Systems and Architectures for Visualization

S Larkin, W T Hewitt, A J Grant
Manchester Visualization Centre, Manchester Computing, University of Manchester, UK.

Abstract
This paper, which is an extract from [1], presents an introduction to scientific visualization and an overview of the associated tools and systems. Section 1 covers the history and background with some discussion on the models which have been developed to describe the visualization process. A classification of visualization systems and a description of their architecture is included in sections 2 and 3.

1 WHAT IS SCIENTIFIC VISUALIZATION?

Scientific visualization [2] is concerned with the analysis and exploration of data and information to gain a greater understanding and insight. It can be split into two broad categories:

- **Visualization: where the researcher is looking to understand the problem;**
- **Presentation: where the scientist is presenting results to other colleagues.**

The term visualization is really an amalgamation of aspects from disciplines such as image and signal processing, computer graphics, human computer interface (HCI) etc. This combination of techniques can be seen in the multitude of visualization systems which have emerged.

1.1 History and background

Clearly the ideas, algorithms and techniques encapsulated by visualization have been in use earlier than the late 1980's, but it is the report published in 1987 [3] which is much cited as the start of the visualization era. The report was the outcome of a two day workshop organised by NSF on "Visualization in Scientific Computing". It summarises the conclusions and recommendations of the workshop and defines visualization and the areas it covers. The main recommendation from the report was to coordinate the development of scientific visualization tools and their provision to the scientific community.

However, these statements do not mean that all visualization algorithms and techniques were already in place. Over the past few years there has been a great deal of work into processing and visualizing volumetric (3D scalar) data and new techniques for visualising multivariate data and tensor fields.

1.2 Models for the visualization process

Over the years, several models related to scientific visualization have been proposed. Watson's [4] model addressed the entire process of scientific investigation with the computational and analysis aspects being further explored by Upson [5] and Carpenter [6]. Haber and McNabb provided a more detailed look at the scientific visualization process with their particular model [7].

1.2.1 Haber and McNabb model

The model of Haber and McNabb provides a general classification of the visualization process from a users point of view. It divides visualization into three broad processes, each of which acts on some data to produce a new set of data, shown in figure 1. The example data shown on the right of this figure are temperature samples taken from a cross section through a polymer injection moulding system [8].

Data preparation, acts on the raw data (measured or simulated). This process creates a model of the data from which a new derived data set can be produced e.g., calibration, smoothing, interpolation.

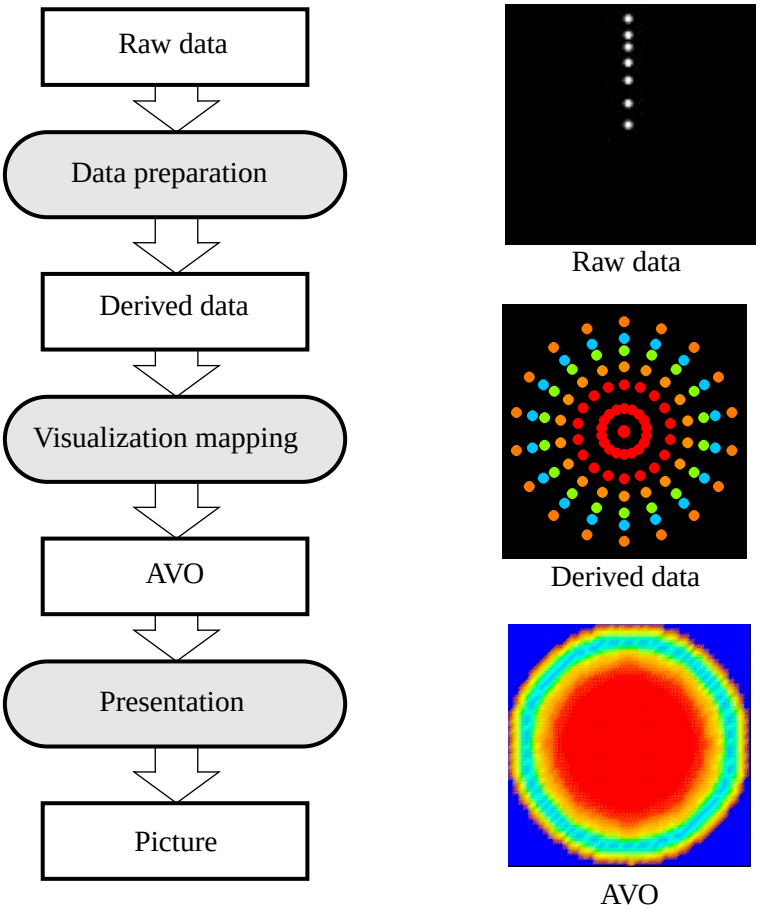


Fig. 1 Haber and McNabb visualization process

The raw scattered data shown on the right of the figure is replicated and interpolated to form a uniform 2D grid.

