NOTICE

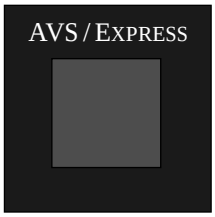
ACKNOWLEDGEMENTS

The use of portions, however modified, of the WWWLibrary code is in compliance with the MIT/W3C Copyright notice at *http://www.w3.org/pub/WWW/COPYRIGHT.html*. If you want to obtain a complete, unmodified version of WWWLibrary you can do so at *http://www.w3.org/pub/WWW/Library/* .

This software is shipped with software that is licensed by UNISYS. Use of this software for providing LZW capability fro any purposes is not authorized unless user first enters into a license agreement with Unisys under U.S. Patent No. 4,558,302 and foreign counterpartrs. For informatio concering licensing, please contact: Unisys Corporation. Welch Licensing Department - CISW19, Township Line & Union Meeting Road,s P.O. Box 500, Blue Bell, PA 19424

Advanced Visual Systems Inc.
300 Fifth Ave.
Waltham, MA 02154

Printed in U.S.A.

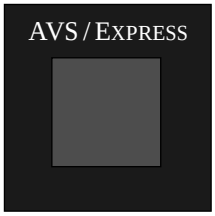


Contents

How To Use This Book

1 An Introduction to the AVS/Express Visualization Edition

2 Using Modules



How To Use This Book

This book contains tutorials and procedures that introduce some of the basic interaction techniques and some of the capabilities of the AVS / Express Release 3.1 Visualization Edition.

Audience

This book is intended for all AVS / Express Visualization Edition users. You should be familiar with the following concepts:

- You should know how to use a Motif- or Windows-style point-and-click interface.
- Use of the mouse on your system.
- Basic operating system commands and procedures.
- You should have a basic understanding of graphics concepts and data visualization techniques and principles.
- Since AVS / Express can be used as a development tool, it is helpful if you have an understanding of programming techniques; an understanding of object-oriented terminology is also useful.

Online help

AVS / Express has an integrated context-sensitive, hypertext help system that provides online access to the applicable documentation (depending on whether you have the Developer or Visualization Edition) as well as providing point-and-click access to information on any objects in the AVS / Express libraries.

Conventions

The following conventions are used in this guide.

Convention Example	Explanation
OK	**Plain Bold Text** This typeface indicates a button, a menu, or a menu option.
File → Save... **File -> Save...**	→ or **->** Indicates a menu selection. In these examples, you display the **File** menu and select the **Save...** option.
... edit the *.cshrc* file Refer to the *Getting Started* section.	*Italic* This typeface indicates a file name, a reference to another section or document, or it may be used for emphasis.
`Syntax Error: Line 17` `> cd ..`	`Computer Font` This typeface indicates a message or prompt, the contents of a file, or text that you must type.
`cd <directory>` `The file <filename> was successfully saved.`	<brackets> Brackets identify a place holder. In some cases, you supply the information, in other cases, the place holder identifies a value that is assigned by the application.
The line continuation symbol \ indicates…	\ The line continuation character indicates that although the command or text string appears on multiple lines, it *must* be entered on a single line.
Click	Click the left mouse button.

Convention Example	Explanation
Double-click	Rapidly click the indicated mouse button twice. (If a mouse button is not specified, assume the left mouse button).
Shift-click	Hold the **Shift** key down and click the indicated mouse button. (If a mouse button is not specified, assume the left mouse button).
Middle-click	Click the middle mouse button.
Right-click	Click the right mouse button.
Drag	Position the pointer over an item, press and hold the mouse button, move the mouse while the mouse button is still pressed, and then release the mouse button when the move is complete.

File Naming Conventions

File naming conventions differ between UNIX and Microsoft Windows (NT and 95) environments.

- UNIX file names are case-sensitive and pathnames use the / slash.
- Microsoft Windows file names are **not** case-sensitive and the pathnames use the \ slash.

This document uses UNIX conventions except where specifically referring to Windows system syntax.

Customer Support

Do not hesitate to contact Advanced Visual Systems if you need any assistance while using AVS/Express.

Depending on the nature of your problem, you can:

- Send e-mail to the support group at Advanced Visual Systems
  - support@avs.com
- Refer to the Advanced Visual Systems web page for support information (http://www.avs.com).
- Call a customer support representative on:
  - 800-428-7001 (within the continental US) or
  - +1-617-890-8192 x2400 (from elsewhere)
- Contact your local sales representative
  - Refer to the information on the back cover of this manual.

An Introduction to the AVS/Express Visualization Edition

This chapter contains a step-by-step procedure that introduces you to the AVS / Express interface, some of the objects, and some of the techniques associated with developing an “application” for visualizing your data.

Some of the topics that are covered in this chapter include:

- Starting the AVS / Express Visualization Edition
- Working with the Network Editor
- Instancing an object
- Reading data
- Viewing data
- Experimenting with objects
- Connecting objects
- Saving your work

1.1 Starting the AVS/Express Visualization Edition

The following steps outline how to start AVS/Express. These steps assume that:

- AVS/Express is installed
- AVS/Express is licensed
- All platform-appropriate environment variables are set (PATH, MACHINE, DISPLAY, LD_LIBRARY_PATH, SHLIB_PATH, and XP_PATH).

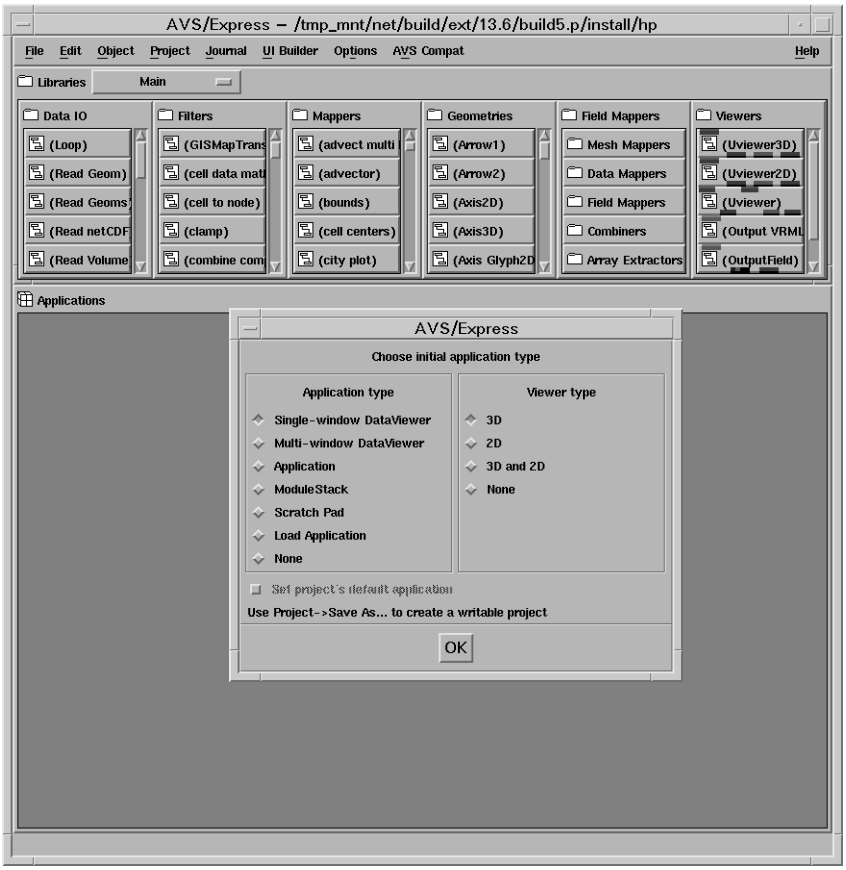
If these assumptions are not met, refer to the following AVS/Express books for instructions before continuing: *System Prerequisites*, *Installation and Licensing*, *User's Guide*, and *Getting Started*.

