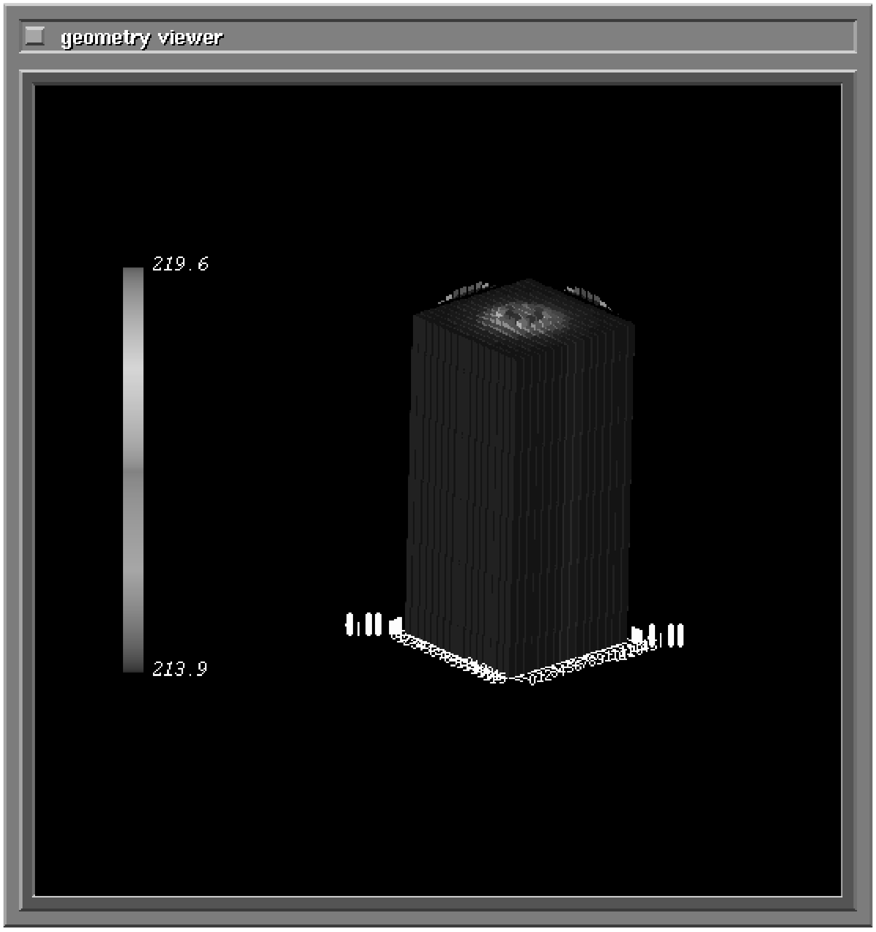
# Changing the downsizing

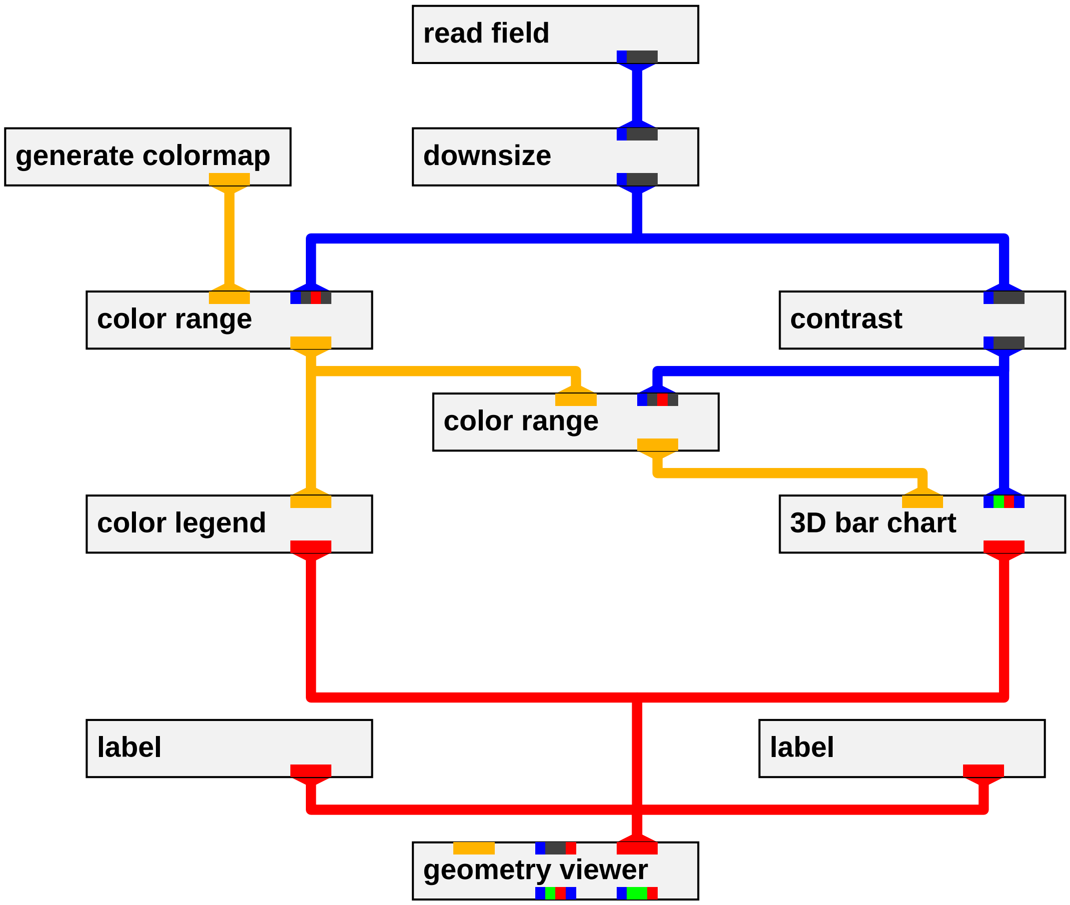



## Changing the downsizing

The image displayed is based on a subset of the imported data as `downsize` samples the data, taking every n'th element in the dataset.

Select the `downsize` parameters and change the value to see what happens. N.B., the larger the dataset the longer the network takes to process.

# Performing a linear transform

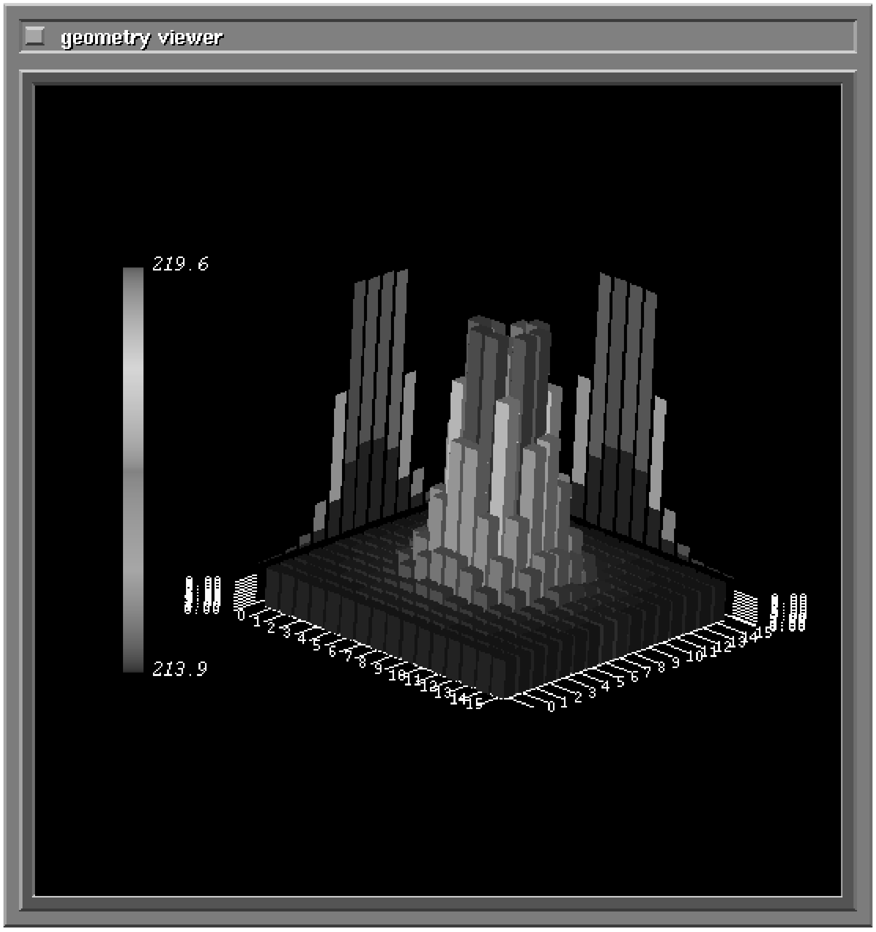



## Performing a linear transform

You may have noticed that the differences in the temperatures of the dataset is concentrated around 213.9 C and 219.6 C. This is shown by the legend which has as its input a colormap that is tailored to the datasets range by `color_range`.

Hence, we need to concentrate on this range and this is done using a module which performs a linear transformation such as `contrast`. Amend the network as above, the module `contrast` is a filter, and inlude another `color_range` module.

# Performing a linear transform (2)

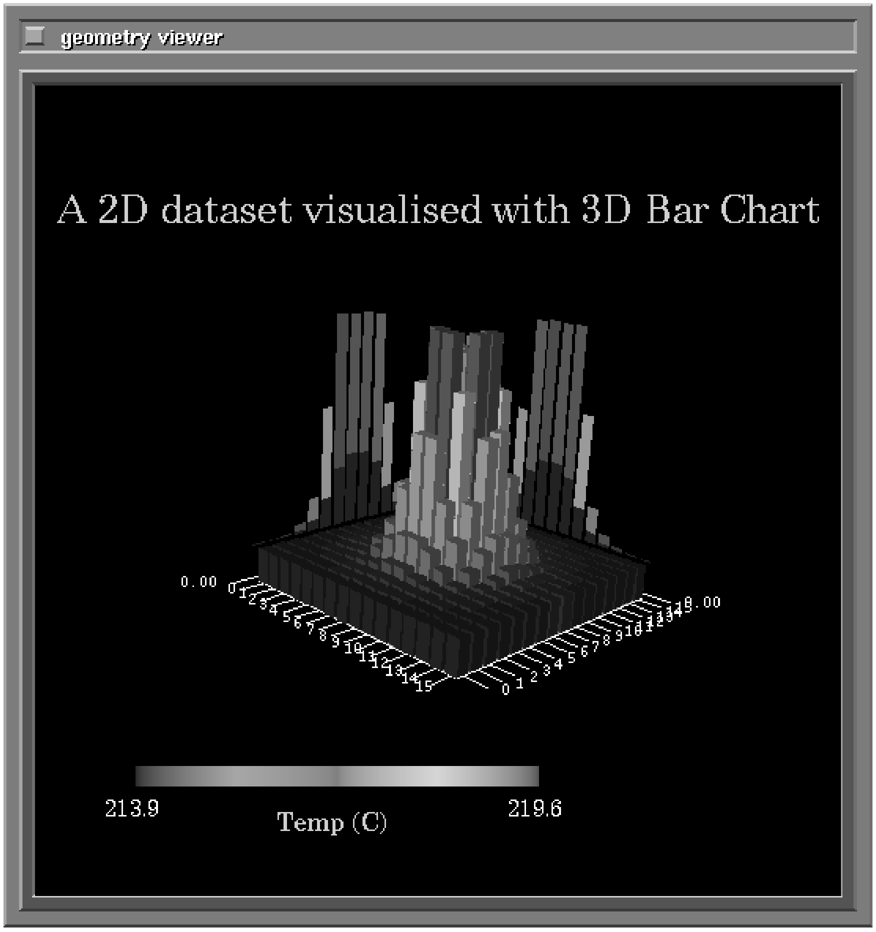



## Performing a linear transform (2)

Select the `contrast` parameters and change the values to read

- cont_in_min: 213
- cont_in_max: 220
- cont_out_min: 0
- cont_out_max: 100

# Enhancing the picture




## Enhancing the picture

1. Select the `3D_bar_chart` parameters and change the 'tic value' to 0.
2. Select the `color_legend` parameters, opt for 'horizontal' legend control. Re-position and alter the label font and height.
3. Select one of the `label` parameters, and enter 'Temp (C)' in the title string box. Re-position and alter the label font and height.
4. Select the other `label` parameters, and enter 'A 2D dataset visualised with 3D Bar Chart' in the title string box. Re-position and alter the label font.
5. Select the geometry_viewer's parameters and opt for 'Cameras' then choose 'Perspective' from the menu.

