

# gViz – Visualization and Steering for the Grid

Jason Wood, Ken Brodlie  
School of Computing, University of Leeds

Jeremy Walton,  
NAG Ltd

## Abstract

This paper describes work in the e-Science Core Programme project, gViz, to develop visualization middleware. The focus in this paper is the evolution of our work to Grid-enable IRIS Explorer, a widely used dataflow visualization system from NAG. A crucial feature of IRIS Explorer is the ability to include user-written simulation code as part of the dataflow pipeline, offering the opportunity of computational steering. The evolution has gone through three phases. An early demonstrator showed how, for a specific application, IRIS Explorer could be adapted to launch a simulation on a remote Grid resource, and maintain contact with it to enable steering of control parameters and visualization of results. In the second phase, we extended IRIS Explorer to run in a distributed fashion on the Grid, with different modules in the pipeline able to run on different resources – thus the pipeline spans the Grid. In a final phase, we are developing a computational steering library that allows a flexible coupling of a simulation with a visualization system such as IRIS Explorer. This approach allows the front-end visualization system to connect and disconnect as required from a long running simulation, and takes a service-oriented approach to visualization and steering. In all of the phases we are able to exploit the COVISA collaborative working feature of IRIS Explorer, enabling teams of e-scientists to jointly steer a simulation and visualize the results.

## 1. Introduction

Visualization has become a key enabler for e-science. It allows the large quantities of data generated either by direct observation or by simulation, to be analysed visually, providing insight and understanding that would be impossible otherwise. Computational steering, in which simulation and visualization are tightly coupled, has also become an important paradigm, encouraging efficient use of expensive Grid resources.

In this paper we describe work in the UK e-Science Core Programme project, gViz, which aims to study visualization middleware for e-Science. A particular interest for this paper is how we can best extend existing visualization systems so that they can be used effectively in Grid environments.

The paper begins with a review of current visualization systems and previous work on computational steering. We describe IRIS Explorer, a visualization system from NAG Ltd, which has been used as the basis for most of our experiments. We then show how our thinking has evolved through three stages over our past two

years' involvement in e-science. First, we outline the development of a demonstrator, widely used as an early illustration of the potential of e-science; one can view this demonstrator as representing the state of the art in visualization and computational steering, at the start of the UK e-science programme. It coupled the simulation and visualization, but in a one-off, hand-crafted manner, with the simulation process running externally to the visualization system. Second, we describe how we have been able to extend IRIS Explorer to run securely across several machines, realising our aim of making it 'Grid-enabled'. In doing this work, some limitations of IRIS Explorer as a complete Grid computing environment have been identified. Therefore in a third phase, we are developing a library for computational steering. This is designed specifically to support the control and monitoring of remote simulations, from a desktop interface. This interface can be any visualization system, including indeed IRIS Explorer, but we have made a clean separation between the roles of visualization system and simulation. We conclude with a summary, and the main issues to be pursued in the future.

## 2. Review

We divide our review into four sections: the development of visualization systems; the notion of computational steering, including mention of other e-science projects which involve steering; an outline of the gViz project; and a description of a typical e-science application which we shall use as a driver for the research reported in later sections.

### 2.1 Visualization Systems

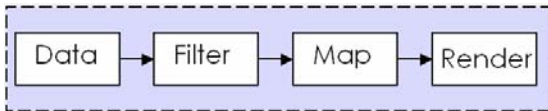


Figure 1 – Haber and McNabb Reference Model

The landmark report on Visualization in Scientific Computing by McCormick et al (1987) sparked the development of a number of interactive visualization systems, based on the dataflow paradigm. The reference model for this type of system was elegantly described by Haber and McNabb (1990a), and is shown diagrammatically in Figure 1. Raw visualization data is fed into a pipeline of processes, progressively filtering the data, converting it to an abstract geometric representation and finally rendering the geometry. This model is the basis for a number of visualization systems, including the system we shall be using here, namely IRIS Explorer from NAG (IRIS Explorer, 2003). Figure 2 shows a screenshot, where modules from a library are loaded into a visual workspace and connected into a network reflecting the Haber-McNabb model.

