Visualization and its Use in Scientific Computation

K W Brodlie, M Berzins, P M Dew, A Poon, H Wright
School of Computer Studies
University of Leeds
LEEDS, UK

1 Introduction

The importance of visualization in scientific computation is now widely recognised. Recent advances in large-scale, parallel computing have increased the potential of numerical simulation as a means of experimentation in the applied sciences; visual techniques of data presentation play a key role in the understanding and evaluation of the simulation results.

The birth of the subject is often associated with the the publication of the NSF-commissioned report 'Visualization in Scientific Computing' (McCormick et al, 1987). This report argued the case for increased attention to be paid to visualization, if full value was to be obtained from the investment in the US national supercomputer centres. Several of these centres then initiated major research and development projects in visualization, the results of which are now beginning to appear. A recent survey volume (Nielson et al, 1990) provides a good overview of these activities.

It is therefore tempting to think of 'scientific visualization' as a new discipline. But there is a case for arguing that it is only the term which is new. West (1991) claims that historically many original thinkers in the physical sciences have relied heavily on visual modes of thought, using images rather than words and numbers. For example, James Clerk Maxwell, founder of thermodynamics, built 3D clay models as an aid to his understanding of functions of two independent variables. (Some interesting computer simulations of Clerk Maxwell's work have recently been done (Jolls and Coy, 1990)).

From the early days of computing too, scientists have used computer graphics as a key part of their experimentation. In the UK, much pioneering graphics work, involving the production of animated sequences on film, was carried out at the Rutherford Appleton and Culham Laboratories in support of physicists in the 1960s. The Culham work led to the development of the GHOST system (GHOST, 1982) which has been widely used in the UK scientific research community over two decades.

While one can debate whether scientific visualization is a new subject or merely a new name, there is no doubt its importance has increased significantly in recent years. This can be attributed to a number of key developments:

- The increased computing power offered by modern systems, especially parallel and novel architectures, has extended the range of numerical simulation experiments which scientists can carry out. For example, meteorologists are tackling increasingly large atmospheric models. These simulations in turn generate vast amounts of data - 'firehoses' of data as they have been called; this data has to be evaluated as efficiently as possible. It is simply impossible for the human brain to comprehend more than

1992

a tiny fraction of the data in numerical form. However by converting entire fields of variables to a colour image, the brain is able to assimilate global information about the simulation;

- The increase in automatic data collection equipment, in particular medical scanners and remote sensing devices, has likewise led to large quantities of data requiring rapid processing;
- The trend away from batch processing to interactive working on a window-based workstation has brought graphics display technology to the scientist's desk as the norm.

While much of the recent activity in visualization has stemmed from the US supercomputer centres, there is an increasing contribution from Europe. A Eurographics working group has held two successful workshops, while in the UK, a recent workshop produced a status report on the subject (Brodlie et al, 1991). The authors of this paper are working in a collaborative research project, called GRASPARC, which is examining the close integration of computation and visualization (Brodlie et al, 1990). The project involves NAG Ltd, University of Leeds and Quintek Ltd.

This paper arises from our work on GRASPARC. Section 2 gives an overview of current work in scientific visualization which has provided input and stimulus to the project. The subject has still to mature - there are different views of its scope and underlying model, and no standards for the functionality to be provided by a system. A small prototype has been built within the GRASPARC project, to help our understanding of the subject and tease out the problems. This prototype is described in section 3, with future plans and conclusions in section 4.

2 Current Work in Scientific Visualization

2.1 Scope and Definition

A number of attempts have been made to define the term 'visualization'. A common flavour to these definitions is the idea that visualization is an aid to understanding: it is part of the experimental process, not simply a means of presenting the final results.

A succinct definition is given by Haber and McNabb (1990):

> Visualization is the use of computer imaging technology as a tool for comprehending data obtained by simulation or physical measurement.

This definition fails however to convey the value of visualization in allowing the scientist to control or steer the simulation, an important aspect certainly in the GRASPARC project. Marshall et al (1990) distinguish three modes in which visualization can be used:

Post-processing where the simulation results are stored and viewed at a later time, the scientist having interactive control over how the data are displayed;

Tracking where the simulation results are fed directly to the graphics module, the scientist again controlling how the results are displayed;

Steering where the simulation results are again fed to the graphics module, but with the scientist having interactive control over both simulation and visualization.

This suggests an extension to the Haber and McNabb definition, along the lines that visualization can be seen as a tool for guiding the computational process.

