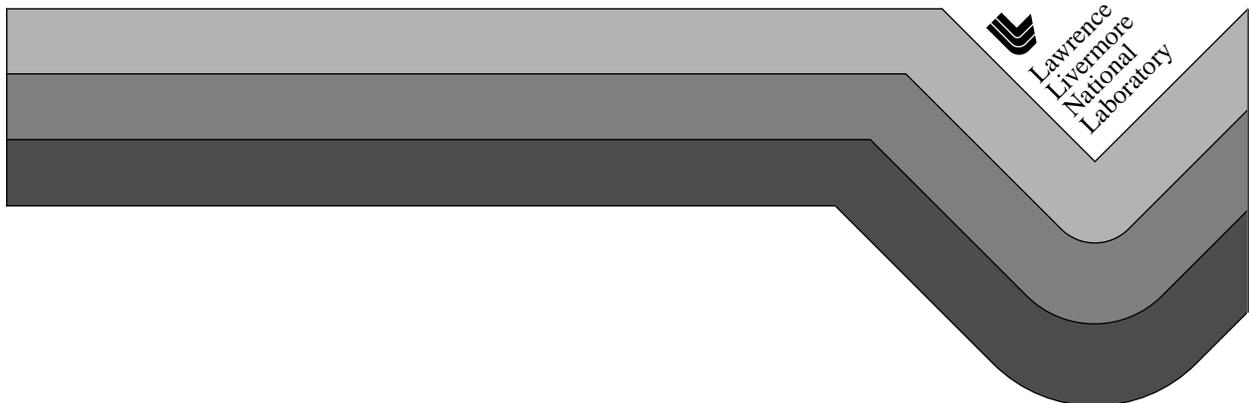


VisIt User's Manual

October 2005

Version 1.5



DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

Table of Contents

Introduction to VisIt

Manual chapters	13
Manual conventions	15
Understanding how VisIt works	15
Starting VisIt	16
The Main Window	17
Posting a window	19
Using the main menu	19
Viewing status messages	21
Applying settings	21
Getting started	22

Working with databases

Supported File Types	23
File extensions	23
File Selection Window	28
Changing hosts	29
Changing directories	30
Default directory	30
Changing filters	30
Managing the selected file list	31
Grouping files	31
Virtual databases	31
Refreshing the file list	33
Clearing out recently visited paths	33
Connecting to a running simulation	33
File panel	34
Opening a file	35
Opening a file on a remote computer	36
Opening a time-varying database	36
Setting the active time step	37
Playing animations	37
Reopening a database	37
Replacing a database	37
Overlaying a database	38
File Information Window	38
Advanced file options	39
Closing a database	39

Plots

Plots	41
Managing plots	41
Standard Plot Types	44
Boundary and FilledBoundary Plots	45
Contour Plot	49
Curve Plot	51
Histogram Plot	52
Label Plot	54
Mesh Plot	58
Pseudocolor plot	60
Scatter Plot	64
Streamline Plot	67
Subset Plot	72
Surface Plot	75
Tensor plot	78
Truecolor plot	79
Vector plot	80
Volume plot	82

Operators

Operators	87
Managing operators	87
Operator Types	92
Box operator	93
Clip operator	94
Cone operator	96
Cylinder operator	98
Decimate operator	98
DeferExpression operator	100
Displace operator	100
Elevate operator	101
ExternalSurface operator	103
Index Select operator	103
InverseGhostZone operator	105
Isosurface operator	106
Isovolume operator	108
Lineout operator	109
Merge operator	109
OnionPeel operator	109
Project operator	111
Reflect operator	112
Revolve operator	114
Resample operator	115
Slice operator	118
Smooth operator	120

SphereSlice operator	121
ThreeSlice operator	122
Threshold operator	123
Transform operator	125
Tube operator	128

Saving and Printing

Saving the visualization window	129
The Save Window	130
Picking an output directory for saved files	130
Setting the save file name	131
Setting the file type	131
Saving images with screen capture	132
Setting image resolution	132
Saving stereo images	132
Saving binary geometry files	132
Saving tiled images	133
Saving movies	133
Choosing movie formats	134
Making a stereo movie	135
Choosing the movie name	136
Choosing movie generation method	137
Exporting databases	138
Exporting variables	139
Choosing an export file format	139
Printing	140
The Printer Window	140
Setting the printer destination	141
Changing the color settings	141
Setting paper format	141
Setting the number of printed copies	141

Visualization Windows

Managing vis windows	143
Adding a new vis window	143
Deleting a vis window	144
Clearing plots from vis windows	144
Changing window layouts	145
Setting the active window	145
Using vis windows	148
Navigate mode	148
Zoom mode	148
Lineout mode	149
Pick mode	149
Interactor settings	149
Zoom interactor settings	149
Navigation styles	150

The Popup menu and the Toolbar	150
Hiding toolbars	151
Moving toolbars	151
Switching window modes	151
Activating tools	152
View options	153
Animation options	155
Window options	156
Clear options	158
Plot options	159
Operator options	160
Lock options	162

Subsets

What is a subset?	163
Subset Inclusion Lattice	164
Using the Subset Window	164
Browsing subsets	165
Changing a SIL restriction	166
Creating complex subsets	167
Turning multiple sets on and off	168
Material Interface Reconstruction	169
Choosing a MIR algorithm	169
Finding materials with low volume fractions	171
Simplifying heavily mixed cells	171
Smoother material boundary interfaces	172
Forcing material interface reconstruction	172
Mixed variables	173
Species	173
Plotting species	175
Turning off species	175

Quantitative Analysis

Expressions	177
Expression Window	177
Expression grammar	180
Built-in expressions	183
Query	207
Query types	208
Built-in queries	209
Executing a query	214
Querying over time	214
Pick	216
Pick mode	216
Pick Window	218
Lineout	220
Lineout mode	220

Curve plot	222
Lineout Operator	223

Making it Pretty

Annotations	227
Annotation Window	228
General Annotations	228
2D Annotations	229
3D Annotations	231
Annotation Colors	235
Annotation Objects	236
Color tables	244
Color table window	245
Lighting	247
Lighting Window	248
Rendering Options	250
Making lines look smoother	250
Specular lighting	251
Shadows	252
View	252
View Window	253
Advanced view features	258
View and data limits	259

Animation and Keyframing

Animation	261
The .visit file	261
Flipbook animation	262
Animation Window	263
Keyframing	264
Keyframing Window	264
Session files	267
Saving session	268
Restoring session	268
Scripting	268
Command Window	268
Movie tools	270

Interactive Tools

Introduction to interactive tools	273
Box Tool	274
Line Tool	275
Plane Tool	276
Point Tool	278
Sphere Tool	278

Multiple Databases and Windows

Databases	281
------------------------	------------

Activating a database	281
Multiple time sliders	283
Database correlations	285
Database correlations and time sliders	285
Types of database correlations	285
Managing database correlations	289
Database comparison	292
The role of expressions	292
Plotting the difference between two databases	292
Plotting values relative to the first time state	293
Plotting time derivatives	294
Multiple Window Operations	295
Reflection and Translation	295
Copying Windows	296
Locking Windows	297
 Remote Visualization	
Distributed mode	301
Passwords	302
Environment	303
Launch progress window	303
Host Profiles	304
Host Profile Window	304
Setting parallel options	307
Advanced host profile options	310
Engine Option Window	312
Managing compute engines	313
Compute Engine Window	314
Simulation Window	315
 Setting Preferences	
How VisIt uses preferences	317
Setting default values	317
How to save your settings	318
Appearance Window	319
Changing GUI colors	319
Changing GUI Style	319
Changing GUI Orientation	320
Plugin Manager Window	322
Enabling and Disabling Plugins	322
Preferences Window	322
Rendering Options Window	326
Changing surface representations	326
Using display lists	327
Stereo images	327
Scalable rendering	328
Frames per second	329

Triangle count.....	329
Plot Extents.....	329

Help

About VisIt	331
Help Window.....	332
Help Window Toolbar	332
Selecting a help page	333

Chapter 1

Introduction to VisIt

1.0 Overview

VisIt is a free, open source, platform independent, distributed, parallel, visualization tool for visualizing data defined on two- and three-dimensional structured and unstructured meshes. VisIt's distributed architecture allows it to leverage both the compute power of a large parallel computer and the graphics acceleration hardware of a local workstation. VisIt's user interface is often run locally on a Windows, Linux, or MacOS X desktop computer while its compute engine component runs in parallel on a remote computer. VisIt's distributed architecture allows VisIt to visualize simulation data where it was generated, eliminating the need to move the data to a visualization server. VisIt can be controlled by its Graphical User Interface (GUI), through the Python and Java programming languages, or from a custom user interface that you develop yourself. More information about VisIt can be found online at <http://www.llnl.gov/visit>.

This manual explains how to use the VisIt GUI. You will be given a brief overview on how VisIt works and then you will be shown how to start and use VisIt.

2.0 Manual chapters

This manual is broken down into the following chapters:

Chapter title	Chapter description
Introduction	This chapter.
Working with files	Describes how to select and open files for visualization.

Chapter title	Chapter description
Plots	Describes the concept of a plot and how to create them as well as detailed information about all VisIt plots.
Operators	Describes the concept of an operator and how to create them as well as information on all VisIt operators.
Saving and Printing	Describes how to save and print images.
Visualization windows	Describes how to interact with VisIt's visualization windows.
Subsetting	Describes the concept of a subset and tells how to create subsets.
Quantitative Analysis	Describes how to extract quantitative data from visualizations. This includes pick and reference lines.
Making it Pretty	Describes how to improve the presentation quality of visualizations using colors, annotations, lighting and view.
Animation and Keyframing	Describes how to view time varying databases as animations.
Interactive Tools	Describes how to interactively slice plots using VisIt's interactive tools.
Multiple Databases and Windows	Describes how to use database correlations and how to visualize databases using multiple visualization windows.
Remote Visualization	Describes how to run VisIt's compute engine on remote computers.
Preferences	Describes how to set GUI look and feel preferences.
Help	Describes how to use VisIt's online help.
Appendix A	Describes VisIt's command line options.
Appendix B	Describes how to set up password-less ssh.

Chapter title	Chapter description
Appendix C	Describes how to install VisIt.

3.0 Manual conventions

This manual uses the following conventions:

Element	All GUI elements, like windows, menus, and buttons will use bold helvetica .
Chapters	All references to other chapters will use Bold Times .
<i>Documents</i>	All document names will be <i>italicized</i> .

4.0 Understanding how VisIt works

VisIt visualizes data by creating one or more plots in a visualization window, also known as a vis window. A plot is a visual representation of the data being examined. Examples of plots include Mesh plots, Contour plots and Pseudocolor plots. Plots take as input one or more scalar or vector variables, which you can modify, by applying operators before passing them to a plot. Examples of operators include arithmetic operations or taking slices through the mesh. It is also possible to restrict the visualization of the data to subsets of the mesh.

VisIt supports up to 16 visualization windows. Each vis window is independent of the other vis windows. VisIt uses an active window concept; all changes made in VisIt's **Main Window** or one of its popup windows apply to the currently active vis window.

VisIt reads its data and performs most of its processing in compute engine processes. A compute engine is launched on each machine where data to be visualized is located. The **Host Profiles Window** is used to specify properties about the compute engines for different machines, such as the number of processors to use when running the engine. The status of a compute engine is displayed in the **Compute Engines Window**. The **Compute Engines Window** can also be used to interrupt pending operations.

VisIt's architecture can be broken down into four main components, though there are some other less important components. The first component, and the one covered by this document, is the GUI. The GUI provides the user interface and menus that let you easily choose what to visualize. The Viewer displays all of the visualizations in its vis windows and is responsible for keeping track of VisIt's state and for talking to the rest of VisIt's components. Both the GUI and the Viewer are meant to run on the local client computer so they can take advantage of the client computer's fast graphics hardware.

The next two components can also be run on the client computer but they are more often run on a remote, parallel computer or cluster where the data files were generated. The first such component is the database server, which is responsible for reading the remote file system and passing information about the files there to the GUI on the local computer. The database server also opens a file to determine its list of variables and other metadata that are useful in creating visualizations. Finally, the compute engine is the component that actually reads the data files, processes them, and sends back either images or geometry to be drawn by the viewer using the local computer's fast graphics hardware. Figure 1-1 shows connections between VisIt components.

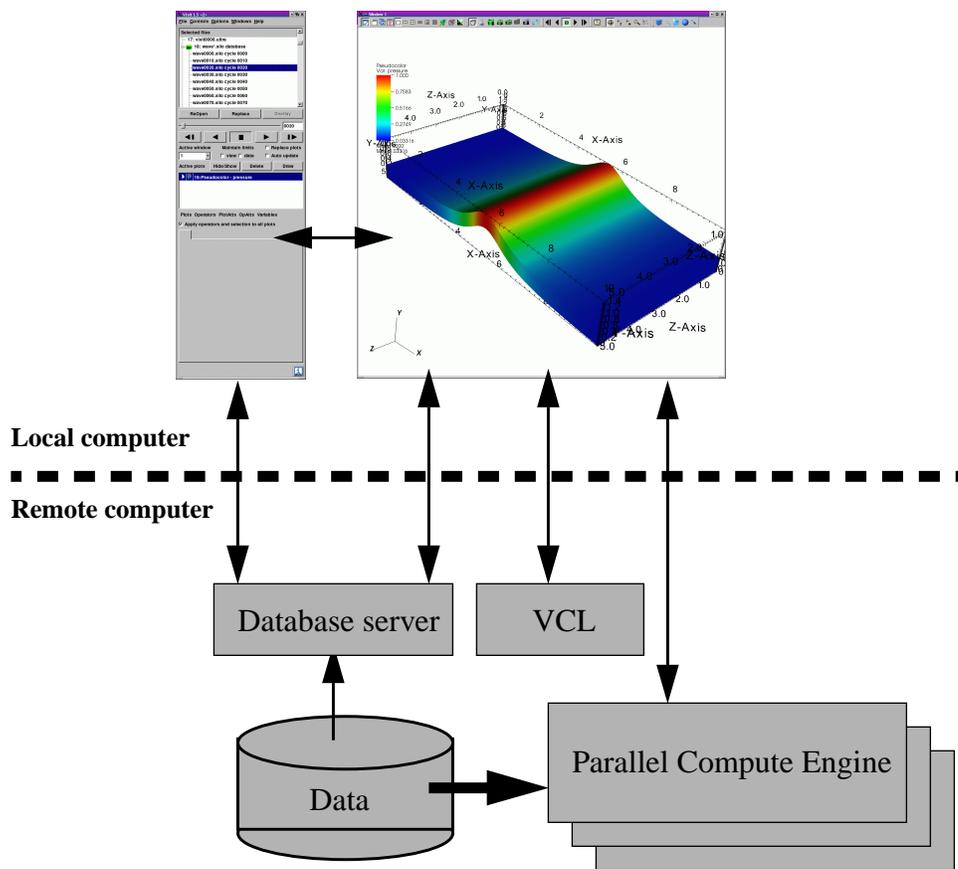


Figure 1-1: VisIt's architecture

5.0 Starting VisIt

You can invoke VisIt from the command line by typing: **visit**

On the Windows platform, the VisIt installation program adds a VisIt program group to the Windows Start menu and it adds a VisIt shortcut to the desktop. You can double-click on the desktop shortcut or use the VisIt option in the Start menu's VisIt program group to

launch VisIt. In addition to creating shortcuts, the VisIt installation program creates file associations for *.silo*, *.visit*, and *.vses* files so double-clicking on files with those extensions opens them with VisIt.

When you run VisIt at the command line, you can provide various command line options, which are listed in **Appendix A**. It is best to have VisIt in your default search path instead of specifying the absolute path to VisIt when starting it. Having VisIt in your default search path isn't important when VisIt is run locally, but VisIt may not run properly in distributed mode if the `visit` command isn't in your default search path on all the machines on which you are running VisIt. When VisIt first launches, it opens two windows that fill as much of the screen as possible. Figure 1-2 contains the most common window layout.

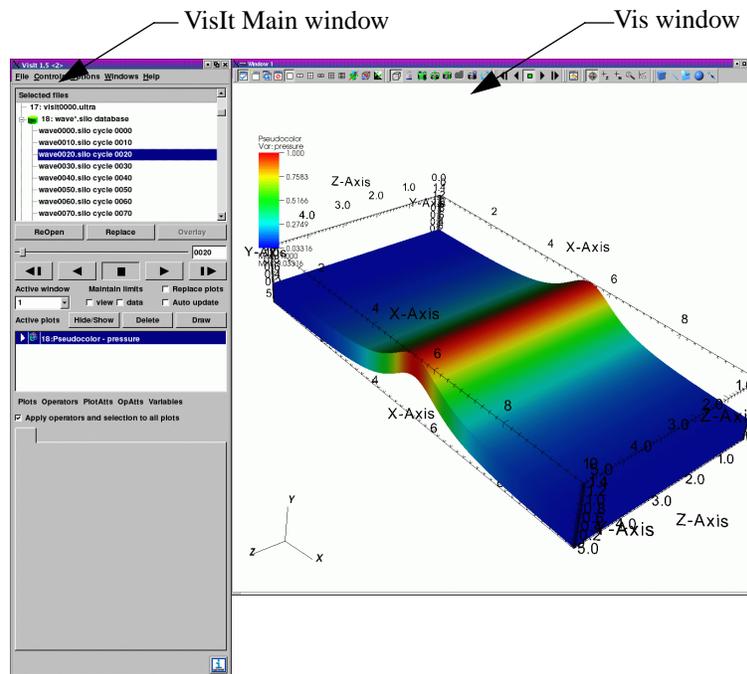


Figure 1-2: VisIt windows

6.0 The Main Window

VisIt's **Main Window**, shown in Figure 1-3, contains three main areas: the **File panel**, **Plot Manager**, and **Notepad** area. The **File panel** is located at the top of the **Main Window** and it allows you to open databases and set the active time step for animations. The middle area of the **Main Window** is the **Plot Manager** area. The **Plot Manager** area contains controls that allow you to create and modify plots and operators. The bottom area of the **Main Window** is the **Notepad** area. The **Notepad** area is a blank area to which various VisIt windows can post. Each time a window posts to the **Notepad** area, a

new tab is created in the **Notepad** and the posted window's contents are added to the new tab. Clicking on a tab in the notebook displays a posted window so that it can be used.

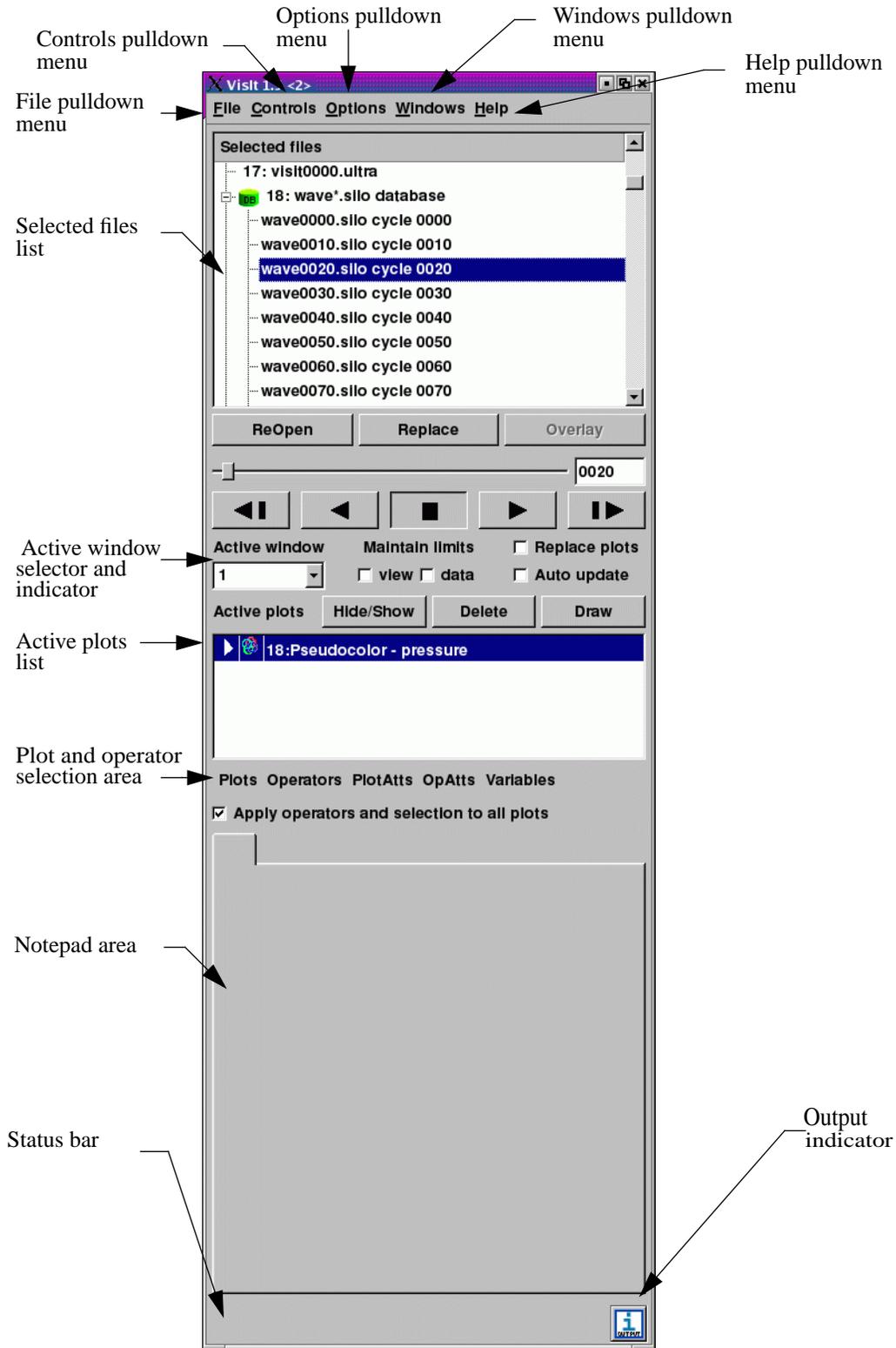


Figure 1-3: VisIt's Main Window

6.1 Posting a window

Windows that have a **Post** button can be posted to the **Main Window's Notepad** area. Clicking on a window's **Post** button hides the window and adds its controls to a new tab in the **Notepad** area. Posting windows allows you to have several windows active at the same time without cluttering the screen. When a window is posted, its **Post** button turns to an **UnPost** button that, when clicked, removes the posted window from the **Notepad** area and displays the window in its own window. Figure 1-4 shows an example of a window with a **Post** button and also shows the same window when it is posted to the **Notepad** area.

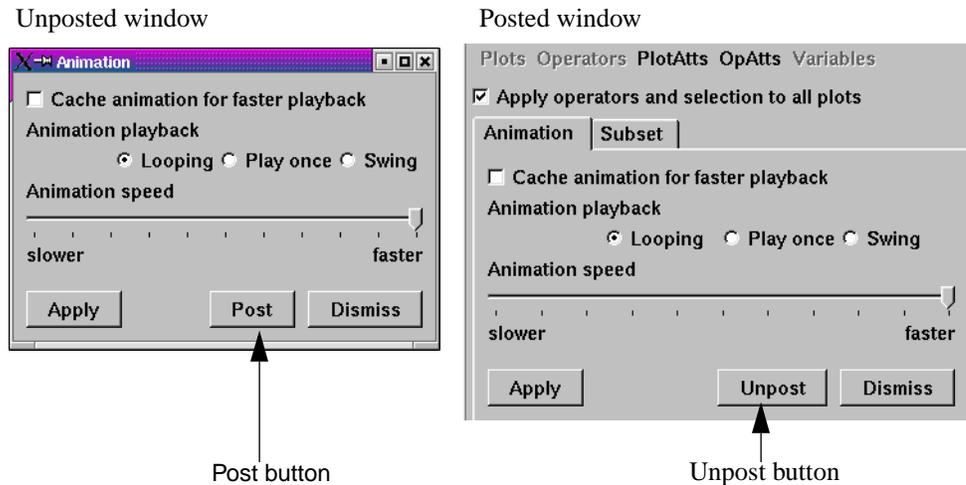


Figure 1-4: Animation Window with Post button

6.2 Using the main menu

VisIt's **Main Menu** contains five menu options that allow you to access many of VisIt's most useful features. Each menu option displays a submenu when you click it. The options in the submenus perform an action such as saving an image. Menu options that contain a name followed by ellipsis open another VisIt window. Some menu options have keyboard shortcuts that activate windows. The **File** menu contains options that open VisIt windows that allow you to open files, manage compute engines and host profiles, save images, and print images. The **Controls** menu contains options that open VisIt windows that, for the most part, set the look and feel of VisIt's visualization windows. The colors, annotations, lighting, and view can be set through some of the options available in the **Controls** menu. The **Options** menu contains options that allow you to set the appearance of the GUI, manage VisIt plugins, and save VisIt's settings to a configuration file. The **Windows** menu contains controls that manage visualization windows. The **Help** menu provides options for viewing online help, VisIt's copyright agreement, and release notes which describe the major enhancements and fixes in each new version of VisIt. The options for each menu are shown in Figure 1-5 and will be described in detail later in this manual.

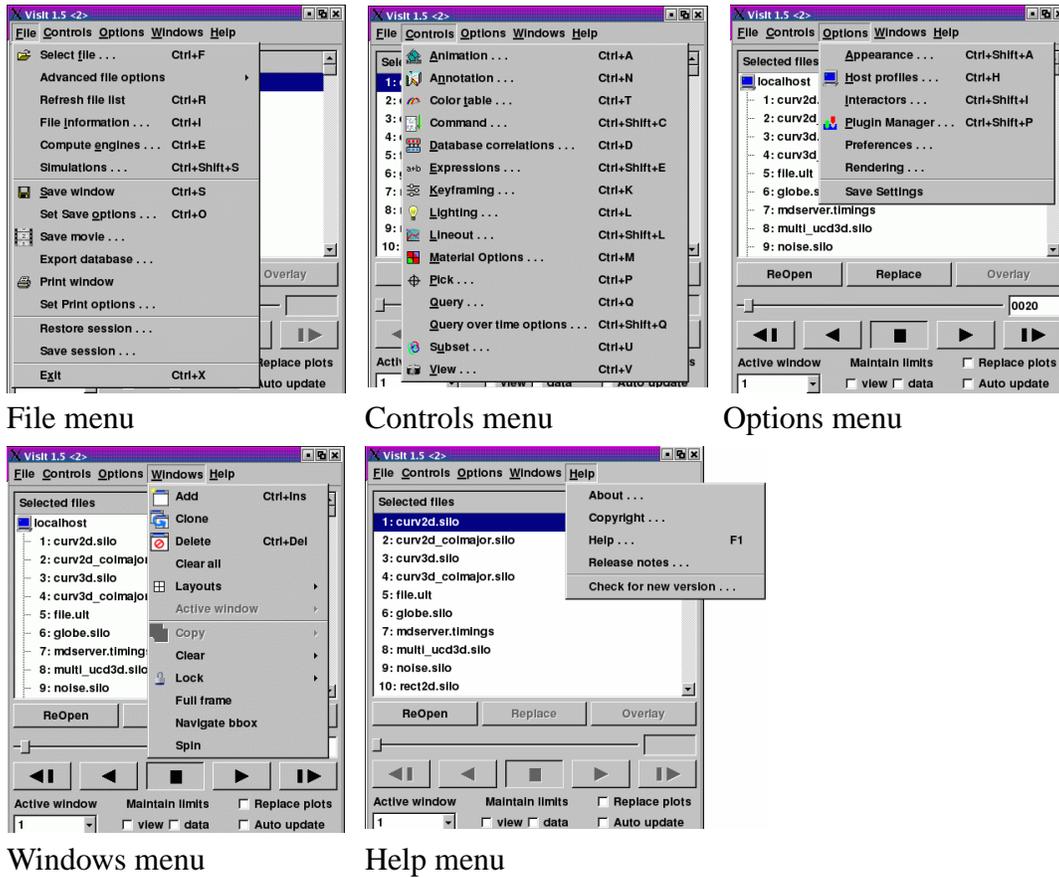


Figure 1-5: VisIt's main menus

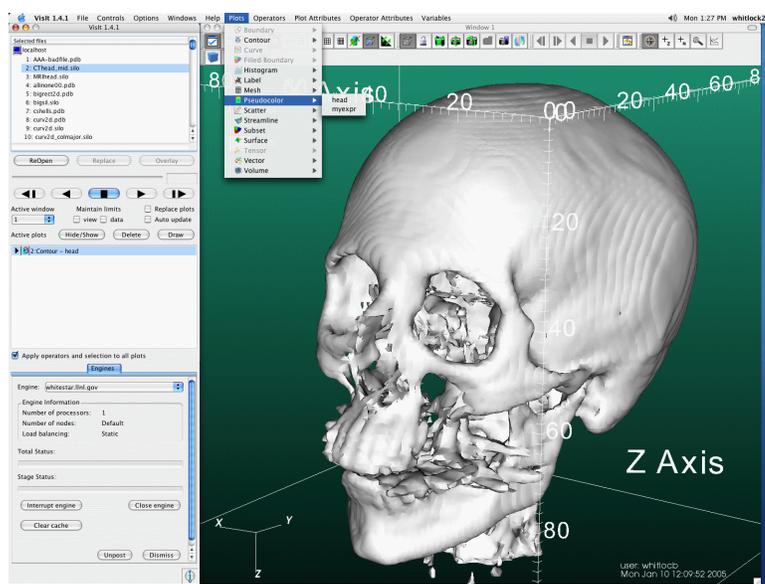


Figure 1-6: Main Menu and Plot and Operator menu are combined in MacOS X

The **Main Menu** and the **Plot and operator** menu are merged in the MacOS X version of VisIt because MacOS X applications always have all menus in the system menu along the top of the display.

6.3 Viewing status messages

VisIt informs the user of its progress as it creates a visualization. As work is completed, status messages are displayed in the bottom of the **Main Window** in the status bar. In addition to status messages, VisIt sometimes displays error or warning messages. These messages are displayed in the **Output Window**, shown in Figure 1-7. To open the **Output Window**, click the **Output Indicator** in the lower, righthand corner of the **Main Window**. When the **Output Window** contains an unread message, the **Output Indicator** changes colors from blue to red..

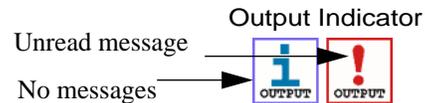


Figure 1-7: Output window and output indicator

6.4 Applying settings

When using one of VisIt's control windows, you must click the **Apply** button for the new settings to take effect. All control windows have an **Apply** button in the lower left corner of the window. By default, new settings are not applied until the **Apply** button is clicked because it is more efficient to make several changes and then apply them at once. VisIt has a mode called *auto update* that makes all changes in settings take place immediately. Auto update is not enabled by default because it can cause plots to be regenerated each time settings change and for the database sizes for which VisIt is designed, autoupdate may not always make sense. If you prefer to have new settings apply immediately, you can enable auto update by clicking on the **Auto update** check box in the middle of the **Main Window**. If auto update is enabled, you do not have to click the **Apply** button to apply changes.

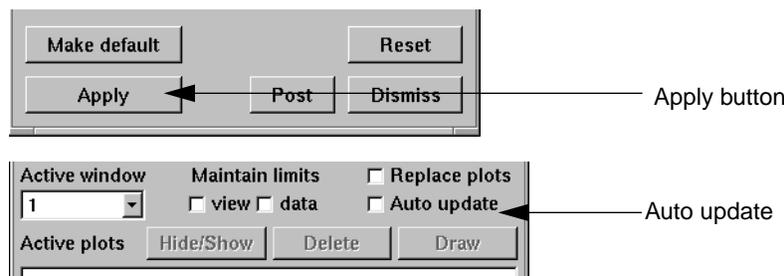


Figure 1-8: Apply button and Autoupdate check box

7.0 Getting started

The rest of this manual details the ins and outs to using VisIt, but you can also very quickly visualize your data by opening a database and creating plots. You must first select databases to visualize. Sample data files are usually installed with VisIt in a data directory in the directory in which VisIt was installed. If you are running VisIt on the Windows platform, you can double-click on one of the sample Silo data files to open it in VisIt or you can run VisIt and open the **File Selection Window** from the **Main Window's File** menu. Highlight some files and click the **Ok** button in the **File Selection Window**. The **Main Window's File panel** should now contain some files. To open a file, click on its entry in the File panel and then click the **Open** button. If the database was successfully opened, the **Plot and operator** menu will be enabled.

Once you have opened a database, you can use it to create a plot by selecting a plot type and database variable from the **Plots** menu. Once a plot is created, the **Active plots list** will show that the new plot has been added by displaying a description of the plot drawn in green text. The color green indicates that the plot is in the new state and has not been drawn yet. To draw the plot, click the **Draw** button in the middle of the **Main Window**. That's all there is to creating a plot using VisIt. For more detailed information on creating plots and performing specific actions in VisIt, refer to the other chapters in this book.

1.0 Overview

In this chapter, we will discuss how to work with databases in VisIt. A database can be either a set of files on disk or a running simulation. You can manage both types of databases using the same VisIt windows. First we'll learn about the **File Selection Window** which allows databases to be selected and grouped. Next, we'll learn how to open databases for visualization using the **File Panel** before learning how to examine information about a database using the **File Information Window**. Finally, we'll learn how to reopen a file and close a database.

2.0 Supported File Types

VisIt can create visualizations from databases that are stored in many types of underlying file formats. VisIt has a database reader for each supported file format and the database reader is a plugin that reads the data from the input file and imports it into VisIt. If your data format is not listed in Table 1, "File formats supported by VisIt," on page 24 then you can first translate your data into a format that VisIt can read (e.g. Silo, VTK, etc.) or you can create a new database reader plugin for VisIt. For more information on developing a database reader plugin, refer to the *VisIt Plugin Developer's Guide* or send an e-mail inquiry to visit-help@llnl.gov.

2.1 File extensions

VisIt uses file extensions to decide which database reader plugin should be used to open a particular file format. Each database reader plugin has a set of file extensions that are used to match a filename to it. When a file's extension matches (case sensitive except on MS Windows) that of a certain plugin, VisIt attempts to load the file with that plugin. If the

plugin cannot load the file then VisIt attempts to open the file with the next suitable plugin, before trying to open the file with the default database reader plugin. If your files do not have file extensions then VisIt will attempt to use the default database reader plugin. You can provide the `-default_format` command line option with the name of the database reader plugin to use if you want to specify which reader VisIt should use when first trying to open a file. For example, if you want to load a PDB/Flash file, which usually has no file extension, you could provide: `-default_format PDB` on the command line.

TABLE 1 File formats supported by VisIt

Database plugin	File extension(s)	Description
ANALYZE	img, hdr	Rectilinear grid of integer values containing MRI and fMRI data for human brains.
ANSYS	inp	ANSYS is a popular commercial suite of multi-physics codes. The VisIt ANSYS reader reads in the ASCII version of the ANSYS mesh input file format.
AUXFile	aux	LLNL ASCII file format containing target and beam diagnostic data for National Ignition Facility (NIF) lasers.
BOV	bov	Brick of values format where the database can be broken down into small bricks of ZLIB compressed scalar arrays.
Boxlib2D	boxlib2d, boxlib2D	Structured AMR format for 2D. Requires a .visit file to group multiple time states into a single time-varying database.
Boxlib3D	boxlib3d, boxlib3D	Structured AMR format for 3D. Requires a .visit file to group multiple time states into a single time-varying database.
CGNS	cgns	CFD General Notation System (CGNS) files contain platform-independent binary CFD (computational fluid dynamics) data for both unstructured and structured grids. For more information on CGNS, visit http://www.cgns.org .
CMAT	cmat	ASCII output of CMAT Fortran simulation.
Cosmos	cosmos	Astronomical simulation data saved in HDF4 format.
CosmosPP	cosmospp, cosmos++	Astronomical simulation data saved in HDF5 format.

Database plugin	File extension(s)	Description
Curve2D	curve, ultra, ult	ASCII text file containing 2 columns of X,Y pairs that describe a curve. Each curve is preceded by a “#” comment containing the name of the variable.
Dune	Dat	ASCII file containing point meshes and variables written by Dune simulation code.
EnSight	case	EnSight Gold case files.
Enzo	boundary, hierarchy	Stanford University AMR Astrophysics simulation output saved in HDF4 format.
Exodus	ex, e, exo, ex2, exII, exii, gen, EX, E, EXO, EX2, EXII, GEN, exodus, EXODUS, nemesis, NEMESIS	Sandia National Laboratory’s file format for storing simulation data.
FLASH	none	AMR data stored in HDF5 file format produced by the FLASH code from the University of Chicago. The <i>-default_format FLASH</i> command line options must be specified to select this database reader plugin because the plugin is not associated with any file extensions.
GDAL	adf, asc, bt, ddf, dem, ecw, gxf, jp2, map, mem, mpl, mpr, n1, nat, ntf, pix, rsw, sid, vrt, xpm	GDAL is a translator library for raster geospatial data formats common in the Geographic Information Systems (GIS) field. VisIt’s GDAL reader can read the formats that GDAL itself can read including popular GIS formats such as: ArcInfo binary grid and Digital Elevation Map (DEM). For a complete list of the file formats that VisIt can access through the GDAL reader plugin, refer to http://www.gdal.org/formats_list.html . Note that many of the formats that GDAL can read are actually image file formats. VisIt reads image file formats through its Image database reader plugin unless you tell the GDAL plugin to read image files by setting the VISIT_READ_IMAGES_WITH_GDAL environment variable.

Database plugin	File extension(s)	Description
Image	BMP, JPEG, JPG, PNG, PNM, PPM, SDT, SPR, TIF, TIFF, bmp, imgvol, jpeg, jpg, png, pnm, ppm, sdt, spr, tif, tiff	The Image database reader plugin can read in many popular image file formats, allowing you to plot images of experimental data or perform image analysis. In addition, the Image database reader plugin supports the creation of image volumes, which are comprised of a set of 2D slice images that can be reassembled into a 3D volume. An image volume file is a text file (ending in .imgvol) that contains the names of the images to be reassembled into a 3D volume.
KullLite	pdb, mkf	Input mesh files for Kull simulation program. Data files are stored in PDB format.
Lines	dat	ASCII text format containing X,Y pairs or X,Y,Z triples.
Mili	m, mili	Popular LLNL engineering format used in Dyna3D, Nike3D simulation codes. Requires use of the <i>visit - makemili</i> utility to create a .mili file that can be opened.
NASTRAN	nas, f06	NASTRAN is a popular commercial finite element tool. The VisIt NASTRAN reader plugin can import geometry from NASTRAN bulk data files.
NETCDF	nc, cdf, elev	NetCDF (Network Common Data Form) files are used to store scientific, array-oriented data in a machine-independent, binary format. NetCDF files often contain data from climate observations and results for climate simulations.
OVERFLOW	dat, save	Binary Fortran output files containing overlaid curvilinear meshes.
PATRAN	neu	PATRAN is a popular commercial finite element code. The VisIt PATRAN reader reads PATRAN neutral files, which are ASCII format files that contain unstructured geometry and simulation results.

Database plugin	File extension(s)	Description
PDB/Flash	pdb, r0000	The PDB database plugin is an umbrella reader for multiple styles of files written in the PDB database format. Flash files are read by this plugin. Equation of state databases in PDB format can also be read using this plugin. Finally, the PDB plugin supports reading wavelet-compressed data files generated with Pf3D simulation code.
Pixie	h5	2D and 3D LANL simulation data saved in HDF5 format.
Plot2D	p2d	Structured data format for simulation data in Plot2D format.
Plot3D	q, x	Structured data format for simulation data in Plot3D format.
Point3D	3D	ASCII file format containing four columns of numbers: X,Y,Z point and a data value.
SAF	saf	Arbitrary simulation data stored in HDF5 file format by Sets and Fields (SAF) library.
SAMRAI	samrai	Structured AMR data stored in HDF5 file format. Requires a <i>.visit</i> file to group multiple time states into a single time-varying database.
SAR	SAR, sar	Specific Absorption Rate image volume files.
STL	stl	Stereolithographic file format containing triangle coordinates
Shapefile	dbf, shp	ESRI Shapefiles are commonly used in GIS applications to store vector data (boundaries, roads, building footprints, etc.).
Silo	pdb, silo	Popular LLNL file format based on PDB that supports scalar, vector fields, and materials stored on rectilinear, curvilinear, unstructured, or point meshes. Data can be split up into multiple domains.
Spheral	sv, spheral	Spheral files contain results of coupled hydrodynamical and gravitational simulations in ASCII form.
TecPlot	plt, tec, tp	ASCII file format for the popular TecPlot plotting package.

Database plugin	File extension(s)	Description
Tetrad	h5, hdf5	Tetrahedral meshes and variables stored in HDF5 file format.
TFT	dat, tft	ASCII output of TFT Fortran simulation.
Time Varying Exodus	exII	Same as Exodus
VTK	vtk	ASCII files containing data from multiple kinds of objects from Visualization Toolkit (VTK). For more information about VTK, visit http://www.kitware.com .
Vista	vista	Hierarchical file format based on Silo that stores data in HDF5 file format.
ViSUS	idx, vis	Research file format that allows dynamic decomposition of the file based on available processors and supports efficient reading of data (and subsets of data) from the file.
Wavefront OBJ	obj	Alias Wavefront Object file format. Contains 3D models.
Xmdv	okc	ASCII file containing columns of data.

3.0 File Selection Window

The **File Selection Window**, shown in Figure 2-1, allows you to select files and simulations by browsing file systems either on your local computer or the remote computer of your choice. You can open the **File Selection Window** by choosing the **Select Files** option from the **Main Window's File** menu. When the window opens, its current directory is set to the current working directory or a directory from VisIt's preferences.

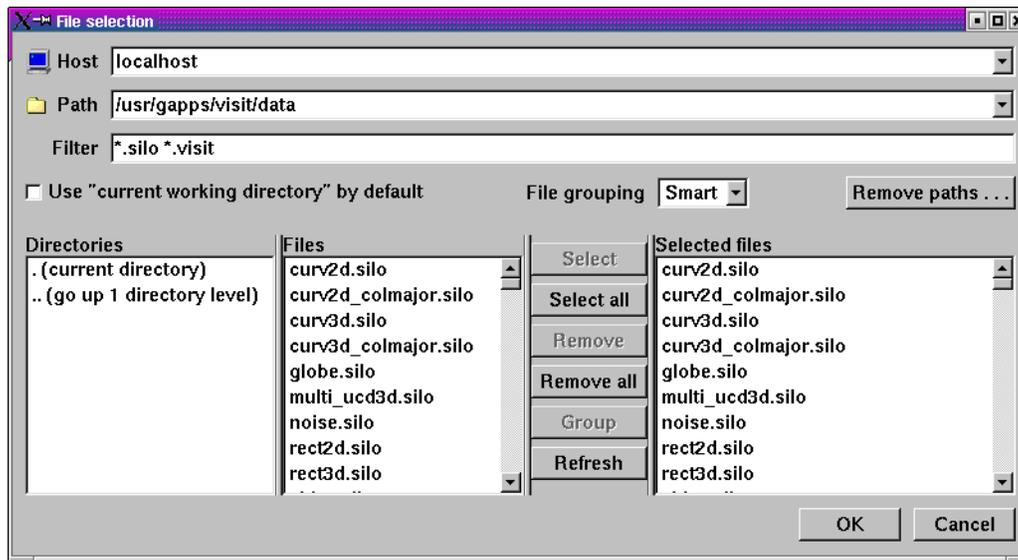


Figure 2-1: File Selection Window

3.1 Changing hosts

One of VisIt’s strengths is its ability to operate on files that exist on remote computers. The default host is: “localhost”, which is a name understood by the system to be the name of your local computer. To access the files on a remote computer, you must provide the name of the remote computer in the **Host** text field by either typing the name of a remote computer and pressing the Enter key or by selecting a remote computer from the list of recently visited hosts. To access the list of recently visited hosts, click on the down-arrow at the far right of the **Host** text field.

Changing the host will cause VisIt to launch a database server on the specified computer so you can access files there. Note that if you do not have an account on the remote computer, or if VisIt is not installed there, you will not be able to access files. Also note that VisIt may prompt you for a password to authenticate your access to the remote computer. To set up password-less access to remote computers, refer to “Setting Up Password-less ssh” on page 339.

Once a database server is running on the remote computer, its file system appears in the directory and file lists. The host name for each computer you access is added to the list of recently visited computers so that you may switch easily to computers you have recently accessed. If you installed VisIt with the provided network configurations then the list of recently visited computers also contains the hosts from the host profiles, which are covered later in this document.

3.2 Changing directories

To select data files, you must often change the active directory. This can be done in two ways. The first way is to enter the entire directory path into the **Path** text field and press Enter. You can use UNIX™ shell symbols, like the “~” for your home directory, or the “..” to go up one directory from your current directory. The directory conventions used depend on the type of computer being accessed. A Windows^(R) computer expects directories to be specified with a disk drive and a path with back slashes (*e.g.* `C:\temp\data`) while a UNIX™ computer expects directories with forward slashes (*e.g.* `/usr/local/data`). Keep the type of computer in mind when entering a path. After a path has been typed into the **Path** text field, VisIt will attempt to change directories using the specified path. If VisIt cannot change to the specified directory, the **Output Window** will appear with an error message and the **Path** text field will revert to the last accepted value. Another way to change directories is to double click the mouse on any of the entries in the directory list. Note that as you change directories, the contents of the **File list** change to reflect the files in the current directory. You can immediately return to any recently visited directory by selecting a directory from the **Path** text field’s pulldown menu.

3.3 Default directory

By default, VisIt looks for files in the current directory. This is often useful in a UNIX™ environment where VisIt is launched from a command line shell in a directory where database files are likely to be located. When VisIt is set to look for files in the current directory, the “**current working directory**” check box is set. If all of your databases are located in a central directory that rarely changes, it is worthwhile to uncheck the check box, change directories to your data directory, and save settings so the next time VisIt runs, it will look for files in your data directory.

3.4 Changing filters

A filter is a pattern that is applied to the files in the **File list** to determine whether or not they should show up in the list. This mechanism allows the user to exclude many files from the list based on a naming convention, which is useful since VisIt’s data files often share some part of their names.

The **Filter** text field controls the filter used to display files in the file list. Changing the filter will often change the **File list** as files are shown or hidden. The **Filter** text field accepts standard UNIX™ C-Shell pattern matching, where, for example, a “*” matches anything, “?” matches any single character, and “#” matches any single digit. The default filter (“*”) shows all files in the **File list**. Note that you can specify more than one filter provided you separate them with a space.

3.5 Managing the selected file list

The **File Selection Window** contains a three text fields along the top that allow you to change the location of where VisIt looks for files. The contents of directories and the selected files list are shown below those text fields. The left of the window lists the names of the directories that are subdirectories of the current directory. This list is commonly empty except for two entries that link to the current and parent directories. The list in the middle of the window contains the files that reside in the current directory and match the supplied file filter. The list on the far right contains the list of selected files. This window allows you to modify the contents of that list by adding and subtracting files from the list. When you are satisfied with the contents of the selected file list, clicking the **Ok** button will apply the **Selected file list**. Otherwise, clicking the **Cancel** button will undo any changes made to the **Selected file list**.

The buttons between the file list and the selected file list add and remove files from the selected files list as well as create grouped files. To add files to the **Selected files list**, you can select multiple files by clicking on a file and holding down the Shift or Control keys before clicking on another file. After selecting files from the **File list**, click the **Select** button to add the selected files to the **Selected files list**. If you want all of the files in the **Files list** moved to the **Selected files list**, you can click the **Select All** button. You can remove files from the **Selected files list** by first selecting the files that you want to remove and then clicking the **Remove** button. To remove all files from the **Selected files list**, click the **Remove All** button.

3.6 Grouping files

Time-varying scientific databases are often organized as a set of files where each file contains the state of a simulation at a particular instant in time. In order for VisIt to play animations, it must know which files are related. The **File Selection Window** allows you to create a “.visit” file, co-located with the database time step files, that groups all of the files together as a time-varying database. To group files, first select a set of files from the **File list** in the middle of the window and then click the **Group** button. This will create a file ending in “.visit” whose name is based on the names of the selected set of files. Once the “.visit” file is created, it appears in the list of available files. A “.visit” file is created each time the **Group** button is clicked but its name does not change unless the names of its constituent files also change. In other words, the “.visit” file can be overwritten if the **Group** button is clicked several times. Later when you want to create an animation, you open the “.visit” file to get the simulation data as it changes over time.

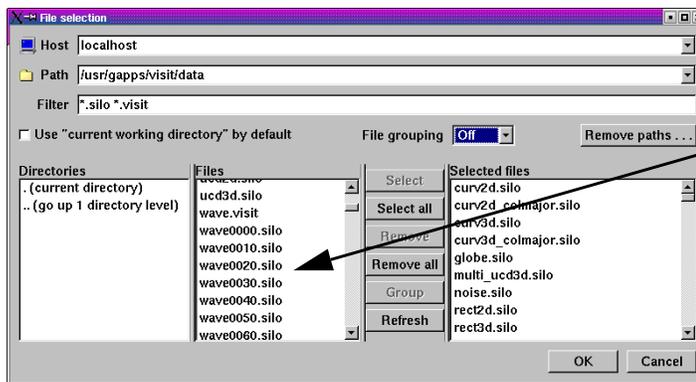
3.7 Virtual databases

A virtual database is a time-varying database that VisIt artificially creates out of smaller, single time step databases that have related filenames. Virtual databases allow you to access time-varying data without having to first create a “.visit” file. The files that are grouped into a virtual database are determined by the file filter. That is, only files that

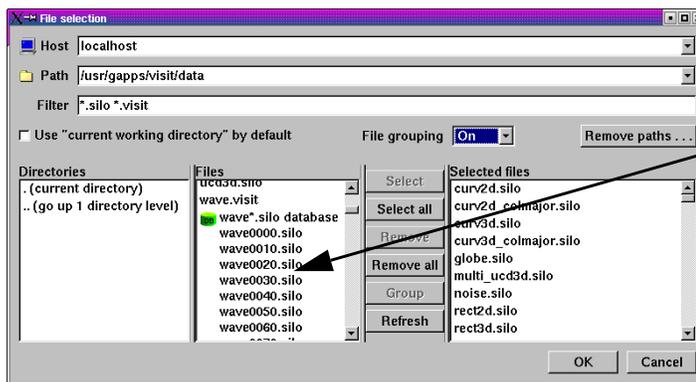
match the file filter are considered for grouping into virtual databases. You can change the definition of a virtual database by changing the file filter before you add the virtual database into your selected files list. A virtual database appears in the file list and the selected files list as a set of filenames that are grouped under a single filename that contains the “*” wildcard character. When you click on any of the filenames in the virtual database, the entire database is selected.

You can tell VisIt to not automatically create virtual databases by selecting the *Off* option in the **File grouping** menu in the **File Selection Window**. When automatic file grouping is turned off, no files are grouped into virtual databases and groups of files that make up a time-varying database will not be recognized as such without a “.visit” file. See Figure 2-2 for the effects of automatic file grouping on files in the **File Selection Window**.

VisIt has two levels of automatic file grouping. The default level is *Smart* file grouping, which enables automatic file grouping but has extra rules that prevent certain groups of files from being grouped into virtual databases. If you find that *Smart* file grouping does not provide the virtual databases that you expect, you can back the file grouping mode down to *On* or turn it off entirely.



Automatic file grouping is off so none of the wave silo files are grouped into a virtual database



Automatic file grouping is on so all of the wave Silo files are grouped into a single virtual database.

Figure 2-2: Automatic file grouping in the File Selection Window

3.8 Refreshing the file list

Scientific simulations often write out new data files as they run. The **Refresh** button makes VisIt re-read the current directory to pick up any new files added by a running simulation. If the active source is a virtual database whose definition was changed by refreshing the file list, then VisIt will close and reopen the active source so information about new time states is made available.

3.9 Clearing out recently visited paths

The **File Selection Window** maintains a list of all of the paths that you've ever visited and adds those paths to the recently visited paths list, which can be accessed by clicking on the down-arrow at the far right of the **Paths** text field. When you click on a path in the recently visited paths list, VisIt sets the database server's path to the selected path retrieves the list of files in that directory. If you visit many paths, the list of recently visited paths can become quite long. Click the **File Selection Window's Remove Paths** button to activate the **Remove Recent Paths Window**. The **Remove Recent Paths Window** allows you to select paths from the recently visited paths list and remove them from the list. The **Remove Recent Paths Window** is shown in Figure 2-3.

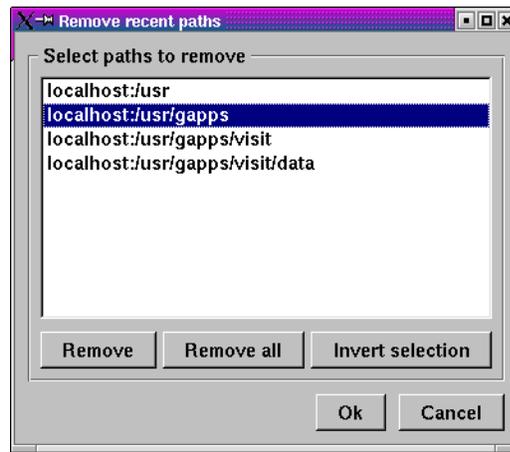


Figure 2-3: Remove Recent Paths Window

3.10 Connecting to a running simulation

Computer simulations often take weeks or months to complete and it is often necessary to visualize data from the simulation before it has completed in order to diagnose potential problems. Newer versions of VisIt come with a simulation interface library that can be linked into your serial or parallel simulation application in order to provide hooks so VisIt can plot data from your running simulation. When instrumented with the VisIt simulation interface library, your simulation can periodically check for incoming VisIt connections. When VisIt successfully connects to your simulation, all of your simulation variables are available for plotting without having to write plot files to disk. During the time that VisIt is connected, your simulation acts as a VisIt compute engine in addition to its regular responsibilities. You can pause the simulation while using VisIt to interact with the data or you can choose to have the simulation continue and push new data to VisIt for plotting. For more information about instrumenting your simulation code with the VisIt simulation library interface, send e-mail to visit-help@llnl.gov.

VisIt currently treats simulations as though they were ordinary files. When the VisIt simulation interface library is enabled in your application, it writes a special file with a `.sim` extension to the `.visit/simulations` directory in your home directory. Each `.sim` file encodes the time and date it was created into the file name so you can distinguish between multiple simulations that VisIt can potentially open. A `.sim` file contains information that VisIt needs in order to connect via sockets to your simulation. If you want to connect to a simulation, you must select the `.sim` files corresponding to the simulations to which you want to connect and add them to the **Selected files list** (Figure 2-4). Once you've done that, connecting to a simulation is the same as opening any other disk file.

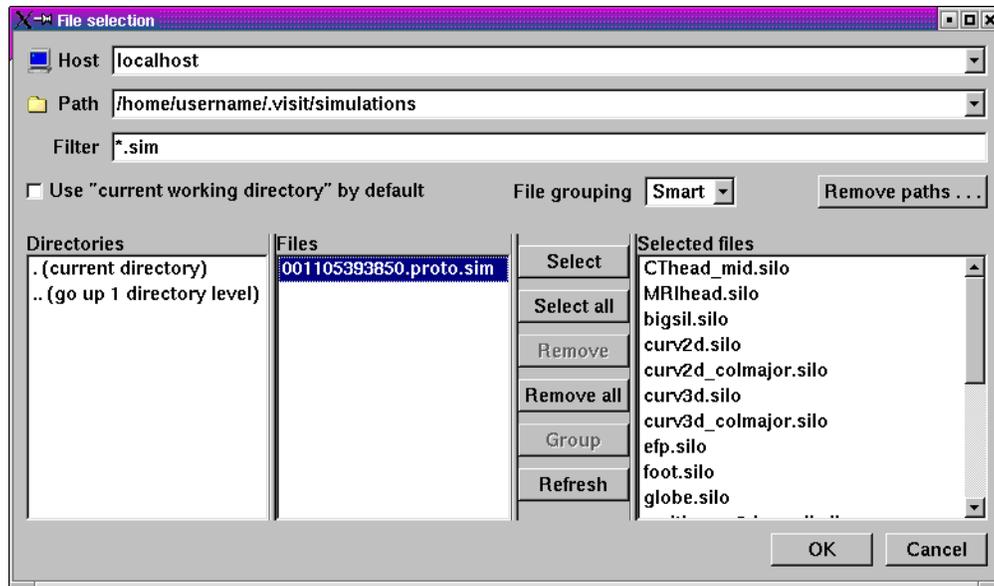


Figure 2-4: Accessing a simulation using the File Selection Window

4.0 File panel

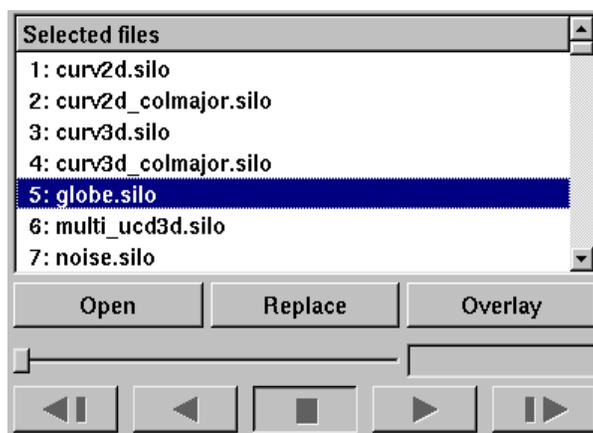


Figure 2-5: File panel with files from a single directory

The **File panel**, near the top of the **Main Window**, displays the files in the **Selected files list** and contains controls for opening files and playing animations. The **File panel** always tries to display the **Selected files list** in the most intuitive manner. If all of the selected files come from the same directory on the local machine, the **File panel** will display the selected files in a simple list like the one shown in Figure 2-5.

When the **Selected files list** contains files from multiple directories, the **File panel** displays the list as a file tree that displays the name of the computer where the files are located as well as the selected files and a minimal amount of directory information needed to uniquely identify files. Directories are identified with a folder icon and they cannot be opened for visualization. Files have no icons; instead they have a number next to them that indicates their index in the **Selected files list**. The file number is used in the **Plot list** to identify the file used by the plot. The file tree is always represented using the fewest possible number of nodes to avoid having to click through levels of empty directories. Displaying the **Selected files list** in tree form makes it easier to distinguish the directory from which files come. This view of the **Selected files list** is shown in Figure 2-6.

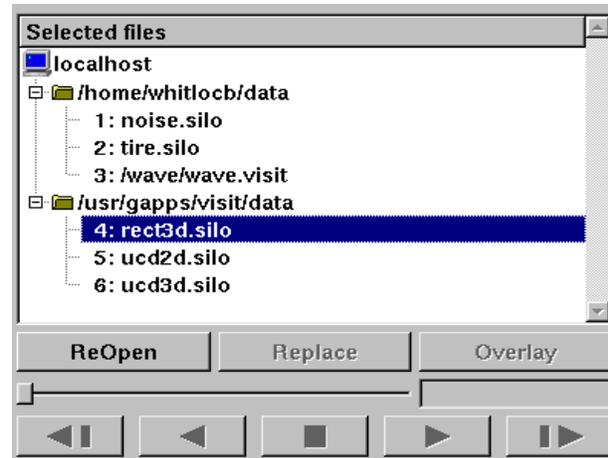


Figure 2-6: File panel with files from multiple directories

4.1 Opening a file

To open a file, you want to visualize, click on its name in the **File panel** and then click on the **Open** button. You can also open files by double-clicking on the file in the **File panel**. Once a file is open, the **Open** button turns into the **ReOpen** button and the file can be visualized. If you click on a file that has been opened before, the **Open** button becomes the **Activate** button. Clicking on the **Activate** button causes the selected file to become the new open database, from which plots can be created.

When the **ReOpen** button is clicked, all cached information about the open database is deleted, the database is queried again for its information, and any plots that use that database are regenerated using the new information. This allows VisIt to access data that was added to the database after VisIt first opened it.

Virtual databases, unlike “.visit” files, are expanded in the **File panel** by default because the time states are usually known before the database is opened. Since the time states are known before the database is opened, and you can click on any of them, it is possible to highlight a later time state of a virtual database and open it at that later time state. Opening a virtual database at a later time state can be useful if the database has variables that are introduced later in the time series.

4.2 Opening a file on a remote computer

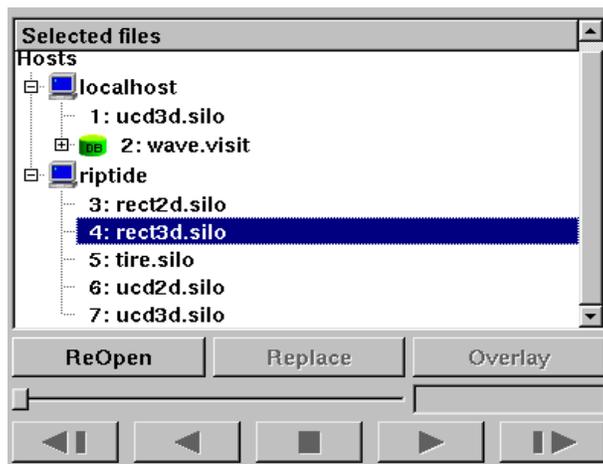


Figure 2-7: File panel with files from multiple computers

Opening a file on a remote computer works like opening a file on your local computer because the files from the remote computer have been placed into the **File panel**. The only difference is that when the **File panel** contains files from multiple computers, it displays the list of files from each computer under a small computer icon that represents the computer where the files are located. An example of this is shown in Figure 2-7.

4.3 Opening a time-varying database

A time-varying database (a “.visit” file or virtual database) is opened the same way as a database with a single time step. A time varying database is created by either grouping files into a “.visit” file in the **File Selection Window** or by turning on automatic file grouping so VisIt creates virtual databases, which are databases made up of multiple single time step databases that have similar filenames. To open a time-varying database, click on it to highlight it and then click the **Open** button in the **File panel**. Double-clicking the time-varying database also will open it. The one difference between opening a single time step database and a time-varying database is that if the time-varying database is also a virtual database then it can be opened at a later time step without having to first open it at the first time step.

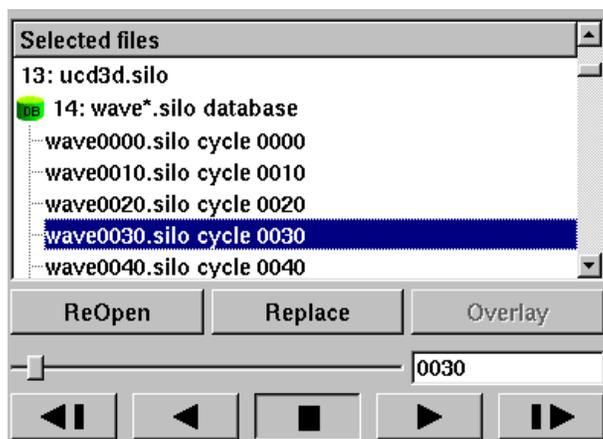


Figure 2-8: File panel with open time-varying database

The **File panel** displays time varying databases with a green database icon next to the name of the database. There often is also a small check box next to the database icon that expands to show the database’s time steps. This is shown in Figure 2-8. When a time-varying database is open, the animation controls are activated so any time step in the database can be used. Note that the animation controls are only active when visualizing a time-varying database or when VisIt is in keyframe animation mode.

4.4 Setting the active time step

Time-varying databases are composed of one or more time steps which contain data to be visualized. The active time step is the time step within a time-varying database that VisIt uses to generate plots. The **File panel** contains a group of animation controls below the **Selected files list** that allow you to set the active time step used for visualization. The **Animation slider** and the **Animation text field** show the active time step. To set the active time step, you can drag the **Animation slider** and release it when you get to the desired time step, or you can type in a cycle number into the **Animation text field**. If you type in a cycle number that is not in the database, the active time step will be set to the time step with the closest cycle number to the cycle that was specified. Another method of setting the active time step is to double click a cycle number under the database in the **Selected file list**.

4.5 Playing animations

The **File panel** contains a set of **VCR buttons** that allow you to put VisIt into an animation mode that plays your visualization using all of the time steps in the database. The **VCR buttons** are only active when you have a time varying database. The leftmost VCR button moves the animation back one frame. The VCR button second from the left plays the animation in reverse. The middle VCR button stops the animation. The VCR button second from the right plays the animation. The VCR button farthest to the right advances the animation by one frame. As the animation progresses, the **Animation Slider** and the **Animation Text Field** are updated to reflect the active time step.

4.6 Reopening a database

Sometimes it is useful to begin visualizing simulation data before the simulation has finished writing out data files for all time steps. When you open a database in VisIt and create plots and later want to visualize new time steps that have been generated since you first opened the database, you can reopen the database to force VisIt to get the data for the new time steps. To reopen a database, click the **ReOpen** button in the **File panel**. When VisIt reopens a database, it clears the geometry for all plots that used that database and cached information about the database is erased so that when VisIt reopens the database, plots are regenerated using the new data files.

4.7 Replacing a database

If you have created a plot with one database and want to see what it looks like using data from another database, you can replace the database using the **File panel's Replace** button. To replace a database, first select a new database by clicking on a file in the **File panel's Selected files list** and then click the **Replace** button. This will make VisIt try to replace the databases used in the plots with the new database. If the replace operation is

a success, the plots are regenerated using the new database and they are displayed in the visualization window.

4.8 Overlaying a database

Overlaying a database is a way to duplicate every plot in the plot list using a new database. To overlay plots, select a new database from the **Selected files list** in the **File Panel** and then click the **Overlay** button. This copies each plot in the **Active plot list** and replaces the database with the specified database. If the operation succeeds, the plots are generated and displayed in the visualization window. It is important to remember that each time the **Overlay** button is clicked, the number of plots in the plot list doubles.

5.0 File Information Window

This **File Information Window**, shown in Figure 2-9, displays information about the currently open file. The **File Information Window** is opened by choosing the **Files information** option from the **Main Window's File** menu. The window displays the names and properties of the open file's meshes, scalar variables, vector variables, and materials. The window updates each time the active file changes such as when switching between plots in the **Active plot list** or opening a new file using the controls in the **File panel**.

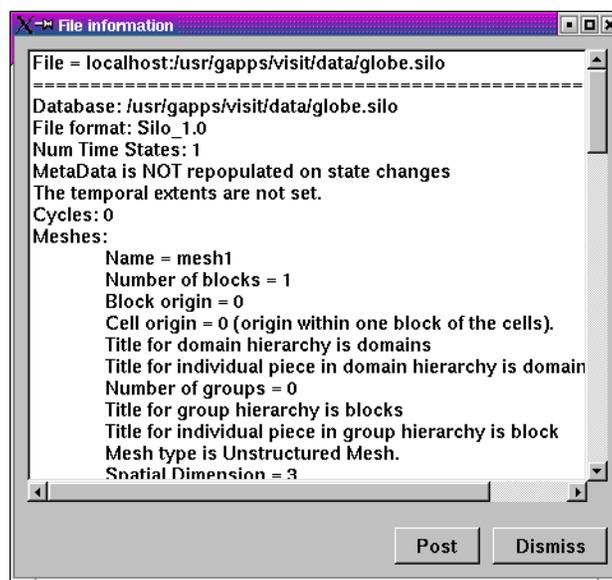


Figure 2-9: File Information Window

6.0 Advanced file options

The **Main Window's File** menu contains an **Advanced file options** menu that allows you to reopen and close databases that have been previously opened. The **Advanced file options** menu is shown in Figure 2-10.

6.1 Closing a database

VisIt allows you to close a database that has been previously opened by selecting a database from the list of databases in the **Close file** menu, which is an option in the **Advanced file options** menu. Only databases that are not used by any plots can be closed. If you attempt to close a database that is being used by at least one plot, VisIt will issue an error message. The **Close file** menu is disabled if there are no open databases.

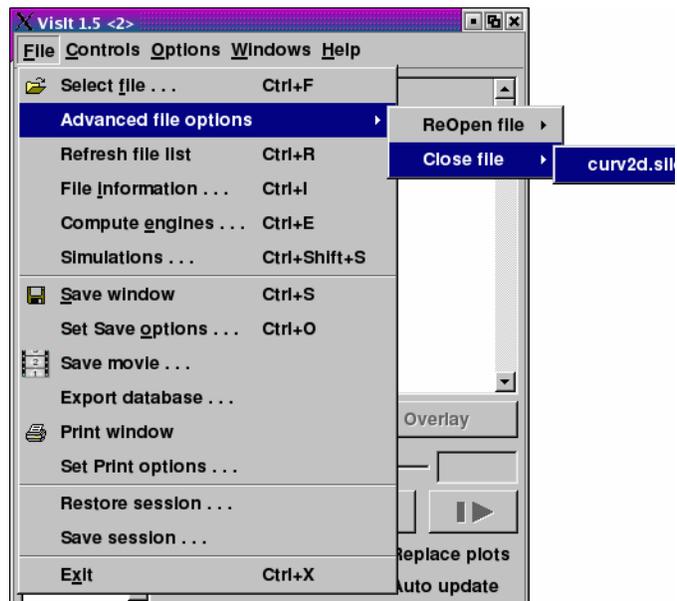


Figure 2-10: Advanced file options menu

1.0 Overview

This chapter explains the concept of a plot and goes into detail about each of VisIt's different plot types.

2.0 Plots

A plot is a viewable object, created from a database, that can be displayed in a visualization window. VisIt provides several standard plot types that allow you to visualize data in different ways. The standard plots perform basic visualization operations like contouring, pseudocoloring as well as more sophisticated operations like volume rendering. All of VisIt's plots are plugins so you can add new plot types by writing your own plot plugins. See the *VisIt Plugin Developer's Guide* for more details on creating new plot plugins or send an e-mail inquiry to visit-help@llnl.gov.

2.1 Managing plots

To visualize your data, you will iteratively create and modify many plots until you achieve the end result. Since plots may be created and deleted many times, VisIt provides controls in its **Main Window** to handle these functions. The **Active plots area**, shown in Figure 3-1, contains the controls for managing plots.

The most prominent feature of the **Active plots area**, the **Plot list**, contains a list of the plots that are in the active visualization window. The entries in the plot list contain the index of the file used for the plot followed by the plot name and variable. Plot list entries change colors depending on the state of the plot. When plots are initially created, their plot list entries are green indicating that they are new and have not been submitted to the

compute engine for processing. When a plot is being created on the compute engine, its plot list entry is yellow. When a plot has finished generating on the compute engine, its plot list entry turns black to indicate that the plot is done. If the compute engine cannot generate a plot, the plot's plot list entry turns red to indicate an error with the plot.

The **Plot list** displays more than just the names of the visualization window's plots. The **Plot list** also allows you to set the active plots, that is, those plots that can be modified. Highlighted plot entries are active.

The **Plot and Operator** menu, an important part of the **Active plots area**, contains the options that create new plots and open plot attribute windows.

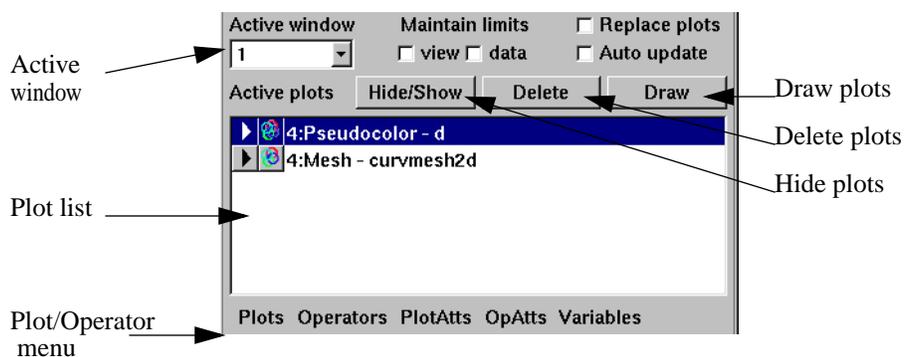


Figure 3-1: Active plots area

2.1.1 Creating a plot

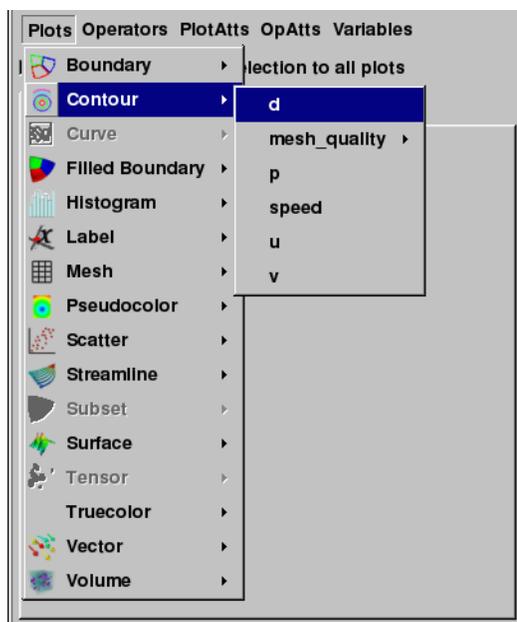


Figure 3-2: Plot menu

To use any of VisIt's capabilities, you must know how to create a plot. First, make sure you have opened a database. To open a database, double-click on the database name in the **Selected files list**. After opening a database, use the **Plots** menu to create a plot. To activate the **Plots** menu, click on the **Plots** option in the **Plots and Operators** menu shown in Figure 3-2.

Selecting the **Plots** menu pops up a list of VisIt plot types. Plots for which the open database has no data are disabled. If a plot type is enabled, pulling the mouse toward the right while holding down the left button shows which variables can be plotted. Release the mouse button when the mouse cursor is over the variable that you want to plot, and a new plot list entry will appear in the **Plot list**. The

new plot list entry will be colored green in the plot list until VisIt is told to draw when you click the **Draw** button. The **Plots** menu is disabled until a database is open.

2.1.2 Deleting a plot

VisIt deletes all the selected plots when you click the **Delete** button. If the **Plot list** has keyboard focus, you can also delete a plot using the *Delete* key.

2.1.3 Selecting a plot

Since VisIt will only let you modify active plots, you must be able to select plots. To select a plot, click on its entry in the **Plot list**. Multiple plots can be selected by holding down the *Ctrl* key and clicking plot entries one at a time. Alternatively, groups of plot entries can be selected by clicking on a plot entry and then clicking another plot entry while holding down the *Shift* key.

2.1.4 Drawing a plot

When you add a plot to the plot list, it won't be drawn until you click the **Draw** button. Once you do, the new plot's plot list entry switches from green to yellow in the **Plot list** to indicate that its results are pending and the compute engine starts generating the plot. Clicking the **Draw** button causes all new plots to be drawn.

2.1.5 Hiding a plot

When you are visualizing your data, you will often have many different plots in the same visualization window. Sometimes you might want to temporarily hide plots from view to more easily view the other plots in the window. To hide the selected plots, click the **Hide/Show** button in the **Active plots area**. When a plot is hidden, its plot list entry is gray and contains the word "*hidden*" to indicate that the plot is hidden. To show a hidden plot, select the hidden plot and click the **Hide/Show** button again. Note that plots must exist for the **Hide/Show** button to be enabled.

2.1.6 Setting plot attributes

Each plot type has its own plot attributes window used to set attributes for that plot type. Plot attributes windows are activated by double-clicking a plot entry in the **Plot list**. You

can also open a plot attribute window by selecting a plot type from the **PlotAtts** (Plot Attributes) menu shown in Figure 3-3.

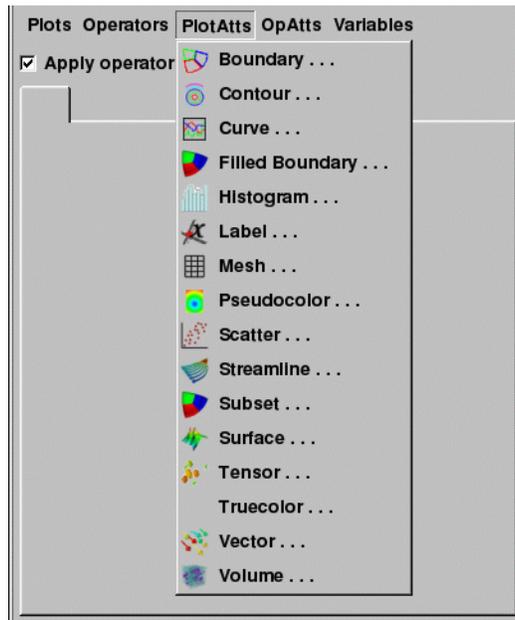


Figure 3-3: Plot attributes menu

2.1.7 Changing plot variables

When examining a plot, you might want to look at another variable. For example, you might want to switch from looking at density to pressure. VisIt allows the plot variable to be changed without having to delete and recreate the plot. To change the plot variable, first make sure the plot is active, then select a new variable from the available variable names in the **Variable** menu (Figure 3-4). The **Variable** menu contains only the variables from the database that are compatible with the plot.

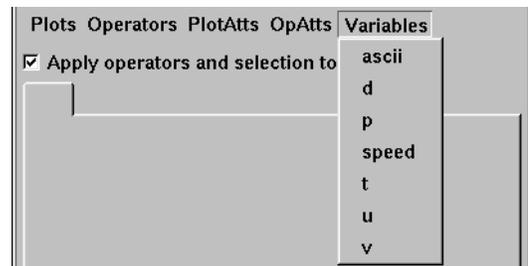


Figure 3-4: Variable menu

3.0 Standard Plot Types

VisIt comes with sixteen standard plots: Boundary, Contour, Curve, FilledBoundary, Histogram, Label, Mesh, Pseudocolor, Scatter, Streamline, Subset, Surface, Tensor, Truecolor, Vector, and Volume. This section explains each plot in detail.

3.1 Boundary and FilledBoundary Plots

The Boundary plot and FilledBoundary plot are discussed together because of their similarity. Both plots concentrate on the boundaries between materials but each plot shows the boundary in a different way. The Boundary plot, shown in Figure 3-5, displays the surface or lines that separate materials, while the FilledBoundary plot (see Figure 3-6) shows the entire set of materials, each using a different color. Both plots perform material interface reconstruction on materials that have mixed cells, resulting in the material boundaries used in the plots.

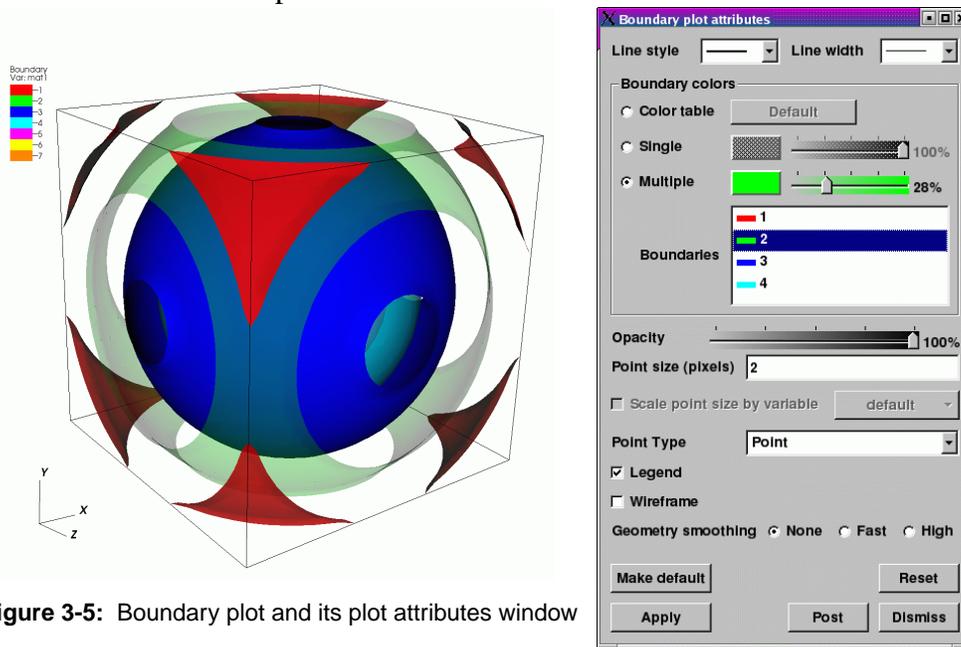


Figure 3-5: Boundary plot and its plot attributes window

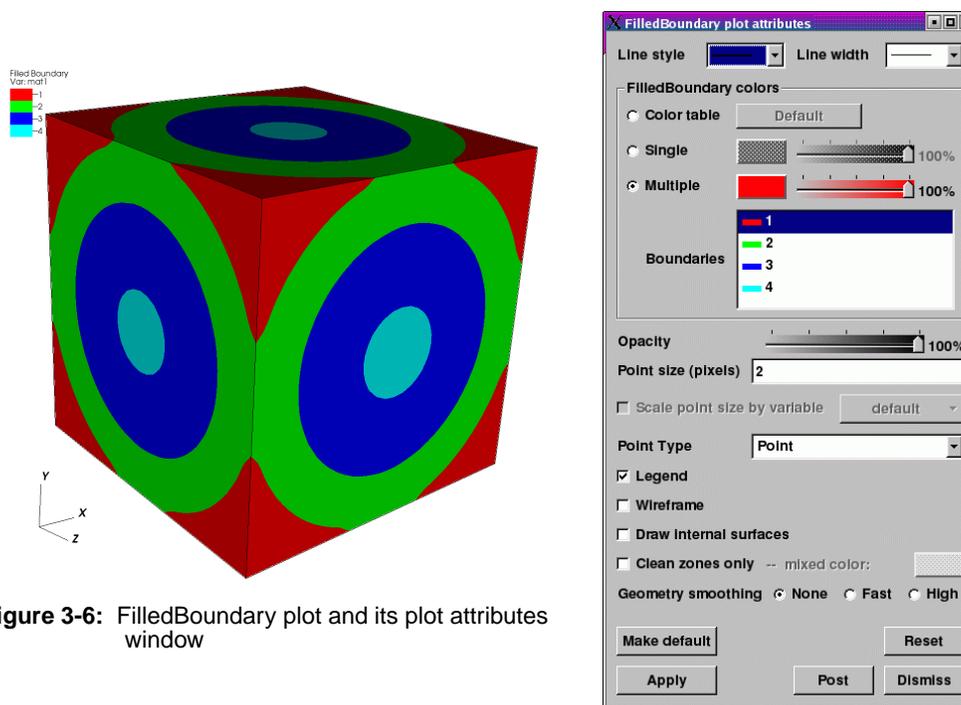


Figure 3-6: FilledBoundary plot and its plot attributes window

3.1.1 Changing colors

The main portion of the **Boundary plot attributes window** and **FilledBoundary plot attributes window**, also known as the **Boundary colors area**, is devoted to setting material boundary colors. The **Boundary colors area** contains a list of material names with an associated material color. Boundary plot and FilledBoundary plot colors can be assigned three different ways, the first of which uses a color table. A color table is a named palette of colors that you can customize to suite your needs. When the Boundary plot or FilledBoundary plot use a color table to color subsets, they selects colors that are evenly spaced through the color table based on the number of subsets. For example, if you have three materials and you are coloring them using the “xray” color table, three colors are picked out of the color table so your material boundaries are colored black, gray, and white. To color a Boundary plot or FilledBoundary plot with a color table, click on the **Color table radio button** and choose a color table from the **Color table menu** to right of the **Color table radio button**.

If you want all subsets to be the same color, click the **Single** radio button at the top of the **Boundary plot attributes window** and select a new color from the **Popup color menu** that is activated by clicking on the **Single color button** . The opacity slider next to the **Single color button** sets the opacity for the single color.

Clicking the **Multiple** radio button causes each material boundary to be a different, user-specified color. By default, multiple colors are set using the colors of the discrete color table that is active when the Boundary or FilledBoundary plot is created. To change the color for any of the materials, select one or more materials from the list of materials and click on the **Color button** to the right of the **Multiple** radio button and select a new color from the **Popup color menu**. To change the opacity for a material, move **Multiple** opacity slider to the left to make the material more transparent or move the slider to the right to make the material more opaque.

The **Boundary plot attributes window** contains a list of material names with an associated color. To change a material’s color, select one or more materials from the list, click the color button and select a new color from the popup color menu.

3.1.2 Opacity

The Boundary plot’s opacity can be changed globally as well as on a per material basis. To change material opacity, first select one or more materials in the list and move the opacity slider next to the color button. Moving the opacity slider to the left makes the selected materials more transparent and moving the slider to the right makes the selected materials more opaque. To change the entire plot’s opacity globally, use the **Opacity** slider near the bottom of the window.