# Importing the `hslice.dat` data

```
# AVS field file
#
ndim = ?
dim1 = ???
dim2 = ???
nspace = ???
veclen = ???
data   = ???
field  = ???
variable 1 file=/usr/avs/auxdata/hslice.dat filetype=??? skip=??? stride=???
```

## Importing the `hslice.dat` data

- Create a file called `hslice.fld` to describe the data contained in `hslice.dat`. N.B., because the two data files are similar you should copy `mri.fld` to `hslice.fld`.
- Use the template description above and the information at the start of this chapter, describing the `hslice.dat` data, to complete the specification.
- Construct a network as previously using `field_to_mesh`.
- To import the data into the network select the file `hslice.fld` you have just created using the `read_field` file browser. If a picture similar to the one above appears then you have been successful.
- If you are having difficulties try connecting the module `print_field`: Data Output to the `read_field` module to analyse the field data which is being imported.

# 4 Visualising 3D arrays of data

## Description of the Data

The data is a downsized version of the AVS sample dataset `bluntfin.fld` which shows the flow over a blunt fin.

The data file is called `/usr/avs/auxdata/fin.dat` and is an ASCII file which represents an $10 \times 8 \times 8$ 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
```

# Importing the `fin.dat` data

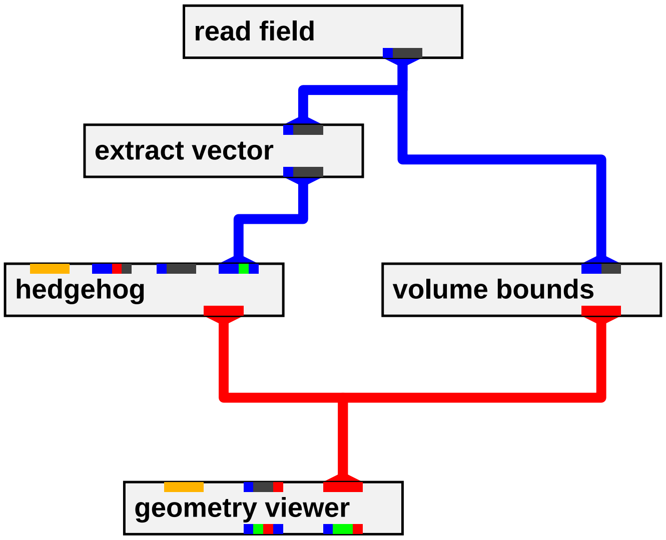
```
# AVS field file
#
ndim = ???
dim1=???
dim2=???
dim3=???
nspace = ???
veclen = ???
data = ???
field = ???
label = ??? ??? ??? ??? ???
coord 1 file= ??? filetype=??? skip=??? offset=??? stride=???
coord 2 file= ??? filetype=??? skip=??? offset=??? stride=???
coord 3 file= ??? filetype=??? skip=??? offset=??? stride=???