V*isualization mapping*, creates an *Abstract Visualization Object* (AVO). Each quantity in the derived data is mapped to an attribute – space and time dimensions, colour, transparency – of the AVO e.g., a height field being mapped onto a 3D coloured surface. There can be multiple AVOs. The derived temperature values in figure 1 are mapped onto an AVO which is a coloured 2D plane.

Presentation, is where one or more views of the AVOs are rendered to produce a picture on an output device, generally the screen of a workstation.

Enhancements have been made to this model, along with the development of new models, but it is still the Haber and McNabb model which provides a general breakdown of the visualization process still applicable when describing the architecture of current visualization systems. As exploitation of the internet and distributed resources continues to grow visualization systems are being adapted to take advantage of this processing environment and new models [29] are being developed.

2 VISUALIZATION SYSTEMS

There are numerous public domain, commercial and research visualization systems available and a common classification is to use the following categories:

– **Libraries/toolkits;**

– **Turnkey packages;**

– **Modular Visualization Environments (MVEs) or Application Builders.**

These classes provide a convenient method for grouping visualization systems when discussing them. In reality, most systems exhibit features in a number of these catergories.

2.1 Libraries/toolkits

Numerous graphical subroutine libraries are available, ranging from PHIGS (PEX), OpenGL, Graphical Kernal System (GKS) and X/Motif, to more object-oriented systems such as Inventor [9]. These provide the most flexible level at which to write applications but most of them require the need to generate large amounts of code to implement basic visualization tasks. Taking this into account they are not an end user solution.

Application subroutine libraries and toolkits such as Numerical Algorithms Group (NAG) graphical subroutine library, UNIRAS Toolmaster, Visualization ToolKit (VTK) [10] provide more support. They include higher level tasks such as different visualization techniques, annotation and axes control. Although these common actions are provided they still require programming knowledge on behalf of the user and large amounts of code to be written.

2.2 Turnkey packages

These are sometimes referred to as *point-and-click* systems since they are usually characterised by a menu driven user interface. Examples of these systems are PV-WAVE, Gsharp, Wavefront DataVisualiser, Analyze, Gnuplot etc.

They have the advantage of being quick and easy to use but present a black-box style of application which is essentially a closed system. This makes extending the system or enhancing the functionality by the integration of application code difficult. However, most of these packages have some command line interface or macro language which provides flexibility to varying degrees. Another problem with turnkey packages is that they are normally targeted towards a particular application domain.

2.3 Modular Visualization Environments (MVEs)

Sometimes referred to as *application builders*, visualization applications are constructed by connecting modules together to form networks (or maps). The systems are based on the dataflow paradigm where data passes through the network and modules are the compute nodes which process/transform the data. These systems can be extended by incorporating application code into the system by writing new modules.

The set of Modular Visualization Environments are the Application Visualization Systems (AVS) [11], Iris Explorer [12], IBM Data Explorer [13], Khoros [14][15] and aPE [16]. Figure 2 shows a sample module from AVS which performs the streamlines visualization technique. Data imported into the module includes the dataset from which the streamlines are generated and optional input values one of which is used to colour the streamlines if needed. The output from the module is a geometric representation of the streamlines.

MVEs have the advantage of being flexible, extensible and address many different application domains. Unfortunately there is a larger initial learning curve and because the systems cover multiple application areas there are many modules to choose from when building an application.

As the MVEs are the most flexible and extensible systems it is these systems which this work concentrates upon.

