THE UNIVERSITY
of MANCHESTER

AVS/Express Visualization Edition

An Introductory Course for Medical Users

Student Workbook

F Lin, S Larkin, A Grant, R Slinger,
J Leng, I Drumm, J Perrin
Manchester and North HPC T&EC

P Lever, H Morphet
International AVS Centre (Manchester)

Edition 3.0 December 1998

Document editor

The original document was compiled and typeset by Fenqiang Lin and Steve Larkin, Computer Graphics Unit, Manchester Computing using the document preparation system LATEX.

The AVS/Express Visualization Edition, An Introductory Course was compiled using the document publishing software FrameMaker. Additional exercises and improvements were supplied by Rachel Slinger. The work was undertaken whilst working in the Computer Graphics Unit as a Summer Student Project 1996.

Additions were made to the text to make the course more suitable to the medical user by Joanna Leng and Ian Drumm in 1997. Further alteration were made by Joanna Leng and James Perrin in April 1998 and by Joanna Leng in November 1998.

The document is published and distributed by the Manchester Visualization Centre.

For further information on the Manchester Visualization Centre see the following World Wide Web page:

    http://www.man.ac.uk/MVC/

or send email to:

    mvc-info@mcc.ac.uk

or by mail to:

```
Manchester Visualization Centre
Manchester Computing
University of Manchester
Manchester
M13 9PL
UK
```

Table of Contents

1 Introduction

1.1 The AVS/Express Visualization Edition course

This edition of the AVS/Express Visualization Edition, An Introductory Course for Medical Users has been developed as a part of the work of the VISUAL project and is aimed at medical users so mainly concentrates on uniform fields with one data element. It is an adaptation of the course AVS/Express Visualization Edition, An Introductory Course that was developed by the International AVS Centre (Manchester) and is based on the *Introduction to AVS* course produced previously by the Manchester and North Training and Education Centre (MAN T&EC) based in the Computer Graphics Unit at The University of Manchester.

1.1.1 What is AVS/Express?

AVS/Express is an object-oriented, visual development tool that enables you to build reusable application components and sophisticated applications. The AVS/Express Visualization Edition provides hundreds of application components for visualizing, analysing, manipulating and displaying data.

1.1.2 Who is this workbook aimed at?

This workbook is aimed at those who have a need to visualize both 2D and 3D data. Techniques covered include: reading data, creating 2D & 3D images, isosurface plots, arrow plots etc.

1.1.3 Are there any prerequisites?

The course is designed for the user with no prior experience of AVS/Express Visualization Edition. The course also assumes no prior experience of AVS5. No programming skills are required but basic experience of working in a workstation environment, i.e. UNIX and X windows, is needed.

1.2 About the workbook

1.2.1 The visualization process

The data visualization process is carried out using a **network**.

A network consists of **modules** and **connections**. Modules carry out certain tasks such as: reading fields, mapping, filtering and data viewing.

The networks are built in the **Network Editor**.

Modules are chosen from the module library and connected in order to carry out a specific application.

Before data can be read into a network a **field file** must be created in order to describe the data to the network. The data is read into the network via the field file, the network carries out its specified function and presents the data visually.

Interaction with the visual data is carried out until the desired analysis has been achieved.

Networks can be saved as **Applications** and reused to analyse similar data sets.

1.2.2 Structure of the exercises

Each chapter of this book works through a series of structured exercises in order to familiarise you with the AVS/Express environment.

- Chapter 2 deals with using the Network Editor: building network, making connections, reading data, saving, loading and deleting applications.
- Chapter 3 and Chapter 4 deal with creating field files to describe 2D & 3D data respectively. Also dealt with are visualization techniques for 2D & 3D data.
- Chapter 5shows the user how to use the Examples library.
- Chapter 7explains how to get output from the DataViewer in hardcopy and vrml format.
- The appendices give further information. The first outlines the difference between application, journal and project and introduces the idea of v code, the second explains how to customise AVS/Express from the command line and the last gives the answers to the exercises.

1.2.3 Datafiles and solutions

Each chapter of exercises uses simple examples of data in order to demonstrate a particular function of AVS/Express. Some data files for these exercises are not part of the standard AVS/Express release. This course assumes they have been made available under the directories:

```
      /usr/express/data/field/med
or
      /usr/express/data/field
or
      /usr/express/auxdata
```

`Directory paths are prone to change and the directory holding the data will be given to you on the day.`

Copies of these data files can be obtained via anonymous FTP from The University of Manchester at

```
ftp.iavsc.org (130.88.100.80)
```

in file

```
pub/course_data/express/intro.tar.Z
```

The answers to the exercises can be found in Appendix III of this workbook.

2 Using the network editor

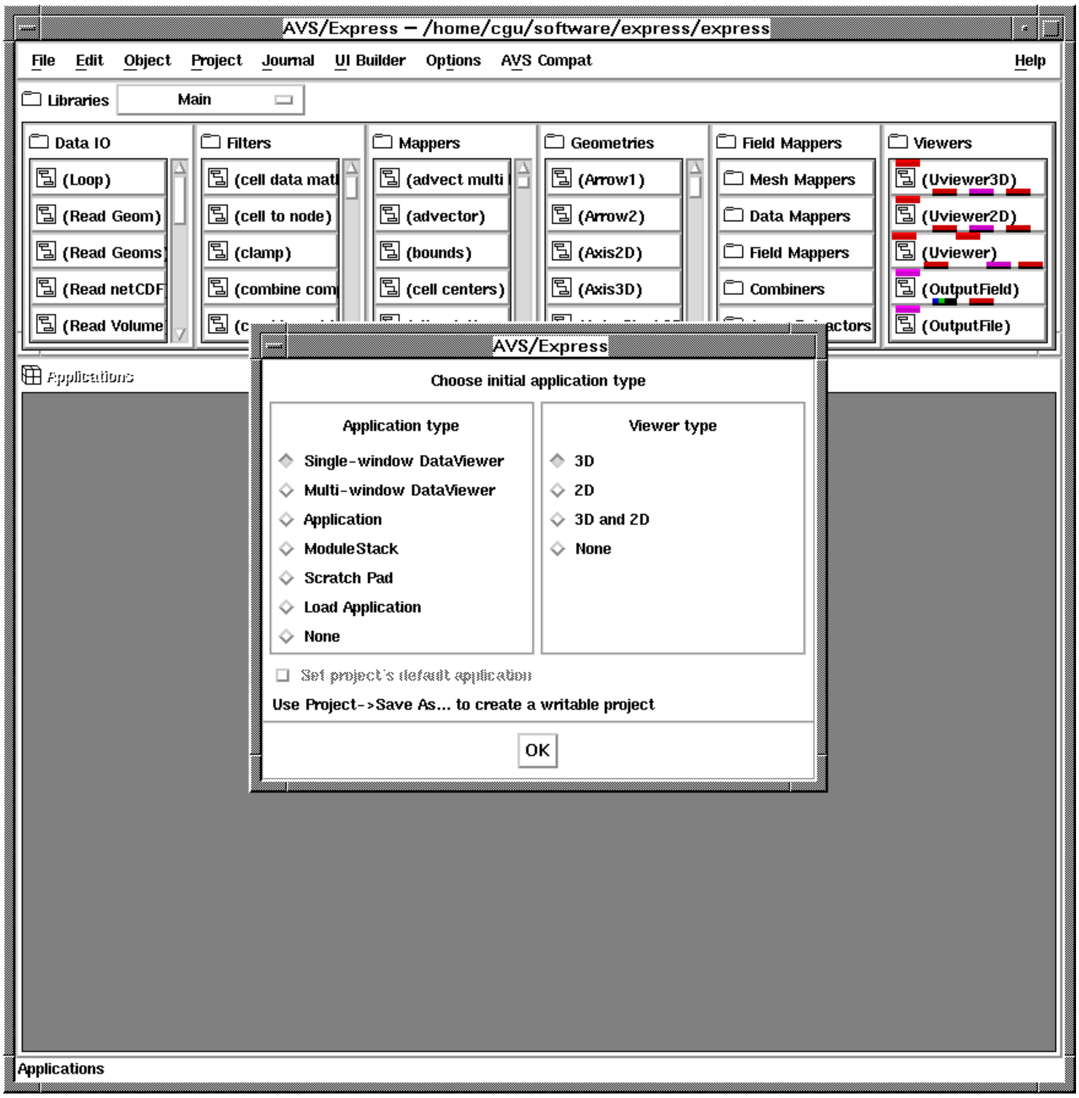
2.1 Aims of the chapter

The exercises in this chapter will illustrate the general features of the AVS/Express network editor. Exercise 1 is a simple example designed to take you through the data visualization process: starting AVS/Express, constructing networks, reading data, saving, loading and deleting applications and exiting from AVS/Express. Exercise 2 is a similar example for you to attempt on your own. In both cases, it is not necessary to create a field file in order to read the data. The creation of field files will be covered in subsequent chapters.

2.2 Exercise 1

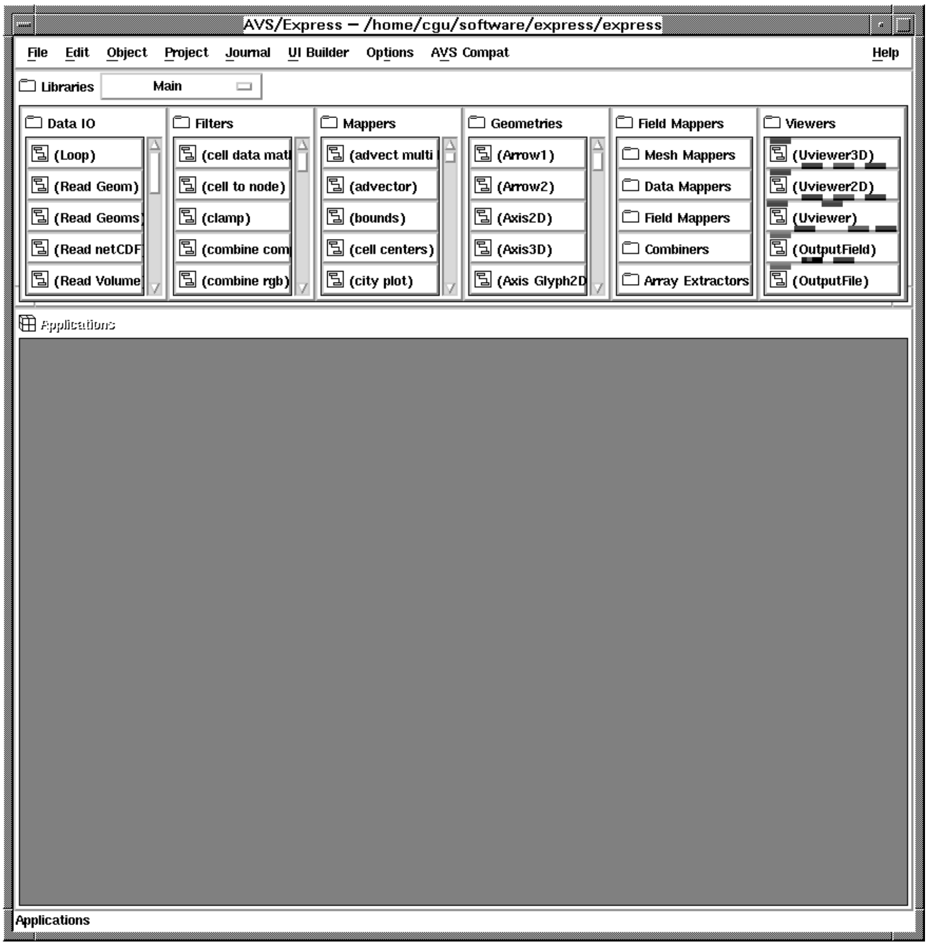
In this example we wish to visualize a MR scan of the head. In particular we want to take a slice through the head and observe the internal structures.

2.2.1 Starting the AVS/Express Visualization Edition

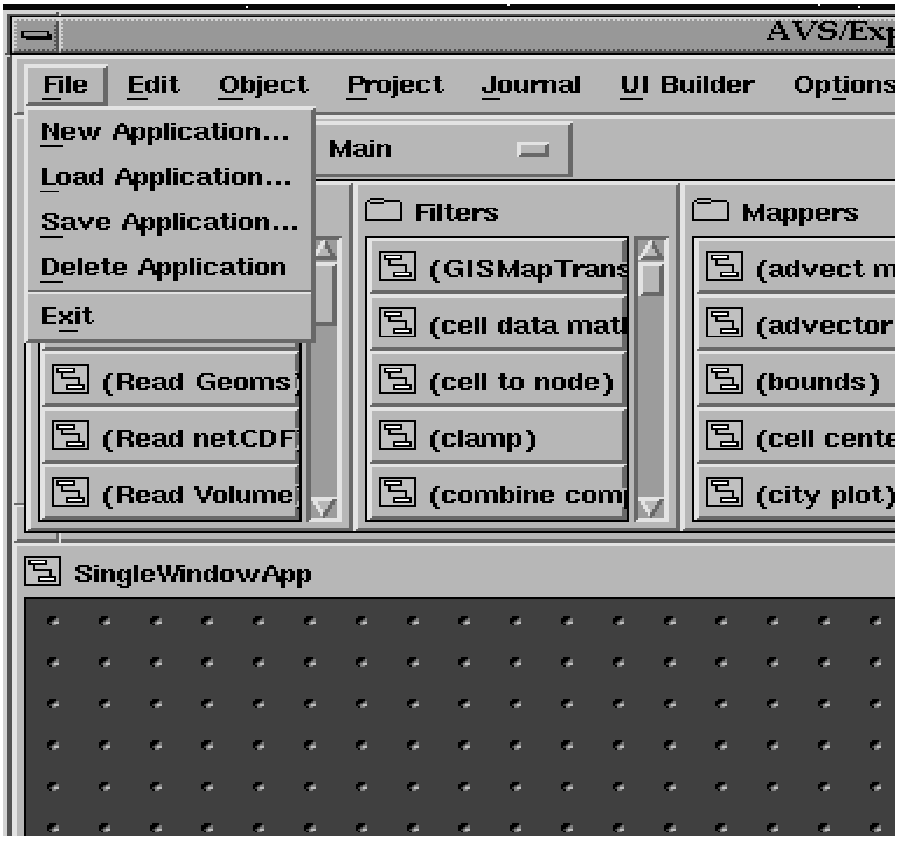


- To start AVS/Express Visualization Edition, simply type the following command:
    - `express -viz`
    - (some machines may have the command "`vxp`")
- AVS/Express displays the network editor and a dialogue box in which you choose the type of application and viewer you want to use for your entire AVS/Express session.
- Choose **None** for **Application Type** (it is often best to select the application type for each application not for the AVS/Express session).
- Click on **OK** to proceed.

2.2.2 Structure Of The Network Editor

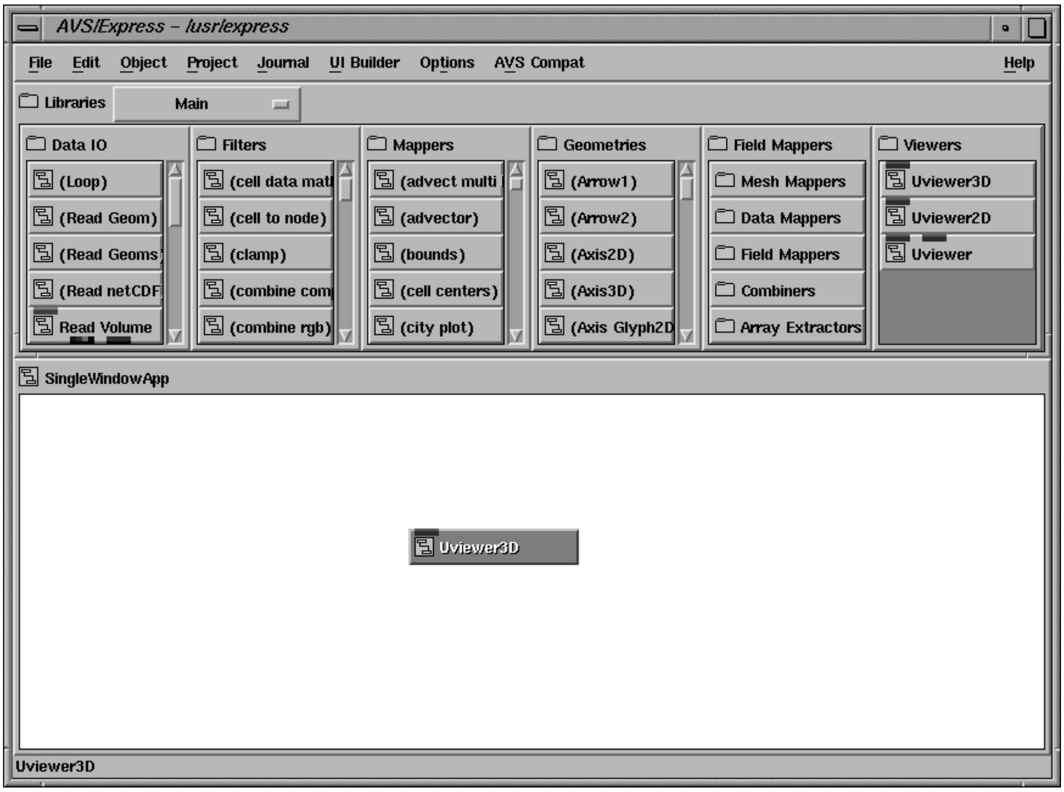


- The screen should now have the above layout, which shows the major components of the network editor.
- Along the top of the window are a number of pull down menus:
  - **File**
  - **Edit**
  - **Object**
  - **Project**
  - **Journal**
  - **UI Builder**
  - **Options**
  - **AVS Compat**
- You will use most of these menus during this course but for now you need to use the **File** menu to start an application.



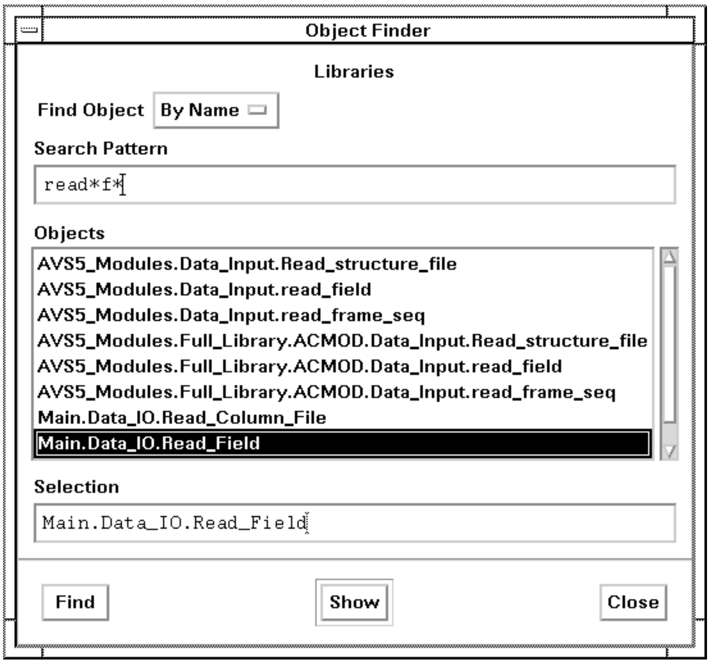
- From the **File** menu select **New Application**.
- The same dialogue box appears as before, this time you will select the application type just for this application.
- Choose the `Single-window DataViewer` with a `3D` viewer type (usually a good choice for a simple application).
- Click on **OK** to proceed.

2.2.3 Building An Application



- The screen should now show the above layout, this time though there is a module in the **Network Editor** which is titled **Uviewer3D**. This module will produce a 3D image in the **Single Window Application**.
- The library of modules on view, above the application workspace, is called **main**. This is the library which contains the major modules for network building. (Look at the popup menu to observe the other available libraries.) Within the main library the modules are divided into sub-libraries:
    - **Data IO**
    - **Filters**
    - **Mappers**
    - **Geometries**
    - **Field mappers**
    - **Viewers**
- **Data IO** contains the various modules for reading data and **Viewers** contains the various modules for viewing data. The other sections are used to build up the intermediate parts of a network in order to manipulate the data for a specific application.

