Review of Visualization Systems

Advisory Group on Computer Graphics Technical Report

K. W. Brodlie, J. R. Gallop, A. J. Grant, J. Haswell,
W. T. Hewitt, S. Larkin, C. C. Lilley, H. Morphet,
A. Townend, J. Wood, H. Wright

Contents

Review of Visualisation Systems

Review of Visualisation Systems

Chapter 1: Overview

Review of Visualisation Systems

2nd Edition

Preface

This technical report arose from the work of a working group of the Advisory Group on Computer Graphics (AGOCG). The following people took part in the study, attended meetings and compiled this report:

K W Brodlie School of Computer Studies, University of Leeds
J Gallop (Chairman) Rutherford Appleton Laboratory, DRAL
A J Grant Computer Graphics Unit, Manchester Computing Centre
J Haswell Rutherford Appleton Laboratory, DRAL
W T Hewitt Computer Graphics Unit, Manchester Computing Centre
S Larkin Computer Graphics Unit, Manchester Computing Centre
P Lever Computer Graphics Unit, Manchester Computing Centre
C C Lilley Computer Graphics Unit, Manchester Computing Centre
H Morphet Computer Graphics Unit, Manchester Computing Centre
A Townend Computing Services, Keyworth, NERC
J Wood School of Computer Studies, University of Leeds
H Wright School of Computer Studies, University of Leeds

While every effort has been made to ensure that this document is accurate it is presented for information only. It is not guaranteed for any particular purpose and neither the editor nor the contributors nor their institutions nor the Advisory Group on Computer Graphics (AGOCG) accept any responsibility.

Published by the Advisory Group on Computer Graphics (AGOCG).

c/o Dr. Anne Mumford, Computer Centre, Loughborough University of Technology, Loughborough, Leics LE11 3TU, UK

Tel: 01509 222312, Fax: 01509 267477, Email: a.m.mumford@lut.ac.uk
URL: /agocg/

Review of Visualisation Systems

1.1 Introduction

One of the responsibilities of the UK Advisory Group on Computer Graphics (AGOCG) is to stimulate and support the effective use of computer-based visualization.

AGOCG therefore requested DRAL Rutherford Appleton Laboratory and the Universities of Leeds and Manchester and NERC Computing Services (Keyworth) to conduct a review of visualization software, to help potential and actual users make effective choices.

AGOCG conducted a previous evaluation of visualization software [15] and the Stichting Academisch Rekencentrum Amsterdam have also produced an evaluation report [10]. At that time, AVS was already becoming available on many workstations, IRIS Explorer and IBM Data Explorer were quite new, Khoros was available free but was restricted to handling images, and apE was undergoing a difficult transition.

One consequence of that evaluation was that AVS was made available to universities and other HEI's by CHEST on favourable site terms.

Why revisit this work after what is a comparatively short time? There are several reasons:

- Revisiting this work was envisaged at the time as it was realised that IRIS Explorer and IBM Data Explorer were new in 1991/2 and it was not possible to evaluate them effectively.
- Since that evaluation IRIS Explorer is now developed and supported by a UK company (NAG Ltd.), moreover a company with strong links with the UK academic community.
- Interest in PV-WAVE has grown in that time. This system is general purpose but at the same time its command language system provides an interesting contrast to the other systems which are in general of the data-flow type.
- Since that evaluation, AVS Inc has taken over the UNIRAS company - which means that CHEST deals with these two distinct companies are now with only one.

The purpose of the work is therefore to review the current market for visualization software as it exists in the UK and to provide information to the UK Academic community for its visualization needs. There are many potential systems and the review had to narrow the field.

The review does not include basic plotting systems. These are already understood in the community. Thus any system under review had to at least treat 3D data adequately. Therefore there had to be:

- good, interactive support for viewing 3D geometry, taking advantage of 3D graphics hardware now widely available to accelerate 3D viewing
- good support for visualizing 3D data - (a common pitfall here is the often quoted 3D plot, which is often a plot of z=f(x,y) a single valued function of 2 variables which we would therefore regard as 2D data)
- The system under review must be general purpose and extensible

Review of Visualisation Systems

1.2 Systems under review

The following systems which were reviewed are listed below (in alphabetical order):

- AVS: Commercial product from AVS Inc., supported in the UK by AVS/UNIRAS Ltd. AVS version 5.01 was reviewed and some comments about version 5.02 and the contents of version 6 have been made.

- IBM Data Explorer: commercial product from IBM Inc., supported in the UK by IBM(UK) Ltd. IBM DX version 2.0 was reviewed.
- IRIS Explorer: commercial product initially from Silicon Graphics Inc., bundled with Silicon Graphics workstations. It is also available from NAG Ltd. on other workstations and is supported by them (not bundled). It has been announced that the next version (3) will be unbundled from Silicon Graphics and available from NAG Ltd. IRIS Explorer version 2.2 was reviewed.
- Khoros: commercial product from Khoral Research Inc., distributed via a Free Access License. Currently the version 2.0 Developer's Release is only available via anonymous ftp. The Khoros Software, including a 7 volume manual set, Khoros 2.0 source code and the most commonly used binaries on CD-ROMs, will be distributed worldwide by Prentice Hall, Inc. in the second quarter of 1995. Khoros 2.0 (Beta release) was reviewed.
- PV-WAVE: commercial product from Visual Numerics Inc., supported in the UK by Visual Numerics Ltd. A related system IDL (http://sslab.colorado.edu:2222/projects/IDL/idl_ssl_) is also available in the UK from Floating Point Systems Ltd. PV-WAVE and IDL have a common origin but have since diverged. We do not cover IDL in this review because of the similarities with PV-WAVE. PV-WAVE version 4.2 was reviewed.

Review of Visualisation Systems

Chapter 2: Getting Started

Review of Visualisation Systems

2.1 Introduction

The following sections provide information for the novice user of each system. In some cases the manuals supplied with the system include suitable tutorial examples and these are subsequently referenced. Other systems have some simple examples included below. The notes assume that the particular system has already been installed correctly on the users target platform.

Review of Visualisation Systems

2.2 Application Visualization System (AVS)

2.2.1 Starting up the system

The command to run AVS is

```
avs
```

Typing this command should start up the copyright notice and result in one window down the left of the screen. If it does not, you might need some extra information in your environment variable PATH. Consult your System Manager for this.

2.2.2 Use a sample network

Place the mouse cursor (a pointing hand) over the panel "Network Editor" and click the left button once. This results in a small window on the left, the Network Control Panel, and a larger one to the right, the Network Editor (see figure 1). In AVS, complete networks do the work of visualizing data and are made up of modules connected together, each module carrying out a specific task. In the Network Editor, the top left hand section is called the main menu, and pressing a button here brings up suboptions. You should find on starting that the main menu option Network Tools is active. Next to the menu options is the module pallette, containing all AVS's individual modules, accessible by scrolling up and down the top of each column.

To use a sample network, place the mouse cursor over the Read Network suboption and click the left mouse button once. A file browser appears showing directories into which you can move by clicking the left mouse button once. Move into the examples directory and click once over bluntfin1.net. The file browser disappears and the network builds automatically. The network reads 3D data representing the flow of air over an aerofoil and displays the pressure in the window labelled AVS Display Pixmap. The image can be manipulated by placing the cursor in the window and pressing the middle mouse button whilst moving the mouse. For more information on manipulating scenes produced from AVS networks you should consult the AVS Tutorial Guide Ch 2: Geometry Viewer Tutorial. This tutorial covers the use of the Geometry Viewer module which is similar in functionality to the coupling of Render Geometry and Display Pixmap used in bluntfin1.net.

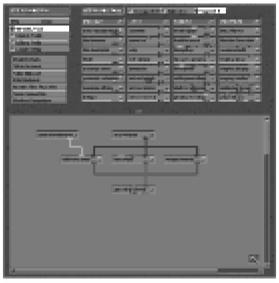


Figure 1. The AVS Network Editor

2.2.3 Build your own network

- First you need to clear the bluntfin network: select the suboption "Clear Network" and confirm.
- Scroll down the data input column until you see **Read Image**, place the cursor over the panel, press the left button and keeping it pressed, drag the module onto the blank workspace below. Release the mouse button and the module appears on the workspace: this combined operation is called module instantiation.
- In the same way, instantiate the modules **Crop** from the Filters column and **Display Image** from Data Output. You now have three modules in the workspace.
- To connect them together, move the cursor onto the coloured output port of **Read Image**, press the middle mouse button and while keeping it pressed, move to the input port of **Crop**. Release the button and the connection will appear in blue. Do the same between the output port of **Crop** and the input port of **Display Image**.
- Move to the Network Control panel and click on **Read Image**. Read Image's file browser appears and you should use the scroll bar and mouse to move into the directory Image, and select the file mandrill.x. The module **Read Image** now imports the data in the file mandrill.x, passes it to the **Crop** module to reduce the size of the data and thence to **Display Image** which shows the results, behind the Network Editor.
- To see the image, press "Close" on the main menu. To see the parameters of the **Crop** module, click on "Crop" in the Network Control Panel. To set a new value on any of the dials simply press with the left mouse button on the pointer, drag it to the required position and release the mouse button.
- To return to the Network Editor press the "Display Network Editor" button in the Network Control Panel. Before exiting AVS you should clear the network as before and then press "Exit" on the Network Control Panel. Then on the main AVS panel press "Exit AVS" and confirm.

2.2.4 Carrying on

The data you have used so far was AVS's own internal data format. You will want to import your own data into the system and for this you will need to use the AVS Data Interchange Application (ADIA). More information on ADIA is available in the AVS Applications Guide [1] Ch. 1.

To find out more about individual modules you can run other demos in AVS. These are explained in the AVS Tutorial Guide [6], Ch. 1

Online help on AVS can be accessed from within the system using the help buttons in both the Network Editor main window and the Network Control Panel. Paper manuals comprise the AVS Users Guide [8], Developers Guide [3], Tutorial Guide [6], Module Reference [4], Applications Guide [1], AVS5 Update manual [7], Chemistry Developers Toolkit Guide [2] and AVS Technical Overview [5].

2.2.5 AVS Training Materials

What do the materials contain?

The training materials contain slides and notes for two courses, the AVS Introductory [43] and Advanced course [42]. These were initially developed as part of the Advisory Group on Computer Graphics (AGOCG) Visualization Support Project at the Computer Graphics Unit, Manchester Computing Centre, University of Manchester.

There are also a number of data files and modules which support the practical exercises described in the course notes.

How do I obtain the materials?

The materials are available in postscript format along with the supporting data files and modules via anonymous FTP from the University of Manchester (ftp.mcc.ac.uk) or the International AVS Center (avs.ncsc.org).

For example to obtain the materials from the University of Manchester you would first type the following:

ftp ftp.mcc.ac.uk When you are connected to the server you should login in as anonymous and supply your email address as the password. To access the training materials you must move to the subdirectory pub/cgu/avs/avs_course and set the transfer to binary mode before getting the files:

```
ftp> cd pub/cgu/avs/avs_course
ftp> binary
ftp> mget *
```

Review of Visualisation Systems

2.3 IBM Data Explorer

A good way to start with Data Explorer (known as dx) is to follow the tutorial described in Appendix A of the User Guide [26].

The tutorial is in a number of separate parts - each can be followed independently of the others. It makes use of some files which should have been installed with the system. If these cannot be found, it is possible that the path names are different on your system - consult your system manager.

2.3.1 Starting up the system

At the UNIX prompt, type

dx

You should see the message below as dx is started:

Starting DX user interface

2.3.2 Building a Network

Appendix A.2 in the User Guide contains a tutorial which shows how to construct a simple network that reads in data, generates an isosurface, and displays it. It also shows how to save the network to file, and deals with the control of parameters (in this case the isolevel).

One thing it does not tell you is how to correct an error in wiring up the network: to remove a connection, click on the destination port and drag away the connection before releasing the mouse button. Figure 2 shows the construction of a simple network.

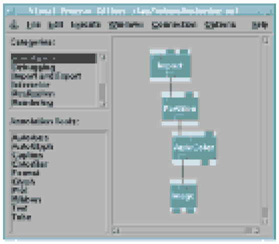


Figure 1: IBM Data Explorer Network

2.3.3 Running Example Programs

Appendix A.1 reads a previously defined network from disk, and explains how to execute it.

2.3.4 Data Input

There is no tutorial on data input and so the new user must refer to the appropriate manual pages. Chapter 4 of the User Guide deals with data import - it is well provided with examples in increasing order of complexity.

2.3.5 Finishing off

To leave dx, click on "File" on the menu bar of the VPE (the Visual Programming Environment) window and select "Quit".

Review of Visualisation Systems

2.4 IRIS Explorer

The manual IRIS Explorer User's Guide [28] has a very good chapter concerned with "Getting Started" for the first time user and an overview of the IRIS Explorer system can be found in pages xx et seq (before Chapter 1 in the User's Guide). The User's Guide also contains chapters on how to use the Map Editor to build IRIS Explorer Maps. The following sections provide suitable references into this manual where more detailed information can be found.

2.4.1 Starting up the System

To start IRIS Explorer you simply type:

explorer

If IRIS Explorer does not start up you may need some additional environment variables defined (see Appendix A of the User's Guide).

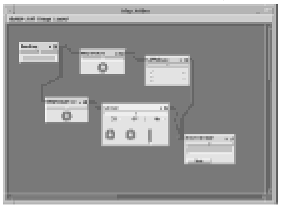


Figure 1: IRIS Explorer Map Editor

2.4.2 Running the Examples

This information can be found at the start of Chapter 1 in the IRIS Explorer User's Guide. In a standard installation the examples can all be found under /usr/explorer/maps and may be invoked from the Map Editor or directly when starting IRIS Explorer:

explorer -map mapname.map

2.4.3 Building your own Map

This is fully described in Chapter 2 and 3 of the IRIS Explorer User's Guide "Working with the Map Editor". An example of the Map Editor is shown in figure 3.

2.4.4 Data Input

Data is imported into IRIS Explorer using the DataScribe tool which is fully described in Chapter 7 of the IRIS Explorer User's Guide. As well as step-by-step instructions, there are samples in:

/usr/explorer/scribe

and data in:

/usr/explorer/data/scribe

2.4.5 Finishing off

To quit from IRIS Explorer you must select "Quit" from the Admin menu in the IRIS Explorer Map Editor window.

Review of Visualisation Systems

2.5 Khoros

The manual Khoros Getting Started Manual [33] should be consulted by the first time user. The following sections provide suitable references into this manual where more detailed information can be found.

2.5.1 Starting up the system

Before using any of the utilities in Khoros you must first configure your startup environment. This can be achieved manually or by running the utility kconfigure. Both of these techniques is described in Chapter 2 of the Getting Started manual. Once this has been done successfully you can try one of the Khoros applications e.g.,

kman kman
putimage -i image:ball

2.5.2 Building a Network

All information processing and visualization programs in Khoros are available via the visual programming language, cantata. Cantata is a graphically expressed, data flow visual language which provides a visual programming environment within the Khoros system.

There is a complete manual [39] dedicated to the Khoros application cantata and an example is shown in figure 4.

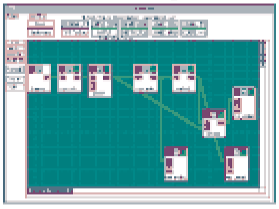


Figure 2-4: Khoros Application/Cantata

2.5.3 Running Example Programs

There are a number of different sample Cantata workspaces available in the Sampledata Toolbox. These may be invoked from the command line by running

Cantata -restore <workspace alias>

with any one of the following aliases:

workspaces:ColorArith1
workspaces:ComplexArith
workspaces:Express
workspaces:FilterEdge

workspaces:FilterMedian
workspaces:Formats
workspaces:Fourier
workspaces:Geometry
workspaces:GeometryElevation
workspaces:Render3dFFT
workspaces:Signal
workspaces:SimpleArith
workspaces:Wavelets
workspaces:5DHead

2.5.4 Data Input

In general, data access is performed through data services using one of the Khoros data models. Data services transparently supports a number of file formats. When a file is opened, data services checks the file to determine if it is one of the supported file formats. If it is, then the data contained in the file will be made available through the various segments in the data model. Applications which use data services can simply open up a file, and if the file is one of the supported formats, it will be able to access it. More information can be found in [36].

Review of Visualisation Systems

2.6 PV-WAVE

PV-WAVE CL is a particularly easy package to get started with, primarily due to having an excellent on-line help and demo system and good introductory documentation.

Once PV-WAVE CL has been started the supplied demo gallery is particularly comprehensive and useful, and the CL code for these examples is availble to the user. An example from a PV-WAVE demonstration is shown in figure 5.

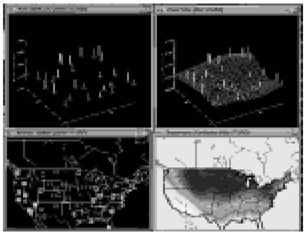


Figure 1: An example from PV-WAVE

There are three getting started manuals provided, although some are not obviously named. These are:

- Getting Started with PV-WAVE Point and Click Motif Version [54]
- PV-WAVE CL Applications Guide - Free Code and Sample Applications [52]
- PV-WAVE Command Language Overview [53]

The latter contains a very useful section, "An Interactive Session with PV-WAVE CL", which introduces the basic concepts of the command language via a simple tutorial.

Caution is required where these manuals refer to machine specific startup of PV-WAVE CL (environment setup and execution), as it is likely that these are site dependent.

Review of Visualisation Systems

Chapter 3: Functionality

Review of Visualisation Systems

3.1 Introduction

In this chapter we summarise the functionality of the visualization systems.

We consider this under the following major headings:

- Data Models: what are the fundamentals of the data that can be handled by each system?
- Algorithms: what visualization operations can be applied to the data, creating a graphical abstraction?
- Presentation: how can the graphical abstraction be presented and manipulated?

Review of Visualisation Systems

3.2 Data Models

It is common for publicity about a visualization system to highlight the 3D or 4D plotting capabilities that it provides. This can be misleading to the potential user for a number of reasons:

- The potential user is primarily interested in the characteristics of the data; the plotting capabilities are a means to the end of analysing the data.
- Emphasising 3D or 4D plotting capabilities may mislead the potential user into believing that the kind of data handled is also 3D or 4D when that is often not the case.

Here we use an approach which starts with the data to be visualized.

3.2.1 General introduction

After importing, the visualization system uses its native form to manipulate a dataset - the external form is not used except to access and import the original data again. The capabilities of a visualization system are limited by the data model offered by this native form. A very powerful data model allows a wide range of visualization possibilities. However it is also possible for this potential not to be realised either in the supported product or by 3rd party software - we review the actual coverage in the sections on Algorithms and Presentation.

Fuller descriptions of the principles of data models are given elsewhere ([11],[63],[18],[17]). Here we confine ourselves to a few practical issues and describe how these are addressed in the visualization systems. We cover type, dimensions and organization of the data.

The scientific investigator who needs visualization is concerned with the values of certain properties defined on a grid.

In simple cases there is a single property, but in many cases the investigator is interested in several properties on the same grid. So the fluid dynamics expert could be concerned with pressure, temperature and flow. This is the dependent data which may represent measurements or the results of computations. In some problems there could be many dependent variables.

Visualization systems normally deal with discrete data. Although it is possible to deal directly with mathematical expressions, this is a subject not well dealt with in general purpose visualization systems. Even if the investigator's mathematical model involves a continuum, for computation or measurement reasons the data has usually been sampled. Hence we refer to a grid, which consists of a set of nodes - the independent data - used to define where the dependent data is sampled.

The coordinate system of the grid often consists of cartesian spatial dimensions and also time. A collection of flow data is a good example of this, using either 2 or 3 spatial dimensions.

However the coordinate system need be neither cartesian nor spatial. Chemical reaction rates may depend on the concentration of the constituent substances, the presence or absence of a catalyst and other quantities such as temperature. For the investigator, these are the independent variables and the values of those quantities, at which chemical reaction rates have been measured or calculated, represent a grid. For some problems, there could be many independent variables.

3.2.2 AVS

This section outlines the basic range of data types which are provided for importing the following classes of data. The information provided is correct for the current version of AVS (AVS5) but an additional section has been added to provide details on the changes that will be provided by the next release of AVS (AVS6) in early 1995. AVS5 provides a number of datatypes:

- Scalars;
- Fields;
- Unstructured Cell Data (UCD);
- Geometry;
- Colourmaps;
- Molecule Data Type (MDT);
- User defined data;

Simple data types (character strings, integers, real, boolean)

The simple scalar types (character strings, integers, real, boolean) are all supported as AVS datatypes.

General 1D/2D/3D arrays of data

The field datatype allows the importing of general arrays of data elements (N dimensional array of M elements). The datatype allows the dimensions of the arrays to be described with an optional mapping/transformation being applied to the data elements of the array as they were being imported. These mapping types were split into three distinct types:

- *uniform*: the mapping is direct and implicit as no transformation of the data elements is applied;
- *rectilinear*: a mapping is applied to the dimensions of the arrays so an additional vector of coordinates is required for each dimension of the array during the mapping stage. This mapping could for example cater for logarithmic axis;
- *irregular*: an explicit mapping is applied to each data element as it is imported into the system. The mapping can also raise the dimensions of the data e.g., mapping a 2D array of elements into 3D space. During the mapping stage an explicit coordinate value is needed for every data element.

Node and Cell based data

The Unstructured Cell Data (UCD) type is aimed at providing support for associating data with discrete geometric structures. The data type consists of nodes and cells to form the overall UCD structure:

- *nodes*: a number of nodal positions in 2D or 3D space;
- *cells*: a number of logical connections are defined between nodes to form cells which hence form the overall model. AVS supports the cell types: point, line, triangle, quadrilateral, pyramid, tetrahedral, hexahedral.

Data can then be optionally associated at node positions, part way along a connection between two nodes (mid-edge) or with a complete cell or UCD structure.

Geometric data

The AVS Geometry data type is provided to support the display of 2D/3D graphical objects. The primitive datatypes provided are: disjoint lines, triangle and quadrilateral meshes, polyhedron definitions and spheres. There is also support for the definition of light sources, texture mapping and control over the camera parameters to perform clipping, depth cueing and perspective viewing of scenes.

Chemistry application data

The Molecule Data Type (MDT) is provided to support chemistry applications and consists of a number of objects arranged in a hierarchical structure:

- CHEMmolecule: root object, contains name and data;
- CHEMatoms: data for component atom;
- CHEMchemunits: molecular substructures;
- CHEMquantums: quantum chemistry;
- some support for the addition of user-defined data fields.

Colourmaps

The AVS colourmap data type is an arbitrary sized one-dimensional array of 4D vectors. The vector components are normalised floating point values which represent hue, saturation, brightness and opacity.

User defined data

The AVS system provides a variant of defining structures in the C programming language which encapsulates a number of the scalar data types into a user defined structure. The structure can also contain arrays of these scalar types.

Errors and undefined values

Currently there is no real support for errors or undefined values but this is provided for in AVS6.

AVS6

This section details the changes to the data types which will appear with the release of AVS6 in Q2 1995. In AVS5 different data types were required for different classes of data (i.e., 2/3D data, finite elements, geometric). AVS6 now has one unified data type which can be used to represent these classes of data within the same data structure. This removes the need for different visualization modules to perform the same function on different types of data e.g., there is now only one isosurface module for both field and unstructured cell data.

Another improvement is the facility to extend this base data structure with user defined fields. For example you could add patient attribute information to image scans and the modules within AVS would still recognise and process this extended data type as an image.

Other features include support for cylindrical, polar and spherical coordinate systems and the provision for users to specify NULL (undefined) data values. These underlying data types will also be supported by the visualization modules in AVS6.

With AVS6, there will be a number of new features to improve the handling of large data sets. These features include:

- data reference instead of data flow;
- direct rendering of large data sets;
- data chunking for processing large data sets a section at a time.

3.2.3 IBM Data Explorer

Data Explorer presents a unified data model based on the concept of a field. This concept brings together the grid positions used to define where the data is sampled, the connectivity and the property data. From the point of view of the user, a module may be defined on any kind of data where it is reasonable to do so - there is no artificial distinction between types of data.

Simple data types (character strings, integers, real, boolean)

DX does not really treat these as data types. Some modules do process all these data types - as standard C items - but are generally parameters to determine how to process data. A DX object can be a string, but better support is provided for strings and numbers that are associated *attributes* of a more complicated data structure (e.g. to associate the data with a name or number).

General 1D/2D/3D arrays of data

The easiest way to import array data into DX is to use the prompter tool. Figure 6 shows the interface to import an ascii file containing data values on a uniform 3D grid. The data file consists of data values only, and has a float value per line.

This generated the following general array format header:

```
file =./hipiph.ascii
grid = 64 x 64 x 64
format = ascii
interleaving = series-vector
majority = row
field = hipip
structure = scalar
type = float
dependency = positions
positions = 0, 1, 0, 1, 0, 1
end
```

The import module converts the data into DX objects, in this case, a Field with 5 components:

- *data* a generic array containing the data values.
- *positions* as it is a uniform grid this contains the start value for x y and z and the delta (spacing) between each value in x y and z.
- *connections* describes the connections and element type for the data.
- *box* defines the corners of the bounding box.

- *data statistic* contains statistics on the data.

which converts to the following DX format when imported with the import module and exported with the export module (the actual data is now binary and not shown here):

```
object 1 class array type float rank 0 items 262144 msb ieee data 0
attribute "dep" string "positions"
#
object 2 class gridpositions counts 64 64 64
origin 0 0 0
delta 1 0 0
delta 0 1 0
delta 0 0 1
attribute "dep" string "positions"
#
object 3 class gridconnections counts 64 64 64
attribute "element type" string "cubes"
attribute "ref" string "positions"
#
object "hipip" class field
component "data" value 1
component "positions" value 2
component "connections" value 3
attribute "name" string "hipip"
#
end
```

Node and Cell based data

Both node and cell based data are supported. To define node data, state the data is dependent ("dep") on the positions and for cell based say it is dependent on connections. Current cell types include: line, triangle, tetrahedron and cube (the latter is a general purpose shape defined by 8 vertices). Currently no mid edge data is supported.

It is possible to define a DX header to import node and or cell-based data into DX. For example the following lines have been extracted from the standard example dataset in /usr/lpp/dx/samples/data/irregular.dx data file:

```
# The irregular positions, which are 24 three-dimensional points.
object 1 class array type float rank 1 shape 3 items 24 data follows
0 0 0
0 0 1
0 0 2
0 2 0
0 2 1
0 2 2
1 0.841471 0
............
3 2.14112 2
# The irregular connections, which are 30 tetrahedra
object 2 class array type int rank 1 shape 4 items 30 data follows
10 3 4 1
3 10 9 6
10 1 7 6
.........
```

```
17 20 23 22
attribute "element type" string "tetrahedra"
attribute "ref" string "positions"
# The data, which is a one-to-one correspondence with the positions
object 3 class array type float rank 0 items 24 data follows
1 3.4 5 2 3.4
5.1 0.3 4.5 1 2.3
4.1 2.1 6 8 9.1
2.3 4.5 5 3 4.3
1.2 1.2 3 3.2
attribute "dep" string "positions"
# the field, with three components: "positions", "connections", and # "data"
object "irregular positions irregular connections" class field
component "positions" value 1
component "connections" value 2
component "data" value 3
end
```

In this example the data is included with the description but does not have to be.

User defined data

DX has a rich collection of data objects, it is also easy for the user to group a number of objects into a more complex structure (although the description header may be difficult to read).