3.1.3 Wireframe mode

The Boundary plot and the FilledBoundary plot can be modified so that they only display outer edges of material boundaries. This option usually leaves lines that give only the rough shape of materials and where they join other materials as seen in Figure 3-7. To make the Boundary or FilledBoundary plots display in wireframe mode, check the **Wireframe** check box near the bottom of the window.

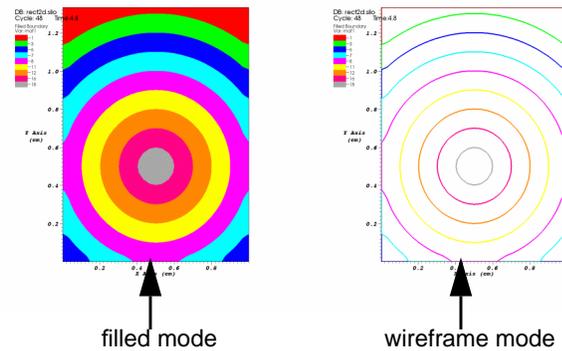


Figure 3-7: Filled mode and wireframe mode

3.1.4 Drawing internal surfaces

When you make one or more material boundaries transparent, you might want to also draw internal surfaces so you can see interior plot details that are normally removed to make the plot draw faster. To draw internal surfaces, check the **Draw internal surfaces** check box near the bottom of the **Boundary plot attributes window** or **FilledBoundary plot attributes window**.

3.1.5 Geometry smoothing

Sometimes visualization operations such as material interface reconstruction can alter mesh surfaces so they are pointy or distorted. The Boundary plot and the FilledBoundary plot provide an optional Geometry smoothing option to smooth out the mesh surfaces so they look better when the plots are visualized. Geometry smoothing is not done by default, you must click the **Fast** or **High** radio buttons to enable it. The **Fast** geometry smoothing setting smooths out the geometry a little while the **High** setting works produces smoother surfaces.

3.1.6 Drawing only clean zones

The FilledBoundary plot, since it deals almost exclusively with plotting materials, has an option to only draw clean zones, which are zones that contain a single material. When only clean zones are drawn, all clean cells are drawn normally but all zones that contained more than one material are drawn with a color that can be set to match the vis window's background color (see Figure 3-8). Drawing clean zones is primarily used to examine how materials mix in 2D databases. To make VisIt draw only the clean zones, click the **Clean zones only** check box. After that, you can set the mixed color by clicking on the **Mixed color** color button and selecting a new color from the popup color palette.

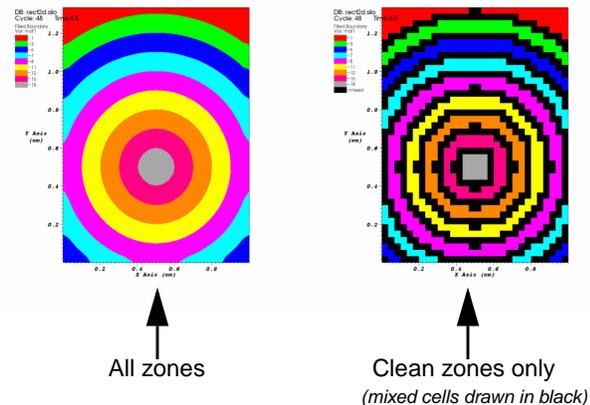


Figure 3-8: All zones and clean zones

3.1.7 Setting point properties

Albeit rare, the Boundary and FilledBoundary plots can be used to plot points that belong to different materials. Both plots provide controls that allow you to set the representation and size of the points. You can change the points' representation using the different **Point Type** radio buttons. The available options are: **Box**, **Axis**, **Icosahedron**, **Point**, and **Sphere** (see Figure 3-9). The default point type is **Point** because that is the fastest to draw, followed by **Sphere**. The other point types create additional geometry and can take longer to appear on the screen and subsequently draw. To change the size of the points when the point type is set to **Box**, **Axis**, or **Icosahedron**, you can enter a new floating point value into the **Point size** text field. When the point type is set to **Point** or **Sphere**, the **Point size** text field becomes the **Point size (pixels)** text field and you should enter your point size in terms of pixels. Finally, you can opt to scale the points' glyphs using a scalar expression by turning on the **Scale point size by variable** check box and by selecting a scalar variable from the **Variable** button to the right of that check box. Note that point scaling does not occur when the point type is set to **Point** or **Sphere**.

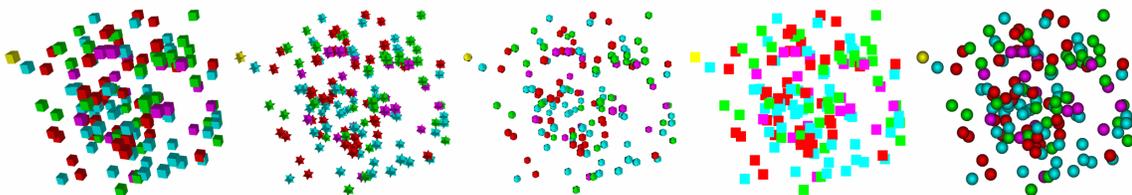


Figure 3-9: Point types: Box, Axis, Icosahedron, Point, Sphere

3.2 Contour Plot

This plot, shown in Figure 3-10, displays the location of values for scalar variables like density or pressure using lines for 2D plots and surfaces for 3D plots. In visualization terms, these plots are isosurfaces. VisIt's Contour plot allows you to specify the number of contours to display as well as the colors and opacities of the contours.

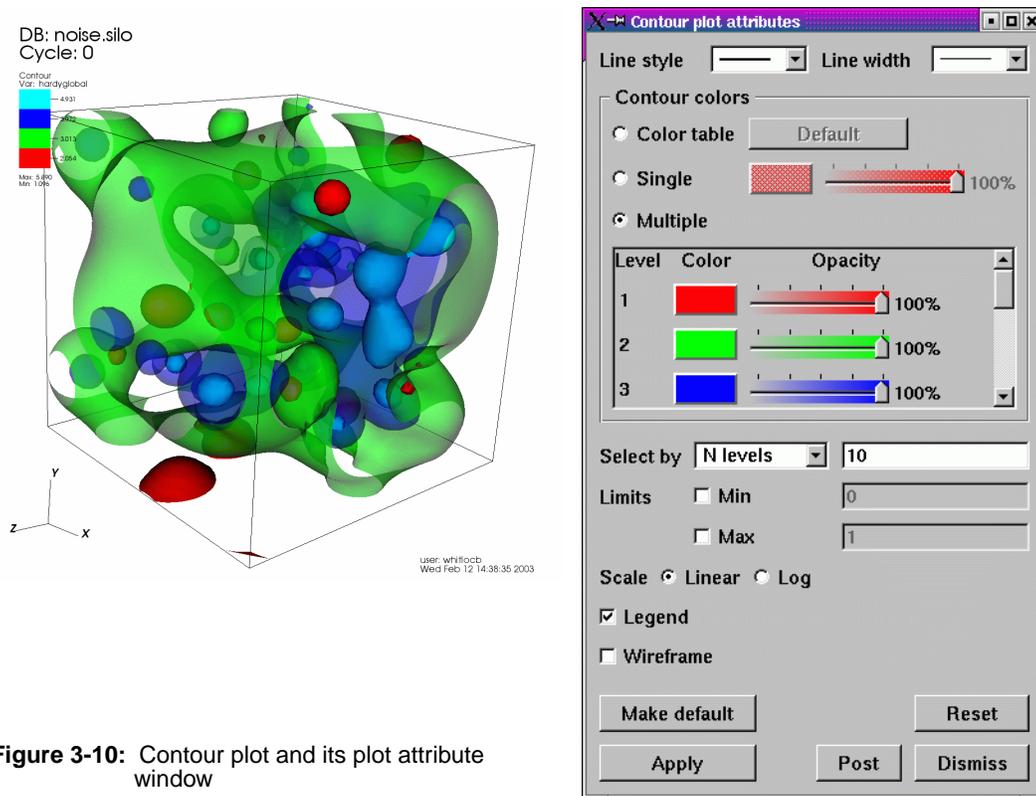


Figure 3-10: Contour plot and its plot attribute window

3.2.1 Setting the number of contours

By default, VisIt constructs 10 levels into which the data fall. These levels are linearly interpolated values between the data minimum and data maximum. However, you can set your own number of levels, specify the levels you want to see or indicate the percentages for the levels.

To choose how levels are specified, make a selection from the **Select by** menu. The available options are: **N levels**, **Levels**, and **Percent**. **N levels**, the default method, allows you to specify the number of levels which will be generated, with 10 being the default. **Levels** requires you to specify real numbers for the levels you want to see. **Percent** takes a list of percentages like 50.5 60 40. Using the numbers just mentioned, the first contour would be placed at the value which is 50.5% of the way between the minimum and maximum data values. The next contour would be placed at the value which is 60% of the way between the minimum and maximum data values, and so forth. You

specify all values for setting the number of contours by typing into the text field to the right of the **Select by** menu.

3.2.2 Setting Limits

The **Contour plot attributes window** provides controls that allow you to specify artificial minima and maxima for the data in the plot. This is useful when you have a small range of values that are of interest and you only want the contours to be generated through that range. To set the minimum value, click the **Min** check box to enable the **Min** text field and then type a new minimum value into the text field. To set the maximum value, click the **Max** check box to enable the **Max** text field and then type a new maximum value into the text field. Note that either the min, max or both can be specified. If neither minimum nor maximum values are specified, VisIt uses the minimum and maximum values in the database.

3.2.3 Scaling

The Contour plot typically creates contours through a range of values by linearly interpolating to the next value. You can also change the scale to a logarithmic function to get the list of contour values through the specified range. To change the scale, click either the **Linear** or **Log** radio buttons in the **Contour plot attributes window**.

3.2.4 Setting contour colors

The main portion of the **Contour plot attributes window**, also known as the **Contour colors area**, is devoted to setting contour colors. Contour plot colors can be assigned three different ways, the first of which uses a color table. A color table is a named palette of colors that you can customize to suite your needs. When the Contour plot uses a color table to color the levels, it selects colors that are evenly spaced through the color table based on the number of levels. For example, if you have five levels and you are coloring them using the “rainbow” color table, the Contour plot picks five colors out of the color table so your levels are colored magenta, blue, cyan, green, yellow, and red. The colors change when increasing or decreasing the number of levels when you use a color table because VisIt uses the new number of levels to sample different locations in the color table. As a rule, increasing the number of levels results in coloration that is closer to the color table because more colors from the color table are represented. To color a Contour plot with a color table, click on the **Color table radio button** and choose a color table from the **Color table menu** to right of the **Color table radio button**.

If you want all levels to be the same color, click the **Single** radio button at the top of the **Contour plot attributes window** and select a new color from the **Popup color menu** that is activated by clicking on the **Single color button** . The opacity slider next to the **Single color button** sets the opacity for the single color.

Clicking the **Multiple** radio button causes each level to be a different, user-specified color. By default, multiple colors are set using the colors of the discrete color table that is active

when the Contour plot is created. To change the color for any of the levels, click on the level's **Color button** and select a new color from the **Popup color menu**. To change the opacity for a level, move its opacity slider to the left to make the level more transparent or move the slider to the right to make the level more opaque.

3.2.5 Wireframe view

The **Contour plot attributes window** provides a **Wireframe** toggle button used to draw only the lines along the edges of the contour. This option only has an effect on 3D Contour plots.

3.3 Curve Plot

The Curve plot, shown in Figure 3-11, displays a simple group of X-Y pair data such as that output by 1D simulations or data produced by Lineouts of 2D or 3D datasets. Curve plots are useful for visualizations where it is useful to plot 1D quantities that evolve over time.

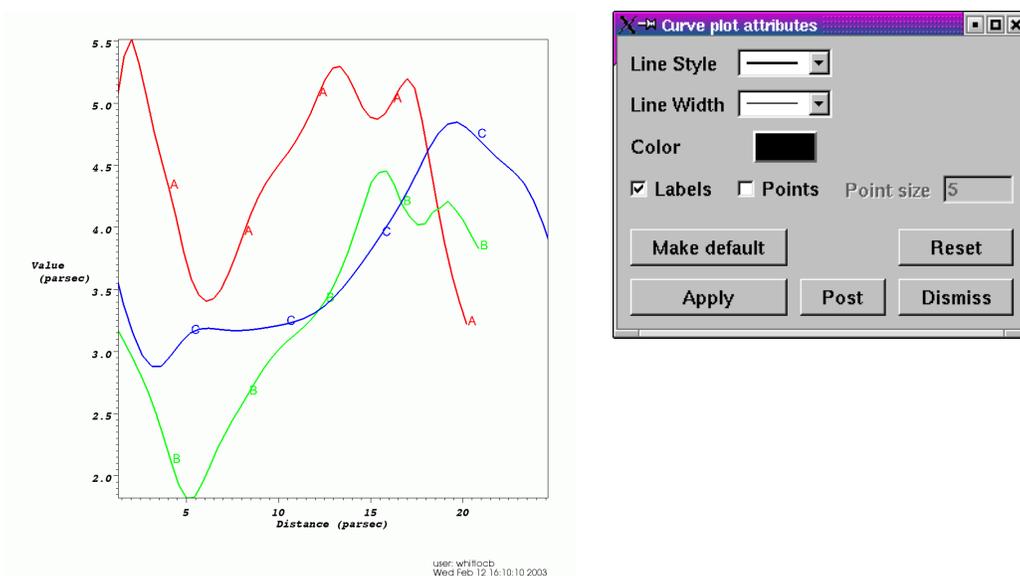


Figure 3-11: Curve plot and its plot attributes window

3.3.1 Setting line style and line width

Several Curve plots are often drawn in the same visualization window so it is necessary that Curve plots can be distinguished from each other. Fortunately, VisIt provides controls to change the line style and line width so that Curve plots can be told apart. Line style is a pattern used to draw the line and it is solid by default but it can also be dashed, dotted, or dash-dotted. You choose a new line style by making a selection from the **Line Style**

menu. The line width, which determines the boldness of the curve, is set by making a selection from the **Line Width menu**.

3.3.2 Setting curve color

The Curve plot's color can be changed by clicking on the **Color button** and making a selection from the **Popup color menu**.

3.3.3 Showing curve labels

In addition to line style and line width, Curve plots have a label that can be displayed to help distinguish a Curve plot from other Curve plots. Curve plot labels are on by default, but if you want to turn the label off, you can uncheck the **Labels** check box.

3.3.4 Drawing points on the Curve plot

The Curve plot is composed of a set of (X,Y) pairs through which line segments are drawn to form a curve. To make VisIt draw a point glyph at the location of each (X,Y) point, click the **Points** check box. You can control the size of the points by typing a new point size into the **Point size** text field.

3.4 Histogram Plot

The Histogram plot divides the data range of a scalar variable into a number of bins and groups the variable's values, weighted by cell area or revolved volume, into different bins. The values in each bin are then used to create a bar graph or curve that represents the distribution of values throughout the variable's data range. The Histogram plot can be used to determine where data values cluster in the range of a scalar variable. The Histogram plot is shown in Figure 3-12.

3.4.1 Setting the histogram data range

By default, the Histogram plot profiles a variable's entire data range. If you want to restrict the Histogram plot so it only takes a subset of a variable's data range into consideration when assigning values to bins, you can set the minimum and maximum values that will be considered by the Histogram plot. To specify a data range, click the **Specify Range** check box and then type in floating point numeric values into the

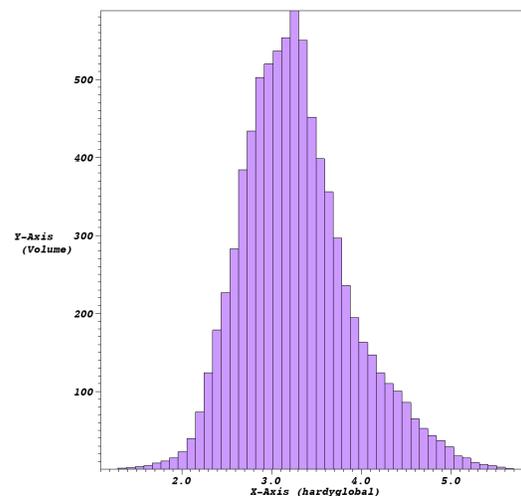


Figure 3-12: Histogram plot

Minimum and **Maximum** text fields in the **Histogram plot attributes window** (see Figure 3-13) before clicking its **Apply** button. Once the data range is set, the Histogram plot will restrict the values that it considers to the specified data range.

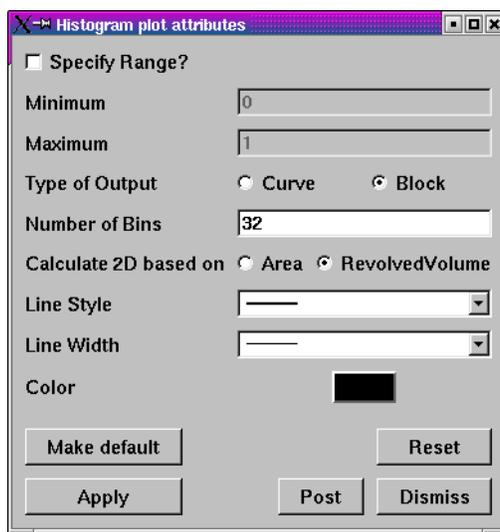


Figure 3-13: Histogram plot attributes window

3.4.2 Setting the type of graph

The Histogram plot has two mode in which it can appear: curve and block. When the Histogram plot is drawn as a curve, it looks like the Curve plot. When the Histogram plot is drawn in block mode, it is drawn as a bar graph where each bin is plotted along the X-axis and the height of each bar corresponds to the number of values that were assigned to that bin. You can set change the Histogram plot's appearance by clicking the **Curve** or **Block** radio buttons.

3.4.3 Setting the number of bins

The Histogram plot divides a variable's data range into a number of bins and then counts the weighted values that fall within each bin. The bins and the counted data are then used to create a graph that represents the distribution of data within the variable's data range. As the Histogram plot uses more bins, the graph of data distribution becomes more accurate. However, the graph can also become rougher because as the number of bins increases, the likelihood that no data values fall within a particular bin also increases. To set the number of bins for the Histogram plot, type a new number of bins into the **Number of Bins** text field and click the **Apply** button in the **Histogram plot attributes window**.

3.4.4 Setting the histogram calculation method

When the Histogram plot groups data values into bins, it weights the data value by the surface area or revolved volume of the cell so contributions from different sizes of cells

are compared fairly. To change the calculation method used to weight the cells, click on the **Area** radio button to make VisIt use surface area or click on the **Revolved volume** radio button to make VisIt use the revolved volume of a 2D cell as the weighting multiplier used to group cells into the right bins.

3.5 Label Plot

The Label plot, shown in Figure 3-14, can display mesh information, scalar fields, vector fields, tensor fields, array variables, subset names, and material names. The Label plot is often used as a debugging device for simulation codes since it allows the user to see labels containing the exact values at the computational mesh's nodes or cell centers. Since the Label plot's job is to display labels representing the computational mesh or the fields defined on that mesh, it does not convey much information about the actual mesh geometry. Since having a Label plot by itself does not usually give enough information to understand the plotted dataset, the Label plot is almost always used with other plots.

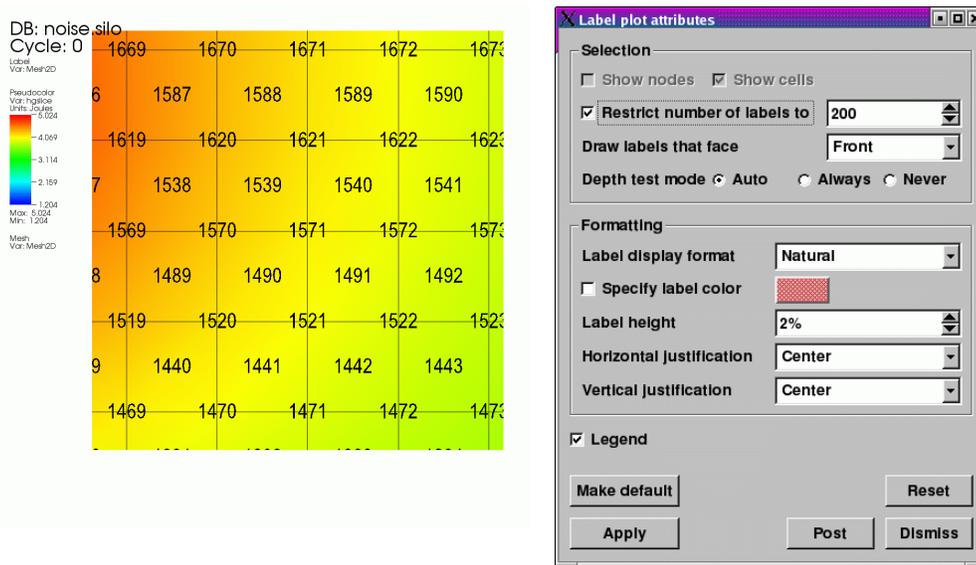


Figure 3-14: Label plot of the mesh overlaid on Pseudocolor and Mesh plots and the Label plot attributes window

3.5.1 Choosing the Label plot's variable

You can choose the Label plot's variable using the **Variable** menu under the **Plot list** the same way as you would with any other type of plot. One special property that distinguishes the Label plot from some of VisIt's other plots is that it can plot multiple types of variables. The Label plot can display information for meshes, scalars, vectors, tensors, array variables, subsets, and materials so you will typically find more variables available for the Label plot than you would for other plots. When you choose a mesh variable for the Label plot, you can display both the mesh node numbers and cell numbers otherwise you are limited to displaying only the variable being plotted.

3.5.2 Showing node and zone numbers

The Label plot can display the node and cell numbers for the computational mesh if you have selected a mesh variable to plot. By default, the Label plot will display cell numbers only. The cell numbers will be displayed in the format most natural to the underlying mesh representation, which means that unstructured meshes will have cell numbers that are displayed as single integers while structured meshes will be displayed in *i,j,k format* when possible. If you want the Label plot to show a mesh's node numbers in addition to its cell numbers, you can click on the **Show nodes** check box. If you no longer want the Label plot to show the mesh's cell numbers, you can turn off the **Show cells** check box.

3.5.3 Restricting the number of labels

Most computational meshes contain many thousands, millions, or even billions of nodes and cells. Adding that many labels would quickly become burdensome on the computer and would result in a Label plot so dense that individual labels could no longer be read or even associated with their cell or node.

VisIt's Label plot restricts the number of labels by default to some user-settable number of labels that can comfortably fit on the screen. The method used to restrict the number of labels differs for 2D and 3D plots. For 2D plots, the viewable portion of world space is periodically subdivided, based on the zoom level, into some number of bins to which labels are then assigned. As you zoom in on the Label plot, labels that go beyond the viewport are no longer drawn and new labels that were previously hidden take their place. This allows the Label plot to efficiently draw many labels without crowding the labels on top of each other. For 3D plots, the Label plot divides up the screen into a user-settable number of bins. All label coordinates are transformed so that they can be assigned to a screen bin and the label wins the screen bin if it is closer than the label that was previously in the bin. This ensures that a small subset of all possible labels is drawn and that they do not usually overlap on the screen. If you find that the labels appear to be from the back of the mesh instead of from the front, it's quite possible that the normals generated for your mesh were inverted for some reason. To combat this problem, select **Back** or **Front or Back** from the **Draw labels that face** combo box.

If you want to set the number of labels that the Label plot will draw, you can type in a new value into the spin box next to the **Restrict number of labels to** check box or use the up and down arrows on the spin box. If you want to force the Label plot to draw all labels, you can turn off the **Restrict number of labels to** check box. Sometimes making the Label plot draw all of the labels can be faster than drawing a subset of labels.

3.5.4 Depth testing for 3D Label plots

When VisIt draw plots in the visualization window, the plots' geometries often correspond to only the outer surfaces of the originating datasets when those datasets are 3D. This means that the majority of plots consist of convex geometry and the normal test for only drawing labels that face front is often adequate to remove any labels that appear on faces

that point away from the current camera. Some plots have geometries that consist of many concave regions, which the afore-mentioned test does not handle well. Plots with concave geometries will often have various pieces be incorrectly visible because though the surfaces may face the camera, they may be obscured by other geometry. When VisIt's Label plot draws 3D geometry, it tries to enable additional depth testing to prevent front-facing labels in back of other surfaces from being drawn. Depth testing can degrade performance so, by default, it is allowed only when you are running VisIt on your local workstation. You can set the Label plot's depth test mode to tell VisIt when to enable depth testing. To change the values for the depth test mode, click on one of the **Auto**, **Always**, **Never** radio buttons to the right of the **Depth test mode** label. If VisIt wants to use depth testing but is not allowed to then a warning message will be issued and you can set the depth test mode to **Always**.

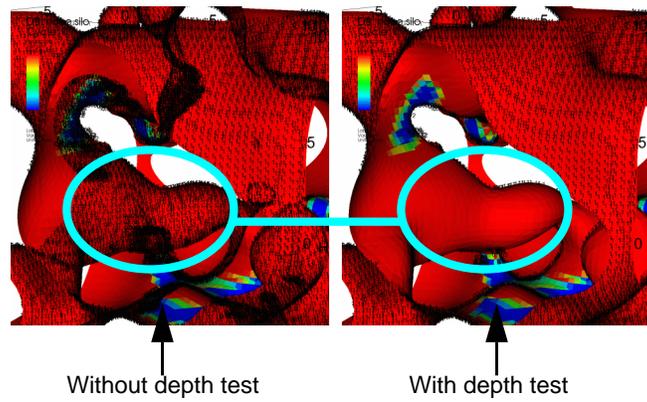


Figure 3-15: Removing extra labels with depth test

3.5.5 Formatting labels

The Label plot provides several options for setting label format. First and foremost, you can set the label display format, which is how mesh node and cell numbers are displayed. By default, the Label plot will display labels in their most appropriate format with cell and node numbers for structured meshes displayed as logical i,j,k indices. Setting the label format is only possible for Label plots of structured meshes. To change the label format, select a new option from the **Label display format** combo box.

The Label plot's default behavior is to use the vis window's foreground color but if you want labels to be a specific color, you can turn off the **Use foreground color** check box and select a new label color by clicking on the **Label color** color button.

The height of the Label plot's labels is determined as a percentage of the height of the vis window containing the Label plot. The default label height is set at 2% but you can change the label height by entering a new value into the **Label height** spin box or by using the up or down arrows on the **Label height** spin box. Note that when you are plotting a mesh variable, VisIt will make more controls in the **Label plot attributes window** so you can set the color and height for cells and nodes independently (see Figure 3-16).

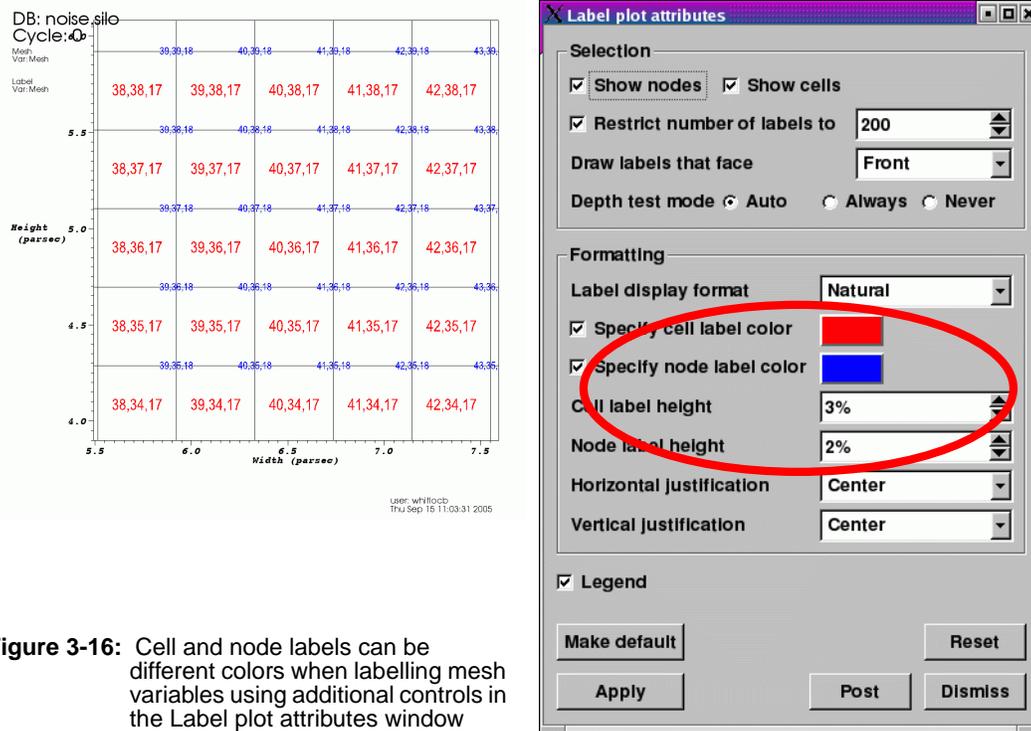


Figure 3-16: Cell and node labels can be different colors when labelling mesh variables using additional controls in the Label plot attributes window

Finally, the **Label plot attributes window** provides controls to determine the horizontal and vertical text justification used when drawing each label. To change the horizontal text justification, select a new value from the **Horizontal justification** combo box. To change the vertical text justification, select a new value from the **Vertical justification** combo box.

3.5.6 Labelling subset names and material names

The Label plot can label subset names and material names in addition to meshes and fields defined on those meshes. To add subset names or material names to your visualization, be sure to create a Label plot using a variable of either of those types. An example of a Label plot of material names is presented in Figure 3-17.

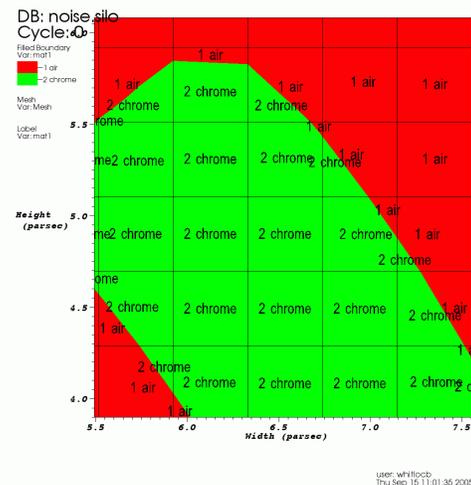


Figure 3-17: Label plot of materials

3.6 Mesh Plot

The Mesh plot, shown in Figure 3-18, displays the computational mesh over which a database's variables are defined. The mesh plot is often added to the visualization window when other plots are visualized to allow individual cells to be clearly seen.

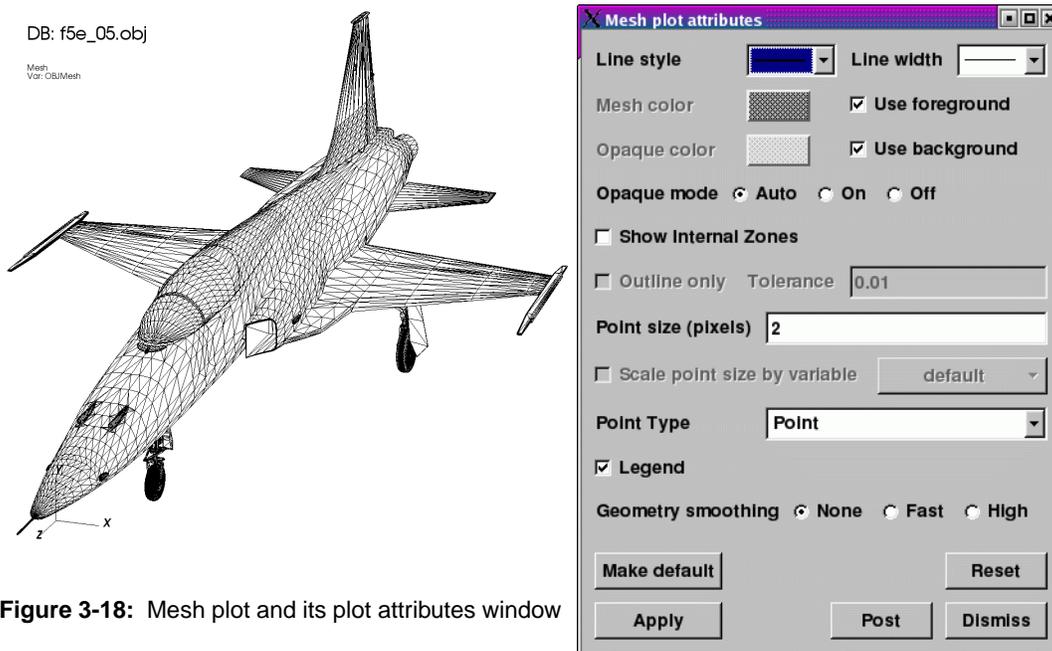


Figure 3-18: Mesh plot and its plot attributes window

3.6.1 Mesh plot opaque modes

By default, VisIt's Mesh plot draws in opaque mode so that hidden surface removal is performed when the plot is drawn and each face of the externally visible cells are outlined with lines. When the Mesh plot's opaque mode is set to automatic, the Mesh plot will be drawn in opaque mode unless it is forced to share the visualization window with other plots, at which point the Mesh plot is drawn in wireframe mode. When the Mesh plot is drawn in wireframe mode, only the edges of each externally visible cell face are drawn, which prevents the Mesh plot from interfering with the appearance of other plots. In addition to having an automatic opaque mode, the Mesh plot can be forced to be drawn in opaque mode or wireframe mode by clicking the **On** or **Off** Radio buttons to the right of the **Opaque mode** label in the **Mesh plot attributes window**.

3.6.2 Showing internal zones

Sometimes it is useful to create mesh plot that shows all internal zones for a 3D database. Rather than plotting just the externally visible zones, which is the Mesh plot's default behavior, you can click the **Show internal zones** check box to force the Mesh plot to draw the edges of every internal zone.

3.6.3 Changing the opaque color

An opaque Mesh plot uses the background color of the visualization window for the Mesh plot faces. To set the opaque color to a color other than the visualization window's background color, uncheck the **Use background** check box and click on the **Opaque color** button and select a new color from the **Popup color menu**.

3.6.4 Changing the mesh color

The mesh color is the color used to draw the mesh lines. The mesh lines normally use the visualization window's foreground color. To use a different color, uncheck the **Use foreground** check box, click the **Mesh color** button, and select a new color from the **Popup color menu**.

3.6.5 Changing mesh line attributes

The Mesh plot's mesh lines have two user-settable attributes that control their width and line style. You can set the line width and line style are set by selecting new options from the **Line style** or **Line width** menus at the top of the **Mesh plot attributes window**.

3.6.6 Changing the point size

Some databases contain point meshes, which are meshes of topological dimension zero. When the point mesh is 2D, VisIt draws it using small points. When the point mesh is 3D, VisIt draws its points as 3D cubes. To control how large the points appear, change the point size attribute by typing a new number into the **Point size** text field. Larger point size values result in larger points in the visualization window. The point size can also be scaled by a scalar variable if you check the **Scale point size by variable** check box and optionally select a variable name from the **Variable** button's menu. The initial value of "default" must be replaced with a valid scalar variable name in order for VisIt to scale the point size with a variable.

3.6.7 Changing the point type

The Mesh plot can use five different point types for drawing point meshes. The Mesh plot draws an object of the desired point type, scaled by the point size, for each point in the point mesh. Setting the point type has no effect if the plotted mesh is not a point mesh. The different point types are as follows: Box, Axis, Icosahedron, Point, and Sphere. Examples of the different point types are shown in Figure 3-19. To set the point type, select a new point type from the **Point Type** combo box. When the point type is set to Box, the Mesh plot draws a small cube for each point in the point mesh. When the point type is set to Axis, the Mesh plot draws there small axis-aligned planes for each point in the point mesh. When the point type is set to Icosahedron, the Mesh plot draws small icosahedra at each point in the point mesh. When the point type is set to Point, the Mesh plot uses flat quadrilateral points. When the point type is set to Sphere, the Mesh plot uses flat quadrilateral points with an applied texture to make them look like spheres. In general, setting the point type to: Point will cause the Mesh plot to have the fastest rendering

performance. The sphere point type is the second fastest but perhaps the best looking. Other point types can take longer to generate and render because they use additional geometry.

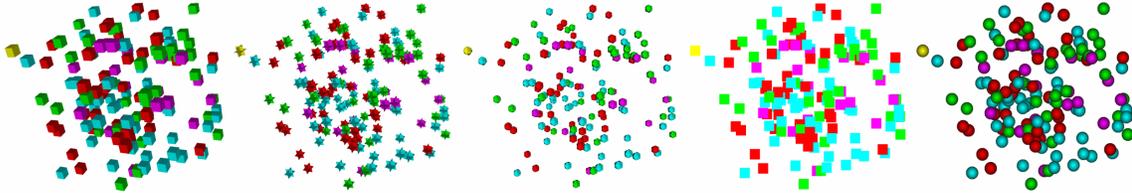


Figure 3-19: Point types: Box, Axis, Icosahedron, Point, Sphere

3.6.8 Geometry smoothing

Sometimes visualization operations such as material interface reconstruction can alter mesh surfaces so they are pointy or distorted. The Mesh plot provides an optional Geometry smoothing option to smooth out the mesh surfaces so they look better when the mesh is visualized. Geometry smoothing is not done by default, you must click the **Fast** or **High** radio buttons to enable it. The **Fast** geometry smoothing setting smooths out the geometry a little while the **High** setting works produces smoother surfaces.

3.7 Pseudocolor plot

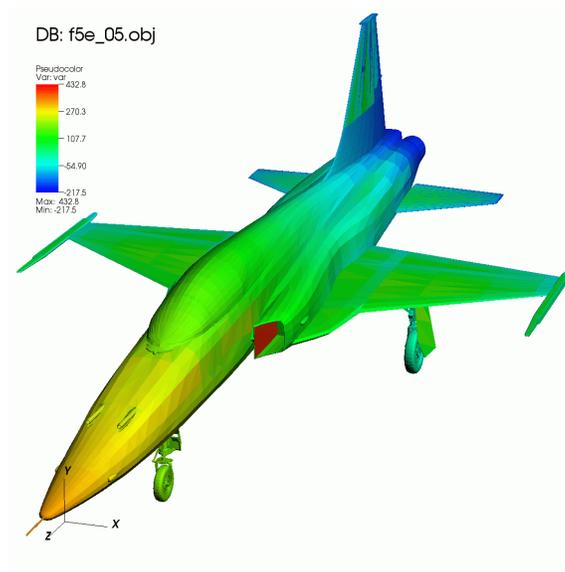


Figure 3-20: Pseudocolor plot

The Pseudocolor plot, shown in Figure 3-20, maps a scalar variable's data values to colors and uses the colors to “paint” values onto the variable's computational mesh. The result is a clear picture of the database geometry painted with variable values that have been mapped to colors. You might try this plot first when examining a scientific database for the first time since it reveals so much information about the plotted variable.

3.7.1 Variable centering

Variables in a database can be associated with a mesh in various ways. Databases supported by VisIt allow variables to be associated with a mesh's

zones (cells) or its nodes. When a variable is associated with a mesh's zones, the variable field consists of one value for each zone and is said to be *zone-centered*. When a variable

is associated with a mesh's nodes, there are values for each vertex making up the zone and the variable is said to be *node-centered*.

VisIt's **Pseudocolor plot attributes window** (shown in Figure 3-21) allows you to specify how variables should be centered. There are three settings for variable centering: **Natural**, **Nodal**, and **Zonal**. **Natural** variable centering displays the data according to the way the variable was centered on the mesh. This means that node-centered data will be displayed at the nodes with colors being linearly interpolated between the nodes, and zone-centered data will be displayed as zonal values, giving a slightly “blocky” look to the picture. If **Nodal** centering is selected, all data is displayed at the nodes regardless of the variable's natural centering. This will produce a smoother picture, but for variables which are actually zone-centered, you will lose some data (local minima and maxima). If you select **Zonal** centering, all data is displayed as if they were zone-centered. This produces a blockier picture and, again, it blurs minima/maxima for node-centered data.

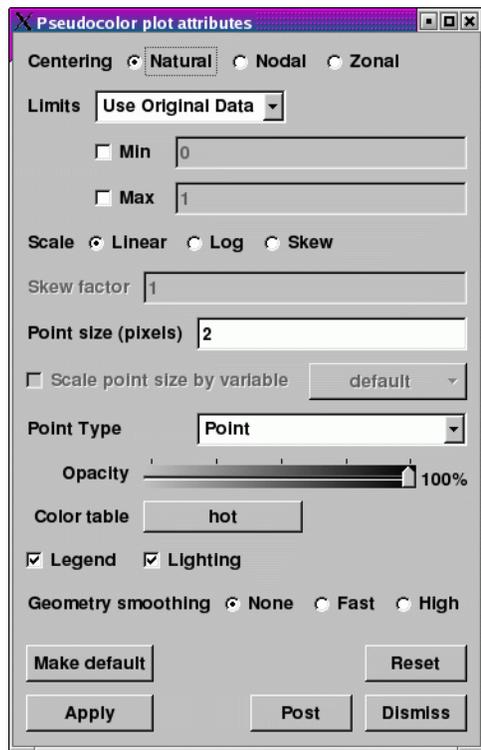


Figure 3-21: Pseudocolor plot attribute window

3.7.2 Limits

Setting limits for the plot imposes artificial minima and maxima on the plotted variable. This effectively restricts the range of data used to color the Pseudocolor plot. You might set limits when you are interested in only a small range of the data or when data limits need to be maintained for multiple time steps, as when playing an animation. In fact, we recommend setting the limits when producing an animation so the colors will correspond to the same values instead of varying over time with the range of the plotted variable. Setting limits often highlights a certain range in the data by assigning more colors to that data range.

To set the limits for the Pseudocolor plot, you must first select the limit mode. The limit mode determines whether the original data extents (data extents before any portions of the plot are removed), are used or the current plot data extents (data extents after any portions of the plot are removed), are used. To select the limit mode, choose either **Use Original Data** or **Use Current Plot** from the **Limits** menu.

The limits for the Pseudocolor plot consist of a minimum value and a maximum value. You may set these limits, and turn them on and off, independently of one another. That is, the use of one limit does not require the use of the other. To set a limit, check the **Min** or

Max check box next to the **Min** or **Max** text field and type a new limit value into the **Min** or **Max** text field.

3.7.3 Scaling the data

The scale maps data values to color values. VisIt provides three scaling options: **Linear**, **Log**, and **Skew**. **Linear**, which is the default, uses a linear mapping of data values to color values. **Log** scaling is used to map small ranges of data to larger ranges of color. **Skew** scaling goes one step further by using an exponential function based on a skew factor to adjust the mapping of data to colors. The function used in skew scaling is $(s^d - 1) / (s - 1)$ where s is a skew factor greater than zero and d is a data value that has been mapped to a range from zero to one. The mapping of data to colors is changed by changing the skew factor. A skew factor of one is equivalent to linear scaling but values either larger or smaller than one produce curves that map either the high or low end of the data to a larger color range. To change the skew factor, choose **Skew** scaling and type a new skew factor into the **Skew factor** text field.

3.7.4 Changing the color table



The Pseudocolor plot can specify which VisIt color table is used for colors. To change the color table, click on the **Color table** button, shown in Figure 3-22, and select a new color table name from the list of color tables. The list of color tables always represents the list of available VisIt color tables. If you do not care which color table is used, choose the Default option to use VisIt's active continuous color table. New color tables can be defined using VisIt's **Color table window** which is described later in this manual.

Figure 3-22: Color table button

3.7.5 Lighting

Lighting adds detail and depth to the Pseudocolor plot, two characteristics that are important for animations. The **Lighting** check box in the lower part of the **Pseudocolor plot attributes window** turns lighting on and off. Since lighting is on by default, uncheck the **Lighting** check box to turn lighting off.

3.7.6 Opacity

You can make the Pseudocolor plot transparent by changing its opacity using the **Opacity** slider. Moving the opacity slider to the left makes the plot more transparent while moving the slider to the right makes the plot more opaque. The default is fully opaque.

3.7.7 Changing the point size

Some databases scalar variables defined on meshes of topological dimension zero. When these point variables are 2D, VisIt draws them using small points that are colored by the value of the variable. When a point variable is 3D, VisIt draws it as a set of 3D cubes

colored by the value of the variable. To control how large the points appear, change the point size attribute by typing a new number into the **Point size** text field. Larger point size values result in larger points in the visualization window. The point size can also be scaled by a scalar variable if you check the **Scale point size by variable** check box and select a new scalar variable from the **Variable** button. The value “default” must be replaced with the name of another scalar variable if you want VisIt to scale the points with a variable other than the one being plotted by the Pseudocolor plot.

3.7.8 Changing the point type

The Pseudocolor plot can use five different point types for drawing point meshes (see Figure 3-23). The Pseudocolor plot draws an object of the desired point type, scaled by the point size, for each point in a point mesh. Setting the point type has no effect if the plotted mesh is not a point mesh. The different point types are as follows: Box, Axis, Icosahedron, Point, and Sphere. To set the point type choose a new point type from the **Point Type** combo box. When the point type is set to Box, the Pseudocolor plot draws a small cube for each point in the point mesh. When the point type is set to Axis, the Pseudocolor plot draws three small axis-aligned planes for each point in the point mesh. When the point type is set to Icosahedron, the Pseudocolor plot draws small icosahedra at each point in the point mesh. When the point type is set to Point, the Pseudocolor plot uses flat quadrilateral points. When the point type is set to Sphere, the Pseudocolor plot uses flat quadrilateral points with an applied texture to make them look like spheres. In general, setting the point type to Point will cause the Pseudocolor plot to have the fastest rendering performance.

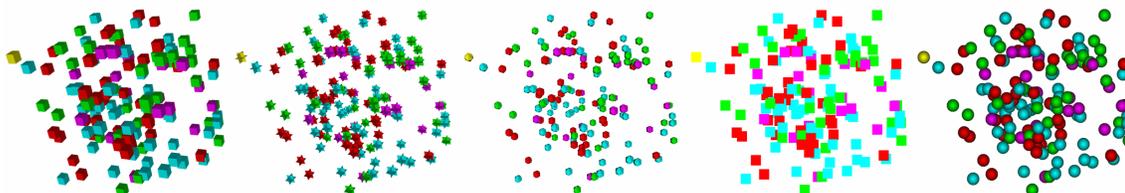


Figure 3-23: Point types: Box, Axis, Icosahedron, Point, Sphere

3.7.9 Geometry smoothing

Sometimes visualization operations such as material interface reconstruction can alter mesh surfaces so they are pointy or distorted. The Pseudocolor plot provides an optional Geometry smoothing option to smooth out the mesh surfaces so they look better when the plot is visualized. Geometry smoothing is not done by default, you must click the **Fast** or **High** radio buttons to enable it. The **Fast** geometry smoothing setting smooths out the geometry a little while the **High** setting works produces smoother surfaces.

3.8 Scatter Plot

The Scatter plot (see Figure 3-24) allows you to combine multiple scalar fields into a point mesh so you can investigate the relationships between multiple input variables. You might, for example, want to see the behavior of pressure vs. density colored by temperature. The Scatter plot can take up to four scalar fields as input and can use up to three of them as coordinates for the created point mesh while one input variable can be used to assign colors to the point mesh. The Scatter plot provides individual controls for setting the limits of each input variable and also allows each input variable to be scaled so that all of the resulting points from disparate data ranges fit in a unit cube.

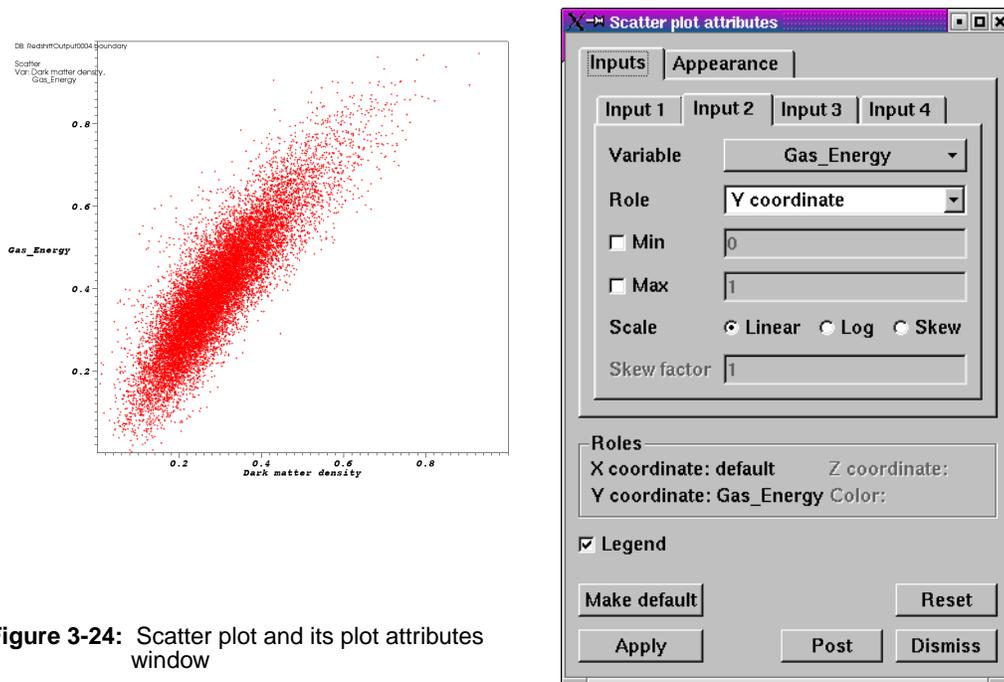


Figure 3-24: Scatter plot and its plot attributes window

The **Scatter plot attributes window** is divided into two tabs: **Inputs** and **Appearance**. The **Inputs** tab is further subdivided into tabs for each input variable. Each tab for an input variable contains controls that pertain to selecting the input variable, settings its limits, or setting the role that the input variable will perform within the Scatter plot. Each input variable can have one of five roles that will be covered later. The **Appearance** tab contains controls for changing the Scatter plot's appearance. Under the two main tabs, the **Scatter plot attributes window** features a small section that lists the roles that are used in the plot and which input variables are assigned to each role.

3.8.1 Scatter plot wizard

Plots are typically created in VisIt when you choose a variable from one of the **Plot menus**. Since the Scatter plot takes as input up to four input variables and typical plot creation only initializes one variable, you can imagine that if a Scatter plot was created the usual way, only one of its many input variables would be initialized. Furthermore, to initialize the plot, you would have to open the **Scatter plot attributes window** and select the other variables. Since that would not be a very straightforward way to create a Scatter plot, VisIt now has support for plot wizards. A plot wizard is a simple dialog window that pops up when you select a variable to plot. A plot wizard leads you through a series of questions that allow VisIt to more fully initialize a new plot. The Scatter plot is the first of VisIt's plots to support plot wizards. The **Scatter plot wizard** prompts you for the scalar variable to use for the Y-Axis, the variable to use for the Z-Axis (optional), and the variable to use for the plot's colors (optional).

3.8.2 Selecting a variable

Three of the Scatter plot's four input variables can be set in the **Scatter plot attributes window**. The first input variable cannot be changed from within the **Scatter plot attributes window** because that is the default variable used by the plot. If you want to change the first input variable, you can use the **Variables** menu under the **Plot list**. If you want to select a different variable for any of the other input variables, you would first click on the input variable's tab and then you would select a new variable by making a selection from the tab's **Variable** button. Note that any combination of nodal and cell-centered variables can be chosen. The Scatter plot will recenter any input variables whose centering does not match the first input variable's centering.

3.8.3 Setting an input variable's role

Each of the Scatter plot's input variables has a role that you can set which determines how the input variable is used by the Scatter plot. An input variable can be used for the X, Y, Z coordinates, for the color, or it can have no role. The role of the input variable is not fixed

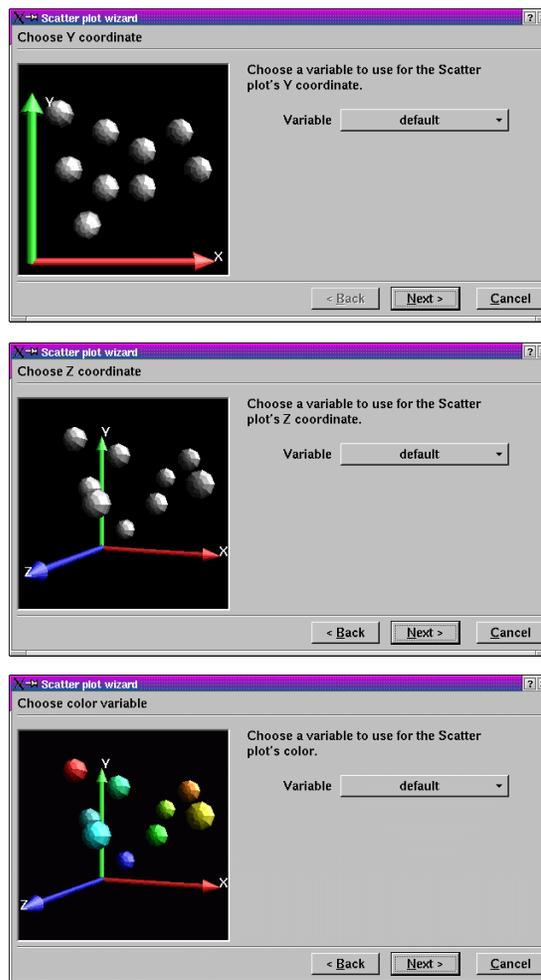


Figure 3-25: Pages from the Scatter plot wizard

because you might want to change roles many times and it is much less work to change only the roles instead of reselecting variables, limits, and scaling for an input variable. The flexibility of selecting a role for an input variable makes it convenient to turn off colors or the Z coordinate with little effort. To change the role for an input variable, select a new role from the input variable's **Role** combo box. If you select a role that is already played by another input variable, VisIt will give the current input variable the selected role and set the input variable that previously had the selected role so that it has no role.

Each of the Scatter plot roles and their associated input variables are listed in the bottom of the **Scatter plot attributes window**. Roles that have an input variable have the name of the input variable printed next to the name of the role so looking through all of the input variable tabs to determine what the Scatter plot should look like is not required. Roles that have no assigned input variable are grayed out.

3.8.4 Setting the minimum and maximum values

The Scatter plot allows you to set minimum and maximum limits on the values considered for inclusion into the created point mesh. If an input variable's data value does not lie in the specified minimum/maximum value data range then the point is not included in the created point mesh. Note that setting limits does not cause points to be removed when data values in the color role fall outside of the specified limits. To set the minimum value to be allowed in the created point mesh, click on the **Min** check box and type a new minimum value into the **Min** text field. To set the maximum value to be allowed in the created point mesh, click on the **Max** check box and type a new value into the **Max** text field.

3.8.5 Scaling an input variable

Sometimes input variable data values are clustered in a certain range of the data. When this is the case, the points in the Scatter plot will bunch up in one or more dimensions. For more uniformly spaced points, you might try scaling one or more input variables. Each input variable can be scaled in the three common ways: Linear, Log, and Skew. To set the scaling method used for the input variable, click on the **Linear**, **Log**, or **Skew** radio buttons. If you choose the Skew scaling method then you should also enter a value greater than zero into the **Skew factor** text field to determine the function used for skew scaling.

Since the Scatter plot's input variables are likely to have wildly different data ranges, the Scatter plot provides an option to independently scale each input variable so it is in the range [0,1] so the resulting plot fits entirely in a cube. If you prefer to see the Scatter plot without this corrective scaling, you can turn off the Scale to cube check box on the **Scatter plot attribute window's Appearance** tab.

3.8.6 Setting point properties

The Scatter plot can draw its points in five different styles: Box, Axis, Icosahedron, Point, and Sphere. The default value of Point is the fastest and forces the Scatter plot to draw all of its points as tiny points. When the Scatter plot uses the Sphere point type, it draws points but applies textures to the points so it is nearly as fast as the Point point type. Any of the other point styles place a glyph at each point in the Scatter plot's created point mesh, taking longer to render. To change the point type used to draw the Scatter plot's points, click on the **Appearance** tab in the **Scatter plot attributes window** and choose a new option from the **Point Type** combo box shown in Figure 3-26. If you choose any of the glyphed point types (all except Point and Sphere) then you can also specify a point size by typing a new value into the **Point size** text field. The point size is used to determine the size of the glyph. For example, if you choose the Box point type and you enter a point size of: 0.1 then the length of all of the edges on the Box glyphs will be 0.1. If you use Point or Sphere point types then the **Point size** text field becomes the **Point size (pixels)** text field and you can set the point size in terms of pixels.

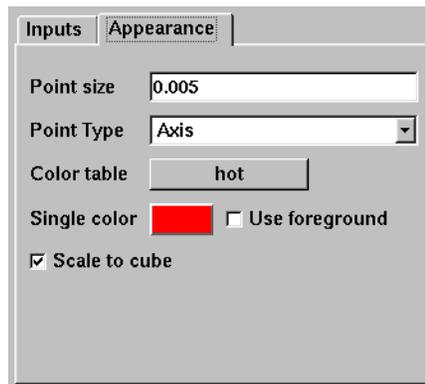


Figure 3-26: Scatter plot attributes window's Appearance tab

3.8.7 Setting the colors

The Scatter plot can map scalar values to colors like the Pseudocolor plot (page 60) does or it can color all points using a single color. If you have set one of the input variables to have a color role then the Scatter plot will map that input variable's data values to colors using the specified color table. To change the color table used by the Scatter plot, click on the **Color table** button and select a new color table from the list of available color tables. If the Scatter plot has been configured such that none of the input variables is playing the color role then the Scatter plot's points will be drawn using one color. When the Scatter plot draws its points using a single color, its default behavior is to color the points using the vis window's foreground color. If you want to instead use a different color, turn off the **Use foreground** check box and click on the **Single color** color button to select a new color.

3.9 Streamline Plot

The Streamline plot (example shown in Figure 3-27) shows the behavior of particles in a vector field. The Streamline plot calculates the value of the vector field at seed locations, which are produced by point sources, and integrates through the vector field to create a streamline. Streamlines can be displayed as a line, a tube, or as a ribbon if you also want to show the vorticity of the vector field. Once a set of streamlines is produced, vortices, etc present in the vector field are apparent in the visualization. The Streamline plot is special

among VisIt's plots in that it can be modified using VisIt's interactive tools. For instance, if the Streamline plot is set to use a Plane source for its seed points, moving VisIt's Plane tool will move the plane that the Streamline plot uses forcing VisIt to recalculate the plot's streamlines using the new plane. For more information on interactive tools, see the **Interactive Tools** chapter on page 273.

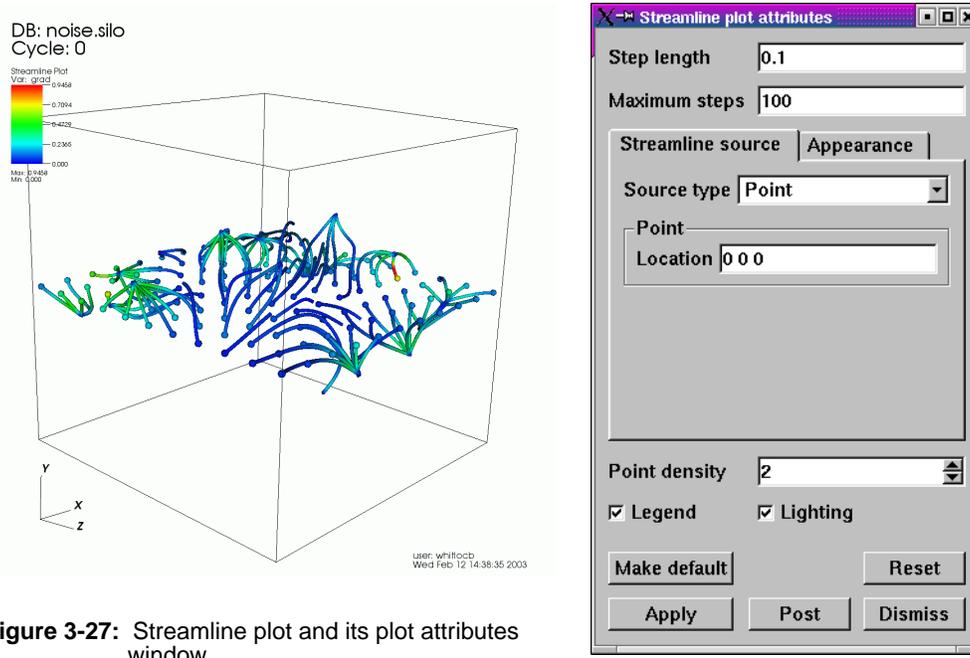


Figure 3-27: Streamline plot and its plot attributes window

VisIt's Streamline plot must be initialized more completely before you click the **Draw** button to generate the plot. First of all, you must take special care to set the step length, which is the maximum length that will be integrated through the vector field for each step. The step length should be set to a value approximating the average width of the cells in the mesh to ensure that the streamlines are as accurate as possible. After choosing a suitable step length, be sure to specify a maximum number of steps. The maximum number of steps is the number of integrations that will be performed for each streamline. In a vector field that is free of attractors, a larger number for maximum steps will generate longer streamlines.

3.9.1 Making longer streamlines

Longer streamlines can be generated by typing a larger value into the **Maximum steps** text field. The maximum steps value indicates the maximum number of integrations through the vector field for each streamline. Increasing the maximum steps value allows streamlines to become longer.

3.9.2 Making smoother looking streamlines

VisIt provides a default value of 1 for the step length taken when integrating a vector field to create streamlines. The step length is the added to the starting location of the streampoint in the direction of the vector field to give the point in the streamline where the

process repeats. The value that you need to provide is very much tied to the spatial and data extents of your data but usually smaller values produce smoother looking streamlines. You should take care not to reduce the step length too rapidly because smaller step lengths also result in streamlines that take longer to calculate. Larger step length values result in streamlines that have visible sharp bends but they are faster to calculate. To change the step length, you type a new length into the **Step length** text field.

3.9.3 Setting the streamline source

The Streamline plot's source type determines the geometry of its seed points, which are the points from which streamlines are calculated. Five different source types: Point, Line, Plane, Sphere, and Box are available in the **Source type menu** on the **Streamline source** tab of the **Streamline plot attributes window**. You can select a new source type by selecting one of the source types in the **Source type menu**. Once a new source type is selected, the area immediately below the **Source type menu**, known as the **Source type area**, changes to reflect the source type that was selected. For example, if a Box source type was selected, the **Source type area** changes to show the text fields used to specify coordinates for the box.

For Point source types, the **Source type area** displays a single text field into which you can type a 3D point to specify the location of the point.

When the source type is set to Line (see Figure 3-28), the **Source type area** displays two text fields that allow you to enter 3D points for the start end and locations of the source line. The Streamline plot creates a number of points, specified by the point density, along the length of the source line. More seed points can be added along the length the source line by increasing the point density, which you can do by typing a larger number into the **Point density** text field or by clicking on its up arrow. The endpoints of the source line can also be modified using VisIt's interactive Line tool. First make sure that the Streamline plot is selected in the **Main Window's Plot list** and then enable the Line tool using the visualization window's toolbar or its popup menu. When the Line tool is enabled, moving its endpoints supplies the Streamline plot with new endpoints for its source line and the plot calculates new streamlines.

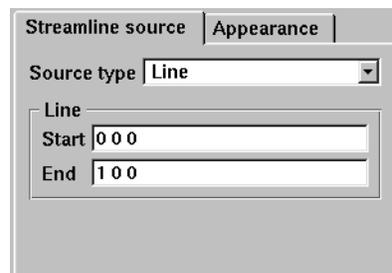


Figure 3-28: Line source type

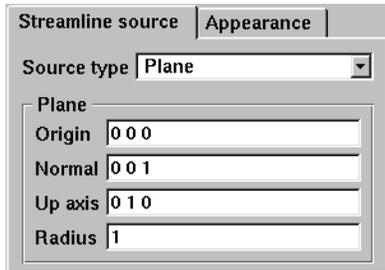


Figure 3-29: Plane source type

When the source type is set to Plane (see Figure 3-29), the **Source type area** displays four text fields that allow you to enter a plane equation for the source plane. To move the source plane, you type a new 3D point into the **Origin text** field. To change the orientation of the source plane, you type a new 3D vector, the plane normal, into the **Normal text** field. The plane normal is a vector that is perpendicular to the source plane. Together with the plane origin, this is all that is necessary to specify a plane in 3D. However, more information is needed to specify the bounds of the source plane. For this purpose, the **Streamline plot attributes window** provides the **Up axis** and **Radius** text fields. Imagine a square, centered at the plane source origin, that lying in the same plane as the plane source. The distance to one of the square's corners is the plane radius. The plane radius determines the size of the plane source and it can be changed by typing a new value into the **Radius** text field. The up axis is a vector lying in the plane that determines which way is up for the square. You can change the up axis by typing a new 3D vector into the **Up Axis** text field. The Streamline plot creates a grid of evenly spaced seed points in the square defined by the source plane and source plane radius. The grid has a number of points equal to the point density plus one in both dimensions. The plane source can be modified using the controls in the **Streamline plot attributes window** or by using VisIt's interactive Plane tool. When a Streamline plot is selected the **Plot list**, moving the plane tool changes the source plane and the plot calculates new streamlines.

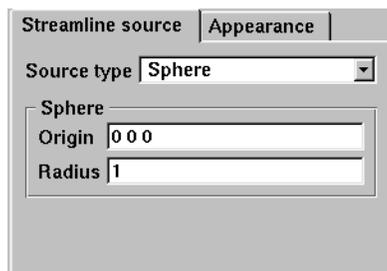


Figure 3-30: Sphere source type

When the source type is set to Sphere (see Figure 3-30), the **Source type area** displays two text fields that allow you to enter a sphere origin and radius to specify the source sphere. To move the source sphere, type a new 3D point into the **Origin** text field. To change the radius of the sphere, you type a new radius value into the **Radius** text field. The Streamline plot creates a number of seed points on the surface of the sphere and ,again, the number of points is a function of the point density. The sphere source can be modified using the controls in the window or by using VisIt's interactive sphere tool. When a Streamline plot is selected in the **Plot list**, moving the sphere tool changes the source sphere and the plot calculates new streamlines.

When the source type is set to Box (see Figure 3-31), the **Source type area** displays three text fields that allow you to enter the minimum and maximum values for the x,y,z dimensions that are used to specify the box source. You can enter new minimum or maximum values by typing in the **X Extents**, **Y Extents**, or **Z Extents** text fields. The Streamline plot creates a 3D grid of evenly spaced seed points in the source box. The grid has a number of points equal to the point density plus one in all three dimensions. The box source can be modified using the window's controls or by using VisIt's interactive Box tool. When a Streamline plot is selected in the **Plot list**, moving the box tool changes the source box and calculates new streamlines.

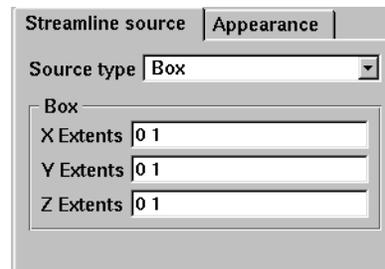


Figure 3-31: Box source type

3.9.4 Changing the number of streamlines

The number of streamlines varies depending on the source type but all source types use the point density to determine the number of streamlines. To change the number of streamlines, type a new point density into the **Point density** text field in the **Streamline plot attributes window**. The point density can easily be incremented or decremented by clicking the up or down arrows next to the **Point density** text field.

3.9.5 Setting streamline appearance

The Streamline plot's streamlines can be drawn as lines, tubes, or ribbons (see Figure 3-32). Drawing the streamlines as lines, which is the default, is faster than drawing them as tubes or ribbons but it is not as visually appealing. When streamlines are drawn as tubes, the streamlines appear thicker and they have a small sphere that indicates the streamline's starting point. When the streamlines are drawn as ribbons, they twist in response to the vector field's vorticity.

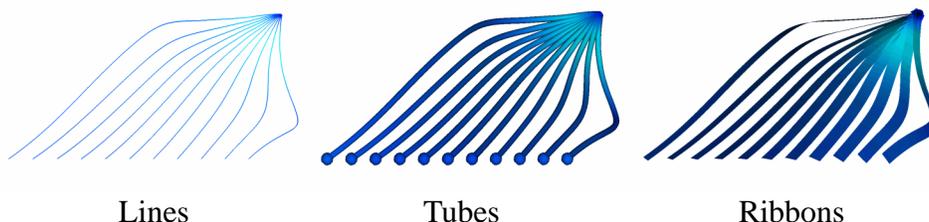


Figure 3-32: Streamline display methods

To display the streamlines as tubes, you can select Tubes from the **Display as** combo box on the **Streamline plot attributes window's Appearance tab** (see Figure 3-33). To display the streamlines as ribbons, you can select Ribbons from the **Display as** combo box. When streamlines are drawn as tubes, you can elect to show their start position by clicking the **Show start** check box. The look of streamline tubes and ribbons can be further modified by changing the radius, which affects the thickness of tubes and ribbons

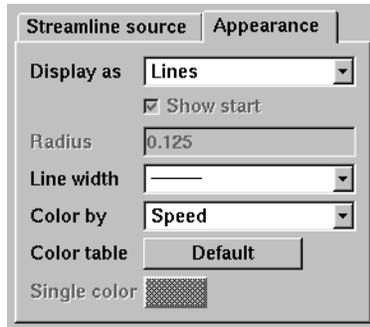


Figure 3-33: Streamline appearance

by typing a new value into the **Radius** text field. When the Streamline plot is drawn as lines, the line width can be set by selecting a new line width from the **Line width** combo box.

3.9.6 Changing the streamline color

Streamline plots can be colored by the vector field's velocity magnitude, vorticity magnitude, or they can be colored using a constant color. To select the coloring method for the streamlines, select an option from the **Color by** combo box and select a color table by selecting a color table from the **Color table** menu. If a single color is used for all streamlines, you can set a new color by clicking on the **Single color** button and selecting a new color from the **Popup color menu**.

3.9.7 Lighting

Lighting adds detail and depth to the Streamline plot when streamline tubes are used. The **Lighting** check box in the lower part of the **Streamline plot attributes window** turns lighting on and off. Since lighting is on by default, uncheck the **Lighting** check box to turn lighting off.

3.10 Subset Plot

The Subset plot (example in Figure 3-34) is used to display subset relationships. The typical scientific database can be decomposed into many different subsets. Frequently a database is decomposed into material subsets or assembly subsets. The Subset plot draws the database with its various subsets color coded so they can be distinguished. For more information about subsets, see the **Subsetting** chapter on page 163. For information on the Boundary or FilledBoundary plots, which are related to the Subset plot, refer back to page 45.

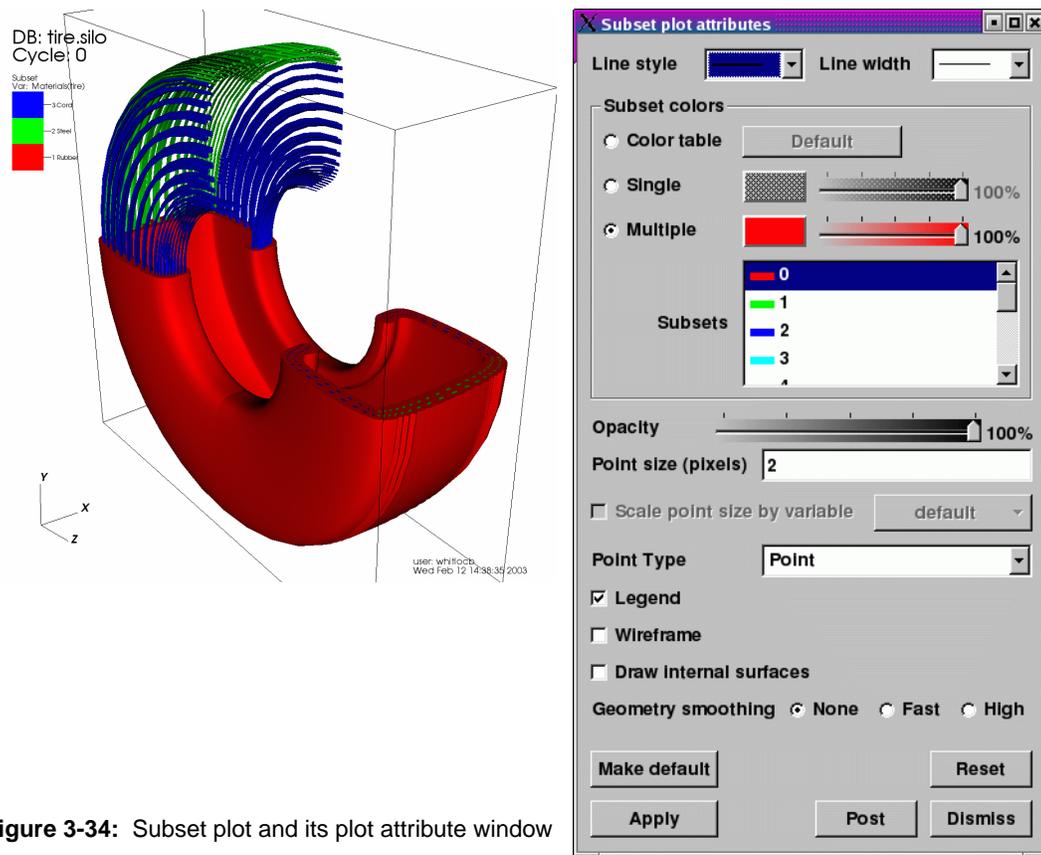


Figure 3-34: Subset plot and its plot attribute window

3.10.1 Changing colors

The main portion of the **Subset plot attributes window**, also known as the **Subset colors area**, is devoted to setting subset colors. The **Subset colors area** contains a list of subset names with an associated subset color. Subset plot colors can be assigned three different ways, the first of which uses a color table. A color table is a named palette of colors that you can customize to suite your needs. When the Subset plot uses a color table to color subsets, it selects colors that are evenly spaced through the color table based on the number of subsets. For example, if you have three subsets and you are coloring them using the “xray” color table, the Subset plot picks three colors out of the color table so your levels are colored black, gray, and white. To color a Subset plot with a color table, click on the **Color table radio button** and choose a color table from the **Color table menu** to right of the **Color table radio button**.

If you want all subsets to be the same color, click the **Single** radio button at the top of the **Subset plot attributes window** and select a new color from the **Popup color menu** that is activated by clicking on the **Single color button**. The opacity slider next to the **Single color button** sets the opacity for the single color.

Clicking the **Multiple** radio button causes each subset to be a different, user-specified color. By default, multiple colors are set using the colors of the discrete color table that is active when the Subset plot is created. To change the color for any of the subsets, select one or more subsets from the list of subsets and click on the **Color button** to the right of the **Multiple** radio button and select a new color from the **Popup color menu**. To change the opacity for a subset, move **Multiple** opacity slider to the left to make the subset more transparent or move the slider to the right to make the subset more opaque.

The **Subset plot attributes window** contains a list of subset names with an associated subset color. To change a subset's color, select one or more subsets from the list, click the color button and select a new color from the popup color menu.

3.10.2 Opacity

The Subset plot's opacity can be changed globally as well as on a per subset basis. To change subset opacity, first select one or more subsets in the subset list and move the opacity slider next to the color button. Moving the opacity slider to the left makes the selected subsets more transparent and moving the slider to the right makes the selected subsets more opaque. To change the entire plot's opacity globally, use the **Opacity** slider near the bottom of the window.

3.10.3 Setting point properties

Albeit rare, the Subset plot can be used to plot points that belong to different subsets so the **Subset plot attributes window** provides controls that allow you to set the representation and size of the points. You can change the points' representation using the **Point Type** combo box. The available options are: **Box**, **Axis**, **Icosahedron**, **Point**, and **Sphere**. To change the size of the points, you can enter a new floating point value into the **Point size** text field. Finally, you can opt to scale the points' glyphs using a scalar expression by turning on the **Scale point size by variable** check box and by selecting a scalar variable from the **Variable** button to the right of that check box.

3.10.4 Wireframe mode

The Subset plot can be modified so that it only displays outer edges of subsets. This option usually leaves lines that give only the rough shape of subsets and where they join other subsets. To make the Subset plot display in wireframe mode, check the **Wireframe** check box near the bottom of the **Subset plot attributes window**.

3.10.5 Drawing internal surfaces

When you make one or more subsets transparent, you might want to make the Subset plot draw internal surfaces. Internal surfaces are normally removed from Subset plots to make them draw faster. To make the Subset plot draw internal surfaces, check the **Draw internal surfaces** check box near the bottom of the **Subset plot attributes window**.

3.10.6 Geometry smoothing

Sometimes visualization operations such as material interface reconstruction can alter mesh surfaces so they are pointy or distorted. The Subset plot provides an optional Geometry smoothing option to smooth out the mesh surfaces so they look better when the plot is visualized. Geometry smoothing is not done by default, you must click the **Fast** or **High** radio buttons to enable it. The **Fast** geometry smoothing setting smooths out the geometry a little while the **High** setting works produces smoother surfaces.

3.11 Surface Plot

The Surface plot (example shown in Figure 3-35) takes 2D scalar databases as input and adds a height component to the variable's mesh, resulting in a height map that is then pseudocolored by the plotted variable. You might want to use this plot to examine 2D datasets because features of the plotted variable are highlighted by the height of the plot in addition to being highlighted by the plot's colors.

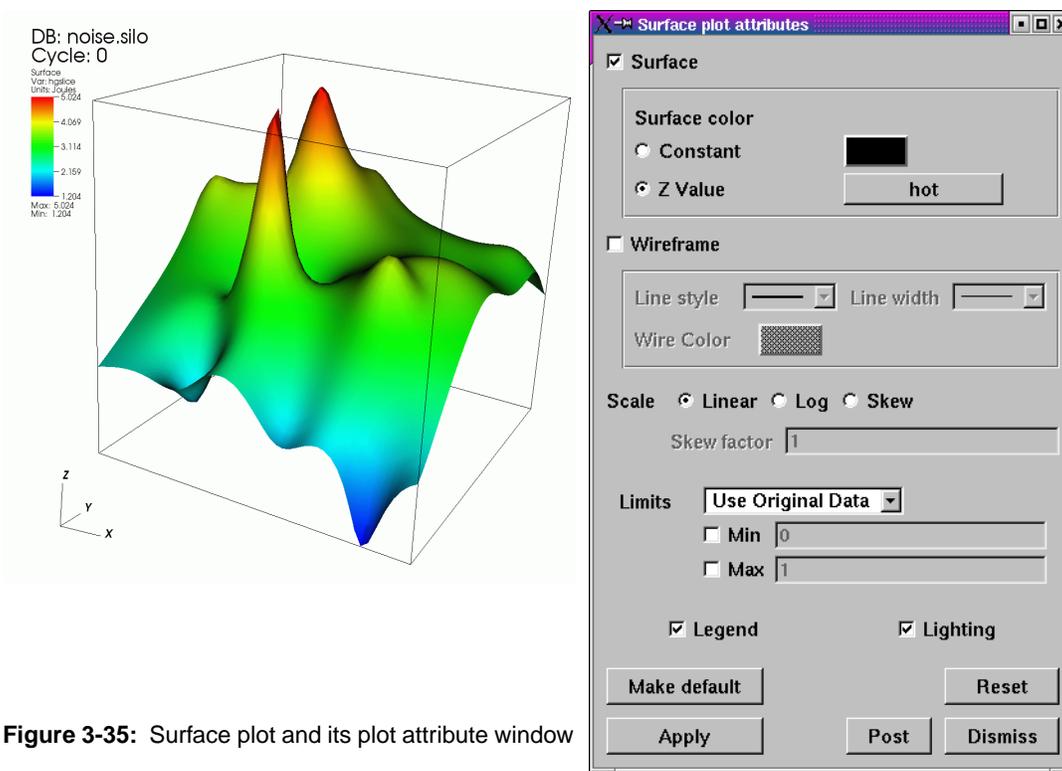


Figure 3-35: Surface plot and its plot attribute window

3.11.1 Surface and Wireframe modes

The Surface plot can be drawn in different ways. The default appearance of the Surface plot draws the surface only. Wireframes, which are essentially mesh lines, can also be drawn. You can draw wireframes with or without the surface. To select which parts of the

plot are drawn, check the **Surface** and **Wireframe** check boxes in the **Surface plot attributes window**.

3.11.2 Surface color

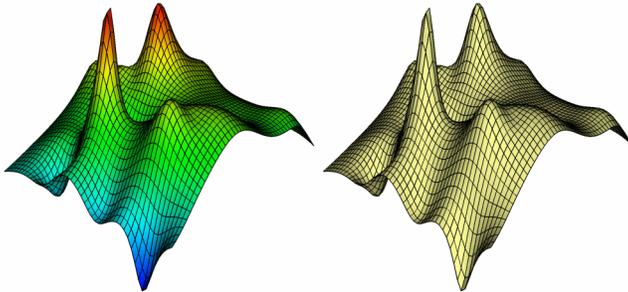


Figure 3-36: Surface plot colored by Z-value (left) and constant color (right)

By default, the Surface plot is colored by the variable value or Z-value as with the Pseudocolor plot. The second coloration scheme uses constant coloration where the entire surface is the same color (see Figure 3-36). To choose a coloration scheme, click on the **Constant** or **Z-Value** radio buttons. When the plot uses constant coloring, you can change the color by clicking on the color button next to the **Constant** radio button and

selecting a new color from the popup color menu. When the plot uses coloring based on the **Z-Value**, you change colors by selecting a new color table name from the color table button next to the **Z-Value** radio button. The available color table names are an up-to-date list of VisIt color table names.

3.11.3 Wireframe properties

The Surface plot's wireframe lines can have different line styles, widths, and colors. To set the line width, make a selection from the **Line width** menu. To set the line style, make a selection from the **Line style** menu. To change the wireframe color, click on the **Wire color** button and choose a new color from the popup color menu. Note that wireframes must be enabled to set their properties.

3.11.4 Scaling the data

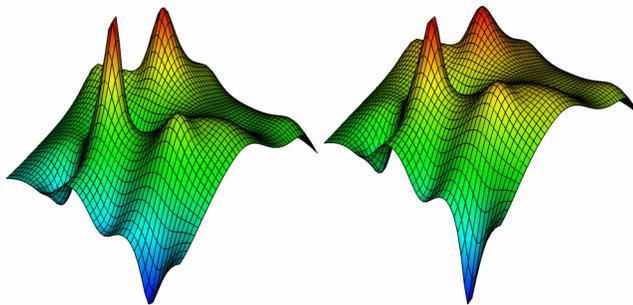


Figure 3-37: Linear scale (left) and Log scale (right)

The scale is map data values to color values and surface height. VisIt provides three scaling options: **Linear**, **Log**, and **Skew**. **Linear**, which is the default, uses a linear mapping of data values to color values. **Log** scaling is used to map small ranges of data to larger ranges of color (see Figure 3-37). **Skew** scaling (Figure 3-38) goes one step further by using an exponential function based on a skew factor to adjust the mapping of data to colors

and surface height. The function used in skew scaling is $(s^d - 1) / (s - 1)$ where s is a skew

factor greater than zero and d is a data value that has been mapped to a range from zero to one. **Skew** scaling can be customized by changing the skew factor. A skew factor of one is equivalent to linear scaling but values either larger or smaller than one produce curves that map either the high or low end of the data to a larger color range. To change the skew factor, choose **Skew** scaling and type a new skew factor into the **Skew factor** text field.

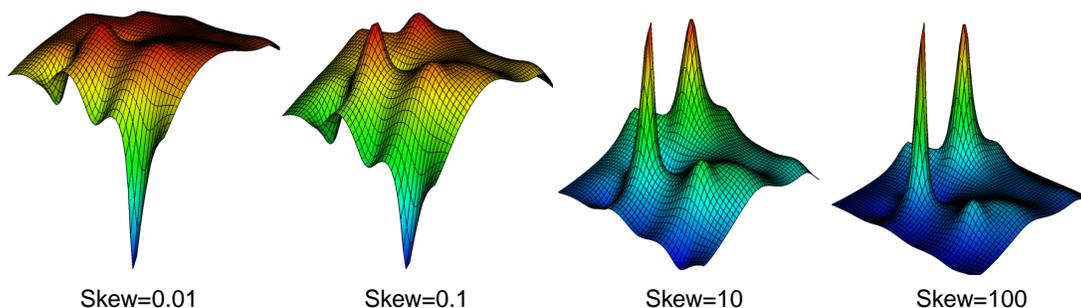


Figure 3-38: Effects of skew factor on Surface plot

3.11.5 Limits

Setting limits for the plot imposes artificial minima and maxima on the plotted variable. This effectively restricts the range of data used to color and set the height for the Surface plot. You might set limits when you are only interested in a small range of the data or when data limits need to be maintained for multiple time steps, as when playing an animation. Setting limits often highlights a certain range in the data by assigning more colors to that data range.

