THE UNIVERSITY
of MANCHESTER

AVS/Express Visualization Edition

An Introductory Course

Student Workbook

International AVS Centre
Manchester Visualization Centre

September 2003

Document editor

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

The document was compiled using the document publishing software FrameMaker. Additional exercises and improvements to the original text were supplied by Rachel Slinger. The work was undertaken whilst working in the Manchester Visualization Centre as a Summer Student Project 1996.

This document has been updated by J Irwin, H Morphet and P Lever, Manchester Visualization Centre, in October 1998 and Tobias Schiebeck, Manchester Visualization Centre, in September 2003.

The document is published and distributed by the International AVS Centre (IAC).

For further information on IAC see the following World Wide Web page:

```
http://www.iavsc.org
```

or send email to:

```
avs@iavsc.org
```

or by mail to:

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

© September 2003, Manchester Visualization Centre, Manchester Computing. The use of registered names, trademarks etc. in this document does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Introduction 1
The AVS/Express Visualization Edition course 1
About the workbook 2
Using the network editor 5
Aims of the chapter 5
Exercise 1 5
Exercise 2 23
Visualizing 2D arrays of data 25
Aims of the chapter 25
Creating field files 26
Exercises 30
Visualization of 2D data 31
Moving to your home directory 31
Exercise 1 32
Exercise 2 40
Exercise 3 51
Visualizing 3D arrays of data 53
Aims of the chapter 53
Exercise 1 54
Exercise 2 62
Using the software renderer 75
Exercise 3 76
Exercise 4 79
Exercise 5 80
Visualizing UCD data 83
What is UCD Data? 83
Aims of the chapter 84
Exercise 1 84
Further exercises 91
Exercise 2 91
AVS/Express examples 101
Aims of the chapter 101
Exercise 1 101
Other exercises 106
Appendix I 107
Obtaining hardcopy from AVS/Express 107
Customising AVS/Express 108
Journaling in AVS/Express 109
Appendix II 111
Solutions to chapter 3: field file exercises 111
Solutions to chapter 3: 2D data exercises 113
Solutions to chapter 4: 3D data exercises 114

1 Introduction

1.1 The AVS/Express Visualization Edition course

The AVS/Express Visualization Edition course has been 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 5 deals with visualization of Unstructured Cell Data (UCD).
- Finally Chapter 6 deals with the AVS/Express example demonstrations.

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 directory

```
/usr/express/auxdata
```

Copies of these data files can be obtained via the International AVS Centre website:

```
http://www.iavsc.org/training/express/intro/data/intro.tar.gz
```

or

```
http://www.iavsc.org/training/express/intro/data/intro.zip
```

The answers to the exercises in Chapter 3 and Chapter 4 can be found in Appendix II of this workbook.

2 Using the network editor

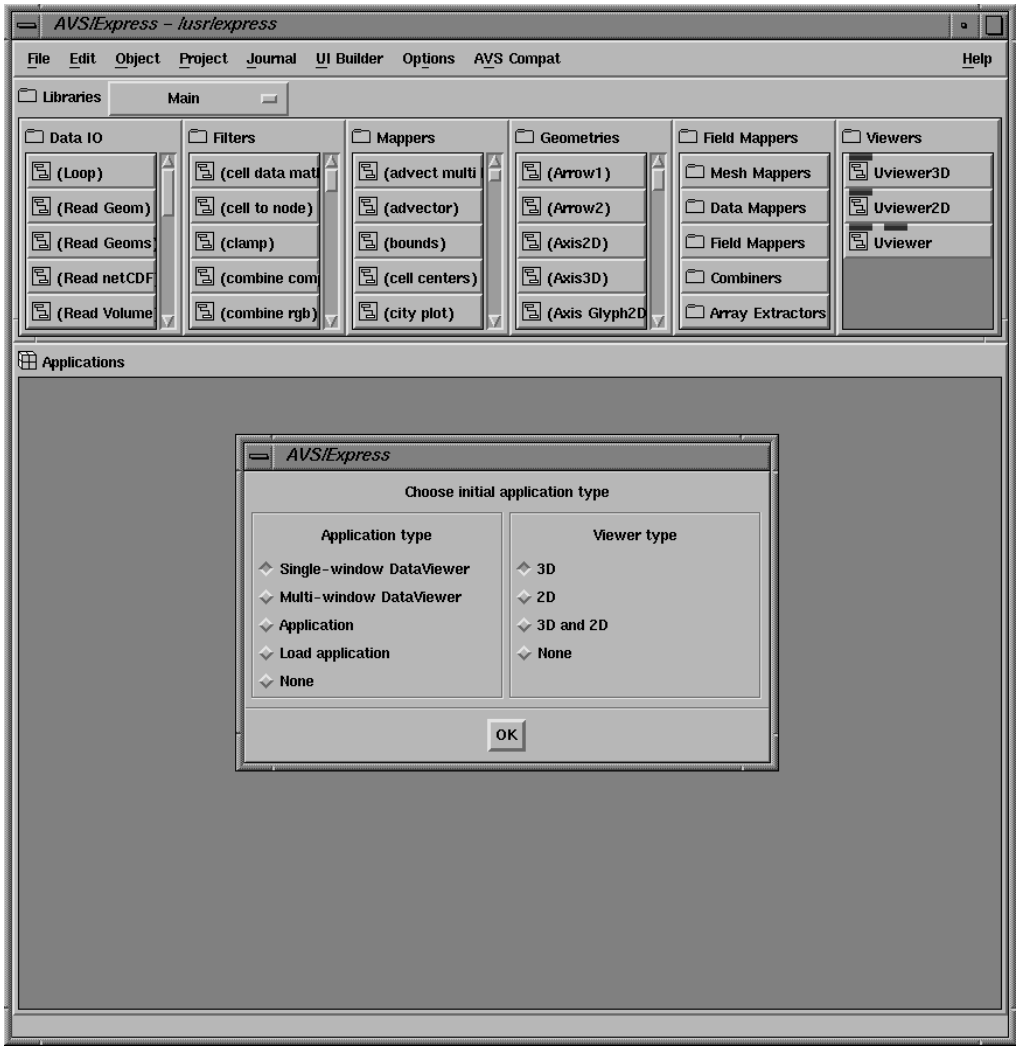
2.1 Aims of the chapter

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

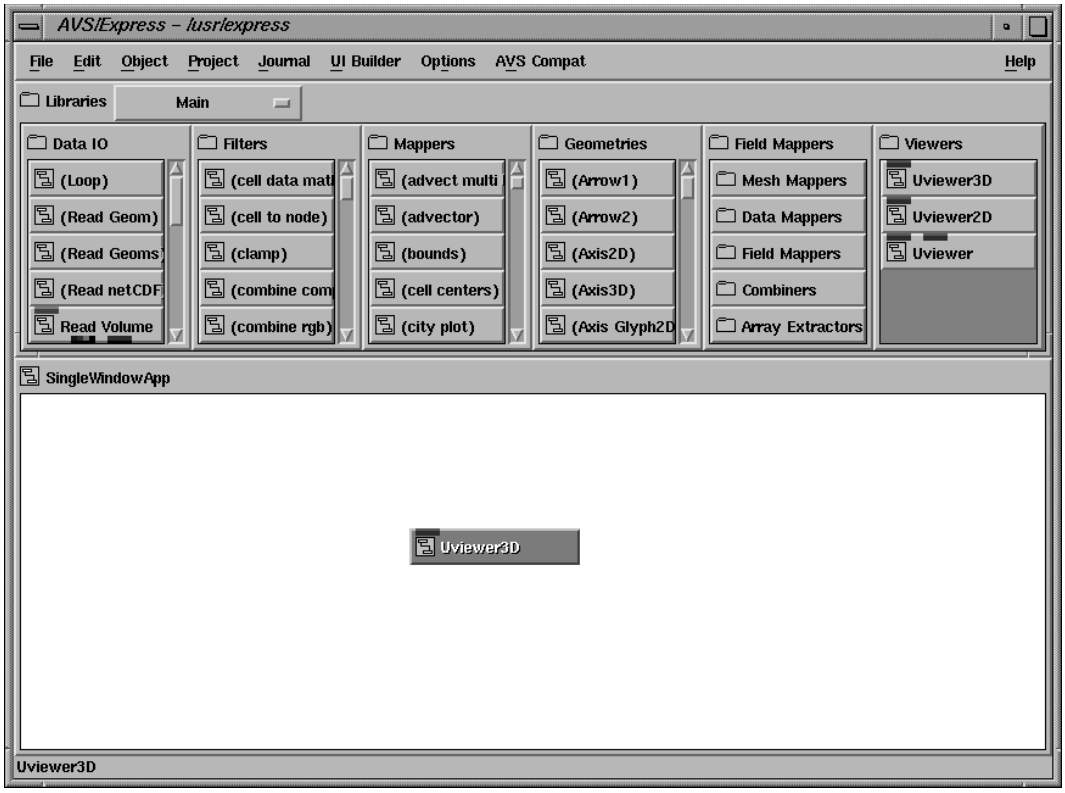
In this example we wish to visualize the electron density within a hydrogen atom. In particular we want to take a slice through the atom and observe differing electron densities throughout the atom.

2.2.1 Starting the AVS/Express Visualization Edition



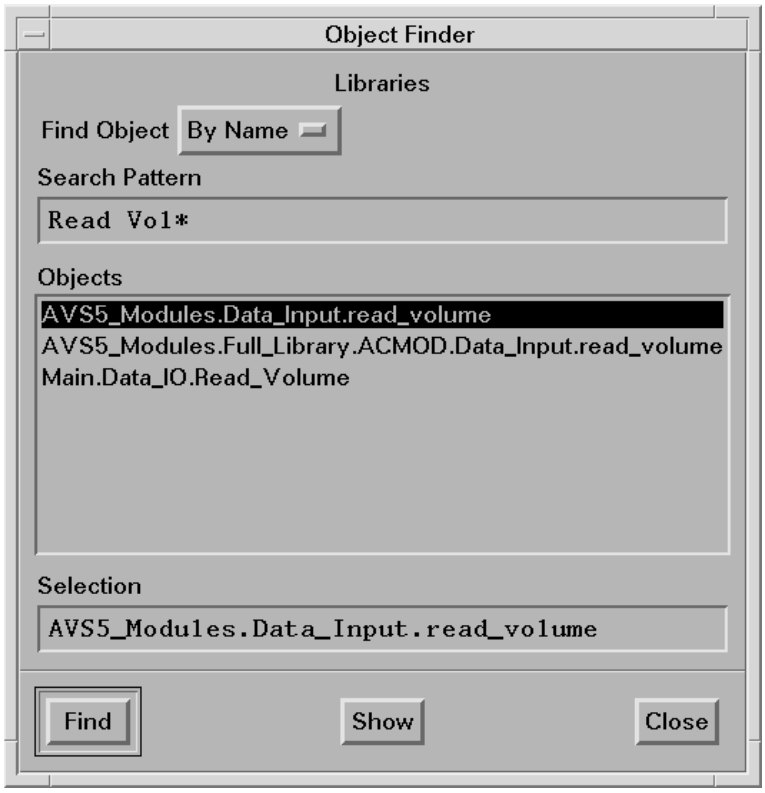
- To start AVS/Express Visualization Edition, simply type the following command:
  - `vxp`
- AVS/Express displays the network editor and a dialogue box in which you choose the initial type of application and viewer you want to use.
- Choose `Single-window DataViewer` with a `3D` viewer type (usually the best choice when constructing simple applications).
- Click on **OK** to proceed.

2.2.2 Structure of the network editor



- The screen should now show the above layout, which shows the major components of the network editor.
- The library of modules on view, above the application workspace area, 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.

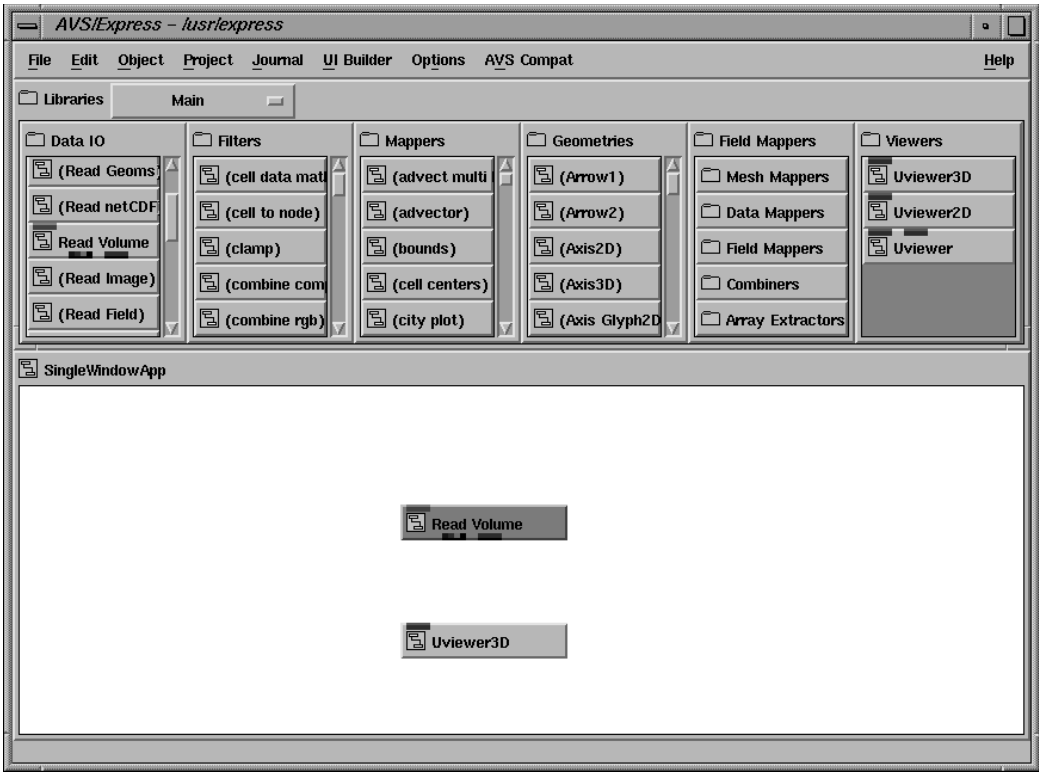
2.2.3 Using the object finder



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.

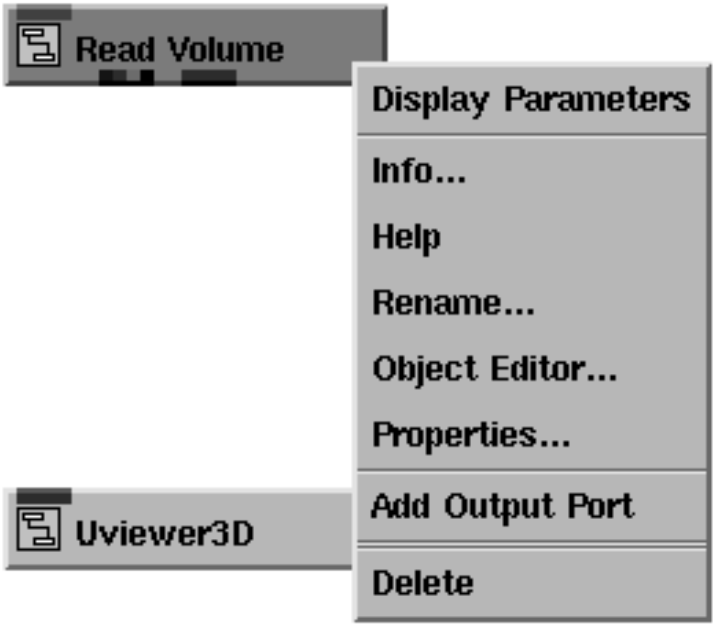
- *To search through all the AVS/Express libraries*: choose, **Find in All Libraries** from the **Object** menu, (located in the menu bar along the top of the network editor), and type in the name of the module in the **Search Pattern** box. Click **Find** and the location of the module will be given 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. For example the above diagram shows a search for any modules which read volume data - type "`Read Vol*`**"** in the **Search Pattern** field**.**

2.2.4 Selecting a module



- Find the `Read_Volume` 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.

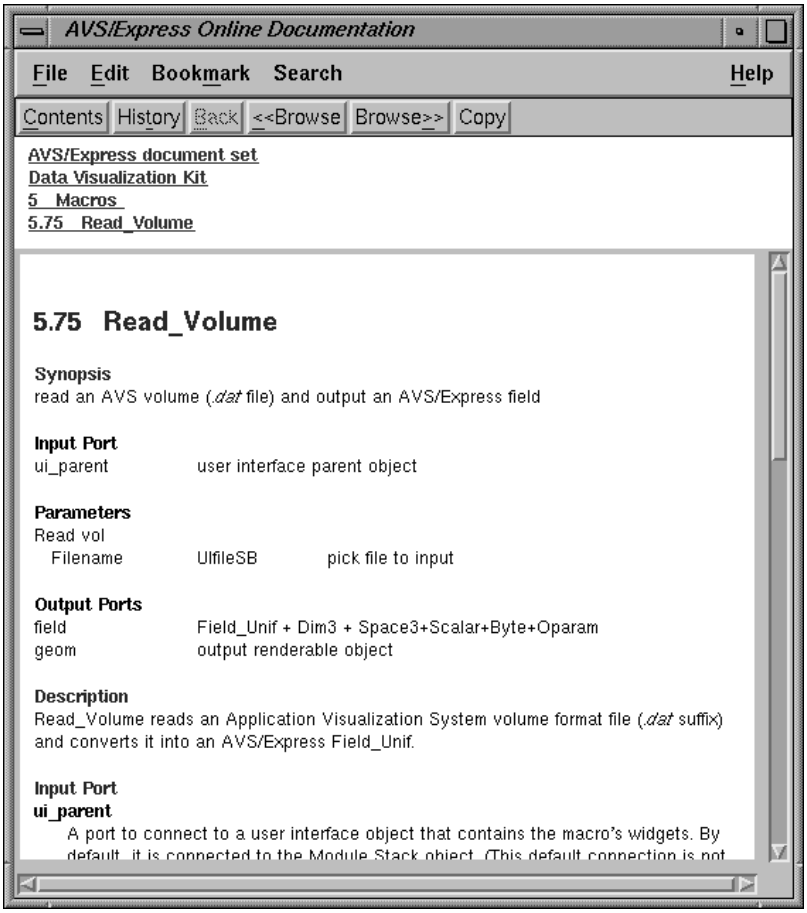
2.2.5 Deleting a module



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.

2.2.6 Accessing the help pages



To access the help pages for a particular module:

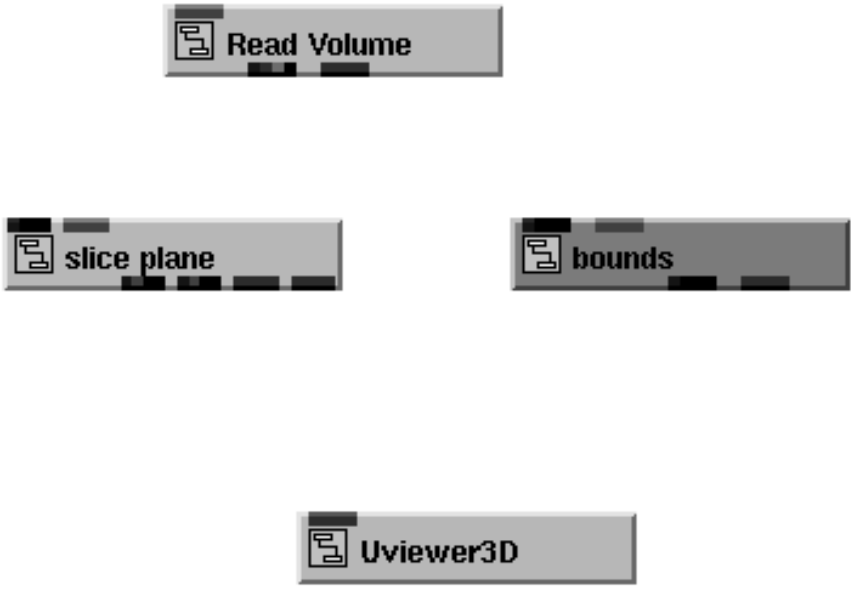
- Select the module and hold down the **right** mouse button to display the popup menu
- Choose the **Help** command.

Alternatively:

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

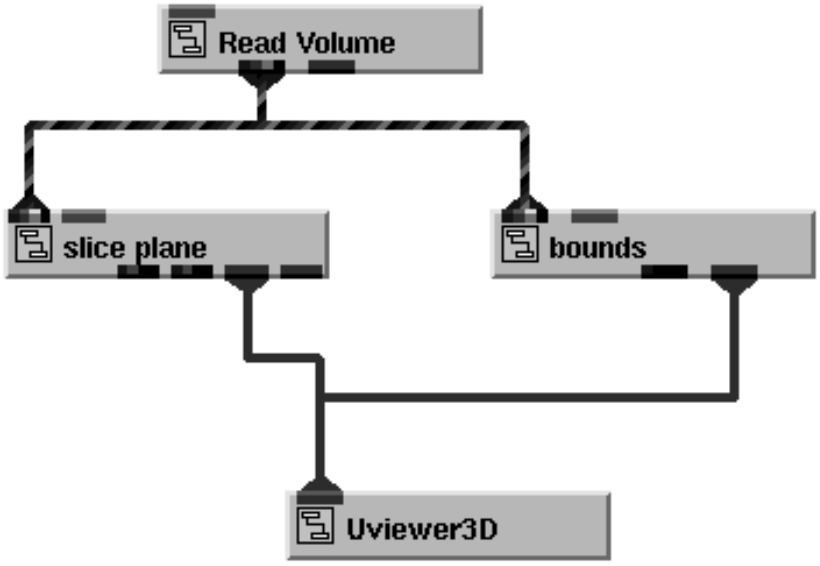
Both these routes will invoke the appropriate help page to appear.

2.2.7 Selecting the other modules



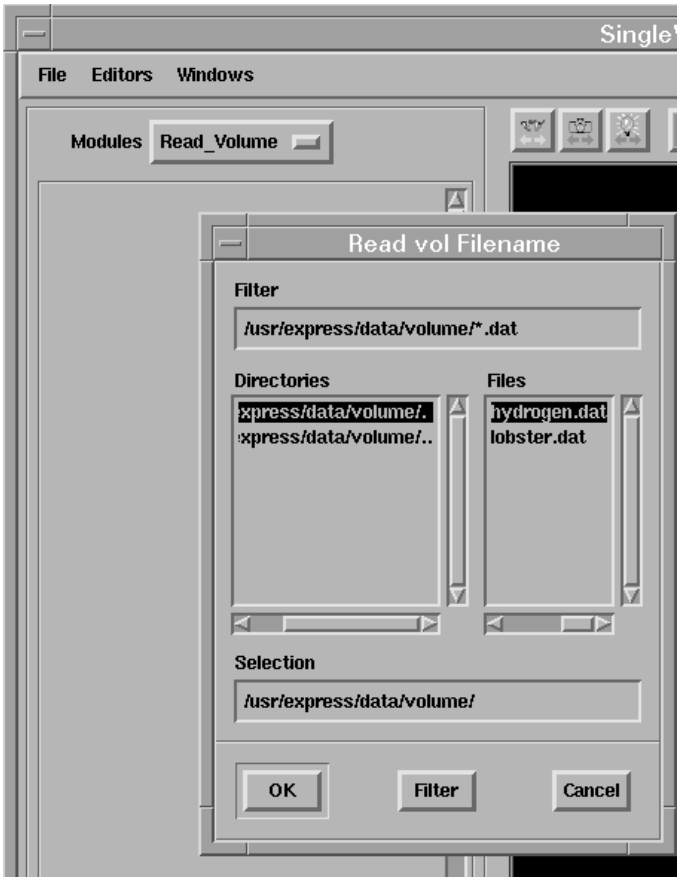
- Select the modules in the list below from the given sub-libraries and arrange them in the workspace as shown above.
    - `Read_Volume` (Data IO): reads a volume format file and converts it to a 3D AVS/Express field.
    - `bounds` (Mappers): generates a bounding box of a 3D field.
    - `slice_plane` (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.**

2.2.8 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.