Errors and undefined values

It is possible to specify the data is invalid, by including a component *invalid positions* which lists the locations, or *invalid components* to list *invalid cells* (where cells can be any cell, e.g. an FE cell or a cell from a uniform grid). This means holes can appear in both regular and irregular grids, or allows the user to define regions where another grid applies in a particular region (e.g. to replace parts of a grid with areas of greater detail). The standard example network /usr/lpp/dx/samples/programs/InvalidData.net shows that invalid data is not rendered - e.g. holes appear in the shaded object, contour lines do not pass through invalid points and streamlines stop at invalid points.

When using DX it is also possible to set a *valid* range of values for use by the autocolour/autogreyscale and specify colours to be used for out of range data (this could be used to highlight unsuspected invalid data). Also the Include module can aid the viewing of invalid data and data ranges. This module allows the user to select only those points which lie within or outside a given range. Non selected points can be marked as invalid (Cull).

Any comments

DX also has good support for *series* of data, such as time series. It can also handle the B-rep model for CSG as it is possible to have faces, loops and edges components in an object (e.g. the CSG examples in the public domain).

Although facilities for importing data by using a header description are very powerful in Data Explorer, it is also possible to generate filters (e.g. the FLUENT filter in the public domain) or import modules (e.g. the PHOENICS import module from the public domain). To develop code to import data both as filters and modules see Chapter 6, "Modules that Import Data", and Chapter 10, "The Data Model", of the Programmers Reference manual.

3.2.4 IRIS Explorer

IRIS Explorer - like AVS - also presents a distinction between major data types. As well as geometry, IRIS Explorer provides lattices, pyramids and parameters.

This section outlines the basic range of data types which are provided for importing the following classes of data. The Explorer data types covered are:-

- Parameter
- Lattice
- Pyramid - Finite Element and Molecule Pyramids.
- Geometry
- Pick
- User defined types

Simple data types (character strings, integers, real, boolean)

The Parameter data type handles all of these explicitly, with the exception of boolean (although those values can of course be handled as some other data type).

General multidimensional arrays of data

This form of data is handled by Explorer's Lattice data type. The Lattice is sub-divided into 3 parts:-

The values of nDim and dims must be the same for each section. There are 5 primitive types:- byte, short, long, float and double.

Lattices have 3 coordinate types:-

- *Uniform*: these can be multidimensional with nDataVar data variables of any primitive type at each node, but with no explicit coordinates. The shape is defined by the number of nodes in each direction with a uniform cell size. Explorer uses a bounding box to set the size and aspect ratio of Lattice coordinates.
- *Perimeter*: these can be multidimensional with nDataVar data variables of any primitive type at each node, and a list of coordinates sufficient to specify an irregularly spaced rectangular structure. It has a 1, 2 or 3 coordinate dimensions depending whether it is a 1D, 2D or 3D Lattice.
- *Curvilinear*: these can be multidimensional with nDataVar data variables of any primitive type at each node, and with M coordinate variables to describe the position of each data point. The dimensionality of the Lattice gives an implied connectivity to the data.

Colour maps are just specific instances of a 1D uniform lattice with either 1 data variable (grey scale), 3 data variables (RGB) or 4 data variables (RGB & opacity).

Node and Cell based data

The Explorer Pyramid data type holds two types of data: irregular or unstructured data and molecular modeling data. A pyramid consists of 3 main parts:

- the several layers of pyramidal data; for example, points, lines and faces. These values are collected in cxLattice structures.
- the relationship between these lattices, described by cxConnection. The cxLattice and cxConnection together form a layer.
- optional references to predefined pyramid elements which are stored in a dictionary, cxPyramidReference.

Irregular or Unstructured Data: In IRIS Explorer, the Pyramid data type is primarily used for finite element modelling and creating irregular grids and most Explorer pyramid modules handle only this kind of data. The basic layers to create a Tetrahedral grid include:

- the base lattice which holds the coordinate and data at each node.
- layer 0 (1D) which lists the connections between nodes to form edges.
- layer 1 (2D) which lists the connections between edges from layer 0 to form faces.
- layer 2 (3D) which lists the connections between the faces from layer 1 to form tetrahedral elements.
- layer 3 which lists the connections between the tetrahedrons of layer 2 to form complex structures composed of tetrahedral elements.

Molecular Modeling

Chemistry pyramids are used to construct objects according to information pertaining to molecule structures. This pyramid structure is more narrowly defined than that for finite elements and represents (in the 3D layer, layer 2) not a volume, but a ball-and-stick construction. There are a number of modules associated with this type such as **BallStick** which takes a molecule pyramid and outputs it, in Geometry, as a ball and stick construction, and **ReadPDB** which links in to the Brookhaven Protein DataBase and can search for named molecules and builds Chemistry Pyramids.

A number of chemistry modules have been written for IRIS Explorer at Imperial College; these are bundled with version 2.2 in the unsupported modules category. These include modules for reading data from a variety of chemistry packages (see section 4.4.1), for the creation of geometry representing molecules as ball and stick and ribbon constructions, and for the animation of molecular vibration modes.

Geometry data type

As noted above in section 3.4.7, geometry in IRIS Explorer is implemented using Inventor, an object-oriented 3D toolkit. Geometry objects which can be created via the IRIS Explorer geometry API include points, lines, polygons, spheres, cones, cylinders, triangle meshes, NURBS patches, octree volumes, and text; this interface also offers control over colours, lighting and geometric transformations. It is possible to also create and manipulate the geometry via the Inventor API.

Geometry is read into and written out of IRIS Explorer as an Inventor object, which makes it available to other Inventor applications such as SGI's Showcase presentation package. The Open Inventor file format is being promulgated by SGI as a de-facto standard for 3D scenes, and the number of Inventor applications, readers and translators is apparently on the increase.

Pick Data

This is a specialised data type that is used only in 2D and 3D modules. It is used to enable the user to pick, or select, a particular location in an image (2D) or rendering (3D) module display window and obtain information about it. For example, you may want to query part of a picture of an isosurface. You could find out the coordinates of a spot on the surface or the data value at that point. It is also possible to use the pick data type to create an object or move an existing object to a new location in a rendering module display window. As the user clicks on the position in the window the information goes to an upstream module which creates the new geometry and sends it to the Render module.

User defined data

IRIS Explorer provides a language, called the Explorer Typing Language (ETL) which can be used to build new user-defined data types that can be passed between modules in the same way as other IRIS Explorer data types. ETL was used to create the standard Explorer root types such as Lattice and Pyramid etc., which contain subtypes such as cxData and cxCoord. A user-defined root type can reference these Explorer subtypes in its definition. ETL is syntacticly very similar to C.

Once the user has defined the data type using the ETL, IRIS Explorer compiles it. This consists of the automatic translation of the ETL file into C and creates a library of accessor functions for the different components of the type, together with the appropriate header files. Information about the new type is automatically picked up by tools such as the Module Builder (used in the creation of new modules - see below) and the Map Editor (used in the creation of new applications). The user can then write their own modules to process data using the new type, manipulated via the new API which has been automatically generated by IRIS Explorer in the compilation of the type.

Errors and undefined values

Having read in an Explorer Lattice there are several ways to deal with erroneous values. Using **ScaleLatNode** it is possible to define a min and max for the data and then either Clamp, stretch or threshold the erroneous values. Clamping fixes the value to either the min or the max value, stretching rescales the data to fit the parameter min and max and thresholding produces a binary array where 1.0 is inserted for a correct value and 0.0 for an error. This can be used later to take different courses of actions for correct and error values.

3.2.5 Khoros

Khoros actually provides two data models, a polymorphic data model and a geometry data model. These data models are implemented within the Khoros data services libraries. All Khoros data processing and visualization routines are written to operate on these data models via data services. The low-level functionality of data services give these operators the ability to operate on data independent of data type, size, and file format.

Polymorphic Data Model

The polymorphic model is so named because it is capable of storing data from several different domains. By capitalizing on the commonality of data interpretation across these different domains, the polymorphic model facilitates interoperability of data manipulation routines. In other words, processing routines which use the polymorphic data model will be able to process data objects containing anything from signals to images and from to volumes to animations.

The polymorphic model consists of data which exists in three-dimensional space and one-dimensional time. You can picture the model most easily as a time-series of volumes in space. This time-series of volumes is represented by five different data segments. Each segment of data has a specific meaning dictating how it should be interpreted. Specifically, these five segments are value, location, time, mask, and map. All of these segments are optional; a data object may contain any combination of them and still conform to the polymorphic model.

The value segment is the primary data segment, consisting of data element vectors organized implicitly into a time-series of volumes. The value data may be given explicit positioning in space and time with the location and time segments. The remaining two segments are provided for convenience. The mask segment is used to mark the validity of each point of value data. The map segment is provided as an extension to the value data; the value data can be used to index into the map data. Figure 7 provides an overview of the Khoros Polymorphic Data Model.

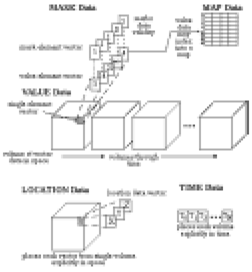


Figure 1: An overview of the Khoros Polymorphic Data Model

Value Segment

The value data segment is the primary storage segment in the polymorphic data model. Most of the data manipulation routines are specifically geared toward processing the data stored in this segment. In an imaging context, the individual pixel RGB values would be stored in here. In a signal context, regularly sampled signal amplitudes would be stored here.

The value segment consists of a time-series of volumes where each volume is composed of element vectors. Each element vector is composed of a number of value points. The size of the value segment is determined by the width, height, and depth of the volume, by the number of volumes through time, and by the number of points in the element vector. This makes the value segment, and the polymorphic data model, inherently five-dimensional.

Location Segment

The value points in the value segment are stored implicitly in a regularly gridded fashion. Explicit location information can be added using the location segment. If the value data is irregularly sampled in space, the explicit location of each sample can be stored here. Specifically, the information stored in this segment serves to position each the value data in explicit space. Note that the location data only explicitly positions a single volume; the position then holds for each volume through time.

The location segment consists of a volume of location vectors. The width, height, and depth of the volume are identical to the volume size of the value segment. Different location grid types are also supported. A curvilinear grid allows for explicit locations to be specified for each vector in the value data. A rectilinear grid allows for explicit locations to be given for the width, height, and depth axes. A uniform grid allows for explicit location corner markers to be specified.

Time Data

Explicit time information can be added using the time segment. If each volume of value data is irregularly sampled in time, an explicit timestamp for each volume can be stored here. This is useful in animations where each frame of the animation occurs at a different time.

The time segment consists of a linear array of timestamps. The number of timestamps matches the time size of the value segment.

Mask Data

The mask segment is available for flagging invalid values in the value segment. If a processing routine produces values, such as NaN or Infinity, these values can be flagged in the mask data so that later routines can avoid processing them. A mask point of zero is used to mark invalid value points, while a mask point of one is used to mark valid value points. The mask segment identically mirrors the value segment in size; there is one mask point for each value point.

Map Data

In cases where the value data contains redundant vectors that are duplicated in different positions, the map segment may be used. The value vectors are replaced with values which index into the map; the map then contains the actual data vectors. In this sense, the map is an extension of the value segment.

The map segment consists of a number of width-height planes. The values from the value segment map into the map height indices. The map vector runs along the map width. A simple map would consist of just a single width-height plane; a more complicated map would have a width-height plane for every depth, time, and element plane in the value segment. This provides a great deal of mapping flexibility. For example, every plane in a volume or every image in an animation could have a separate map.

Geometry Data Model

The geometry data model supports the storage and retrieval of a number of standard geometric primitives, such as spheres, triangles, and lines. Other non-geometric primitives such as octmeshes and textures are also supported.

The geometry data model is centered around a primitive list. This list is able to store any combination of geometric primitives such as spheres or polylines. Each geometric primitive consists one or more different types of data, suchas location data and color data. The types of data required depend on the primitive; all primitives have location and most have color while only some have radii or normals. Most data is explicitly provided, although colors may be provided indirectly via a colourmap. Quadmesh and octmesh primitives, which are not illustrated here, are also available. These mesh primitives are overlaid on top of the polymorphic data model. Thus, from the point of view of the polymorphic data model, a quadmesh will appear to be an image, and an octmesh will appear to be a volume.

General 1D/2D/3D Arrays of Data

The polymorphic data model can be used to represent up to five dimensional arrays of data complete with auxiliary information such as explicit location and time data, validity data, and map data.

Node and Cell Based Data

Explicit nodes are provided for vertices or cells in the Geometry data model, however the connectivity is implied by the geometry primitive which the data represents. There is no support for arbitrary connectivity.

The polymorphic data model allows an explicit location to be assigned to every data vector in the Value segment, thus sparsely distributed node data can be represented. Connectivity between the nodes can be specified only through the implicit organization of the Value data. There is no support for explicit connectivity between arbitrary nodes.

User Defined Data

The data services provides library calls for manipulating a generic abstract data object containing arbitrary segments of any dimensionality. While new data models could be constructed using this, it is not recommended as no existing processing operators would be able to operate on the data. In general, if the data can be expressed in a five-dimensional space, it is best to use the polymorphic data model.

Errors and undefined values

The Mask data can be used to specify the validity of the data (undefined data). There is no explicit mention of support for errors within data values but the application could simply allocate a portion of the data vector to support this feature. Even a combination of the mask and value data could be used for indicating if data has an error component associated with it.

3.2.6 PV-WAVE

In principle, PV-WAVE's data model could be as general as its command language. This includes arrays and structures, which in combination allows a very flexible data definition. In practice the data is confined to what can reasonably be handled by the plotting procedures. To give an example, although it is possible to defined unstructured data using PV-WAVE's data types, no procedures to plot such data are available. More detailed information can be found in Chapter 4: Data Import section 4.6.1

Review of Visualisation Systems

3.3 Algorithms

3.3.1 Introduction

This section deals with the algorithms which can be used to visualize a particular data set. It is structured according to the following scheme:

- first, we characterise the nature of the data - this effectively gives us classes of problems, such as scalar fields in 3D etc., (see section 3.3.2);
- next, we review briefly interpolation techniques - this gives us a means of creating a model from the data that is defined everywhere (see section 3.3.3);
- then, for each of the classes, we describe some of the possible visualization techniques (see section 3.3.4 to section 3.3.7);
- finally, we look at the availability of the different techniques in each system (see section 3.3.8 to section 3.3.12).

3.3.2 Classification

In visualization, we begin with a set of data. This will come either from observation (as in satellite recording or medical scanner) or from simulation of some physical process (as in CFD). Either way, the data is a sample from some underlying field - which we do not know, but which we are looking to visualization to help us understand. Thus a fundamental step in visualization is the creation of an empirical model, guided by the scientific investigator's understanding of the underlying field.

When selecting a visualization algorithm, it is useful to first consider the nature of the underlying field. What are the dimensions of the independent variables? What are the datatypes of the dependent variables, and so on.

Indeed we shall use the nature of the underlying field as our means of classifying the visualization algorithms. We use a subset of the classification derived at the AGOCG Visualization Workshop in 1991 [11].

The letter E represents the entity to be visualized, and a subscript denotes the dimension of the independent variables. Thus E_3 represents an entity defined over a 3D region. Time is treated specially - so in the above example, if the entity varies over time, we write $E_{3;t}$.

The type of dependent variable can be scalar (S), vector of dimension k (V_k) or tensor (T_{kk}), and this is written as a superscript.

So a vector field over 3D is written as: $E_3^{V_3}$.

3.3.3 Interpolation

Interpolation is the process by which the empirical model is created from the data.

There are a variety of methods and a good reference is the book by Lancaster and Salkauskas [44]. Here we give only a brief summary.

Consider first just E_1^S . We shall be given a set of points $(x_i, f_i), i = 1, 2, \ldots, N$. The interpolation problem is to construct a model F(x) which matches the data at the given points.

The simplest method is *nearest neighbour*, in which the value of F is taken to be the f-value of the nearest datapoint to x. Notice two features: it is very quick; but F(x) has a discontinuity midway between each datapoint.

Continuity can be improved by *linear interpolation,* in which F(x) is a linear function (that is, a straight line) between data points. This requires some computation so is slower, but F(x) is now continuous. Note however the slope of F(x) is not continuous, so if we know the underlying entity is smooth, then our recreation of it may be misleading in this sense.

Slope continuity (continuity of the first derivative of F(x)) can be achieved, but at the expense of more computation. A usual method is to estimate the slopes at the data points (for example, as the average of the slopes of the lines to adjacent data points) and then to fit a cubic function in each interval between data points. There are a variety of techniques for doing this *piecewise cubic interpolation* [12].

Second derivative continuity can be achieved, at even more computational expense, using *cubic splines*.

There are analogs of these methods in 2D and 3D. Consider first data on a rectilinear grid. The nearest neighbour extends in an obvious manner - it is quick, but discontinuous as in 1D.

Linear interpolation extends to *bilinear (2D)* or *trilinear (3D)*. Bilinear interpolation proceeds thus: find the grid square of interest; use linear interpolation in x for both extreme values of y; use these two calculated values in a further linear interpolation step in the y-direction. Trilinear is an obvious extension to 3D. These give continuity of function value, at extra computation expense. Hill [21] describes computational aspects of linear, bilinear and trilinear interpolation.

It is important to note that the bilinear interpolant in 2D is deceptively complex to visualize: it is a curved surface with contour lines which are hyperbolic. Similarly, the trilinear interpolant in 3D is likewise complex: surfaces of constant value are hyperbolic in nature. For this reason it can be useful to split the rectangles into triangles, or cuboids into tetrahedra, and fit simpler interpolants in each triangle or tetrahedron. These have straight line contours, or planar surfaces of constant value, respectively.

Piecewise cubic interpolation extends to *piecewise bicubic* and *piecewise tricubic*; these provide first derivative continuity, but are relatively rare (especially tricubic) on account of the computation involved.

Suppose now the data does not lie on a rectilinear grid. A variety of techniques have been suggested for scattered data, and a good recent review is by Foley and Nielson [16]. A very simple and reliable method is *multiquadric* (MQ) interpolation. The MQ interpolant is continuous in all derivatives, and is defined (in 2D) as:

$$F(x,y) = \sum_{1}^{N} a_i B_i(x,y)$$

where

$$B_i(x,y) = \left(R^2 + d^2\right)^{0.5}$$

with R a constant and d the distance of (x,y) from the ith data point (xi,yi). The coefficients ai are found by solving the N linear equations:

$$f_i = F(x_i, y_i)$$

The extension to 3D (and indeed higher dimensions) is straightforward.

Another good method is the *quadratic Shepard's method.* This has the form:

$$F(x,y) = \sum_1^N w_i(x,y)\, L_i(x,y)$$

where Li(x,y) is a quadratic function constructed to be a good approximation to the underlying function near the corresponding data point (xi,yi), and wi(x,y) is a weighting based on the distance of the interpolation point (x,y) from the corresponding data point. Again the extension to 3D is straightforward.

Another approach is to construct, in 2D a triangulation of the data, or in 3D a tetrahedral decomposition. There are well known algorithms for this: the Delaunay triangulation has optimal properties in terms of avoiding long, skinny triangles - traditionally thought to be a good thing. (Note however that views on this are changing - data dependent triangulations that align the triangles with features of the data, rather than across them, can give superior results in practice - see the paper by Dyn et al [14])

The triangulation is useful because local interpolants can be fitted within each triangle. Again these can be linear to give function value continuity, or higher order to give slope continuity.

There are many other techniques for interpolating scattered data, and the reader is referred to [16] for detail. Renka [60] gives an efficient implementation of the quadratic Shepard method.

3.3.4 Algorithms for scalar data over 3D

E_3^S

A large class of applications require the visualization of a scalar field over a 3D region. For example, data from medical scanners, temperature or pressure data from CFD.

There are essentially two approaches:

- *Surface Extraction:* In this approach, a 2D surface of interest is extracted. The surface can be extracted as a 2D planar *slice* through the 3D region, with 2D algorithms being used to display the data. Alternatively, the surface can be extracted as an *isosurface,* comprising all points with the same value of the scalar field. The slice therefore extracts in the space of the independent variable; the isosurface extracts in the space of the dependent variable.

  Of course, these techniques display only a 2D subset of the data. However, the third dimension can be visualized through animation - in the case of slicing, the cross-section can moved in time through the 3D region; in the case of isosurfacing, the isolevel can be moved in time from minimum to maximum.

  Note that in the case of an isosurface, it is possible to visualize a second scalar field over the extracted surface, by assigning colour values to the surface corresponding to the second scalar field.

- *Volume Rendering*: In this approach, the entire 3D volume is visualized. This is achieved by mapping the scalar value to colour and opacity, to form a coloured jelly-like *material* which can then be rendered. This allows the interior of the volume data to be inspected, and varying the classification of data by colour and opacity enables different features of the data to be extracted. For example, in an extreme case, an isosurface can be extracted by classifying all values greater than a threshold to be opaque.

We now look at these approaches in greater detail:

Surface Extraction

Slicing

This is relatively elementary: the slice can be positioned anywhere in the volume, a special case being the *orthogonal slice* which is perpendicular to one axis. A typical visualization technique on the slice is the image display, in which each pixel is coloured according to the scalar value at the corresponding position on the slice; but other 2D techniques are possible (e.g. contour lines). The 2D algorithms are described in section 3.3.6.

Isosurfacing

The extraction of isosurfaces has been studied in detail by researchers over the last few years. An excellent review article has recently appeared [49].

The best known algorithm is *marching cubes* [47]. This assumes data defined on a set of cubical cells, and constructs a polygonal approximation to the isosurface on a cell-by-cell basis. The extraction uses linear interpolation along edges of the cube cells, and a simple strategy to determine the surface across faces and in the interior. This strategy needs some care however, as topological ambiguities can occur, and as a consequence there can be holes in the resulting surface. Ning and Bloomenthal [49] explain how this can occur, and the remedies which can be taken. Certainly a good algorithm will contain some robust disambiguation strategy.

One such strategy is to decompose each cell into a number of tetrahedra. If this is done carefully - again see Ning and Bloomenthal - then linear interpolation along edges is sufficient to uniquely define a piecewise linear interpolant within each tetrahedron, and hence an unambiguous surface over the entire volume. This is known as *marching tetrahedra*.

Other strategies are possible, and some are more efficient than marching tetrahedra because they generate fewer triangles.

If the data are not given on a structured mesh, then there are techniques to construct a tetrahedral mesh from the data

Marching tetrahedra can then be used on this mesh.

A common feature of all algorithms is the generation of a set of triangles which approximate the isosurface. These are passed to a geometry renderer for display. For lighting and shading, it is important to calculate surface normals at the triangle vertices, and different strategies are possible:

- Calculate the gradient of the scalar field (which is in the direction of the surface normal) at the data points - for cubical cells, central differences can be used. Linear interpolation along edges will give estimates of the normal at the triangle vertices.
- Calculate the normal at a vertex as the average of the normals to the triangles which share that vertex.

Either strategy will generate normals which can be used for Gouraud or Phong shading to create a visually smooth isosurface.

As mentioned earlier, the colour of the isosurface may be assigned by some transfer function of a second scalar field.

Volume Rendering

In volume rendering, the intent is to display a representation of the entire 3D volume data - rather than extract a surface. The data is modelled as a coloured jelly substance, of varying opacity or transparency. It involves the following two initial steps:

- *Colour Classification*: The range of scalar data values are mapped to a range of colours, using some transfer function.
- *Opacity Classification*: The range of scalar values are mapped to a range of opacity values, using again some transfer function.

This classification step is under the control of the user, and is the key to successful visualization. As an example, consider a medical application where the scalar data values represent density of tissue. Then a certain range of data values will be known to correspond to bone, say, other ranges to skin, fat etc., and these can be mapped to specific colour and opacity values. Other data values may be *fuzzy*, for example, maybe skin, maybe fat, with certain probabilities; these are mapped to colour and opacity values intermediate between those for definite skin and fat. This ability to handle fuzzy classification is an important aspect of volume rendering. There are two major approaches to rendering the jelly substance: direct ray casting and splatting which are described in section 3.4.2.

3.3.5 Algorithms for vector field over 3D

This is the type: $E_3^{V_3}$. The field is defined by position only, that is, it is time-invariant or steady.

There are a variety of techniques for the display of vector field data. For a good review, see [50] and [22]. We summarise the main techniques here:

- *Arrows*: A glyph showing the flow speed and direction, usually as an arrow of variable length, is drawn at a given set of points. One may choose to draw the glyph at some or all data points, in which case the data provides sufficient information. Alternatively, one may wish to draw the arrows at user-specified positions, in which case interpolation will be needed.

  There are serious perception problems when this technique is used in 3D. It is more successful when a 2D surface is extracted, and the arrows are shown only for data points on the surface - indeed it is often effective to draw only lines without an arrowhead giving a spiked appearance (hence the term "hedgehogs"). Some success has been obtained in 3D by using 3D glyphs.
- *Particle Advection*: This is based on a traditional experimental technique for flow visualization. A light-emitting particle is released into the field and its progress is filmed over a specified time interval. In experimental flow visualization, this is commonly called a *pathline*.

  In practice, this visualization technique is implemented as follows. An initial particle position is specified: say (x_0, y_0, z_0), at time $t = 0$. Then its progression is governed by the system of Ordinary Differential Equations (ODEs):

  $$\frac{dx}{dt} = v_x \qquad \frac{dy}{dt} = v_y \qquad \frac{dz}{dt} = v_z$$

  with the initial values:

  $$x(0) = x_0 \qquad y(0) = y_0 \qquad z(0) = z_0$$

Because the flow is steady, the velocities vx, vy and vz are dependent only on position, not on time. The solution of these equations is done numerically, and the quality of the ODE solver will determine the accuracy of the resulting path. Euler's method is simple and quick, but generally inaccurate. Better results will be gained by a Runge-Kutta method. There are a family of R-K methods, with differing orders of accuracy. Second order R-K is commonly used.

The solution will also need interpolation to calculate the velocity at the points required by the ODE solver, and the different algorithms described earlier can be used.

The presentation can be as the animated movement of the particle, or the trace of the particle path can be rendered. In this latter case, the path is identical to a stream line as explained below.

- *Stream lines*: These are lines which are everywhere tangential to the velocity field. The speed of flow is also indicated by the relative closeness of the streamlines. The method is better in 2D than 3D, some depth cueing being needed to assist perception.

  For steady flows, particle paths and stream lines are identical, and so the computation process is identical. The user will typically provide a set of starting positions from which the stream lines are generated.
- *Stream ribbons*: This is a variation in which a pair of adjacent stream lines are considered to be the edges of a ribbon, and rendered as such. This is useful to display twist in a flow field [23].
- *Stream surfaces*: This is another variation, where a number of adjacent streamlines are connected into a polygonal surface [24].
- *Time lines*: This is based on the experimental technique of releasing a line of hydrogen bubbles into the flow at a given time. The position of this line at a number of successive time intervals is then displayed - as an advancing front.

  In practice, this can be computed by selecting initial positions on a line in the flow. These are imagined as the start points of particle paths, and the positions of the particles at regular time intervals are recorded - interpolation through the positions at any time gives the corresponding time line.
- *Streak lines*: In experimental flow visualization, this is defined as a line composed of those particles which have passed through a specified location in a specified period of time. In computational visualization, the term is used somewhat loosely. Sometimes it is used to refer to the release of a *line of particles* whose position is tracked through the flow; in a steady flow, these particle tracers will follow stream lines. On other occasions, it is used to indicate particle paths through a time-varying field. When we look at different systems, we shall try to indicate the meaning of the term in the context of the system.
- *Topology Methods*: This approach identifies the critical points where velocity vector is zero - and classifies them as sources, sinks, etc. The connection of critical points divides the space into regions of common flow properties. This is described in [20]. This has been implemented in the FAST [9] visualization environment developed at NASA Ames Research Center, but to our knowledge is not available in other visualization systems.

