

University of Leeds
SCHOOL OF COMPUTER STUDIES
RESEARCH REPORT SERIES

Report 95.17

Improving Visualisation Through Collaboration

by

Jason Wood, Helen Wright & Ken Brodlie
Division of Computer Science

May 1995

Presented at the Sixth Eurographics Workshop on Visualisation in Scientific
Computing, Chia, Italy, 3-5 May 1995

Abstract

Scientific visualization has become an essential tool in helping the researcher to understand the vast quantities of data which arise in computational science and engineering. Yet visualization to date has been a solitary occupation, with little support for group working. This paper describes work to provide ‘added value’ to IRIS Explorer in the form of new modules to facilitate data sharing and explains their use in conjunction with video and audio conferencing tools.

1 Introduction

This paper will describe work carried out as part of the COVISA project, currently being undertaken at the University of Leeds. COVISA (Co-Operative working in VISualization and Scientific Analysis) is a study funded by the Engineering and Physical Sciences Research Council, with the aim of understanding requirements for group working in visualization. Key themes of the work are its emphasis on practical experience through case studies and the feeding back of requirements to industry.

Computer supported co-operative working (CSCW) spans many activities. In [1] it is referred to as ‘people working together on a product, research area, topic or scholarly endeavour with help from computers’. Scientific visualization has to date been largely a solitary process - there is little support in existing visualization systems for shared working. There could be numerous benefits from group working in visualization, for example: two scientists with different, complementary skills could work on the same problem; a large data set can be shared amongst a number of researchers; or a lecturer could use a visualization system to demonstrate some feature to a class of students, perhaps allowing them individual, controlled access to a data set.

Pagendarm and Walter [2] and Gerald-Yamasaki [3] have approached the problem in the field of Computational Fluid Dynamics, providing the capability for several workers to view the results of a simulation. Pagendarm and Walter have broadened this scenario to allow multiple simulations and have also studied the case of synchronised dual control, where each investigator in turn can steer the visualization. However, in our view these two modes of working represent two extremes of collaboration. Although it undoubtedly is useful to see what a colleague can see, or to witness a co-worker's manipulation of the visualization process, there are occasions when only certain elements of data or control are to be passed amongst the group. For example, two workers might share intermediate data, employ subsequent different visualization idioms and then compare results visually.

Other work in this area by Wierse, Lang and Rühle [4] has resulted in a complete collaborative visualization system built from scratch, though with the large investment in user time required to learn a particular visualization package we feel that there are advantages in extending commercially available products to include collaborative elements. This of course is balanced by the more limited CSCW facilities which we can impart compared with writing a whole new system.

Taking our two aims together, the approach we describe in the present paper has therefore been to provide a toolkit for IRIS Explorer [5], such that different aspects of the visualization process can be shared (or not) at will, within an existing system.

2 An illustration

As an illustration of the tools we might need, let us take the scenario of two people with different skills working in collaboration. The first one, who we will call the *scientist*, is attempting to visualize a data set of temperature of airflow over an aircraft wing. He has a general knowledge of visualization techniques and is looking at isosurfaces but is using a graphics package with which he is unfamiliar. The other, who we will call the *visualization expert*, is a computer scientist, maybe working in a support role, who is familiar with the program being used. She first became involved when she set up the visualization for the scientist, using data which he emailed to her. Not long afterwards the scientist contacts the visualization expert over a videophone to explain that he has a problem.

The scientist, instead of getting a smooth isosurface as he had expected, got a strange shape stretching at right angles to the wing. The visualization expert attempts to re-create the problem – she still has a copy of his data so she loads it into the visualization program and starts both a local and a remote display device, so that they can look at the same output. The isosurface looks normal - not at all what the scientist saw. Using a pointer device which displays on both outputs, he tries to describe where the sheet came from. The expert doesn't understand and shakes her head! Seeing this on the videophone, the scientist abandons that approach and sketches what he saw on a shared whiteboard. The expert begins to realise what might be the difficulty: that this effect is just due to a particular threshold value generating an open, rather than closed, surface. She suggests that he investigate the isosurfaces around that value while they are in contact, but unfortunately he doesn't have a copy of the program running. To avoid unnecessary delay he opens a link and sends his values of the threshold parameter to her copy. He has some difficulty reproducing the effect but by initiating a two-way connection they alter the values collaboratively.