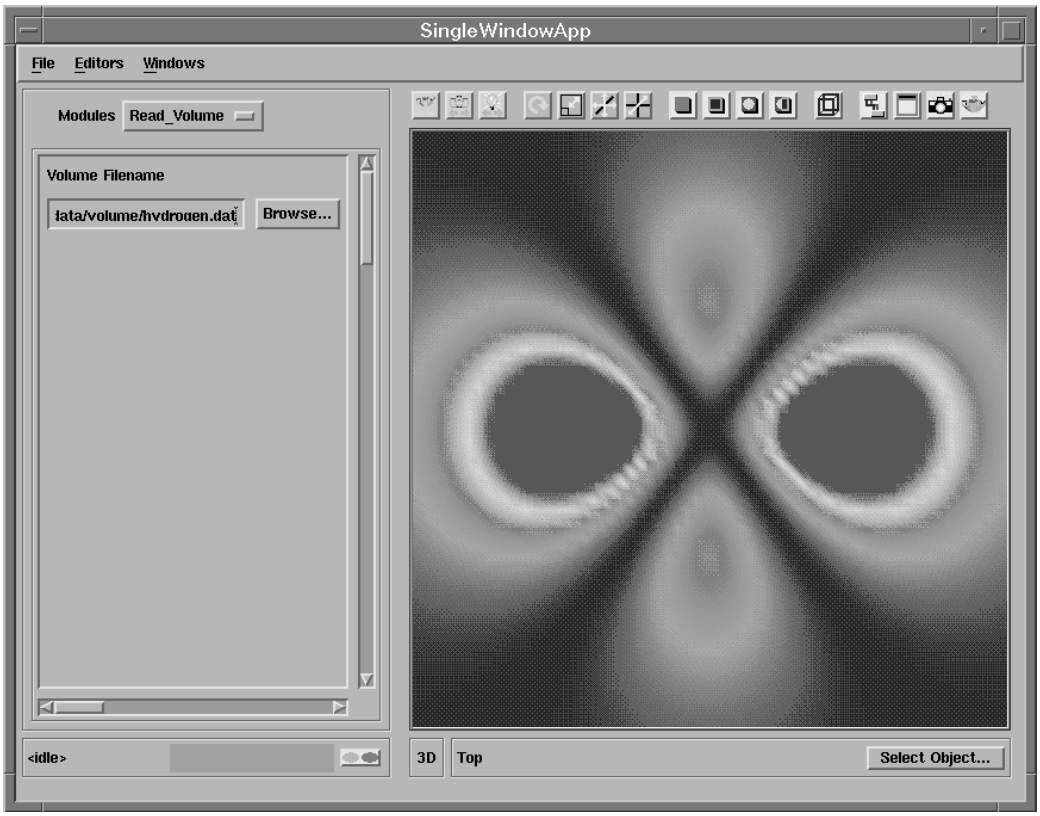
2.2.9 Reading a data file (1)



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_Volume` module and press the browse button. The file browser should now appear.
- Use the file browser to change to the directory `/usr/express/data/volume` and click on the file called `hydrogen.dat`.
- The network will execute producing a picture in the viewer window located in the **DataViewer Pad.**

2.2.10 Reading a data file (2)

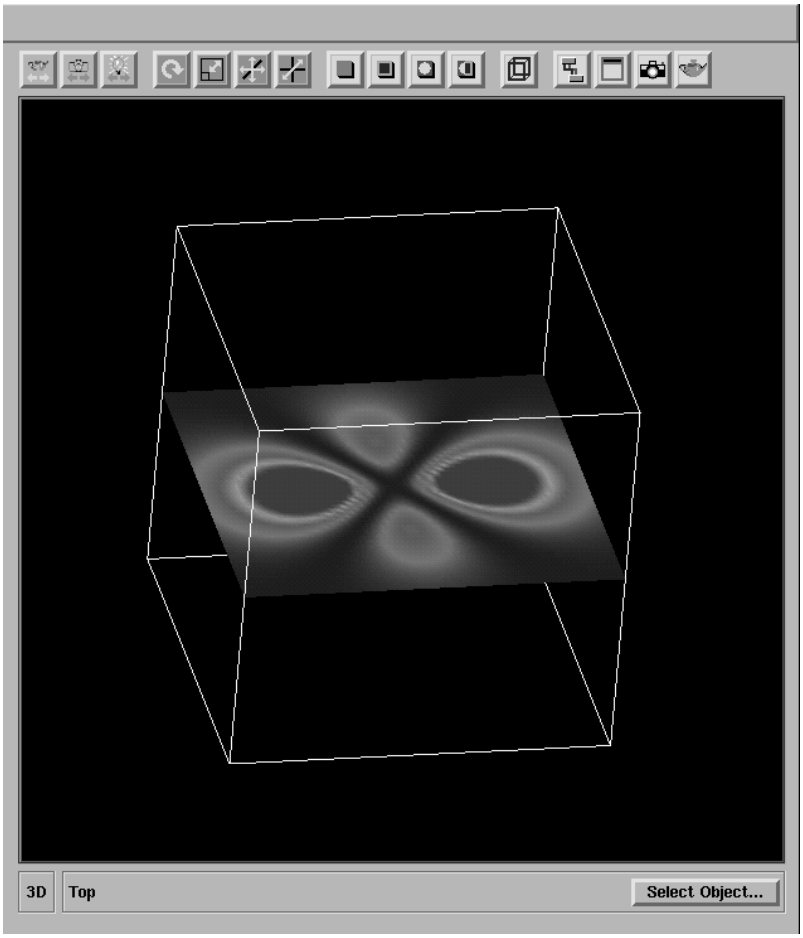


- The **DataViewer** window should display the above picture.
- Now that an image has been obtained,we want to manipulate the picture to the desired format.

2.2.11 Interacting with the objects (1)

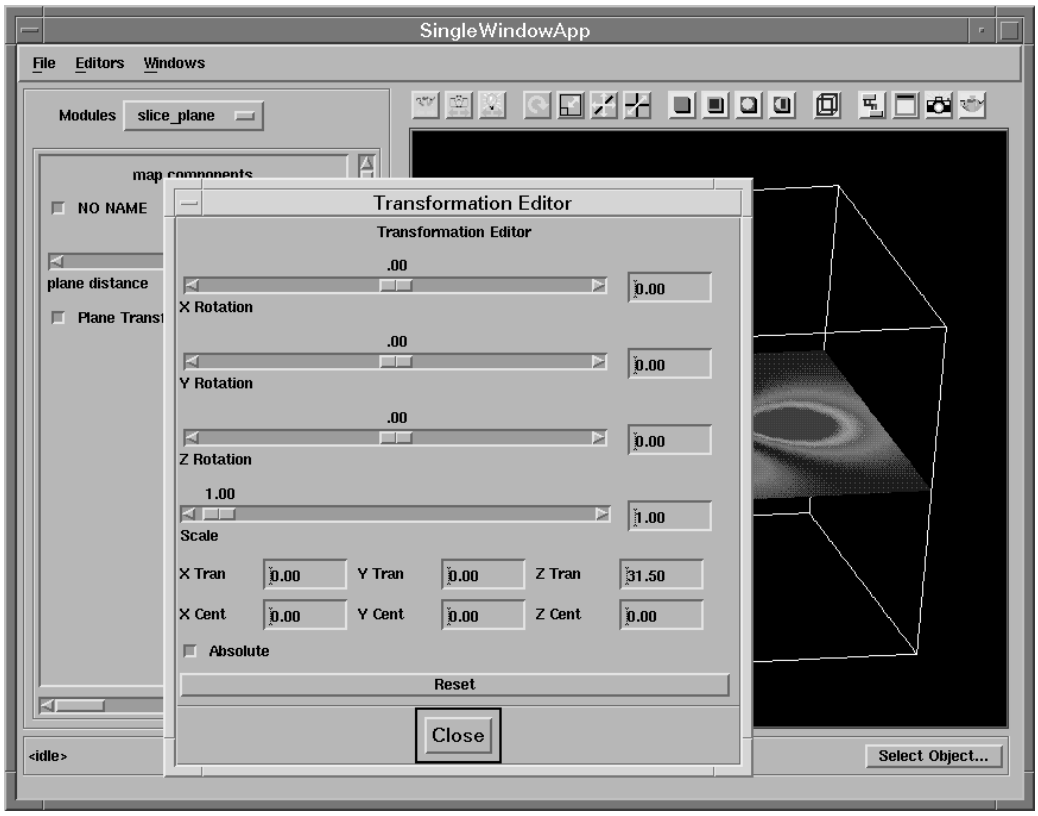
- The Toolbar, located in the **DataViewer** interface, provides a quick way to open **DataViewer** components and to manipulate images in the **DataViewer** window.
- Move the mouse over these icons to reveal a small label indicating the action the icon provides a short-cut to, e.g.
    - Transform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

2.2.12 Interacting with the objects (2)



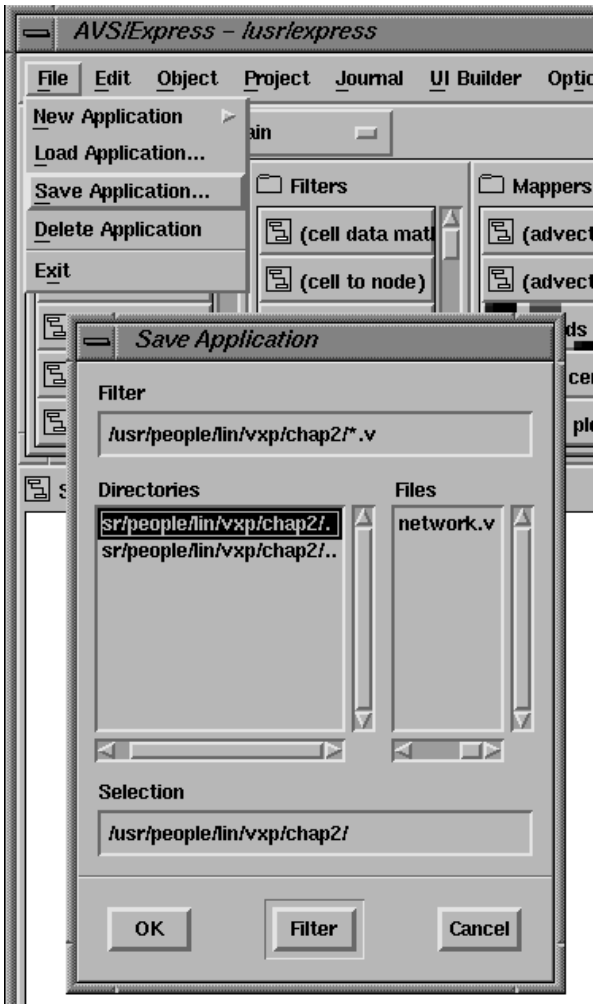
- Click on the **Reset/Normalize/Center** icon, and the full picture should appear in the viewer window.
- Now you can perform a few simple transformations on the image in the **DataViewer** window:
  - **scaling**: click on the `Scale` icon in the Toolbar, place the cursor in the window and, while holding 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.

2.2.13 Changing the `slice_plane` parameters



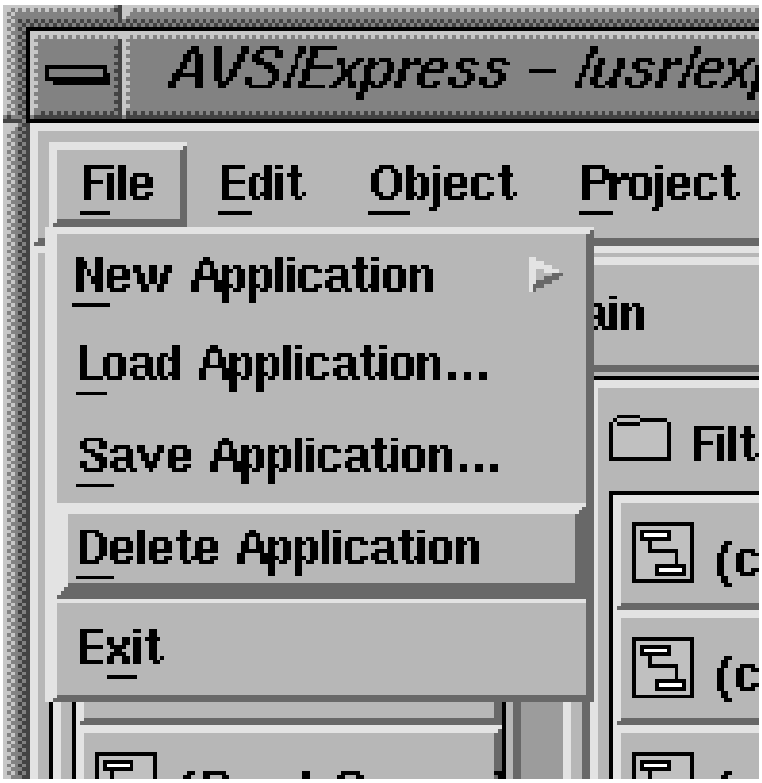
- Select the **Modules** command from the **Editors** pull-down menu, (or the Toolbar), and choose the `slice_plane` module. You can now try altering some of the parameters of the slice through the atom.
- You can alter the position of the slice through the atom by changing the value on the `plane distance` 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.

2.2.14 Saving applications



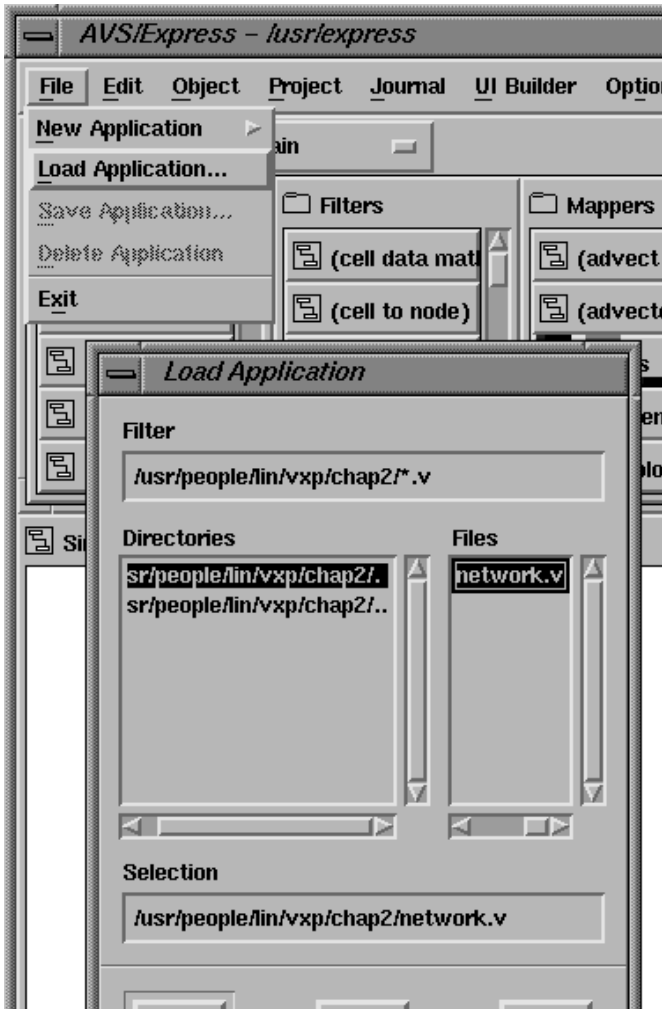
- The network can be saved as an application by selecting the **Save Application...** command from the **File** pull-down menu.
- A file browser will appear showing the files in your current working directory. Use the file browser to change the current directory to the one you want to save the file in. (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"`.

2.2.15 Deleting an application



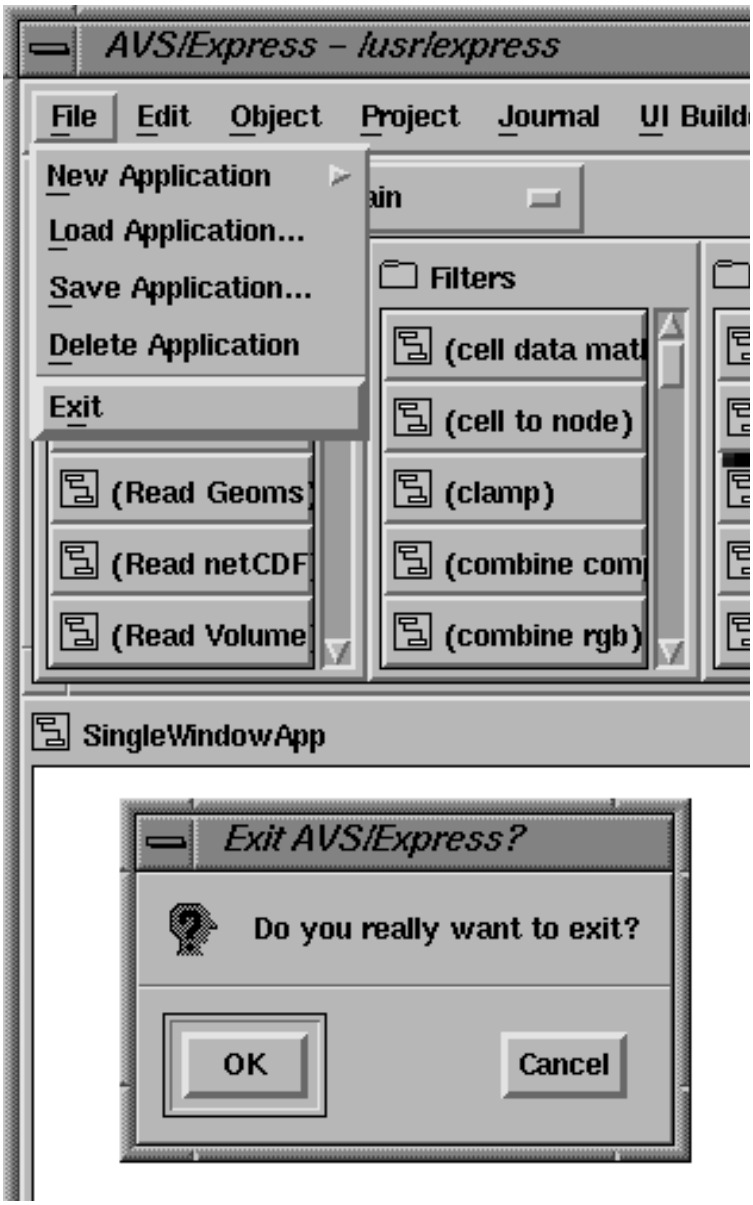
- To clear an existing application network from the workspace select the **Delete Application** command from the **File** pull-down menu.

2.2.16 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.

2.2.17 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.

- Use the same network as the one constructed in Exercise 1.
- When using the volume browser select the file `lobster.dat` and click **OK**.
- Using similar manipulation techniques as those described in Exercise 1, investigate the bone density of the lobster within its exoskeleton.
- 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 II . Secondly, for you to work through several structured exercises which cover visualization techniques for 2D data: creating field files, constructing networks and manipulating images. The solutions to the field files in the 2D data examples are also given in Appendix II .

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. The solutions are given in Appendix II .

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`. 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.

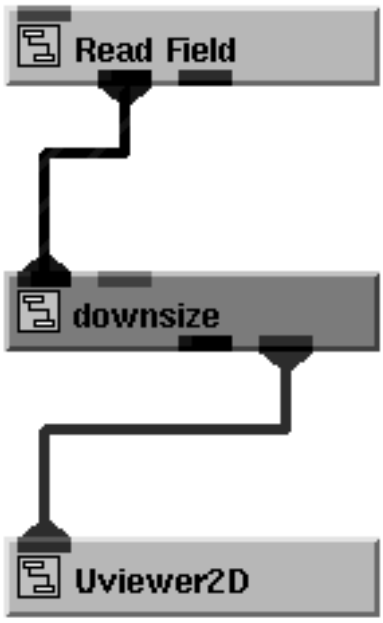
3.6.1 Importing the `mri.dat` data (1)