It should be noted that it is often useful to calculate scalar quantities from the vector field, for example:

- *magnitude of velocity*
- *magnitude of vorticity*

These scalar quantities are of type E_3^S and can be visualized as such.

3.3.6 Scalar field over 2D

There are three approaches:

- *Line extraction*: In this approach, a slice is taken in a 1D subspace. The extraction can be in the space of the independent variables, giving us a cross-section through the 2D region, and a 1D technique can then be used - for example, a graph. Animation can provide a view of the entire 2D region, by sweeping the slice through the space. The slice can be orthogonal to an axis, or at an arbitrary angle.

Alternatively, the line can be extracted in the space of the dependent variable, giving a line (an isoline or contour line) along which the scalar field has constant value. Several isolines can be drawn, within a range of values of the scalar function, so as to give a view of the whole region. A variation is to shade with constant colour between isolines.

- *Surface drawing*: In this approach, the scalar value is mapped to a third spatial dimension, giving a surface in 3D space which can then be rendered as a geometric object. Note this also allows a second scalar field to be displayed, by colouring the surface according to the corresponding value of the other field.
- *Image display*: In this approach, the 2D region is mapped to an area on the display, and each pixel coloured according to the value of the scalar function - using some suitable transfer function.

These approaches are described in more detail:

Line Extraction

- *Slicing*

  This is elementary: typically values along the slice direction are extracted, and a simple graph is drawn. The 1D interpolation techniques mentioned earlier can be used to *fill in* between the extracted samples.
- *Contouring*

  There is a vast literature on contouring which has been a popular 2D visualization technique for geographers and other scientists for many years. A good review paper is by Sabin [61].
- If the data is defined on a regular grid, then there is a simple method (the analogue of marching cubes in 3D) in which linear interpolation is used to estimate the intersection of isolines and grid lines - with some strategy for determining the lines within grid cells. Ambiguities can again occur, so care is needed (see Sutcliffe [62]).

  Methods based on linear interpolation will have slope discontinuities as the isolines move between grid cells. For smoother lines, one needs a method based on bicubic interpolation - see Preusser [51].

  For scattered data, one approach is to create a triangulation of the data points. Within any triangle, the intersection of the isoline with the edges can be found. Joining the intersections with straight lines gives the isoline of a linear interpolant within the triangle.

  An alternative approach for scattered data is to use an interpolant to estimate the underlying field on a grid, and then use a gridded contouring method. It is often useful to enhance the appearance by shading the regions between isolines - indeed the isolines may be removed altogether. As with volume rendering, one needs a colour transfer function to associate colour values with values of the underlying field.

Surface Drawing

Another traditional graphics technique for 2D data is the *carpet plot* which shows 2D scalar data as a surface in 3D space - the value being mapped to the height axis. In earlier days this was drawn as a projection of a network of 1D curves parallel to the x and y axes, giving the carpet like effect. Modern implementations will draw this as a smoothly shaded surface.

It is possible to show a second scalar field by draping a colour shaded contour map over the surface - that is, one scalar variable is represented by height, the other by colour.

Image Display

This is a very simple technique in which the sample region is mapped to a corresponding region on the device, and the colour of each pixel is determined by the associated value, or interpolated value, at that point. Again a colour transfer function will achieve this.

3.3.7 Scalar field over 1D

This we mention largely for completeness: the conventional approach is to draw a graph, relying on one of the interpolation methods described in section 3.3.3 to fill in between data points.

3.3.8 AVS

Scalar field over 3D

Note: AVS has separate modules for different datatypes - ucd data is treated by a separate set of modules from AVS field data - field data can be uniform, rectilinear or irregular grids.

- *Surface Extraction*

  Slicing: The **orthogonal slicer** module takes a slice through a 3D scalar field, perpendicular to one of the axes.

  The **arbitrary slicer** module takes a slice at any orientation. The **thresholded slicer** module is similar, but any values outside a range are mapped to zero in the output slice.

  For ucd data, the slicing module is **ucd rslice**.
- *Isosurface*

  The **isosurface** module generates an isosurface from a 3D scalar field. The particular algorithm used is not described in the manual. Likewise the way that gradients are calculated is not described.

  The **ucd iso** module does the same for ucd data.

  The surface may be coloured by a second scalar field.
- *Volume Render*

  The **tracer** module creates a ray cast image from volume data. There are two interpolation methods: *voxel approximation* which is similar to nearest neighbour in that the voxels are deemed to have constant colour and opacity, but the value is taken from the upper left corner of the cell; and *trilinear* interpolation.

  The **cube** module is a general tool, adapted from SunVision, which has four options: texture - which shows texture-mapped exterior surfaces of the volume; maximum - which just projects the maximum voxel along each ray; ray cast and create surfaces - which render surfaces at different density levels by ray casting.

  The **xray** module is a fast volume renderer, giving orthographic views only of the data.

  The **volume render** module - the exact technique being used is not clear from manual - allows the rendering of volume and surface data in combination.

Vector field over 3D

Note: The AVS modules described below apply to static vector fields.

- *Arrows*

  The **hedgehog** module shows arrows at locations in the 3D volume - for AVS node data. These locations may be generated first by the **sampler** module, and can be: single point; points on line or circle; points on plane; points in volume; data points themselves.

  The arrows can be drawn with or without arrowheads.

  Note: A scalar field can be viewed in conjunction with the hedgehog by colouring the arrows according to the value of the scalar field at that point.

  The **ucd hog** module does the same for data defined in the AVS ucd data structure.
- *Particle advection*

  The **particle advector** module releases a grid of particles into the field. As with hedgehog, the initial sample of points can be generated by the **sampler** module. The particles can be displayed as a tracer of specified length, and batches can be released.

  The ODEs can be solved either by Euler's method, or Runge-Kutta (order of R-K method not specified in manual). Interpolation method is not specified.

  There is no standard UCD Particle Advector module but a public domain module **ucd_particle** makes use of the **ucd_streamline** module to display particles along each streamline segment.
- *Stream lines*

  The **stream lines** module generates stream lines for a vector field. The user can specify the initial sample of points from which the stream lines are drawn.

  The ODEs can be solved either by Euler's method, or Runge-Kutta (order of R-K method not specified in manual). Interpolation method is not specified.

  The **ucd streamline** module is similar for ucd data. The ODE method has choice between Euler, 2nd order R-K and 3rd order R-K.
- *Stream surfaces and ribbons*

  The **stream lines** module can optionally create a surface connecting the stream lines.

  The **ucd streamline** module can optionally create ribbons of specified width.

Scalar field over 2D

- *Contouring*

  The **contour to geom** module constructs isolines from a 2D scalar field for a specified threshold value. The particular technique used is not described.

  The **ucd isolines** module creates isolines on the exterior boundary of a UCD structure.
- *Image Display*

  The **colorizer** module converts data at each point of a scalar field to a colour.

Scalar field over 1D

AVS provides the module **AVS/Graph** (contained in AVS release 5.02 which is now available) for producing traditional 2D plots from 1D data. The module has been built using some of the Toolmaster-agX libraries and is an example of AVS and UNIRAS integration. The module provides a number of different representations of the data:

- line and curves
- scatter plots
- bar and area charts
- polar and pie charts

The module provides the usual control over parameters and annotation facilities: title, axes, limits, line/curve/marker styles etc.

3.3.9 IBM Data Explorer

Scalar field over 3D

- *Surface Extraction*

  Slicing: The **Slice** module extracts an orthogonal slice through a volume of data, and this can be passed to a 2D visualization technique.**Slice** extracts an n-1 dimensional slice from an n dimensional dataset. The nth dimension is removed, i.e. a Slice from a 2D dataset produces a 1D line, a Slice through a 3D dataset produces a 2D grid (with no third dimension).

  Slab does a similar job to **Slice** except it maintains the nth dimension of the data and allows the slab thickness to be controlled.

  MapToPlane allows users to define non-orthogonal slices by controlling both the position of and normal to the plane

  Isosurface: The **Isosurface** module creates an isosurface from a *scalar field*. The particular algorithm used is an implementation of marching tetrahedra [41].

  The gradient vector (for shading) can be generated internally (again how it does it is not specified), or it can be supplied explicitly. The **Gradient** module will calculate the gradient of the scalar field, and this can feed into the Isosurface for shading.
- *Volume Render*

  The **Render** module creates a volume rendering of volume data (alternative modules are **Display** and **Image**. The data is of the "scalar field" datatype. The algorithm used is specified in [48].

  Note that the module can handle a combination of volume data and surfaces.

Vector field over 3D

Static Field

- *Particle Advection, Streamlines, Stream Ribbons*

The **Streamline** module produces streamlines for static vector fields. It traces the flow of a particle released from a defined point. The tracing is done by proceeding step-by-step, each step being in the direction of the vector field at the current point; this direction is found by interpolation from the vector data. The user has control over the step length. (This would appear equivalent to Euler's method.)

There is a sophisticated set of controls. The starting points can be given as a list of points, or as a geometric entity - which could be an isosurface for example. The lines themselves can be drawn as lines, ribbons (by connecting to the **Ribbon** module), or 3D tubes (by connecting to the **Tube** module).

Time Varying Field

- *Streaklines*

  The **Streaklines** module produces streaklines for time-varying vector fields. This works as follows. Successive vector fields advancing in time are passed to the module. The streaklines are traced step-by-step as for streamlines, but the direction is calculated now by interpolation in space and time.

Scalar field over 2D

- *Contouring*

  The **Isosurface** module will create isolines from 2D data. A number of threshold values can be specified. The technique used is not described.

  The **Band** module, together with the **Autocolor** module, will create a visualization with coloured regions denoting ranges of values.
- *Surface View*

  The **Rubber Sheet** module can be used to create a surface view.
- *Image Display*

  The **Autocolor** module converts data at each point of a scalar field to a colour.

Scalar field over 1D

IBM Data Explorer provides a **Plot** module to provide traditional 2D graphics plot of 1D data. The module provides all the facilities to alter parameters and annotation features associated with the plots. These include: axes labels, title, axes style (linear and logarithmic), tickmarks and colour legends.

There are also some other related modules that can be used in conjunction with the **Plot** module to produce other 2D representations of data.

- **Histogram** regroups your data into a specified number of bins. The output of **Histogram** is a new field which are the bars on the histogram
- Connecting **Histogram** through **AutoColor** then **Plot** produces a colored plot of the data distribution.

3.3.10 IRIS Explorer

Scalar field over 3D

- *Surface Extraction*

  Slicing: The **Slice** module slices a uniform 3D lattice by taking regular samples on a cutting plane, which may be at any orientation. The output is a 2D lattice of the same datatype and number of channels as the input lattice.

  The **SliceLat** module takes a slice through a 3D lattice and outputs the coloured plane as geometry. The lattice cells are intersected with the slice plane, forming polygons. The polygon vertices are coloured by values interpolated from the lattice values.

  Other slicing options are provided by **Orthoslice**, which generates 2D slices which are oriented with the coordinates of the input 3D lattice, and **MultiSlice**, which gives geometry output from a number of slice planes.
- *Isosurface*

  The **IsosurfaceLat** module generates an isosurface from a scalar 3D lattice. The algorithm used in earlier releases of IRIS Explorer was based on Marching Cubes, although this has been updated in version 2.2.

  The gradient vector (for shading) can be generated internally from the gradient of the scalar field (the Smooth parameter "on"). Alternatively the renderer can generate vertex normals from the geometry of the triangular mesh (the Smooth parameter "off").

  The surface may be optionally coloured according to the value of another lattice. The colours used can be controlled by means of a colourmap, which may optionally be passed to the module.

  The **Contour** module generates a set of isosurfaces (i.e. 3D contours) in wireframe. The user has control over the minimum and maximum isolevels, and the number of contours to be calculated. They are generated by calculating contour lines in orthogonal 2D slices of the 3D dataset; the user can control the direction of the slices to be used.
- *Volume Render*

  The **VolumeToGeom** module The VolumeToGeom module uses a volume rendering algorithm to convert a 3D lattice into a geometry. The method used is the so-called hierarchical splatting algorithm of Laur and Hanrahan [45], which fills the space within the volume using screen-oriented planar shapes (or splats). The algorithm uses a small number of large splats in uniform regions of the volume, and fills more detailed areas with a lot of smaller splats. The user has control over the error tolerance associated with the subdivision of the volume, and also over the size and type of the splat, which gives interactive control over rendering time and quality. The geometry produced may be combined with other geometry in the rendering of the final scene (see section 3.4.7), unlike the **VolumeRender** module (below) which performs direct volume rendering on the input lattice.

  The **VolumeRender** module performs volume rendering on a 3D byte lattice. It offers the choice between two algorithms offering different speed and quality advantages, so the user can trade off between interactivity and quality. The "Transform" algorithm gives interactive rendering speeds for moderately sized datasets, while the "Slicer" algorithm produces high quality images with longer rendering times. In the latter case, the user has interactive control over the number of slices (and so, the quality of the final image).

Vector field over 3D

- *Arrows*

The **Vectors** module displays a 3D vector field for a 1D, 2D or 3D lattice having 1,2 or 3 channels (i.e. vectors having 1,2 or 3 components). Vectors are located at the lattice coordinate locations and point in the direction of the data vector field. The vectors may be optionally coloured using a scalar lattice of the same dimensionality and size. They may be displayed as lines, tubes (cylinders) or arrows (cylinders plus cones).

The **VectorGen** module is a simplified version of the same module - here, each vector is represented as a bi-coloured line segment.

- *Particle Advection, Streamlines, and Streaklines*

  The **Streakline** module calculates a streakline through a velocity field. It stops when either the new velocity is 0, or when the streak intersects the bounds of the velocity field. The user has control over how each iteration is calculated and displayed. The full solution can be shown or each iteration as it is calculated. The current particle path is shown in red with the old paths in green.

  Particle advection is provided by the module **ParticleAdvect** which was developed as part of the NCSA Pathfinder project (http://redrock.ncsa.uiuc.edu/PATHFINDER/pathrel2/explorer/ParticleAdvect/ParticleAdvect.html)

Scalar field over 2D

- *Contouring*

  The **contour** module generates contour lines for a 2D lattice (or a 3D lattice - see above). For lattices with several channels, the user can select the channel to be contoured. The lines are coloured according to an internal colourmap; this can be optionally overridden by passing another colourmap to the module.
- *Surface View*

  The **DisplaceLat** displaces the coordinates of one input lattice by the data values of the other. The two input lattices can be the same. This can be used to create a surface view of a 2D lattice. The user can control the amount of displacement interactively.
- *Image Display*

  The **LatToGeom** module creates geometry from a 1D or 2D lattice i.e., it produces lines or sheets, depending on the dimensionality of the input lattice. The user can control whether the geometry is produced as points, lines or (in the case of sheets) polygons. The geometry can be optionally coloured via an input colourmap.

  The **DisplayImg** module displays 2D lattices as images. IRIS Explorer also contains a number of image processing modules, which may be used to filter or modify the image before display. See section 3.4.7, below, for more details.

Scalar field over 1D

IRIS Explorer currently has two modules which support traditional 2D representations of 1D scalar data. These are:

- **Gnuplot**: This is an implementation of gnuplot 3.0. Some additions to standard gnuplot have been made to allow the plot/splot commands to extract data and coordinate information from the input lattice.
- **Graph**: This module plots an X-Y graph of one or more channels of data from a 1-D lattice. The Draw Mode selector allows the user to select between single channel plotting and drawing of all channels.

There is also a **Histogram** module which can be used in conjunction with the above.

IRIS Explorer release 3.0 will also have **NAGGraph** with more advanced features than **Graph**.

3.3.11 Khoros

All the image operators are written tooperate on width-height planes of the polymorphic model. If the depth, time, or elements dimensions of data object are greater than one, the operation is repeated for each width-height plane.

Scalar field over 3D

There is an isosurface operator in the Geometry Toolbox which produces an isosurface constructed of triangles. An orthogonal slicer operator is also available. Mapping operators are provided which can map the scalar values contained in the field into RGB-alpha values.

Vector field over 3D

There are currently no operators which directly address the visualization of vector fields, although it is possible to produce scalar fields from the vector fields using the various arithmetic operators found in the Datamanip Toolbox.

Scalar field over 2D

A two-dimensional field can be produced from scattered location points using the gridding operator found in the Geometry Toolbox. The orthogonal slicer in the geometry toolbox can be used to slice 1D lines from the 2D fields. Two dimensional fields are generally visualized as images or 3D plots, but this is done through various interactive applications.

There are numerous operators in the Image Toolbox which are general image processing operators and more information is given in section 3.4.8.

Xprism can also produce line, mesh, coloured contour and shaded 3D plots. There are also facilities to alter annotation and parameters associated with these plots within Xprism.

Scalar field over 1D

Xprism can produce a number of traditional 2D plots from 1D data. The Xprism application in the Envision Toolbox provides fifteen different 2D plot types and these include:

- line and scatter plots
- bar charts
- discrete plots

The Xprism application also provides full control over fonts, colours, axes, marker types, line types, titles etc.

3.3.12 PV-WAVE

Scalar field over 3D

- *Surface Extraction*

  Isosurface: The **SHADEVOLUME** procedure calculates an isosurface from data in cubical cell arrangement. The algorithm used is that described in [40]. It is said to be variation on marching cubes.

  The gradient calculation for shading is not described in the manual.
- *Volume Render*

  The **VOLREND** function in the Advanced Rendering Library creates a volume rendering from data in cubical cell arrangement.

Vector field over 3D

There is only an "arrow" capability for this visualization datatype.

- *Arrows*

  The **VECTORFIELD3** function in the Advanced Rendering Library plots vector glyphs at a set of points specified by user. Form of interpolation not specified in the manual.

Scalar field over 2D

- *Contouring*

  The **CONTOUR** procedure draws a contour plot from a uniform 2D grid, with very sophisticated control over the annotation and presentation. There are two methods: one draws all lines per cell then moves on to next cell; the other traces a line through all cells, then moves to next line. No further algorithmic details are given.

  The **IMAGE_CONT** procedure overlays a contour plot onto an image display. Bilinear interpolation is used in image display.

  The **SHOW3** procedure overlays image, surface view and contours.
- Surface View

  The **SURFACE** procedure draws a surface view, with sophisticated control over the annotation and presentation.
- *Image Display*

  See contouring above.

Scalar field over 1D

PV-WAVE has a number of specific view windows for producing 2D plots from 1D data:

- 2D line
- 2D scatter
- 2D histogram

Each view window provides control over numerous annotation and parameters associated with each view type.

Review of Visualisation Systems

3.4 Presentation

3.4.1 Introduction

Presentation is the computer graphics part of Visualization. It includes the rendering (typically to a computer screen) of images, geometric models, volumes and hybrids of these. It also encompasses direct interaction or manipulation, i.e. selecting and moving objects, probing, and so on. This section does not address rendering to other media such as PostScript, video, etc. which is covered elsewhere.

3.4.2 Rendering

Genuine 3D display devices do exist which allow you to walk around and see objects from different angles. However they are large, expensive, experimental and often have low spatial and colour resolution. In most cases Abstract Visualization Objects [19] are rendered to a 2D screen or, for stereoscopic vision, two screens - one for each eye. When mounted into moveable, trackable object such as a boom box or helmet one can once again appear to walk around the objects. This virtual reality is currently a more promising direction than true 3D displays and potentially has great application to visualization e.g., the NASA Virtual Wind Tunnel project [13].

Images

Rendering images typically involves a one-to-one mapping from image pixels to screen pixels. Panning is readily provided for in a windowing environment, and zoom is often provided by bilinear or bicubic interpolation.

Geometry

Rendering geometric models uses classic computer graphics principles, with hidden surface removal, local illumination models, shading, and transformation to a 2D image with perspective. The algorithms for doing this are well established and some may be implemented in hardware.

Volume

To render volume data, it is assumed that classification has already been done. Volume datasets which have had isosurfaces extracted produce a geometric model which is readily rendered as described above. A problem with surface extraction methods is that they do not allow views inside volumes; even if many semi-transparent shells are rendered, detail is lost between the layers. Rendering the volume data without recourse to intermediate geometric objects avoids these problems and also permits weak, fuzzy surfaces or gradual gradients to be rendered. There are two major approaches to rendering volume data: direct ray casting and splatting.

In *Direct Ray Casting* a ray is cast from the eyepoint through each pixel into the volume. Along each ray, regular samples of the colour and opacity are taken. As the volume need not be axis aligned with the viewing plane, this step involves interpolation, and can use any of the techniques described in the earlier section. The final colour for the pixel is obtained by accumulating the colour and opacity values at the samples. A classic paper on this approach is that by Levoy [46]. This differs from ray tracing in that rays are not reflected from the surface of objects; all rays are perpendicular to the image plane.

A variation, optimized for speed rather than accuracy, is *Maximum Intensity Projection*. This is where the volume has axes aligned with the view plane and the highest value along a ray is used as the colour of the pixel - there is no accumulation.

In *Splatting, rather than project from the image plane into* the volume, the footprint of each voxel is projected onto the image plane. This footprint is typically a Gaussian distribution but this may be simplified to a triangle or step function, trading accuracy for rendering speed. The method was first described by Westover [65].

Hybrid

Hybrid methods are used to present scenes containing a mixture of image, geometric and volume elements. For example, a prosthetic hip replacement may be matched up with a CT scan of the patients hip. Problems arise when mixing the different rendering types together, to ensure correct occlusion of objects and to allow picking of interpenetrating geometric and volume objects. In some cases, all the data must be re-rendered if one small part of the model changes. Typical strategies include converting all objects to one type (surface extraction turns volumes into geometric objects, conversely geometric objects can be voxelised) or merging the intermediate results of rendering using some depth sorting method.

3.4.3 Manipulation

Viewpoint selection

The simplest form of direct manipulation is selecting a different view of the scene, either by moving the camera or moving the objects in the scene. On a 2D screen this is typically done with sliders, dials etc. in a dialog box, or (better) by gesturing with the pointing device. A particularly useful analogy used by some systems is a *glass trackball* which conceptually encases the displayed scene and is moved by dragging the mouse. Spaceballs may be used to achieve a similar result. In a 3D environment, gesturing with a dataglove is often used.

Picking

A refinement on moving the whole scene is to select a component of it. This may indicate to the visualization software that a particular object is to be used for the next interactive operation. It represents information flowing backwards through the visualization process, from rendering to mapping or to filtering.

Probing

An object is inserted into the scene where it samples the underlying model and reports back the data values. For example colour may be used to depict pressure on the surface of an object. Colours could of course be compared by eye with the colour scale, but a probe can be positioned anywhere and will give a reading directly in kiloPascals or other suitable units. A probe reports its geometric position in the scene to earlier stages of the visualization process, which then generate the requested data. This need not be numeric; other Abstract Visualization Objects are often produced. For example a probe for a flow field might produce a solid arrow whose direction and length indicated the components of the vector flow field. Other quantities such as the curl or divergence of the field might be probed and represented as twisting of the arrow shaft, or colour of its tip.

3.4.4 Hardware support

High performance graphics workstations have dedicated hardware for accelerating various stages of the geometric rendering pipeline such as hidden line and surface removal, perspective transformation, texture mapping, and local illumination models. Less commonly, they have support for image rendering, such as bicubic zoom, or for volume rendering, such as fast, hardware assisted trilinear interpolation. This is particularly important as direct volume rendering is compute intensive and real-time performance is a much sought-after goal. Support for hybrid rendering is currently rare, particularly if picking is to be supported. General purpose visualization software may not support all the capabilities of a particular platform.

Some workstations have support for presenting stereo images - using either liquid crystal shutters to present alternate images or using twin displays in a headset - and for 3D interaction using devices like spaceballs, datagloves, and the like. These capabilities are currently rather manufacturer specific, so again it may be difficult for general visualization systems to take advantage of them.

3.4.5 Application Visualization System (AVS)

AVS uses three separate components to view different types of object. There is an image viewer, a geometry viewer, and a tracer for volume data, which produces an image. This means that hybrid scenes can only be displayed and interacted with by converting all data to geometric form. 24 bit TrueColor or DirectColor displays are supported. 8 bit displays will use 216 colours from a 6x6x6 RGB cube; while fine for general use, this is severely sub-optimal for greyscale image display with only six grey levels. Typical medical imaging systems use 250 grey levels with the other six colours mapped onto bright primary and secondary colours for annotation

Image display

Collections of images can be displayed. Zoom and stacking are supported; images can be labelled and there is a flipbook animation capability. A variety of dithering techniques can be selected. The unsupported alpha blend module allows the compositing of stacked images with variable transparency but this requires hardware support. Separate modules provide two probes: one gives the colour at a point, the other measures the distance between points.

Geometric objects

The geometry viewer supports both a hardware renderer and a software renderer. The latter supports the full functionality of AVS but may be slow. The hardware renderer will take advantage of the native graphics system - PHIGS, GL, Starbase - but in this case there is no software fallback for missing functionality. If a particular hardware system does not support, for example, transparency or texture mapping, the hardware renderer will display all objects as opaque or plain, respectively. Spaceballs are supported on SGI platforms. Stereo is supported on Silicon Graphics, Evans & Sutherland and Kubota platforms.

The next release of AVS (AVS6) will extend the graphics library support to include PEXlib and OpenGL.

Objects may be translated, rotated and scaled using a glass trackball paradigm. They may be coloured, texture mapped and their other surface properties such as specularity altered. Lights of various colours may be positioned in the scene.

Pick information can be sent to other modules and a variety of other modules provide probes. The probe module reports data values and can use either nearest neighbour or trilinear interpolation.

Volumetric objects and hybrid rendering

These are dealt with in a variety of ways. One option is to use the volume render module, which have variable colour and opacity and may be sent to the geometry viewer. This is one way to do hybrid scenes. There is no control over the lighting although the gradient shade module helps with this, and rendering can be slow. This method really requires hardware support for 3D texture mapping and transparency.

Fast views of axis aligned volumetric data are provided by the **xray** module which provides options for finding the sum, minimum, mean, median, minimum or maximum value for each pixel.

A third alternative is the cube module from the SunVision toolkit, intended for classified volumetric data. This does not appear well integrated with the other modules. Four rendering methods are supported: texture mapped external surfaces, a maximum intensity projection method which need not be axis aligned, ray casting and surface extraction. The last two methods classify voxels using values set in the edit substances module. Interpolation is nearest neighbour by default but can be set to trilinear. Rotation and translation is by specifying 4x4 matrices although other utility modules can generate these.

Lastly, a pair of modules give direct raycasting. The tracer module does the raycasting and can accept a colourmap for greyscale volumes. The display tracker provides a direct manipulation interface on the resulting image using a glass trackball paradigm. Bilinear zoom can be used on the image without having to repeat the raycasting operation.

Currently there does not appear to be a module to perform splatting.

3.4.6 IBM Data Explorer

DX uses a raft of rendering modules which co-operate to present and interact with objects. There are often alternative ways of doing similar things, depending on the data and interaction required. Unlike other systems, all object types are displayed in a simple and natural way in the same window. This consistency of user interface is a definite benefit.

Three modules are central to the rendering process. **Display** is the most basic module; it simply displays an image which may be a 2D regular dataset or the output of the **Render** module. **Display** does no rendering; it is simply a mechanism to put images on the screen.