After seeing a series of isosurfaces the scientist realises that an animation or movie of a sequence would be helpful to his understanding of the physics involved. The expert plays back just such a movie developed for a different client, sending the sequence of images to the scientist's workstation. Although it is for a different problem, he is sufficiently convinced to commission a movie of his own.

3 Toolkit approach

The scenario described above is a hypothetical one, but it serves to crystallise some of the issues surrounding group working in general and visualization in particular.

Our first assertion is that it shows it is worthwhile: had the scientist and expert simply corresponded or telephoned, the problem would have been solved less efficiently. Interaction, both verbal and visual, contributed significantly to the assimilation of the situation. More importantly, the interaction precipitated a totally new idea for the scientist: that of making a movie of isosurfaces.

Given that it is worthwhile, the next issue is: How feasible is it to provide the necessary tools? Some will be familiar already - email is ubiquitous and the videophone is an obvious development of the telephone - but how can we support collaboration to explore a data set visually?

We have approached this challenge by considering the nature of the data being shared and the different levels of interaction this imparts. Our practical studies have centred on application builder visualization systems, in particular IRIS Explorer, since the architecture of these is open, giving us access to the data which is flowing. IRIS Explorer already has a comprehensive data model and process management facilities, so by developing new IRIS Explorer modules which share data items across machines, we have laid the foundations of a toolkit for group working in visualization. Our goal throughout has been to build on what is already provided in order to create a combined tool of general utility.

The following sections explain the visualization system in more detail, describe the tools which we have developed so far and indicate how we shall extend the toolkit subsequently.

3.1 IRIS Explorer

IRIS Explorer is a visualization tool from the ‘application builder’ family, originally produced by Silicon Graphics Inc. (SGI) and now distributed by NAG Ltd. An application builder follows the data flow paradigm and provides the user with a set of modules to carry out constituent operations of the visualization pipeline [6] [7]. The user connects modules together to make a map, the connections between modules forming the path along which the data will flow. Data enters the map on the left, gradually being altered as it flows through, emerging on the right in either a 2D (image) or 3D (object) form for viewing.

