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, and `dimn` specifies the size of dimension n.
- 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 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 ... ... ... ... ...
```

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.

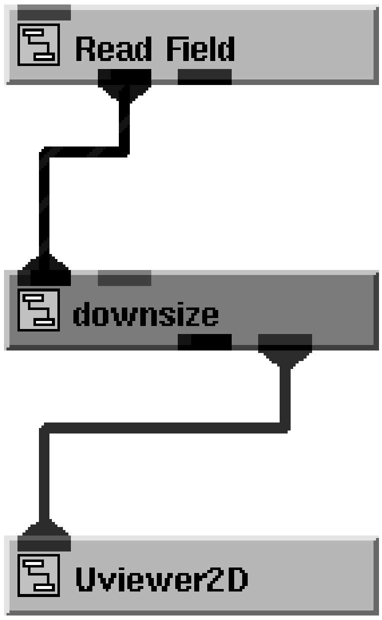
3.6.1 Importing the `mri.dat` data (1)

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.
```

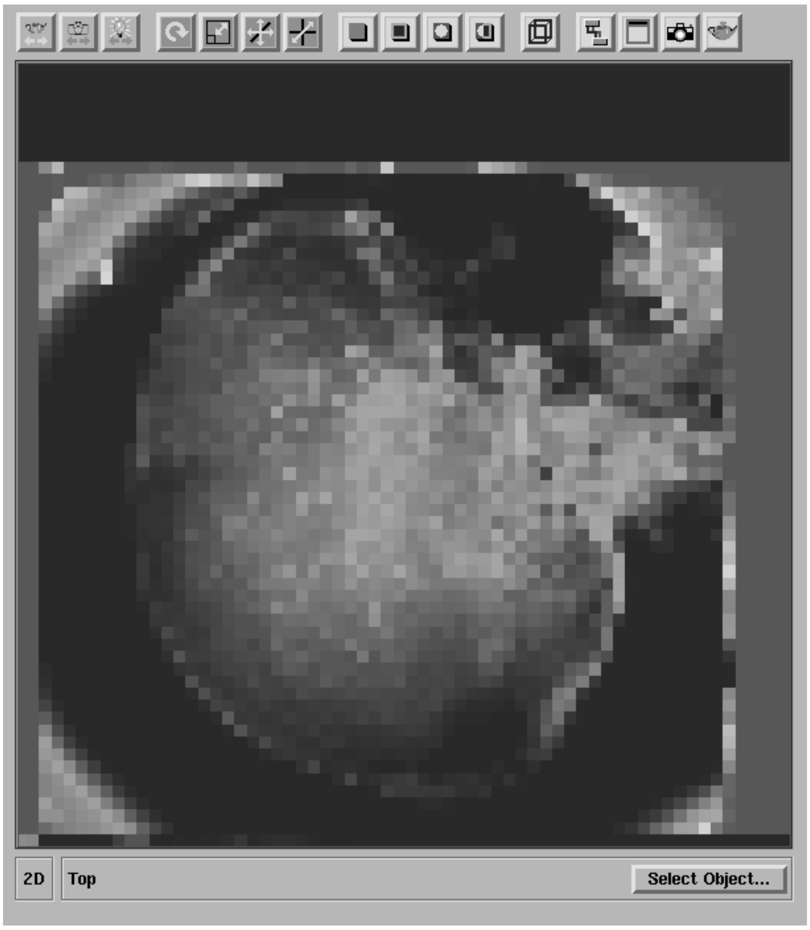
- Create a file called `mri.fld` to describe the data contained in `mri.dat`.
- Use the template field file given on page 26 and the information given above to complete the specification.

3.6.2 Displaying the MRI data



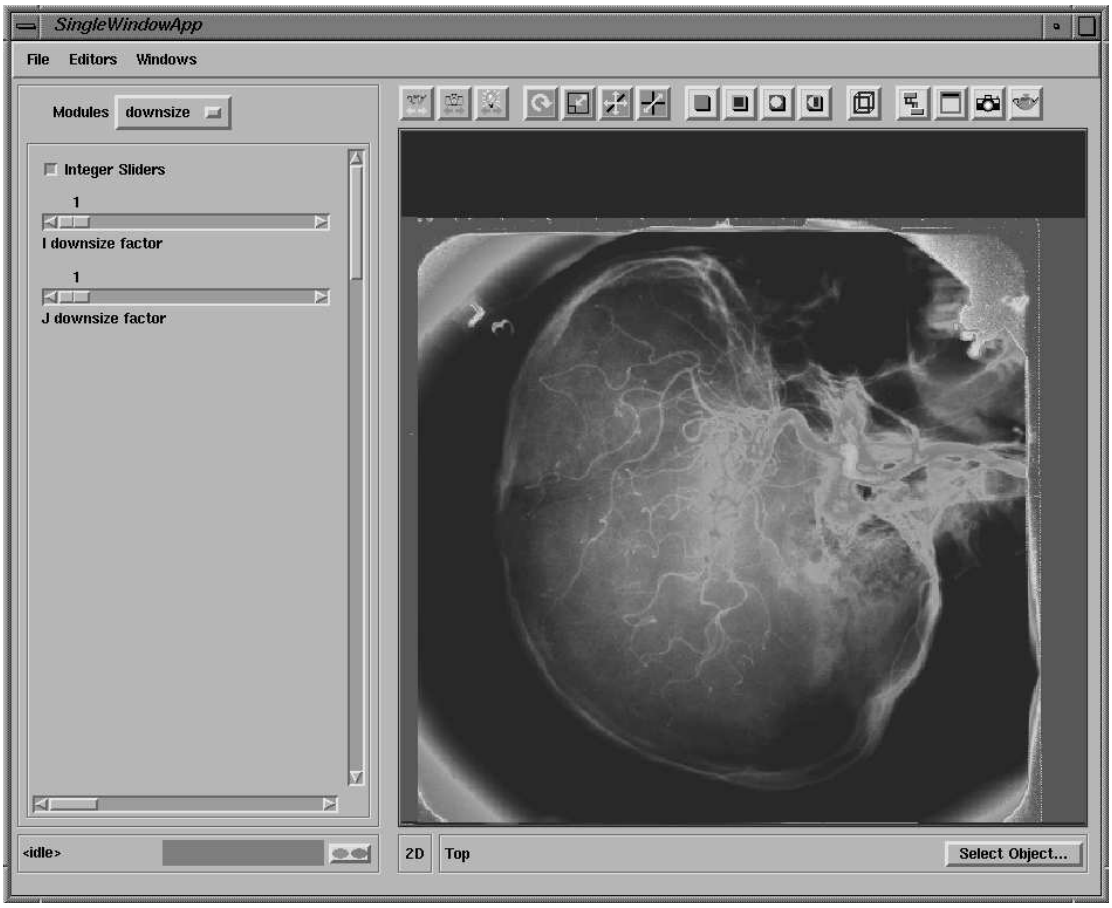
- Using the network editor, instance the modules in the list below into the workspace and construct the above network (be careful to connect the correct ports).
  - `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
  - `downsize` (Filters): resamples a field using 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.)