Image display and Geometric objects

Image is a DX Macro and contains modules such as: **AutoAxes**, **Render**, **Display**, **Camera**. The Image tool is the most frequently used method of rendering and display in DX as it supports direct manipulations of the displayed scene such rotation, translation, zoom, navigate etc. Users should avoid passing images e.g. a Landsat image, to the Image tool as it will be rendered as if it where a 2D dataset prior to display. To simply display image data the **Display** module should be used.

Render is the most powerful module as it renders one or more geometric or voxelised objects and presents them.

Objects and cameras can be translated and rotated. The viewing model is easy to use, being based on a look-to point rather than a camera position. Object properties such as colour, normals, specularity can be modified. Point lights and ambient light are supported. The renderer appears to use Gouraud and Phong local illumination models.

Pick data may be sent to other modules, and a variety of probes are available. These send their 3D position, or a list of positions, to other modules. A measuring probe calculates the area and volume of objects.

Volumetric objects and hybrid rendering

Rendering hybrid scenes is readily performed subject to a few limitations - interpenetrating volumes are not supported, and volumes are not rendered with perspective. However, volumes need not conform to a regular rectangular grid. The documentation stated that volumes were rendered by `one of a variety of irregular and regular volume rendering algorithms' which appeared to mean direct ray casting using a dense emitter model: opacity gives the absorption per unit thickness and the colour relates to light emission per unit thickness: a self-luminous gel. Volumes are composited front to back with geometric objects.

3.4.7 IRIS Explorer

Image display

Images in IRIS Explorer are passed through the system as 2D multichannel lattices. Thus, all modules that accept lattices as input can be used to manipulate images - either by modifying the coordinates part of the lattice (for image cropping, scaling, rotation, etc.) or the data part of the lattice (for image filtering, blurring, edge detection, etc.). Much of the image processing functionality in IRIS Explorer is provided via the ImageVision library, an object-oriented toolkit for the manipulation, processing and display of image data. A special feature here is that ImageVision modules can be chained together to make use of ImageVision's so-called pull model for passing only the region of interest of an image along the chain. This leads to greater efficiency, especially when dealing with large images.

The **DisplayImg** module displays 2D lattices as images. The module accepts multiple input lattices; each is displayed as a separate image, and each can be separately managed, manipulated and updated. Lattices of any datatype and any number of channels are allowed - thus, single-channel input is displayed as monochrome, while 3-channel input is displayed as RGB.

Geometric objects

Geometry in IRIS Explorer is implemented using Inventor, an object-oriented 3D toolkit. Earlier releases of IRIS Explorer were based on IRIS Inventor 1.0, which used the IRIS GL for rendering geometry. The latest release of IRIS Explorer (2.2) is based on Open Inventor 2.0, which uses OpenGL for rendering.

The geometry type in IRIS Explorer is an Inventor object. This means that geometry can be shared between IRIS Explorer and other Inventor applications (for example, SGI's Showcase presentation package) for display and manipulation outside IRIS Explorer. Similarly, 3D geometries from other packages can be read into IRIS Explorer once they have been translated into Inventor file format (see chapter 4, below). It also means that modules can be written (see Chapter 6: Incorporating Application Code) which make calls to the Inventor API to create and modify geometry within IRIS Explorer (a simplified geometry API is also supplied with IRIS Explorer which provides some of the same functionality). Finally, IRIS Explorer is able to inherit and make use of much of the functionality of Inventor for 3D geometry creation, manipulation and display, which is very sophisticated.

The main module for geometry display and interaction is Render. This allows a rich set of controls over the scene, including

- Changing camera parameters via a viewing model. By default, this model allows for examining the scene through the `glass trackball' analogy mentioned above in section 3.4.3, although other viewers are available which mimic flying, walking, etc. through the scene.
- Selecting objects in the scene. The user can then edit the object's appearance (using Inventor controls over components of the lighting model), colour (using Inventor colour controls), and its scale, orientation and location (using Inventor 3D manipulators).
- Picking objects. Here, information about the selected object is written to an IRIS Explorer data type which can be passed to another module for further processing. Open Inventor's 3D picking returns useful information such as the point of intersection, normal at the point of intersection, the nearest vertex, and more.
- Controlling lighting. Besides being able to manipulate the properties of the material of which objects in the scene are composed, the user is able to create and edit various types of lights in the scene - again, making use of the sophisticated Inventor interface to do this.

Finally, it should be noted that the Render module can combine multiple geometries into a single scene, irrespective of their origin. Thus, a user could, for example, display a scene made up of a wireframe box, an isosurface, a volume rendered lattice, an imported 3D model and a slice through a vector field. This is another benefit of the flexibility of the Inventor 3D toolkit which IRIS Explorer uses for its geometry data type

Volumetric objects and hybrid rendering

A volume to geometry module does direct volume rendering by splatting. There is support for hybrid rendering as the output may be fed into the render module.

3.4.8 Khoros

The interactive data presentation routines in the Khoros system can be found in the Envision and Geometry Toolboxes. The Envision Toolbox provides a number of applications for interactively exploring multidimensional data. The data can be visualized as images, surfaces, 2D plots, or 3D plots. The Geometry Toolbox contains an interactive geometry and volume renderer. All data presentation routines interpret data according to the polymorphic and geometry data models.

Image display

There are a number of image display applications which are all in the Envision toolbox. These applications are Editimage, Putimage, Animate, and Spectrum. Editimage is an interactive image display program which provides a zooming capability, colourmap editing, false colouring capabilities, and image value display. Animate is an sequence display tool. Putimage is a non-interactive image display program. Spectrum is an interactive program for exploring multi-dimensional data.

These applications all use the Khoros image visual object. This visual object is capable of displaying image stored in any data type. Data types other than byte are automatically normalized between 0 and 255. A standard 256 entry colourmap can then easily be applied to the normalized image through the image visual object. Complex data types are converted to floating point using real, imaginary, magnitude, or log magnitude conversion before the normalization occurs. The image visual object is capable of displaying to either 8 or 24 bit displays. A private colourmap is used. On an 8 bit display, 24 bit images will be displayed using a fast 332 quantization. Large images may be displayed using a pan icon. The image data is cached such that the entire large image is never all in memory at any one time. This is also true of the animation display with large animations.

An image probing capability is available in Editimage. The data values at and around the pixel indicated by the mouse are displayed. The explicit world-coordinate position of the pixel is also displayed if explicit location data is available.

Geometric objects

The RenderMonster application in the Geometry Toolbox provides the geometry visualization capability to the Khoros system. Implemented as a software renderer, RenderMonster interactively produces either 24 bit true-color rendered images, or 8 bit rendered images. Alpha compositing is used to render solid and semi-transparent objects together.

The Xprism application in the Envision Toolbox provides fifteen different 2D plot types and nine different 3D plottypes. It supports multiple plots per area and multiple plot areas. All details of the axis, tic marks, labels, colors, line styles, and annotations are easily modified by the user. The built-in expression parser allows the user to create complex data arrays interactively.

Geometry Transformations and Viewpoint Selection in RenderMonster can be performed interactively on the rendered image using the mouse. Pressing different mouse buttons on the rendered image and moving the mouse will perform scalings, rotations, and translations. A bounding box is used to show the transformation interactively as it is being performed. When the bounding box has been transformed to the desired position, a new rendering is done.

Khoros runs on any hardware with an X display, regardless of it being 8 bit or 24 bit. No special hardware capabilities are required, nor are they used if present. There are plans however on doing a GL port of RenderMonster.

Volumetric objects and hybrid rendering

In addition to being able to render geometry, the RenderMonster application in the Geometry Toolbox is also capable of rendering volumetric data directly. A voxel splatting approach is used for volumetric rendering. A voxel dot approach is also available for faster, more interactive rendering.

The RenderMonster application does not make a strong distinction between geometric data and volumetric data and is capable of rendering both geometric data and volumetric data together in a single rendered scene.

3.4.9 PV-WAVE

The presentation capabilities of this package are slanted towards presenting 1D/2D data and images with some facilities for 3D arrays of data.

Image display

Besides displaying images they may be warped, frequency domain filtered, and similar image processing tasks applied. It appeared that only indexed colour was supported, in other words images were required to have a colour table.

Geometric objects

3D data may be plotted as surface and contour plots, with either a network (wire mesh) surface or shaded surfaces (flat or Gouraud). There is no facility even for Phong shading.

Translation and rotation of objects is specified with 4x4 matrices using a command language. There does not appear to be a direct manipulation interface for this. A form of pick is available but this gives a 2D position in pixel coordinates rather than a 3D position in world coordinates.

Volumetric objects and hybrid rendering

Volume processing produces a 2D image from a particular view. The package does not support the mixing of volumetric and geometric objects.

Review of Visualisation Systems

Chapter 4: Data Import

Review of Visualisation Systems

4.1 Introduction

The tables which follow are used to list the currently available data readers for each visualization system. It has not been possible to verify the robustness of each reader or the availability across the platforms each system supports during the review. The following key is used to indicate the source of the reader:

- standard(s): is supplied with the visualization system by the vendors;
- public domain(p): is obtained from an anonymous FTP site. The full list of sites and FTP addresses can be found in Chapter 8: Additional Information;
- commercial(c): can be obtained from a commercial supplier. These readers have a cost associated with them and more information can be obtained by contacting the supplier.

Review of Visualisation Systems

4.2 Application Visualization System (AVS)

4.2.1 Supported data readers

This section outlines the data readers for specific formats which are available as part of the visualization system.

4.2.2 Tools for importing data

There are three main methods of importing data into AVS:

Standard AVS modules: there are a number of modules provided to read data files in one of the defined AVS datatype formats e.g., **read_ucd**, **generate_colormap**, **read_geom**;

AVS read_field: this module parses an ASCII header file which describes the organisation and structure of a binary/ascii datafile. The data is then imported and mapped onto the AVS field datatype. As the AVS field datatype is used then it is restricted to general N dimensional arrays of M data elements;

AVS Data Interchange Application (ADIA): this tool provides the ability to import data which matches the AVS field datatype but has a number of advantages over the **read_field** module:

- interactive graphical user-interface;
- supports variables and expressions removing the need for absolute values;
- reads information from the datafile itself e.g., dimensions, length;
- searches the file for token keywords which specify format values e.g., length=72;
- the forms created can be saved or loaded for re-use.

Figure 8 shows a sample of the ADIA interface.

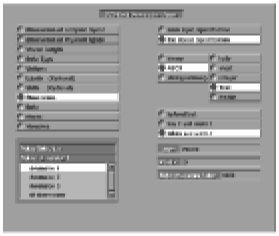


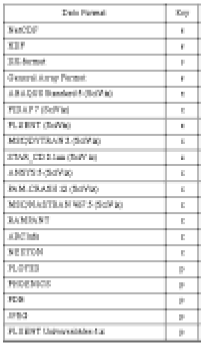
Figure 8: Example of the ADIA interface

Review of Visualisation Systems

4.3 IBM Data Explorer

4.3.1 Data readers

A number of modules and macros are available via anonymous ftp from the DX repository at Cornell (ftp.tc.cornell.edu) under the directory pub/Data.Explorer more details are given after the table.



Data Format	Key
NetCDF	r
HDF	r
DX-format	r
General Array Format	r
ABAQUS Standard 5 (SciVis)	c
PDA P7 (SciVis)	c
FLUENT (SciVis)	c
MSC/DYTRAN 2 (SciVis)	c
STAR_CD 2.1aa (SciVis)	c
ANSYS 5 (SciVis)	c
PAM-CRASH 12 (SciVis)	c
MSC/NASTRAN 67.5 (SciVis)	c
RAMPANT	c
ARCInfo	c
NEKTON	c
PLOT3D	p
PHOENICS	p
PDB	p
JPEG	p
FLUENT Universal files 4.x	p

4.3.2 Tools for importing data

Data Explorer being an object-oriented based systems allows objects to be one of the following classes: group, series, multigrid, compositefield, field, array, constantarray, gridpositions, regulararray, productarray, gridconnections, patharray, mesharray, xform, string, light, camera, clipped, screen.

For the majority of data input, DX relies on a general array importer. A simple text header needs to be created to describe the structure and location of the data then the **import** module can read in the data directly. To use a graphical interface to define this header, type:

```
dx -prompter
```

This interface can import many simple arrays, both binary and ascii. For binary data it is possible to specify whether the data is in most significant byte (MSB) or least significant byte (LSB) first. The 2 common forms of vector interleaving are also supported:

As part of the prompter it is possible to browse through a data file to set marks and determine offsets in bytes or lines which can be useful when describing ascii data. Although the prompter cannot completely describe irregular grids (e.g. FE data) it can be useful to determine some of the description then complete the header by editing the resulting text.

For a more complete guide to the data model see Chapter 3, "Understanding the Data Model" and for importing data into DX see Chapter 4, "Importing Data", from the User's Guide.

Review of Visualisation Systems

4.4 IRIS Explorer

4.4.1 Data Readers

The following is a list of readers supplied with the package or available in the public or commercial domain:

In addition, the following CFD-related readers for IRIS Explorer should soon be available commercially from SciViz:

- ABAQUS
- ANSYS
- FIDAP
- FLUENT
- LS-DYNA3D
- MSC/DYTRAN, MSC/NASTRAN

Finally, NAG has developed readers for a number of formats, including PHOENICS, USGS, NTF, etc. These will be made available as part of the next release (3.0) of IRIS Explorer.

4.4.2 Tools for importing data

Explorer provides a number of tools for importing data.

Standard IRIS Explorer Datatype Readers: There are a set of modules which are capable of reading standard Explorer data types from files, one for Pyramids, Lattices and Geometry. They are capable of reading data saved in either ascii or binary format.

Basic Ascii Format to Explorer Format: There are several modules which take files in simple ASCII formats and produce lattices. For example, the module ReadXData creates a 1D lattice from the data in an ascii file in the following format: n, x1, f(x1), x2, f(x2),... xn, f(xn), where n is the total number of data pairs. These modules are all written using DataScribe (see below).

DataScribe: This is IRIS Explorer's visual programming interface for creating modules to read data in a variety of user-defined formats. It takes ASCII or binary data files and outputs lattices (or vice versa). An example from DataScribe is shown in figure 9. Some of the features of DataScribe are:

- It is possible to use pattern matching to skip text headers or actively search for character strings within a file to find formatting values. The pattern matching includes the use of UNIX "*" and "?" wildcards.
- DataScribe provides a *ruler* for formatting ASCII data that makes reading and writing formatted files easier. It allows the interleaving of text and values at specified columns and reading values only from specific columns.
- Values describing the size and shape of the data can be read in from the data file or set to a constant value. Also, it is possible to read to EOF removing the need to know the whole extent of the data.
- A number of different files can be read by one DataScribe module and combined into one or more output lattices. DataScribe can also be used to read Explorer formats and transcribe sections or channels of data.

- Parameter values can be attached to widgets on the control panel of a data input module and can be used to section the data. The control panel can be edited from DataScribe by altering the appearance, default values and position of the widgets.
- A number of example DataScribe readers come with the system and the scripts they use are available as a starting point for making further readers. Having created a data reader the script and module resource files are saved and can then be inserted as a module in the Explorer map.

DataScribe only works with Lattices. However, there are examples in the manual (with the source provided on-line) of readers which produce Pyramids, plus the module **ComposePyr** can be used to build a pyramid from a set of lattices.

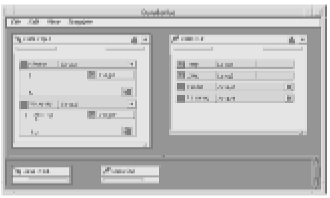


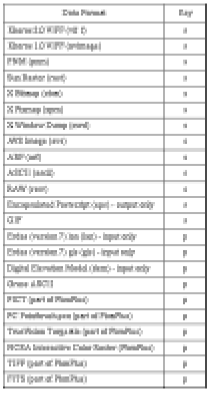
Figure 1: Example of DataScribe

Review of Visualisation Systems

4.5 Khoros

4.5.1 Data readers

The specific formats which are supported within Khoros 2.0:



Data Format	Key
Khoros 2.0 VIFF (viff 2)	s
Khoros 1.0 VIFF (xvimage)	s
PNM (pnm)	s
Sun Raster (rast)	s
X Bitmap (xbm)	s
X Pixmap (xpm)	s
X Window Dump (xwd)	s
AVS Image (avs)	d
ARF (arf)	s
ASCII (ascii)	d
RAW (raw)	d
Encapsulated Postscript (eps) - output only	d
GIF	s
Erdas (version 7) lan (lan) - input only	p
Erdas (version 7) gis (gis) - input only	p
Digital Elevation Model (dem) - input only	p
Grass ASCII	p
PICT (part of PbmPlus)	p
PC Paintbrush/pcx (part of PbmPlus)	p
TrueVision Targa file (part of PbmPlus)	p
NCSA Interactive Color Raster (PbmPlus)	p
TIFF (part of PbmPlus)	p
FITS (part of PbmPlus)	p

A number of public domain utilities are available; for example the PBM Plus utilities could be used for importing data into a pnm format which can then be read by all Khoros operators.

4.5.2 Tools for importing data

Khoros has a different view on data import. In general, data access is performed through data services using one of the Khoros data models. Data services transparently supports a number of file formats. When a file is opened, data services checks the file to determine if it is one of the supported file formats. If it is, then the data contained in the file will be made available through the various segments in the data model. Applications which use data services can simply open up a file, and if the file is one of the supported formats, it will be able to access it. while most file formats can be automatically understood by Khoros operators, some file formats may need extra information to be imported correctly. The following routines are available for importing ASCII and raw data.

- *kimportasc*: this routine allows you read an ASCII file into the Khoros 2.0 VIFF format, while specifying the size, data type, and index order of the ASCII data. Support for skipping points in the ASCII data is also provided. Simplified versions of this routine which read ASCII data into each of the polymorphic data segments are also available.
- *kimportraw*: this routine allows you to specify details of the raw data format (such as what machine the data was created on) and import the data file into a Khoros 2.0 VIFF file.
- *kformats*: this routine allows you to convert a file stored in any of the supported file formats to any of the other supported file formats.

Review of Visualisation Systems

4.6 PV-WAVE

4.6.1 - Data readers and importing data

4.6.1 Data readers and importing data

PV-WAVE CL provides a number of procedures for data input, which are reasonably comprehensive and easy to use, although readers for standard data formats are not available other than for TIFF.

The data input procedures and DC functions include:

- READ (formatted read on the standard input stream)
- READF (formatted read from a specified file)
- READU (binary unformatted read from a specified file)
- DC_READ_FIXED (ASCII fixed format)
- DC_READ_FREE (ASCII free format)
- DC_READ_8_BIT (8-bit image data)
- DC_READ_24_BIT (24-bit image data)
- DC_READ_TIFF (Tag Image File Format)

Equivalent functions broadly exist for data output (please see the data output chapter).

Data import/export error messages and associated options can be specified through the functions:

- DC_ERROR_MSG (returns text string associated with error condition)
- DC_OPTIONS (set error message reporting level)

In general data input is simple and straight forward and the supplied documentation provides useful examples.

Review of Visualisation Systems

Chapter 5: Data Output

Review of Visualisation Systems

5.1 Application Visualization System (AVS)

5.1.1 Hardcopy facilities

There are two main methods within AVS to generate postscript files (monochrome and colour) from scenes and images in the Geometry Viewer and Image Viewer modules are:

Image to Postscript Module

This module takes as its input a standard AVS image and converts the output to Postscript in a file on the system. The module supports two types of postscript output:

- An 8 bit gray scale image for monochrome printers
- A 24 bit true colour image for colour printers supporting the level 1 Postscript operator colorimage or any Postscript Level 2 printer.

The module also allows the user to specify: orientation: landscape or portrait style: encapsulated (for inclusion into other packages) size: the page x and y size in inches

One of the problems with this module though is that it generates images at a resolution of the screen (100 dpi) whereas most printers have a higher (300 dpi) resolution. One method of increasing the resolution the module can produce is to enlarge the image using the **geometry_viewer** facilities.

geom_save_postscript CLI command

This is the preferred method of generating output from the geometry_viewer module. This CLI command will output the contents of the **geometry_viewer** window using Postscript lines and text where possible to improve the quality of output. There is however no conversion of polygons to the appropriate postscript representations. To use the facility you must run AVS with the command line interpreter running and when you are happy with the scene in the **geometry_viewer** execute the CLI command.

5.1.2 Files

There are a number of modules which produce output files from various stages in an AVS network. These are listed below with a brief description:

Standard AVS modules

Image to CGM: converts the input image into the Computer Graphics Metafile (CGM) format. All three encodings, binary, character and clear text are supported and control is provided for the page height and width and orientation.

Write field: writes an AVS field data type to disk. The file has two sections: an ASCII header and the data written in a binary format. The module allows control on the format of the binary portion and it can be written in the machine's native format or in Sun's (external data representation) XDR format which is useful for transporting files across machine ranges.

Write UCD: writes unstructured cell data to a file in either binary (compact) or ASCII (human readable) format.

Write image: writes an AVS image data structure to a file. These files can be read by the AVS module read_image and some public domain image conversion software e.g., customised versions of xv and it is a supported format for the San Diego Image Toolkit.

Write volume: writes an AVS volume data structure to disk. These files can be read by the AVS module read_volume.

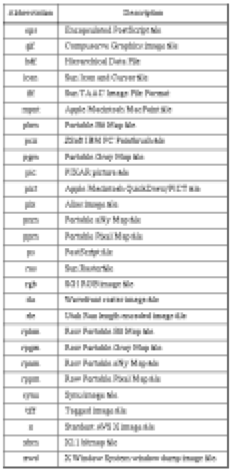
ip write vff: converts an AVS image data structure into a SunVision binary vff image format file.

Write structure file: convert an AVS Molecule Data Type (MDT) data structure into a structure file (and associated formal charge file). A structure file is one of the formats supported within the AVS Chemistry Developers Kit.

Public domain modules

Create_MPEG: creates MPEG movies from a series of AVS images. It makes use of Andy Hung's MPEG which is available by anonymous ftp from Stanford. Geom_to_Wavefront: generates an equivalent Wavefront (.obj) file for each polyhedron and polytriangle in an AVS Geometry object.

WRITE_ANY_IMAGE: This module writes an image from an AVS Network in a variety of formats which the San Diego Supercomputing Center's image tools support. Any of the following image file formats can be written by this module:



Abbreviation	Description
eps	Encapsulated PostScript file
gif	CompuServe Graphics image file
hdf	Hierarchical Data File
icon	Sun Icon and Cursor file
iff	Sun TAAC Image File Format
mpnt	Apple Macintosh MacPaint file
pbm	Portable Bit Map file
pcx	ZSoft IBM PC Paintbrush file
pgm	Portable Gray Map file
pic	PIXAR picture file
pict	Apple Macintosh QuickDraw/PICT file
pix	Alias image file
pnm	Portable aNy Map file
ppm	Portable Pixel Map file
ps	PostScript file
ras	Sun Rasterfile
rgb	SGI RGB image file
rla	Wavefront raster image file
rle	Utah Run length encoded image file
rpbm	Raw Portable Bit Map file
rpgm	Raw Portable Gray Map file
rpnm	Raw Portable aNy Map file
rppm	Raw Portable Pixel Map file
synu	Synu image file
tiff	Tagged image file
x	Stardent AVS X image file
xbm	X11 bitmap file
xwd	X Window System window dump image file

field2_to_Math: allows a two-dimensional scalar field to be imported into Mathematica from AVS. The session is initialised with a package of Mathematica commands which start a MathLink communication channel to AVS. Other initialisations are made so that the Mathematica command AVSReadField will read a two-dimensional scalar field from AVS.

field_to_EXCEL: ASCII Excel spreadsheet format files are produces containing field data in a table format. These files can be imported into a suitable excel format for analysis and plotting. Separator characters may be specified for differing output versions, three possible floating point output formats are selectable, and the maximum number of values per line is adjustable.

Output a60: The output a60 module takes an AVS image data structure as input, converts it into Abekas YUV format and sends it to an Abekas a60 digital disk recorder.

ucd_to_wave: The ucd to wave module writes a ucd structure to disk, in wavefront format, which is supported by Data Visualizer. Ucd to wave works only with node data, therefore if you have cell data, you have to transform the cell data to node data using the cell to node module.

write_Dore_i: converts an AVS image data structure into Dore format

write_KSWAD: converts an AVS image data structure into a KSWAD format

write_MooV: Create QuickTime sequences from AVS images. This code currently requires, the QuickTime Movie Exchange Toolkit, which is available from Apple, Inc, and a C++ compiler (you also need the C++ compiler to build the QuickTime libraries.

write_jpeg: Compresses an image with the JPEG compression standard and writes it to a file.

5.1.3 Animation and Video facilities

Standard AVS

With standard AVS users can control the behaviour of downstream modules in an AVS network by using the **animated float** and **animated integer** modules. These modules both output a stream of numbers which can be used to control the parameters of modules downstream in the network. The modules allow the user to input the minimum and maximum range of the numbers along with the number of intervals or steps to proceed with. This allows the control and generation of *flipbook* style of animation.

The modules **image_viewer** and **geometry viewer** both have facilities for the creation of simple flipbook animation by recording a sequence of images or geometries and then playing them back from memory. The playback speed and performance is highly dependent on the platforms rendering speed and memory configuration.

Public domain modules

There are a few public domain modules which provide animation facilities extra to the ones in standard AVS. The **create_mpeg** module has been mentioned in the previous section on file output.

CICA Keyframe module: this module is connected to the geometry viewer module and controls the of objects by manipulating their transformation matrixes. The user can define a number of keyframes and then interpolate between them using linear or spline interpolation.

Fast Animate: this module is designed animate a series of AVS geometries which have been written to disk. The files are defined with a prefix and then a frame sequencing number with controls to step forward, back and play a sequence.

AVS Animator

The AVS animator package consists of a number of modules and a separate license to the standard AVS one is needed to use the AVS animator module.

The package consists:

- **AVS Animator**
- **Read Frame Seq**
- **Write Frame Seq**
- **Output ImageNode**
- **Output VideoCreator**
- **Prepare Video**

AVS Animator is the main module within the package and provides a front end to the keyframe style of animation. The animator module can control the rendering modules: **image viewer**, **geometry viewer**, **graph viewer** etc. It can also control the parameters associated with modules in the same AVS network upstream of these rendering modules.

The user generates an animated sequence or script by defining a number of keyframes and then asking the AVS Animator to in-between or interpolate between these keyframes. The keyframe definition consists of the objects attribute information within the rendering module and also the parameter settings in the modules upstream of the renderer. An example scenario could be a rotating dataset with the isosurface value changing from its minimum to maximum value. Obviously more complex animations are possible.

When an animation has been generated the user has to hand a more complex user interface to control the aspects affected at each keyframe (see figure 10). This allows the user to edit the final sequence.

Figure 1: AVS Animator Interface

As the animator module generates an animated sequence useful information is sent out of its output ports: frames/second, frame number and current time.

Finally the animator allows the user to save and load scripts that they have generated.

Along with the main animator module come a number of support modules:

Write Frame Sequence: compress a series of images which make up a flipbook style of animation into a single file. The module can also add and delete frames from an existing frame sequence file. It is useful for collecting images from a simulation where each time step takes a time to calculate.

Read Frame Sequence: read a series of files written to a frame sequence file by the write frame sequence module. The module has the animator style control panel and optional controls over the playback speed and current frame.