2.4 Modes of interaction

An important aspect of the visualization procedure is the interaction the system provides between the user, data generation/collection and the visualization system. There are three principle modes of interaction with a visualization system:

– **post-processing**

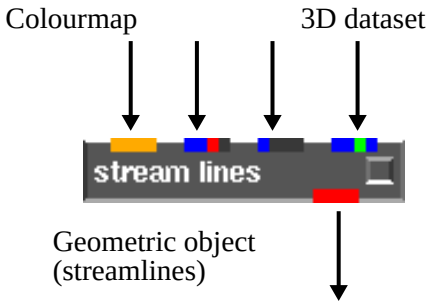


Fig. 2 An example of a module from the MVE AVS

– **interaction/tracking**

– **steering.**

2.4.1 Post-processing

This is the simplest of the three modes of interaction. Data is generated and collected off-line and then processed by importing into a visualization system. There is no direct communication between the data generation task and the visualization system, see figure 3.

Post processing has the advantage of being simple to use but it does not provide any method of reacting to atypical data sets and may produce unsuitable, wasted output. Most visualization systems are capable of this mode of operation.

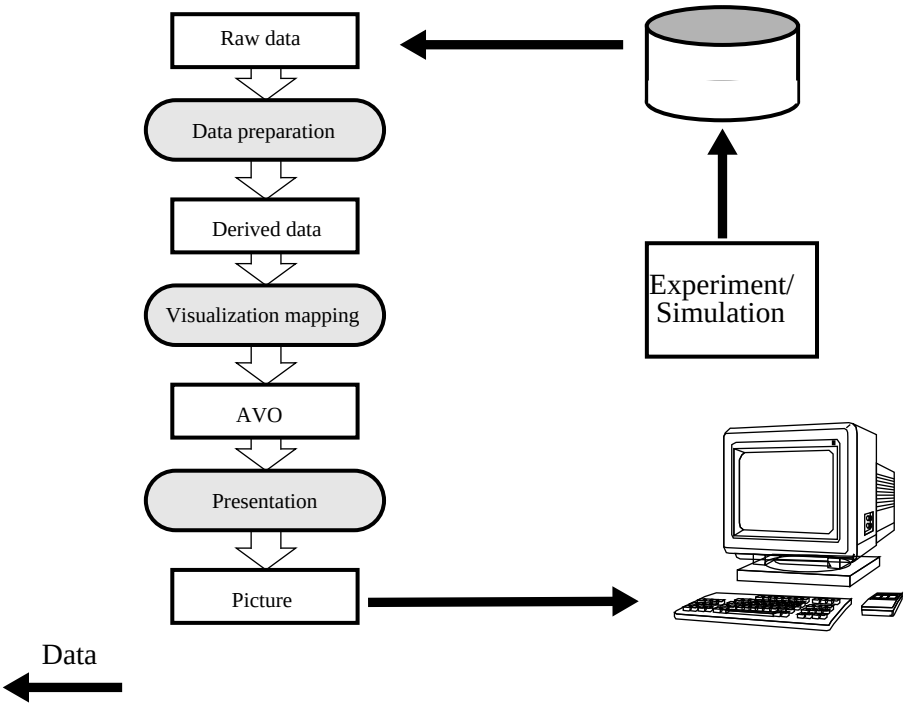


Fig. 3 Post processing

2.4.2 Interaction/tracking

As in post-processing, the data production is separate from the visualization system but the user has interactive control over the visualization task.

Interaction allows views of several different visualizations of the same data set and examination of the model from different angles. Functions such as probing and measuring the model can also provide greater insight. The main disadvantage is that if original data is flawed, the data generation has to be started all over again.