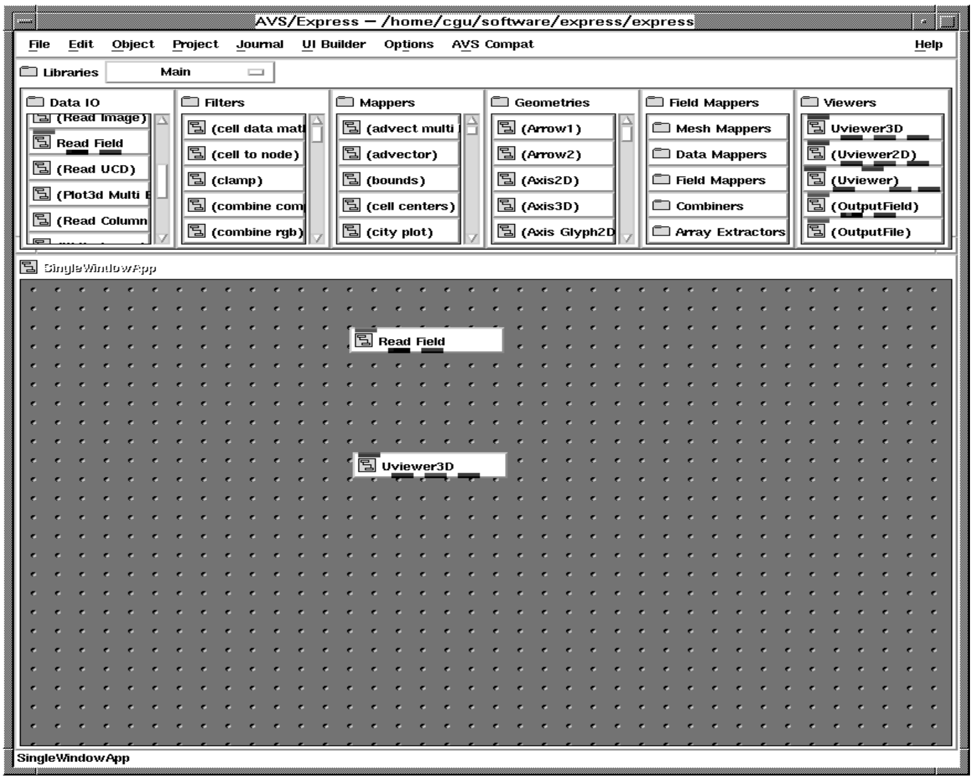
2.2.4 Using the object finder



When selecting modules for a network you can either scroll through the sub-libraries until you locate the correct one or alternatively you can use the **Object Finder** to search for a specific module. You can either search for a module in all libraries or in a specific library.

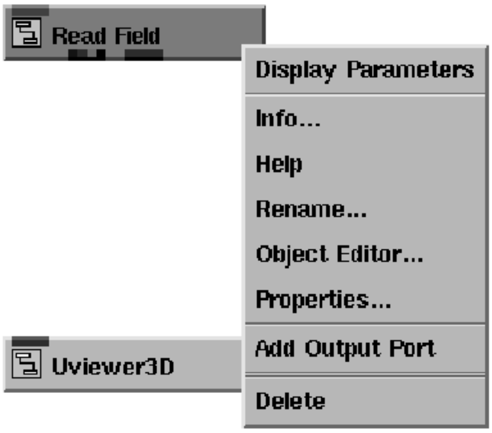
- *To search through all the AVS/Express libraries*: choose, **Find in All Libraries** from the **Object** menu, (located in the menu bar along the top of the network editor), and type in the name of the module in the **Search Pattern** box. Click **Find** and the location of the module will be given in the **Object box**. If you wish the location of the module to be highlighted in the network editor library click **show**.
- *To limit your search*: select the library or library page you want to search by clicking on it using the **left** mouse button. Now hold down the right mouse button to display the popup menu, and choose **Find** popup command. Alternatively, you can select the library page, and then choose **Find in Selected Library** from the **Object** menu and follow the same procedure as before.
- You can use the wildcard ***** to increase search flexibility. The * stands for one or any number of unknown characters in a name. For example the above diagram shows a search for any modules which read field data - type "`read*f*`" in the **Search Pattern** field**.**
- It is best to avoid using old AVS5 modules, they are there to help convert AVS5 networks to AVS/Express. The AVS/Express modules use the full functionality of the new AVS/Express structure and will continue to be supported by AVS in future versions.

2.2.5 Selecting a module



- Find the `Read_Field` module which is located in the **Data IO** library.
- Point to the module using the mouse and press the **left** mouse button.
- While keeping the mouse button pressed drag the module onto the application workspace. When you are happy with its location release the button.
- The module will be instanced to the workspace.

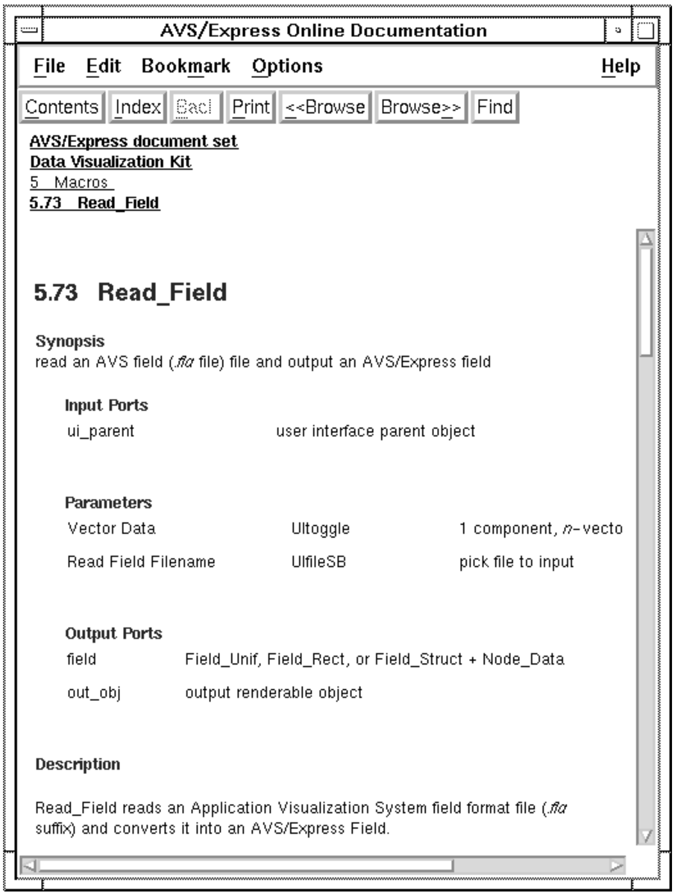
2.2.6 Deleting a module



To delete a module:

- Select it
- Hold down the **right** mouse button to display the popup menu
- Choose the **Delete** popup command
- When you release the mouse button the module will be deleted.

2.2.7 Accessing the help pages



To access the help pages for a particular module:

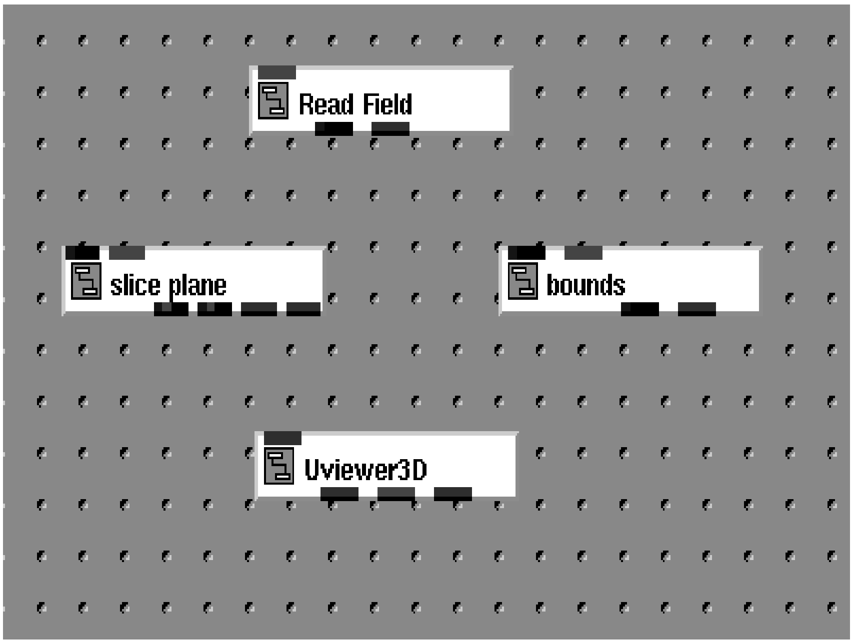
- Select the module and hold down the **right** mouse button to display the popup menu
- Choose the **Help** command.

Alternatively:

- Select the module
- From the **Help** menu on the menu bar at the top right of the network editor choose **On Selected Object**.

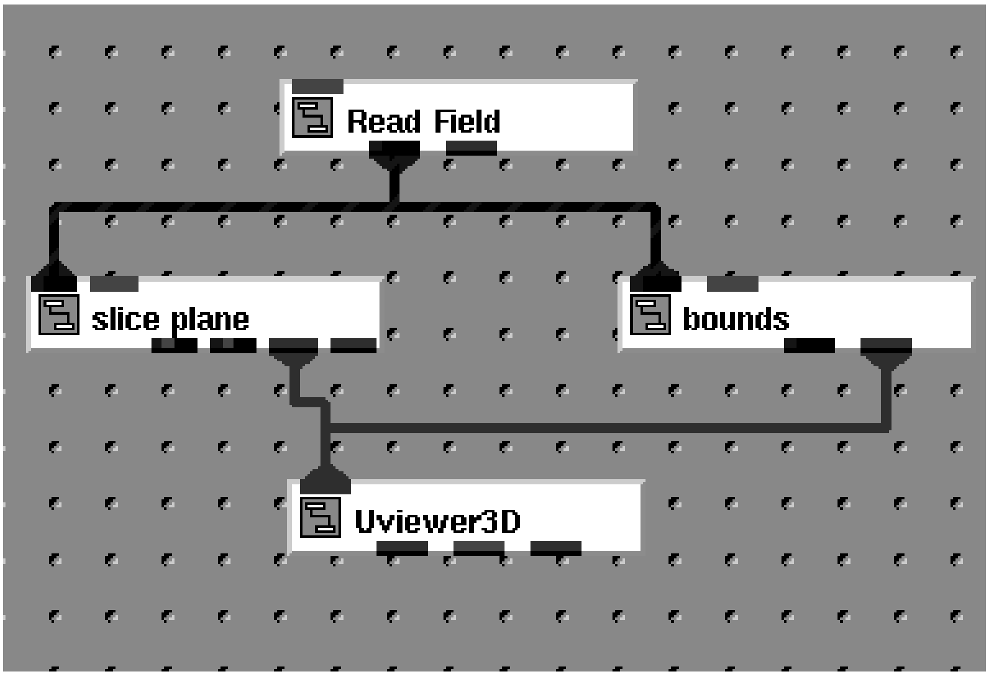
Both these routes will invoke the appropriate help page to appear.

2.2.8 Selecting the other modules



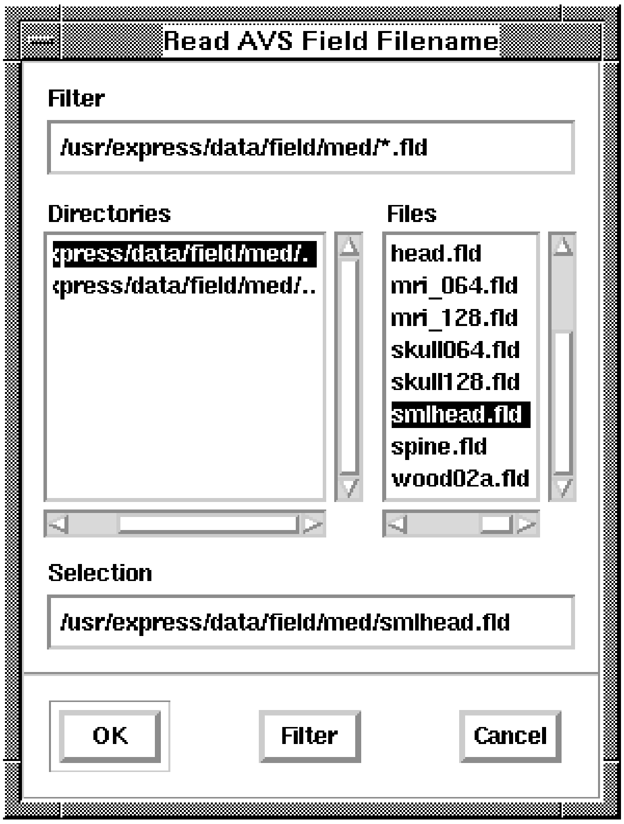
- Select the modules in the list below from the given sub-libraries and arrange them in the workspace as shown above.
  - `Read_Field` (Data IO): reads a AVS field format file (*.fld* suffix) and converts it to a AVS/Express field.
  - `bounds` (Mappers): generates a bounding box of a 3D field.
  - `slice_plane` (Mappers): extracts a 2D slice from a 3D field with an arbitrarily positioned slice plane.
  - `Uviewer3D`: (Viewers): creates a 3D image in the DataViewer.
- Try to locate at least one of the modules using the **Object Finder.**

2.2.9 Connecting the ports



- Connect the modules together to form the above network. It is important to make sure that the correct input/output ports are connected or the network will not perform the correct function. (Often AVS/Express will not let you make an incorrect connection or will produce an error message if one is made.)
- **Connecting modules**:
    - To connect module ports together move onto one of the ports you wish to connect with the mouse.
    - Now press the left button - all the possible connections are shown.
    - While keeping the button pressed move the mouse toward the other module port you wish to make the connection to. The connection should now become highlighted.
    - Release the mouse button and the connection is made.
- **Disconnecting modules:**
    - To disconnect a module connection move onto the module which is connected via its output port and “remake” the connection i.e. press the **left** button while positioned on this port, move toward the other port and when the connection between the two ports becomes highlighted, release the button.
    - The connection will be removed.

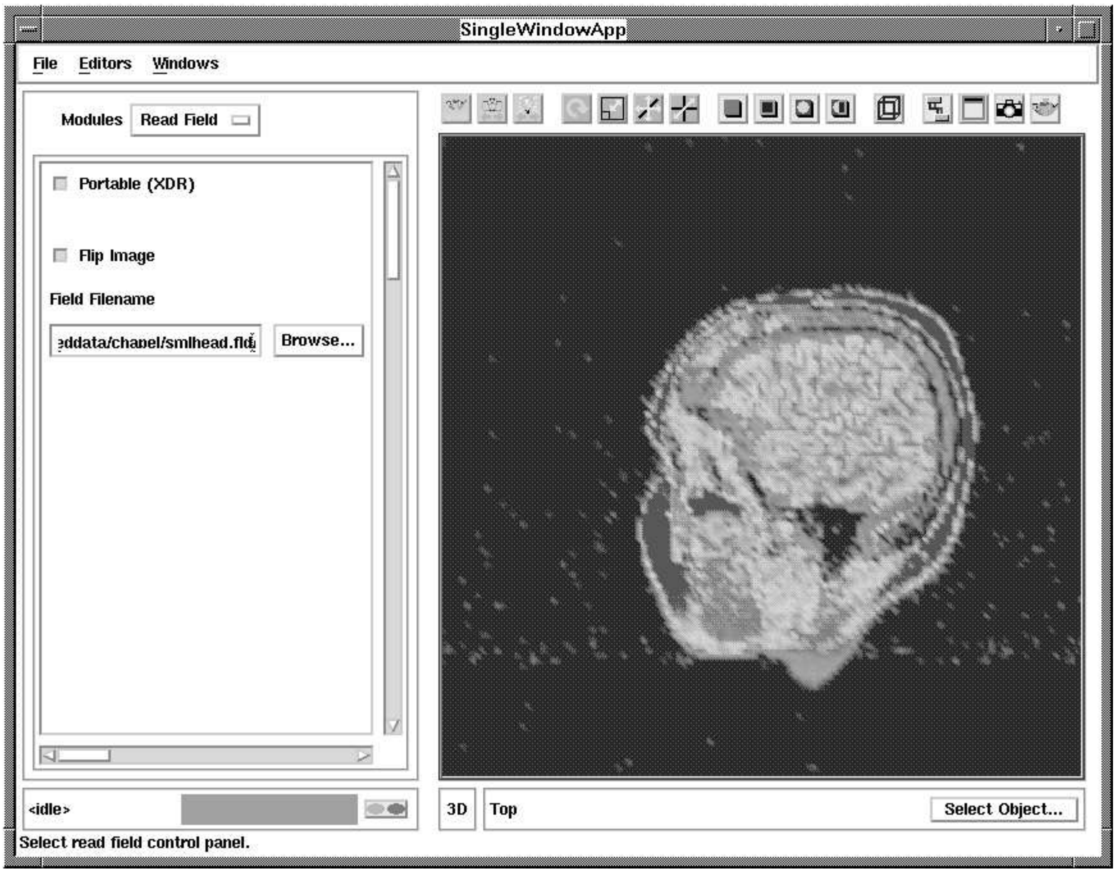
2.2.10 Reading a data file (1)



When a module is instanced to the workspace you can access controls for the module in the editor panel of the **DataViewer Pad**.

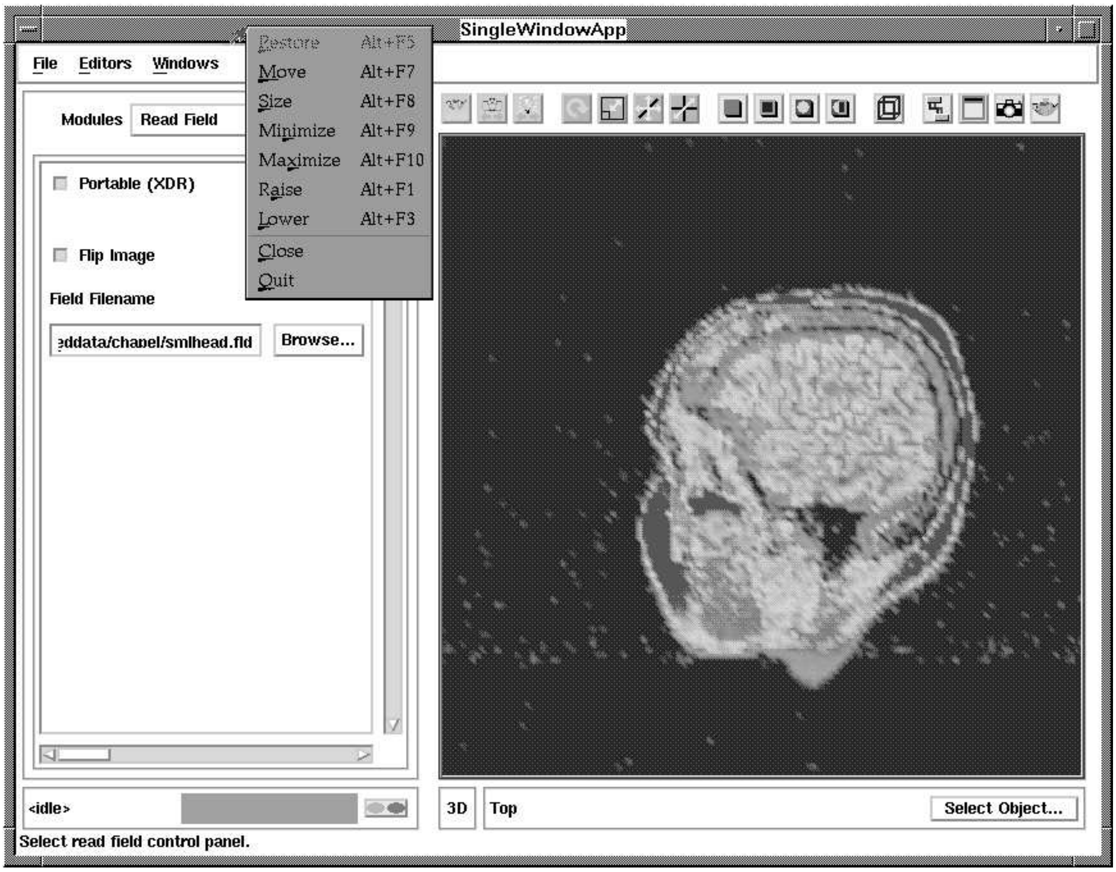
- Select the **Modules** command from the **Editors** pull-down menu. A menu of all modules instanced to the workspace will appear.
- Choose the `Read_Field` module and press the browse button. The file browser should now appear.
- Use the file browser to change to the directory with the data in it (e.g. `/usr/express/data/field/med)` and click on the file called `smlhead.fld.` The directory holding the data will be given to you on the day.
- The network will execute producing a picture in the viewer window located in the **DataViewer Pad.**

2.2.11 Reading a data file (2)



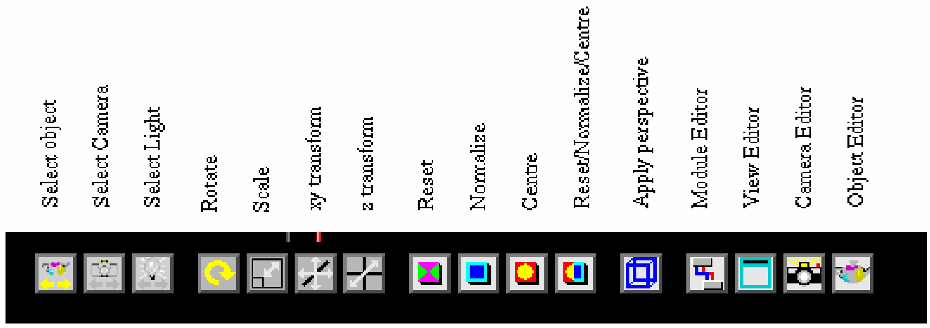
- The **DataViewer** window should display the above picture.
- Now that an image has been obtained, we want to manipulate the picture to the desired format.