When setting the limits for the Surface plot, the first option to set is the limit mode. The limit mode determines whether the original data extents are used or the current plot data extents, which may vary, are used. To select the limit mode, choose either **Use Original Data** or **Use Current Plot** from the **Limits** menu.

The limits for the Surface plot consist of a minimum value and a maximum value. You may set these limits, and turn them on and off, independently of one another. That is, the use of one limit does not require the use of the other. To set a limit, check the **Min** or **Max** check box next to the **Min** or **Max** text field and type a new limit value into the **Min** or **Max** text field.

3.11.6 Lighting

Lighting adds detail and depth, two characteristics that are important for animations, to the Surface plot. You can click the **Lighting** check box in the lower part of the **Surface plot attributes window** to turn lighting on and off. Since lighting is on by default, uncheck the **Lighting** check box to turn lighting off.

3.12 Tensor plot

The Tensor plot, shown in Figure 3-39, displays tensor variables using ellipsoid glyphs to convey information about a tensor variable's eigenvalues. Each glyph's scaling and rotation is controlled by the eigenvalues/eigenvectors of the tensor as follows: for each tensor, the eigenvalues (and associated eigenvectors) are sorted to determine the major, medium, and minor eigenvalues/eigenvectors. The major eigenvalue scales the glyph in the x-direction, the medium in the y-direction, and the minor in the z-direction. Then, the glyph is rotated so that the glyph's local x-axis lies along the major eigenvector, y-axis along the medium eigenvector, and z-axis along the minor.

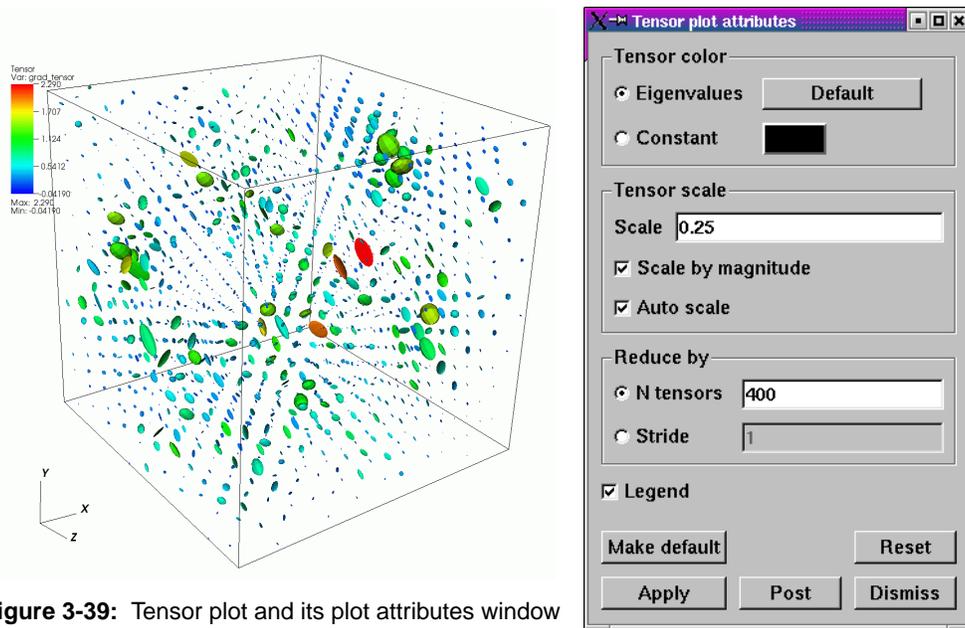


Figure 3-39: Tensor plot and its plot attributes window

3.12.1 Changing the tensor colors

The Tensor plot can be colored by a solid color or by the corresponding to the largest eigenvalue. To color the Tensor plot by eigenvalues, click the **Eigenvalues** radio button and then select a color table name from the color table button to the right of the **Eigenvalues** radio button. To make all tensor glyphs be the same color, click the **Constant** radio button and choose a color by clicking on the **Constant color button** and selecting a new color from the **Popup color menu**.

3.12.2 Setting the tensor scale

The Tensor plot's tensor scale affects how large the ellipsoidal glyphs that represent the tensor are drawn. By default, VisIt computes an automatic scale factor based on the length of the bounding box's diagonal to multiply by the user-specified scale factor. This ensures that the tensors are some reasonable size independent of the size of the mesh. To change the tensor scale, type a new floating point number into the **Scale** text field and click the **Apply** button in the **Tensor plot attributes window**. If you want to turn off automatic

scaling so the size of the tensors is solely determined by the scale in the Scale text field, turn off the Auto scale check box. Yet another scaling option for tensors is scaling by magnitude. When the **Scale by magnitude** check box is checked, the magnitude of the tensor's longest eigenvector is used as a scale factor that is multiplied into the scale determined by the user-specified scale and the automatic scale factor.

3.12.3 Setting the number of tensors

When visualizing a large database, a Tensor plot will often have too many tensors to effectively visualize so the Tensor plot provides controls to reduce the number of tensors to a number that looks appealing in a visualization. You can accomplish this reduction by setting a fixed number of tensors or by setting a stride. To set a fixed number of tensors, select the **N tensors** radio button and enter a new number of tensors into the **N tensors** text field. To reduce the number of tensors by setting the stride, select the **Stride** radio button and enter a new stride value into the **Stride** text field.

3.13 Truecolor plot

The Truecolor plot, shown in Figure 3-40, is used to plot images of observational or experimental data so they can be compared to other plots, possibly of related, simulated data, in the same visualization window. The Truecolor plot takes in a color variable, represented in VisIt as a three or four component vector, and uses the vector components as the red, green, blue, and alpha values for the plotted image. This allows you access to many more colors than other plots like the Pseudocolor plot, which can be used only to plot a single color component of an image.

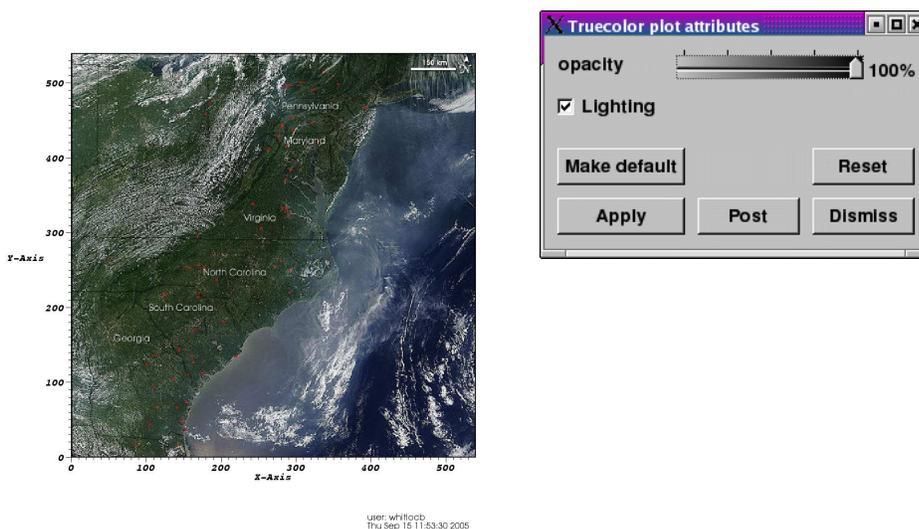


Figure 3-40: Truecolor plot and its plot attributes window. The Truecolor plot in the above example displays satellite imagery of the U.S. east coast.

3.14 Vector plot

The Vector plot (example shown in Figure 3-41) displays vector variables as small glyphs that indicate the direction and magnitude of vectors in a vector field.

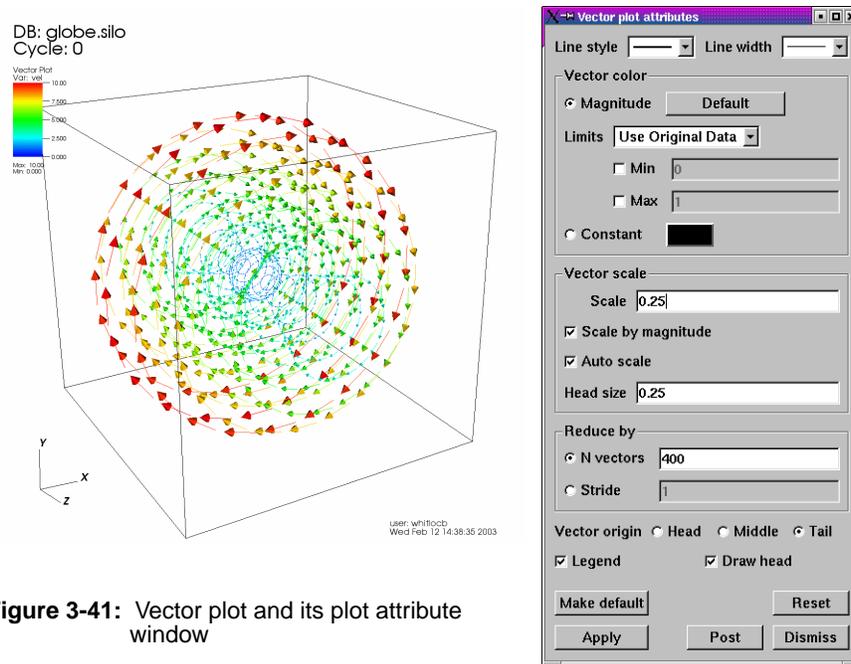


Figure 3-41: Vector plot and its plot attribute window

3.14.1 Setting vector color

The vectors in the Vector plot can be colored by the magnitude of the vector variable or they can be colored using a constant color. Choose the coloring method by clicking on either the **Magnitude** radio button or the **Constant** color button. When vectors are colored by a constant color, you can change the color by clicking on the color button next to the **Constant** radio button and choosing a new color from the **Popup color menu**. When vectors are colored by magnitude, the color is determined by one of VisIt's color tables, which can be chosen from the **Color table** button next to the **Magnitude** radio button.

If you choose to color the vectors by their magnitudes, you have the option of also specifying minimum and maximum values to aid in the mapping of vector magnitude to color. The options that are used to aid coloring are collectively known as limits. Limits can apply to all vectors that exist in the dataset or just the vectors that have been drawn by the Vector plot. To specify which, choose the appropriate option from the **Limits** combo box. When you specify a minimum value all vectors with magnitudes less than the minimum value are colored using the color at the bottom of the color table. When you specify a maximum value all vectors with magnitudes larger than the maximum value are colored using the color at the top of the color table. To provide a minimum value, check the **Min** check box and type a new minimum value into the **Min** text field. To provide a maximum value, check the **Max** check box and type a new maximum value into the **Max** text field.

3.14.2 Vector scaling

The size of the vector glyphs has a tremendous effect on the Vector plot's readability. VisIt uses an automatically computed scaling factor based on the diagonal of the bounding box as the size for the largest vector. You can augment this size by entering a new scale factor in to the **Scale** text field. It is also possible to turn off automatic scaling by turning off the **Auto scale** check box. When automatic scaling is turned off, the vectors in the Vector plot are the length specified in the **Scale** text field.

If you want each vector to be further scaled by its own magnitude, you can turn on the **Scale by magnitude** check box. When the **Scale by magnitude** check box is off, all vectors are the same length as determined by the automatically computed scale factor and the user-specified scale.

3.14.3 Heads on the vector glyph

You can control the vector head size by typing a new value into the **Head size** text field, which is the fraction of the entire vector's length that will be devoted to the vector's head. Vectors in the Vector plot can be drawn without vector heads so that only the line part of the vector glyph is drawn. This results in cleaner plots, but the vector direction is lost. To turn off vector heads, uncheck the **Draw head** check box at the bottom of the **Vector plot attributes window**.

3.14.4 Tails on the vector glyph

The length of the tails on the vector glyph are determined by the vector scaling factors that have been enabled. You can also set properties that determine the location and line properties used to draw a vector glyph's tail. First of all, you can set the line style used to draw the vector glyph's tail by choosing a line style from the **Line style** combo box. You can choose a new line width for the vector glyph's tail by choosing a new line width from the **Line width** combo box. Finally, you can determine where the origin of the vector is on the vector glyph. The vector origin is a point along the length of the vector that is aligned with the node or cell center where the vector glyph will be drawn. The available options are: Head, Middle, and Tail. You can choose a new Vector origin by clicking on one of the **Head**, **Middle**, or **Tail** radio buttons.

3.14.5 Setting the number of vectors

When visualizing a large database, a Vector plot will often have too many vectors. The Vector plot becomes incomprehensible with too many vectors. VisIt provides controls to thin the number of vectors to a number that looks appealing in a visualization. You can accomplish this reduction by setting a fixed number of vectors or by setting a stride. To set a fixed number of vectors, select the **N vectors** radio button and enter a new number of vectors into the **N vectors** text field. To reduce the number of vectors by setting the stride, select the **Stride** radio button and enter a new stride value into the **Stride** text field.

3.15 Volume plot

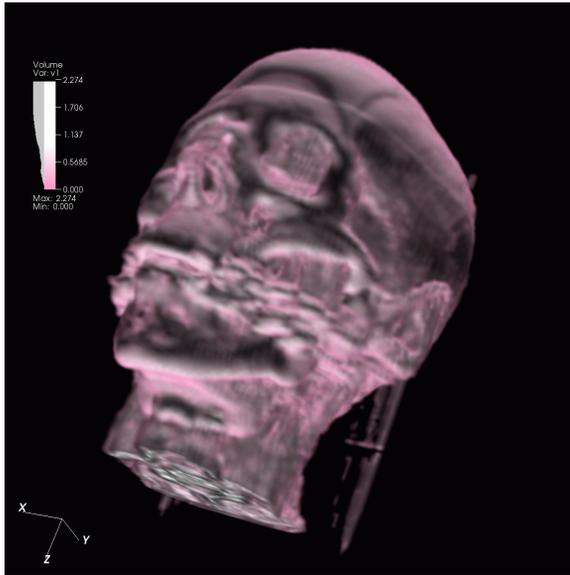


Figure 3-42: Volume plot

This plot, shown in Figure 3-42, uses both color and transparency to visualize 3D scalar fields. The values in the data range have associated color and opacity values that allow parts of the dataset to become partially or completely transparent. This plot captures internal details while keeping the whole dataset at least partially visible.

The Volume plot uses a visualization technique known as volume-rendering, which assigns color and opacity values to a range of data values. The colors and opacities are collectively known as a volume transfer function. The volume transfer function determines the colors of the plot and which parts are visible. The Volume plot uses three types of volume-rendering to visualize data.

The first volume rendering method, hardware-accelerated splatting, resamples the entire database onto a small rectilinear grid and then, at each node in the grid, draws a small textured polygon. The polygon gets its colors and opacity from the transfer function. This method is fast due to its use of graphics hardware but it can require a large number of points in the resampled mesh to look accurate.

Like the first volume rendering method, the second method, hardware-accelerated 3D texturing, resamples the entire database onto a small rectilinear grid. Once the data has been resampled, it is converted into a 3D texture using the Volume plot's volume-transfer function and gets loaded into the video card's texture memory. The Volume plot then draws a set of planes that are perpendicular to the view vector from back to front, with each plane getting the pre-loaded texture mapped onto it. The resulting image is very crisp and captures details not evident when the splatting method is used.

The third volume-rendering technique, called ray-casting, used by the Volume plot is not hardware accelerated. In ray-casting, a ray is followed in reverse from the computer screen into the dataset. As a ray progresses through the dataset, sample points are taken and the sample values are used to determine a color and opacity value for the sample point. Each sample point along the ray is composited to form a final color for the screen pixel. Rays are traced from closest to farthest to allow for early ray termination which stops the sampling process when the pixel opacity gets above a certain threshold. This method of volume-rendering yields superior pictures at the cost of speed and memory use.

The **Volume plot attributes window**, shown in Figure 3-43, is divided into three main areas. The top **Color area** sets the colors that go along with the plot's data values. The **Opacity area** sets the opacity for the plot's data values. The bottom area contains controls which set the level of detail used to draw the plot.

3.15.1 Setting colors

You can design the color component of the volume transfer function using the controls in the top of the **Volume plot attributes window**. The controls are similar to the controls for the **Color Table Window**. There is a color spectrum that has color control points which determine the final look of the color table. Color control points are added and removed using the “+” and “-” buttons. Dragging control points with the mouse moves them and changes their order. Right-clicking on a color control point displays a popup color menu from which a new control point color can be chosen.

3.15.2 Limits

The **Volume plot attributes window** provides controls for setting the limits of the variable being plotted. Limits are artificial minima or maxima that are specified by the user. Setting the limits to a smaller range of values than present in the database cause the plot's colors to be distributed among a smaller range of values, resulting in a plot with more color variety.

You set the limits for the variable being plotted in the **Color area** of the window. To set the limits are set by first clicking the **Min** or **Max** check box next to the **Min** or **Max** text field. Clicking a check box enables a text field into which you can type a new minimum or maximum value.

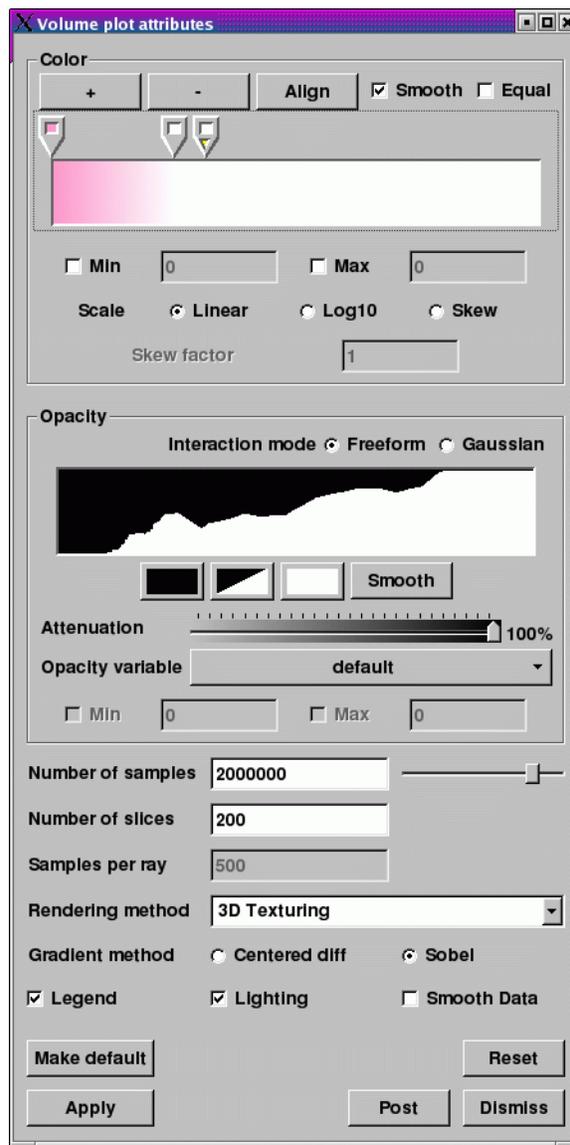


Figure 3-43: Volume plot attributes window

Like VisIt's other plots that map scalar values to colors, the Volume plot allows for the data values to be scaled using Linear, Log, and Skew functions. To select a scaling

function other than linear where values in the data range are mapped 1:1 to values in the color range, click on the **Log10** or **Skew** radio buttons.

3.15.3 Setting opacities

The **Volume plot attributes window** provides several controls that allow you to define the opacity portion of the volume transfer function. The opacity portion of the volume transfer function determines what can be seen in the volume-rendered image. Data values with a lower opacity allow more to be seen and give the plot a gel-like appearance, while data values with higher opacity appear more solid and occlude objects behind them. The controls for setting opacities are located in the center of the window in the **Opacity area**.

You can set opacity two ways. You can hand-draw an opacity map, or create it by designing curves that specify the opacity when they are added together. Both methods use the controls shown in Figure 3-44.

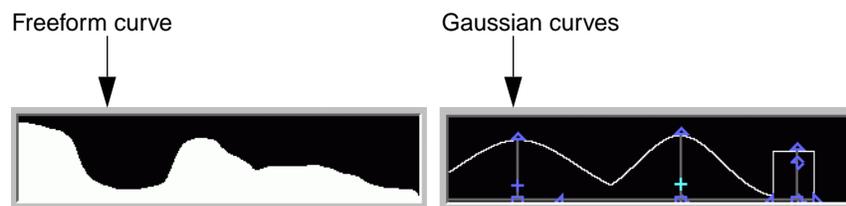


Figure 3-44: Opacity controls

The interaction mode determines how opacity is set. Clicking on the **Freeform** or **Gaussian** radio buttons selects the interaction mode. If the interaction mode switches from **Gaussian** to **Freeform**, the shape constructed by the **Gaussian** controls is copied to the **Freeform** control. Both controls pretend that the plot's data range is positioned horizontally such that the values on the left of the control correspond to the low data values while the values on the right of the control correspond to high data values. The vertical direction corresponds to the opacity for the given data value. Taller curves are more opaque while shorter curves are more transparent.



Figure 3-45: Freeform buttons

To design an opacity map using the **Freeform** control, position the mouse over it and click the left mouse button while moving the mouse. The shape traced by the mouse is entered into the **Freeform** control so you can draw the desired opacity curve. Immediately under the **Freeform** control, there are four buttons, shown in Figure 3-45, which can be used to manipulate the curve. The first three buttons initialize a new curve. The black button makes all data values completely transparent. The ramp button creates a linear ramp of opacity that emphasizes high data values. The white button makes all data values completely opaque. The **Smooth** button smooths out small bumps in the opacity curve that occur when drawing the curve by hand.

The **Gaussian** control used during Gaussian interaction mode is complex but it provides precise control over the shape of a curve.

The basic paradigm followed by the

Gaussian control is that new curves are added and reshaped to yield the desired opacity curve. You add new curves by clicking and dragging in the control. Right clicking with the mouse on an existing curve removes the curve. Each curve has five control points which can change the curve's position and shape. The control points are shown in figure 3-46 along with the shapes that a curve can assume. A control point changes color when it becomes active so there you know which control point is used. Curves start as a smooth Gaussian shape but they can change between the shapes shown in Figure 3-46 by moving the shape control point up and down or left and right. Opacity maps are typically created by adding several curves to the window and altering their shapes and sizes until the desired image is obtained in the visualization window. The **Attenuation slider**, the final control involved in creating an opacity map, controls the opacity of the entire opacity map defined by the **Freeform** or **Gaussian** controls. It provides a knob to scale all opacities without having to modify the opacity map.

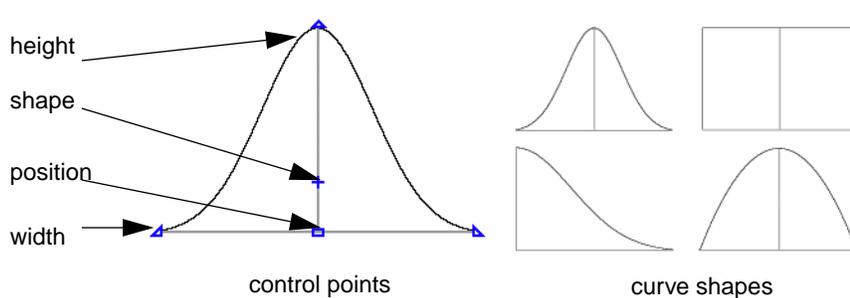


Figure 3-46: Gaussian control points and curve shapes

3.15.4 Changing the opacity variable

The variable used to determine opacity does not have to be the plotted variable. Having a different opacity variable than the plotted variable is useful for instances in which you want to determine the opacity using a variable like density while coloring the plot by another variable such as pressure. To change the opacity variable, select a new variable from the **Opacity variable** variable menu. By default, the plotted variable is used as the opacity variable. This is implied when the **Opacity variable** variable button contains the word *default*. Even when “default” is chosen, it is possible to set artificial data limits on the opacity variable by entering new values into the **Min** or **Max** text fields.

3.15.5 Controlling image quality

When the Volume plot is drawn with graphics hardware, the database is resampled onto a rectilinear grid that is used to place the polygons that are drawn to produce the image. You can control the coarseness of the resampled grid with the **Number of samples** text field and slider. To increase the number of sample points, enter a larger number into the **Number of samples** text field or move the slider to the right. Note that the slider is on an exponential scale and moving it to the right increases the number of sample points exponentially.

In addition to setting the number of samples, when the Volume plot is using the 3D texturing method, you can set the number of planes to be drawn from back to front. Increasing the number of planes can help to reduce the amount of aliasing in the resulting image. However, as the Volume plot uses a higher number of planes, more work must be done to draw the plot and it takes a little longer to draw. To set the number of planes, enter a new number of planes into the **Number of slices** text field.

When the Volume plot is drawn in ray casting mode, the number of samples along each ray that is cast through the data becomes important. Having too few sample points along a ray gives rise to sampling artifacts such as rings or voids. You should adjust this number until you are satisfied with the image. More samples generally produce a better image, though the image will take longer to produce. To change the number of samples per ray, enter a new number of samples per ray into the **Samples per ray** text field.

When using lighting, the gradient calculation method that the Volume plot uses influences the quality of the images that are produced. By default, VisIt uses the Sobel operator, which uses more information from adjacent cells to calculate a gradient. When the Sobel operator is used to calculate the gradient, lighting usually looks better. The alternative gradient calculation method is centered-differences and while it is much less compute intensive than the Sobel operator, it also produces lesser quality gradient vectors, which results in images that are not lit as well. To change the gradient calculation method, click on either the **Centered diff** or **Sobel** radio buttons.

3.15.6 Software rendered images

The Volume plot uses hardware-accelerated graphics by default. While you will want to operate in this mode most of the time, since it's faster, images drawn by software are more accurate. To get a more accurate image, select **Ray casting** from the **Rendering method** combo box. When the Volume plot is set to use ray casting as its rendering mode, VisIt recalculates what the image should look like in software mode. Note that this can be a time-consuming process if the database being used is large or if the visualization window is large. We recommend shrinking the size of the visualization window before changing the rendering method to ray casting to reduce the time and resources required to draw the plot. It is worth noting that if you have a large dataset with intricate details, the software volume rendering method is the best method to use because it scales well in parallel. Using a parallel compute engine can greatly speed up the rate at which software volume rendering operates as long as the dataset is domain-decomposed into equal-sized pieces.

3.15.7 Lighting

The Volume plot can use lighting to enhance the look of the plot. Lighting is enabled by default but you can disable it by unchecking the **Lighting** check box near the bottom of the window. Note that lighting is not currently available when the Volume plot is using the ray casting volume renderer.

Chapter 4

Operators

1.0 Overview

This chapter explains the concept of an operator and goes into detail about each of VisIt's operators.

2.0 Operators

An operator is a filter applied to a database variable before the compute engine uses that variable to generate a plot. VisIt provides several standard operator types that allow various operations to be performed on plot data. The standard operators perform data restriction operations like planar slicing, spherical slicing, and thresholding, as well as more sophisticated operations like peeling off mesh layers. All of VisIt's operators are plugins and you can write your own operator plugins to extend VisIt in new ways. See the *VisIt Plugin Developer's Guide* to find out more information about creating new operator plugins or send an e-mail inquiry to visit-help@llnl.gov.

2.1 Managing operators

When an operator is applied to a plot, it modifies the data that the plot uses to generate a visualization. Any number of operators can be applied to a plot. Each operator added to a plot restricts or modifies the data that is supplied to the plot. By using a series of operators, you can create very sophisticated visualizations.

Since operators are applied to plots and the controls for managing plots are in the **Main Window's Active plots area**, the controls for the operator are found in the same location as the plot controls. The **Plot list**, which displays the list of plots found in the current visualization window, also displays the operators applied to each plot. Each entry

in the **Plot list** displays the database being plotted, the plot type, the variable, and all operators that are applied to the plot. When an operator is applied to a plot, the name of the operator is inserted in front of the plot variable. If multiple operators are applied to a plot, the most recently added operator appears first when reading left to right while the operator that was applied first appears just to the left of the variable name. Plot list entries can also be expanded to allow you to change the order or operators or remove operators from any place in the pipeline.

In addition to containing controls for managing plots, the **Plot list** contains controls that manage operators. Operators can be added, removed, and have their attributes set using controls found in the **Active Plots area**.

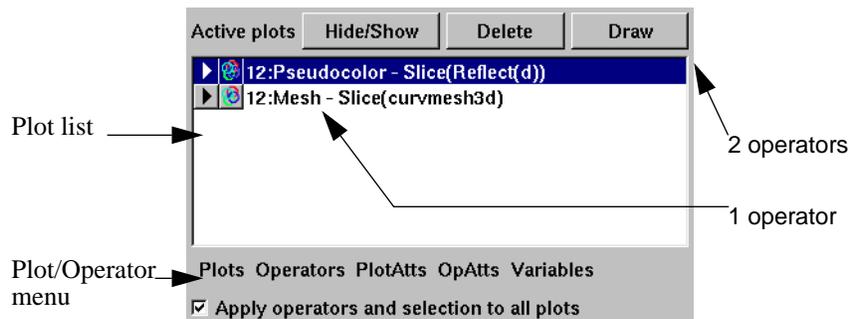


Figure 4-1: Active plots area

2.1.1 Adding an operator

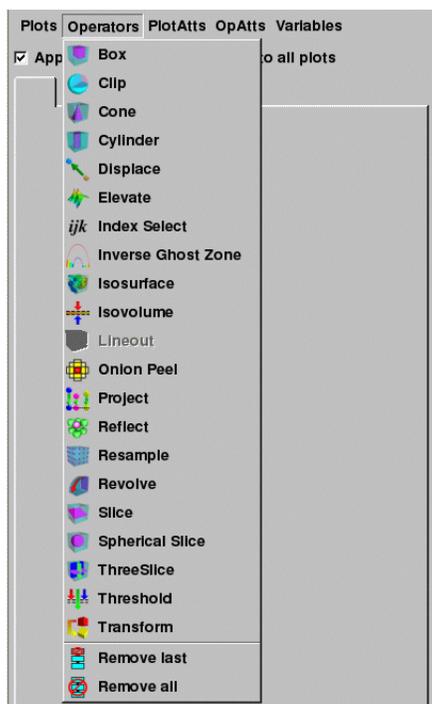


Figure 4-2: Operators menu

You add operators by making a selection from the **Operators** menu, shown in Figure 4-2, which you activate by clicking on the **Operators** option in the **Plots and Operators** menu. If you do not see an operator listed in this chapter in the **Operators** menu then the operator might not be loaded by default. If you want to enable additional operators, use the **Plugin Manager Window** (covered on page 322). When you select an operator, it applies operator to the selected plots in the **Plot list** unless the **Apply operators to all plots** check box is checked, in which case, the selected operator is applied to all plots in the **Plot list**. By default, operators are applied to all plots in the **Plot list**.

When you add an operator to a plot, the name of the operator appears in the plot list entry to the left of the variable or any previously

applied operator. If you apply an operator to an already generated plot, the plot is regenerated immediately with the applied operator. If a plot has not yet been generated (its entry is green), then the operator does not take effect until the plot is generated. This provides time for setting the operator attributes.

It is also possible to apply an operator by clicking an operator attributes window's **Apply** button. If you click the **Apply** button when there is no operator of the specified type has been applied to the selected plot, you can apply the operator and any changes you made by clicking the **Yes** button in a dialog window (see Figure 4-3) that asks whether or not the operator should be applied to the plot.

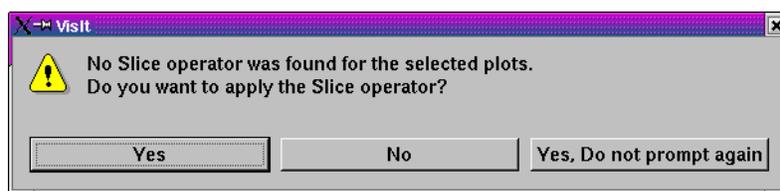


Figure 4-3: Add operator dialog

2.1.2 Expanding plots

Plot list entries are normally collapsed by default so the operators applied to plots are shown in the **Plot list** as a series of nested operators, which finally take a variable as an argument. The **Plot list** allows plot list entries to be expanded on a per-plot basis so you can get to each individual operator that is applied to a plot. To expand a plot list entry, click on its expand button, shown in Figure 4-4. When a plot list entry is expanded, the plot's database, all operators, and finally the plot get their own line in the plot list entry. This is significant because it allows operators to have additional controls to let you reposition them in the pipeline or remove them from the middle of the pipeline without having to first remove other operators.

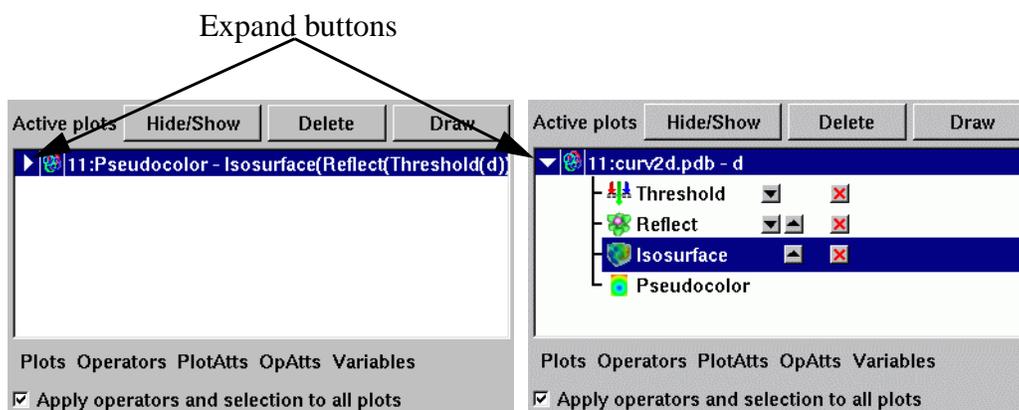


Figure 4-4: Plot list entry before and after being expanded

2.1.3 Changing the order of operators

Sometimes when you've applied several operators, it is useful to change the order of the operators around. For example, you might want to apply a Slice operator before a Reflect operator instead of after it so you can reduce the amount of data that VisIt must process in order to draw your plot. The order in which operators are applied often has a significant impact on the visualization. Using the previous example, suppose a plot is sliced before it is reflected. The resulting visualization is likely to have a reflected slice of the original data. If the order of the operators was reversed so the Reflect operator came first, the Slice operator's slice plane might not intersect the reflected data in the same way, which could result in a totally different looking visualization.

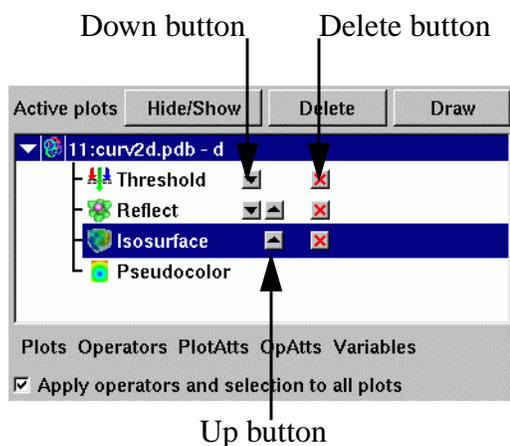


Figure 4-5: Controls for changing operator order

You must expand a plot list entry in order to change the order of its operators. Once the plot list entry is expanded, each operator is listed in the order in which they were applied and each operator has small buttons to the right of its name that allow you to move the operator up or down in the pipeline. To move an operator closer to the database so it is executed before it would have been executed before, click on the **Up** button next to an operator's name. Moving the operator closer to the database in the pipeline is called demoting the operator. If you click the **Down** button next to an operator's name, that operator is moved to a later stage of the pipeline. Moving an operator to a later stage of the pipeline is known as promoting the

operator since the operator appears closer to the plot in the expanded plot entry. Operators in the plot list entry that can only be moved in one direction have only the **Up** button or the **Down** button while operators in the middle of the pipeline have both the **Up** button and the **Down** button.

2.1.4 Removing operators

You do not remove operators by clicking the **Delete** button as you do with plots. There are two ways that you can delete an operator from a plot. As its last two options, the **Operators** menu has options that remove one or more operators. To remove only the last applied operator, select the **Remove last** option from the **Operators** menu. To remove all operators applied to a plot, select the **Remove all** option from the **Operators** menu. Unless the **Apply operator to all plots** check box is checked, operators are only removed from selected plots. If you remove operators using the controls in the **Operators** menu, the plots that are in the completed state are immediately recalculated by the compute engine and redisplayed using the modified list of operators.

The controls in the **Operators** menu allow you to remove operators from the end of a plot's operator list or remove all of a plot's operators. VisIt also provides controls that let you remove specific operators from the middle of a plot's operator list. First expand the plot list entry by clicking its **Expand** button and then click on the red **X** button next to the operator that you want to delete. The red **X** button deletes the operator to which it is attached. When an operator is deleted using the red **X** buttons, the plot is reset back to the new state so you must click the **Draw** button to tell VisIt to regenerate the plot. See Figure 4-6 for an example of deleting an operator from the middle of a plot's operator list.

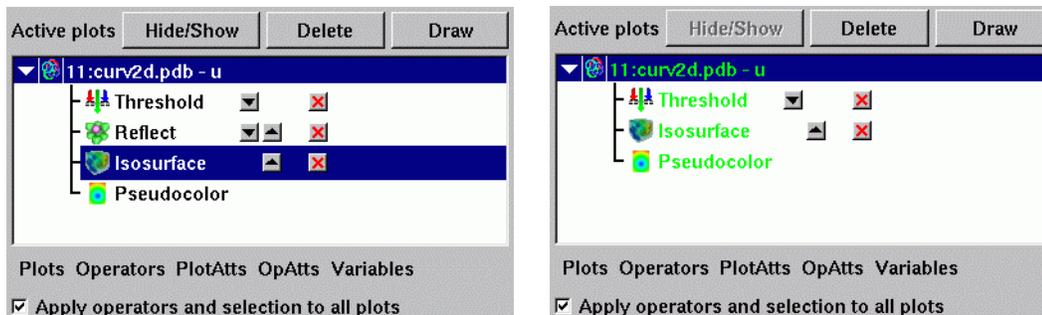


Figure 4-6: Before and after removing an operator from the middle of the pipeline

2.1.5 Setting operator attributes

Each operator type has its own operator attributes window used to set attributes for that operator type. You activate Operator attribute windows by selecting the operator type from the **OpAtts** (Operator attributes) menu shown in Figure 4-7.

When there is only one operator of a given type in a plot's operator list, setting the attributes for that operator type will affect that one operator. When there are multiple instances of the same type of operator in a plot's operator list, only the active operator's attributes are set if the active operator is an operator of the type whose attributes are being set. The active operator is the operator whose attributes are set when using an operator attributes window and can be identified in an expanded plot entry by the highlight that is drawn around it (see Figure 4-8). To set the active operator, expand a plot entry and then click on an operator in the expanded plot entry's operator list.

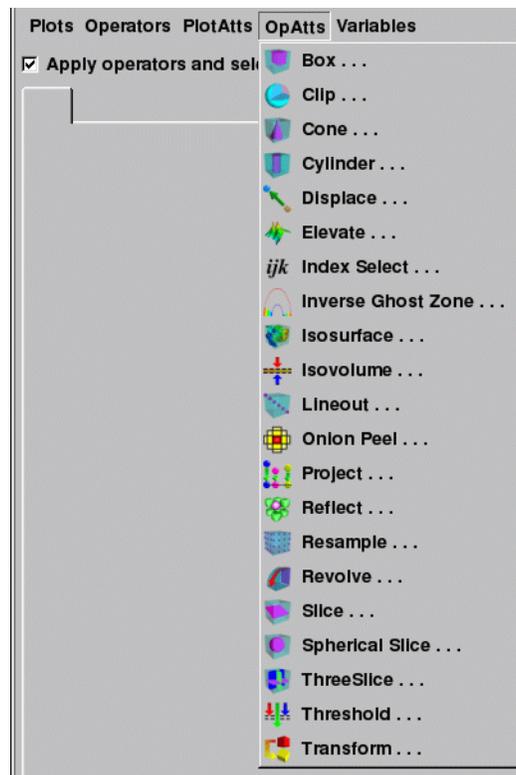


Figure 4-7: Operator attributes menu

Setting the active operator is useful when you have multiple operators of the same type applied to the same plot. For example, you might have applied two Transform operators so you can scale a plot with one operator and then rotate the plot with the second Transform operator. If there was no way to set an active operator, changing the attributes for the Transform operator would cause both instances of the operator to get the same operator attributes. You can make sure the first operator only gets scaling information by making it the active operator. To set the attributes in the second instance of the Transform operator, you can click on that second Transform operator in the expanded plot entry, to make it the active operator, and then set the rotation attributes for that second Transform operator.

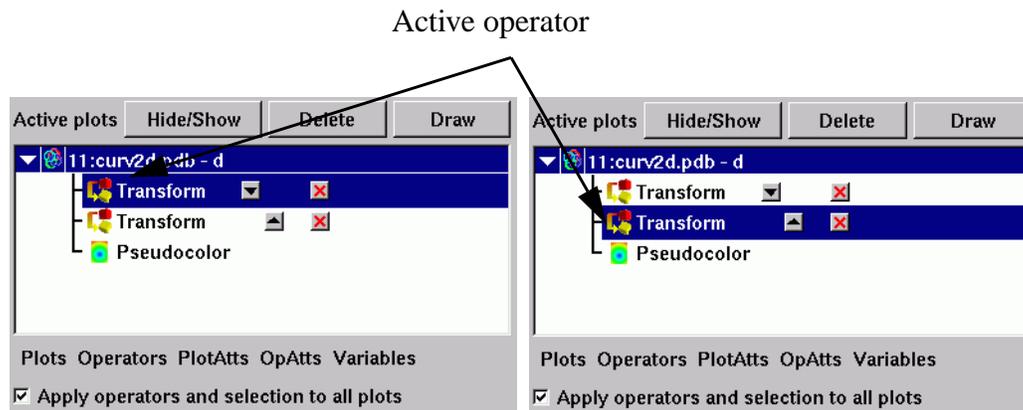


Figure 4-8: Setting the active operator

3.0 Operator Types

VisIt is installed with operator plugins, which perform a wide variety of functions. Some of the operators are not enabled by default so they do not show up in the **Operator** menu. Use the **Plugin Manager Window**, which can be opened by clicking on the **Plugin Manager** option in the **Main Window's Preferences menu**, to enable additional operators or disable operators that you rarely use. This section explains each operator in alphabetical order; not in terms of an operator's importance.

Operator plugins that are enabled by default			
Box	Elevate	Lineout	Slice
Clip	IndexSelect	OnionPeel	SphereSlice
Cone	InverseGhostZone	Project	ThreeSlice
Cylinder	Isosurface	Reflect	Threshold
Displace	Isovolume	Revolve	Transform

3.1 Box operator

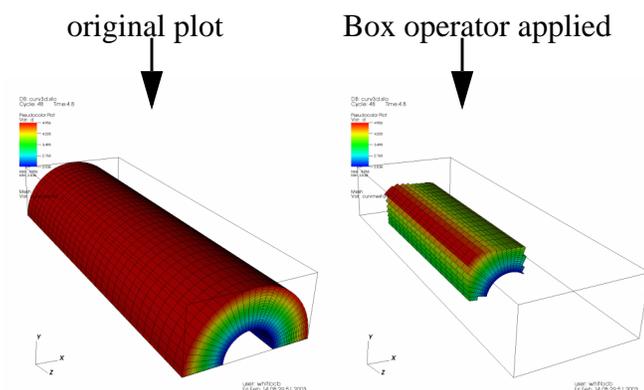


Figure 4-9: Box operator example

The Box operator, which is mostly intended for use with 3D datasets, removes areas of a plot that are either partially or completely outside of the volume defined by an axis-aligned box. The Box operator does not clip cells that straddle the box boundary, it just removes the cells from the visualization leaving jagged edges around the edges of the box where cells were removed.

3.1.1 Setting how cells are removed

The Box operator can either remove cells that are totally outside of the box or it can remove those cells outside of the box and cells that are only partially outside of the box. By default, the Box operator only removes cells that are completely outside of the box. To make the Box operator also remove cells that are partially outside of the box, you click the **All** radio button in the **Box operator attributes window** (shown in Figure 4-10).

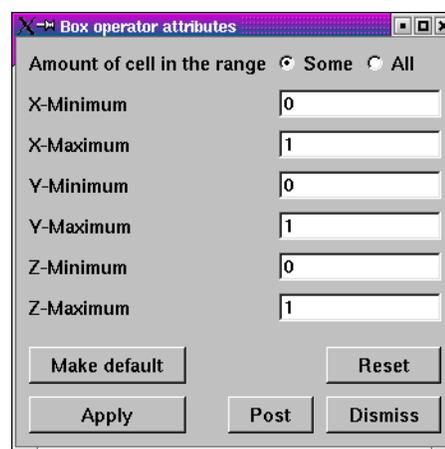


Figure 4-10: Box operator attributes window

3.1.2 Resizing the box

The Box operator uses an axis aligned box to remove cells from the visualization so the box can be specified as a set of minimum and maximum values for X, Y, and Z. To set the size of the box using the **Box operator attributes window**, you type new coordinates into the **X Minimum**, **X Maximum**, **Y Minimum**, **Y Maximum**, **Z Minimum**, or **Z Maximum** text fields.

The Box operator can also be resized interactively with VisIt's Box tool (for more information, see the **Interactive Tools** chapter). If you want to use the Box tool to resize the Box operator's box, first make sure that you've selected the plot that uses the Box operator in the Plot list and then enable the Box tool. When the Box tool appears, it uses the same box as the Box operator. Moving or resizing the Box tool causes the Box operator to also move or be resized and the plots in the visualization window get regenerated with the new box.

3.2 Clip operator

The Clip operator can remove certain shapes from a database before it is plotted. More specifically, the Clip operator can clip away box- or sphere-shaped regions from a database. The database remains in its original dimension after being clipped by the Clip operator and since the Clip operator manipulates the database before it is plotted, the surfaces bounding the removed regions are preserved in the final plot. While being geared primarily towards 3D databases, the Clip operator also clips 2D databases. When applied to 2D databases, the Clip operator can remove rectangular or circular regions from the database. Figure 4-11 shows a Pseudocolor and Mesh plots with a Clip operator.

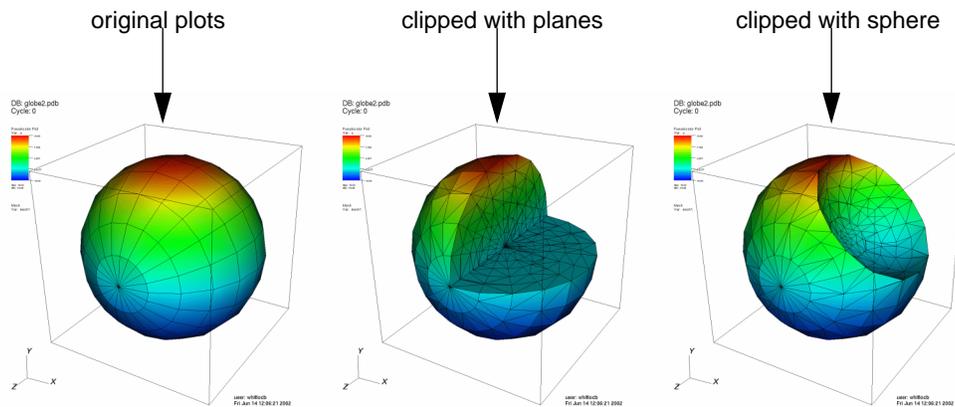


Figure 4-11: Clip operator example

3.2.1 Removing half of a plot

The Clip operator uses up to three planes to define the region that is clipped away. Each plane is specified in origin-normal form where the origin is a point in the plane and the normal is a vector that is perpendicular to the plane. When a plane intersects a plot, it serves as a clipping boundary for the plot. The plane's normal determines which side of the plane is clipped away. The region on the side of the plane pointed to by the normal is the region that the Clip operator clips away. If more than one plane is active, the region that is left as a result of the first clip operation is clipped by the next plane, and so on.

Only one plane needs to be used to remove half of a plot. Find the center of the database by inspecting the 3D axis annotations in the visualization window. Type the center as the new plane origin into the **Origin** text field for plane 1 then click the **Plane 1** check box for plane 1 (see Figure 4-12). When the **Apply** button is clicked, half of the plot should be removed. You can rotate the clipping plane by entering a new normal vector into the **Normal** text field. The normal is specified by three floating point values separated by spaces.

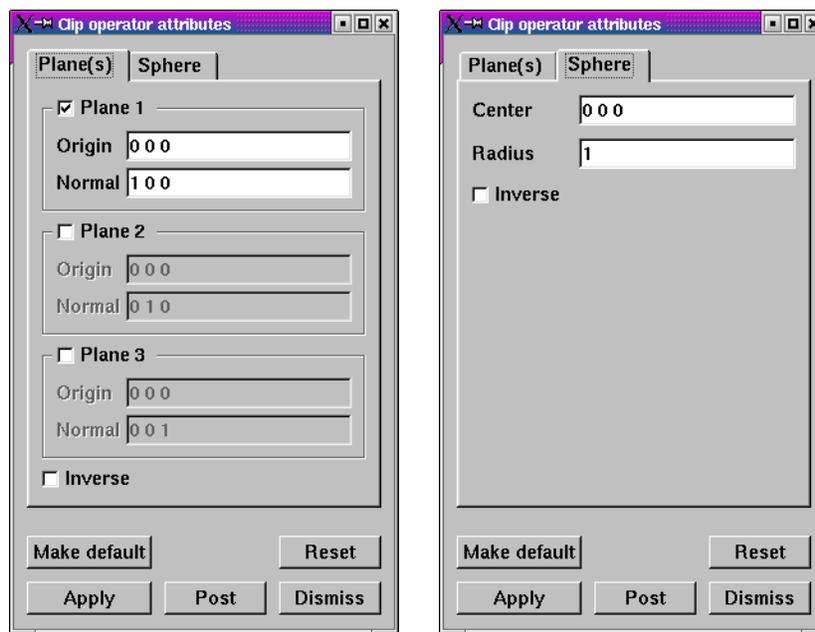


Figure 4-12: Clip operator attributes window

3.2.2 Removing one quarter of a plot

To remove a quarter of a plot, you need two clipping planes. To remove one quarter of the plot, first remove one half of the plot. Now, enable the second clipping plane and make sure that it has the same origin as the first clipping plane but a different normal. To remove exactly one quarter of the plot, make sure that the normal is perpendicular to plane 1's normal. Also make sure that plane 2's new normal points into the region that was clipped away by plane 1. The two planes, when considered together, remove one quarter of the plot. For an illustration of this, see Figure 4-13. In general, the Clip operator removes regions defined by the intersection of the regions removed by each clipping plane. Follow the same procedure with the third clipping plane to remove only one eighth of the plot.

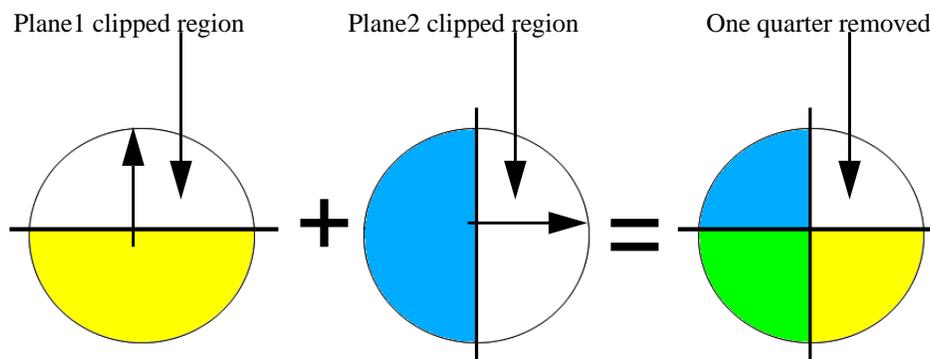


Figure 4-13: Removing one quarter of a plot using two clip planes

3.2.3 Spherical clipping

The Clip operator not only uses sets of planes to clip databases, it can also use a sphere. To make the Clip operator use a clipping sphere, click on the **Sphere** tab. To specify the location and size of the sphere, enter a new center location into the **Center** text field on the Sphere tab of the Clip operator attributes window and then enter a new sphere radius.

3.2.4 Inverting the clipped region

Once the Clip operator has been applied to plots and a region has been clipped away, clicking the **Invert** check box brings back the clipped region and clips away the region that was previously unclipped. Using the **Invert** check box is an easy way to get only the clipped region back so it can be used for other operations.

A common trick when creating animations is to have two identical plots with identical Clip operators applied and then switch one Clip operator to have an inverted clipping region. This will make the plot appear whole. The plot with the inverted clipping region can then be transformed independently of the first plot so it appears to slide out of the first plot. Then it is common to fade out the second plot and zoom in on the first plot's clipped region.

3.3 Cone operator

Like the Slice operator, the Cone operator is also a slice operator. The Cone operator slices a 3D database with a cone, creating a surface that can be left in 3D or be projected to 2D. Plots to which the Cone operator has been applied become surfaces that exist on the surface of the specified cone. The resulting plot can be left in 3D space or it can be projected to 2D space where other operations can be done to it. A Pseudocolor plot to which a Cone operator has been applied is shown in Figure 4-14.

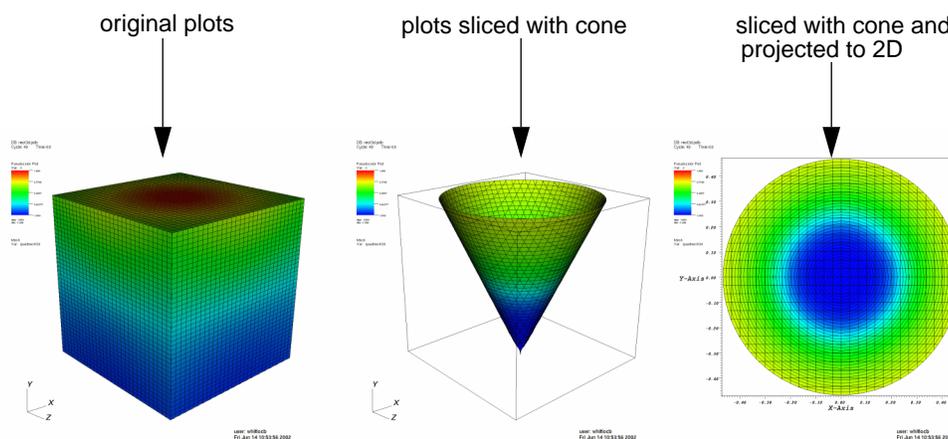


Figure 4-14: Cone operator example

3.3.1 Specifying the slice cone

You can specify the slice cone by setting various fields in the **Cone operator attributes window**, shown in Figure 4-15. To specify how pointy the cone should be, type a new angle (in degrees) into the **Angle** text field. The cone is defined relative to its origin, which is the point at the tip of the cone. To move the cone, type in a new origin vector into the **Origin** text field. The origin is represented by three floating point numbers separated by spaces. Once the cone is positioned, you can set its direction (where the cone points) by entering a new direction vector into the **Direction** text field.

The cone can extend forever or it can be clipped at some distance along its length. To clip the cone at a certain length, check the **Cut cone off** check box and enter a new length value into the **Length** text field.

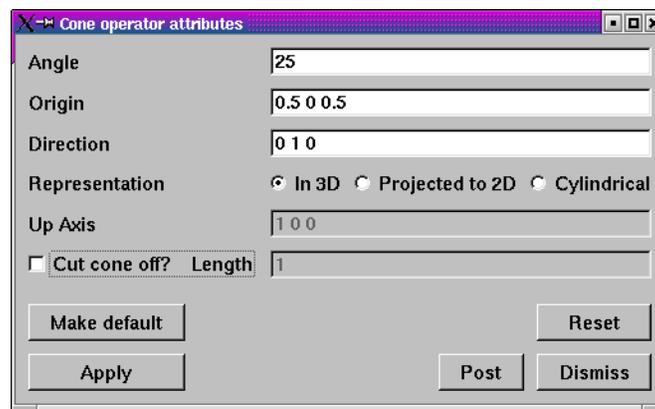


Figure 4-15: Cone operator attributes window

3.3.2 Projecting the slice to 2D

The Cone operator usually flattens sliced plots to 2D along the cone's direction vector. This results in circular 2D plots in the visualization window. The Cone operator can also unfold sliced plots into a cylinder and then into rectangular 2D plots. Alternatively, the Cone operator can leave the sliced plots in 3D space where their cone shape is obvious. To set the cone projection mode, click on one of the following radio buttons: **In 3D**, **Project to 2D**, or **Cylindrical**.

3.4 Cylinder operator

The Cylinder operator, shown in Figure 4-16, clips a database with a cylinder whose size and orientation are specified by the user.

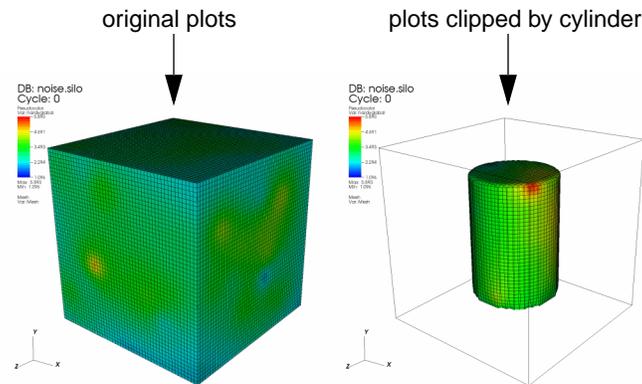


Figure 4-16: Cylinder operator example

3.4.1 Setting the cylinder's endpoints

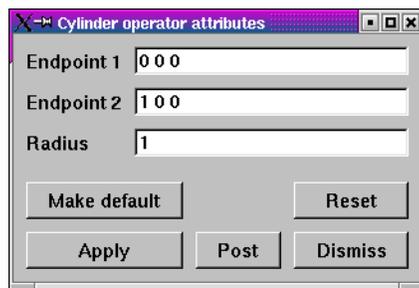


Figure 4-17: Cylinder operator attributes window

There are two ways to set the endpoints for the Cylinder operator. First of all, you can open the **Cylinder operator attributes window** (see Figure 4-17) and type new 3D points into the **Endpoint 1** and **Endpoint 2** text fields. The second, and more interactive way to set the endpoints for the Cylinder operator is to use VisIt's interactive Line tool, which is discussed in the **Interactive Tools** chapter. The Line tool lets you interactively place the Cylinder operator's endpoints anywhere in the visualization. The Line tool's endpoints correspond to the centers of the cylinder's top and bottom circular faces.

3.4.2 Setting the radius

To set the radius used for the Cylinder operator's clipping cylinder, type a new radius into the **Radius** text field in the **Cylinder operator attributes window**.

3.5 Decimate operator

The Decimate operator, shown in Figure 4-18, removes nodes and cells from an input mesh, reducing the cell count while trying to maintain the overall shape of the original mesh. The Decimate operator can currently operate only on the external surfaces of the

input geometry. This means that in order to apply the Decimate operator, you must first apply the ExternalSurfaces operator, which will be covered later in this chapter.

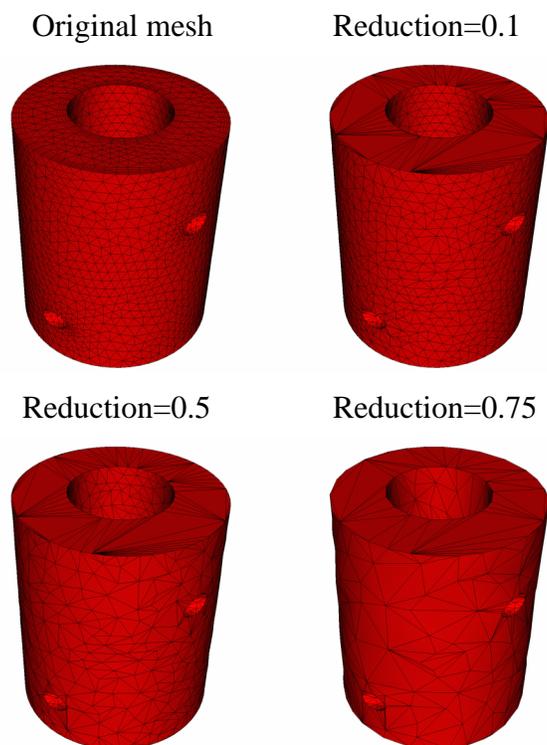


Figure 4-18: Decimate operator applied to reduce the number of cells in the mesh

3.5.1 Using the Decimate operator

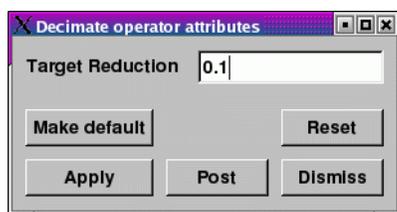


Figure 4-19: Decimate operator attributes window

The Decimate operator simplifies mesh geometry. This can be useful for producing models that have lower polygon counts than the model before the Decimate operator was applied. Models with lower polygon count can be useful for speeding up operations such as rendering. The Decimate operator has a single knob that influences how many cells are removed from the input mesh. The **Target Reduction** value is a floating point number in the range (0,1) and it can be set in the

Decimate operator attributes window (see Figure 4-19). The number specified is the proportion of number of polygonal cells in the output dataset "over" the number of polygonal cells in the original dataset. As shown in Figure 4-18, higher values for **Target Reduction** value cause VisIt to simplify the mesh even more.

3.6 DeferExpression operator

The DeferExpression operator is a special-purpose operator that defers expression execution until later in VisIt's pipeline execution cycle. This means that instead of expression execution taking place before any operators are applied, expression execution can instead take place after operators have been applied.

3.6.1 Plotting surface normals

VisIt can use the DeferExpression operator in conjunction with the ExternalSurface operator and the surface_normal expression to plot surface normals for your plot geometry. To plot surface normals, first create a vector expression using the surface_normal expression, which takes the name of your plot's mesh as an input argument. Once you have done that, you can create a Vector plot of the new expression.

Be sure to apply the

ExternalSurface operator first to convert the plot's 2D cells or 3D cells into polygonal geometry that can be used in the surface_normal expression. Finally, apply the DeferExpression operator and set its variable to your new vector expression. This will ensure that the surface_normal expression is not evaluated until after the ExternalSurface operator has been applied.

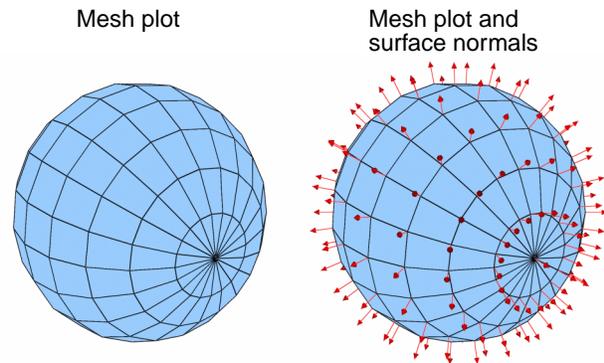


Figure 4-20: DeferExpression operator example

3.7 Displace operator

The Displace operator deforms a mesh variable using a vector field that is defined on the nodes of that mesh. Many engineering simulation codes write a mesh for the first time state of the simulation and then write vector displacements for the mesh for subsequent time states. The Displace operator makes it possible to use the mesh and the time-varying vector field to observe the behavior of the mesh over time. The Displace operator provides a multiplier that can amplify the effects of the vector field on the mesh so slight changes in the vector field can be exaggerated. An

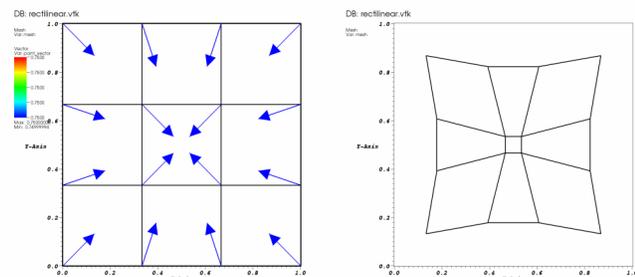


Figure 4-21: Mesh and Vector plots and a Mesh plot that uses the Displace operator to deform the mesh using a vector field.

example showing a mesh and a vector field, along with the results of the mesh displaced by the vector field is shown in Figure 4-21.

3.7.1 Using the Displace operator

The Displace operator takes as inputs a mesh variable and a vector variable and a displacement multiplier value. For each node in the mesh, the Displace operator adds the vector field defined at that node to the node's coordinates. Before adding the vector to the mesh, VisIt multiplies the vector by the displacement multiplier so the effects of the vector field can be exaggerated. To set a new value for the displacement multiplier, type a new value into the **Displacement multiplier** text field in the **Displace operator attributes window** (see Figure 4-22). To set the name of the vector variable that VisIt uses to displace the mesh, select a new vector variable from the **Displacement variable** variable button.

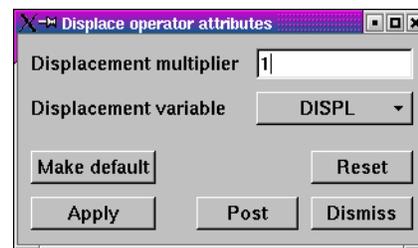


Figure 4-22: Displace operator attributes window

3.8 Elevate operator

The Elevate operator uses a scalar field on a 2D mesh to elevate each node in the input mesh, resulting in a topologically 2D surface in 3D. The Elevate operator allows you to perform much of the same functionality as a Surface plot and it allows you to do additional things like elevate plots that do not accept scalar variables. The Elevate operator can also elevate plots whose input data was produced from higher dimensional data that has been sliced. Furthermore, the Elevate operator allows you to display multiple scalar fields in a single plot such as when a Pseudocolor plot of scalar variable A is elevated by scalar variable B (see Figure 4-23).

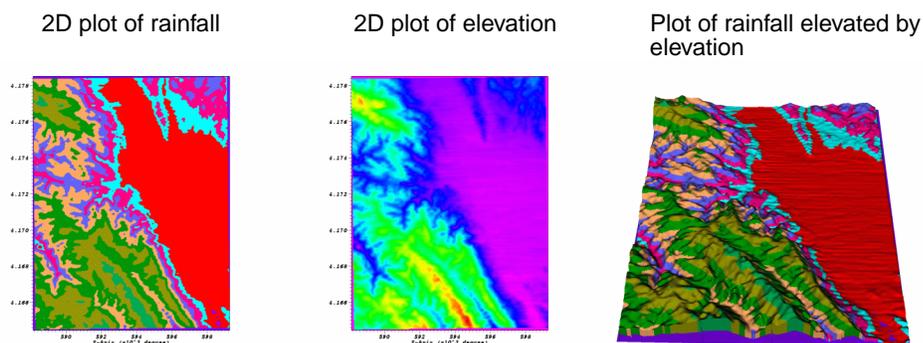


Figure 4-23: Elevate operator example

3.8.1 Using the Elevate operator

The Elevate operator can be used to create plots that look much like a Surface plot if you simply apply the Elevate operator to a plot that accepts scalar values. The Elevate operator is more flexible than a Surface plot because whereas the Surface plot limits you to elevating by one variable and coloring by the same variable, the Elevate operator can be used with any plot and still achieve the Surface plot's elevated effect. You could use the Elevate operator to elevate a Pseudocolor plot of rainfall by elevation. You could also take Vector or FilledBoundary plots (among others) and elevate them by a scalar variable.

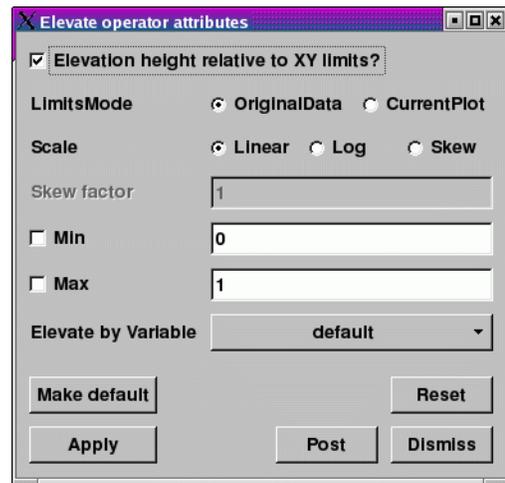


Figure 4-24: Elevate operator attributes window

Since the Elevate operator uses a scalar variable to elevate all of the points in the mesh, the Elevate operator has a number of controls related to scaling scalar data. For example, the Elevate operator allows you to artificially set minimum and maximum values for the scalar variable so you can eliminate data that might otherwise cause your elevated plot to be stretched undesirably in the Z direction. To set minimum and maximum values for the Elevate operator, click on the **Min** or **Max** check boxes in the **Elevate operator attributes window** (see Figure 4-24) and type new values into the adjacent text fields. The options for scaling the plots created using the Elevate operator are the same as those for scaling Surface plots. For more information on scaling, see the Surface plot documentation on page 76.

The most useful feature of the Elevate operator is its ability to elevate plots using an arbitrary scalar variable. By default, the Elevate operator uses the plotted variable in order to elevate the plot's mesh. This only works when the plotted variable is a scalar variable. When you apply the Elevate operator to plots that do not accept scalar variables, the Elevate operator will fail unless you choose a specific scalar variable using the **Elevate by Variable** variable menu in the **Elevate operator attributes window**.

3.8.2 Changing elevation height

The Elevate operator uses a scalar variable's data values as the Z component when converting a mesh's 2D coordinates into 3D coordinates. When the scalar variable's data extents are small relative to the mesh's X and Y extents then you often get what appears to be a flat 2D version of the data floating in 3D space. It is sometimes necessary to scale the scalar variable's data extents relative to the spatial extents in order to produce a visualization where the Z value differs noticeably. If you want to exaggerate the Z values that the scalar variable contributes to make differences more obvious, you can click on the

Elevation height relative to XY limits check box in the **Elevate operator** attributes window.

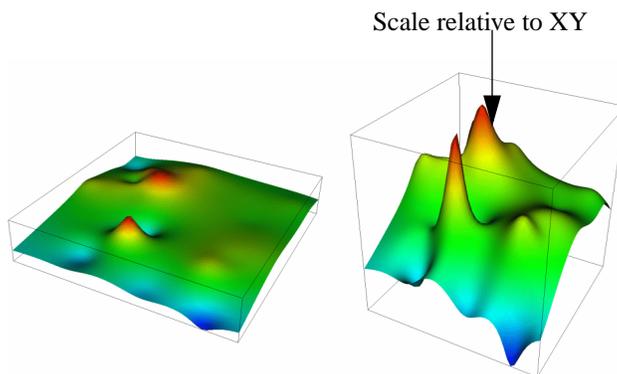


Figure 4-25: Effect of scaling relative to XY limits

3.9 ExternalSurface operator

The ExternalSurface operator takes the input mesh and calculates its external faces and outputs polygonal data. The ExternalSurface operator is not enabled by default but it can be turned on in the **Plugin Manager Window**. The ExternalSurface operator can be useful when creating plots that only involve the external geometry of a plot - such as when you create a Vector plot of surface normals.

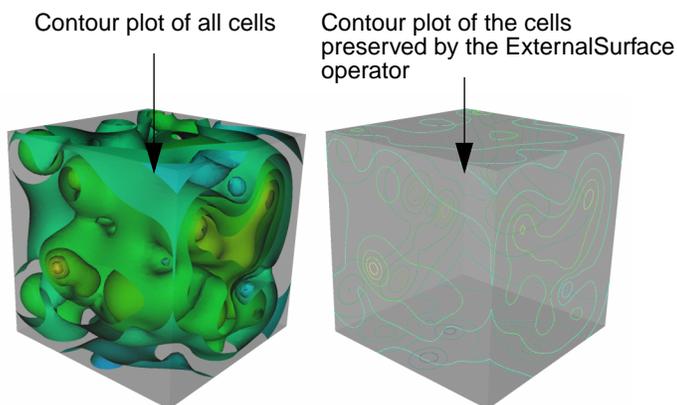


Figure 4-26: ExternalSurface operator example

3.10 Index Select operator

The Index select operator selects a subset of a 2D or 3D structured mesh based on ranges of cell indices. Structured meshes have an implied connectivity that allows each cell in the mesh to be specified by an i,j or i,j,k index depending on the dimension of the mesh. The Index select operator allows you to specify different ranges for each mesh dimension. The

ranges are used to select a brick of cells from the mesh. In addition to indices, the Index select operator uses stride to select cells from the mesh. Stride is a value that allows the operator to count by 2's, 3's, etc. when iterating through the range indices. Stride is set to 1 by default. When higher values are used, the resulting mesh is more coarse since it contains fewer cells in each dimension. The Index select operator attempts to preserve the size of the mesh when non-unity stride values are used. An example of the Index select operator appears in Figure 4-27.

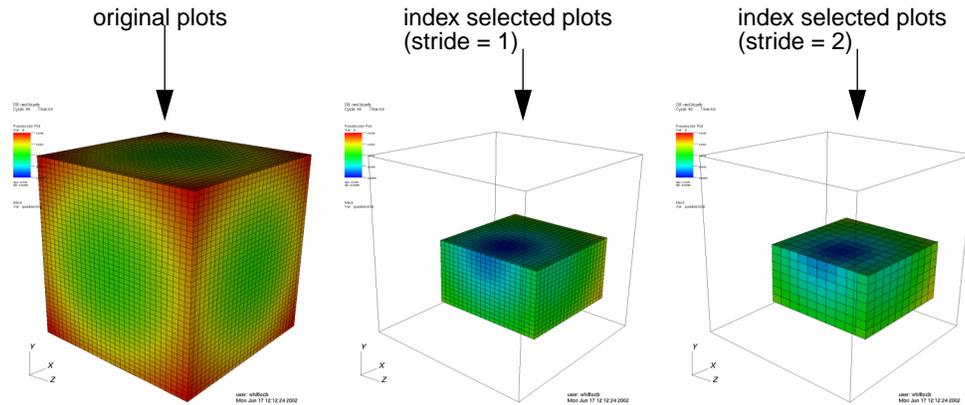


Figure 4-27: Index select operator example

3.10.1 Setting a selection range

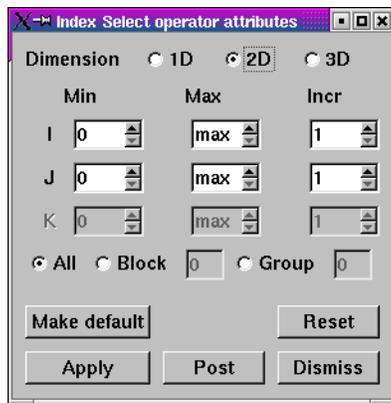


Figure 4-28: Index select operator

The **Index select operator attributes window**, shown in Figure 4-28, contains nine spin boxes that allow you to enter minimum and maximum ranges for i, j, k . To select all cells in the **X** dimension whose index is greater than 10, you would enter 10 into the spin box in the **I** row and **Min** column. Then you would enter max into the spin box in the **Max** column in the **I** row. Finally, you would enter a stride of 1 into the spin box in the **Incr** column in the **I** row. If you wanted to subselect cell ranges for the **Y** dimension, you could follow a similar procedure using the spin boxes in the **J** row and so forth.

To set a range, first select the maximum number of dimensions to which the Index select operator will apply. To set the dimension, click on the **1D**, **2D**, **3D** radio buttons. Note that if the chosen number of dimensions is larger than the number of dimensions in the database, the extra dimension ranges are ignored. It is generally best to select the same number of dimensions as the database. The three range text fields are listed in i, j, k order from top to bottom. To restrict the number of cells in the **X**-dimension, use spin boxes in the **I** row. To restrict the number of cells in the **Y**-dimension, use the spin boxes in the **J** row. To restrict the number of cells in the **Z**-dimension, use the spin boxes in the **K** row.

3.10.2 Block number

Some databases are composed of multiple meshes, often called domains or blocks. The **Index select operator attributes window** calls these submeshes blocks. Often when examining a database, you might want to look at only one block at a time. By default, the Index select operator is applied to all blocks in the database. This means that each index range is applied to each block in the database and will probably result in an image featuring several small chunks of cells. When the Index select operator is set to apply to just one block, the index ranges are relative to the specified block. To make the Index select operator apply to just one block, click on the **Block** radio button and type a new block number into the **Block** text field.

3.10.3 Group number

Some databases are composed of multiple groups of meshes, which are often called groups. The **Index select operator attributes window** calls these groups of meshes groups. The Index select operator can be used to examine parts of a group of meshes by clicking the **Groups** radio button and typing a group number into the **Group** text field.

3.11 InverseGhostZone operator

The InverseGhostZone operator makes ghost cells visible and removes real cells from the dataset so plots to which the InverseGhostZone operator have been applied show only the mesh's ghost cells. Ghost cells are a layer of cells around the mesh that usually correspond to real cells in an adjacent mesh when the whole mesh has been decomposed into smaller domains. Ghost cells are frequently used to ensure continuity between domains for operations like contouring. The InverseGhostZone operator is useful for debugging ghost cell placement in simulation data and for database reader plugins under development.

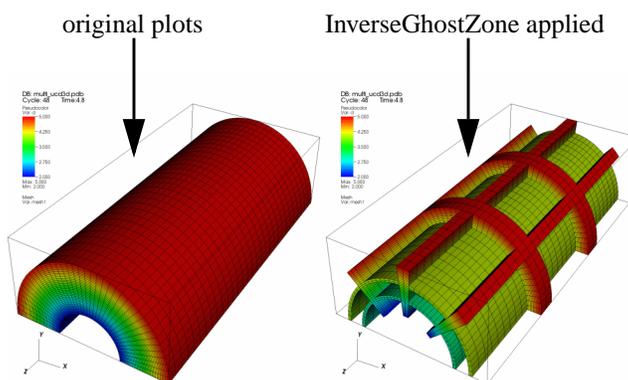


Figure 4-29: InverseGhostZone example

3.11.1 Making all cells visible

The InverseGhostZone operator's only purpose is to make ghost cells visible and real cells are usually stripped out. If you want to ensure that real cells are not removed while still making the ghost cells be enabled, click the **Both ghost zones and real zones** radio button in the **InverseGhostZone operator attributes window** (see Figure 4-30).

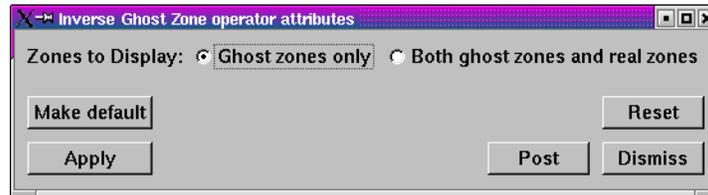


Figure 4-30: InverseGhostZone operator attributes window

3.12 Isosurface operator

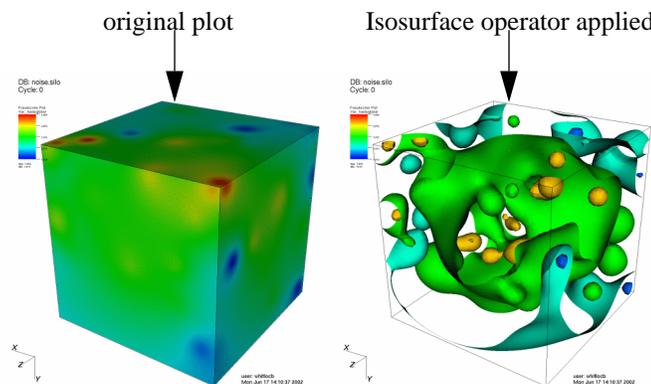


Figure 4-31: Isosurface operator example

The Isosurface operator extracts surfaces from 2D or 3D databases and allows them to be plotted. The Isosurface operator takes as input a database and a list of values and creates a set of isosurfaces through the database. An isosurface is a surface where every point on the surface has the same data value. You can use an isosurface to see a surface through cells that contain a certain value. The Isosurface operator performs essentially the same visualization operation as the

Contour plot, but it allows the resulting data to be used in VisIt's other plots. For example, an Isosurface operator can be applied to a Pseudocolor plot where the Isosurface variable is different from the Pseudocolor variable. In that case, not only are the isosurfaces shown, but they are colored by another variable. An example of the Isosurface operator is shown in Figure 4-31.

3.12.1 Setting isosurface levels

By default, VisIt constructs 10 levels into which the data fall. These levels are linearly interpolated values between the data minimum and data maximum. However, you can set

your own number of levels, specify the levels you want to see or indicate the percentages for the levels.

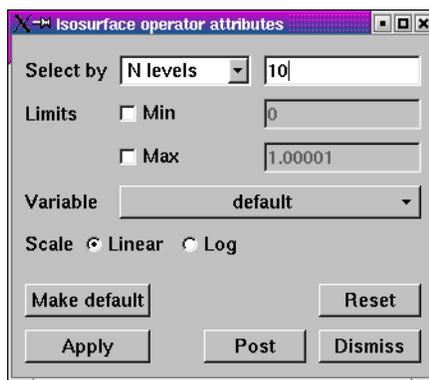


Figure 4-32: Isosurface operator attributes

To choose how levels are specified, make a selection from the **Select by** menu. The available options are: **N levels**, **Levels**, and **Percent**. **N levels**, the default method, allows you to specify the number of levels that will be generated, with 10 being the default. **Levels** requires you to specify real numbers for the levels you want to see. **Percent** takes a list of percentages like 50.5 60 40. Using the numbers just mentioned, the first isosurface would be placed at the value which is 50.5% of the way between the minimum and maximum data values. The next isosurface would be placed at the value that is 60% of the way between the minimum and maximum data values, and so forth. You specify all values for setting the number of isosurfaces by typing into the text field to the right of the **Select by** menu.

3.12.2 Setting Limits

The **Isosurface operator attributes window**, shown in Figure 4-32, provides controls that allow you to specify artificial minima and maxima for the data in the plot. You might set limits when you have a small range of values that you are interested in and you only want the isosurfaces to be generated through that range. To set the minimum value, click the **Min** check box to enable the **Min** text field and then type a new minimum value into the text field. To set the maximum value, click the **Max** check box to enable the **Max** text field and then type a new maximum value into the text field. Note that either the min, max or both can be specified. If neither minimum nor maximum values are specified, VisIt uses the minimum and maximum values in the dataset.

3.12.3 Scaling

The Isosurface operator typically creates isosurfaces through a range of values by linearly interpolating to the next value. You can also change scales so a logarithmic function is used to get the list of isosurface values through the specified range. To change the scale, click either the **Linear** or **Log** radio buttons in the **Isosurface operator attributes window**.

3.12.4 Setting the isosurfacing variable

The Isosurface operator database variable can differ from the plotted variable. This enables plots to combine information from two variables by having isosurfaces of one variable and then coloring the resulting surfaces by another variable. You can change the isosurfacing variable, by selecting a new variable name from the **Variable** variable button.

Sometimes it is useful to set the isosurfacing variable when the plotted variable is not a scalar. For example, you might want to apply the Isosurface operator to a Mesh plot but the Mesh plot's plotted variable is not a scalar so the Isosurface operator does not know what to do. To avoid this situation, you can set the isosurfacing variable to one you know to be scalar and the operator will succeed.

3.13 Isovolume operator

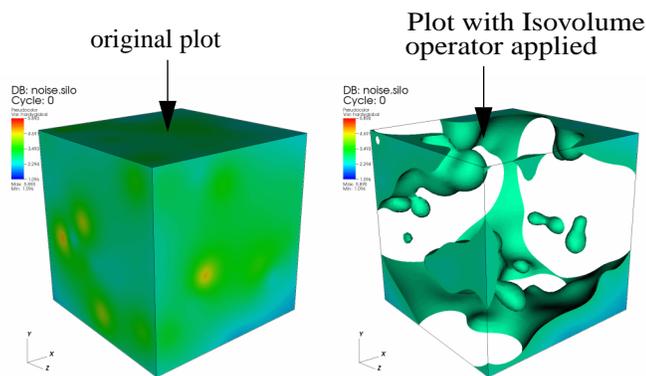


Figure 4-33: The effects of the Isovolume operator on a Pseudocolor plot

The Isovolume operator creates a new unstructured mesh using only cells and parts of cells from the original mesh that are within the specified data range for a variable. The resulting mesh can be used in other VisIt plots. You might use this operator when searching for cells that have certain values. The Isovolume operator can either use the plotted variable or a variable other than the plotted variable. For instance, you might want to see a Pseudocolor plot of pressure

while using the Isovolume operator to remove all cells and parts of cells below a certain density. An example of a plot to which an Isovolume operator has been applied is shown in Figure 4-33.

3.13.1 Using the Isovolume operator

The Isovolume operator iterates over every cell in a mesh and determines which parts of the cell, if any, contain a value that falls within a specified data range. If any parts of the cell are within the specified data range, they are kept as part of the operator's output. The Isovolume operator uses an isosurfacing algorithm to determine the interfaces where cells should be split so the interfaces for neighboring cells are all continuous and fairly smooth. To specify a data range, type new upper and lower bounds into the **Lower bound** and

Upper bound text fields in the **Isovolume operator attributes window**, which is shown in Figure 4-34.