1. Open a command shell or an xterm window.
2. Navigate to the *express* directory and then start AVS/Express. If you are running on a Windows platform, you can click on the application icon. If you prefer to use commans, you can start AVS/Express using the appropriate command for your platform:

UNIX	**Windows**
vxp	start bin\pc\vxp

Note: *Do not start AVS/Express in the background. You start AVS/Express in the foreground because AVS/Express' V Command Processor (VCP) uses the shell window.*

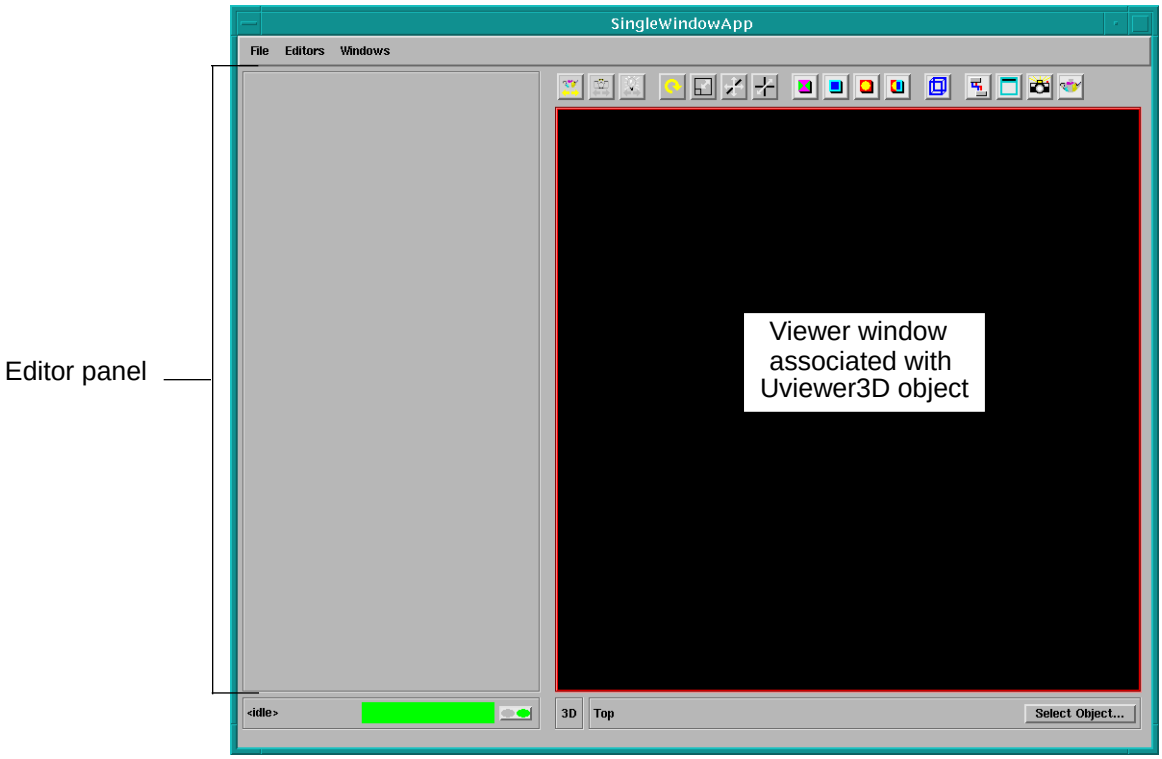
When you enter either command, the AVS/Express interface and a dialog box are displayed.

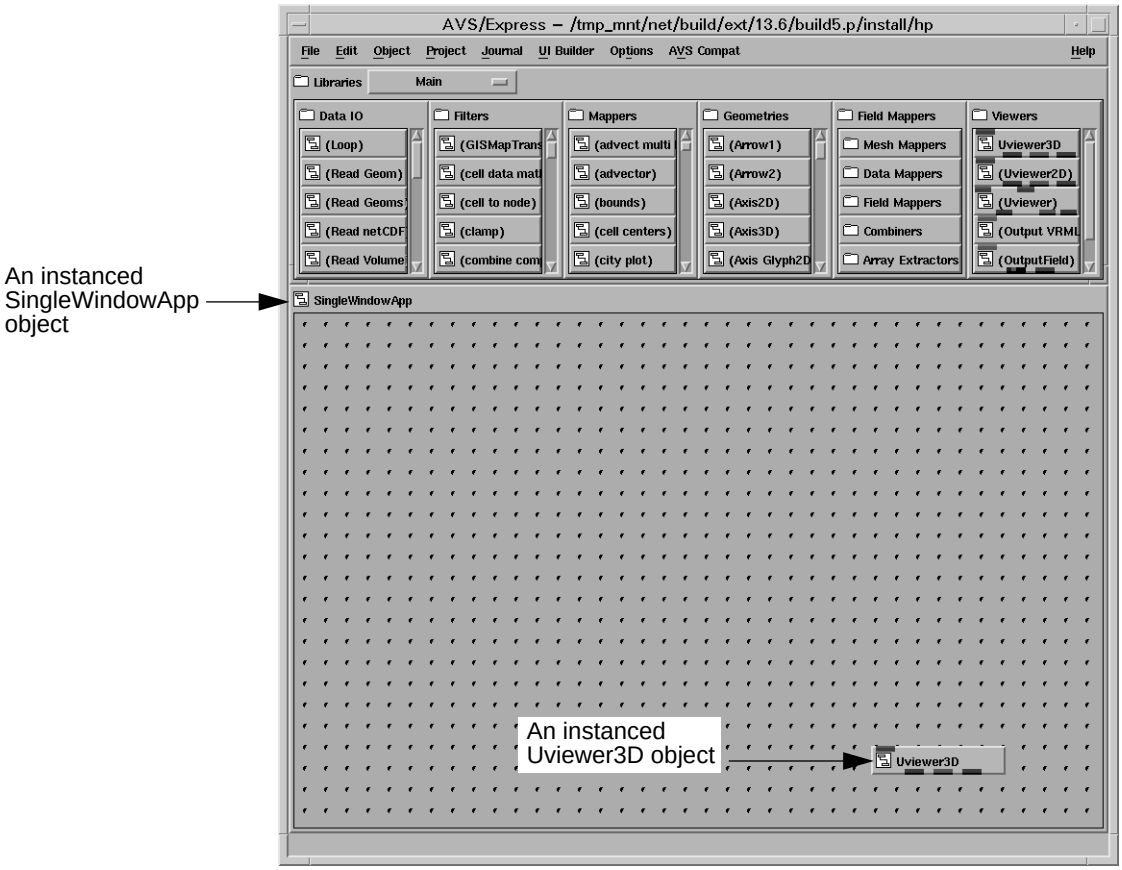


The AVS/Express main interface is called the Network Editor. The dialog box that is displayed provides options that allow you to specify your application environment and the type of viewer you will use in that application.

3. For the purposes of this tutorial, press **OK** to accept all of the default settings from the dialog box.

The SingleWindowApp window is displayed (see page 1-4), the *SingleWindowApp* object and the *Uviewer3D* object are instanced in the Network Editor (see page 1-5), and the V Command processor is started in the window where you started AVS/Express.