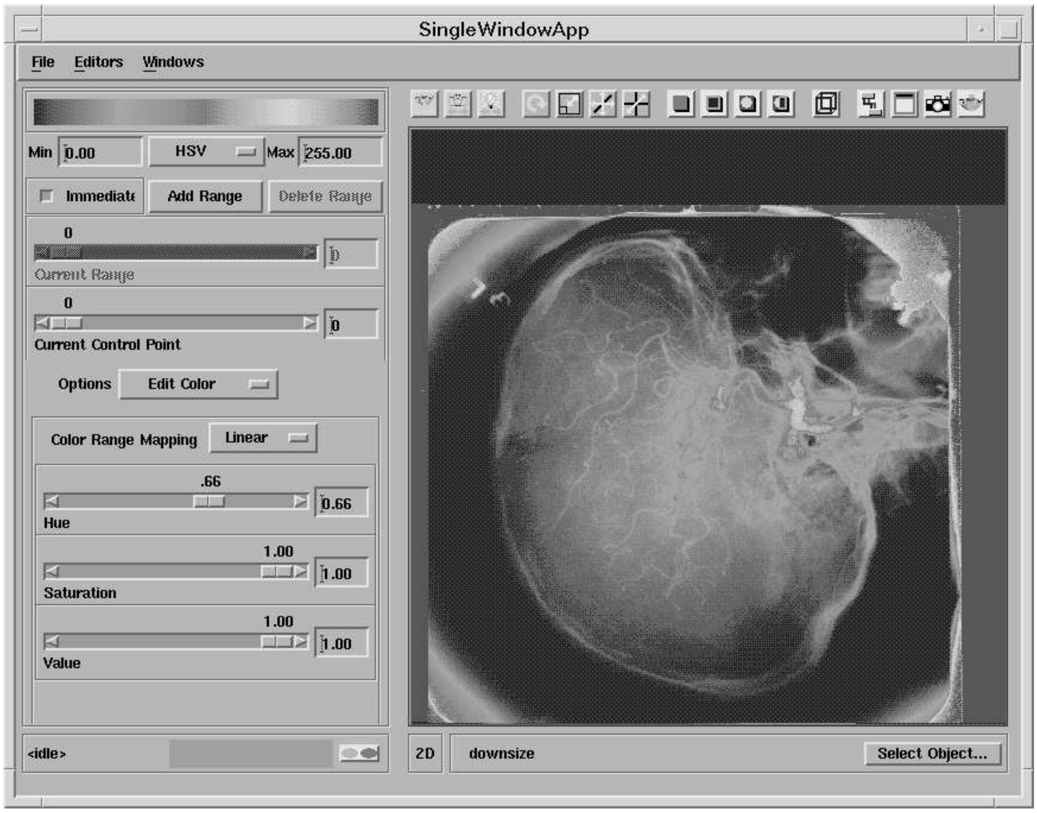
3.6.5 Changing the colour map (1)

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:

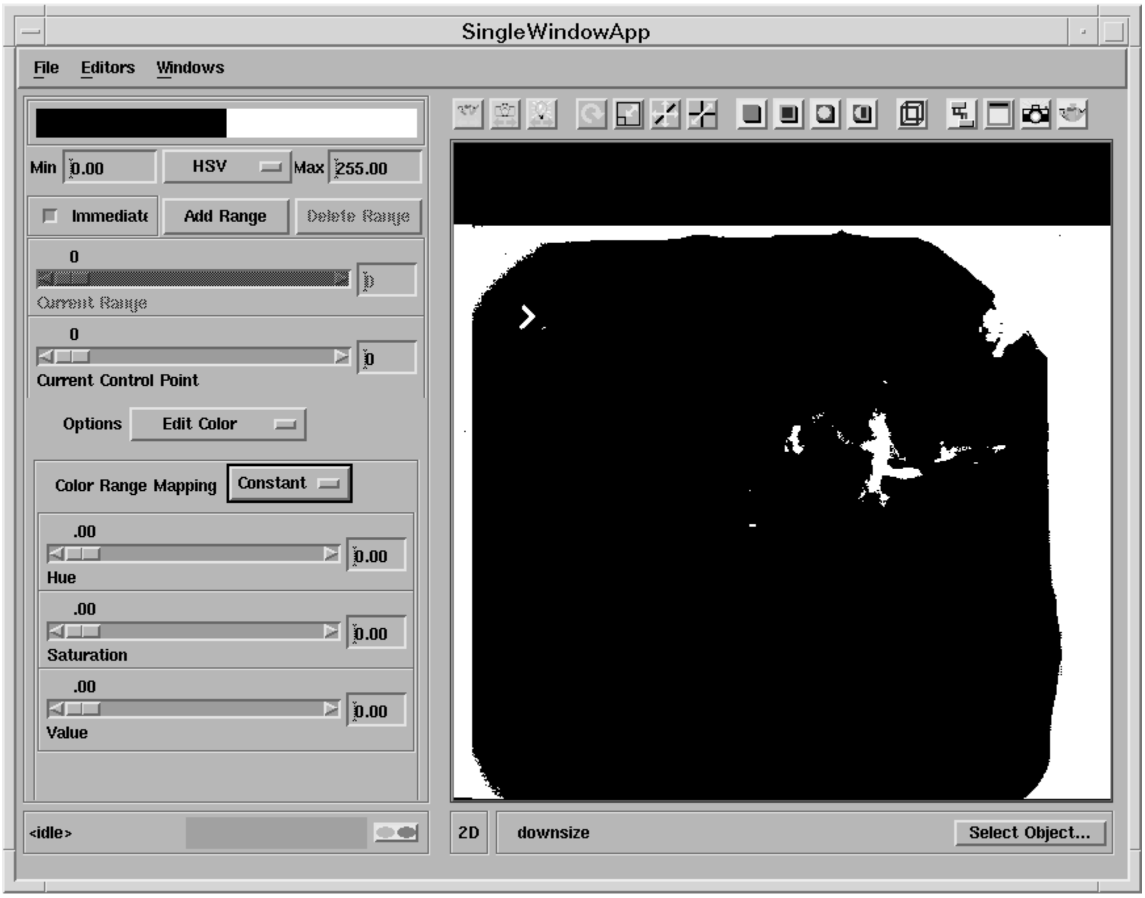
- There are two colour models which may be used
    - **HSV** (**H**ue **S**aturation **V**alue)
    - **RGB** (**R**ed **G**reen **B**lue)
- The **colour range maps** can either be **linear** (interpolated between max and min values of data) or **stepped** (constant colour for each step). The mapping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

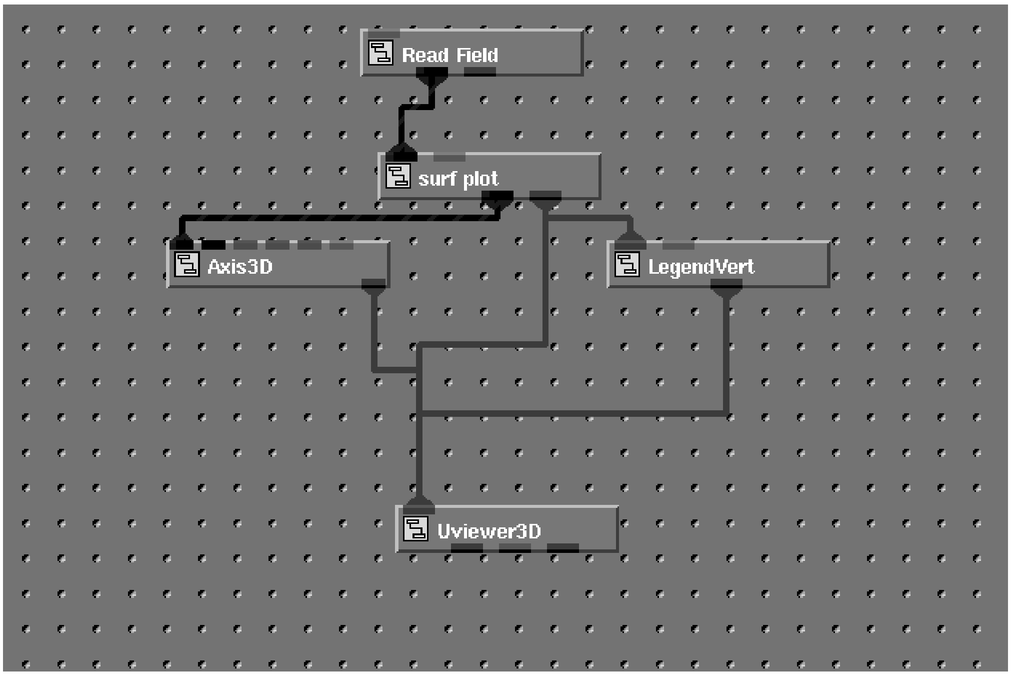
3.7.1 Importing the `temps.dat` data (1)

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.