Prepare video: pre-process animations before sending them to a video device. The processing that is supported:

- Low pass filter
- Interlacing
- Gamma correction

Output ImageNode and **Output VideoCreator**: two example modules for the DiaQuest Inc. and Silicon Graphics VideoCreator video boards. These modules serve as examples for users who wish to interface to other video output hardware.

Review of Visualisation Systems

5.2 IBM Data Explorer

5.2.1 Hardcopy

There are 3 modules in DX that will produce an image from some data: **render**, **display** and **image**. Render produces the image but doesn't display it while **display** will display it too but requires a camera module to produce transformations on the viewpoint. The image tool combines both display and a camera, in addition allows the user to save and print images. If the render tool is used it would be necessary to output the image from the **render** module into **write image** or the **image tool**. The **write image** requires the user to open its control panel and type in the name of the format (new users will need to refer to manual pages to determine what this string is and may find the control panels a little difficult to understand). The **image tool** uses pull down menus to select the format. Using print image under the "file" menu of the Image tool, the standard formats provided to send directly to a printer are: Colour PS, Greyscale PS, Encapsulated Colour PS and Encapsulated Greyscale PS. In addition it is possible to save images using "save image" under the "file" menu: as a PS file, in RGB (grouped together), R+G+B (all R, then all G then all B), or TIFF. Having saved the image this can be converted to other typical hardcopy image formats using a public domain image tool, e.g. San Diego Supercomputing Centre (SDSC) tools. Example macros and networks are available at the Cornell anonymous ftp site (ftp.tc.cornell.edu), under pub/Data.Explorer/Extensions/Import.and.Export.Macros. The macros automatically create a temporary tiff file, convert it to the requested format using the SDSC imconv routine, and finally delete the temporary tiff file.

5.2.2 Files

Apart from the creation of image formats, as detailed above, the only other standard module for creating output files from dx is the **export** module. This exports data from Data Explorer but only produces dx formatted or bin files (dx is the default, needs to have the text "dx" or "bin" in the "format" entry of the control panel). The bin format is only recommended on IBM POWER Visualization Systems. With "dx" format additional modifiers apply:

- MSB or LSB (most or least significant byte order)
- 2 formats of dx are currently supported ieee (or binary) and text (or ascii).

5.2.3 Animation/Video Creation

The primary tool used for the creation of Animation sequences is the **sequencer -** this can be found under the category "Special". Given a min, max and step value (passed from other modules or set explicitly), this basically creates a sequence of integers. Although like more advanced animation utilities, the sequence can be played forwards, backwards, set to a particular value, paused, or set to loop (either to the last then jump back to the first or bounce forwards then backwards). The default interface resembles a simple VCR control panel, see Figure 11

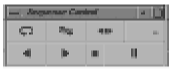


Figure 11: VCR interface of the sequencer

To set the values used by the sequencer, an additional control panel needs to be opened. By selecting the box containing "...", the panel shown in Figure 12 becomes visible.

Figure 1: Additional control panel for sequencer

On its own this module is not particularly useful for general purpose animations, but in combination with others can become a powerful animation aid. For example, when used with the transformation tool **Compute** it can produce a sequence of float values that could modify camera position, or with the interactor **Selector** could cycle through time series data. Several networks exist in the standard example directory, /usr/lpp/dx/samples/programs, to illustrate this. Only one sequencer is allowed in a network, but more experienced users can build networks such that this produces side by side animations.

Review of Visualisation Systems

5.3 IRIS Explorer

5.3.1 Hardcopy

Hardcopy output of visual information is provided by colour Encapsulated PostScript through the **Render** module. Other modules producing visual output, such as **Graph**, **Histogram**, **DisplayImage**, do not output PostScript, though a new module **NAGGraph**, available soon, will do so. The **Render** module File menu has a Print selection with options to set an output size in inches and orientation (Landscape or Portrait), sending directly to the printer or to a file. When sending to a file it is also possible to output RGB.

5.3.2 File output

DataScribe

DataScribe is a visual programming interface for creating user-defined data readers. It takes ASCII or binary data files and turns them into IRIS Explorer Lattice data types, or vice versa. It works only to/from the various Lattice types (see Data Import section).

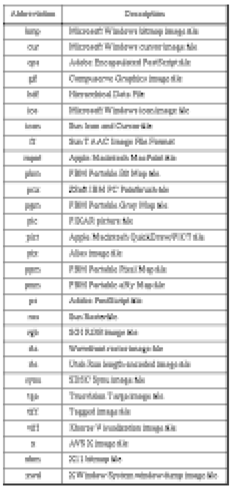
Image Formats

ReadImg and **WriteImg** module

These IRIS Explorer modules read/write a lattice containing a 2D image to a file. The lattice can have 1, 3 or 4 data variables denoting greyscale, RGB or RGB Opacity.

ImportImage and **ExportImage** modules

These user-contributed modules convert between an IRIS Explorer Lattice containing a 2D image and one of the following formats:



Abbreviation	Description
bmp	Microsoft Windows bitmap image file
cur	Microsoft Windows cursor image file
eps	Adobe Encapsulated PostScript file
gif	Compuserve Graphics image file
hdf	Hierarchical Data File
ico	Microsoft Windows icon image file
icon	Sun Icon and Cursor file
iff	Sun TAAC Image File Format
mpnt	Apple Macintosh MacPaint file
pbm	PBM Portable Bit Map file
pcx	ZSoft IBM PC Paintbrush file
pgm	PBM Portable Gray Map file
pic	PIXAR picture file
pict	Apple Macintosh QuickDraw/PICT file
pix	Alias image file
ppm	PBM Portable Pixel Map file
pnm	PBM Portable aNy Map file
ps	Adobe PostScript file
ras	Sun Rasterfile
rgb	SGI RGB image file
rla	Wavefront raster image file
rle	Utah Run length-encoded image file
synu	SDSC Synu image file
tga	Truevision Targa image file
tiff	Tagged image file
viff	Khoros Visualization image file
x	AVS X image file
xbm	X11 bitmap file
xwd	X Window System window dump image file

Render

The File menu of **Render** can also be used to output files in IRIS Inventor format. The file can subsequently be read into a Showcase document on an SGI machine whilst retaining the 3D relationships recorded therein.

ReadXXX and WriteXXX modules

IRIS Explorer's internal data formats (Lattices, Pyramids, Geometry including camera and lighting specifications) have an associated module for writing out to a file, whether in ASCII or binary. Binary output typically is used where the file is to be read back into Explorer at a later time, using the corresponding **ReadXXX** module. ASCII format is useful to elucidate the data structures and for visual inspection during debugging. Storage of a colourmap is as a specific instance of a Lattice. Parameters and Pick data types can be output to the IRIS Explorer log window using **PrintXXX**

5.3.3 Video/Animation

The **AnimateCamera** module is provided to animate the viewing position. Typically, camera positions are output from the **Render** module and are input to **AnimateCamera** in *Learn* mode. In *Run* mode, **AnimateCamera** interpolates at user-specified intervals to produce camera geometry which is then input to **Render**. Camera positions can also be stored in a file and read into **AnimateCamera** using **FileList**.

Loop control modules such as **For**, **Trigger**, **While**, *and* **Repeat** allow the generation of a sequence of parameters to drive, for example, automatic isosurface calculation using the **IsosurfaceLat** module.

WriteAnimation writes incoming 2D lattices into a single 3D image in FIT format, which is defined and used by the IRIS ImageVision Library. This format is accepted by SGI moviemaker in which it can be edited and saved in *movie* format which is accepted by SGI movieplayer. This forms flip book animation.

The module pair **WriteMovie** and **ReadMovie** allow an SGI movie file of concatenated images to be written out and read back in frame by frame. When used in conjunction with the loop control constructs this allows flipbook animation of images.

A suite of modules is available to control an external video recorder connected to a hardware video board. The hardware supported includes the *VideoLab Board*, *Starter Video*, *VFR* and *VC* and the options available are frame rate, timing and output format, all of which can all be set from the **VideoDevice** module.

The **VideoControl** module controls an external video device and has support for V-LAN as well as VC and VFr. There are 3 modules designed to work with the Starter Video board, **VideoStarterIn**, **VideoStarterInWin** and **VideoStarterOut**:

- **VideoStarterIn** takes input video via the Starter Video board and outputs a series of user specified frames in Lattice format. Various settings may be altered including Hue and Saturation of the image as well as the scale. Both NTSC and PAL formats are supported.
- **VideoStarterIn** and **VideoStarterInWin** do the same job except that **VideoStarterInWin** has a drawing region on which to display single frames before they are grabbed.
- **VideoStarterOut** sends video output via the Starter Video board and has inputs either of a lattice or a user specified region of the hi-res display. It also supports both NTSC and PAL formats.

There are similar versions of **VideoStarterInWin** and **VideoStarterOut** for the VideoLab board called **VideoLabInWin** and **VideoLabOut** with only slightly different options.

Review of Visualisation Systems

5.4 Khoros

5.4.1 - Hardcopy

By default, all Khoros operators write out in the Khoros 2.0 VIFF file format. This is done by default because not all formats are capable of storing the full polymorphic data model. For example, the PBM format is only able to store map and value data. Thus, if you create a data object with explicit location data and then save the object using the PBM format, your location data will be lost. The Khoros 2.0 VIFF format is the only supported format which is capable of generally supporting all data segments and attributes. Note, however, that since most of the supported formats are designed for storing images, this is typically not a limitation if you are working with image data.

5.4.1 Hardcopy

The kformats operator can be used to convert any of the supported file formats to any of the other supported file formats.

The specific formats which are supported within Khoros 2.0:

- Khoros 2.0 VIFF (viff)
- Khoros 1.0 VIFF (xvimage)
- PNM (pnm)
- Sun Raster (rast)
- X Bitmap (xbm)
- X Pixmap (xpm)
- X Window Dump (xwd)
- AVS Image (avs)
- ARF (arf)
- ASCII (ascii)
- RAW (raw)
- Encapsulated Postscript (eps) - output only
- Support for GIF will be in available with the CD release.

Review of Visualisation Systems

5.5 PV-WAVE

5.5.1 Hardcopy

PV-WAVE CL provides a number of options for producing hardcopy output and includes support for, Postscript, HPGL, PCL, QMS QUIC, CGM, Tektronix 4510 (rasterizer(*) DEC LJ250, SIXEL and PICT.

Five steps are required to produce hardcopy output and these are (PV-WAVE CL Commands in brackets):

Specific formats and configurations are specified through the use of keywords to the above commands. Where commands are not specified e.g. no DEVICE following a SET_PLOT, default values are taken. The command HELP can be used to obtain information about the currently selected device.

In general hardcopy output is easily created and the supplied documentation provides useful examples.

5.5.2 File output

File output from PV-WAVE CL is created by the use of the data export procedures and DC functions, which include:

- **WRITEU** (binary unformatted)
- **DC_WRITE_FIXED** (ASCII fixed format)
- **DC_WRITE_FREE** (ASCII free format)
- **DC_WRITE_8_BIT** (8-bit image data)
- **DC_WRITE_24_BIT** (24-bit image data)
- **DC_WRITE_TIFF** (Tag Image File Format)

Equivalent functions broadly exist for data input (please see the data input chapter).

Data import / export error messages and associated options can be specified through the functions:

- **DC_ERROR_MSG** (returns text string associated with error condition)
- **DC_OPTIONS** (set error message reporting level)

In general file output is easily created and the supplied documentation provides useful examples.

5.5.3 Video/Animation

The programming constructs available provide the mechanism to produce animation sequences, and a sample animation procedure is provided with the supplied documentation. The constructs and procedure enable simple flip book animation sequences to be created.

Review of Visualisation Systems

Chapter 6: Incorporating Application Code

Review of Visualisation Systems

6.1 Introduction

For most of the visualization systems there are public domain repositories of examples and demonstrations to illustrate the inclusion of application code. The location of these sites is given in Chapter 8: Additional Information.

Review of Visualisation Systems

6.2 Application Visualization System (AVS)

6.2.1 Programming language

AVS provides direct support for the following programming languages:

- C (K & R and ANSI styles);
- Fortran 77;
- C++

6.2.2 General overview and structure

Programmers can include user-written modules into the AVS system. A module consists of source code files and an associated help page. Within the source code of the module the user specifies what input/output data ports the module has and what parameters are associated with the module.

There are two types of modules which differ in the way their execution is controlled:

- Subroutine: these modules are controlled by the AVS Kernel (Flow Executive) which will invoke a module only when its input ports or parameters have received new or changed data;
- Coroutine: these modules execute independently of AVS and explicitly inquire data on the input ports and parameter settings whenever they require this information. They also send data into an AVS network asynchronously; this could cause problems if large amounts of data are sent so coroutines have synchronisation routines made available to them.

6.2.3 Automatic generation

There are two tools associated with the code generation and user interface aspects of AVS modules:

Module generator

The user provides a specification of the module (see figure 13) they wish to construct via a number of menus. This includes information such as programming language (currently C and Fortran 77 options), input/output port definitions, parameters and widgets (slider, dials). When the specification is complete the module generator will output a skeleton source code file utilising the information specified along with a makefile, manual page and optional programming hints;

Figure 1: AVS Module Generator Interface.

Layout editor

The parameters associated with a particular module all appear in a stack fashion towards the left-hand side of the network editor by default. The layout editor allows a user to interactively change, group and position these parameters to create a custom layout definition for a particular AVS network. This information is saved with an AVS network thus allowing the interface to a particular module to be customised differently in other AVS networks.

6.2.4 General topics

Application control of the module

An AVS coroutine module can execute independently of the AVS Kernel allowing it to provide a useful method of interfacing an external applications or devices into AVS.

Shared Data

AVS makes use of shared memory and Unix sockets for data communication between modules on the same local machine.

User interface

As mentioned earlier the Layout editor can be used to interactively change, position and group a modules parameters within an AVS network. There are also examples of modules which provide their own user interface through X11 and associated widgets and toolkits.

The current version of AVS (AVS5) was based upon their own graphical user interface, LUI. AVS6 will now have a Motif user interface on the workstation platforms and AVS6 is being planned for release onto PC platforms with the first release based on Windows NT and later versions for Windows 95 and DOS.

Compilation

You can perform cross compilation from within an AVS module and make calls to external subroutine libraries.

Debugging support

AVS provides mechanisms a number of mechanisms for debugging:

- An AVS module can be debugged using the native system debugging tools;
- Facilities to track down problems with memory allocation errors;
- Verification of the integrity of geometry data types a user has built;
- Modules exist which output the data representations to ASCII files;

Training

Two AVS courses, introductory and advanced, have been developed as part of the Advisory Group on Computer Graphics (AGOCG) Visualization Support Project at the Computer Graphics Unit, Manchester Computing Centre, University of Manchester.

The materials are available in postscript format along with the supporting data files and modules via anonymous FTP from the University of Manchester ftp.mcc.ac.uk (130.88.203.12) under the directory pub/cgu/avs/avs_course or the International AVS Center (avs.ncsc.org).

Review of Visualisation Systems

6.3 IBM Data Explorer

6.3.1 Programming language

C is the main language but using C wrappers both Fortran and C++ code can be incorporated into modules.

6.3.2 Overview of modules

There are two specific types of modules that can be implemented:

- Inboard: these are linked with the existing DX executive and do not require any socket communication;
- Outboard: these are not linked with the DX executive and can sometimes be easier to manage than the Inboard modules. Outboard modules are easier for Application Developers to ship extensions;

The API that is made available to users of DX is the same one used by the IBM DX development team.

On SGI & SUN Symmetric Multi Processor (SMP) machines such as Onyx DX modules will run in parallel within a module. Using data decomposition the DX executive spawns tasks on each of the available processors to execute the module function against a sub-area of the dataset. The sub-areas are grown into each other to cater for boundary conditions.

This is an important area for DX as it was originally written to run on a parallel machine.

6.3.3 Automatic generation

DX has a module building tool that will automatically generate most of the C code, along with the modules description file and makefile. To run this the user needs to type:

dx -builder

This is an easy to use tool, for example Figure 14 shows how the builder was used to outline a transformation module that takes 1 input and 1 output.

Apart from describing the data types so DX can set up the appropriate data, the user also includes comments to describe what the module does and what the input and output data are.

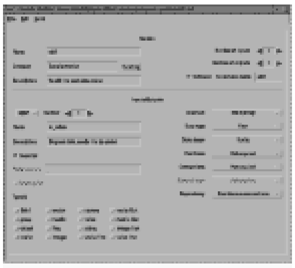


Figure 1. IBM Data Explorer module builder interface

A text version of this description can be saved for future use:

```
MODULE_NAME = add1
CATEGORY = Transformation
MODULE_DESCRIPTION = To add 1 to each data value
OUTBOARD_EXECUTABLE = add1
OUTBOARD_PERSISTENT = FALSE
ASYNCHRONOUS = FALSE
PINNED = FALSE
SIDE_EFFECT = FALSE

INPUT = in_value
DESCRIPTION = Original data, needs 1 to be added
REQUIRED = TRUE
STRUCTURE = Field/Group
DATA_TYPE = float
DATA_SHAPE = Scalar
POSITIONS = Not required
CONNECTIONS = Not required
ELEMENT_TYPE = Not required
DEPENDENCY = Positions or connections

OUTPUT = out_val
DESCRIPTION = in_value with 1 added
STRUCTURE = Field/Group
DATA_TYPE = float
DATA_SHAPE = Scalar
POSITIONS = Not required
CONNECTIONS = Not required
ELEMENT_TYPE = Not required
DEPENDENCY = Positions or connections
```

The Module Description file generated is as follows:

```
MODULE add1
CATEGORY Transformation
DESCRIPTION To add 1 to each data value
OUTBOARD add1;
INPUT in_value; group; (none); Original data, needs 1 to be added
OUTPUT out_val; group; in_value with 1 added
```

The Makefile:

```
FILES_add1 = add1.o
BASE = /usr/lpp/dx
CFLAGS = -Dindigo -O -I$(BASE)/include LDFLAGS= -L$(BASE)/lib_indigo
LIBS = -lDX -lsun -lgl_s -ly -ll -lX11 -lm -lmpc
OLIBS = -lDXlite -lm
```

```
add1: $(FILES_add1) outboard.o
$(CC) $(LDFLAGS) $(FILES_add1) outboard.o $(OLIBS) -o add1

# how to make the outboard main routine

outboard.o: $(BASE)/lib/outboard.c
$(CC) $(CFLAGS) -DUSERMODULE=m_add1 -c $(BASE)/lib/outboard.c

# make the user files

useradd1.c: add1.mdf
mdf-c add1.mdf > useradd1.c
```

The C code generated was over 500 lines and so too long to list here. This may appear to be very long, but this is well commented code and the user is only required to modify the tail end of the code:

```
int add1_worker
(
int in_value_knt,
float *in_value_data,
int out_val_knt,
float *out_val_data
)
{
/*
* The arguments to this routine are:
*
*
* The following are inputs and therefore are read-only.The default
* values are given and should be used if the knt is 0.
*
* in_value_knt, in_value_data: count and pointer
* for input"in_value"
* no default value given.
*
* The following are outputs and therefore are writable.
*
* out_val_knt, out_val_data: count and pointer
* for output "out_val"
*/

/*
* User's code goes here
*/
}
```

When 2 inputs were used the number of lines of code increased to over 650 lines but again the user is only required to insert code into the *worker* routine.

6.3.4 Further examples

There are further examples which can be found in the IBM Data Explorer Programmer's Reference Manual. There is also an FTP site of public domain modules available ftp.tc.cornell.eduunder pub/Data.Explorer.

Review of Visualisation Systems

6.4 IRIS Explorer

6.4.1 Programming language

The programming languages that are supported directly by IRIS Explorer are:

- C
- Fortran77
- C++
- Fortran90: The addition of Fortran 90 to this list is currently under consideration at NAG, although the timescale would be determined by the degree of user interest.

6.4.2 General overview and structure

Modules are written as a user function and associated subroutines called from the user function. These are then compiled and automatically embedded within an Explorer wrapper. The input and output ports of the final module are expressed as mappings from the parameters of the user function to Explorer data types. It is possible to create *hook* functions that can be called in specific circumstances i.e. the initialisation hook function is called when the module is launched into the map and is used to do initialisation for the module.

Module compilation is done using a graphical interface called Module Builder (mbuilder) which requires the name of the source code file, the Module name, any flags, any Libraries and any User Makefile. There are then subsequent visual interfaces for:-

- Input Ports: List the names, types and optionality (required or optional) of input ports. Data must be present on all required input ports before firing.
- Output Ports: List the names and types of the output ports.
- Function arguments: List the names and types of the arguments to the user function. Also specify their type, either scalar or array (or pointers to either), the return type (if any) and the language.
- Connections: This lists the input ports, function arguments and output ports in 3 columns and allows the user to wire the input ports to the required function argument and then to the output port. It also allows subtypes and structure members of the root type on the port to be wired either to a function argument or, if left unaltered by a module, direct to an output port. I.e. It is possible to wire the data portion of a lattice into a function argument, do some calculation on it, then wire it to the data portion of an output lattice while just wiring the coordinate portion directly to the output.
- Control Panel: See User Interface later in this section.

Once all aspects of the module have been entered into the Module Builder, selecting build will cause it to begin compiling the module. On building, a set of makefiles are produced and all error messages are returned to the screen. Once these makefiles are produced the module can be recompiled externally to Module Builder by using UNIX make. Building a module not only produces the executable, but also the C source code for the interface between the user function and the remainder of Explorer.

Explorer uses three controlling interfaces or *wrappers* to mediate between the module's user function and the rest of the system. These are the Module Control Wrapper (MCW), the Module Data Wrapper (MDW) and the Generic Wrapper. All three of these are generated automatically on building, though the MDW can be turned off but the user must then incorporate the MDW functions in their own code. The MCW is the module data manager, contains the firing algorithm and provides interfaces to input and output functions, including port and widget settings. The MDW performs conversions between Explorer and user-defined data types on the ports and the data format required by the user. The user can thus develop modules with customised interfaces without having to interact with Explorer data types. The Generic Wrapper is not accessible to the module writer, except indirectly through Hook Functions menu.

As an alternative to writing C/Fortran code and having to build, link and compile after each alteration, Explorer provides a method of generating user defined modules that act on the Lattice data type, without having to use Module Builder. The **LatFunction** module is an interpreter for a lattice manipulation language called Shape that makes it easy to operate on Lattices. It allows the user to interactively change the way **LatFunction** acts on the data it receives, which provides a quick way to prototype a new module without having to actually compile it. The user passes the name of the Shape program file as a parameter to the **LatFunction** module. Shape is an interpreted, array-oriented language with a C-like syntax which makes it easy to create, process and manipulate lattices. Since the program is interpreted, changing the output lattice can be done by changing the name of the file passed to **LatFunction -** i.e. a new module doesn't have to be launched every time the lattice needs to be changed. This makes **LatFunction** useful in prototyping or testing new modules.

How is control passed to/from a module?

The modules are all free to fire independently of each other, the only constraint being that they have the data on their input ports that they require. In practice this means that control is passed from upstream modules to downstream modules as data (or parameters) are generated/altered and passed on.

How is data passed to/from a module?

Data flow between modules is specified by the user by wiring modules in the Map Editor together. The connections are made from the output port of one module (always on the right of a module control panel) to the input port of another module (always on the left). This is done by pointing and clicking over the relevant port which pops a list of available output ports, selecting one of these then causes the other modules with compatible input ports to be highlighted. A *wire* is then displayed in the map editor linking the two modules together. On workstations supporting shared memory only the pointers are copied, not the data itself.

6.4.3 Automatic generation

The tool for this with Explorer is Module Builder (see figure 15), used as described above, filling in details of ports and function arguments to describe the flow of data through the module. However, instead of using the build option, "Create function prototype" is selected from the Prototypes menu and a source code template is created with gaps for the user to write their function.

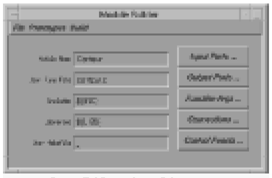


Figure 1. IRIS Explorer Module Builder Interface

One of the menu options on mbuilder is for creating document prototypes. This option produces a standard document file with all the input/output data structures listed, also, mbuilder automatically produces the template for the module help files in the standard format with all the port data types listed as for the doc pages. Gaps are left for the user to fill in specific details of the module and port descriptions using any standard editor.

6.4.4 General topics

Application control of the module

The firing algorithm in the Module Control Wrapper (MCW) is responsible for determining when the module is ready to fire. It is based on whether all the required frames of data are present on the input ports. The user can make a module fire by selecting the fire option from the pop up menu associated with each module. Explorer provides a means of registering file descriptors with the Explorer Kernel through the API function cxInputAdd. This descriptor is then *watched* by Explorer and when there is data to read a callback function is called. This in turn can then request that its associated user function be fired by calling the API function fireASAP(). Through this method it is possible to connect to external UNIX sockets and hence to other applications/devices.

Remote execution

In IRIS Explorer this is achieved by simply calling up a module Librarian on an alternative machine. This is done through the host pull down menu which gives a dialogue box into which the host name is entered. This new Librarian is used in the same way as the local one where modules are selected from it and placed in the map. Explorer takes care of the rest. These modules are marked with the name of the machine they are running on.

Shared Data

Explorer uses a shared memory model for storing data. This allows modules to share a single copy of, say a Lattice, which remains in memory until it is no longer being referenced by any modules. This is achieved by means of a reference count stored within the data structure.

User interface

In Module Builder there is an option for editing the control panel of a module. Selecting this gives a visual tool for designing and arranging the widgets associated with a module. It is possible to set size, default values, positions, visibility, sizeability and widget type. Also it is possible to add text labels, frames, and separators. The widgets are associated with the user function by expressing them in the argument list as cxParameter types. The P-func Editor can be used to calculate an input parameter value from one or more connected parameters on other modules in the map.

There is an option to build a module with X-MCW which uses the X mechanism for handling scheduling. Using this option allows Explorer to recognise and manipulate X Window or Motif widgets in a module. The advantages of this are that you have complete freedom in creating the user interface.

At a higher level, it is possible to group a set of modules that make up a map and edit a control panel made up from some (or all) of the widgets available from all the modules in the group. This includes the graphical output windows of DisplayImg and Render. This is done from within the map editor and can be saved as a map. This map can then be run in application mode which is where the map editor and librarian are unused and just the user defined control panel is visible.

Compilation

Module Builder allows the cross compilation of C and Fortran and also the inclusion of object code. All the required files are listed in the text box "User Func File" as a space separated list. Mbuilder needs the language type of the actual user function, but the rest are defined by their file extension. External libraries can be linked by inserting them in the Libraries text slot in the same way they are listed in the UNIX link command. It uses the form -l<library> for individual libraries.

Debugging support

Debugging support is provided by compiling the module with an environment variable DEBUG set to 1 to force the use of the -g flag. A module is then launched into a map and any of a number of debuggers can be attached through the modules process ID.

Explorer does some runtime checking of data and there are a few options as to the course of action taken if an error is found. These are selected at run time from the -action <INT> option where

- 0=ignore,
- 1=warning,
- 2=halt,
- 3=abort

and -level <0/1> which sets the level of data checking.

Training

The module writer's guide contains an initial chapter going through building a module with a fairly trivial example (the source of which can be found on line) explaining what the basic elements of Module Builder do. The next chapter runs in parallel with the first and gives a more general overview of Module Builder. There are chapters for each of the data types with examples of code in both C and Fortran. In addition to the Module Writer's Guide there are UNIX man pages in 2 sections, the first covering all the standard modules, the second covering all the API routines.

There are also available courses from NAG which include an introductory and advanced course. For more information you should contact NAG directly.