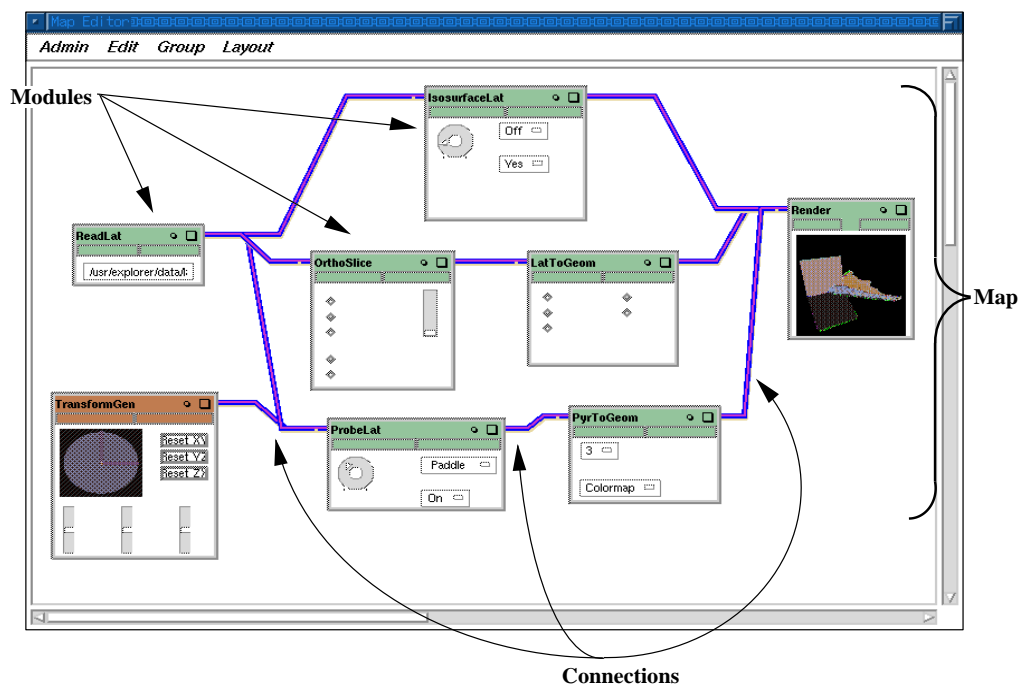


Figure 1 IRIS Explorer Map Editor

Map building is carried out by means of a visual programming tool called the Map Editor (Figure 1), with individual modules represented graphically as glyphs. Each module has widgets which allow the user to set parameters governing its operation, e.g. IsoSurfaceLat has a dial widget to set the threshold value for the surface. If a required operation cannot be accomplished by a standard module, the user can implement their own algorithm as a new module to add to the system.

3.2 Display sharing

Modules are available to view the processed data locally; DisplayImg for images and Render for objects. There is no standard module to allow the sharing of images, but this is achieved using new tools described in 3.3. Remote viewing of objects however is possible using the standard RenderRemote module which opens a render window on a specified remote display. IRIS Explorer's Geometry datatype is passed describing the objects in the scene which are rendered and displayed by the remote machine. The remote user is free to manipulate the object in the render window independently of the local user.

This means two or more people see the end results of the visualization process and can discuss them by means of other tools. A problem arises, however, when users are free to manipulate their views independently, since this can lead to collaborators talking about different objects or features in the scene. A user's view of the scene in Render or RenderRemote is controlled by the position of a camera, which in turn is passed out of Render as the Geometry datatype, so one way round this problem is to pass cameras between the two renderers. A second approach is the use of a shared pointer. The render windows accept input from the mouse and return details about the point on an object that was selected. This is fed to a new upstream module, Pointer, which generates an arrow shaped object and places it at the selected point in all render windows. This ensures that users are talking about the same object or feature. All collaborators are free to move the position of the pointer.

Further interaction between the user and the object in the renderer takes place within the scene graph which is controlled by Open Inventor [8]. Consequently, changes to material properties, lights, object transforms and so on remain internal to Render and are not propagated to the IRIS Explorer interface in the form of the Geometry datatype. This makes full collaborative control of the renderer unsuited to the data sharing tools we currently have available. We are considering a more integrated method for synchronising views which will extend the existing Render module to share changes directly with other copies of itself. Alterations to object parameters and views will be distributed either by a central server or a number of distributed servers in concert and users will be able to decouple the shared controls to allow periods of free investigation.

3.3 Control and data sharing

The tools described in 3.2 give us a means to share the end result of the visualization process between collaborators, but not the control of the visualization process itself. Since in IRIS Explorer the individual modules are controlled by widgets which in turn deliver scalar values, the issue of collaborative control merges with the general issue of data sharing.

In IRIS Explorer a process called the Global Controller (GC) is responsible for managing connections between modules within a single Map Editor; however, connections to modules from a second Map Editor are beyond its scope. One method of allowing inter-map connections would be for a sharer module to require a specific port number so that connections could be made to a known location. The disadvantage of this is that only a single copy of the module could be run, as the specific port would then be in use. Also there would be no control over which, of possibly many, remote connections could be made. Our solution to these problems has been to write a module, called PortServer, to handle connections between modules in separate Map Editors. This means that only a single reserved port number is required so that modules know where to connect initially. Only one PortServer is required

between collaborators - the machine containing it is defined to be the 'local' machine and all others connecting to it to be 'remote' machines. All machines have an environment variable set to point to the 'local' machine.

The PortServer accepts connections from 'remote' modules and extracts an address, consisting of port number and machine name, and a description of intended use; the remote modules then wait to be connected. A 'local' version of the same module connects to the PortServer and copies the entire list of remote connections which it presents to the 'local' user. One connection is then selected from the list by the 'local' user and this choice is deleted from the list held by the PortServer. The 'local' module uses the address to connect directly to the chosen 'remote' module by means of a UNIX socket [9], so the pair are now no longer concerned with the PortServer. Figure 2 shows a step by step illustration of the process.