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.
```

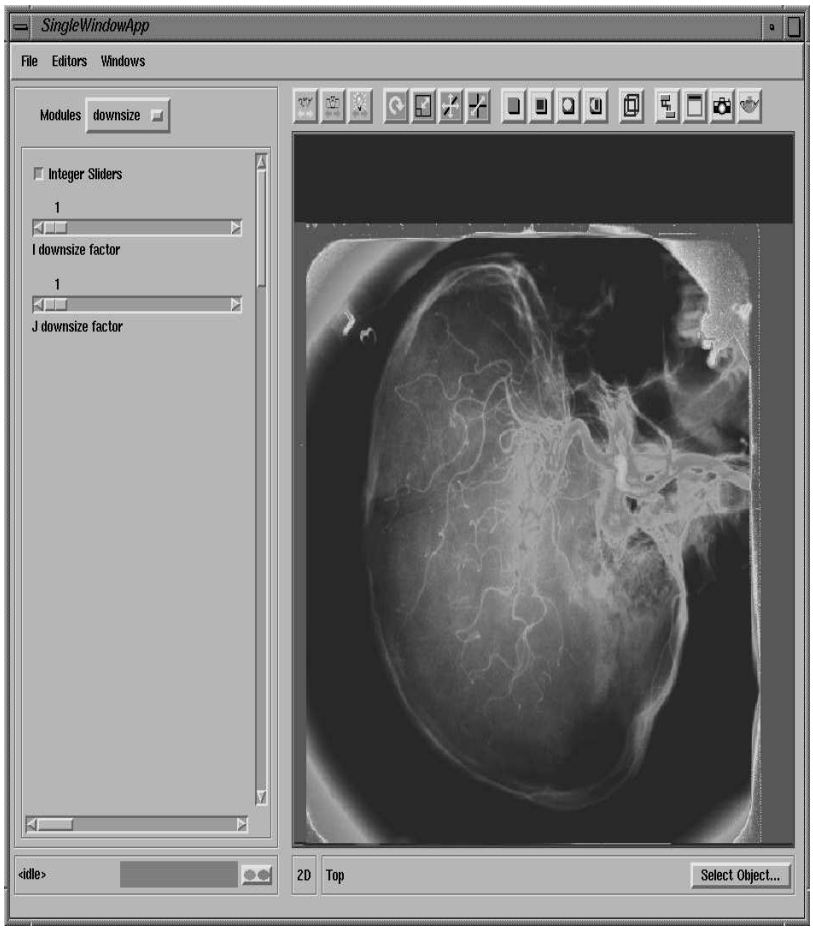
- Create a file called `mri.fld` to describe the data contained in `mri.dat`.
- Use the template field file given on page 26 and the information given above to complete the specification.

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.)

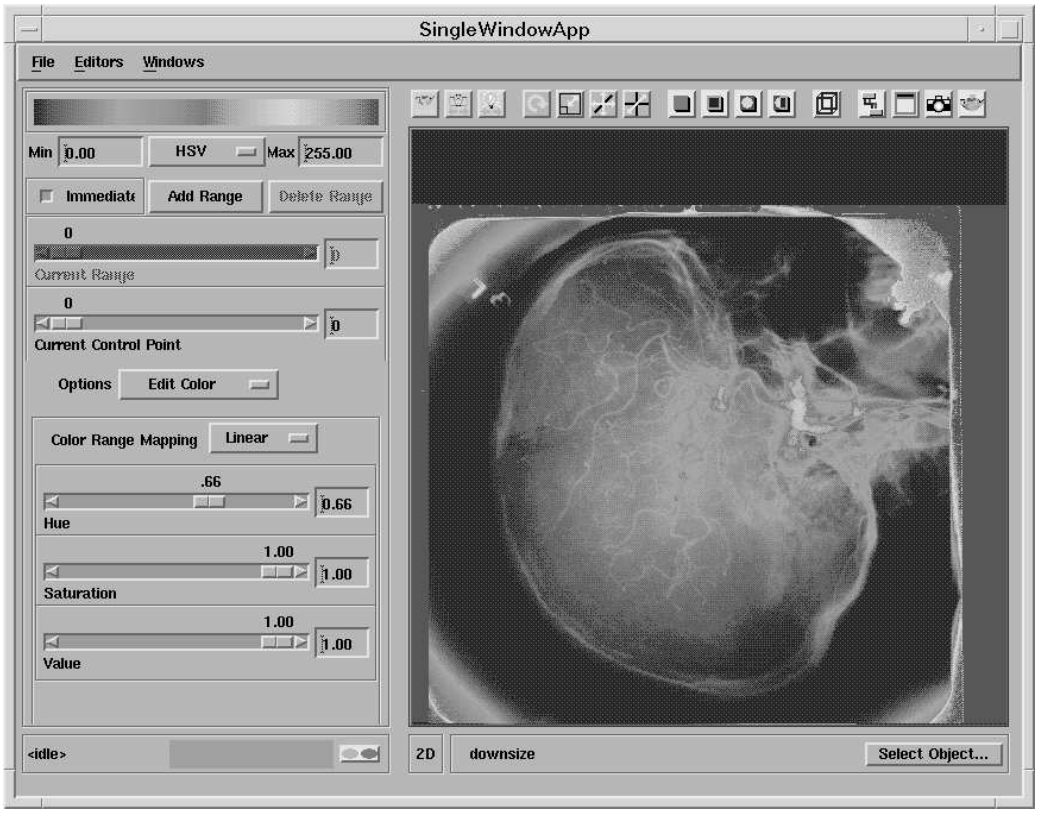
3.6.5 Changing the colour map (1)

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 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:

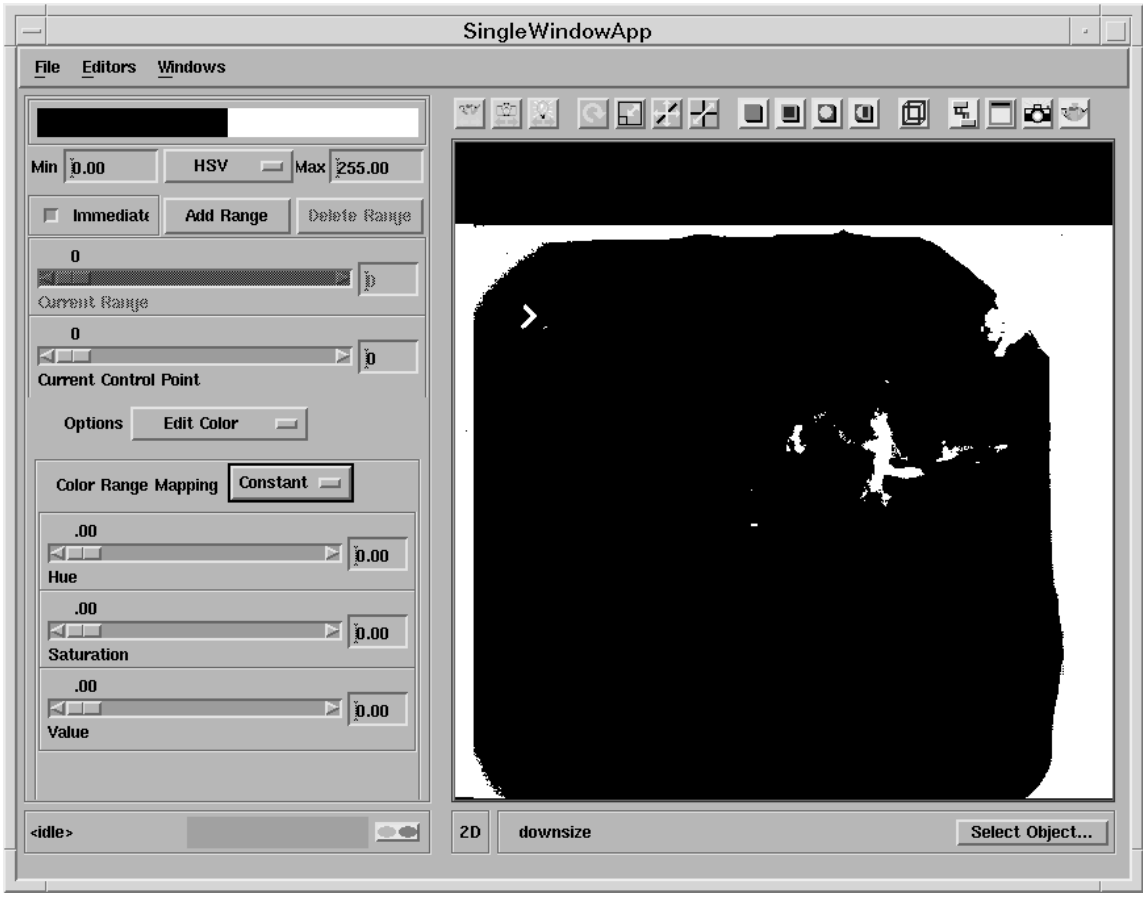
- There are two colour models which may be used
  - **HSV** (**H**ue **S**aturation **V**alue)
  - **RGB** (**R**ed **G**reen **B**lue)
- The **colour range maps** can either be **linear** (interpolated between max and min values of data) or **stepped** (constant colour for each step). The mapping is 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.

3.7 Exercise 2

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

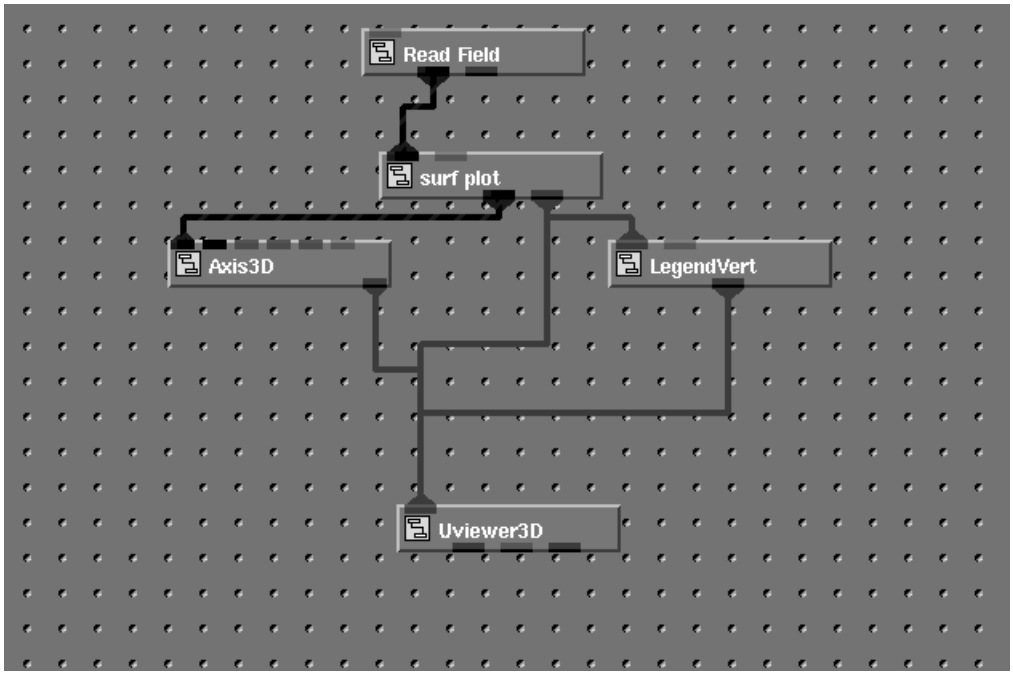
3.7.1 Importing the `temps.dat` data (1)

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.

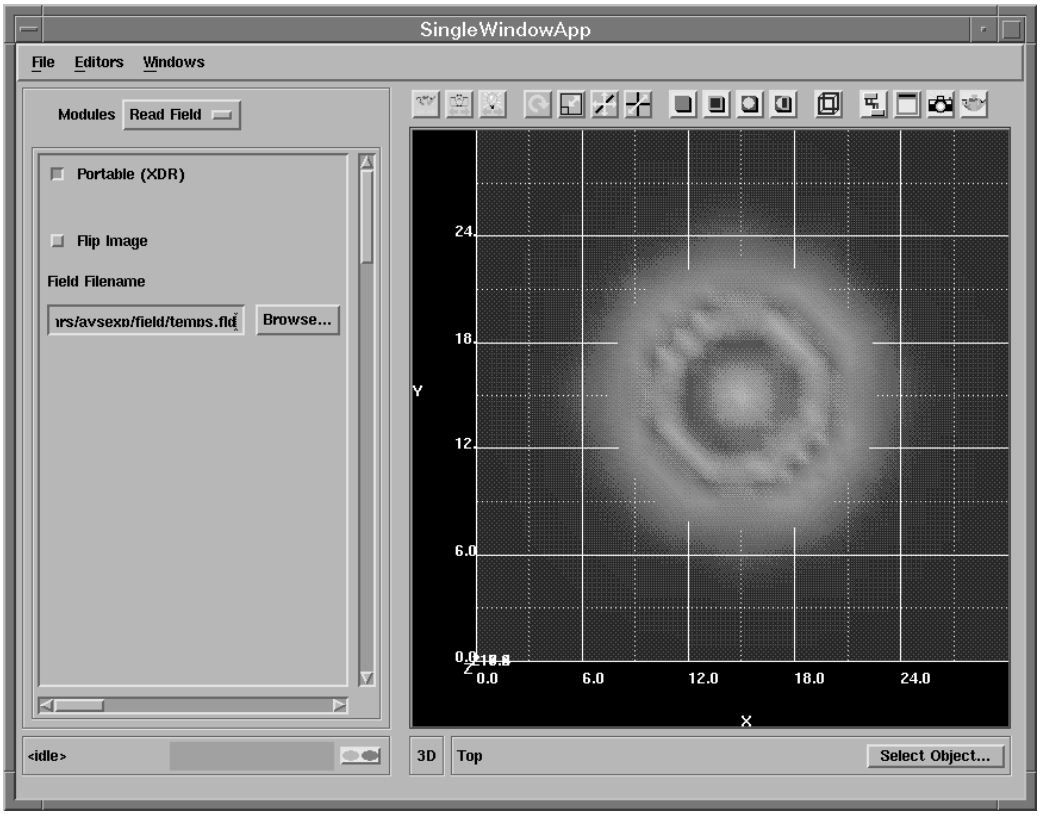
3.7.2 Displaying the temperature data with `surf_plot`



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

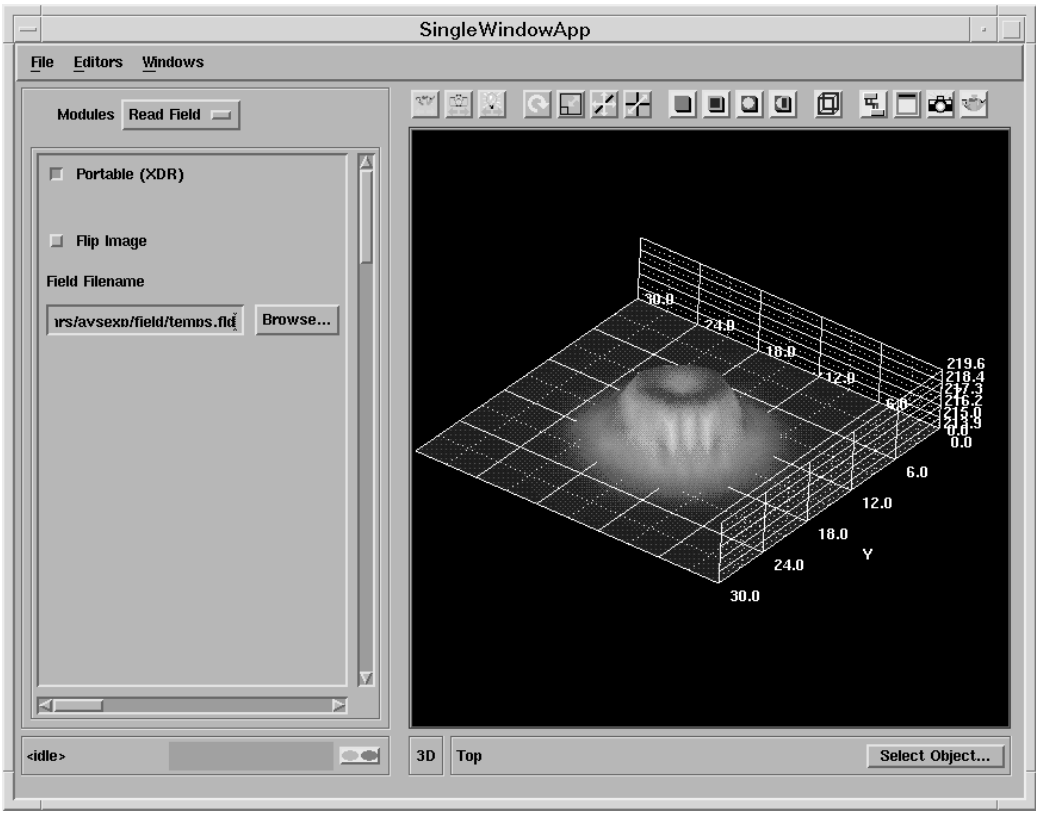
- `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
- `LegendVert` (Geometries): produces a color legend which displays the object's datamap.
- `surf_plot` (Mappers): creates a 2D or 3D mesh whose height is proportional to the scalar data value at each point. 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)



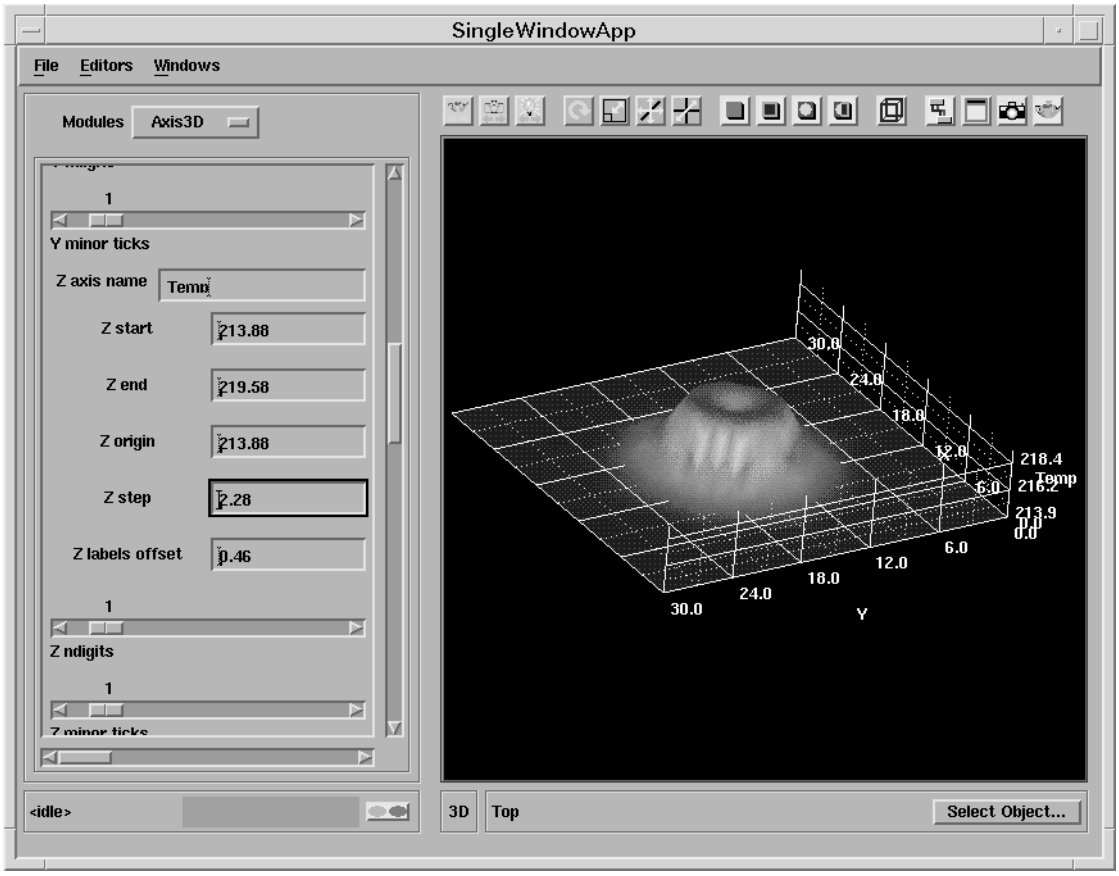
- Select the `Read_Field` module from the **Modules** editor panel and press the browse button. The file browser should now appear.
- To import the data into the network, select the file `temps.fld` you have just created using the file browser.
- If a picture similar to the one above appears then you 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



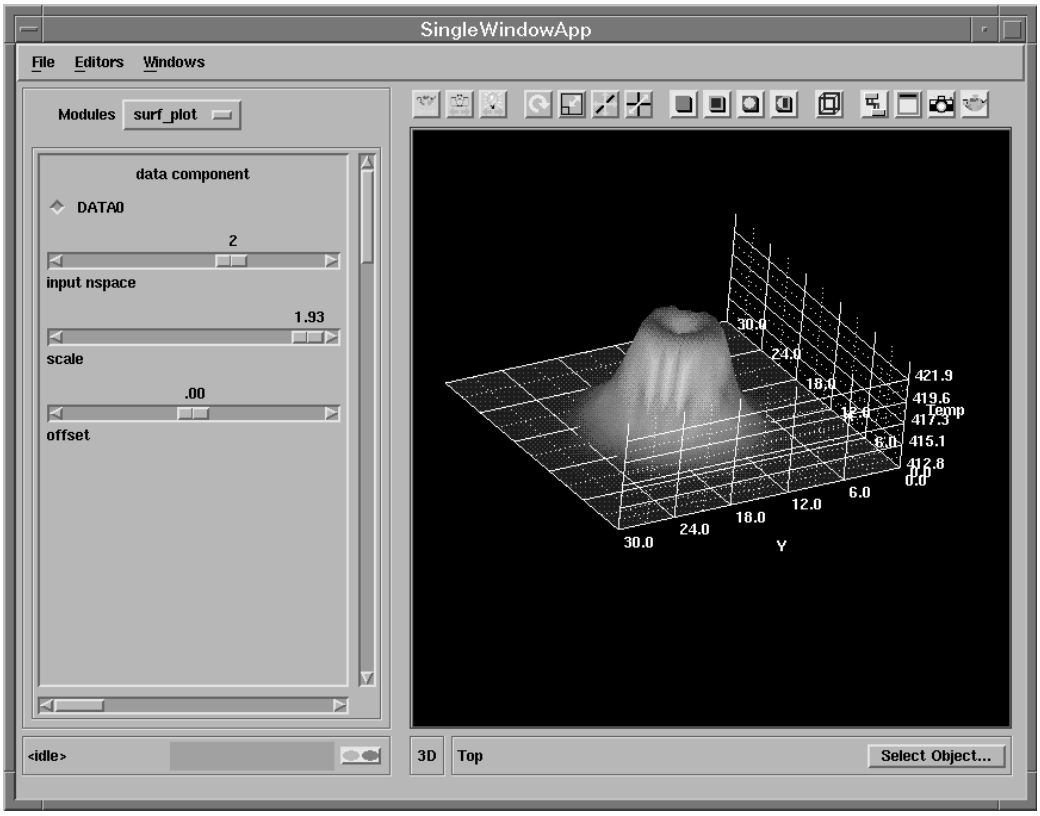
- Click on the **Reset/Normalize/Center** icon in the Toolbar so you can fit the whole object in the window.
- Now using the mouse, transform the object so it appears as above.

3.7.5 Altering `LegendVert` and `Axes3D`



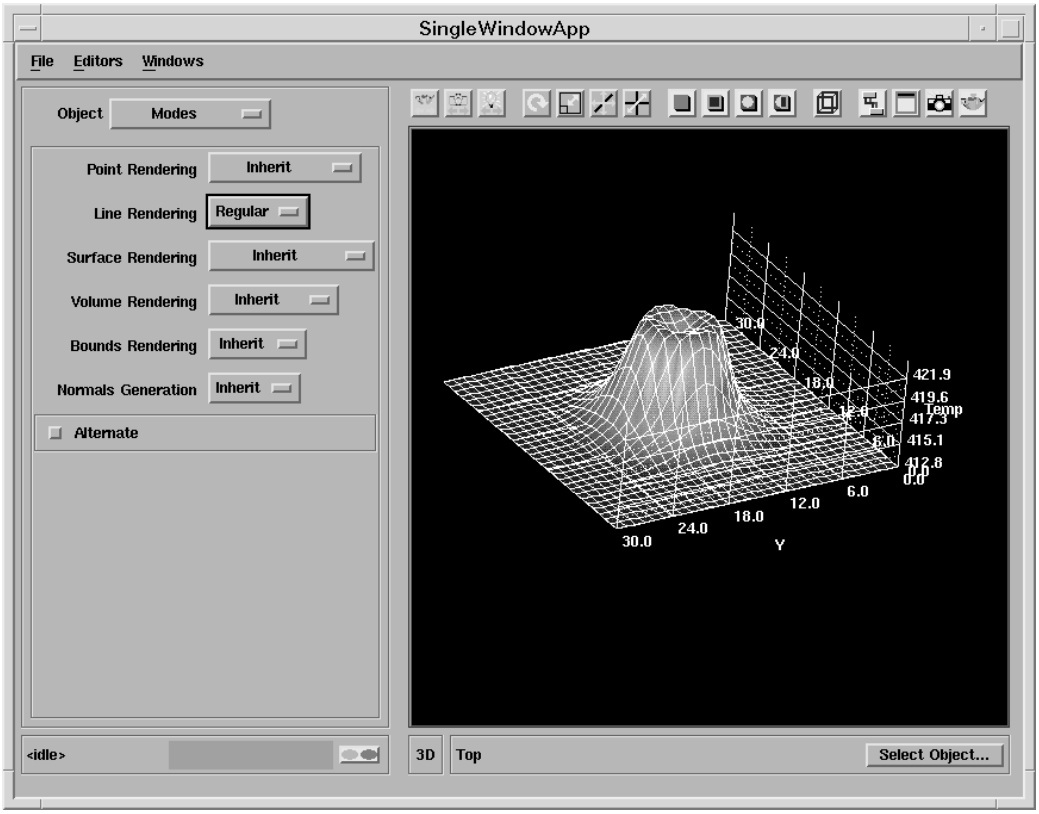
- Select the `LegendVert` module from the **Modules** editor panel
- Change the parameters of the colour bar or legend to make it look optimal in the picture.
- Similarly, select the `Axes3D` module from the **Modules** editor panel:
    - Reduce the number of steps on the z-axis.
    - Label the z-axis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

3.7.6 Using `surf_plot`



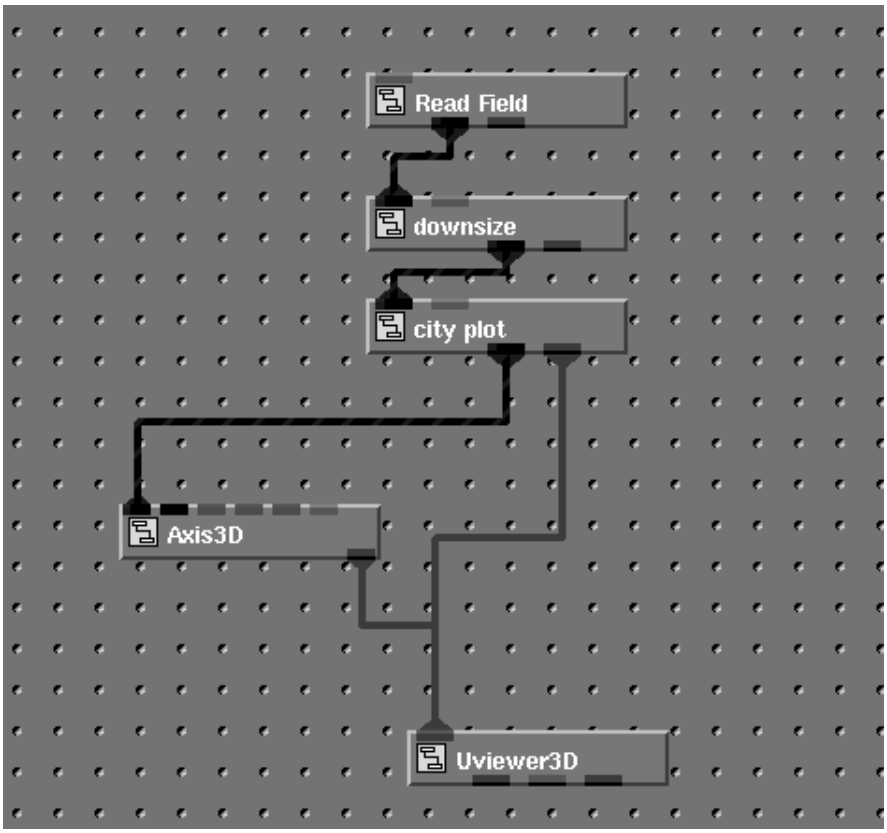
- Select the `surf_plot` module from the **Modules** editor panel
- Alter the slider to increase the scaling.
- The module should now produce a picture as shown above.

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.

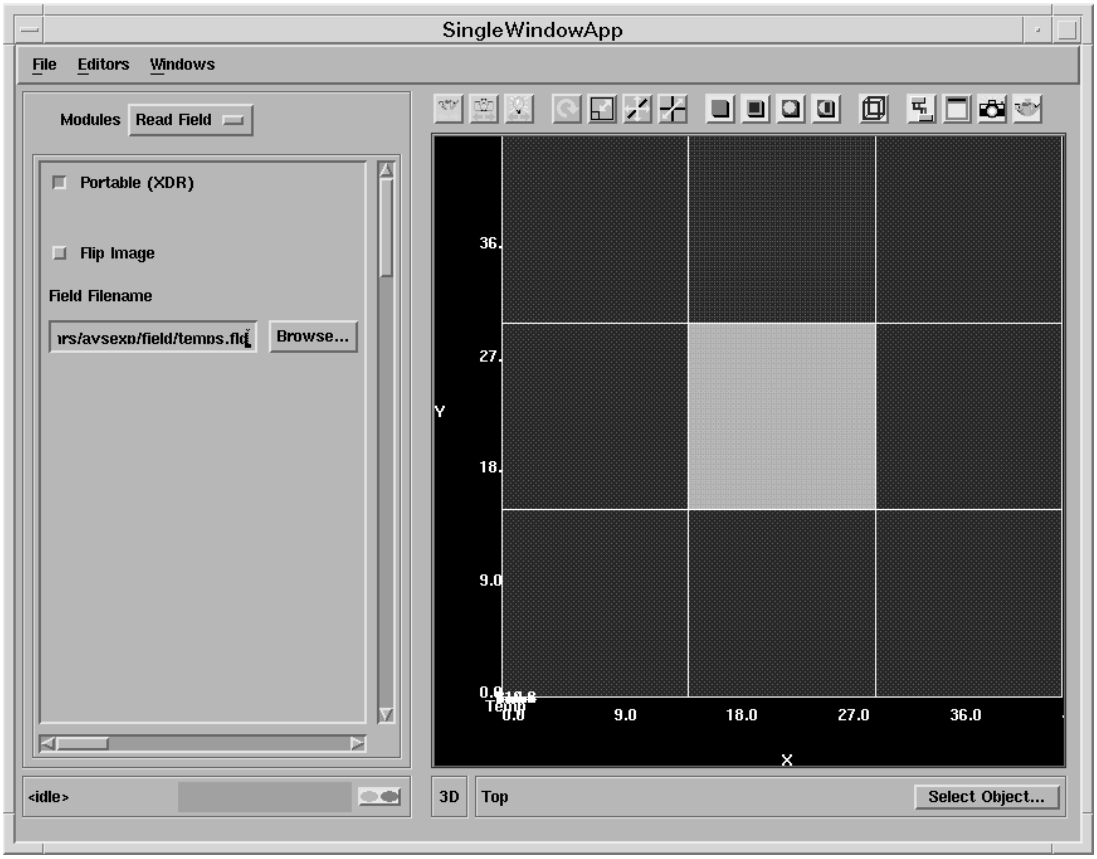
3.7.8 Displaying the temperature data with `city_plot`



Using the network editor construct the above network. The location of the modules are listed below:

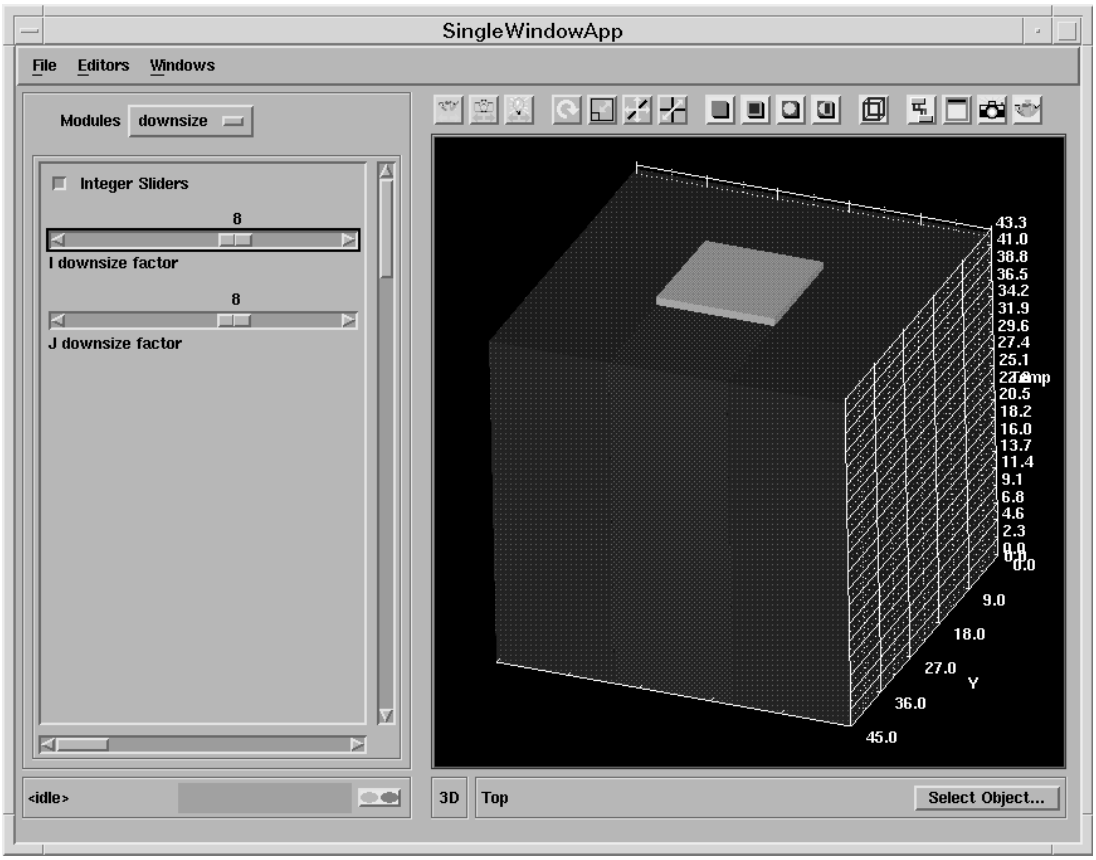
- `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
- `downsize` (Filters): resamples a field using a scaling factor. Usually the factor is greater than 1 and the size of the field is reduced to save processing time and memory.
- `city_plot` (Mappers): creates a plot of blocks whose height is based on the value of a 2D field.
- `Axes3D` (Geometries): creates an XYZ axis grid with labels based on mesh extents.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

3.7.9 Importing the `temps.dat` data



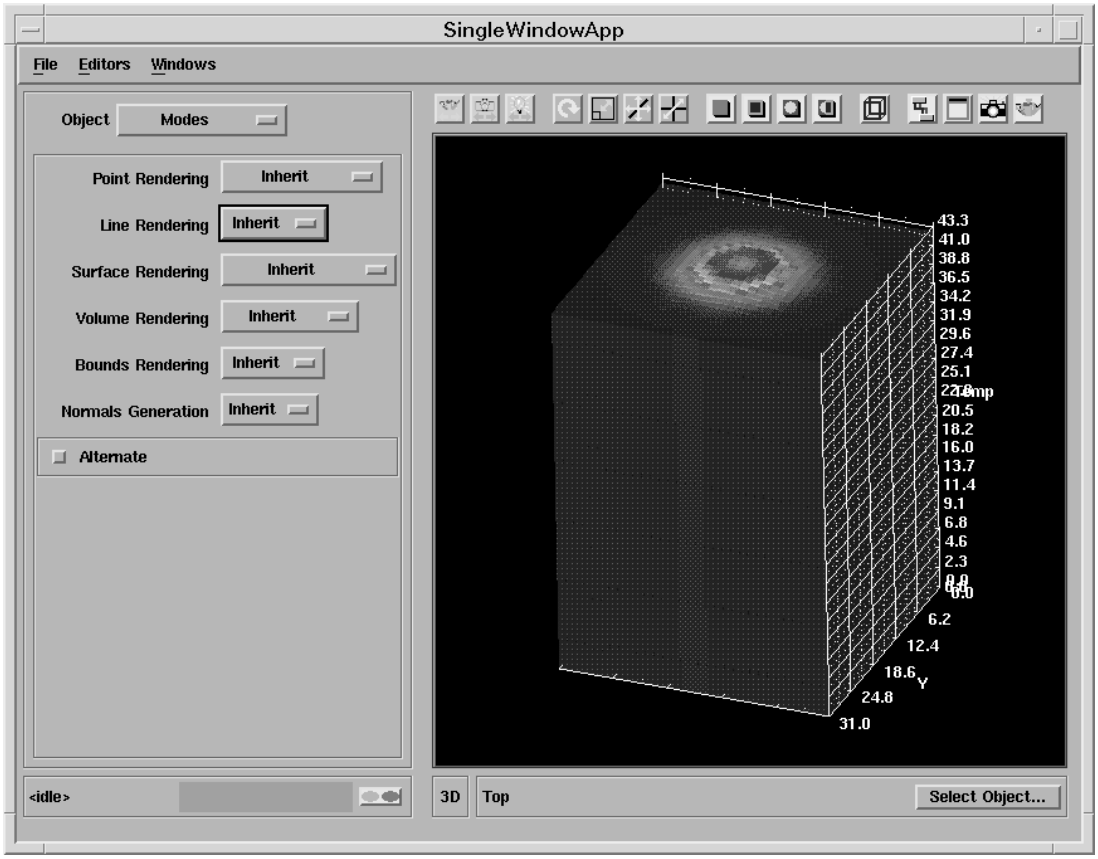
- Select the `Read_Field` module from the **Modules** editor panel and press the browse. The file browser should now appear.
- To import the data into the network, select the file `temps.fld` you have just created using the file browser.
- If a picture similar to the one above appears then you have been successful.

3.7.10 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.
- To reduce the bar chart's length select the `city_plot` module from the **Modules** editor panel. Change the height scale value on the corresponding slider.
- Alter some of the other scale values as well to see what happens.

3.7.11 Changing the downsizing



- The image displayed has been produced by a subset of the data due to the `downsize` module function.
- Select the `downsize` module from the **Modules** editor panel.
- Try altering the values of the I & J downsize factors. (Again you will find that the best resolution of the image is obtained if both values are set to 1.)
- Click on the **Perspective** toggle icon from the Toolbar to observe the effect.

3.8 Exercise 3

3.8.1 Importing the `hslice.dat` data

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