variable 1 file=??? filetype=??? skip=??? offset=??? stride=???
variable 2 file=??? filetype=??? skip=??? offset=??? stride=???
variable 3 file=??? filetype=??? skip=??? offset=??? stride=???
variable 4 file=??? filetype=??? skip=??? offset=??? stride=???
variable 5 file=??? filetype=??? skip=??? offset=??? stride=???
```

## Importing the `fin.dat` data

- Create a file called `fin.fld` to describe the data contained in `fin.dat`.
- Use the template description above and the information at the start of this booklet, describing the `fin.dat` data, to complete the specification.

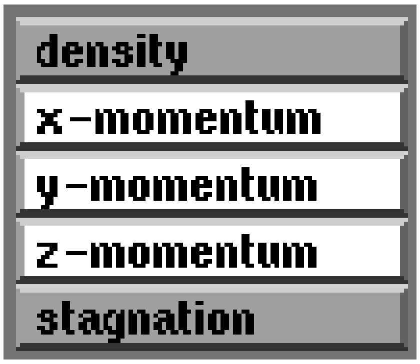
# Displaying the vector data




## Displaying the vector data

- Create the above network using the following modules:
  - `read_field`: Data Input
  - `extract_vector`: Filters
  - `volume_bounds`: Mappers
  - `hedgehog`: Mappers
  - `geometry_viewer`: Data Output
- Select the `read_field` module's control panel and select the file `fin.fld` to import the data.

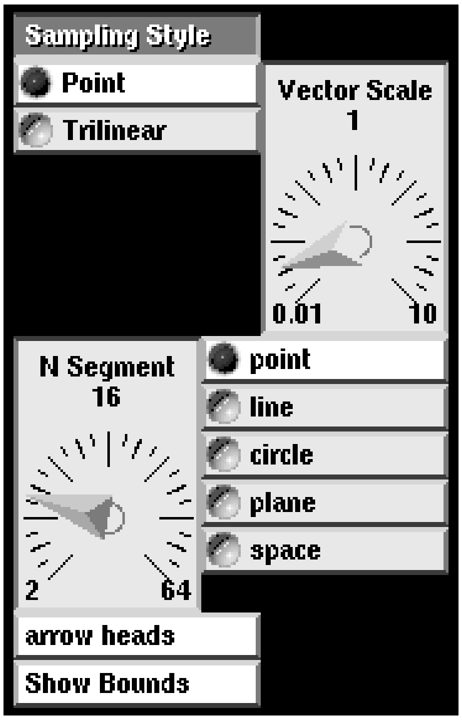
# Extracting the vector data




## Extracting the vector data

- Select the control panel for the `extract_vector` module and make active the three data components x-momentum,y-momentum and z-momentum. (see above)
- 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 network.

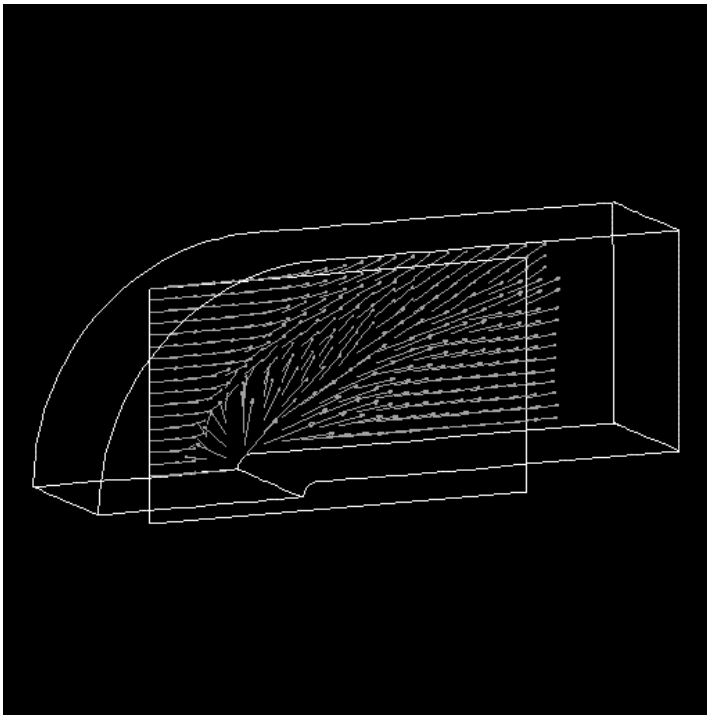
# Using the `hedgehog` module




## Using the `hedgehog` module

- Select the **`hedgehog`** modules control panel.
- Click on the button labelled “line” to change the probe type.
- Try “circle” and “plane” leaving the probe type as “plane”.
- Try altering some of the other parameters using the module help to understand the effect.

# Moving the `hedgehog` probe



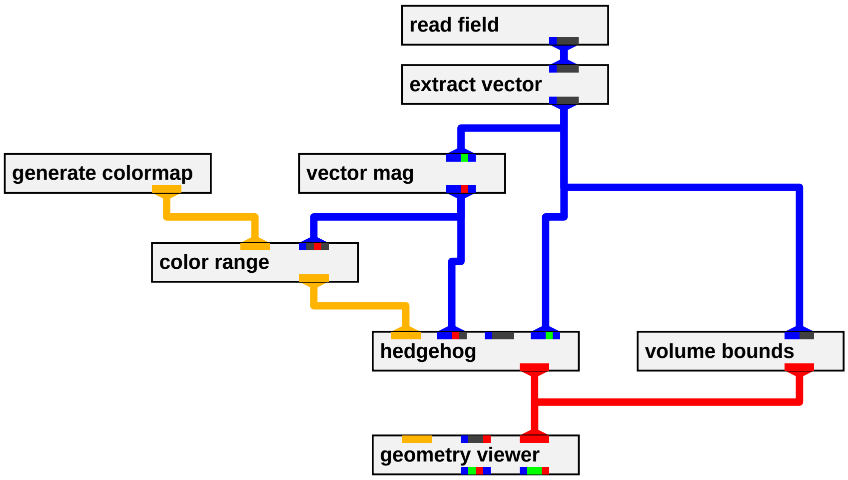
## Moving the `hedgehog` probe

- Select the `geometry_viewer` modules control panel.
- Place the cursor in the small window representation of the scene in the `geometry_viewer` control panel and press the **left** mouse button to cycle through the list of objects in the scene.

  Choose the probe by selecting the object called `hedgehog.x`; where `x` is a single integer.
- Now using the mouse in the `geometry_viewer`'s main window you can scale, rotate and translate the probe to a desired position in the dataset.

  Note that the vector representation changes as the probe moves and samples different data elements.

# Colouring the arrows




## Colouring the arrows

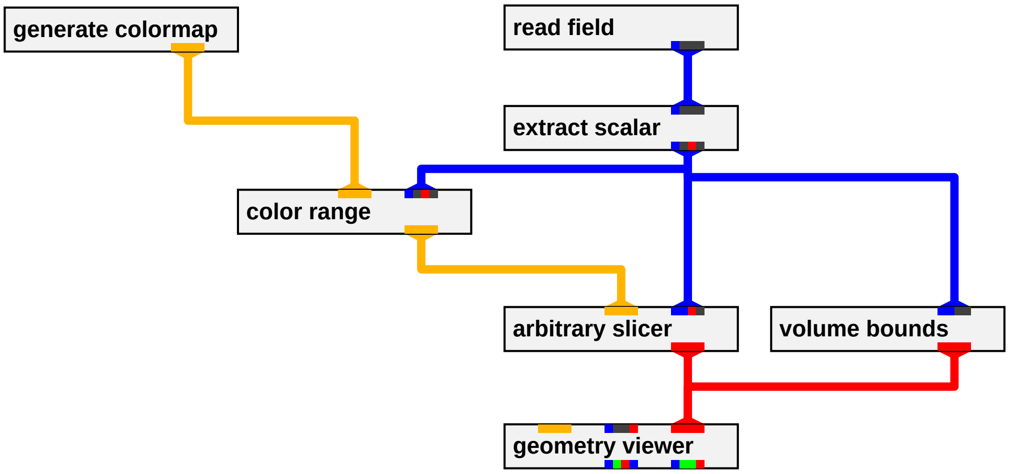
- Add the following modules to create the network shown above:
  - generate_colormap (data input)
  - color_range (data input)
  - vector_mag (filter)

  N.B., Click the "OK" button on the pop-up window which appears to ignore the error report. This message will always appear when you connect up modules to these optional ports on the hedgehog module as a reminder to also connect the other optional port.

- The vector_mag module calculates a 3D field containing the vector magnitude at each data element and uses this to colour the arrows according to the magnitude.

  The color_range module restricts the colourmap from generate_colormap to the minimum and maximum extent of the data.

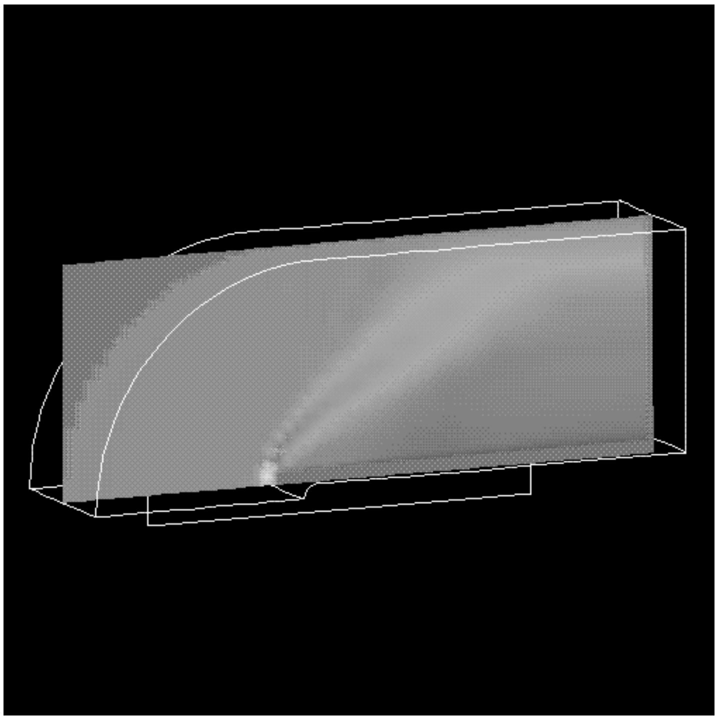
# Visualising the scalar data




## Visualising the scalar data

- Change the network to create the above network:
  - `arbitrary_slice`: Mappers
  - `extract_scalar`: Filters
- Select the control panel for the `extract_scalar` module and make active the data component density.
- This module will now pass on a 3D field with a scalar data element consisting of density to the other modules in the AVS network.

# Using the `arbitrary_slice` module



## Using the `arbitrary_slice` module

- The `geometry_viewer` should now show something similar to the above figure.

  If the `hedgehog` probe is still visible in the `geometry_viewer`'s window then use the `geometry_viewer` control panel to select and delete the probe.

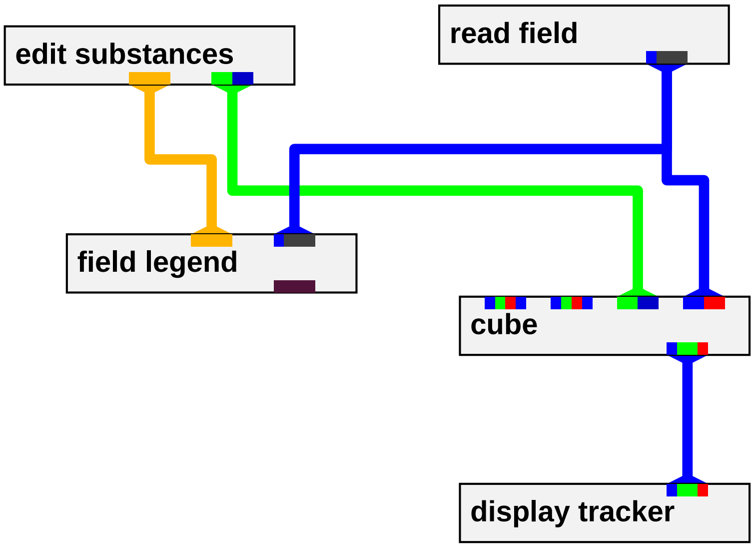
- Use the `arbitrary_slice` modules control panel to change the slice definition and position.

- Try extracting the other scalar components using the `extract_scalar` module.

**If you finish early**

Investigate using the `streamlines` module and try to extract a 2D slice of the data and use `colorizer` to display the data on this 2D slice. Hint: use `orthogonal_slicer`.

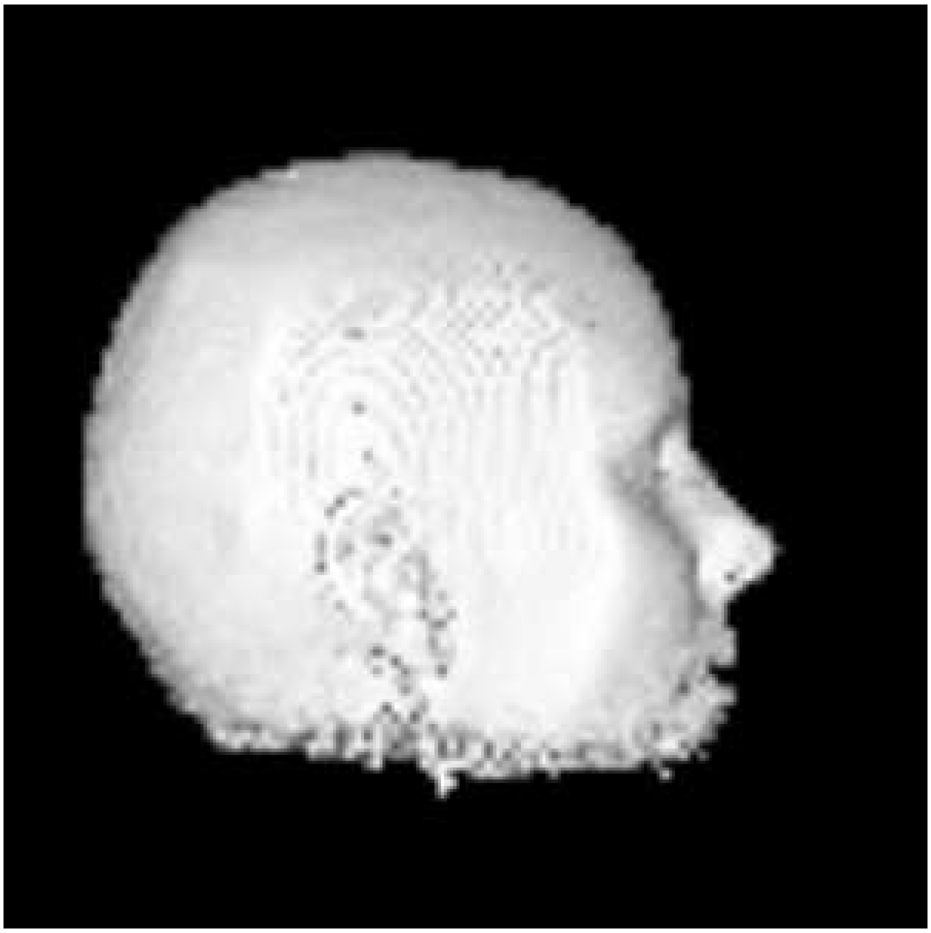
# Displaying the `head.fld` data using `cube`




## Displaying the `head.fld` data using `cube`

- Create the above network using the following modules:
  - `read_field`: Data Input
  - `cube`: Mappers
  - `edit_substances`: Data Input
  - `field_legend`: Mappers
  - `display_tracker`: Data Output
- Select the `read_field` module's control panel and select the file `head.fld` to import the data.

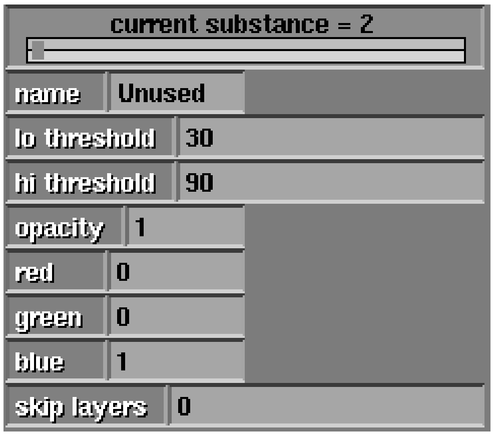