2.2 Underlying Model

Just as different authors have suggested different definitions of visualization, so there have been several attempts to define a model of the processes involved. A common thread is to distinguish different phases through which a physical problem passes towards its computational solution.

Haber and McNabb (1990) present the underlying model which is the basis of the National Center for Supercomputing Applications (NCSA) RIVERS project. They draw a parallel between the different stages in numerical simulation, and the corresponding stages in scientific experimentation. Thus they separate the simulation into three phases: *modelling, solution,* and *interpretation / evaluation.*

The modelling phase involves a two-step process in which the original, perhaps loosely defined, problem is first posed as a well defined physical model, and is then translated into an idealized mathematical formulation. The solution phase typically involves a discretisation step, where the continuous mathematical model is approximated by some discrete model to which numerical techniques can be applied. The output from this phase is a field solution for the unknown quantities in the model. Finally the interpretation and evaluation phase involves the analysis of the computed results, leading perhaps to some modification in the physical model, its mathematical idealization or the numerical solution technique. It is in this final phase that to date visualization has been most successfully employed, taking the field solution and displaying it in different ways.

Haber and McNabb see visualization itself as a three stage process. These stages transform the solution data to a displayable image on a graphics device, the stages being: *data enhancement, visualization mapping,* and *rendering.*

The data enhancement stage takes the raw solution data, typically defined on a mesh, and converts it to a form suitable for visualization. This frequently involves constructing a continuous model of the underlying field using an interpolation process. This interpolation process can be regarded as *filling out* the solution data with sufficient information; in the case of data obtained from scanners and remote sensing equipment however, this phase is more one of *data reduction* - the data is often collected to finer detail than the eventual display resolution.

The visualization mapping stage involves the selection of an appropriate visualization abstraction: for example, a 2D scalar field can be displayed as a contour map or a surface view. The choice of abstraction needs to be made so as to best understand the data.

The final rendering stage takes the abstract visualization object such as a contour map, and

renders it on the display surface.

This three-stage visualization pipeline is summarized in Figure 1.

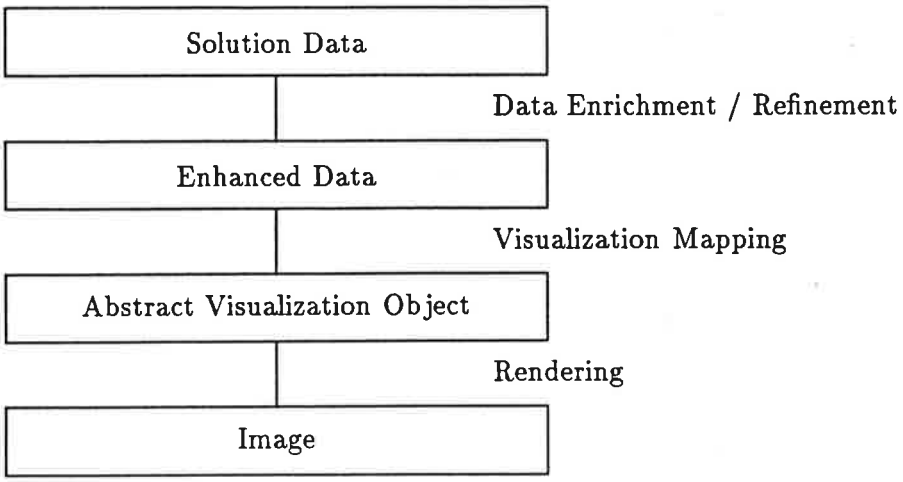

Figure 1 - Haber-McNabb Visualization Model

A similar visualization model was put forward by Upson et al (1989) as a basis for the Application Visualization System (AVS). They distinguish two cycles in the problem solving process: a *computation* cycle and an *analysis* cycle. The computation cycle consists of the following steps:

1. A theoretical stage, where the appropriate physical laws are determined.
2. A programming stage, where the physical laws are transformed to a computer program.
3. A specification stage, where details such as the computational grid, boundary conditions, etc are defined - these are essentially parameters of the computer program.
4. A computation stage, where the program is executed under the specified conditions.
5. An analysis stage where the results are examined and either deemed acceptable, or suggest further work. The cycle may be resumed from any of the first three steps, depending on the analysis of what is required.

The analysis stage is itself a cycle of three steps, roughly equivalent to the three-stage visualization pipeline of Haber and McNabb. The model is illustrated in Figure 2.