```
This data file represents a single slice through a hydrogen atom where the data
values represent the electron density. The dataset is 2D and contains 64 x 64
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 `hslice.fld` to describe the data contained in `hslice.dat` (rather than re-type the field format edit an already existing field file 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.
- Construct a network as in Exercise 2 using `surf_plot`.
- Select the `Read_Field` module from the **Modules** editor panel and press the browse button. The file browser should now appear.
- Import the data into the network and investigate the possible alterations to the image, given in Exercise 2.

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 II .

4.2 Exercise 1

The aim of this exercise is to visualize a downsized version of the AVS/Express sample dataset `fin.fld` which shows the flow of air over a blunt fin.

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 data file format is as follows:

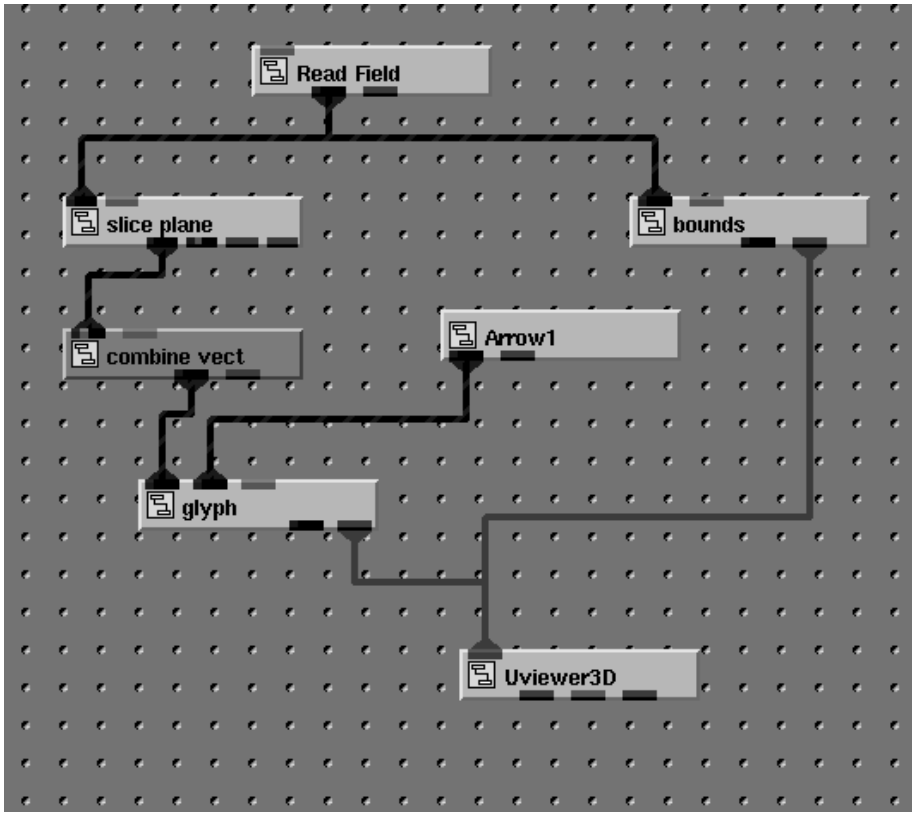
```
density x-momentum y-momentum z-momentum stagnation x-coord y-coord z-coord
```

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 26 and the information given above to complete the specification.

4.2.2 Displaying the vector data



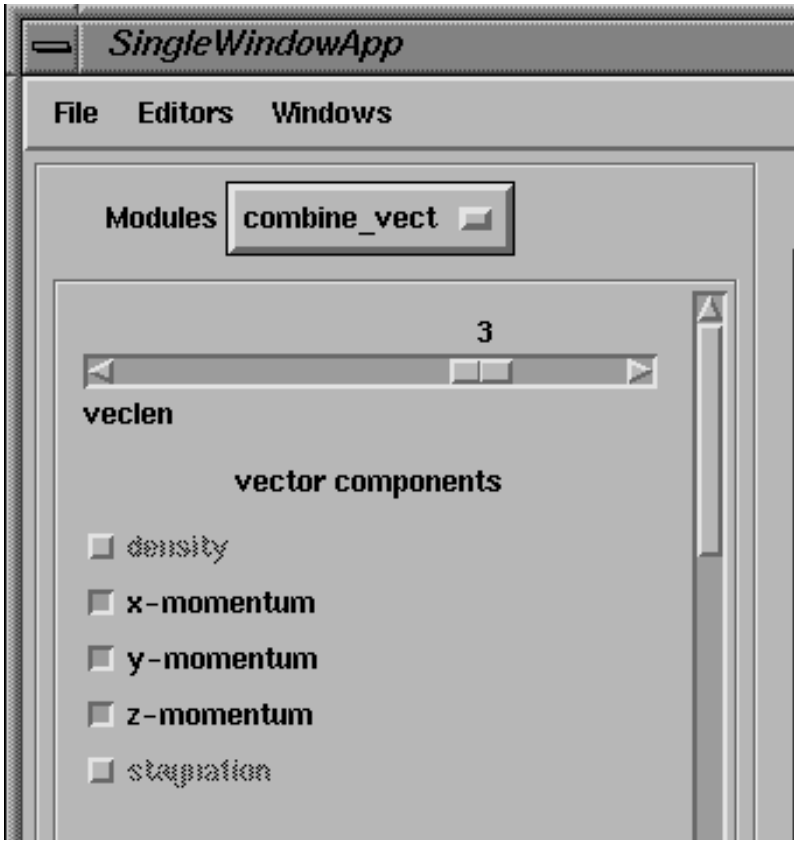
Create the above network using the following modules:

- `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

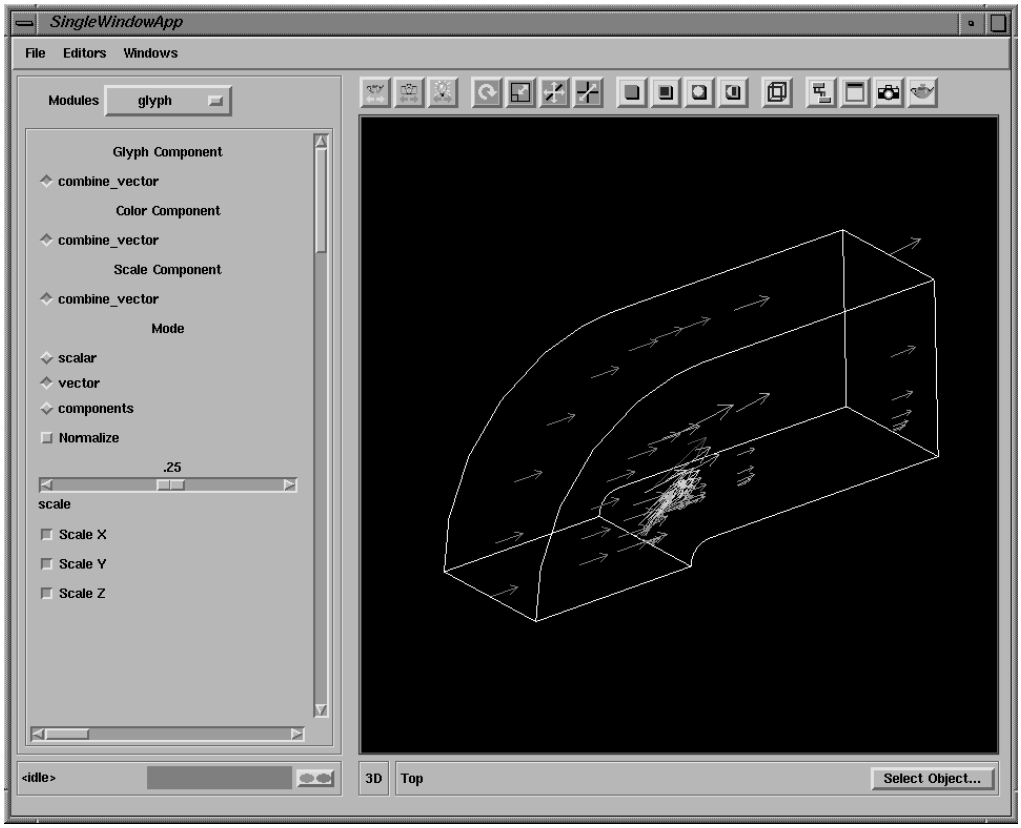
- Select the `Read_Field` module from the **Modules** editor panel and then select the file `fin.fld` which you have just created to import the data using the file browser.
- Select the control panel for the `slice_plane` module and make active all data components, as shown above. For now ignore any 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



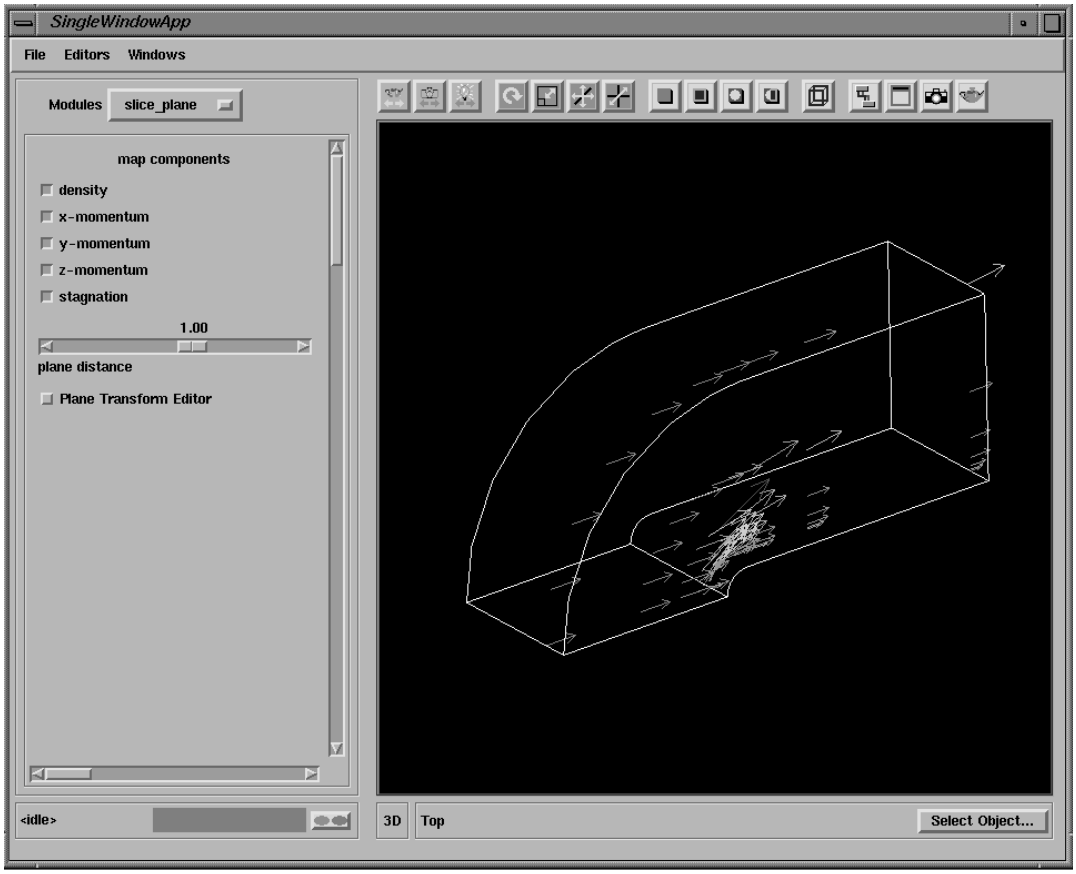
- Select the control panel for the `combine_vect` module and make active the three data components x-momentum, y-momentum and z-momentum, as shown above. Since `veclen` is set to 3 it is only possible to have 3 vector components active at a time.
- This module will now pass on a 3D field with a 3-vector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



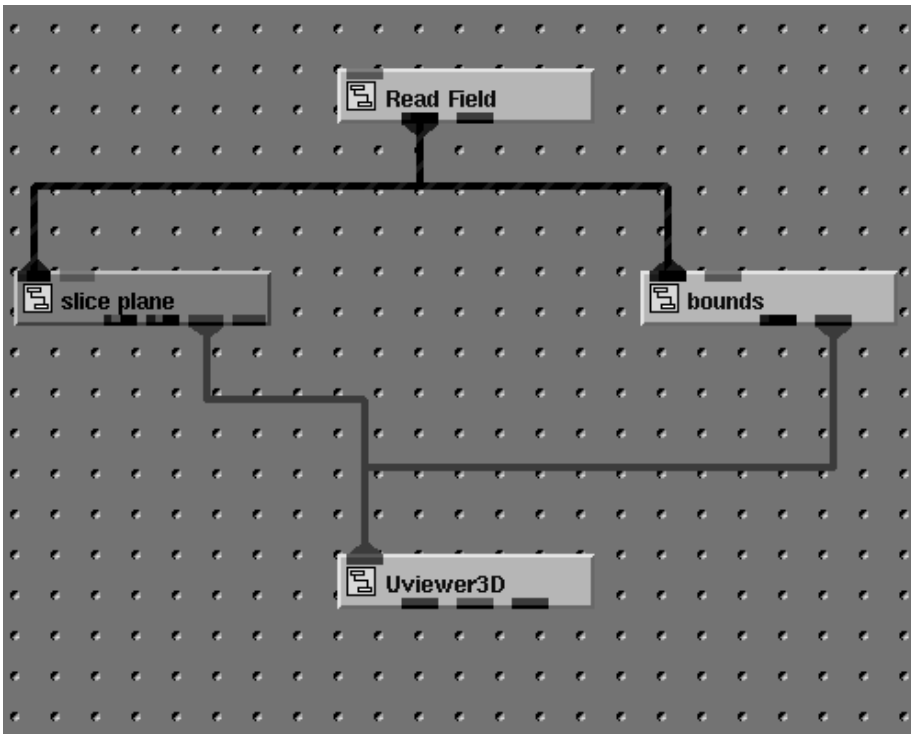
- Select the `glyph` module's control panel.
- Reduce the scale value using the slider to make the arrows a more acceptable size
- Try altering some of the parameters - use the module help facility to understand their effect.

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.

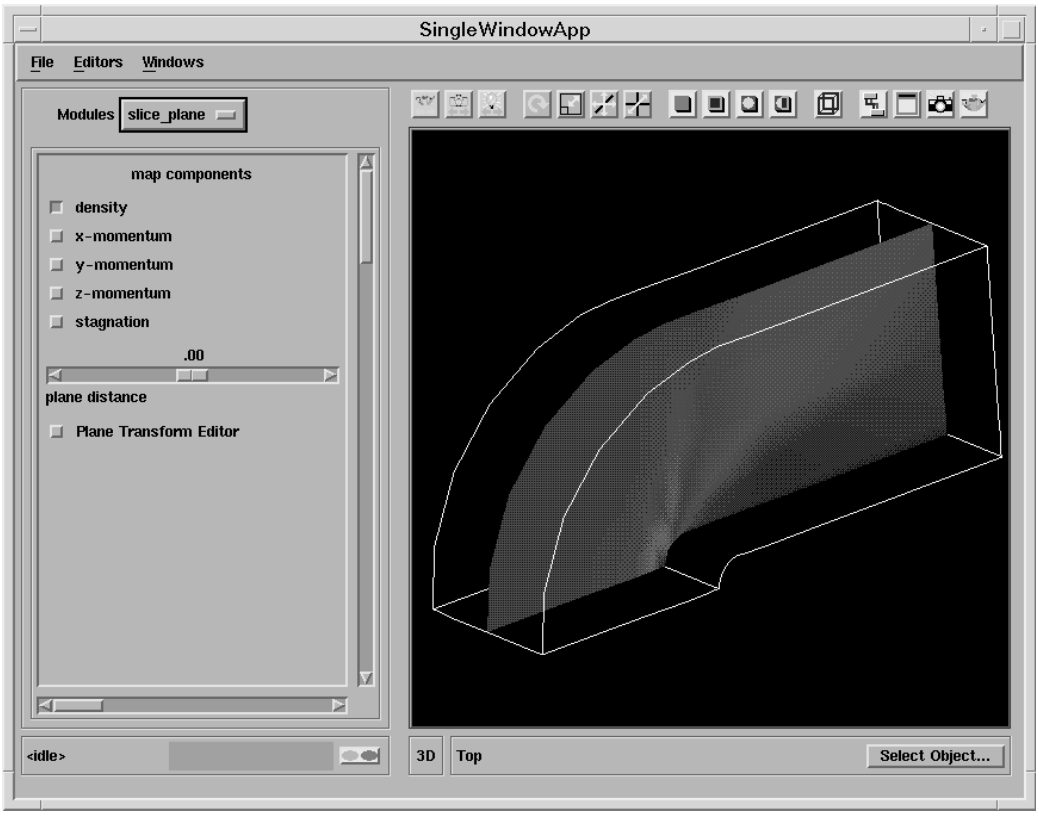
4.2.7 Visualizing the scalar data



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 Exercise 2

The aim of this exercise is to visualize some data obtained from a climate modelling experiment. The data contains values of temperature, specific humidity and u, v, & w components of wind velocity, which are sampled over the surface of the globe and at several altitude levels. There are two sets of coordinates for the data sampling: rectangular and spherical.

Firstly, you are to present the scalar data - temperature and specific humidity - using the spherical coordinate system. Secondly you are to present the vector data, the resultant wind velocity, using the rectangular coordinate system. The solutions to the field files for this exercise are given in Appendix II .

4.3.1 Importing the climate data

Below is a description of the data files containing the climate modelling data.

Coordinates: The data is irregularly spaced on a 49x12x40 grid. The coordinates are provided in ASCII files ccm1.rect which gives the coordinates in rectangular space and ccm1.sph which gives the coordinates in spherical space.

Both files have one line of header information and are arranged with all the coordinates of the first dimension followed by all the coordinates of the second dimension, followed by all the coordinates of the third dimension. There are eight values on each line of the files i.e.

```
x1 x2 x3 x4 x5 x6 x7 x8
x9 x10 ............
y1 y2 y3 y4 y5 y6 y7 y8
y9 y10 ............
z1 z2 z3 z4 z5 z6 z7 z8
z9 z10 ............
```

Temperature and Specific Humidity: Temperature and specific humidity data are contained in binary files called temp.dat and humid.dat, respectively. The data consists of a single floating point value at each grid point. There are 16 bytes of header information in each file.

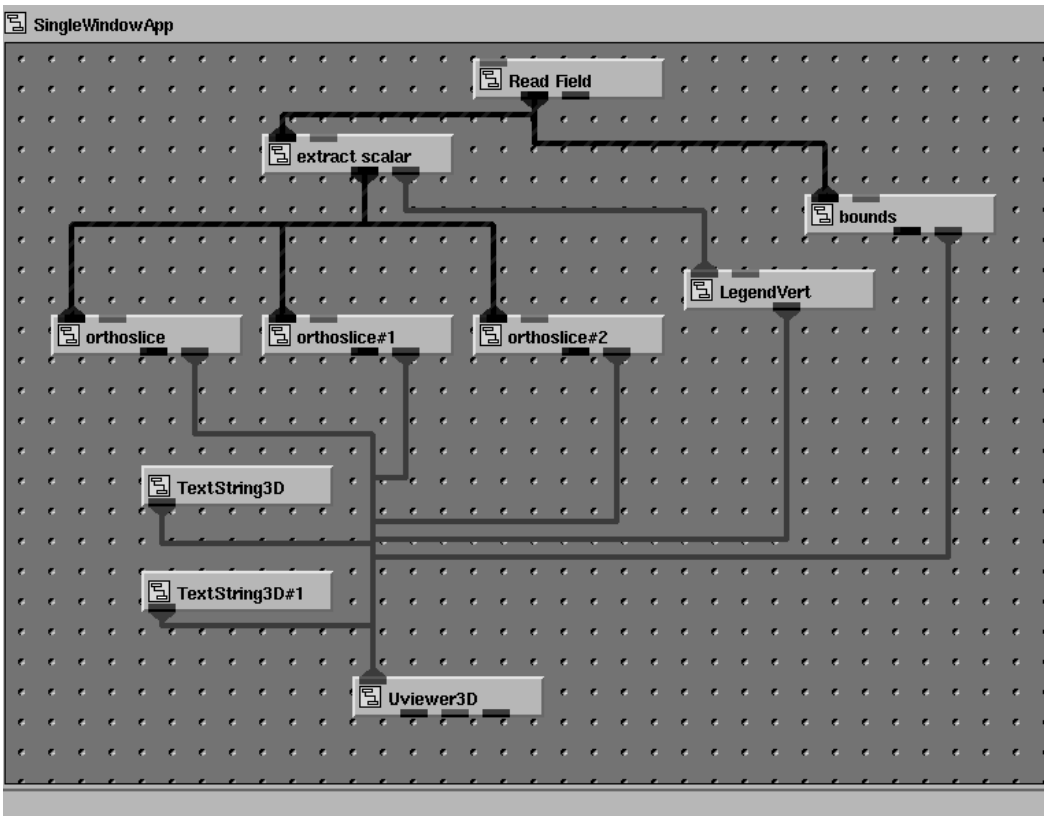
Wind Velocities: The wind velocity data is contained in three binary files of floating point values for the u, v & w components. The files are called bi.u.49, bi.v.49 and bi.w.49, respectively. Each file contains 16 bytes of header information.