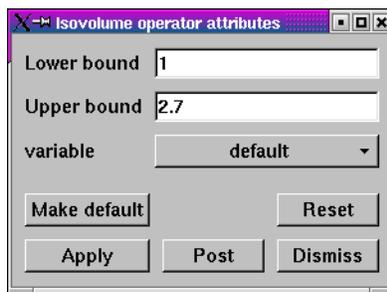


Figure 4-34: Isovolume operator attributes window

The variable that the Isovolume operator uses does not necessarily have to match the plotted variable. If the plotted variable is to be used, the **Variable** text field must contain the word: *default*. If you want to make the Isovolume operator use a different variable so you can, for example, plot temperature but only look at regions that have a density greater than 2g/mL, you can set the Isovolume’s variable to temperature. To make the Isovolume operator use a different variable, select a new variable from the **Variable** variable button in the **Isovolume operator attributes window**.

If you apply this operator to a plot that does not operator on scalar variables such as the Mesh or Subset plots, be sure to set the variable because the default variables for those plots is never a scalar variable. Without a scalar variable, the Isovolume operator will not work.

3.14 Lineout operator

The Lineout operator samples data values along a line, producing a 1D database from databases of greater dimension. This operator is used implicitly by VisIt’s Lineout capability and cannot be added to plots. For more information on Lineout, see the **Quantitative Analysis** chapter.

3.15 Merge operator

VisIt’s Merge operator merges all geometry that may exist on separate processors into a single geometry dataset on a single processor. The Merge operator can be useful when applying other operators like the Decimate operator or when creating Streamline plots. The Merge operator is not enabled by default.

3.16 OnionPeel operator

The OnionPeel operator creates a new unstructured mesh by taking a seed cell or node from a mesh and progressively adds more layers made up of the initial cell’s neighboring

cells. The resulting mesh is then plotted using any of VisIt's standard plots. The OnionPeel operator is often useful for debugging problems with scientific simulation codes, which often indicate error conditions for certain cells in the simulated model. Armed with the cell number that caused the simulation to develop problems, the user can visualize the simulation output in VisIt and examine the bad cell using the OnionPeel operator. The OnionPeel operator takes a cell index or a node index as a seed from which to start growing layers. Only the seed is shown initially but as you increase the number of layers, more of the cells around the seed are added to the visualization. An example of the OnionPeel operator is shown in Figure 4-35.

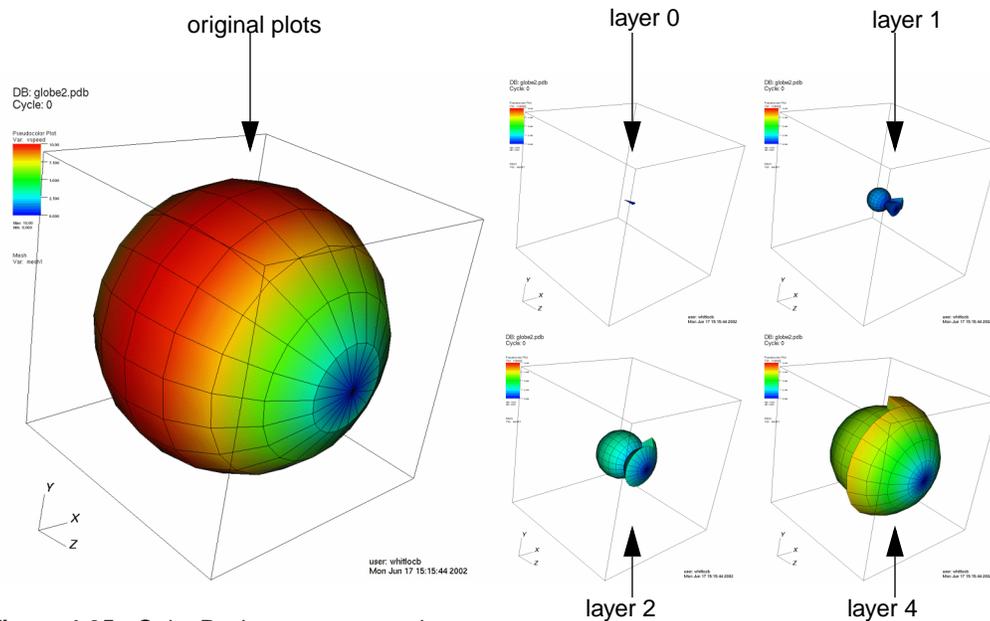


Figure 4-35: OnionPeel operator example

3.16.1 Setting the seed

The OnionPeel operator uses a seed cell or a seed node as the seed to which all cells from other layers are added. When a layer is added around the seed, the new cells are those immediately connected to the seed. You specify the seed as a cell index or a node index by typing a new seed value into the **Seed# or ij[k]** text field. VisIt interprets the seed as a cell index by default. If you want to start growing cell layers around a given node, click on the **Node** radio button before entering a new seed value. The form of the seed index depends on how the underlying mesh is organized. Unstructured meshes, which are a collection of independent cells, require only a single integer value for the seed while structured meshes are indexed with i,j or i,j,k indices depending on the dimension of the mesh. To set the seed using i,j,k indices, type the i and j and k indices, separated by spaces, into the **Seed# or ij[k]** text field.

Some meshes that have been decomposed into multiple smaller meshes known blocks or domains have an auxiliary set of cell indices and node indices that allow cells and nodes from any of the domains to be addressed as though each domain was part of a single,

larger whole. If you have such a mesh and want to specify seed indices in terms of global cell indices or global node indices, be sure to turn on the **Seed# is Global** check box.

The OnionPeel operator can only operate on one domain at a time and when the operator grows layers, they do not cross domain boundaries. The seed cell index is always relative to the active domain. To make a cell in a different domain the new seed cell, change the domain number by selecting a new domain from the **Set** drop down list.

3.16.2 Growing layers

The OnionPeel operator starts with a seed and adds layers of new cells around that seed. The added cells are determined by the layer number and the adjacency information. The cell adjacency rule determines the connectivity between cells. Cells are next to each other if they share a cell face or a cell node. The visualization will differ slightly depending on which adjacency rule is used. To change the adjacency rule, click the **Node** or the **Face** radio buttons in the **OnionPeel operator attributes window**, shown in Figure 4-36

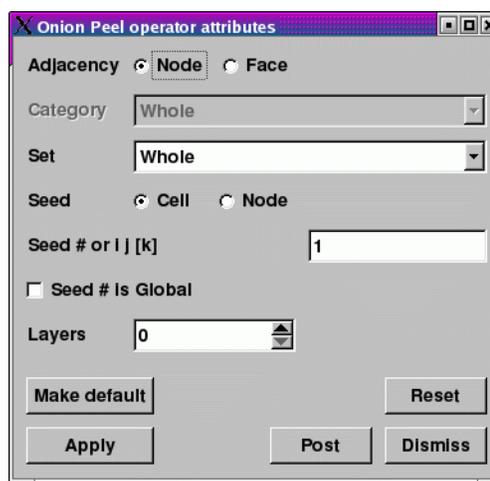


Figure 4-36: OnionPeel operator attributes window

The OnionPeel operator initially shows zero layers out from the seed, so only the seed is shown in the visualization when the OnionPeel operator is first applied. Consequently, the visualization might appear to be empty since some seed cells are very small. To add more layers around the seed, enter a larger layer number into the **Layer Number** text field. Clicking the up or down buttons next to the **Layer Number** text field also increments or decrements the layer number.

3.17 Project operator

The Project operator sets all of the Z values in the coordinates of a 3D mesh to zero and reduces the topological dimension of the mesh by 1. The Project operator is, in essence, an

operator to make 2D meshes out of 3D meshes. An example of the Project operator is shown in Figure 4-37.

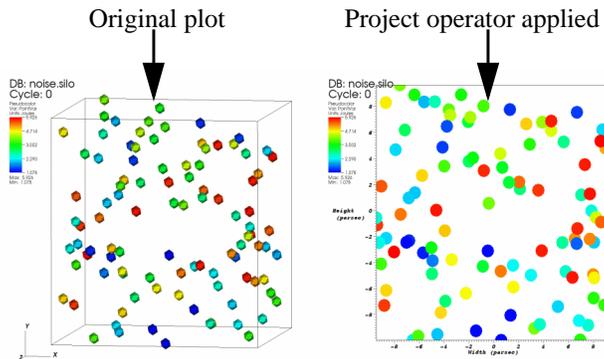


Figure 4-37: Project operator example

3.17.1 Setting the projection type

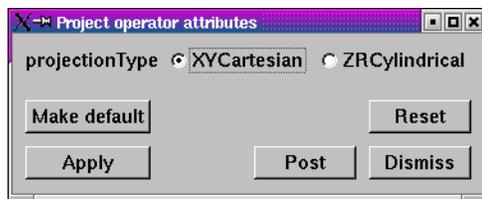


Figure 4-38: Project operator attributes window

The Project operator can project 3D down to 2D using two different transforms: XY Cartesian, and ZR Cylindrical. To specify which of these transforms you want to use when using the Project operator, click on either the **XYCartesian** or **ZRCylindrical** radio buttons in the **Project operator attributes window** (see Figure 4-38).

3.18 Reflect operator

Use the Reflect operator to reflect database geometry across one or more axes. Scientific simulations often rely on symmetry so they only need to simulate part of the problem. When creating a visualization, most users want to see the entire object that was simulated. This often involves reflecting the database geometry to create the full geometry of the simulated object. VisIt's reflect operator can be applied to both 2D and 3D databases and

can reflect them across one or more plot axes. An example of the Reflect operator is shown in Figure 4-39.

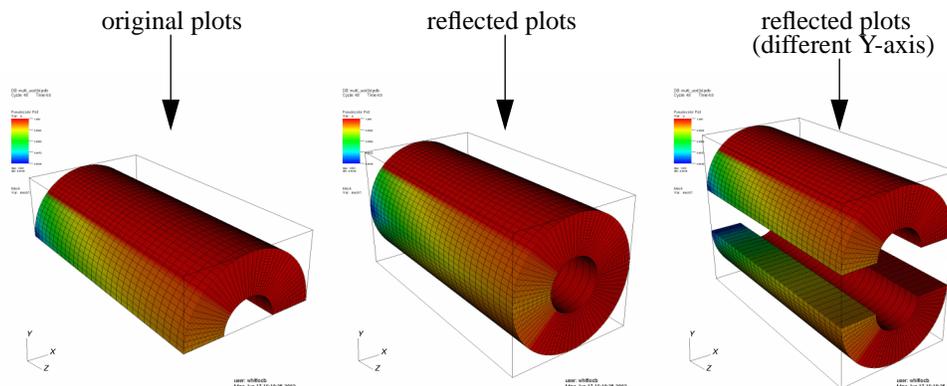


Figure 4-39: Reflect operator example

3.18.1 Setting the Reflect operator attribute window's input mode

The **Reflect operator attributes** window, shown in Figure 4-40, has two input modes.

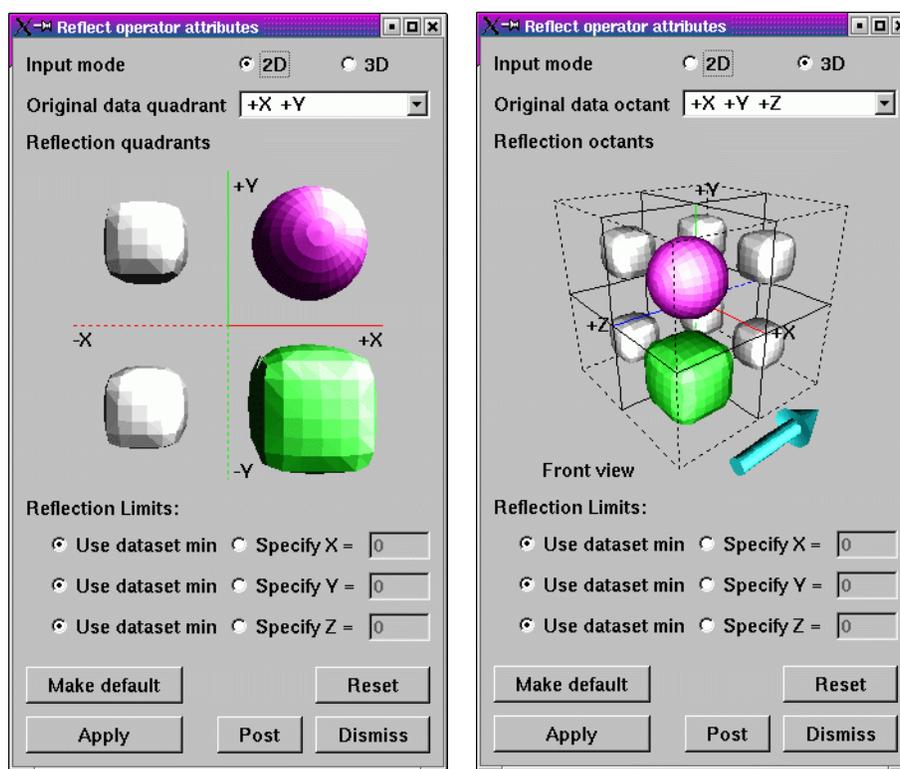


Figure 4-40: Reflect operator attributes window

One input mode is for 2D data, in which only reflection quadrants are shown, and the second input mode is for 3D data for which the window shows 3D octants. In either input mode, clicking on the brightly colored shapes turns on different reflections and in the 3D

input mode, clicking on the cyan arrow rotates the view so you can more easily get to reflections in the back. To set the input mode, click either the **2D** or **3D** radio buttons.

3.18.2 Setting the data octant

The Reflect operator assumes that the database being reflected resides in the +X+Y+Z octant when performing its reflections. Sometimes, due to the orientation of the database geometry, it is convenient to assume the geometry exists in another octant. To change the data octant, make a new selection from the **Original data is in octant** menu in the **Reflect operator attributes window**. The **Reflection operator attributes window** graphically depicts the original data octant as the octant that contains a sphere instead of a cube, which correspond only to reflections.

3.18.3 Reflecting plots

Once the Reflect operator has been applied to plots, you must usually specify the direction in which the plots should be reflected. To set the plot direction, click on the check boxes below the **Original data is in octant menu**. The possible reflections are shown by the cube and sphere glyphs. When a reflection is set to be on, the glyph in the octant or quadrant will be green or magenta. When a reflection is not on, its glyph is smaller and silver. To turn a reflection on or off, just click on its glyph. If the window is in its 3D input mode and you need to access octants in the back that are obscured by other octants, clicking on the cyan arrow will rotate the glyphs to the octants in the back will be more accessible.

3.18.4 Reflection limits

Reflection limits determine the axes about which the database geometry is reflected. The Reflect operator attributes window has three reflection limits controls; one for each dimension. You will usually want to reflect plots using the dataset min value, which you set by clicking the **Use dataset min** radio button. When using the dataset min value to reflect plots, the reflected plots will touch along the reflected edge. You can also specify another axis of reflection. When using a custom axis of reflection, the reflected plots will not necessarily touch. This option, though not normally needed, can produce interesting effects in animations. To specify a custom axis of reflection, click the **Specify X**, **Specify Y**, or **Specify Z** radio buttons and enter a new X, Y, or Z value into the appropriate text field.

3.19 Revolve operator

The Revolve operator is for creating 3D geometry from 2D geometry by revolving the 2D about an axis. The Revolve operator is useful for incorporating 2D simulation data into a

visualization along with existing 3D data. An example of the Revolve operator is shown in Figure 4-41.

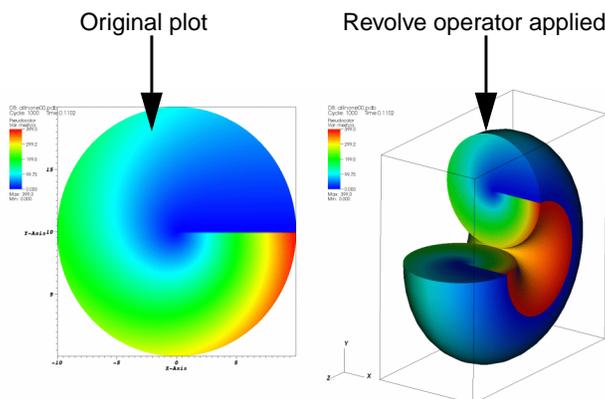


Figure 4-41: Revolve operator example

3.19.1 Using the Revolve operator

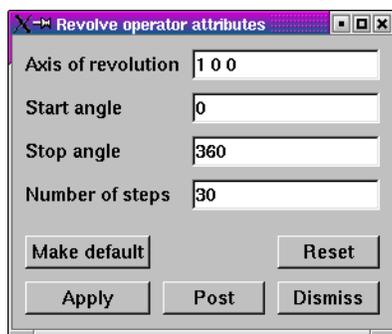


Figure 4-42: Revolve operator attributes window

To use the Revolve operator, the first thing to do is pick an axis of revolution. The axis of revolution is specified as a 3D vector in the **Axis of revolution** text field (see Figure 4-42) and serves as the axis about which your 2D geometry is revolved. If you want to revolve 2D geometry into 3D geometry without any holes in the middle, be sure to pick an axis of revolution that is incident with an edge of your 2D geometry. If you want 3D geometry where the initial 2D faces do not meet, be sure to specify start and stop angles in degrees in the **Start angle** and **Stop angle** text fields. Finally, the number of steps determines how many times the initial 2D geometry is

revolved along the way from the start angle to the stop angle. You can specify the number of steps by entering a new value into the **Number of steps** text field.

3.20 Resample operator

The Resample operator extracts data from any input dataset in a uniform fashion, forming a new 2D or 3D rectilinear grid onto which the original dataset has been mapped. The Resample operator is useful in a variety of contexts such as downsampling a high

resolution dataset (shown in Figure 4-43), rendering Constructive Solid Geometry (CSG) meshes, or mapping multiple datasets into a common grid for comparison purposes.

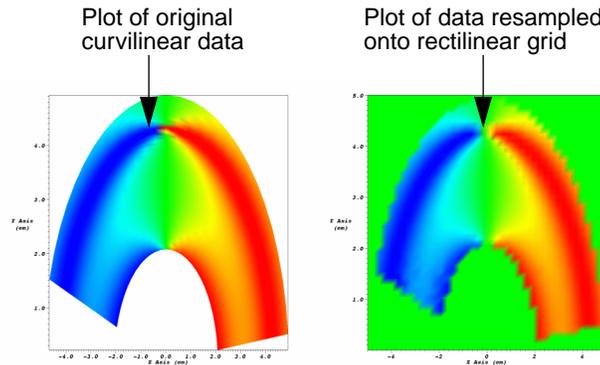


Figure 4-43: Resample operator example

3.20.1 Resampling onto a rectilinear grid

Resampling a high resolution dataset onto a rectilinear grid is the most common use case for the Resample operator. When a Resample operator is applied to a plot, the Resample operator clips out any data values that are not within the operator's bounding box. For the data that remains inside the bounding box, the operator samples it using the user-specified numbers of samples for the X, Y, and Z dimensions. The bounding box is specified by entering new start and end values for each dimension. For example, if you want to change the locations sampled in the X dimension then you could type new floating point values into the **Resample operator attributes window's Start X** and **End X** text fields. The same pattern applies to changing the locations sampled in the Y and Z dimensions. One difference between resampling 2D and 3D datasets is that 3D datasets must have the **3D resampling** check box enabled to ensure that VisIt uses the user-specified Z-extents and number of samples in Z.

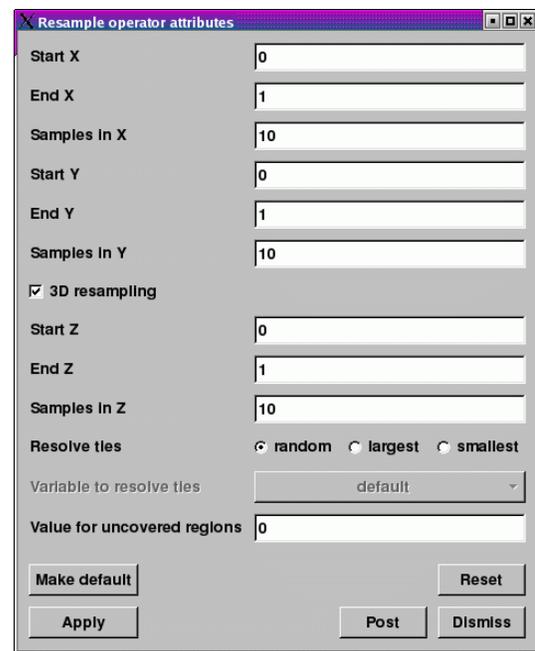


Figure 4-44: Resample operator attributes window

Samples for which there was no data in the original input dataset are provided with a default value that you can change by typing a new floating point number into the **Value for uncovered regions** text field.

3.20.2 Using Resample with CSG meshes

Constructive Solid Geometry (CSG) modeling is a method whereby complex models are built by adding and subtracting primitive objects such as spheres, cubes, cones, etc. When you plot a CSG mesh in VisIt, VisIt resamples the CSG mesh into discrete cells that can be processed as an unstructured mesh and plotted. The Resample operator can be used to tell VisIt the granularity at which the CSG mesh should be sampled, overriding the CSG mesh's default sampling. Naturally, higher numbers of samples in the Resample operator produce a more faithful representation of the original CSG mesh. Figure 4-45 depicts a CSG model that contains a disc within a smooth ring. Note that as the number of samples in the Resample operator increases, the model becomes smoother and jagged edges start to disappear.

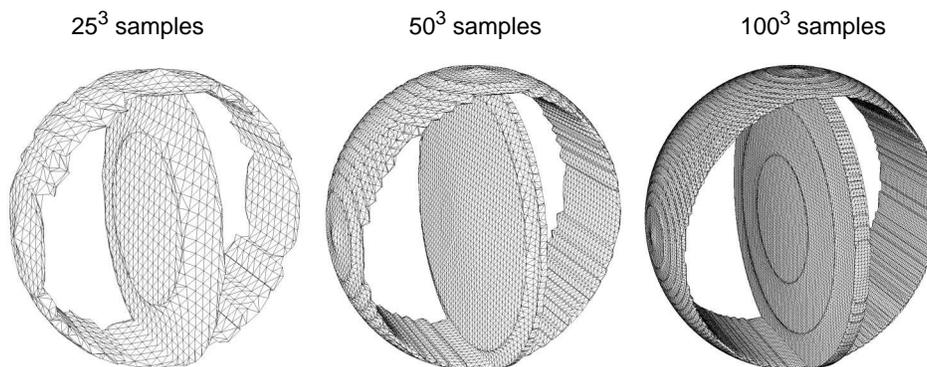


Figure 4-45: The Resample operator can be used to control the resolution of CSG meshes

3.20.3 Resampling surfaces projected to 2D

Sometimes it is useful to project complex surfaces into 2D and resample them onto a 2D mesh so queries and other analysis can be performed.

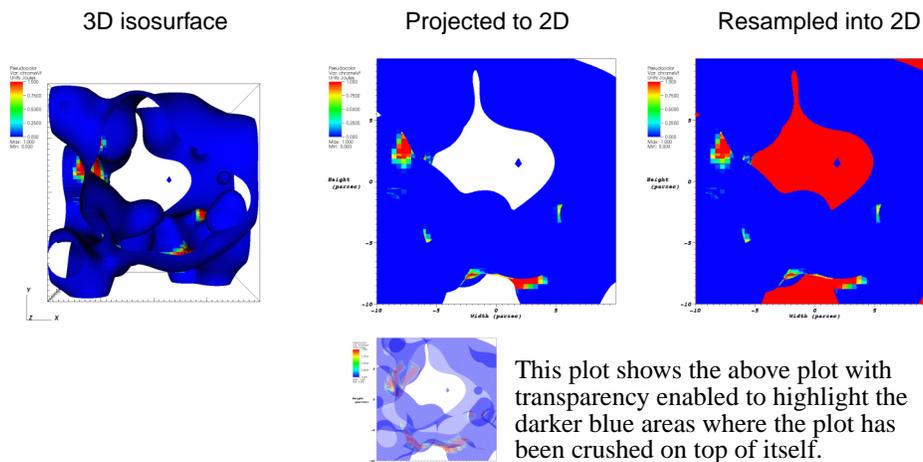


Figure 4-46: Using the Resample operator to create a 2D projection

When you project a complex surface to 2D using the Project operator, all of a plot's geometry remains and its Z coordinates are set to zero. This results in some areas where the plot is essentially crushed on top of itself, as shown in Figure 4-46. When resampling the plot onto a new 2D grid, these overlapping areas can be treated in three different ways. You can ensure that the top value is taken if you choose the random option by clicking on the **random** button in the **Resolve ties** button group. You can use a mask variable to decide ties by clicking on the **largest** or **smallest** buttons and by selecting an appropriate variable using the **Variable to resolve ties** menu.

3.21 Slice operator

This operator slices a 3D database with a plane that can have an arbitrary orientation. Plots to which the Slice operator has been applied are turned into 2D planar surfaces that are coplanar with the slice plane. The resulting plot can be left as a 2D slice in 3D space or it can be projected to 2D space where other operations can be done to it. A Pseudocolor plot to which a Slice operator has been applied is shown in Figure 4-47.

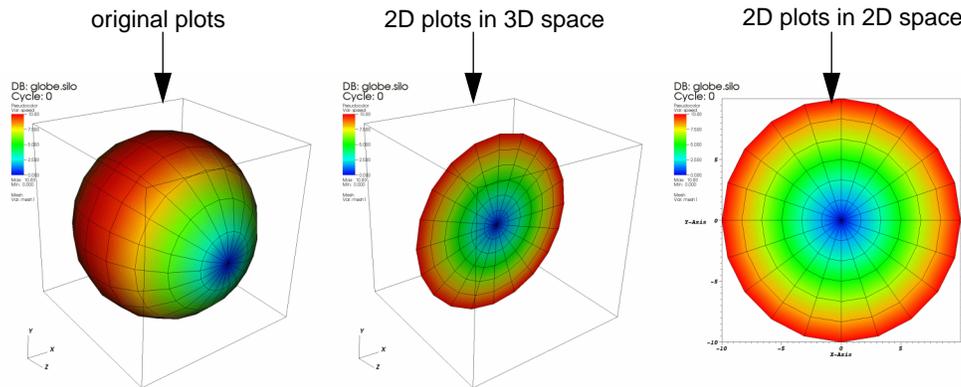


Figure 4-47: Slice operator example

3.21.1 Positioning the slice plane

You can position the slice plane by setting the origin, normal, and up-axis vectors in the **Slice operator attributes window**, shown in Figure 4-48. The slice plane is specified using the origin-normal form of a plane where all that is needed to specify the plane are two vectors; the origin and the normal. The origin of the plane is a point in the slice plane. The normal vector is a vector that is perpendicular to the slice plane.

VisIt allows the slice plane normal to be aligned to a specific axis or it can be set to any arbitrary vector. If you want the slice plane to be along any of the three axes, click the **X-Axis**, **Y-Axis**, or **Z-Axis** radio button. If you want to make a slice plane that does not align with the principle axes, click the **Arbitrary** radio button and then type a direction vector into the text field to the right of the **Arbitrary** radio button. The vector need not be normalized since VisIt will normalize the vector before using it.

The slice plane's origin, which specifies the location of the slice plane, can be set five different ways. The middle of the **Slice operator attributes window**, or **Origin area** (see Figure 4-49), provides the necessary controls required to set the slice plane origin. The **Origin area** provides five radio buttons: **Point**, **Intercept**, **Percent**, **Zone**, and **Node**. Clicking on one of these radio buttons causes the **Origin area** to display the appropriate controls for setting the slice plane origin.

To set the slice plane origin to a specific point, click the **Point** radio button in the **Origin area** and then type a new 3D point into the **Point** text field. To set the slice plane origin to a specific value along the principle slice axis (usually an orthogonal slice), click the **Intercept** radio button and then type a new value into the **Intercept** text field.

If you don't know a good value to use for the intercept, consider using the percent slice mode. Percent slice mode, which is most often used for an orthogonal slice, allows you to slice along a particular axis using some percentage of the distance along that axis. For example, this allows you to see what the slice plane looks like if its origin is 50% of the distance along the X-Axis. To set the origin using a percentage of the distance along an axis, click the **Percent** radio button and then type a new percentage value into the **Percent** text field or use the **Percent** slider.

Sometimes it is useful to slice through a particular zone or node. The Slice operator allows you to pick an origin for the slice plane so a specific zone or node lies in the slice plane. To make sure that a particular zone is sliced by the Slice operator, click on the **Zone** radio button and then enter the zone to be sliced into the **Zone** text field. Be sure to also enter the domain that contains the zone into the **Domain** text field if you are slicing a multi-domain database. If you want to make sure that the slice plane's origin is at a specific node in a mesh, click the **Node** radio button and enter a new node number into the **Node** text field. Note that you must also specify a domain if you are slicing a multi-domain database.

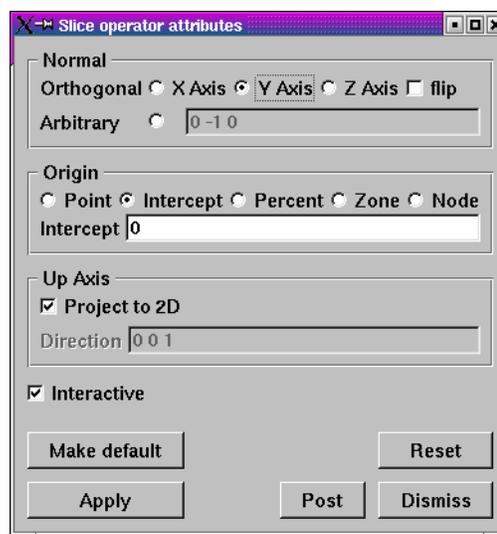


Figure 4-48: Slice operator attributes

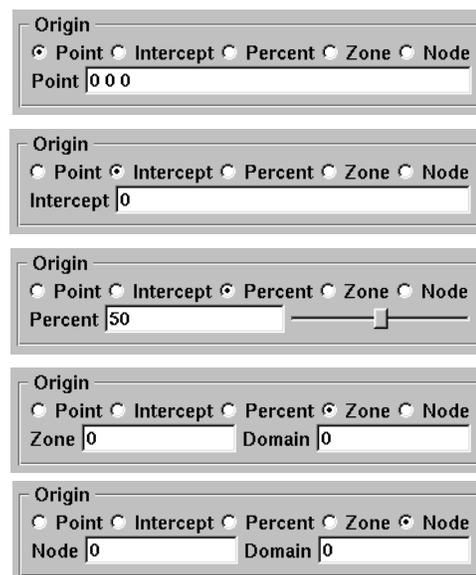


Figure 4-49: Origin area appearance

Use the up-axis vector when you want the slice plane to be projected to 2D. The up-axis vector is a vector that lies in the slice plane and defines a 2D coordinate system within the plane where the up-axis vector corresponds to the Y-axis. To change the up-axis vector, type a new 3D vector into the **Direction** text field in the **Up Axis** area of the window.

3.21.2 Positioning the slice plane using the Plane Tool

You can also position the slice plane using VisIt's interactive plane tool. The plane tool, which is available in the visualization window's popup menu, allows you to position a slice plane interactively using the mouse. The plane tool is an object in the visualization window that can be moved and rotated. When the plane tool is changed, it gives its new slice plane to the Slice operator if the operator is set to accept information interactively. To make sure that the Slice operator can accept a new slice plane from the plane tool, check the **Interactive** check box in the **Slice operator attributes window**. For more information about the plane tool, read the **Interactive Tools** chapter.

3.21.3 Projecting the slice to 2D

The Slice operator usually leaves sliced plots in 3D so you can position the slice with the plane tool. However, you might want the plot projected to 2D. When a sliced plot is projected to 2D, any 2D operation, like **Lineout**, can be applied to the plot. To project a plot to 2D, check the **Project 2D** check box in the **Slice operator attributes window**.

3.22 Smooth operator

The Smooth operator smooths a mesh to improve areas plagued by jagged edges or sharp peaks.

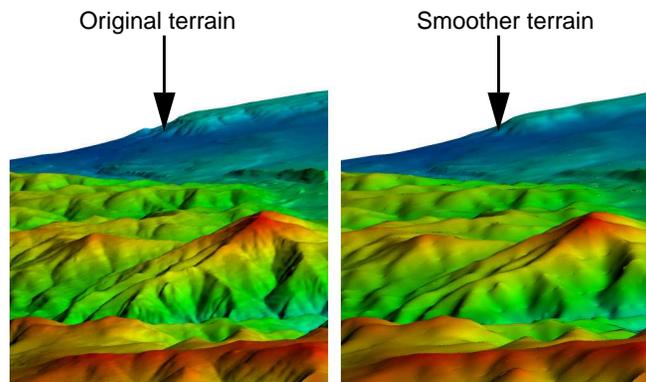


Figure 4-50: Smooth operator example

3.22.1 Using the Smooth operator

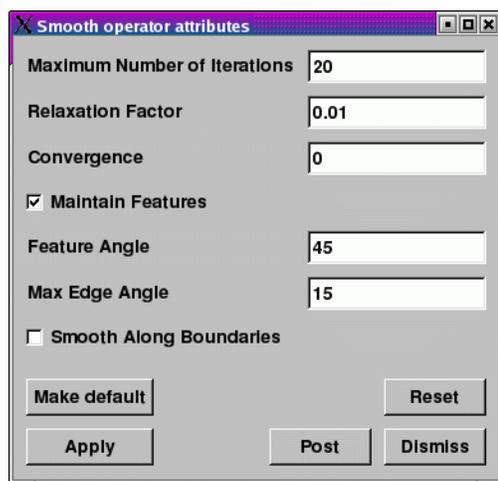


Figure 4-51: Smooth operator attributes window

The Smooth operator has a number of controls that can be used to tune mesh smoothness. The such control is the number of iterations, which controls the number of times the mesh relaxation algorithm is applied to the input mesh. Larger numbers of iterations will produce smoother meshes but will also take more time to compute. To change the number of iterations, type a new integer value into the **Maximum number of iterations** text field in the **Smooth operator attributes window** (see Figure 4-51). The relaxation factor is a floating point number in the range [0,1] and it controls how much the mesh is relaxed. Values near 1 produce a mesh that is very smooth relative to the input mesh. To use a new relaxation factor, type a floating point

number into the **Relaxation Factor** text field. The **Maintain Features** check box allows you to tell VisIt to preserve sharp peaks in the mesh while still smoothing out most of the mesh. The angle in the **Feature Angle** text field determines which features are kept. Any mesh angles less than the feature angle are preserved while others are smoothed.

3.23 SphereSlice operator

The SphereSlice operator slices a 2D or 3D database with an arbitrary sphere. Plots to which the SphereSlice operator have been applied become 2D surfaces that are coincident with the surface of the slicing sphere. The resulting plots remain in 3D space. You can use the SphereSlice operator to slice objects to judge their deviation from being perfectly spherical. An example of the SphereSlice operator is shown in Figure 4-52.

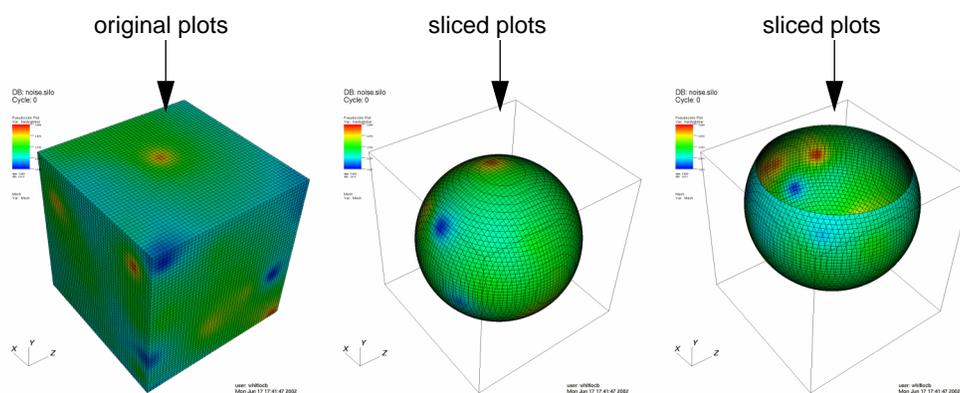


Figure 4-52: SphereSlice operator example

3.23.1 Positioning and resizing the slice sphere

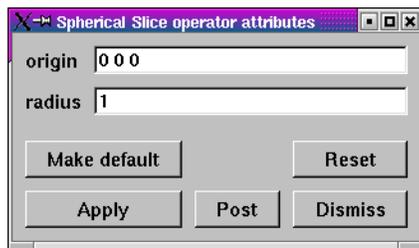


Figure 4-53: SphereSlice operator attributes window

You can position the slice sphere by setting its origin in the **SphereSlice operator attributes window** shown in Figure 4-53. The slice sphere is specified by a center point and a radius. To change the slice sphere's center, enter a new point into the **Origin** text field. The origin is a 3D coordinate that is represented by three space-separated floating point numbers. To resize the sphere, enter a new radius number into the **Radius** text field.

3.23.2 Positioning the slice sphere using the Sphere tool

You can also position the slice sphere using VisIt's interactive sphere tool. The sphere tool, available in the visualization window's popup menu, allows you to position and resize a slice sphere interactively using the mouse. The sphere tool is an object in the visualization window that can be moved and resized. When the sphere tool is changed, it gives its new slice sphere to the SphereSlice operator. For more information about the sphere tool, read the **Interactive Tools** chapter.

3.24 ThreeSlice operator

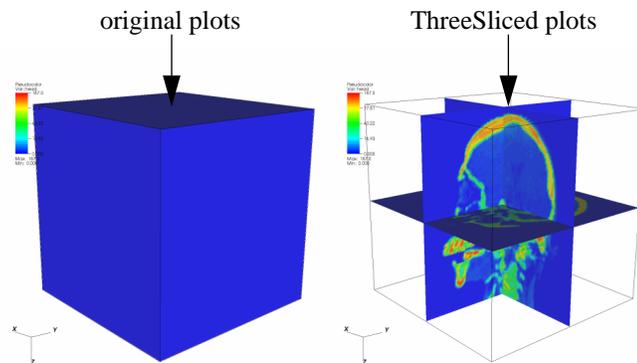


Figure 4-54: ThreeSlice operator example

The ThreeSlice operator slices 3D databases using three axis-aligned slice planes and leaves the resulting planes in 3D where they can all be viewed at the same time. The ThreeSlice operator is meant primarily for quick visual exploration of 3D data where the internal features cannot be readily observed from the outside.

3.24.1 Moving the ThreeSlice operator

The ThreeSlice operator is controlled by moving its origin, which is the 3D point where all axis-aligned slice planes intersect. There are two ways to move the ThreeSlice operator's origin. First, you can directly set the point that you want to use for the origin by entering new x, y, z values into the respective **X**, **Y**, **Z** text fields in the **ThreeSlice operator attributes window**, shown in Figure 4-55. You can also make sure that the **Interactive** toggle is turned on so you can use VisIt's interactive Point tool to set the ThreeSlice operator's origin. When you use the Point tool to set the origin for the ThreeSlice operator, the act of moving the Point tool sets the ThreeSlice operator's origin and causes plots that use the ThreeSlice operator to be recalculated with the new origin.

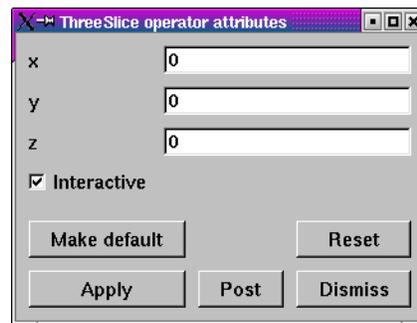


Figure 4-55: ThreeSlice operator attributes window

3.25 Threshold operator

The Threshold operator extracts cells from 2D and 3D databases where the plotted variable falls into a specified range. The resulting database can be used in other VisIt plots. You might use this operator when searching for cells with certain values. One such example is searching for the cell with the minimum or maximum value for the plotted variable. The Threshold operator removes all cells that do not have values in the specified range, making it easy to spot cells with the desired values. The Threshold operator can also use variables other than the plotted variable, for instance, you might want to see a Pseudocolor plot of pressure while using the Threshold operator to remove all cells below a certain density. By specifying a different threshold variable, it is possible to visualize different quantities over the subset of cells specified by the threshold variable and range. An example of the Threshold operator is shown in Figure 4-56.

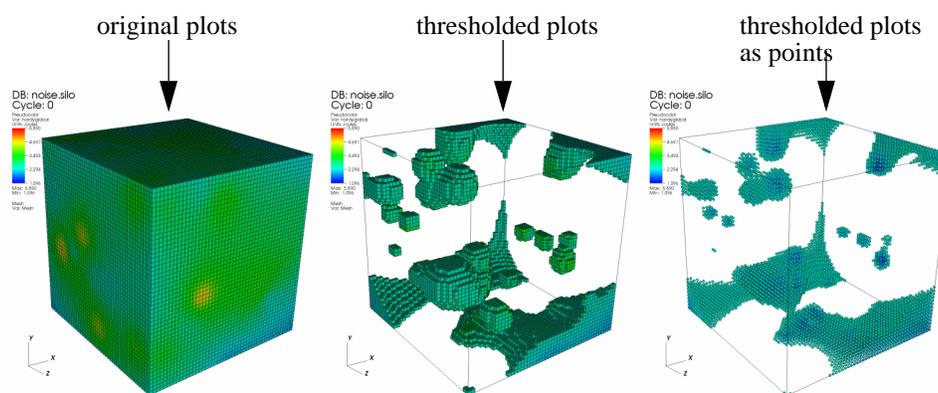


Figure 4-56: Threshold operator example

3.25.1 Setting the variable range

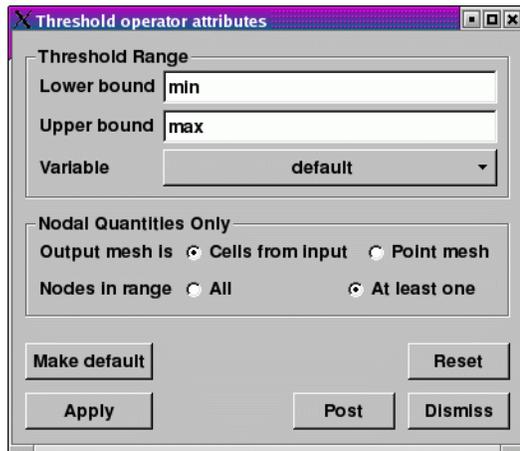


Figure 4-57: Threshold operator attributes window

The Threshold operator uses a range of values to determine which cells from the database should be kept in the visualization. You specify the range of values by lower and upper bounds on the threshold variable. Cells with values below the lower bound or with values above the upper bound are removed from the visualization. To specify a new lower bound, type a new number or the special keyword: *min* into the **Threshold operator attributes window's** (Figure 4-57) **Lower bound** text field. To specify a new upper bound, type a new number or the special keyword: *max* into the **Upper bound** text field.

When the threshold variable is a nodal quantity, the cell being considered by the Threshold operator has values at each node in the cell. In this case, the Threshold operator provides a control that determines whether or not to keep the cell if some nodes have values in the threshold range or if all nodes have values in the threshold range. More cells are usually removed from the visualization when all nodes must be in the threshold range. Click the **Some** radio button to allow cells where at least one value is in the threshold range into the visualization. Click the **All** radio button to require that all nodal values exist in the threshold range.

3.25.2 Setting the threshold variable

The Threshold operator uses the threshold variable to determine whether cells remain in the visualization. The threshold variable is usually the plotted variable in which case the **Variable** variable button displays: *default*. To specify a threshold variable other than the plotted variable, click on the **Variable** variable button and select a new scalar variable from the list of available variables.

You might set the threshold variable when you apply the Threshold operator to plots which do not take scalar variables as input. An example of this is the Mesh plot. When you apply the Threshold operator to a Mesh plot, you must set the threshold variable to a valid scalar variable for cells to be removed from the plot. You can also use the threshold variable to remove cells based on one variable while viewing the plotted variable.

3.25.3 Setting the output mesh type

The Threshold operator removes all cells that do not meet the threshold criterion, leaving behind a set of cells that are gathered into an unstructured mesh. Sometimes, it can be useful to transform the remaining cells into a point mesh. You can specify the desired

output mesh type using the **Cells from input** and **Point mesh** radio buttons in the **Threshold operator attributes window**.

3.26 Transform operator

The Transform operator manipulates a 2D or 3D database's coordinate field by applying rotation, scaling, and translation transformations. The operator's transformations are applied in the following order: rotation, scaling, translation. The Transform operator is applied to databases before they are plotted. You might use the Transform operator to rotate database geometry to a more convenient orientation or to scale database geometry to make better use of the visualization window. You can also use the Transform operator to make objects rotate and move around the visualization window during animations. This works well when only one part of the visualization should move while other parts and the view remain fixed. An example of the Transform operator is shown in Figure 4-58.

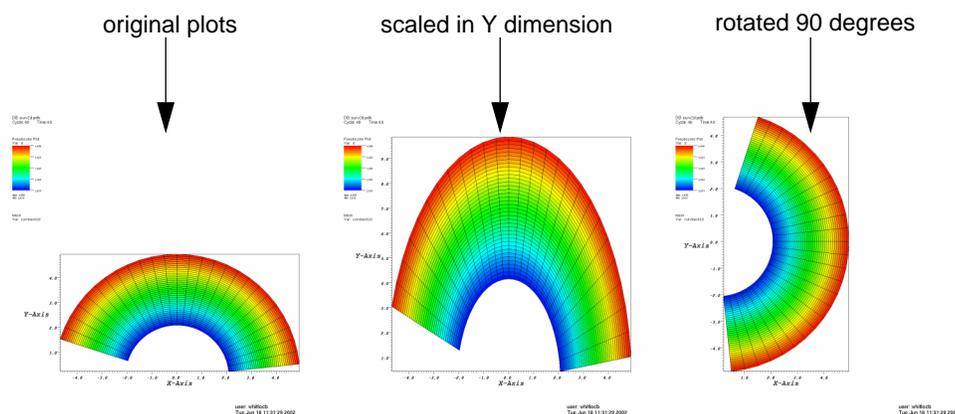


Figure 4-58: Transform operator example

3.26.1 Rotation

You can use the Transform operator to rotate plots around an arbitrary axis in 3D and around the Z-axis in 2D. To apply the rotation component of the Transform operator, be sure to check the **Rotate** check box in the **Transform operator attributes window** (Figure 4-59). An origin and normal are needed to specify the axis of rotation. The origin serves as a reference point for the object being rotated. The axis of rotation is a 3D vector that, along with the origin, determines the 3D axis that will serve as the axis of rotation. You must supply an origin and an axis vector to specify an axis of rotation. To change the origin, type a new 3D vector into the top **Origin** text field. To change the 3D axis, type a new 3D vector into the **Axis** text field. Both the origin and the axis are represented by three space-separated floating point numbers.

When applying the Transform operator to plots, you probably want to make the origin the same as the center of the plot extents which can be found by looking at the axis annotations. When the Transform operator is applied to 3D plots, the axis of rotation can

be set to any unit vector. When the Transform operator is applied to 2D plots, the axis of rotation should always be set to the Z-axis ($0\ 0\ 1$).

Once you specify the axis of rotation, you must supply the angle of rotation. The default angle of rotation is zero degrees, which gives no rotation. To change the angle of rotation, enter a number in degrees or radians into the **Amount** text field and click the **Deg** radio button for degrees or the **Rad** radio button if the angle is measured in radians.

3.26.2 Scale

You can use the Transform operator to scale plots. Each dimension can be scaled independently by entering a new scale factor into the **X**, **Y**, **Z** text fields. Each scale factor is a multiplier so that a value of 1 scales plots to their original size while a value of 2 scales plots to twice their original size. To apply the scale component of the Transform operator, be sure to check the **Scale** check box in the **Transform operator attributes window**. All dimensions are scaled relative to a scaling origin which can be changed by typing a new origin into the middle lower **Origin** text field.

3.26.3 Translation

You can use the Transform operator to translate plots. To apply the translation component of the Transform operator, be sure to check the **Translate** check box in the **Transform operator attributes window**. To translate plots in the X dimension, replace the default value of zero in the **X** translation text field. Translations in the Y and Z dimensions are handled in the same manner.

3.26.4 Coordinate system conversion

In addition to being able to rotate, scale, and translate plots, the Transform operator can also perform coordinate system conversions. A plot's coordinates can be specified in terms of Cartesian, Cylindrical, or Spherical coordinates (illustrated in Figure 4-60). Ultimately, when a plot is rendered in the visualization window, its coordinates must be

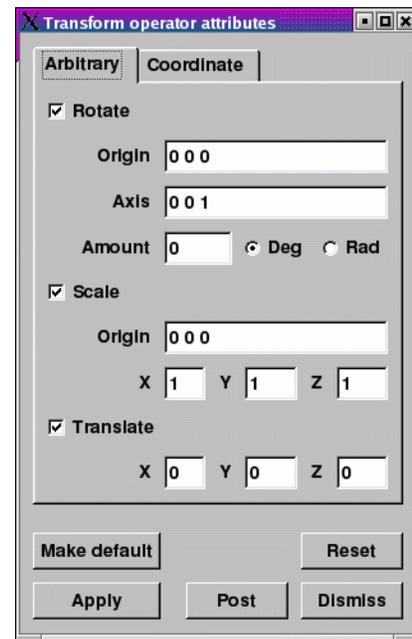


Figure 4-59: Transform operator attributes window

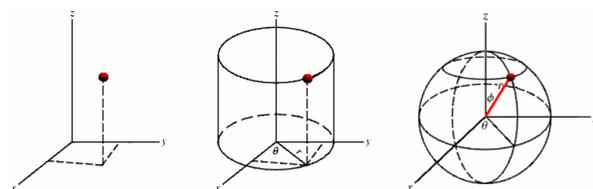


Figure 4-60: Cartesian, Cylindrical, and Spherical coordinate systems

specified in terms of Cartesian coordinates due to the implementation of graphics hardware. If you have a database where the coordinates are not specified in terms of Cartesian coordinates, you can apply the Transform operator to perform a coordinate system transformation so the plot is rendered correctly in the visualization window.

Figure 4-61 shows a model of an airplane that is specified in terms of spherical coordinates. When it is rendered initially, VisIt assumes that the coordinates are Cartesian, which leads to the plot getting stretched and tangled. The Transform operator was then applied to convert the plot's spherical coordinates into Cartesian coordinates, which allows VisIt to draw the plot as it is intended to look.

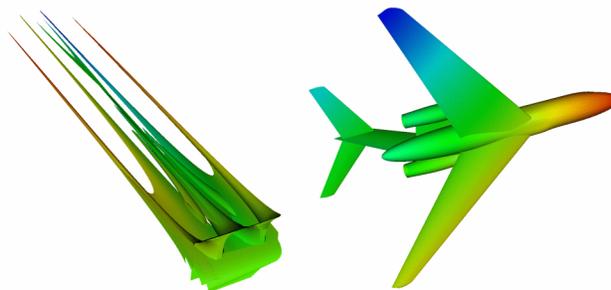


Figure 4-61: Coordinate system conversion using the Transform operator

The Transform operator allows coordinate system transformations between any of the three supported coordinate systems, shown in Figure 4-62. To pick a coordinate system transformation, you must first pick the coordinate system used for the input geometry. Next, you must pick the desired output coordinate system. In the example shown in Figure 4-61, the input coordinate system was Spherical and the output coordinate system was Cartesian. Note that if you use the Transform operator to perform a coordinate system transformation then you cannot also perform rotation, scaling, or translation. If you must perform any of those operations, add a second Transform operator to your plots.

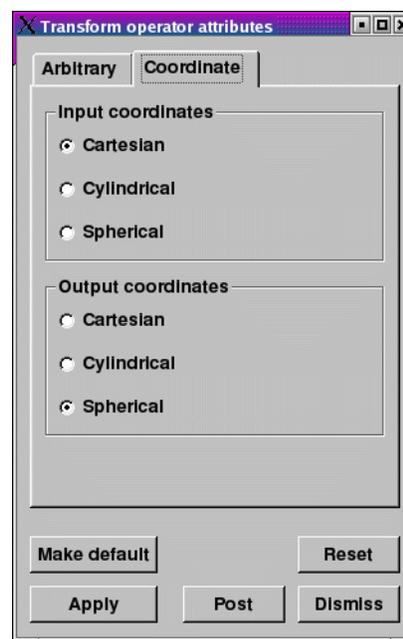


Figure 4-62: Supported coordinate systems

3.27 Tube operator

The Tube operator is a little-used operator that applies turns line geometry into tubes, making the lines appear fatter and shaded.



Figure 4-63: Lines transformed into tubes with the Tube operator

3.27.1 Changing tube appearance

The Tube operator provides a few knobs that control the appearance of the generated tubes. First of all, the tube width can be set by typing a new width into the **Width** text field in the **Tube operator attributes window** (Figure 4-64). The number of polygons used to make up the circumference of the tube can be altered by typing a new number of sides into the **Fitness of tube** text field. Finally, the ends of tubes can be capped instead of remaining open by turning on the **Cap Tubes** check box.

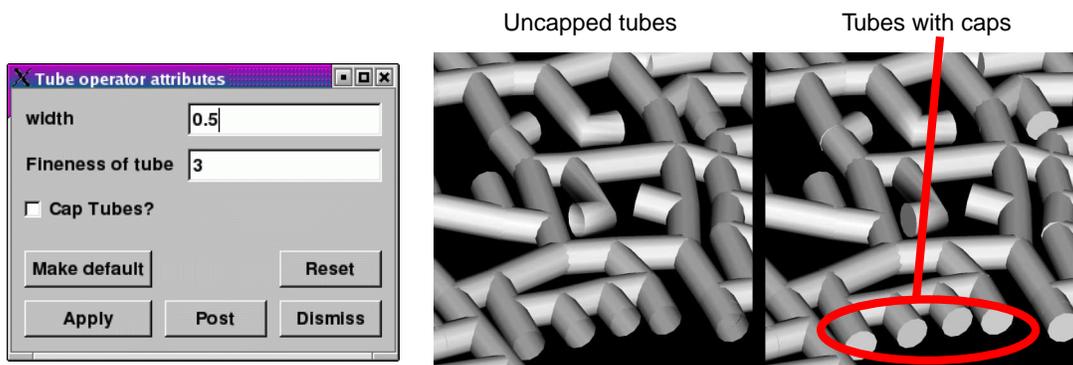


Figure 4-64: Tube operator attributes window and the effects of capping tubes

1.0 Overview

In this chapter, we discuss how to save and print files from within VisIt. The section on saving files is further broken down into three main areas: saving images, saving movies, and exporting databases. We first learn about saving images of visualizations using the **Save Window** and then we move on to saving movies and sets of image files using the **Save movie wizard**. After learning to save images and movies, this chapter concentrates on exporting databases to different file formats using the **Export Database Window**. Finally, we learn to print images of visualizations using the **Printer Window**.

2.0 Saving the visualization window

VisIt allows you to save the contents of any open visualization window to a variety of file formats. You can save visualizations as images so they can be imported into presentations. Alternatively, you can save the geometry of the plots in the visualization window so it can be imported into other computer modeling and visualization programs.

VisIt currently supports the image files formats: *BMP*, *JPEG*, *PNG*, *PPM*, *Raster Postscript*, *RGB*, and *TIFF*.

VisIt currently supports the geometry file formats: *Curve*, *Alias WaveFront Obj*, *STL*, *ULTRA*, and *VTK*. The *Curve* and *ULTRA* file formats are specially designed to store the data created from curve plots and can be used with other Lawrence Livermore National Laboratory visualization software. The *Alias Wavefront Obj* file format is supported so visualizations produced with VisIt can be imported into rendering programs such as *Maya*. VisIt can save visualizations into *STL* files, which are used with stereolithographic printers to fabricate three-dimensional parts. Finally, VisIt can save visualizations into

VTK (Visualization Toolkit) format for reading back into VisIt and other VTK-based applications at a later date.

When saving the geometry of plots in the visualization window into any of the aforementioned formats, you are performing a type of database export operation. However, saving geometry in this manner differs from exporting databases using the **Export Database Window**. Only the external faces of the plots are saved out when saving plot geometry whereas during a database export, 3D cells are preserved in the final exported database. The topic of exporting databases is covered later in this chapter.

2.1 The Save Window

You can open the **Save Window** by selecting **Set save options** from the **Main Window's File menu**. The **Save Window** contains the controls that allow you to set the options that govern how visualizations are saved.

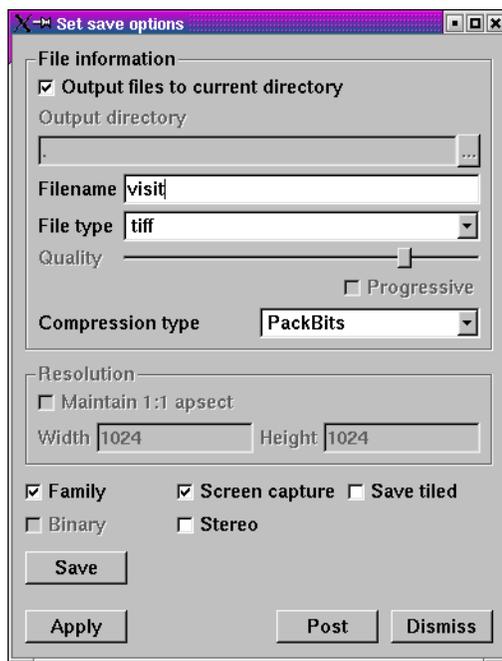


Figure 5-1: Save Window

The **Save Window**, shown in Figure 5-1, contains three basic groups of controls. The first group allows you to set the file information. Use the file information controls to set the name and destination of the saved file, as well as its file type and any optional quality parameters that may exist for the selected file type. Use the second group of controls when saving out an image file to specify the dimensions of the saved image. Finally, a group of check boxes near the bottom of the window control miscellaneous file saving features. When the save options are set and applied by clicking the **Apply** button, the active visualization can be saved either through the **Save Window** option in the **Main Window's File menu**, by the keyboard shortcut *Ctrl+S*, or by clicking the **Save** button in the **Save window**.

2.2 Picking an output directory for saved files

On most platforms, VisIt's default behavior is to save output files to the current directory, which is the directory where VisIt was started. On the Windows platform, VisIt saves images to the "My images" directory, which is a directory under the VisIt installation directory. If you want to specify a special output directory for your output files, you can turn off the **Output files to current directory** check box and type in the path to the

directory where you want VisIt to save your files in the **Output directory** text field. If you want to browse the file system to find a suitable directory in which to save your images, click on the “...” button to the right of the **Output directory** text field to bring up a **Directory chooser** dialog. Once you select a suitable directory using the **Directory chooser** dialog, the path that you chose is inserted into the **Output directory** text field.

2.3 Setting the save file name

To set the file name that will be used to save files, type a file name into the **Filename** text field. The file name that you use may contain a path to a directory where you want to write the saved files. If no path is specified, the saved files are written to the directory from which VisIt was launched. A file extension appropriate for the type of file being generated is automatically appended to the file name. For example, a *BMP* file will have a “.bmp” extension, while a *JPEG* file will have a “.jpeg” extension, and so on.

The file name that VisIt uses to save visualizations is based on the specified file name, the file format, and also the family toggle setting. The family toggle setting is set by checking the **Family** check box towards the lower left part of the **Save Window**. The family toggle setting allows you to save series of files that all have essentially the same name except for a number that is appended to the file name. The number increases by one each time an image is saved. If the family toggle setting is on then a file named “visit” of type *TIFF* will save out as “visit0000.tif”. If the family toggle setting is off, the file will save as “visit.tif”.

2.4 Setting the file type

You set the file type by making a selection from the **File type** menu. You can choose from image file types or geometry file types. Note that some areas of the **Save Window** become enabled for certain file types.

Choosing *JPEG* format files enables the **Quality** slider and the **Progressive** check box. These controls allow you to specify the desired degree of quality in the resulting *JPEG* images. A lower quality setting results in blockier images that fit into smaller files. The progressive setting stores the *JPEG* images in such a way that they progressively refine as they are downloaded and displayed by Web browsers.

Choosing *TIFF* format files enables the **Compression type** combo box. The available compression types are: None, PackBits, *JPEG*, and Deflate. When compression is enabled for *TIFF* files, they are smaller than they would be without compression.

Choosing *STL* or *VTK* file formats saves visualizations as geometry files instead of images and also enables the **Binary** check box. The **Binary** check box tells these formats to write their geometry data as binary data files instead of human-readable ASCII text files.

In general, files written with the binary option are smaller and faster to load than their non-binary counterparts.

2.5 Saving images with screen capture

The **Screen capture** check box tells VisIt to grab the image directly off of the computer screen. This means that the saved image will be exactly the same size as the image on the screen. There are advantages and disadvantages to using screen capture. An advantage is that capturing the image from the screen does not require VisIt to redraw the image to an internal buffer before saving, which usually results in a faster save. A disadvantage of screen capture is that any other windows on top of VisIt's visualization window occlude portions of the image. Screen capture can also be very slow over a sluggish network connection. Finally, using screen capture might not provide images that have enough resolution. Weigh the advantages and disadvantages of using screen capture for your own situation. Screen capture is on by default.

2.6 Setting image resolution

You set image resolution using the controls in the **Resolution** control group. These controls are disabled unless the file being saved is an image format and screen capture is not being used. You specify the image height and width by typing new values into the **Height** and **Width** text fields. If the **Maintain 1:1 aspect** check box is on, VisIt forces the image's height and width to be the same, yielding a square image. Turn off this setting if you want to save rectangular images. The image resolution is ignored unless you turn off the **Screen capture** check box.

2.7 Saving stereo images

When the **Stereo** check box is turned on and you save an image, VisIt will save a separate image for the left eye and for the right eye. The cameras used to generate each image are offset such that when the images are played together at high rates, they appear to have more depth. To enable saving of stereo images, click the **Stereo** check box in the **Save window** before you try to save an image.

2.8 Saving binary geometry files

Some geometry file formats such as *STL* and *VTK* have both ASCII and binary versions of the file format. The ASCII file formats are human-readable and are larger and more complex for programs to process than binary formats, which are not human-readable but are smaller and easier for programs to read. When geometry file formats support both ASCII and binary formats, the **Save Window's Binary** check box is enabled. By default VisIt writes ASCII geometry files but you can click the **Binary** check box to make VisIt write binary geometry files.

2.9 Saving tiled images

A tiled image is a large image that contains the images from all visualization windows that have plots. If you want to save tiled images, make sure to check the **Save tiled** check box in the **Save window**. To get an idea of how VisIt saves your visualization windows into a tiled image, see Figure 5-2.

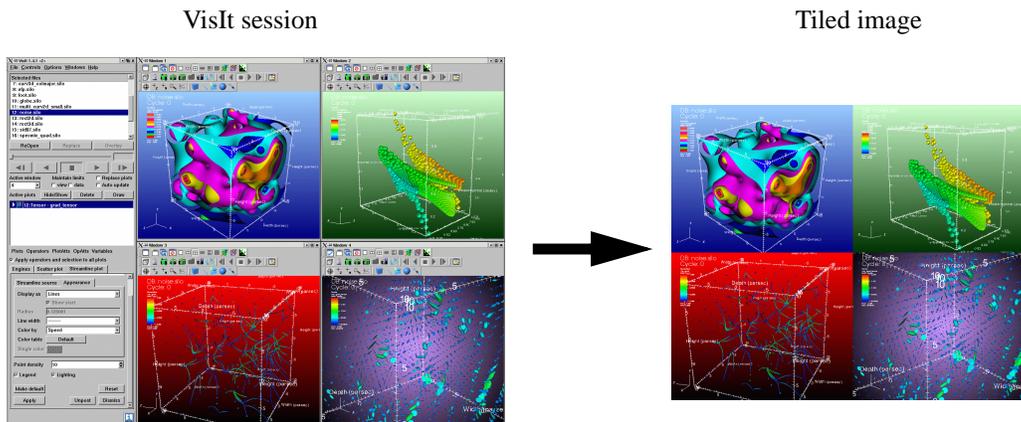


Figure 5-2: Saving tiled images example

3.0 Saving movies

In addition to allowing you to save images of your visualization window for the current time state, VisIt also allows you to save movies and sets of images for your visualizations that vary over time. There are multiple methods for saving movies with VisIt. This section introduces the **Save movie wizard** and explains how to use it to create movies from within VisIt's GUI. The **Animation and Keyframing** chapter on page 261 will explain some auxiliary methods that can be used to create movies.

The **Save movie wizard** (see Figure 5-3) is available in the **Main Window's Files** menu. The **Save movie wizard's** purpose is to lead you through a set of simple questions that allow VisIt to gather the information required to create movies of your visualizations. For example, the **Save movie wizard** asks which image and movie formats you want to generate, where you want to store the movies, what you want to call the movies, etc. Each of these questions appears on a separate screen in the **Save movie wizard** and once you answer the question on the current screen, clicking the **Next** button advances you to the next screen. You can cancel saving a movie at any time by clicking on the **Cancel** button. If you advance to the last screen in the **Save movie wizard** then you have successfully provided all of the required information that VisIt needs to make your movie. Clicking the **Finish** button at that point invokes VisIt's movie-making script to make the movie. If you want to make subsequent movies, you can choose to use the

settings for the movies that you just made or you can choose to create a new movie and provide new information.

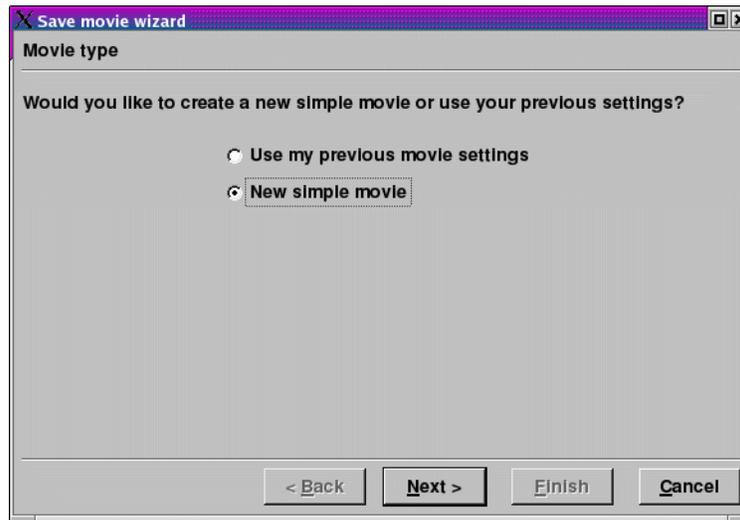


Figure 5-3: Save movie wizard (screen 1)

3.1 Choosing movie formats

The **Save movie wizard's** second screen, shown in Figure 5-4, allows you to pick the types of movies that you want to create. You can select as many image and movie formats as you want and you can even specify multiple resolutions of the same movie. VisIt allows you to order multiple versions of your movie because it is often easier to create different versions of the movie all at once as opposed to doing it later once you've discovered that you need a new version to play on a laptop computer or a tiled display wall.

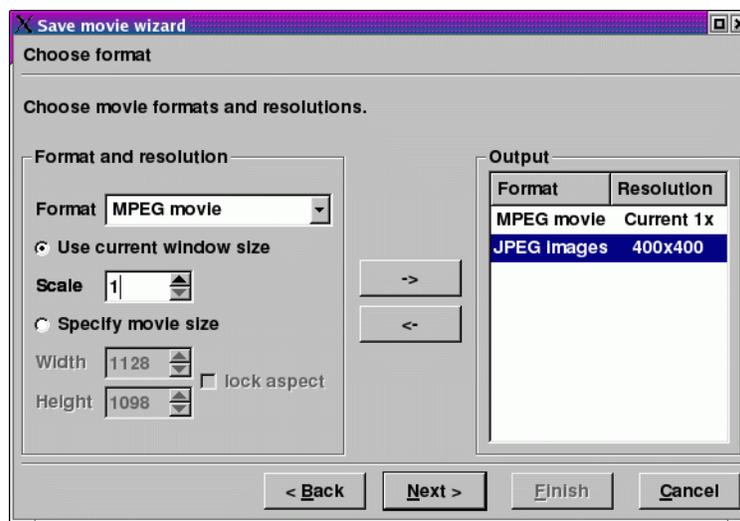


Figure 5-4: Save movie wizard (screen 2)

The **Save movie wizard's** second screen is divided vertically into two main areas. On the left you will find the **Format and resolution** area, which displays the format and resolution for the current movie. On the right, you will find the **Output** area, which lists the formats and resolutions for all of the movies that you have ordered. By default no movie formats are present in the **Output** area's list of movies. You cannot proceed to the next screen until you add at least one movie format to the list of movies in the **Output** area.

To add a movie format to the list of movies in the **Output** area, first choose the desired movie format from the **Format** combo box in the **Format and resolution** area. Next, choose the movie resolution. The movie resolution can be specified in terms of the visualization window's current size or it can be specified in absolute pixels. The default movie resolution uses the visualization window's current size with a scale of 1. You can change the scale to shrink or grow the movie while keeping the visualization window's current aspect ratio. If you want to specify an absolute pixel size for the movie, click on the **Specify movie size** radio button and type the desired movie width and height into the **Width** and **Height** text fields. Note that if you specify a width and height that causes the movie's shape to differ from the visualization window's shape, you might want to double-check that the view used for the visualization window's plots does not change appreciably.

Once you have selected the desired movie format, width, and height, click on the right-arrow button that separates the **Format and resolution** area from the **Output** area. Clicking the right-arrow button adds your movie to the list of movies that you want to make. Once you have at least one movie in the **Output** area, the screen's **Next** button will become active. Click the **Next** button to go to the next screen in the **Save movie wizard**.

3.2 Making a stereo movie

The **Save movie wizard** allows you to create stereo movies if you choose the **Yes** button on the wizard's third screen. The default is to create regular, non-stereo movies because stereo movies are not widely supported. The only movie format that VisIt produces that is compatible with stereo movies is the "Streaming movie" format, which is an LLNL format commonly used for tiled displays. The "Streaming movie" format can support stereo movies where the image will flicker between left and right eye versions of the movie, causing a stereo effect if you view the movie using suitable liquid-crystal goggles. The stereo option has no effect when used with other movie formats. However, if you choose to save a stereo movie in any of VisIt's supported image formats, VisIt will save images for the left eye and images for the right eye. You can then take the left and

right images into your favorite stereo movie creation software to create your own stereo movie.



Figure 5-5: Save movie wizard (screen 3)

3.3 Choosing the movie name

Once you have specified options that tell VisIt what kinds of movies that you want to make, you must provide the base name and location for your movies. By default, movies are saved to the directory in which you started VisIt. If you want to specify an alternate directory, you can either type in a new directory path into the **Output directory** text field (see Figure 5-6) or you can select a directory from the **Choose directory** dialog box activated by clicking on the “...” button.

The base filename for the movie is the name that is prepended to all of the movies that you generate. When generating multiple movies with differing resolutions, the movie resolution is often encoded into the filename. VisIt may generate many different movies

with different names but they will all share the same base filename that you provided by typing into the **Base filename** text field.

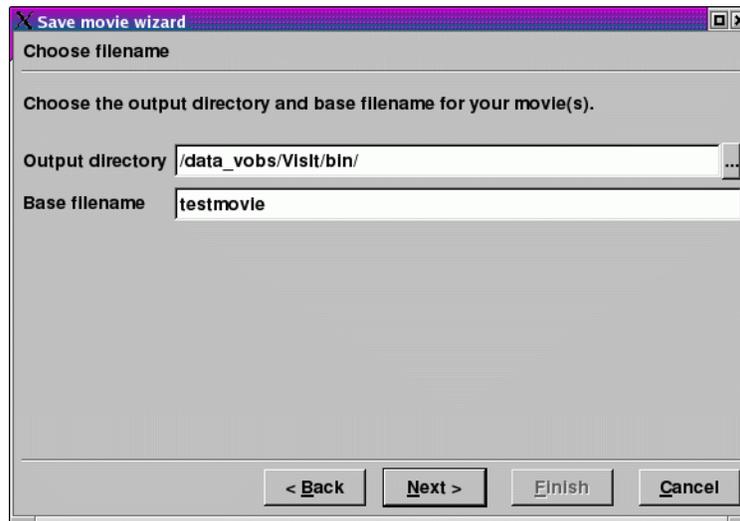


Figure 5-6: Save movie wizard (screen 4)

3.4 Choosing movie generation method

After you've specified all of your movie options, VisIt prompts you how you would like your movie made. At this point, you can click the **Finish** button to make VisIt start generating your movie. You can change how VisIt creates your movie by clicking a different movie generation method on the **Save movie wizard's** fifth screen, shown in Figure 5-7.

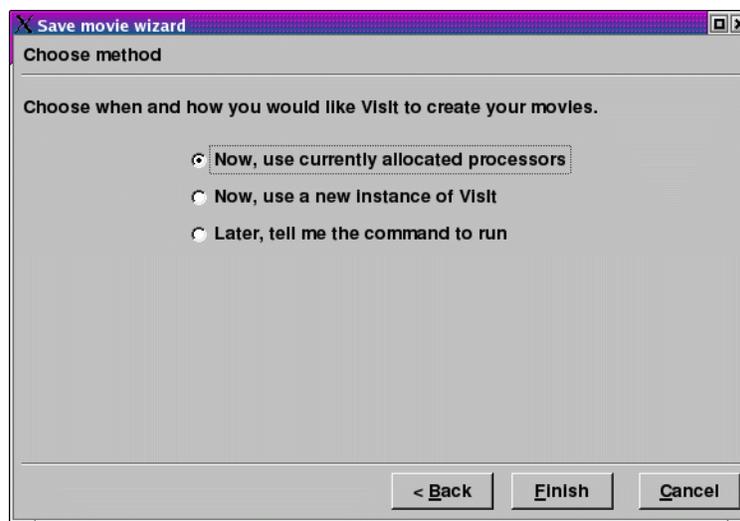


Figure 5-7: Save movie wizard (screen 5)

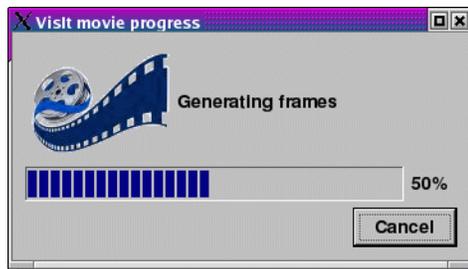


Figure 5-8: Movie progress dialog

The default option for movie creation allows VisIt to use your current VisIt session to make your movies. This has the advantage that it uses your current compute engine and allocated processors, which makes movie generation start immediately. When you use this movie generation method, VisIt will launch its command line interface (CLI) and execute Python movie-making scripts in order to generate your movie. This means that you have both the VisIt GUI and CLI controlling the viewer. If you use this movie generation method, you will be able to watch your movie as it is generated. You can track the movie's progress using the **Movie progress dialog**, shown in Figure 5-8. The downside to using your currently allocated processors is that movie generation takes over your VisIt session until the movie is complete. If you want to regain control over your VisIt session, effectively cancelling the movie generation process, you can click the **Movie progress dialog's Cancel** button.

The second movie generation method will cause VisIt to save out a session file containing every detail about your visualization so it can be recreated by a new instance of VisIt. This method works well if you want to create a movie without sacrificing your current VisIt session but you cannot watch the movie as it is generated and you may have to wait for the second instance's compute engine to be scheduled to run. The last movie generation option simply makes VisIt display the command that you would have to type at a command prompt in order to make VisIt generate a movie of your current visualizations.

4.0 Exporting databases

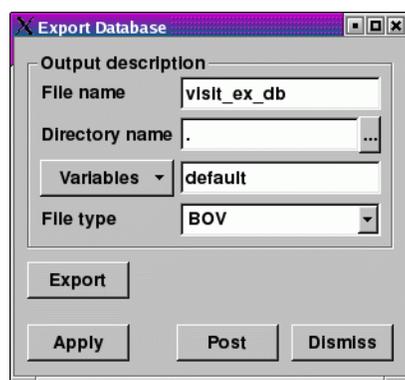


Figure 5-9: Export Database Window

Plot geometry can be saved to a handful of geometric formats by saving the plots in the window to a format such as VTK. Often saving the plot geometry, which only consists of the visible faces required to draw the plot, is not enough. When interfacing VisIt to other tools you may want to save out the database that you've plotted in a different file format. For instance, you might plot a 3D database and want to export actual 3D cells for the entire database instead of just the externally visible geometry. You might also want to save out additional variables that you did not plot. VisIt allows this kind of data export via the **Export Database Window**, shown in Figure 5-9.

You can find the **Export Database Window** in the **Main Window's Files** menu. To save a database, you must first have opened a database and created a plot. Note that the data transformations applied by plots or operators will affect the data that you export. This

allows you to alter the data using sophisticated chains of operators before you export it for use in another tool.

4.1 Exporting variables

The **Export Database Window** allows you to export a subset of the variables for your active plot's database by letting you specify which variables are to be exported. To choose which variables should be exported, you can type the names of the variables to export into the **Variables** text field or you can select from the available variables in the **Variables** menu depicted in Figure 5-10. You can select as many variables as you want from the menu. Each time you select a variable from the **Variables** menu, VisIt will append it to the list of variables to be exported.

4.2 Choosing an export file format

The **Export Database Window** lists the names of the database reader plugins that can also write data back into their native file formats.

A small handful of the total number of database plugins currently support this feature but in the future most formats will support this capability more fully, making VisIt not only a powerful visualization tool but a powerful database conversion tool.

You can try to use any of the supported export formats to export your data but some of the file formats may not be able to accept certain types of data. The Silo file format can safely export any type of data that you may want to export. If you want to export data to other applications and the data must be stored in an ASCII file that contains columns of data, you might want to choose the Xmdv file format. If you want to choose a specific database plugin to export your data files, make a selection from the **File Type** menu shown in Figure 5-11.

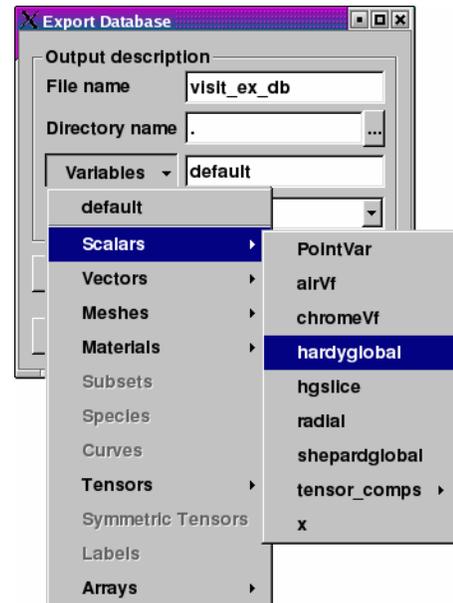


Figure 5-10: Variables menu

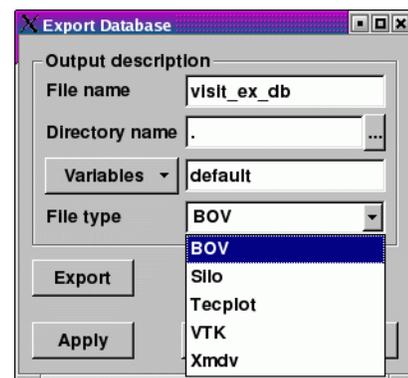


Figure 5-11: Export file types

5.0 Printing

VisIt allows you to print the contents of any visualization window to a network printer or to a *PostScript* file.

5.1 The Printer Window

Open the **Printer window** by selecting **Set Print options** from the **Main Window's File menu**. The **Printer Window's** appearance is influenced by the platform on which you are running VisIt so you may find that it looks somewhat different when you use the Windows, UNIXTM, or MacOS X versions of VisIt. The Linux version of the **Printer Window** is shown in Figure 5-12.

The **Printer Window** contains a number of options to set the destination printer, number of pages to print, and paper orientation. After changing the settings, click the **Apply** button to apply the settings or click **Cancel** to revert to the previous settings. To print an image, select the **Print window** option from the **Main Window's File menu**.

Note that the **Print window** (see Figure 5-13) looks and behaves a little differently on MacOS X. When it comes up, clicking the **Print** button causes immediate printing unlike printing on other platforms. This was done deliberately to be more consistent with other MacOS X applications.

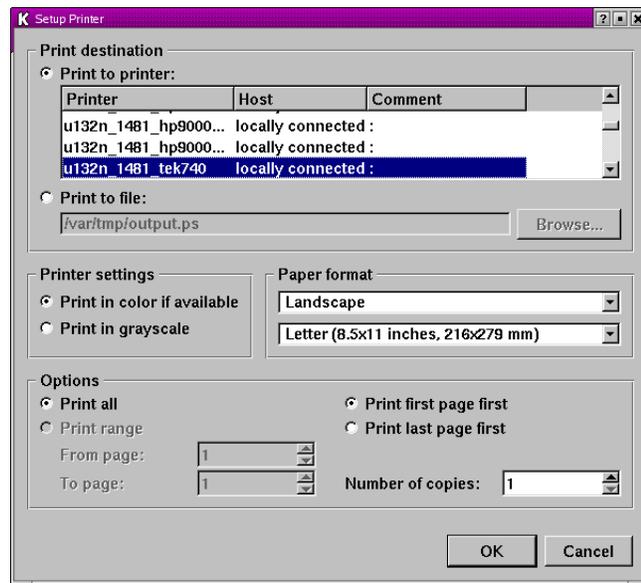


Figure 5-12: Printer Window

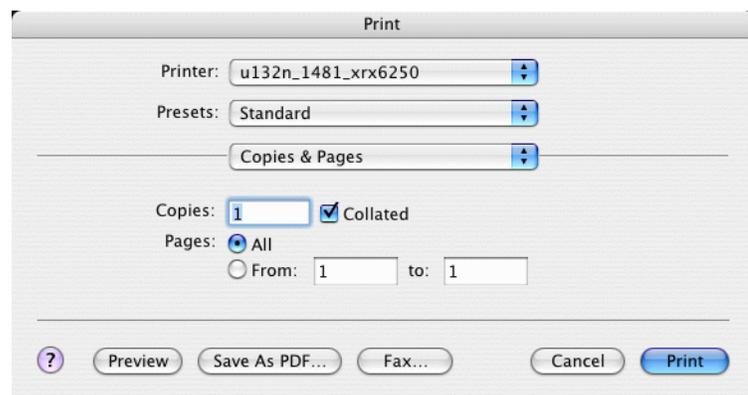


Figure 5-13: Printer window on MacOS X

5.2 Setting the printer destination

You use the printer destination to tell VisIt where you print your file. You have the option of printing to a network printer or a file. If you want to print to a network printer, select the **Print to printer** radio button and choose the name of the printer from the list of available network printers. The list of printers is read from your system settings and should be up-to-date. If you want to print to a file, select the **Print to file** radio button and type a valid file name into the file text field. You can also click on the **Browse** button to select a directory in which to write the *PostScript* file.

5.3 Changing the color settings

The **Printer Window** has two radio buttons that allow you to set whether you want to print images in color or in grayscale. The buttons, located in the middle left of the **Printer Window**, are in a group called **Printer Settings**.

5.4 Setting paper format

The **Printer Window** allows you to set the type of paper on which you will print. This helps VisIt determine how it should scale images when printing. You can set the size of the paper by choosing from the lower menu in the **Paper format** group located in the middle right of the **Printer Window**. You can also set the printed image orientation to *Portrait* or *Landscape* by choosing an option from the top menu in the **Paper format** group.

5.5 Setting the number of printed copies

The **Printer Window** allows you to request multiple copies of the printed image. To request multiple copies, locate the **Number of copies** text field in the lower right of the **Printer Window** and enter an integer greater than one or click on the up arrow button next to the text field.

1.0 Overview

A visualization window, also known as a vis window, is a window that displays plots and allows you to interact with them using the mouse. The vis window not only allows for direct manipulation of plots but it also provides a popup menu and toolbar that allow you to switch window modes, activate interactive tools, and perform commonly used operations. This chapter explains how to manage and use vis windows.

2.0 Managing vis windows

VisIt allows you to create up to 16 vis windows and to manage those vis windows, VisIt provides controls to add vis windows, remove vis windows or alter their layout. The controls for managing vis windows are located in the **Main Window's Windows** menu (see Figure 6-1), as well as in the vis window's **Toolbars** and **Popup menu**.