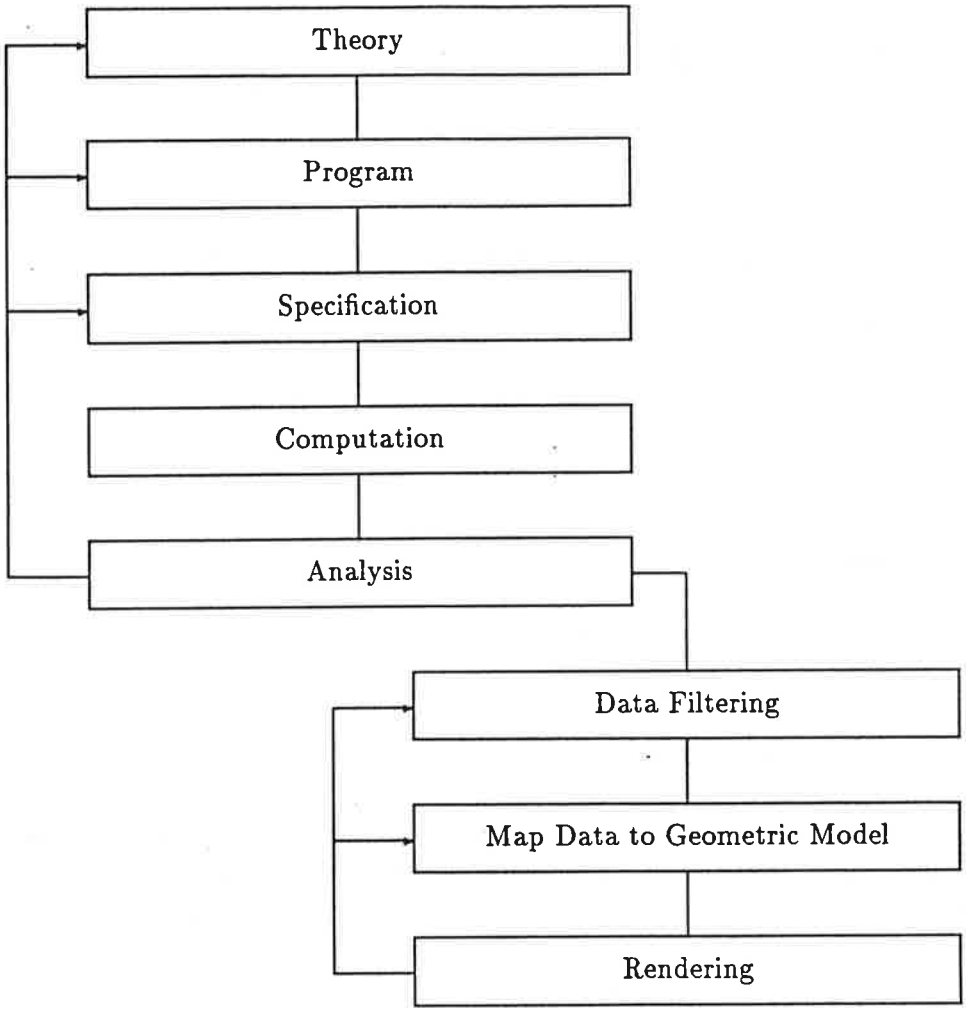

Figure 2 - AVS Visualization Model

Another view of the 'computational' and 'analysis' cycles is given in the paper by Carpenter (1990).

2.3 Classification of Techniques

With the variety of types of data and techniques available, it is useful to have a classification scheme to place some order on the subject. A number of schemes have been proposed. For example, the Bergeron and Grinstein (1989) scheme is based on the data that is to be visualized. They introduce the concept of *lattices* of data. A p-dimensional lattice of q-dimensional data is written L_q^p. The dimension of the lattice indicates the ordering of the data - zero-dimensional is unordered (eg a set of points), one-dimensional is a vector of data elements (eg a list of points to be connected by a polyline), and two-dimensional indicates an array of data elements (eg height and pressure at the nodes of a rectangular grid). The

dimension of the data refers to the number of components in a data element. Thus a list of points in 3D is a one-dimensional lattice of three-dimensional data, an object of type L_3^1.

Bergeron and Grinstein go on to describe the visualization process as a sequence of transformations on lattices. The process of interpolating scattered data on to a rectangular grid, for example, would convert a lattice L_2^0 to a lattice L_2^2.