# Normalizing the picture



## Normalizing the picture

1. Select the `display_tracker` parameters and set 'Scale' to 3, this increases the size of `display_tracker`.
2. Select the `cube` parameters and choose 'ray cast' of the four volume rendering techniques.
3. The head is not yet visible, it can only be seen from the opposite side. Therefore, with the middle mouse button click in the display of `display_tracker` and while holding down the button rotate the newly appearing cube until the '-z' face appears. The head should now be visible.
4. The picture's resolution can be improved by
   - Opting for 'trilinear' in the `cube` parameters
   - Changing the number of pixels in the display from 64 to 100 in the `cube` parameters
   - Select the `display_tracker` parameters and choose 'interpolate'.

# Using edit_substances




## Using edit_substances

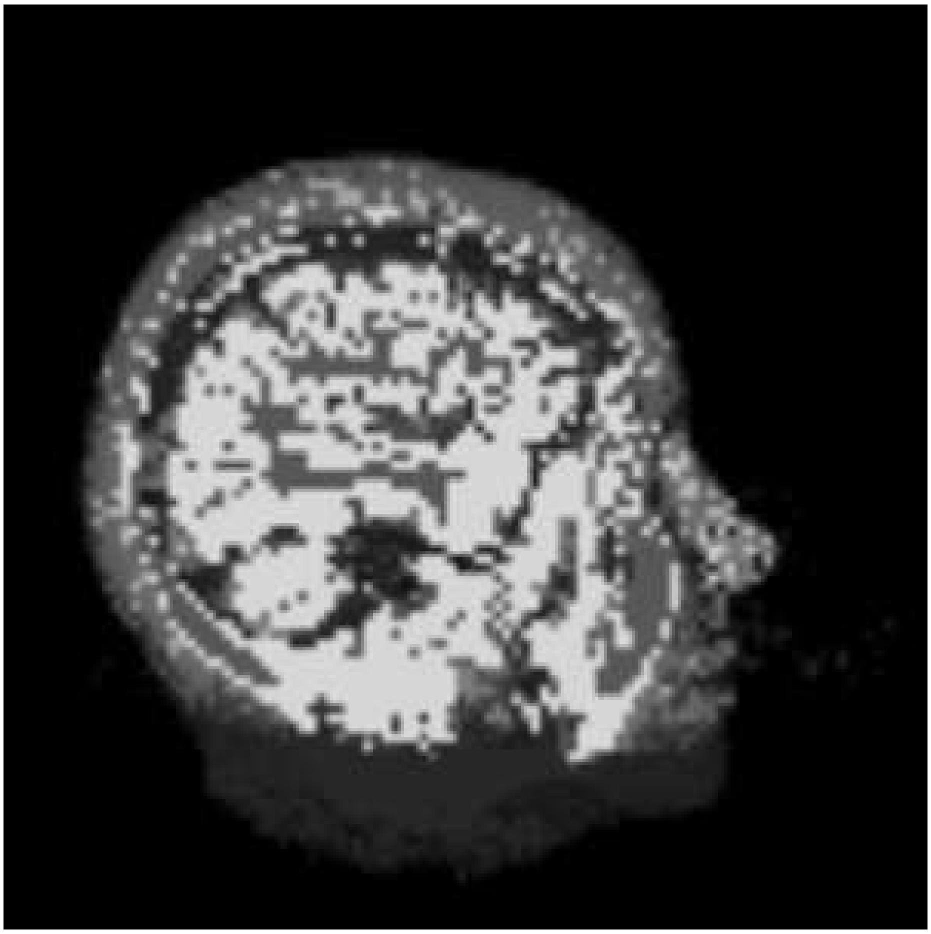
To create a substance table for cube and 'colour in' certain features of the head dataset. The module `edit_substances` must be amended and the alterations are visually displayed by the `field_legend`.

Select the `edit_substances` parameters then

1. Change the 'current substance' to 2 (as shown above)
   - lo threshold: 50
   - hi threshold: 120
   - red: 0, green: 0, blue: 1
2. Change the 'current substance' to 3
   - lo threshold: 120
   - hi threshold: 220
   - red: 0, green: 1, blue: 0
3. Change the 'current substance' to 4
   - lo threshold: 220
   - hi threshold: 255
   - red: 1, green: 0, blue: 0

N.B., a certain amount of 'noise' can be found around the dataset.

# Viewing the interior

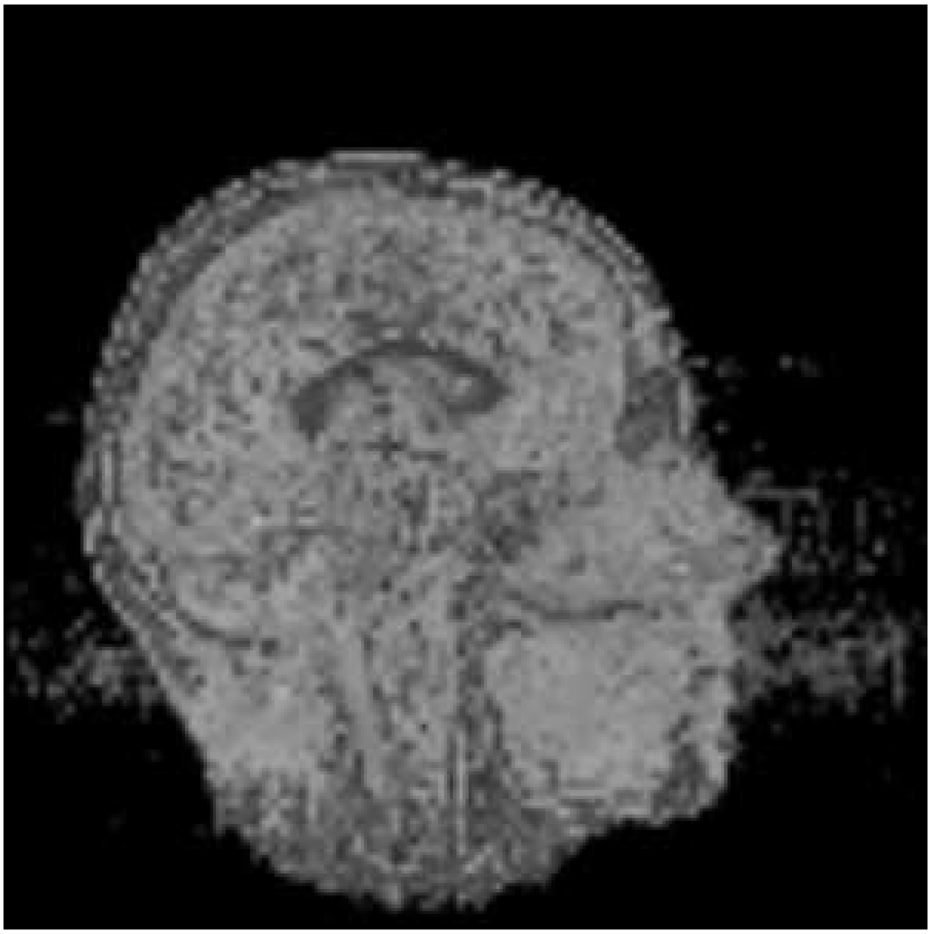


## Viewing the interior

Select the `cube` parameters and gradually increase the value of the slider 'distance', to reveal inside the head. Notice certain features, such as the brain, are outlined.

Also, select the `edit_substances` module and change the 'current substance' to 2. Find the 'opacity' type-in box and change the value from 1 to 0.5, and observe the effect. Try this with the other substances.

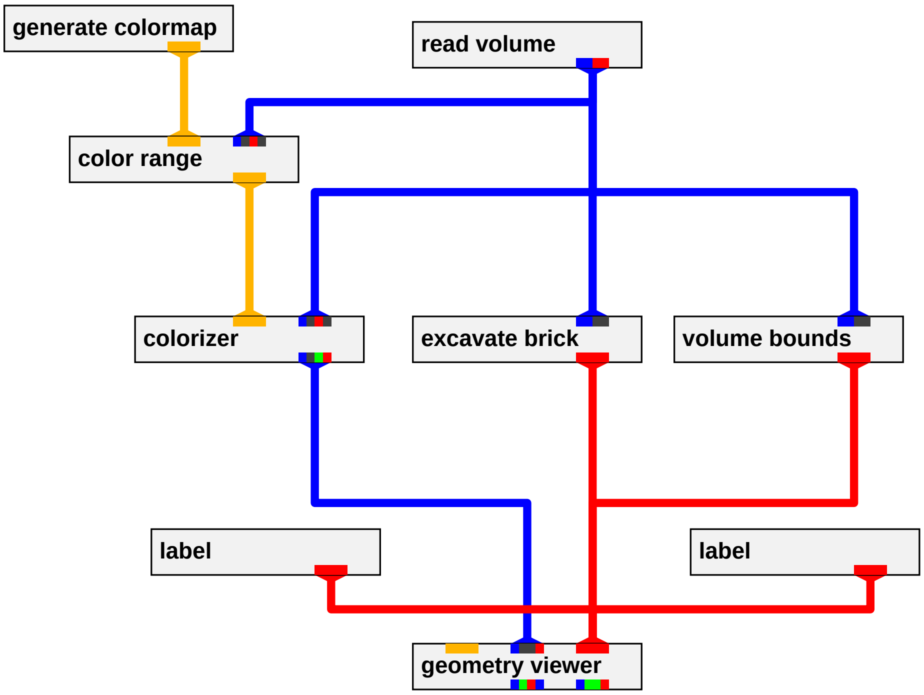
# Using edit_substances (2)



## Using edit_substances (2)

Select the `edit_substances` parameters and using the file browser opt for ‘colours.sub’ in the path ‘usr/avs/auxdata’, then click on ‘read file’. This is a dataset of a head, volume rendered using a ray cast technique.

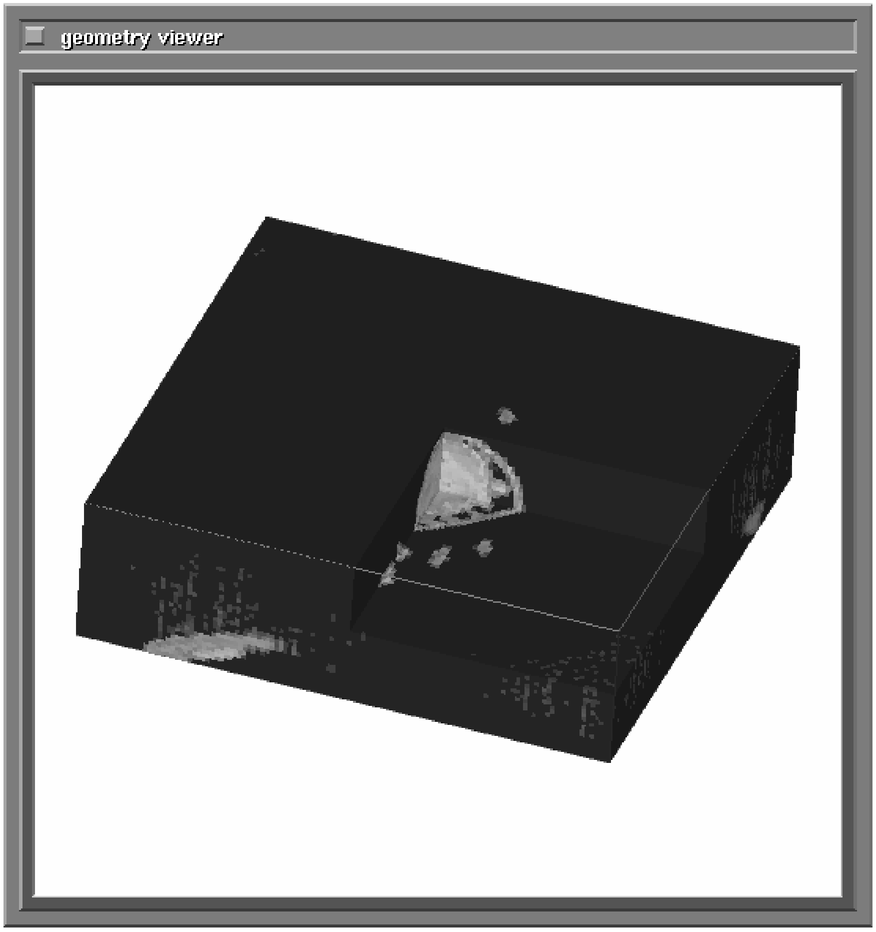
# Displaying the `lobster.dat` data using `excavate_brick`




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

- Create the above network using the following modules:
  - `read_volume`: Data Input
  - `generate_colormap`: Data Input
  - `color_range`: Data Input
  - `colorizer`: Filters
  - `excavate_brick`: Mappers
  - `volume_bounds`: Mappers
  - `label`: Data Input
  - `geometry_viewer`: Data Output
- Select the `read_volume` module's control panel and select the file `lobster.dat` to import the data.

# Normalizing the picture

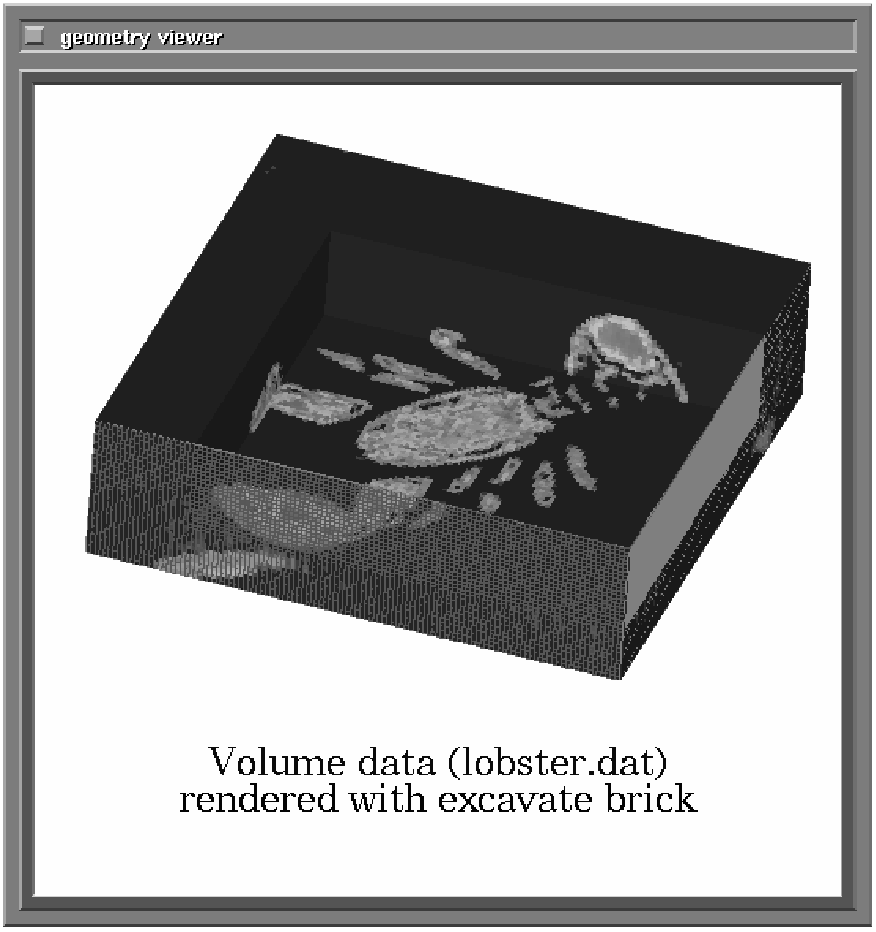



## Normalizing the picture

Select the geometry_viewers parameters and using the mouse reduce the image and rotate. Choosing the 'bounding box' may help in the transformation.

N.B., when transforming the image ensure 'top' is always selected in the `geometry_viewer` parameters. By clicking on the dimple in the box above the small window, a list appears of all the objects in the `geometry_viewer`, and 'top' can then be selected.

# Using excavate_brick



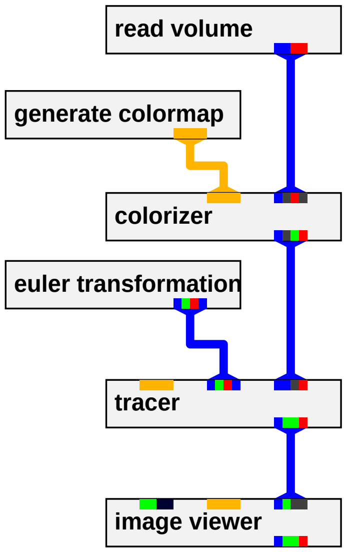

## Using excavate_brick

The image displayed is of a lobster encased in a brick. Select the `excavate_brick` parameters and change the values to see the effect.

The `volume_bounds` module allows the sides of the brick to be drawn.

Select the `label` parameters to enter and re-position the text.

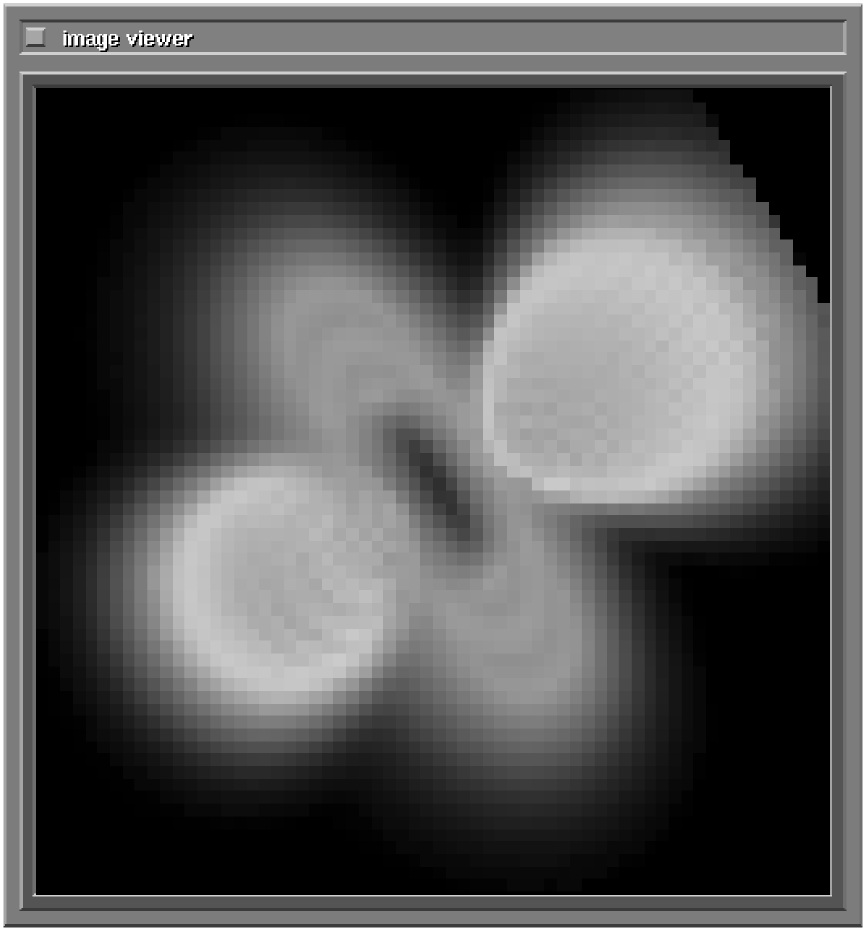
# Displaying the `hydrogen.dat` data using `tracer`




## Displaying the `hydrogen.dat` data using `tracer`

- Create the above network using the following modules:
  - `read_volume`: Data Input
  - `generate_colormap`: Data Input
  - `colorizer`: Filters
  - `tracer`: Mappers
  - `euler_transformation`: Data Input
  - `image_viewer`: Data Output
- Select the `read_volume` module's control panel and select the file `hydrogen.dat` to import the data.

# Using `tracer` and `euler_transformation`




## Using `tracer` and `euler_transformation`

- Normalise the image with the parameters of `image_viewer`.
- Change the parameters of `tracer`. Refer to the on-line module documentation to discover the effects of changing these values of the ray- traced image.
- The module `euler_transformation` allows the image to be rotated, a facility not available in the `image_viewer`. For instance, the 'rho' parameter allows rotation about an axis which would appear to come 'out' the screen

# 5 Visualising UCD Data

## Description of the Data

The module `read_ucd` imports data files in the following format:

```
<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
```

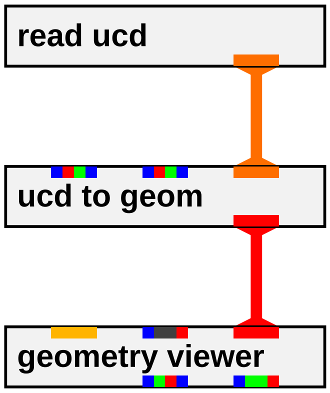
An example of this is the `hex.inp` file which defines a single hexahedral cell with stress at each node:

```
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
```

Some of the other files containing sample UCD data can be found in `/usr/avs/data/ucd`:

- `pyr.inp`: example of a pyramid shaped cell
- `tet.inp`: example of a tetrahedral shaped cell
- `bluntfin.inp`: flow of air over a fin
- `vec1.1000.inp`: velocity at each node
- `avs.inp`: AVS logo

# Displaying the UCD geometry

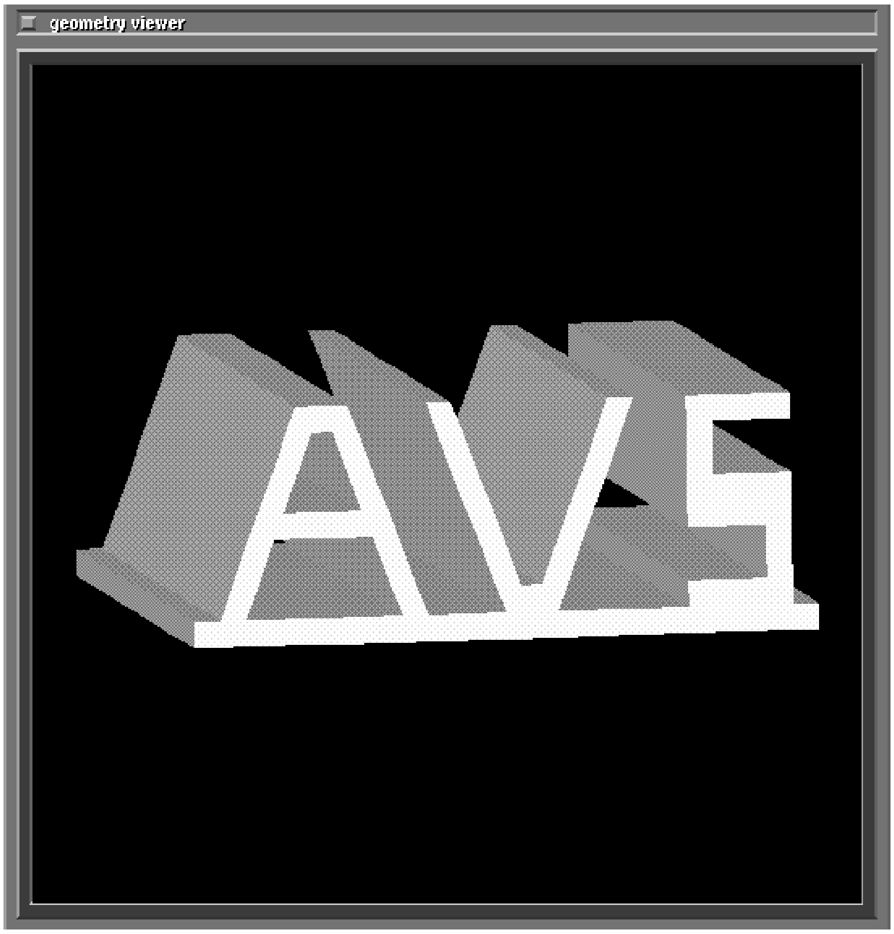



## Displaying the UCD geometry

Create the network above using the following modules:

- `read_ucd`: Data Input
- `ucd_to_geom`: Mappers
- `geometry_viewer`: Data Output

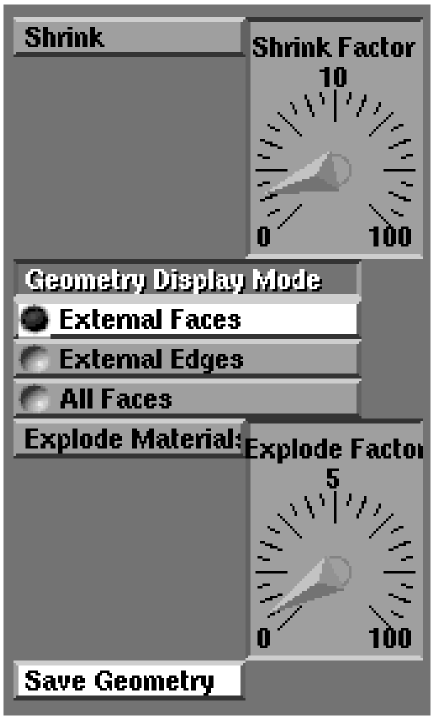
# Importing a UCD file




## Importing a UCD file

1. Select the file `hex.inp` using the `read_ucd` file browser. This file can be found in `/usr/avs/data/ucd`.
2. Using the mouse transform and rotate the object.
3. Try importing the some of the other files. The above picture shows the file `avs.inp`.

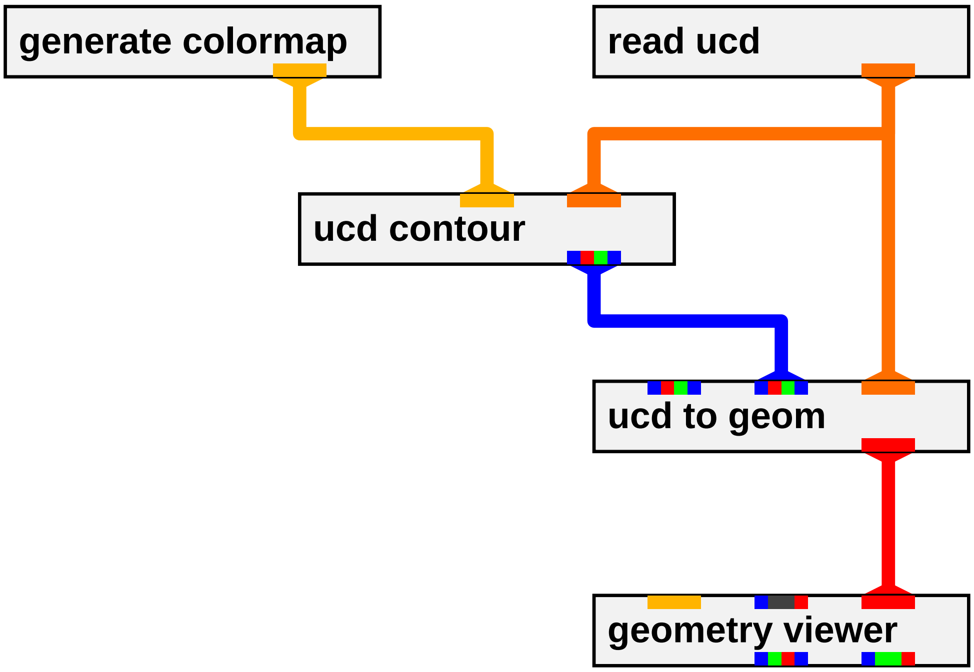
# Using the `ucd_to_geom` module




## Using the `ucd_to_geom` module

1. Select the `ucd_to_geom` modules parameters.
2. Press the "shrink" button and the cells making up the UCD structure will be shrunk by 10% illustrating how the model is constructed.
3. Alter the "Shrink Factor" dial and the cells size will change.
4. Choose "All Faces" and the inside cells are also rendered.
5. Choose "External Edges" and only certain edges are displayed.
6. Alter the "Edge Angle" dial and the number of edges displayed will increase and decrease.

# Displaying Scalar Data

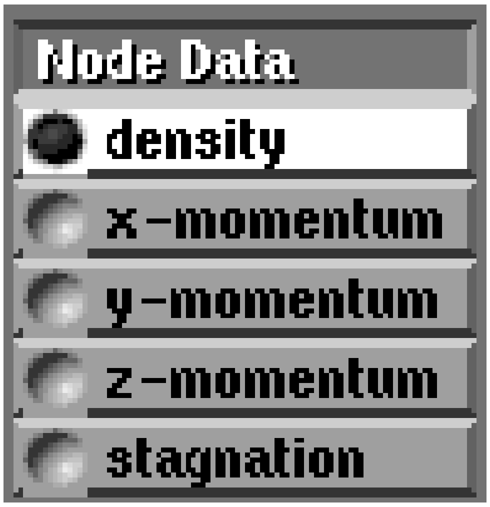



## Displaying Scalar Data

Add the `ucd_contour` module to the network as above:

- ucd_contour: Mappers
- generate_colormap: Data Input

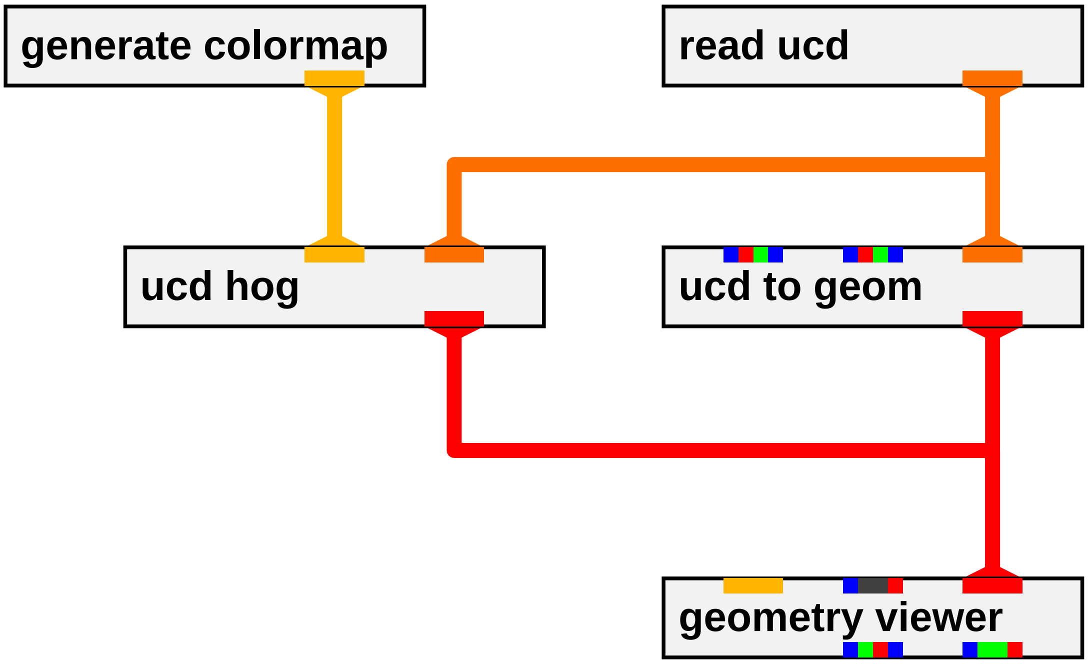
# Using the `ucd_contour` module




## Using the `ucd_contour` module

1. Select the file `bluntfin.inp` using the `read_ucd` modules file browser.
2. Select the `ucd_contour`'s parameters.
3. To change the data value which is contoured select the appropriate component from the `ucd_contour` panel.
4. Try some other data files.

# Displaying vector data

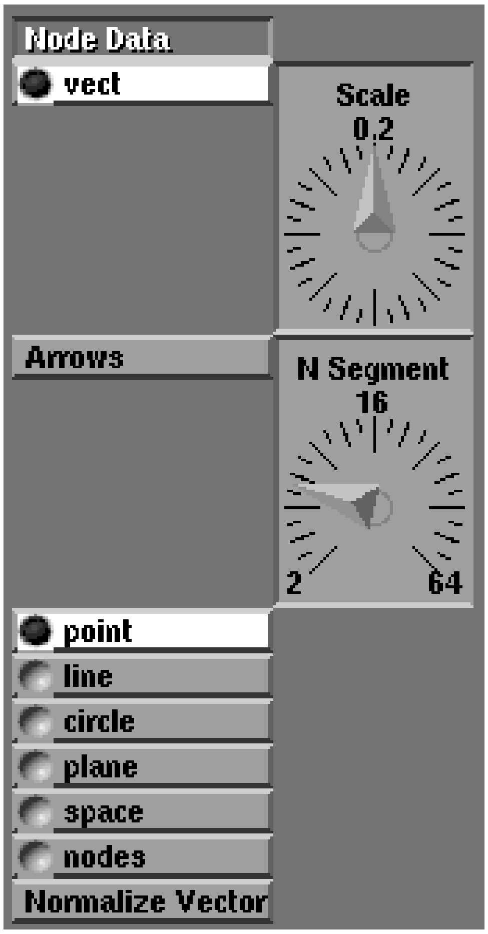



## Displaying vector data

Create the network above using the modules listed below:

- ucd_hog: Mappers

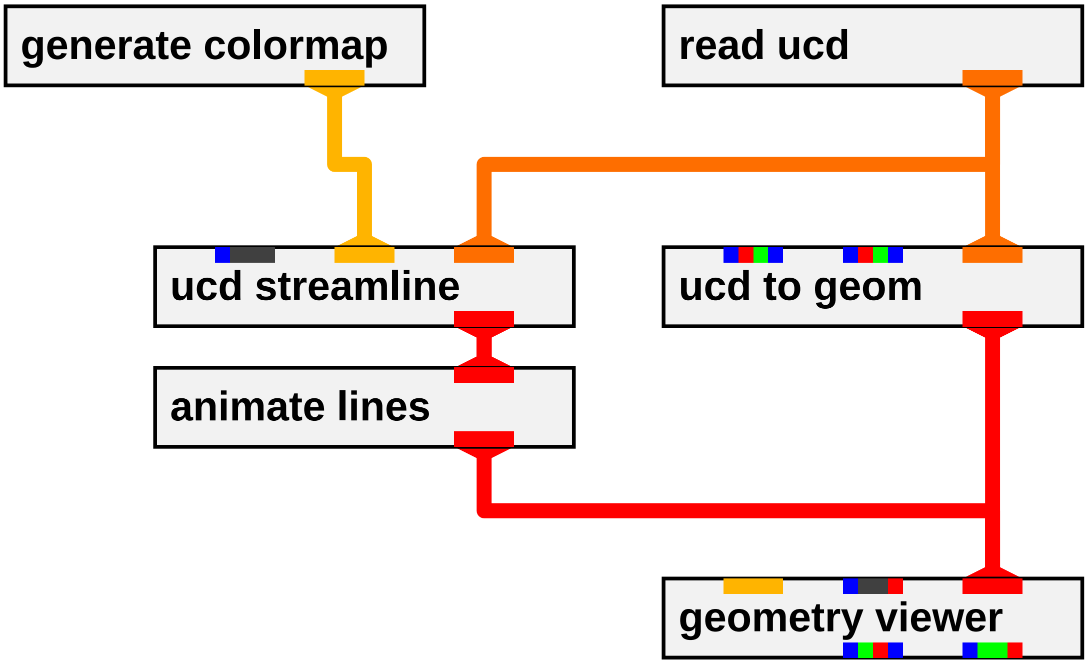
# Using the `ucd_hog` module




## Using the `ucd_hog` module

1. Select the file `vec1.1000.inp` using the file browser on the `read_ucd` module.
2. Select the `ucd_hog` parameters and change the probe type to a plane.
3. Select the probe and move it through the structure.
4. Try some of the other parameters using the help facility for more information.

# Animating vector flow

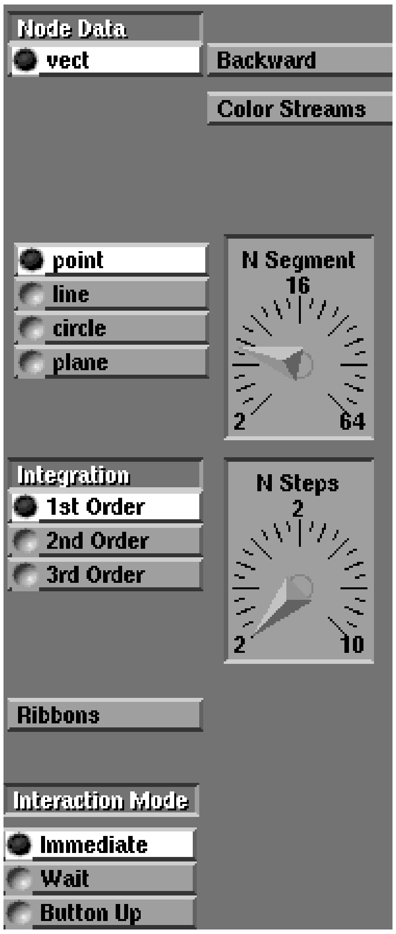



## Animating vector flow

Create the network above using the modules listed below:

- ucd_streamline: Mappers
- animate_lines: Filters

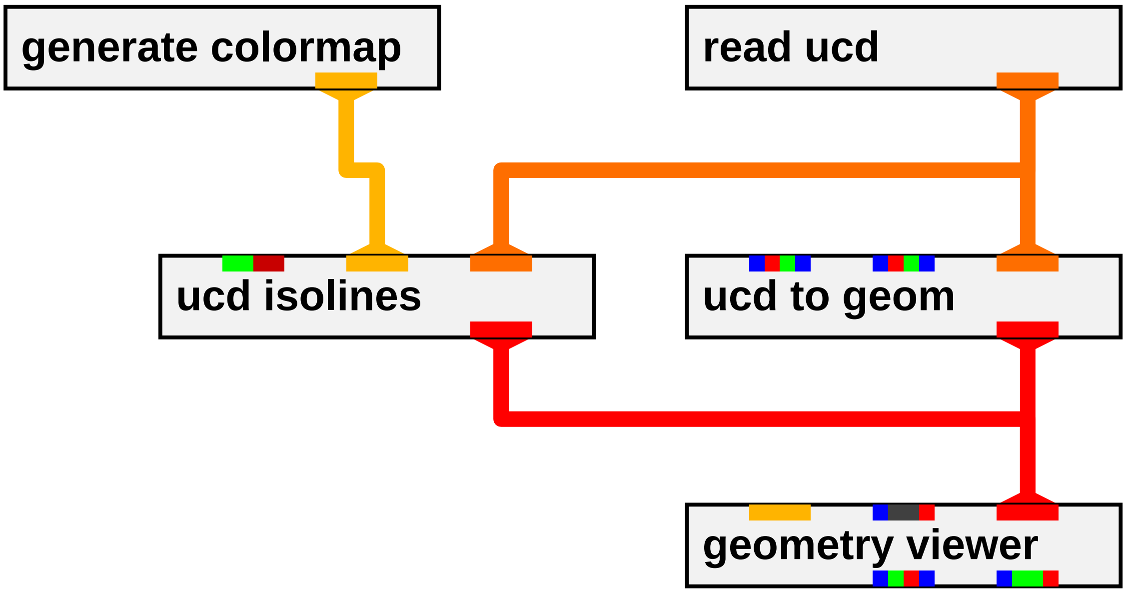
# Using the `ucd_streamline` module




## Using the `ucd_streamline` module

1. Select the `ucd_streamline`'s parameters.
2. Change the sample area by choosing the "plane" option. Do not alter any of the other parameters.
3. Select the parameters for the `animate_lines` module and click on the "animate" button. The flow should be animated on screen.

# Other exercises




## Other exercises

1. Create the network above and try some of the sample files in `/usr/avs/data/ucd`.
2. Experiment with some of the other UCD modules.
   - ucd_iso
   - ucd_print
   - ucd_crop
   - ucd_slice

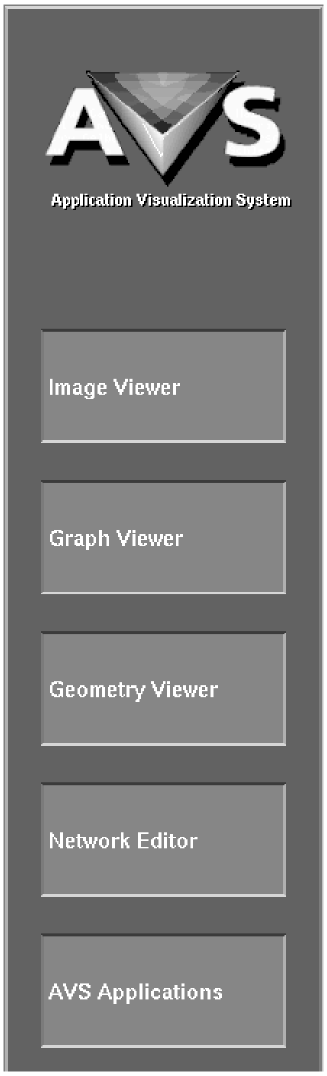
   The help facility lists some example networks for each module.