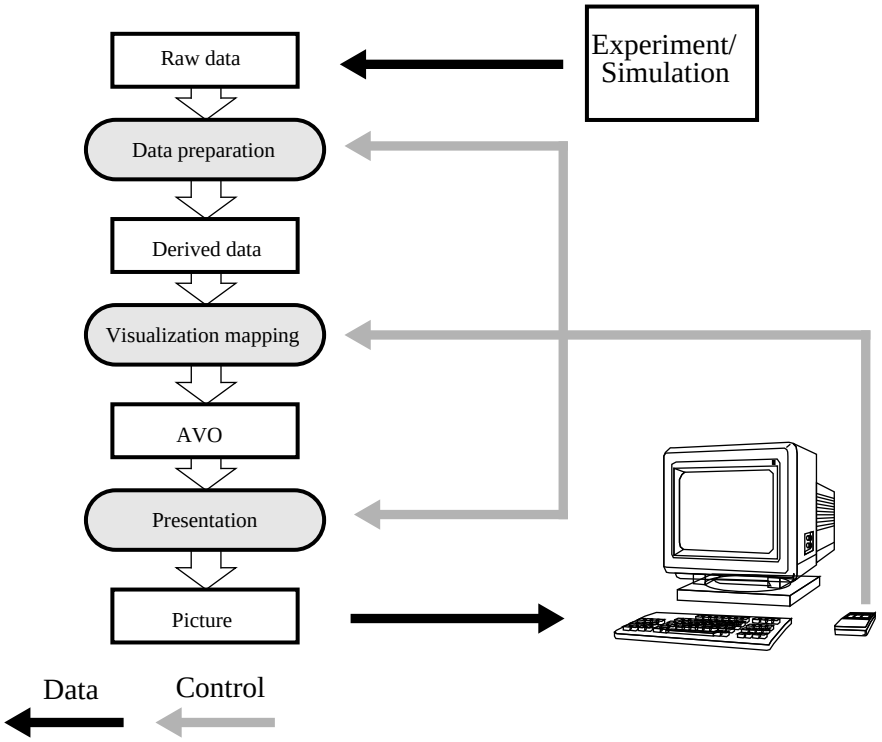


Fig. 4 Interaction

2.4.3 Steering

Steering is the most advanced mode and combines the interactive control of both the data generation and visualization tasks. This mode effectively *closes the loop* as results at one stage can affect the preceding as well as the following stages. It has the advantage that the user can rapidly abandon or adjust simulations that are going wrong and allows iterative homing in on an optimal solution. The disadvantages are that it often requires high bandwidth network connection between the simulation compute engine and machine hosting the visualization systems. Some changes to the simulation code may be needed. Research has been made into investigating the design of systems for computational steering and implementations of some of these ideas have been developed e.g., GRASPARC [17] and VASE [18]

3 A CLOSER LOOK AT MODULAR VISUALIZATION ENVIRONMENTS (MVES)

At an abstract level the current MVEs provide similar features and all use the idea of a visual editor to construct networks of modules, see figure 6. In practical terms the systems differ in price, functionality, support, robustness etc. There are many ad-hoc comparisons of different MVEs and turnkey packages by individuals but it is not the purpose of this paper to compare and contrast the systems. Instead the

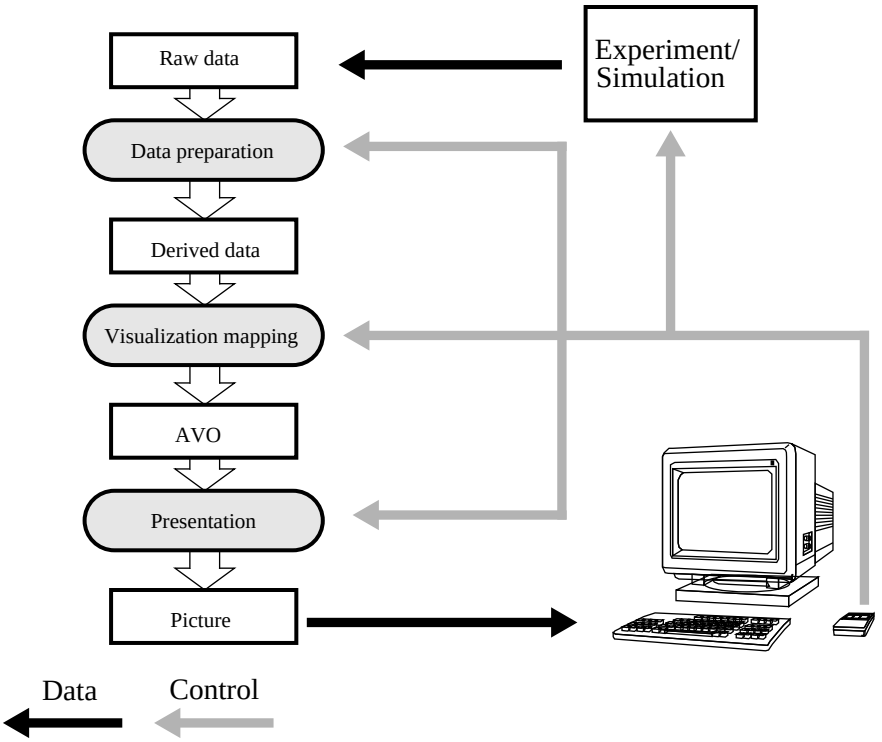


Fig. 5 Computational Steering

reader is recommended to consult two comparisons which have been compiled by groups to evaluate the various systems. These are the UK Advisory Group on Computer Graphics (AGOCG) [19] and Stichting Academisch Rekencentrum Amsterdam (SARA) [20] reports.