Review of Visualisation Systems

6.5 Khoros

Khoros 2.0 was designed to be a software development environment, so the support for incorporating new application code is very strong. Khoros is composed of toolboxes, which are simply collections of programs and libraries. Additional application code can be easily added to the system by creating new programs and libraries within a new toolbox [38] [39]. Since the Khoros system comes with complete source code, programming examples are always handy.

6.5.1 Programming language

Khoros is based on the C programming language with support for compiling with Fortran and C++. Support for sh, ksh, csh, and perl scripts as program objects is also available.

6.5.2 General overview and structure

In Khoros, application code is contained within distinct software objects. A software object a complete encapsulation of a program or library and consists of application source code, documentation, user interface information, and configuration. All software objects except libraries can be run independently from the command line, or from the visual programming language Cantata. The following types of software objects can be created.

- *kroutine*: A kroutine object is a non-interactive data processing routine.
- *xvroutine*: An xvroutine is an interactive program with its own graphical user interface.
- *pane*: A pane object is a special type of program object which is simply an alternate user interface for an existing program. Pane objects may be used as a Khoros wrapper for the easy integration of existing non-khoros program.
- *script*: A script object is a utility program which is written in some scripting language. The user interface of the script must be written and maintained by the shell script programmer.
- *library*: A library object is simply a collection of functions which can be called by other libraries or programs.
- *workspace*: A workspace object is a Cantata workspace which has been collapsed down into a software object. Note that a workspace object can also be run from the command line, as well as within Cantata.

6.5.3 Software lifecycle

The Khoros software development tools provide a complete environment which support the iterative process of developing, maintaining, delivering, and sharing software. These tools act as the programmer's assistant by providing automation where possible, enforcing consistency as necessary, and hiding underlying complexity of software configuration, code generators, and documentation formatters. This functionality is provided in the form of three applications which are described below.

- *Craftsman*: The Craftsman application is used to manage toolboxes and software objects. Toolboxes can be created or deleted from here. Software objects within any given toolbox can be created or deleted from here. Editing of a software object is initiated here by launching the Composer application.
- *Composer*: The Composer application is used to edit, manipulate, and compile existing software objects. Composer lets you edit the different parts of the software object such as the source code and the documentation. The user interface of the software object can be interactively edited by launching the Guise application.
- *Guise*: The Guise application is used to interactively create graphical and command-line user interfaces. These applications are all used in a dynamic way, allowing a developer to move easily between the different stages of development.

6.5.4 General Topics

Application control of the module

Each program object is an application unto itself giving full control to the user.

Shared data

Khoros provides data transport between processes via files shared memory, pipes, and streams.

User interface

As mentioned earlier, interactive editing of Khoros user interfaces is done with the Guise application. The resulting user interface will be created at run time using widgets from one of the supported widget sets (Athena, Motif, OLIT).

Compilation

Compilation of Khoros software objects is performed at a button press from the Composer application.

Debugging support

Khoros has no actual debugger, but provides compilation rules for linking against Purify from Puresoft and CodeCenter from Centerline.

6.5.5 Training

Khoral Research Inc. offers a "Khoros 2.0 Software Development Course" which provides a complete tour of the Khoros system, detailed coverage on each of the Programing Services, as well as the Khoros Software development system and how to write new Khoros routines.

E-mail ktraining@khoros.unm.edu for more information.

Review of Visualisation Systems

6.6 PV-WAVE

Application code can be incorporated into PV-WAVE CL at two levels, via user developed command language procedures or more appropriately user developed C or FORTRAN code. The description here will concentrate on the incorporation of user developed C or FORTRAN code as the incorporation of command language procedures is considered an integral part of the PV-WAVE CL package.

Two supplied PV-WAVE CL procedures, SPAWN and LINKLOAD enable application code to be incorporated into PV-WAVE CL and support the following languages:

- SPAWN: C and FORTRAN
- LINKLOAD: C

The SPAWN procedure provides a simple and efficient way to augment PV-WAVE CL with user developed C and FORTRAN code, and moreover provide a general mechanism for accessing the operating system. The procedure SPAWN spawns a child process to execute the application code, to which it communicates via a Unix pipe. Various input and output keywords are used to specify precise requirements. Where possible it is recommended that the required operation is performed entirely within PV-WAVE CL, as PV-WAVE CL is likely to be more efficient, particularly will small datasets.

The LINKLOAD procedure is used to access external functions in an external shared object i.e. the external code must be linked as a shared object. LINKLOAD is more efficient than SPAWN when communicating with application code and is the simplest method for attaching application code (C only) to PV-WAVE CL. Given that external objects must be linked as shared objects, LINKLOAD is only available on systems capable of supporting dynamic linking.

Useful examples are provided in the supplied documentation, although the manual descriptions do tend to be rather technical.

Review of Visualisation Systems

Chapter 7: Distributed Support

Review of Visualisation Systems

7.1 Application Visualization System (AVS)

7.1.1 Remote module execution

AVS supports the remote execution of modules on heterogenous platforms. To transfer the data between machines the external data representation (XDR) is used to represent the data and the Unix socket mechanism to transport the data. If the machines both use the same data representation then the XDR translation is bypassed. All this is made transparent to the user and remote modules are used in the same fashion as local modules in an AVS network.

7.1.2 Remote access

AVS can be executed on a machine as an X client with the display set to point to a remote X server (display). When the geometry viewer tries to start up on the remote display it will try to initialize a hardware renderer of the same type on that machine. Currently only PEX/PEX and DGL/DGL pairings are supported. The software renderer within AVS can be used on remote 8 bit colour displays and is independent of the hardware capabilities of the remote machine.

Review of Visualisation Systems

7.2 IBM Data Explorer

7.2.1 Remote module execution

Modules are grouped into Execution Groups which can then be directed to run on any machine accessible to the user. The distribution can then exploit a mix of machine architectures. As with all distributed processing care must be taken not to require too much data to be passed between machines.

7.2.2 Remote access

Since IBM Data Explorer is based on the X Windowing System and Motif it can be accessed as a remote client with its display pointed to a local X server (display).

Review of Visualisation Systems

7.3 IRIS Explorer

7.3.1 Remote module execution

In IRIS Explorer this is achieved by simply calling up a module Librarian on an alternative machine. This is done through the host pull down menu which gives a dialogue box into which the host name is entered. This new Librarian is used in the same way as the local one where modules are selected from it and placed in the map. Explorer takes care of the rest. These modules are marked with the name of the machine they are running on.

7.3.2 Remote display

Version 2.0 of IRIS Explorer allows remote display via the xhost/setenv DISPLAY mechanism, but this requires the remote machine to be SGI with the proprietary graphics GL. This limitation is fixed at V2.2, which uses the de facto standard OpenGL. IRIS Explorer provides a module RemoteRender which allows the user to display geometry on another workstation in addition to their own, provided that the second machine has remote X connections allowed, and subject to the proviso mentioned above.

7.3.3 Printing

The Print menu of Render automatically offers the names of networked printers on which to produce output.

Review of Visualisation Systems

7.4 Khoros

The Khoros installation guide provides more detailed information [35]

7.4.1 Remote module exectution

A remote daemon, phantomd, is executed on the remote machine and handles all the requests to execute a remote task and the transporting of data to/from the remote machine. The transports supported for the transferring of data to remote machines include sockets and Sys V Transport Layer Interface (tli). In Khoros 2.0, local processes now communicate through a local phantom daemon. The local phantom daemon then communicates with remote phantom daemons. This method has the benefit of enabling more advanced distributed computing features to be added such as load balancing, security, encryption, process groups. This functionality is not availble in the current ftp release of Khoros 2.0, but is expected to be on line for the CD release next year.

7.4.2 Remote access

Khoros, since it is based on X, can be accessed as a remote client with its display pointed to a local X server (display). Khoros supports all visual types of X servers.

Review of Visualisation Systems

7.5 PV-WAVE

7.5.1 Remote execution

PV-WAVE does not provide true distributed support as available within AVS and IRIS-Explorer (i.e the ability to run specific procedures as part of the system but on different machines). However, various methods of interapplication communication are provided within the PV-WAVE CL environment, and these include:

- External program execution from within PV-WAVE CL (see also Module writing)
- Execution of PV-WAVE CL from within a C or FORTRAN program (without data transfer)
- External function call from within PV-WAVE CL via dynamic linking of shareable objects (see also Module Writing)
- Execution of PV-WAVE CL from within a C or FORTRAN program (with data transfer) via static linking.
- Client Server capability to remote machines via RPC's

7.5.2 Remote Display

Distributed support in the form of X Windows client server access is supported within PV-WAVE CL.

Review of Visualisation Systems

Chapter 8: Additional Information

Review of Visualisation Systems

8.1 Introduction

This chapter is intended to provide information on:

- Availability: includes cost, platforms supported and contact address for the products;
- Support
- Information: includes on-line help facilities, manuals, network services;
- User Groups

The information below may have changed since the compilation time of this report and the appropriate vendors should be contacted for the most up-to-date information on cost, special deals, etc. Some of the price information and reference to CHEST deals is only applicable to UK Higher Education Institutes. Commercial users and academic sites outside the UK should contact the vendors for price and availability information.

Review of Visualisation Systems

8.2 AVS

8.2.1 Availability

General information on availability can be found on WWW at:

http://www.avs.com/ The International AVS Center, North Carolina Supercomputing Center, US can also be accessed on:

ftp:avs.ncsc.org
http://www.mcnc.org/HTML/ITD/IAC/IAC.html

Supplier

The UK office of Advanced Visual Systems Inc is:

AVS/UNIRAS Ltd.
Montrose House
Chertsey Boulevard
Hanworth Lane
Chertsey
Surrey KT16 9JX

Tel: 01932 566608
Fax: 01932 568842
Email:

Platforms

A brief summary at time of writing is given here, but please see the following URL for current information. Workstation platforms supported are:

- Data General
- DEC Alpha AXP, MIPS and VAX
- Evans & Sutherland
- Hewlett Packard PA-RISC
- IBM RS/6000
- Kubota Denali
- Silicon Graphics
- Sun Microsystems SPARC and also on supercomputers e.g., CS-6400
- Convex
- Cray Y-MP

- Fujitsu VPX
- Intel
- Meiko
- NEC
- Thinking Machines

The next release of AVS (AVS6) will be released on PC platforms including both Windows NT and Windows 95.

For more up-to-date information access the URL:

http://www.avs.com/products/avs.html

Costs

The following is an extract from the NISS Bulletin Board; refer to the Bulletin Board for full details. CHEST and AVS/UNIRAS Ltd. have concluded an Amendment to the CHEST/AVS Agreement which offers a per-platform licence, as an alternative to the multi-platform licence originally offered. This deal is applicable only to UK Higher Education Institutes.

The annual charges are as follows:

- Multi-Platform Licence: 4,800 for each year that the Site is licensed; and: AVS Animator 900 per site licence.
- Per-Platform Licence: 2,150 for first platform ordered: And: 1,150 for each subsequent platform ordered; and: AVS Animator site licence 400 for first platform ordered. And: AVS Animator site licence 225 for each subsequent platform.

8.2.2 Support

UK Support Arrangements

The following is an extract from the NISS Bulletin Board; refer to the Bulletin Board for full details.

International AVS Center

The International AVS Center serves as a catalyst for expanding the AVS user base and for increasing AVS functionality by fostering discipline-specific module development and new AVS uses. Located at the North Carolina Supercomputing Center, the worldwide clearinghouse collects, ports, and distributes user-contributed, public-domain modules and acts as liason between users and vendors. The International AVS Center also publishes a quarterly magazine called AVS Network News and a yearly module catalog. It also hosts the yearly International AVS User Group conference and coordinates User Group activities.

The AVS Consortium is made up of seven AVS sponsors and two affiliates who are funding and providing direction for the International AVS Center. The seven sponsors are Advanced Visual Systems Inc., Digital Equipment Corporation, IBM, Hewlett Packard Company, Kubota Computer Corporation, Kubota Pacific Incorporated, and Sun Microsystems. The two affiliates are Mobile Research and Development and Oki Electric, Inc.

International AVS Center North Carolina Supercomputing Center 3021 Cornwallis Road Research Triangle Park, NC 27709 Phone (US) 919-248-1100

The main email alias for the center is

8.2.3 Information

Manuals

- AVS Applications Guide
- AVS Chemistry Developers Toolkit Guide
- AVS Developers Guide
- AVS Module Reference
- AVS Technical Overview
- AVS Tutorial Guide
- AVS5 Update manual
- AVS Users Guide

On-line Information

AVS 5.02 has online reference information for all the modules in the system and a series of pages for various topics within AVS. The next version of AVS (AVS6) will have online, context-sensitive, hypertext help system based on the Bristol Hyperhelp for Unix and Winhelp for Windows.

Module Repository

The International AVS Center at the North Carolina Supercomputing Center (NCSC) was established to provide support for AVS. One of the functions the center provides is a repository of AVS modules supplied by users which are available via anonymous FTP avs.ncsc.org. Manchester Computing Centre (MCC) provides a shadow of the repository of modules on ftp.mcc.ac.uk. The repository contains well over 600 public domain modules available in source code form and a number of sample datasets.

Frequently Asked Questions

This can be accessed by:

http://www.mcnc.org:80/HTML/ITD/IAC/faq.html There is also the Usenet newsgroup comp.graphics.avs

8.2.4 User groups

UK AVS User Group

There is a UK AVS User Group which meets approximately twice a year in the UK. The current charge is 25 per year for individual membership and 125 per year for organizations. The chairman is Julian Gallop, Rutherford Appleton Laboratory (email: jrg@inf.rl.ac.uk).

International AVS Center User Group

You can join the International AVS Users Group for a yearly fee of $100.00 which includes subscription to the AVS magazine, the yearly AVS electronic catalog of modules (user donated and commercial), a $50.00 reduction on attending the yearly International AVS Users Group conference and have special rates for additional services as they become available. To join, send check or money order for $100.00 (add $10.00 if out of continental USA) to:

The International AVS Center PO Box 12889 3021 Cornwallis Road RTP, NC 27709

Review of Visualisation Systems

8.3 IBM Data Explorer

8.3.1 Availability

On WWW access to general information on DX can be found at:

http://www-i.almaden.ibm.com/dx/ This also links to the ftp repository at Cornell, or can be accessed directly as:

http://www.tc.cornell.edu/DX/

Supplier

DX is from IBM or any IBM Dealer. In the UK the contact is:

Andy Hey
IBM AIX Scientific & Technical Solutions,
1, New Square,
Bedfont Lakes,
Feltham,
Middlesex, TW14 8HB.

Tel: 0171 202 5266
email:

Platforms

DX is currently available on the following (colour) platforms:

- IBM RISC System/6000 all models platforms IBM SP1 & SP2
- Multiprocessor systems Silicon Graphics: Indigo, Crimson, Onyx, and Challenge, On Symmetric Multiprocessor machines DX supports intra module parallelism i.e. most DX modules are written to run in parallel
- Sun: SPARC 2 & 10, including SMP (see SGI)
- Digital Alpha models 351
- Hewlett Packard: 9000/700 Series, (all models) 800 Series: 817,827, 837, 847, 857, 867
- Data General AViiON

Costs

- Node locked license (multiple instances on one CPU) List price 4,700.

- Floating license (up to n instances on any of the supported platforms in the user's network) List price 5,895
- Generally IBM Academic discounts can be as much as 80%

8.3.2 Support

The UK based product specialists in IBM Hursley at no additional cost.

Training

IBM run training courses as required, cost depends on student numbers location, length of and type of course (overview or detailed). Typically a 3 day course would cost approximately 250 per day.

8.3.3 Information

Manuals

There are 3 manuals: The Users Guide, Users Reference and Programmers Reference [25], [26] and [27].

The Users Guide includes how to use DX, its data types and the data importer. The tutorial to get new users started with DX is in Appendix A.

The Users Reference lists and describes the modules available.

The Programmers Guide describes how to write your own DX modules and includes how to use the module builder to automatically generate the template for new tools.

On-line Information

The on-line help for DX is very good. Help can be accessed on all aspects - including the modules, on using DX but also to get help on complete networks too. The help system uses hypertext so the user can navigate by selecting keywords from one item to the next until they find what they want (it automatically saves a history so the user can reverse the route back step by step). This was found extremely useful since looking at the entry for one module may require the user to follow links to related modules.

Network Information Services

Apart from the IBM and Cornell Theory Centre WWW pages, location given at the beginning of this section, Cornell also acts as a repository for news (see the directory ftp:ftp.tc.cornell.edu:/pub/Data.Explorer/news or equivalent from the WWW) where up to date information can be obtained on various aspects of DX, this includes a Frequently Asked Questions file.

The Usenet group exists in comp.graphics.data-explorer.

8.3.4 User groups

No formal user group exists but Birds of a Feather sessions occur at appropriate conferences, e.g. Visualization `94, and the usenet group gives a route to contact users in similar areas of work.

Review of Visualisation Systems

8.4 IRIS Explorer

8.4.1 Availability

General information on availability can be found on:

http://www.nag.co.uk/1h/Welcome_IEC.html

Supplier

IRIS Explorer was originally developed by Silicon Graphics Inc, but it is now developed, distributed and supported by NAG Ltd.

The software can be ordered from:

IRIS Explorer Center
PO Box 50
OXFORD OX2 8DR
United Kingdom

Tel: 01865 516377
Fax: 01865 516388
Email:

Platforms

A list of platforms on which IRIS Explorer is available can be found on:

http://www.nag.co.uk/1h/Welcome_IEC.html A brief summary at time of writing is given here, but please see the above URL for current information.

Workstation platforms supported are:

- Hewlett Packard Workstations
- IBM RS6000
- SUN (SunOS and Solaris)
- Silicon Graphics
- Cray Y-MP

The platforms for SUN (Solaris), Hewlett-Packard HP9000/700 and DEC Alpha are under development.

Costs

Please note that for current pricing structure users should contact their nearest IRIS Explorer Centre (which are run by NAG Ltd).

http://www.nag.co.uk/0h/visual/IE/iecbb/contact Prices for educational users is normally a 50% discount on the industrial price.

A preferential rate for is offered to Silicon Graphics users until Release 3.0 of IRIS Explorer is available. After that, NAG's standard pricing will apply.

8.4.2 Support

Provision

Support is provided through a IRIS Explorer Center help desk which can be accessed by the telephone, fax or email addresses given earlier.

8.4.3 Information

Manuals

- IRIS Explorer User's Guide: contains a "Getting Started" chapter;
- IRIS Explorer Module Writer's Guide: provides information on how to build your own modules with the Module Builder; introduces the Explorer datatypes; and shows how to use the shape language to prototype modules;
- IRIS Explorer Module Definitions: lists all the modules available with Explorer;

On-line Information

There is on-line help for each module, and man pages for the API routines.

Network Information Services

Details of all NAG products and services, including IRIS Explorer, are available from the NAG WWW server. Information is updated regularly and includes Technical Reports, Product Availability Details and Free Software.

Connection details:

http://www.nag.co.uk/ There is an FTP site at Edinburgh Parallel Computing Centre which maintains a repository of modules and an FAQ list. This can be accessed via the NAG WWW server, at the URL:

http://www.nag.co.uk/1/visual/IE/iecbb/archive Other IRIS Explorer files are available at other sites, e.g.

ftp.ncsa.uiuc.edu:/SGI/PATHFINDER/Explorer pdb.pdb.bnl.gov To subscribe to the IRIS Explorer mailing list send mail to explorer-request@castle.ed.ac.uk There is an active Usenet group for IRIS Explorer: comp.graphics.explorer

Repositories of examples

In the UK, NAG provide a bulletin board called the IRIS Explorer Center which can be accessed using mosaic (a WWW interface) or gopher. This homepage provides links, via an Explorer map looking point and click interface, to FTP sites in the UK (ftp://ftp.epcc.ed.ac.uk/pub/explorer) and in the US (ftp://swedishchef.lerc.nasa.gov/), User Group Info and FAQ, Documentation and Announcements.

The FTP sites have a mixture of binaries and source code for a variety of modules that can be downloaded. There are currently binaries for SGI and SUN systems. Also documents are available.

8.4.4 User groups

There is an IRIS Explorer User Group. Meetings are held in conjunction with the SIGGRAPH and Eurographics conferences. Details from the IRIS Explorer Center, address above, or from the WWW home page for IRIS Explorer.

Review of Visualisation Systems

8.5 Khoros

8.5.1 Availability

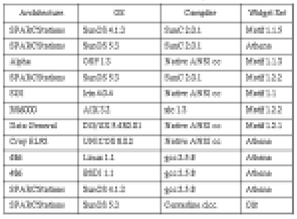
There are currently two version of Khoros available, Khoros 2.0 Developer Release and Khoros 1.0.5. The Khoros 2.0 Developer Release is available via FTP only. The Khoros 2.0 book and CD-ROM will be distributed by Prentice Hall, and should be in local book stores by the second quarter of 1995. Khoros 1.0.5 is available via both anonymous FTP and CD-ROM.

Supplier

The supplier is Khoral Research, Inc (KRI).

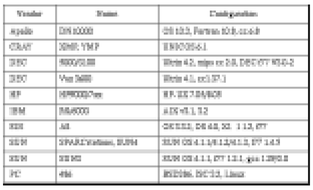
Platforms

KRI is developing on and has ported the Khoros 2.0 Developer Release to the following architectures:



Architecture	OS	Compiler	Widget Set
SPARCStations	SunOS 4.1.3	SunC 2.0.1	Motif 1.1.5
SPARCStations	SunOS 5.3	SunC 2.0.1	Athena
Alpha	OSF 1.3	Native ANSI cc	Motif 1.1.3
SPARCStations	SunOS 5.3	SunC 2.0.1	Motif 1.2.2
SGI	Irix 4.0.4	Native ANSI cc	Motif 1.1
RS6000	AIX 3.2	xlc 1.3	Motif 1.2.2
Data General	DG/UX 5.4R2.01	Native ANSI cc	Motif 1.2.1
Cray EL92	UNICOS 8.0.2	Native ANSI cc	Athena
486	Linux 1.1	gcc 2.5.8	Athena
486	BSDI 1.1	gcc 2.5.8	Athena
SPARCStations	SunOS 4.1.3	gcc 2.5.8	Athena
SPARCStations	SunOS 5.3	Centerline clcc	Olit

Khoros 1.0.5 runs on these architectures:

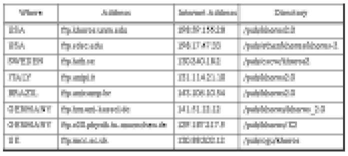


Vendor	Name	Configuration
Apollo	DN 10000	OS 10.3, Fortran 10.8, cc 6.8
CRAY	XMP, YMP	UNICOS 6.1
DEC	5000/3100	Ultrix 4.2, mips cc 2.0, DEC f77 V3.0-2
DEC	Vax 3600	Ultrix 4.1, cc1.37.1
HP	HP9000/7xx	HP-UX 7.05/8.05
IBM	RS/6000	AIX v3.1, 3.2
SGI	All	OS 3.3.1, OS 4.0, X11 1.1.3, f77
SUN	SPARCstations, SUN4	SUN OS 4.1.1/4.1.2/4.1.3, f77 1.4.3
SUN	SUN3	SUN OS 4.1.1, f77 1.3.1, gcc 1.39/2.2
PC	486	BSD386, BSDI 1.1, Linux

Costs

Khoros 1.0.5 and Khoros 2.0 are distributed via the Internet as free access software; that is, Khoros is available throughout the world, free of charge, via Unix File Transfer Protocol (FTP). In addition, Khoros 1.0.5 is also available for low-cost on CD-ROM. Although there is free access to Khoros, it is not in the public domain. The software is owned by KRI, and does carry a License and Copyright. While Khoros may be used by any organization free of charge, it can not be distributed without a license. Consult the Khoros Free Access License for additional information on use and distribution. KRI is committed to maintaining the Free Access Distribution of Khoros.

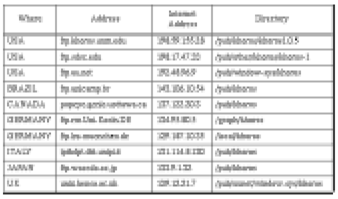
The Khoros 2.0 Developer Release is available from the following anonymous ftp sites:



Where	Address	Internet Address	Directory
USA	ftp.khoros.unm.edu	198.59.155.28	/pub/khoros2.0
USA	ftp.eitc.edu	198.17.47.33	/pub/vtbasch/khoros/khoros-1
SWEDEN	ftp.luth.se	130.240.18.2	/pub/unix/khoros2
ITALY	ftp.unipi.it	131.114.21.18	/pub/khoros2.0
BRAZIL	ftp.unicamp.br	143.106.10.54	/pub/khoros2.0
GERMANY	ftp.hrz.uni-kassel.de	141.51.12.12	/pub/khoros/khoros_2.0
GERMANY	ftp.e18.physik.tu-muenchen.de	129.187.217.8	/pub/khoros/V2
UK	ftp.mcc.ac.uk	130.88.200.12	/pub/cvg/khoros

Pull back the file $KHOROS_FTP/release/install and read it first. Please use the site closest to you.

Khoros 1.0.5 is available from the following anonymous ftp sites:



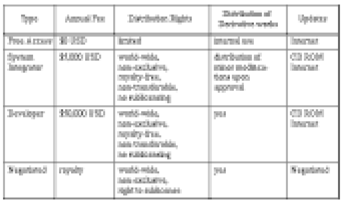
Where	Address	Internet Address	Directory
USA	ftp.khoros.unm.edu	198.59.155.28	/pub/khoros/khoros1.0.5
USA	ftp.eitc.edu	198.17.47.33	/pub/vtbasch/khoros/khoros-1
USA	ftp.uu.net	192.48.96.9	/pub/window-sys/khoros
BRAZIL	ftp.unicamp.br	143.106.10.54	/pub/khoros
CANADA	popcyn.gpsid.uottawa.ca	137.122.30.5	/pub/khoros
GERMANY	ftp.rus.Uni-Essen.DE	134.91.80.1	/graph/khoros
GERMANY	ftp.lrz-muenchen.de	129.187.10.33	/local/khoros
ITALY	ipifidpt.difi.unipi.it	131.114.8.130	/pub/khoros
JAPAN	ftp.waseda.ac.jp	133.9.1.32	/pub/khoros
UK	unix.hensa.ac.uk	129.12.21.7	/pub/uunet/window-sys/khoros

Pull back the file $KHOROS_FTP/release/install.ftp and read it first. Please use the site closest to you.

A CD-ROM and manual set of Khoros 1.0.5 may be purchased for $375 USD from KRI. Contact khoros-request@khoros.unm.edu for more information.

Licensing Costs

There are three standard Khoros distribution licenses; Free Access, System Integrator, and Developer. Rights in addition to those provided by the standard licenses are negotiated on a case-by-case basis. The Terms and Conditions of each license are summarized below:



Type	Annual Fee	Distribution Rights	Distribution of Derivative works	Updates
Free Access	$0 USD	limited	internal use	Internet
System Integrator	$1,000 USD	world-wide, non-exclusive, royalty-free, non-transferable, no sublicensing	distribution of minor modifications upon approval	CD ROM Internet
Developer	$10,000 USD	world-wide, non-exclusive, royalty-free, non-transferable, no sublicensing	yes	CD ROM Internet
Negotiated	royalty	world-wide, non-exclusive, right to sublicense	yes	Negotiated

8.5.2 Support

Provision

No cost email support is available over the Khoros mailing list and comp.soft-sys.khoros usenet group.