2.2.12 Resizing The DataViewer



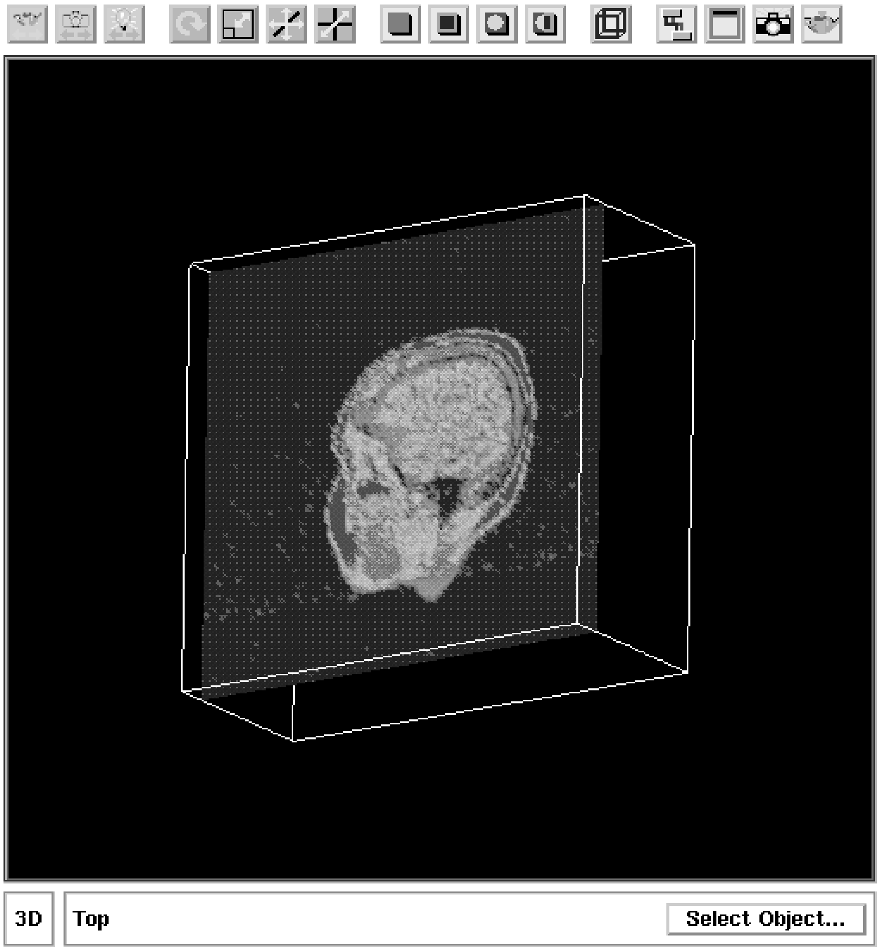
- The window around the DataViewer is a motif window which means that the size of the Viewer can be increased in a standard well known manner. AVS/Express will alter the image size to correspond to the window size by using interpolation so an accurate smooth image is always displayed.
- The size of the Viewer is increased by clicking on the top bar of the Viewer with the right mouse button and selecting Maximise from the pull down menu. The Viewer now covers the whole screen.
- Alternatively the size of the window can be adjusted by clicking on any of the corners of the window with the left mouse button and dragging it to the desired location.
- The window can be minimised to an icon by clicking on the square button on the top bar in the left hand corner with the small square on it. The iconised window can be increased to full size by double clicking on it with the left hand mouse button.

2.2.13 Interacting with the objects (1)



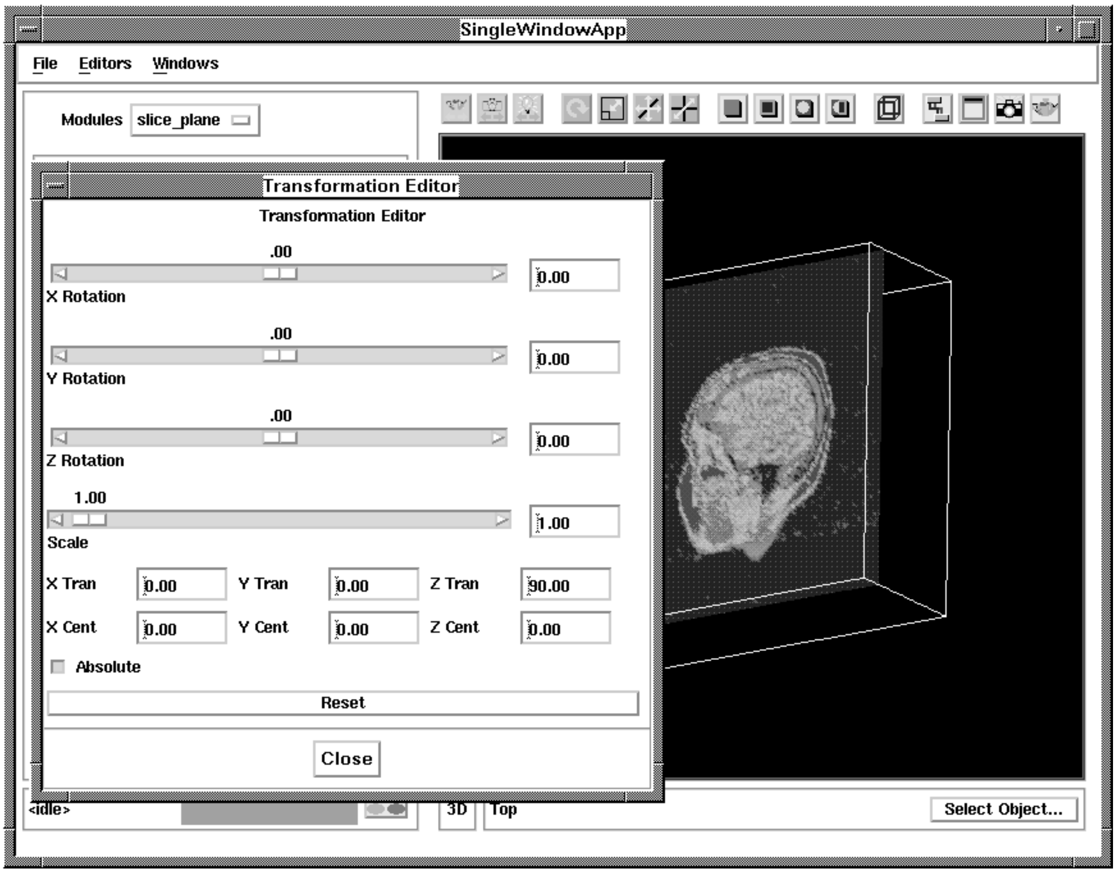
- The Tool bar, located in the **DataViewer** interface, provides a quick way to open **DataViewer** components and to manipulate images in the **DataViewer** window.
- Move the mouse over these icons to reveal a small label indicating the action the icon provides a short-cut to, e.g.
  - Transform mode selection, rotation, translate xy, translate z, and scale
  - Reset, normalize, and centre objects

2.2.14 Interacting with the objects (2)



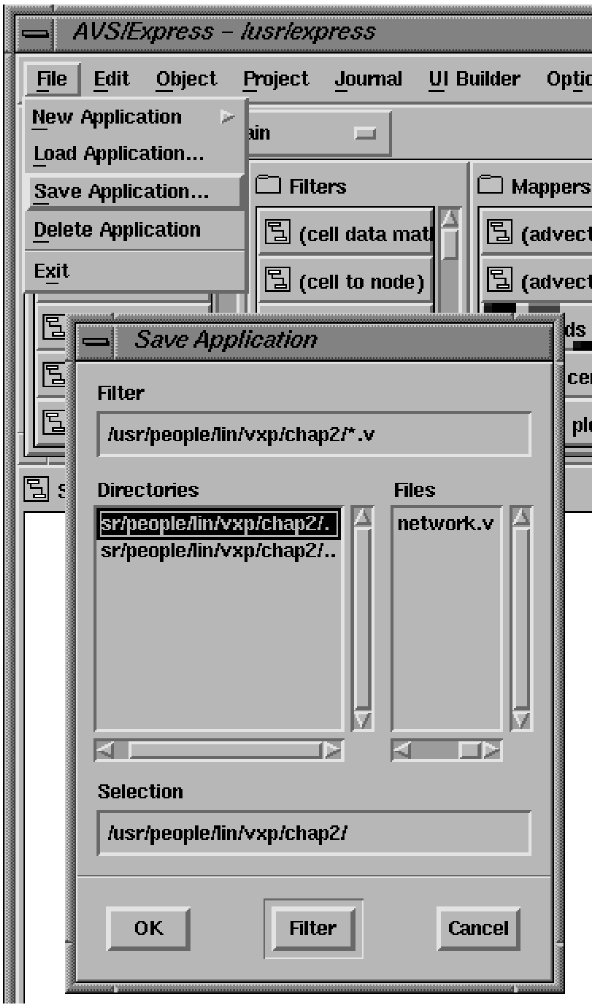
- Click on the **Reset/Normalize/Center** icon, and the full picture should appear in the viewer window.
- Now you can perform a few simple transformations on the image in the **DataViewer** window:
  - **scaling**: click on the `Scale` icon in the Toolbar, place the cursor in the window and, while holding down the **left** mouse button, move the cursor up and down to enlarge and shrink the object.
  - **rotating**: click on the `Rotate` icon in the Toolbar, place the cursor in the window and, while holding down the **left** mouse button, rotate the object.
  - **translating**: click on the `Translate` icon in the Toolbar, place the cursor in the window and, while holding down the **left** mouse button move the mouse to translate the object in the window.
- Manipulate the object until the scene appears similar to the one shown in the diagram above.

2.2.15 Changing the `slice_plane` parameters



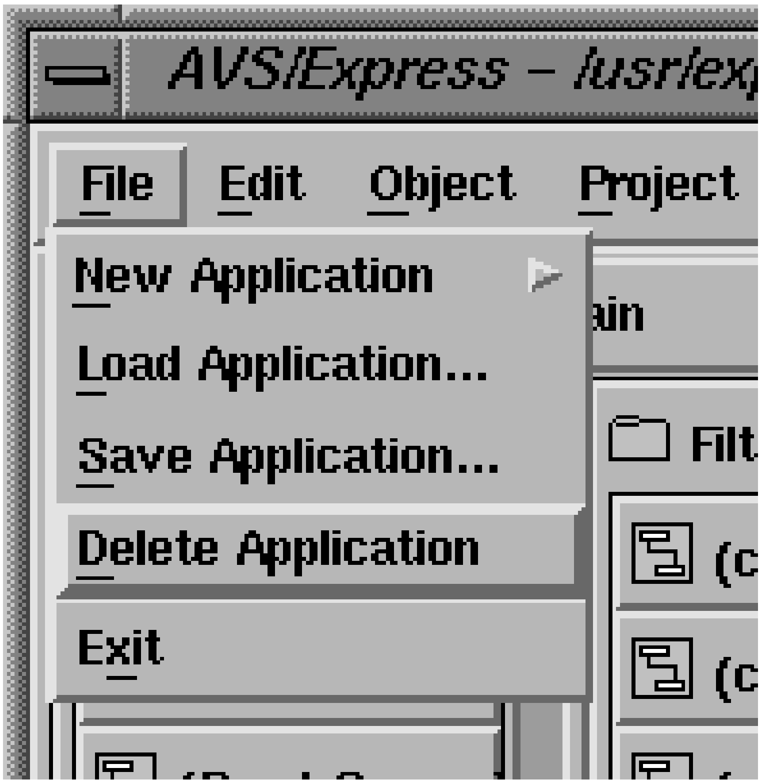
- Select the **Modules** command from the **Editors** pull-down menu, (or the Toolbar), and choose the `slice_plane` module. You can now try altering some of the parameters of the slice through the head.
- You can alter the position of the slice through the head by changing the value on the `plane distance` slider.
- Turn on the **Plane Transform Editor** - a series of sliders should appear. These sliders can be used to rotate and transform the slice in the x, y, & z directions and to scale it.
- Try altering some of the other slider parameters to alter the position of the slice.

2.2.16 Saving applications



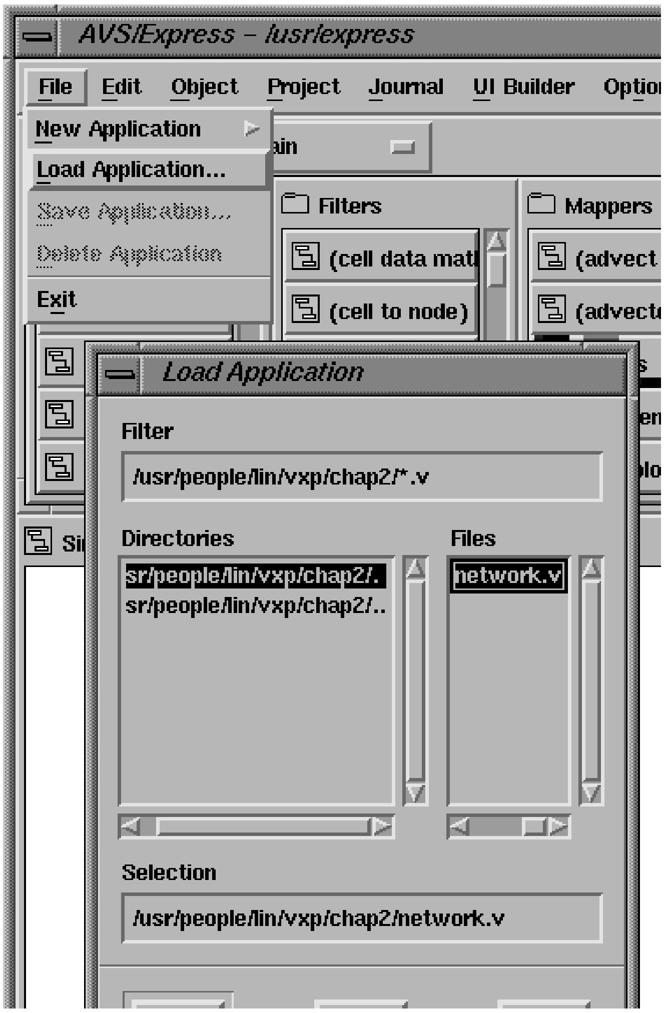
- The network can be saved as an application by selecting the **Save Application...** command from the **File** pull-down menu.
- A file browser will appear showing the files in your current working directory. Use the file browser to change the current directory to the one you want to save the file in. (It is best to save networks in your home directory or try `/tmp`.)
- A good convention is to always save the application network by typing in the file name followed by the file extension "`.v`", e.g. "`fred.v`".

2.2.17 Deleting an application



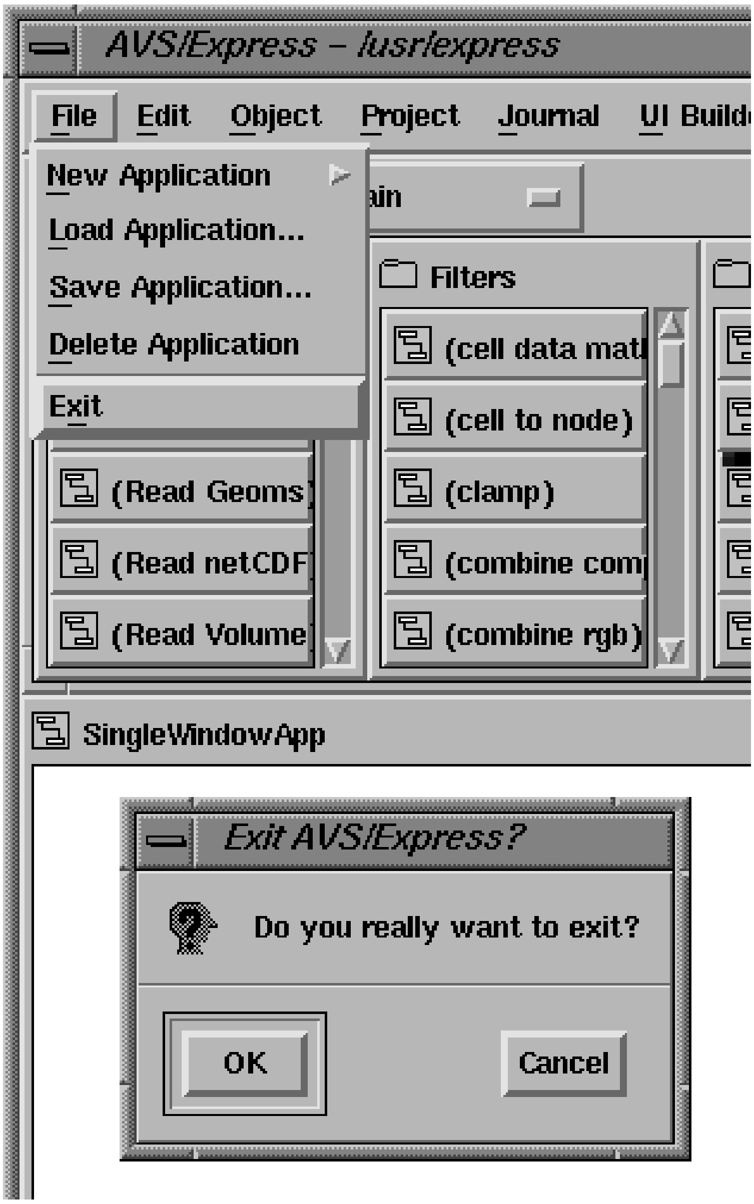
- To clear an existing application network from the workspace select the **Delete Application** command from the **File** pull-down menu.

2.2.18 Loading an application



- To load an application network select the **Load Application...** command from the pull-down **File** menu. A file browser will appear showing the files in your current working directory.
- Use the file browser to change the current directory to the one containing the saved application network.
- Click on the file name in the file browser window to load the application network.

2.2.19 Exiting from AVS/Express



- To exit from AVS/Express select the **Exit** command from the pull-down **File** menu.
- You will then be asked "Do you really want to exit?" in a popup window. Click the **OK** button if you still wish to exit.

2.3 Exercise 2

In this exercise we want to visualize the variation in bone-density through the body of a lobster.

- Restart AVS/Express
- Use the same network as the one constructed in Exercise 1, reload it into AVS/Express.
- When using the Read Field browser select the file `lobster.fld` or `hydrogen.fld` in the directory the browser goes to by default and click **OK**.
- Using similar manipulation techniques as those described in Exercise 1, investigate the internal structure of your data set.
- Select the `bounds` module from the **Modules** menu and try altering some of the parameters.

3 Visualizing 2D arrays of data

3.1 Aims of the chapter

There are two aims to this chapter. Firstly, for you to learn how to create field files and to understand their structure. Further understanding will be gained by attempting the field file examples, the answers to which are given in Appendix III .

Secondly, for you to work through a structured exercise which covers visualization techniques for 2D data: creating field files, constructing a networks and manipulating images.

3.2 Creating field files

Field files are used to represent arrays of data in AVS/Express. The data is described by the field file which in turn reads the data into the network via the `Read_Field` module.

3.2.1 Field file format

The basic field file has the following format, although a particular field file need not have all the components shown:

```
# AVS field file
#
ndim = ?
dim1 = ???
dim2 = ???
nspace = ???
veclen = ???
data = ???
field = ???
label = ???
unit = ???
variable n file = ??? filetype = ??? skip = ??? offset = ??? stride = ???
coord n file = ??? filetype = ??? skip = ??? offset = ??? stride = ???
```

- The file should have the file extension ".fld" (you may want to create a sub-directory to keep all your field files in).
- The first line should read "`# AVS field file`".
- The next line, also beginning "#" could be a line describing the data. This way you know which application this field file refers to.

3.2.2 Defining the data structure

- The dimensions of the data must be specified, both in computational and, as it is passed to AVS/Express, in coordinate space. `ndim` and `nspace` give the number of dimensions in computational and coordinate space, respectively. `dim1`, `dim2` and `dim3` specify the size of the x, y and z dimensions.
- The number of data values per element is specified by `veclen.` For example, if there is a value of temperature and pressure for each element in the data set then `veclen=2.`
- The type of the data components is given by `data.` The data components can be `integer`, `float`, `double` or `byte`.
- The mapping type of the field is given by `field.` The mapping type can be one of 3 types:
  - **uniform**: data is not transformed as it is imported. Used to import regular arrays of data. Number of dimensions in computational and coordinate space must be the same.
  - **rectilinear**: each dimension in the data has explicit coordinate mapping, but the spacing along coordinate axes need not be the same. Number of dimensions in computational and coordinate space must be the same.
  - **irregular**: each element is mapped to a point in coordinate space, e.g. 1D scatter to 3D coordinates. Number of dimensions of computational and coordinate space may be different.
- The data components in the dataset can be labelled and given units by `label` and `unit`, e.g. for 2 components, temperature and pressure,
  - `label=temperature pressure`
  - `unit=C Pa`

3.2.3 Specifying the format of the data

Once the data structure has been defined, it is necessary to specify the location and format of the datafiles to be read.