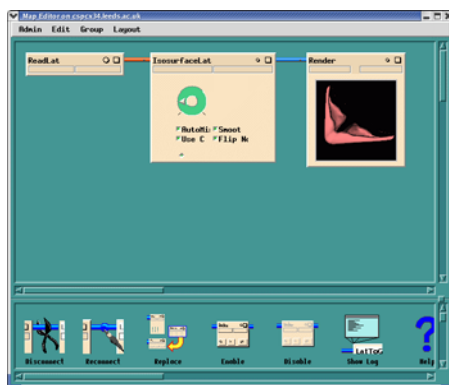


Figure 2 – Simple visualization pipeline in IRIS Explorer

An important aspect of these systems is their openness and extensibility. While many standard modules are delivered with the system, a user can encapsulate their own code as a module, add this to the library, and load into the workspace like any standard module. This extensibility has allowed these systems to evolve with advances in computing technology, and IRIS Explorer, for

example, remains widely used today even though more than a decade old. One particular extension is important in the context of e-science: we are able to create special collaborative modules that link separate pipelines being executed by users at different geographical locations. This was first achieved in the EPSRC COVISA project for IRIS Explorer (Wood et al, 1997), and is now an integral part of the system.

### 2.2 Computational Steering

Computational steering has become an attractive paradigm for e-science. By close coupling of simulation and visualization, it becomes possible to control the execution of the simulation through observation of the visualization of current output. Steering is also being actively studied in the RealityGrid e-science pilot project (RealityGrid, 2003), and in the ICENI framework being developed at the London e-Science Centre (ICENI, 2003).

Steering is not a new idea. The concept was dramatically demonstrated at the 1989 SIGGRAPH Technical Reception by Bob Haber and David McNabb (Haber, 1990b). A supercomputer simulation running at NCSA in Illinois was steered by a user at the conference in Boston. The notion of steering was well described by Marshall et al (1990), where they identify three approaches to the visualization of output from a simulation. First, a simple *post-processing* approach where the simulation is run first, and the results later visualized. An advantage is that results can be examined in depth, at the scientist's own pace, and different visualization techniques used. The second approach is *tracking*, in which images are viewed as the calculation proceeds, allowing errors to be spotted and fruitless calculations aborted. The scientist can interact with the visualization to control the view that is taken. The third approach is *steering*, where the scientist interacts with both the simulation and the visualization. The model for steering is illustrated in Figure 3, where we see the simulation as part of the pipeline. Again it is the extensibility of dataflow visualization systems that makes this possible – the e-scientist's simulation code can be incorporated as a module in the dataflow network.

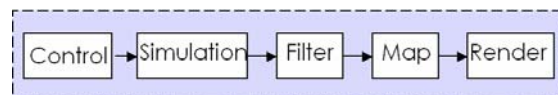


Figure 3 – Simulation in the dataflow pipeline

A number of key developments followed in the 1990s. The EPSRC/DTI funded GRASPARC project (Brodie et al, 1993) extended the notion of steering to a more powerful concept, whereby the

path to the solution was recorded as the simulation proceeded, allowing not only change of course as in the simple Marshall model, but also backtracking to earlier checkpoints so as to allow a 'reverse gear' as well as a 'steering wheel'. The resulting set of paths formed what was called a *History Tree*, recording the progress of an e-science investigation.

The CUMULVS library (Geist et al, 1996; Kohl, 1997) allows a group of scientists to collaboratively steer a simulation. As in GRASPARC, there is the opportunity to checkpoint results.

Other work extended the notion of steering to include control of performance, notably that by Vetter and Schwan in the development of Progress (Vetter and Schwan, 1995), Magellan (Vetter and Schwan, 1997) and Falcon (Gu et al, 1995).

Another key advance in the 1990s was the development of the SCIRun Computational Steering System (Parker et al, 1998; Parker, 1999). This is a dataflow system designed specifically to act as a problem-solving environment. Therefore it is more than just a visualization system: it also includes modules for setting up models (for example, creating boundary conditions) and for numerical computation (such as iterative linear solvers for the differential equations associated with a simulation). Particular attention has been given to the granularity of the modules, so that users can achieve effective control of the simulation.