KRI offers monthly training courses for both developers and users. In addition, KRI can arrange for on-site training.

KRI also offers specialized consulting arrangements.

Costs

E-mail support is free.

The Developer's Training Course runs $1600 USD per person and the user training course runs $1200 USD per person.

The cost of on-site training and consulting agreements are negotiated.

8.5.3 Information

Manuals

Khoros 2.0 manuals are available via anonymous ftp from ftp.khoros.unm.edu in the /pub/khoros2.0/manual directory.

Note that the khoros_manual/ subdirectory contains the combined manuals for the bootstrap, design, and dataserv toolboxes, which together comprise the core Khoros system. Thus, in this directory you will find:

- Installation Guide [34]
- Getting Started Manual [33]
- Toolbox Programmer's Manual [38]
- Visual Programming Manual [39]
- Introduction to Application Toolboxes [32]
- Programming Services Volume I: Foundation Services [35]
- Programming Services Volume II: Data Services [36]
- Programming Services Volume III: GUI & Visualization Services [37]

Then, there is a different subdirectory for each additional toolbox that is distributed with the Khoros system. Thus, the datamanip/ directory contains the manual for the datamanip toolbox, the envision/ directory contains the manual for the envision/ toolbox, and so on.

KRI distributes printed Khoros 2.0 manuals only to Licensees and Khoros Consortium members. We are currently working on a Book and CD release; KRI has made an agreement with Manning Publications to produce the Khoros 2.0 manual set which will include a CD ROM of the Khoros 2.0 distribution. The manual set will be distributed by Prentice Hall in the summer of 1995.

On-line Information

All Khoros programs have complete man pages available on-line as well as help pages which can be browsed with an interactive viewer. HTML converters are available for the documentation if local hypertext versions are desired.

Network Information Services

Khoral Research, Inc. has a home page on the world-wide-web at

http://www.khoros.unm.edu KRI maintains up to date information on the Web Site. In addition to information on Khoral Research and Khoros there are pointers to many other home pages that may be of interest.

For additional information on Khoros, training, licensing, and support, users can mail to

khoros-request@khoros.unm.edu

Repositories of examples

The Khoros system comes with complete source code so programming examples are always available. Contributed toolboxes containing source are also available.

8.5.4 User groups

There are two ways to get questions answered and to interact with other Khoros users: The Khoros mailing list and the Khoros USENET group. The Khoros mailing list and USENET group are bi-directionally gatewayed to each other (all messages are cross posted). That way you only need to participate on either the mailing list or the USENET group.

To be added to, or removed from, the Khoros Mailing List, send e-mail to khoros-request@khoros.unm.edu with your request. To participate on the Khoros USENET group, subscribe to comp.soft-sys.khoros.

Khoral Research also provides a digestifed version of the Khoros mailing list. Once a day you will receive a single email message with a digest of the previous 24 hours' worth of articles. This message includes a list of topics discussed. To be added to, or removed from the digestified list, send e-mail to khoros-request@khoros.unm.edu with your request.

When you become a System Integrator or Developer Licensee of Khoros, you automatically become a member of the Khoros Consortium. The Consortium began in 1991 to facilitate the free access distribution of Khoros and to further the research and development of software development environments. Today the Consortium focuses on the rapid advancement of the Khoros software infrastructure.

Review of Visualisation Systems

8.6 PV-WAVE

8.6.1 Availability

General information on availability can be fond on WWW at:

http://www.vni.com

Supplier

The UK Academic contact is Nigel Brown:

Visual Numerics Ltd.
New Tithe Court
23 Dictate Road
Slough
BerkshireSL3 7LL

Tel: 01753 790600
Fax: 01753 790601

Email: n.brown@vniuk.co.uk

Platforms

PV-WAVE CL is available as part of the Visual Numerics Ltd. CHEST agreement and is available for the following platforms:

- Convex Unix
- DEC VAX/VMS
- DEC Ultrix
- HP 9000 HP-UX
- IBM RS6000 AIX
- SGI Irix
- SUN SunOS + Solaris

PV-WAVE "Personal Edition" is available for the PC under Windows 3.1 but is not included in the CHEST agreement. The full terms and conditions of the PV-WAVE CL CHEST agreement are detailed below, including costs.

Costs

This is a 5 year CHEST Agreement which commenced on the 23rd March 1994 and terminates on the 22nd March 1999. Institutions may participate at any time during the period; however they will be bound until the end of the Agreement. The Agreement covers both the PV-WAVE family of products and the IMSL C and FORTRAN numerical and graphical libraries. New platforms will be provided within the charges if and when commercially available from Visual Numerics.

The Options for pricing are as follows:

Please note that this information was correct at the time of printing.

8.6.2 Support

Provision

Visual Numerics Ltd. are a small organisation and support for PV-WAVE CL is currently restricted to Telephone contact for those nominated as CHEST technical contacts (see directory below). Email access to Visual Numerics Ltd. is possible but it is preferred that this is used for general information requests only (see directory below).

Costs

8.6.3 Information

Manuals for PV-WAVE Command Language

- PV-WAVE CL Applications Guide - Free Code and Sample Applications: Describes the online application examples provided with PV-WAVE and provides sample code in the book for other examples. Also contains a useful worked interactive session example which is a useful way of getting started.
- PV-WAVE Command Language Guide to Advanced Rendering Library: Provides description and reference guide to the Advanced Rendering Library, which is a group of procedures and functions that provide additional data visualization capabilities for the PV-WAVE Command Language, of which it assumes an existing knowledge.
- PV-WAVE Command Language Overview: Describes the basic features of the Command Language with many examples.
- PV-WAVE Command Language Reference: Consists of a Reference Guide to all the functions, procedures, keywords and system variables in the Command Language. Approx 650 pages.
- PV-WAVE Command Language User's Guide: Description of all the facilities in the Command Language. Similar size to the Reference Manual.
- PV-WAVE Command Language Unix Installation Guide: For the person who has to install the PV-WAVE Command Language product, this describes how to do it on the Unix operating system.

Manuals on the PV-WAVE Point & Click product

- Getting Started with PV-WAVE Point & Click Motif Version: Introduces the Motif version of the PV-WAVE Point & Click product.
- Getting Started with PV-WAVE Point & Click OPENLOOK version: Introduces the OPENLOOK version of the PV-WAVE Point & Click product.
- PV-WAVE Point & Click User's Guide: Provides a detailed description of PV-WAVE Point & Click.
- Installing PV-WAVE Point & Click: Provides instructions on installing PV-WAVE Point & Click on Unix workstations.

PV Software Licensing

For the person installing Precision Visuals software, this guide is concerned with unlocking the software.

On-line Information

On-line help is provided within PV-WAVE CL in the form of a procedure called HELP, and a help system. A separate help utility is available outside of PV-WAVE; "man" pages are not provided.

The help procedure can be used with a number of command line parameters to return status information with regards to the current session and variables in use. The help system can be accessed to obtain on-line information with regards the functions and procedures available both within PV-WAVE CL and outside via a help utility.

Network Information Services

There is a WWW site for VNI Inc.,:

http://www.vni.com

Information Email: info@vniuk.co.uk

There are two Mailbase lists:

chest-pvwave@mailbase
chest-imsl@mailbase

8.6.4 User groups

There is not a formal PV-WAVE User Group but occasional user meetings are arranged by Visual Numerics.

Review of Visualisation Systems

Chapter 9: Strengths Weaknesses Opportunities Threats (SWOT)

Review of Visualisation Systems

9.1 Introduction

This Chapter incorporates SWOT analyses (Strengths, Weaknesses, Opportunities, Threats) for PV-WAVE, AVS, Khoros and IRIS Explorer. The intention of the analysis is to predict how each system will fare in view of anticipated changes in usage patterns which have been judged important by the evaluation group. The analysis is intended to be forward looking and thus where appropriate has been extended to cover features of systems which will become available soon. In the case of Khoros, the analysis refers to Version 2 which at the time of analysis was not generally released yet and was carried out from the documentation available.

9.1.1 What is SWOT Analysis?

SWOT analysis [31] is a technique commonly used in business circles to assist in identifying strategic issues for a company or organisation. If the analysis is to be applied to visualization products some modifications to the technique will be required, however, potentially it will yield useful information about the future viability of various systems. The predictive capabilities of the technique come about from the consideration of each system's strengths and weaknesses in the context of the environment which is seen to present opportunities and threats. The intention is to determine how each system will fare in the light of changes taking place around it.

9.1.2 Strengths and Weaknesses

In business, SWOT analysis is usually applied to one company in order to determine its own strategic direction. When applied to many systems it will be necessary to adopt some model of visualization products which captures all important issues regardless of which product is being analysed. It is then up to the analyst to measure the particular strengths and weaknesses of each product using this model. For instance, if one model parameter is Documentation, the analyst will record this as a strength if this is known to be comprehensive and easy to follow for the system under consideration, and as a weakness if it is poor or non-existent.

For convenience the model has been constructed under a number of sub-headings; Availability, Input/Output, Basic Usage and Advanced Usage. Figure 16 shows the complete model in broad terms. The section describing the *SWOT kit* issued to each analyst goes into more detail about what is required under each sub-heading.

9.1.3 Opportunities and Threats

The next stage is to analyse the environment and this also is done under sub-headings for convenience. An important difference though from the determination of strengths and weaknesses is that the environment is fixed for each visualization system. The sub-headings are Economic Variations, Changing User Population and Changing Hardware and Software Environment, under each of which there are a number of factors against which to consider each system's strengths and weaknesses.

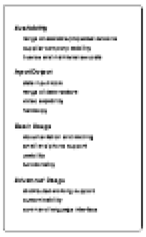


Figure 1: The Complete Model

Economic Variations

- *Purchasing ability* of users and their representatives will remain the same for some years
- *Vendors of visualization systems* will generally experience growth in the coming years. This situation is not to be confused with the stability of an individual company which is assessed as a strength or a weakness
- *Competition* in the visualization field will narrow but become increasingly fierce

Changing User Population

- *Size* of the user population will increase, opening up new areas of interest
- *Skill range* of users will widen, the existing population becoming more competent whilst at the same time novices join the field
- *Use* of visualization systems will increase
- *Rising expectations* amongst users will be the norm
- *Problem sizes* will increase

Changing Hardware and Software Environment

- *Desktop* equipment will become more common
- *Network bandwidths* will continue to increase
- *De facto standards* will have widening support
- *New algorithms and techniques* for visualization will emerge

9.1.4 Analysis

Having classified each feature from the model as a strength or weakness for the particular system under study, these are written down the left hand side of a table as in Figure 17. Along the top are put the environmental factors.

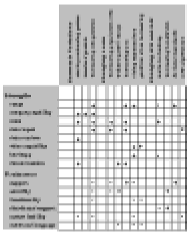


Figure 1: An example of a SWOT table

The analyst then examines the rows against the columns, putting "+" or "-" as follows:

The prevalence of "+" and "-" indicate which qualities will have the most marked effect in the context of the environment and conversely which aspects of the environment will provide the greatest opportunities or threats. Note that features originally thought to be major strengths or weaknesses might register as fairly neutral in the final reckoning.

9.1.5 SWOT Kit

This section describes the SWOT kit issued to each analyst.

Model details

Figure 16 gives a broad view of the features of systems which are of interest. More specifically the following features were considered for each system:

- *range of versions*: the supplier and operating system combinations
- *company stability*: is it stable, viable and in a tenable market position
- *costs*: the capital and recurrent costs, the availability of deals. Consider the difference if any between runtime and development licences
- *data input tools*: any limitations and their ease of use
- *data readers*: pre-written readers for other packages/systems
- *video*: facilities for animation and playback
- *hardcopy*: how easy/difficult is it to produce, what formats are available
- *documentation:* the costs, availability, quality. Include on-line information sources and training
- *support:* include both email and telephone support. Consider the presence of a bulletin board or email list - are facilities free
- *usability:* how easy to use or flexible is the system. How does the performance scale with increasing problem size
- *functionality*: what types of visualization are supported. Are there limitations on the number of independent or dependent variables
- *distributed working*: both remote process execution and via X windows
- *customisability:* is it easy to add new features, what are the tools provided to assist
- *command language:* the facilities for running scripts of commands, batch working, audit.

Preparing the table

Each of the model features were then considered in turn to decide whether it represented a strength or weakness for the system being analysed. The feature was placed in the appropriate section of the table, remembering that no feature can appear in both sections.

Filling in the table

Returning to the first feature for the system being analysed, consideration was given to each predicted environmental change in turn. Any warranting a "+" or "-" were filled in, with comments to justify the entry. It was not intended to fill in every box on every row, only those features of special note. Comments had to be as specific as possible, since any statement which is true in every context has no value in differentiating between systems. The process was repeated for each feature of the model in turn and a summary given commenting upon noteworthy features which had emerged.

9.1.6 Mediation

The SWOT analyses represent the considered views of individuals or small groups working on each system. The results were discussed by all the authors together at an evaluation meeting held at the University of Manchester on 19 and 20 July 1994, where changes were agreed to try to ensure consistency between the analyses. Subsequently each analysis was made available to the system supplier in order to check for factual correctness. The results are presented in good faith, however, it must be recognised that the nature of the technique precludes any claim to absolute certainty in respect of the future of the systems considered. Tables summarising the results for each system can be found in figures 18, 19, 20 and 21.

Review of Visualisation Systems

9.2 Application Visualization System (AVS)

9.2.1 SWOT analysis

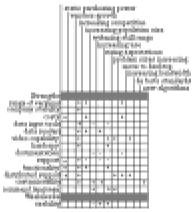


Figure 1: AVS SWOT Table

9.2.2 Comments

Range of versions: Strength

AVS appears on a wide range of UNIX workstations (but needs colour), and on Dec VMS, Cray, and Convex. It has also been ported to the KSR by University of Manchester. A version for powerful PCs (Windows NT and Windows 95) will appear with AVS6 next year.

- Increasing Competition: AVS has the advantage of having been in use on commonly available platforms for several releases and as such is a mature product
- Increasing Population size: AVS is likely to support hardware platforms available to the new user
- Moving to the Desktop: AVS is available on a wide range of platforms, including typical desktop machines, such as colour Suns, SG Indigos, HPs, DECs and PCs (next year) and has been tailored to take advantage of special hardware features and local graphics libraries

Company stability: Strength

- Increasing Competition: Although AVS are reliant on just two products (i.e. AVS and UNIRAS) they appear to have a strong position in the general visualization market. They were the first to market such a product and have held on to their early customers. Their proven track-record continues to attract new customers from a wide range of application areas. The development team grows every year and is now in the 20s, with the company apparently committed to vigorous and continuing improvement of their software.

  AVS/UNIRAS Ltd., is the UK subsidiary of Advanced Visual Systems Inc. and currently has eight full time employees with an expected increase in 1995.

Costs: Strength

- Static Purchasing Power: A CHEST deal (2.5 years remaining) is available whereby a single fixed charge each year allows academic sites unlimited use of the base software on a wide range of platforms. The current CHEST deal does not include the Chemistry or UCD viewers but example networks are provided. It is also possible if there is enough interest in a particular *add-on* that it can be added to the CHEST deal, e.g. the Animator is obtained at an additional but reasonable cost.
- Increasing Population Size: The CHEST deal makes AVS attractive to a growing population
- Move to the Desktop: With the CHEST deal no further costs are involved in moving to a desktop system. It is not necessary to buy additional licences or new versions as common desktop platforms are included in the site-wide deal.

Data input tools: Strength

- Widening Skill Range / Rising Expectations: AVS has an interactive facility, the AVS Data Interchange Application (ADIA), to import data into AVS fields. To access this use the **field descriptor** module from the network editor. Having saved the data format as a template it is then possible to re-use this for other datafiles with the **data dictionary** module. These tools only allows the user to import field data and not polygonal or UCD (e.g. Finite element) data, so it may prove limiting in view of users' rising expectations. It can cause problems to novice users too that may need to specify offsets in the data file by number of bytes (a graphical interface where the user can select with the mouse where new data starts would be easier to use).
- Increasing Competition: Although the data input tool will not handle all data and may cause problems for novice users, it is considered an asset to have an interactive tool to import data.

Data readers: Strength

- Increasing Competition: AVS have a large number of data readers, both in the public domain and from commercial sources (e.g. AVS, Tessella)
- Increasing Population size: With the large amount of example code available, developing a new reader can be a relatively easy task for a new user.
- Increasing Use: As the user uses AVS for more data visualization, there is a good chance that a data reader will already be available for the new tasks.

Video capability: Strength

- Increasing Competition / Rising Expectations: There are a number of ways to create animation and video with AVS. For simple animations modules such as **animated integer** can step through a range of values, which can be used to change parameters. For more complicated animations the Animator tool is of great benefit, and can be used to control changes in parameters, data input, camera positions and even networks. The **Animator** is a key-frame animator, and has 2 levels of control panels, one for newer users or less complicated animations, the other for advanced use. The simpler panel shows a small subset of fairly simple functionality, such as add/delete keyframes and play. The full panel gives greater control over the time between each keyframe. Although the **Animator** is probably beyond absolute novices, slightly more experienced users will find this of great benefit.

  In addition it is possible to generate an MPEG movie from an animation (i.e. sequence of images) using a public domain module from the International AVS Centre (IAC). This module can be found in avs modules/data output/**Create MPEG**, but note this does not generate MPEG directly but calls the standard mpeg encoder.

- Problem size increasing: The ability to create quality animations stored to video is an advantage in the case of increasing problem sizes since it is not always possible to view the visualization in real time
- Increasing Bandwidth: Increased bandwidths will make the strong animation facilities even more attractive

Hardcopy: Strength

- Increasing Competition / Rising Expectations: Support for PostScript for hardcopy output is good. It is also possible to convert an image to CGM. The expected addition of UNIRAS functionality in AVS6 will improve hardcopy support to cover many more devices and formats. Advanced facilities are available, for example, views which have a mixture of shaded rendering and line drawing can be saved as a mixture of lines and image in PostScript so as to make best use of hardcopy devices.

Documentation: Strength

- Widening Skill Range / Increasing Use: The documentation can prove difficult for novice users (e.g. no Getting Started guide) and as a reference source can be missing necessary information. However it is readable and for many users it is possible to find the required information and so on balance the documentation is considered a strength.
- Move to the Desktop: The user will probably have to purchase a paper copy of all of the documentation as it seems unlikely they could manage with the small amount of on-line information available

  The next version of AVS (AVS6) though will have online, context-sensitive, hypertext help system based on the Bristol Hyperhelp for Unix and Winhelp for Windows.

Support: Strength

- Static Purchasing Power: The support of AVS is included in the CHEST licence arrangements
- Increasing Competition: AVS currently provides good technical support in the UK and access to the larger support team in the US when necessary.
- Increasing Population Size and Use / Rising Expectations: Could overstretch the resources of the support team.

Usability: Weakness

- Widening Skill Range: Novices often find the system difficult to use, for example, where several modules appear to do the same thing, such as **isosurface** and **ucd iso**. In the next release of AVS, 6.0, expected next year, an object oriented approach and improved general data types means a single **isosurface** module will be available.
- Increasing Use: Poor usability becomes less noticeable once the user understands which modules to choose and why.
- Rising Expectations: Apart from the many example visualizations available in the form of networks, some of which are packaged into a menu-driven viewer (**dataviewer**), it is also possible to purchase customised data viewers (e.g. for UCD, and chemistry). Therefore it may be possible to visualize new data without the development of new code.

Functionality: Strength

- Static Purchasing Power: With strong functionality the need for alternative systems is reduced
- Increasing Competition: A large number of modules are available in the Public Domain (IAC) and within AVS itself.
- Rising Expectations/Increasing population size: As the users' expectations rise there is a good chance with the large number of modules available that they can find what they need

Distributed support: Strength

- Static Purchasing Power: AVS has good support for distributing modules across a number of platforms to share the workload for an application, allowing the user to make better use of the available resources
- Increasing Competition: The ability to distribute modules is considered an asset.
- Problem size increasing: As the problem size increases it is possible to distribute the task across a number of platforms, either by taking advantage of larger, more powerful machines or splitting the task so each part can be processed separately.
- Move to the Desktop: When a user moves to a desktop system they can take advantage of the distributed support in AVS to farm out difficult and large problems.
- Increasing Bandwidth: With increased bandwidth the transfer of data will improve. Since distributing applications often involves transferring data from one system to another greater benefits in speed can be expected from distributing large, complex problems.

Customisability: Strength

- Widening Skill Range: A user with relatively little experience can expect to be able to customise AVS. With a small amount of experience users can produce an application to process simple data. The user interface can be modified to some extent without any programming (e.g. by using the mouse the user can switch between typed in values and dials or sliders). For more experienced users it becomes possible to add new tools. For advanced users with access to the developers' kit it is possible to tailor all AVS modules or produce an application that just uses the required parts of AVS (e.g. a scaled-down geometry viewer). This will become easier with the next developers' kit (Express) which is the underlying system for AVS6.
- Increasing Competition/Increasing Use: The ability to customise applications to end-users' needs is of great benefit. With AVS6 there will be even more control for the expert, including the ability to modify Motif-type interfaces too.
- New Algorithms: Since AVS can be customised it is possible for new algorithms to be incorporated as modules into AVS. There is also a large amount of example source code in the public domain (e.g. IAC) which may be tailored to include new algorithms.

Command language: Strength

- Widening Skill Range: Although not considered suitable for novices the command language, CLI, is extremely useful for scripting applications and demonstrations. Less experienced users can rely on AVS itself to record scripts, which can then be tailored. CLI scripts can be used to build and change networks as well as to change parameters. It is possible to journal a sequence of events by starting AVS with the -cli option and opening a script for saving events with script -open file - this does not record all events (e.g. geometry animations) but they can be added in afterwards by hand. It is also possible to develop modules that call CLI commands so networks can change to handle different events/data.

  V, the replacement for CLI in AVS6, will have even more powerful commands and should be easy to master for those already familiar with Unix shell scripts.

- Rising expectations: For the reasons above (in Widening Skill Range) CLI can be a powerful facility. In AVS6 the command language will be even more powerful with programming type control and expression evaluation.
- Problem size increasing: With increasing problem sizes it is often useful to be able to produce a batch of commands which can be left unattended to produce a visualization (although this does require the use of a graphics screen for display).

9.2.3 Summary

The AVS product is a system which allows users to visualise their data by constructing applications from a series of software components called modules. Each module performs a specific task and some of these include:

- importing data
- processing and filtering data
- mapping data onto geometric primitives or an image representation.
- rendering the geometry or images

Some of the features of the software allow the system to be extended by users integrating application and visualization code into AVS by writing new modules. The system also allows the user to change and edit the user interface and layout to customise the application. This makes it a very powerful tool for prototyping.

The maturity of the AVS product is one of its strengths. The first version of AVS (AVS1) was released in April 1989 and it is now currently in its fifth version (AVS5). The new development environment AVS/Express was released during 1994 and the summer of 1995 will see AVS6 which will be based upon the new architecture of AVS/Express bringing many new improvements.

There is also a very good infrastructure in place for the AVS system in the public domain with the International AVS Center's activities in the US collecting and porting user contributed modules.

One of the disadvantages, as with most of the application builders, is the complexity of the system for the first time user. This leads to an initial steep learning curve which can be an obstacle for novices. Improvements in on-line help and tutorial facilities would alleviate this and the releases of AVS6 may address these.

Review of Visualisation Systems

9.3 IRIS Explorer

9.3.1 SWOT analysis

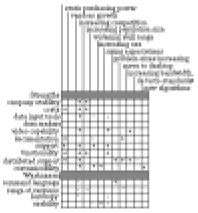


Figure 1: IRIS Explorer SWOT Table

9.3.2 Comments

Version

The version of Explorer used for this analysis was 2.0 running on Silicon Graphics equipment. All comments are based on this version unless otherwise stated.

Range of versions: Weakness

This is only a temporary weakness - SGI, SUN (SunOS) and IBM RS/6000 are available now, HP 9000/700 and SUN (Solaris) are promised by the end of the summer and DEC Alpha sometime after that. There is also a Cray version, and other supercomputer versions are being discussed.

- Increasing Competition/Increasing Population Size: A weakness since the range of machines currently supported is less than the competition.
- Move to Desk Top: Some notable workstation vendors missing, and no PC version.

Company stability: Strength

NAG have been in existence for over 20 years; their turnover is steadily increasing; their staff complement has been stable over the recent difficult period in the economy. It is not-for-profit so surpluses are fed back to the benefit of the company.

- Increasing Competition: Should do well since NAG has a high professional reputation built over many years to trade on.
- Increasing Population Size: The company are well experienced in the support of a large user community and thus we can expect them to be able to scale their activities to match an increasing user base.

Costs: Strength

It is currently bundled with SGI workstations. This will change, but NAG Ltd. have publicly committed to a non-expensive pricing strategy. Educational sites will receive a 50% discount on both the one-time fee & the support service. Support costs will be approximately 15% of the one-time fee. Current SGI users, who up to now have received it free, will be offered support and maintenance for an unlimited number of users for the price of the 4-user support service up to Release 3.0. For the period of probably a year after Release 3.0, these sites will be offered the opportunity to increase the scope of their licence to include more users for the difference in support charge only. Those who do not take up the offer before Release 3.0, and new sites wishing to receive IRIS Explorer, will have to pay the one-time payment which ranges from 1000 for a single user licence to 20,000 for an unlimited user licence. These costs refer only to 1 implementation; an additional fee of 500 (licence) and 75 per annum (support) is charged per extra implementation.

- Static Purchasing Power: Once the one time fee is paid for one version, there is only a very small additional fee for additional implementations. Also, it is possible to pay a one time unlimited user fee.
- Increasing Population Size/Increasing Competition: Relatively low price product will attract new customers.

Data input tools: Strength

DataScribe is a flexible, visual programming tool. It allows the user to easily create a data reader that will convert from any data file format to an Explorer data type. This does require, however, knowledge of the data file format and some knowledge of Explorer data types and their uses. DataScribe can also be used in *reverse* to output data from IRIS Explorer according to any desired format. Indeed, it can be used as a general data conversion tool between two external formats.

- Increasing Competition: Having good input tools and the ability to create data readers is a plus, but at present, as with other similar systems, the tools provided are less than ideal. This is due mainly to the fact that they only support Lattice datatypes, not Pyramids, and can be difficult for novice users.
- Widening Skill Range: Data input is rarely easy, and DataScribe would be difficult for novice users.
- Rising Expectations: Users expect tools that cover everything, but DataScribe does not handle some of Explorer's data types.

Data readers: Strength

A range of readers are provided (e.g. AVS types, HDF, PDB, Plot3D). Others (e.g. **ReadMarc**, **ReadMesh1**, **ReadMultiPlot3D**, **ReadNastran**, **ReadSEGY**) are available as either unsupported (they come on the installation disk, but are user donated modules) in version 2.2 or available through Explorer ftp sites (e.g. netCDF).

Video capability: Strength

Facilities to animate the viewing position are provided; and there are loop controls in the visual programming language. There are modules to control a video recorder; and a module to interface to MovieMaker on SGI.

- Increasing Competition / Rising Expectations: This is increasingly important in scientific visualization. Linking them into existing software is a plus.
- Increasing Bandwidth: Distributed multimedia become technically feasible, and IRIS Explorer has tools to exploit them.

Hardcopy: Weakness

At present, it does not seem possible to generate CGM. Postscript is only available through the Render module: it will generate colour Postscript with options to set an output size in inches and setting the DPI; it can be sent direct to a printer or into a file. It is also possible to output GIF.

