-format displays. *IEEE Computer Graphics and Applications*, 20(4):20–21, July/August 2000. 14

47. J. R Gallop. *Virtual Reality–Its Application to Scientific Visualization*. Eurographics Tutorial Notes, EG96 TN3. Eurographics Association, 1996. ISSN 1017-4656. 14

48. GAPtk project website. `http://www.e-science.clrc.ac.uk/web/projects/gaptk`. 22

49. Globus website. `http://www.globus.org/`. 15, 22

50. GODIVA project website. `http://www.e-science.clrc.ac.uk/web/projects/godiva`. 22

51. S. Greenberg, S. Hayne, and R. Rada. *Groupware for Real-time Drawing: A Designer's Guide*. McGraw-Hill, 1995. 2

52. gViz project website. `http://www.visualization.leeds.ac.uk/gViz/`. 13, 22

53. R. B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In B. Shriver, G. M. Neilson, and L. J. Rosenblum, editors, *Visualization In Scientific Computing*, pages 74–93. IEEE Computer Society Press, 1990. 10, 18

54. O. Hendin, N. John, and O. Shochet. Medical Volume Rendering Over the WWW. In J. Westwood, editor, *Proceedings of Medicine Meets Virtual Reality 1997*. IOS Press, 1997. 20

55. I. Herman and D. Duke. Minimal Graphics. *IEEE Computer Graphics and Applications*, 21(6):18–21, November/December 2001. 14

56. R. D. Hersch, B. Gennart, O. Figueiredo, M. Mazzariol, J. Tarraga, S. Vetsch, V. Messerli, R. Welz, and L. Bidaut. The Visible Human Slice Web Server: a First Assessment. In *Proceedings of IS and T/SPIE Conference on Internet Imaging, SPIE Vol 3964*, pages 253–258, 2000. 21

57. W. Hibbard, C. R. Dyer, and B. E. Paul. Display of scientific data structures for algorithm visualization. In A. E. Kaufman and G. M. Neilson, editors, *Proceedings of Visualization '92*, pages 139–146. IEEE Computer Society Press, 1992. 18

58. G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. Kirchner, and J. T. Klosowski. Chromium: A Stream Processing Framework for Interactive Rendering on Clusters. In *Proceedings of SIGGRAPH 2002*, pages 693–702. ACM Press, 2002. 19

59. ICENI and RealityGrid. `http://www.lesc.ic.ac.uk/projects/reality.html`. 23

60. ICENI project website. `http://www.lesc.ic.ac.uk/iceni/index.html`. 22

61. IDL website. http://www.rsinc.com/idl/. 16

62. inSORS website. http://www.insors.com/. 2

63. S. Jacobs, M. Gebhardt, S. Kethers, and W. Rzasa. Filling HTML forms simultaneously: CoWeb—architecture and functionality. *Computer Networks and ISDN Systems*, 28(7-11):1385–1395, 1996. 4

64. N. W. John. High Performance Visualization in a Hospital Operating Theatre. In *Theory and Practice of Computer Graphics (TPCG03)*, 2003. 19

65. C.R. Johnson, S. Parker, D. Weinstein, and S. Heffernan. Component-Based Problem Solving Environments for Large-Scale Scientific Computing. *Journal on Concurrency and Computation: Practice and Experience*, (14):1337–1349, 2002. 18

66. G. Johnson. Collaborative Visualization 101. *Computer Graphics*, 32(2):8–11, 1998. 15

67. JWAVE website. http://www.vni.com/products/wpd/jwave/. 17

68. JXTA website. http://www.jxta.org. 19

69. A. Kee. Visualization over WWW using Java. Master's thesis, School of Computer Studies, University of Leeds, U.K., 1996. 21

70. W. Lefer. A Distributed Architecture for a Web-based Visualization Service. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing*. Eurographics Association, 1998. 20

71. T. Liao. WebCanal: a Multicast Web Application. In *Proceedings of the Sixth International World Wide Web Conference*, 1997. 4