The Pathfinder system (Hart and Kraemer, 1999) discusses issues of consistency in steering of simulations. One needs to maintain consistency, in the presence of two forms of *lag*: the lag between applying a steering parameter and the simulation responding; and the lag between results being generated by the simulation and their display by the visualization system.

Today, the steering paradigm fits very well into the context of modern Grid environments, where computing is distributed between desktop and remote servers. The simulation can be executed remotely but the visualization executed locally, giving the e-scientist control over the progress of the calculation from their desktop.

### 2.3 The gViz Project

The gViz project is studying visualization middleware for e-Science. It brings together a range of academic partners (Universities of Leeds, Oxford and Oxford Brookes; and CLRC), together with small, medium and large industrial companies

(Streamline Computing, NAG Ltd and IBM UK). Different parts of the project are exploring different issues, including the use of XML in visualization (see paper by Duce and Sagar at this conference); the Grid-enabling of pV3; and the compression of geometry to minimize data traffic in distributed visualization.

The part of the project described here is the Grid – enabling of IRIS Explorer. Our aim is to study to what degree existing visualization systems can be extended to operate within a Grid framework. We would again hope to exploit the extensibility of IRIS Explorer in being able to create new modules. The advantages are substantial: e-scientists do not have to change their way of working to adapt to a new visualization system and we can exploit a decade of development effort that has seen high quality algorithms being included as standard modules in IRIS Explorer. Equally we hope to identify those features of IRIS Explorer which inhibit its use, and these can drive new development work.

### 2.4 Pollution Alert

We have found it useful to develop our ideas against what we regard as a typical e-science application.

Imagine an industrial accident at a chemical plant resulting in the sustained release of a toxic chemical gas. This gas is dispersed under the action of the wind and could be blown over neighbouring inhabited regions. The question to be answered is which regions, if any, need to be evacuated? The answer is to simulate the event faster than real time in order to predict gas concentrations in particular regions over time.

The Grid can help by providing off-the-shelf compute power to allow scientists to simulate the event in faster than real time. A domain expert can run their atmospheric gas dispersion model and examine the results using data visualization. However, as well as computational resources, the scientist needs information about current weather conditions to steer the simulation. This data can be provided by experts from, for example, the meteorological office. Since time is short, rather than co-locating the scientists, they must be able to collaboratively steer the simulation and visualize the results. Since the decision to order an evacuation will be taken by the local authorities, the visualized data need to be presented in an easy-to-digest form. This requires the visualization tools to be flexible enough to allow different visualizations of the same data to occur, while still remaining under collaborative control.

### 3. An Early e-Science Demonstrator

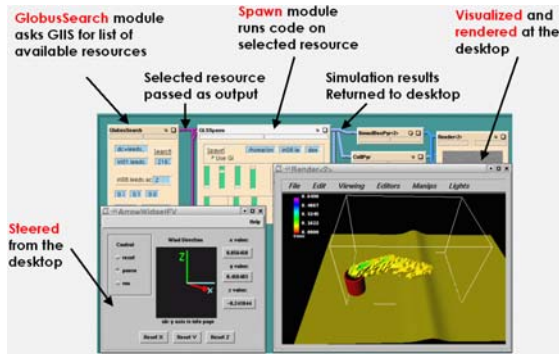


Figure 4 – Pollution demonstrator

In late 2001, we created an early e-science demonstrator based on the pollution application, and this has been presented at many e-science events (Brodli et al, 2002). The demonstrator was built using IRIS Explorer, and made use of the Globus Toolkit version 2. It illustrated a number of important e-science concepts:

- **Resource discovery:** a special IRIS Explorer module was created to display to the user available Grid resources (by enquiry to the GIIS).
- **Launching a simulation on the Grid:** a special IRIS Explorer module used Globus authentication to move simulation code from the desktop, to execute on a remote compute server. The simulation ran outside IRIS Explorer but socket connections allowed two-way data transfer.
- **Steering a simulation on the Grid:** control parameters (in fact the wind direction) were fed from the desktop to the remote simulation code.
- **Visualization:** results were returned from the simulation to the desktop for visualization using IRIS Explorer.
- **Collaborative visualization and computational steering:** making use of IRIS Explorer's collaborative visualization facilities, different people could both steer the simulation, and create a visualization of the results.