World Map: A simple map of the world showing land/ocean outlines is contained in the file world.asc. The file is in ASCII format and irregularly spaced on the same grid as the other files. There is one line of header information.

In order to visualize this information you will need to write three field files:

- Create a field file called `climscal.fld` to describe the scalar data - temperature and specific humidity - using the spherical coordinate system.
- Create a field file called `climvect.fld` to describe the vector data - wind velocity - using the rectangular coordinates
- Create a field file called `worldrec.fld` to describe the world map using the rectangular coordinates.

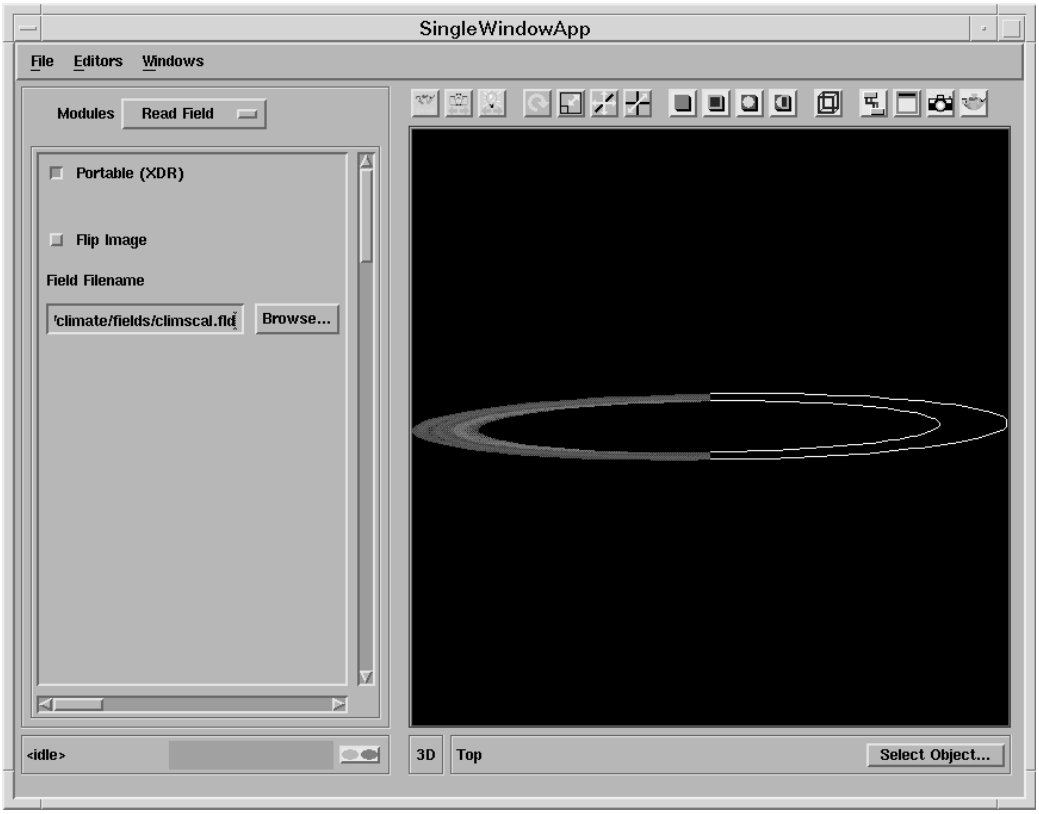
4.3.2 Visualizing the scalar data



Create the above network using the following modules:

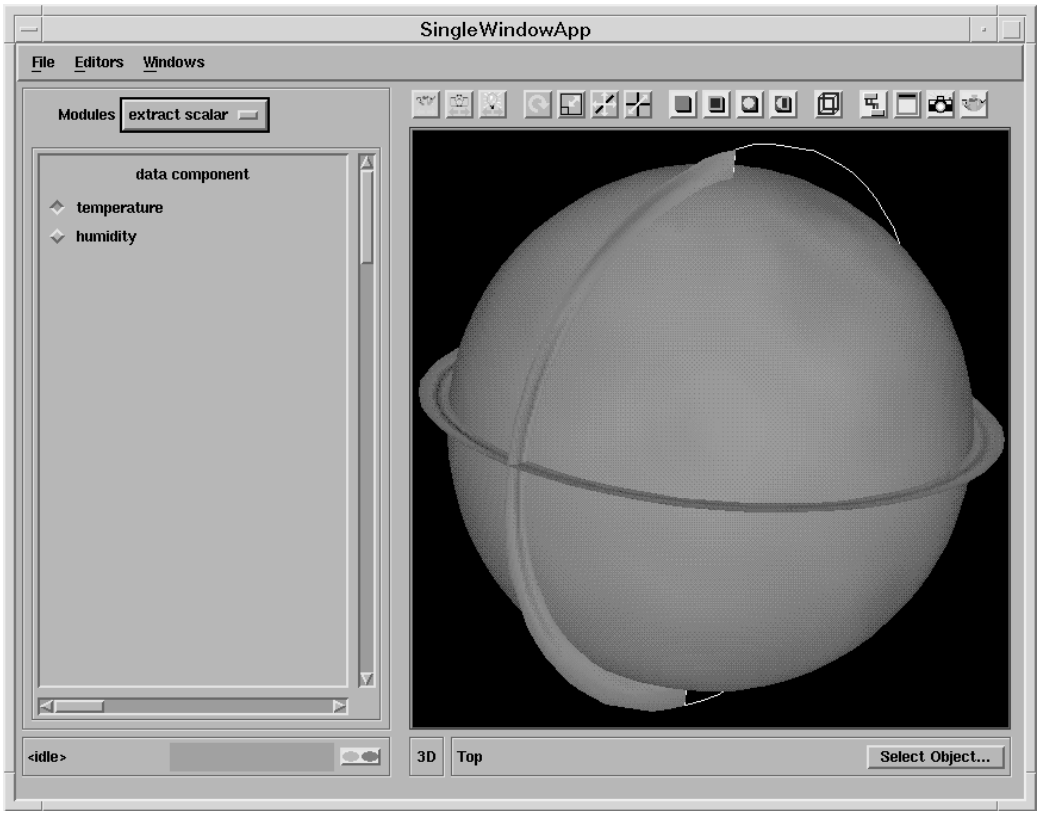
- `Read_Field` (Data Input): reads an AVS field file and converts it into an AVS/Express field.
- `extract_scalar` (Filters): extracts a single scalar data element from a field's vector component.
- `orthoslice` (Mappers): produces a slice of a structured field perpendicular to a selected coordinate axis.
- `bounds` (Mappers): generates a bounding box of a 3D field.
- `LegendVert` (Geometries): produces a color legend which displays the object's datamap.
- `TextString3D` (Geometries): produces a label.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

4.3.3 Changing the `orthoslice` parameters (1)



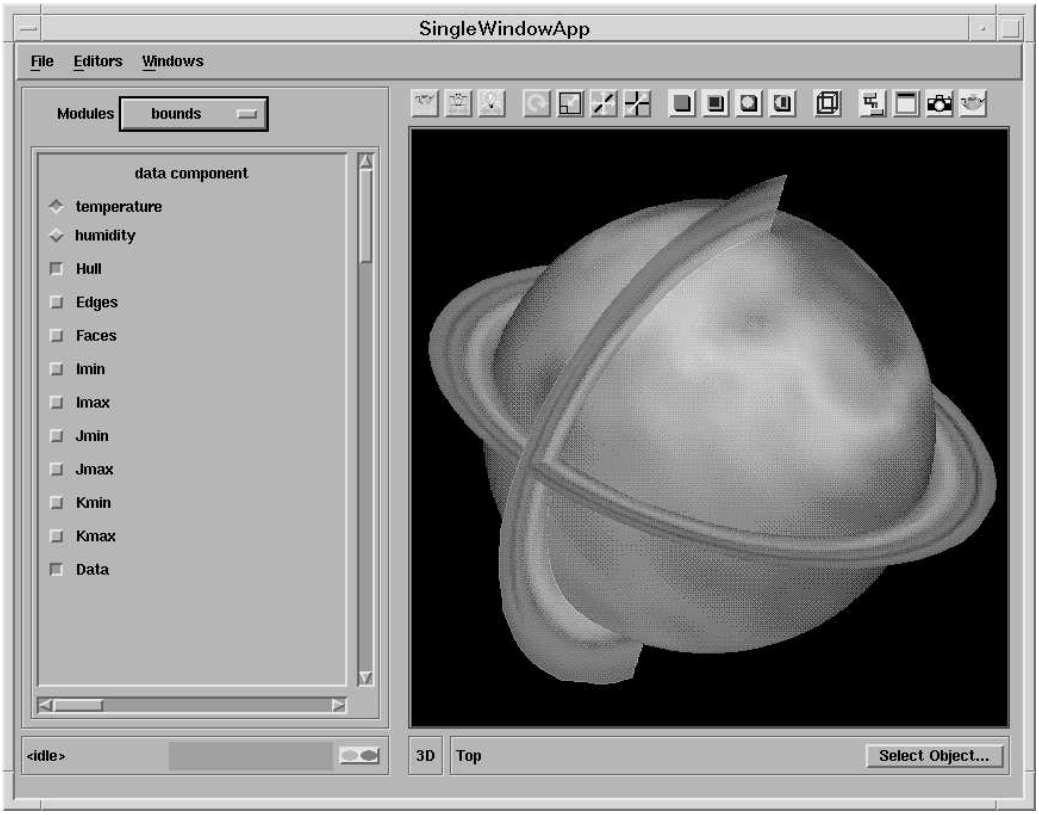
- Select `Read_Field` from the **Module Editors** menu and choose `climscal.fld` from the file browser.
- You should obtain a picture similar to the one above.
- Select the `extract_scalar` module and check that **temperature** is selected.
- Select the `orthoslice, orthoslice#1` and `orthoslice#2` modules in turn and set the **axis** values to 0, 1 and 2, respectively.

4.3.4 Changing the legend and label parameters



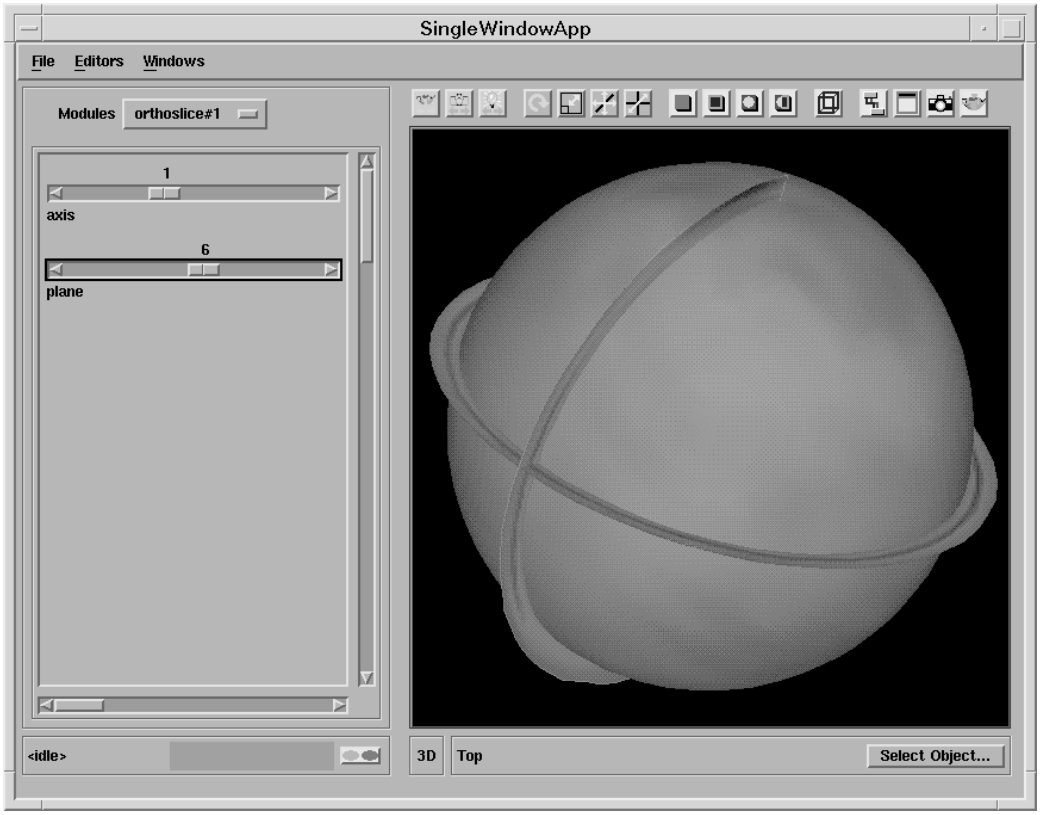
- Rotate the picture so that it appears as above.
- Select the `LegendVert` module and change the parameters to make them look optimal in the picture.
- Select the `TextString3D` modules and enter suitable text into the **string** box. Also change the position variables if necessary.

4.3.5 Changing the `orthoslice` parameters (2)



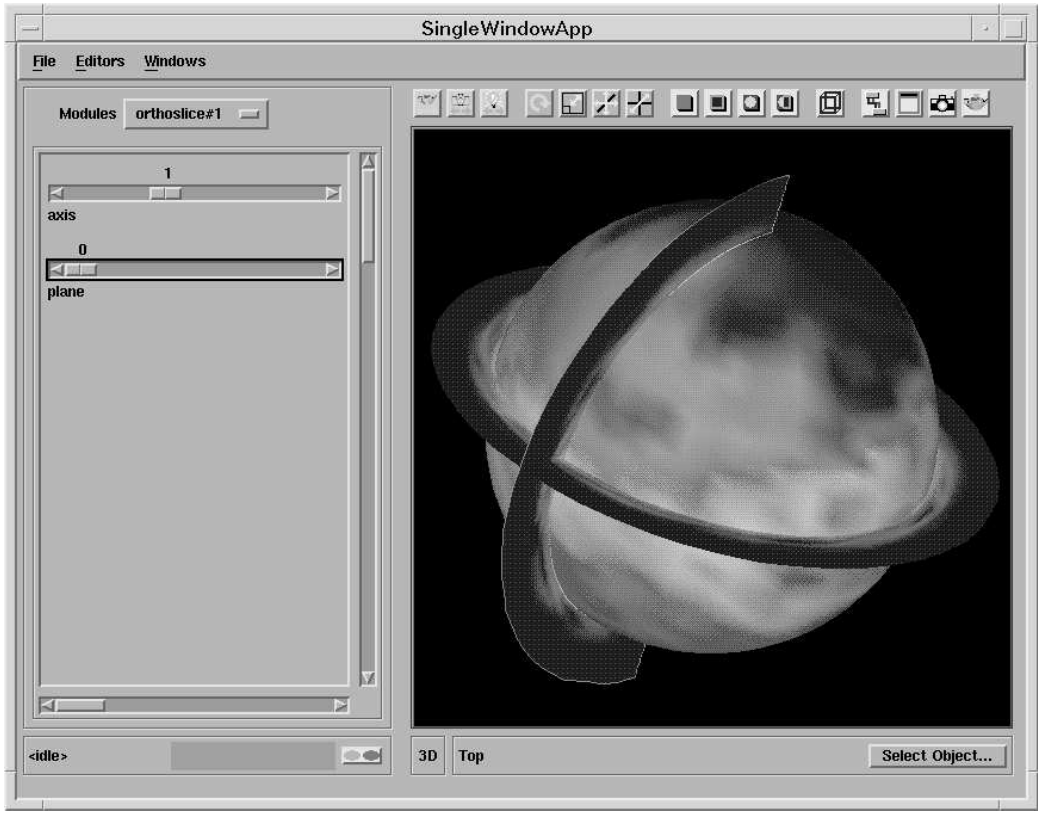
- Select the `orthoslice` and `orthoslice#1` modules, (or the modules with axes set to 0 and 1), and reduce the **plane sliders** to 0.
- You should obtain the above picture.
- You may wish to improve the picture further by selecting the `bounds` module and either removing the **hull** or selecting **data**, to colour the bounds to the boundary value of the data.

4.3.6 Looking at the temperature data



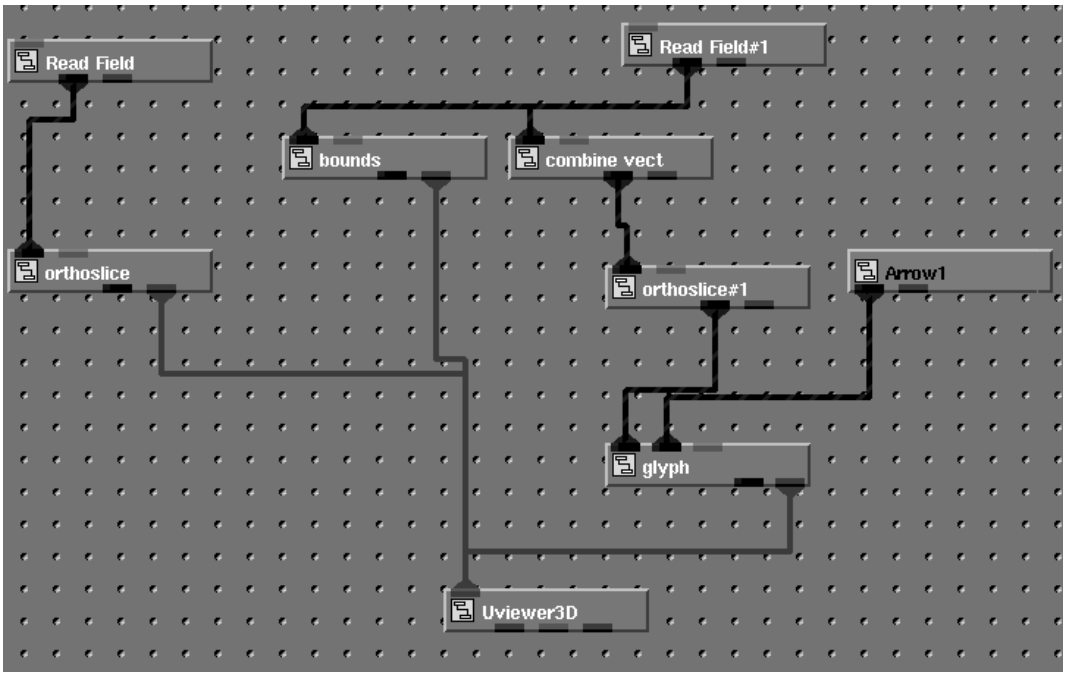
- Select the `orthoslice#1` module whose **axis** is set to 1, you should have just set the **plane slider value** to 0.
- This is the temperature around the surface of the earth.
- Increase the value of the **plane slider** and observe how the temperature of the atmosphere changes as altitude increases.

4.3.7 Looking at the humidity data



- Select the `extract_scalar` module and select **specific humidity**.
- Investigate how specific humidity varies with altitude.

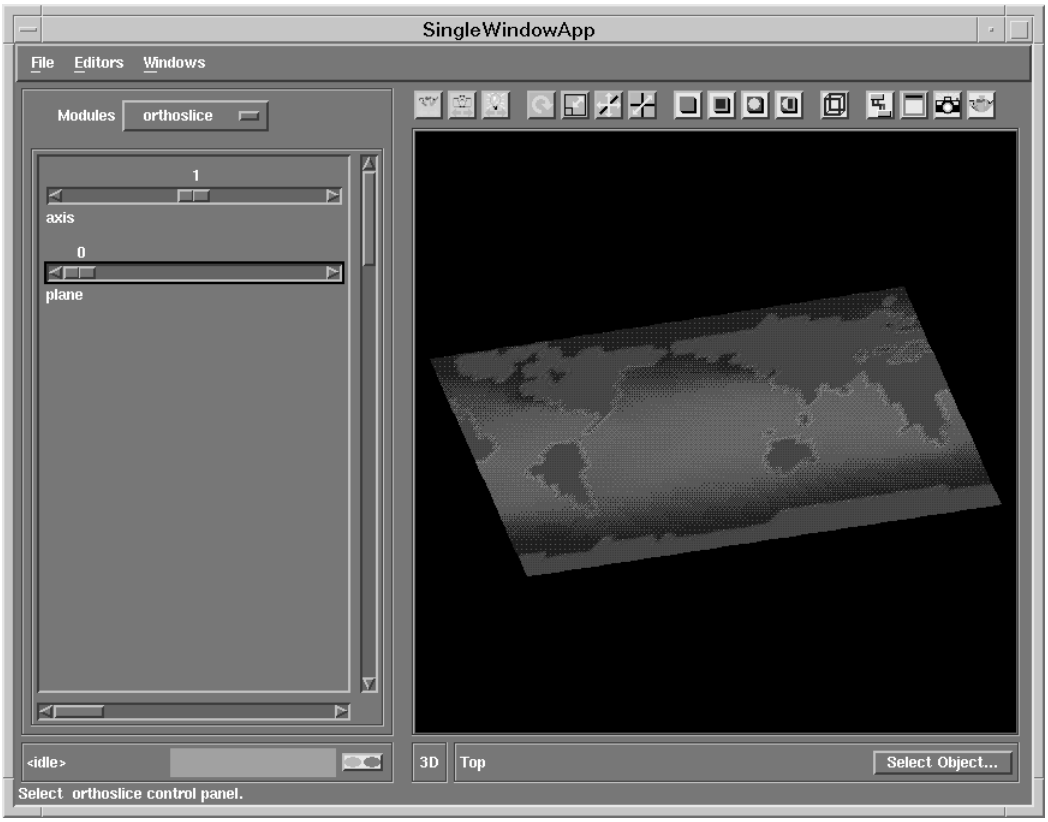
4.3.8 Visualizing the vector data



Create the above network using the following modules:

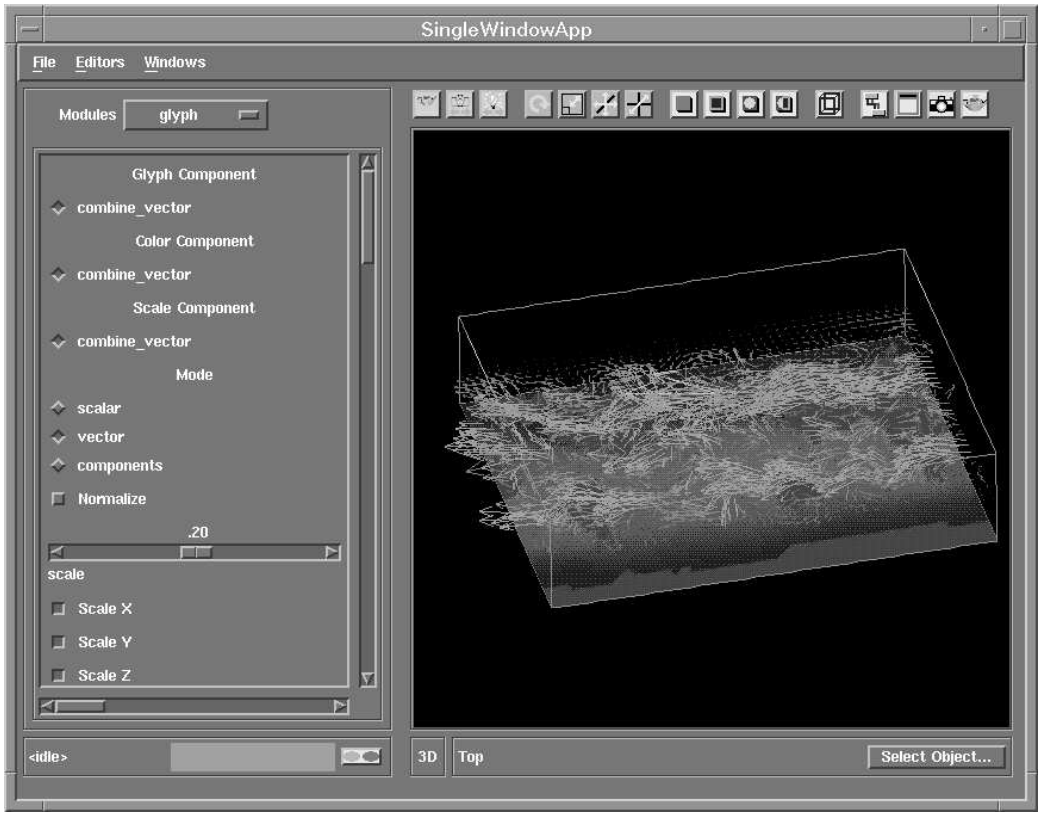
- `Read_Field` (Data Input): reads an AVS field file and converts it into an AVS/Express field.
- `orthoslice` (Mappers): produces a slice of a structured field perpendicular to a selected coordinate axis.
- `combine_vect` (Filters): takes all selected scalar data components and combines to output a 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.