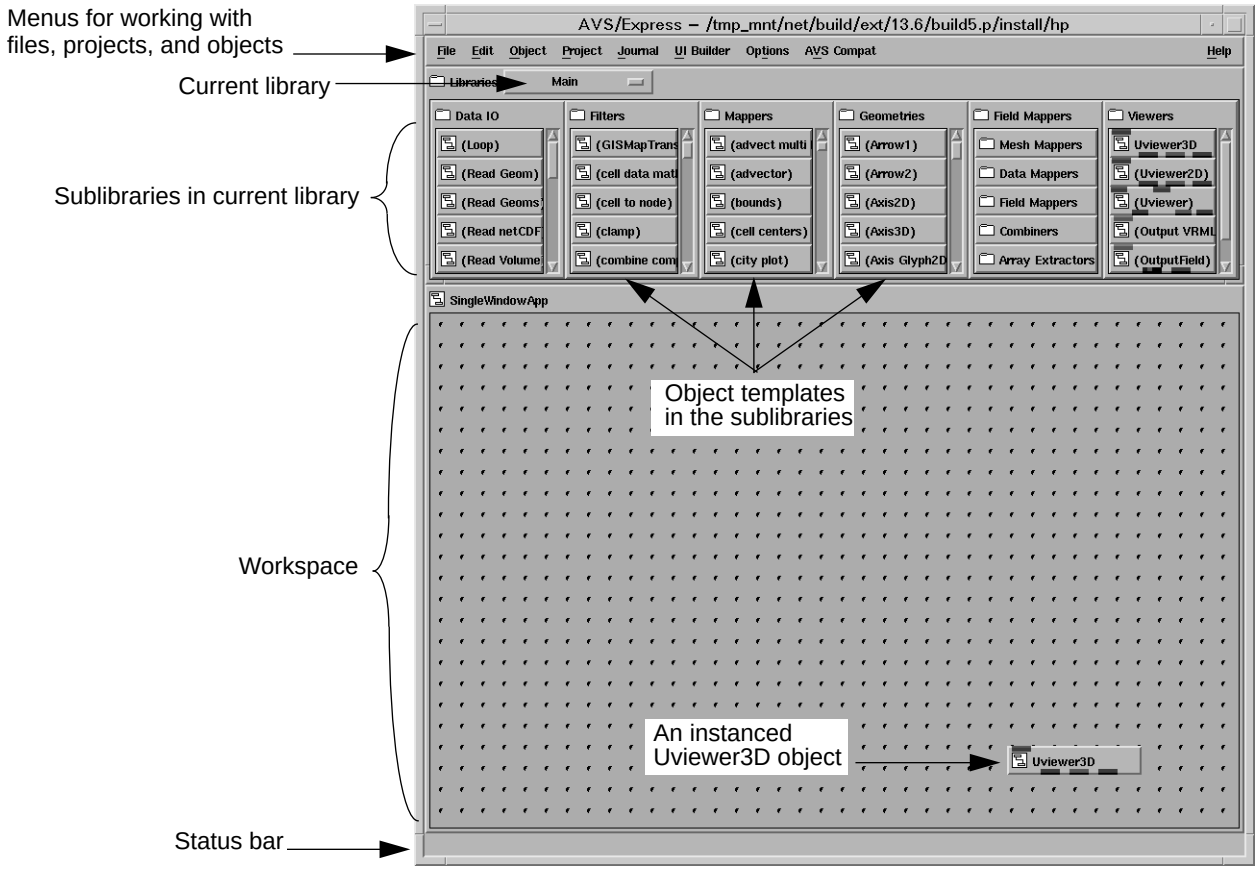




Note: *The V Command processor is a command-line interface to AVS/Express. You will be introduced to some of the capabilities of this command line interface in other books.*

1.2 Working with the Network Editor

The Network Editor is your primary means of creating your application. It provides access to all of the AVS/Express object **templates**, and it contains the workspace where you organize and connect **instanced objects** to form the application's **network**.



Before you go on, notice that all of the object templates are named, but the names are enclosed in parentheses. The parentheses indicate that the object's information is not yet loaded into memory. This is important to note, because the first time

you click on a template or attempt to create an instance of a template there is a slight delay. The delay is the actual “loading into memory” time. Once the template information is loaded into memory, the parentheses are no longer displayed, and the next time you use that template icon, all interactions are almost immediate.

1.2.1 Instancing an object

You create an instance of an object by dragging the object’s template icon to the workspace. Follow the procedure below create an instance of the *Read Field* object.

1. Scroll through the Data I/O sublibrary until you see the Read Field icon. It willinitially have parentheses around its name.

2. Click on the *Read Field* icon.

   Notice the delay while the object’s information is loaded into memory. Also notice that the parentheses are removed once the object’s definition is loaded into memory.

3. Point at the *Read Field* icon, press and hold the left mouse button, and then drag the object’s icon to the workspace.

   As you drag the *Read Field* object, the border around the Data I/O sublibrary is highlighted, and then when *Read Field* object crosses over to the workspace area, the *SingleWindowApp* object is highlighted.

4. When the *SingleWindowApp* object is highlighted and the outline of the *Read Field* object is displayed in the workspace area, release the mouse button.

What do the different highlighting colors mean?

There are several different types of highlighting in AVS/Express:

- As you move a template object from one area to another, the highlighting indicates the parent object.

  For example, while you were moving the *Read Field* object, the Data I/O library was highlighted (indicating that it was the parent object) up until the

point when the *Read Field* object was positioned in the *SingleWindowApp* object area, at which point, the *SingleWindowApp* object was highlighted to indicate that it would be the parent of that instance of the *Read Field* object.

- If you click on an object icon, it indicates that it is selected.

1.3 Creating an Application: A Short Tutorial

This tutorial will teach you more about how to use the AVS/Express Visualization Edition, and it will demonstrate the importance of being familiar with your data and with data visualization techniques when creating applications.

Before you continue, recall that AVS/Express automatically instanced the *Uviewer3D* object and the *SingleWindowApp* object. Also recall that you instanced a *Read Field* object. The *Read Field* will be used to read data into the AVS/Express **field** data structure.

You will also create an instance of the *downsize*, *crop*, and *isosurface* objects.

- The *downsize* object is a data preprocessor; it resizes a field to either reduce or increase the amount of information in the original field.
- The *crop* object is also a data preprocessor; it extracts a subset of the data in the field within a specified range so only the data of interest is displayed.
- The *isosurface* object performs the actual visualization of the data.

As you will see, it is important that you know about your data and that you know if you need to perform any preprocessing on that data. It is also important that you know how you want to visualize the data, because there are many visualization techniques, and knowing both what is available and what is appropriate will dictate the effectiveness of your application.

Let's begin.

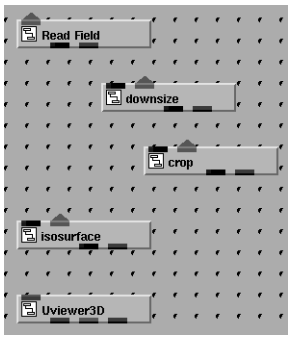
1. Look in the Filters sublibrary, and locate the *downsize* object.
2. Create an instance of the *downsize* object by dragging its icon to the SingleWindowApp workspace area.

   The *downsize* object is a child of the *SingleWindowApp* object.
3. Look again in the Filters sublibrary, locate the *crop* object, and create an instance of the *crop* object by dragging its icon to the SingleWindowApp workspace area.

The *crop* object is a child of the *SingleWindowApp* object, and is a sibling of *downsize*.

4. Now look in the Mappers sublibrary, locate the *isosurface* object, and create an instance of *isosurface* by dragging its icon to the SingleWindowApp workspace area.

Your workspace should look similar to what is shown below.



1.3.1 Reading Data

This tutorial will read a sample data file into AVS/Express's field data structure. Follow the procedure below to read the *hydrogen.fld* file.

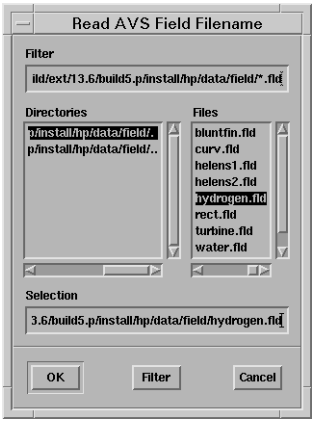
1. Bring the SingleWindowApp window to the front; in other words, make it the active window.

2. Look at the editor side of this window (the left side), and if the Modules option is not indicating Read Field, click on the Modules drop down menu and select Read Field.

3. Click **Browse**.

The Read AVS Field Filename dialog is displayed.

4. Select *hydrogen.fld* from the Files list, and then press **OK** to load the data into the field data structure.



The data associated with the *hydrogen.fld* file is loaded into the field data structure.