An annotated screen shot of the demonstrator is illustrated in Figure 4, and a schematic view is shown in Figure 5.

Although a useful illustration of what is possible, it also highlighted some difficulties:

- It was specially engineered for the application, rather than being based on a toolkit that others could use.
- The front-end application (IRIS Explorer) had to remain 'alive' for the duration of the simulation; for long-running simulations, one wants to be able to connect and disconnect from the simulation as it executes over a period of days.
- In order to check whether any steering change had been made, the simulation had to be interrupted frequently so that it could poll the IRIS Explorer interface (across the network) to check for changes. Performance analysis showed that 50% of the time that could have been used for simulation was spent checking for steering changes or sending data.
- All the data is transmitted across the network to the local machine for visualization. For large data problems it is more efficient to push the visualization towards the data so that the visualization computation is also happening remotely.

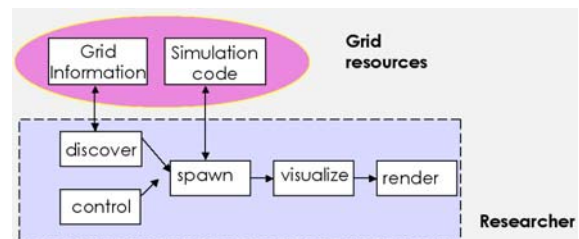


Figure 5 – Pollution demonstrator schematic

### 4. Grid-enabled IRIS Explorer

As a first response to these difficulties, we have extended IRIS Explorer so that it can run in a distributed fashion. The prime instance of IRIS Explorer runs as before on the desktop, providing a library of modules and a workspace in which these modules can be connected into a dataflow pipeline. However the e-scientist can now call up a secondary instance of IRIS Explorer, running on a remote Grid resource. Authentication to allow this is handled either by the `ssh` utility, or by the Globus Toolkit version 2.

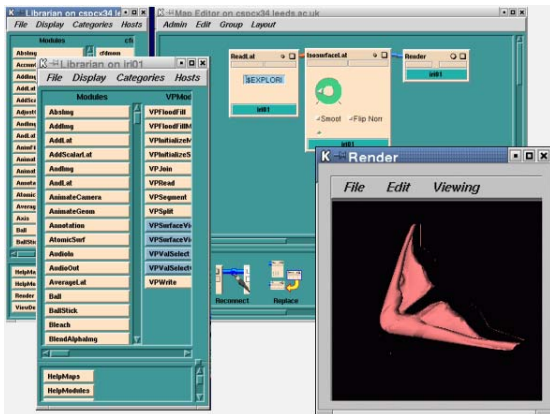


Figure 6 – Remote execution in IRIS Explorer

The e-scientist is now provided with a second library of routines, displayed on their desktop, from which they can select modules for inclusion in the workspace as part of the processing pipeline.

These modules form part of the single visualization application, but execute on the remote host. We illustrate this with the screenshot in Figure 6, and the schematic view is shown in Figure 7, where part of the pipeline executes remotely, across the Grid.

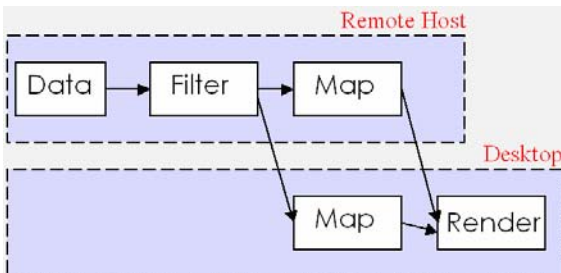


Figure 7 – Schematic of Grid-enabled Dataflow Pipeline Model

This addresses the first of the three limitations noted with the early demonstrator. The application is built using the IRIS Explorer toolkit that enables user code to be encapsulated as modules, and provides an easily created and consistent user interface.