- Each data component needs a line,
  `variable n <number of format keywords>`
- Each item of coordinate information needs a line,
  `coord n <number of format keywords>`
- Although each new variable and each new coordinate begin on new lines, all format keywords for a particular item must be on the same line.
- Format keywords:
  - **file**: filename of the datafile to be read, (full pathname required)
  - **filetype**: format of the data, `binary` or `ASCII`
  - **skip**: number of lines or bytes in the datafile before the data is actually found
  - **offset**: number of columns to jump before the data is read (ASCII only)
  - **stride**: how many steps to jump before the next item is reached

3.2.4 Summary

Field file format keywords:

- `ndim`: number of dimensions in computational space
- `dim1`: size of dimension 1
- `dim2`: size of dimension 2
- `nspace`: number of dimensions in coordinate space
- `veclen`: number of data components per element
- `data`: type of the data components
- `field`: mapping method
- `label`: label for each data component
- `unit`: units for each data component
- `variable n file = ??? filetype = ??? skip = ??? offset = ??? stride = ???`
- `coord n file = ??? filetype = ??? skip = ??? offset = ??? stride = ???`

where:

- `file`: location of data file
- `filetype`: format of datafile
- `skip`: number of lines (ASCII) or bytes (binary) before the data is found
- `offset:` number of columns to jump before data is read (ASCII only)
- `stride`: number of steps forward to reach next item

3.3 Exercises

Write field files to describe the datafiles[1] described in Exercises 1, 2, 3 & 4. The solutions are given in Appendix III .

3.3.1 Exercise 1

The dataset called `image.dat` is a 2D image (512x256) consisting of greyscale values in the range 0 - 255,

```
0 10 0 15 200 255 ...
...
...
```

3.3.2 Exercise 2

The datafile called `volume.dat` consists of a 3D array of density components in a regular volume, 128x128x128. The file has one line of header information.

```
Patient: Steve 10/11/93
0 255 2 1 7 ...
5 6 ... ... ... ...
...
```

3.3.3 Exercise 3

The data is a 2D array (5x10) with floating point values of temperature and pressure at each point in space. The field is irregular. The temperature and pressure data is contained in file `flow.dat` and the coordinates are contained in file `coord.dat`. Neither file contains any header information.

```
flow.dat
      t1 p1 t2 p2 t3 p3 ... ... ... ... ...
coord.dat
      x1 y1 x2 y2 x3 y3 ... ... ... ... ...
```

1. These files are examples and are not supplied as part of the data files supporting this workbook.

3.4 Visualization of 2D data

Now that you can create field files to read data you are ready to move onto creating applications which will process the data described by your field files. The exercises given in the rest of this chapter will take you through several simple examples of the visualization of 2D data.

3.5 Moving to your home directory

In this chapter you will want to read field files which you will create in sub-directories within your home directory. By default when you first instance a version of the `Read_Field` module, or any other module which reads data, it will point to the directory `/usr/express/data/field`. In order to move rapidly to your home directory:

- Delete the contents of the `filter` box.
- Type `"~"` and press `return`.
- You will now be placed in your home directory and can easily move to the sub-directory which contain your field files.

3.6 Exercise 1

The aim of this exercise is to visualize the data obtained from the MRI scan of a brain.

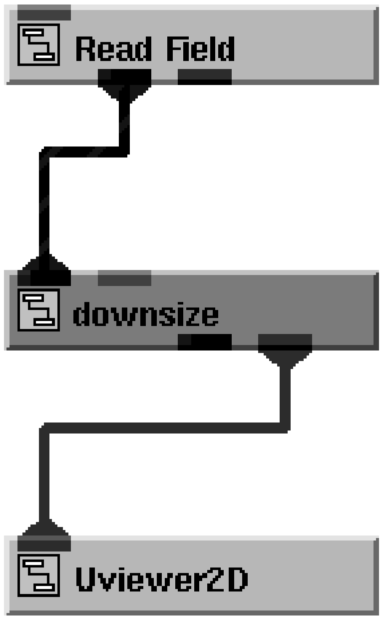
3.6.1 Importing the `mri.dat` data (1)

Below is a description of the data contained in the datafile `mri.dat`,

```
The dataset is 2D and contains 512x512 single scalar bytes for each data value.
The data file is binary and the data is arranged contiguously with no header
information.
```

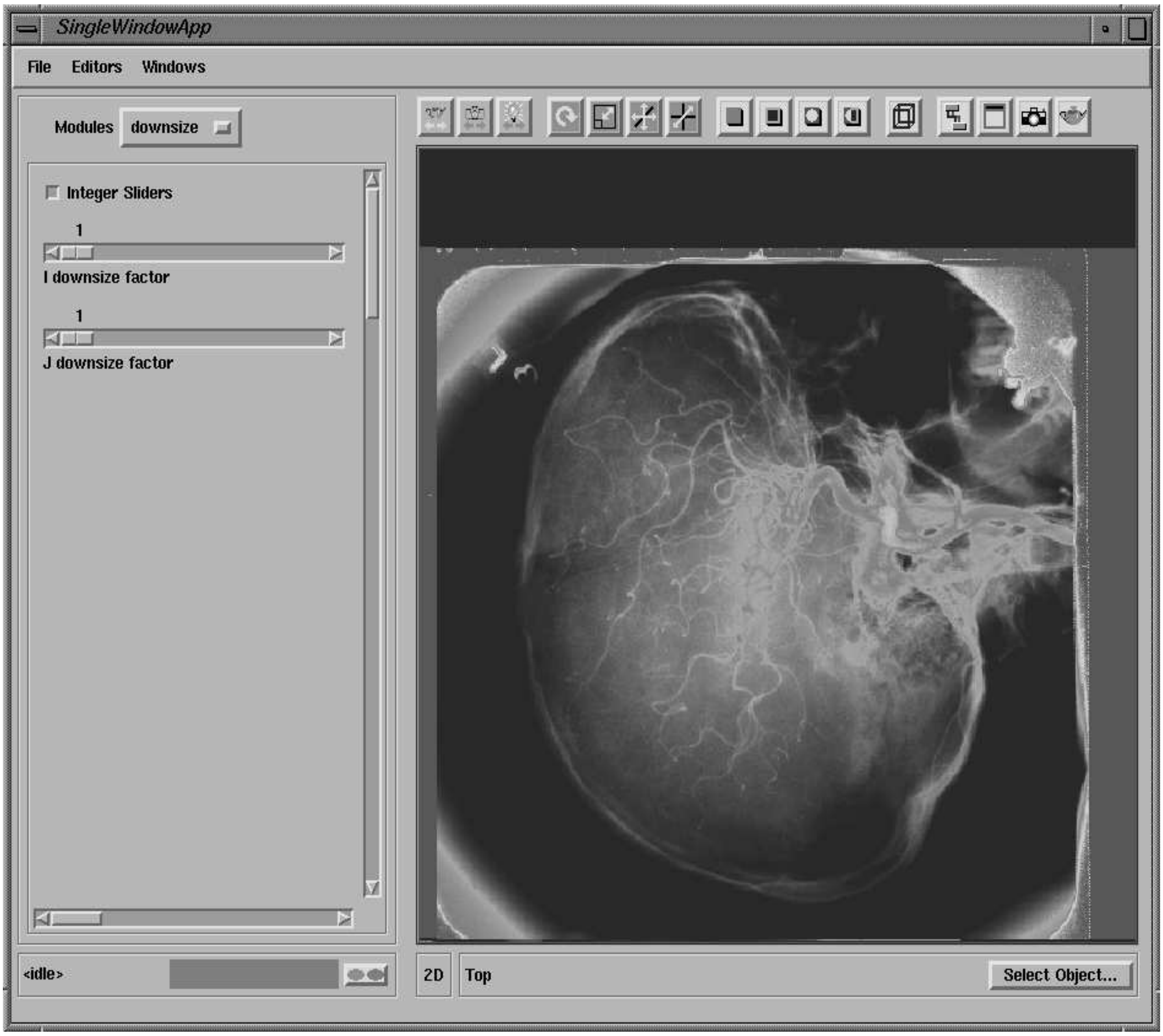
- Create a file called `mri.fld` to describe the data contained in `mri.dat`.
- Use the template field file given on page 28 and the information given above to complete the specification.

3.6.2 Displaying the MRI data



- Using the network editor, instance the modules in the list below into the workspace and construct the above network (be careful to connect the correct ports).
    - `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
    - `downsize` (Filters): resamples a field using a scaling factor. Usually the factor is greater than 1 and the size of the field is reduced to save processing time and memory.
    - `Uviewer2D` (Viewers): creates a 2D image in the DataViewer.

3.6.3 Importing the `mri.dat` data (2)



- In the DataViewer window select **Modules** from the **Editors** pull-down menu and select the `Read_Field` module.
- To import the data into the network use the file browser to move to the correct directory and select the file `mri.fld` you have just created.
- You should obtain a picture similar to the one shown above.

3.6.4 Changing the downsizing

The image displayed has been produced by a subset of the data due to the `downsize` module function. This image is likely to be poor.

- Select the `downsize` module from the **Module** editor panel.
- Try altering the values of the I & J downsize factors. (You will find that the best resolution of the image is obtained if both values are set to 1.)

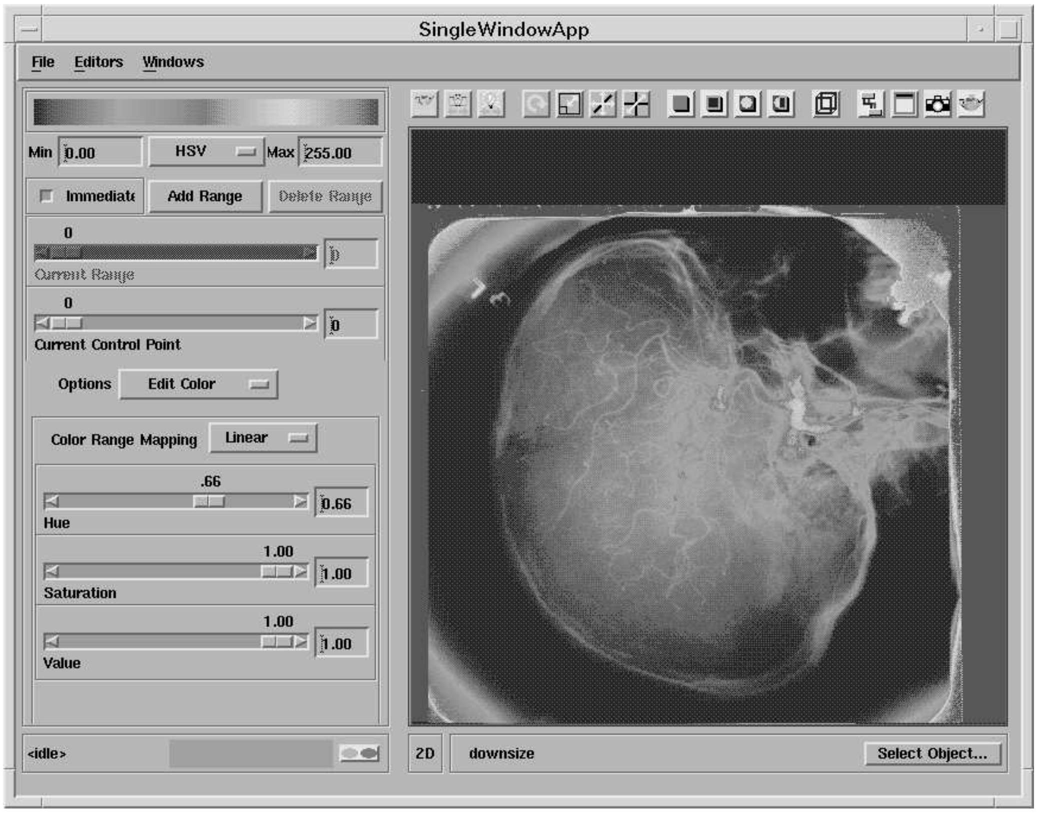
3.6.5 Changing the colour map (1)

The image is coloured using the **Default AVS/Express datamap**. It is possible to alter the colour map of the image to a more suitable format.

In AVS/Express, each object in the image hierarchy can be selected and manipulated separately. *In order to change the datamap, the image itself should be selected as the current object.*

- Click the **Select Object...** button, which is found in the bottom right corner of the **DataViewer**.
- Select the object i.e. the **downsize** object in the **object selector dialog** displayed and click **apply** or **OK.**

3.6.6 Changing the colour map (2)



- Bring up the **Editors** pull down menu and select **datamap.** The datamap menu is shown in the above diagram.

Below are listed some of the main properties of the datamap:

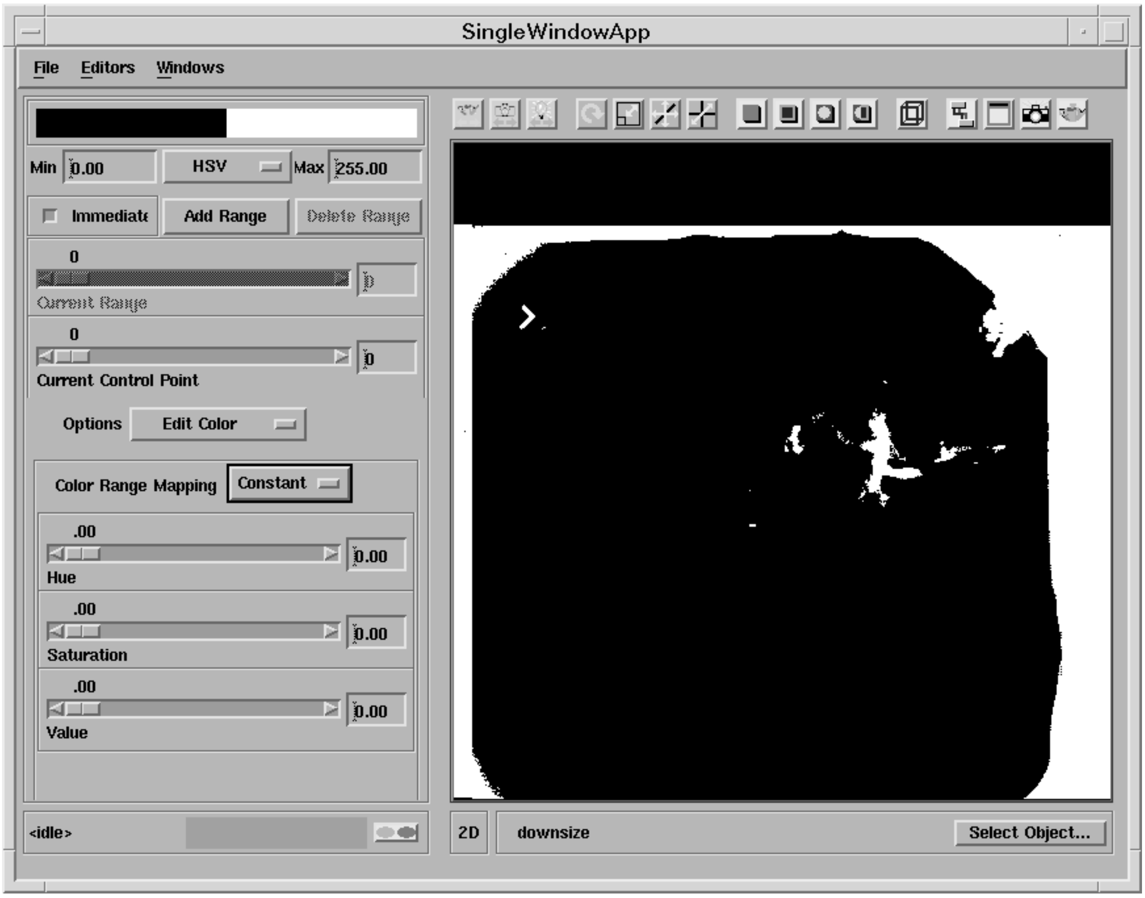
- There are two colour models which may be used
    - **HSV** (**H**ue **S**aturation **V**alue)
    - **RGB** (**R**ed **G**reen **B**lue)
- The **colour range maps** can either be **linear** (interpolated between max and min values of data) or **stepped** (constant colour for each step). The mapping is set by **control points**.
- The datamap range and the data range can be altered or additional sub-ranges added to give a more precise representation of the data.

3.6.7 Changing the colour map (3)

There are several pre-defined datamaps which can be used to view the image. Now you are going to alter the datamap to obtain a greyscale image.

- In the **Options** menu select the **input/output** option.
- In the **Action** menu select **Read Datamap**.
- In the datamap options select **Greyscale** and click apply.
- The image in the dataviewer should now be a greyscale image, with black set to 0 and white set to 255.

3.6.8 Changing the colour map (4)



Now you are going to turn the greyscale image into a binary black and white image.

- From the **Options** menu select **Edit Colour**.
- Change the **Colour mapping range** from **linear** to **constant**. A binary map should be obtained as shown above.
- Select **Edit Range/Data** from the **Options** menu.
- Select sub-range and alter the values of max and min to alter the position of the black/white division.

Try out some of the other datamap parameters yourself.2D Visualization Techniques

There are many 2D visualization techniques. There is a library of modules designed for 2D images called `Image_Processing` in the `Accessories` library (this is written `Accessories.Image_Processing`) which have a range of image processing functionality.

3.7 Exercise 2

The aim of this exercise is to represent the temperatures taken from a cross section of an injection moulding system.

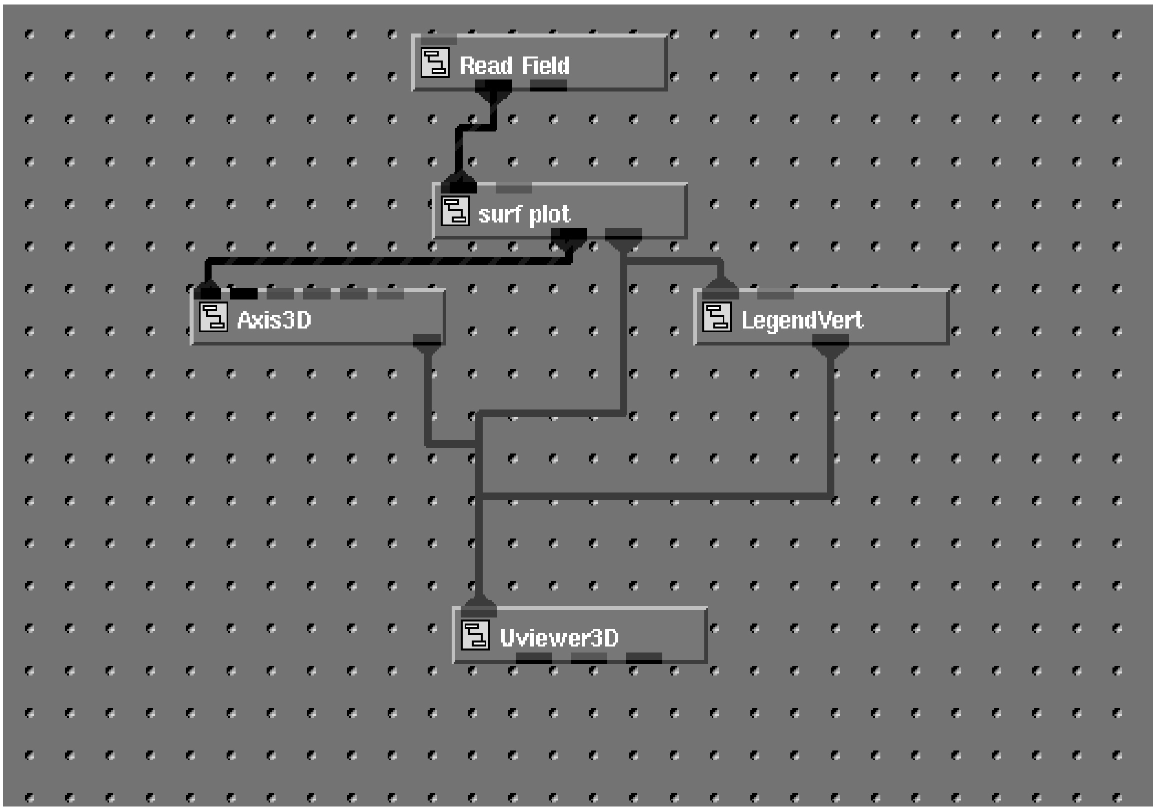
3.7.1 Importing the `temps.dat` data (1)

Below is a description of the data contained in the datafile `temps.dat`:

```
The ASCII data file contains data on a regular, 31 x 31 grid. The data
component at each element is a single floating point value and the first two
numbers in the file indicate the X and Y dimensions of the grid respectively.
```

- Create a file called `temps.fld` to describe the data contained in temps.dat (rather than re-type the field format edit the existing `mri.fld` and save it under a new filename).
- Use the template field file given on page 26 and the information given above to complete the specification.