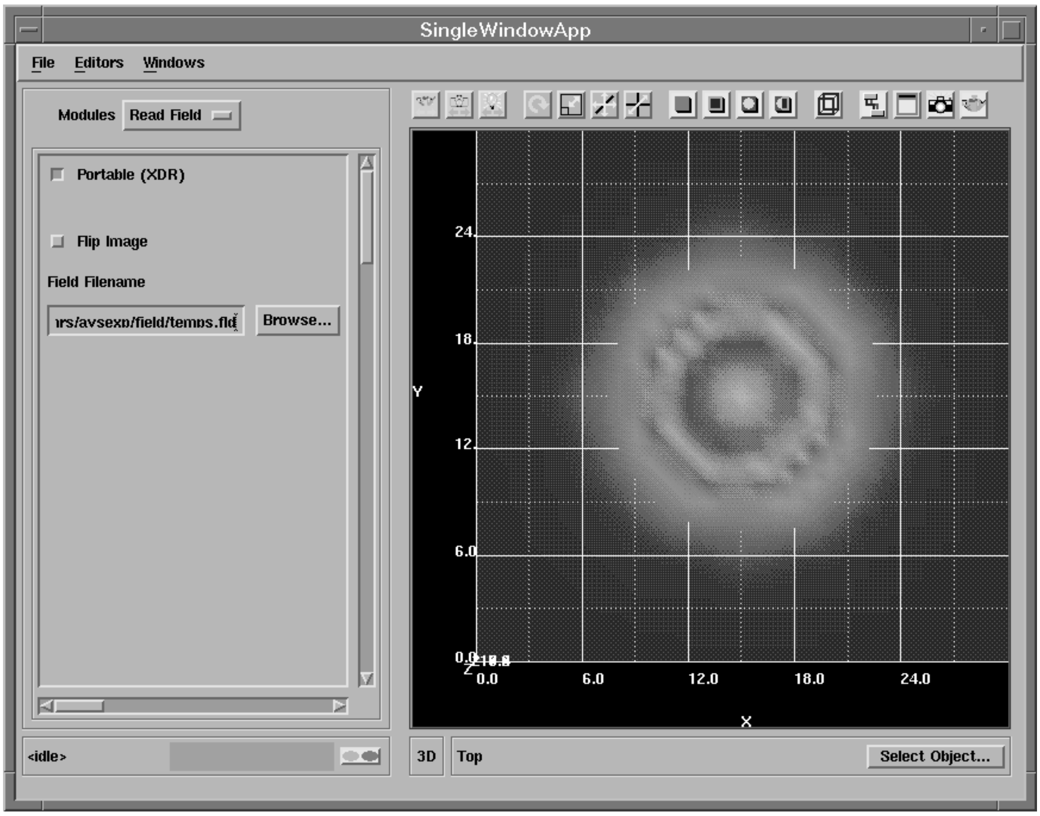
3.7.2 Displaying the temperature data with `surf_plot`



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

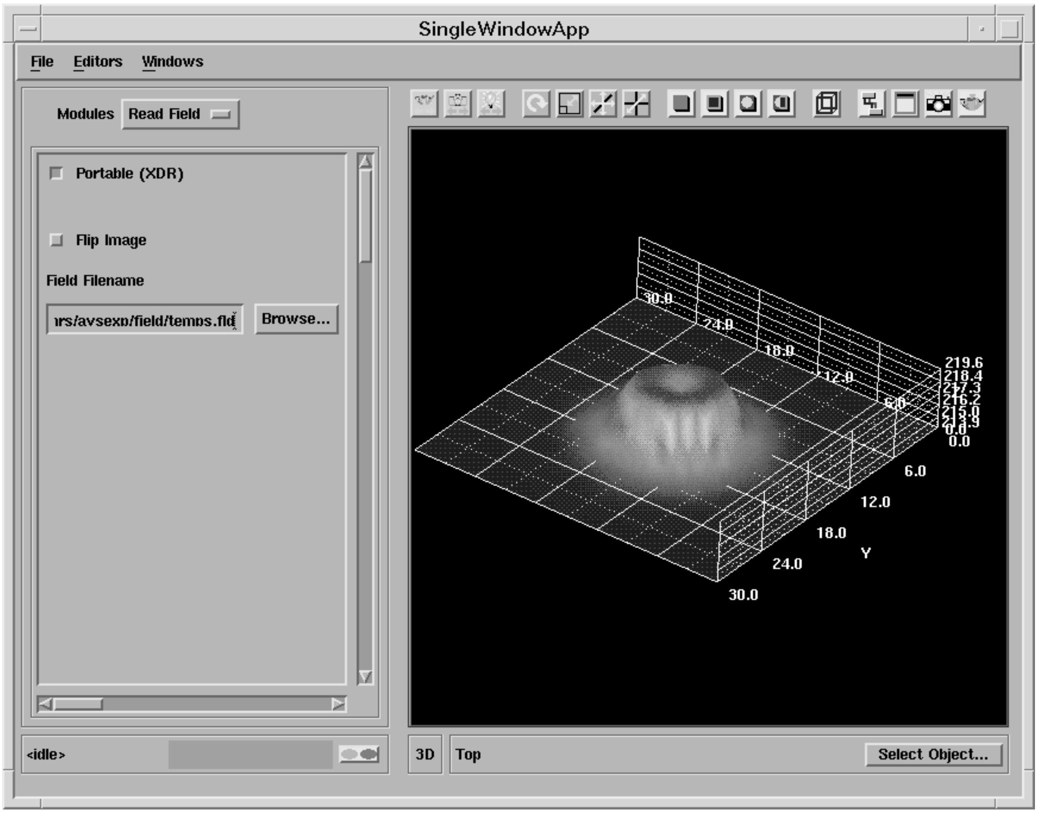
- `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
- `LegendVert` (Geometries): produces a color legend which displays the object's datamap.
- `surf_plot` (Mappers): creates a 2D or 3D mesh whose height is proportional to the scalar data value at each point.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