2.1 Adding a new vis window

You can add a new vis window in a few different ways, the first of which is by selecting the **Add** option from the **Main Window's Windows** menu. You can also click on the **Add window icon** in the vis window's **Toolbar** or you can select the **Add window** option from the **Windows** submenu in the vis window's **Popup menu** to add a

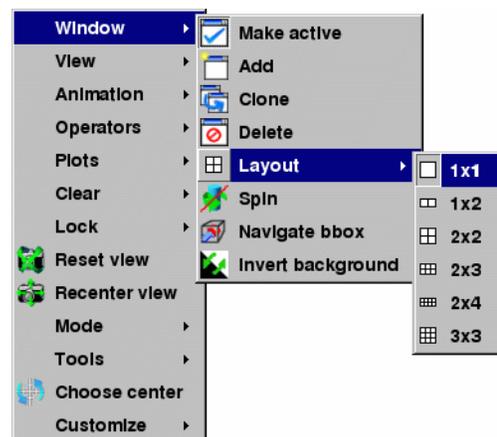


Figure 6-1: Window menu

new vis window. When you add a new window, it will be sized according to the window layout so if you have only a single, large vis window, the new vis window will also be large. You can change the window layout to shrink the vis windows so that they both fit on the screen. Vis windows are numbered 1 to 16 so the new window will have the first available number for which there is not already a window. If you have windows 1, 2, and 4, vis window 3 would be created by adding a new window. Adding a new window also makes the new window the active window.

A new vis window can also be added by cloning the active window. You can clone the active window by selecting the Clone option from the **Main Window's Windows** menu or you can click the **Clone window icon** in the vis window's **Toolbar**. When you clone the active window, VisIt creates a new window as if you had clicked the Add option but it also copies the plots, annotations, and lighting from the active window so that the new window is identical in appearance to the active window. When plots are copied to the new cloned window, they have not yet been generated so their plot list entries in the **Plot list** are green. You can force the plots to be generated by clicking the **Draw** button in the **Main Window**.

2.2 Deleting a vis window

There are four ways to delete a vis window. The first way is to select the **Delete** option from the **Main Window's Window menu**. When you delete a window in this manner, the active window gets deleted and VisIt makes the window with the smallest number the new active window. The second way to delete a window is to click on the **close window button** in the window decorations provided by the windowing system. The window decorations' appearance varies based on the platform and windowing system used to run VisIt, but the button used to close windows is commonly a button with an X in it. An example of a **close window button** is shown in Figure 6-2.



Figure 6-2: Window decorations with close button

The third way to delete a vis window is to click on the **Delete window icon** in the vis window's **Toolbar**. The fourth way to delete a vis window is to use the **Delete** option in the vis window's **Popup menu**. When you use the **Toolbar** or the **Popup menu** to delete a window, the window does not need to be the active window as when other controls are used.

2.3 Clearing plots from vis windows

The **Main Window's Windows menu** provides a **Clear all** option that you can use to clear the plots from all vis windows. Selecting this option does not delete the plots from a vis window's plot list but it does clear the plots so they have to be regenerated by VisIt's

compute engine. You can also clear the plots for just the active window by selecting the **Plots** option from the **Clear** submenu in the **Main Window's Windows** menu (see Figure 6-3). You might find clearing plots useful when you want to make several changes to plot attributes because, unlike plots that are already generated, setting attributes of cleared plots does not force them to regenerate when you change their attributes.

In addition to clearing plots, you can also clear pick points and reference lines from a vis window. A pick point is a marker that VisIt adds to a vis window when you click on a plot in pick mode. The marker indicates the location of the pick point. A reference line is a line that you draw in a vis window when it is in lineout mode. You can clear a vis window's pick points or reference lines, by selecting the **Pick points** or **Reference lines** options from the **Clear** submenu in the **Main Window's Windows** menu.

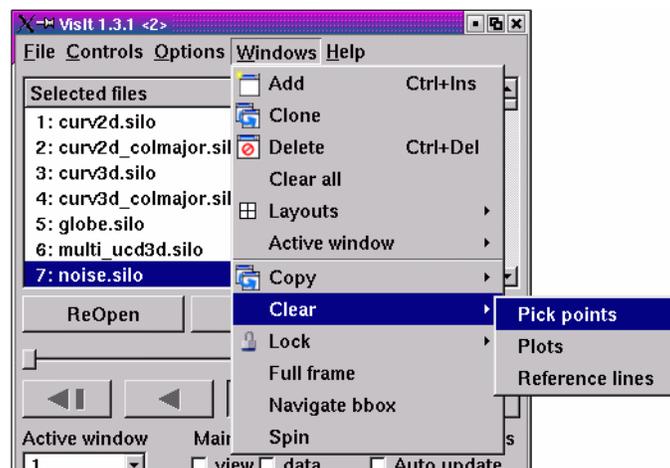


Figure 6-3: Clear menu

2.4 Changing window layouts

VisIt uses different window layouts to organize vis windows so they all fit on the screen. Changing the window layout typically resizes all of the vis windows and moves them into a tiled formation. If there are not enough vis windows to complete the desired layout, VisIt creates new vis windows until the layout is complete. You can change the layout selecting a new layout from the **Layouts** menu located in the **Main Window's Window menu** or you can click on a layout icon in the vis window's **Toolbar**.

2.5 Setting the active window

VisIt has the concept of an active window that is the window to which new plots are added. You can change the active window by selecting a window number from the **Active window** menu located in the center of the **Main Window**. Setting the active window updates the GUI so that it displays the state for the new active window. The **Active window** menu is shown in Figure 6-4. You can also set the active window using the

Active window submenu in the **Main Window's Windows** menu or you can click on the **Active window icon** in the vis window's **Toolbar**.

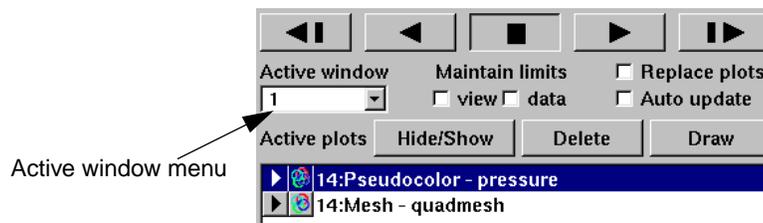


Figure 6-4: Active window menu

2.5.1 Copying window attributes

VisIt allows you to copy window attributes and plots from one window to another when you have more than one window. This can be useful when you are comparing plots generated from similar databases. The **Copy** menu, shown in Figure 6-5, contains options to copy the view, lighting, annotations, plots, or everything from other from other vis windows. Under each option, the **Copy** menu provides a list of available vis windows from which attributes can be copied so, for example, if you have two windows and you want to copy the view from vis window 1 into vis window 2, you can select the **Window 2** option from the **View from** submenu. The list of available windows depends on the vis windows that you have created. You can copy the lighting from one window to another window by using the **Lighting from** submenu or you can use the **Annotations from** or **Plots from** to copy the annotations or plots, respectively. If you make a selection from the **Everything from** submenu, all attributes and plots are copied into the active vis window.

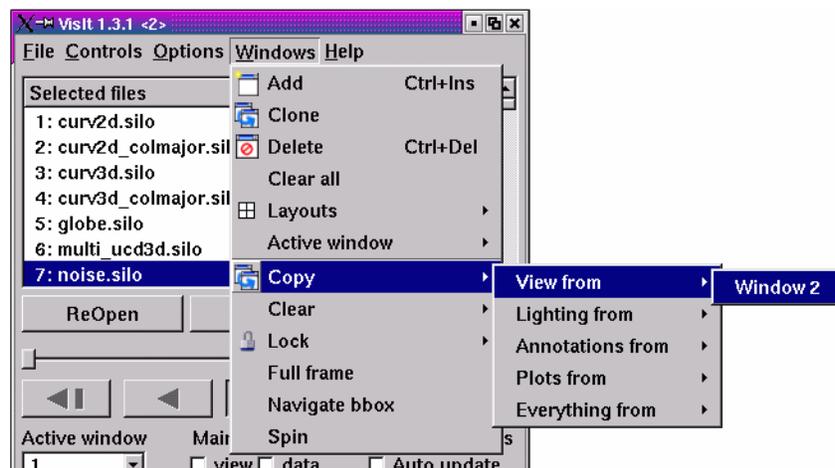


Figure 6-5: Copy menu

2.5.2 Locking vis windows together

When you use VisIt to do side by side comparisons of databases, you may find it useful to lock vis windows together. Vis windows can be locked together in time so that when you change the active database timestep in one database, as when viewing an animation, all vis windows that are locked in time switch to the same database timestep. You can lock vis windows together in time by selecting the **Time** option from the **Lock** menu (see Figure 6-6) in the **Main Window's Windows** menu. Any number of windows can be locked together in time and you can turn off time locking at any time.

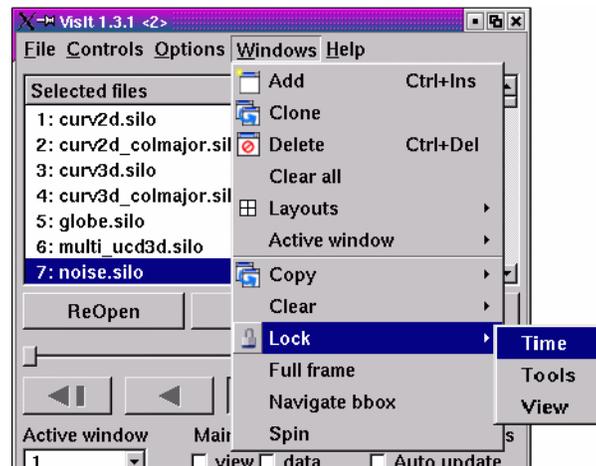


Figure 6-6: Lock menu

You can also lock interactive tools together so that updating a tool in one window updates the tool in other windows that have enabled tool locking. This can be useful when you have sliced a database using the plane tool in more than one window and you want to be able to change the slice using plane tool in either window and have it affect the other vis windows. You can enable tool locking by selecting the **Tools** option from the **Lock** menu.

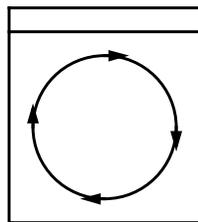
In addition to locking vis windows in time, or locking their tools together, you can also lock vis windows' views together so that when you change the view in one vis window, other vis windows get the same view. When you change the view in a vis window that has view locking enabled, the view only effects other vis windows that also have view locking enabled and have plots of the same dimension. That is, when you change the view of a vis window that contains 3D plots, it will only have an effect on other locked vis windows if they have 3D plots. Vis windows that contain 2D plots are not affected by changing the view of a vis window containing 3D plots and vice-versa. When you enable view locking, the vis window snaps to the view used by other vis windows with locked views or it stays the same if no other vis windows have locked views. To enable view locking, select the **View** option from the **Lock** menu or click on the **Lock view icon** in the vis window's **Toolbar**.

3.0 Using vis windows

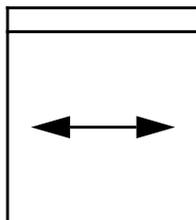
The first thing to know about using a vis window is how to change window modes. A window mode is a state in which the vis window behaves in a specialized manner. There are four window modes: *Navigate*, *Zoom*, *Lineout*, and *Pick*. Vis windows are in navigate mode by default. This means that most mouse actions are used to move, rotate, or zoom-in on the plots that the vis window displays. Each vis window has a **Popup menu** that can be activated by clicking the right mouse button while the mouse is inside of the vis window. The **Popup menu** contains options that can put the vis window into other modes and perform other common operations. To put the vis window into another window mode, open the **Popup menu**, select **Mode** and then select one of the four window modes. You can also change the window mode using the vis window's **Toolbar**, which has buttons to set the window mode. You can find out more about the **Popup menu** and **Toolbar** later in this chapter.

3.1 Navigate mode

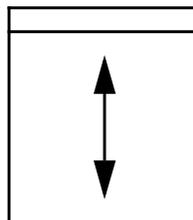
Navigate mode is VisIt lingo for moving and zooming-in on plots. When the vis window is in navigate mode, clicking the left mouse button and dragging with the mouse will perform an action that moves, rotates, or zooms the plot. The mouse motions used to rotate plots are shown in Figure 6-7. You can translate plots by holding down the *Shift* key before left-clicking and dragging the plot. You zoom in on plots by clicking the middle button and moving the mouse up or down. Sometimes the controls are modified based on the interactor settings. For more information, look at the section on *Interactor settings* on page 149.



Rotating about the z-axis of the screen.



Rotating about the y-axis of the screen.



Rotating about the x-axis of the screen.

Figure 6-7: Mouse motions used to rotate plots in navigate mode

3.2 Zoom mode

When the window is in zoom mode, you can draw a box around the area of the vis window that you want drawn larger. Press the left mouse button and move the mouse to sweep out a box that will define the area to be zoomed. Release the mouse button when the zoom box covers the desired area. If you start zooming and decide against it before releasing the left mouse button, clicking one of the other mouse buttons cancels the zoom operation.

Changes to the view can be undone by selecting the **Undo view** option from the popup menu's **View** menu. Sometimes the zoom controls can change based on the interactor settings, which are described further on in *Interactor settings*.

3.3 Lineout mode

Lineout mode is only available when the vis window contains 2D plots. A lineout is essentially a slice of a two dimensional dataset that produces a one dimensional curve in another vis window. When a vis window is in lineout mode, pressing the left mouse button in the vis window creates the first endpoint of a line that will be used to create a curve. As you move the mouse around, the line to be created is drawn to indicate where the lineout will be applied. When you release the mouse button, VisIt adds a lineout to the vis window and a curve plot is created in another vis window.

3.4 Pick mode

When a vis window is in pick mode, any click with the left mouse button causes VisIt to calculate the value of the plot at the clicked point and place a pick point marker in the vis window to indicate where you clicked. The calculated value is printed to the **Output Window** and the **Pick Window**.

4.0 Interactor settings

Some window modes such as Zoom mode and Navigate mode have certain interactor properties that you can set. Interactor properties influence how user interaction is fed to the controls in the different window modes. For example, you can set zoom interactor settings that clamp a zoom rectangle to a square or fill the viewport when zooming. VisIt provides the **Interactors window** so you can set properties for window modes that have interactor properties. The **Interactors window** is shown in Figure 6-8.

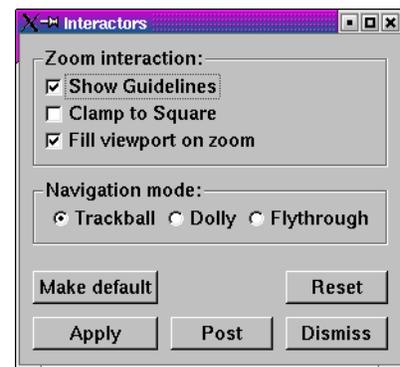


Figure 6-8: Interactors window

4.1 Zoom interactor settings

The zoom interactor settings are mostly used when the vis window is in zoom mode. When the vis window is in zoom mode, clicking in the vis window will anchor a point that becomes one of the corners of a zoom rectangle. When you release the mouse, the point over which the mouse was released becomes the opposite corner of the zoom rectangle. VisIt's default behavior is to show guidelines that extend from the edges of the zoom rectangle to the edges of the plots' bounding box when the vis window is in 2D mode. If

you want to turn off the guidelines, click off the **Show Guidelines** check box in the **Interactors window**.

When sweeping out a zoom rectangle in zoom mode, VisIt allows you to draw a rectangle of any proportion. The relative shape of the zoom rectangle, in turn, influences the shape of the viewport drawn in the vis window. If you hold down the *Shift* key while sweeping out the zoom rectangle, VisIt will constrain the shape of the zoom rectangle to a square. If you want VisIt to always force a square zoom rectangle so that you don't have to use the *Shift* key, you can click on the **Clamp to Square** check box, click **Apply** in the **Interactors window** and save your settings.

Using the **Clamp to Square** zoom mode is a good way to maximize the amount of the vis window that is used when you zoom in on plots and when the vis window is in zoom mode. When the vis window is in navigate mode, the middle mouse button also effects a zoom. By default, zooming with the middle mouse button zooms into the plots but keeps the same vis window viewport which may, depending on the aspect ratio of the plots, not make the best use of the vis window's pixels. Fortunately, you can turn on the **Fill viewport on zoom** check box to force middle mouse zooming to also enlarge the viewport to its largest possible size in order to make better use of the vis window's pixels.

4.2 Navigation styles

When VisIt displays 3D plots, there are a few navigation styles from which you can choose by clicking on the following radio buttons in the **Interactors window**: **Trackball**, **Dolly**, and **Flythrough**. The default navigation style for 3D plots is: **Trackball** and it allows you to interactively rotate plots and move around them but it keeps the camera at a fixed distance from the plots and while it can get infinitely close to plots when you zoom in, it can never touch them or go inside of them. The **Dolly** navigation style behaves like the trackball style except that the when the camera zooms, it is actually moved. The **Flythrough** navigation style moves the camera and allows you to fly into plots and out the other side.

5.0 The Popup menu and the Toolbar

Each vis window contains a **Popup menu** and a **Toolbar**, which can be used to perform several categories of operations such as window management, setting the window mode, activating tools, manipulating the view, or playing animations. Options in the **Popup menu** exist in the **Toolbar** and vice-versa. A group of actions that is represented in the **Popup menu** as a menu usually maps to a toolbar in the vis window's **Toolbar**. To perform an action using the **Toolbar**, you can just click on its buttons. Access the **Popup menu** by pressing the right mouse button in the vis window. Select the desired item, then release the mouse button.

5.1 Hiding toolbars

The **Popup menu** has a **Customize** menu that lets you customize the vis window's **Toolbar**. For instance, you can choose to hide all of the toolbars so that they do not take up any of your screen space if you use a small monitor. If you want to hide all toolbars, you can select the **Hide toolbars** option from the **Customize** menu. If you want to show the toolbars again, you can click the **Show toolbars** option in the **Customize** menu. Note that when you select the **Show toolbars** option, VisIt only shows the toolbars that were enabled before they were hidden. If you want to enable or disable individual toolbars, you can select from the **Toolbars** menu under the **Customize** menu so VisIt only shows the toolbars that you routinely need. Once you tell VisIt which toolbars you want to use, you can save your preferences using the **Save settings** option in the **Main Window's Options** menu so that the next time you run VisIt, it only shows the toolbars that you enabled.

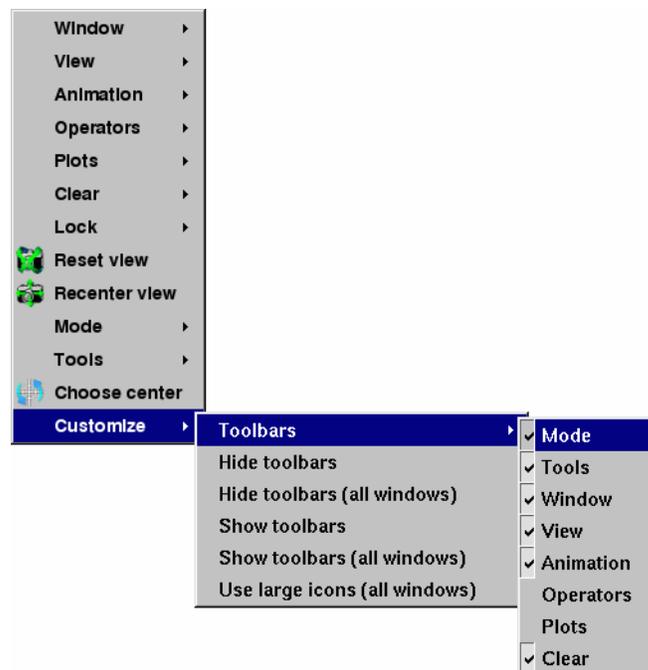


Figure 6-9: Customize menu

5.2 Moving toolbars

Each of the vis window **Toolbar's** smaller toolbars can be moved to other edges of the vis window by clicking the small tab on the left or top side of the toolbar and dragging it to other edges of the vis window.

5.3 Switching window modes

The **Popup menu** contains a **Mode** menu (see Figure 6-10) that contains the 5 window modes. You can select a window mode from the **Mode** menu to change the vis window's

mode. If you want to move or zoom the plot, choose navigate or zoom modes. If you want to extract data from the plots in the vis window, choose lineout mode or one of the pick modes. You can also use the **Mode toolbar** to change the vis window's window mode.

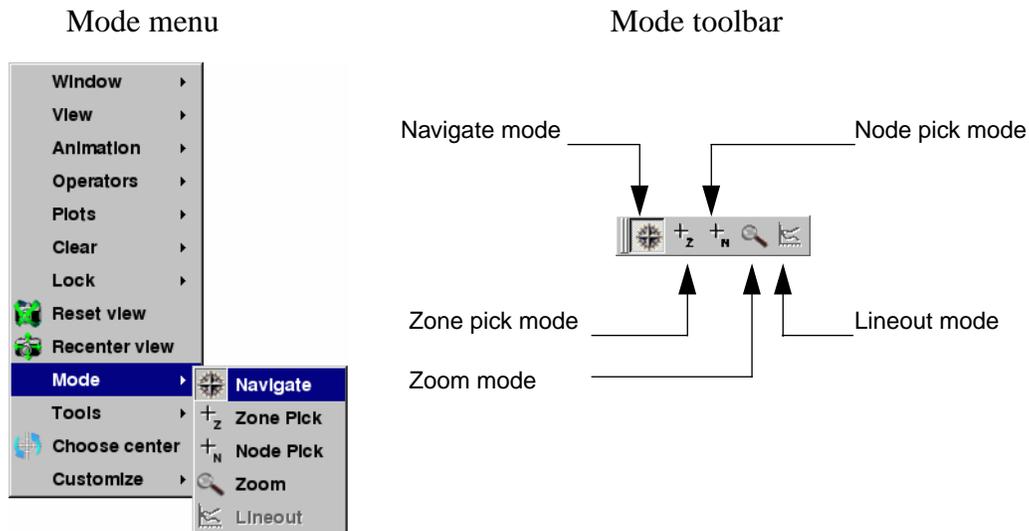


Figure 6-10: Mode menu and toolbar

5.4 Activating tools

The **Popup menu** contains a **Tools** menu (see Figure 6-11) that lists of all of VisIt's interactive tools. Each tool shown in the menu has an associated icon that is used to indicate if the tool is enabled and if it is available in the vis window. Some tools are not available if the vis window does not contain plots or if the plots in the vis window are the wrong dimension to be used with the tool. In that event, the tool cannot be activated and the menu and toolbar entries for that tool are disabled. If a tool is available, its icon is bright blue; otherwise the icon is grayed out. If a tool is enabled, its icon has a selection rectangle around it. To activate a tool, choose an inactive tool from the **Tools** menu or

click on its button in the **Toolbar**. To deactivate a tool, choose the tool that you want to deactivate from the **Tools** menu or click on its button in the **Toolbar**.

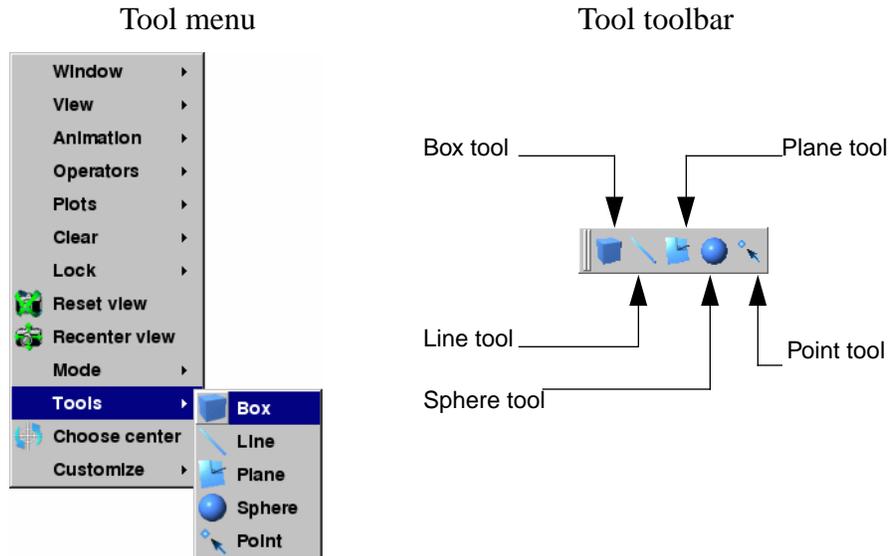


Figure 6-11: Tool menu and toolbar

5.5 View options

VisIt's **Popup menu** and **Toolbar** (see Figure 6-12) have several options that are available for manipulating the view. You can reset the view, recenter the view, undo a view change, toggle perspective viewing, save and reuse useful views, or choose a new center of rotation.

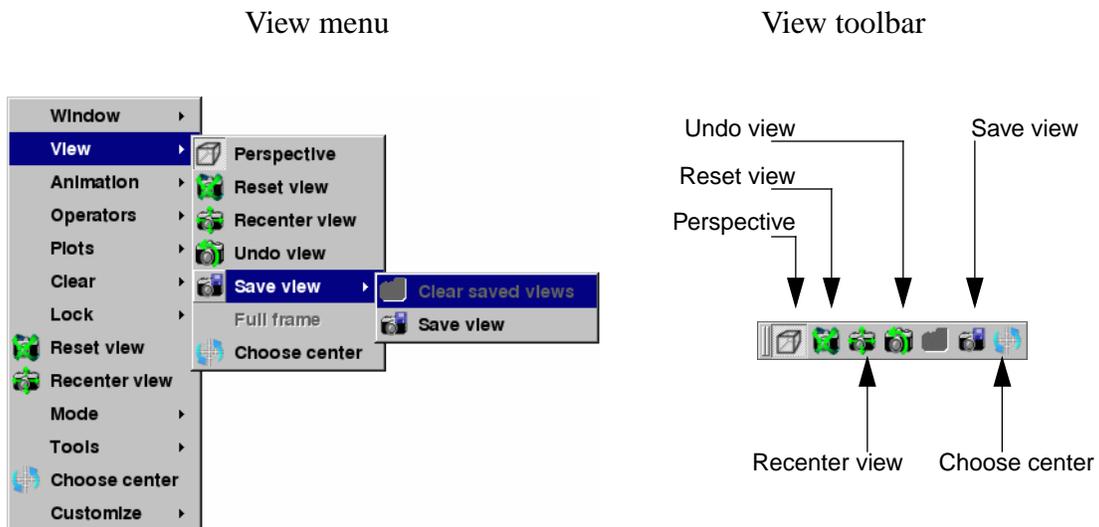


Figure 6-12: View menu and toolbar

5.5.1 Resetting the view

The **Popup menu** has a **Reset view** option (see Figure 6-12) that resets the view used to view the plots in the vis window. The view is typically reset to look down the $-Z$ axis in a right-handed coordinate system. You can reset the view by selecting the **Reset view** option from the **Popup menu** or by clicking on the **Reset view icon** in the **Toolbar**.

5.5.2 Recentering the view

Sometimes adding a plot to a vis window that already contains plots can result in a lopsided visualization. This happens when the spatial extents of the plots do not match. The **Popup menu** has a **Recenter view** option (see Figure 6-12) to calculate a new center of rotation for the plots so they are drawn in the center of the window. You can also recenter the view by clicking on the **Recenter view icon** in the **Toolbar**. To make sure that the view updates appropriately when new plots are added to the vis window, you may also want to check the **Auto center view** check box that is available in the **View Window**.

5.5.3 Undo view

The vis window saves the last ten views in a buffer so that you can restore them if you make an unintended change to the view. You can undo a view change, by selecting the **Undo view** option in the **Popup menu's View** menu or by clicking the **Undo view icon** in the **Toolbar** (see Figure 6-12).

5.5.4 Changing view perspective

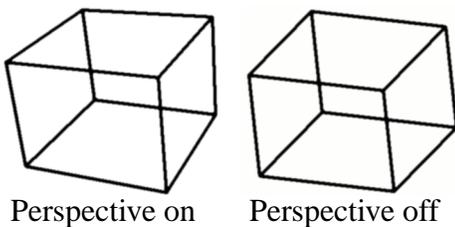


Figure 6-13: Perspective examples

When the vis window contains 3D plots, the perspective setting can be used to enhance how 3D the plot looks. In a perspective projection, graphics grow smaller as they recede into the distance which makes them look more realistic. To change the perspective setting, click on the **Perspective** option in the **Popup menu's View** menu (see Figure 6-12). When the vis window uses a perspective projection, the **Popup menu's Perspective** option will have a selection rectangle

around its icon. You can also turn perspective on or off by clicking on the **Perspective icon** in the **Toolbar**. The difference in appearance having perspective and not having it is shown in Figure 6-13.

5.5.5 Locking views

The vis window can lock its view to other vis windows. When this toggle is set, making a change that affects the view in the active vis window will cause other vis windows that have the lock views toggle set to receive the same view as the active window. To lock the view, select the **Lock view** option from the **Popup menu's View** menu (see Figure 6-

12) or click on the **Lock view icon** in the **Toolbar**. Note that you can lock 2D and 3D windows separately.

5.5.6 Saving and reusing views

Sometimes when analyzing a database, it is useful to be able to toggle between several different views. VisIt allows you to save up to 15 views that you can then use to look at different parts of your visualization. When you navigate to a view that you like, click the **Save view** icon in the **View** toolbar or click the **Save view** option in the **Popup menu's View** menu to save the view. When you save a view, VisIt adds a new numbered camera icon to the **View** toolbar and the **Popup menu**. Clicking on a view icon makes VisIt use the view that is associated with the clicked icon so you have one-click access to all of your saved views. You can preserve the saved views across VisIt sessions if you save your settings. If you want to delete the saved views so you can create different saved views, click the **Clear saved views** icon next to the **Save views** icon in the **View** toolbar.

5.5.7 Fullframe mode

Some databases yield plots that are so long and skinny that they leave most of the vis window blank when VisIt displays them. VisIt provides Fullframe mode to stretch the plots so they fill more of the vis window so it is easier to see them. It is worth noting that Fullframe mode does not preserve a 1:1 aspect ratio for the displayed plots because they are stretched in each dimension so they fit better in the vis window. To activate Fullframe mode, click on the **Fullframe** option in the **Popup menu's View** menu.

5.5.8 Choosing a new center of rotation

When you are working with a 3D database and you have created plots and zoomed in on them, you should set the center of rotation. The center of rotation is the point about which the plots are rotated when you rotate the plots in navigate mode. Normally, the center of rotation is set to the center of the plots being visualized. When you zoom way in on plots and attempt to rotate them, the default center of rotation often causes plots to whiz off of the screen when you rotate because the center of rotation is not close enough to the geometry that you are actually viewing. To set the center of rotation to something more suitable, VisIt provides the **Choose center** button, which can be accessed in the **Popup** menu or in the **View** toolbar. Once you click the **Choose center** button, VisIt temporarily switches to pick mode so you can click on the part of your visualization that you want to become the new center of rotation. Once you click on a plot, VisIt exits pick mode and uses the picked point as the new center of rotation. After setting the center of rotation, VisIt will make sure that the picked point is visible at all times.

5.6 Animation options

The animation controls in VisIt's **Main Window** are not the only controls that are provided for playing animations. Each vis window's **Popup menu** and **Toolbar** has

options for playing and stepping through animations. To play an animation, select the Play option from the **Popup menu's Animation** menu or click on the **Play icon** in the **Toolbar**, shown in Figure 6-14. To play the animation in reverse, select the **Reverse play** option or click on the **Reverse play icon** in the **Toolbar**. To stop the animation from playing, select the **Stop** option in the **Animation** menu or click on the **Stop icon** in the **Toolbar**. If you want to advance or reverse one frame at a time, use forward or reverse step.

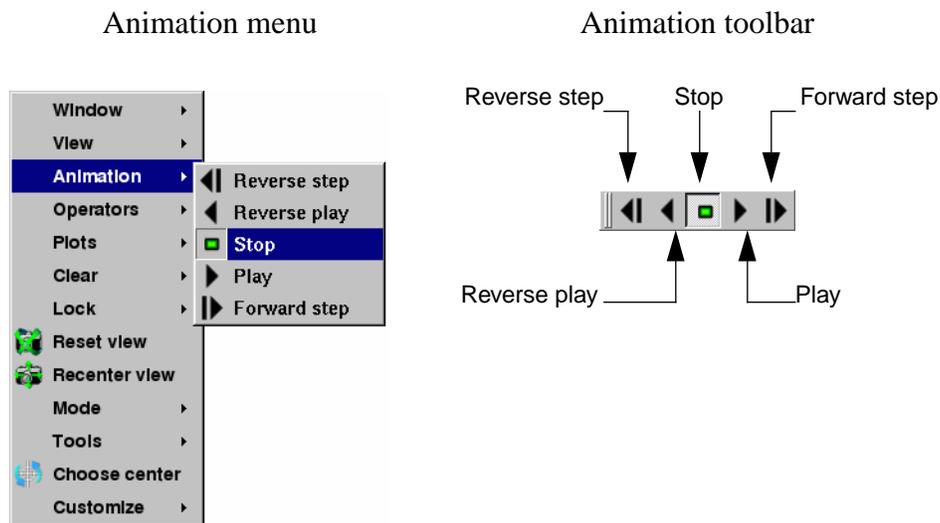


Figure 6-14: Animation menu and toolbar

5.7 Window options

Many window options have previously been explained in this chapter so this section describes some additional options that were not covered. Many of the options in the **Main**

Window's Windows menu are also present in the **Popup menu's Window** menu and toolbar (see Figure 6-15).

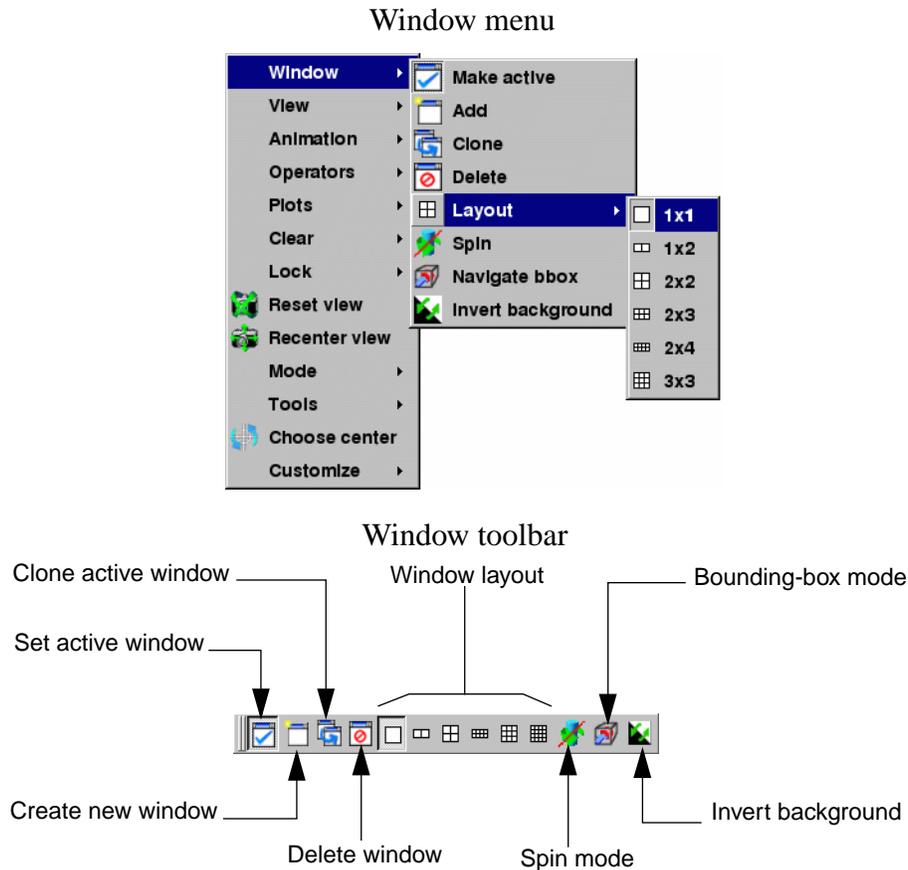


Figure 6-15: Window menu and toolbar

5.7.1 Changing bounding-box mode

The vis window allows a simple wireframe box to be substituted for complex plots when you want to rotate or move them. This is called bounding-box navigation and you can use it during navigate mode for complex plots so you can navigate faster when a vis window contains plots that take a long time to redraw. You can change the bounding-box mode by selecting the **Navigate bbox** option from the **Popup menu's Window** menu shown in Figure 6-15. You can also change the bounding-box mode by clicking on the **Bounding-box icon** in the **Toolbar**.

5.7.2 Engaging spin

Spin is a setting that makes plots spin after the user stops rotating them and it provides a nice, easy way to see the entire plot without having to actively rotate it. To spin a 3D plot, turn on the **Spin** option in the **Popup menu's Windows** menu and then rotate the plot as you would in navigate mode. The plot will continue to spin after you release the mouse buttons. You can also engage spin using the **Spin** option in the **Main Window's**

Windows menu or by clicking the **Spin icon** in the vis window's **Toolbar**. You can stop plots from spinning by turning off spin.

5.7.3 Inverting the foreground and background colors

Sometimes it is useful to swap the vis window's foreground and background colors. You can invert the background and foreground colors by clicking on the **Windows** menu's **Invert background** option. Note that this option is disabled when the vis window has a gradient background.

5.8 Clear options

The **Clear** menu (see Figure 6-16) in the **Popup menu** contains options that cause certain items such as: plots, pick points, and reference lines to be removed from a vis window. The **Clear** menu also appears in the **Main Window's Windows** menu.

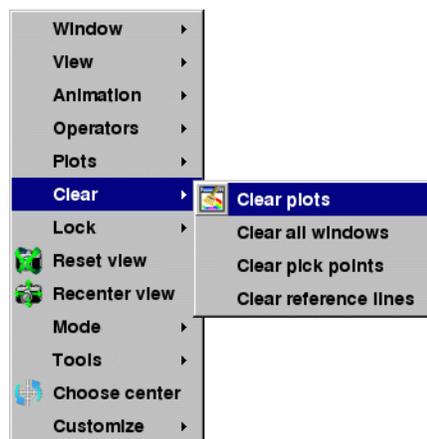


Figure 6-16: Clear menu

5.8.1 Clearing plots from all windows

Sometimes it is useful to clear all plots from the vis window. Clearing plots from the vis window does not delete the plots but instead deletes their computed geometry and returns them to the new state so they appear green in the **Plot list**. An example of when you might want to clear plots is when you change material interface reconstruction options since changing them requires a plot to be regenerated. Rather than deleting plots that existed before changing the material interface reconstruction parameters, you can clear the plots and force them to be completely regenerated by clearing the plots.

5.8.2 Clearing pick points

Click on the **Clear** menu's **Clear pick points** option if you want to remove all of the pick labels that were added when you picked on the plots in the vis window. Clearing the pick points also removes any pick information related to those pick points in the **Pick** window.

5.8.3 Clearing reference lines

Click on the **Clear** menu's **Clear reference lines** option if you want to remove all of the reference lines that were added to the vis window when you performed lineouts on the plots in the vis window.

5.9 Plot options

The **Plot** toolbar and **Plot** menu let you create new plots using variables from the open databases and also let you hide, delete, and draw the plots that correspond to the selected plot entries in VisIt's **Plot list**. The **Plot** menu is always available in the **Popup menu** but the **Plot** toolbar is not visible by default. If you want to make the **Plot** toolbar visible, you can turn it on in the **Popup menu's Customize** menu. The **Plot** menu and toolbar are shown in Figure 6-17.

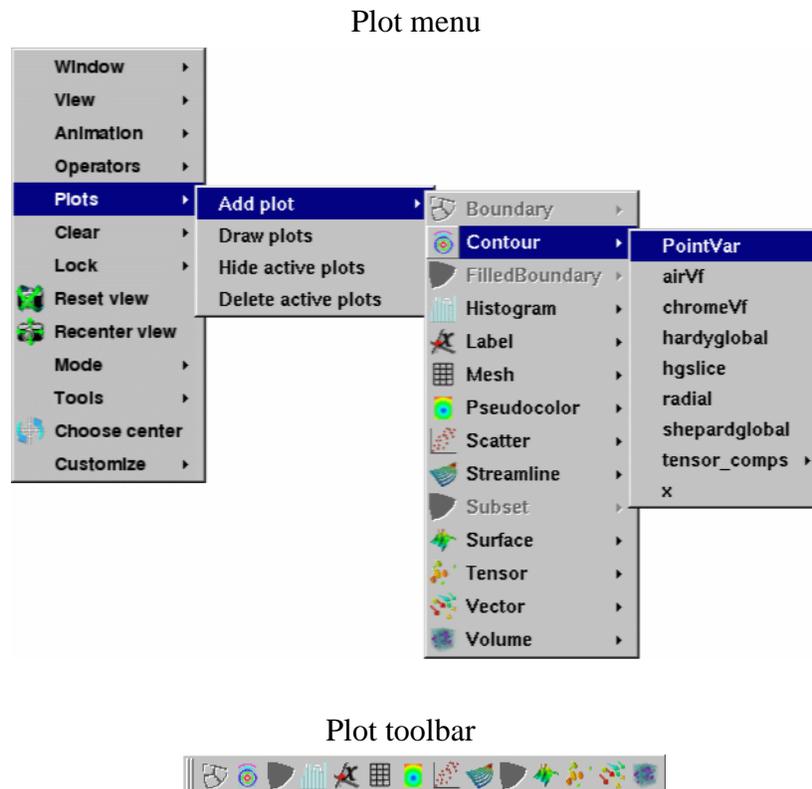


Figure 6-17: Plot menu and toolbar

5.9.1 Adding a plot

The **Plot** menu and toolbar both provide options for you to add new plots. Each plot has its own menu option or icon that contains the variables that can be plotted from the open database. To add a new plot using the **Plot** menu, click the **Add plot** option to activate the list of available plots and then select a variable for the desired plot type. To add a new plot using the **Plot** toolbar, click on the icon for the desired plot type and select a variable from

its variable menu. A new plot will appear in the **Main Window's Plot list** and it will be in the new state. To draw the plot, click the **Draw** button.

5.9.2 Drawing a plot

All plots added using the **Plot** menu or toolbar are in the new state, indicating that they have not been generated yet. To generate a plot once it has been created, click the **Draw** plots option in the **Plot** menu.

5.9.3 Hiding active plots

To hide the active plots, which are the plots that are highlighted in the **Main Window's Plot list**, click the **Plot** menu's **Hide active plots** option. Once clicked, the selected plots are made invisible until you hide them again to show them.

5.9.4 Deleting active plots

To delete the active plots, which are the plots that are highlighted in the **Main Window's Plot list**, click the **Plot** menu's **Hide active plots** option. Once a plot has been deleted, you can't get it back.

5.10 Operator options

The **Operator** menu and toolbar allow you to add new operators and remove operators from plots. The **Operator** menu is always available in the **Popup menu** but the **Operator toolbar** is not visible by default. If you want to make the **Operator toolbar** visible, you can turn it on in the **Popup menu's Customize menu**. The **Operator** menu and **Operator toolbar** are shown in Figure 6-18 and Figure 6-19.

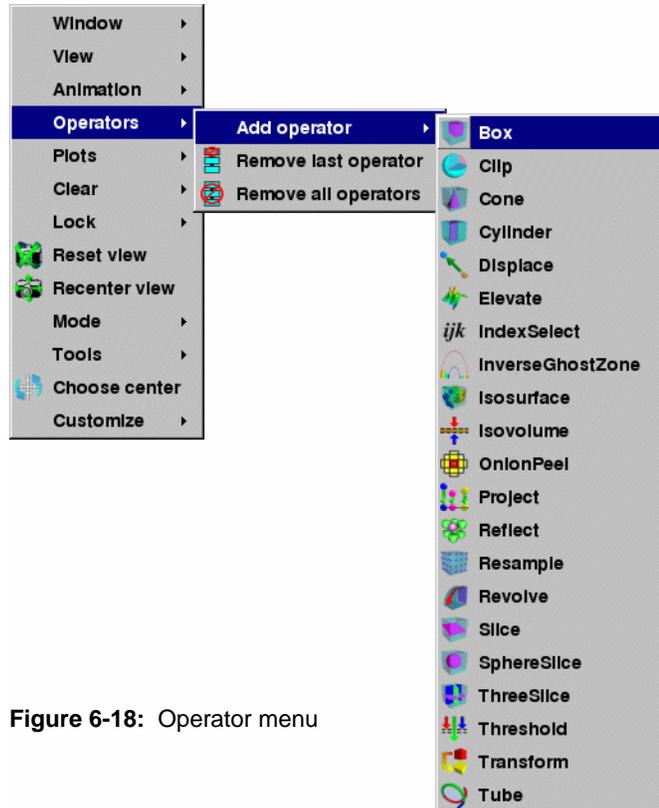


Figure 6-18: Operator menu

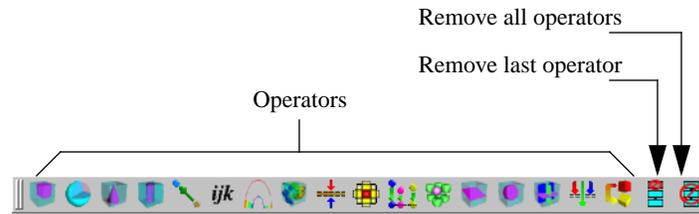


Figure 6-19: Operator toolbar

5.10.1 Adding an operator

The **Operator** menu and toolbar both provide options for you to add new operators. Each operator has its own menu option or icon that adds an operator of that type to the selected plots when you click its menu option or icon.

5.10.2 Removing the last operator

The **Operator** menu and toolbar both have options for you to remove the last operator from a plot. Each plot has a list of applied operators and clicking the **Remove last operator** menu option or icon will remove the last operator from each plot that is selected in the **Plot list**. Plots that have been drawn are regenerated.

5.10.3 Removing all operators

The **Operator** menu and toolbar both have options for you to remove all operators from a plot. Each plot has a list of applied operators and clicking the **Remove all operators** menu option or icon will remove all operators from each plot that is selected in the **Plot list**. Plots that have been drawn are regenerated.

5.11 Lock options

The **Lock menu** and toolbar, both shown in Figure 6-20, allow you to lock certain visualization window attributes so that when you change them, other locked visualization windows also update. Currently, you can lock the view and you can lock time.

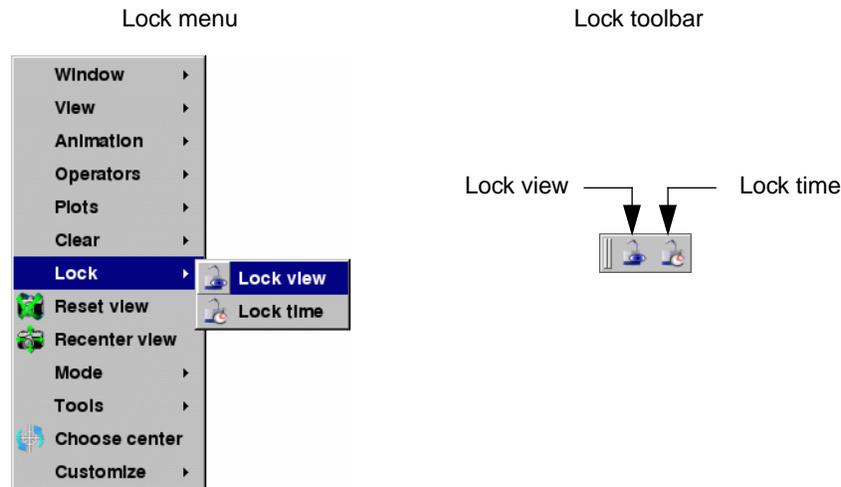


Figure 6-20: Lock menu and toolbar

5.11.1 Locking views

If you have created plots from related databases in multiple visualization windows, you can lock the views for the visualization windows together so as you change the view in one of the visualization windows with a locked view, the other visualization windows with locked views also update to have the same view. There are three types of views in VisIt: curve, 2D, and 3D. If you have 2D plots in a visualization window, the visualization window is considered to be 2D. Locking that 2D visualization window's view will only update other visualization windows that are also 2D and vice-versa. The same is true for curve and 3D views. To lock a visualization window's view, select the **Lock->View** option from the **Main Window's Window** menu or use the visualization menu's **Popup menu** or **Toolbar**.

5.11.2 Locking time

If you have created plots from related databases in multiple visualization windows, you can lock the visualization windows together in time so as you change time in one visualization window, it updates in all other visualization windows that are locked in time. To lock a visualization window in time, select the **Lock->Time** option from the **Main Window's Window** menu.

1.0 Overview

Many scientific databases can be decomposed into subsets that represent only part of the database. It is common to decompose databases into subsets based on Materials (regions), Domains, Groups, AMR patches, AMR levels, Species, Assemblies, etc. Databases are decomposed for a variety of reasons including breaking the database into smaller sized pieces that can be calculated in parallel using a computer with multiple processors. When VisIt is used to visualize the results of a simulation that has been divided into smaller subsets, it is often useful to restrict the size of the database, to increase performance, by using VisIt's **Subset Window**.

2.0 What is a subset?

A subset is a set each of whose elements is an element of an inclusive subset. In other words, a subset is a smaller part of something. In simulation terms, a subset might be the cells containing a particular material in a mesh. A subset might be the part of the simulation mesh that ran on computer processor 10. A subset might be the part of a simulated vehicle that is a tire. In many input database formats, this subset information is encoded in the file or can be inferred from how the file is structured. This leads to an organization of the data where many possible subsets may exist. If there is more than one way to create a subset, you can start asking questions like where in the mesh is there aluminum on only the parts that were computed on processor 2.

3.0 Subset Inclusion Lattice

VisIt relates all possible subsets in a database using what is called a Subset Inclusion Lattice (SIL). Ultimately the subsets in a database are cells that can be grouped into different categories such as material region, domain, patch, refinement level, etc. Each category has some number of possible values when taken together form a collection. A collection lets you group the subsets that have different values but are still part of the same category. For example, the mesh shown in Figure 7-1 is broken down into domain and material categories and there are 3 domain subsets in the domain category. VisIt uses the SIL to remove pieces of a database from a plotted visualization by turning off bottom level subsets that are arrived at through turning off members in various collections or turning off entire collections. When various subsets have been turned off in a SIL, the collective on/off state for each subset is known as a SIL restriction.

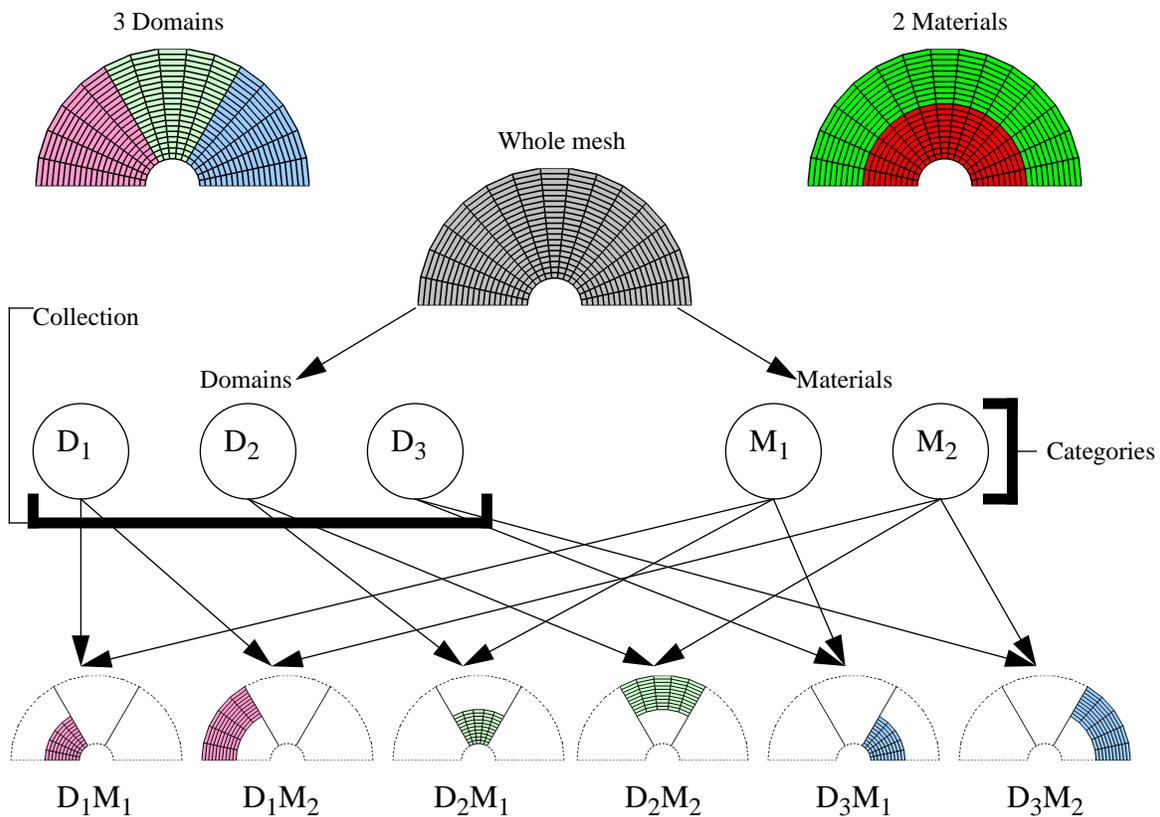


Figure 7-1: Whole mesh divided up into domains and materials

4.0 Using the Subset Window

You can open the **Subset Window**, shown in Figure 7-2, by clicking on the **Subset** option in the **Main Window's Controls** menu or by clicking on the **Subset** icon next to

the name of a plot in the **Plot lists's** plot entries. VisIt's **Subset Window** displays subset relations and provides controls that allow you to alter a plot's SIL restriction by turning off parts of the database.

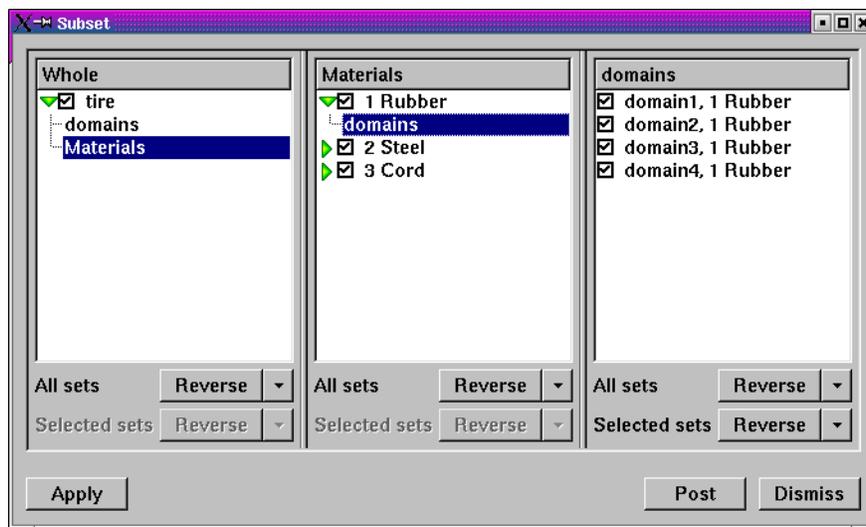


Figure 7-2: Subset Window

The **Subset Window** initially has three panels that display the sets contained in a database though the window can display an unlimited number of panels as subsets are browsed. Each successive panel serves to further subset the database. The leftmost panel contains the top level set for the database being examined. The top level set, which includes all elements in the database, can usually be decomposed in various ways. For example, it can be decomposed by material, processor domain, etc. The various ways in which a database can be decomposed are called subset categories. The subset categories will vary according to the file format in which the database was written as well as the organization and naming conventions used in the database.

4.1 Browsing subsets

To browse the subsets for a database, you must first have created a plot. Once a plot is created and selected, open the **Subset Window**. The left panel in the **Subset Window** contains the database's top level set and may also list some subset categories. Some simple databases lack subset categories thus VisIt cannot create subsets. To start browsing the available subsets, click on one of the subset categories to display the subsets of that category. For instance, clicking on a "Material" subset category will list all of the database's materials in the next panel to the right. The materials are subsets of the top level set and decompose the top level set based on material. Double clicking on a set or clicking on a green turndown arrow lists any subset categories that can be used to further break down the set.

4.2 Changing a SIL restriction

Each set in the **Subset Window** has a small check box next to it that allows you to turn the set on or off. The check box not only displays whether a set is on or off, but it also displays whether or not a set is partially on. When a set is partially on, it means that at least one (but not all) of the subsets that comprise the set is turned on. When a set is partially on, its check box shows a small slash instead of a check or an empty box. Uncheck the check box next to a set name to turn the set off.

Suppose you have a database that contains 4 domains (subsets that correspond to computer processor) numbered 1 through 4. If you want to turn off the subset that is domain1, first click on the “domains” subset category to list the subsets in that category. Next, click the check box next to the subset name “domain1” and click the **Apply** button. The result of this operation, shown in Figure 7-3, removes the “domain1” subset from the visualization. Note that the **Subset Window** “domain1” set’s check box is unchecked and the top level set’s check box has a slash through it to show that some subsets are turned off.

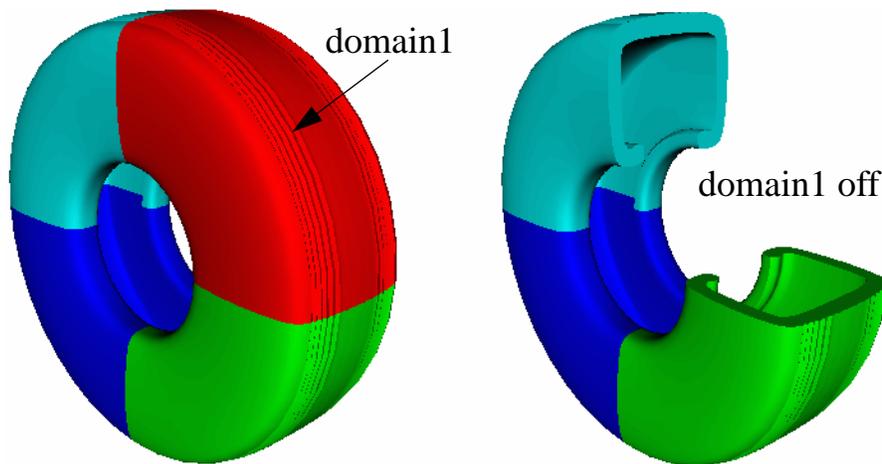


Figure 7-3: Removing one subset.

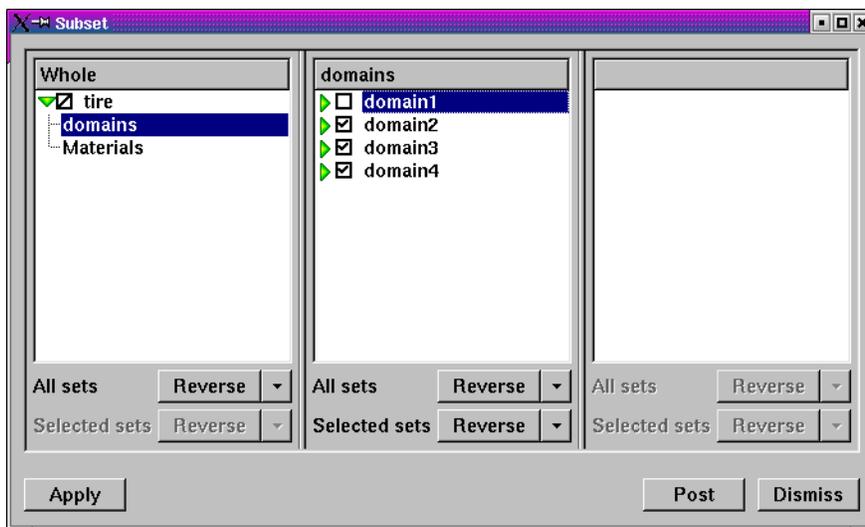


Figure 7-4: Subset Window with one subset removed.

4.3 Creating complex subsets

When visualizing a database, it is often useful to look at subsets created by subsetting more than one category. Suppose you have a database that has two subset categories: “Materials”, and “domains” and that you want to turn off the “domain1” subset but you also want to turn off a material in the “domain4” subset. You can do this by clicking on the “domains” category and then unchecking the “domain1” check box in the second panel. Now, to turn off a material in the “domain4” subset, you click on the “domains” category in the left panel. Next, double-click on the “domain4” subset in the second panel. Select the “Materials” subset category in the second panel to make the third panel list the materials that you can remove from the “domain4” subset. Turning off a couple materials from the list in the third panel will only affect the “domain4” subset. An example of a complex subset is shown in Figure 7-5 and the state of the **Subset window** is shown in Figure 7-6.

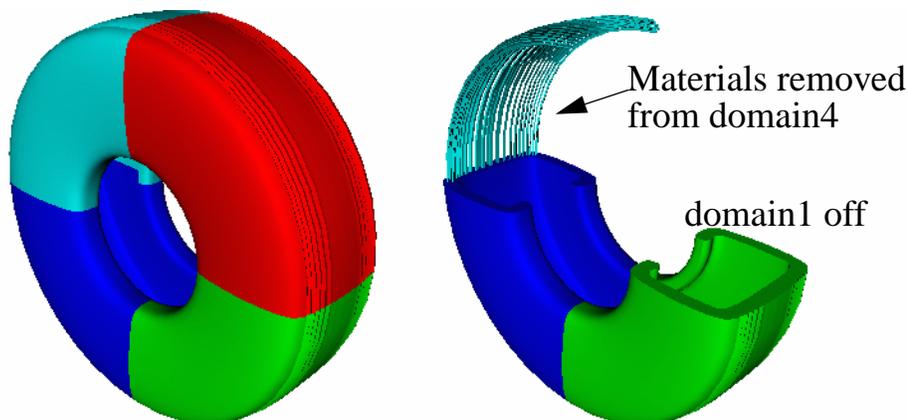


Figure 7-5: Example of a complex subset.

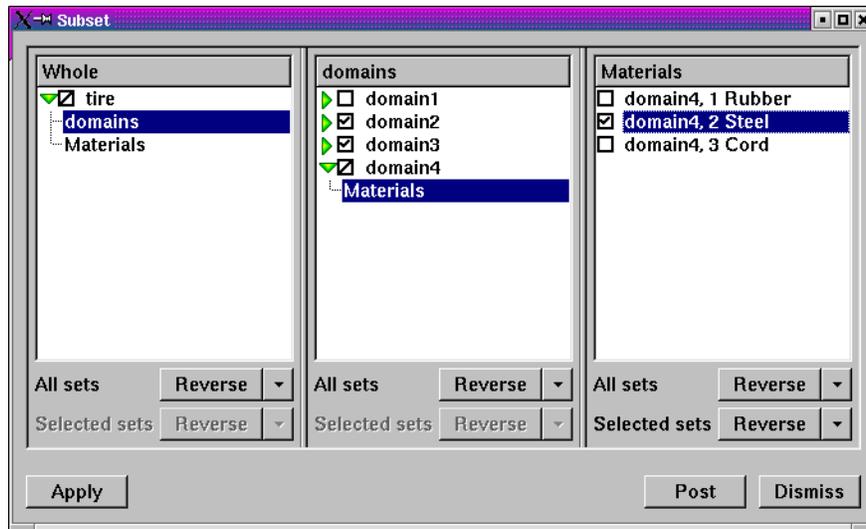


Figure 7-6: Subset Window for complex subset example.

4.4 Turning multiple sets on and off

When databases contain large numbers of subsets, it is convenient to turn groups of them on and off at the same time. You can select ranges of subsets by clicking on the name of a subset using the left mouse button and dragging the mouse up or down to other subsets in the list while still holding down the left mouse button. Alternatively, you can click on a subset to select it and then you can click on another subset while holding down the *Shift* key to select all of the subsets in the middle. Finally, you can select a group of multiple nonconsecutive subsets by holding down the *Ctrl* key while you click on the subsets that you want to select.

Once you have selected a group of subsets, you can use the buttons at the bottom of the pane whose subsets you selected. There are two action buttons at the bottom of each subset pane. The top button applies an action to all of the sets in the pane regardless of how they have been selected. The bottom button applies an action to only the subsets that you have selected. Each action button has three possible actions: Turn on, Turn off, and Reverse. You can change the action for an action button by clicking on the down-arrow button to its right and selecting one of the **Turn on**, **Turn off**, and **Reverse** menu options. When you use an action button that is set to **Turn on**, the appropriate subsets will be turned on. When you use an action button that is set to **Turn off**, the appropriate subsets will be turned off. When you use an action button that is set to **Reverse**, which is the default action, the appropriate subsets' enabled state will be reversed.

5.0 Material Interface Reconstruction

Many databases store out their computational meshes with associated materials. The materials are often used to break meshes into subsets that correspond to physical parts of a model. Materials are commonly stored out as a list of materials and material volume fractions for each cell in the database. If a cell has only one material then it is a clean cell. If a cell has more than one material, it has some fraction of each of the materials and it is known as a mixed cell. The fraction of the material in a cell is accounted for by the material volume fraction. Since only the volume fractions are known, and not any information about how the materials are distributed in the cell, VisIt must make a guess at the location of the boundaries between materials.

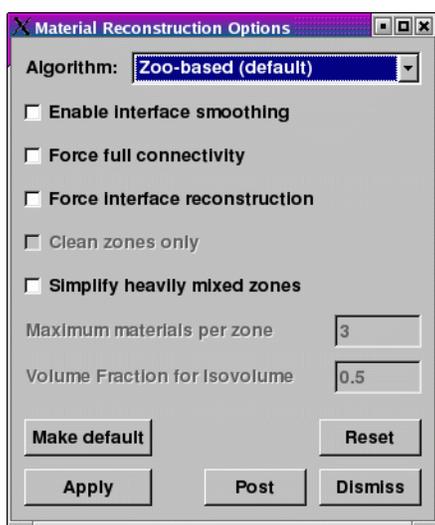


Figure 7-7: Material Reconstruction Options Window

Material interface reconstruction (MIR) is the process of constructing the boundaries between materials, in cells with mixed materials, from the material volume fraction information stored in the database. MIR is not usually needed when you visualize the entire database but when you start to subset the database by removing materials, VisIt must perform MIR to remove only the parts of the database that contain the material to be removed. Without MIR, visualizations containing mixed materials would be very blocky when materials are removed. VisIt's MIR algorithms have several settings, which you can change using the controls in the **Material Reconstruction Options Window** (see Figure 7-7), that influence the appearance of the final plot. To open the **Material Reconstruction Options Window**, click on the **Materials** option in the **Main Window's Controls menu**.

5.1 Choosing a MIR algorithm

VisIt currently provides three MIR algorithms: Tetrahedral, Zoo-based, and Isovolume. Each MIR algorithm reconstructs the interfaces between materials using a different method and one method may work better or worse than another based on the complexities of the input data. You can select your preferred MIR algorithm by choosing from the **Algorithm** combo box in the **Material Reconstruction Options Window**. Note that if you have plots that have already been generated, the new material options will not take effect for those plots unless you clear the plots and redraw them.

The Tetrahedral algorithm breaks up each mixed cell into tetrahedra and computes the interfaces through the original cell by recursively subdividing the tetrahedra until the approximate volume fractions, which determine the amount of material in a cell, are reached. The Tetrahedral MIR algorithm results in a high cell count so it is not often used.

The Zoo-based MIR algorithm breaks up mixed cells into elements based on supported finite elements (tetrahedra, prisms, pyramids, wedges, cubes). The resulting reconstruction results in far fewer cells than other methods while also producing superior material boundaries. The Zoo-based algorithm is the default because of the quality of the material boundaries and because the zoo-based cell representation saves memory and ultimately leads to faster pipeline execution due to the smaller cell count.

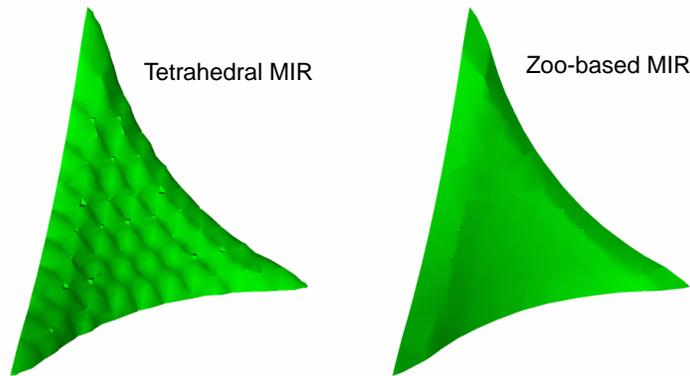


Figure 7-8: Tetrahedral MIR vs. Zoo-based MIR

The Isovolume algorithm computes an isovolume containing portions of cells that contain a user-specified fraction of materials. The Isovolume approach to MIR does not generally produce very good looking results since there are gaps where several materials join. However, the Isovolume algorithm does do a better job than the other two algorithms when it comes to finding cells that contain very small fractions of a certain material when the cells are heavily mixed. If you use the Isovolume MIR algorithm, you can specify the amount of material required to be present before VisIt creates a material interface for a material. The amount of material is specified as a volume fraction in the range $[0,1]$. Specifying smaller values in the **Volume Fraction for Isovolume** text field will find materials that may be omitted by other MIR algorithms.

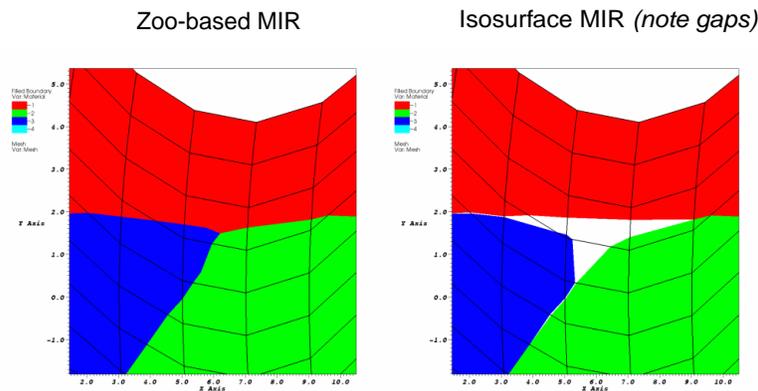


Figure 7-9: Zoo-based MIR vs. Isovolume MIR

5.2 Finding materials with low volume fractions

When mixed cells contain several materials, the Zoo-based MIR algorithm will often omit materials with very small volume fractions, leaving only the materials in the mixed cell that had the highest volume fractions. If you want to plot materials in mixed cells where the volume fraction is very small then you can try using the Isovolume MIR algorithm since it can be used to find materials whose volume fractions are above a user-specified threshold. Figure 7-10 shows an example of a dataset containing five mixed materials where the first four mixed materials are roughly equal in the amount of area that they occupy. The fifth material has a volume fraction that never exceeds 0.08 so it is omitted by the Zoo-based MIR algorithm due to its comparatively low volume fraction. To ensure that VisIt plots the fifth material, the Isosurface MIR algorithm is used with a **Volume Fraction for Isovolume** setting of 0.02. Using the Isovolume MIR algorithm with a low **Volume Fraction for Isovolume** value can find materials that have been distributed into many heavily mixed cells.

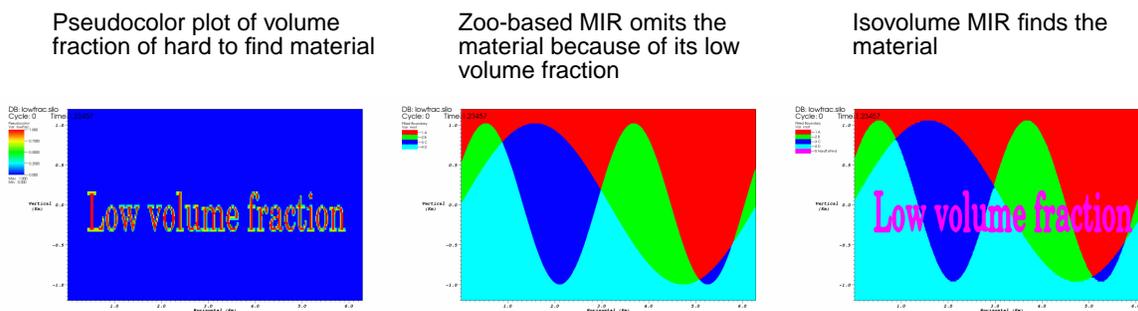


Figure 7-10: Materials with low volume fractions can be found with the Isosurface MIR algorithm

5.3 Simplifying heavily mixed cells

VisIt provides the **Simplify heavily mixed cells** check box in the **Material Reconstruction Options Window** so you can tell VisIt to throw away information materials that have low volume fractions. When you tell VisIt to omit these materials, VisIt will use less memory and will also finish MIR faster because fewer materials have to be considered. The **Simplify heavily mixed cells** check box is especially useful for databases where most of the cells are mixed or where there are many cells that contain tens of materials. When you tell VisIt to simplify heavily mixed cells, you can tell VisIt how many of the top materials to keep from each cell by entering a new number of materials into the **Maximum materials per zone** text field. By keeping the N top materials, VisIt will be sure to preserve the features that are contributed by the most dominant materials.

5.4 Smoother material boundary interfaces

VisIt's material interface reconstruction algorithm sometimes produces small, pointy outcroppings on reconstructed material boundaries next to where clean cells are located. Since these are often distracting features when looking at a visualization, VisIt provides an interface smoothing option that allows materials to bleed a little bit into clean cells to improve how they look when their material boundary is reconstructed. Figure 7-11 shows a plot that has not been smoothed next to a plot that has been smoothed. To enable interface smoothing, check the **Enable interface smoothing** check box. Note that changing this setting will not affect plots that have already been generated. If you want to make your current plots regenerate with smoother interfaces, you must also clear them out of the visualization window by choosing the **Plots** option from the **Clear** submenu located in the **Main Window's Windows** menu.

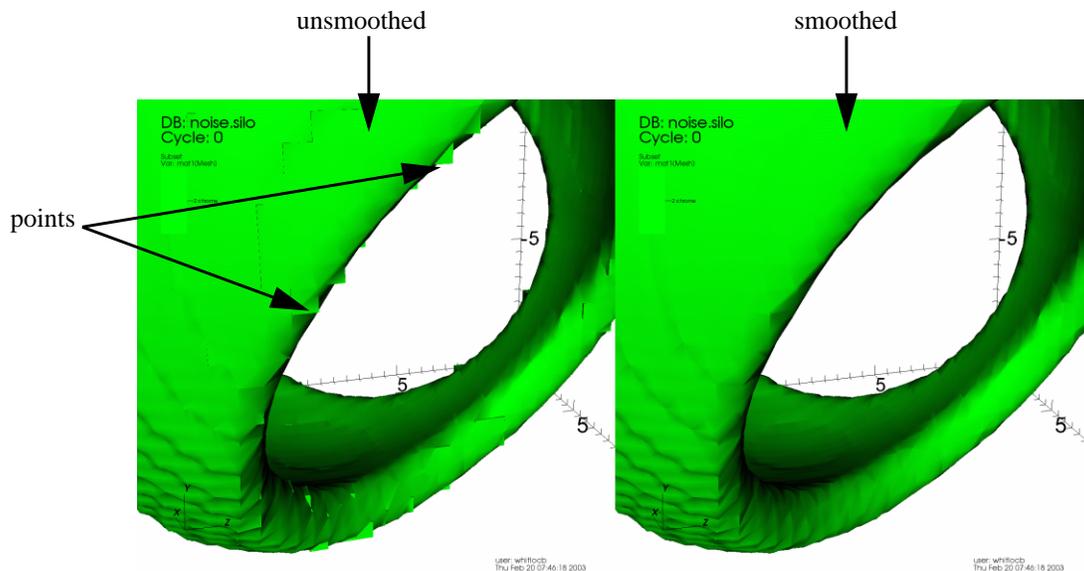


Figure 7-11: Effect of material interface smoothing

5.5 Forcing material interface reconstruction

VisIt tries to minimize the amount of work that it must do to generate a plot so that it can be done quickly. Sometimes databases have variable information for each material in a cell instead of just having a single value for each cell or node. Because the variable is defined for each material in the cell, these variables are known as mixed variables. VisIt tends to just plot the value for the entire cell since it is more work to go through the material interface reconstruction (MIR) stage, which is usually only done when removing material subsets but is required to plot mixed variables correctly. You can force VisIt to always do MIR by checking the **Force interface reconstruction** check box. This will make mixed variables plot correctly even when you are not removing any material subsets.

5.6 Mixed variables

Some simulations write out multiple scalar values for cells that contain mixed materials so each material in the cell can have its own scalar value. Once a cell has undergone MIR, it is split into multiple cells if the original cell contained more than one material. Each split cell gets its corresponding scalar value from the original mixed variable data. The resulting plot can then display each split cell's actual value, taking into account the material boundaries. Suppose you are simulating the interaction between hot lava and ice and you have a material interface that happens to cross in the middle of a cell. Obviously each material in the cell has its own temperature. Plotting mixed variables allows the visualization to more faithfully depict the material boundaries while preserving the actual data so the multiple mix values do not have to be averaged in the cell (see Figure 7-12). Note that VisIt does not use mixed variable values for variables that have them unless the **Force interface reconstruction** check box is enabled because most scalar fields are not mixed variables and automatically performing MIR can be expensive. If your scalars are mixed variables and you want to visualize them as such, be sure to enable the **Force interface reconstruction** check box.

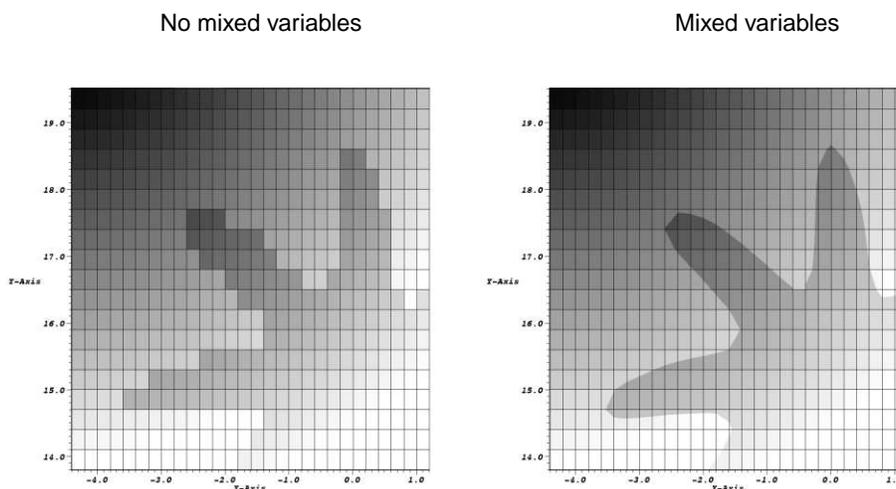


Figure 7-12: Mixed variables can improve a visualization

6.0 Species

VisIt adds species, which are components of materials, to the SIL when they are available. Air is a common material in simulations since many things in the real world are surrounded by air. The chemical composition of air on Earth is roughly 78% Nitrogen, 21% oxygen, 1% Argon. You can say that if air is a material then it has species: Nitrogen, Oxygen, and Argon with mass fractions 78%, 21%, 2%, respectively. Suppose one of the calculated quantities in a database with the afore-mentioned air material is atmospheric

temperature. Now suppose that we are examining one cell that contains only the air material from the database and its atmospheric temperature is 100 degrees Fahrenheit. If we wanted to know how much the Nitrogen contributed to the atmospheric temperature, we could multiply its concentration of 78% times the 100 degrees Fahrenheit to yield: 78 degrees Fahrenheit. Species are often used to track chemical composition of materials and their effects on various calculated quantities.

When species are available, VisIt creates a scalar variable called Species and it is available in the variable menus for each plot that can accept scalar variables. The Species variable is a cell-centered scalar field defined over the whole mesh. When all species are turned on, the Species variable has the value of 1.0 over the entire mesh. When species are turned off, the Species variable is set to 1.0 minus the mass fraction of the species that was turned off. Using the previous example, if we plotted the Species variable and then turned off the air material's Nitrogen species, we would be left with only Oxygen's 21% and Argon's 1% so the species variable would be reduced to 22% or 0.22. When species are turned off, the amount of mass left to be multiplied by the plotted variable drops so the plotted variable's value in turn drops.

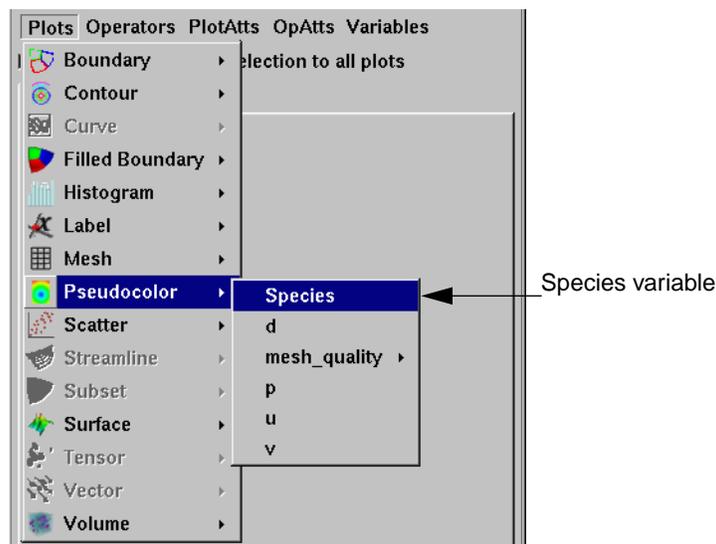


Figure 7-13: Species variable

VisIt adds species to the SIL as a category that contains the various chemical constituents for all materials that have species. Since species are handled using the SIL, you can use VisIt's Subset Window to turn off species. Turning off species has quite a different effect than turning off entire materials. When materials are turned off, they no longer appear in the visualization. When species are turned off, no parts of the visualization disappear but the plotted data values may change due to drops in the Species variable.

6.1 Plotting species

VisIt provides the *Species* scalar variable so you can plot or create expressions that involve species. If you create a Pseudocolor plot of the Species variable, the resulting plot will have a constant value of 1.0 over the entire mesh because when no species have been removed, they all sum to 1.0. Once you begin removing species by turning off species subsets in the **Subset Window**, the plotted value of Species changes, causing plots that use it to also change. If you remove all but one species, the plots that use the Species variable will show zero for all areas that do not contain the one selected species (see Figure 7-14). For example, if you had air for a material and then you removed every species except for oxygen, the plots that use the Species variable would show zero for every place that had no oxygen.

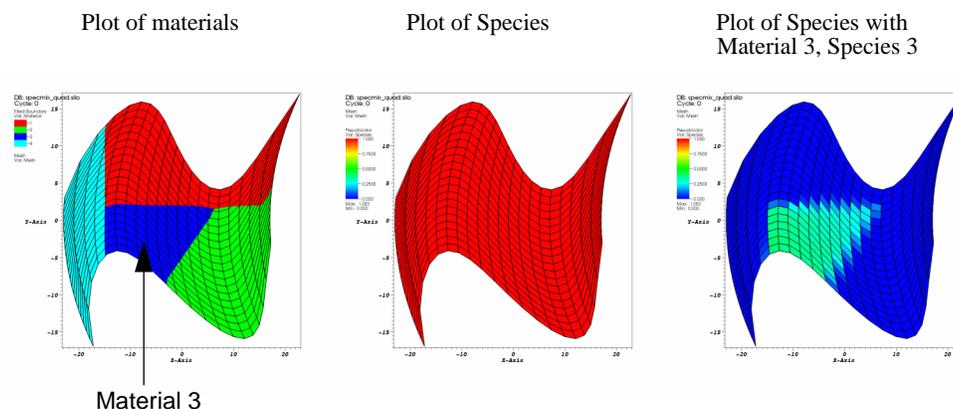


Figure 7-14: Plots of materials and species

6.2 Turning off species

VisIt adds species information to the SIL as new subsets under a category called: Species. Since species are part of the SIL, you can use the **Subset Window** (see Figure 7-15) to turn off species. To access the list of species, select the Species category under the whole mesh. Once the Species category is clicked, the second pane in the **Subset Window** is populated with the species for all materials. Turn off the species that you don't want to look at by clicking off the check box next to the name of the species subset. When you

apply your changes, the values for the Species variable are recalculated to include only the mass fractions for the species that are still turned on.