72. H. D. Lord. Improving the Application Development Process with Modular Visualization Environments. *Computer Graphics*, 29(2):10–12, 1995. 3

73. T. Mayer. New Options and Considerations for Creating Enhanced Viewing Experiences. *Computer Graphics*, 31(2):32–37, 1997. 13

74. R. F. McCloy and N. W. John. Remote Visualization of Patient Data in the Operating Theatre during Helpatopancreatic Surgery. In *CARS 2003 Computer Assisted Radiology and Surgery, London, UK*, 2003. 19

75. B. H. McCormick, T. A. DeFanti, and M. D. Brown. Visualization in Scientific Computing. *Computer Graphics*, 21(6), 1987. 1

76. C. K. Michaels and M. J. Bailey. VizWiz: A java applet for interactive 3d scientific visualization over the web. In R. Yagel and H. Hagen, editors, *Proceedings of IEEE Visualization '97*, pages 261–268. IEEE Computer Society Press, 1997. 21

77. J. Mortensen, V. Vinayagamoorthy, M. Slater, A. Steed, B. Lok, and M. C. Whitton. Collaboration in Tele-Immersive Environments. In *Eighth Eurographics Workshop on Virtual Environments*, pages 93–101, May 2002. 14

78. MSN Messenger website. http://messenger.microsoft.com. 19

79. OpenDX weather applications. http://www.research.ibm.com/weather/vis/w_vis.htm. 17, 20

80. OpenDX website. http://www.opendx.org/. 3, 17

81. H.-G. Pagendarm. Visualization Within Environments Supporting Human Communication. *Future Generation Computer Systems*, 15:109–117, 1999. 20

82. H.-G. Pagendarm and F. H. Post. Comparative Visualization—Approaches and Examples. In M. Göbel, Müller H., and B. Urban, editors, *Visualization in Scientific Computing*. Springer-Verlag, Wien, 1995. 2

83. A. Pang and K. Smith. Spray rendering: Visualization using smart particles. In G. M. Nielson and D. Bergeron, editors, *Proceedings of IEEE Visualization '93*, pages 302–309. IEEE Computer Society press, 1993. 16

84. A. Pang, C. K. Wittenbrink, and T. Goodman. CSpray: A Collaborative Scientific Visualization Application. In *Proceedings of Multimedia Computing and Networking*, 1995. 16

85. A. T. Pang and C. M. Wittenbrink. Collaborative 3D visualization with CSpray. *IEEE Computer Graphics and Applications*, 17(2):32–41, 1997. 12

86. S. G. Parker and C. R. Johnson. SCIRun: A scientific programming environment for computational steering. In H. W. Meuer, editor, *Proceedings of Supercomputer '95*, New York, 1995. Springer-Verlag. 17

87. P. Parnes, M. Mattsson, K. Synnes, and D. Schefstrom. The mWeb Presentation Framework. In *Proceedings of the Sixth International World Wide Web Conference*, 1997. 4

88. RealityGrid project website. http://www.realitygrid.org. 13, 23

89. PV-Wave website. http://www.vni.com/products/wave/. 16

90. pV3 website. http://raphael.mit.edu/pV3/pV3.html. 17

91. RealVNC website. http://www.realvnc.com. 19

92. B. E. Rogowitz. The Psychology of Visualization (panel session position statement). In G. M. Nielson

and D. Bergeron, editors, *Proceedings of Visualization '93*. IEEE Computer Society Press, 1993. 1

93. E. Saxon, Z. Wood, M. O'Neil, C. Oates, J. Story, S. Djurcilov, and A. Pang. Integrated Visualization of Realtime Environmental Data. In *Proceedings of Spring Conference on Computer Graphics*, pages 135–143, Comenius University, Bratislava, Slovakia, 1997. 16

94. W. J. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*. Kitware, Inc., 3rd edition, 2003. 18, 20

95. SCIRun and VNC. `http://web.gat.com/pubs-ext/MISCONF02/A23983.pdf`. 18

96. A. Scott and H. Wolf. Collaborative Browsing in the World Wide Web. In *Proceedings of the 8th Joint European Networking Conference*, 1997. 4