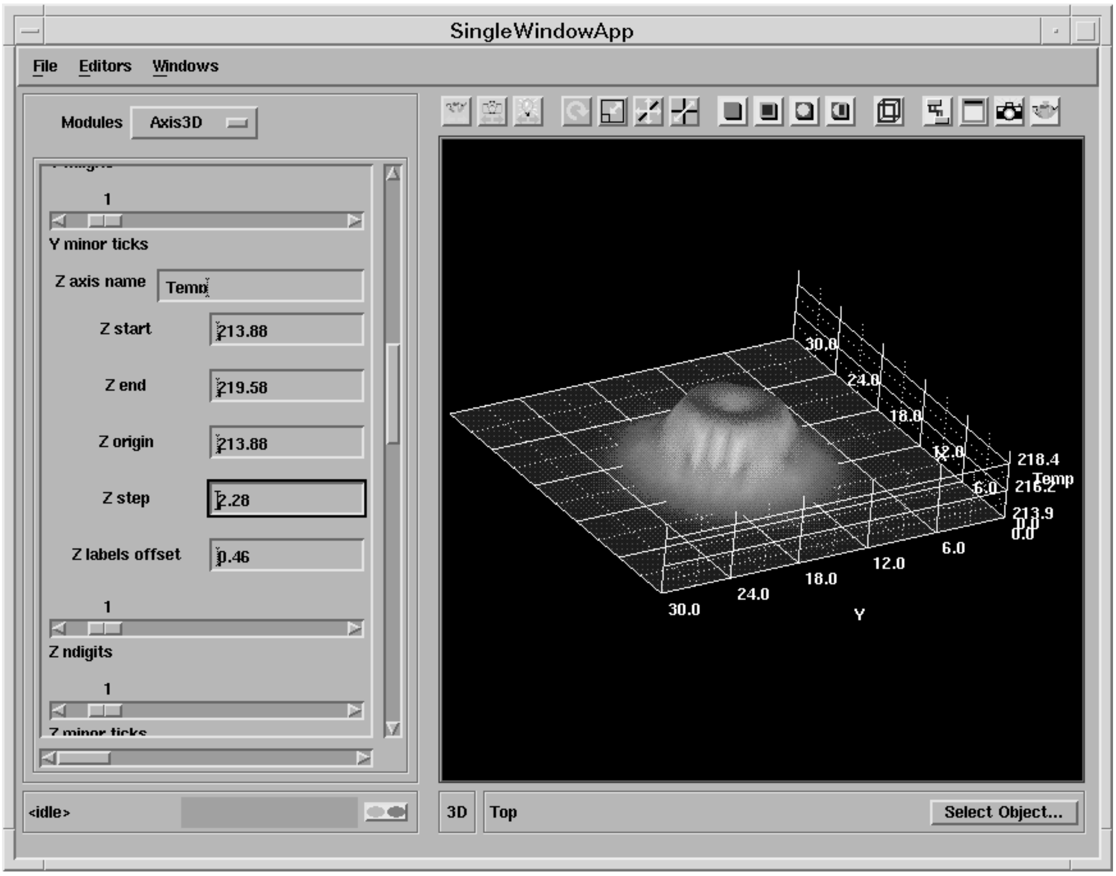
- Select the `Read_Field` module from the **Modules** editor panel and press the browse button. The file browser should now appear.
- To import the data into the network, select the file `temps.fld` you have just created using the file browser.
- If a picture similar to the one above appears then you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



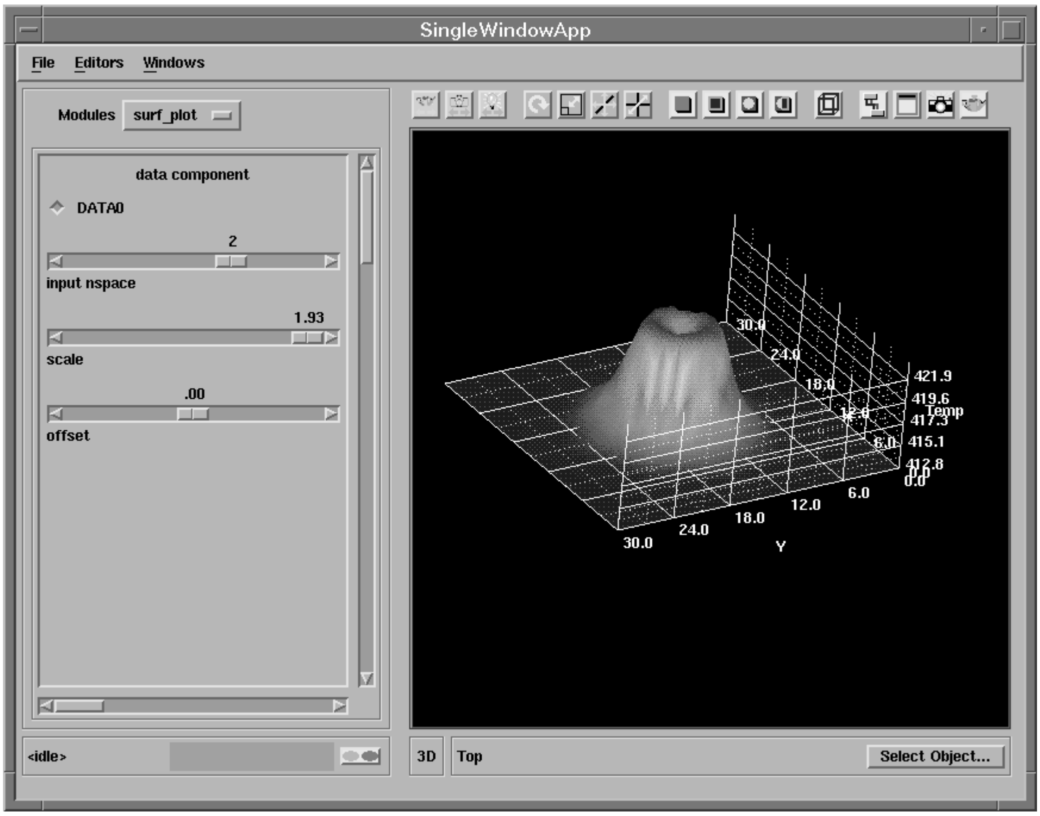
- Click on the **Reset/Normalize/Center** icon in the Toolbar so you can fit the whole object in the window.
- Now using the mouse, transform the object so it appears as above.