3.1 Next generation systems - a move from the dataflow paradigm

AVS, aPE, Iris Explorer, IBM Data Explorer and Khoros are all MVEs which are based on the dataflow principle. If the optimizations vendors have added to the systems are ignored then dataflow inherently means that many copies of the data are made as the dataset is processed through a network of modules. To address this and other issues Advanced Visual Systems upgrade to AVS, AVS/Express [21][22], is based on a completely new architecture of data-referencing.

The two figures 7 and 8 show the effect of copying data in a very simple network.

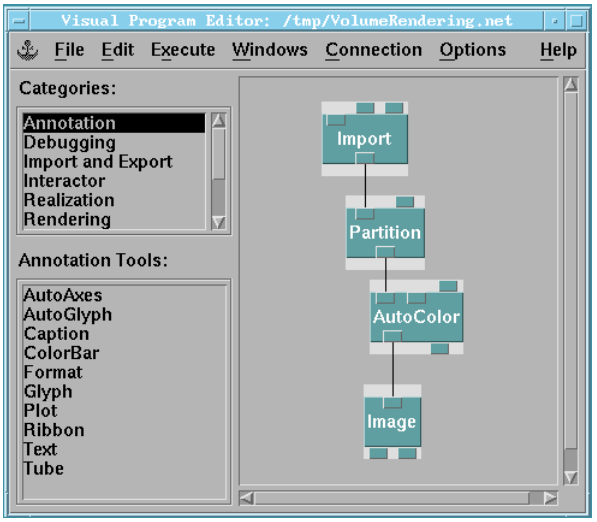
Dataflow: The filter module in figure 7 transforms the original data ($Data^{O}$) and creates a copy of the dataset which contains the unchanged coordinate mesh and new data ($Data^{T}$). The mapper module represents a visualization technique by a set of primitives which use the same coordinate mesh. Once again, the new dataset contains a copy of the original coordinates.

Data-referencing: Figure 8 shows the same application but using a data reference scheme. The filter module only operates on the original data creating a new object containing the results. The coordinate data is represented by a link to the object containing the coordinate mesh. The same action is taken for the mapper module. This also has the advantage that if the original coordinate mesh is altered this change is propagated to the other modules via the references and not by generating new copies.

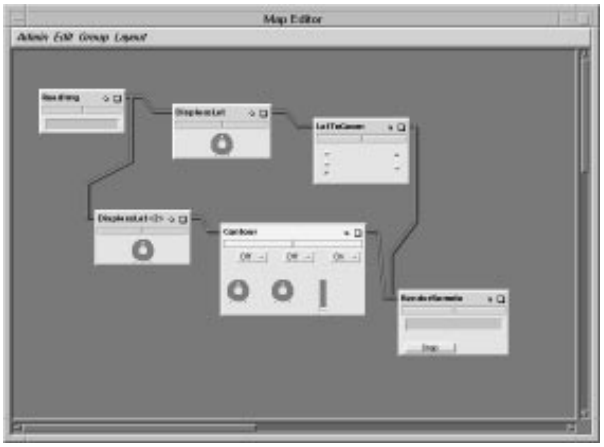
3.2 Overview of AVS/Express

AVS/Express enables you to create/instance objects and connect them to form higher level objects and visualization applications.