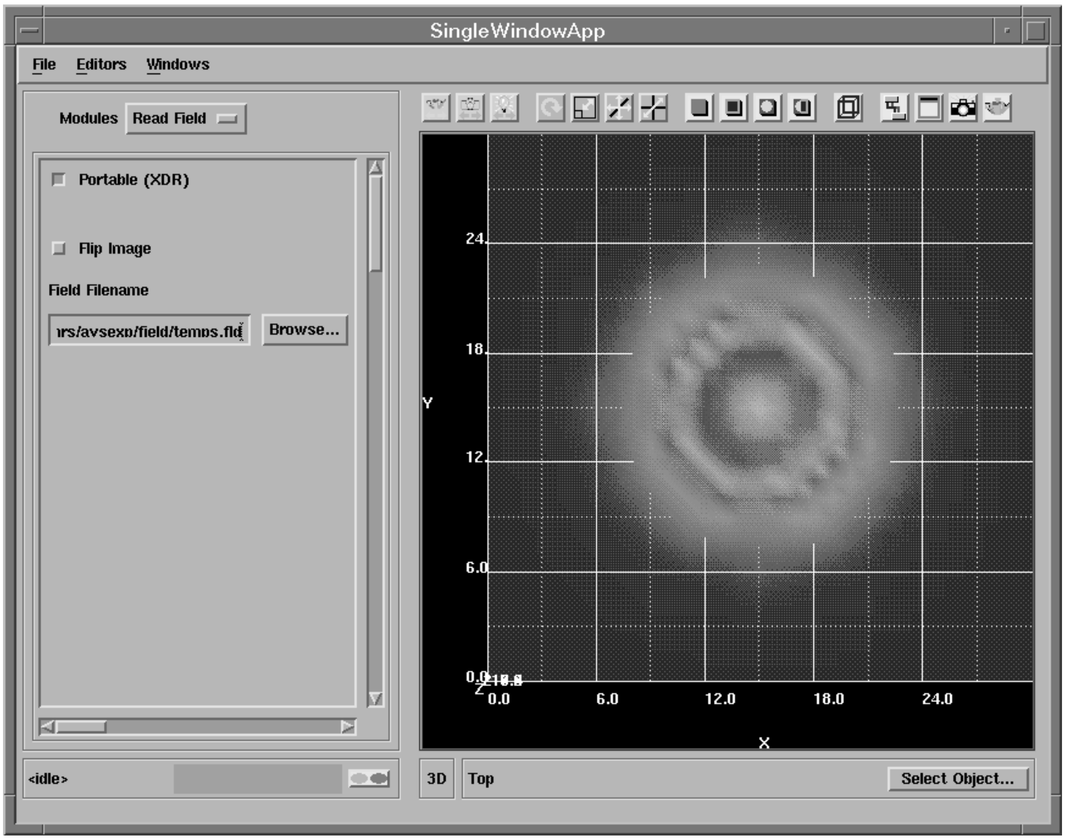
3.7.2 Displaying the temperature data with `surf_plot`



Using the network editor construct the above network. Descriptions of the modules are given below:

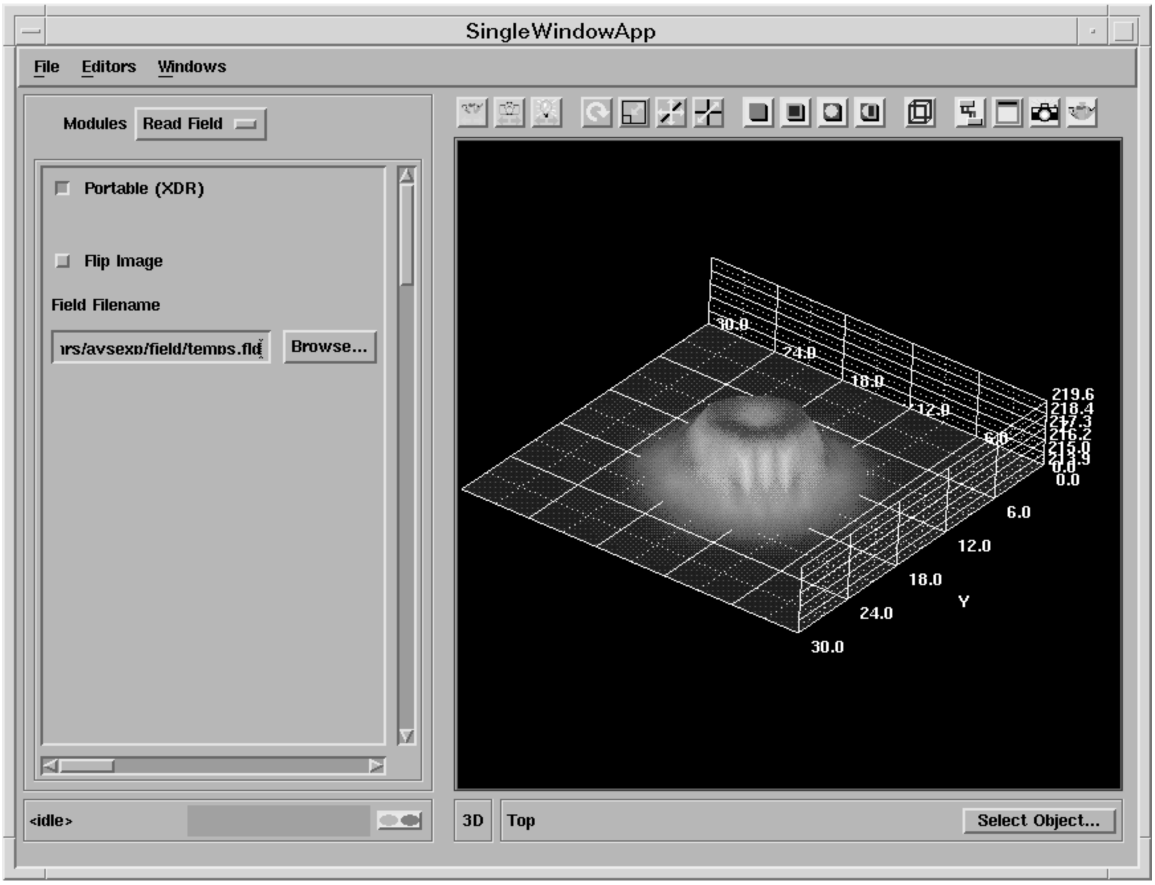
- `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
- `LegendVert` (Geometries): produces a color legend which displays the object's datamap.
- `surf_plot` (Mappers): creates a 2D or 3D mesh whose height is proportional to the scalar data value at each point. 2D data produces a 3D mesh and 1D data produces a 2D mesh.
- `TextString3D` (Geometries): produces a label.
- `Axes3D` (Geometries): creates an XYZ axis grid with labels based on mesh extents.
- `Uviewer3D` (Viewers): produces a 3D image in the DataViewer

3.7.3 Importing the `temps.dat` data (2)



- Select the `Read_Field` module from the **Modules** editor panel and press the browse button. The file browser should now appear.
- To import the data into the network, select the file `temps.fld` you have just created using the file browser.
- If a picture similar to the one above appears then you have been successful.
- It often improves the appearance of the image upon rotation to change the light to bi-directional:
    - In the **Editors** menu select **Light**.
    - In the **light-type** menu select **bi-directional**.

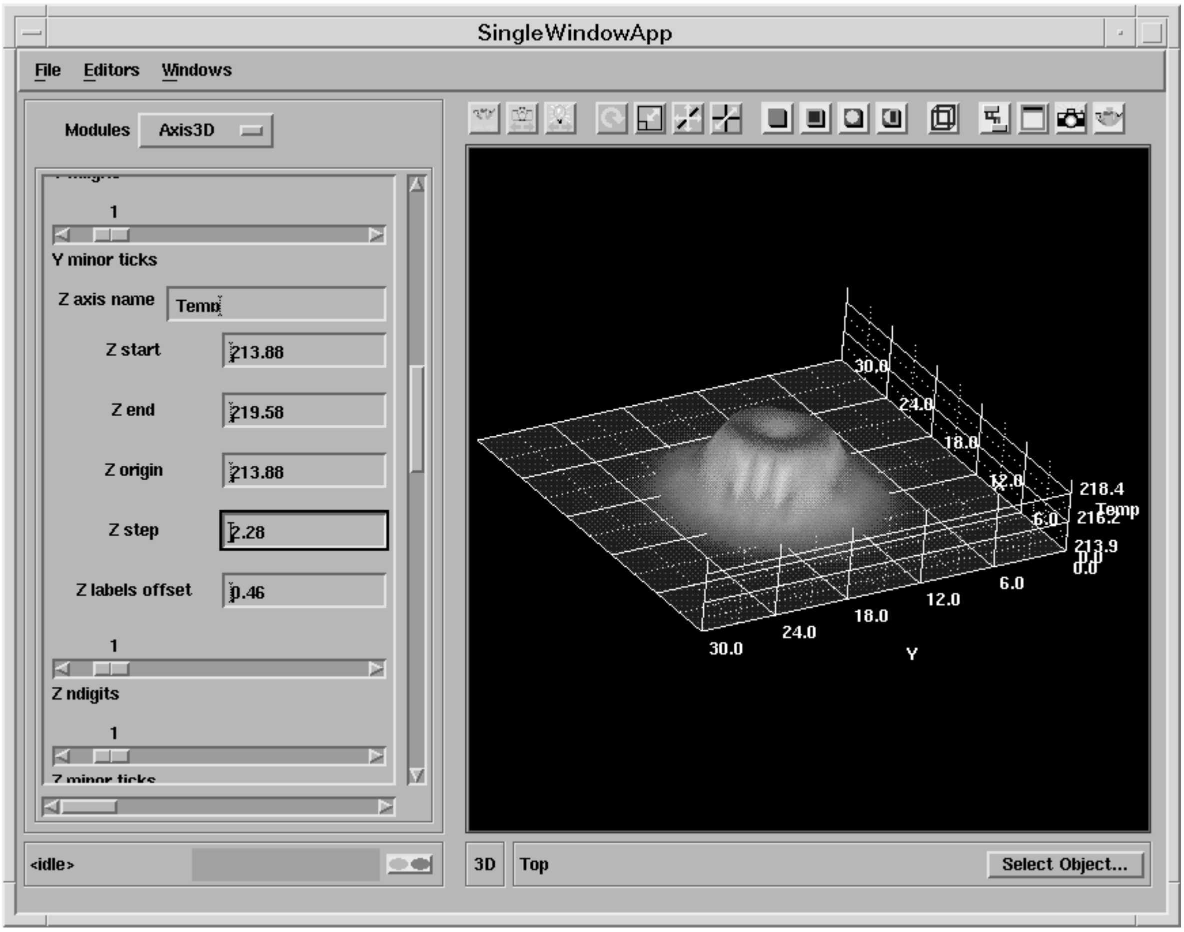
3.7.4 Normalizing the picture



Click on the **Reset/Normalize/Center** icon in the Toolbar so you can fit the whole object in the window.

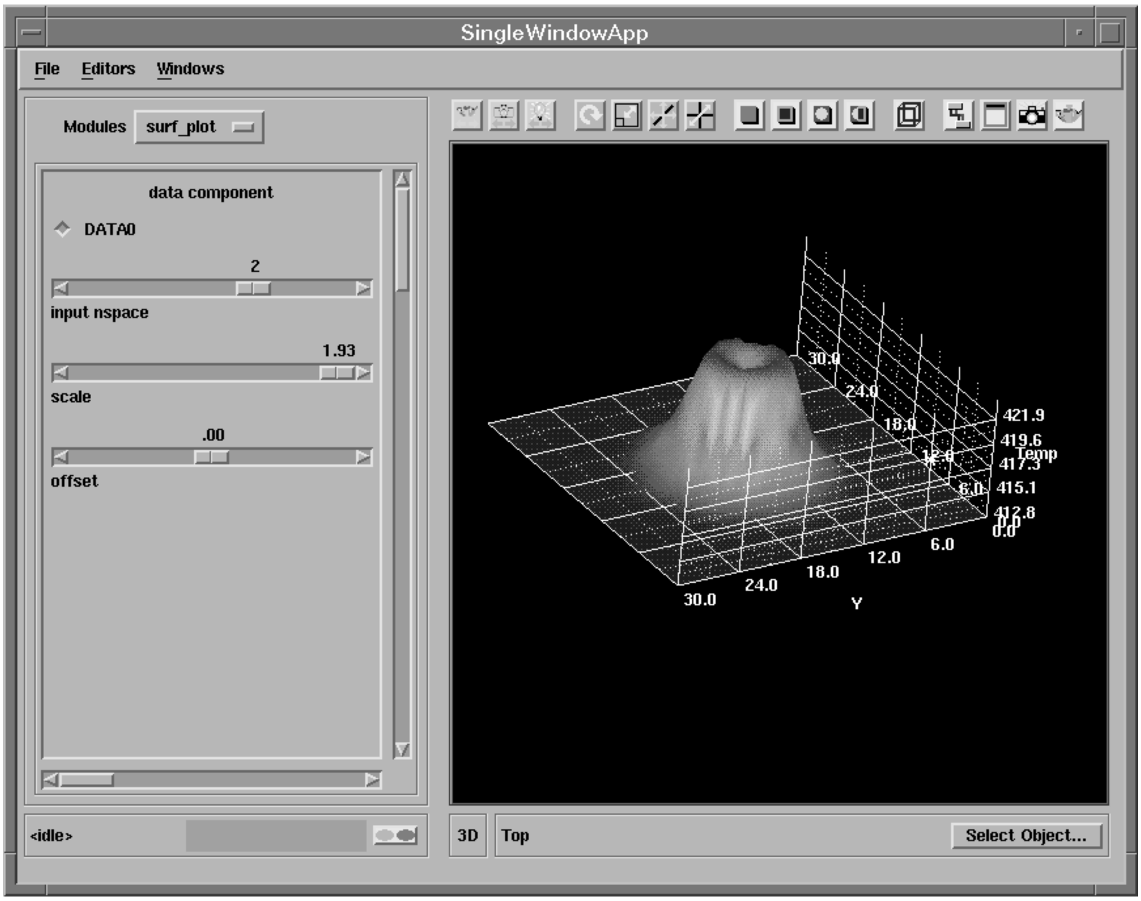
Now using the mouse, transform the object so it appears as above.

3.7.5 Altering `LegendVert` and `Axes3D`



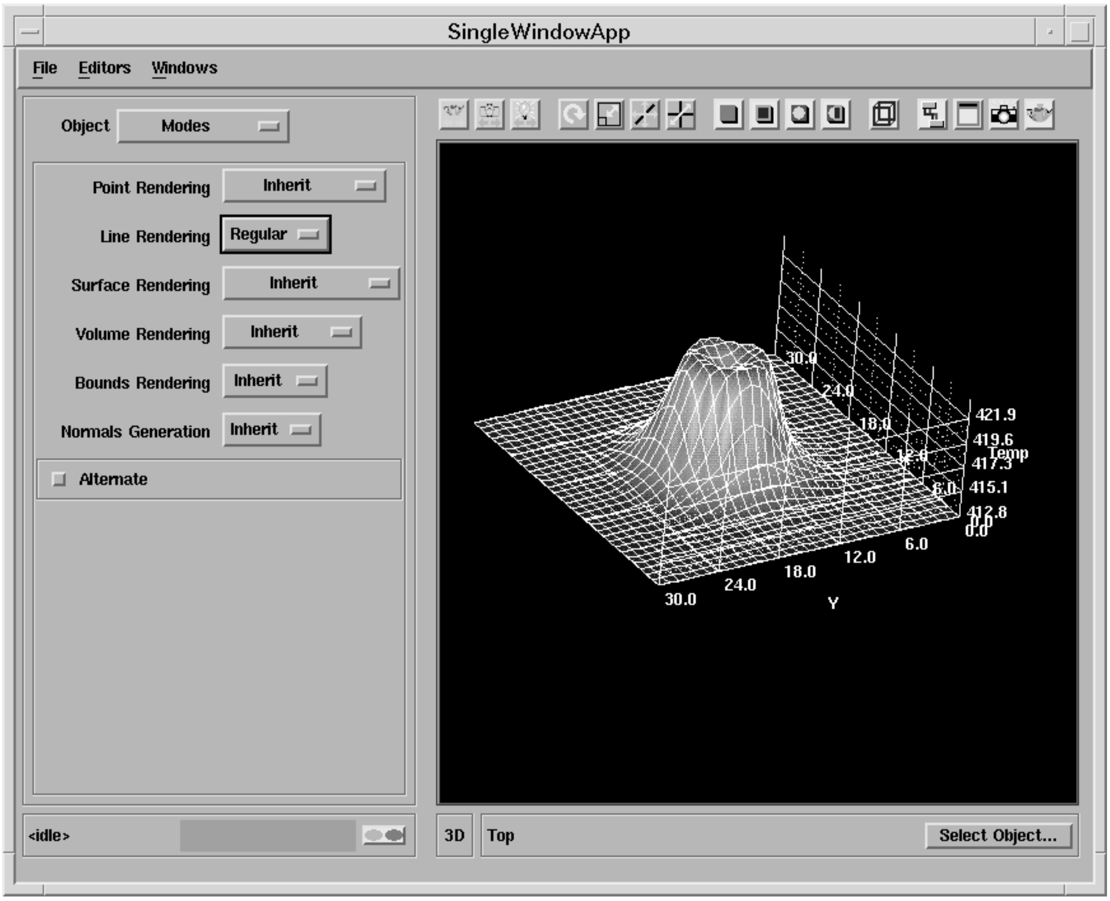
- Select the `LegendVert` module from the **Modules** editor panel
- Change the parameters of the colour bar or legend to make it look optimal in the picture.
- Similarly, select the `Axes3D` module from the **Modules** editor panel:
    - Reduce the number of steps on the z-axis.
    - Label the z-axis "Temperature".

3.7.6 Using `surf_plot`



- Select the `surf_plot` module from the **Modules** editor panel
- Alter the slider to increase the scaling.
- The module should now produce a picture as shown above.

3.7.7 Enhancing the picture



- Select the **Object** command from the **Editors** pull-down menu or the Toolbar to invoke the **Object editor**.
- Select **Modes** from the **Object** menu.
- Select the mode of **Line Rendering** as **Regular**. This shades the image and superimposes a wireframe image on the `surf_plot` image.
- Select one of the `TextString3D` parameters, and enter "Temp (C)" in the string box. Re-position next to the legend and alter the other parameters, e.g. font, if necessary.
- Select the other `TextString3D` parameters, and enter "temp.dat visualized with surf_plot" in the string box. Again re-position and alter the other parameters if necessary.

4 Visualizing 3D arrays of data

4.1 Aims of the chapter

The aim of this chapter is for you to learn techniques which are used to visualize 3D data. You are to work through several structured exercises which will demonstrate the visualization process: creating field files, constructing applications and manipulating data in 3D. The solutions to the field files for 3D data are given in Appendix III .

4.2 Exercise 1

The aim of this exercise is to see how vector fields can be visualized, such fields may be similar to multi-spectral MR scans which have several data values for each data point. To do this you will visualize a downsized version of the AVS/Express sample dataset `fin.fld` which shows the flow of air over a blunt fin as no multi-spectral MR datasets were available when this course was produced.