3.7.5 Altering `LegendVert` and `Axes3D`



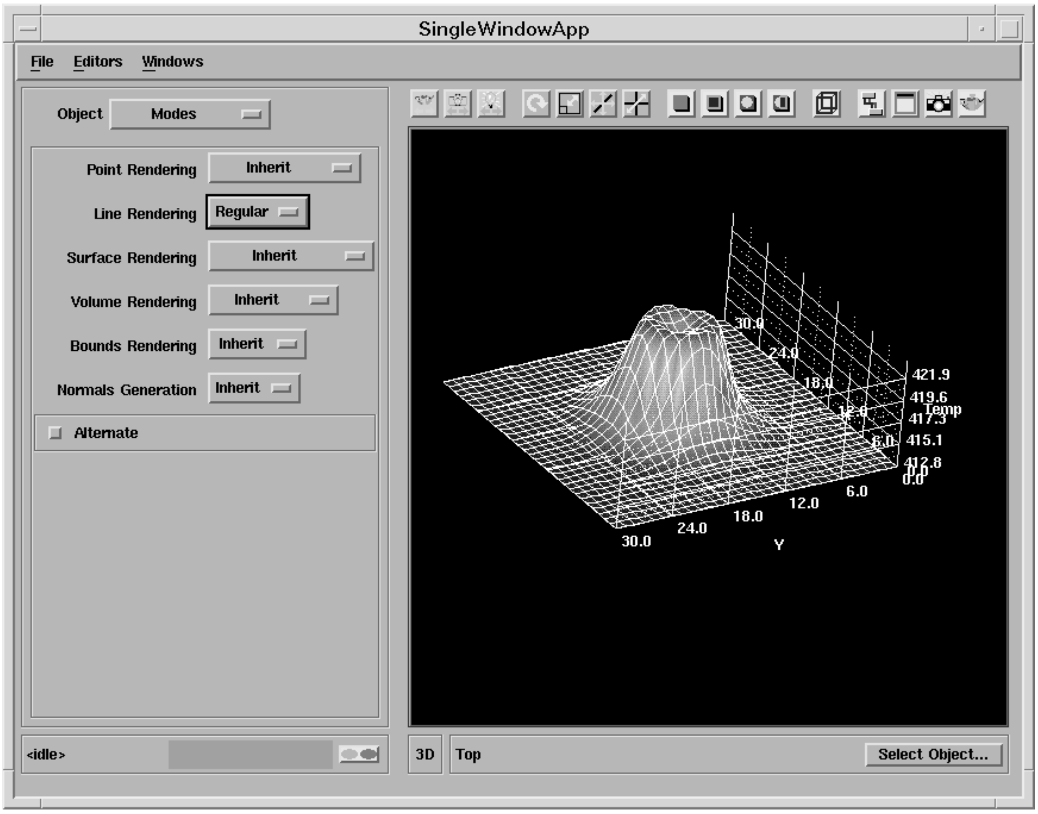
- Select the `LegendVert` module from the **Modules** editor panel
- Change the parameters of the colour bar or legend to make it look optimal in the picture.
- Similarly, select the `Axes3D` module from the **Modules** editor panel:
    - Reduce the number of steps on the z-axis.
    - Label the z-axis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

3.7.6 Using `surf_plot`



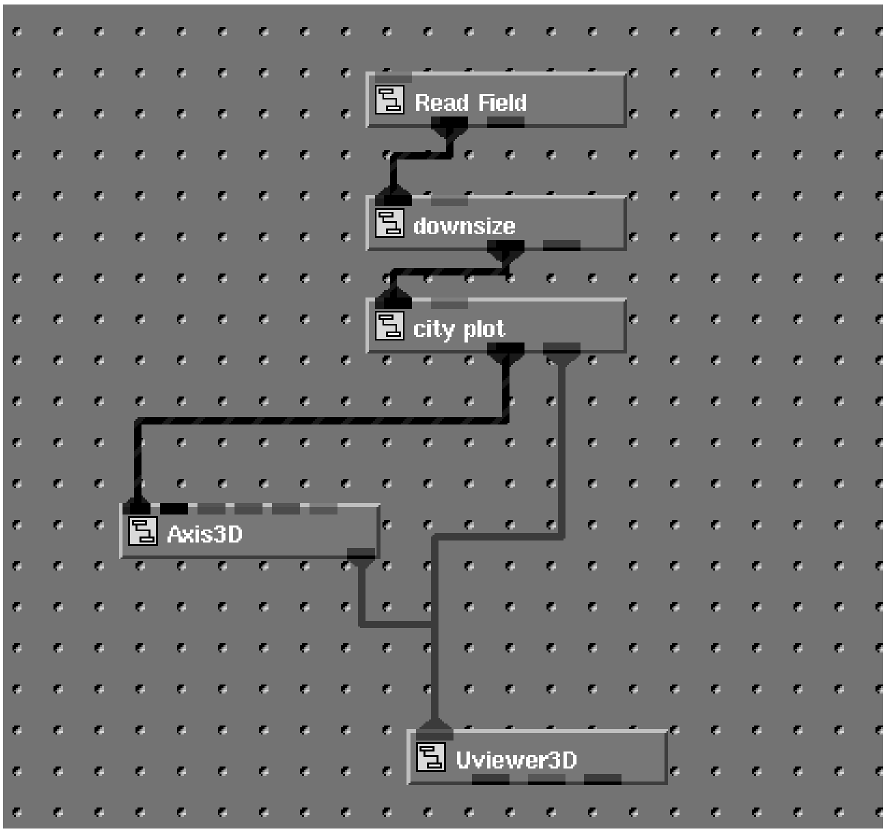
- Select the `surf_plot` module from the **Modules** editor panel
- Alter the slider to increase the scaling.
- The module should now produce a picture as shown above.