Another classification scheme was developed at a UK workshop on scientific visualization (Brodlie, 1991). This is based, not on the properties of the sampled data, but on the properties of the entity, or field, underlying the data. This is the empirical model created in the 'data enrichment' stage of the Haber and McNabb pipeline. The entity is written as E, with a superscript to indicate the nature of the field, which can be Point (P), Scalar (S), Vector (V) or Tensor (T). In the case of vector and tensor fields, the dimension is indicated as a subscript; so E^{V_3} indicates a 3D vector field.

The other element of the classification is the dimension of the domain over which the entity is defined. Thus a scalar field in 2D is written E_2^2, a 3D vector field over 3D space as $E_3^{V_3}$. Further distinction is made according to the nature of the domain: if the entity is defined over ranges of values (as in a histogram or choropleth map), the subscript is written as [n]; if the entity is defined over an enumerated set (as in a bar chart), the subscript is written {n}.

It has been found that this classification scheme can code all the popular visualization techniques, from a 2D scatter plot (E_2^P) to a volume visualization (E_3^S), to a shaded contour map ($E_{[2]}^S$).

A third classification scheme uses the mathematics of fiber bundles (Butler and Pendley, 1989). A fiber bundle is a space constructed from a base space and a fiber space: one can imagine a fiber through each point of the base space. Essentially the base space corresponds to the independent variables, the fiber to the dependent variables; the bundle is the Cartesian product of the base and the fiber. For example, a line graph is formalised as follows. The base is the real line (x-axis) and a fiber is a real line through any point on the base; the fiber bundle is the set of all such lines. A section through a bundle is obtained by picking a point on each fiber - giving a line graph. Butler and Pendley develop a taxonomy of techniques, classifying them according to dimension of base and dimension of fibre. A nice feature of the scheme is that operators can be defined on the bundles to obtain, for example, cross-sections which are bundles of lower dimension.

2.4 Visualization Systems

A number of software systems have been developed, all following the visualization paradigm expressed by Haber and McNabb - with the data enhancement, visualization abstraction and rendering being implemented as a pipeline of processes. They are often given the generic term of 'dataflow systems', reflecting the passage of data through the pipeline; they are also referred to as 'application builders', since they provide a toolkit from which different applications can be created.

The first of these systems to make a major impact was AVS, developed by Stellar (Upson et al, 1989) - now part of Stardent. It provides a number of modules for accepting data into the system, filtering and enhancing the data, defining the abstract representation and

rendering on the display. To build a particular application, the user connects modules together using a *network editor*. The resulting pipeline will typically have a data source at one end, and a display module at the other end. Networks can be dynamically reconfigured to allow a scientist to experiment with different representations. The system has an X-Window based user interface with Motif-like look and feel. An early criticism was the fact that it was restricted to Stellar workstations; but this situation has dramatically changed with announcements that it has been licensed by several major workstation vendors.

AVS is an active product: AVS3 is the latest release. This release has increased support for distributed visualization in which different modules run on different processors. This has been used for example in joint work by Cray and Stardent, to allow compute-intensive simulation on a Cray and visualization on a Stardent workstation (Curington and Coutant, 1991).

The proprietary nature of AVS has encouraged the development of further visualization systems, which offer a similar way of working but are intended to be freely available at least in the educational community. A major product is apE from Ohio State University, one of the US supercomputer centres (Dyer, 1990). This works in much the same way as AVS, the user constructing a pipeline of processes. A large number of modules are provided with the system, including a volume visualizer, and the user can incorporate external modules as with AVS. The system is built to allow distributed processing, with modules connecting *via* UNIX sockets.

Another product is Khoros from University of New Mexico (Rasure, 1991). This has its roots more in image processing but is being used for general 2D data visualization. A consortium (Khoros, 1991) has been established to distribute Khoros and carry out further research. A stated aim is to provide Khoros free of charge to any organisation *via* network access. Consortium members will have the privilege of early releases and certain rights to develop enhancements and redistribute the software.

2.5 Data Management Systems

Visualization can be seen as a navigation process through a large database. It is clearly important that the data be well organised and structured so as to make this navigation efficient.

The traditional approach of using flat sequential files for data storage is inefficient in storage and access. Commercial relational database systems such as INGRES and ORACLE have been used by some groups of scientific users, but they are largely oriented to business applications. There is a need to store and retrieve the data objects that naturally occur in scientific computation, particularly multidimensional grids. Thus an important development over the past few years has been the emergence of data management systems targetted at the particular needs of the scientific computation community.