# 6 Accessing the AVS demo suite

## The particle advector module

There are numerous demonstration scripts included with AVS to illustrate the operation and use of most of the AVS modules. This exercise will illustrate the use of the demo suite to show the particle advector module as an example.

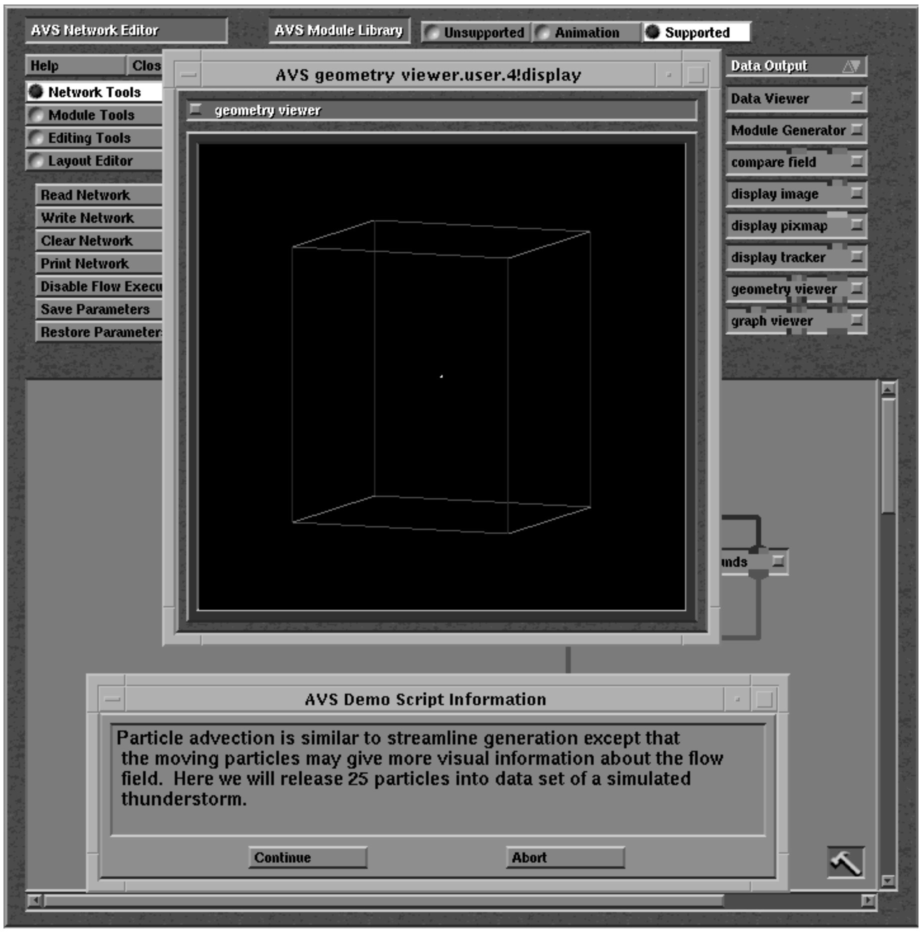
# Entering the AVS Demos




## Entering the AVS Demos

1. Start the AVS software by typing `avs`
2. Click on the panel labelled "AVS Applications" using the mouse and **left** button.
3. Click on the panel labelled "AVS Demo" using the mouse and **left** button.

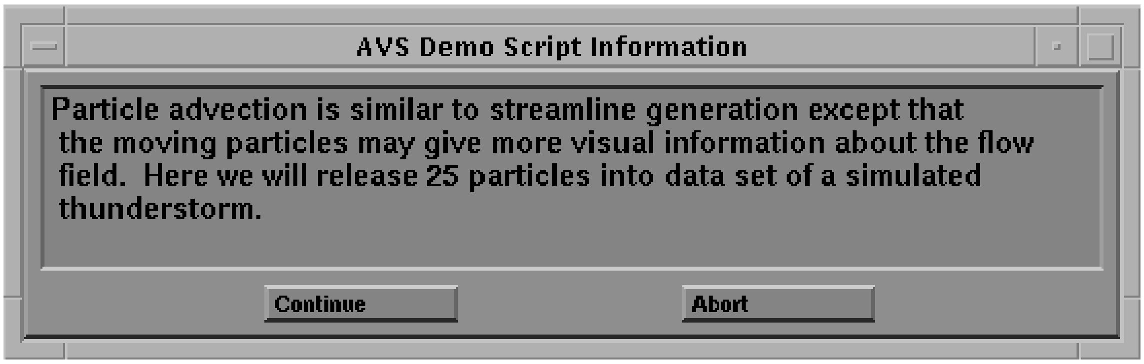
# Starting the Fluid Flow demo




## Starting the Fluid Flow demo

1. Place the cursor on the demo panel labelled "Techniques" at the top of the screen.
2. Press the **left** mouse button and while holding it down select the panel labelled "Fluid Flow" and release the mouse button.
3. Now place the cursor on the panel labelled "Fluid Flow" at the top of the screen.
4. Press the **left** mouse button and while holding it down select the panel labelled "Particle Advection" and release the mouse button.
5. The screen should appear as above.

# Executing each stage of the demo




## Executing each stage of the demo

**Read this section but do not do anything yet**

1. When the demo starts up, and before each stage in the demo, a window similar to the one above will appear. The contents of the window briefly explain what you will see happen next when you click on the button labelled “Continue” using the **left** mouse button
2. If at some point you wish to stop the demo simply click on the button labelled “Abort” using the **left** mouse button.
3. The following page explains the demo of the particle advector module. Please read this as you progress through the demo.

# The particle_advector demo


## The particle_advector demo

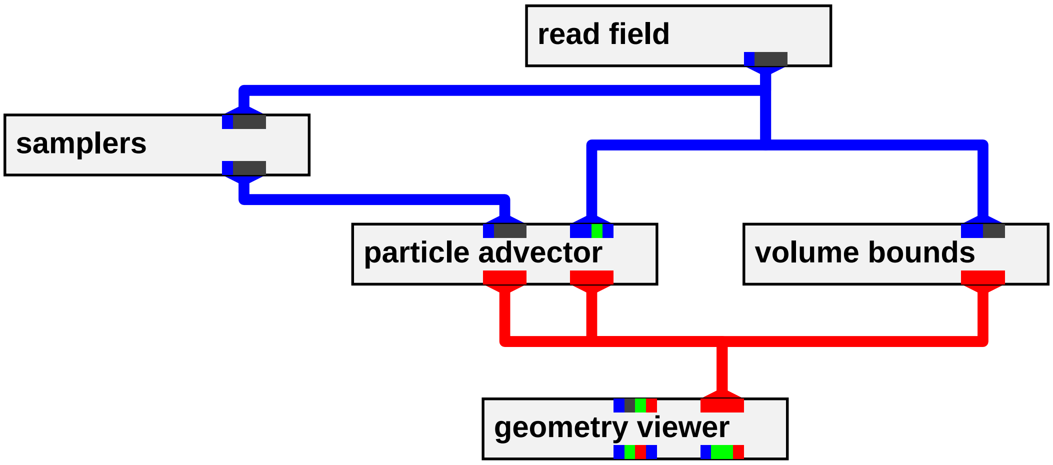
Follow the demo through by reading the explanatory comments which appear on screen and below. Only click on the button labelled “Continue” when indicated in this document. Do not change any of the parameters yourself until you have completed the full demonstration.

The demo shows the release of 25 particles into a dataset which represents a “simulated thunderstorm”. The particles are initially positioned evenly around a circular probe before they are released. The particles path is shown by tracers which are connected to each particle.

**Click the “Continue” button**

The demo has now finished, please click the “Continue” button.

# The `particle_advector` network




## The `particle_advector` network

Rearrange the network in the network editor so it appears similar to the layout shown above. The text below will explain the way the network you have just seen works.

The `read_field` module imports the vector data from a file into AVS.

The `samplers` module simply produces a number of locations which are used as the initial positions for the particles in the `particle_advector` module. There are a number of sample positions available and these are point, line, circle, plane, and space. The number of sample points can be altered by changing the dial labelled "N Segment" and the placement of the initial positions by manipulating the object in the `geometry_viewer` module.

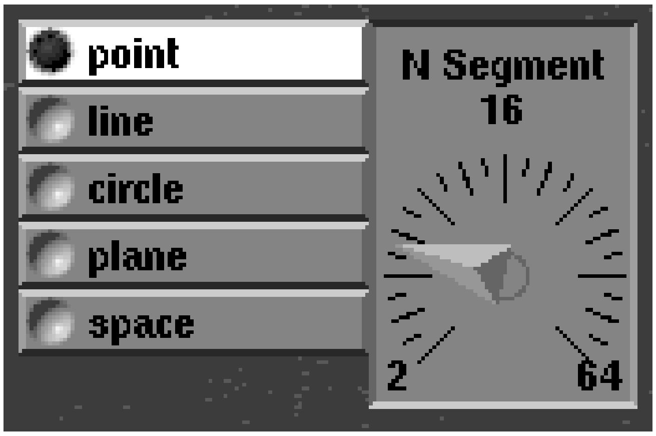
The `particle_advector` module takes as its inputs:

1. A field to define the particles initial positions. This is usually the output from the `samplers` module.
2. The vector data which is used when calculating the path of the particles.

The `volume bounds` module calculates a line boundary to represent the extent of the data. This provides feedback to the user when manipulating the starting positions as to where in the data the positions are located.