Technically the Grid-enabled version of IRIS Explorer has been achieved by reworking the existing facility in IRIS Explorer for distributed working. This was based on the deprecated `rsh` facility and had fallen into disuse. The user can now specify to use either `ssh` or Globus to launch the remote components of IRIS Explorer. In the case of `ssh`, authorisation is handled via a user's certificate being transparently authenticated through `ssh-agent`. In the case of Globus, it uses the user's e-science Globus certificate proxy initialised by the user.

Advantages of this approach include:

- **Consistent working environment:** the e-scientist works entirely within a single environment, IRIS Explorer.
- **Exploiting existing technology:** we exploit all the powerful features of IRIS Explorer, including the ability to work collaboratively across the network.
- **Performance:** we can allocate the modules to resources so as to maximize performance. Essentially we have a logical pipeline describing the dataflow, and the e-scientist can map modules (simulation and visualization) to different Grid resources so as to provide the most efficient configuration.

However some limitations from the early demonstrator remain:

- The IRIS Explorer execution model is based on the principle that a module completes execution before examining its interface to check for parameter changes. This is somewhat against the idea of continuous steering. It is possible to work around this limitation by allowing the module to run for several iterations of the simulation and then stop. The module can check its user interface for changes and then re-fire to continue from where it previously stopped. However, there is a performance penalty on the simulation in having to keep stopping and starting.
- The primary instance must remain alive throughout the simulation.

## 5. The gViz Computational Steering Library

These limitations have therefore motivated our work to develop a library of software for use in computational steering applications.

The library is in two parts: one part provides an API which the e-scientist can use to instrument their simulation code; the other part provides an API which allows matching capability to be integrated into a front-end visualization system. Our development environment for this is IRIS Explorer, but other systems could be used in its place.

Our design aims include:

- **Lack of intrusion:** we want to impact as little as possible on the simulation, so that

integrating current simulation codes with the library is as easy as possible.

- **Minimise performance loss:** we want to maximise the time the simulation spends using the processor for computation by allowing the library to handle external interactions.
- **Breadth of scope:** we want to support the steering of both short and long-running simulations (support for long-running implies an ability to connect and disconnect).
- **Exploit service-oriented concepts:** we want to be able to view computational steering from a web services angle (thus we aim to see simulations registering with a web service, depositing information about how to connect – allowing e-scientists to locate running simulations, and connect to them from a front-end application).
- **Exploit existing visualization systems:** we take a view that we want to re-use existing visualization systems, which are now quite mature, but do not want to prescribe any particular system.
- **Well specified protocol for communication between front and back ends:** we have defined a simple XML-based language for this communication.
- **Distinguish fixed and variable parameters:** some parameters must remain fixed for the duration of a simulation to maintain integrity of the physics, or the numerical approximation (some of these are output by the simulation itself, such as current integration step-size and are of interest, but in a ‘read-only’ sense); others are truly steerable (such as frequency of output).
- **Manage different rates of producer-consumer:** the simulation produces data which the front-end interface consumes, but the rates will be different – therefore we need to have a strategy that accommodates this. For example, if the simulation generates data sets faster then the visualization system can visualize them then we need to either: skip intermediate data; or cache data between the simulation and the visualization system; or request the simulation slows down while the visualization system is attached.
- **Support collaboration:** we want to allow many simultaneous front-end processes to be able to connect, and we want the output to them to be synchronised.

A first implementation of this design has been completed. It provides a threaded library to handle external communications to keep interruption of the simulation to a minimum. It provides threads to receive changes to steerable parameters and holds them in a queue until requested by the simulation. It also provides threads to serve current parameter values to connected clients and threads to serve computed data.

A second component of the library provides API routines for clients to use to create connections to the simulation. These connections are then used to send/receive parameter values in an XML format, or to receive data.

Communication between simulation and client codes is currently handled through Unix sockets in an open way. Work has begun on a web services interface using the gSOAP library for communications with a GSI plugin to perform Globus authentication. It is also intended to provide a Globus authenticated and secured version of the current communications mechanism.