We have explored two ways of sharing IRIS Explorer's datatypes: the first relies on an explicit knowledge of the internal structure of the data whilst the second uses the automatically generated interface routines for reading and writing types. We addressed ourselves in the first case to the Parameter and Lattice datatypes since these are widely used in IRIS Explorer and represent the minimum requirement for sharing data and control. They are also the two simplest types – essential in the first method since we have to unpack and repack the type within the module code:

- ShareParam - Pairs of these modules, once connected by means of the PortServer as described above, send IRIS Explorer's Parameter datatype between them via a UNIX socket. This enables widgets on module control panels in separate maps to be connected together giving both users control over visualization parameters, such as the threshold value for an isosurface.
- ShareLat - A pair of these modules are able to send any variant of IRIS Explorer's Lattice datatype between them, and hence between separate maps. The Lattice is the most commonly used datatype and encodes data of many forms. For example an image is stored as a 2D lattice, hence a movie sequence can be shared using this module.

Both of these modules allow bi-directional data passing, but when connected in this way steps are taken to prevent infinite cycling of data. This method of sharing data is ideal in the early stages of toolkit development since all the elements of the type can be tracked as they are transferred. The drawbacks are in the time taken to develop the code and sensitivity to future changes in the internal structure of the type.

To address these problems we have also investigated the use of IRIS Explorer's api (application programmer interface) routines to read and write each type transparently. This method is independent both of the complexity of a datatype and of future changes it might undergo, since the api always keeps pace with updates to a type. It also allows 'Share' modules for user-added datatypes to be easily constructed since the api routines are automatically generated when the new type is built. These api routines are normally used in conjunction with disk files and some technical problems remain to be resolved when they are applied to socket connections. However, the method shows promise and modules to allow uni-directional sharing of the three remaining datatypes were quickly developed in this way:

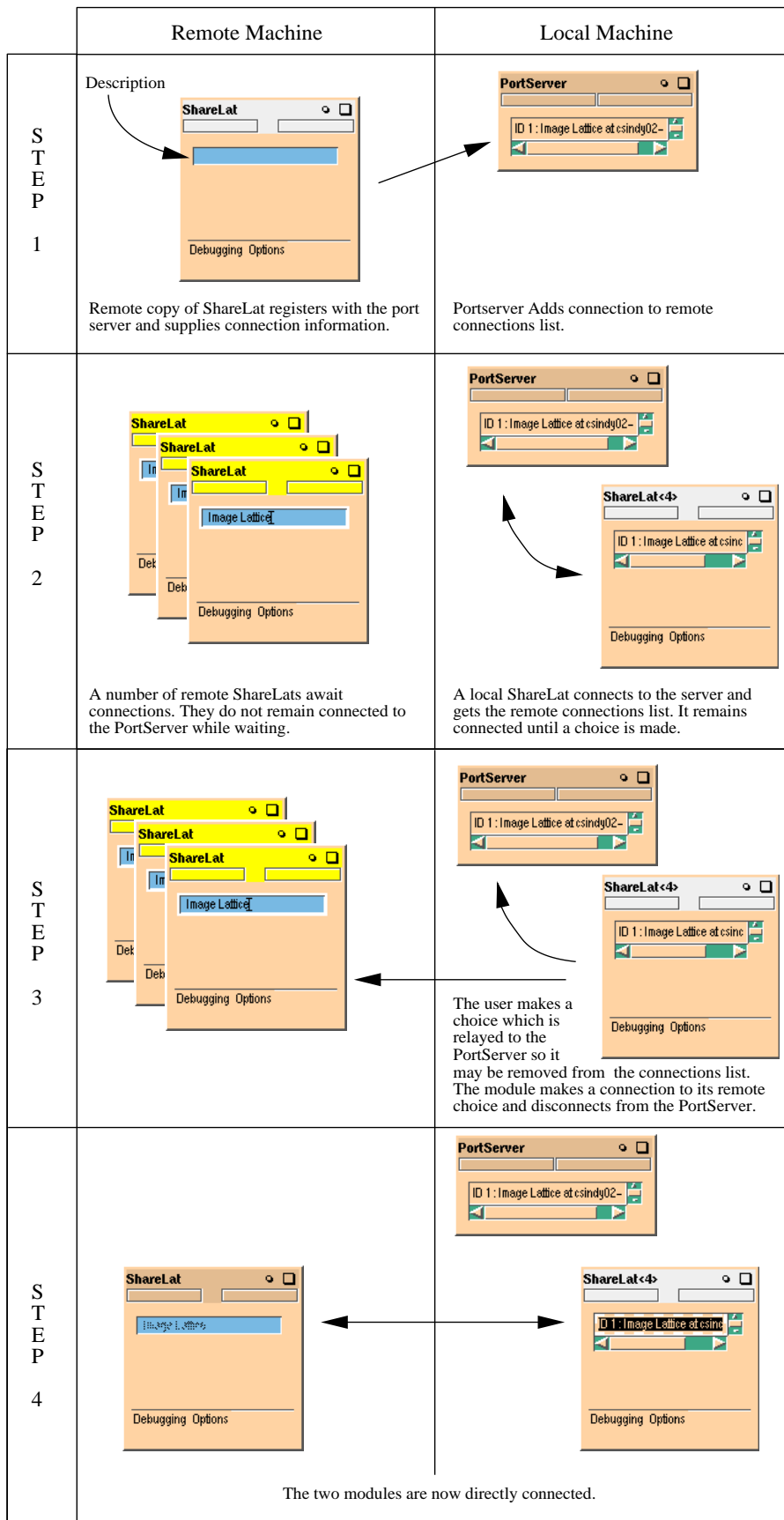


Figure 2 PortServer Operation

- SharePyr, SharePick, ShareGeom - Allow sharing of IRIS Explorer's Pyramid, Pick and Geometry datatypes respectively. The Pyramid datatype is often used for Finite Element data and for chemistry data. Pick is used to feed back information about selected points in the renderer. The Geometry datatype is used to describe the objects viewed in Render and to pass camera positions between renderers.

Although instances of the same sharer module are run on both the 'local' and 'remote' machines, they autoconfigure their control panels (Figure 2) to accept information relevant to their method of connection to the PortServer. Thus the 'remote' module requires a text string to distinguish it from other 'remote' connections and hence has a text type-in widget, while the 'local' version displays a scrolling list widget which displays all of the remote connections from which the local user makes a selection. If the environment variable containing the machine name of the 'local' machine is missing, the modules open a text slot and request the user to supply it.

3.4 Conferencing tools

To complement the collaboration system certain conferencing tools are required. To date we have used

- Videoconferencing - We are using a composite tool comprising audio connection via the INRIA IVS Videoconferencing System and video connection via an 'in house' tool written as part of the Virtual Science Park project [10].
- A shared whiteboard is provided by WSCRAWL.

All of these tools run over Ethernet and our trial ATM network.

4 A test case

To test the tools in practice an experiment was set up between two scientists at the University of Leeds. One scientist works in the field of fuel engineering and makes only limited use of visualization as a tool; the other works in the School of Computer Studies on a project to produce scientific simulation software. The simulation scientist also has good knowledge of IRIS Explorer as a visualization system. Normally these scientists would have a meeting to visualize and discuss the results from the simulator, with the fuel engineer using her expertise to check the validity of the output. The experiment required both scientists to stay in their own offices and connect together by means of the videophone, using the tools we have described to evaluate their results. Each was shadowed by a member of the COVISA team who observed and assisted when necessary.

Our first observation of this experiment is the importance of establishing initial video and audio links transparently. On this occasion there was a difficulty in setting up a connection which led at first to scepticism on the part of the users, however, once this was resolved the potential for this way of working was accepted. Some prompting was also required to select the best tool for a particular job, for example, to use ShareLat to send a complete colour map rather than attempting to share the many parameters on the GenerateColormap module. The Pointer module proved useful in identifying object features to the other participant, though the other limitations already noted in sharing object displays were observed. Undoubtedly a fully sharable Render module such as we envisage in 3.2 would be of value. Nonetheless,

collaboration was achieved without the participants having to meet and some of the criticisms of the tools will diminish as familiarity with them grows.

Perhaps the most serious deficiency noted was an inability to build maps collaboratively. Thus the simulation scientist had to build the same map as that being used by the fuel engineer by means of directions given over the videophone. Similarly, when one participant made an alteration to their map, precise instructions had to be passed to the other to recreate the change. This required a lot more effort to be put into visualizing the results than would normally be the case – something which was rightly perceived as a backward step by the participants.