4.3.9 Reading the world map



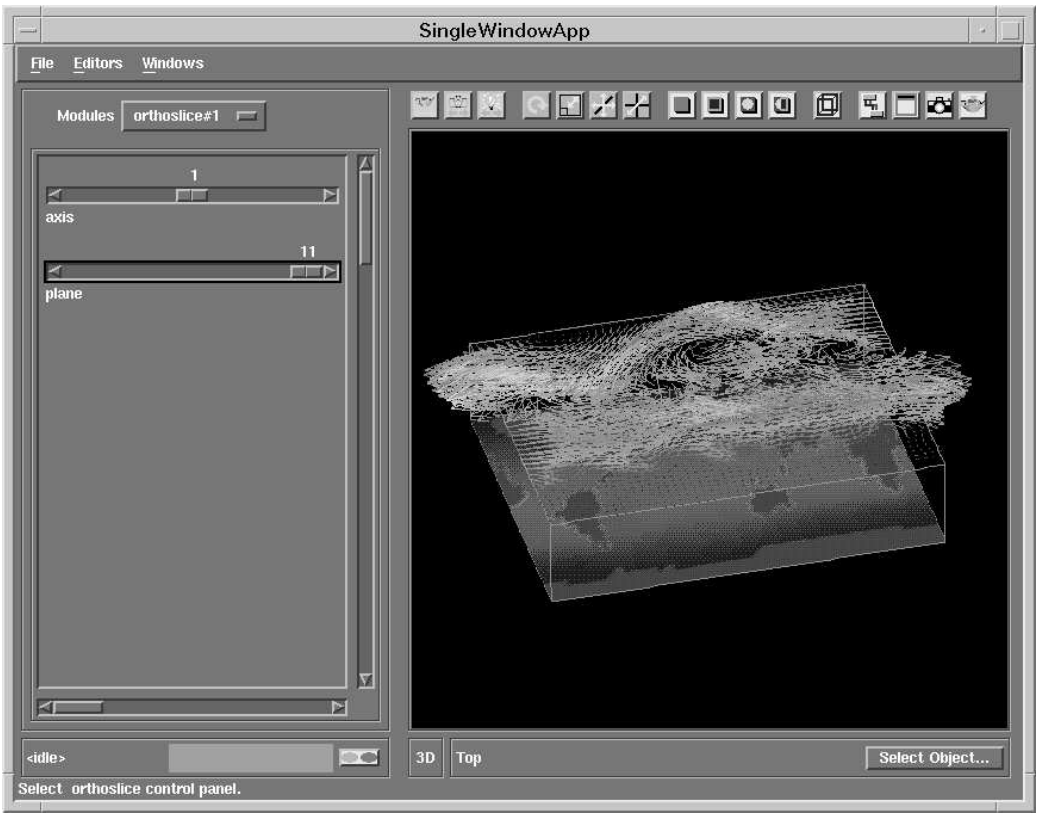
- Select the module `Read_Field` the **Module Editors** menu and choose `worldrec.fld` from the file browser.
- Select the `orthoslice` module which is connected to the `Read_Field` module and set the axis value to 1.
- Reduce the plane slider to 0.
- Using the mouse, rotate the picture until it is orientated as shown above.
- Hopefully the above picture will look familiar.

4.3.10 Reading the velocity data



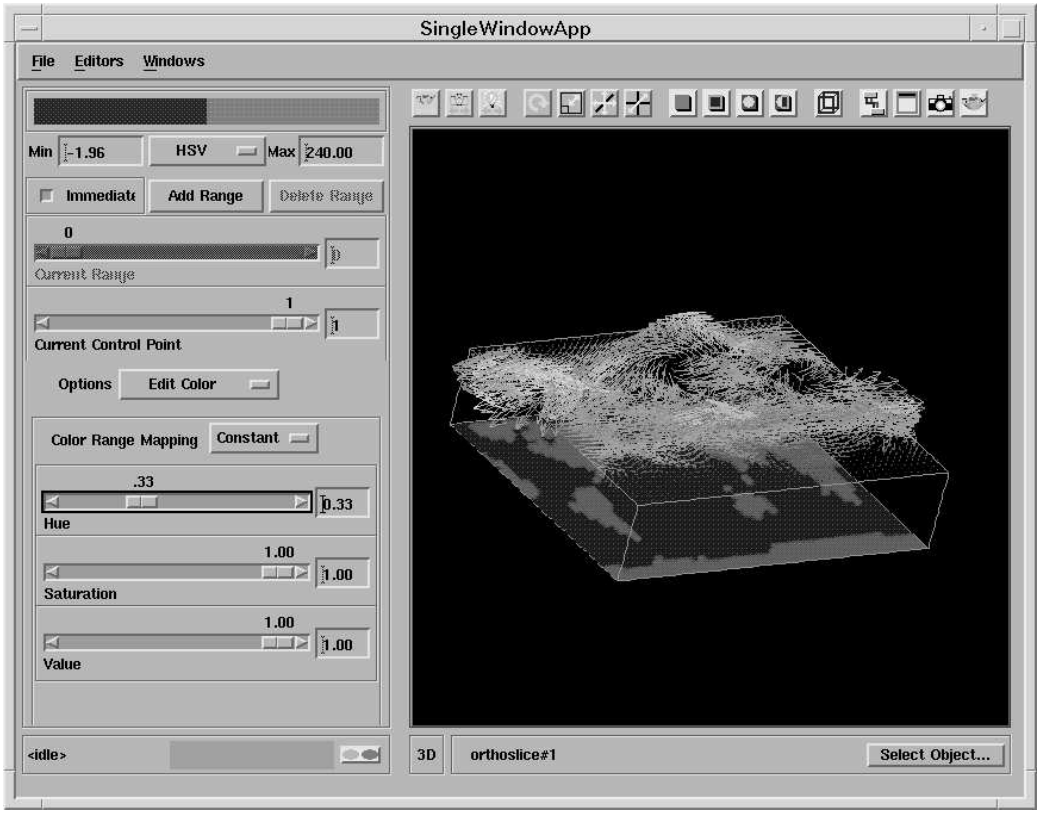
- Select the `Read_Field#1` module and choose `climrect.fld` from the file browser.
- Select the `combine_vect` module and check that **u-comp, v-comp** and **w-comp** are all selected.
- Select the `orthoslice` module which is attached to the `Read_Field` and set the **axis** to 1.
- Select the `bounds` module which is attached to the `orthoslice` module and remove the **hull**.
- Select the other `bounds` module and select **data** to colour the bounds.
- Select the `glyph` module and reduce the **scale** to ~0.2 or until the arrows are a sensible size.
- You should have obtained the above picture.

4.3.11 Looking at the velocity data



- Select the `orthoslice` module concerned with velocities and set the **plane slider** value to 0.
- This shows the velocities on the surface of the earth.
- Increase the plane value and observe how the wind velocities change with increasing altitude.
- The picture above shows the wind velocities at the highest altitiude level.

4.3.12 Changing the datamap



- To improve the appearance of the picture, attempt to alter the **datamap** of the world map so that the land is coloured green and the oceans are coloured blue.
- You may want to use the notes from Section 3.6.5 on page 36 to help you.

(Hint: Select the world map as the current object and make the colour map constant. Alter the values of the control points.)

4.4 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.

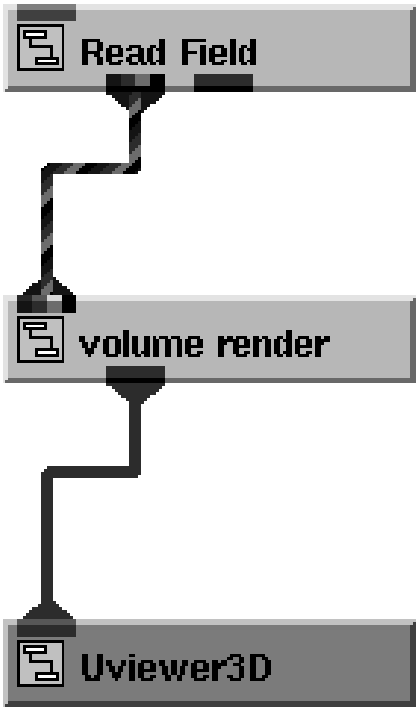
The rest of the exercises in this chapter are concerned with 3D representation of various fields. The examples use the `volume_render` module and all require the use of the software renderer.

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

4.5 Exercise 3

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

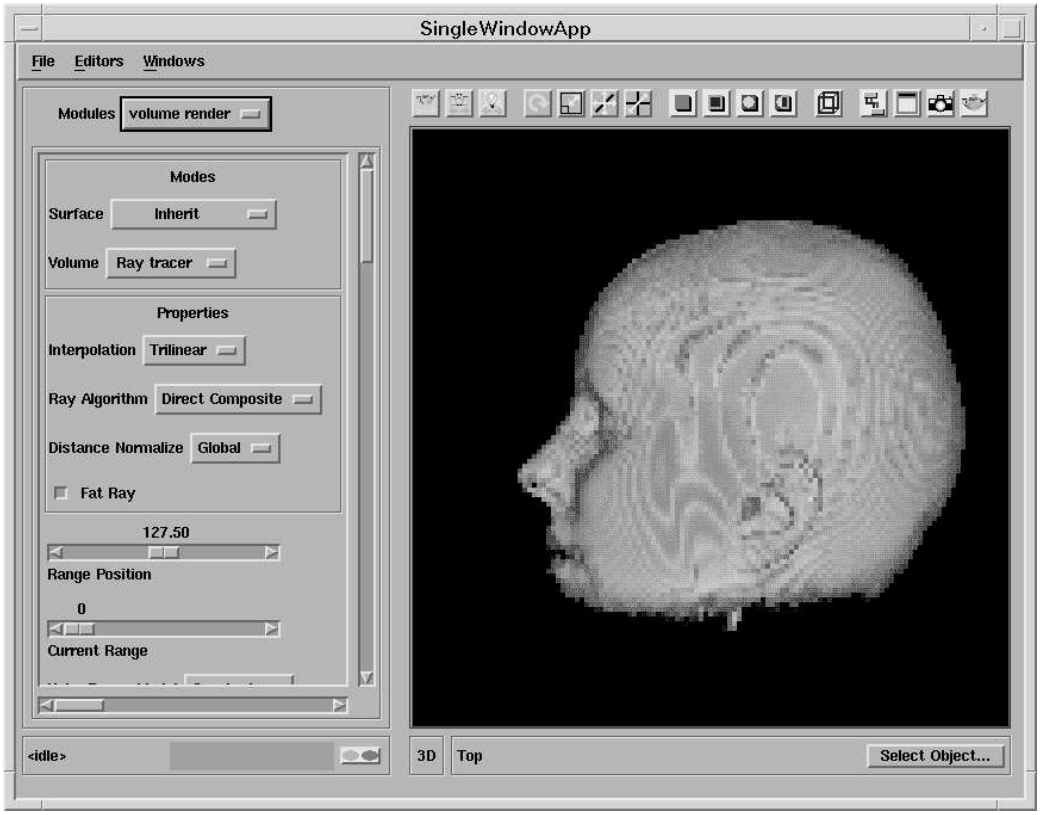
4.5.1 Displaying the `head.fld` data using `volume_render`



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.

4.5.2 Normalizing the picture



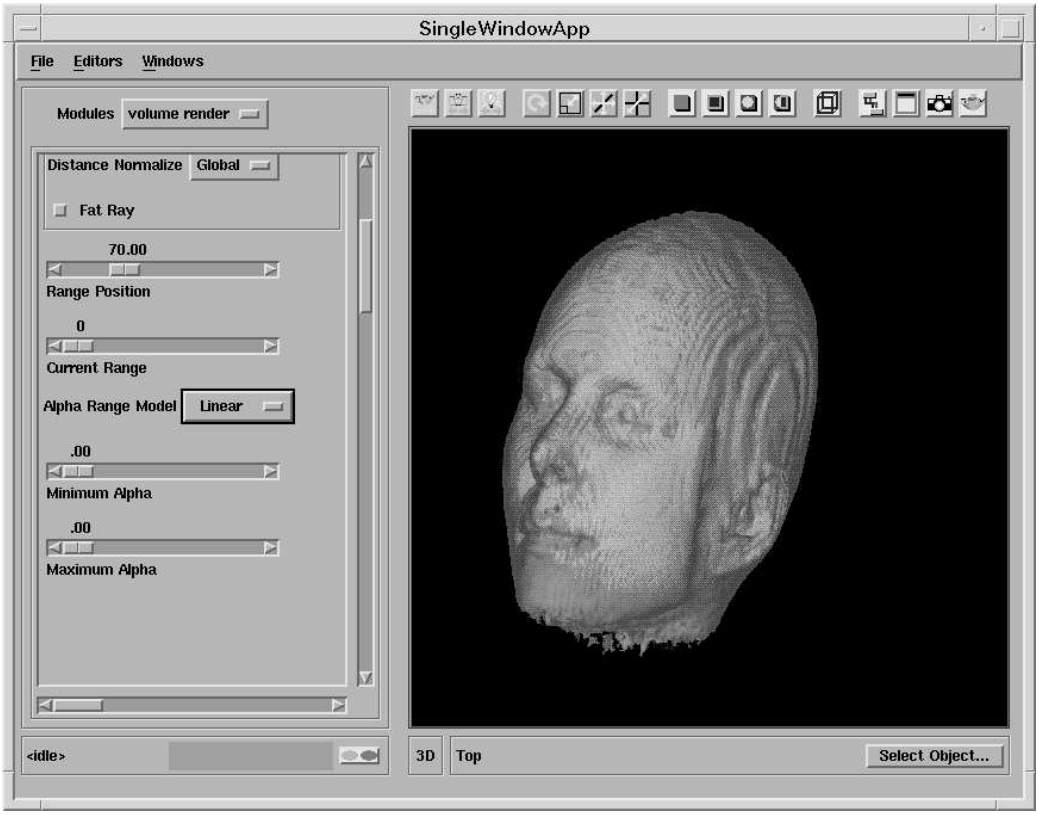
- Select the `Read_Field` module's control panel and select the file `head.fld,` from the file browser, to import the data. The `head.fld` file can be found under the path `/usr/avs/auxdata/...`

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 Toolbar 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 appearence of the image and explain some of the properties of `volume_render.`

4.5.3 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.

4.6 Exercise 4

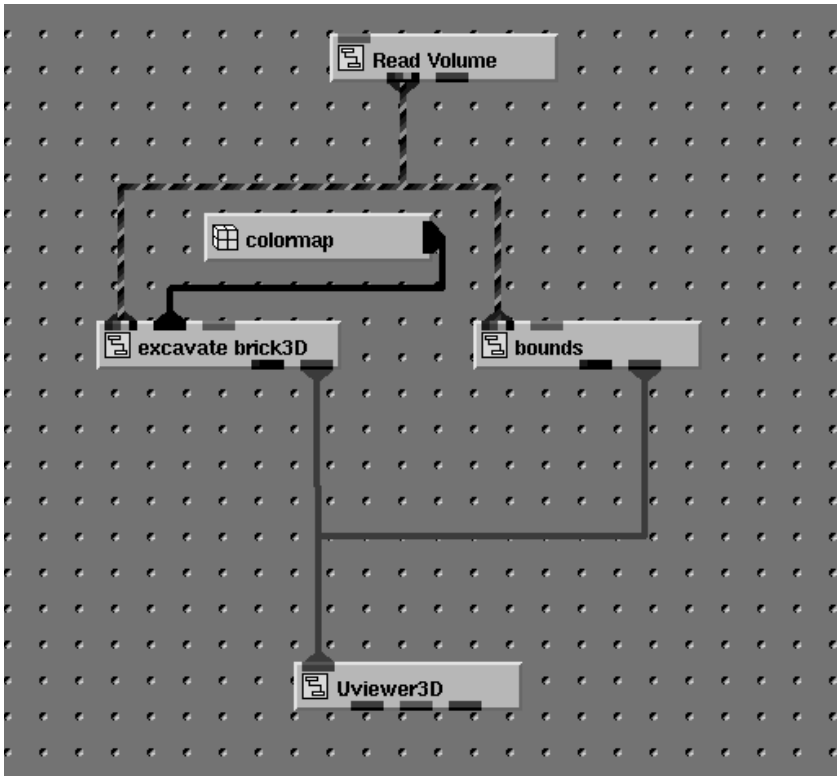
In this Exercise the electron density of the hydrogen atom will be represented, this time in 3D.

- Use the network constructed in Exercise 3.
- Select the `Read_Volume` module's control panel and select the file `hydrogen.dat,` in the file browser, to import the data.
- Click on the **Reset/Normalize/Center** icon in the Toolbar so you can fit the whole object in the window.
- Change some of the parameters of `volume_render` to improve the appearance of the image. Use the explanations of the properties of the `volume_render` module given in the previous exercise.
- Using the mouse, transform the object.
- Finally, try adding a `slice_plane` module to the network and investigate the results obtained.

4.7 Exercise 5

This example will use the `excavate_brick` module to observe the 3D image of a lobster encased in a brick.

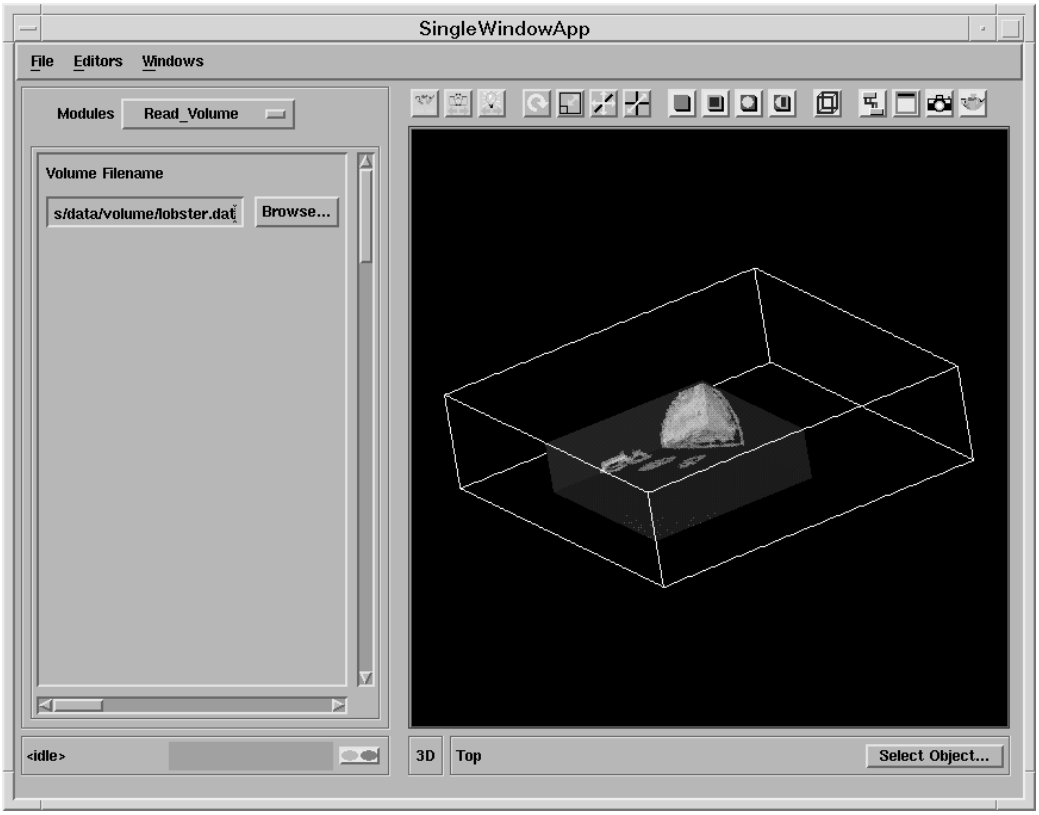
4.7.1 Displaying the `lobster.dat` data using `excavate_brick`



Create the above network using the following modules:

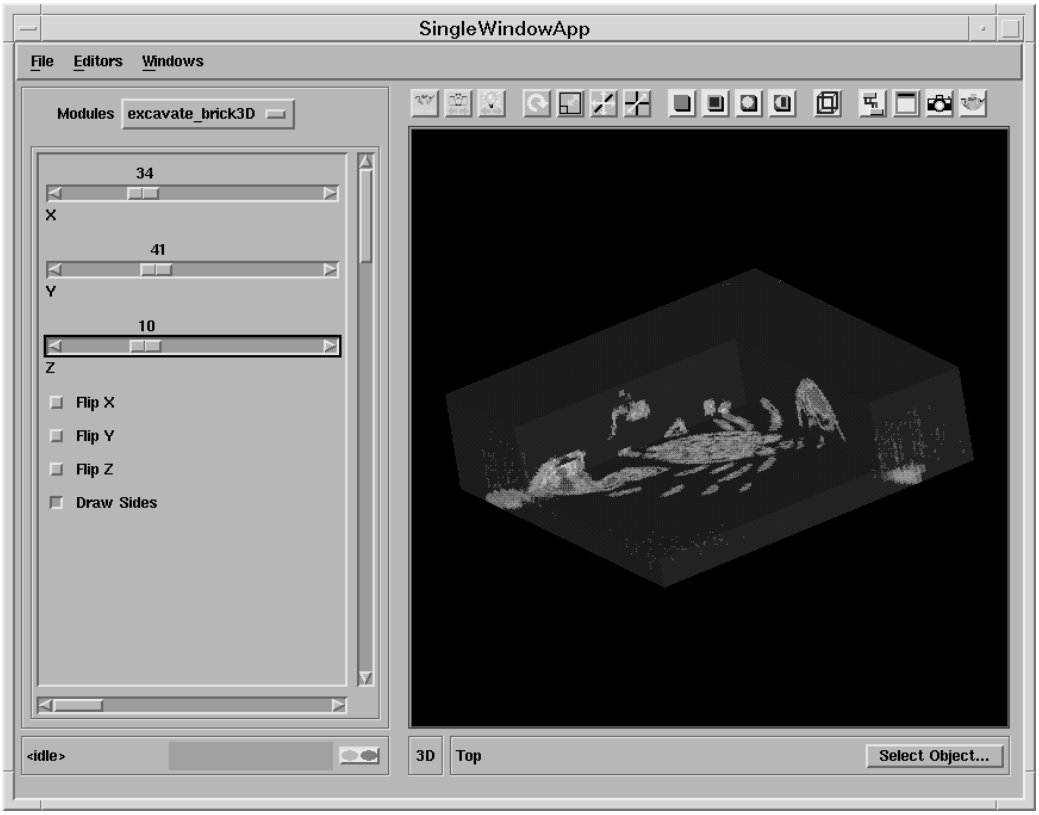
- `Read_Volume` (Data IO): reads a volume format file and outputs an AVS/Express field.
- `colormap` (Data IO): generates a coloured image.
- `excavate_brick3D` (Mappers): technique for visualizing 3D uniform volume. Volume is displayed with an X, Y & Z slice plane, which removes a rectangular sub-volume of the field, revealing the internal structure.
- `bounds` (Mappers): generates a bounding box of a 3D field.
- `Uviewer3D` (Viewers): produces a 3D image in the Data Viewer.

4.7.2 Normalizing the picture



- Select the `Read_Volume` module's control panel and select the file `lobster.dat` to import the data.
- 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.

4.7.3 Using `excavate_brick`



- Select the `excavate_brick` control panel from the **Modules** menu.
- Turn on **Draw Sides** - this allow the sides of the brick to be drawn.
- Try altering some of the other parameters and observe their effect.
- Select the `bounds` control panel and remove the **Hull**.

5 Visualizing UCD data

5.1 What is UCD Data?

UCD stands for Unstructured Cell Data. The format of UCD is just a different way for you to read your datafiles and visualize the information. UCD consists of:

- **Nodes**: contain data, which can be a collection of scalar and vector data, and position information.
- **Cells**: have geometric shape and are made up of reference nodes.
- **Structure**: collection of cells.

The UCD format is shown below:

```
<num of nodes> <num of cells> <data per node> <data per cell> <model data>
<nodeid> coordinates
<cellid> <cell shape> nodes
<data label> <data units>
<nodeid> data
```

UCD files have the suffix ".inp". An example of this is the `hex.inp` file which defines a single hexahedral cell with stress at each node. An extract from the file is given below:

```
8 1 1 0 0
   1  0.000   0.000   1.000
   2  1.000   0.000   1.000
   3  1.000   1.000   1.000
   4  0.000   1.000   1.000
   5  0.000   0.000   0.000
   6  1.000   0.000   0.000
   7  1.000   1.000   0.000
   8  0.000   1.000   0.000
   1  1 hex 1 2 3 4 5 6 7 8
   1 1
   stress, lb/in**2
   1  4999.9999
   2  18749.9999
   3  37500.0000
   4  56250.0000
   5  74999.9999
   6  93750.0001
   7  107500.0003
   8  5000.0001
```

5.2 Aims of the chapter

The writing of UCD format files is not covered by this course, but it is useful to know a little about their structure.

UCD is read into AVS/Express via the `Read_UCD` module. In order to help you to understand the process, there are several prepared UCD format files which can be found in the directory,

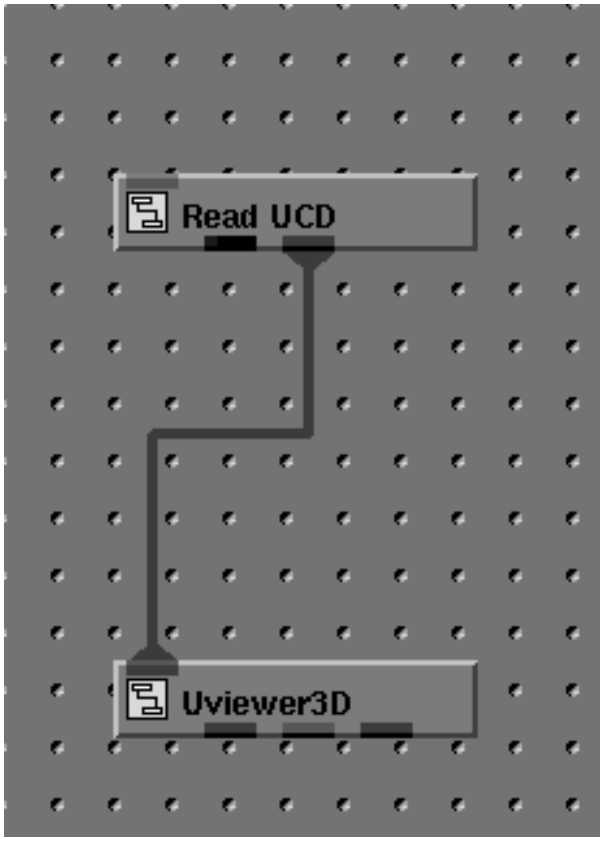
```
/usr/express/data/ucd...
```

The exercises in this chapter are designed to take you through several examples of UCD visualization and demonstrate the principal uses of the format.

5.3 Exercise 1

The aim of this exercise is to use the UCD file containing the AVS logo to demonstrate some simple networks which will display UCD data.