1.3.2 Viewing the Data

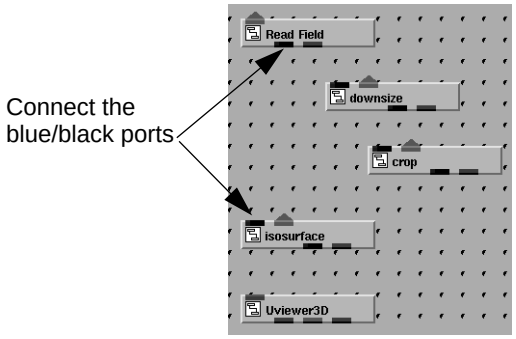
Now that you have data, you can experiment with the objects to learn more about AVS/Express and data visualization.

First, let's send the data to the visualization object (*isosurface*) and then send those results to the *Uviewer3D* object so they can be displayed in the viewer window.

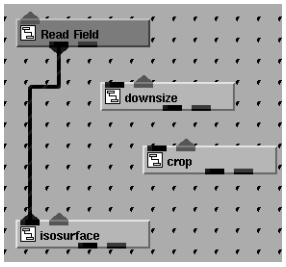
1. First, adjust the windows on your screen so you can see both the Network Editor and the viewer side (the right side) of the SingleWindowApp window.

2. In the Network Editor, point to the blue/black output port on the *Read Field* object, press and hold the left mouse button, and then drag the mouse toward the blue/black input port on the *isosurface* object.

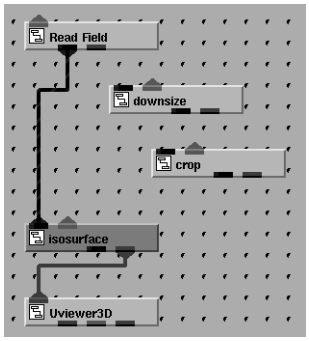
   Note: *Output ports are on the bottom of the object icon, input ports are on the top of the object icon.*



A **connection line** is drawn between the two objects, but notice that there is no change to the viewer area in the SingleWindowApp window. (A connection line indicates a reference between an object's parameters. If the parameters change, the objects that are connected downstream are automatically updated with the change.)



3. Now connect the red output port on *isosurface* to the red input port on *Uviewer3D*.



Once the connection to the *Uviewer3D* object is made, look at the SingleWindowApp window and notice that the viewer area is updated with a visual representation of the data.

1.3.3 Experimenting with the network

Look closely at the image; you'll be experimenting with the *downsize* and the *crop* objects and you'll want to notice how including those objects affect the image.

1. In the Network Editor, disconnect the blue / black output port on *Read Field* from the blue / black input port on *isosurface.* Use either of the methods below to remove the connection line.

Either

- Point at one of the connected ports, press and hold the left mouse button, and move the mouse slightly until all possible connections are displayed. Move the mouse so that the connection you want to break is highlighted, and then release the mouse button.

Or

- Point at the connection line, press and hold the right mouse button, and when the pop-up menu is displayed, select **Delete Connection**.

2. Now connect the blue / black output port on *Read Field* to the blue / black input port on *downsize*.

3. Connect the blue / black output port on *downsize* to the blue / black input port on *isosurface*.

   Again, look at the SingleWindowApp window and notice that the viewer area is updated with a visual representation of the data. Look closely at the image and notice that "less" data is being visualized.

4. Look at the editor side of the SingleWindowApp window (the left side), click on the Modules drop down menu and select downsize.

1.3.3.1 Experiment with *downsize*

1. Experiment with the sliders to see how the downsize factors affect the image.

   a. Set all downsize factors to 1. Notice that the image appears as it did before you connected the *downsize* object.

   b. In the viewer area, rotate the image by pressing the middle mouse button (**MB2**) and dragging the mouse slightly to the left.

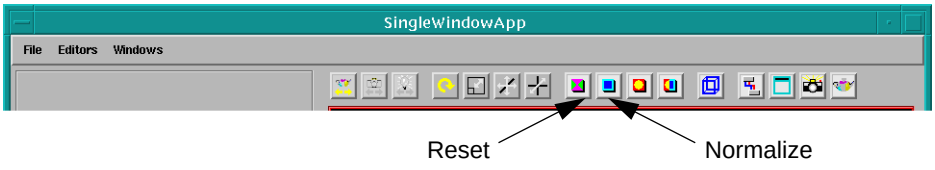
   c. Adjust the *I downsize factor* one increment at a time and notice the results.

   d. Set *I downsize factor* back to 1 and then adjust the *J downsize factor* one increment at a time and notice the results.

   e. Set *J downsize factor* back to 1 and then adjust the *K downsize factor* one increment at a time and notice the results.

2. Return all downsize factors to 8.

3. Look at the buttons that run above the viewer area of the SingleWindowApp window and click **Reset** and then click **Normalize**.



4. Make the Network Editor the active window.

5. In the Network Editor, disconnect the blue / black output port on *Read Field* from the blue / black input port on *downsize*.

6. Disconnect the blue / black output port on *downsize* from the blue / black input port on *isosurface*.

1.3.3.2 Experiment with *crop*

1. Connect the blue / black output port on *Read Field* to the blue / black input port on *crop*.

2. Connect the blue / black output port on *crop* to the blue / black input port on *isosurface*.

3. Bring the SingleWindowApp window to the front, click on the Modules drop down menu and select crop.

4. Notice the six min and max sliders, and experiment with each one individually.

   a. Set *I min* to 30, look for the effect of this change and then set *I min* back to 0.

   b. Set *I max* to 36, look for the effect of this change and then set *I min* back to 63.

   c. Set *I min* to 30, set *I max* to 36, and look for the effect of this change.

   d. Set I min back to 0 and I max to 63.

   e. Repeat the Steps A through D for *J min* and *J max*.

   f. Rotate the image slightly, and then repeat Steps A through D for *K min* and *K max*.

5. Look at the buttons that run above the viewer area of the SingleWindowApp window and click **Reset** and then click **Normalize**.

1.3.4 Connect all objects

1. In the Network Editor, disconnect the blue / black output port on *Read Field* from the blue / black input port on *crop*.
2. Connect the blue / black output port on *Read Field* to the blue / black input port on *downsize*.
3. Connect the blue / black output port on *downsize* to the blue / black input port on *crop*.
4. Bring the SingleWindowApp window to the front, and look at the image in the viewer area.
5. Look at the editor side of the SingleWindowApp window and notice how the slider values for *crop* have been adjusted automatically to account for preprocessing that is being performed by *downsize*.

1.3.5 Add *bounds* to the network

1. In the Network Editor, look in the Mappers sublibrary, locate the *bounds* object, and create an instance of *bounds* by dragging its icon to the SingleWindowApp workspace area.
2. Connect the blue / black output port on *crop* to the blue / black input port on *bounds*.
3. Connect the red output port on *bounds* to the red input port on *Uviewer3D*.
4. Bring the SingleWindowApp window to the front, and look at the image in the viewer area.

   The *bounds* object shows the bounds or limits of the data being visualized. Now that you can see where the data “ends”, you might find it useful to repeat the steps in *Experiment with downsize* on page 1-14 and *Experiment with crop* on page 1-15.

1.3.6 Save your network

1. Select **File -> Save Application**.

   The Save Application dialog box is displayed.

2. Navigate to the desired directory.

3. Enter a file name, for example, *tutorial1.v*, in the Selection field and then press **OK**.

4. Now that your application is saved, select *downsize*, *crop*, and *isosurface*.

5. Select **Edit -> Delete**.

AVS / EXPRESS

2

Using Modules

This chapter consists of several examples that introduce you to using the Network Editor to create a network of objects for the purpose of visualizing data.