5 Conclusions and further work

This paper has described how an existing visualization system can be extended to support group working. In particular we have shown how our extensions can support a form of collaboration in which a visualization expert advises a scientist. This is only one possible use of collaborative visualization - i.e. the support role.

The tools described here work well between two people and extend easily to greater numbers, but the system works at a very informal level. If the tools are to be used as part of a formal collaboration system then some form of ‘conference management’ tool will be needed. This would allow participants to join and leave the session, set the control policies and set participants’ permissions within a session. It would give collaborators the facilities to temporarily work alone while still being in touch with the collaboration and then rejoin at a later time and integrate the results of their individual work. It should also allow private and public sections of a map so that collaborators with different skill bases can view different results.

Our case study revealed a need for robust, easy-to-use tools, and also for a shared map-building facility. We are currently assessing the feasibility of the latter for IRIS Explorer and intend to investigate other systems which might allow this, such as Khoros2.0. We will look at tools that extend from the simpler case of visualizing a data set, to wider problem solving environments where the group can also control the simulation. To show that these tools have ‘real world’ value, we will be testing them in further case studies. Furthermore, in order to understand the complexities of group working it is helpful to try to determine a theoretical model for the processes taking place. This will lead to the development of a reference model for collaborative visualization in which the different styles of collaboration can be positioned – a so-called ‘sliding scale of collaboration’.

Visualization is already a major factor in scientific productivity, particularly in the technologically intensive industries. To date, it has been a task for a single person. This project will work towards realising the potential for group working in scientific visualization - the benefits being greater speed of analysis, simply through more human processing power, and more effective analysis through exploiting a group with a range of expertise. Indeed, there is the potential for major scientific breakthroughs which would not be possible through single person working

Acknowledgements

COVISA involves thirty staff-months effort expended over two years. The project is in its first year.

Our thanks go to all who have contributed through discussion and technical assistance, notably Alison Tomlin and Justin Ware for participating in the case study and our reviewers for a number of helpful comments. Special thanks to Richard Drew and others from the Virtual Science Park project for the video software used as part of the videoconferencing tools. IVS is produced by Thierry Turletti at INRIA and Brian Wilson wrote WSCRAWL. Grateful thanks, too, to Jeremy Walton of NAG Ltd. for writing Pointer specifically for the project and for other invaluable help with aspects of IRIS Explorer.

References

- [1] James D. Palmer and N. Ann Fields, "Computer Supported Cooperative Work", *Computer*, Volume 27(5), pp 15-17, 1994.
- [2] H.G. Pagendam and B. Walter, "A Prototype of a Cooperative Visualization Workplace for the Aerodynamicist", *Computer Graphics Forum*, Volume 12(3), EG 93 Conference Issue, pp 485-496, 1993.
- [3] M.J. Gerald-Yamasaki, "Cooperative Visualization of Computational Fluid Dynamics", *Computer Graphics Forum*, Volume 12(3), EG 93 Conference Issue, pp 497-508, 1993.
- [4] A.Wierse, U.Lang, R.Rühle, "Architectures of Distributed Visualization Systems and their Enhancements", *Eurographics Workshop on Visualization in Scientific Computing*, Abingdon 1993.
- [5] IRIS Explorer User's Guide, Silicon Graphics Inc., 1993.
- [6] Robert B. Haber and David A. McNabb, "Visualization Idioms: A Conceptual Model for Scientific Visualization Systems", *Visualization in Scientific Computing*, IEEE, pp 74-93, 1990.
- [7] Craig Upson, Thomas Faulhaber, Jr., David Kamins, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz, Andries van Dam, "The Application Visualization System: A Computational Environment for Scientific Visualization", *Computer Graphics & Applications*, IEEE, Volume 9(4), pp 30-42, 1989.
- [8] The Inventor Mentor, Addison Wesley, 1994.
- [9] IRIX Programmer's Reference Manual, Volume 1, section 2, Silicon Graphics Inc.
- [10] Peter Dew, Christine Leigh, Richard Drew, David Morris, Jayne Curson, "Virtual Working Systems to Support R & D Groups", *Multimedia Computing and Networking*, San Jose. California, Feb 5 -10 1995.