97. T. Sgouros. *DODS User Guide, Version 1.11*. University of Rhode Island, Graduate School of Oceanography, 2003. Available at `http://www.unidata.ucar.edu/packages/dods/`. 8

98. J. Shalf and E. W. Bethel. The Grid and Future Visualization System Architectures. *IEEE Computer Graphics and Applications*, 23(2):6–9, March/April 2003. 3, 23

99. SOAP details. `http://www.w3.org/TR/SOAP/`. 22

100. Space Physics and Astronony Research Collabatory website. `http://intel.si.umich.edu/sparc/`. 9

101. D. A. Stainforth, D. Frame, and J. P. R. B. Walton. Visualization For Public-Resource Climate Modeling Research. In *IEEE Visualization 2003*, 2003. submitted. 22

102. J. Stanton, S. Newhouse, and J. Darlington. Implementing a Scientific Visualisation Capability Within a Grid Enabled Component Framework. In *Proceedings of 8th International Euro-Par Conference, volume 2400 of Lecture Notes in Computer Science, Paderborn, Germany*, 2002. Available at `http://www.lesc.ic.ac.uk/iceni/pdf/europar2002.pdf`. 23

103. P. Stanton. Java Visualization Software. Final year project, School of Computer Studies, University of Leeds, U.K., 1997. 21

104. S. Stegmaier, M. Magallon, and T. Ertl. A Generic Solution for Hardware-Accelerated Remote Visualization. In D. Ebert, P. Brunet, and I. Navazo, editors, *Joint EUROGRAPHICS-IEEE TCVG Symposium on Visualization (2002)*, pages 87–94. Eurographics, 2002. 14, 19

105. Studierstube website. `http://www.cg.tuwien.ac.at/research/vr/studierstube/AVS/html/`. 14

106. I. Taylor, M. Shields, and R. Philp. GridOneD: Peer to Peer visualization using Triana: A Galaxy Formation Test Case. In *Proceedings of UK e-Science All Hands Meeting, September 2002*, 2002. 19

107. The UK e-Science website. `http://www.escience-grid.org.uk/index.htm`. 22

108. M. Thorson, J. Leigh, G. Maajid, K. Park, A. Nayak, P. Salva, and S. Berry. AccessGrid-to-Go: Providing AccessGrid access on Personal Digital Assistants. In *Proceedings of the Access Grid Retreat, 2002*. Available at `http://www.evl.uic.edu/paper/pdf/AG2GoFinal2002.PDF`, 2002. 14

109. TightVNC website. `http://www.tightvnc.com`. 19

110. J. Trapp and H.-G. Pagendarm. A prototype for a WWW-based visualization service. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 21–30. Springer, Wien, 1997. 20

111. C. Upson, T. Faulhaber, D. Kamins, D. Schlegel, D. Laidlaw, J. Vroom, R. Gurwitz, and A. van Dam. The Application Visualization System: a Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, 1989. 10, 15

112. S. P. Uselton. Case study: The 'Office of Real Soon Now' for Visualization. In H. Pfister and M. Bailey, editors, *Proceedings of Visualization 2002*. IEEE Computer Society Press, 2002. 14