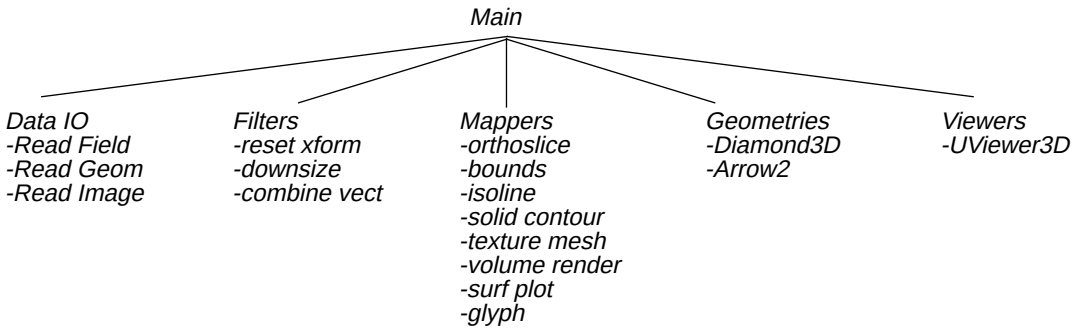
Some of the topics that are covered in this chapter include:

- Using modules
- Slicing a volume
- Creating isolines
- Creating solid contours
- Performing volume rendering
- Using the texture mesh module
- Specifying node locations with glyphs
- Combining components

2.1 Modules

This tutorial requires you to use modules from the *Main* library. The *Main* library consists of several sublibraries. The modules you *instance* in this chapter are found in the *Data IO*, *Filters*, *Mappers*, *Geometries*, and *Viewers* sublibraries of *Main*.

The following figure shows the modules you will be using in this chapter:

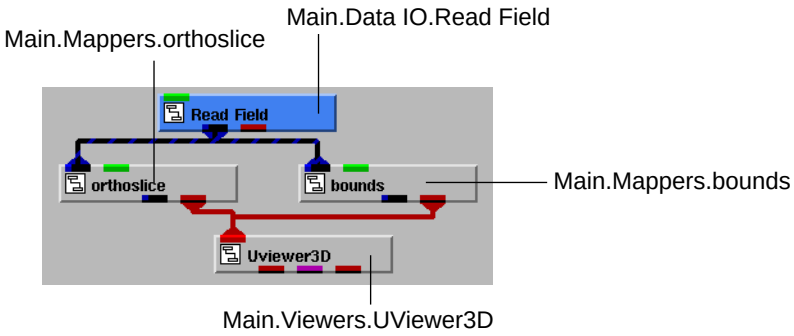


- *Data IO* modules read the data you want to visualize. Examples of how the data can be stored include field objects (*.fld* files), geometric objects (*.geo* files), and image objects (*.x* files).
- *Filters* modules perform data processing. You typically use these modules to modify the data so that it is easier to view it in the data viewer.
- *Mappers* modules perform various functions to extract various aspects or dimensions of the data for viewing.
- *Geometries* modules contain the mesh of various geometric objects.
- *Viewers* modules contain all the different types of viewers available with AVS/Express.

2.2 Slicing a Volume

The *orthoslice* module creates a two-dimensional slice from a 3D structured field. This infinitely-thin slice is perpendicular to the coordinate axis you specify. Use the Network Editor to set up the network as shown:

1. Instance and connect the *Read Field*, *orthoslice*, *bounds*, and *Uviewer3D* modules, as shown in the following figure.



2. Using *Read Field*, read the structured field data stored in *hydrogen.fld*.

   a. In the *SingleWindowApp* window, select **Read Field** from the **Modules** pull-down list.

   b. Click the **Browse...** button and select *hydrogen.fld*.

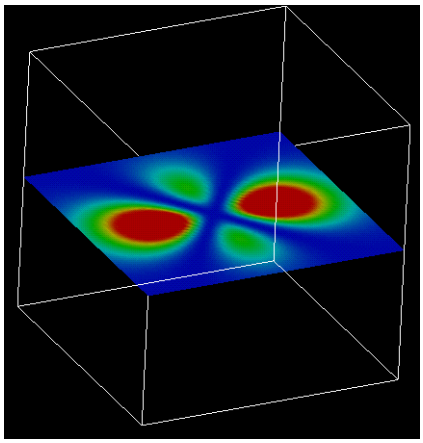
3. Set the slice so that it is perpendicular to the y-axis. Note that the default is 0, where: 0 = x-axis; 1 = y-axis; 2 = z-axis.

   a. In the *SingleWindowApp* window, select **orthoslice** from the **Modules** pull-down list.

   b. Using the slider, set the *axis* parameter to 1.

4. Rotate and scale the image in the viewer for better visibility.

The following figure shows the resulting image in the viewer.

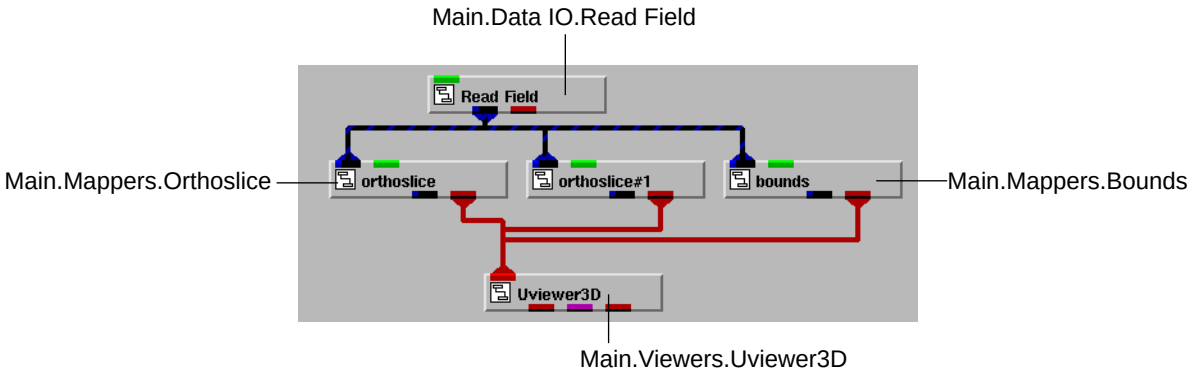


2.2.1 Using Two *orthoslice* Modules

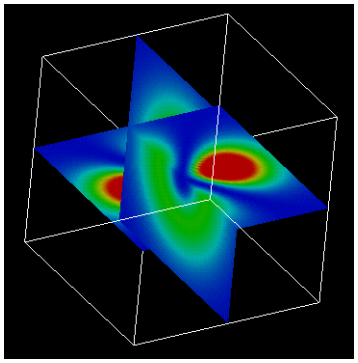
To obtain another slice of the structured field, this time perpendicular to the x-axis, perform the following:

1. Instance another *orthoslice* module. With this module (*orthoslice#1*) do not modify the slice axis; keep it at the default value of 0 (that is, perpendicular to the x-axis).

2. Connect it to the existing network as follows:



The resulting image shows two slices of the structured field: one perpendicular to the x-axis and the other perpendicular to the y-axis.



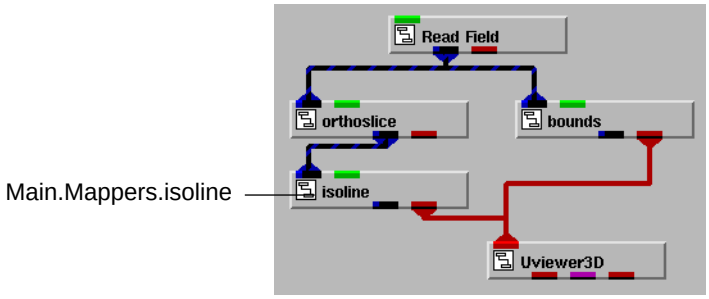
2.3 Creating Isolines

The *isoline* module creates isolines—lines that connect points that have the same value. In this case, the isolines are created on the surface created by the *orthoslice* module.

1. Remove *orthoslice #1* from the network.

In the network editor, right-click *orthoslice #1* and select **Delete**.

2. Instance *isoline* and connect it to the network as shown in the following figure.



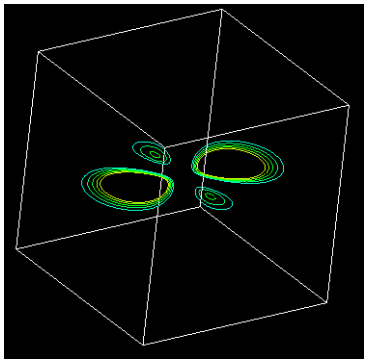
3. Set the number of isolines to 5.

a. In the *SingleWindowApp* window, select **isoline** from the **Modules** pull-down list.

b. Using the slider, set the *number of contours* parameter to 5.

isoline creates five equal ranges from the range of values defined by the *min level* and *max level* parameters. Values that fall within the same range are connected by one isoline.