4.2.1 Importing the `fin.dat` data

Below is a description of the data contained in the datafile `fin.dat`:

```
The data file is called fin.dat and is an ASCII file which represents an 10x8x8
irregular grid. Each data element has 5 floating point data components and each
element has some associated coordinate data to position it in 3D space. The file
fin.dat is in /usr/express/data/field/med.
```

The data file format is as follows:

```
density x-momentum y-momentum z-momentum stagnation x-coord y-coord z-coord
```

and a small extract from the file is shown below:

```
2.171 0 0 0 10.7602  0 0 0
2.1523 0 0 0 10.6423  0.0160473 0.129677 0
1.9742 0 0 0 9.61927  0.0757717 0.266559 0
1.6659 0 0 0 7.96548  0.173189 0.379764 0
1.2728 0 0 0 5.94705  0.299646 0.45922 0
0.86119 0 0 0 3.92577  0.443802 0.497866 0
0.7654 0 0 0 3.6417  0.618551 0.501378 0
0.73957 0 0 0 3.52176  1.08654 0.501378 0
0.55002 0 0 0 2.62795  2.47197 0.501378 0
0.45067 0 0 0 2.20615  6.57433 0.501378 0
```

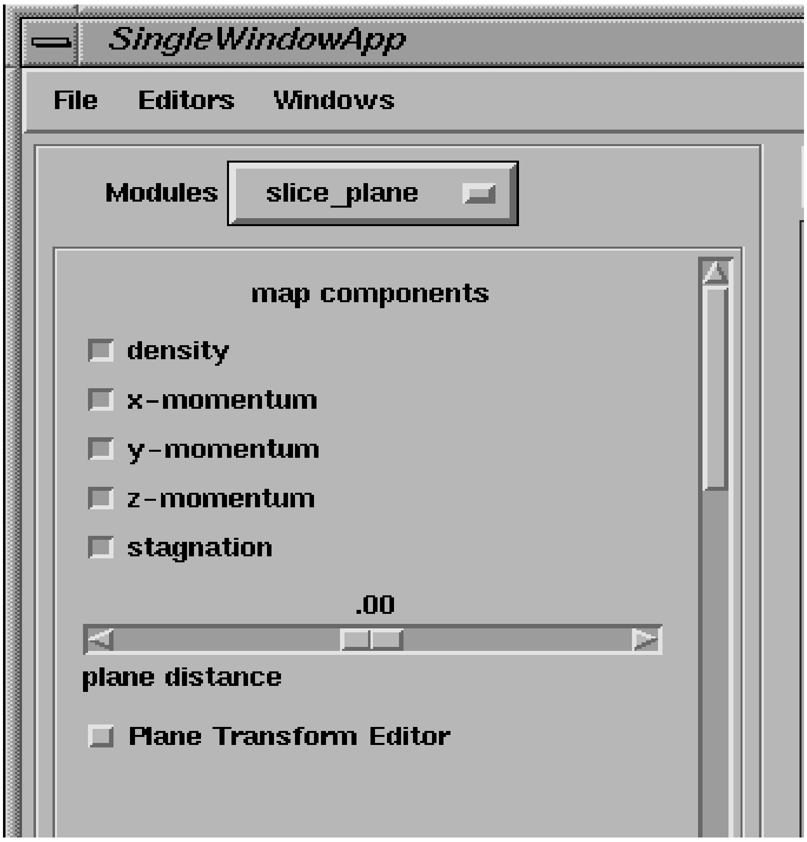
- Create a file called `fin.fld` to describe the data contained in `fin.dat`.
- Use the template field file given on page 28 and the information given above to complete the specification.

4.2.2 Displaying the vector data

Create the above network using the following modules:

- `Read_Field` (Data Input): reads an AVS field file and converts it into an AVS/Express field.
- `slice_plane` (Mappers): extracts a 2D slice from a 3D field with an arbitrarily positioned slice plane.
- `combine_vect` (Filters): takes all selected scalar data components and combines to output a single vector component
- `bounds` (Mappers): generates a bounding box of a 3D field.
- `Arrow1` (Geometries): creates a wireframe arrow shaped mesh
- `glyph` (Mappers): places a geometrical object at each node of an input field. The glyph is coloured and sized according to the magnitude of the data component at that point.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

4.2.3 Selecting the map components

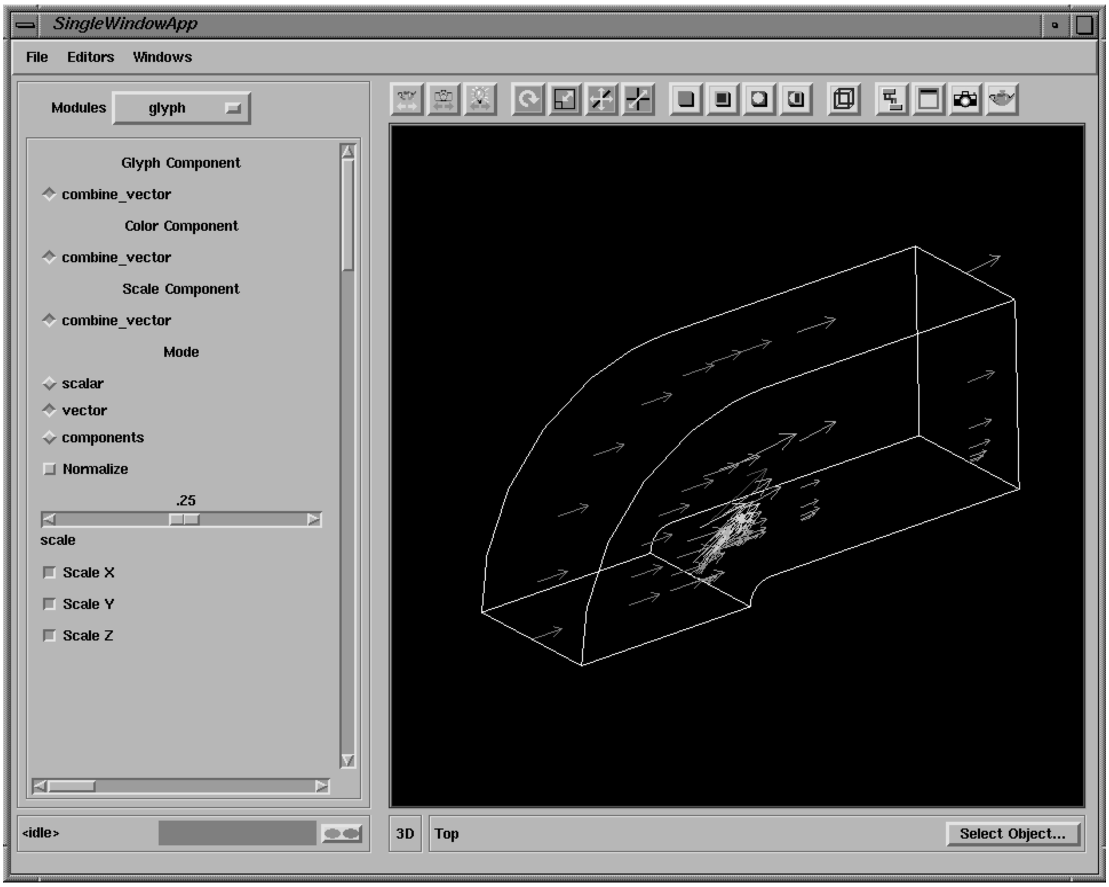


- Select the `Read_Field` module from the **Modules** editor panel and then select the file `fin.fld` which you have just created to import the data using the file browser.
- Select the control panel for the `slice_plane` module and make active all data components, as shown above. For now ignore any error messages which appear.
- This module will now pass on a 3D field with a 5-vector data element consisting of density, x, y and z momentum, and stagnation to the other modules in the AVS/Express network.

4.2.4 Extracting the vector data

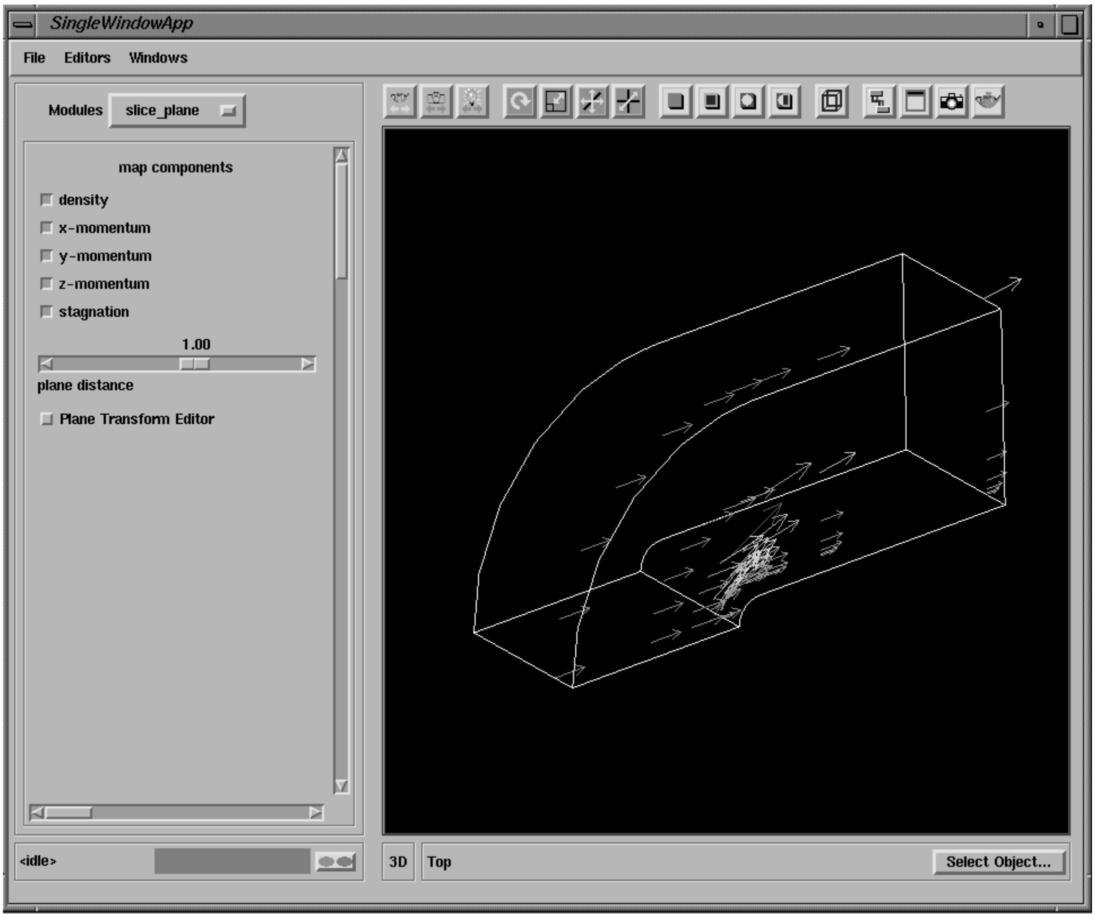
- Select the control panel for the `combine_vect` module and make active the three data components x-momentum, y-momentum and z-momentum, as shown above. Since `veclen` is set to 3 it is only possible to have 3 vector components active at a time.
- This module will now pass on a 3D field with a 3-vector data element consisting of x, y and z momentum to the other modules in the AVS/Express network.

4.2.5 Using the **`glyph`** module



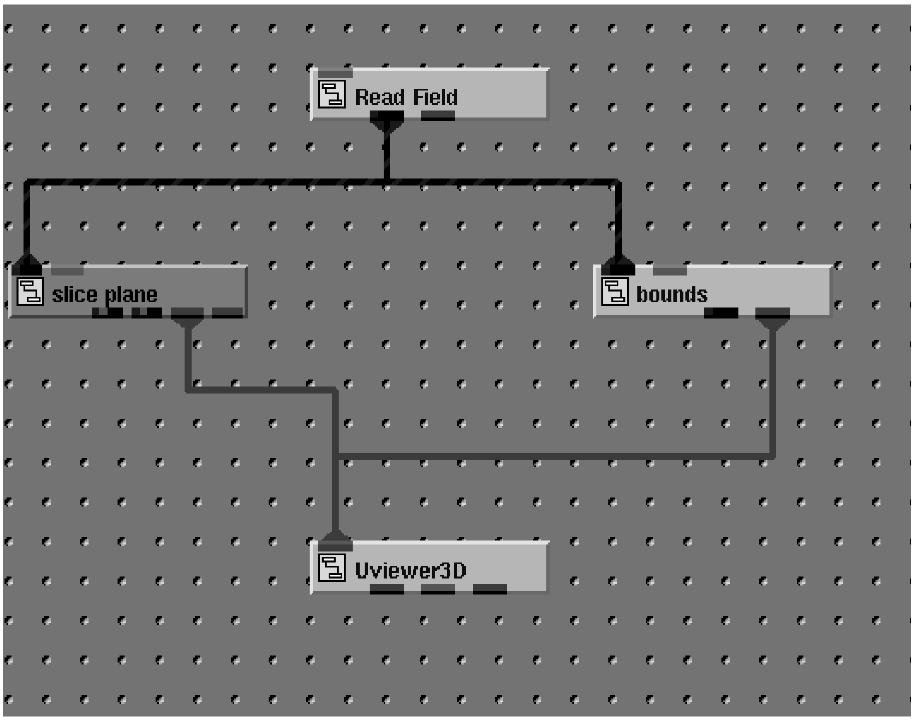
- Select the `glyph` module's control panel.
- Reduce the scale value using the slider to make the arrows a more acceptable size
- Try altering some of the parameters - use the module help facility to understand their effect.

4.2.6 Moving the `slice_plane`



- Select the `slice_plane` module's control panel.
- Alter the plane distance value to observe the value of momentum at different positions within the data set.
- Try altering some of the plane transformation parameters to change the orientation of the slice plane.

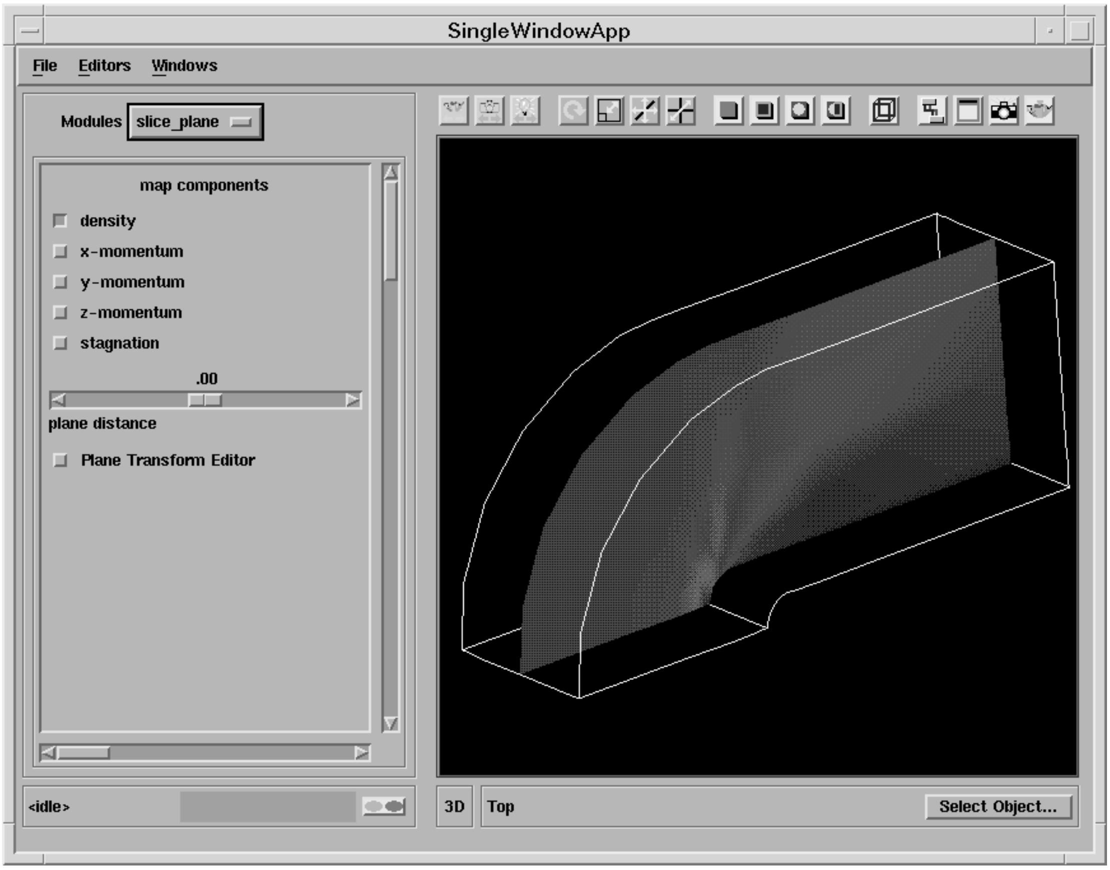
4.2.7 Visualizing the scalar data



Change the network to create the above network:

- `Read_Field` (Data Input): reads an AVS field file and converts it into an AVS/Express field.
- `slice_plane` (Mappers): extracts a 2D slice from a 3D field with an arbitrarily positioned slice plane.
- `bounds` (Mappers): generates a bounding box of a 3D field.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

4.2.8 Using the `slice_plane` module



- Select the `Read_Field` module from the **Modules** editor panel and then select the file `fin.fld` to import the data using the file browser.
- Select the control panel for the `slice_plane` module and make active the density data component.
- This module will now pass on a 3D field with a scalar data element to the other modules in the AVS network.
- The viewer window should now show something similar to the above figure.
- Use the module's control panel to change the slice position.
- Try extracting the other scalar components using the module.

4.3 Using the software renderer

You can use the renderer option to alter the renderer which is used to draw the display. There are several options:

- **Software**: software renderer which uses its own graphics rendering techniques.
- **Hardware**: uses the workstations own software and hardware graphics techniques.

The hardware renderer is used as the default. However, when using the volume renderer to look at 3D images, it is often necessary to use the software renderer. The software renderer is selected in the following way:

- Select the **View editor** from the **Editors** pull down menu.
- In the **Renderer** menu select **Software**.
- Proceed with the example as usual.

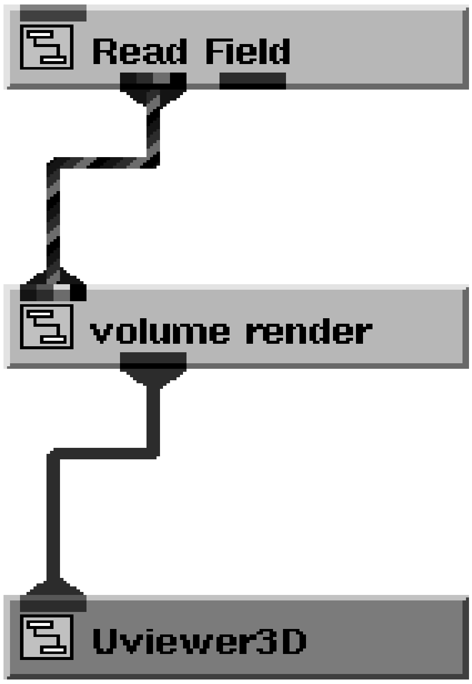
The rest of the exercises in this chapter are concerned with 3D representation of various fields. The examples use the `volume_render` module and all require the use of the software renderer.

Note: Use of the software renderer will significantly slow down the visualization process - but for volume rendering this is unavoidable.

4.4 Exercise 3

The volume to be represented in this example is a 3D image of a human head.

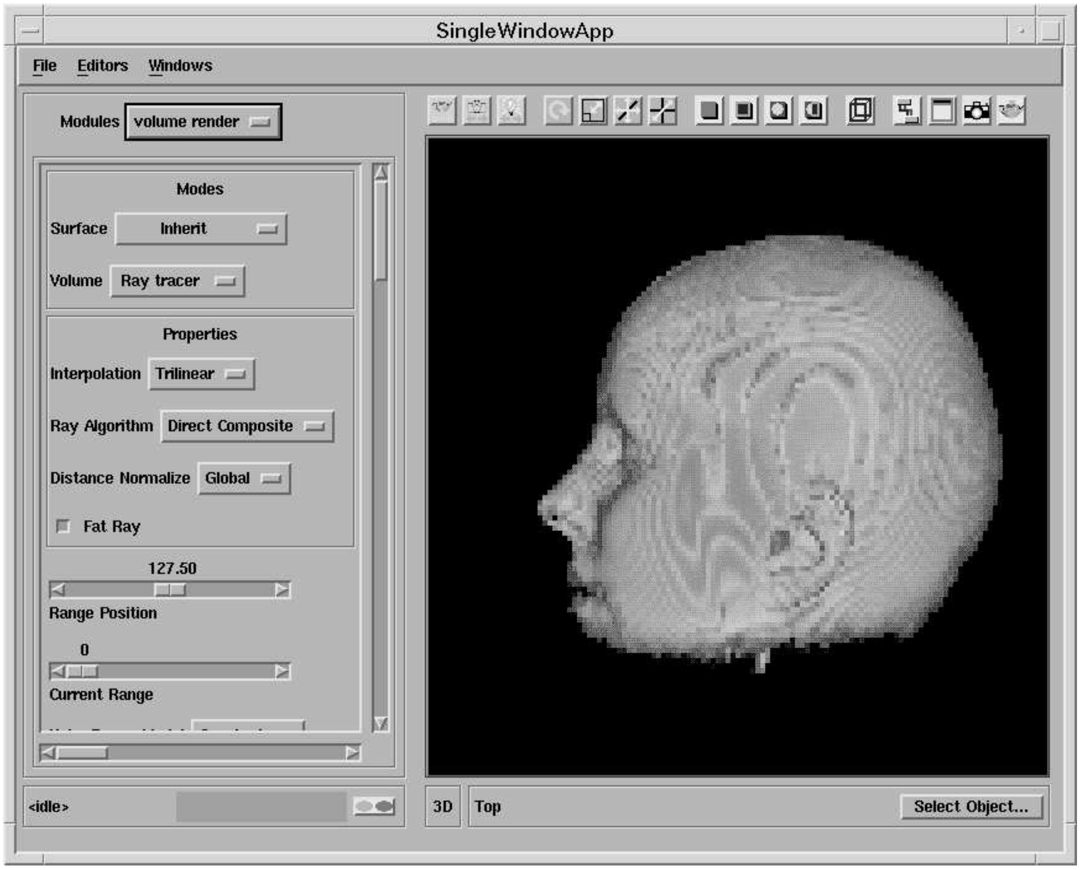
4.4.1 Displaying the `smlhead.fld` data using `volume_render`



Create the above network using the following modules:

- `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
- `volume_render` (Mappers): directly renders a volume.
- `Uviewer3D` (Viewers): creates a 3D image in the Data viewer.