One of the first examples of a system based on a scientific data model was NASA's Common Data Format (CDF). Another example is the Hierarchical Data Format, or HDF, developed at NCSA (NCSA, 1989). HDF defines a multi-object file format for transfer of graphical and floating-point data between different systems. It allows, for example, a grid of data to be stored together with scales, annotation, etc; slices through datasets can be extracted.

Simple Fortran and C calling interfaces are provided, and so the package provides a simple scientific data management system. Moreover, NCSA also supply some useful visualization tools which accompany HDF. The software is in the public domain.

A good review of data management for scientific visualization is given by Carpenter (1991). Another useful source is the report of a workshop at SIGGRAPH 90 (Treinish, 1991) which includes a comparison of different systems.

2.6 Conclusions

AVS, apE and Khoros represent the current state of the art in scientific visualization environments. Their view of the world is dominated by data visualization: a source of data is fed through a pipeline of processes reaching a display process at the end of the pipe. An application module can be inserted as the data source, and some interaction allowed with that module (Marshall et al, 1990), but the notion of a single linear pipeline from source to sink persists.

This style of working does not necessarily reflect the exploratory nature of mathematical modelling. Here the view of the world is dominated by the modelling processes, with visualization as a set of windows onto these processes. These windows control aspects at the different levels of the model (the physical, mathematical and discretised levels of Haber-McNabb) and similarly provide views of the solution at different levels. Since the process is an experimental one, recording of the data is essential so that experiments can be restarted from intermediate points with change of parameters.

Thus it seems to us that data management must play a key role in visualization for mathematical modelling. It is significant that the existing dataflow visualization systems have been developed separately from the data management systems described in the previous section. Can an improved system be created by merging the two ideas?

3 Prototype Demonstrator for GRASPARC Project

3.1 Aims of GRASPARC

The aim of the GRASPARC project is to explore how best to provide an integrated environment for numerical simulation and visualization. This environment should support interaction not just with the solution data, but also with the physical model, its mathematical idealisation and the numerical solution. Data management is to be given special attention so that a history of the computational process can be properly maintained.

This leads us to study a number of issues:

- Can we define a visualization reference model that clearly distinguishes the different phases in problem solving, and identifies the interaction and data objects appropriate to each phase?
- Can we build a system that offers these levels of interaction, and allows the scientist to explore interactively throughout the solution process?

- Are there applicable standards that will enhance the portability of the system?
- Can the system be implemented efficiently on modern computer architectures?

A first step in the GRASPARC project has been to build a prototype demonstrator which can help resolve some of these issues.

3.2 Demonstrator Problem

A relatively simple mathematical modelling problem was chosen for the prototype demonstrator: the motion of a particle in a potential field coupled to a heat bath (Cartling, 1987). The physical problem is Brownian motion; its mathematical idealization is the Fokker-Planck equation:

$$\frac{\partial P}{\partial t} = -v\frac{\partial P}{\partial x} + \frac{1}{m}\frac{dU}{dx}\frac{\partial P}{\partial v} + \beta\left(\frac{\partial}{\partial v}(vP) + \frac{\kappa T}{m}\frac{\partial^2 P}{\partial v^2}\right)$$

$$P = P(x, v; t)$$

where:

$P(x, v; t)$ is a probability density function, expressing the probability P that the particle has a given position x and velocity v;

$U(x)$ is the supplied potential (bistable); and

β is a damping factor (strength of coupling to the heat bath).

The numerical formulation is the method of lines, in which derivatives in the spatial variables x and v are replaced by differences, and the resulting system of ordinary differential equations solved by some numerical technique.

Given an initial distribution, the problem is to find out what happens over time. The potential defines two minima, representing product and reactant states separated by a barrier, across which thermally activated transitions take place. The damping factor β determines the motion of the particle: for weak damping (small β) the motion is deterministic, and for strong damping (large β) stochastic. An aim is to ascertain the behaviour of the solution P as β varies.

The scientist will want to interact at a number of levels - for example:

- to modify the mathematical idealization by changing β
- to control the numerical solution by changing the spatial discretization

To achieve this interaction, the visualization of the solution should be presented at the same level. For control of the numerical solution, the results on the computational grid should be

highlighted; for control of the mathematical idealization, the corresponding approximation to the continuous solution is required.