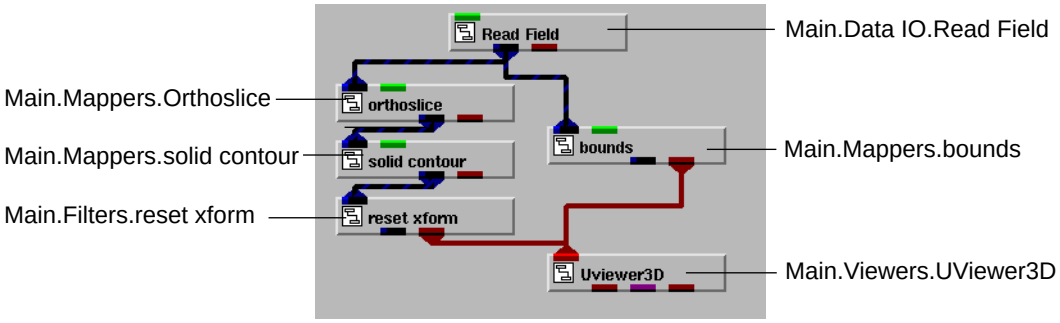
The resulting image shows contour lines on an orthoslice.



2.4 Creating Solid Contours

When you visualize data with contours, you are not restricted to using contour lines. You can create contours of distinct colors by using the *solid_contour* module. Use the Network Editor to set up the network as shown:

1. Instance and connect the modules as shown in the following figure.



2. Read the field data from *hydrogen.fld*.

a. In the *SingleWindowApp* window, select **Read Field** from the **Modules** pull-down list.

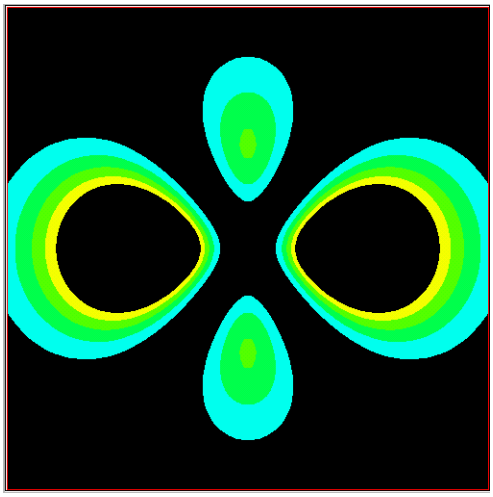
b. The *Read AVS Field Filename* dialog box appears. Click the **Browse...** button and select *hydrogen.fld*.

3. Select the y-axis as the orthoslice axis (the default is the x-axis).

a. In the *SingleWindowApp* window, select **orthoslice** from the **Modules** pull-down list.

b. Using the slider, set the *axis* parameter to 1.

The image in the viewer should appear as shown below:



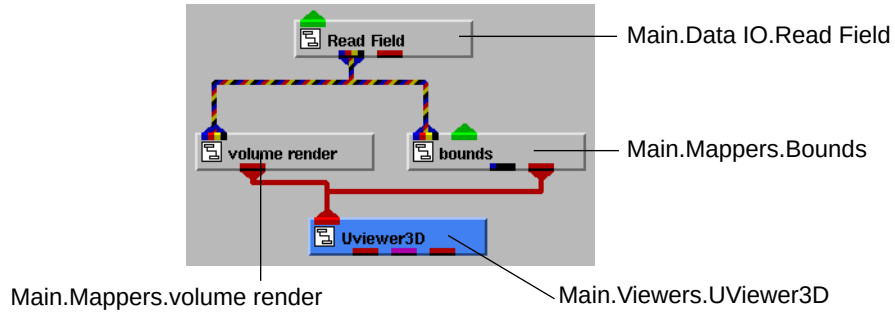
2.5 Performing Volume Rendering

The *volume render* module creates an image from 3D volume data. A field is volume renderable if it has the following properties:

- It is uniform field
- It has node data of the type *byte*.
- Its components has scalar values (that is, *veclen* = 1)
- It is a 3D, 3Space object

This section shows how you volume render the *hydrogen.fld* field, a field dataset that has the properties mentioned above.

1. Instance and connect the modules as shown in the following figure.



2. Read the field data from *hydrogen.fld*.

a. In the *SingleWindowApp* window, select **Read Field** from the **Modules** pull-down list.

b. The *Read AVS Field Filename* dialog box appears. Click the **Browse...** button and select *hydrogen.fld*.

3. Select to use the software renderer. If you select one of the hardware renderers (such as OpenGL or Pex), you must be sure that the graphics board in your system is capable of performing volume rendering.

   a. In the *SingleWindowApp* window, select **Editors -> View** to display the view editor.

   b. From the **View** pull-down list, select **General**.

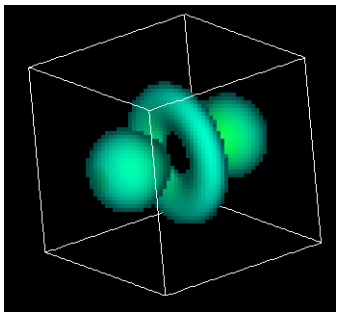
   c. From the **Renderer** pull-down list, select **Software**.

4. Set the maximum alpha parameter to 0.4.

   a. In the *SingleWindowApp* window, select **Editors -> Modules** to display the module editor.

   b. Using the slider, set the *Maximum Alpha* parameter to 0.4.

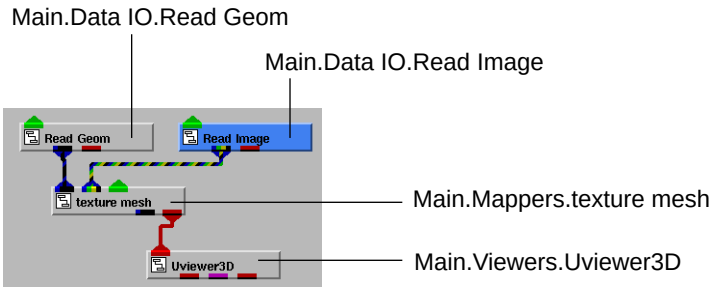
The image in the viewer should appear as in the figure below.



2.6 The *texture mesh* Module

The *texture mesh* module allows you to use a two-dimensional image to cover a surface. The surface can be that of a two- or three-dimensional mesh. In this section, you will use an image (depicting a marble surface) to serve as the surface of a teapot mesh.

1. Instance and connect the modules as shown in the following figure.



2. Read the AVS geometry object contained in *teapot.geo*.

a. In the *SingleWindowApp* window, select **Read Geom** from the **Modules** pull-down list.

b. The *Read GEOM Filename* dialog box appears. Click the **Browse...** button and select *teapot.geo*.

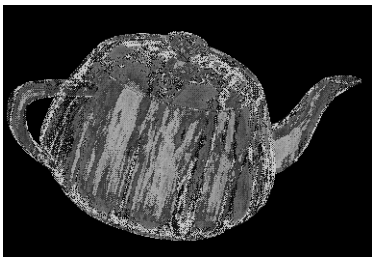
Read Geom reads a binary file containing an AVS geometry object. Such files are named with the *.geo* extension.

3. Read the image contained in *marble.x*.

a. In the *SingleWindowApp* window, select **Read Image** from the **Modules** pull-down list.

b. The *Read IMAGE Filename* dialog box appears. Click the **Browse...** button and select *marble.x*.