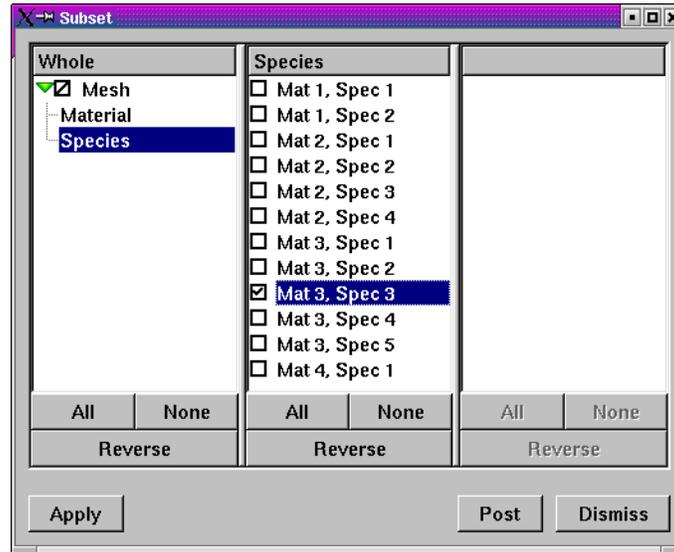


Figure 7-15: Turning off species in the Subset Window

1.0 Overview

Simulation data must often be compared to experimental data so VisIt provides a number of features that allow quantitative information to be extracted from simulation databases. This chapter explains how to visualize derived variables created with expressions and query information about a database. This chapter also explains VisIt's Pick and Lineout capabilities which allow quantitative data to be obtained from the visualization.

2.0 Expressions

Scientific simulations often keep track of several dozen variables as they run. However, only a small subset of those variables are usually written to a simulation database to save disk space. Sometimes variables can be derived from other variables using a variable expression. VisIt provides variable expressions to allow scientists to create derived variables using variables that are stored in the database. Expressions are extremely powerful because they allow you to analyze new data without necessarily having to rerun a simulation. Variables created using expressions behave just like variables stored in a database; they appear in the plot menu and can be visualized using VisIt's plots.

2.1 Expression Window

VisIt provides an **Expression Window**, shown in Figure 8-1, that allows you to create new variables that can be used in visualizations. You can open the **Expression Window** by clicking on the **Expressions** option in the **Main Window's Controls** menu. The **Expression Window** is divided vertically into two main areas: **Expression list** and **Definitions**. The **Expression list** contains the list of expressions. The **Definitions**

area displays the definition of the expression that is highlighted in the **Expression list** and provides controls to edit the expression definition.

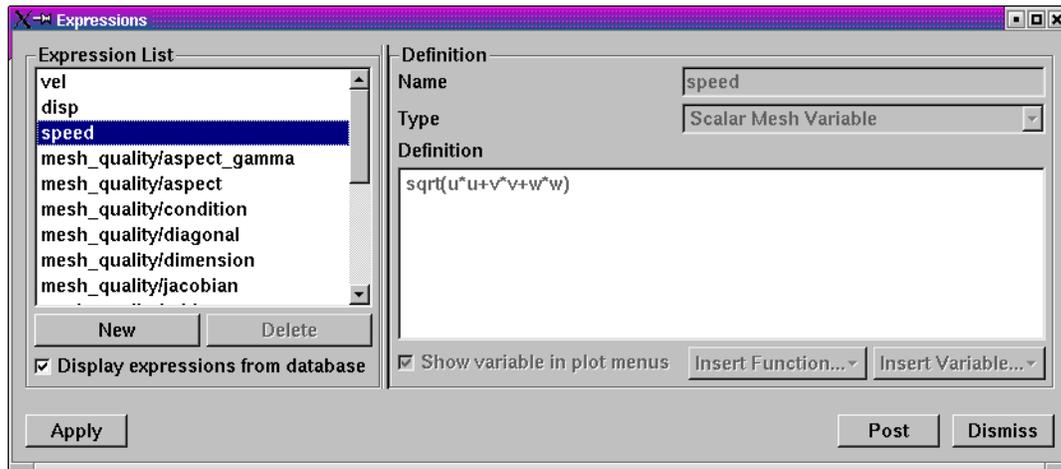


Figure 8-1: Expression Window

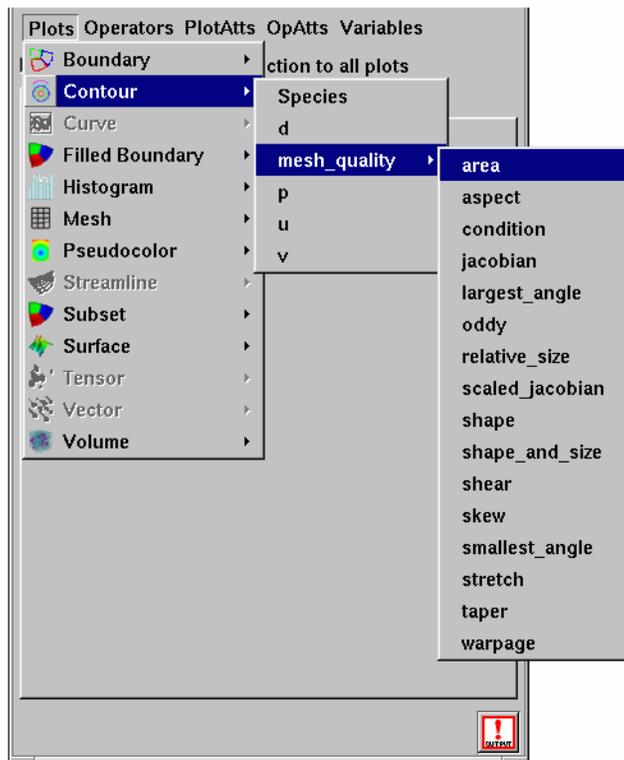


Figure 8-2: Mesh quality expressions

open database contains meshes that are unstructured or curvilinear, VisIt adds special mesh quality expressions to the expression list to let you more easily plot metrics of mesh quality such as the volume of cells or how much cells are skewed. The mesh quality expressions are primarily used to evaluate the fitness of computational meshes and VisIt

In addition to displaying expressions that you create yourself, the **Expression Window** displays expressions that were written to the database or expressions that were added by one of VisIt's database reader plugins. Expressions that came from a database change when you change open databases but expressions that you defined yourself remain in the window regardless which database is open. Expressions that came from a database are not usually shown in the **Expression list** by default but you can choose to show them by clicking on the **Display expressions from database** check box.

There is one special case when VisIt adds expressions to the expression list that were not in the database and were not added by a database reader plugin: mesh quality expressions. When the

adds them to the expression list to make them more accessible to users since they are so commonly used. Figure 8-2 shows the how the mesh quality expressions appear in the variable list for plots that accept scalar variables.

2.1.1 Creating a new expression

You can create a new expression by clicking on the **Expression Window's New** button. When you click on the **New** button, VisIt adds a new expression and shows its new, empty definition in the **Definitions** area. The initial name for a new expression is “unnamed” followed by some integer suffix. As you type a new name for the expression into the **Name** text field, the expression's name in the **Expression list** will update.

Each expression has an expression type that determines the variable menu in which the new variable appears. The available expression types are: *Scalar Mesh Variable*, *Vector Mesh Variable*, *Mesh variable*, *Tensor variable*, *Symmetric Tensor mesh*. Since the expression type determines the menu in which the variable appears, it also determines the plots that can operate on the variable. Scalar mesh variables and species variables can be used in the Contour, Pseudocolor, and Volume plots. Vector mesh variables are used in the Streamline and Vector plots. Tensor mesh variables are used in the Tensor plot.

To edit an expression's actual definition, you can type a new expression comprised of constants, variable names, and other VisIt expressions into the **Definition** text field. The expression definition can span multiple lines as the VisIt expression parser ignores whitespace. For a complete list of VisIt's built-in expressions, refer to page 183. You can also use the **Insert Function...** menu, shown in Figure 8-3, to insert any of VisIt's built-in expressions directly into the expression definition. The list of built-in expressions divided into certain categories as evidenced by the structure of the **Insert Function...** menu.

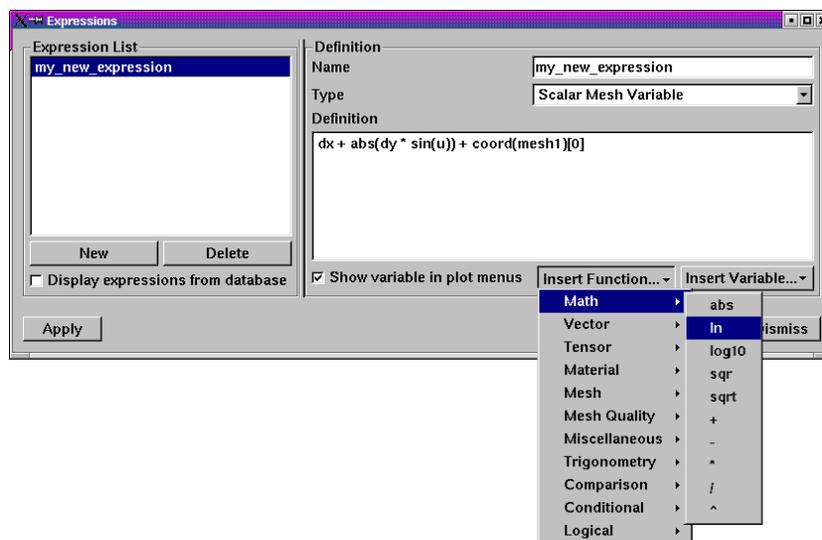


Figure 8-3: Expression Window's Insert Function... menu

In addition to the **Insert Function...** menu, which lets you insert built-in functions into the expression definition, VisIt's **Expression Window** provides an **Insert Variable...** menu that allows you to insert variables for the active database into the expression definition. The **Insert Variable...** menu, shown in Figure 8-4, is broken up into Scalars, Vectors, Meshes, etc. and has the available variables under the appropriate heading so they are easy to find.

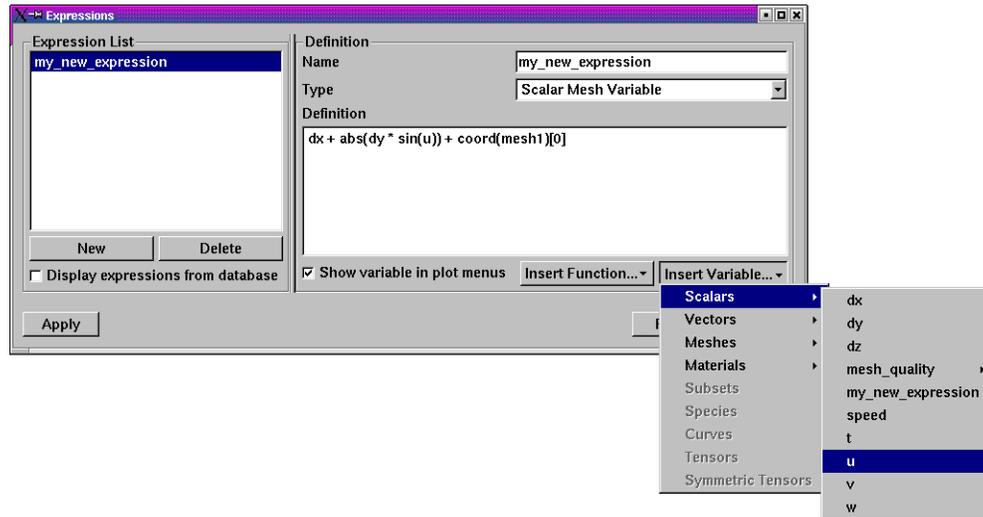


Figure 8-4: Expression Window's Insert Variable... menu

Some variables can only be expressed as very complex expressions containing several intermediate subexpressions that are only used to simplify the overall expression definition. These types of subexpressions are seldom visualized on their own. If you want to prevent them from being added to the **Plot** menu, turn off the **Show variable in plot menus** check box.

2.1.2 Deleting an expression

You can delete an expression by clicking on it in the **Expression list** and then clicking on the **Delete** button. Deleting an expression removes it from the list of defined expressions and will cause unresolved references for any other expression that uses the deleted expression. If a plot uses an expression with unresolved references, VisIt will not be able to generate it until you resolve the reference or change the active plot variable.

2.2 Expression grammar

VisIt allows expressions to be written using a host of unary and binary math operators as well as built-in and user-defined functions. VisIt's expressions follow C-language syntax, although there are a few differences. The following paragraphs detail the syntax of VisIt expressions.

2.2.1 Math operators

These include use of +, -, *, /, ^ as addition, subtraction, multiplication, division, and exponentiation as infix operators, as well as the unary minus, in their normal precedence and associativity. Parentheses may be used as well to force a desired associativity.

Examples: $a+b^c$ $(a+b)*c$

2.2.2 Constants

Scalar constants include floating point numbers and integers, as well as booleans (true, false, on, off) and strings.

Examples: $3e4$ 10 "mauve" true false

2.2.3 Vectors

Expressions can be grouped into two or three dimensional vector variables using curly braces.

Examples: {xc, yc} {0,0,1}

2.2.4 Lists

Lists are used to specify multiple items or ranges, using colons to create ranges of integers, possibly with strides, or using comma-separated lists of integers, integer ranges, floating points numbers, or strings.

Examples: [1,3,2] [1:2, 10:20:5, 22] [silver, gold] [1.1, 2.5, 3.9] [level1, level2]

2.2.5 Identifiers

Identifiers include function names, defined variable and function names, and file variable names. They may include alphabetic characters, numeric characters, and underscores in any order. Identifiers should have at least one non-numeric character so that they are not confused with integers, and they should not look identical to floating point numbers such as 1e6.

Examples: density x y z 3d_mesh

2.2.6 Functions

These are used for built in functions, but they may also be used for functions/macros defined by the user. They take specific types and numbers of arguments within the parentheses, separated by commas. Some functions may accept named arguments in the form *identifier=value*.

Examples: $\sin(\pi / 2)$ $\text{cross}(\text{vec1}, \{0,0,1\})$ $\text{my_xform}(\text{mesh1})$ $\text{subselect}(\text{materials}=[a,b])$

2.2.7 Database variables

These are like identifiers, but may also include periods, plus, and minus characters. A normal identifier will map to a file variable when it is not defined as another expression. To force variables that look like integers or floating point numbers to be interpreted as variable names, or to force variable names which are defined by another expression to map to a variable in a file, they should be enclosed with `<` and `>`, the left and right carats/angle brackets. Note that quotation marks will cause them to be interpreted as string constants, not variable names. In addition, variables in files may be in directories within a file, so they may include slashes in a path when in angle brackets.

Examples: density $\langle \text{pressure} \rangle$ $\langle a.001 \rangle$ $\langle a.002 \rangle$ $\langle \text{domain1/density} \rangle$

2.2.8 Databases

A database specification looks similar to a database variable contained in angle brackets, but it is followed by a colon before the closing angle bracket, and it may also contain extra information. A database specification includes a file specification possibly followed a machine name, a time specification by itself, or a file/machine specification followed by a time specification. A file specification is just a file name with a path if needed. A machine specification is an at-sign `@` followed by a host name. A time specification looks much like a list in that it contains integer numbers or ranges, or floating point numbers, separated by commas and enclosed in square brackets. However, it may also be followed by a letter `c`, `t`, or `i` to specify if the time specification refers to cycles, times, or indices, respectively. If no letter is specified, then the parser guesses that integers refer to cycles, floating point numbers refer to times. There is also an alternative to force indices which is the pound sign `#` after the opening square bracket.

Examples: $\langle \text{/dir/file} \rangle$ $\langle \text{file@host.gov} \rangle$ $\langle \text{[# 0:10]} \rangle$ $\langle \text{file[1.234]} \rangle$ $\langle \text{file[000, 023, 047]} \rangle$ $\langle \text{file[10]c} \rangle$

2.2.9 Qualified file variables

Just like variables may be in directories within a file, they may also be in other timesteps within the same database, within other databases, and even within databases on other machines. To specify where a variable is located, use the angle brackets again, and prefix the variable name with a database specification, using the colon after the database specification as a delimiter.

Examples: $\langle \text{file:var} \rangle$ $\langle \text{/dir/file:/domain/var} \rangle$ $\langle \text{file@192.168.1.1:/var} \rangle$ $\langle \text{[#0]:zerocyclevar} \rangle$

2.3 Built-in expressions

The following table lists built-in expressions that can be used to create more advanced expressions. Unless otherwise noted in the description, each expression takes scalar variables as its arguments.

Expression	Meaning	Usage
{ }	Associate a list of expressions into a single variable. Usually used for defining a vector or tensor variable.	$\{ \text{expr}_1, \text{expr}_2 [, \dots] \}$ expr_i can be an expression of variables, another expression, or a constant defined over a mesh. Examples: $\text{vector} = \{a,b,c\}$ $\text{tensor} = \{ \{a,b,c\}, \{d,e,f\}, \{g,h,i\} \}$
[]	Access a subscript of a vector or tensor variable.	$\text{expr}[\text{index}]$ expr must be an expression that evaluates to a vector or tensor variable and index must be an integer greater than or equal to zero and less than the number of components in the variable being indexed. Example: $\text{velocity}[0]$
()	Associative parenthesis.	Group mathematical operations to influence the order in which expressions are evaluated.
-	Unary negation	$-\text{expr}$ expr is any expression that evaluates to a scalar or vector field defined on a mesh. Example: $-\text{charge}$
-	Subtraction	$\text{expr}_1 - \text{expr}_2$ expressions involving subtraction can be database variables, constants, or other expressions. Example: $a - b$
+	Addition	$\text{expr}_1 + \text{expr}_2$ Expressions involving addition can be database variables, constants, or other expressions. Example: $a + b$

Expression	Meaning	Usage
*	Multiplication	$\text{expr}_1 * \text{expr}_2$ Expressions involving multiplication can be database variables, constants, or other expressions. Multiplication is most often used with two scalar inputs but one of the inputs to the multiplication operator can be a vector if you want to scale a vector using a scalar or a constant. Example: $a * b$
/	Division	$\text{expr}_1 / \text{expr}_2$ Expressions involving division can be database variables, constants, or other expressions. Division is most often used with two scalar inputs but the first input can be a vector if you want to scale a vector using a scalar or a constant. Example: a / b
^	Exponentiation	$\text{expr}_1 ^ \text{expr}_2$ Expressions involving exponentiation can be database variables, constants, or other expressions. Example: $a ^ b$ (a^b)
abs	Absolute value	$\text{abs}(\text{expr})$ expr can be an expression of database variables, constants, or other scalar expressions.
acos	Arccosine	$\text{acos}(\text{expr})$ expr can be an expression of database variables, constants, or other scalar expressions. The acos expression returns an angle in radians. Example: $\text{angle} = \text{acos}(\text{value})$

Expression	Meaning	Usage
and	Logical and	<p><code>and(expr₁, expr₂)</code></p> <p>The logical and function returns a value of 1 (true) if both scalar input expressions <code>expr₁</code> and <code>expr₂</code> are not equal to zero. Otherwise, the logical and function returns zero. The logical and function is often used with other conditionals such as the <i>if</i> expression.</p> <p>Example: <code>if(and(gt(pressure, 2.0), lt(pressure, 4.0)), pressure, 0.0)</code></p> <p>Meaning: if pressure is in the range (2.0, 4.0) then use the value for pressure. Otherwise, use zero.</p>
area	Cell Area	<p><code>area(expr)</code></p> <p><code>expr</code> must evaluate to a 2D mesh expression so VisIt can calculate the area of each 2D cell in the mesh and returns scalar values.</p> <p>Example: <code>density = mass / area(hydro_mesh2d)</code></p>
array_compose	Array compose	<p><code>array_compose(expr₁, expr₂, ..., expr_N)</code></p> <p>Each argument to the <code>array_compose</code> expression, <code>expr_i</code>, must evaluate to a scalar expression and all of the input expressions must have the same centering. The <code>array_compose</code> expression creates a new array variable from the input expressions. Array variables are collections of scalar variables that are commonly used with certain plots to display the contents of multiple variables simultaneously. For example, the Label plot can display the values in an array variable.</p> <p>Example: <code>array_compose(density, pressure, velocity[0], velocity[1], velocity[2])</code></p>

Expression	Meaning	Usage
array_decompose	Array decompose	<p>array_decompose(expr1, expr2)</p> <p>expr1 must evaluate to an array variable. expr2 must evaluate to a number between 0 and the number of scalar components in the array variable minus 1. This expression isolates one scalar variable from an array variable.</p> <p>Example: array_decompose(array, 0)</p>
asin	Arcsine	<p>asin(expr)</p> <p>expr can be an expression of database variables, constants, or other scalar expressions. The asin expression returns an angle in radians.</p> <p>Example: angle = asin(value)</p>
aspect	Cell aspect ratio	<p>aspect(expr)</p> <p>expr must be an expression that evaluates to a mesh. The aspect function computes the aspect ratio for each cell in the mesh and returns scalar values. Cells with high aspect ratios (long and skinny) are often considered to be less fit than more squat cells.</p> <p>Example: aspect(hydro_mesh)</p>
aspect_gamma	Cell aspect gamma	<p>aspect_gamma(expr)</p> <p>expr must be an expression that evaluates to a mesh.</p> <p>Example: aspect_gamma(hydro_mesh)</p>
atan	Arctangent	<p>atan(expr)</p> <p>expr can be an expression of database variables, constants, or other scalar expression variables. The atan expression returns an angle in radians.</p> <p>Example: angle = atan(value)</p>

Expression	Meaning	Usage
ceil	Ceiling	<p>ceil(expr)</p> <p>expr must evaluate to a scalar expression. The ceil expression calculates the ceiling function. The ceiling is defined to be smallest integer that is bigger than the current value.</p> <p>Example: ceil(pressure)</p>
condition	Condition number	<p>condition(expr)</p> <p>expr must be an expression that evaluates to a mesh. The condition expression returns the maximum condition number of the Jacobian matrix at the nodes of each cell in the mesh, resulting in a zone-centered, scalar expression.</p> <p>Example: condition(hydro_mesh)</p>
conn_cmfe	Connectivity-based common mesh field evaluation	<p>conn_cmfe(expr1, expr2)</p> <p>expr1 is an expression commonly from a different database and it contains the field that will be mapped onto the mesh expression that is created by expr2. The conn_cmfe expression is used to map variables from one mesh onto another mesh. The mesh that donates the field being mapped can be a different time state of the same database or it can be in a different file than the destination mesh. You can use the conn_cmfe expression to compare the results from different simulation runs. For more information on using the conn_cmfe expression, see page 292.</p> <p>Examples:</p> <p><i>Map wave0020.silo's pressure variable the current mesh.</i></p> <pre>conn_cmfe(<wave0020.silo:pressure>, quadmesh)</pre> <p><i>Subtract <mesh/ireg> variable from the last time state from the current time state.</i></p> <pre><mesh/ireg> - conn_cmfe(<allinone00.pdb[-1]id:mesh/ireg>, mesh)</pre>

Expression	Meaning	Usage
conservative_smooth	Conservative smooth	<p>conservative_smooth(expr)</p> <p>expr must be an expression that evaluates to a scalar field. This expression applies the conservative smooth filter (an image processing filter) to a scalar field. The filter only considers a value and its immediate neighbors (26 neighbors in three dimensions, 8 in two dimensions). The filter only works on structured meshes. When performing a conservative smooth operation, a value is only replaced if it is outside the range of its immediate neighbors. If so, it is replaced with the closest value from its immediate neighbors.</p> <p>Example: conservative_smooth(intensity)</p>
coord, coords	Mesh coordinates	<p>coord(expr)</p> <p>expr must be an expression that evaluates to a mesh. The coord expression extracts the coordinate fields from a mesh and returns them as a vector variable with 3 components. The resulting vector can be used to extract the x,y,z components of the mesh's coordinate field.</p> <p>Example: x = coord(Mesh)[0]</p>
cos	Cosine	<p>cos(expr)</p> <p>Compute the cosine of an angle in radians. Expr can be an expression of database variables, constants, or other scalar expression that evaluate to an angle in radians. The cos expression returns a scalar expression.</p> <p>Example: x = radius * cos(angle)</p>
cross	Vector cross product	<p>cross(expr₁, expr₂)</p> <p>The cross expression returns the vector cross product of the expr₁ and expr₂ vector expressions.</p> <p>Example: viewvec = cross(right_vec, up_vec)</p>

Expression	Meaning	Usage
curl	Curl	<p>curl(expr)</p> <p>The curl expression calculates the curl of the input expression, which must evaluate to a vector field. The result is also a vector unless the input data is 2D. When the input data set is 2D, the resulting curl vector always is (0,0,V) so the curl expression instead returns the scalar V. The curl expression must be declared as a vector for 3D data sets, but as a scalar for 2D data sets.</p> <p>Example: curl(vecfield)</p>
cylindrical_radius	Cylindrical radius	<p>cylindrical_radius(expr)</p> <p>expr must be an expression that evaluates to a mesh. The cylindrical radius expression converts the coordinates of the input mesh from cartesian coordinates to cylindrical coordinates and returns the radius component of the cylindrical coordinates.</p> <p>Example: cylindrical_radius(hydro_mesh)</p>
cylindrical_theta	Cylindrical theta	<p>cylindrical_theta(expr)</p> <p>expr must be an expression that evaluates to a mesh. The cylindrical theta expression converts the coordinates of the input mesh from cartesian coordinates to cylindrical coordinates and then returns the theta component of the cylindrical coordinates.</p> <p>Example: cylindrical_theta(hydro_mesh)</p>
deg2rad	Degrees to radians	<p>deg2rad(expr)</p> <p>The deg2rad expression converts the input scalar expression, which is assumed to be in degrees, to radians.</p> <p>Example: angle_rad = deg2rad(angle)</p>
degree	Mesh degree	<p>degree(expr)</p> <p>expr must be an expression that evaluates to a mesh. The degree expression creates a nodal scalar field that contains the number of cells that use each node.</p> <p>Example: degree(Mesh)</p>

Expression	Meaning	Usage
det, determinant	Matrix determinant	<p>determinant(expr)</p> <p>expr must evaluate to a 3x3 tensor. The determinant expression computes the determinant of a 3x3 matrix and returns the scalar result.</p> <p>Expression: <code>detA = determinant(A)</code></p>
diagonal	Diagonal ratio	<p>diagonal(expr)</p> <p>expr must be an expression that evaluates to a mesh. The diagonal expression computes the minimum and maximum diagonals for 3D hexahedral cells and returns the minimum diagonal length divided by the maximum diagonal length. Cells that have equal aspect ratios will have diagonal ratios of 1.0 while more oblong cells will have differing values. For cells that are not hexahedral, the diagonal expression returns -1.</p> <p>Example: <code>diagonal(Mesh)</code></p>
dimension	Pronto-specific length of stable time step	<p>dimension(expr)</p> <p>expr must be an expression that evaluates to a mesh. The dimension expression computes the characteristic length for stable time step calculation in the Pronto simulation code.</p> <p>Example: <code>dimension(Mesh)</code></p>
divergence	Divergence	<p>divergence(expr)</p> <p>The divergence filter calculates the divergence, which is the tendency of a fluid to spread out. The input expression must evaluate to a vector field. Divergence results in a scalar field.</p> <p>Example: <code>divergence(vec)</code></p>
dot	Vector dot product	<p>dot(expr₁, expr₂)</p> <p>The dot expression takes 2 vector inputs expr₁ and expr₂ and computes their vector dot product and returns the resulting scalar expression.</p> <p>Example: <code>dot(vector1, vector2)</code></p>

Expression	Meaning	Usage
effective_tensor	Effective tensor	<p>effective_tensor(expr)</p> <p>expr must evaluate to a tensor expression. The effective_tensor expression determines the effective part of a 3x3 tensor and returns the resulting scalar expression.</p> <p>Example: effective_tensor(tensor1)</p>
eigenvalue	Eigenvalue	<p>eigenvalue(expr)</p> <p>expr must evaluate to a 3x3 tensor. The eigenvalue expression returns the eigenvalues of a matrix as a scalar expression.</p> <p>Example: scalars = eigenvalue(tensor)</p>
eigenvector	Eigenvector	<p>eigenvector(expr)</p> <p>expr must evaluate to a 3x3 tensor. The eigenvector expression returns the eigenvectors of a matrix as a vector expression.</p> <p>Example: vectors = eigenvector(tensor)</p>
eq, equal, equals	Equality	<p>equal(expr₁, expr₂)</p> <p>The equal expression returns a value of 1 (true) if both input scalar expressions expr₁ and expr₂ are equal. Otherwise, the equal expression returns zero. The equal expression is often used with other conditionals such as the if expression.</p> <p>Example: if(eq(density, 1.0), density*2.0, 0.0)</p>
external_node	External node	<p>external_node(expr)</p> <p>The external_node expression marks every node that is incident to an external face as '1', every node incident to only internal faces as '0'. Expr must be a mesh. If expr is a two-dimensional mesh, then the expression returns '1' if a node is incident to an external edge.</p> <p>Example: external_node(hydro_mesh)</p>

Expression	Meaning	Usage
floor	Floor	<p>floor(expr)</p> <p>expr must be an expression that evaluates to a scalar expression. The floor expression takes the floor of its expression argument, which is defined to be biggest integer that is smaller than the current value.</p> <p>Example: floor(pressure)</p>
ge, gte	Greater than or equal	<p>ge(expr₁, expr₂)</p> <p>The ge expression returns a value of 1 (true) if expr₁ >= expr₂. Both input expressions must be scalar expressions.</p> <p>Example: if(ge(pressure, density), 1.0, 0.0)</p>
global_nodeid	Global node numbers	<p>global_nodeid(expr)</p> <p>The global_nodeid expression returns a scalar field containing the global node numbers for a domain-decomposed mesh so that each node in the mesh is numbered such that it is part of the whole mesh. expr can be any expression that ultimately involves a single mesh.</p> <p>Example: global_nodeid(MESH)</p> <p>N = global_nodeid(x)</p>
global_zoneid	Global zone numbers	<p>global_zoneid(expr)</p> <p>The global_zonid expression returns a scalar field containing the global cell numbers for a domain-decomposed mesh so that each cell in the mesh is numbered such that it is part of the whole mesh. expr can be any expression that ultimately involves a single mesh.</p> <p>Example: global_zoneid(MESH)</p> <p>N = global_zoneid(x)</p>

Expression	Meaning	Usage
gradient	Gradient	<p>gradient(expr)</p> <p>The gradient expression computes the gradient, which results in a vector expression, from expr, which must evaluate to a scalar expression. The gradient calculation method varies depending on the type of the mesh over which the input is defined.</p> <p>Example: volume_normals = gradient(vals)</p>
gt	Greater than	<p>gt(expr₁, expr₂)</p> <p>The gt expression returns a value of 1 (true) if expr₁ > expr₂. Both input expressions must be scalar expressions.</p> <p>Example: density = mass / if(gt(volume, 0.0), volume, 0.00001)</p>
if	Conditional	<p>if(expr₁, expr₂, expr₃)</p> <p>The if expression is used to select values based on a condition. Expr₁ must evaluate to a scalar. When expr₁'s values are not equal to zero then the condition is true and the if expression returns the value for expr₂. When expr₁'s values are equal to zero, values for expr₃ are returned.</p> <p>Example: inv_pressure = 1.0 / if(gt(pressure,0.0), pressure, 0.001)</p>
inverse	Matrix inverse	<p>inverse(expr)</p> <p>expr must evaluate to a 3x3 tensor expression. The inverse expression calculates the matrix inverse of the input matrix.</p> <p>Example: invA = inverse(A)</p>
jacobian	Jacobian	<p>jacobian(expr)</p> <p>expr must be an expression that evaluates to a mesh. The jacobian expression returns a scalar expression.</p> <p>Example: jacobian(hydro_mesh)</p>

Expression	Meaning	Usage
laplacian, Laplacian	Laplacian	<p>laplacian(expr)</p> <p>expr must be an expression that evaluates to a scalar. The laplacian expression returns a scalar expression containing the Laplacian of the input scalar field.</p> <p>Example: Laplacian(pressure)</p>
largest_angle	Largest angle	<p>largest_angle(expr)</p> <p>expr must be an expression that evaluates to a 2D mesh. The largest_angle expression calculates a cell-centered scalar field containing the value of the largest interior angle in degrees. Only triangle and quadrilateral cells are considered.</p> <p>Example: largest_angle(hydro_mesh)</p>
le, lte	Less than or equal	<p>le(expr₁, expr₂)</p> <p>The le expression returns a value of 1 (true) if expr₁ ≤ expr₂. Both input expressions must be scalar expressions.</p> <p>Example: if(le(pressure, density), 1.0, 0.0)</p>
ln	Natural logarithm	<p>ln(expr)</p> <p>expr can be a database variable, constant, or other scalar expression. The ln expression computes the natural logarithm of the input scalar expression.</p> <p>Example: ln(density)</p>
log, log10	Base 10 logarithm	<p>log(expr)</p> <p>expr can be a database variable, constant, or other scalar expression. The log expression computes the base 10 logarithm of the input scalar expression.</p> <p>Example: log(density)</p>
lt	Less than	<p>lt(expr₁, expr₂)</p> <p>The lt expression returns a value of 1 (true) if expr₁ < expr₂. Both input expressions must be scalar expressions.</p> <p>Example: if(lt(pressure, density), 1.0, 0.0)</p>

Expression	Meaning	Usage
magnitude	Vector magnitude	<p>magnitude(expr)</p> <p>expr can be a database variable, constant, or other vector expression. The magnitude expression computes the magnitude of the vector and returns a scalar expression.</p> <p>Example: magnitude(vector1)</p>
materror	Material error	<p>materror(expr₁, expr₂)</p> <p>expr₁ must be an expression that evaluates to a material. expr₂ must be an identifier for the name of a material, which can be either numeric or a string constant matching the name of a material in the database. The materror expression computes the difference between the volume fractions stored in the database and those that were used by VisIt's material reconstruction algorithm.</p> <p>Example: materror(mat1, 1)</p>
matvf	Material volume fraction	<p>matvf(expr₁, expr₂)</p> <p>expr₁ must be an expression that evaluates to a material. expr₂ must be an identifier for the name of a material, which can either be numeric or can be a string constant matching the name of a material in the database. The matvf expression extracts the material volume fractions from a material for a specified material name and returns the results in a scalar expression.</p> <p>Examples:</p> <p>percent_chrome = matvf(mat1, "2 chrome")</p> <p>percent_1 = matvf(Material, 1)</p>
max_edge_length	Maximum edge length	<p>max_edge_length(expr)</p> <p>expr must be an expression that evaluates to a mesh. The max_edge_length expression calculates the edge length for each edge in a cell, assigning the length of the longest edge to the entire cell.</p> <p>Example: max_edge_length(hydro_mesh)</p>

Expression	Meaning	Usage
max_side_volume	Maximum side volume	<p>max_side_volume(expr)</p> <p>expr must be an expression that evaluates to a three-dimensional mesh. The max_side_volume expression calculates the side volume for each side in a cell, assigning the value of the biggest side volume to the entire cell. A "side" is a tetrahedron that covers one edge of a cell plus parts of the surrounding faces. When a cell has negative side volume, it is usually twisted.</p> <p>Example: max_side_volume(hydro_mesh)</p>
mean	Mean average	<p>mean(expr)</p> <p>expr must evaluate to a scalar expression. The mean expression calculates the mean average of a scalar field. The mean is calculated using the value and its immediate neighbors. The mean expression only considers the value and its 26 neighbors in three dimensions, or just the eight neighbors in two dimensions. <i>This expression only works on structured meshes.</i></p> <p>Example: mean(intensity)</p>
median	Median	<p>median(expr)</p> <p>expr must evaluate to a scalar expression. The median expression Calculates the median of a scalar field. The median is calculated using the value and its immediate neighbors. The median expression only considers the value and its 26 neighbors in three dimensions, or just the eight neighbors in two dimensions. <i>This expression only works on structured meshes.</i></p> <p>Example: median(intensity)</p>
min_edge_length	Minimum edge length	<p>min_edge_length(expr)</p> <p>expr must be an expression that evaluates to a mesh. The min_edge_length expression calculates the edge length for each edge in a cell, assigning the length of the smallest edge to the entire cell.</p> <p>Example: min_edge_length(hydro_mesh)</p>

Expression	Meaning	Usage
min_side_volume	Minimum side volume	<p>min_side_volume(expr)</p> <p>expr must be an expression that evaluates to a three-dimensional mesh. The min_side_volume expression calculates the side volume for each side in a cell, assigning the value of the smallest side volume to the entire cell. A "side" is a tetrahedron that covers one edge of a cell plus parts of the surrounding faces. When a cell has negative side volume, it is usually twisted.</p> <p>Example: max_side_volume(hydro_mesh)</p>
mirvf	Material interface reconstruction volume fraction	<p>mirvf(expr₁, expr₂, expr₃, expr₄)</p> <p>expr₁ must be an expression that evaluates to a material. expr₂ must be an expression that evaluates to a scalar field containing the zone id's for the mesh. expr₃ must be an expression variable that evaluates to a scalar field containing the volumes or areas of each cell in the mesh. expr₄ must be an expression that evaluates to a list of material names or numbers.</p> <p>The mirvf expression returns the volume fractions computed by VisIt's material interface reconstruction algorithm. The volume fractions computed by VisIt's material interface reconstruction algorithm do not always match what is stored in the database because VisIt's algorithm strives to maintain continuous interfaces rather than accuracy.</p> <p>Example: mirvf(mat1, zoneid(curvmesh2d), area(curvmesh2d), [1])</p>
mod	Modulo	<p>mod(expr₁, expr₂)</p> <p>expr₁ and expr₂ must be expressions that evaluate to integer scalar fields. The mod expression calculates expr₁ modulo expr₂. This expression is typically used in conjunction with the expressions ceil, floor, or round.</p> <p>Example: mod(var1, var2)</p>

Expression	Meaning	Usage
neighbor	Neighbor	<p>neighbor(expr)</p> <p>expr must be an expression that evaluates to a mesh. Neighbor creates a node-centered scalar on a point mesh representation of the original mesh.</p> <p>Expression: neighbor(hydro_mesh)</p>
nmats	Number of materials	<p>nmats(expr)</p> <p>expr must be an expression that evaluates to a material. The nmats expression creates a cell-centered scalar variable containing the number of materials in each cell. This expression can be used with the Threshold operator to remove cells that do not have a desired number of materials.</p> <p>Example: nmats(material)</p>
node_degree	Node degree	<p>node_degree(expr)</p> <p>expr must be an expression that evaluates to a mesh. The node_degree expression calculates how many edges each node it part of and stores that value as a node-centered scalar expression.</p> <p>Example: node_degree(MESH)</p>
nodeid	Node number	<p>nodeid(expr)</p> <p>expr can be any expression that involves a single mesh. The nodeid expression creates a node-centered scalar field containing the node index of each mode in the mesh. The nodeid expression's primary use is in debugging VisIt plots.</p> <p>Example: nodeid(MESH)</p>
not	Logical not	<p>not(expr)</p> <p>expr must be a database variable, constant, or other scalar expression. The not expression returns 1 (true) if the input expression equals zero and zero otherwise.</p> <p>Expression: outside = not(inside)</p>

Expression	Meaning	Usage
notequal, notequals. ne	Not equal	<p>notequal(expr₁, expr₂)</p> <p>The notequal expression returns 1 (true) if expr₁ != expr₂ and zero otherwise. Both expr₁ and expr₂ must be scalar expressions.</p> <p>Example: if(notequal(mass, 12.3), 1, 0)</p>
oddy	General distortion measure	<p>oddy(expr)</p> <p>expr must be an expression that evaluates to a mesh. The oddy expression calculates a general distortion measure based on the left Cauchy-Green tensor.</p> <p>Example: oddy(MESH)</p>
or	Logical or	<p>or(expr₁, expr₂)</p> <p>The logical or expression returns a value of 1 (true) either of the scalar input expressions expr₁ and expr₂ are not equal to zero. If both input expressions are equal to zero then the expression returns zero. The logical or expression is often used with other conditionals such as the if expression.</p> <p>Example: if(or(lt(pressure, 2.0), gt(pressure, 4.0)), pressure, 0.0)</p> <p>Meaning: if pressure is in the range (,2.0) or (4.0,) then use the value for pressure. Otherwise, use zero.</p>
polar	Convert to polar coordinates	<p>polar(expr)</p> <p>expr must be a database variable, constant, or other vector expression. The polar expression converts a vector, which is assumed to represent cartesian coordinates (x, y, z), into polar coordinates (r, theta, phi).</p> <p>Example: vec = polar(coord(MESH))</p>

Expression	Meaning	Usage
polar_phi	Phi component of polar coordinate representation of mesh.	<p>polar_phi(expr)</p> <p>expr must evaluate to a mesh. The polar_phi expression converts the coordinates of the input mesh from cartesian coordinates to polar coordinates and then returns the phi component of the cylindrical coordinates.</p> <p>Example: polar_phi(hydro_mesh)</p>
polar_radius	Radius component of polar coordinate representation of mesh.	<p>polar_radius(expr)</p> <p>expr must evaluate to a mesh. The polar_radius expression converts the coordinates of the input mesh from cartesian coordinates to polar coordinates and then returns the radius component of the cylindrical coordinates.</p> <p>Example: polar_radius(hydro_mesh)</p>
polar_theta	Theta component of polar coordinate representation of mesh.	<p>polar_theta(expr)</p> <p>expr must evaluate to a mesh. The polar_theta expression converts the coordinates of the input mesh from cartesian coordinates to polar coordinates and then returns the theta component of the cylindrical coordinates.</p> <p>Example: polar_theta(hydro_mesh)</p>
principal_deviatoric_tensor	Principal deviatoric vector of tensor	<p>principal_deviatoric_tensor(expr)</p> <p>expr can be a database variable, constant, or other 3x3 tensor expression. The principal_tensor computes the deviatoric principals of a tensor, which result in a vector expression.</p> <p>Example: principal_deviatoric_tensor(tensor1)</p>
principal_tensor	Principals of tensor	<p>principal_tensor(expr)</p> <p>expr can be a database variable, constant, or other 3x3 tensor expression. The principal_tensor computes the principals of a tensor, which result in a vector expression.</p> <p>Example: principal_tensor(tensor1)</p>

Expression	Meaning	Usage
procid	Processor ID	<p>procid(expr)</p> <p>expr must be an expression that evaluates to a mesh. The procid expression returns the processor id of the compute engine that is handling the current domain of the specified mesh.</p> <p>Example: pid = procid(MESH)</p>
rad2deg	Radians to degrees	<p>rad2deg(expr)</p> <p>The rad2deg expression converts the input scalar expression, which is assumed to be in radians, to degrees.</p> <p>Example: angle_deg = rad2deg(angle)</p>
random, rand	Random number	<p>random(expr)</p> <p>expr must be an expression that evaluates to a mesh. The random expression creates a scalar field over the specified mesh where each node in the mesh gets a different random number in the range: [0,1].</p> <p>Expression random(MESH)</p>
recenter	Recenter variable	<p>recenter(expr)</p> <p>expr must be a database variable or other expression that eventually involves a database variable. Two common types of variable centering are cell-centered and node-centered. The recenter expression switches the centering for the input expression. Use the recenter expression to explicitly force a specific variable centering or to make another varia</p> <p>Expression: density + recenter(temp)</p>
relative_difference	Relative difference	<p>relative_difference(expr₁, expr₂)</p> <p>The relative_difference expression computes the relative difference between two scalar expressions using the following formula:</p> <p>if (A == B && A == 0) then 0. else (A-B) / (abs(A) + abs(B))</p> <p>Example: relative_difference(x,y)</p>

Expression	Meaning	Usage
relative_size	Relative cell size	<p>relative_size(expr)</p> <p>expr must be an expression that evaluates to a mesh. The relative_size calculates the relative size of cells in the mesh by finding the minimum value of J or $1/J$ where J is the determinant of a weighted Jacobian matrix.</p> <p>Example: relative_size(MESH)</p>
revolved_surface_area	Revolved surface area	<p>revolved_surface_area(expr)</p> <p>expr must be an expression that evaluates to a 2D mesh. The revolved_surface_area expression revolves triangle and quad cells about the X-axis and calculates the surface area of the revolved cell. The result is a cell-centered scalar expression.</p> <p>Example: revolved_surface_area(mesh2d)</p>
revolved_volume	Revolved cell volume	<p>revolved_volume(expr)</p> <p>expr must be an expression that evaluates to a 2D mesh. The revolved_volume expression calculates the volume of a triangle or quad cell revolved about the X-axis. The result is a cell-centered scalar expression.</p> <p>Example: vol = revolved_volume(mesh2d)</p>
round	Round	<p>round(expr)</p> <p>expr must be an expression that evaluates to a scalar expression. The round expression rounds the value of its expression argument to the nearest integer value. If two integers are equally close then the round expression rounds up. For example, 8.5 gets rounded up to 9.</p> <p>Example: round(pressure)</p>
scaled_jacobian	Scaled jacobian	<p>scaled_jacobian(expr)</p> <p>expr must be an expression that evaluates to a mesh. The scaled jacobian expression calculates a scalar expression based on the minimum of the Jacobian divided by the lengths of a cell's edge vectors.</p> <p>Example: scaled_jacobian(MESH)</p>

Expression	Meaning	Usage
shape	Cell shape	<p>shape(expr)</p> <p>expr must be an expression that evaluates to a mesh. The shape expression calculates 2 divided by the condition number of the weighted Jacobian matrix.</p> <p>Example: shape(MESH)</p>
shape_and_size	Cell shape and size	<p>shape_and_size(expr)</p> <p>expr must be an expression that evaluates to a mesh. The shape_and_size expression calculates the shape and size expressions and multiplies their values.</p> <p>Example: shape_and_size(MESH)</p>
shear	Shear	<p>shear(expr)</p> <p>expr must be an expression that evaluates to a mesh. The shear expression calculates mesh shear and results in a scalar expression.</p> <p>Example: shear(MESH)</p>
sin	Sine	<p>sin(expr)</p> <p>Compute the sine of an angle in radians. Expr can be an expression of database variables, constants, or other scalar expression that evaluate to an angle in radians. The sin expression returns a scalar expression.</p> <p>Example: y= radius * sin(angle)</p>
skew	Mesh skew	<p>skew(expr)</p> <p>expr must be an expression that evaluates to a mesh. The skew expression calculates the skew for each cell in the mesh and stores the result in a cell-centered scalar expression. The skew is the maximum $\cos A$ where A is the angle between the edges at the cell center. Cells with high skew are generally created in error.</p> <p>Example: skew(MESH)</p>

Expression	Meaning	Usage
smallest_angle	Smallest angle	<p>smallest_angle(expr)</p> <p>expr must be an expression that evaluates to a 2D mesh. The smallest_angle expression calculates a cell-centered scalar field containing the value of the smallest interior angle in degrees. Only triangle and quadrilateral cells are considered.</p> <p>Example: smallest_angle(hydro_mesh)</p>
specmf	Species mass fraction	<p>specmf(expr₁, expr₂, expr₃ [, expr₄])</p> <p>The specmf expression extracts the mass fractions for a material species from a species variable and returns a mesh-sized scalar field that can be used in other expressions or plotted.</p> <p>expr₁ must a species variable; the default name for a species variable is: Species. expr₂ must be a single material number or material name. expr₃ must be a single species name or number or list of species names or numbers for the given material. Lists of species are enclosed in square brackets [] and are comma-separated. The expr₄ argument is optional and is a boolean value that weights the species masses by the material volume fraction if it is set to true. When expr₄ is not provided, its value is assumed to be false.</p> <p>Example: specmf(Species, "Air", "Argon")</p>
sq, sqr	Square	<p>sqr(expr)</p> <p>expr can be a database variable, constant, or other scalar expression. The sqr expression multiplies the input values by themselves, squaring them.</p> <p>Example: x_times_x = sqr(x)</p>
sqrt	Square root	<p>sqrt(expr)</p> <p>expr can be a database variable, constant, or another scalar expression. The sqrt expression computes the square root of the input expression.</p> <p>Example: dist = sqrt(x*x + y*y + z*z)</p>

Expression	Meaning	Usage
stretch	Cell stretch	<p>stretch(expr)</p> <p>expr must be an expression that evaluates to a mesh. The stretch expression calculates the square root of 3 times the minimum edge length divided by the maximum diagonal length. Stretch is only calculated for hex and quad cells.</p> <p>Example: stretch(HexMesh)</p>
surface_normal	Surface normal	<p>surface_normal(expr)</p> <p>Calculates the direction normal to each vertex. This incorporates the normal directions of each face a vertex is incident to. Expr must be a mesh that is a three dimensional surface. Note that you will have to apply the External-Surface operator and the DeferExpression operator in order for the surface_normal expression to be evaluated on the plot as it is displayed in the visualization window.</p> <p>Example: surface_normal(surface_mesh)</p>
tan	Tangent	<p>tan(expr)</p> <p>Compute the tangent of an angle in radians. Expr can be an expression of database variables, constants, or other scalar expression that evaluate to an angle in radians. The tan expression returns a scalar expression.</p> <p>Example: $y = \text{radius} * \sin(\text{angle})$</p>
taper	Mesh taper	<p>taper(expr)</p> <p>expr must be an expression that evaluates to a mesh. The taper expression calculates cell taper, which is the maximum ratio of lengths derived from opposite edges. Taper is only calculated for hex and quad cells.</p> <p>Example: taper(MESH)</p>
tensor_maximum_shear	Maximum tensor shear	<p>tensor_maximum_shear(expr)</p> <p>expr can be a database variable, constant, or other 3x3 tensor expression. The result is a scalar expression.</p> <p>Example: tensor_maximum_shear(tensor)</p>

Expression	Meaning	Usage
trace	Trace of tensor	<p>trace(expr)</p> <p>Expr can be an expression of database variables, constants, or other tensor expressions. The trace expression calculates the trace of a 3x3 tensor, which is the sum of the diagonal components. The result is a scalar expression.</p> <p>Example: trace_times_3 = trace(tensor) * 3.0</p>
var_skew	Apply skew scaling	<p>var_skew(expr₁, expr₂)</p> <p>Both expr₁ and expr₂ must evaluate to scalar expressions. More commonly, expr₂ is a floating point constant indicating the skew factor to be used in the calculation. The var_skew expression applies skew scaling, as performed in the Pseudocolor plot, to the input data. Skew scaling can be used to highlight either small values or large values, depending on the skew factor that you provide.</p> <p>Expression: var_skew(density)</p>
volume	Cell volume	<p>volume(expr)</p> <p>expr must be an expression that evaluates to a 3D mesh or database variable. The volume expression calculates the volume of each cell in the mesh and creates a cell-centered scalar expression containing those values. Volumes of 2D meshes are zero.</p> <p>Example: volume(ucdmesh)</p>
warpage	Cell warpage	<p>warpage(expr)</p> <p>expr must be an expression that evaluates to a mesh. The warpage expression calculates how much quad cells deviate from a plane. Warpage is only calculated for quad cells.</p> <p>Example: warpage(quadmesh)</p>

Expression	Meaning	Usage
zoneid	Zone number	<p>zoneid(expr)</p> <p>expr can be any expression involving a single mesh. The zoneid expression creates a zone-centered scalar field for the mesh used by the input expression. The zoneid expression is primarily used to debug new VisIt components such as plot plugins or database reader plugins.</p> <p>Example: zoneid(density)</p>

3.0 Query

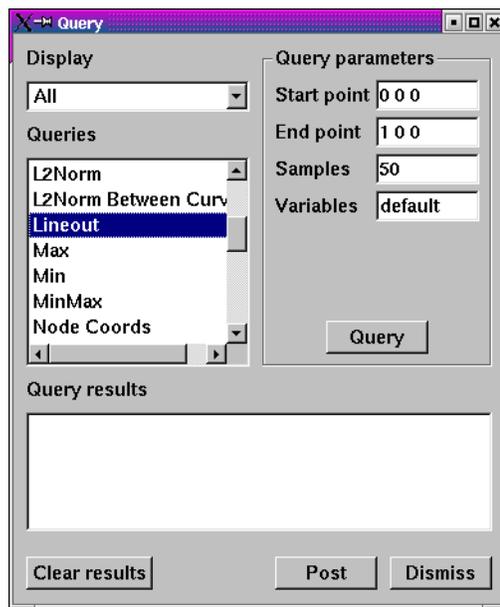


Figure 8-5: Query window

VisIt allows you to gather quantitative information from the database being visualized through the use of queries. A query is a type of calculation that can either return values from the database or values that are calculated from data in the database. For example, VisIt's Pick and Lineout capabilities (described later in this chapter) are specialized point and line queries that print out the values of variables in the database at points or along lines. In addition to point and line queries, VisIt provides database queries that return values that are based on all of the data in a database.

Some queries can even be executed for all of the time states in a database to yield a Curve plot of the query's behavior over time. This feature will be covered in more detail a little later.

VisIt's queries are available in the **Query Window** (shown in Figure 8-5), which you can open by clicking the **Query** option in the **Main Window's Control** menu. The **Query window** consists of upper and lower areas where the upper area allows you to select a query and set its query parameters. The controls for setting a query's parameters change as required and some queries have no parameters and thus have no controls for setting parameters. The bottom area of the window displays the results of the query once VisIt has finished processing it. The results for new queries are appended to the output from previous queries until you clear the **Query results** by clicking the **Clear results** button.

3.1 Query types

VisIt's queries can be divided into three types: database queries, point queries, and line queries. Database queries usually calculate information for the database as a whole instead of concentrating on a single zone or node but some Pick-related database queries do concentrate on cells and nodes. Point queries calculate information for a point in the database and several types of variable picking queries fall into this category. Line queries calculate information along a line. Each type of query has different controls in the **Query parameters** area (see Figure 8-6) and as you highlight different queries, the controls in the **Query parameters** area may change.

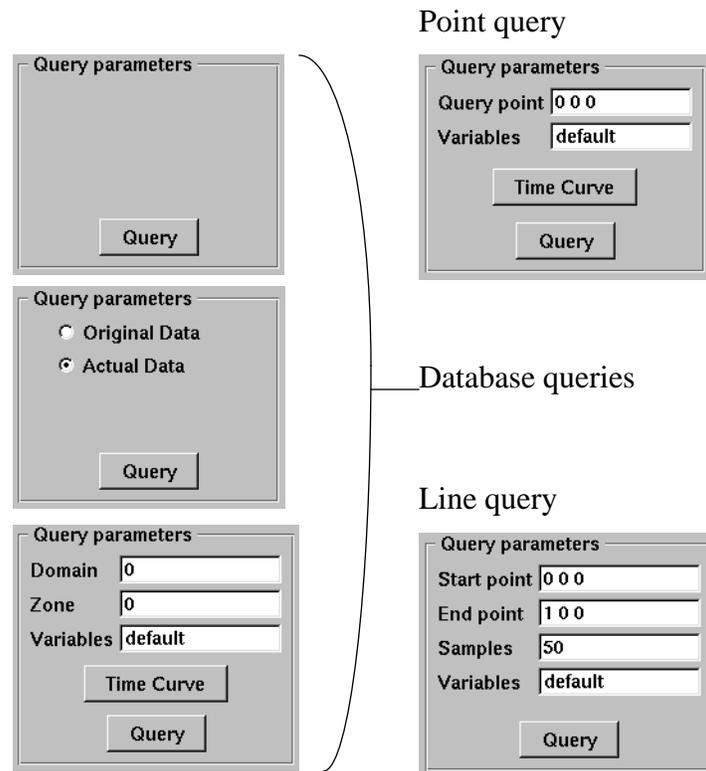


Figure 8-6: Query parameters area

Database queries provide a few different interfaces depending on the query. Many database queries require no additional input so they have no controls except for the **Query** button. Other database queries ask whether the query is to be performed with respect to the original data or the actual data, which is that data that is left in the plot after subsets have been removed and operators have transformed the data. Finally, some database queries ask for a specific domain number and zone or node number.

Point queries provide interfaces in the **Query parameters** area that allow you to enter a 3D point or a screen space point to use as the point for the query. Line queries provide an interface that lets you specify the start and end positions of the line as well as the number

of sample points to consider along the length of the line. Nearly all query types allow you to provide additional variables to query in a **Variables** text field.

3.2 Built-in queries

The following table lists the query name, type, and description for the built-in queries that VisIt supports. The query type for database queries, point queries, and line queries are respectively marked as: DB, Point, and Line.

Query name	Query type	Description
2D area	DB	The 2D area query calculates the area of the 2D plot highlighted in the Plot list and prints the result to the Query results . VisIt can produce a Curve plot of this query with respect to time.
3D surface area	DB	The 3D surface area calculates the area of the plot highlighted in the Plot list and prints the result to the Query results . VisIt can produce a Curve plot of this query with respect to time.
Area Between Curves	DB	The Area Between Curves query calculates the area between 2 curve plots. The plots that will serve as input to this query must both be highlighted in the Plot list or VisIt will issue an error message. Once the area has been calculated, the result is printed to the Query results .
Centroid	DB	This query will calculate the centroid of a dataset. The contribution of each cell is calculated assuming its mass all lies at the center of the cell. If the query is performed on a Pseudocolor plot, the plot's variable will be assumed to be density. If the query is performed on a plot such as a Mesh plot or FilledBoundary plot, uniform density will be used. The results are print to the Query results .
Compactness	DB	The Compactness query calculates mesh metrics and prints them in the Query results .
Cycle	DB	The Cycle query prints the cycle for the plot that is highlighted in the Plot list to the Query results .
Eulerian	DB	The Eulerian query calculates the Eulerian number for the mesh that is used by the highlighted plot in the Plot list . The results are printed to the Query results .

Query name	Query type	Description
Integrate	DB	The Integrate query calculates the area under the Curve plot that is highlighted in the Plot list and prints the result to the Query results .
Kurtosis	DB	The Kurtosis query calculates the kurtosis of a normalized distribution function. The normalized distribution function must be represented as a Curve plot in VisIt. Kurtosis measures the variability of a distribution by comparing the ratios of the fourth and second central moments. The results are print to the Query results .
L2Norm	DB	The L2Norm query calculates the L2Norm, or square of the integrated area, of a Curve plot. The Curve plot must be highlighted in the Plot list . The results are printed to the Query results .
L2Norm Between Curves	DB	The L2Norm query takes two Curve plots as input and calculates the L2Norm between the 2 curves. Both Curve plots must be highlighted in the Plot list or VisIt will issue an error message. The results are printed to the Query results .
Lineout	Line	The Lineout query creates a new instance of the highlighted plot in the Plot list , applies a Lineout operator, and copies the plot to another vis window. The properties of the Lineout operator such as the start and end points are set using the controls in the Query parameters area of the Query Window . Creating Lineouts in this manner instead of using VisIt's interactive lineout allows you to create 1D Curve plots from 3D databases.
Min	DB	The Min query calculates the minimum value for the variable used by the highlighted plot in the Plot list and prints the value and the logical and physical coordinates where the minimum value was found to the Query results .
Max	DB	The Max query calculates the maximum value for the variable used by the highlighted plot in the Plot list and prints the value and the logical and physical coordinates where the maximum value was found to the Query results .

Query name	Query type	Description
MinMax	DB	The MinMax query calculates the minimum and maximum values for the variable used by the highlighted plot in the Plot list and prints the values and their logical and physical coordinates in the Query results .
Moment of inertia	DB	This query will calculate the moment of inertia tensor for each cell in a three-dimensional dataset. The contribution of each cell is calculated assuming its mass all lies at the center of the cell. If the query is performed on a Pseudocolor plot, the plot's variable will be assumed to be density. If the query is performed on a plot such as a mesh plot or FilledBoundary plot, uniform density will be used. The results are print to the Query results .
NodeCoords	DB	The NodeCoords query prints the node coordinates for the specified node and prints the values in the Query results .
NodePick	Point	The NodePick query performs node picking at the specified world coordinate which, if used in 3D, need not be on the surface of a 3D dataset. The plot to be picked must be highlighted in the Plot list . Information about the picked node, if there is one, is printed to the Query results and the Pick Window .
NumNodes	DB	The NumNodes query prints the number of nodes for the mesh used by the highlighted plot in the Plot list to the Query results .
NumZones	DB	The NumZones query prints the number of zones for the mesh used by the highlighted plot in the Plot list to the Query results .
Pick	Point	The Pick query performs zone picking at the specified world coordinate which, if used in 3D, need not be on the surface of a 3D dataset. The plot to be picked must be highlighted in the Plot list . Information about the picked node, if there is one, is printed to the Query results and the Pick Window .

Query name	Query type	Description
PickByNode	Point	The PickByNode query performs node pick using the highlighted plot in the Plot list and specified domain and node values. You can give a global node number if you turn on the Use Global Node check box. A pick point is added to the vis window and the query results appear in the Query results and the Pick Window . Note: this is the query to use if you want to query the database for the value of a variable at a certain node. VisIt can produce a Curve plot of this query with respect to time.
PickByZone	Point	The PickByZone query performs zone pick using the highlighted plot in the Plot list and specified domain and zone values. You can give a global node number if you turn on the Use Global Zone check box. A pick point is added to the vis window and the query results appear in the Query results and the Pick Window . Note: this is the query to use if you want to query the database for the value of a variable at a certain cell. VisIt can produce a Curve plot of this query with respect to time.
Revolved surface area	DB	The Revolved surface area query revolves the mesh used by the highlighted plot in the Plot list about the X-axis and prints the plot's revolved surface area to the Query results .
Revolved volume	DB	The Revolved volume area query revolves the mesh used by the highlighted plot in the Plot list about the X-axis and print's the plot's volume to the Query results .
Skewness	DB	The Skewness query calculates the skewness of a normalized distribution function. The normalized distribution function must be represented as a Curve plot in VisIt. Skewness measures the symmetry of a distribution using its second and third central moments. The results are print to the Query results .

Query name	Query type	Description
Spatial Extents	DB	The Spatial Extents query calculates the original or actual spatial extents for the plot that is highlighted in the Plot list . Whether the original or actual extents are calculated is determined by setting the options in the Query parameters area. The spatial extents are printed to the Query results when the query has finished.
Spherical compactness factor	DB	This query attempts to measure how spherical a three dimensional shape is. The query first determines what the volume of a shape is. It then constructs a sphere that has that same volume. Finally, the query positions the sphere so that the maximum amount of the original shape is within the sphere. The query returns the percentage of the original shape that is contained within the sphere. The results are print to the Query results . VisIt can produce a Curve plot of this query with respect to time.
Time	DB	The Cycle query prints the cycle for the plot that is highlighted in the Plot list to the Query results .
Variable Sum	DB	The Variable Sum query adds up the variable values for all cells using the plot highlighted in the Plot list and prints the results to the Query results . VisIt can produce a Curve plot of this query with respect to time.
Volume	DB	The Volume query calculates the volume of the mesh used by the plot highlighted in the Plot list and prints the value to the Query results . VisIt can use this query to produce a Curve plot of volume with respect to time.
Watertight	DB	The Watertight query determines if a three-dimensional surface mesh, of the plot highlighted in the Plot list , is "watertight", meaning that it is a closed volume with mesh connectivity such that every edge is indcident to exactly two faces. This means that no edge can have a duplicate in the exact same position. The result of the query is printed in the Query results .

Query name	Query type	Description
Weighted Variable Sum	DB	The Weighted Variable Sum query adds up the variable values, weighted by cell size, for all cells using the plot highlighted in the Plot list and prints the results to the Query results . VisIt can produce a Curve plot of this query with respect to time.
ZoneCenter	DB	The ZoneCenter query calculates the zone center for a certain cell in the database used by the highlighted plot in the Plot list . The cell center is printed to the Query results and the Pick Window .

3.3 Executing a query

VisIt has many queries from which to choose. You can choose the type of query to execute by clicking on the name of the query in the **Queries list**. The **Queries list** usually displays the names of all of the queries that VisIt knows how to execute. If you instead want to view a subset of the queries, grouped by function, you can make a selection from the **Display as** combo box. Once you have clicked on a query in the **Query list**, the **Query parameters** area updates to show the controls that you need to edit the parameters for the query. In the case of a point query like Pick, the only parameters you need to specify are the 3D point where VisIt will extract values and the names of the variables that you want to examine. Once you specify the query parameters, click the **Query** button to tell VisIt to process the query. Once VisIt has fulfilled your request, the query results are displayed in the **Query results** at the bottom of the **Query Window**.

3.4 Querying over time

Many of VisIt's queries can be executed for every time state in the database used by the queried plot. The results from a query over time is a Curve plot that plots the query results with respect to time. The **Query parameters** area contains a **Time Curve** button when the selected query can be plotted over time. Clicking the **Time Curve** button executes the selected query for each time state in the database used by the plot highlighted in the **Plot list**. VisIt then creates a new Curve plot in a new vis window and uses the query results versus time as the curve data.

By default, querying over time will force VisIt to execute the selected query on every time state in the relevant database. If you want to restrict the number of time states used when querying over time or if you want to set some general options that also affect how time curves are created, you can set additional options in the **Query Over Time Window** (see Figure 8-7). If you want to open the **Query Over Time Window**, click on the **Query over time option** in the **Controls** menu in VisIt's **Main Window**.

3.4.1 Querying over a time range

You can restrict the range of time states that are considered when VisIt is performing a query over time if you specify a start or end time state in the **Query Over Time Window**. To set a starting time state, click the **Starting timestep** check box and enter a new time state into the adjacent text field. To set an ending time state, click the **Ending timestep** check box and enter a new ending time state into the adjacent text field.

In addition to setting the starting and ending time states, you can also specify a stride so VisIt can skip frames in the middle and consider every N^{th} frame instead of every frame. If you want to specify a stride, enter a new stride into the **Stride** text field in the **Query Over Time Window** and click the **Apply** button.

3.4.2 Setting the axis title

When VisIt creates a new Curve plot, after having calculated a query over time, the horizontal axis label is labeled with the database cycles. If you prefer to think about time in terms of time state or simulation time then you can change the axis label by clicking one of the following radio buttons in the **Query Over Time Window**: **Cycle**, **Time**, **Timestep**.

3.4.3 Setting the time curve's destination window

When VisIt creates a Curve plot using the results of a query over time, the Curve plot is placed in a vis window designated for Curve plots. If there is no vis window into which the Curve plot can be added, VisIt creates a new vis window to contain the Curve plot. If you want VisIt to always place the new Curve plot in a specific window, turn off the **Use 1st unused window or create new one** check box and enter a new window number into the **Window#** text field. After setting these options, subsequent Curve plots created by querying over time will be added to the specified vis window.

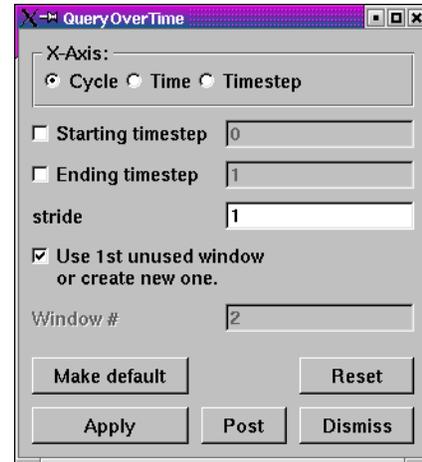


Figure 8-7: Query Over Time Window

4.0 Pick

VisIt provides a way to interactively pick values from the visualized data using the visualization window's Zone Pick and Node Pick modes. When a visualization window is in one of those pick modes, each mouse click in the visualization window causes VisIt to find the location and values of selected variables at the pick point. When VisIt is in Zone pick mode, it finds the variable values for the zones that you click on. When VisIt is in node pick mode, similar information is returned but instead of returning information about the zone that you clicked on, VisIt returns information about the node closest to the point that you clicked. Pick is an essential tool for performing data analysis because it can extract exact information from the database about a point in the visualization.

4.1 Pick mode

You can put the visualization window into one of VisIt's pick modes by selecting **Zone Pick** or **Node Pick** from the **Popup menu's Mode** submenu. After the visualization window is in pick mode, each mouse click causes VisIt to determine the values of selected variables for the zone that contains the picked point or the node closest to the picked point. Each picked point is marked with an alphabetic label which starts at A, cycles through the alphabet and repeats. The pick marker is added to the visualization window to indicate where pick points have been added in the past. To clear pick points from the visualization window, select the **Pick points** option from the **Clear** menu in the **Main Window's Window menu**. The dimension of the plots in the visualization does not matter when using pick mode. Both 2D and 3D plots can be picked for values. However, when using pick mode with 3D plots, only the surface of the plots can be picked for values. If you want to obtain interior values then you should use one of the Pick queries or apply operators that expose the interiors of 3D plots before using pick. An example of the

visualization window with pick points is shown in Figure 8-8 and an example of node pick and zone pick markers is shown in Figure 8-8.

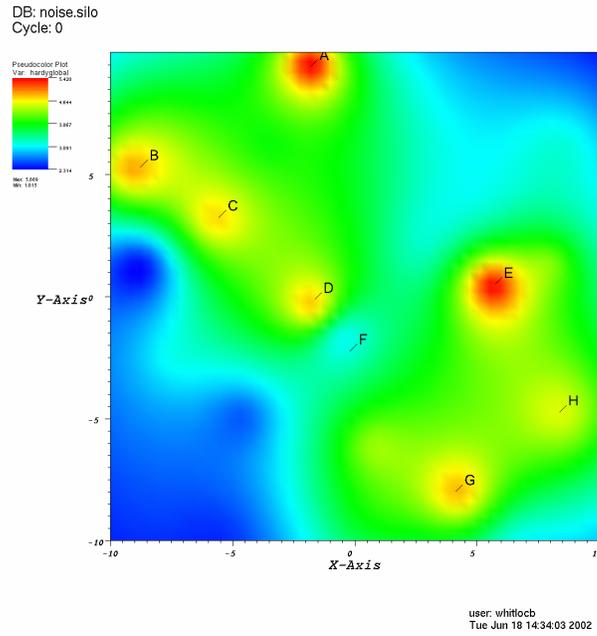


Figure 8-8: Visualization window with pick points

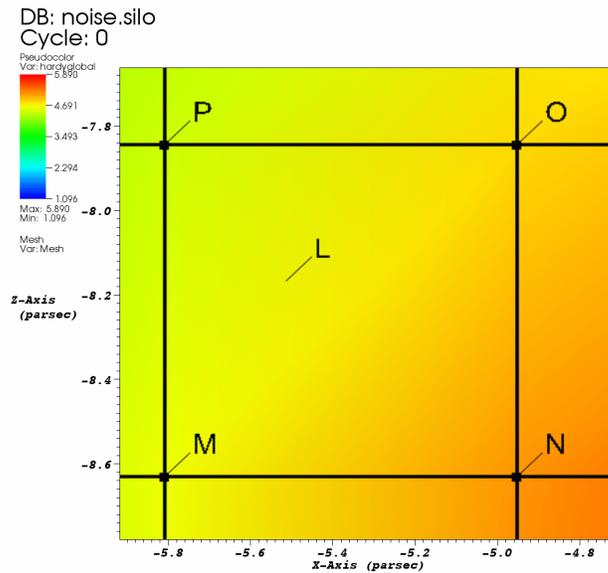


Figure 8-8: Zone pick marker L and node pick markers M,N,O,P

4.2 Pick Window

Each time a new pick point is added to the visualization window by clicking on a plot, VisIt extracts information about the pick point from the plot's database and displays it in the **Pick Window** (Figure 8-9) and the **Output Window**. If the **Pick Window** does not automatically open after picking, you can open the **Pick Window** by selecting the **Pick** option from the **Main Window's Controls** menu.

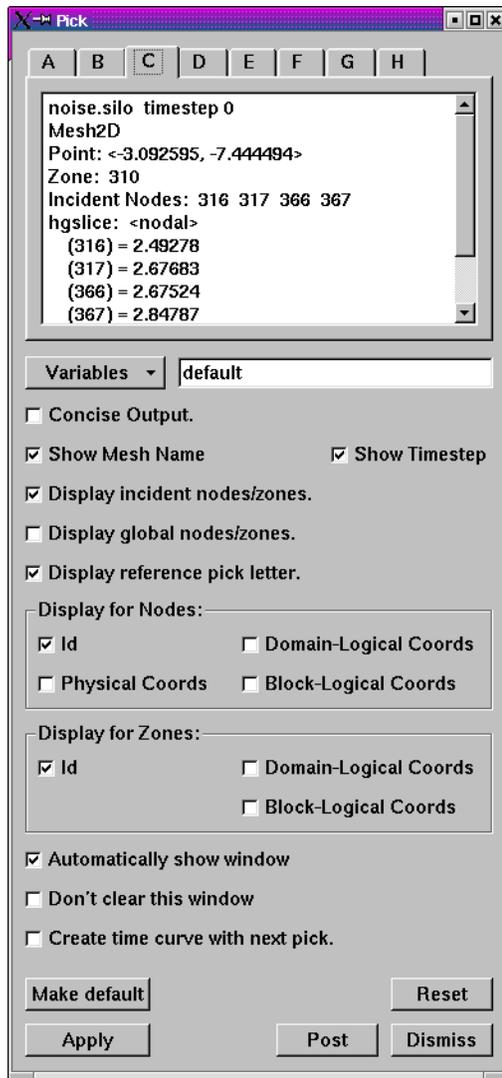


Figure 8-9: Pick Window

replace the default pick variable by typing one or more valid variable names, separated by spaces, into the **Variables** text field. You can also select additional pick variables by selecting a new variable name from the **Variables** variable button to the left of the **Variables** text field. When more than one variable is picked, multiple variables appear in the pick information displayed in the information tabs.

The **Pick Window** mainly consists of a group of tabs, each of which displays the values from a pick point. The tab label A, B, C, etc. corresponds to the pick point label in the visualization window. Since there is a fixed number of tabs in the **Pick Window**, tabs are recycled as the number of pick points increases. When a pick point is added, the next available tab, which is usually the tab to the right of the last unused tab, is populated with the pick information. If the rightmost tab already contains pick information, the leftmost tab is recycled and the process repeats. To see a complete list of picked points, open the **Output Window**.

The information displayed in each tab consists of the database name and timestep, the coordinates of the pick point, the zone/cell that contains the pick point, the nodes that make up the cell containing the pick point, and the picked variables. The rest of the **Pick Window** is devoted to setting options that format the pick output.

4.2.1 Setting the pick variable

The **Pick Window** contains a **Variables** text field that allows you to specify pick variables. Most of the time, the value in the text field is the word “default” which tells VisIt to use the plotted variables as the pick variables. You can

4.2.2 Concise pick output

Pick returns a lot of information when you pick on a plot. The **Pick Window** usually displays the pick output one item per line, which can end up taking a lot of vertical space. If you want to condense the information into a smaller area, click the **Concise output** check box. Sometimes, not all of the information is relevant for your analysis so VisIt provides options to hide certain items in the pick output. If you don't want VisIt to display the name of the picked mesh, turn off the **Show Mesh Name** check box. If you don't want VisIt to show the time state, turn off the **Show timestep** check box.

4.2.3 Turning off incident nodes and cells in pick output

When VisIt performs a pick, the default behavior is to show a lot of information about the cell or node that was picked. This information usually includes the nodes or cells that were incident to the node or cell that was picked. The incident nodes and cells are included to give more information about the neighborhood occupied by the cell or node. If you want to turn off incident nodes and cells in the pick output, click off the **Display incident nodes/zones** check box.

4.2.4 Displaying global node and cell numbers

Many large meshes are decomposed into smaller meshes called domains that, when added together, make up the whole mesh. Each domain typically has its own range of cell numbers that begin at 0 or 1, depending on the mesh's cell origin. Any global cell numbering scheme that may have been in place before the original mesh was decomposed into domains is often lost. However, some meshes have auxiliary information that allows VisIt to use the original global node and cell numbers for the domains. If you want the pick output to contain global node and cell numbers if they are available, click on the **Display global nodes/zones** check box.

4.2.5 Turning off pick markers for new pick points

Some queries that perform picks create pick markers by default, as do VisIt's regular pick modes. If you want to prevent pick queries from creating pick markers, click off the **Pick Window's Display reference pick letter** check box.

4.2.6 Returning node information

In addition to printing the values of the pick variables, pick can also display information about the nodes or cells over which the pick variables are defined. By default, VisIt only returns the integer node indices of the nodes contained by the picked cell. You can make VisIt return the node coordinates in other formats by checking the **Id** check box in the **Display for Nodes** area. The node coordinates can be displayed 4 different ways: Node indices, physical coordinates, domain-logical coordinates, or block-logical coordinates. Click the check boxes in the **Display for Nodes** area that correspond to the types of node information that you want to examine.

4.2.7 Returning zone information

The **Pick Window** has controls in its **Display for Zones** area that allow you to specify how you want VisIt to display zone information. Click the check boxes that correspond to the types of information that you want to examine.

4.2.8 Automatically showing the Pick Window

When you pick on a plot, VisIt automatically opens the **Pick Window** to display the results of the pick operation. You can prevent VisIt from automatically showing the **Pick Window** after a pick operation by turning off the **Automatically show window** check box in the **Pick Window**. If the **Pick Window** does not automatically appear after picking then you can turn on the **Automatically show window** check box.

4.2.9 Picking over time

Querying over time is normally done using the controls in the **Query Window** but you can also pick over time to generate curves that show the behavior of a picked zone or node over time. To pick over time, you must click the **Create time curve with next pick** check box in the **Pick Window**. Once that check box is turned on, each pick operation will result in a new Curve plot that shows the behavior of the most recently picked zone or node over time.

5.0 Lineout

One-dimensional curves, created using data from 2D or 3D plots, are popular for analyzing data because they are simple to compare. VisIt's visualization windows can be put into a mode that allows you to draw lines, along which data are extracted, in the visualization window. The extracted data are turned into a Curve plot in another visualization window. If no other visualization window exists, VisIt creates one and adds the Curve plot to it. Curve plots are often more useful than 2D Pseudocolor plots because they allow the data along a line to be seen spatially as a 1D curve instead of relying on differences in color to convey information. Furthermore, the curve data can be exported to curve files that allow the data to be imported into other Lawrence Livermore National Laboratory curve analysis software such as *Ultra*.

5.1 Lineout mode

You can put the visualization window into lineout mode by selecting **Lineout** from the **Popup menu's Mode** submenu. Note that lineout mode is only available with 2D plots in this version though you can create 3D lineouts using the Lineout query in the **Query Window**. After the visualization window is in lineout mode, you can draw reference lines in the window. Each reference line causes VisIt to extract data from the database along the prescribed path and draw the data as a Curve plot in another visualization window. Each reference line is drawn in a color that matches the initial color of the Curve plot so the

reference lines, which have no labels, can be easily associated with their corresponding Curve plots. If lineout's dynamic mode is also set then changes such as a variable or time state change made to the plot that originated a Curve plot via a lineout operation are also reflected in the Curve plot. To clear the reference lines from the visualization window, select the **Reference lines** option **Clear** submenu in the **Main Window's Window** menu. An example of the visualization window with reference lines and Curve plots is shown in Figure 8-10.

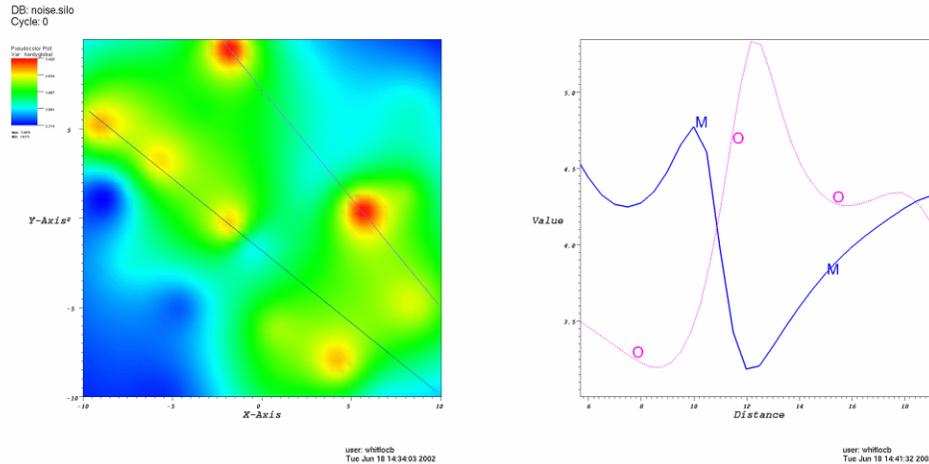


Figure 8-10: Visualization windows with reference lines and Curve plots

5.1.1 Dynamic lineout

Normally when you perform a lineout operation, the Curve plot that results from the lineout operation is in no way connected to the plots in the window that originated the Curve plot. If you want to make variable or time state changes made to the originating plots also affect the Curve plots that were created via lineout, you should set the dynamic lineout flag for lineouts in the **Lineout Options Window** (see Figure 8-11). To open the **Lineout Options Window** (not the **Lineout operator attributes** window), select **Lineout** from the **Controls** menu in the **Main Window**. Click the **Dynamic** check box in the **Lineout Options Window** to make new Curve plots, created via lineout, be linked to their originating plots so they will be affected by variable or time state changes made to the originating plots.

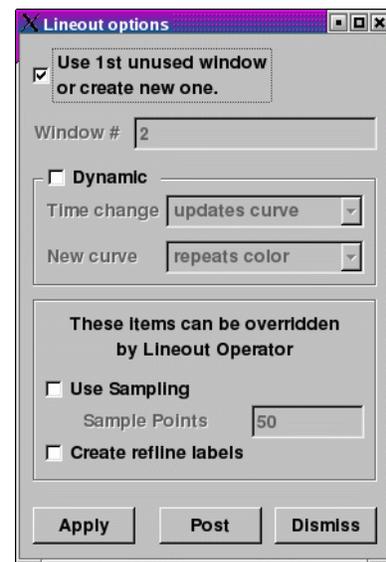


Figure 8-11: Lineout Options Window

The **Lineout Options Window** also contains some other options affecting dynamic lineouts. When you change the variable in the plot the originated the lineout, the lineout will change. When you change time states for the plot that originated the lineout, the lineout will change.

However, you can instead make VisIt create a new Curve plot for the lineout each time the you change the originating plot's time state by changing the Time change behavior in the **Lineout Options Window** from **updates curve** to **creates new curve**. When VisIt creates a new curve each time you advance to a new time state, VisIt will put a new curve in the lineout destination window, resulting in many Curve plots (see Figure 8-12). By default, VisIt will make all of the related Curve plots be the same color. You can override this behavior by selecting **creates new color** instead of **repeats color** from the **Lineout Options Window's New Curve** combo box.

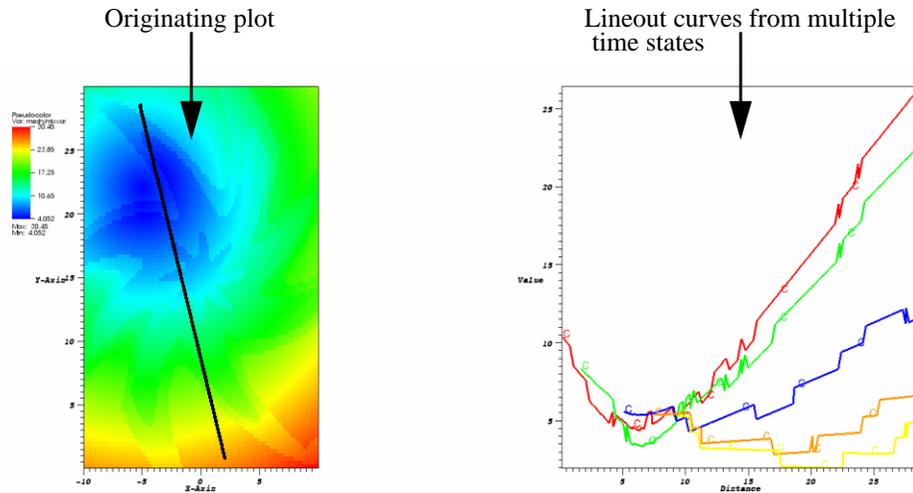


Figure 8-12: Dynamic lineout can be used to create curves for multiple time states

5.2 Curve plot

Curve plots are created by drawing reference lines. The visualization window must be in lineout mode before reference lines can be created. You can create a reference line by positioning the mouse over the first point of interest, clicking the left mouse button and then moving the mouse, while pressing the left mouse button, and releasing the mouse over the second endpoint. Releasing the mouse button creates a reference line along the path that was drawn with the mouse. When you draw a reference line, you cause a Curve plot of the data along the reference line to appear in another visualization window. If another visualization window is not available, VisIt opens a new one before creating the Curve plot. The Curve plot in the second window can be modified by setting the active window to the visualization window that contains the Curve plots.

5.2.1 Setting curve attributes

Curve plots have a few options that can be set to alter their appearance. You can set the curve's line style, which can be solid, dotted, dashed, or dash-dotted, by making a selection from the **Line Style** menu in the **Curve plot attributes window** (Figure 8-13). If the Curve plot is difficult to see in the visualization window, try increasing the line thickness by making a selection from the **Line Width** menu. You can change the curve's color, by clicking on the **Color** button and selecting a new color from the **Popup color menu**. If you don't want the curve to have labels, you can turn them off by unchecking the **Labels** check box. Finally, you can make Curve plots display a small point glyph at each data point along the curve if you check the **Points** check box and set a point size in the **Point size** text field.