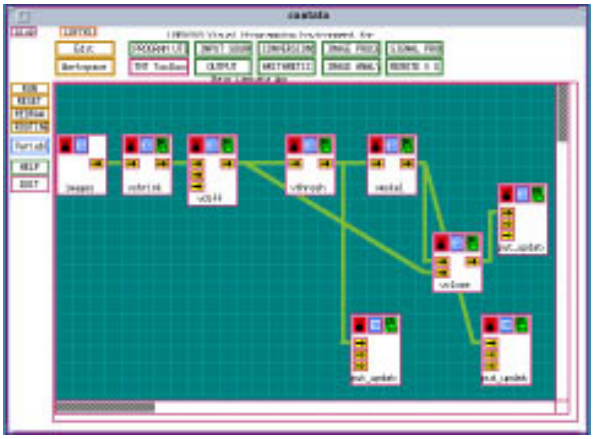
Application Visualization System



IBM Data Explorer



Iris Explorer



Khoros

Fig. 6 Visual programming paradigm for MVEs

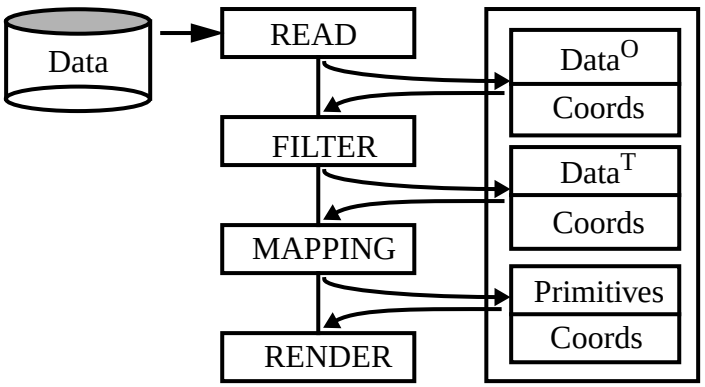


Fig. 7 Dataflow and implicit copying of data

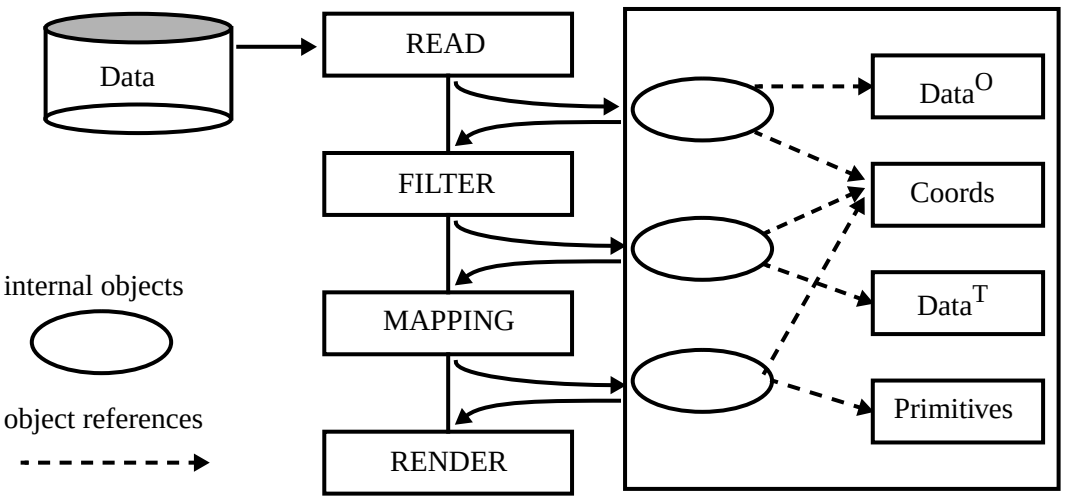


Fig. 8 The removal of copying data using data referencing

There are a large number of different object types available in AVS/Express and the following are the base types:

- ***library*: collection of objects that can be used to build applications;**
- ***macro*: combinations of modules, macros, and connections;**
- ***module*: parameters and methods to implement objects that interface to C or C++ code**
- ***parameters*: maintain data usually operated on by methods. Data can be scalar, array, and hierarchical;**
- ***method*: objects that interface directly to a C function or C++ method.**

AVS/Express has special objects for representing scientific visualization datasets, referred to as an Express field. It is also possible for the user to create new objects or enhance and re-use existing objects in the system. This provides the facilities to extend the system creating new data structures or integrate application code as new modules.

An application is constructed using the Network Editor, shown in figure 9, which is similar in use to the other MVE systems. AVS/Express extends the visual programming paradigm by also allowing fine grain application development using the network editor e.g., creating/defining new objects, drilling down through layers of macro modules, integrating application code into modules.

3.3 Summary

Visualization systems, in particular MVEs, have been in existence since the early nineties and have become mature for producing visualization applications [23][24][25][26].

Current research is looking towards the integration of these systems with existing and emerging technologies such as parallel processing [27][28], virtual reality, World Wide Web (WWW) [29] and Java [30].