4. Rotate the teapot object in the Viewer for better visibility. The image in the viewer should appear as shown below.



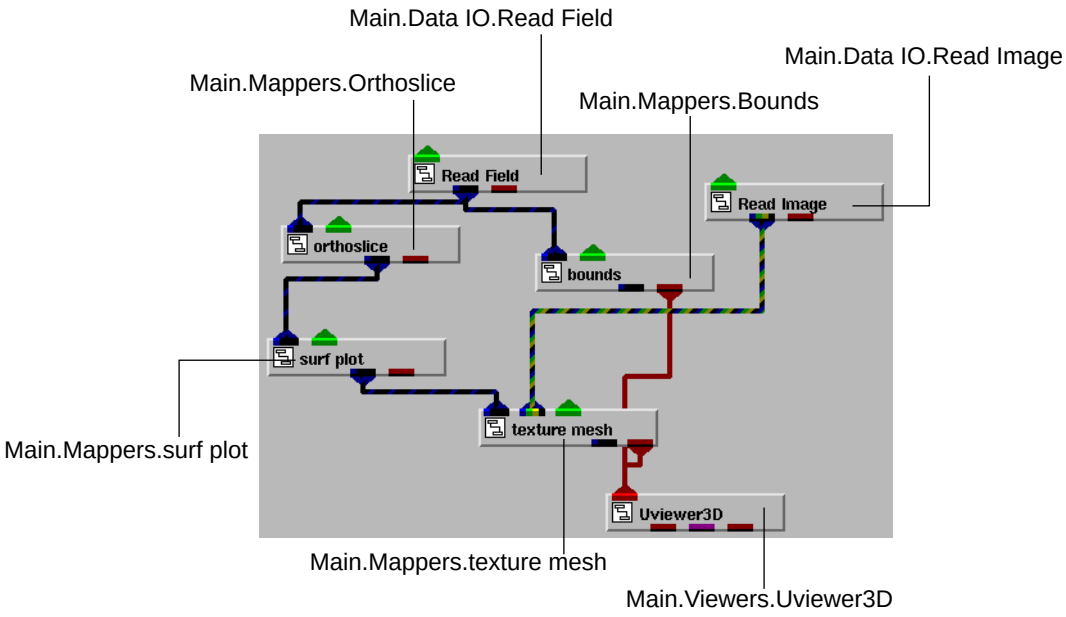
2.6.1 Surface Plot and Texture Mesh

This section shows how you can map texture to a surface that is created from a slice of a field. You will use the following modules: *orthoslice*, *surf plot*, and *texture mesh*.

- *orthoslice* extracts a 2D slice from the field you specify.
- *surf plot* creates a (3Space) surface from the slice by using its contained values.
- *texture mesh* adds texture to the surface by mapping a specified image to the surface.

To create the application:

1. Delete all unnecessary modules in the Network Editor and instance the following modules, connecting them as shown:



2. Read the field data from *hydrogen.fld*.

   a. In the *SingleWindowApp* window, select **Read Field** from the **Modules** pull-down list.

   b. The *Read AVS Field Filename* dialog box appears. Click the **Browse...** button and select *hydrogen.fld*.

3. Set the slice so that it is perpendicular to the z-axis.

   a. In the *SingleWindowApp* window, select **Modules -> orthoslice**.

   b. Using the slider, set the *axis* parameter to 2.

4. Set the height of the surface plot to a lower value.

   a. In the *SingleWindowApp* window, select **Modules -> surf_plot**.

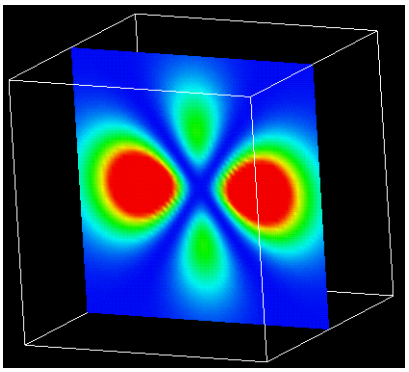
   b. Using the slider, set the *scale* parameter to 0.2.

5. Read the image in *mandrill.x* and use *texture_mesh* to map it onto the surface created in the previous step.

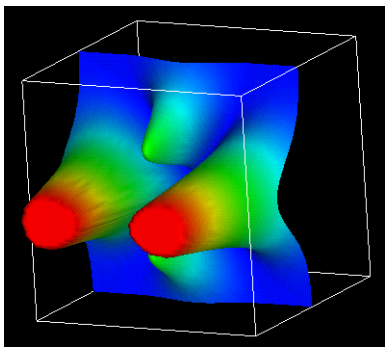
   a. In the *SingleWindowApp* window, select **Read Image** from the **Modules** pull-down list.

   b. The *Read AVS Field Filename* dialog box appears. Click the **Browse...** button and select *mandrill.x*.

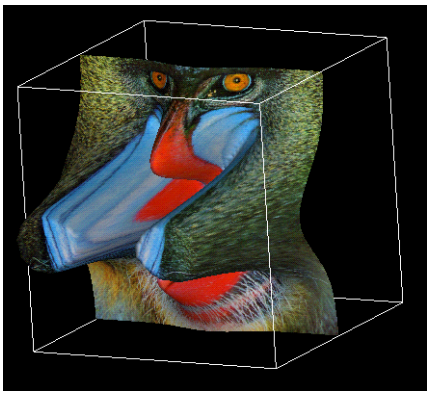
The following figure shows the progression of the application and the resulting image in the viewer.



Using *orthoslice*



Using *surf plot*



Using *texture mesh*

2.7 Specifying Node Locations with Glyphs

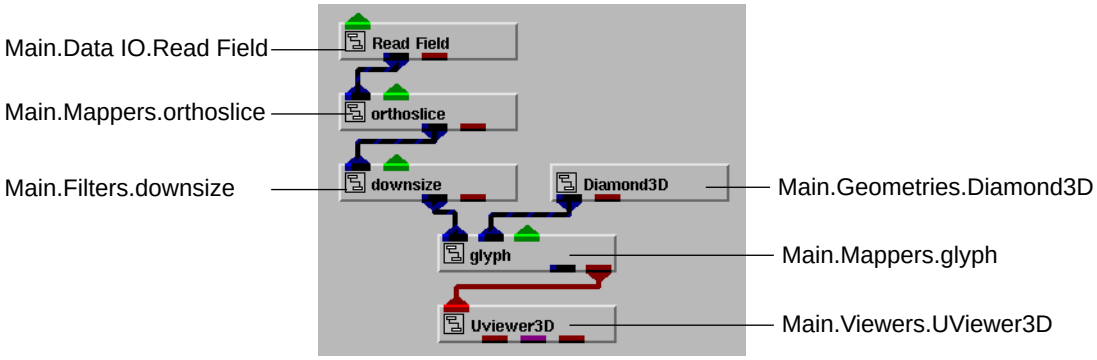
The *glyph* module places a visual geometric object (such as a diamond) at each node location in the mesh portion of a provided field. You can set the size of the glyphs according to component values.

In this section, you will read the *hydrogen.fld* field file and use the *glyph* and other modules as follows:

- *orthoslice* extracts a 2D slice from the field you specify.
- *downsize* subsamples the 2D image, reducing the number of nodes in the mesh.
- *glyph* places a geometric object at each node location.
- *Diamond3D* specifies that the geometric object used by *glyph* is a diamond.

To create the application:

1. Delete all unnecessary modules in the Network Editor and instance the following modules, connecting them as shown:



2. Read the field data from *hydrogen.fld*.

   a. In the *SingleWindowApp* window, select **Read Field** from the **Modules** pull-down list.

b. The *Read AVS Field Filename* dialog box appears. Click the **Browse...** button and select *hydrogen.fld*.

3. Set the slice so that it is perpendicular to the z-axis.

a. In the *SingleWindowApp* window, select **Modules -> orthoslice**.

b. Using the slider, set the *axis* parameter to 2.

4. Set the downsize factor to reduce the number of nodes from 4096 to about 100.

a. In the *SingleWindowApp* window, select **Modules -> downsize**.

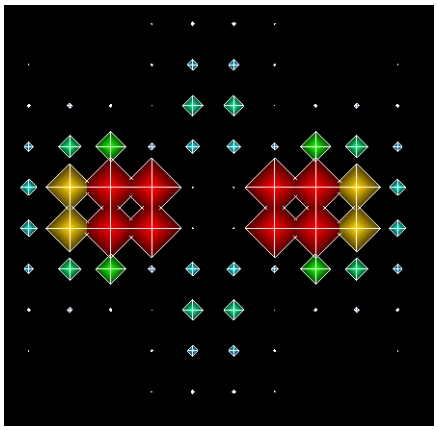
b. Using the slider, set the *I downsize factor* and the *J downsize factor* parameters to 6. *downsize* retains every 6^{th} node along each axis.