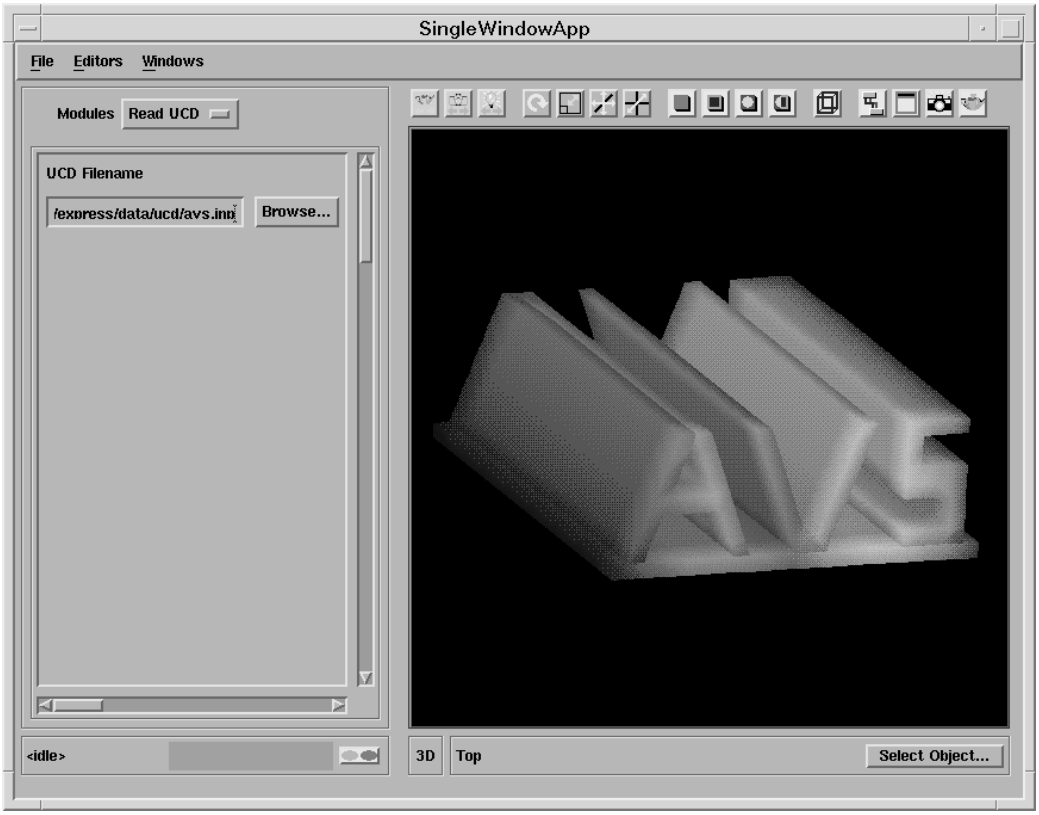
5.3.1 Displaying the UCD geometry (1)



Create the network above using the following modules:

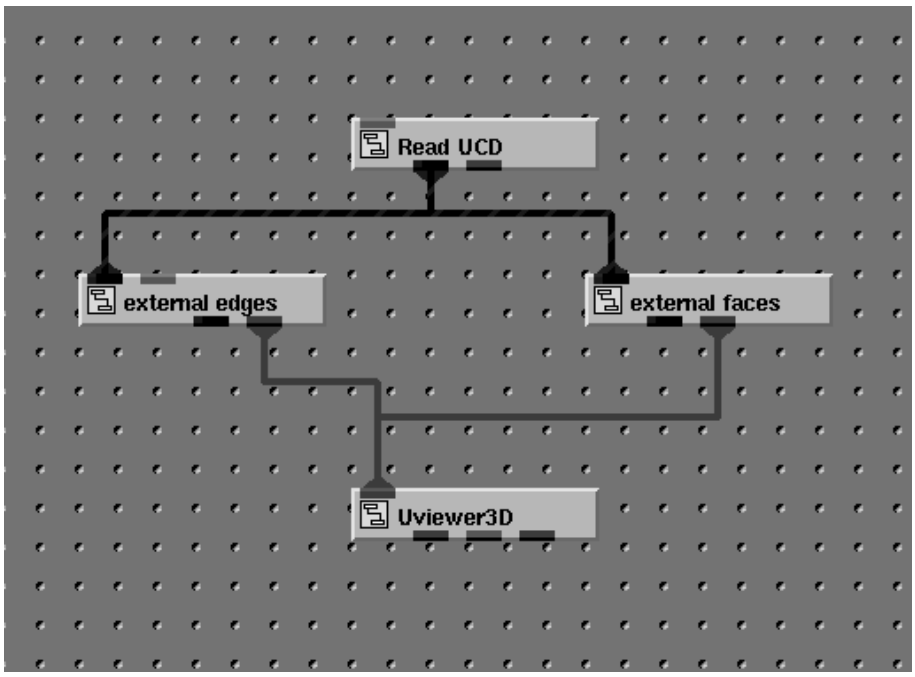
- `Read_UCD` (Data IO): reads an AVS UCD (.inp) file and outputs an AVS/Express field.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

5.3.2 Displaying the UCD geometry (2)



- Import a UCD file by selecting the file `avs.inp` using the `Read_UCD` file browser.
- Using the mouse transform and rotate the object until the picture is similar to that shown above.

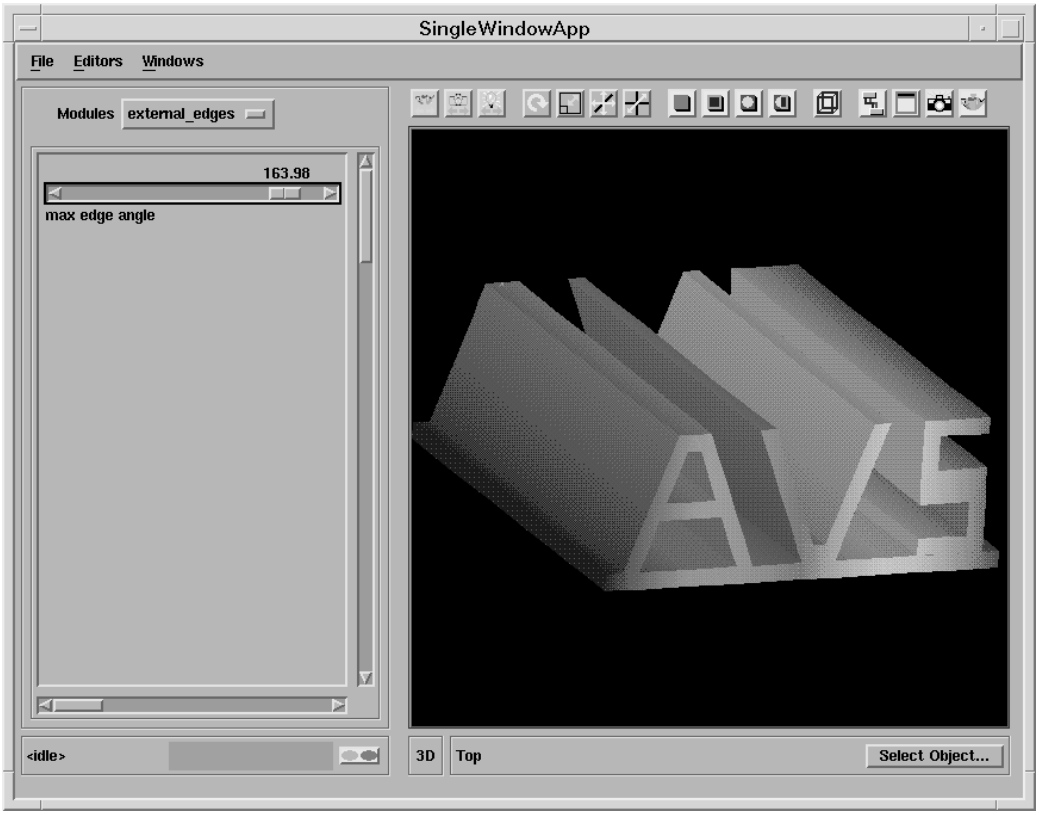
5.3.3 Displaying the UCD external edges and faces (1)



Create the network above using the following modules:

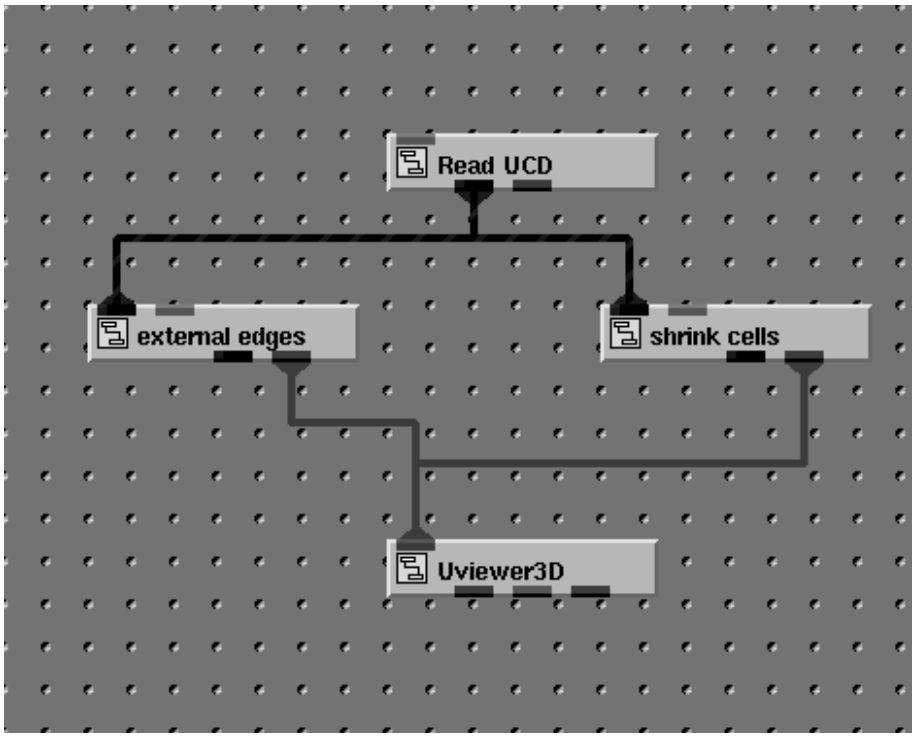
- `Read_UCD` (Data IO): reads an AVS UCD (.inp) file and outputs an AVS/Express field.
- `external_edges` (Mappers): produces a wireframe representation of the outside of an unstructured mesh to reveal objects inside.
- `external_faces` (Mappers): produces a mesh which represents the exterior, visible faces of an unstructured mesh.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

5.3.4 Displaying the UCD external edges and faces (2)



- Select the `avs.inp` UCD file using the file browser on the `Read_UCD` module.
- Select the `external_edges` module's control panel.
- Alter the max line angle and observe the effect.

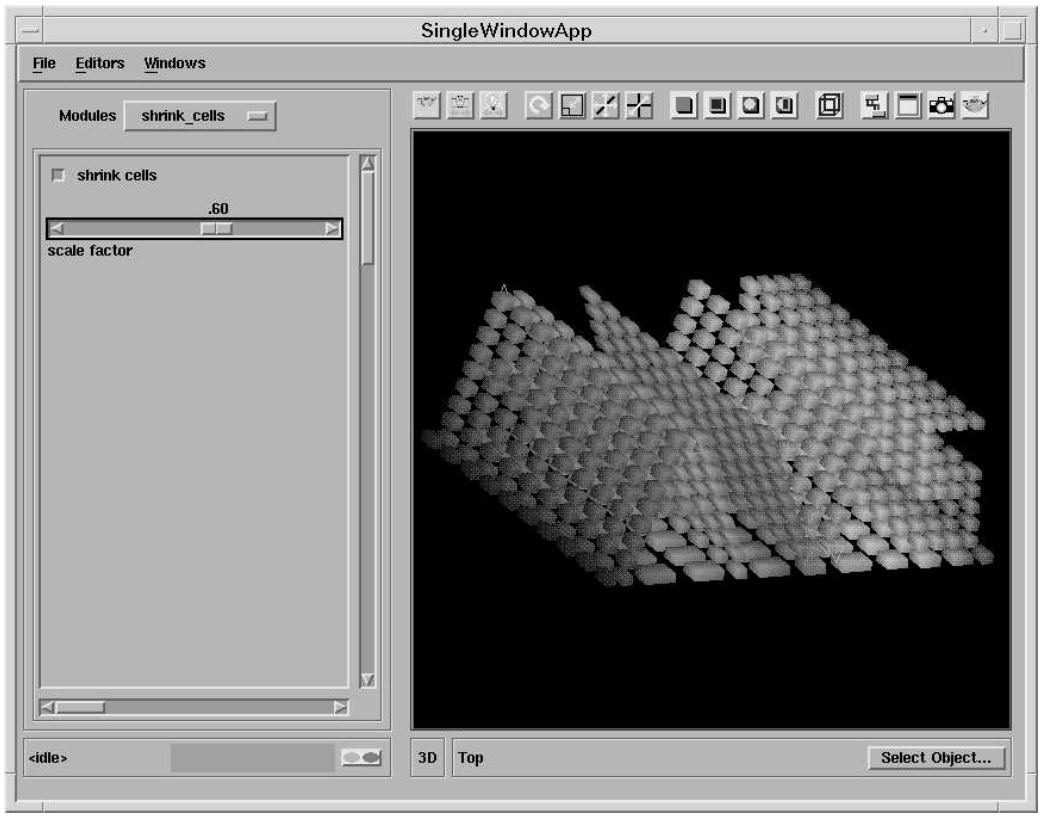
5.3.5 Shrinking the UCD cells (1)



Create the network above using the following modules:

- `Read_UCD` (Data IO): reads an AVS UCD (.inp) file and outputs an AVS/Express field.
- `external_edges` (Mappers): produces an wireframe representation of the outside of an unstructured mesh to reveal objects inside.
- `shrink_cells` (Mappers): produces a mesh with cells shrunk relative to their geometric centres.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

5.3.6 Shrinking the UCD cells (2)



- Select the `avs.inp` UCD file using the file browser on the `Read_UCD` module.
- Alter the shrink factor and observe the effect.

5.4 Further exercises

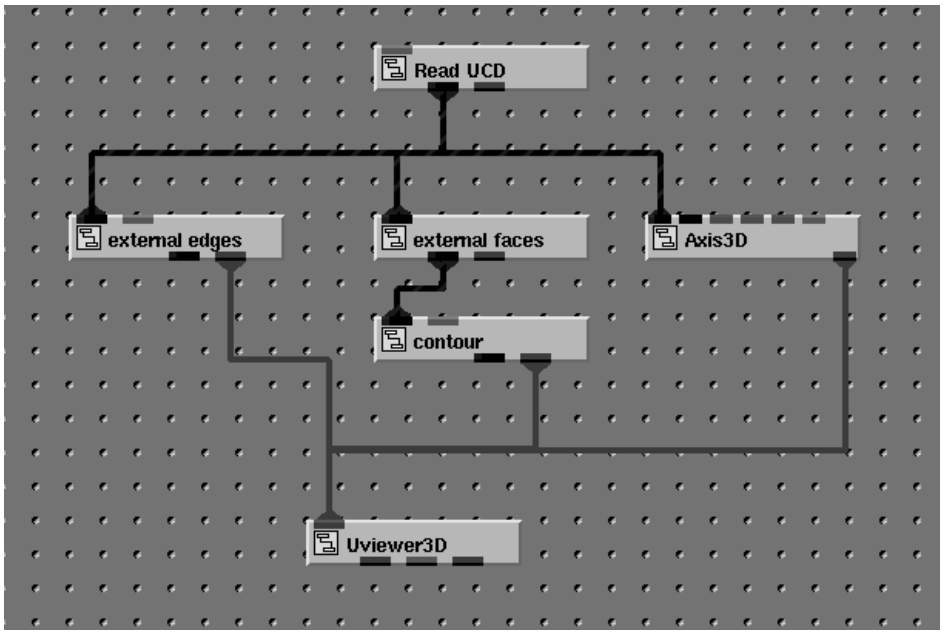
Now that you have worked though some basic UCD examples, you may wish to try out your networks on some other UCD files. Below is a list of several UCD files you may like to look at, they are to be found in the same directory as the AVS logo file.

- `pyr.inp`: example of a pyramid shaped cell
- `tet.inp`: example of a tetrahedral shaped cell
- `bluntfin.inp`: flow of air over a fin

5.5 Exercise 2

In this exercise you will look at several different networks which display the bluntfin data as a UCD.

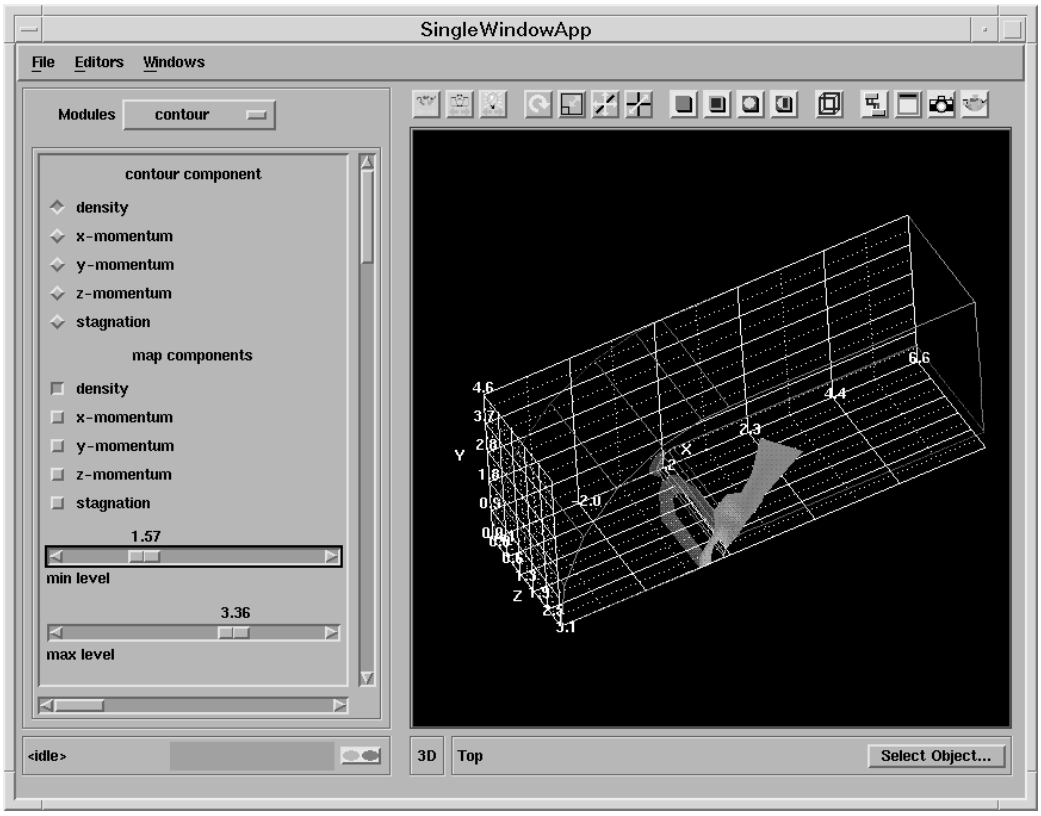
5.5.1 Displaying scalar data using `contour` module (1)



Create the network above using the following modules:

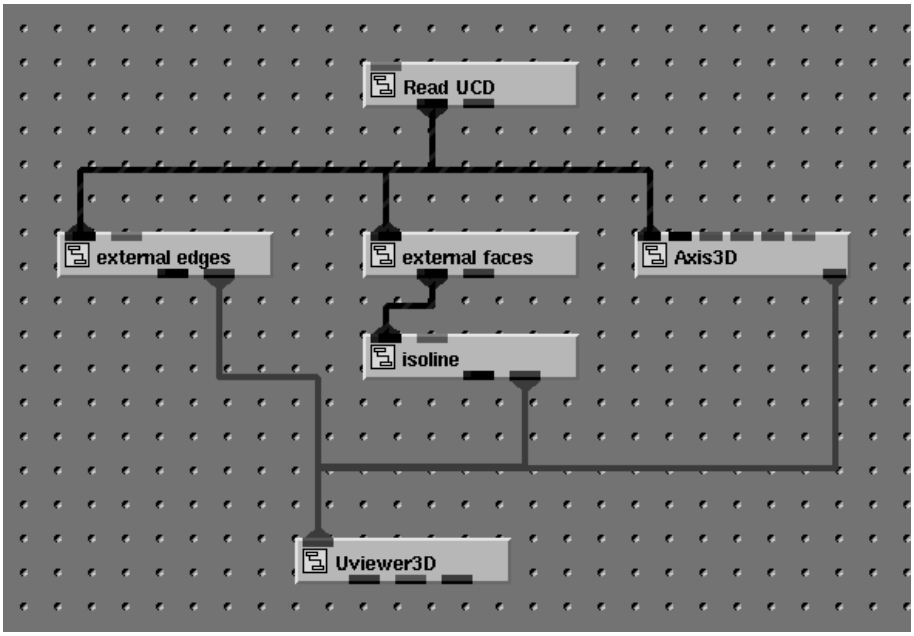
- `Read_UCD` (Data IO): reads an AVS UCD (.inp) file and outputs an AVS/Express field.
- `external_edges` (Mappers): produces an wireframe representation of the outside of an unstructured mesh to reveal objects inside.
- `external_faces` (Mappers): produces a mesh which represents the exterior, visible faces of an unstructured mesh.
- `Axes3D` (Geometries): creates an XYZ axis grid with labels based on mesh extents.
- `contour` (Mappers): creates an isovolume bounded by two isosurfaces (3D) or isolines (2D).
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

5.5.2 Displaying scalar data using `contour` module (2)



- Select the file `bluntfin.inp` using the `Read_UCD` module's file browser.
- Using the mouse, rotate the object so that it appears as above.
- Select the `contour` module's control panel
- Make active the data component density
- Try altering the max and min values to be contoured and observe the effect.

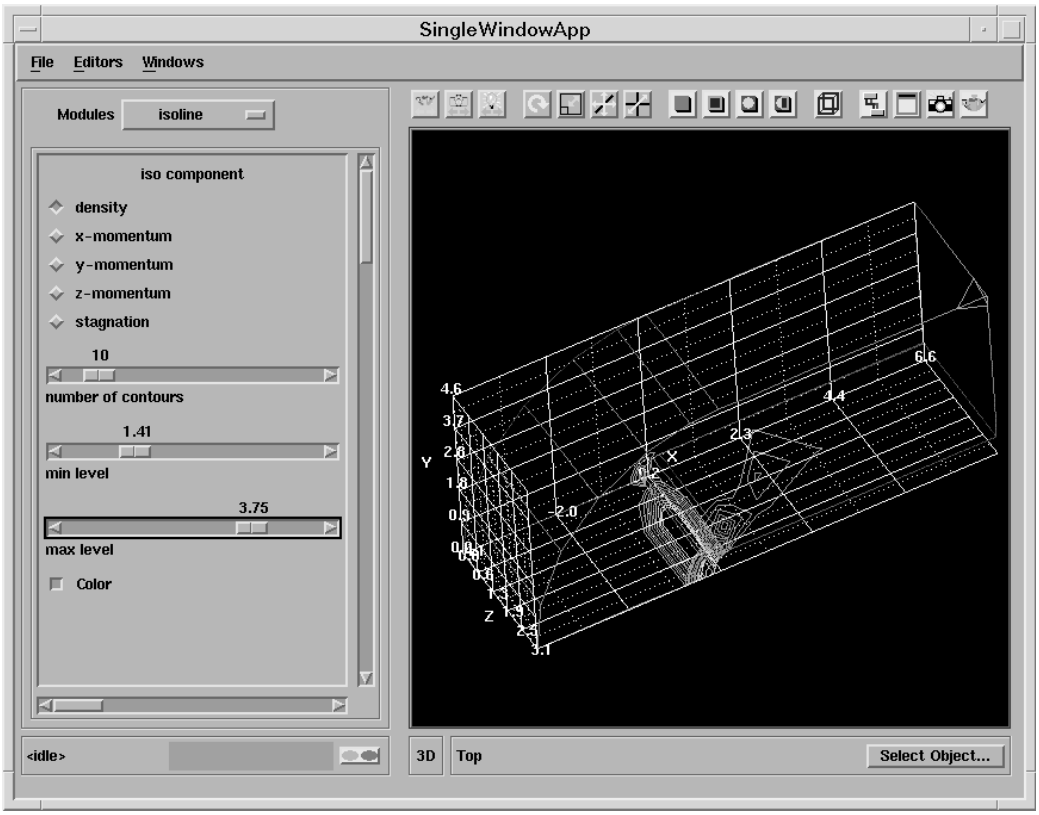
5.5.3 Using the `isolines` module (1)



Create the network above using the following modules:

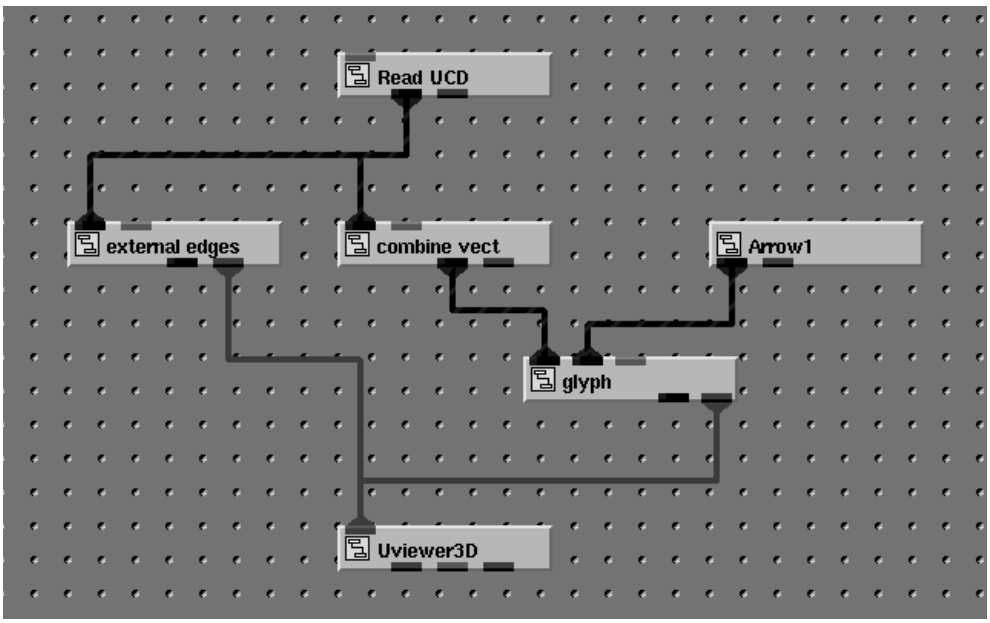
- `Read_UCD` (Data IO): reads an AVS UCD (.inp) file and outputs an AVS/Express field.
- `external_edges` (Mappers): produces an wireframe representation of the outside of an unstructured mesh to reveal objects inside.
- `external_faces` (Mappers): produces a mesh which represents the exterior, visible faces of an unstructured mesh.
- `Axes3D` (Geometries): creates an XYZ axis grid with labels based on mesh extents.
- `isoline` (Mappers): creates contour lines of constant value.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

5.5.4 Using the `isolines` module (2)



- Select the file `bluntfin.inp` using the `Read_UCD` module's file browser.
- Select the `isoline` module's control panel.
- Make active the data component density.
- You should have obtained a picture similar to the one above.
- Alter the min and max data values to be contoured and observe the effect.