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.

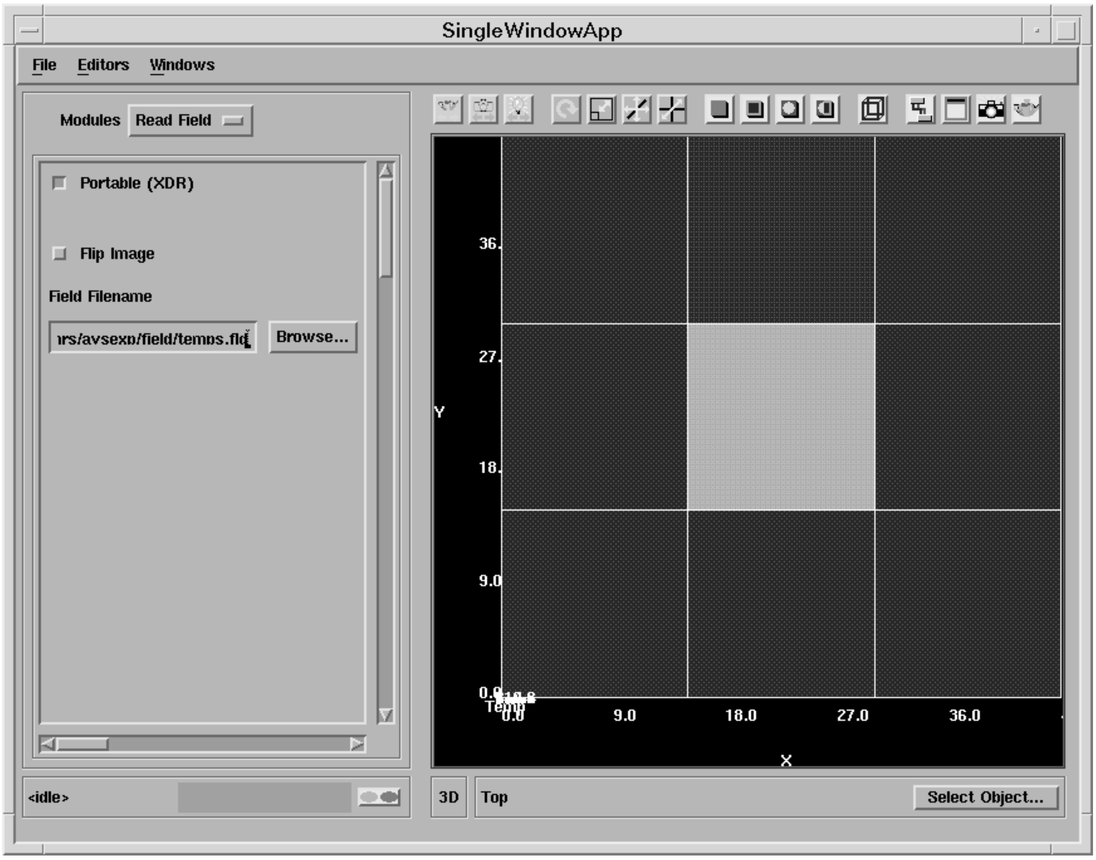
3.7.8 Displaying the temperature data with `city_plot`