5. Set the size of the diamond glyphs.

a. In the *SingleWindowApp* window, select **Modules -> glyph**.

b. Using the slider, set the *scale* parameter to 0.04.

Rotate the resulting image in the viewer to see the effect of the glyphs. The image should appear as shown below.



2.8 Combining Multiple Components

The *combine vect* module combines multiple components of scalar *node data* values to create one-component vector *node data* values. For example, you can use *combine vect* on the following two nodes' (scalar) *node data* values:

- Node0_component0 = 31
- Node0_component1 = 43
- Node0_component2 = 23
- Node1_component0 = 52
- Node1_component1 = 46
- Node1_component2 = 66

The following is the resulting *node data* values after using *combine vect*:

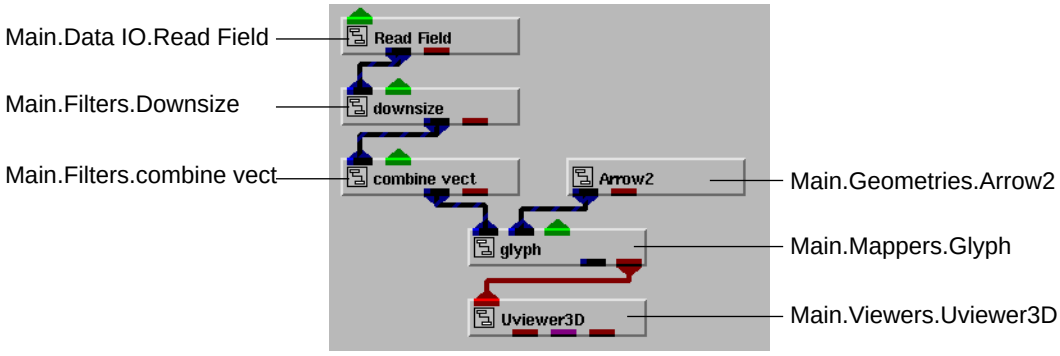
- Combined_node0= {31, 43, 23}
- Combined_node1= {52, 46, 66}

In this section, you will read the *wind.fld* field file and use the *combine vect* and other modules as follows:

- *downsize* subsamples the 3D image, reducing the number of nodes in the mesh.
- *combine vect* combines multiple components of the downsized *scalar* node data structure to create a one-component *vector* node data structure.
- *glyph* places a geometric object at each node location.
- *Arrow2* specifies that the geometric object used by *glyph* is a flat arrow, that depicts the vector value (at that node location) by its size and direction.

To create the application:

1. Delete all unnecessary modules in the Network Editor and instance the following modules, connecting them as shown:



2. Read the field data from *wind.fld*.

   a. In the *SingleWindowApp* window, select **Read Field** from the **Modules** pull-down list.

   b. The *Read AVS Field Filename* dialog box appears. Click the **Browse...** button and select *wind.fld*.

3. Set the downsize factor to reduce the number of nodes.

   a. In the *SingleWindowApp* window, select **Modules -> downsize**.

   b. Using the slider, set the *I downsize factor*, the *J downsize factor*, and *K downsize* parameters to 8. *downsize* retains every 8^{th} node along each axis.

4. Specify that all (three) scalar components of each node are combined to form a single vector component with three elements.

With *combine vect*, the maximum number of scalar components you can combine is 4.

a. In the *SingleWindowApp* window, select **Modules -> combine vect**.

b. Select all the option buttons in the *veclen* parameter. Do not use the slider.

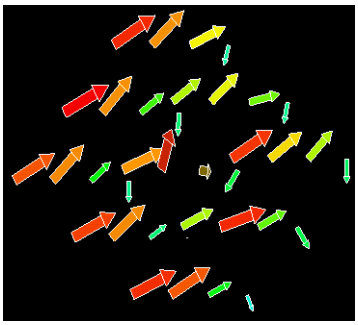
In the output of *combine vect*, each node has data consisting of one component: the component is a vector value with three elements (output value of *veclen* = 3).

5. Set the size of the arrow glyphs.

a. In the *SingleWindowApp* window, select **Modules -> glyph**.

b. Using the slider, set the *scale* parameter to 0.2.

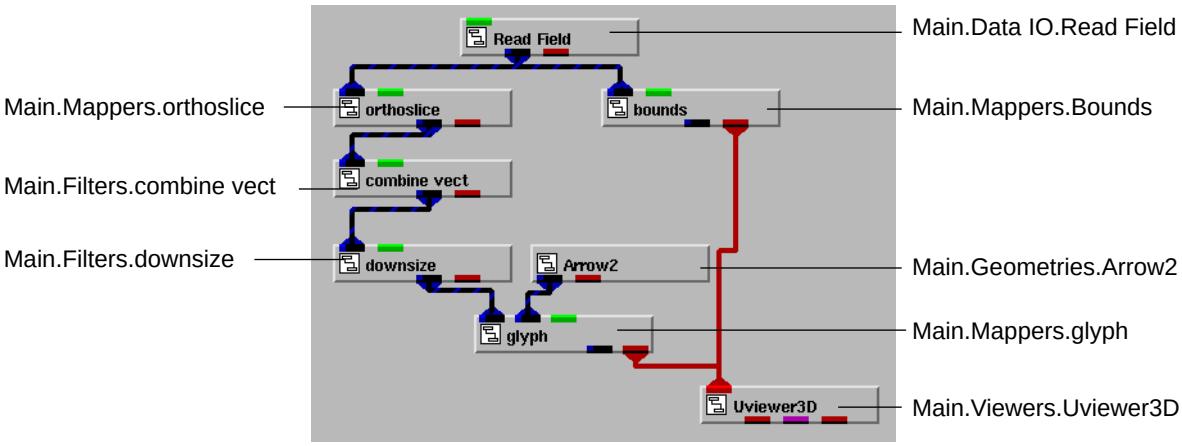
Rotate the resulting image in the viewer to see the effect of the glyphs. The image should appear as shown below.



2.9 Combining Scalar Components of a Slice

This section shows how you can use *combine vect* on a two-dimensional slice of the volume (in *wind.fld*) from the previous section. Instead of combining the three scalar components of the nodes of a volume, *combine vect* combines the three scalar components of only those nodes that fall on a specified two-dimensional slice.

1. Disconnect the modules, instance *orthoslice* and *bounds*, and reconnect the modules in the Network Editor as shown:



2. Set the slice so that it is perpendicular to the y-axis. The resulting node data consists of two components.

a. In the *SingleWindowApp* window, select **Modules -> orthoslice**.

b. Using the slider, set the *axis* parameter to 1.

3. Specify that all (three) scalar components of each node are combined to form a single vector component with three elements. With *combine vect*, the maximum number of scalar components you can combine is 4.

a. In the *SingleWindowApp* window, select **Modules -> combine vect**.

b. Select all the option buttons in the *veclen* parameter. Do not use the slider.

In the output of *combine vect*, each node has data consisting of one component: the component is a vector value with three elements (output value of *veclen* = 3).

4. Set the downsize factor to reduce the number of nodes.

a. In the *SingleWindowApp* window, select **Modules -> downsize**.

b. Using the slider, set the *I downsize factor* and the *J downsize factor* parameters to 5. *downsize* retains every 5^{th} node along each axis.

Notice that because *orthoslice* was used on the volume, the number of axes is now 2.

5. Set the size of the arrow glyphs.

a. In the *SingleWindowApp* window, select **Modules -> glyph**.

b. Using the slider, set the *scale* parameter to 0.2.

Rotate the resulting image in the viewer to see the effect of the glyphs. The image should appear as shown below.

Notice that since *bounds* is connected to *Read Field*, the bounds of the box enclose data that was excluded because of orthoslice. Also notice that the glyphs in the image are positioned on a plane that is perpendicular to the y-axis.