4.4.2 Normalizing the picture



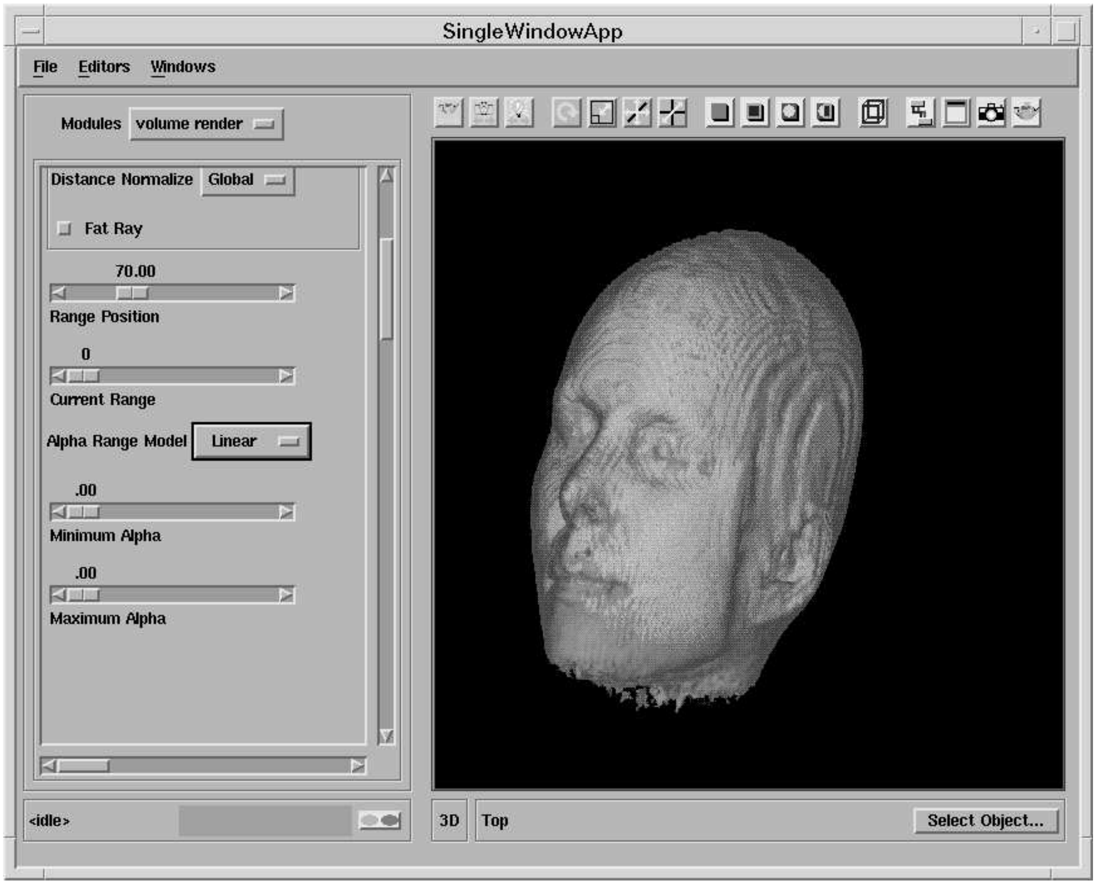
- Select the `Read_Field` module's control panel and select the file sml`head.fld`, from the file browser, to import the data. The `smlhead.fld` file can be found under the path `/usr/express/data/field/med.`

Note - if an error message is reported when `Read_Field` is selected, or if no image appears, it is likely that you have forgotten to switch on the software renderer

- Click on the **Reset/Normalize/Center** icon in the Tool bar so you can fit the whole object in the window. The head should now be visible.
- Select the `volume_render` module's control panel as shown above.

The steps listed on the next page will improve the appearance of the image and explain some of the properties of `volume_render.`

4.4.3 Using `volume_render`

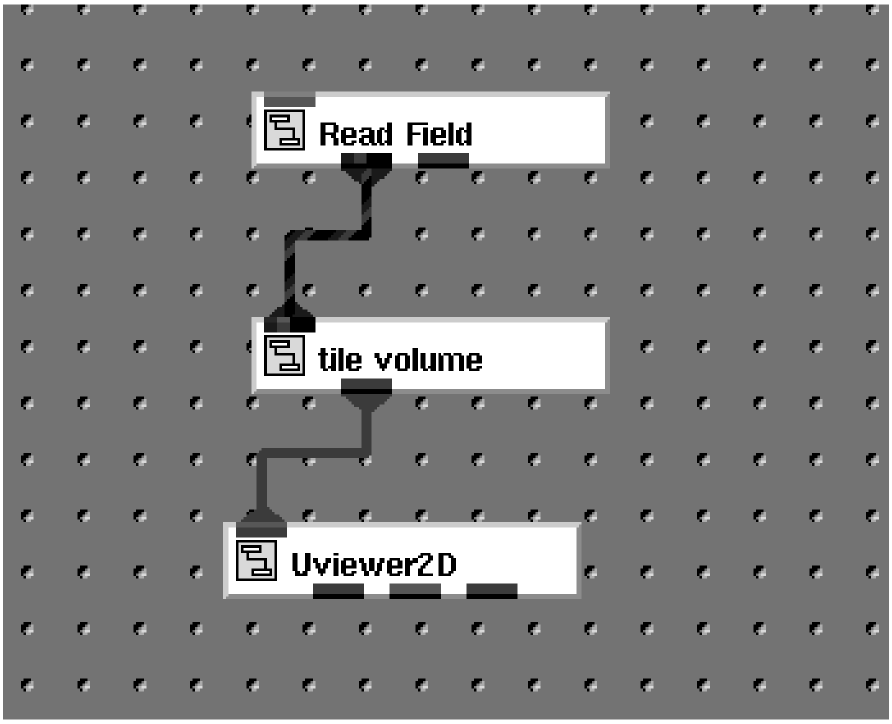


- Switch the **surface mode** to **Gouraud**. This selection provides the highest quality rendering - including both lighting and interpolation.
- Switch off the **Fat Ray** option. The resolution of the image should improve.
- The `volume_render` module has two default datamaps. The range to be edited can be selected by altering the value of **Current Range**. The **Range position** value allows the mid-point of the two ranges to be altered. Set the Range position value to approximately 70.
- Set the **Alpha Range Model** to **Linear**. Now the alpha component is interpolated between the maximum and minimum of the current data range. The other option is **constant**, the alpha component for the whole data range is set to the minimum alpha value.
- Rotate the head so it appears as above.
- Try altering some of the other `volume_render` parameters and observe the effect.

4.5 Exercise 4

This example will use the `tile_volume` module to let you decompose and a volume into a series of images and display them.

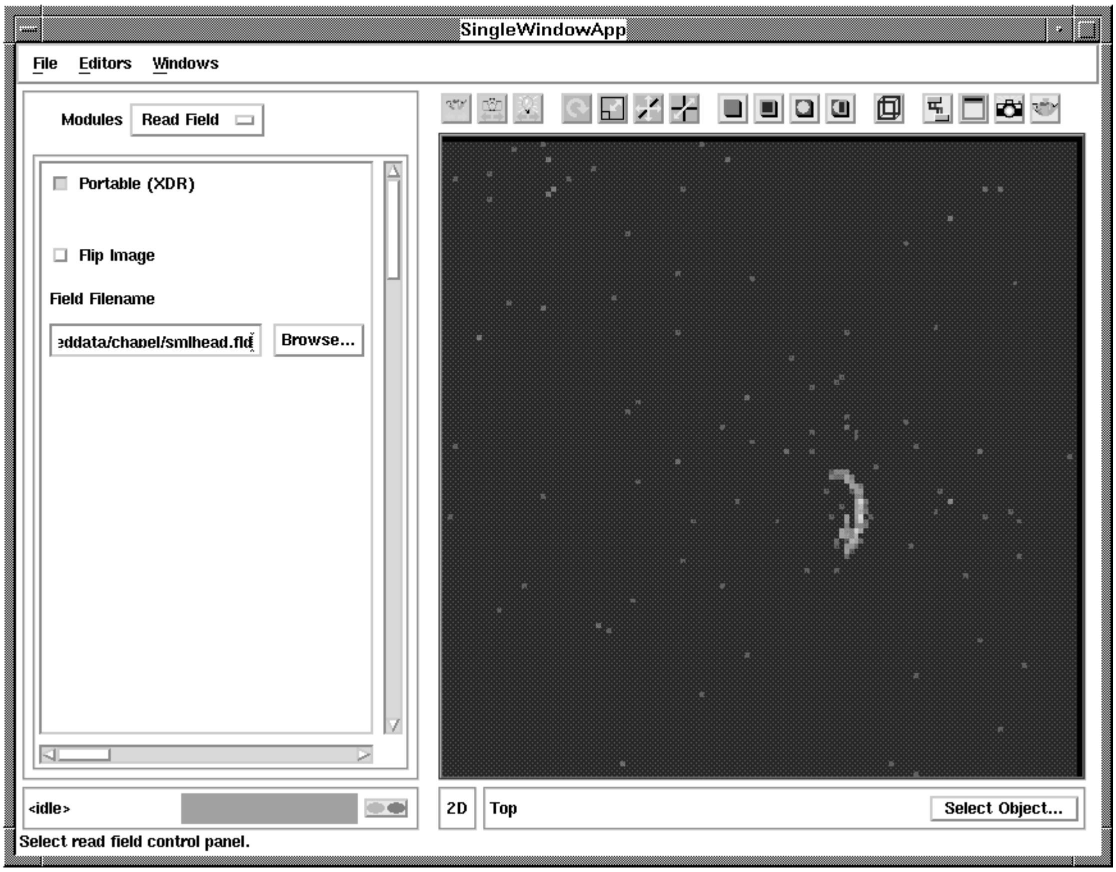
4.5.1 Displaying the `smlhead.fld` data as a series of its component images using tile-volume



Create the above network using the following modules:

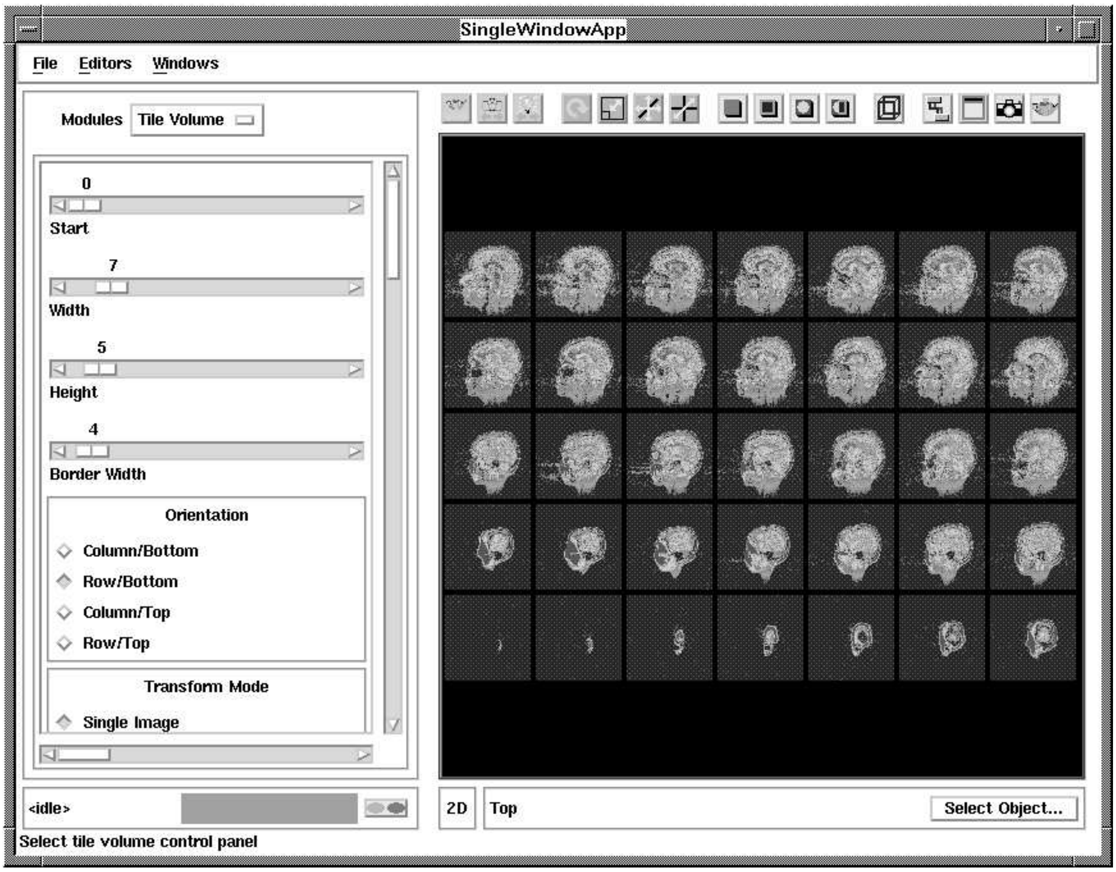
- `Read_Field` (Data IO): reads as AVS field file and converts it into an AVS/Express field.
- `tile_volume` (Mappers): creates a tiled display from a volume.
- Uviewer2D (Viewers): creates a 2D image in the Data viewer.

4.5.2 Importing The Data `smlhead.fld`



- Select the `Read_Field` module's control panel and select the file `smlhead.fld` from `/usr/express/data/field/med`.
- The viewer shows just one slice of the data, the first one.

4.5.3 Using `tile-volume`



- Select the `tile_volume` control panel from the **Modules** menu.
- Alter the **Width** and **Height** values to control the number of tiles to be displayed. If the product of the **Width** and **Height** values are greater than the **dim3** value in the field file then the screen will appear blank, the `smlhead.fld` data has a **dim3** value of 55.
- The **Boarder Width** value controls the width of a boarder around each tile.
- The **Orientation menu** gives four options for the layout of the tiles. The parameter specifies the position of the start slice and the direction of the display, either along a row or column.
- The value of **Start** selects the slice in the volume used to begin the tiled display.

6 AVS/Express examples

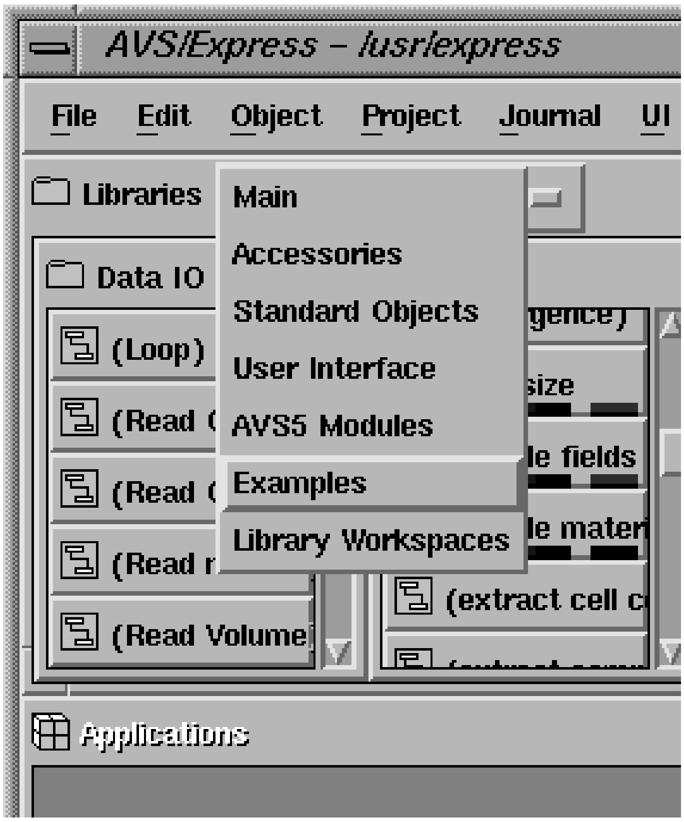
6.1 Aims of the chapter

There are numerous on-line examples which are included with AVS/Express to illustrate the operation and applications of most of the AVS/Express modules. In this chapter you will investigate several of these example networks.

6.2 Exercise 1

This exercise will illustrate the use of these on-line examples using the `Isosurface` module as a demonstration. The `Isosurface` creates a surface in a data field around the data points with the same value. The `Isosurface` module is used to produce a 3D solid model from a volume of data values.

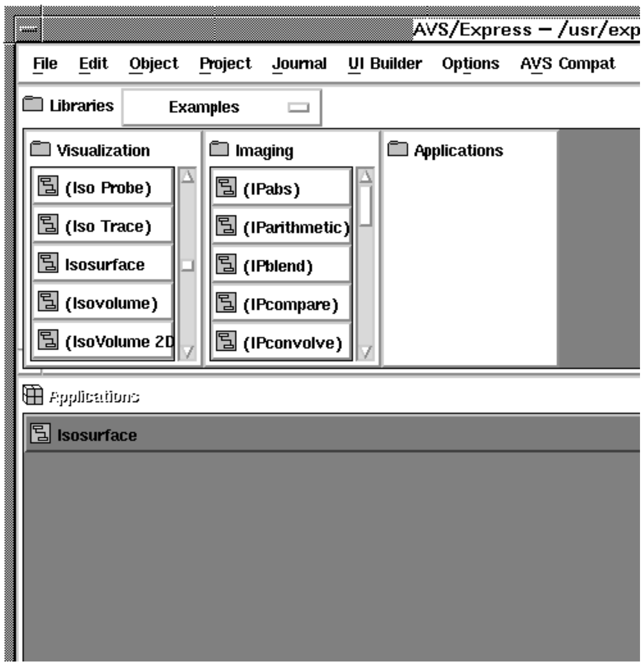
6.2.1 Entering the AVS/Express examples library



To access to the AVS/Express on-line examples

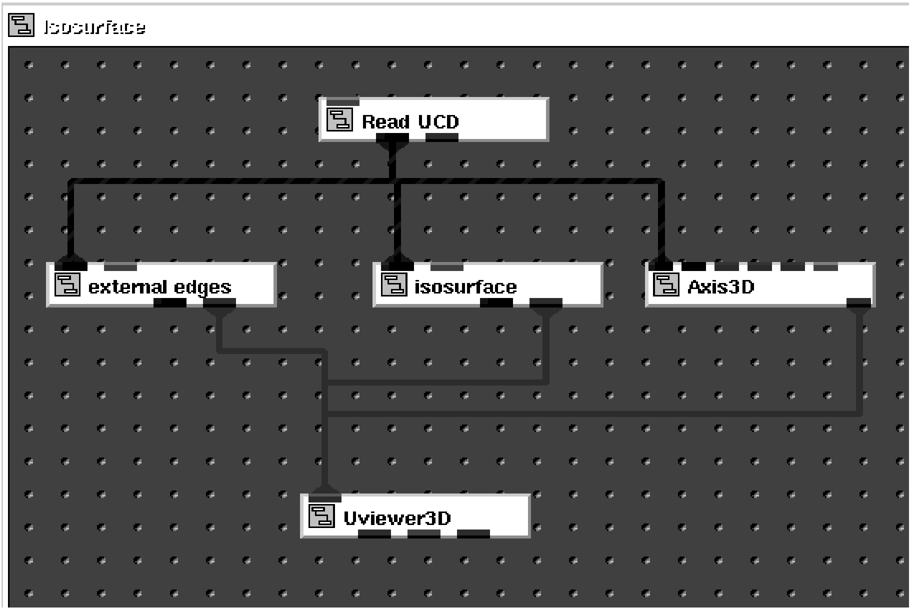
- Select the **Examples** library page from the **Library page** selector on the top of the **Network Editor window.**

6.2.2 Starting the `Isosurface` demo



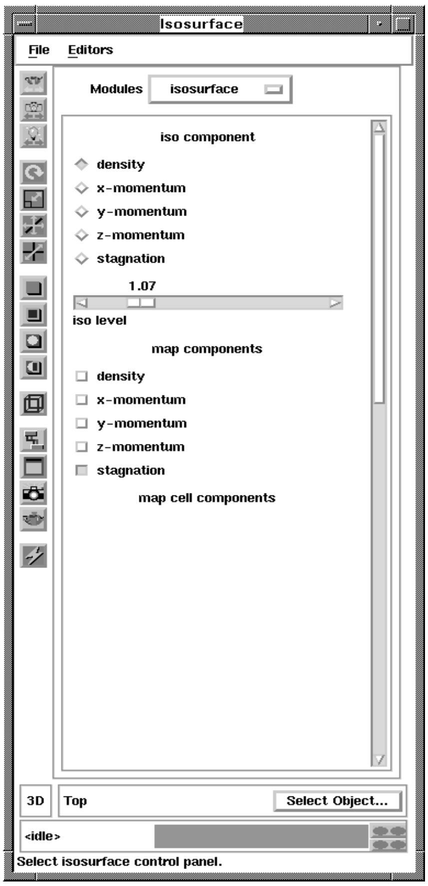
- Double click the **Visualization** sub-library to open it. (It may already be open).
- Use the left mouse button to drag the button to the right of the **Visualization library** and leaf through the library.
- Instance the `Isosurface` demo module to the workspace.
- The screen should appear as shown below.