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

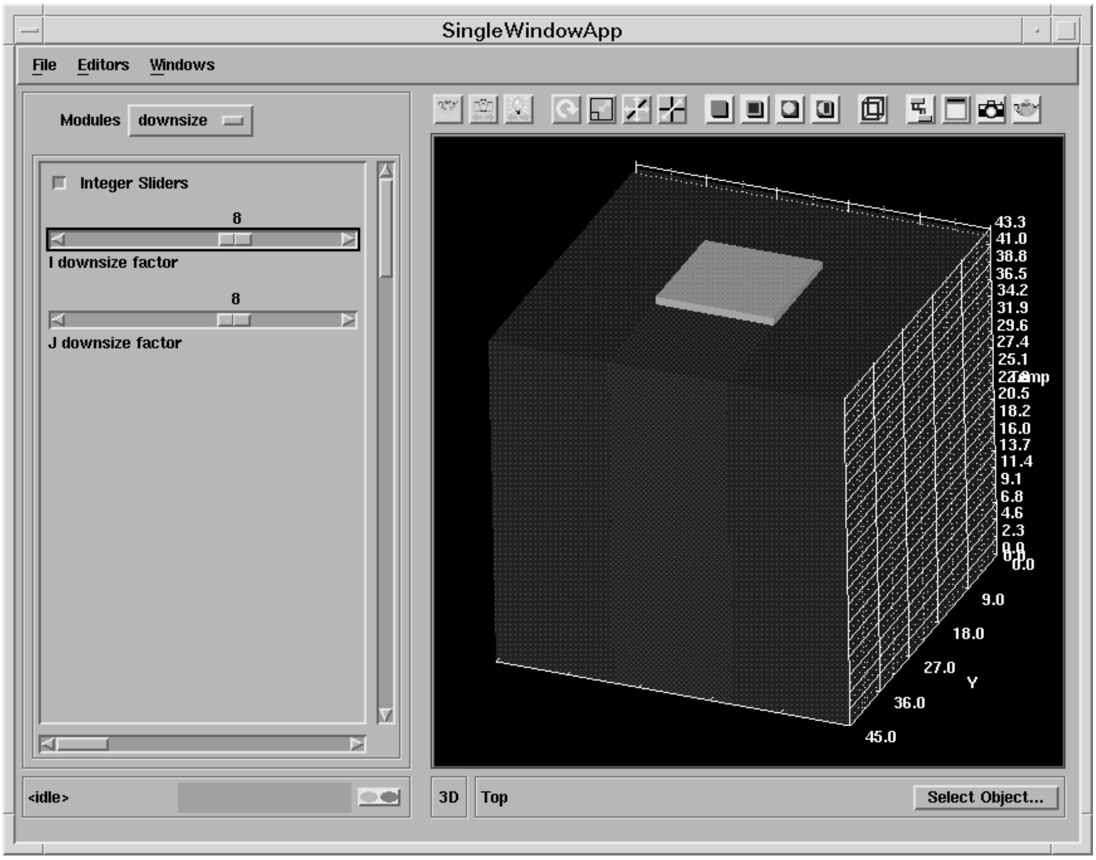
- `Read_Field` (Data IO): reads an AVS field file and converts it into an AVS/Express field.
- `downsize` (Filters): resamples a field using a scaling factor. Usually the factor is greater than 1 and the size of the field is reduced to save processing time and memory.
- `city_plot` (Mappers): creates 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



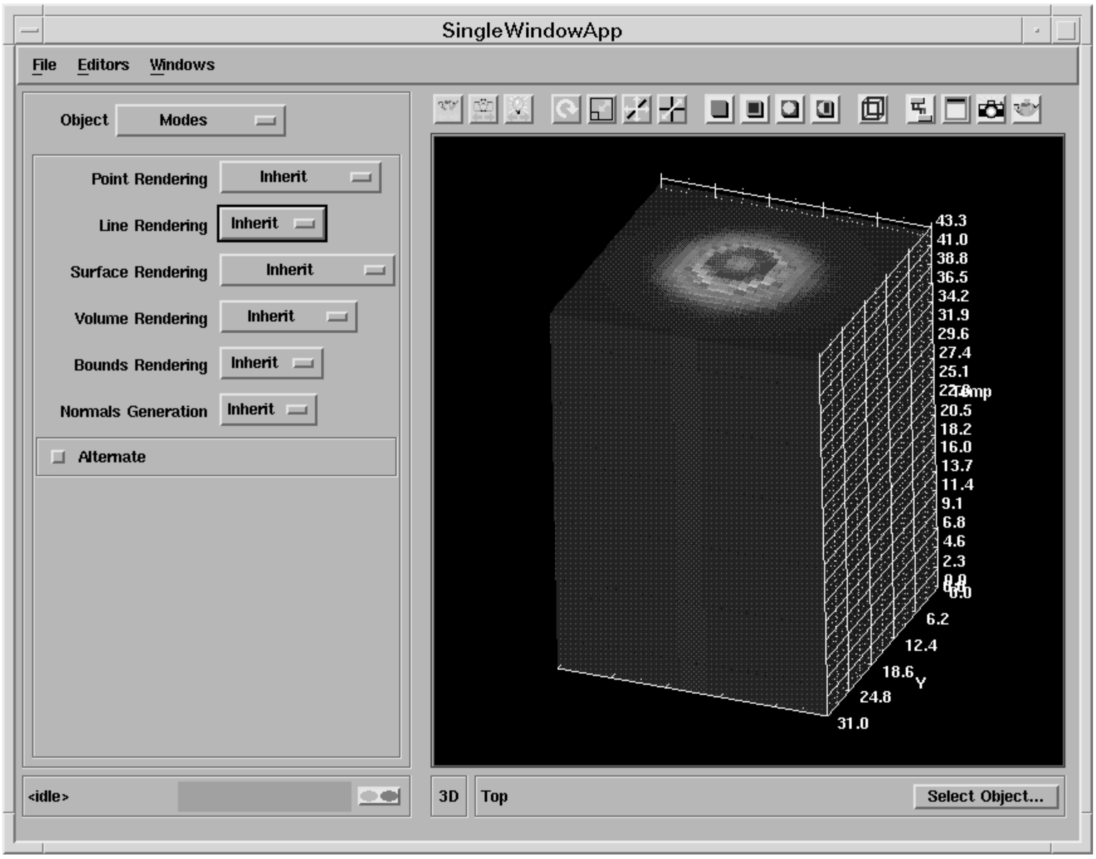
- Select the `Read_Field` module from the **Modules** editor panel and press the browse. The file browser should now appear.
- To import the data into the network, select the file `temps.fld` you have just created using the file browser.
- If a picture similar to the one above appears then you have been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