- Increasing Competition / Rising Expectations: to be able to produce output which can be sent directly to a hardcopy device should be standard. The only method of doing this is from the **Render** module. Others such as **Graph**, **DisplayImg** and **Histogram** do not support this.
- A new module, available soon from NAG, **NAGGraph** will be able to output postscript.

Documentation: Strength

Detailed manuals (in the form of a User's Guide, Module Reference Manual and a Module Writer's Guide), plus on-line help (UNIX man pages) for Explorer subroutines and on-line help pages with each module.

- Static Purchasing Power: The documentation is to be found on line, hence no need to buy large quantities of manuals.
- Widening Skill Range: It should have a "Getting Started" guide as this is increasingly important.
- Move To Desk Top: On-line documentation becomes more important as we move to large numbers of desk-top users and the cost of hardcopy documentation becomes prohibitive.

Support: Strength

IRIS Explorer Centres have been established in the UK and Japan, and there is support in the US. These act as one-stop shops for the product. The support service licence also includes phone/fax/E-mail support, Users Newsletter and regular CDs containing tested and implemented modules. Also NAG Bulletin Board is a source of up-to-date information with an IRIS Explorer section; this is available on WWW at: http://www.nag.co.uk/1/visual/IE/iecbb

- Static Purchasing: Support comes in the package at present, but after Release 3.0 comes at 15% of the one-time fee.
- Increasing Competition / Rising Expectations: Support is well organised and is an important selling point.
- Widening Skill: Naive users and expert users can use same query mechanism.

Usability: Weakness

IRIS Explorer (for the naive user) needs some time to learn, and is not suitable for the occasional user (who would be better with a menu driven system). Conversely its flexibility makes it highly usable by the expert. It needs a workstation of reasonable processor power and memory for worthwhile use.

- Rising Expectations: This is an obvious weakness as users expect to be able to "turn it on and go".
- Widening Skill Range: Naive users will be exposed to the steep learning curve.
- Increasing Use: Familiarity makes the system easier to use. Its flexibility means that experts cannot easily outgrow the system.

Functionality: Strength

- Static Purchasing Power: With strong functionality the need for alternative systems is reduced.
- Increasing Competition: Explorer provides a large range of modules and more may be found on public ftp sites.

- Increasing Population: Rich functionality should enable it to be used throughout the community.
- Rising Expectations: As user expectations rise there is a good chance that with the large numbers of modules around they will find what they need.

Distributed support: Strength

The design of IRIS Explorer is focussed on a distributed execution model. It is easy to place computationally intensive modules on separate compute engines. The **Render Remote** module explicitly provides for rendering to be carried out on a separate workstation. At present, on SGI, it uses GL rather than X so remote execution with local viewing can be a problem. Version 2.2 and above uses OpenGL so remote viewing is no longer a problem (with IRIX5 and above).

- Static Purchasing Power: As the range of versions increases it will be possible to run modules on a variety of platforms so distributing the load.
- Increasing Competition: To be able to distribute the load is what future users will expect so is an obvious strength.
- Problem Size Increasing / Move To Desk Top / Increased Bandwidth: Can use remote high performance computer to *number crunch* but display locally.
- De Facto Standard: Does not as yet adhere to X standard.

Customisability: Strength

IRIS Explorer has two important customisation features:- (1) Module Builder is a visual programming tool that allows the user to write their own code and then compile it into a new module. (2) Modules can be grouped to provide simpler interfaces for novice users (cf turnkey).

- Increasing Competition / Increasing Use: It is an obvious strength to be able to produce your own modules to do tasks related to your specific area of interest.
- Widening Skill Range: At one extreme it can be customized for the naive user by grouping modules and providing a simplified interface; at other extreme, the advanced user can write their own modules
- New Algorithms: Can add new algorithms as modules as they become available.

Command language: Weakness

IRIS Explorer provides a command language called SKm (scheme) which can be run interactively or from a script file. The language allows the functionality of the map editor with the launching/destroying of modules, connection/disconnection of ports, the setting of parameter values and loading/saving maps etc. It is also possible to define procedures involving modules and values. The system allows a limited form of batch working.

- Increasing Competition: There is no facility present to script an interactive session so that it can be played back.
- Problem Size Increasing: The production of scripts for limited batch working allows large problems to be worked on without interaction, viewing only the end result.

Summary

A state of the art visualization system from a UK company with a long history of supplying the higher education community with high quality software. IRIS Explorer is an application builder which makes it highly flexible as a visualization tool. It uses a visual programming interface to build networks of modules to form visualization maps, it allows the incorporation of new/user written modules to extend the functionality if the need arises and it supplies tools to aid the generation of these new modules.

There are options to extensively customise the appearance of maps by grouping the modules together and allowing the user to define their own control panel by selecting and re-arranging widgets. This customisability is extended by allowing these groups to be saved and used by other, more novice, users as turnkey applications.

One of the strengths of the IRIS Explorer software is that the system architecture was explicitly designed to ensure its application across hardware platforms in heterogeneous network environments. This, combined with NAG's expertise in porting software, will ensure that the user is able to fully exploit the easy to use remote module execution system of Explorer for distributed working.

Explorer provides a good graphical user interface, DataScribe, for writing data import modules. Although it is difficult for novice users and applies only to Lattices it is still a powerful tool with many useful features. These features should allow Explorer to do well with respect to the increase in competition.

The down side, as with most other flexible systems, is the rather steep learning curve. Also a lack of good hardcopy output, this being a screen dump only available from the render module in encapsulated postscript format, and a poor command language do not do it justice.

Review of Visualisation Systems

9.4 Khoros

The analysis for the SWOT was performed from Beta release versions of the manuals. At the time of compilation of the results there was no access to information on the Data Input Tools and Data Readers.

9.4.1 SWOT analysis

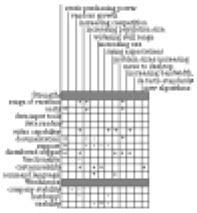


Figure 1: Khoros SWOT Table

9.4.2 Comments

Range of versions: Strength

- Increasing Competition: Khoros is available on a wide range of machines which increases its appeal in the face of competition. All versions are directly supported by Khoral Research, Inc.
- Increasing Population Size: likely to be supported on hardware platforms available to the new user
- Move to Desktop: some desktop machines are supported

Company stability: Weakness

- Vendors' Growth / Increasing Competition: revenue for Khoral Research, Inc. comes from government contracts, training, product sales, support, and licensing. KRI is a startup company, and does not yet have all the resources that other, more-established vendors have.

Costs: Strength

- Static Purchasing Power: users will always be able to afford the software because it is free, distributed via the Internet as source code and binaries
- Increasing Population Size: being freely available it is attractive to a growing population
- Move to Desktop: no further costs are involved in moving to a desktop system

Video capability: Strength

- Increasing Competition / Rising Expectations: the Animate program is an interactive image sequence display tool which allows the user to either input a sequence of images or supply a basename that describes a set of images contained within a number of files. The user is presented with a video recorder style interface with which to move through the image sequence. Using this program together with the tools to annotate images and control within the visual programming language such as do-loops and if-then-else constructs provides strong support for video
- Problem Size Increasing: strong video capabilities are an advantage in the case of increasing problem sizes since it is not always possible to view the visualization in real time
- Increasing Bandwidth: increased bandwidths make the strong video capabilities even more attractive

Hardcopy: Weakness

- Increasing Competition / Rising Expectations: weak provision for hardcopy will detract from the system's attractiveness in the face of competition and users' rising expectations

Documentation: Strength

- Static Purchasing Power: documentation is freely available in PostScript by FTP for printing locally in desired amounts. The Khoros 2.0 manual set will be available for at low-cost from bookstores world-wide in the second quarter of 1995.
- Move to Desktop: it is also available on-line
- Widening Skill Range / Increasing population size: there is a lack of documentation at the novice and expert ends of the spectrum

Support: Strength

- Static Purchasing Power: support is via email, from users and the Khoral Research, Inc. who also maintain extensive FAQ (Frequently Asked Questions) files. This service is free and response is normally very good.
- Increasing Population Size / Use: could overload newsgroups and email lists
- Rising Expectations / Widening Skill Range: users will expect more than just email and newsgroup support; technical queries will be increasingly difficult to handle in this way

Khoral Research, Inc. offers monthly training courses and various consulting contracts.

Usability: Weakness

The Khoros 2.0 version of Cantata contains something called a Finder, which allows for module searches based on keywords.

- Widening Skill Range / Increasing Use: the systems comprising Khoros all support the data flow model, but it can be difficult to find the appropriate module for a certain task or data. This will be more marked for novices, but poor usability becomes less noticeable once the user understands which modules to choose and why

- Rising expectations: users would not expect to learn about specific data types onto which they must map their data; they would rather have the system import the data

Functionality: Strength

- Increasing Competition: Khoros has strong functionality but there are gaps in respect of Finite Element analysis. There is however a contributed toolbox from LBL is available which contains basic FEA functionality.

Distributed support: Strength

- Static Purchasing Power: distributed support exists in the form of a daemon which handles remote task execution and data transfer. Users can make best use of existing hardware, e.g. to share the workload of an application
- Increasing competition: these facilities will prove to be an advantage when comparing to other systems
- Problem Size Increasing: jobs can be distributed across a number of platforms
- Move to Desktop: local processing can be carried out on the desktop with compute-intensive parts carried out remotely
- Increasing Bandwidth: benefits in speed can be expected when distributing processing

Customisability: Strength

- Increasing Use / New Algorithms: Khoros is customisable at a number of levels. New functionality can be added or existing functions can be altered to suit users needs
- Widening skill range: can customise the system for novice users

Command language: Strength

- Problem Size Increasing: all programs in Khoros can be activated and controlled from the command line using the command line user interface (CLUI). This provides *true* batch facilities to produce a visualization unattended Furthermore, the encapsulated workspace functionality of Cantata makes the creation of a batch process automatic. Users can collapse a workspace down into a command line program.
- Widening Skill Range: The journalling capabilities in the Khoros 2.0 Developers Release were not robust across the different widget sets and have been disabled. This functionality may be re-enabled in the future.

9.4.3 Summary

Khoros is an integrated software development environment that allows users to compose and perform a variety of tasks related to image and signal processing, medical imaging, remote sensing, ecological research, data exploration, scientific visualization, X-window application development and other application specific domains.

Khoros includes a visual programming language, a suite of software development tools that extend the visual language and help you create new applications, an interactive user interface editor, an interactive image display package, 2D/3D plotting, and an extensive suite of image processing, data manipulation, scientific visualization, geometry and matrix operators.

Review of Visualisation Systems

9.5 PV-WAVE

9.5.1 SWOT analysis

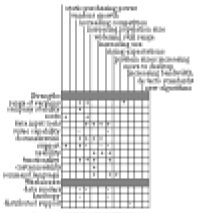


Figure 1: PV-WAVE SWOT Table

9.5.2 Comments

Range of versions: Strength

- increasing competition: the software is available on a wide range of platforms, from PC to Super Computer, more readily than some of the competition
- increasing population: readily available and usable on commonly available H/W in the community
- move to desktop: MS Windows 3.1 version available. Development taking place under Windows NT

Company stability: Strength

- increasing competition: Visual Numerics Inc formed in 1993 through merger of Precision Visuals and IMSL means the new company should have greater stability and increased product range

Costs: Strength

- increasing population / static purchasing power: due to be announced as a CHEST deal in the very near future, ensure pricing is competitive.

Data input tools: Strength

- changing users (all factors): data input facility is flexible and easy to use, suiting all user environment factors.

Distributed support: Weakness

- increasing bandwidth: Graphics based on X11 only, no true distributed support as with AVS, Explorer etc.

Data readers: Weakness

- increasing competition: data readers for data from other applications not readily available, leaves package susceptible to competition
- increasing population: lack of data readers may mean the package is less attractive given an environment change.
- rising expectations: data readers might not meet rising expectations

Video capability: Strength

- increasing competition / rising expectation: animation sequences can be built facilitating flip book animation.

Hardcopy: Weakness

- increasing competition / rising expectations: limited H/C facilities likely to mean package is susceptible to competition, unlikely to attract increasing use and not meet expectations

Documentation: Strength

- increasing competition: existing documentation is very good with compact easy to follow manuals
- increasing use / widening skill range: manuals are structured, from overview, getting started, users guide to comprehensive reference guide
- increasing population: overview and getting started documentation is very good

Support: Strength

- static purchasing power / increasing competition: technical support in the UK is very good; responses are prompt and support staff are very knowledgeable - problems can be referred to US if required although the UK office does appear self sufficient in many ways
- increasing population / use / rising expectations: limited number of support staff in the UK could mean problems with these factors e.g. softkeys are currently generated in the US only

Usability: Strength

- widening skill range: Command Language is both easy to use and is extensible for the competent / computer literate user
- increasing use: Command Language is easy to use, particularly for anyone with programming skills
- rising expectations: extensibility of the Command Language environment should fulfill rising expectations

Functionality: Strength

In addition to the standard functionality the Advantage product has the full IMSL C library routines integrated into it.

- increasing competition: functionality is comprehensive. Should enable PV-WAVE to remain at the forefront
- widening skill range: functionality suited to technical and non-technical users
- increasing population: rich functionality should enable PV-WAVE to be useful throughout the community
- rising expectations: rich functionality should match rising expectations

Customisability: Strength

- widening skill range: extensibility is attractive to competent technical users

Command language: Strength

- widening skill range: command language is usable across a wide skill range, readily suited to the novice and specialist alike.
- rising expectations: command language likely to meet rising expectations, given the available programming constructs.
- problem size increasing: ability to use in batch mode

9.5.3 Summary

A very good general purpose package, providing rich functionality suited to a wide skill range of users, although more particularly those with programming experience. PV-WAVE is complementary to Uniras and AVS in the community, and not in competition with them.

Strengths which will carry the system forward are its data input tools and functionality. Some aspects of support will require attention by the suppliers in view of anticipated environmental changes. Ready-made data readers, and hardcopy facilities, represent the most significant weaknesses. The expected widening skill range of users is the chief environmental factor showing advantage for this system.

Annex to Chapter 9:
Strengths Weaknesses Opportunities Threats

Review of Visualisation Systems

Annex to Chapter 9: Strengths Weaknesses Opportunities Threats

9.6 IBM Data Explorer

9.6.1 Introduction

During the review of the visualization systems an in-depth SWOT analysis was not performed for IBM Data Explorer. This omission should *not* be taken as a reflection on the features provided by the IBM Data Explorer system. This annex will provide similar information for IBM Data Explorer that can be found in chapter 9 for other visualization systems. This annex will either provide that information under the equivalent section headings or references to other areas in this report where that information can be found.

The final section of this annex contains some information on the powerful data manipulation facilities available in IBM Data Explorer.

9.6.2 Additional Information for IBM Data Explorer

*Ra*nge of versions

IBM Data Explorer is available on a range of Unix workstations and the complete range of versions can be found in section 8.3.1 on page 111.

Company stability

International Business Machines Corporation (IBM) has a long history in the computing industry and is involved in almost every area of computing hardware and software. For more specific information on the company and products you can access the IBM WWW home page:

http://www.ibm.com

Costs

Price information can be found in section 8.3.1 on page 111.

Data input tools

Information concerning importing data into IBM Data Explorer can be found in section 3.2.3 on page 24 and section 4.3 on page 66.

Data readers

Information on data readers and their location can be found in section 4.3 on page 66

Hardcopy and Animation Facilities

These features are detailed in section 5.2 on page 80.

Documentation

The full range of manuals are listed in section 8.3.3 on page 112.

Support

Contact addresses and arrangements for support are in section 8.3.2 on page 112.

Functionality

A review of the functionality in IBM Data Explorer can be found in section 3.2.3 on page 24 and some information on data manipulation is contained in section 9.6.3.

Distributed working

Details of these facilities can be found in section 7.2 on page 103.

Customisability

Information on incorporating application code into the IBM Data Explorer system can be found in section 6.3 on page 92.

Command language

IBM Data Explorer has an extensive scripting language. To execute IBM DX in script mode you simply type:

```
dx -script
```

More details on the scripting language can be found in the IBM Data Explorer User Guide [26].

9.6.3 Data Manipulation

Sometimes the scientist using the visualization system wishes to select a subset of the data to process or wants to apply an expression to the dataset to produce a resultant data component e.g., vector magnitude from a vector field. Most of the visualization systems provide support in the form of modules to slice and crop the data and other modules to apply simple mathematical expressions to data components.

If the scientist however wants to select specific data components from a dataset based on some conditional expression then there is only minor support through modules in the various systems. The user must resort to coding the task in the form of an additional module if any extensive manipulation is required for example see [64].

IBM Data Explorer provides extensive support for both of the above cases through the **Compute** module. This module applies an expression point-by-point to the input data producing an output dataset. The module also allows multiple input fields to be combined via the expression supplied and individual data components within a dataset can be referenced in the expression. Some of the operations available in the **Compute** module are:

- Trigonometric, Hyperbolic and Logarithmic functions
- Unary, Binary and Vector functions
- Type conversion
- Logical and Conditional expressions
- Bitwise operations

The user can construct complex expressions for example, Field1 is a vector field, Field2 and Field3 are scalar fields. The expression evaluates on a point-by-point basis an output field which is the magnitude of Field1 added to the quantity Field2 divided by 4.5 times Field3:

```
output_field=Compute("mag($0) + $1/($2*4.5)", Field1, Field2, Field3);
```

More information on the **Compute** module can be found in [27].

Review of Visualisation Systems

Chapter 10: Conclusions and Summary

Review of Visualisation Systems

10.1 Conclusions

We set out to review a number of visualization software packages that are available for use in the UK higher education community. We also required that the software we reviewed met certain minimal requirements - which we described in Chapter 1: Overview.

Conventionally, the conclusions would be our summing up of the comparative performance of the systems. In this report, we have taken a different approach. Following the basic factual information in chapters 2 to 8, we have made a detailed SWOT study, in Chapter 9, which in effect stand as our conclusions.

One could make an overall general comment, that, since AGOCG published the previous visualization software evaluation, the visualization software on the market has been comparatively stable - the changes have taken place in the companies, but not the products.

We offer this report in the hope that it may be useful to potential users deciding what to buy and to new users trying to put their chosen system in context.

Review of Visualisation Systems

10.2 Acknowledgments

The review members would like to thank the Advisory Group on Computer Graphics (AGOCG) for funding during the production of this report.

We would also like to thank the vendors of the various visualization systems for their comments on early draft versions of this document.

The people involved with the review would also like to thank Ms Mary McDerby at the Computer Graphics Unit, MCC for all her help with local meeting arrangements and mailings to members of the group and Paul Lever for his assistance and input to the review. Also thanks to all the staff in the Reprographics Department in Manchester Computing Centre for their help with the production of the many draft versions of this document.

Review of Visualisation Systems

Chapter 11: References

[1]
"AVS Applications Guide", Advanced Visual Systems Inc.

[2]
"AVS Chemistry Developers Toolkit Guide", Advanced Visual Systems Inc.

[3]
"AVS Developers Guide", Advanced Visual Systems Inc.

[4]
"AVS Module Reference", Advanced Visual Systems Inc.

[5]
"AVS Technical Overview", Advanced Visual Systems Inc.

[6]
"AVS Tutorial Guide", Advanced Visual Systems Inc.

[7]
"AVS5 Update manual", Advanced Visual Systems Inc.

[8]
"AVS Users Guide", Advanced Visual Systems Inc.

[9]
G. Bancroft and F. Merrit and T. Plessel and P. Kelaita and R. McCabe and A. Globus, "FAST: A Multi-Processing Environment for Visualization of CFD", Proceedings of Visualization 90, IEEE Press, 1990.

[10]
A J C Belien, "Comparison of Visualization Techniques and Packages", Stichting Academisch Rekencentrum Amsterdam, ISBN 90-72490-08-8 (1993)

[11]
K W Brodlie et al, "Scientific Visualization: Techniques and Applications", Springer Verlag 1992

[12]
K W Brodlie, "Methods for drawing curves", Fundamental Algorithms for Computer Graphics, Ed. R.A. Earnshaw, pages 303-324, Springer-Verlag 1985.

[13]
S. Bryson, C. Levit, "The Virtual Windtunnel:An Environment for the Exploration of Three Dimensional Unsteady Flows", NASA Ames Research Center RNR Technical Report RNR-92-013,April,1992.

[14]
N.Dyn and D.Levin and S.Rippa, "Data dependent triangulations for piecewise linear interpolation", IMA Journal of Numerical Analysis, 1990.

[15]
"Evaluation of Visualization Software", AGOCG Technical Report 9, January 1992.

[16]
T.A.Foley and G.M. Nielson, "Modelling of Scattered Multivariate Data", Eurographics 94 State of the Art Reports, Eds C. Giertsen and P.A. Fevang, pages 38--59, Eurographics Association 1994.

[17]
J R Gallop, "Underlying Data Models and Structures for Visualization", from Scientific Visualization: Advances and Challenges, edited by L Rosenblum et al, Academic Press, 1994

[18]
R. B. Haber, B. Lucas and N. Collins, "A Data Model for Scientific Visualization with Provisions for Regular and Irregular Grids", Proceedings of IEEE Visualization `91, IEEE Computer Society Press 1991.

[19]
R. B. Haber and D.A. McNabb, "Visualization Idioms: A Conceptual Model for Scientific Visualization Systems", Proceedings of IEEE Visualization `90, IEEE Computer Society Press 1990.

[20]

J. L. Helman and L. Hesselink, "Representation and Display of Vector Field Topology in Fluid Flow Data Sets", IEEE Computer, pages 27--36, 1989.

[21]
S Hill, "Tri-linear interpolation", Graphics Gems IV, Ed P. S. Heckbert. pages 521-525 AP Professional 1994.

[22]
A.J.S. Hin, "Visualization of Turbulent Flow", 1994.

[23]
J.P.M. Hultquist, "Interactive Numeric Flow Visualization Using Stream Surfaces", Computing Systems in Engineering, pages 349--353, Vol. 1(2-4), 1990

[24]
J.P.M. Hultquist, "Constructing Stream Surfaces in Steady 3D Vector Fields", Visualization `92 Proceedings, pages 171---177 IEEE, Computer Society Press 1992.

[25]
"IBM Data Explorer Programmer's Reference Manual", International Business Machines Corporation 1993.

[26]
"IBM Data Explorer User's Guide", International Business Machines Corporation 1993.

[27]
"IBM Data Explorer User's Reference Manual", International Business Machines Corporation 1993.

[28]
"IRIS Explorer User's Guide", Silicon Graphics Computer Systems.

[29]
"IRIS Explorer Module Writer's Guide", Silicon Graphics Computer Systems.

[30]
"IRIS Explorer Module Definitions", Silicon Graphics Computer Systems.

[31]
G Johnson and K Scholes, "Exploring Corporate Strategy: Text and Cases", Prentice Hall, 1989.

[32]
Khoros Manual: "Application Toolboxes", Khoral Research, Inc. 1994-95.

[33]
Khoros Manual: "Getting Started", Khoral Research, Inc. 1994-95.

[34]
Khoros Manual: "Installation Guide", Khoral Research, Inc. 1994-95.

[35]
Khoros Manual: "Programming Services Volume I - Foundation Services", Khoral Research, Inc. 1994-95.

[36]
Khoros Manual: "Programming Services Volume II - Data Services", Khoral Research, Inc. 1994-95.

[37]
Khoros Manual: "Programming Services Volume III - GUI and Visualization Services", Khoral Research, Inc. 1994-95.

[38]
Khoros Manual: "Toolbox Programming", Khoral Research, Inc. 1994-95.

[39]
Khoros Manual: "Visual Programming", Khoral Research, Inc. 1994-95.

[40]
Klemp, McIrvin and Boyd, "*Polypaint - a three-dimensional rendering package",* American Meteorological Society Proceedings, 6th International Conference on Interactive Animation and Processing Systems, 1990.

[41]
A. Koide, A. Doi, K Kajioka, "Polyhedral approximation approach to molecular orbital graphics", Journal of Molecular Graphics, Volume 4, Number 3, pages 149-155, September 1986

[42]
S. Larkin, A. J. Grant, F. Lin, N. Hill, "Advanced AVS Course Training Materials", Manchester and the North High Performance Computing Training and Education Centre, December 1994.

[43]
S. Larkin, A. J. Grant, F. Lin, N. Hill, "Introductory AVS Course Training Materials", Manchester and the North High Performance Computing Training and Education Centre, December 1994.

[44]
Lancaster, P. and Salkauskas, K. "Curve and Surface Fitting: An Introduction", Academic Press, London 1986.

[45]
Laur, D. and Hanrahan, P., SIGGRAPH `91 proceedings, in Computer Graphics, 1991, 25, page 285.

[46]
M. Levoy, "Display of Surfaces from Volume Data", IEEE Computer Graphics and Applications, pages 29--37, Volume 8, Number 3, 1988.

[47]
W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm", pages 163--169, Vol 21, Number 4, Computer Graphics 1987.

[48]
Lucas, B, "A Scientific Visualization Renderer", Proceedings of Visualization `92, pages 227-234, IEEE Computer Society Press, 1992.

[49]
Paul Ning and Jules Bloomenthal, "An Evaluation of Implicit Surface Tilers", IEEE Computer Graphics and Applications, pages 33--41, Vol 13, Number 6, 1993

[50]
F.H. Post and T. van Walsum, "Fluid Flow Visualization", Focus on Scientific Visualization, Eds. H. Hagen and H. Muller and G.M. Nielson, pages 1--40, Springer-Verlag 1993.

[51]
A. Preusser, "Algorithm 671 - FARB-E-2D: Fill Area with Bicubics on Rectangles - A Contour Plot Program", ACM Transactions on Mathematical Software, Vol 15, no. 1, 1989.

[52]
"PV-WAVE CL Applications Guide - Free Code and Sample Applications", Precision Visuals

[53]
"PV-WAVE Command Language Overview", Precision Visuals

[54]
"PV-WAVE: Getting Started with PV-WAVE Point and Click Motif Version", Precision Visuals

[55]
"PV-WAVE Command Language Guide to Advanced Rendering Library", Precision Visuals

[56]
"PV-WAVE Command Language Overview", Precision Visuals

[57]
"PV-WAVE Command Language Reference", Precision Visuals

[58]
"PV-WAVE Command Language User's Guide", Precision Visuals

[59]
"PV-WAVE Point & Click User's Guide", Precision Visuals

[60]
R.L.Renka, "Algorithm 660: QSHEP2D: Quadratic Shepard method for bivariate interpolation to scattered data", ACM Trans Math Soft, Vol 14, pages 149--150, 1988

[61]
M.A. Sabin, "A Survey of Contouring Methods", Computer Graphics Forum, Vol 5, pages 325-339, 1986.

[62]
D.C. Sutcliffe, "Contouring over rectangular and skewed rectangular grids - an introduction", Mathematical Methods in Computer Graphics and Design, Ed. K.W. Brodlie, pages 39--62, Academic Press, New York and London, 1980.

[63]
L A Treinish, "Unifying principles of data management for scientific visualization", in Animation and Scientific Visualization: Tools and Applications, R A Earnshaw and D Watson, Academic Press 1993

[64]
T. van Walsum, F. H. Post, "Selective Visualization of Vector Fields", Proceedings from Eurographics `94, Computer Graphics Forum, Volume 13, Number 3, September 1994.

[65]

L A Westover, "Footprint Evaluation for Volume Rendering", Computer Graphics Vol. 24, no.4 August 1990.