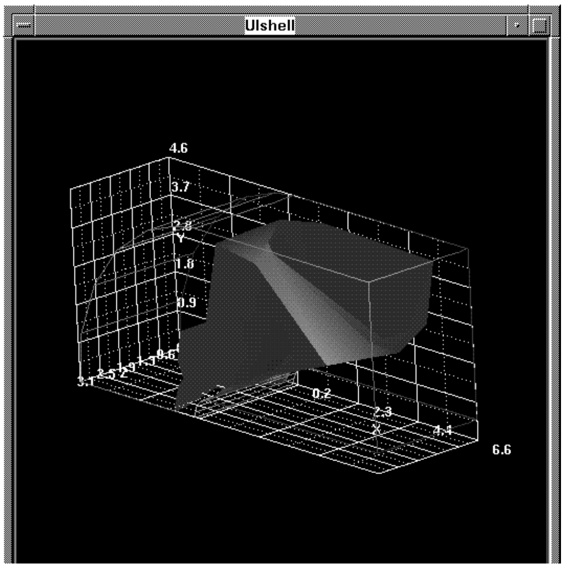
6.2.3 The `Isosurface` demo



- Double click the `Isosurface` in the workspace to maximize it. By doing so, the complete `Isosurface` network is shown in the workspace.
- Refer to each module's help documentation to understand the way the network operates, especially the `Isosurface` module.
- This time the viewer and the control panel are in two separate windows. This is a Mutiwindow DataViewer and the control panel is not connected to the DataViewer/DataViewers.

6.2.4 Changing the parameters yourself





- Select the Isosurface module from the Modules menu.

The data used in this example is the same one used in Chapter 4 Exercise 1. This data set has many data values for each data point. An isosurface can be produced for anyone of these components

- The default value for **iso component** is **density**. This is the component for which the isosurface will be produced.
- The slider **iso level** gives the data value for the isosurface.
- The isosurface can be coloured by the data values of another component. This is the **map component.**

6.3 Other exercises

Try out some of the other examples in the Examples library, all of the ones with the prefix *iso* may be useful in for medical applications. Use the module documentation to help you understand their functions.

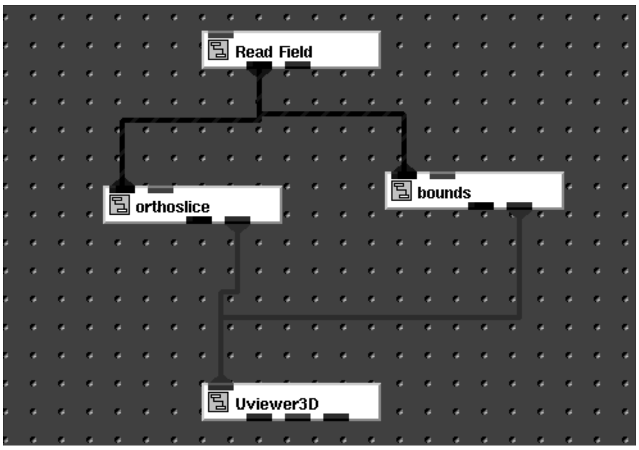
6 Obtaining Output From The Data Viewer

It is useful to obtain a copy of the image seen in the DataViewer either by the getting a hardcopy (printout) or by obtaining a 3D model or it (in the from of a VRML file).

6.1 Exercise1

The aim of this exercise is produce a postscript file that could be sent to a printer to produce a paper copy of the image.

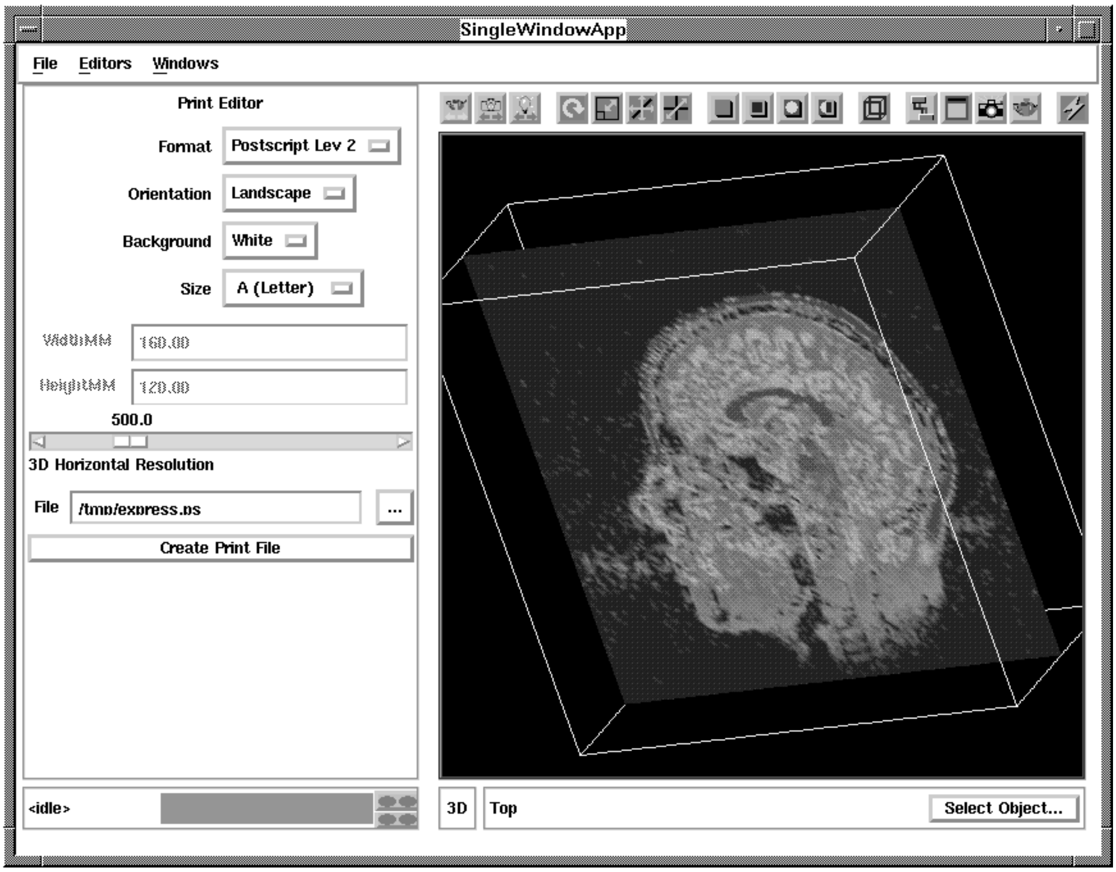
6.1.1 Obtaining hardcopy from AVS/Express



Create the above network and read in the `smlhead.fld` data (Although any network with a DataViewer could be used)

- `Read_Field` (Data IO): reads an AVS field file and converts into an AVS/Express field.
- `orthoslice` (Mappers): produces a slice of a structured field perpendicular to the coordinate axis.
- `bounds` (Mappers): generates a 3D bounding box of a structured field.
- `Uviewer3D` (Mappers): creates a 3D image in the Data viewer.

6.1.2 Producing The Postscript File



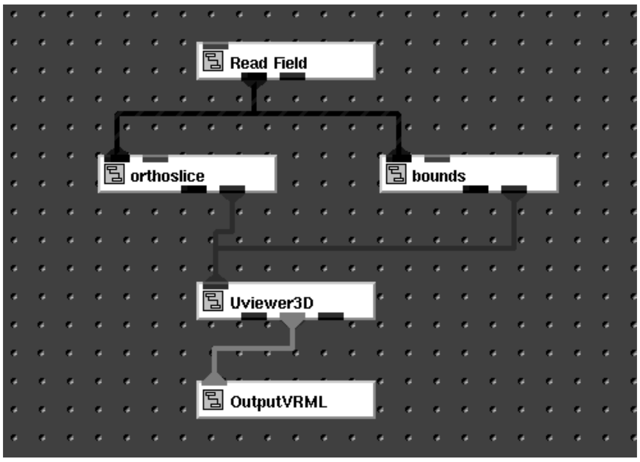
You can print the contents of the **Viewer** window by using the interactive tool - **Print Editor**.

- From the **Editors** menu select **Print** (For applications with more than one DataViewer there is one **Print Editor** for each view).
- Select the appropriate parameters.
    - **Format** gives the file type, postscript is suitable for most printers and colour or black and white can be created.
    - **Orientation** is **Landscape** or **Portrait**.
    - **Background** can be set to **Black** or **White** or the original background from the DataViewer.
    - The printout can be in a variety of sizes.
- To alter the default file name and directory delete the default address and type the new one into the **File** field box. Alternatively get a dialogue box by double clicking with the left mouse button on the button to the right of the **File** field box labelled with three dots.
- When you are happy with the parameter settings and file name click "Create Print File".
- From a shell window you can view and the image file and send it to the printer. Most Unix machines have xv, ghostview or ghostscript which will view a postscript file, Silicon Graphics machines may have imgview instead.

6.2 Exercise 2

You can create a VRML file of the contents of any Viewer window by using the OutputVRML module, the resulting VRML file can be viewed by any suitable web or VRML browser.

6.2.1 Obtaining VRML From AVS/Express

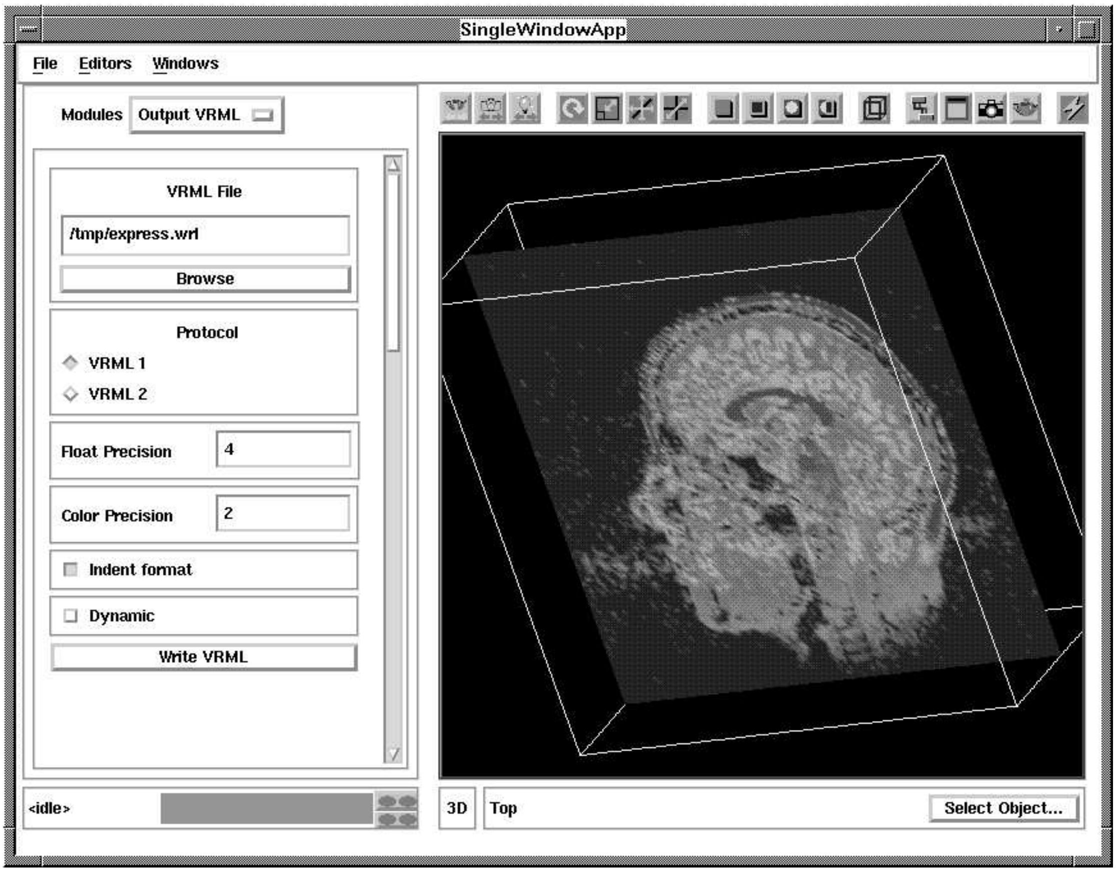


Use the same application as the previous exercise and add the `OutputVRML` module:

- `OutputVRML` (Viewers): generates a virtual reality modelling language (VRML) file by capturing the contents of the view

- The `OutputVRML` module can be connected to any **Uviewer** by the purple port.

6.2.2 Producing the VRML file



- Select the module control panel for the `Output VRML` module.
- Select the appropriate parameters.
    - **Protocol** for the resulting file can be of standard one or two.
    - **Float Precision** specifies the accuracy of all the parameters (e.g. coordinates except colour).
    - **Colour Precision** specifies the accuracy of just the colour.
    - **Indent** puts an indent on the assci vrml file so that the code is easier to understand.
    - **Dynamic** produces a new VRML file each time a new view is rendered.
- To alter the default file name delete the address and type the new one into the **VRML File** field box. Alternatively get a dialogue box by double clicking with the left mouse on the **Browse** button.
- When you are happy with the parameter settings and file name click “Write VRML”.
- To view the file you can use a suitable web browser or a specific tool for displaying 3D VRML scenes, such as vrweb or cosmoplayer.

7 Appendix I

7.1 Obtaining Output From AVS/Express

There are several ways of saving AVS/Express and its settings; applications, journals and projects. They are all similar but it is useful to understand how they differ.

AVS/Express has its own language called V which describes the state of all the AVS/Express components. All of these techniques use V code, but it is difficult to separate any part of AVS/Express from the rest. The IAC (International AVS Centre) is currently working on guide lines for the production of free standing modules that can be transferred between users.

7.1.1 Applications

The first chapter went through creating and saving applications. An application is a network of connected modules, its DataViewer and parameter settings. A saved application will produce exactly the same scene in the DataViewer so saving an application is a good way to keep a scene which you may wish to work on with a colleague.

To create an application:

- From the **File** menu on the Network Editor bar select **New Application...**
- A dialogue box appears allowing you to select an application type:
  - **Single-window DataViewer**: the viewer is connected to the control panel.
  - **Multi-window DataViewer**: the viewer is not connected to the control panel.
  - **Application**: no viewer comes with the application, although you can add one.
  - **Module Stack**: used for module development.
  - **Scratch Pad**: used for module development.
- A variety of Viewer Types are also available:
  - **3D**
  - **2D**
  - **3D and 2D**
  - **None**

To save an application:

- From the **File** menu select **Save Application...**.
- A dialogue box appears so you can select an appropriate location for the application's file.

7.1.2 Journals

Journaling lets you record a set of operations you perform on an application and replays them as a series of views in the DataViewer. AVS/Express records these operations as V statements and commands in a journal file with a*.v* extension. The file only stores the parameters so the application must be loaded before a journal is played back, if the wrong application is loaded or the wrong data set used then a fatal error may occur and crash the session.

To record a series of Network Editor operations:

- From the **Journal** menu on the Network Editor menu bar select the **Record** pull-down command. The **Record File** dialogue box will appear.
- Specify a journal filename.
- Select **OK** to begin recording. Perform the series of operations, e.g. build a network, set values and manipulate the object. (Any actions performed outside the workspace will be ignored).
- To stop recording select the **Stop** pull-down command from the **Journal** menu. The recording will stop and the resulting V file will be saved.

To play back the V file:

- Make sure that you are no longer recording and that your network is in the same state it was in when you began recording.
- Select the **Playback** pull-down command from The **Journal** menu. The **Playback File** dialogue box will appear.
- Select the journal file and click **OK**. The recording will be executed and cannot be stopped part way through.

7.1.2.1 Projects

Projects are used for module development and this was demonstrated in chapter 7. A project is saved as a group of files and directories which hold information about parameters and source code for your modules.

The modules are stored in the **Library Workspaces** and many copies of each module can be instanced from the library and used to produce a variety of new applications.

To produce a new project:

- From the **Project** menu on the Network Editor bar select **Save Project**.
- New modules are created in the Network Editor and dragged into the **Library Workspaces**. A new module may be created in a variety of ways:
    - Altering the default parameters of a standard module and dragging it into the **Library Workspace**.
    - Producing a network and dragging it into a **Macro** module and then dragging that **Macro** into the **Library Workspace**.
    - Producing a module which contains your own source code.
- After any additions to a project the whole project must be resaved.

To use a project:

- To use a project start AVS/Express the main project directory.

8 Appendix II

8.1 Customising AVS/Express

You can include a number of command line options when you enter the AVS/Express start-up command “express -viz” or “vxp”. Below is a list of a few of the available command line options which you may find useful:

- The -version option indicates which version of AVS/Express is installed. Does not start AVS/Express.
- The -usage option lists all the possible options. Print a summary of command line options. does not start AVS/Express.
- The filename option loads a specified V file into the Applications object. You can use this filename command option to start an application created with AVS/Express and saved with the Save application command. By default neither the Network Editor nor the VCP appear but this can be altered using the following commands,
    - -ne: requests the network editor
    - -none: suppresses the network editor
    - -vcp: requests the VCP
    - -novcp: suppresses the VCP

9 Appendix III

9.1 Solutions to chapter 3: field file exercises

9.1.1 Field descriptor for Exercise 1

```
# AVS field file
# Image
ndim=2
dim1=512
dim2=256
nspace=2
veclen=1
data=integer
field=uniform
variable 1 file=image.dat filetype=ascii skip=0 offset=0 stride=1
```

9.1.2 Field descriptor for Exercise 2

```
# AVS field file
# Volume
ndim=3
dim1=128
dim2=128
dim3=128
nspace=3
veclen=1
data=integer
field=uniform
variable 1 file=volume.dat filetype=ascii skip=1 offset=0 stride=1
```

9.1.3 Field descriptor for Exercise 3

```
# AVS field file
# Flow
ndim=2
dim1=5
dim2=10
nspace=2
veclen=2
data=float
field=irregular
label=temp press
units=K kg/m**2
variable 1 file=flow.dat filetype=ascii skip=0 offset=0 stride=2
variable 2 file=flow.dat filetype=ascii skip=0 offset=1 stride=2
coord 1 file=coord.dat filetype=ascii skip=0 offset=0 stride=2
coord 2 file=coord.dat filetype=ascii skip=0 offset=1 stride=2
```

9.2 Solutions to chapter 3: 2D data exercises

9.2.1 Field descriptor for Exercise 1: `mri.fld`

```
# AVS field file
# MRI
ndim=2
dim1=512
dim2=512
nspace=2
veclen=1
data=byte
field=uniform
variable 1 file=/usr/express/auxdata/mri.dat filetype=binary skip=0 stride=1
```

9.2.2 Field descriptor for Exercise 2: `temps.fld`

```
# AVS field file
# Temp
ndim=2
dim1=31
dim2=31
nspace=2
veclen=1
data=float
field=uniform
variable 1 file=/usr/express/auxdata/temps.dat filetype=ascii skip=1 stride=1
```

9.3 Solutions to chapter 4: 3D data exercises

9.3.1 Field descriptor for Exercise 1: `fin.fld`

```
# AVS field file
# Bluntfin
ndim = 3
dim1 = 10
dim2 = 8
dim3 = 8
nspace = 3
veclen = 5
data = float
field = irregular
label = density x-momentum y-momentum z-momentum stagnation
variable 1 file=/usr/express/field/med/fin.dat filetype=ascii skip=0 offset=0 -
  stride=8
variable 2 file=/usr/express/field/med/fin.dat filetype=ascii skip=0 offset=1 -
  stride=8
variable 3 file=/usr/express/field/med/fin.dat filetype=ascii skip=0 offset=2 -
  stride=8
variable 4 file=/usr/express/field/med/fin.dat filetype=ascii skip=0 offset=3 -
  stride=8
variable 5 file=/usr/express/data/med/fin.dat filetype=ascii skip=0 offset=4 -
  stride=8
coord 1 file=/usr/express/data/med/fin.dat filetype=ascii skip=0 offset=5 -
  stride=8
coord 2 file=/usr/express/data/med/fin.dat filetype=ascii skip=0 offset=6 -
  stride=8
coord 3 file=/usr/express/data/med/fin.dat filetype=ascii skip=0 offset=7 -
  stride=8
```