113. Using Vizserver for remote medical visualization. `http://www.sgi.com/features/2002/sep/manchester/`. 19

114. Vestac project website. `http://www.kuleuven.ac.be/ucs/java/`. 21

115. Vircinity company website. `http://www.vircinity.com/`. 16

116. Vis-5D website. `http://www.ssec.wisc.edu/~billh/view5d.html`. 21

117. Vis5d source. `http://vis5d.sourceforge.net/`. 21

118. VisAD collaborations. `http://www.ssec.wisc.edu/~dglo/visad-collab/`. 18

119. VisAD events. `http://www.ssec.wisc.edu/~dglo/visad-events/`. 18

120. VisAD website. http://www.ssec.wisc.edu/~billh/visad.html. 18

121. Visible Human visualization. http://visiblehuman.epfl.ch. 21

122. VistaPortal website. http://www.vistaportal.com. 19

123. Visual3 website. http://raphael.mit.edu/visual3/visual3.html. 17

124. http://www.tessella.co.uk/projects/vivre/index.htm. 14

125. Vizserver website. http://www.sgi.com/software/vizserver/. 19

126. VTK—Multi-pipe rendering. http://brighton.ncsa.uiuc.edu/~prajlich/vtkActorToPF/. 18

127. VTK and CAVEs. http://www.evl.uic.edu/cavern/cavernpapers/viz98/. 18

128. VTK website. http://public.kitware.com/VTK/. 18

129. J. P. R. B. Walton. Get The Picture–New Directions In Data Visualization. In R. A. Earnshaw and D. Watson, editors, *Animation And Scientific Visualization: Tools & Applications*, pages 29–36. Academic Press, 1993. 16

130. J. P. R. B. Walton. World processing: data sharing with VRML. In R. A. Earnshaw and J. A. Vince, editors, *The Internet in 3D: Information, Images and Interaction*. Academic Press, 1997. 20

131. J. P. R. B. Walton. NAG's IRIS Explorer. In C. R. Johnson and C. D. Hansen, editors, *Visualization Handbook*. Academic Press, 2003, in press. Available from http://www.nag.co.uk/doc/TechRep/Pdf/tr2_03.pdf. 3, 17

132. J. P. R. B. Walton and D. Knight. Rock 'n' Roll: Using VRML2.0 for Visualization. IRIS Explorer Technical Report TR27 IETR (NP3159), NAG Ltd, 1997. Available from http://www.nag.co.uk/doc/TechRep/HTML/tr2_97.html. 8

133. R. Weggenkittl and E. Groeller. Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet. In R. Yagel and H. Hagen, editors, *Proceedings of IEEE Visualization '97*, pages 309–316. IEEE Computer Society Press, 1997. 21

134. G. Welch, H. Fuchs, R. Raskar, H. Towles, and M. S. Brown. Projected imagery in your 'office of the future'. *IEEE Computer Graphics and Applications*, 20(4):62–67, July/August 2000. 14

135. J. Wernecke. *The Inventor Mentor. Programming Object-Oriented Graphics with Open Inventor, Release 2*. Addison-Wesley, 1994. 15, 22

136. A. Wierse, U. Lang, and R. Rühle. Architectures of Distributed Visualization Systems and their Enhancements. presented at 4th Eurographics Workshop on Visualization in Scientific Computing, Abingdon, U.K., 1993. 15

137. J. D. Wood. *Collaborative Visualization*. PhD thesis, School of Computer Studies, University of Leeds, U.K., 1998. 12

138. J. D. Wood, K. W. Brodlie, and H. Wright. Visualization Over The World Wide Web and its Application to Environmental Data. In R. Yagel and G. M. Nielson, editors, *Proceedings of IEEE Visualization '96*, pages 81–86. IEEE Computer Society Press, 1996. 20

139. J. D. Wood, H. Wright, and K. W. Brodlie. Collaborative Visualization. In R. Yagel and H. Hagen, editors, *Proceedings of IEEE Visualization '97*, pages 253–259, 1997. 10, 12, 17, 20

140. A. Yeo. Client-based Web Visualization. Master's thesis, School of Computer Studies, University of Leeds, U.K., 1998. 9, 21

141. M. Young, D. Argiro, and J. Worley. An Object Oriented Visual Programming Language Toolkit. *Computer Graphics*, 29(2):25–28, 1995. 3

142. H. Zhou, M. Chen, and M. F. Webster. Comparative evaluation of visualization and experimental results using image comparison metrics. In H. Pfister and M. Bailey, editors, *Proceedings of IEEE Visualization 2002*. IEEE Computer Society Press, 2002. 2