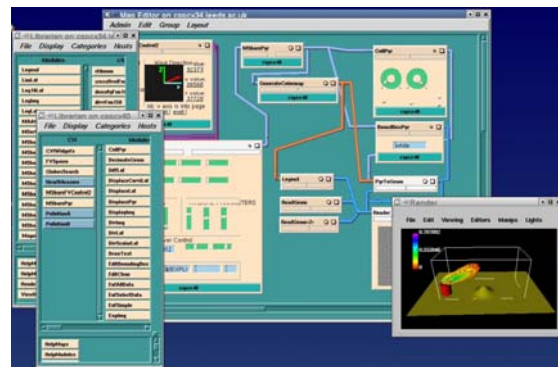


Figure 8 – New pollution demonstrator using gViz library

Figure 8 shows a screenshot of the pollution demonstrator; figure 9 shows a schematic representation, where we are running the simulation remotely on a Grid resource and also exploiting the idea of the previous section of distributing the dataflow visualization pipeline.

This library has been used to re-implement the original pollution demonstrator to remove many of its limitations. It can now be launched onto the Grid and allowed to run with no connection to external processes. IRIS Explorer modules can connect to the running simulation and view the values of its parameters as well as change those that are available for steering. Further modules can connect to receive data. All modules can disconnect, leaving the simulation to continue. It has also been used in GOSPEL, another e-science project, (also presented at this event in the paper by Goodyer et al). It allows the scientist to launch

long running parallel optimization problems on the Grid and to subsequently monitor, steer and visualize the data from their desktop.

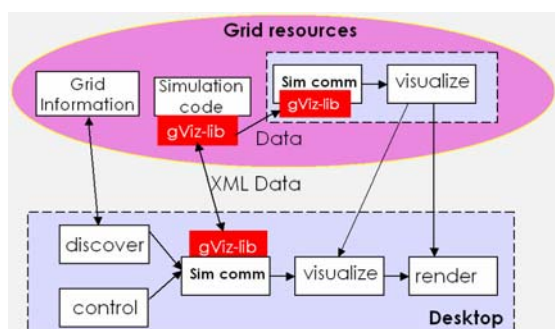


Figure 9 – Schematic of visualization and computational steering using the gViz library

This library, combined with the previously described extensions to IRIS Explorer gives us the flexibility to place simulations and visualization components in close proximity on the grid. Additionally, it allows us to communicate with the simulation and direct the visualization process from the desktop.

Finally by using the COVISA collaborative working facilities of IRIS Explorer, we allow teams of scientists to jointly work with the visualization interface.

## 6. Conclusions, Issues and Future Work

The gViz project has successfully extended IRIS Explorer to become Grid-enabled. This allows a computational steering application to be developed within IRIS Explorer, with the simulation running remotely, and with the steering and visualization running on the e-scientist's desktop. A positive advantage of this approach is that we are able to exploit the existing, well developed IRIS Explorer environment, including both its visualization modules and its facility for collaborative working.

There remain some limitations in this approach which have encouraged us to pursue a complementary strategy. We are developing a library for computational steering which allows the simulation code to run in a Grid environment, with IRIS Explorer limited to a visualization role. Indeed the approach can be used with any visualization system.

We have applied the different approaches to a pollution alert scenario, within the environment provided by the White Rose Grid. Our next aim is to use the gViz technology in helping the Leeds

Computational Biology group visualize heart modelling applications.

During our initial work on Grid-enabling IRIS Explorer, a number of issues have emerged. These include:

- Level of security: our use of Globus in section 4 is limited to the authentication of a user on the host machines that IRIS Explorer is running on – should we also worry about secure data transfer between modules within IRIS Explorer, when that transfer goes across machine boundaries?
- Collaboration: we use the COVISA collaborative extensions of IRIS Explorer without any modification for Grid computing – should we integrate this with Globus?
- Standardisation: IRIS Explorer has a proprietary set of datatypes. For better interworking between visualization systems, it would be useful to have standards in this area.
- Advanced steering: The GRASPARC History Tree mentioned in section 2 provided a level of sophistication in computational steering that remains even more valid today. It requires the storage of intermediate data from a simulation.

We hope to address these and other issues in the final part of the gViz project, and within other visualization activities in the White Rose Grid e-Science Centre of Excellence.