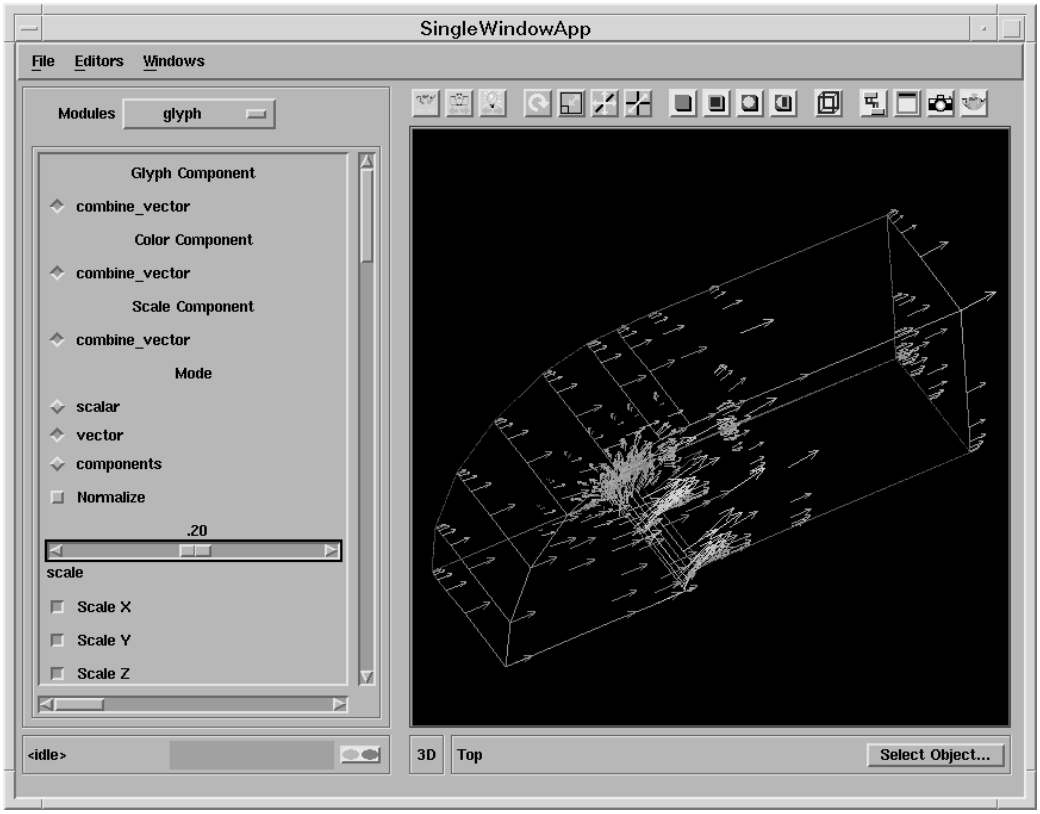
5.5.5 Displaying vector data (1)



Create the network above using the modules listed below:

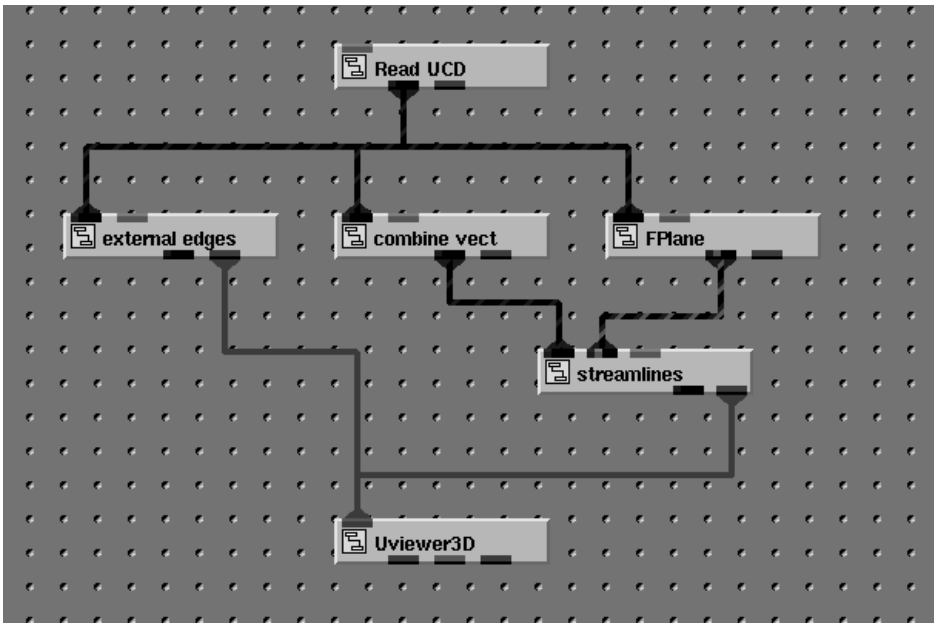
- `Read_UCD` (Data IO): reads an AVS UCD (.inp) file and outputs an AVS/Express field.
- `external_edges` (Mappers): produces an wireframe representation of the outside of an unstructured mesh to reveal objects inside.
- `combine_vect` (Filters): takes all selected scalar data components and combines to output a single vector component.
- `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.

5.5.6 Displaying vector data (2)



- Select the file `bluntfin.inp` using the `Read_UCD` module's file browser.
- Select the `combine_vectors` module and select x-momentum, y-momentum and z-momentum.
- Select the `glyph` module and reduce the **scale** to ~0.2 or until the arrows are a sensible size.
- You should have obtained a picture similar to the one above.

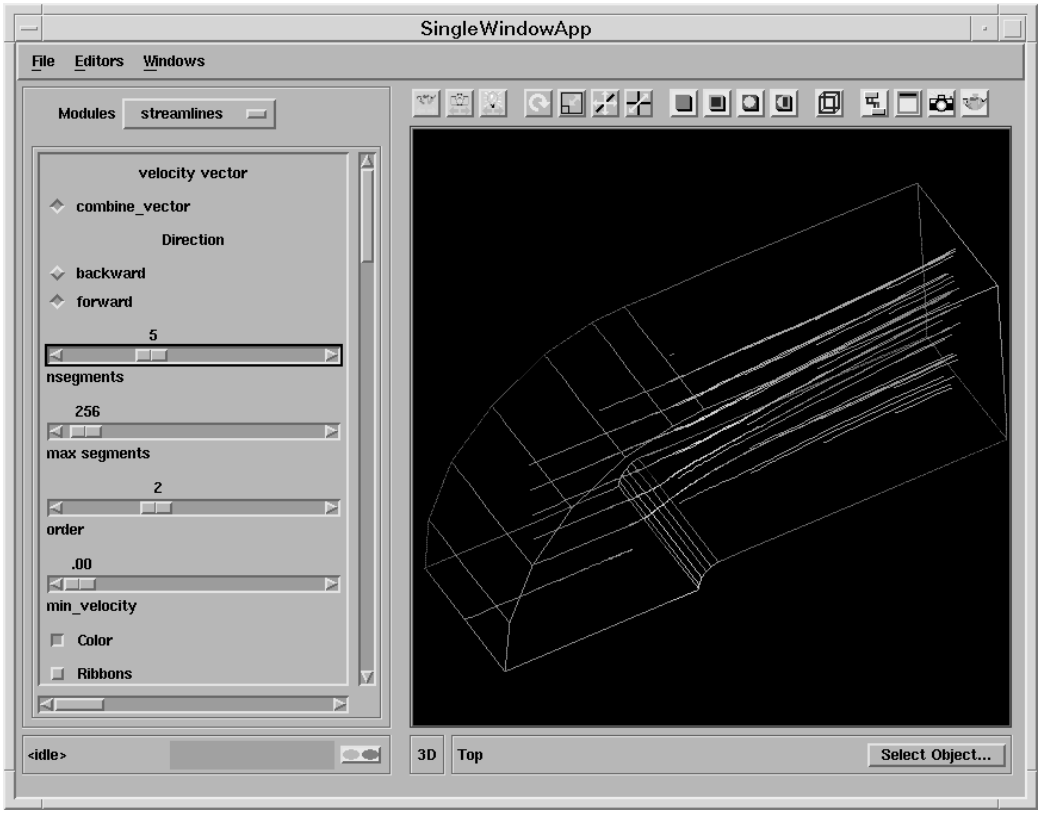
5.5.7 Using the `streamlines` module (1)



Create the network above using the following modules:

- `Read_UCD` (Data IO): reads an AVS UCD (.inp) file and outputs an AVS/Express field.
- `external_edges` (Mappers): produces an wireframe representation of the outside of an unstructured mesh to reveal objects inside.
- `FPlane` (Geometries): generates a 3D plane of variable size which can be transformed in three dinensions.
- `combine_vect` (Filters): takes all selected scalar data components and combines to output a single vector component.
- `streamlines` (Mappers): generates streamlines or streamribbons based on a field with a one, two or three element vector component.
- `Uviewer3D` (Viewers): creates a 3D image in the DataViewer.

5.5.8 Using the `streamlines` module (2)



- Select the file `bluntfin.inp` using the `Read_UCD` module's file browser.
- Select the `combine_vect` module and ensure that x, y & z momentum are selected.
- Select the `streamlines` module's control panel.
- Try altering the Nsegment value to increase the numbers of streamlines.
- Investigate the effect of changing the minimum velocity value.
- Change the streamlines to streamribbons.
- Try altering some of the `FPlane` module parameters.

6 AVS/Express examples

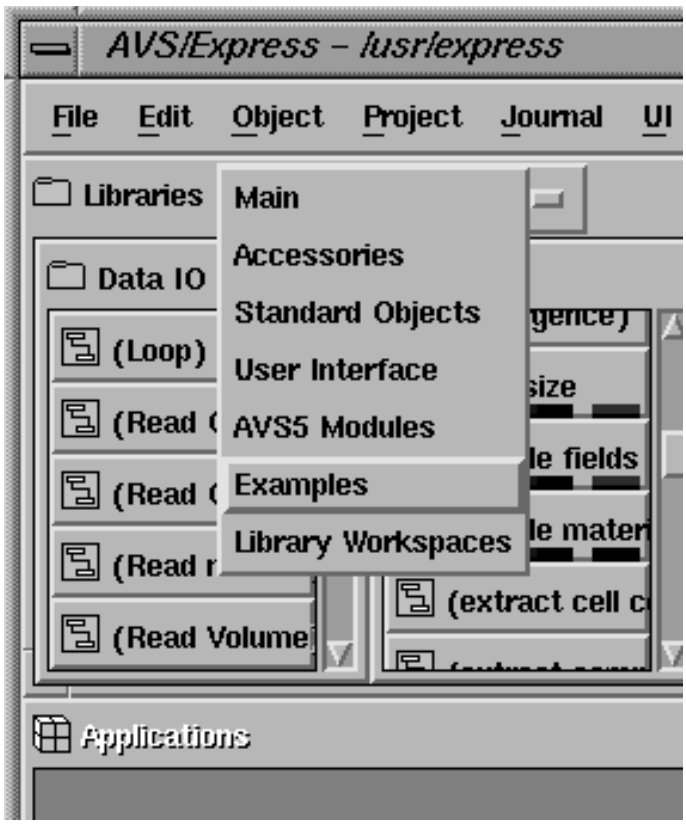
6.1 Aims of the chapter

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 `Advector` module as a demonstration. The `Advector` releases a sample of massless particles into a field. The particles move through the field according to the magnitude and direction of the vectors at the nodes. The `Advector` module is used to demonstrate fluid flow.

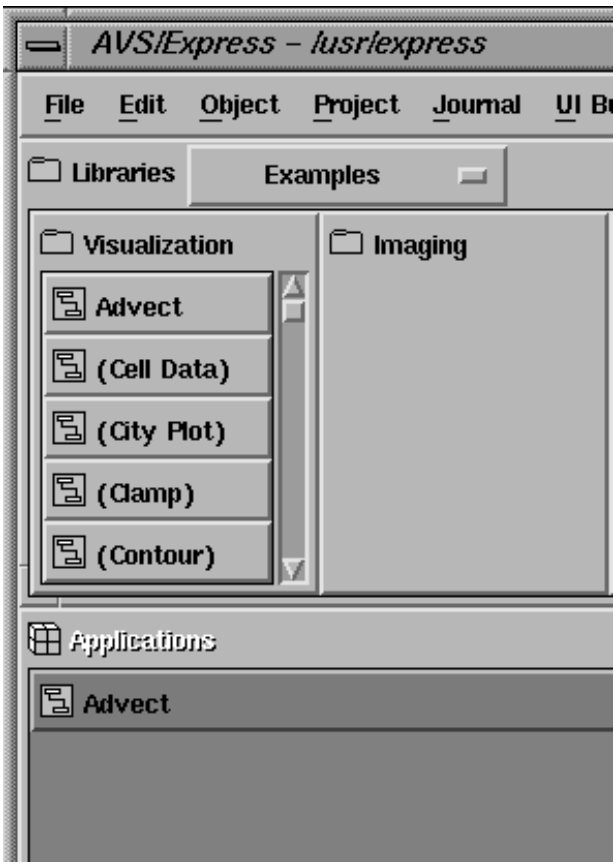
6.2.1 Entering the AVS/Express examples library



To access to the AVS/Express on-line examples

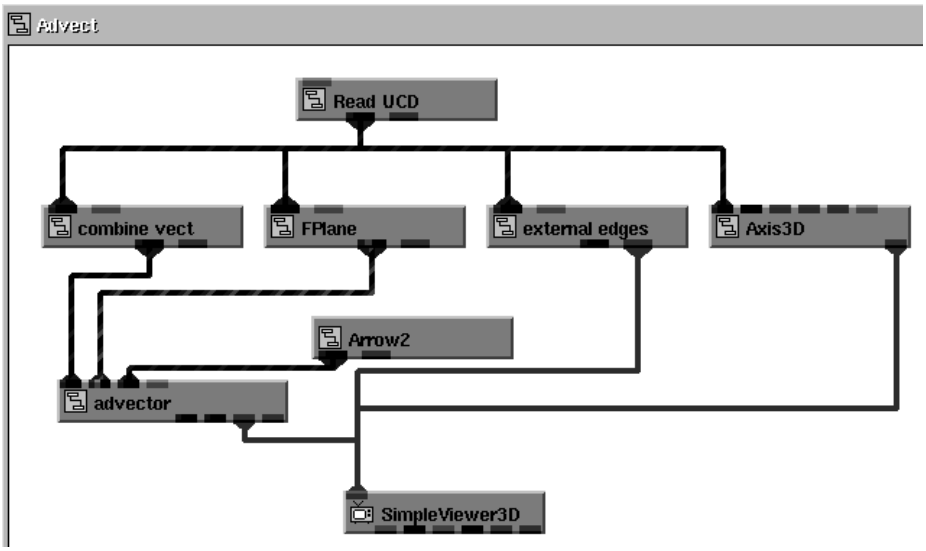
- Select the **Examples** library page from the **Library page** selector on the top of the **Network Editor window.**

6.2.2 Starting the `Advect` demo



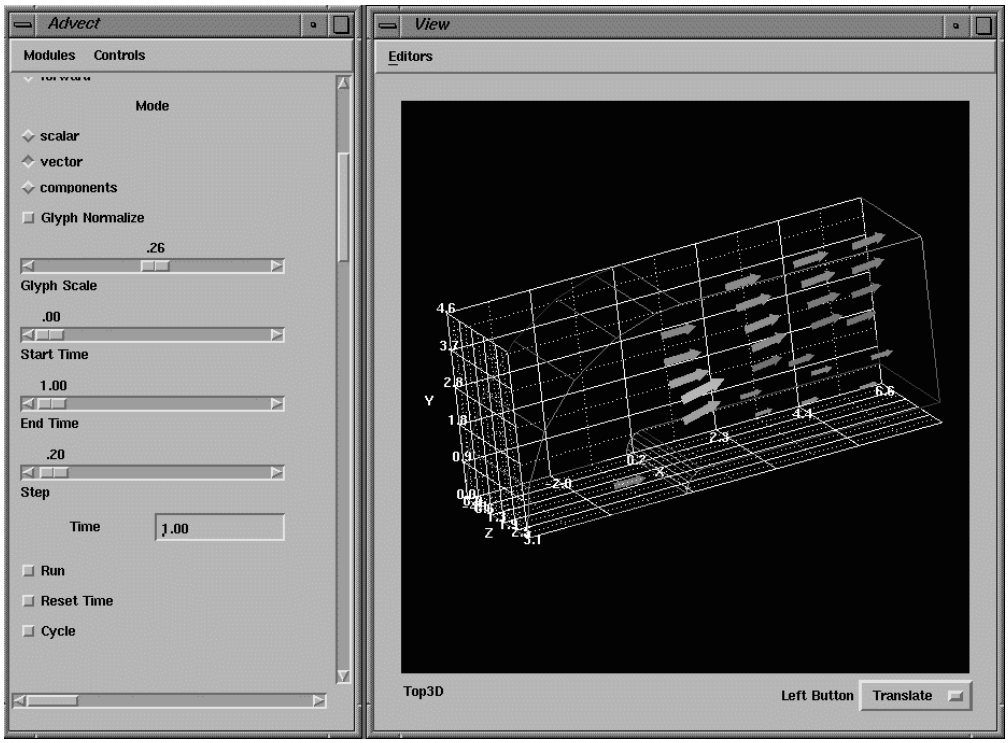
- Double click the **Visualization** sub-library to open it. (It may already be open).
- Instance the `Advect` demo module to the workspace.
- The screen should appear as shown below.

6.2.3 The `Advector` demo



- Double click the `Advect` in the workspace to maximize it. By doing so, the complete `Advect` network is shown in the workspace.
- Refer to each module's help documentation to understand the way the network operates, especially the `Advector` module.
- Rearrange the network so it appears similar to the layout shown above. You will need to select the **main** library page in order to select new modules.

6.2.4 Changing the parameters yourself



- Select the `Advector` module from the Modules menu.

The following steps will run the `Advector` demonstration.

- Before releasing the advectors select the **Reset Time** button.
- To release the advectors click the **Run** button.
- You can also use the **Cycle** mode. To stop the cycle mode click the **Run** button to turn it off.

You may also wish to,

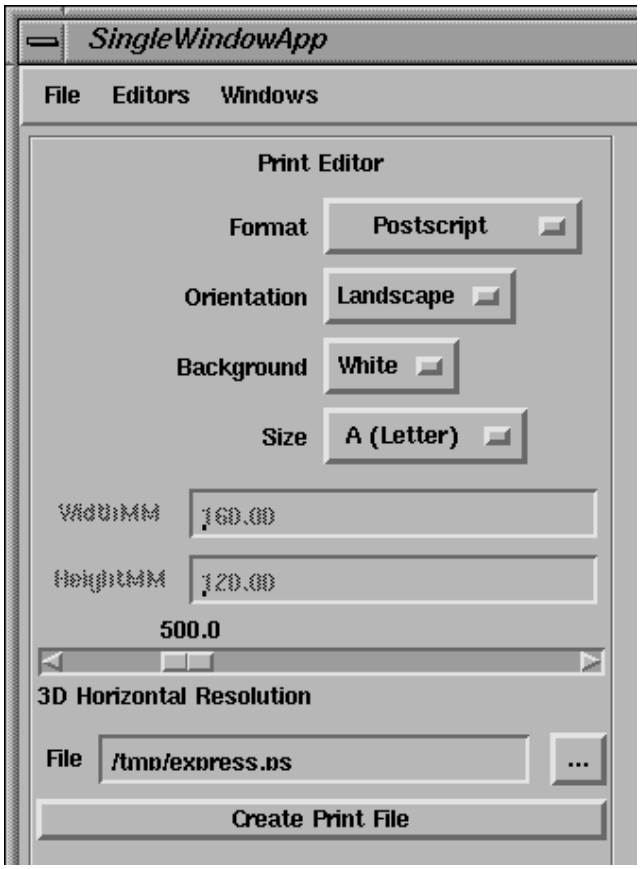
- Alter the size of glyph by changing the **Glyph scale** value.
- Try altering some of the other parameters in the `Advector` module's control panel to see the effect on the demonstration.

6.3 Other exercises

Try out some of the other examples in the Examples library. Use the module documentation to help you understand their functions.

7 Appendix I

7.1 Obtaining hardcopy from AVS/Express



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

- Alter the parameters to those shown above.
- Click “Create Print File”. This will create a postscript file called `/tmp/express.ps`.
- Go to a shell window and execute the appropriate system command to create the postscript file.

7.2 Customising AVS/Express

You can include a number of command line options when you enter the AVS/Express start-up command “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

7.3 Journaling in AVS/Express

Journaling lets you record a set of operations you perform on an object and replay them. AVS/Express records these operations as V statements and commands on a journal file.

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.

8 Appendix II

8.1 Solutions to chapter 3: field file exercises

8.1.1 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
```

8.1.2 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
```

8.1.3 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
```

8.2 Solutions to chapter 3: 2D data exercises

8.2.1 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
```

8.2.2 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
```

8.2.3 Field descriptor for Exercise 3: `hslice.fld`

```
# AVS field file
# Hydrogen
ndim=2
dim1=64
dim2=64
nspace=2
veclen=1
data=byte
field=uniform
variable 1 file=/usr/express/auxdata/hslice.dat filetype=binary skip=0 stride=1
```

8.3 Solutions to chapter 4: 3D data exercises

8.3.1 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/auxdata/fin.dat filetype=ascii skip=0 offset=0 -
   stride=8
variable 2 file=/usr/express/auxdata/fin.dat filetype=ascii skip=0 offset=1 -
   stride=8
variable 3 file=/usr/express/auxdata/fin.dat filetype=ascii skip=0 offset=2 -
   stride=8
variable 4 file=/usr/express/auxdata/fin.dat filetype=ascii skip=0 offset=3 -
   stride=8
variable 5 file=/usr/express/auxdata/fin.dat filetype=ascii skip=0 offset=4 -
   stride=8
coord 1 file=/usr/express/auxdata/fin.dat filetype=ascii skip=0 offset=5 -
   stride=8
coord 2 file=/usr/express/auxdata/fin.dat filetype=ascii skip=0 offset=6 -
   stride=8
coord 3 file=/usr/express/auxdata/fin.dat filetype=ascii skip=0 offset=7 -
   stride=8
```

8.3.2 Field descripters for Exercise 2

8.3.2.1 Scalar data

```
# AVS field file
# Climate Modelling - scalar data
ndim = 3
dim1 = 49
dim2 = 12
dim3 = 40
nspace = 3
veclen = 2
data = float
field = irregular
label = temperature specific humidity
variable 1 file=/usr/express/auxdata/temp.dat filetype=binary skip=16 offset=0
  stride=1
variable 2 file=/usr/express/auxdata/humid.dat filetype=binary skip=16 offset=0
  stride=1
coord 1 file=/usr/express/auxdata/ccm1.sph filetype=ascii skip=1 offset=0
  stride=1
coord 2 file=/usr/express/auxdata/ccm1.sph filetype=ascii skip=2941 offset=0
  stride=1
coord 3 file=/usr/express/auxdata/ccm1.sph filetype=ascii skip=5881 offset=0
  stride=1
```

8.3.2.2 Vector Data

```
# AVS field file
# Climate Modelling - vector data
ndim = 3
dim1 = 49
dim2 = 12
dim3 = 40
nspace = 3
veclen = 3
data = float
field = irregular
label = u-comp v-comp w-comp
variable 1 file=/usr/express/auxdata/bi.u.49 filetype=binary skip=16 offset=0
  stride=1
variable 2 file=/usr/express/auxdata/bi.w.49 filetype=binary skip=16 offset=0
  stride=1
variable 3 file=/usr/express/auxdata/bi.v.49 filetype=binary skip=16 offset=0
  stride=1
coord 1 file=/usr/express/auxdata/ccm1.rect filetype=ascii skip=1 offset=0
  stride=1
coord 2 file=/usr/express/auxdata/ccm1.rect filetype=ascii skip=2941 offset=0
  stride=1
coord 3 file=/usr/express/auxdata/ccm1.rect filetype=ascii skip=5881 offset=0
  stride=1
```

8.3.2.3 World Map

```
# AVS field file
# World Map
ndim = 3
dim1 = 49
dim2 = 12
dim3 = 40
nspace = 3
veclen = 1
data = float
field = irregular
variable 1 file=/usr/express/auxdata/world.asc filetype=ascii skip=1 offset=0
  stride=1
coord 1 file=/usr/express/auxdata/ccm1.rect filetype=ascii skip=1 offset=0
  stride=1
coord 2 file=/usr/express/auxdata/ccm1.rect filetype=ascii skip=2941 offset=0
  stride=1
coord 3 file=/usr/express/auxdata/ccm1.rect filetype=ascii skip=5881 offset=0
  stride=1
```

(Hint: For coordinates there are 49x12x40 values with 8 values on each line.)