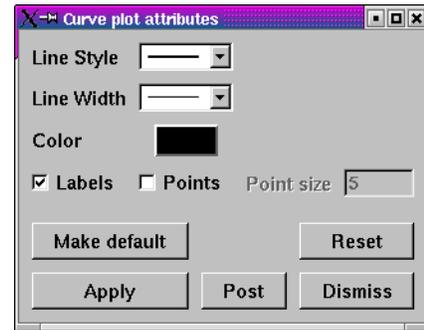


Figure 8-13: Curve plot attributes window

5.2.2 Saving curves

Once a curve has been generated, it can be saved to a curve file. A curve file is an ASCII text file that contains the X-Y pairs that make up the curve and it is useful for exporting curve data to other curve analysis programs. To save a curve, make sure you first set the active window to the visualization window that contains the curve. Next, save the window using the *curve* file format. All of the curves in the visualization window are saved to the specified curve file.

5.3 Lineout Operator

The Curve plot uses the Lineout operator to extract data from a database along a linear path. The Lineout operator is not generally available since curves are created only through reference lines and not the **Plot menu**. Still, once a curve has been created using the Lineout operator, certain attributes of the Lineout can be modified. Note that when you modify the Lineout operator attributes, it is best to turn off the **Apply operator to all plots** check box in the **Main Window** so that all curves do not get the same set of Lineout operator attributes. Some Lineout operator attributes can be set globally in the **Global Lineout Window**, which is accessed by clicking on Lineout option in the **Main Window's Controls** menu. The **Global Lineout Window** contains mainly the same options as the **Lineout operator attributes window** but it applies to all Lineout operators.

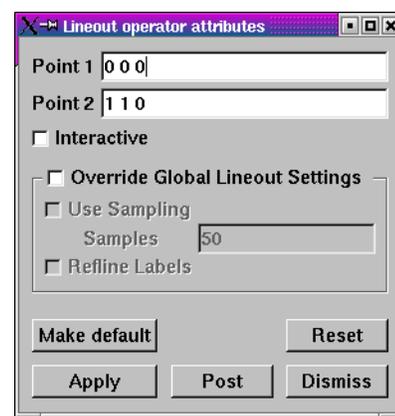


Figure 8-14: Lineout operator attributes window

5.3.1 Setting lineout endpoints

You can modify the line endpoints by typing new coordinates into the **Lineout operator attribute window's Point 1** or **Point 2** text fields (Figure 8-14). Each endpoint is a 3D coordinate that is specified by three space-separated floating point numbers. If you are performing a Lineout operation on 2D data, you can set the value for the Z coordinate to zero.

5.3.2 Setting the number of lineout samples

The Lineout operator works by extracting sample points along a line. The sample points are then used to create Curve plots. The Lineout operator's default sampling scheme is to sample data values at the intersections between the sampling line and the cell boundaries encountered along the way. This method gives rise to jagged Curve plots favored by many VisIt users. See Figure 8-15 for a comparison between the sampling methods. If you instead want to smoothly sample the cells along the sampling line with some number of evenly spaced sample points, you can make the Lineout operator use evenly spaced sampling by clicking on the **Override Global Lineout Settings** check box in the **Lineout operator attributes window**. Once you've told the Lineout operator to override the global lineout settings click on the **Use Sampling** check box and enter a number of sample points. The number of sample points taken along the line determine the fidelity of the Curve plot. Generally, it is best to set the number of sample points such that each cell is sampled at least once. To set the number of sample points, type a new number into the **Samples** text field in the **Lineout operator attributes window**.

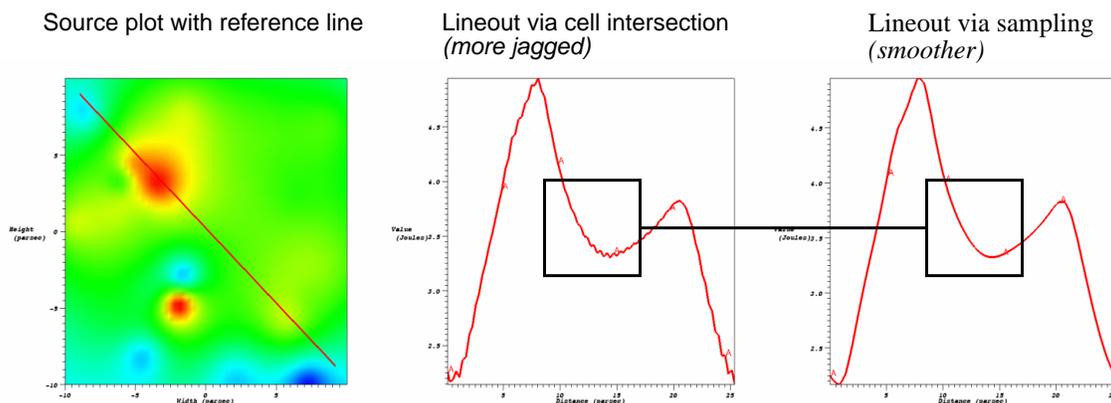


Figure 8-15: Lineout via cell intersection and lineout via sampling

5.3.3 Interactive mode

When the **Interactive** check box is checked, changes to the Lineout operator cause similar changes to the reference line that generated the curve. For example, changing the line endpoints causes the reference line to update.

5.3.4 Reference line labels

You can make the reference lines in the window that caused Curve plots to be generated to have labels by checking the Lineout operator's **Refline Labels** check box.

1.0 Overview

Now that you know how to visualize databases, it is time to learn how to make presentation quality visualizations. This chapter explains what options are available for making professional looking visualizations and introduces new windows that allow you to control annotations, colors, lighting, and the view.

2.0 Annotations

Annotations are objects in the visualization window that convey information about the plots. Annotations can be global objects that show information such as database name, or they can be objects like plot legends that are directly tied to plots. Annotations are an essential component of a good visualization because they make it clear what is being visualized and they make the visualization appear more polished.

VisIt supports a few different annotation types that can be used to enhance visualizations. The first category of annotations includes global annotations like database name, user name, and plot legends. These annotations convey a good deal of information about what is being visualized, what values are in the plots, and who created the visualization. The second category of annotations include the plot axes and labels. This group of annotations comes in two groups: 2D, and 3D. The attributes for the 2D annotations can be set independently of the 3D annotation attributes. Colors can greatly enhance the look of a visualization so VisIt provides controls to set the colors used for annotations and the visualization window that contains them. The third and final category includes annotation objects that can be added to the visualization window. You can add as many annotation objects as you want to a visualization window. The currently supported annotation objects are: 2D text and time slider annotations.

2.1 Annotation Window

The **Annotation Window** (see Figure 9-1) contains controls for the various annotations that can appear in a visualization window. You can open the window choosing the **Annotation** option from the **Main Window's Controls menu**. The **Annotation Window** has a tabbed interface which allows the three basic categories of annotations to be grouped together. The first tab lets you control the 2D annotation options. The second tab lets you control the 3D annotation options. The third tab lets you control the colors used for annotations and the visualization window. Finally, the fourth tab lets you create arbitrary numbers of annotation objects such as 2D text annotations or time slider annotations.

In addition to the four tabs that contain annotation controls, there are three check boxes at the top of the **Annotation Window** for turning global annotations, such as database name, on and off. If you want to turn off all annotations then you can click the **No annotations** button.

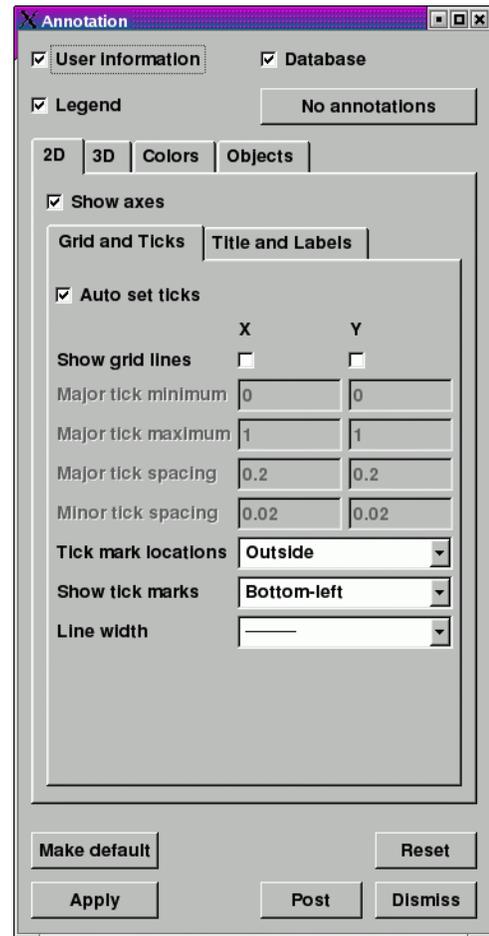


Figure 9-1: Annotation Window

2.2 General Annotations

VisIt has a few general annotations that describe the visualization and are independent of the type of database in the visualization. General annotations encompass user name, database name, and plot legends. The **Annotation Window** does not provide too much flexibility for these types of annotations, but it does provide controls that allow you to disable these annotations. Figure 9-2 shows common locations for some general annotations.

2.2.1 Turning user information on/off

When add plots to the visualization window, your username is shown in the lower right corner. The user information annotation is turned on or off using the **User information** check box at the top of the **Annotation Window**. You may want to turn off user information when you are generating a movie.

2.2.2 Turning database information on/off

When plots are displayed in the visualization window, the name of the database used in the plots is shown in the visualization window's upper left corner. You can turn the database information on or off using the **Database** check box at the top of the **Annotation Window**.

2.2.3 Turning plot legends off globally

Plot legends are special annotations that are added by plots. An example of a plot legend is the color bars and title that the Pseudocolor plot adds to the visualization window. Normally, plot legends are turned on or off by a check box in a plot attribute window but VisIt also provides a check box in the **Annotation Window** that can turn off the plot legends for all plots in the visualization window. You can use the **Legend** check box at the top of the **Annotation Window** to turn plot legends off if they are present.

2.3 2D Annotations

VisIt has a number of controls in the **Annotation Window** to control 2D annotations on the **2D** options tab (Figure 9-1). The 2D annotation settings are primarily concerned with the appearance of the 2D axes that frame plots of 2D databases. Figure 9-2 shows a plot with various annotations.

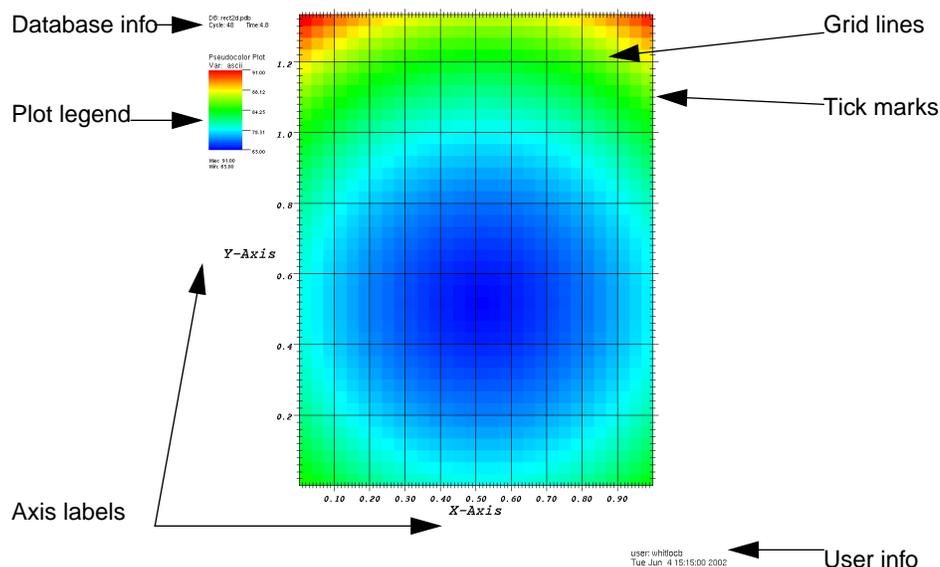


Figure 9-2: 2D plot with annotations

2.3.1 Setting grid line properties

The 2D grid lines are a set of lines that make a grid over the 2D viewport. The grid lines are disabled by default but you can enable them by checking the **Show grid lines** check boxes on the **Grid and Ticks** sub-tab in the **Annotation Window's 2D Options tab**. Grid lines can be enabled for one or more dimensions.

2.3.2 Setting tick mark properties

The tick marks are small lines that are drawn along the edges of the 2D viewport. Tick marks can be drawn on a variety of axes by selecting a new option from the **Show tick marks** menu. Tick marks can also be drawn on the inside, outside, or both sides of the plot viewport by selecting a new option from the **Tick mark locations** menu in the **Annotation Window**.

Tick mark spacing is usually changed to best suite the plots in the visualization window but you can explicitly set the tick mark spacing by first unchecking the **Auto set ticks** check box and then typing new tick spacing values into the **Major tick minimum**, **Major tick maximum**, **Major tick spacing**, and **Minor tick spacing** text fields. The text fields on the left correspond to the horizontal axis while the rightmost text fields set the tick spacing for the vertical axis.

2.3.3 Setting axis label properties

The axis labels are the labels that appear along the 2D plot viewport. This includes the title of the axis as well as the numeric labels that indicate the plot's spatial dimensions. By default, the axis labels are enabled and set to appear for both the X and Y axes. You can turn the labels off for one or more axes unchecking the **Show labels** check boxes on the **Annotation Window's 2D Options tab**. You can change the size of the axis labels and titles by entering new font sizes, specified as a percentage of vis window height, into the **Label font height** and **Title font height** text fields. Note that you can set these label properties for the X and Y dimensions independently.

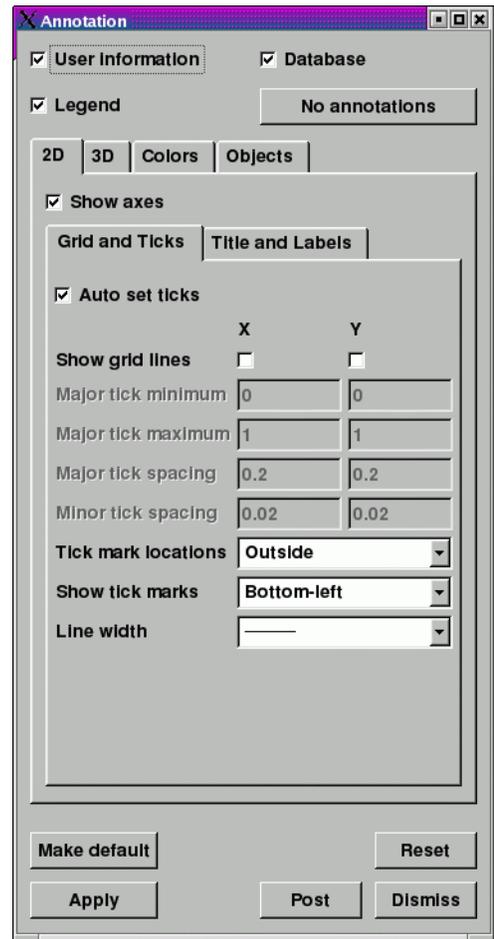


Figure 9-3: Grid and Ticks tab on Annotation Window's 2D Options tab

2.3.4 Setting axis titles and units

The axis titles are the names that are drawn along each axis, indicating the meaning of the values shown along the axis. Normally, the names used for the axis titles come from the database being plotted so the axis titles are relevant for the displayed plots. Many of VisIt's database readers plugins read file formats that have no support for storing axis titles so VisIt uses default values such as: "X-Axis", "Y-Axis". VisIt's **Annotation Window** provides options that allow you to override the defaults or the axis titles that come from the file. If you want to override the axis titles that VisIt uses for 2D visualizations, turn on the **Set X-Axis title** or **Set Y-Axis title** check boxes on the **Title and Labels** sub-tab (see Figure 9-4) on the **Annotation Window's 2D Options tab**. Next, type the new axis titles into the adjacent text fields.

On addition to overriding the names of the axis titles, you can also override the units that are displayed next to the axis titles. Units are displayed only when they are available in the file format and like axis titles, they are not always stored in the file being plotted. If you want to specify units for the axes, turn on the **Set X-Axis units** or **Set Y-Axis units** check boxes and type new units into the adjacent text fields.

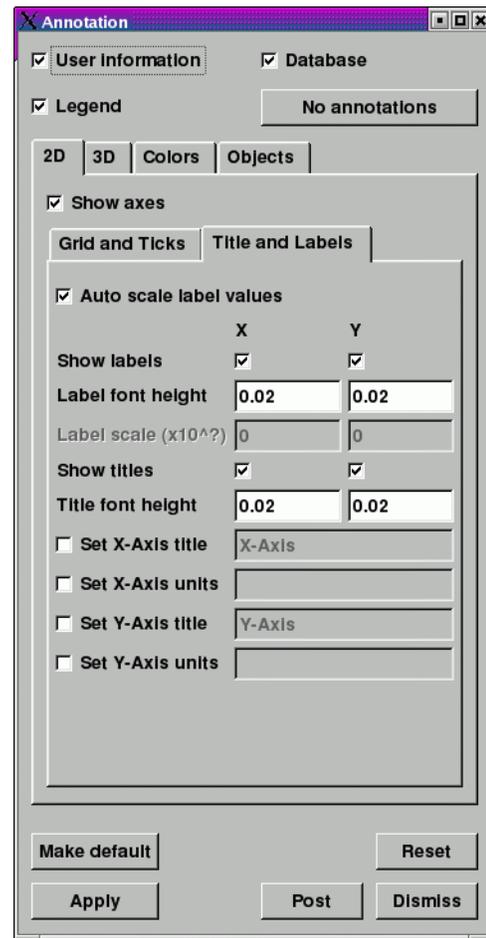


Figure 9-4: Title and Labels tab on Annotation Window's 2D Options tab

2.4 3D Annotations

VisIt has a number of controls, located on the **3D Options tab**, in the **Annotation Window** for controlling annotations that are used when the visualization window contains 3D plots. Like the 2D controls, these controls focus mainly on the axes that are drawn around plots. Figure 9-5 shows an example 3D plot with the 3D annotations labeled and Figure 9-6 shows the **Annotation Window's 3D Options tab**.

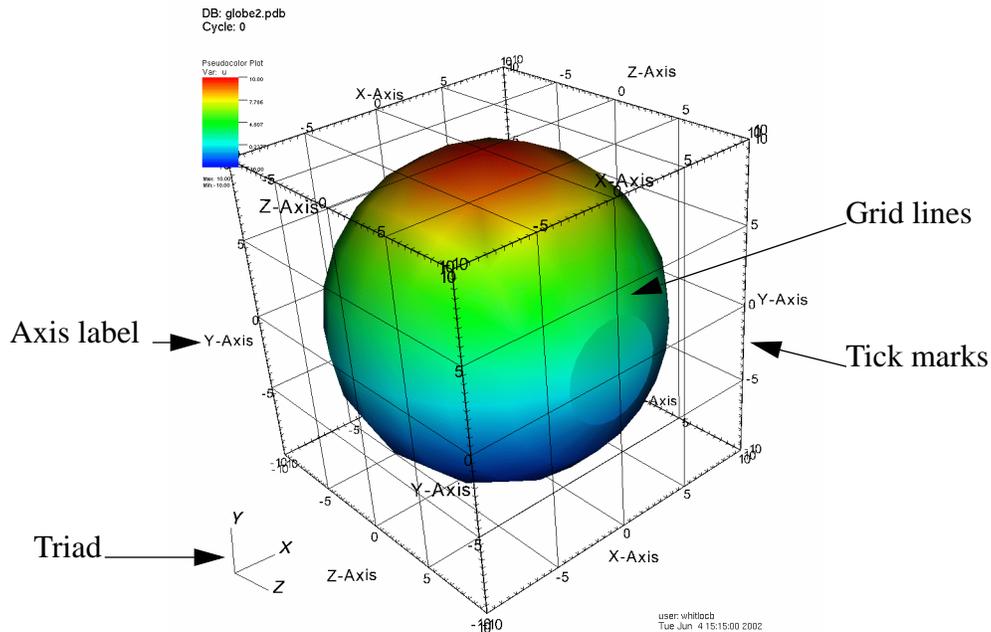


Figure 9-5: 3D plot with annotations

2.4.1 Hiding all axes

As a convenience, VisIt provides a **Draw axes** check box that can be used to turn all axes off without having to turn off each individual annotation setting. To hide all of the 3D axes, uncheck the **Draw axes** check box.

2.4.2 Turning off the triad

The triad annotation contains a small set of axes and it is drawn in the lower left corner of the visualization window. It is displayed so you can get your bearings in 3D. The triad can be turned off by unchecking the **Triad** check box in the **Annotation Window**.

2.4.3 Setting grid line properties

The 3D grid lines are a set of lines that make a grid around the 3D bounding box. The grid lines are disabled by default but you can enable them using the **Show grid lines** check boxes in the **Annotation Window**. Grid lines can be enabled for one or more dimensions.

2.4.4 Setting tick mark properties

The tick marks are small lines that are drawn along the edges of the 3D bounding box. Tick marks can be drawn on a variety of axes by checking the **Show tick marks** check boxes. Tick marks can also be drawn on the inside, outside, or both sides of the bounding

box by selecting a new option from the **Tick mark locations** menu in the **Annotation Window**.

2.4.5 Setting the plot axis type

VisIt provides a few different types of 3D plot axes. You can set the plot axis type by making a selection from the **Axis type** menu on the **Grid and Ticks** sub-tab in the **Annotation Window's 3D Options tab**.

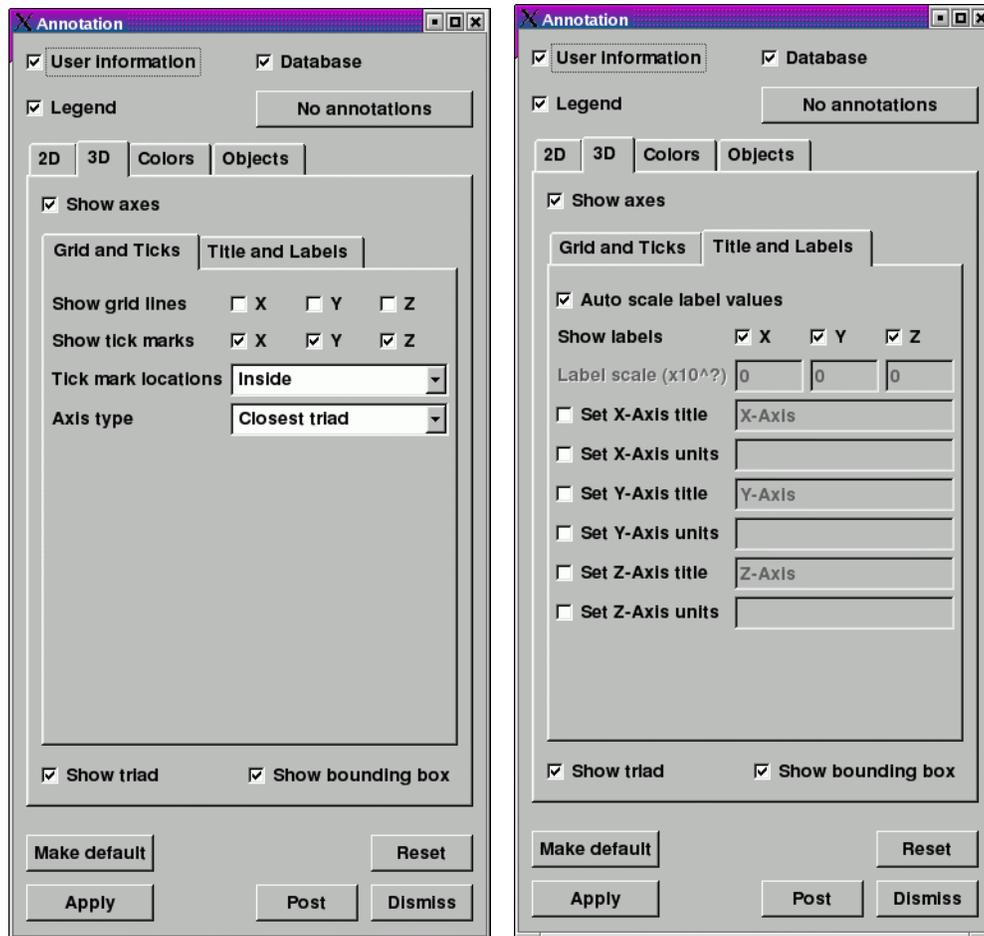


Figure 9-6: 3D Options tab is divided into Grid and Ticks / Title and Labels tabs

2.4.6 Setting axis label properties

The 3D annotation options provide three **Show labels** check boxes that allow you to turn individual axis labels on and off. Click the check box next to each axis that you want to have axis labels or leave them unchecked if you do not want axis labels.

You can specify a label scale, which affects how VisIt displays the numbers used in the axes. To specify a label scale, turn off the **Auto scale label values** check box and type new scaling exponents into the **Label scale** text fields for the X, Y, and Z dimensions.

2.4.7 Setting axis titles and units

The axis titles are the names that are drawn along each axis, indicating the meaning of the values shown along the axis. Normally, the names used for the axis titles come from the database being plotted so the axis titles are relevant for the displayed plots. Many of VisIt's database readers plugins read file formats that have no support for storing axis titles so VisIt uses default values such as: "X-Axis", "Y-Axis", "Z-Axis". VisIt's **Annotation Window** provides options that allow you to override the defaults or the axis titles that come from the file. If you want to override the axis titles that VisIt uses for 3D visualizations, turn on the **Set X-Axis title**, **Set Y-Axis title**, or **Set Z-Axis title** check boxes on the **Title and Labels** sub-tab (see Figure 9-6) on the **Annotation Window's 3D Options tab**. Next, type the new axis titles into the adjacent text fields.

On addition to overriding the names of the axis titles, you can also override the units that are displayed next to the axis titles. Units are displayed only when they are available in the file format and like axis titles, they are not always stored in the file being plotted. If you want to specify units for the axes, turn on the **Set X-Axis units**, **Set Y-Axis units**, or **Set Z-Axis units** check boxes and type new units into the adjacent text fields.

2.5 Annotation Colors

Colors are very important in a visualization since they help to determine how easy it is to read annotations. VisIt provides a tab in the **Annotation Window**, shown in Figure 9-7, specifically devoted to choosing annotation colors. The **Colors** tab contains controls to set the background and foreground for the visualization window which, in turn, sets the colors used for annotations. The **Colors** tab also provides controls for more advanced background colors called gradients which are colors that bleed into each other.

2.5.1 Setting background and foreground colors

The **Colors** tab has two color buttons that allow you to set the background and foreground colors. To set the background or foreground color, click the **Background** or **Foreground** color button and select a color from the **Popup color menu**. Releasing the mouse outside of the **Popup color menu** (see Figure 9-8) cancels color selection and the color is not changed.

Once you select a new color and click the **Apply** button, the colors for the active visualization window change. Note that each visualization window can have different background and foreground colors.

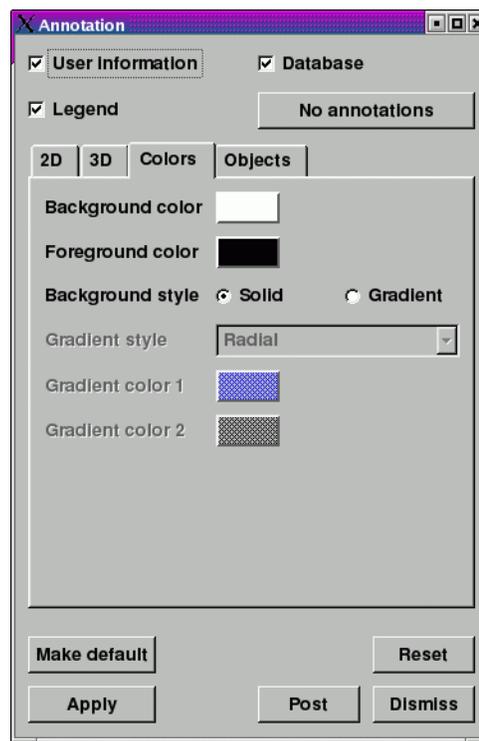


Figure 9-7: Colors tab

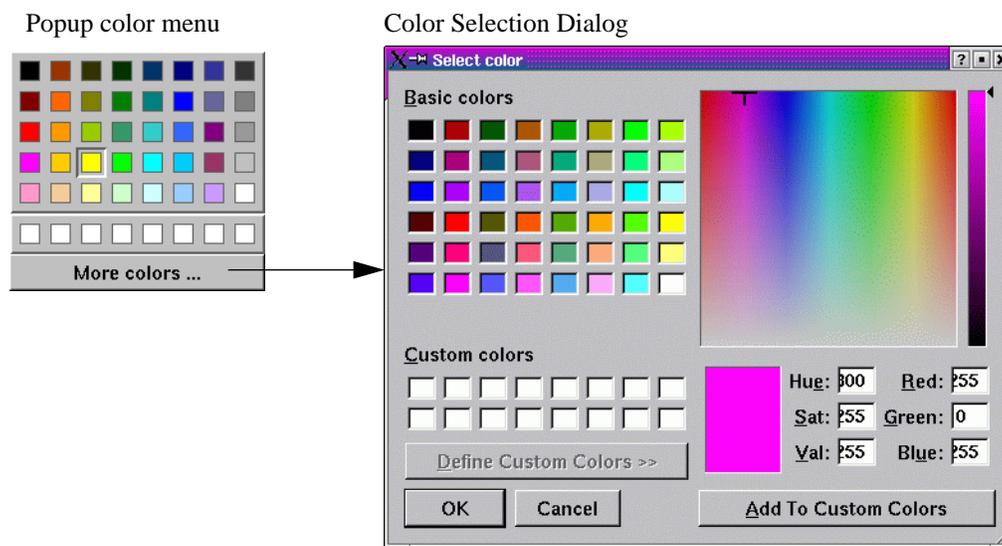


Figure 9-8: Popup color menu and the Color Selection Dialog

2.5.2 Changing the background style

VisIt has two possible background styles from which to choose. The default background style is solid where the entire background is a single color. Another background style is a gradient background. In a gradient background, two colors are blended into each other in various ways. The resulting background offers differing degrees of contrast and can enhance the look of many visualizations. To change the background style, click the **Background style** radio buttons in the **Annotation Window**. Choosing **Solid** selects a solid background while choosing **Gradient** selects a gradient background.

2.5.3 Customizing gradient backgrounds

VisIt provides controls for setting the colors and style used for gradient backgrounds. There are two color buttons: **Gradient color 1** and **Gradient color 2** that are used to change colors. To change the gradient colors, click on the color buttons and select a color from the **Popup color menu**. The gradient style is used to determine how colors blend into each other. To change the gradient style, make a selection from the **Gradient style** menu. The available options are Bottom to Top, Top to Bottom, Left to Right, Right to Left, and Radial. The first four options blend gradient color 1 to gradient color 2 in the manner prescribed by the style name. For example, Bottom to Top will have gradient color 1 at the bottom and gradient color 2 at the top. The radial gradient style puts gradient color 1 in the middle of the visualization window and blends gradient color 2 radially outward from the center. Examples of the gradient styles are shown in Figure 9-9.



Figure 9-9: Gradient styles

2.6 Annotation Objects

So far, the annotations that have been described can only have a single instance. To provide more flexibility in the types and numbers of annotations, VisIt allows you to create annotation objects, which are objects that are added to the visualization window to convey information about the visualization. Currently, VisIt supports four types of annotation objects: 2D text objects, time slider objects, 2D line objects, and image objects. All of those types of annotation objects will be described herein. The fourth tab, or

Objects tab, in the **Annotation Window** (Figure 9-10) is devoted to managing the list of annotation objects and setting their properties.

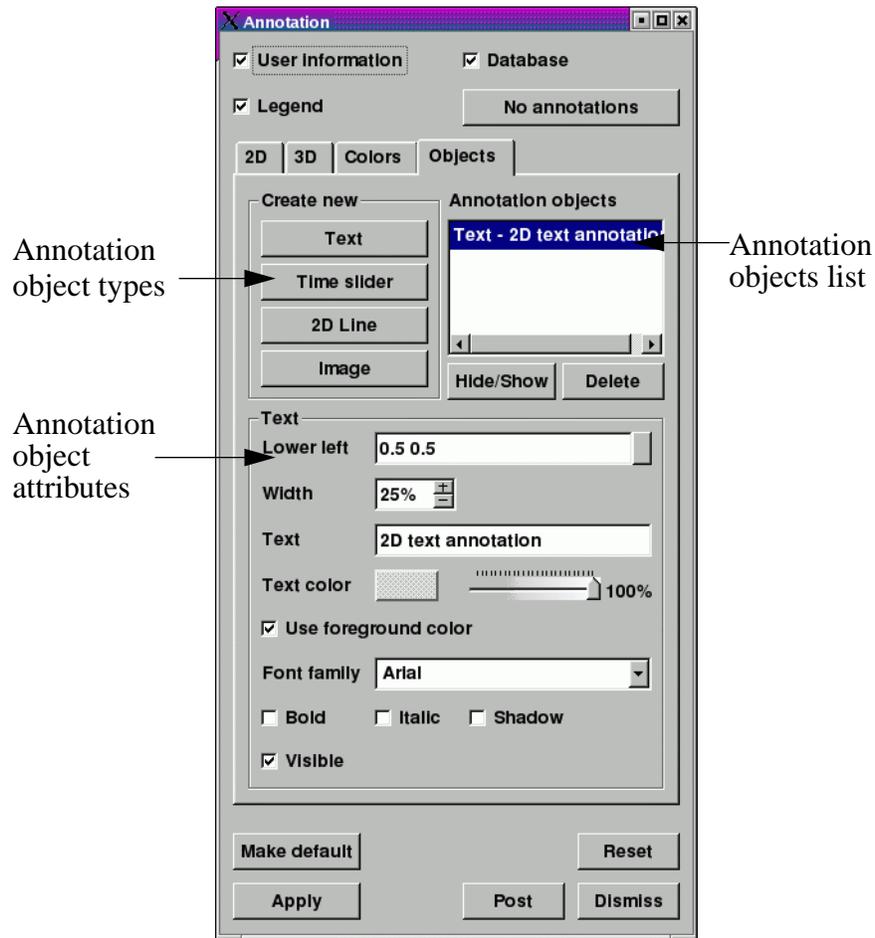


Figure 9-10: Annotation object tab

The **Objects** tab in the **Annotation Window** is divided up into three main areas. The top of the window is split vertically into two areas that let you create new annotation objects and manage the list of annotation objects. The bottom half of the **Objects** tab displays the controls for setting the attributes of the selected annotation object. Each annotation object provides a separate user interface that is tailored for setting its particular attributes. When you select an annotation in the annotation object list, the appropriate annotation object interface is displayed.

2.6.1 Creating a new annotation object

The **Create new** area in the **Annotation Window's Objects** tab contains one button for each type of annotation object that VisIt can create. Each button has the name of the type of annotation object VisIt creates when you push it. After pushing one of the buttons, VisIt creates a new instance of the specified annotation object type, adds a new entry to the

Annotation objects list, and displays the appropriate annotation object interface in the bottom half of the **Objects** tab to display the attributes for the new annotation object.

2.6.2 Selecting an annotation object

The **Objects** tab displays the annotation object interface for the selected annotation object. To set attributes for a different annotation object, or to hide or delete a different annotation object, you must first select a different annotation object in the **Annotation objects** list. Click on a different entry in the **Annotation objects** list to highlight a different annotation object. Once you have highlighted a new annotation object, VisIt displays the object's attributes in the lower half of the **Objects** tab.

2.6.3 Hiding an annotation object

To hide an annotation object, select it in the **Annotation objects** list and then click the **Hide/Show** button on the **Objects** tab. To show the hidden annotation object, click the **Hide/Show** button a second time. The interfaces for the currently provided annotation objects also have a **Visible** check box that can be used to hide or show the annotation object.

2.6.4 Deleting an annotation object

To delete an annotation object, select it in the **Annotation objects** list and then click the **Delete** button on the **Objects** tab. You can delete more than one plot if you select multiple plots in the **Annotation objects** list before clicking the **Delete** button.

2.6.5 Text annotation objects

Text annotation objects, shown in Figure 9-11, are created by clicking the **Text** button in the **Create new** area on the **Objects** tab. Text annotation objects are simple 2D text objects that are drawn on top of plots in the visualization window and are useful for adding titles or classification levels to a visualization. Text annotation objects can be placed

anywhere in the visualization window and you can set their size, text, colors, and font properties.

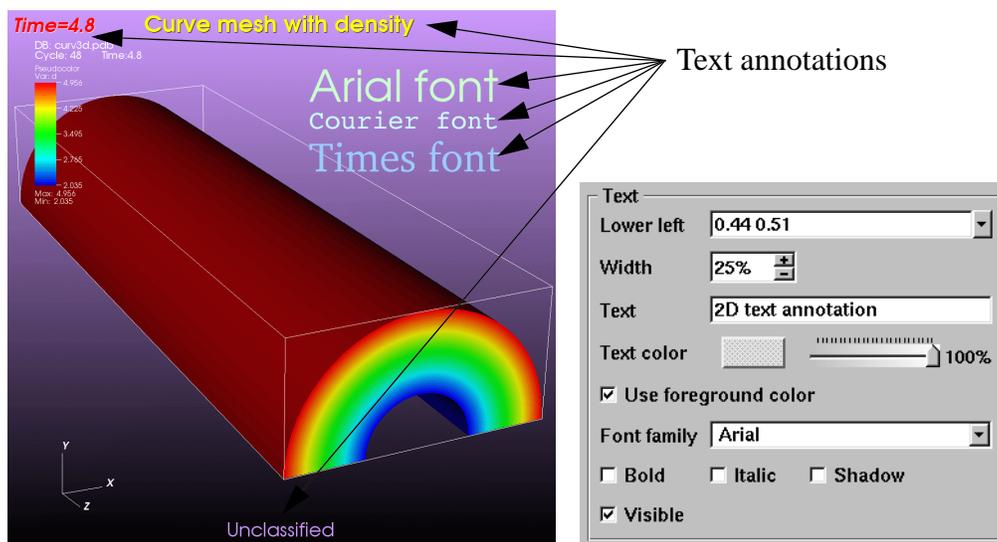


Figure 9-11: Text annotations and text annotation interface

Text annotation objects are placed using 2D coordinates where the X, and Y values are in the range [0,1]. The point (0,0) corresponds to the lower left corner of the visualization window and the point (1,1) corresponds to the upper right of the visualization window. The 2D coordinate used to position the text annotation matches the text annotation's lower left corner. To position a text annotation object, enter a new 2D coordinate into the **Lower left** text field. You can also click the down arrow next to the **Lower left** text field to interactively choose a new lower left coordinate for the text annotation using the screen positioning control, which represents the visualization window. The screen positioning control, shown in Figure 9-12, lets you move a set of cross-hairs to any point on a square area that represents the visualization window. Once you release the left mouse button, the location of the cross-hairs is used as the new coordinate for the text annotation object's lower left corner.

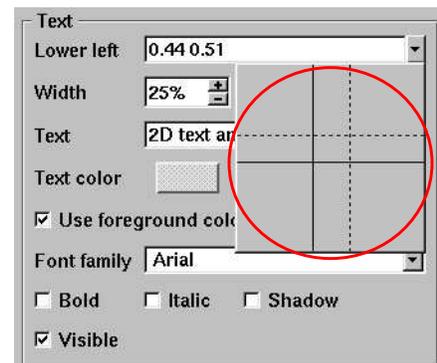


Figure 9-12: Screen positioning control

Text annotations objects are currently sized using a percentage of the visualization window's width. When you specify a width for the text annotation object, you are setting its maximum width. To set the width for a text annotation, type a new width value into the **Width** spin box or use its the +/- arrows to increase or decrease the size of the text annotation object. The height of the text depends on the length and composition of the text

that the text annotation will display. Text annotation objects will likely be changed in the near future so they are specified in terms of visualization window height instead of width so it is easier to make different text annotation objects have the same font size when they display different lines of text.

To set the text that a text annotation object displays, type a new string into the **Text** text field. You can make the text annotation object display any characters that you type in but you can also use the *\$time* wildcard string to make the text annotation object display the time for the current time state of the active database. A text string of the form: *Time=\$time* will display *Time=10* in the visualization window when the active database's time is 10. Whatever text you enter for the text annotation object is used to identify the text annotation object in the **Annotation objects** list.

Text annotation objects can be displayed in any color, including the visualization window's foreground color. You can also set the opacity for text annotation objects to make them transparent. If you want to set the color for a text annotation object, you must first turn off the **Use foreground color** check box. Once that check box is turned off, the text annotation uses the color that you pick for it instead of the visualization window's foreground color. To change the color for a text annotation object, click the **Text color** button and choose a new color from the **Popup color menu**. To change the opacity, use the opacity slider next to the **Text color** button.

In addition to being able to set the position, size, message, and color for the text annotation object, you can also choose from 3 different fonts (Arial, Courier, and Times) and set certain font display properties such as bold, italic, and shadow.

2.6.6 Time slider annotation objects

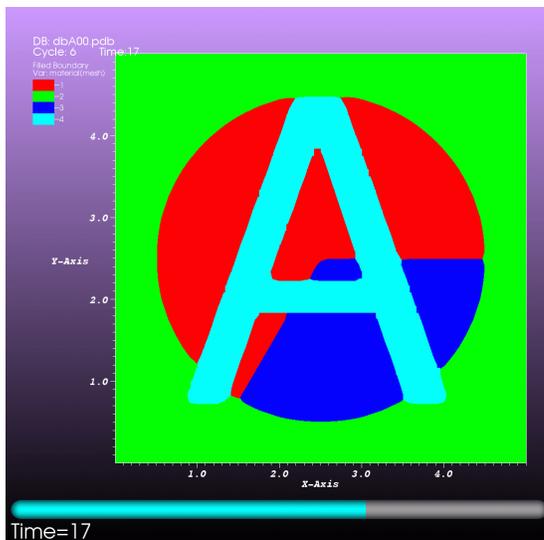


Figure 9-13: Time slider annotation object

Time slider annotation objects, shown in Figure 9-13, are created by clicking the Time slider button in the **Create new** area on the **Objects** tab. Time slider annotation objects consist of a graphic that shows the progress through an animation using animation and text that shows the current database time. Time slider annotation objects can be placed anywhere in the visualization window and you can set their size, text, colors, and appearance properties.

Time slider annotation objects are placed using 2D coordinates where the X, and Y values are in the range [0,1]. The point (0,0) corresponds to the lower left corner of the visualization window and the point (1,1) corresponds to the upper right of the visualization window. The 2D coordinate used to position the text annotation matches the text annotation's lower left corner. To position a text annotation object, enter a new 2D coordinate into the **Lower left** text field. You can also click the down arrow next to the **Lower left** text field to interactively choose a new lower left coordinate for the text annotation using the screen positioning control, which represents the visualization window.

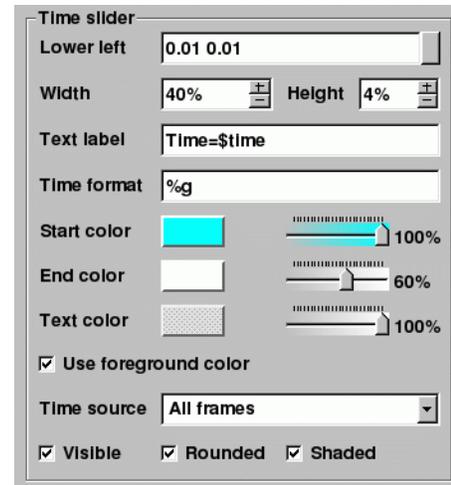


Figure 9-14: Time slider object interface

The size of a time slider annotation object is controlled by settings its height and width as a percentage of the vis window height and width. Type new values into the **Width** and **Height** spin buttons or use the +/- arrows next to the **Width** and **Height** spin buttons to set a new width or height for the time slider annotation object.

You can set the text displayed by the time slider annotation object by typing a new text string into the **Text label** text field. Text is displayed below the time slider annotation object and it can contain any message that you want. The text can even include wildcards such as *\$time*, which evaluates to the current time for the active database. If you use *\$time* to make VisIt incorporate the time for the active database, you can also specify the format string used to display the time. The format string is a standard C-language format string (e.g. "%4.6g") and it determines the precision used to write out the numbers used in the time string. You will probably want to specify a format string that uses a fixed number of decimal places to ensure that the time string remains the same length during the animation, preventing distracting differences in the length of the string from taking the eye away from the visualization. Type a C-language format string into the **Time format** text field to change the time format string.

Time slider annotations have three color attributes: start color, end color, and text color. A time slider annotation object displays time like a progress bar in that the progress bar starts out small and then grows to the right until it takes up the whole length of the annotation. The color used to represent the progress can be set by clicking the **Start color** button and choosing a new color from the **Popup color menu**. As the time slider annotation object shows more progress, the color that is used to fill up the time that has not been reached yet (end color) is overtaken by the start color. To set the end color for the time slider annotation object, click the **End color** button and choose a new color from the **Popup color menu**. Normally, time slider annotation objects use the foreground color of the vis window when drawing the annotation's text. If you want to make the annotation use a special color, turn off the **Use foreground color** check box and click the **Text color** button and choose a new color from the **Popup color menu**.

Time slider objects have two more attributes that affect their appearance. The first of those attributes is set by clicking on the **Rounded** check box. When a time slider annotation object is rounded, the ends of the annotation are curved. The last attribute is set by clicking on the **Shaded** check box. When a time slider annotation object is shaded, simple lighting is applied to its geometry and the annotation will appear to be more 3-dimensional.

2.6.7 2D line annotation objects

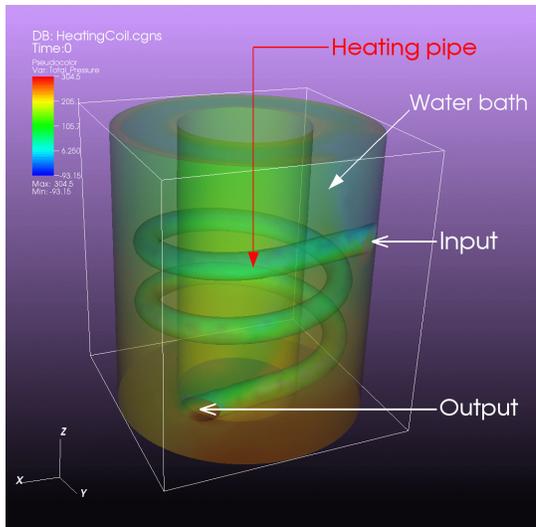


Figure 9-15: 2D line annotations and text annotations can be used to point to features of interest

2D line annotation objects, shown in Figure 9-15, are created by clicking the **2D Line** button in the **Create new** area on the **Objects** tab. 2D line annotation objects are simple line objects that are drawn on top of plots in the visualization window and are useful for pointing to features of interest in a visualization. 2D line annotation objects can be placed anywhere in the visualization window and you can set their locations, arrow properties, and color.

2D line annotations are described mainly by two coordinates that specify the start and end points for the line. The start and end coordinates are specified as pairs of floating point numbers in the range [0,1] where the point (0,0) corresponds to the lower left corner of the visualization window and the

point (1,1) corresponds to the upper right corner of the visualization window. You can set the start or end points for the 2D line annotation by entering new start or end points into the **Start** or **End** text fields in the 2D line object interface. You can also click the down arrow to the right of the **Start** or **End** text fields to interactively choose new coordinates using the screen positioning control.

Once the 2D line annotation has been positioned there are other attributes that can be set to improve its appearance. First of all, if the 2D line annotation is being used to point at important features in a visualization, you might want to increase the 2D line annotation's width to make it stand out more. To change the width, type a new number of pixels into the **Width** spin box or use the +/- buttons to increment or decrement the current width. After changing the width, the color of the 2D line annotation should be chosen to stand out against the plots in the visualization. The color that you use

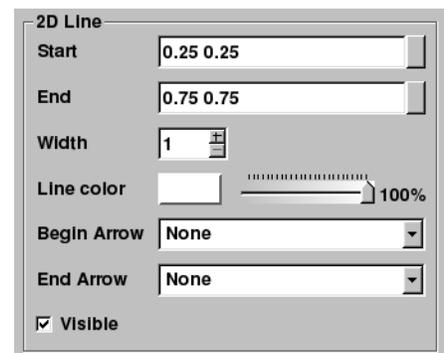


Figure 9-16: 2D line object interface

should be chosen such that the line contrasts sharply with the plots over which it is drawn. To choose a new color for the line, click on the **Line color** button and choose a new color from the **Popup color menu**. You can also adjust the opacity of the line by using the opacity slider next to the **Line color** button.

The last properties that are commonly set for 2D line annotations determine whether the end points of the line have arrow heads. The 2D line annotation supports two different styles of arrow heads: filled and lines. To make your line have arrow heads at the start or the end, make new selections from the **Begin Arrow** and **End Arrow** menus.

2.6.8 Image annotation objects

Image annotation objects, shown in Figure 9-11, are created by clicking the **Image** button in the **Create new** area on the **Objects** tab. Image annotation objects display images from image files on disk in a visualization window. Images are drawn on top of plots in the visualization window and are useful for adding logos, pictures of experimental data, or other views of the same visualization. Image annotation objects can be placed anywhere in the visualization window and you can set their size, and optional transparency color.

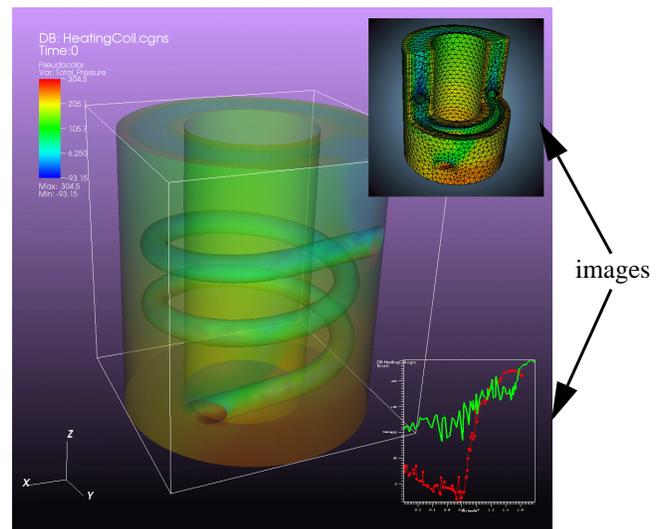


Figure 9-17: Visualization with two overlaid image annotations

The first step in incorporating an image annotation into a visualization is to choose the file that contains the image that will serve as the annotation. To choose an image file for the image annotation, type in the full path and filename to the file that you want to use into the **Image source** text field. You can also use the file browser to locate the image file if you click on the “...” button to the right of the **Image source** text field in the **Image annotation interface**, shown in Figure 9-18. Note that since image annotations are incorporated into a visualization inside of VisIt’s viewer component, the image file must be located on the same computer that runs the viewer.

After selecting an image file, you can position its lower left coordinate in the visualization window. The lower left corner of the visualization window is the origin (0,0) and the upper right corner of the visualization window is (1,1).

Once you position the image where you want it, you can optionally scale it relative to its original size. Unlike some other annotation objects, the image annotation does not scale automatically when the visualization window changes size. The image annotation will

remain the same size - something to take into account when setting up movies that use the image annotation. To scale the image relative to its original size, enter new percentages into the **Width** and **Height** spin boxes or click their +/- buttons. If you want to scale one dimension of the image and let the other dimension remain unchanged, turn off the **Lock aspect** check box.

Finally, if you are overlaying an image annotation whose image contains a constant background color or other area that you want to remove, you can pick a color that VisIt will make transparent. For example, Figure 9-18 shows an image of some Curve plots overlaid on top of the plots in the visualization window and the original background color in the annotation object was removed to make it transparent. If you want to make a color in an image transparent before VisIt displays it as an image annotation object, click on the **Transparent color** check box and then select a new color by clicking on the **Transparent color** button and picking a new color from the **Popup color menu**.

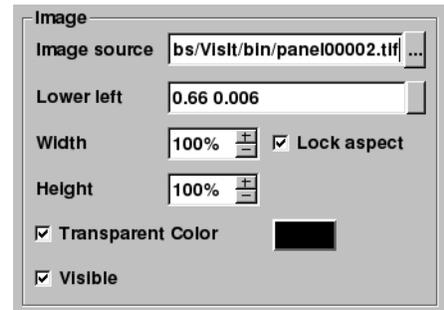


Figure 9-18: Image object interface

3.0 Color tables

A color table is a set of colors that is used by certain plots to color variables. Color tables can be immensely important for understanding visualizations since changes in color can highlight interesting features. VisIt has several built-in color tables that can be used in visualizations. VisIt also provides a **Color table window** for designing custom color tables.

Color tables come in two types: continuous and discrete. A continuous color table is defined a relatively few color control points defined at certain intervals in the color table and the gaps in between the color control points are filled by smoothly interpolating the colors. This makes continuous color tables look smooth since there are several colors that are blended to form the color table. Continuous color tables are used by several plots including the Pseudocolor, Streamline, Tensor, and Vector plots. A plot that uses a continuous color table attempts to use all of the colors in the color table. Some plots that opt to only use a handful of colors from a continuous color table pick colors that are evenly distributed through the color table so that the plots end up with colors that still somewhat resemble the original colors from the continuous color table.

A discrete color table is a set of N colors that can be set individually. There are no other colors in a discrete color table other than the colors that you provide. Discrete color tables are usually used by plots like the Boundary, Contour, FilledBoundary, or Subset plots, which need only a small set of colors. Typically, these plots use a color from a discrete color table to color some object and then they use the next color to color another object,

and so on. When they reach the end of the color table and still need more colors, they start again at the beginning with the first color from the discrete color table.

3.1 Color table window

You can open VisIt's **Color table window**, shown in Figure 9-19, by selecting the **Color table** option from the **Main Window's Controls** menu. The **Color table window** is vertically separated into two main areas. The top area, or manager portion of the window, allows you to set the active color table as well as create or delete new color tables. The bottom area, or editor portion of the window, allows you to edit color tables by adding, removing, moving, or changing the color of color control points. A color control point is a point with a color that influences how the color table will look.

3.1.1 Setting the active color table

VisIt has the concept of active color tables, which are the color tables used to color plots that do not specify a color table. There is both an active continuous color table (for plots that prefer to use continuous color tables) and an active discrete color table (for plots that prefer to use discrete color tables). The active color table can be different for each visualization window. To set the active continuous color table, select a new color table name from the **Continuous** menu in the **Active color table area**. To select a new active discrete color table, select a new color table name from the **Discrete** color table menu in the **Active color table area**.

3.1.2 Creating a new color table

Creating a new color table is a simple process where you first type a new color table name into the **Name** text field and then click the **New** button. This creates a copy of the currently highlighted color table, which is the color table that is selected in the **Manager area**, and inserts it into the color table list with the specified name. After creating the new color table, you can modify the color control points to fashion a new color table.

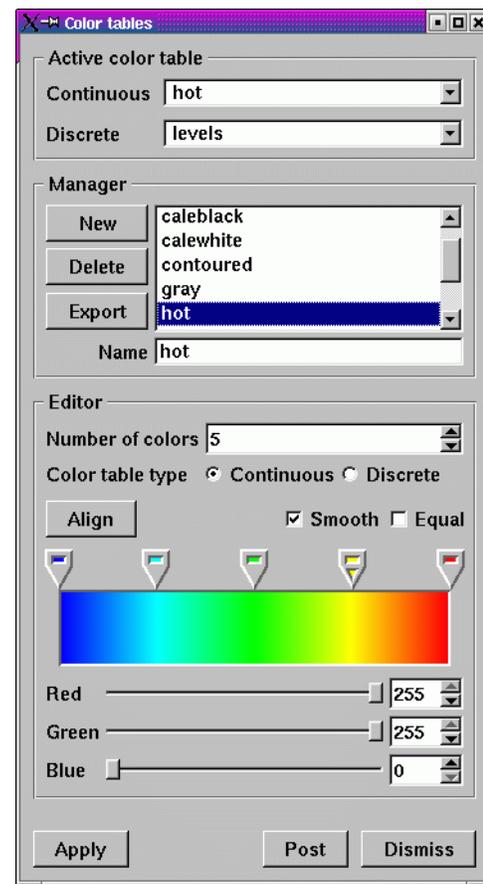


Figure 9-19: Color table window

3.1.3 Deleting a color table

To delete a color table, click on a color table name in the color table list and then click the **Delete** button. You can delete all color tables except for the last color table. VisIt makes no distinction between built-in color tables and user-defined color tables so any color table can be deleted. When you delete a color table, the active color table is set to the color table that comes first in the list. If a color table is in use when it is deleted, plots that used the deleted color table will use the default color table from that point on.

3.1.4 Exporting a color table

If you design a color table that you want to share with colleagues, click the **Export** button in the **Manager area** to save an XML file containing color table definition for the highlighted color table to your .visit directory. The name of a color table file will usually be composed of the name of the color table with a “.ct” extension. Copying a color table file to a user’s .visit directory will allow VisIt to find the color table the next time VisIt runs. Look for the color table file in the directory in which VisIt was installed if you use the Windows version of VisIt.

3.1.5 Editing a continuous color table

There are a handful of controls in the editor portion of the **Color table window**, shown in Figure 9-20, that are used to change the definition of a color table. To change a color table definition, you must alter its color control points. This means adding and removing color control points as well as changing their colors and locations.

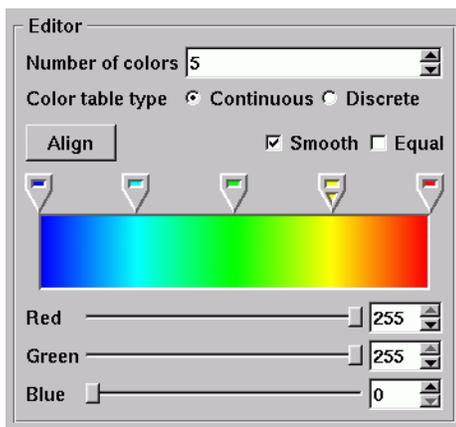


Figure 9-20: Continuous color table editor

To change the number of color control points in a color table, type a new number of color control points into the Number of colors spin box or use the up/down arrows to increase or decrease the number of color control points. When a new color control point is added, it appears to the right of the selected color control point and to the left of the next color control point. Color control points are represented as a pointy box just above the color spectrum. The color control point that has a small triangular mark is the selected color control point. When a color control point is removed, the color control point that was created before the deleted color control point becomes the new selected color control point. Clicking the **Align** button makes all color control points have equal spacing.

Clicking on a color control point makes it active. You can also use the *Space bar* if the color spectrum has keyboard focus. Clicking and dragging on a color control point changes its position. Clicking the arrow keys on the keyboard also moves a color control

point. To change a color control point's color, right click on it and choose a new color from the **Popup color menu** that appears under the mouse cursor.

The **Color table window** also has a couple of toggles that can be set to influence a color table's appearance without having permanent effects on the color table. The **Smooth** toggle turns on color interpolation between color control points. When this toggle is enabled, colors are smoothly blended into each other. The **Equal spacing** toggle can temporarily tell the color table to ignore the positions of its color control points and use equal spacing instead. The **Equal spacing** toggle is often used with **Smooth** turned off.

3.1.6 Editing a discrete color table

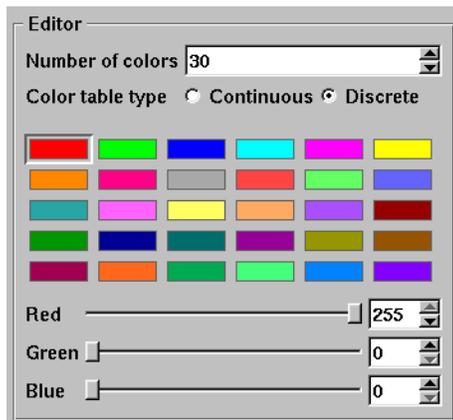


Figure 9-21: Discrete color table editor

The **Color table window's Editor area** looks different when you edit a discrete color table. Instead of showing a spectrum of colors, the window shows a grid of colors that correspond to the colors in the discrete color table. The order of the color control points if left to right, top to bottom. To edit a discrete color table, first left click on the color that you want to edit and then use the **Red**, **Green**, and **Blue** sliders to change the color. You can also right click on a color to select it and also open the **Popup color menu**. so you can choose a new color.

3.1.7 Converting color table types

It is possible to convert a continuous color table to a discrete color table and vice-versa using the **Continuous** and **Discrete** radio buttons in the editor portion of the **Color table window**. Changing the color table type from discrete to continuous does not change the color table's color control points; it only changes how they are used. If you select the levels color table and click the **Continuous** radio button, the color table will be changed into a continuous color table and the **Editor area** will change to continuous mode and show the color table in a spectrum but no color control points will have changed. You can even turn the color table back into a discrete color table and the **Editor area** will show the color table in discrete mode, but the color control points will not have changed.

4.0 Lighting

Lighting is an important element when producing 3D visualizations because all areas of interest in the visualization should be lit so they can be easily seen. To this end, it is often necessary to have multiple light sources so all of the visualization's important areas are bright enough. VisIt can have up to 8 light colored light sources in order to improve the look of 3D visualizations. Each light source can be positioned and colored using VisIt's

Lighting Window. It is also possible to have specular highlights in addition to multiple colored lights. For more information on specular highlights, which can make visualizations appear much more realistic, read about specular lighting in the **Preferences** chapter.

4.1 Lighting Window

You can open the **Lighting Window** (see Figure 9-22) by selecting the **Lighting** option from the **Main Window's Controls** menu. The **Lighting Window** has two modes of operation: edit and preview. When the window is in preview mode, light sources cannot be modified, but they are all visible and illuminate the **Lighting Window's** test sphere so the cumulative effect of the lights can be observed. When the window is in edit mode, light sources can be modified one at a time. You set light properties using the controls in the **Properties** panel and you can position lights interactively by moving them around in the lighting panel to the left of the **Properties** panel.

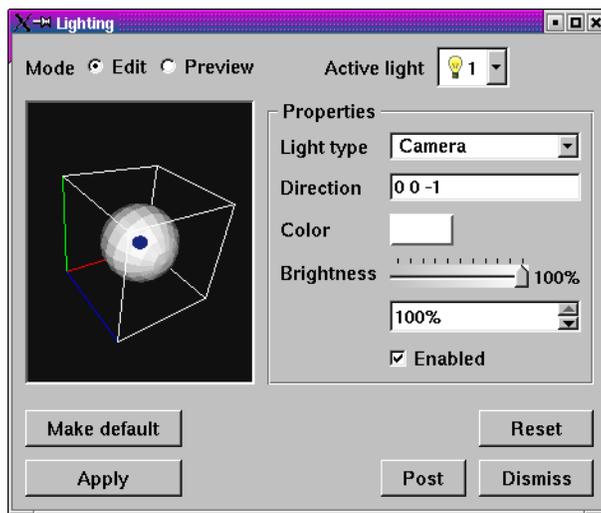


Figure 9-22: Lighting Window

4.1.1 Switching between edit mode and preview mode

Clicking the radio buttons for **Edit** or **Preview** switches the **Lighting Window** into the desired mode. When the **Lighting Window** is in edit mode, one light source at a time is shown in the lighting panel and the lights properties can be set by moving the light interactively or by setting its properties by using the controls in the **Properties** panel. When the **Lighting Window** is in preview mode, all lights are shown in the lighting panel and none of them can be modified.

4.1.2 Choosing the active light

The active light is the light whose properties are shown in the **Lighting Window**. Only the active light can be modified so you must switch active lights each time you want to make changes to a light. To change the active light, select a new light from the **Active light** pulldown (Figure 9-23). The **Active light** pulldown contains a list of eight possible lights of which only light 1 is active by default. When a light is active, it has a small light bulb icon next to it. Inactive lights have no light bulb icon. Once a new light has been selected from the **Active light** pulldown, its properties are displayed in the **Lighting Window's Properties** panel.

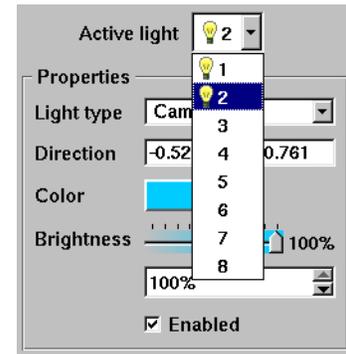


Figure 9-23: Active light pulldown

4.1.3 Turning a light on

You can turn lights on and off using the **Enabled** check box that appears at the bottom of the **Lighting Window's Properties** panel. You can only modify lights when the **Lighting Window** is in edit mode.

4.1.4 Light type

VisIt supports three types of lights. The first type is called an ambient light. An ambient light is a light that has no direction and contributes brightness to the entire visualization. When an ambient light is present, the lighting panel displays a small light bulb. The second type of light and the default light in VisIt is a camera light. A camera light stays fixed in space and always points the same direction regardless of how the objects in the visualization are positioned. Camera lights are represented in the lighting panel as small blue arrows. The third type of light in VisIt is the object light. An object light has a direction that is relative to the orientation of the object in the visualization. When the objects in the visualization are rotated, an object light keeps shining on the same area of the object. Object lights are represented in the lighting panel as small yellow cones. To change the light type for the active light, select a new light type from the **Light type** menu in the **Properties** panel.

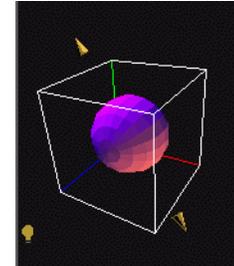


Figure 9-24: Different kinds of lights

4.1.5 Positioning a light

There are two ways to position a light. The first, and most intuitive, way is to interactively position the light by dragging it to the desired location in the lighting panel. Lights move in a sphere around the test sphere. Experiment with the motion until you are comfortable moving the light. The second way to move the light is to type a direction vector into the

Direction text field. The coordinate system for specifying a direction vector is right-handed. Suppose you want to create a light that looks directly into the visualization. Since the Z-axis points directly out of the screen, the negative Z-axis points into the screen. This can be captured by entering a direction vector of: $0\ 0\ -1$. Note that ambient lights have no direction.

4.1.6 Light color and brightness

VisIt allows lights to have colors as well as brightnesses. Colored lighting can produce interesting effects that may be desirable for presentations. To change the light color, click on the light **Color** button and select a new color from the **Popup color menu**. Once a color is picked, you can also set the brightness for the light. The brightness is essentially a knob that allows you to dim the light. If the brightness is set completely to the right then the light will have exactly the color that was picked for it. If the brightness is not set to full intensity then the light will be dimmer. You can set the brightness by adjusting the **Brightness** slider in the **Lighting Window**.

5.0 Rendering Options

VisIt provides support for setting various global rendering options that improve quality and realism of the plots in the visualization. Specifically, VisIt provides controls that let you smooth the appearance of lines, add specular highlights, and add shadows to plots in your visualizations. The controls for setting these options are located in the **Rendering Options Window** (see Figure 9-25) and they will be covered here while other controls in that window will be covered in the **Preferences** chapter on page 317. To open the **Rendering Options Window**, click on the **Rendering** option in the **Main Window's Preferences** menu.

5.1 Making lines look smoother

Computer monitors contain an array of millions of tiny rectangular pixels that light up to form patterns which your eyes perceive as images. Lines tend to look blocky on computer monitors because they are drawn using a relatively small set of pixels. Lines can be made to look better by blending the edges of the line with the color of the background image. This is a form of antialiasing that VisIt can use to make plots which use lines, such as the Mesh plot, look

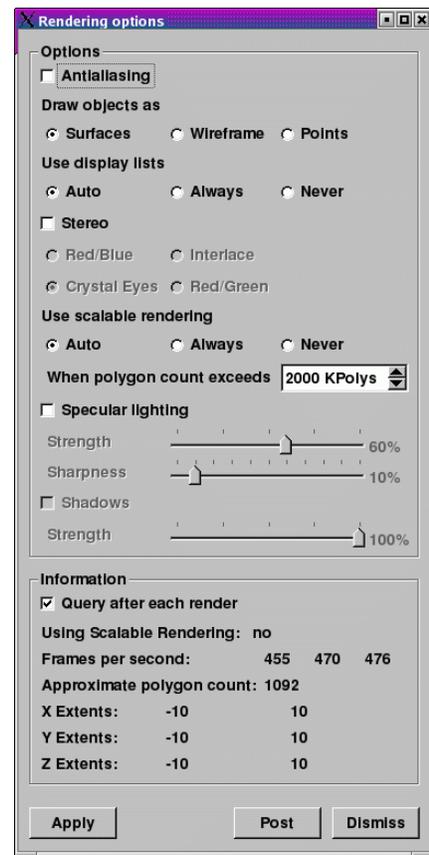


Figure 9-25: Rendering Options Window

better (see Figure 9-26). If you want to enable antialiasing, which is off by default, you check the **Antialiasing** check box in the **Rendering Options Window**. When antialiasing is enabled, all lines drawn in a visualization window are blended with the background image so that they look smoother.

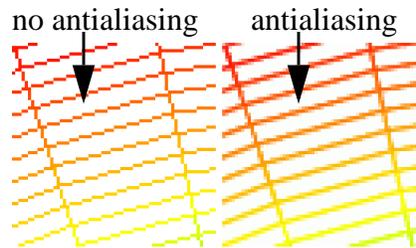


Figure 9-26: Antialiasing example

5.2 Specular lighting

VisIt supports specular lighting, which results in bright highlights on surfaces that reflect a lot of incident light from VisIt's light sources. Specular lighting is not handled in the **Lighting Window** because specular lighting is best described as a property of the material reflecting the light. The controls for specular lighting don't control any lights but instead control the amount of specular highlighting to caused by the plots. Specular lighting is not enabled by default. To enable specular lighting, click the **Specular lighting** check box in the **Rendering Options Window**.

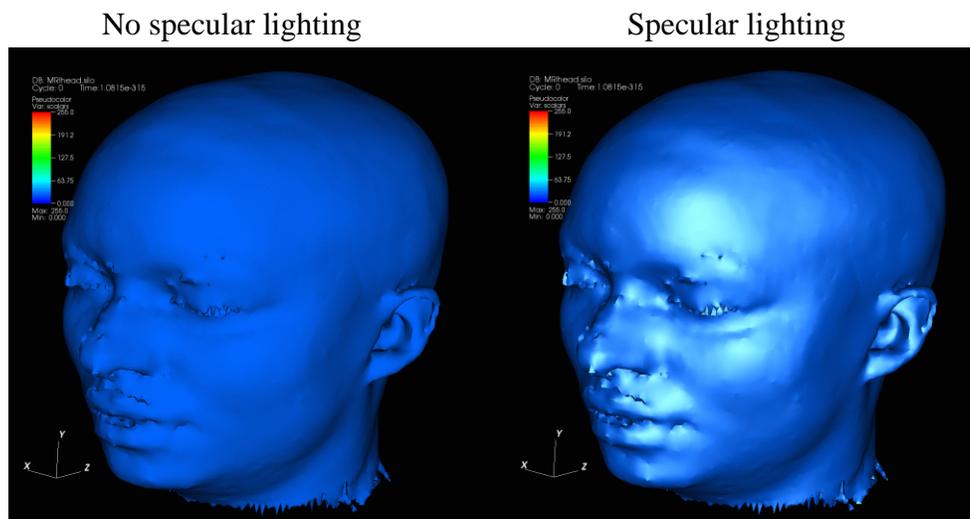


Figure 9-27: Effects of specular lighting on plots

Once specular lighting is enabled, you can change the strength and sharpness properties of the material reflecting the light. The strength, which you can set using the **Strength** slider, influences how glossy the plots are and how much light is reflected off of the plots.

The sharpness, which is set using the **Sharpness** slider, controls the locality of the reflections. Higher the sharpness values result in smaller specular highlights. Specular highlights are a crucial component of lighting models and including specular lighting in your visualizations enhances their appearance by making them more realistic. Compare and contrast the plots in Figure 9-27. The plot on the left side has no specular highlights and the plot on the right side has specular highlights.

5.3 Shadows

VisIt supports shadows when scalable rendering is being used. Shadows can be useful for increasing the realism of your visualization. The controls to turn on shadows can be found in the **Rendering Options Window**. To turn on shadows, you must currently turn on scalable rendering by clicking on the **Always** radio button under the **Use scalable rendering** label. Once scalable rendering has been turned on, the Shadows controls become enabled. The default shadow strength is 50%. If you desire a stronger or weaker shadow, adjust the **Strength** slider until you are satisfied with the amount of shadow that appears in the visualization. The same plot is shown with and without shadows in Figure 9-28.

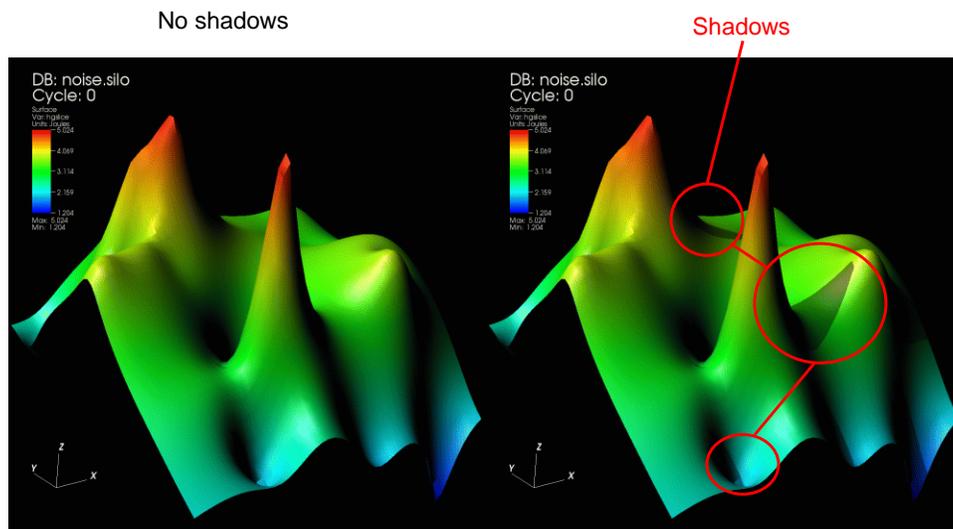


Figure 9-28: Effects of shadow on plots

6.0 View

The view is one of the most critical properties of a visualization since it determines what parts of the dataset are seen. The view is also one of the most difficult properties to set. It isn't that the act of setting the view is difficult, in fact, it is quite the opposite. The problem with setting the view is finding a flattering view for a database that will continue to be a good view for the entire life of the visualization. Many plots will deform or expand over the course of an animation and you have to decide how to pick a good view. You can pick

a view that is zoomed way out and then let your plots expand and deform until they make good use of the visualization window. You can also decide to keep changing the view throughout the animation. A common technique is to interpolate views or do some sort of fly-by animation when the plots in the animation are expanding or not behaving in a static manner. The fly-by animation is used to distract the audience from the fact that you need to change to a more suitable view.

The view in VisIt can be set in two different ways. The first and best way to set the view is to navigate to it interactively in the visualization window. This is the fastest and most direct way of setting the view. The problem with setting the view in this manner is that it is not very reproducible. It is often the case that users want to look at the same feature in their database using the same view. VisIt provides a **View Window** that they can use to set the view information exactly the same every time.

6.1 View Window

You can open the **View Window** by selecting the **View** option from the **Main Window's Controls** menu. The **View Window** is divided into three tabbed sections. The first tab sets the 2D view, the second tab sets the 3D view, and the last tab sets advanced view options.

6.1.1 Setting the Curve view

Visualization windows that contain Curve plots use a special type of view known as a curve view. A curve view consists of: viewport, domain, and range. The viewport is the area of the visualization window that will be occupied by the plots and is specified using X and Y values in the range $[0,1]$. The point $(0,0)$ corresponds to the lower-left corner of the visualization window while the point $(1,1)$ corresponds to the visualization window's upper-right corner. To change the viewport, type new numbers into the **Viewport** text field on the **Curve tab** of the **View Window** (Figure 9-29). The minimum and maximum X values should come first, followed by the minimum and maximum Y values.

The domain and range refer to the limits on the X and Y axes. You can set the domain, which is the range of X values that will be displayed in the viewport, by typing new minimum and maximum values into the **Domain** text field. You should use domain values that use the same dimensions as the Curve plot that will be plotted in the visualization window. You

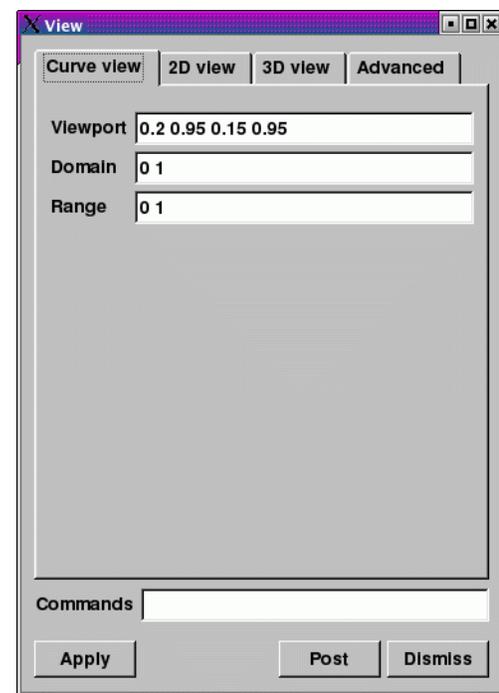


Figure 9-29: Curve view options

can set the range, which is the range of Y values that will be displayed in the viewport, by typing new values into the **Range** text field.

6.1.2 Setting the 2D view

Setting the 2D view is conceptually simple. There are only two pieces of information that you need to supply. The first piece of information that you must enter is the viewport, which is an area of the visualization window in which you want the 2D plots to appear. Imagine that the lower left corner of the visualization window is the origin of a coordinate system and that the upper left and lower right corners both have values of 1. Every point in the visualization window can be characterized as a cartesian coordinate where both values in the coordinate are in the range [0,1]. The viewport is specified by entering four numbers in the form $x0\ x1\ y0\ y1$ where $x0$ is the leftmost X value, $x1$ is the rightmost X value, $y0$ is the lower Y value, and $y1$ is the upper Y value that will be used in the viewport. The window is an area in the space occupied by the 2D plots. You can start with a window that is the same size as the plot's spatial extents and then zoom in from there by making the window values smaller and smaller. The window values are also of the form $x0\ x1\ y0\ y1$. To change the 2D view, type new values into the **Viewport** and **Window** text fields on the **View Window's 2D view tab** (Figure 9-30).

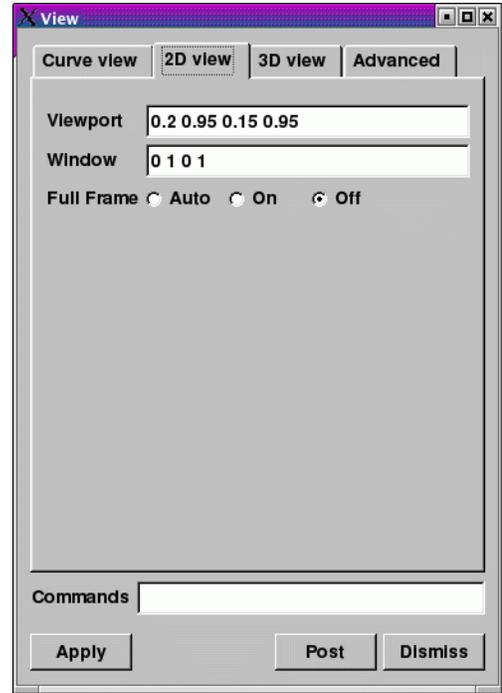


Figure 9-30: 2D view options

6.1.3 Fullframe mode

Some databases yield plots that are so long and skinny that they leave most of the vis window blank when VisIt displays them. A common example is equation of state data, which often has at least 1 exponential dimension. VisIt provides Fullframe mode to stretch long, skinny plots so they fill more of the vis window so it is easier to see them. It is worth noting that Fullframe mode does not preserve a 1:1 aspect ratio for the displayed plots because they are stretched in

Y dimension is so much larger that X diminishes to a line FullFrame mode engaged

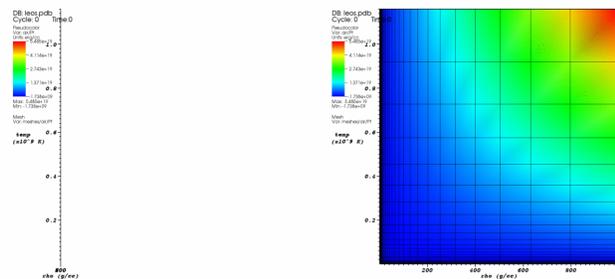


Figure 9-31: Effect of Fullframe mode on an extremely skinny plot

each dimension so they fit better in the vis window. To activate Fullframe mode, click on the **Auto** or **On** buttons in the **View Window**. When Fullframe mode is set to **Auto**, VisIt determines the aspect ratio of the X and Y dimensions for the plots being visualized and automatically scales the plots to fit the window when extents for one of the dimensions are much larger than the extents of the other dimension.

6.1.4 Setting the 3D view

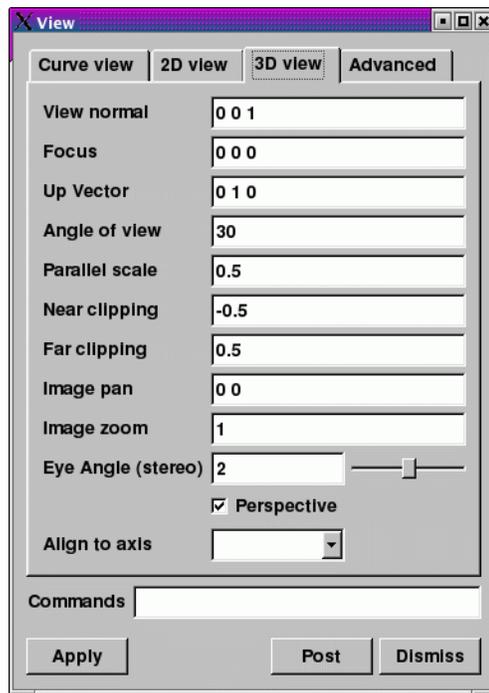


Figure 9-32: 3D view options

Setting the 3D view using controls in the **View Window's 3D view tab** (see Figure 9-32) demands an understanding of 3D views. A 3D view is essentially a location in space (*view normal*) looking at another location in space (*focus*) with a cone of vision (*view angle*). There are also clipping planes that lie along the view normal that clip the near and far objects from the view. Figure 9-33 depicts the various components of a 3D view.

To set the 3D view, first decide on where you want to look from. Type a vector value into the **View normal** text field. Next, type the vector valued location of what you want to look at into the **Focus** text field. The **Up axis** vector is simply a vector that determines which way is up. A good default value for the up axis is $0\ 1\ 0$. VisIt will often calculate a better value to use for the up axis so it is not too important to figure out the right value. The **View Angle** determines how wide the field of view is. The view angle is

specified in degrees and a value around 30 is usually sufficient. **Near clipping** and **Far clipping** are values along the view normal that determine where the near and far clipping planes are to be placed. It is not easy to know that good values for these are so you will have to experiment **Parallel scale** acts as a zoom factor and larger values zoom the camera towards the focus. The **Perspective** check box applies to 3D visualizations and it causes a more realistic view to be used where objects that are farther away are drawn smaller than closer objects of the same size. VisIt uses a perspective view for 3D visualizations by default.

VisIt supports stereo rendering, during which VisIt draws the image in the visualization window twice with the camera eye positioned in slightly different locations to mimic the differences in images seen by your left eye and your right eye. With the right stereo goggles, the image that you see appears to hover in 3D space within your monitor since the effect of the stereo image adds much more depth to the visualization. You can set the angle that VisIt uses to separate the cameras used to draw the images by typing a new angle into the **Eye angle** text field or by using the **Eye angle** slider.