### Acknowledgements

We thank the others in the gViz project team for their many helpful discussions during the course of this work. Special thanks are due to Mark Walkley who wrote the pollution simulator, and to Chris Goodyer who has helped test the ideas through the GOSPEL project. Finally, thanks also to Martin Thompson, John Hodrien and Sally Mason.

### References

K. W. Brodlie, A. Poon, H. Wright, L. Brankin, G. Banecki, and A. Gay. (1993) GRASPARC: A Problem Solving Environment Integrating Computation and Visualization. In G.M. Nielson and D. Bergeron, editors, Proceedings of IEEE Visualization 1993 Conference, pages 102-109,

- Los Alamitos, CA, 1993. IEEE Computer Society Press.
- K.W. Brodlie, S. Mason, M. Thompson, M. Walkley and J.D. Wood. (2002) Reacting to a crisis: benefits of collaborative visualization and computational steering in a grid environment in: Proceedings of UK e-Science All Hands Conference, Sheffield, 2002.
- G.A. Geist, J.A. Kohl and P.M. Papadopoulos. (1996) Cumulvs: Providing fault tolerance, visualization and steering of parallel applications. SIAM, Aug 1996.
- W. Gu, G. Eisenhaur, E. Kramer, K. Schwan, J. Stasko and J. Vetter. (1995) Falcon: Online monitoring and steering of large-scale parallel programs. In Proceedings of 5<sup>th</sup> Symposium of the Frontiers of Massively Parallel Computing, pp 422-429, ACM.
- R. B. Haber and D.A. McNabb. (1990a) Visualization idioms : A conceptual model for scientific visualization systems. In: Visualization in Scientific Computing. IEEE, pp74-93, 1990.
- R.B. Haber and D.A. McNabb. (1990b) Eliminating Distance In Scientific Computing: An Experiment In Televisualization, in The International Journal of Supercomputer Applications, pp 71-89, Vol4(4) Winter 1990.
- D. Hart and E. Kraemer (1999). Consistency considerations in te Interactive Steering of computations. International Journal of Parallel and Distributed Networks and Systems, Vol 2, No 3, pp 171-179.
- ICENI Web Site (2003)  
<http://www.lesc.ic.ac.uk/iceni>
- IRIS Explorer Web Site (2003)  
[http://www.nag.co.uk/visualisation\\_graphics.asp](http://www.nag.co.uk/visualisation_graphics.asp)
- J.A. Kohl, High-Performance computing: Innovative assistant to science. (1997) ORNL Review, Vol 30, Nos 3-4. See <http://www.ornl.gov/ORNLReview/v30n3-4/hpc-ias.htm>
- B.H. McCormack, T. de Fanti and M.D. Brown (1987). Visualization in Scientific Computing Computer Graphics, Vol
- R. Marshall, J. Kempf, S. Dyer and C. Yen. (1990) Visualization Methods and Simulation Steering for a 3D Turbulence Model for Lake Erie, SIGGRAPH Computer Graphics, **24**, 2, pages 89-97.
- S.G. Parker (1999) The SCIRun Problem Solving Environment and Computational Steering System. PhD Thesis, University of Utah.
- S.G. Parker, M. Miller, C.D. Hansen and C.R. Johnson. (1998) An integrated problem solving environment: The SCIRun computational steering system. 31<sup>st</sup> Hawaii International Conference on System Sciences (HICSS-31), vol VII, pp 147-156.
- RealityGrid Web Site (2003)  
<http://www.realitygrid.org>
- J. Vetter and K. Schwan (1995) Progress: A toolkit for interactive program steering. In Proceedings of the 24<sup>th</sup> Annual Conference on Parallel Processing, pp 139-142.
- J. Vetter and K. Schwan (1997) High performance computational steering of physical simulations. In Proceedings of 11<sup>th</sup> International Parallel Processing Symposium, Geneva, Switzerland.
- J.D. Wood, H. Wright and K.W. Brodlie (1997) Collaborative Visualization, Proceedings of IEEE Visualization 1997 Conference, edited by R. Yagel and H.Hagen, pp 253--260, ACM Press.