Finally both the `particle_advector` and `volume_bounds` module output geometric definitions which represent the particles and boundary definitions. These are passed to the `geometry_viewer` module which displays them in a window on the screen and allows the user to interact with these objects.

# Changing the parameters yourself




## Changing the parameters yourself

Try the performing the following actions using the network in the network editor. The figure above shows the parameters associated with the `samplers` module.

1. Change the sampler type by clicking on the buttons labelled "point", "line", "circle", "plane" and "space"
2. Alter the number of samples taken by changing the "N segment" dial
3. Move the probe through the data by selecting the `particle_advect` object in the `geometry_viewer` and transform it as you would any object.
4. Before releasing the particles you must select the "Reset Particles" button. To release the particles select the "Start Advection" button.
5. The "Stop Advection" button acts as a toggle and will only temporarily halt the advection. To completely stop the advection you must select the "Reset Particles" button before turning the "Stop Advection" toggle off again.

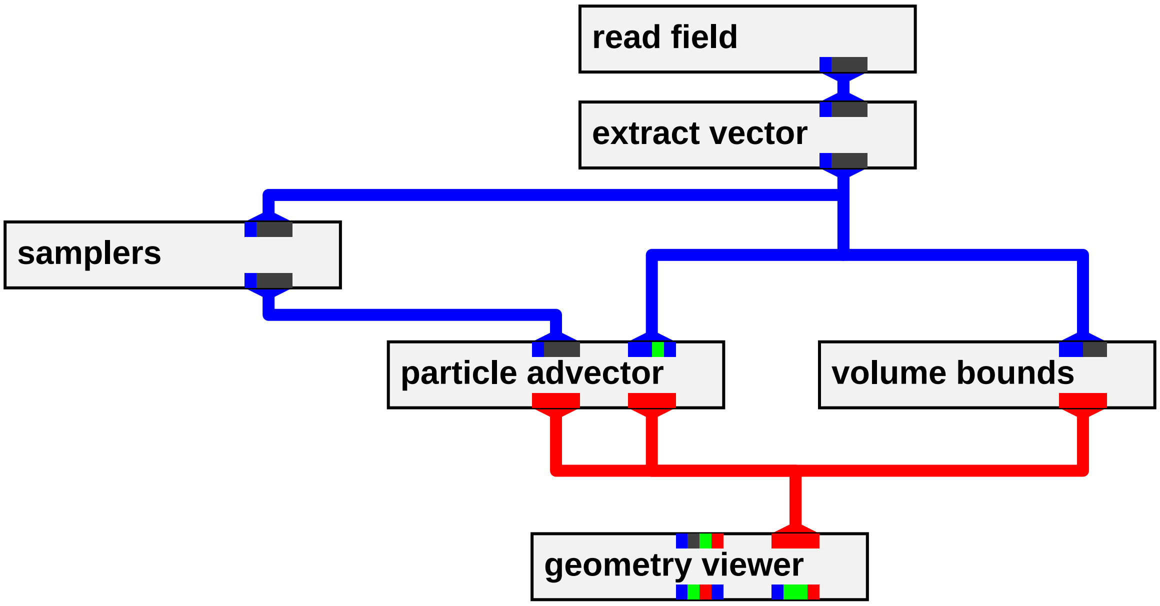
# Changing the parameters yourself


## Changing the parameters yourself

Try the performing the following actions using the network in the network editor. The panel above shows the parameters associated with the `particle_advector` module.

1. Alter the size of each particle by changing the value in the panel marked “Size”. A size of zero will change the particles to points.
2. Change the advection method from “Euler” to “Runge-Kutta” and note the difference.
3. Alter the length of the tracers using the “Tracer Length” dial and the style by selecting either “cap”, “cycle” or “add”. (Note that the “add” style should be “end”)
4. Click on the “Color” button to colour the particles according to their speed. Red particles are the fastest, blue ones have stopped.

# Advanced exercise



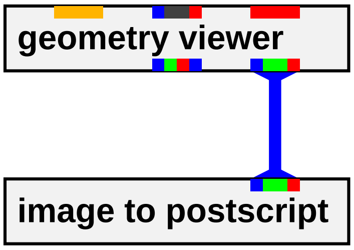

## Advanced exercise

**Only try this if you have enough time left**

1. Alter the network so it includes the `extract_vector` module (Filters).
2. Select the file "bluntfin.fld" from the file browser for the `read_field` module.
3. Select the parameters for the `extract_vector` and then click on the buttons "X momentum", "Y momentum" and "Z momentum". This constructs a vector data component from 3 scalar components. You need to do this before passing the data to the `particle_advector` and `samplers` modules.
4. Now explore the data using the techniques described in the demo.

# 7 Obtaining hardcopy from AVS

# Obtaining hardcopy from `geometry_viewer`

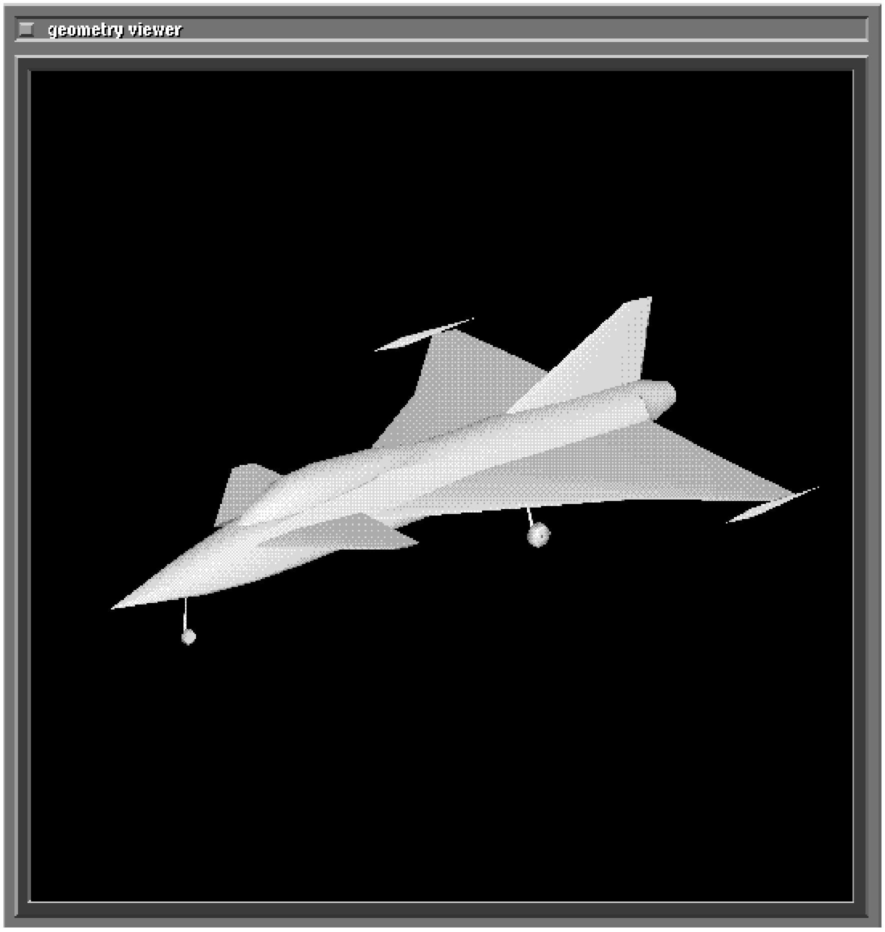



## Obtaining hardcopy from `geometry_viewer`

Create the following network using the modules listed below:

- geometry_viewer: Data Output
- image_to_postscript: Data Output

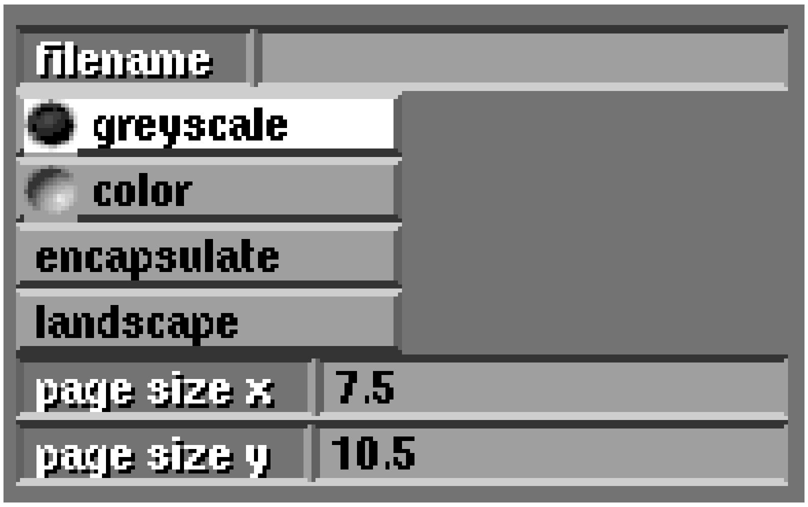
# Read an object




## Read an object

1. Select the `geometry_viewer`'s parameters.
2. Select the "Read Object" option.
3. Move to the geometry directory and click on the file called `jet.geom`.

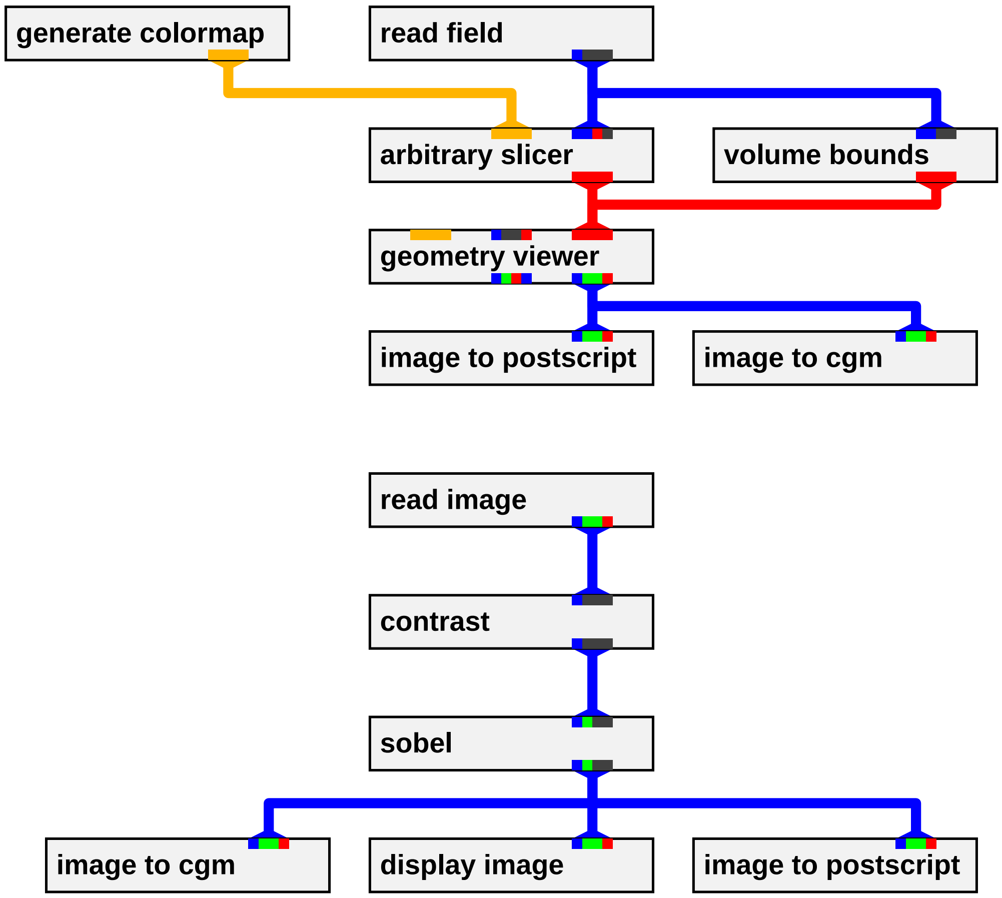
# Creating the hardcopy file




## Creating the hardcopy file

1. Select the `image_to_postscript` parameters.
2. Type in the box the name of the file you wish to create e.g., myps and AVS will generate a postscript version of the image.
3. You can also specify the orientation on the page as well as the size of the image.

# Practical use




## Practical use

1. Any network which uses the `geometry_viewer` can have the `image_to_postscript` module added to capture certain pictures (see above).
2. Any network producing images can also use the module (see above).

# 8 Customising AVS

## The `.avsrc` file

When AVS is started it looks for the file `.avsrc` in your home directory. If it cannot find this file it uses the default file `/usr/avs/runtime/avsrc` which is listed below:

```
DisplayPixmapWindow 512x512+270+20
ModuleLibraries   /usr/avs/unsupp_mods/Unsupported
                                /usr/avs/avs_library/Supported

# These are the defaults which can be reset in this file:
# NetworkDirectory  /usr/avs/networks
# DataDirectory /usr/avs/data
# NetworkWindow 960x960+290+30

# Force software rendering
NoHW 1

#VisualType DirectColor
```

## Some useful changes to your `.avsrc` file

1. Obtain a copy of the file by typing the following:

   ```
   cd
   cp /usr/avs/runtime/avsrc .avsrc
   ```

2. edit the file `.avsrc` and make the following changes.

   ```
   # NetworkDirectory  /usr/avs/networks
   ```

   Uncomment the line and change the directory to your own home directory e.g.,

   ```
   NetworkDirectory  /users/unknown
   ```

3. Change the default data directory:

   ```
   # DataDirectory /usr/avs/data
   ```

   Uncomment the line and change the directory to one of your own data directory:

   ```
   DataDirectory /usr/unknown/data
   ```

4. Change the argument to the NoHW command from a 1 to a 0.

   ```
   NoHW 0
   ```

   This will force AVS to use any special graphics hardware your machine may have.

5. When you select the "print network"command the default action is to use lpr. By adding the following line to your `.avsrc` file you can change this to send the output to your own local printer where "lp -dXXXX" is your command to print postscript.

   ```
   PrintNetwork lp -dXXXX
   ```

## AVS command line options

There are many command line options available from AVS and a few are listed below to try:

1. The `-version` option indicates which version of AVS is installed.
2. The `-network` option takes a network file as an argument and starts up AVS to automatically use this network.
3. The `-nohw` option switches on the software renderer in the geometry viewer.
4. The `-usage` option lists all the possible options.

## The AVS Command Line Interpreter (CLI)

1. Start AVS with the option `-cli` e.g., `avs -cli`.
2. AVS will start up with the prompt `avs>` in the terminal window where you started AVS.
3. At this prompt type

   ```
   avs> script -open myscript
   ```

   A file called `myscript` will be created containing an audit of any commands you have chosen using the mouse and keyboard.
4. Perform some specific task within AVS and watch the commands appear in the terminal window.
5. When you have finished recording the commands type:

   ```
   avs> script -close
   ```

6. Now exit from AVS and restart it again using `avs -cli`.
7. At the prompt type:

   ```
   avs> script -p myscript
   ```

   The script will be played through mimicking the actions you performed earlier.
8. For more help on any of the CLI commands just type `help` at the CLI prompt (`avs>`).

# A Answers to the exercises

## Field descriptor for `mri.dat`

```
# AVS field file
#
ndim=2 # number of dimensions in the field
dim1=512      # dimension of axis 1
dim2=512      # dimension of axis 2
nspace=2 # number of physical coordinates per point
veclen=1 # number of components at each point
data=byte         # portable data format
field=uniform      # field type (uniform, rectilinear, irregular)
variable 1 file=/usr/avs/auxdata/mri.dat filetype=binary skip=0 stride=1
```

## Field descriptor for `hslice.dat`

```
# AVS field file
#
ndim=2 # number of dimensions in the field
dim1=64       # dimension of axis 1
dim2=64       # dimension of axis 2
nspace=2 # number of physical coordinates per point
veclen=1 # number of components at each point
data=byte         # portable data format
field=uniform      # field type (uniform, rectilinear, irregular)
variable 1 file=/usr/avs/auxdata/hslice.dat filetype=binary skip=0 stride=1
```

## Field descriptor for `temps.dat`

```
# AVS field file
#
ndim=2 # number of dimensions in the field
dim1=31       # dimension of axis 1
dim2=31       # dimension of axis 2
nspace=2 # number of physical coordinates per point
veclen=1 # number of components at each point
data=float         # portable data format
field=uniform      # field type (uniform, rectilinear, irregular)
variable 1 file=/usr/avs/auxdata/temps.dat filetype=ascii skip=1 stride=1
```

## Field descriptor for `fin.dat`

```
# AVS field file
#
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
coord 1 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=5 stride=8
coord 2 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=6 stride=8
coord 3 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=7 stride=8

variable 1 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=0 stride=8
variable 2 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=1 stride=8
variable 3 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=2 stride=8
variable 4 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=3 stride=8
variable 5 file=/usr/avs/auxdata/fin.dat filetype=ascii skip=0 offset=4 stride=8
```