Acknowledgments

The authors of the paper would first like to thank EPSRC (GR/K40390) for funding the Visualization in Parallel project as part of the Portable Software Tools for Parallel Architectures (PSTPA) initiative, under which some of this research has been carried out. The authors would also like to acknowledge the support of the staff in the Manchester Visualization Centre, Manchester Computing.

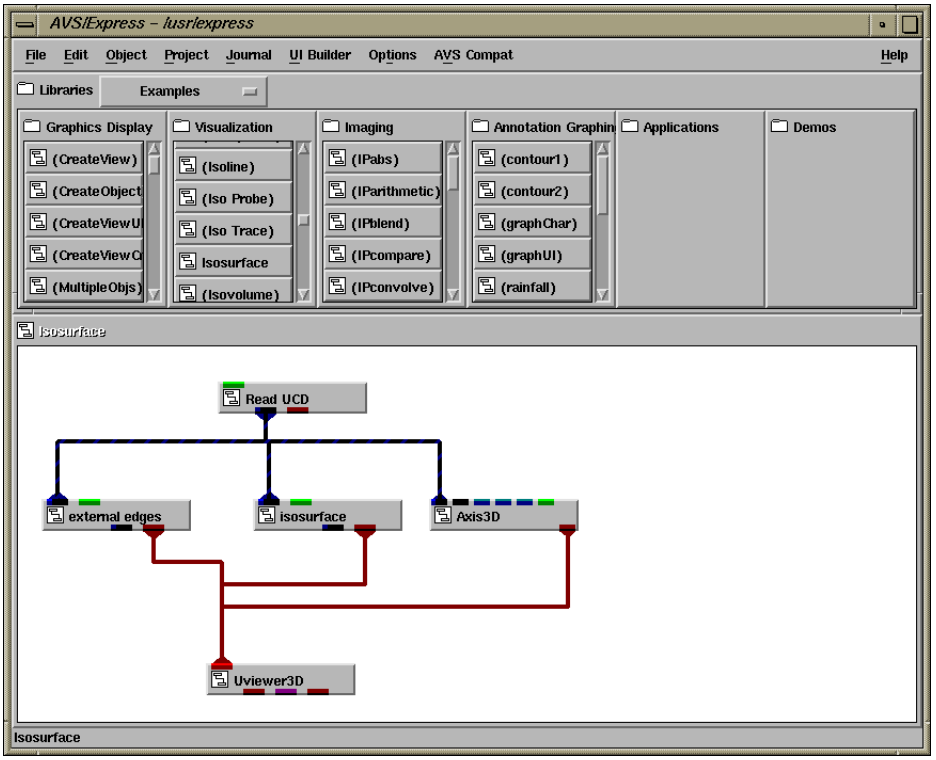


Fig. 9 AVS/Express - Network Editor

References

[1] Larkin S, "*Vipar: An Environment for the Development of Parallel Visualization Modules*", Transfer Report, Dept. of Computer Science, University of Manchester, 1997.

[2] Brodlie K W, Carpenter L A, Earnshaw R A, Gallop J R, Hubbold R J, Mumford A M, Osland C D, Quarendon P, "*Scientific Visualization: Techniques and Applications*", Springer Verlag 1992.

[3] McCormick B, DeFanti T A, Brown M D, "*Visualization in Scientific Computing*", ACM SIGGRAPH Computer Graphics vol 21(6), November 1987.

[4] Watson D, "*The State of the Art of Visualization*", Proc. Supercontinent Europe Fall Meeting Aachen, September 1990.

[5] Upson C, "*Scientific Visualization Environments for the Computational Sciences*", Proc. Compcon 1989, pp 322-327.

[6] Carpenter L A, "*The Visualization of Numerical Computation*", Eurographics Workshop on Scientific Visualization, 1991.

[7] Haber R B, Lucas B and Collins N, "*A Data Model for Scientific Visualization with Provisions for Regular and Irregular Grids*", Proceedings of IEEE Visualization '91, IEEE Computer Society Press 1991.

[8] Sombatsomsop N., Larkin, S., Wood, A. K., "*Real-Time Immurements of Temperature and Velocity Profile of Polymer Melts using Application Visualization System*", In Journal of UK UNIRAS User Group, Vol 2, Issue 4, February 1997, ISSN 1356-515X.

[9] Wernecke J, "*The Inventor Mentor*", Addison-Wesley ISBN 0-201-62495-8.