For any given solution data set, the scientist will also wish to have different views of the data - a further mode of interaction.

Finally the scientist will wish to compare runs with different values of β, and different discretisations, and so data from each computational run must be stored. This gives effectively a tree structure containing the computational history.

3.3 Building the Demonstrator

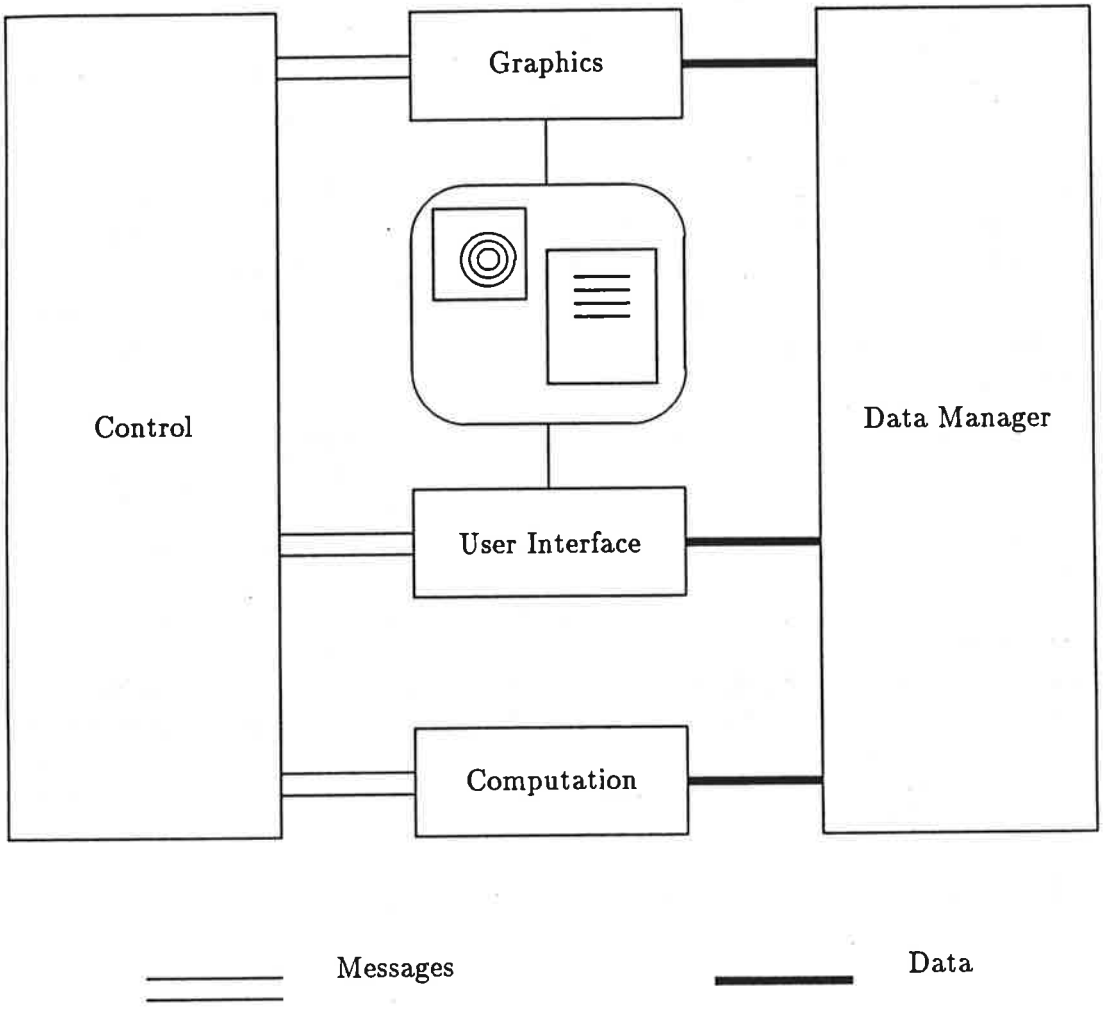

Figure 3 - GRASPARC Prototype Demonstrator

The prototype which has been built with these aims in mind has a structure as shown in Figure 3. Three separate modules implement computation, graphics and user interface components. The modules operate concurrently and communicate with each other by a message passing mechanism. Problem specification and solution data are held in a data store.

The system has been implemented on a Silicon Graphics 4D/240 workstation, with display management under control of the X Window System. Processes communicate *via* UNIX sockets, and so may run on different machines.

The computation module is based on the SPRINT software (Berzins et al, 1989) for solution of differential equations, developed at the University of Leeds in conjunction with Shell Research Ltd, and now available through the NAG Library. At times requested by the user, results are written to the data manager which is based on the HDF system described earlier. Calls to HDF are included directly in the computation module.

The graphics module consists of software to display scalar fields of two variables - contour plots and surface views - over a sequence of times. This software is written in terms of PHIGS PLUS (ISO, 1990), and allows some simple direct interaction on the part of the scientist, such as rotation of the surface.

The user interface module is based on a simple X toolkit called SUIT (Bowers and Brodlie, 1991), which allows the building of form-type interfaces. One form allows the scientist to enter control details for the computation; another, control details for the graphics.

The communications channel is implemented as a UNIX control process with socket connections to the computation, graphics and user interface modules. This control process acts as a switch for messages passed between the modules.

The system runs in the following way. On start-up, the user completes a form specifying details of the computation: mesh-size for discretisation, times at which output is to be reported, and so on. As the computation proceeds, messages are sent at each time step to the user interface module which draws a computation tree showing the progress. This acts as a *monitor window* on the computation. Immediately results are written to the data manager, the scientist may call up the graphics module to display the solution. The computation can be halted at any time, and restarted with different parameters. For example, the scientist can restart the computation with a different start time and change the time interval between time steps at which output is written. This is then displayed in the monitor window as a new branch on the tree. Figure 4 illustrates a series of such branches which together form a history tree structure. The scientist is able to trace the progress of the experiment easily.

4 Conclusions and Further Development

The prototype has achieved its aim of teasing out some of the difficult issues in developing a visual environment for mathematical modelling.

It has provided a basic understanding which is now allowing us to develop a reference model for problem solving, in which the physical, mathematical (or functional) and numerical layers are cleanly distinguished.

It has highlighted the importance of a data store as an intermediary between computation and visualization, with the scientist separately controlling the two processes. The interface to the data store becomes a key part of the system. Currently this is implemented directly in terms of HDF. However HDF does not directly support the computation tree structure that emerges in exploratory working, and so a small interface layer has been designed above HDF. This gives an 'Application Programming Interface' to the data management functions which supports the needs of a GRASPARC-like system. This will have the added advantage of removing the direct dependency on HDF, and allowing the introduction of any standard scientific data format, should that appear in the future.

The graphics module already makes use of a (draft) international standard, namely PHIGS PLUS. The prototype includes just two simple techniques, but the graphics module will eventually contain a range of techniques for displaying the various categories of fields described in section 2. The use of PEX (Gaskins, 1991), an extension of X to support PHIGS PLUS, will give the correct separation between graphics and display. This will allow the inclusion of specialist hardware to act as a PEX server - this to be built by Quintek Ltd as part of the project.

The user interface module will progress to the use of an established *de facto* standard toolkit such as Motif (OSF, 1989).

However a major study required will be the form of interaction which is most useful to the scientist. Some of the issues to be resolved are:

- What level of programmability is required? For example, current visualization systems use the concept of a network editor to dynamically configure the visualization pipeline. Can this idea be extended to the different style of working in GRASPARC?
- What control can usefully be given to the scientist over the numerical solution process - do adaptive algorithms largely remove this need anyway?

The ultimate test of the GRASPARC system will be its usability to the scientist.

Acknowledgements

GRASPARC is a collaborative project funded jointly by the UK Department of Trade and Industry and the Science and Engineering Research Council. The partners are NAG Ltd, University of Leeds and Quintek Ltd.

Our thanks go to all who are contributing to the project, especially Steve Hague, Lesley Carpenter, Richard Brankin, Greg Banecki and Alan Gay of NAG Ltd; and Pat Mills and Mark Powell of Quintek Ltd. Thanks also go to Neil Bowers, an SERC/NAG CASE award student at Leeds University who has developed the SUIT software, and to Justin Ware who assisted with the computation software.

References

R.D. Bergeron and G.G. Grinstein, 1989, *A reference model for visualization of multi-dimensional data*, Eurographics '89 Proceedings, Elsevier Science Publishers BV, p393-399.

M. Berzins, P.M. Dew and R.M. Furzeland, 1989, *Developing software for time-dependent problems using the method of lines and differential-algebraic integrators*, Applied Numerical Mathematics 5, pp375-397.