[10] Schroeder, Martin, Lorenson, "*The Visualization Toolkit: An Object Oriented Approach to 3D Graphics*", Prentice-Hall ISBN 0-13-199837-4.

[11] Upson C, Faulhaber T, Kamins D, Laidlaw D, Schlegel D, Vroom J, Gurwitz R, van Dam A, "*The Application Visualization System: A Computational Environment for Scientific Visualization*", IEEE Computer Graphics and Applications, 9(4), pp 30 -42, 1989.

[12] "*IRIS Explorer - Technical Report*", Silicon Graphics Computer Systems.

[13] Lucas B, Abram G D, Collins N S, Epstein D A, Gresh D L, McAuliffe K P, "*An Architecture for a Scientific Visualization System*", Proceedings of IEEE Visualization '92, pages 107-114, IEEE Computer Society Press, 1992.
[14] Rasure J, Argiro D, Sauer T, Williams C, "*A visual language and software development environment for image processing*", International Journal of Imaging Systems and Technology, 1991.
[15] Rasure J, Young M, "*An Open Environment for Image Processing Software Development*", SPIE/ IS&T Symposium on Electronic Imaging Proceedings, Vol. 1659, February 1992.
[16] Dyer D S, "*A Dataflow Toolkit for Visualization*", IEEE Computer Graphics and Applications vol 10(4), pp 60-69, July 1990.
[17] Brodlie K W, Brankin L, "*GRASPARC - A Problem Solving Environment Integrating Computation and Visualization*", Proceedings of Visualization '93, IEEE Computer Society Press, 1993
[18] Haber R B, Bliss B, Jablonowski D, Jog C "*A Distributed Environment for Run-Time Visualization and Application Steering in Computational Mechanics*", Invited paper at the Symposium on High-Performance Computing for Flight Vehicles, Washington, DC, December 7-9, 1992.
[19] Brodlie K W, Gallop J R, Grant A J, Haswell J, Hewitt W T, Larkin S, Lilley C C, Morphet H, Townend A, Wood J, Wright H, "*Review of Visualization Systems*", AGOCG Technical Report, no. 9, February 1995.
[20] Belien A C, "*Comparison of Visualization Techniques and Packages*", Stichting Academisch Rekencentrum Amsterdam, ISBN 90-72490-08-8 (1993).
[21] Vroom J, "*AVS/Express: A New Visual Programming Paradigm*", Proceedings of AVS 95, pages 65-94, Boston MA, 1995.
[22] Lord H, "*AVS/Express Product Family Overview*", Proceedings of AVS 95, pages 3-13, Boston MA, 1995.
[23] Cox J P, "*The Visualization of 3D Device Simulation: Using AVS with an Existing Simulator*", Proceedings of AVS '93, paper 66, Vol 2, 24-26th May 1993 Lake Buena Vista, Florida, USA.
[24] Chen P. C., "*A Climate Simulation Case Study*", Proceedings of IEEE Visualization 93, IEEE Computer Society Press, pp 397-401, 25-29th October 1993, San Jose, USA.
[25] Kim J J H, Dogan N, McShan D L, Kessler M L, "*An AVS-Based System for Optimization of Conformal Radiotherapy Treatment Plans*", Proceedings of AVS '95, pp 417-431, 19-21st April 1995, Boston USA.
[26] Post F J, van Walsum T, Post F H, "*Iconic Techniques for Feature Visualization*", Proceedings of IEEE Visualization 95, IEEE Computer Society Press, pp 288-295, 29th October 3rd November 1995, Atlanta, USA.
[27] Larkin S, Grant A J, Hewitt W T, "*Vipar Libraries to Support Distributed Processing of Visualization Data*", Proceedings of HPCN '96, April 1996.
[28] Larkin S, Grant A J, Hewitt W T, "*A Data Decomposition Tool for Writing Parallel Modules in Visualization Systems*", Proceedings of Eurographics UK '96, March 1996.
[29] Wood J, Brodlie K W, Wright H, "*Visualization over the World Wide Web and its Application to Environmental Data*", Proceedings of IEEE Visualization 96, IEEE Computer Society Press, pp 81-86, 1996.
[30] Jern M, "*Information Drill-down using Web tools*", Advanced Visual Systems seminar, 1997, `http://www.uniras.dk/info/seminars/Drilldown.htm`