N. Bowers and K.W. Brodlie, 1991, *SUIT: A Portable User Interface Toolkit*, in preparation.

K W Brodlie, P M Dew and M Berzins, 1991, *GRASPARC: A Visual Environment for Numerical Computing*, NAGUA News, Issue 5.

K.W. Brodlie, 1991, editor *Visualization Techniques*, in *Scientific Visualization - Techniques and Applications*, eds Brodlie et al, Springer-Verlag, to appear.

K W Brodlie, L A Carpenter, R A Earnshaw, J R Gallop, R Hubbold, A M Mumford, C D Osland, P Quarendon, eds, 1991, *Scientific Visualization - Techniques and Applications*, Springer Verlag, to appear.

D M Butler and M H Pendley, 1989, *A Visualization Model based on the Mathematics of Fiber Bundles*, Computers in Physics, Vol 3, Number 5, pp45-51.

L A Carpenter, 1990, *The Visualization of Numerical Computation*, in Proceedings of Eurographics Workshop on Scientific Visualization, Clamart.

L.A. Carpenter, 1991, editor *Data Management*, in *Scientific Visualization - Techniques and Applications*, eds Brodlie et al, Springer-Verlag, to appear.

B. Cartling, 1987, *Kinetics of activated processes from non-stationary solutions of the Fokker-Planck equation for a bistable potential*, J. Chem. Phys. 87 (5).

I.J. Curington and M.D. Coutant, 1991, *AVS - A flexible interactive distributed environment for scientific visualization applications*, Proceedings of 2nd Eurographics Workshop on Scientific Visualization, Delft, to appear.

D.S. Dyer, 1990, *A Dataflow Toolkit for Visualization*, IEEE Computer Graphics and Applications, Vol 10 (4), pp60-69.

T. Gaskins, 1991, *PEX Sample Implementation API Overview*, Proceedings of 5th Annual X Technical Conference.

GHOST, 1982, *GHOST80 User Manual*, UKAEA Culham Laboratory.

R.B. Haber and D.A. McNabb, 1990, *Visualization Idioms: A Conceptual Model for Scientific Visualization Systems*, in *Visualization in Scientific Computing*, eds G.M. Nielson, B. Shriver and L.J. Rosenblum, pp74-93, IEEE.

ISO/IEC, 1988, *Information processing systems - Computer graphics - Programmer's Hierarchical Interactive Graphics System - Part 1 - functional description*, ISO/IEC 9592-1.

ISO/IEC, 1990, *Information processing systems - Computer graphics - Part 4: Plus Lumiere Und Surfaces (PHIGS PLUS)*, ISO/IEC DP 9592-4.

K. R. Jolls and D.C. Coy, 1990, *The Art of Thermodynamics*, IRIS UNiverse: The Magazine of Visual Processing, No 12, pp31-36.

Khoros Group,, 1991, *The Khoros Consortium Description*, Department of ECE, University of New Mexico, Albuquerque, USA.

R Marshall, J Kempf, S Dyer and C Yen, 1990, *Visualization Methods and Simulation Steering for a 3D Turbulence Model of Lake Erie*, Computer Graphics, Vol 24, Number 2, pp89-97.

B.H. McCormick et al, 1987, *Visualisation in Scientific Computing*, Computer Graphics, 21 (6).

NAG, 1989, *NAG Graphics Library Handbook - Mark 3*, 1st edition, NAG Ltd.

NCSA, 1989, *NCSA, HDF Calling Interfaces and Utilities*, NCSA HDF Version 3.1, National Center for Supercomputing Applications (NCSA) at the University of Illinois Urbana-Champaign.

G M Nielson, B Shriver and L J Rosenblum, eds, 1990, *Visualization in Scientific Computing*, IEEE Computer Society Press.

Open Software Foundation, 1989, *OSF/MOTIF Manual*, Open Software Foundation.

J. Rasure et al, 1991, *A Visual Language and Software Development Environment for Image Processing*, International Journal of Imaging Systems and Technology.

L A Treinish, 1991, *SIGGRAPH 90 Workshop Report - Data Structures and Access Software for Scientific Visualization*, Computer Graphics, Vol 25, Number 2, pp104-118.

C. Upson et al, 1989, *The Application Visualization System: A Computational Environment for Scientific Visualization*, IEEE Computer Graphics and Applications, Vol 9 (4), pp30-42.

T. West, 1991, *In the Mind's Eye*, Prometheus Books, New York.