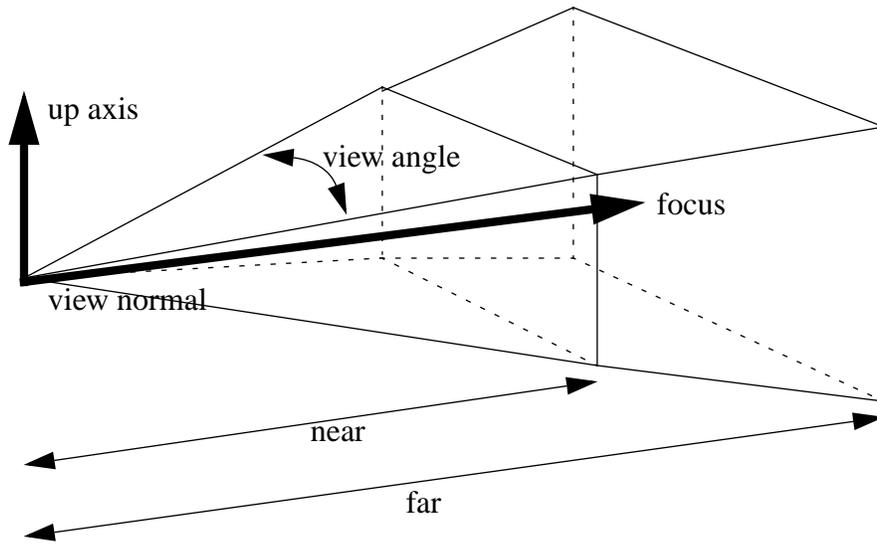


Figure 9-33: 3D perspective view volume

6.1.5 Using axis alignment buttons

The axis alignment buttons in the **3D view tab** set the view normal, which is the direction that the view points, so it is aligned with one of the 3D axes. Clicking them provides a convenient way to get side, top, and bottom views of the plots in the visualization window.

6.1.6 Using view commands

The **View Window** contains a **Commands** text field that allows you to enter one or more semi-colon delimited legacy MeshTV commands to change the view. The following table has a description of the supported view commands:

Command	Description
pan x y	Pans the 3D view to the left/right or up/down. The x, y arguments, which are floating point fractions of the screen in the range [0,1], determine how much the view is panned in the X and Y dimensions.
pan3 x y	Same as pan.
panx x	Pans the 3D view left or right. The x argument is a floating point fraction of the screen in the range [0,1].
xtrans x	Same as panx.
pany y	Pans the 3D view up or down. The y argument is a floating point fraction of the screen in the range [0,1].
ytrans y	Same as pany.

Command	Description
rotx x	Rotates the 3D view about the X-axis x degrees.
rx x	Same as rotx.
roty y	Rotates the 3D view about the Y-axis y degrees.
ry y	Same as roty.
rotz z	Rotates the 3D view about the Z-axis z degrees.
rz z	Same as rotz.
zoom val	Scales the 3D zoom factor. If you provide a value of 2.0 for the val argument, the object being viewed will appear twice as large. A value of 0.5 for the val argument will make the object appear only half as large.
zf	Same as zoom.
zoom3	Same as zoom.
vp x0 x1 y0 y1	Sets the viewport, which is the area of the screen that 2D plots take up, for the 2D view. All arguments are floating point numbers that are fractions of the screen so they are in the range [0,1]. The x0 and x1 arguments are the minimum and maximum values for the edges of the viewport in the X dimension. The y0 and y1 arguments are the minimum and maximum values for the edges of the viewport in the Y dimension.
wp x0 x1 y0 y1	Sets the window, which is how much space relative to the plot will be visible inside of the viewport, for the 2D view. All arguments are floating point numbers that are in the same range as the plot extents. The x0 and x1 arguments are the minimum and maximum values for the edges of the window in the X dimension. The y0 and y1 arguments are the minimum and maximum values for the edges of the window in the Y dimension.
reset	Resets the 2D and 3D views
recenter	Recenters the 3D view
undo	Changes back to the previous view

6.2 Advanced view features

The **View Window's Advanced tab**, shown in Figure 9-34, contains features that are not needed by all users.

6.2.1 View centering

The view can either be based on the original spatial extents of the plot or the actual current extents which are the plot's current extents after it has been subsetting in some way. By default, VisIt bases the view on the plot's original extents which leaves the remaining bits of a plot, after being subsetting, in the same space as the original plot. This makes it easy to see where the remaining pieces of the plot were situated relative to the whole plot but it does not always make best use of the visualization window. To fill up more of the visualization window, you might want to base the view on the actual current extents which you can select by choosing the **Actual current extents** option from the **View based on** menu.

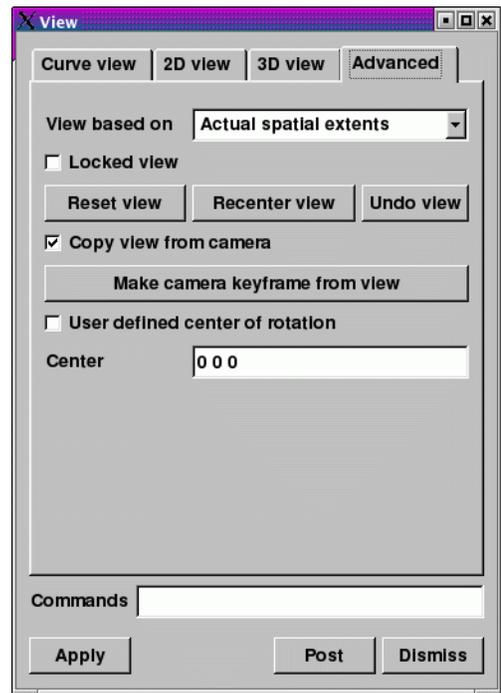


Figure 9-34: Advanced view options

6.2.2 Locking views

When using more than one visualization window, such as when comparing plots using two different databases side by side, it is often useful for the plots being compared to have the same view. VisIt allows you to lock the views together for the two visualization windows so that when you change the rotate, zoom, etc on plots in any window whose view is locked, all other windows with locked views get the new view. To lock the view for a visualization window, click the **Locked view** check box in the **View Window's Advanced tab** or click on the Toolbar button to lock views.

6.2.3 Undo view

If you ever accidentally change the view when you didn't want to change it, you can click on the **Undo view** button on the **View Window's Advanced tab** so set the view back to the previous view. The last 10 views are stored so you can undo up to 10 view changes.

6.2.4 Setting the center of rotation

The center of rotation is the point about which plots are rotated when you set the view. You can type a new center of rotation into the **Center** text field and click the **User defined center of rotation check box** if you want to specify your own center of rotation. The center of rotation is, by default, the center of your plots' bounding box.

When you zoom in to look at smaller plot features and then you rotate the plot, the far away center of rotation causes the changes to the view to be large. Large view changes when you are zoomed in often make the parts of the plot that you were inspecting go out of the view frustum. If you are zoomed in, you should pick a center of rotation that is close to the surface of the plot that you are inspecting. You can also pick a center of rotation using the **Choose center** option in the visualization window's **Popup menu**.

7.0 View and data limits

When visualizing a time-varying database that changes its size, position, or data values over time, it is often difficult to pick view and data limits that work well for all time states. If a plot's extents change over time, VisIt will often recenter the view as appropriate so the plot remains visible. This periodic view recentering, while a good feature for investigative visualization, is not as desirable for generating movies because the view in movies should remain fixed as much as possible unless it is very obvious that a fly-by or some other view device is being used. To ensure that VisIt does not change the view as the plots change size or move through space, VisIt provides the **Maintain view** check box in the **Main Window**, shown in Figure 9-35.

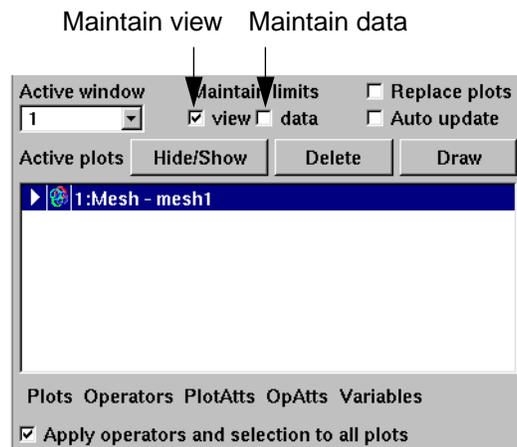


Figure 9-35: Maintain view check box

Data limits that change over time also present a problem when you do investigative work or when you generate a movie. Some plots use the data limits to influence the appearance of the plot. One such plot is the Pseudocolor plot, which maps a variable's data values to a color. If the data range changes over time, the meaning of the colors could change and that is often confusing since it is more intuitive if the meaning attached to each color remains fixed. For example, you might want to adjust the data limits somehow to ensure that the color red always corresponds to the value 50.323. Plots that are affected by the data limits often provide controls for you to specify custom data limits that are used in lieu of the plot's current data limits so the meaning of colors can remain constant throughout the visualization's different time states regardless of how a plot's actual data limits behave. In

addition to the data limit controls provided by plots, VisIt's **Main Window** has a more convenient form for setting the data limits, though you may still prefer to use the plots' data limit controls sometimes. The **Main Window** provides a **Maintain data** check box that forces a plot to use the data limits that it had when the **Maintain data** check box was clicked for all time states in the visualization. If you find that the data limits chosen in this fashion are not suitable for all time states, you can always turn off the **Maintain data** check box and set the limits explicitly in plot attributes windows.

1.0 Overview

This chapter discusses how to use VisIt to create animations. There are three ways of creating animations using VisIt: flipbooks, keyframing, and scripting. For complex animations with perhaps hundreds or thousands of database time steps, it is often best to use scripting. VisIt provides Python and Java language interfaces that allow you to program animation and save image files that get converted into a movie. The flipbook approach is strictly for static animations in which only the database time step changes. This method allows database behavior over time to be quickly inspected without the added complexity of scripting or keyframing. Keyframed animation can exhibit complex behavior of the view, plot attributes, and database time states over time. This chapter emphasizes the flipbook and keyframe approaches and explains how to create animations both ways.

2.0 Animation

Animation is used mainly for looking at how scientific databases evolve over time. Databases usually consist of many discrete time steps that contain the state of a simulation at a specific instant in time. Creating visualizations using just one time step from the database does not reveal time-varying behavior. To be most effective, visualizations must be created for all time steps in the database.

2.1 The `.visit` file

Since scientific databases usually consist of dozens to thousands of time states. Those time states can reside in any number of actual files. Some database file formats support having multiple time states in a single file while other formats require each time state to be

located in its own file. When all time states are in their own file, it is important for VisIt to know which files comprise the database. VisIt attempts to use automatic file grouping to determine which files are in a database but sometimes it is better if you provide the actual list of files in a database when you want to generate an animation using VisIt. You can create a `.visit` file that contains a list of the files in the database. By having a list of files that make up the database, VisIt does not have to guess database membership based on file naming conventions. While this may appear to be inconvenient, it removes the possibility that VisIt will include a file that is not in the database. It also frees VisIt from having to know about dozens of ad-hoc file naming conventions. Having a `.visit` file also allows VisIt to make certain optimizations when generating a visualization. VisIt provides a **Group** button in the **File Selection Window** to assist in the creation of a `.visit` file for databases that have no `.visit` file.

2.2 Flipbook animation

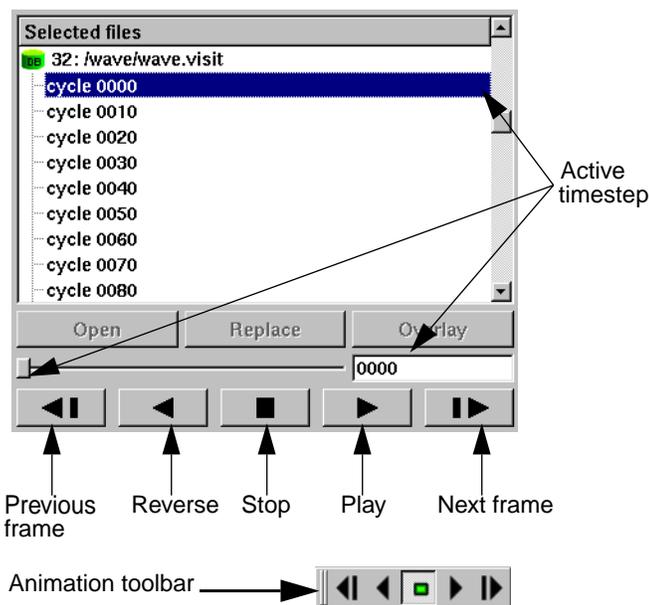


Figure 10-1: Animation controls

All that is needed to create a flipbook animation is a time-varying database. To view a flipbook animation, open a time-varying database, create plots as usual, and click the **Play** button in the **Main Window's File Panel** or in the visualization window's **Animation Toolbar**. A flipbook animation repeatedly cycles through all of the time states in the database displaying the plots for the current time state in the visualization window. The result is an animation that allows you to see the database evolve over time. The **VCR** buttons, shown in Figure 10-1, allow you to control how a flipbook animation plays. The animation controls are also used for controlling

keyframe animations. Clicking the **Play** button causes VisIt to advance the database timestep until the **Stop** button is clicked. As the plots are generated for each database time state, the animation proceeds only as fast as the compute engine can generate plots. You have the option of caching the geometry for each time state so animations will play smoothly according to the animation playback speed once the plots for each database time state have been generated.

2.2.1 Setting the time state

There are several ways that you can set the time state for an animation. You can use the **VCR** controls to play animations or step through them one state at a time. You can also

use the **Time slider** to access a specific animation time state. To set the animation time state using the **Time slider**, click on the time slider and drag horizontally to a new time state. The time state to which you drag it will be displayed in the **Cycle/Time** text field as you drag the time slider so you will know when to let go of the **Time slider**. Once you release the mouse button at a new time state, VisIt will calculate the visualized plots using the data at the specified time state.

If you prefer more precise control over the time state, you can type a cycle or time into the **Cycle/Time** text field to make VisIt jump to the closest cycle or time for the active database. You can also highlight a new time state for the active database in the **Selected files** list and then click the **Replace** button to make VisIt change the time state for the visualization.

2.3 Animation Window

You can open the **Animation Window**, shown in Figure 10-2, by clicking on the **Animation** option in the **Main Window's Controls** menu. The **Animation Window** contains controls that allow you to turn off pipeline caching and adjust the animation playback mode and speed.

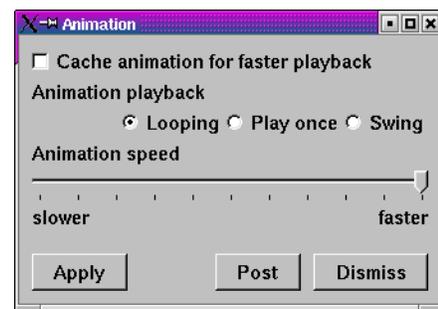


Figure 10-2: Animation Window

2.3.1 Animation playback speed

The animation playback speed is used when playing flipbook or keyframe animations. The playback speed determines how fast VisIt cycles through the database states that make up the animation. Rather than using states per second as a measurement for the playback speed, VisIt uses a simple scale of slower to faster. To set the animation playback speed, use the **Animation speed** slider. Moving the slider to the left and slower setting slows down animations so they change time states once every few seconds. Moving the slider to the right and faster setting will make VisIt play the animation as fast as the host graphics hardware allows.

2.3.2 Pipeline caching

When pipeline caching is enabled, VisIt tries to retain all of the geometric primitives that are used to draw a plot. This greatly speeds up animations once the geometry for all time states is cached. The downside to pipeline caching is that it can consume large amounts of memory. Pipeline caching is enabled by default, but sometimes it makes sense to turn it off. The deciding factors are the size of the database, the number of animation frames, and the number of plots in each animation frame. Try leaving pipeline caching enabled until you notice performance degradation. To turn off pipeline caching, uncheck the **Pipeline caching** check box in the **Animation Window**.

2.3.3 Animation playback mode

The animation playback mode determines how VisIt gets to the next time state after playing until the end of the animation. There are three animation playback modes: looping, play once, and swing. VisIt loops animations by default so once the end of the animation is reached, it starts playing from the beginning. When the animation mode is set to play once, VisIt plays the animation through until the end and then stops playing the animation. When VisIt reaches the end of the animation in swing mode, the animation starts playing in reverse until it gets to the start, at which point, it starts playing forward again. To set the animation mode, click on one of the **Looping**, **Play once**, and **Swing** radio buttons in the **Animation Window**.

3.0 Keyframing

Keyframing is an advanced form of animation that allows you create animations where certain animation attributes such as view or plot attributes can change as the animation progresses. You can design an entire complex animation upfront by specifying a number of animation frames to be created and then you can tell VisIt which plots exist over the animation frames and how their time states map to the frames. You can also specify the plot attributes so they remain fixed over time or you can make individual plot and operator attributes evolve over time. With keyframing, you can make a plot fade out as the animation progresses, you can make a slice plane move, you can make the view slowly change, etc. Keyframe animations allow for quite complex animation behavior.

3.1 Keyframing Window

Keyframe animations are designed using VisIt's **Keyframing Window** (see Figure 10-3), which you can open by selecting the **Keyframing** option from the **Main Window's Controls menu**. The window is dominated by the **Keyframe area**, which consists of many vertical lines that correspond to each frame in the animation and horizontal lines, or **Keyframe lines**, that correspond to the state attributes that are being keyframed. The horizontal lines are the most important because they allow you to move and delete keyframes and set the plot range, which is the set of animation frames over which the plot is defined.

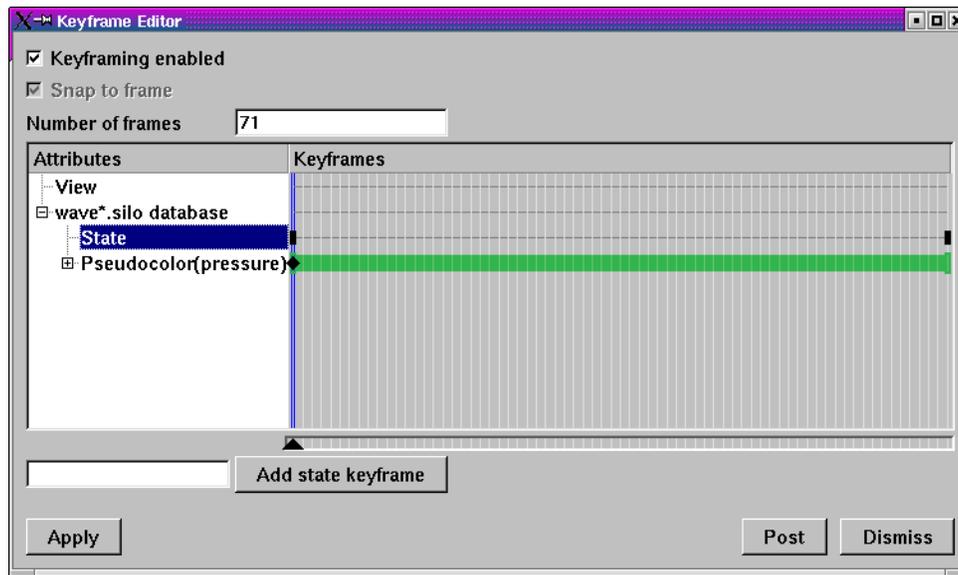


Figure 10-3: Keyframing Window

3.1.1 Keyframing mode

To create a keyframe animation, you must first open the **Keyframing Window** and check the **Keyframing enabled** check box. When VisIt is in keyframing mode, a keyframe is created for the active animation state each time you set plot or operator attributes and time is set using the *Animation* time slider. The *Animation* time slider is a special time slider that is made active when you enter keyframing mode and the animation frame can only be set using it. Changing time using any other time slider results in a new database state keyframe instead of changing the animation frame.

If you have created plots before entering keyframing mode, VisIt converts them into plots that can be keyframed when you enter keyframing mode. When you leave keyframing mode, extra keyframing attributes associated with plots are deleted, the animation containing the plots reverts to a flipbook animation, and the *Animation* time slider is no longer accessible.

3.1.2 Setting the number of frames

When you go into keyframing mode for the first time, having never set a number of keyframes, VisIt will use the number of states in the active database for the number of frames in the new keyframe animation. The number of frames in the keyframe animation will vary with the length of the database with the most time states unless you manually specify a number of animation frames, which you can do by entering a new number of frames into the **Keyframing Window's Number of frames** text field. Once you enter a number of frames, the number of frames will not change unless you change it.

3.1.3 Adding a keyframe

To add a keyframe, you must first have created some plots and put VisIt into keyframing mode by clicking the **Keyframing enabled** check box in the **Keyframing Window**. After you have plots and VisIt is in keyframing mode, you can add a keyframe by opening a plot's attribute window, changing settings, and clicking its **Apply** button. To set a keyframe for a later frame in the animation, move the **Keyframe time** slider, which is located under the **Keyframe area** (see Figure 10-4), to a later time and change the plot attributes again. Each time you add a keyframe to the animation, a small black diamond, called a **Keyframe indicator**, will appear along the **Keyframe line** for the plot. When you play through the animation using any of VisIt's animation controls, the plot attributes are calculated for each animation frame and they are used to influence how the plots look when they appear in the vis window.

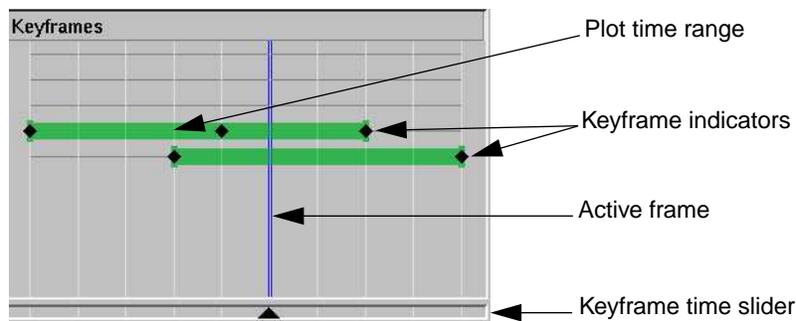


Figure 10-4: Keyframe area

3.1.4 Adding a database state keyframe

Each plot that exists at a particular animation frame must use a specific database state so the correct data will be plotted. When VisIt is in keyframing mode, the database state can also be keyframed so you can control the database state used for a plot at any given animation frame. The ability to set an arbitrary database state keyframe for a plot allows you to control the flow of time in novel ways. You can, for example, slow down time, stop time, or even make time flow backwards for a little while.

There are two ways to set database state keyframes in VisIt. The first way is to move the **Keyframe time** slider to the desired animation frame, enter a new number into the text field next to the **Keyframe Window's Add state keyframe** button, and then click the **Add state keyframe** button. As an alternative, you can use the **Main Window's Time slider** to create a database state keyframe, provided the active time slider is not the *Animation* time slider. To set a database state keyframe using the **Time slider**, select a new database time slider from the Active time slider combo box and then change time states using the **Time slider**. Instead of changing the active state for the plots that use the specified database, VisIt uses the information to create a new database state keyframe for the active animation frame.

3.1.5 Adding a view keyframe

In addition to being able to add keyframes for plot attributes, operator attributes, and database states, you can also set view keyframes so you can create sophisticated fly-bys of your data. To create a view keyframe, you must interactively change the view in the vis window using the mouse or specify an exact view in the **View Window**. Once the view is where you want it for the active animation frame, open the **View Window** and click the **Make camera keyframe from view** button on the **Advanced** tab in order to make a view keyframe. Once the view keyframe has been added, a keyframe indicator will be drawn in the **Keyframing Window**.

VisIt will not use view keyframes by default when you are in keyframing mode because it can be disruptive for VisIt to set the view while you are still adding view keyframes. Once you are satisfied with your view keyframes, click the **Copy view from camera** button on the **Advanced** tab in the **View Window** in order to allow VisIt to set the view using the view keyframes when you change animation frames.

3.1.6 Deleting a keyframe

To delete a keyframe, move the mouse over a **Keyframe indicator** and right click on it with the mouse once the indicator becomes highlighted.

3.1.7 Moving a keyframe

To move a keyframe, move the mouse over a **Keyframe indicator**, click the left mouse button and drag the **Keyframe indicator** left or right to a different animation frame. If at any point you drag the **Keyframe indicator** outside of the green area, which is the plot time range, and release the mouse button, moving the keyframe is cancelled and the **Keyframe indicator** returns to its former animation frame.

3.1.8 Changing the plot time range

The plot time range determines when a plot appears or disappears in a keyframed animation. Since VisIt allows plots to exist over a subset of the animation frames, you can set a plot's plot range in the **Keyframe area** to make a plot appear later in an animation or be removed before the animation reaches the last frame. You may find it useful to set the plot range if you've increased the number of animation frames but found that the plot range did not expand to fill the new frames. To change the plot time range, you left-click on the beginning or ending edges of the **Plot time range** (the green band on the **Keyframe line**) in the **Keyframe area** and drag it to a new animation frame.

4.0 Session files

A session file is an XML file that contains all of the necessary information to recreate the plots and visualization windows used in a VisIt session. You can set up complex

visualizations, save a session file, and then run a new VisIt session later and be able to pick up exactly where you left off when you saved the session file. If you often look at the same types of plots with the same complex setup then you should save a session file for your visualization once you've set it up so you don't have to do any manual setup in the future.

4.1 Saving session

Once you have set up your plots, you can select **Save session** option in the **Main Window's File** menu to open up a **Save file** dialog. Once the **Save file** dialog is opened, select the location and filename that you want to use to store the session file. By default, VisIt stores all session files in your `.visit` directory on UNIX and MacOS X computers and in the directory where VisIt was installed on Windows computers. Once you select the location and filename to use when saving the session file, VisIt writes an XML description of the complete state of all vis windows, plots, and GUI windows into the session file so the next time you come into VisIt, you can completely restore your VisIt session.

4.2 Restoring session

Restoring a VisIt session file deletes all plots, closes all databases, etc before VisIt reads the session file to get back to the state described in the session file. After restoring a session file, VisIt will look exactly like it did when the session file was saved. To restore a session file, click the **Restore session** option from the **Main Window's File** menu to open an **Open file** dialog. Choose a session file to open using the **Open file** dialog. Once you've chosen a file, VisIt restores the session using the selected session file. If you are on the Windows platform, you can double-click session files (*.vses files*) stored on your computer in order to directly open them with VisIt.

5.0 Scripting

Scripting is an alternate method of producing animations that can have the simplicity of flipbook animations with the flexibility of keyframing. Scripting animations is more difficult than other methods because you have to script each event by writing a Python or Java program to control VisIt's viewer. One clear strength of this method is that it is very reproducible and can be used to generate animation frames in a batch computing environment. For in-depth information about writing Python scripts for VisIt, consult the *Visit Python Interface* document.

5.1 Command Window

It is possible for VisIt's GUI and Python Interface to share the same viewer component at runtime. When you invoke `visit` at the command line, VisIt's GUI is launched. When you invoke `visit -cli` at the command line, VisIt's CLI (Python interface) is launched. If you

want to use both components simultaneously then you can use VisIt's **Command Window**. The **Command Window** can be opened by clicking on the **Command** menu option in the **Main Window's Controls** menu. The **Command Window** consists of a set of eight tabs in which you can type Python scripts. When you type a Python script into one of the tabs, you can then click the tab's **Execute** button to make VisIt try and interpret your Python code. If VisIt detects that it has no Python interpreting service available, it will launch the CLI (connected to the same viewer component) and then tell the CLI to execute your Python code. Note that the **Command Window** is just for editing Python scripts. Any output that results from the Python code's execution will be displayed in the CLI program window (see Figure 10-5).

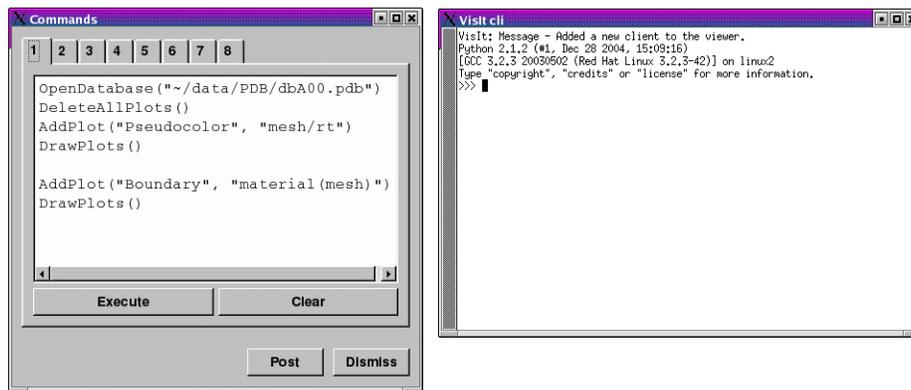


Figure 10-5: Command Window and CLI program window

5.1.1 Saving the Command Window's Python scripts

The **Command Window** is meant to be a sandbox for experimenting with small Python scripts that help you visualize your data. You will often hit upon small scripts that can be used over and over. The scripts in each of the eight tabs in the **Command Window** can be saved for future VisIt sessions if you save your settings. Once you save your settings, any Python scripts that are present in the **Command Window** are preserved for future use.

5.1.2 Clearing a Python script from a tab

If a Python script in one of the **Command Window's** tabs is no longer useful then you can click that tab's **Clear** button to clear out the contents of the tab so you can begin creating a new script in that tab. If you want VisIt to permanently delete the script from the tab then you must save your settings after clicking the **Clear** button.

5.1.3 Using the GUI and CLI to design a script

Writing a Python script that performs visualization from scratch can be difficult. The process of setting up a complex visualization can be simplified by using both the GUI and the CLI at the same time. For example, you can use VisIt's GUI to set up the plots that you initially want to visualize and then you can save out a session file that captures that setup.

Next, you can open a text editor and create a new Python script. The first line of your Python script can use VisIt's *RestoreSession* command to restore the session file that you set up with the GUI from within the Python scripting environment. For more information on functions and objects available in VisIt's Python interface, see the *VisIt Python Interface manual*. After using the *RestoreSession* function to set VisIt situated with all of the right plots, you can proceed with more advanced Python scripting to alter the view or move slice planes, etc. Once you've completed your Python script in a text editor, you can paste it into the **Command Window** to test it or you can pass it along to VisIt's command line movie tools to make a movie.

5.1.4 Quitting VisIt when there are multiple user interfaces

When you use the **Command Window**, VisIt launches the VisIt Python interface (CLI) to interpret the Python scripts that you enter into the **Command Window**. This means that you have two user interfaces attached to the same viewer component. When more than one user interface is attached to the viewer, VisIt asks you if you want to quit all user interfaces when you click the **Quit** option in the **Main Window's File menu** (see Figure 10-6). If you choose to quit all user interfaces then VisIt totally quits. If you choose instead to just quit the GUI then the CLI program will remain active and just the GUI will terminate. This allows you to selectively close user interfaces that you no longer need. When all of the user interface programs (GUI, CLI, Java, etc.) attached to the viewer have closed, the viewer itself closes down.

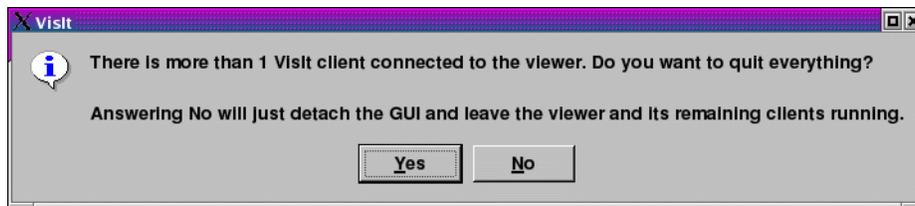


Figure 10-6: Quit prompt when using multiple user interfaces

6.0 Movie tools

VisIt provides a command line utility based on VisIt's Command Line Interface that is called *visit -movie*. The *visit -movie* movie generation utility is installed with all versions of VisIt and can be used to generate movies using session files or Python scripts as input. If you want to design movies based on visualizations that you have created while using VisIt's GUI then you might also want to read about the **Save movie wizard** on page 133. If the *visit* command is in your path then typing: *visit -movie* at the command

line prompt, regardless of the platform that you are using, will launch the *visit -movie* utility. The following table lists *visit -movie*'s command line arguments.

Argument	Description
-format fmt	The format option allows you to set the output format for your movie. The supported values for <i>fmt</i> are: <i>mpeg</i> : MPEG 2 movie. <i>qt</i> : Quicktime movie. <i>sm</i> : Streaming movie format. <i>png</i> : Save raw movie frames as individual PNG files. <i>ppm</i> : Save raw movie frames as individual PPM files. <i>tiff</i> : Save raw movie frames as individual TIFF files. <i>jpeg</i> : Save raw movie frames as individual JPEG files. <i>bmp</i> : Save raw movie frames as individual BMP (Windows Bitmap) files. <i>rgb</i> : Save raw movie frames as individual RGB (SGI format) files.
-geometry size	The geometry option allows you to set the movie resolution. The size argument is of the form <i>WxH</i> where <i>W</i> is the width of the image and <i>H</i> is the height of the image. For example, if you want an image that is 1024 pixels wide and 768 pixels tall, you would provide: <i>-geometry 1024x768</i> .
-sessionfile name	The sessionfile option lets you pick the name of the VisIt session to use as input for your movie. The VisIt session is a file that describes the movie that you want to make and it is created when you save your session from within VisIt's GUI after you set up your plots how you want them.
-scriptfile name	The scriptfile option lets you pick the name of a VisIt Python script to use as input for your movie.
-framestep step	The number of frames to advance when going to the next frame.
-start frame	The frame at which to start.
-end frame	The frame at which to end.
-output	The output option lets you set the name of your movie.

Argument	Description
-fps number	Sets the frames per second at which the movie should be played.

The *visit -movie* utility always supports creation of series of image files but it does not always support creation of movie formats such as MPEG, QuickTime, or Streaming movie. Support for movie formats varies based on the platform. QuickTime and Streaming movie formats are currently limited to computers running IRIX and the appropriate movie conversion tools (*makemovie*, *img2sm*) must be in your path or VisIt will create a series of image files instead of a single movie file. All other platforms except Windows support creation of MPEG movies using software provided in the VisIt distribution. If you want to create movies on Windows, use *visit -movie* to generate the individual movie frames and then use your favorite movie generation software to convert the frames into a single movie file.

While *visit -movie* on Windows does not provide an encoding program to create a single movie file, it does have some usability advantages over other platforms. If you browse the Windows file system and come across a VisIt session file, which on Windows ends with a *.vses* extension, you can right click on the file and choose from several movie generation options. The movie generation options make one-click movie generation possible so you don't have to master the arguments for *visit -movie* like you do on other platforms. After selecting a movie generation option for a VisIt session file, Windows runs *visit -movie* implicitly with the right arguments and saves out the movie frames to the same directory that contains the session file. The movie generations option in a session file's context menu are shown in Figure 10-7

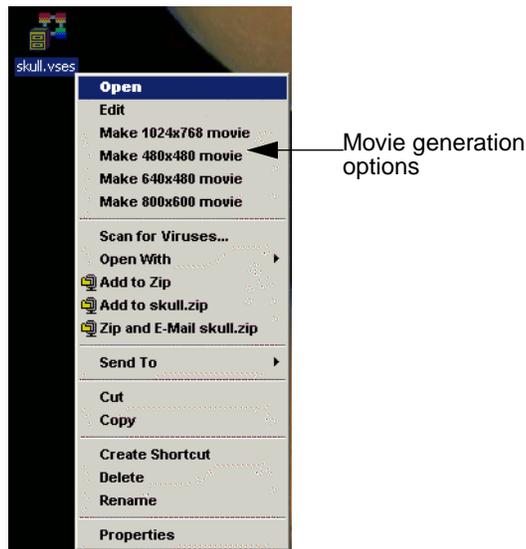


Figure 10-7: Movie generation options for session files on Windows platform

1.0 Overview

In this chapter, we learn about VisIt's interactive tools and how to use them.

2.0 Introduction to interactive tools

An interactive tool is an object that can be added to a visualization window to set attributes for certain plots and operators such as the Streamline plot or Slice operator. You can turn interactive tools on and off by clicking on the tool icons in a visualization window's **Toolbar** or **Popup menu** (see Figure 11-1). Note that some tools prefer to operate in visualization windows that contain plots of a certain dimension so some tools are not always available.

Once you enable a tool, its appears in the visualization window. Tools have one or more small red rectangles called *hot points* that cause the tool to perform an action when you click or drag the hot point with the mouse. When you use the mouse to manipulate a tool's hot point, all mouse events are delivered to the tool so it can respond to the mouse interaction. When the mouse is outside of a hot point, the mouse responds as it would if there were no tools activated so you can still rotate and zoom-in on plots while still having tools enabled.

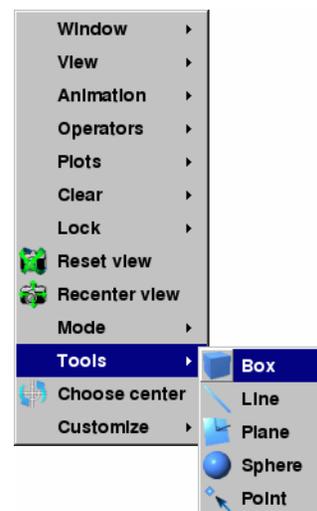


Figure 11-1: Tools menu

3.0 Box Tool

The box tool, which is shown in Figure 11-2, allows you to move an axis-aligned box around in 3D space. You can use the box tool with the Box and Clip operators or the Streamline plot to interactively restrict plots to a certain volume. The box tool is drawn as a box with five hotspots that allow you to move the box in 3D space or resize it in any or all dimensions.

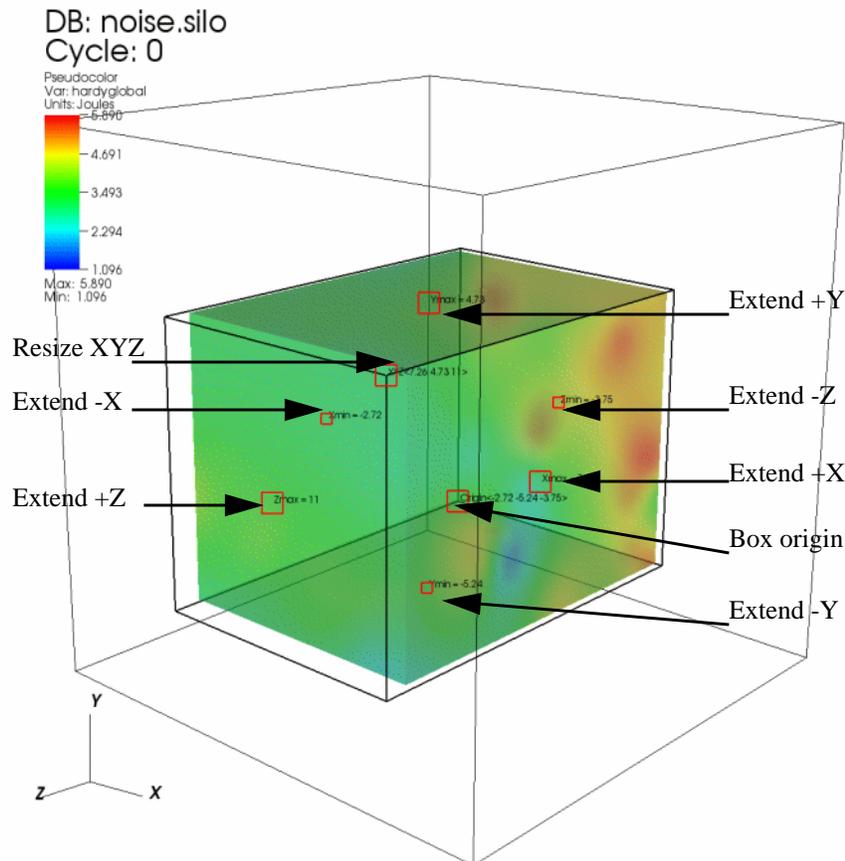


Figure 11-2: Box tool with a plot restricted to the box.

You can move the box tool around the vis window by clicking on the origin hotspot, which has the word “Origin” next to it, and dragging it around the vis window. When you move the box tool, it moves in a plane that is parallel to the screen. You can move the box tool backward and forward along an axis by holding down the keyboard’s *Shift* key before you click and drag the origin hotspot. When the box tool moves, red, green, and blue boxes appear to give a point of reference for the box with respect to the X, Y, and Z dimensions (see Figure 11-3).

You can extend one of the box’s faces at a time by clicking on the appropriate hotspot and moving the mouse up to extend the box or by moving the mouse down to shrink the box in the given dimension. Hotspots for the box’s back faces are drawn smaller than their front-facing counterparts. When the box is resized in a single dimension, reference planes are

drawn in the dimension that is changing so you can see where the edges of the box are in relation to the bounding box for the visible plots. You can also resize all of the dimensions at the same time by clicking on the “Resize XYZ” hotpoint and dragging the mouse in an upward motion to scale the box to a larger size in X,Y, and Z or by dragging the mouse down to shrink the box. When all box dimensions are resized at the same time, the shape of the box remains the same but the scale of the box changes.

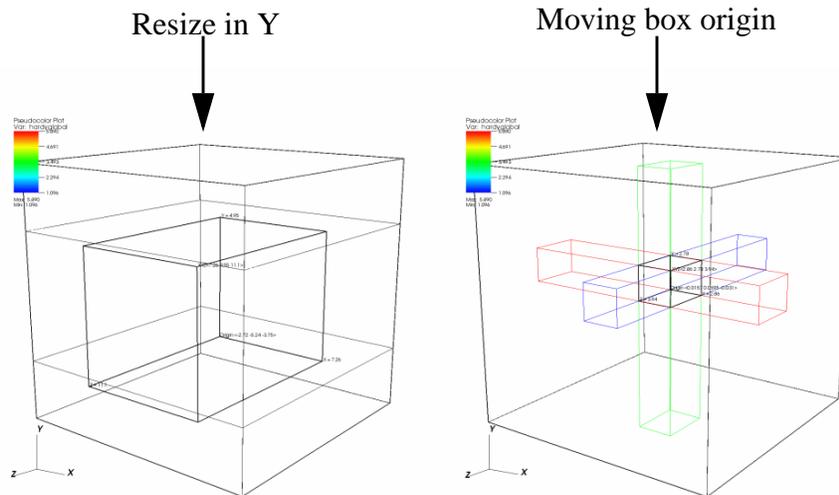
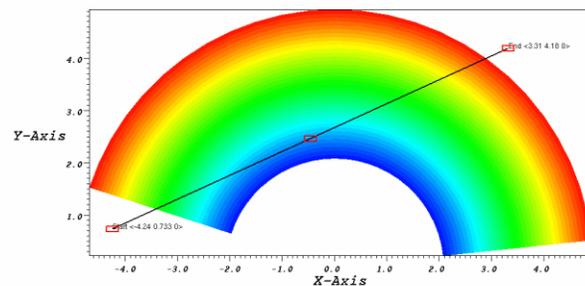
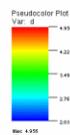


Figure 11-3: Box tool appearance while it is resized or moved

4.0 Line Tool

It is common to create Curve plots when analysing a simulation database. Curve plots are created using VisIt's lineout mechanism where reference lines are drawn in a visualization window and Curve plots are created in another visualization window using the path described by the reference lines. VisIt's line tool allows reference lines to be moved after they are initially drawn. The line tool allows the user to see a representation of a line in a visualization window and position the line relative to plots that exist in the window.

DB: curv2d.pdb
Cycle: 48 Time:4.8



user: whitlocb
Fri Jun 21 10:09:11 2002

Figure 11-4: Line tool with a 2D plot

The line tool is drawn as a thick line with three hot points positioned along the length of the line. Both of the line tool's endpoints, as well as its center, have a hotpoint. Since the line tool can be used for both 2D and 3D databases, the line tool's behavior is slightly different for 2D than it is for 3D. Clicking and dragging on either endpoint will move the selected endpoint causing the line to change shape. Another way of moving an endpoint is to hold down the *Ctrl* key and then click on the point and move the mouse up and down to extend or shorten the line. Clicking and dragging the middle hot point moves the entire line tool.

In 2D, the line endpoints can only be moved in the X-Y plane (Figure 11-4). In 3D, the line endpoints can be moved in any dimension. Since it is more difficult to see how the line is oriented relative to plots in 3D, when the line tool is moved, 3D crosshairs appear. The crosshairs intersect the bounding box and show the position of the line endpoint relative to the plots. Clicking and dragging endpoints will move them in a plane that is perpendicular to the screen. Moving the endpoints, while first pressing and holding down the *Shift* key, causes the selected endpoint to move back and forth in the dimension that most faces the screen. This allows endpoints to be moved in one dimension at a time. An example of the line tool in 3D is shown in Figure 11-5.

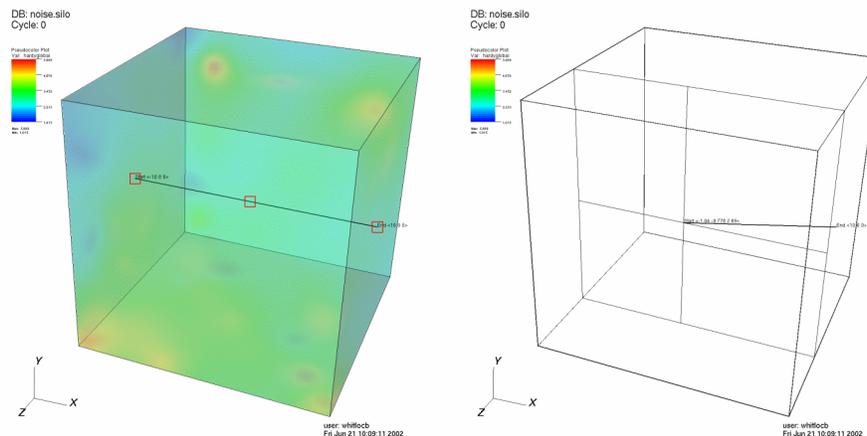


Figure 11-5: Line tool in 3D

The line tool can be used to set the attributes for certain VisIt operators such as VisIt's Lineout operator. If a plot has a Lineout operator applied to it, the line tool is initialized with that operator's endpoints when it is first enabled. As the line tool is repositioned and reoriented, the line tool's line endpoints are given to the Lineout operator and Curve plots that are fed by the Lineout operator are recalculated.

5.0 Plane Tool

The plane tool allows the user to see a representation of a slice plane in a visualization window and position the plane relative to plots that may exist in the window. The plane

tool, shown in Figure 11-6, is represented as a set of 3D axes, a bounding rectangle, and text which gives the plane equation in origin-normal form. The plane tool provides several hot points positioned along the 3D axes that are used to position and orient the tool. The hot point nearest the origin allows the user to move the plane tool in a plane parallel to the computer screen. The hot point that lies in the middle of the plane's Z-axis translates the plane tool along its normal vector when the hotpoint is dragged up and down. The hot point on the end of the Z-axis causes the plane tool to rotate freely when the hot point is moved. When the plane tool is facing into the screen, the Z-axis vector turns red to indicate which direction the plane tool is pointing. The other hot points also rotate the plane tool but they restrict the rotation to a single axis of rotation.

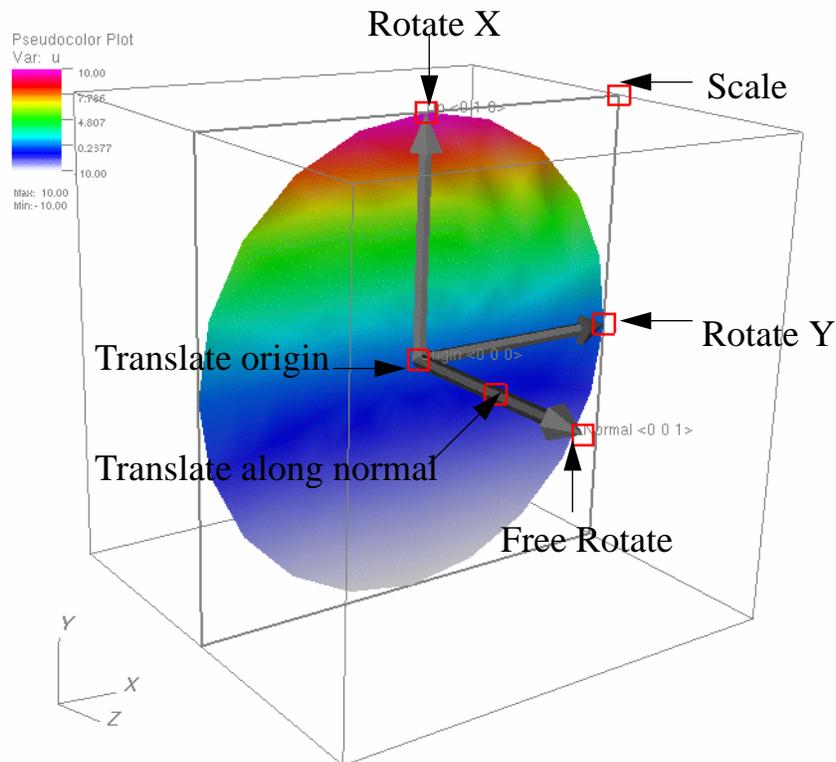


Figure 11-6: Plane tool with sliced plot

You can use the plane tool to set the attributes for certain VisIt plots and operators. The Slice operator, for example, can update its plane equation from the plane tool's plane equation. If a plot has a Slice operator applied to it, the plane tool is initialized with that operator's slice plane when it is first enabled. As the plane tool is repositioned and reoriented, the plane tool's plane equation is given to the operator and the sliced plot is recalculated. You can also use the plane tool with the Streamline plot to modify the source plane that it uses for streamline seed points.

6.0 Point Tool

The point tool allows you to position a single point relative to plots that exist in the visualization window. The point tool, shown in Figure 11-7, provides one hot point at the tool's origin. Clicking on the hot point and moving the mouse moves the point tool's origin in a plane perpendicular to the screen. Holding down the *Shift* key before clicking on the hot point moves the point tool's origin along the plot axis that most faces the user. Holding down the *Ctrl* key moves point tool along the plot axis that points up. The point tool is commonly used to set the origin for the ThreeSlice operator or set the location of a point streamline source for the Streamline plot.

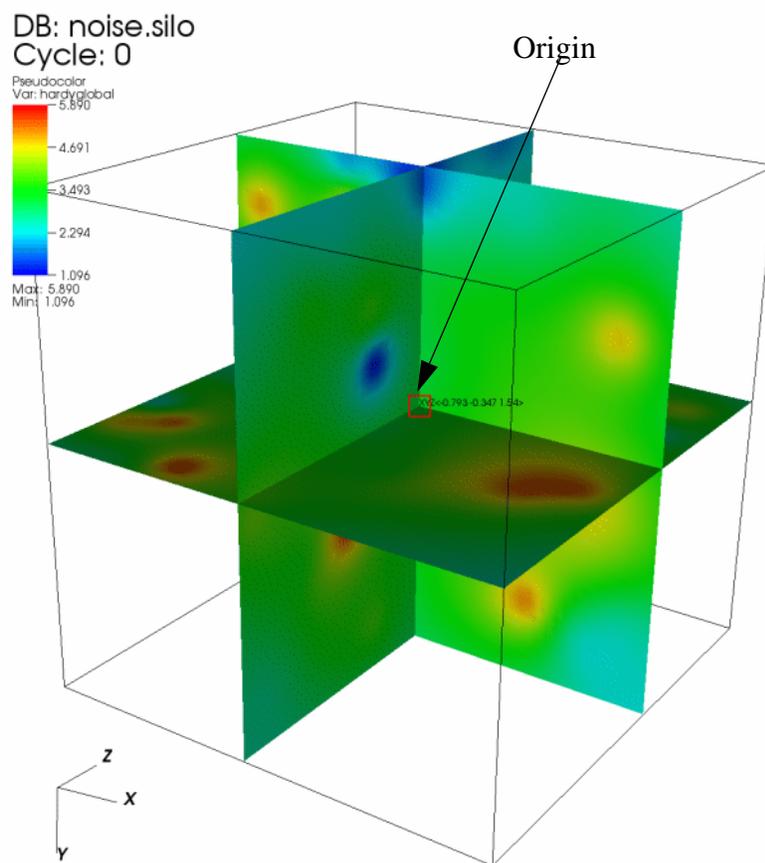


Figure 11-7: Point tool

7.0 Sphere Tool

The sphere tool allows you to position a sphere relative to plots that exist in the visualization window. The sphere tool, shown in Figure 11-8, provides several hot points that are used to position and scale the sphere. The hot point nearest the center of the sphere is the origin hot point and it is used to translate the sphere in a plane parallel to the screen.

The other hot points are all used to scale the sphere. To scale the sphere, click on one of the scaling hot points and move the mouse towards the origin hot point to shrink the sphere or move the hot point away from the origin to enlarge the sphere.

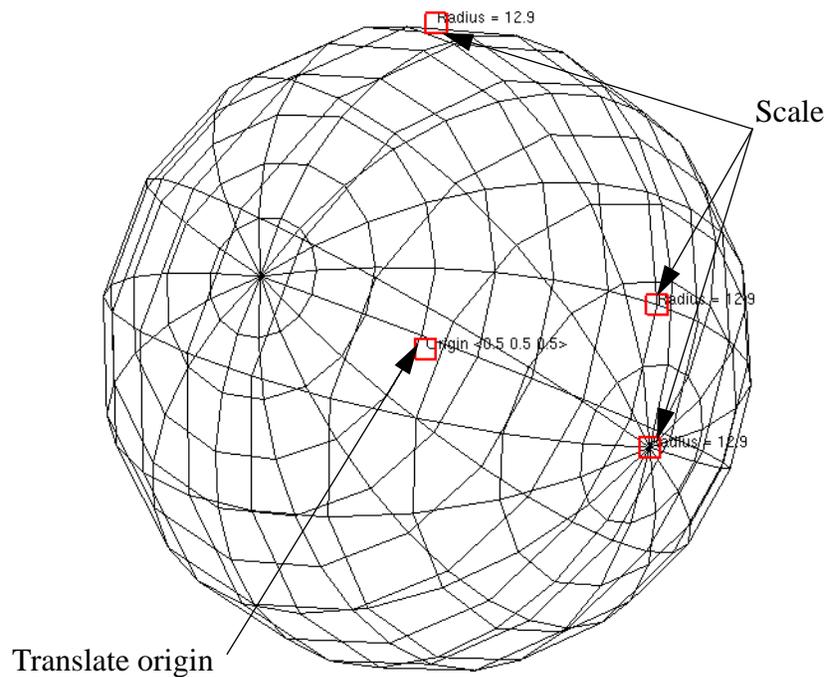


Figure 11-8: Sphere tool

You can use the sphere tool to set the attributes for certain VisIt plots and operators. The sphere tool is commonly used to set the attributes for the Sphere slice operator. After applying a Sphere slice operator to a plot, enable the Sphere tool to interactively position the sphere that slices the plot. You can also use the sphere tool to set the seed points for the Streamline plot.

Chapter 12

Multiple Databases and Windows

1.0 Overview

In this chapter, we discuss how to use VisIt to visualize multiple databases using either a single window or multiple visualization windows that have been locked together. The first topic covered in this chapter is database correlation, which is used relate multiple time-varying databases together in some fashion. The use of database correlations will be explained in detail followed by a description of common useful operations involving multiple visualization windows.

2.0 Databases

One main use of a visualization tool such as VisIt is to compare multiple related simulation databases. Simulations are often run over and over with a variety of different settings or physics models and this results in several versions of a simulation database that all describe essentially the same object or phenomenon. Simulations are also often run using different simulation codes and it is important for a visualization tool to compare the results from both simulations for validation purposes. You can use VisIt to open any number of databases at the same time so you can create plots from different simulation databases in the same window or in separate visualization windows that have been locked together.

2.1 Activating a database

VisIt can have any number of databases open simultaneously but there is still an active database that is used to create new plots. Each time you open a database, the newly opened database becomes the active database and consequently the source database for any new plots that you decide to create. If you want to create a plot using a database that is open but

is not your active database, you must activate that database so it becomes your active database. When you activate a database, its variables are added to the menus for the various plot types.

There are two ways to activate a database and the method that you choose depends on how you have configured the **File panel** to display files. By default, the **File panel** shows the **Selected files** list and the **Open**, **ReOpen**, **Activate**, **Replace**, and **Overlay** buttons. However, you can configure the **File panel** so it does not show the **Selected files** list. You can read more about configuring the **File panel** in the **Preferences** chapter of this book. If the **Selected files** list is shown in the **File panel**, you can click on a database in the **Selected files** list that you've previously opened and click the **Activate** button to set the active database to the highlighted file. The **Activate** button, shown in Figure 12-1, is located in the same place as the **Open** or **ReOpen** buttons but it is only displayed if you have clicked on a file that you have previously opened that is not already the active database.

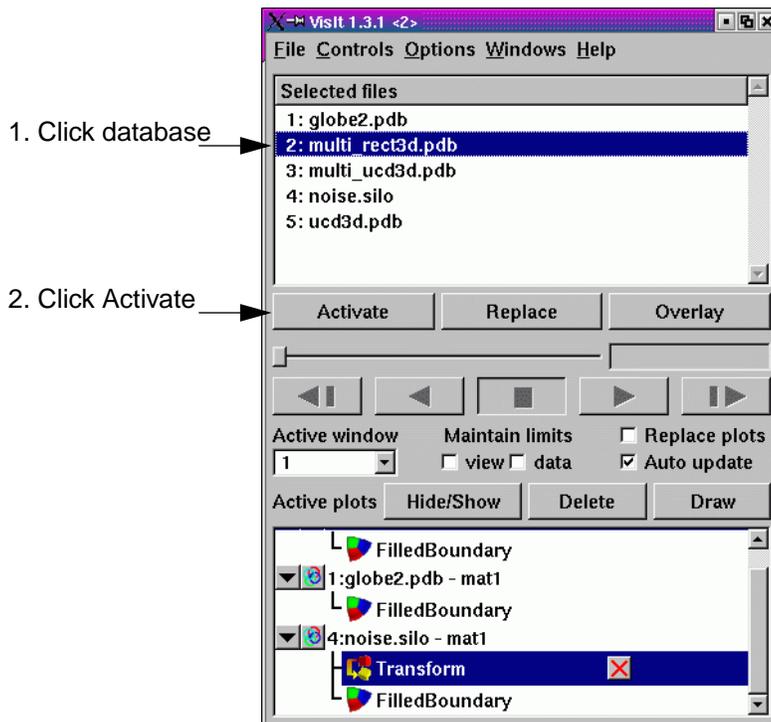


Figure 12-1: Activating a database using the Selected files list

If you have configured VisIt's **File panel** so the **Selected files** list is not shown then you have to use the second method of database activation in order to activate a database since the controls used in the first method are not available. The second method for

activating a database is to select a database from the **Source** combo box, which is located just above the plot list and contains the list of opened databases (see Figure 12-2).

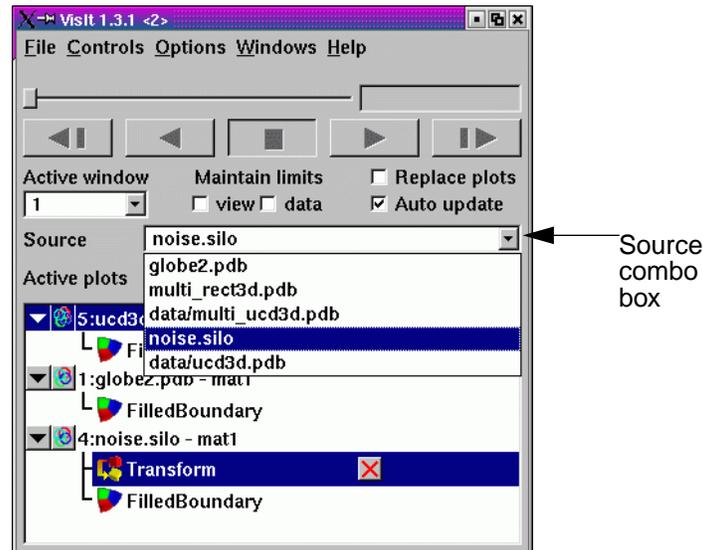


Figure 12-2: Activating a database using the Source combo box

2.2 Multiple time sliders

When your open databases all have only a single time state, the Time slider in the Main Window is disabled. When you have one database that has multiple time states, the Time slider is enabled and can be used exclusively to change time states for the database that has multiple time states; the database does not even have to be the active database. Things get a little more complicated when you have opened more than one time-varying database - especially if you have plots from more than one of them.

When you open a database in VisIt, it becomes the active database. If the database that you open has multiple time states, VisIt creates a new logical time slider for it so you can end up having a separate time slider for every open database with multiple time states. When VisIt has to create a time slider for a newly opened database, it also makes the new database's (also the active database) be the active time slider. There is only one **Time slider** control in the **Main Window** so when there are multiple logical time sliders, VisIt displays an **Active time slider** combo box (see Figure 12-3) that lets you choose which logical time slider to affect when you change time using the **Time slider**.

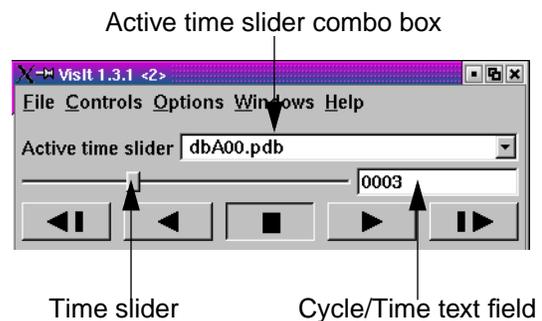


Figure 12-3: Time slider and related controls

Since VisIt allows each time-varying database to have its own logical time slider, you can create plots from more than one time-varying database in a single visualization window and change time independently for each database, an impossible operation in MeshTV and early versions of VisIt. Another benefit of having multiple logical time sliders is that the databases plotted in the visualization windows are free to have different numbers of time states. Suppose you have opened time-varying databases A and B and created plots from both databases in the same visualization window. Assuming you opened database A and then database B, database B will be the active database. If you want to change time states for database A but not for database B, you can select database A from the **Active time slider** combo box and then change the time state using the **Time slider**. If you then wanted to change time states for database B, you could select it in the **Active time slider** combo box and then change the time state using the **Time slider**. If you wanted to change time states for both A and B at the same time, you have to use database correlations, which are covered next.

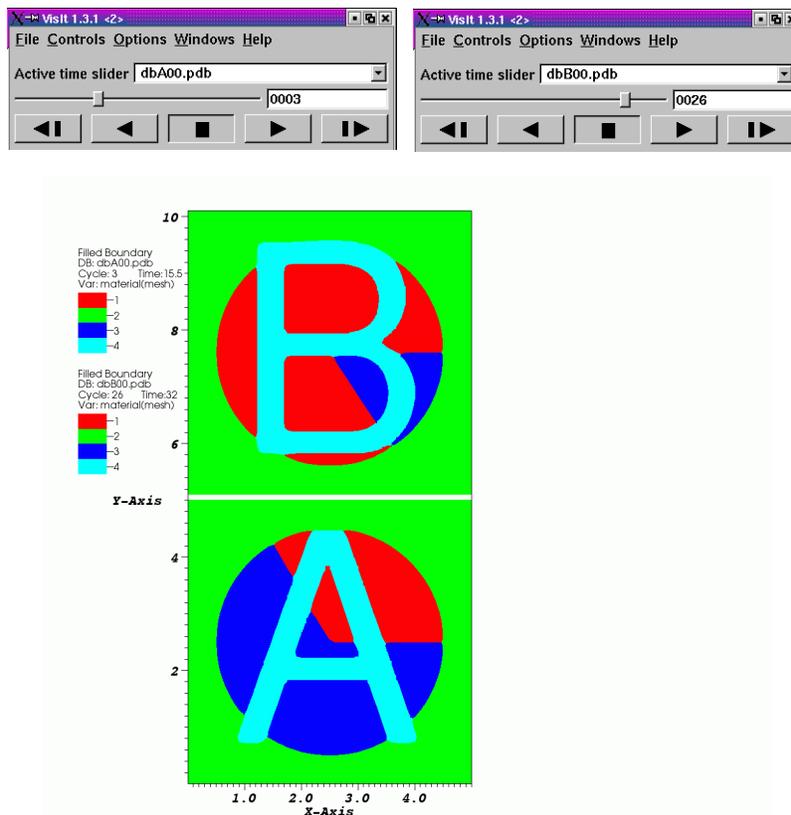


Figure 12-4: Active time slider and time slider controls

3.0 Database correlations

A database correlation is a map that relates one or more different time-varying databases so that when accessed with a common time state, the database correlation can tell VisIt which time state to use for any of the databases in the database correlation. Since VisIt supports multiple logical time sliders, so time states can be changed independently for different time-varying databases in the same window, no time slider for any database can have any effect on another database. Sometimes when comparing two different, but related, time-varying databases, it is useful to make plots of both databases and see how they behave over time. Since changing time for each database independently would be tedious, VisIt provides database correlations to simplify visualizing multiple time-varying databases.

3.1 Database correlations and time sliders

When you open a database for the first time, VisIt creates a trivial database correlation for that single database and creates a new logical time slider for it. Each database correlation has its own logical time slider. Figure 12-5 shows a database correlation as the active time slider.



Figure 12-5: Database correlation as the active time slider

Suppose you have plots from time-varying database A and database B in the same visualization window. You can use the logical time slider for database A to change database A's time state and you can use the logical time slider for database B to change database B's time state. If you want to change the time state for both databases at the same time using a single logical time slider, you can create a database correlation involving database A and database B and then change time states using the database correlation's logical time slider. When you change time states using a database correlation's time slider, the time state used in each plot is calculated by using the database correlation's time slider's time state to look up the plot's expected time state in the database correlation. Thus changing time states using a database correlation also updates the logical time slider for each database involved in the database correlation.

3.2 Types of database correlations

A database correlation is a map that relates one or more databases. When there is more than one database involved in a database correlation, the time states from each database are related using a correlation method. Database correlations currently have 4 supported correlation methods: padded index, stretched index, time, and cycle. This section describes each of the correlation methods and when you might want to use each method.

For illustration purposes, the examples describing each correlation method use two databases, though database correlations can have any number of databases. The examples

refer to the databases as: database A and database B. Both databases consist of a rectilinear grid with a material variable. The material variable is used to identify the database using a large letter A or B and also to visually indicate progress through the databases' numbers of time states by sweeping out a red material like a clock in reverse. At the first time state, there is no red material but as time progresses, the read material increases and finally totally replaces the material that was blue. Database A has 10 time states and database B has 20 time states. The tables below list the cycles and times for each time state in each database so the time and cycle behavior of database A and database B will make more sense later when time database correlations and cycle database correlations are covered.

Database A										
Time state	0	1	2	3	4	5	6	7	8	9
Times	14	14.5	15	15.5	16	16.5	17	17.5	18	18.5
Cycles	0	1	2	3	4	5	6	7	8	9

Database B (part 1)										
Time state	0	1	2	3	4	5	6	7	8	9
Times	16	17	18	19	20	21	22	23	24	25
Cycles	10	11	12	13	14	15	16	17	18	19

Database B (part 2)										
Time state	10	11	12	13	14	15	16	17	18	19
Times	26	27	28	29	30	31	32	33	34	35
Cycles	20	21	22	23	24	25	26	27	28	29

3.2.1 Padded index database correlation

A padded index database correlation, like any other database correlation, involves multiple input databases where each database potentially has a different number of time states. A padded index database correlation has as many time states as the input database with the largest number of time states. All other input databases that have fewer time states than the longest database have their last time state repeated until they have the same number of time states as the input database with the largest number of time states. Using the example databases A and B, since B has 20 time states and A only has 10 time states, database A

will have its last time state repeated 10 times to make up the difference in time states between A and B. Note how database A's last time state is repeated in Figure 12-6.

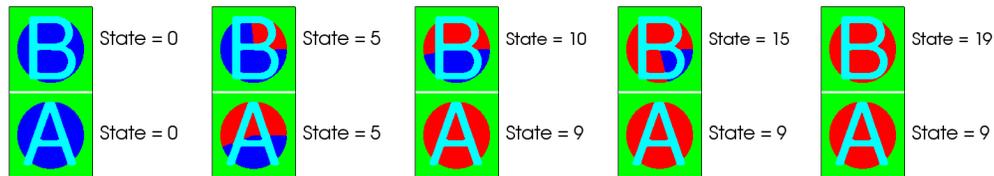


Figure 12-6: Padded index database correlation of A and B (every 5th time state)

3.2.2 Stretched index database correlation

A stretched index database correlation, like any other database correlation, involves multiple input databases where each database potentially has a different number of time states. Like a padded index database correlation, a stretched index database correlation also has as many time states as the input database with the largest number of time states. The difference between the two correlation methods is in how the input databases are mapped to a larger number of time states. The padded index database correlation method simply repeated the last frame of the input databases that needed more time states to be made even with the length of the database correlation. Stretched index database correlations on the other hand do not repeat only the last frame; they repeat frames throughout the middle time states until shorter input databases have the same number of time states as the database correlation. The effect of repeating time states throughout the middle is to evenly spread out the time states over a larger number of time states.

Stretched index database correlations are useful for comparing related related simulation databases where one simulation wrote out data at 2x, 3x, 4x, ... the frequency of another simulation. Stretched index database correlations repeat the data for smaller databases, which makes it easier to compare the databases. Figure 12-7 shows example databases A and B related using a stretched index database correlation. Note how the plots for both databases, even though the databases contain a different number of time states, remain roughly in sync.

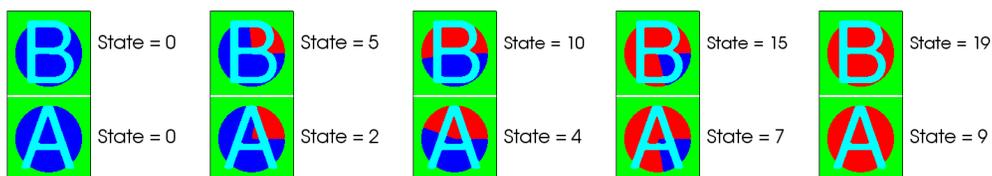


Figure 12-7: Stretched index database correlation of A and B (every 5th time state)

3.2.3 Time database correlation

A time index database correlation, like any other database correlation, involves multiple input databases where each database potentially has a different number of time states. The number of time states in a time database correlation is not directly related to the number of time states in each input database. The number of time states in the database correlation are instead determined by counting the number of unique time values for every time state in every input database. The times from each input database are arranged on a number line and each unique time value is counted as one time state. Time values from different input databases that happen to have the same time value are counted as a single time state. Once the time values have been arranged on the number line and counted, VisIt calculates a list of time state indices for each database that identify the right time state to use for each database with respect to the time database correlation's time state. The first time state for each database is always the first time state index stored for a database. The first time state is used until the time exceeds the first time on the number line, and so on.

Time database correlations are useful in many of the same situations as stretched index database correlations since they are both used to align different databases in time. Unlike a stretched index database correlation, the time database correlation does a better job of aligning unrelated databases in actual simulation time rather than just spreading out the time states until each input database has an equal number. Use a time database correlation when you are correlating two or more databases that were generated with different dump frequencies or databases that were generated by totally different simulation codes. Figure 12-8 shows the behavior of databases A and B when using a time database correlation.

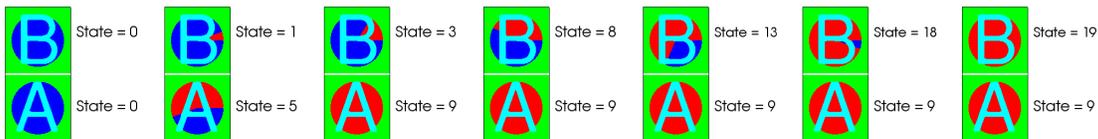


Figure 12-8: Time database correlation of A and B (every 5th time state)

3.2.4 Cycle database correlation

Cycle database correlations operate in exactly the same way as time database correlations except that they correlate using the cycles from each input database instead of using times. Figure 12-9 shows the behavior of databases A and B when using a cycle database correlation.

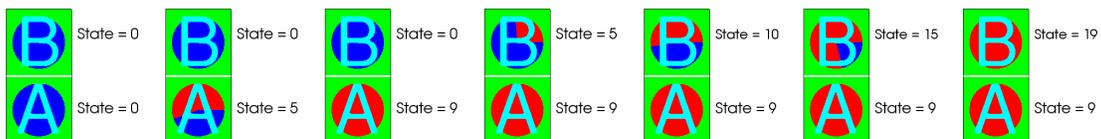


Figure 12-9: Cycle database correlation of A and B (every 5th time state)

3.3 Managing database correlations

If you want to create a new database correlation or edit properties related to database correlations, you can use the Database Correlation Window. You can open the Database Correlation Window, shown in Figure 12-10, by clicking on the Database correlations option in the Main Window's Controls menu. The Database Correlation Window contains the list of database correlations, along with controls that allow you to create new database correlations, edit existing database correlations, delete database correlations, or set global settings that tell VisIt when to automatically create database correlations.

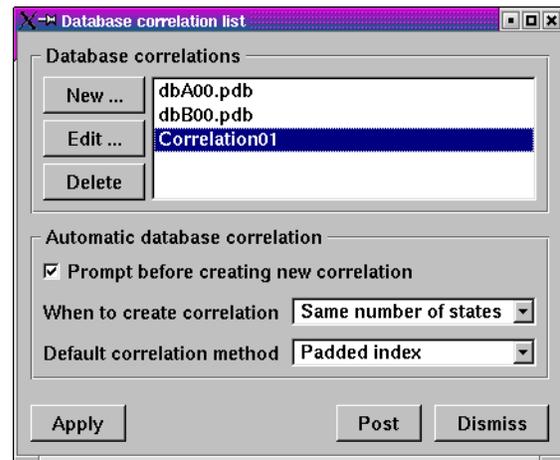


Figure 12-10: Database Correlation Window

3.3.1 Creating a new database correlation

If you want to create a new database correlation to relate time-varying databases that you have opened, you can do so by opening the **Database Correlation Window**. The **Database Correlation Window** contains a list of trivial database correlations for the time-varying databases that you have opened. You can create a new, database correlation by clicking on the New button to the left of the list of database correlations. Clicking the New button opens a **Database Correlation Properties Window** (Figure 12-11) that you can use to edit properties for the database correlation.

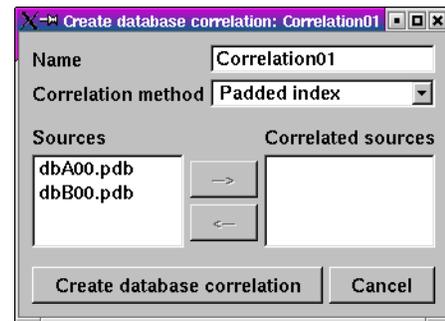


Figure 12-11: Database Correlation Properties Window

New database correlations are automatically named when you first create them but you can change the name of the database correlation to something more memorable by entering a new name into the Name text field. Once you have entered a name, you should set the correlation method that the database correlation will use to relate the time states from all of the input databases. The available choices, shown in Figure 12-12, are: padded index, stretched index, time, and cycle.

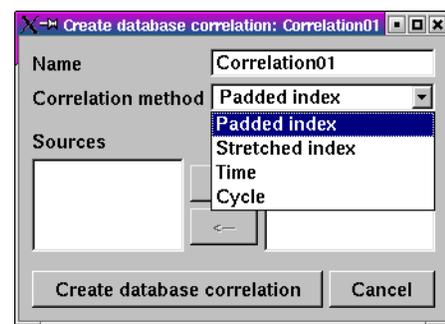


Figure 12-12: Correlation methods

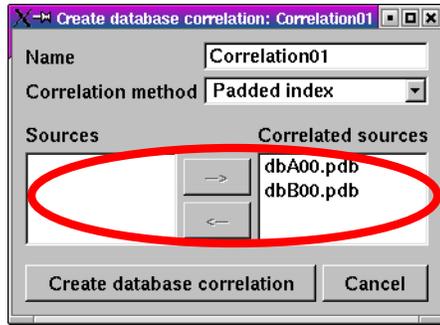


Figure 12-13: Sources list and Correlated sources list

Once you have chosen a correlation method, it is time to choose the input databases for the correlation. The input databases, or sources as they are sometimes called in VisIt, are listed in the **Sources** list (see Figure 12-13). The **Sources** list only contains the databases that you have opened so far. If you do not see a database that you would like to have in the database correlation, you can either click the **Cancel** button to cancel creating the new database correlation or you can continue creating the database correlation and then add the other database to the correlation later after you have opened it. To add databases to the new

database correlation, click on the them in the **Sources** list to highlight then and then click on the **Right arrow** button to move the highlighted databases into the database correlation's **Correlated sources** list. If you want to remove a database from the **Correlated sources** list, highlight the database in the **Correlated sources** list and then click the **Left arrow** button to move it back to the **Sources** list. Once you are satisfied with the new database correlation, click the **Create database correlation** button to create a new database correlation.

When you create a new database correlation, VisIt also creates a new time slider for the new database correlation. The database correlation's active time state is initially set to the first time state, which might not match the time state of individual plots in the vis window. Once you change time states using the **Time slider**, the plots in the vis window will be updated using the correct time state with respect to the correlation's active time state. As always, if you want to update the time state for only one database, you can select a different time slider using the **Active time slider** combo box and then change time states using the **Time slider**. Any time state changes made to an individual database that is also an input database for a database correlation has no effect on the database correlations that involve the changed database. Time state changes for a database correlation can only happen if you have selected the database correlation as your active time slider.

3.3.2 Altering an existing database correlation

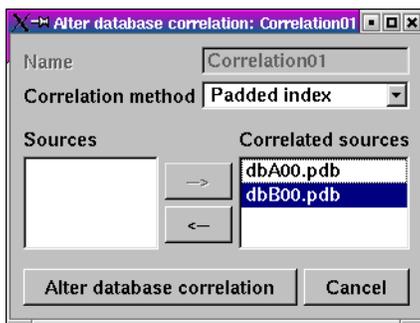


Figure 12-14: Altering a database correlation

Once you've created a database correlation, you can alter it at any time by highlighting it in the **Correlation** list in the **Database Correlation Window** and clicking the **Edit** button to the left of the **Correlation** list. Clicking the **Edit** button opens the **Database Correlation Properties Window** and allows you to change the correlation method and the input databases. Once you've made the desired changes, clicking the **Alter database correlation** button will make the specified database correlation use the new options and all plots in all vis windows

that are subject to the changed database correlation will update to the new time states prescribed by the altered database correlation.

Using the **Database Correlation Properties Window** explicitly alters a database correlation. Reopening a file or refreshing the file list can implicitly alter a database correlation if after reopening the affected databases, there are different numbers of time states in the databases. When reopened databases that are input databases to database correlations have a new number of time states, VisIt recalculates the indices used to access the input databases via the time slider and updates any plots that were affected. In addition to the time state indices changing, the number of time states in the database correlation and its time slider can also change.

3.3.3 Deleting a database correlation

Database correlations are automatically deleted when you close a database that you are not using anymore provided that the closed database is not an input database to any database correlation except for that database's trivial database correlation. You can delete non-trivial database correlations that you have created by highlighting a database correlation in the **Correlation** list in the **Database Correlation Window** and clicking the **Delete** button to the left of the **Correlation** list. When you delete a database correlation, the new active time slider will be set to the active database's time slider if the active database has more than one time state. Otherwise, the new active time slider, if any, will be set to the time slider for the first source that has more than one time state.

3.3.4 Automatic database correlation

VisIt can automatically create database correlations when they are needed if you enable certain global settings to control the creation of database correlations. By default, will prompt you to when it wants to create a database correlation. VisIt can automatically create a database correlation when you add a plot of a multiple time-varying database to a vis window that already contains a plot from a different time-varying database. VisIt first looks for the most suitable existing database correlation and if the one it picks must be modified to accommodate a new input database or if an entirely new database correlation must be created, VisIt will prompt you using a **Correlation question** dialog (Figure 12-15). If you prevent VisIt from creating a database correlation or altering the most suitable correlation, you will no longer be prompted to create a database correlation for the list of databases listed in the **Correlation question** dialog.

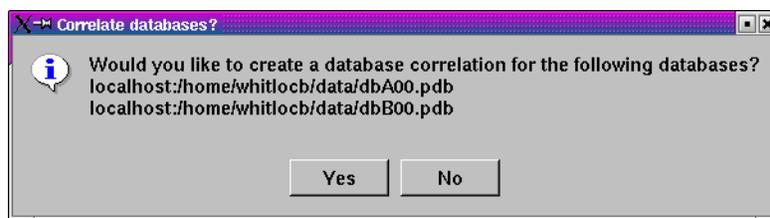


Figure 12-15: Correlation question dialog

By default, VisIt will only attempt to create a database correlation for you if the new plot's database has the same number of time states as the existing plot. You can change when VisIt creates a database correlation for you by selecting a different option from the **When to create correlation** combo box in the **Database Correlation Window**. The available options are: **Always**, **Never**, and **Same number of states**. You can change the default correlation method by selecting a new option from the **Default correlation method** combo box. Finally, you can prevent VisIt from prompting you when it needs to create a database correlation if you turn off the **Prompt before creating new correlation** check box.

4.0 Database comparison

Comparing the results of multiple related simulation databases is one of VisIt's main uses. You can put plots from multiple databases in the same window or you can put plots from each database in adjacent visualization windows, allowing you to compare plots visually. In addition to these visual comparison modes, VisIt supports more direct comparison of databases using database comparisons. Database comparison allows you to plot direct differences between two databases or between different time states in the same database (i.e. time derivatives). VisIt's expression language is used extensively to facilitate the database comparison.

4.1 The role of expressions

Database comparison is accomplished by creating new expressions that involve the contents of multiple databases using VisIt's expression language. Database comparisons use a special expression called *conn_cmfe*, which is capable of mapping a field from one mesh to another mesh using a mesh connectivity-based approach. The name "conn_cmfe" means *connectivity-based common mesh field evaluation* (CMFE) and as the name implies, the expression takes fields from one mesh and maps the field onto another mesh by taking the cell or node-centered values on the donor mesh and mapping them onto the cells or nodes having the same indices in the new mesh. Mismatches in mesh sizes result in not all data values being used or, alternatively, VisIt can pad the remapped field with zeroes. The *conn_cmfe* expression can be used to map fields from one database onto a mesh in another database, which then allows you to create expressions involving the active database.

There are plans to add position-based CMFE, which will resample the field from one mesh onto another mesh by calculating the values of the field on the first mesh using the locations of the cells or nodes in a second mesh.

4.2 Plotting the difference between two databases

Simulations are often run as parts of parameter studies, meaning the initial conditions are changed slightly to see what observable effects the changes produce. The ability to do

direct numerical comparison of multiple simulation databases is required in order to observe the often miniscule differences between the fields in the two databases. VisIt provides the `conn_cmfe` expression, which allows you to map a field from one simulation database onto a mesh from another simulation database. Once the mapping has been done, you can then perform difference operations using an expression like this:

```
<mesh/ireg> - conn_cmfe(</usr/local/visit/data/dbB00.pdb:mesh/ireg>, mesh)
```

The expression above is a simple difference operation of database A minus database B. The assumption made by this expression is that database A is the active database and we're trying to map database B onto it so we can subtract it from database A's `mesh/ireg` variable. Note that the `conn_cmfe` expression takes two arguments. The first argument encodes the name of the file and the field that we're mapping onto a mesh from the active database, where the mesh name is given by the second argument. In this example, we're mapping database B's `mesh/ireg` field onto database A's `mesh`. Figure 12-16 shows a picture that illustrates the database differencing operation.

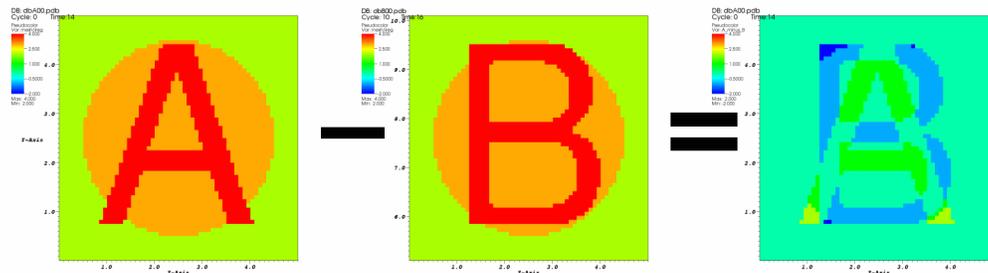


Figure 12-16: Database B subtracted from database A.

4.3 Plotting values relative to the first time state

Plotting a variable relative to its initial values can be important for understanding how the variable has changed over time. The `conn_cmfe` expression is also used to plot values from one time state relative to the values at the current time state. Consider the following expression:

```
<mesh/ireg> - conn_cmfe(</usr/local/visit/data/dbA00.pdb[0]i:mesh/ireg>, mesh)
```

The above expression subtracts the value of `mesh/ireg` at time state zero from the value of `mesh/ireg` at the current time state. The interesting feature about the above expression is its use of the expression language's `[]` operator for specifying a database time state. The expression uses `"[0]i"`, which means use time state zero because the `"i"` suffix indicates that the number inside of the square brackets is to be used as an absolute time state index. As you change the time slider, the values for the current time state will change but the part of the expression using `conn_cmfe`, which in this case uses the first database time state, will not change. This allows you to create expressions that compare the current time state

to a fixed time state. You can, of course, substitute different time state indices into the `conn_cmfe` expression so you don't have to always compare the current time state to time state zero.

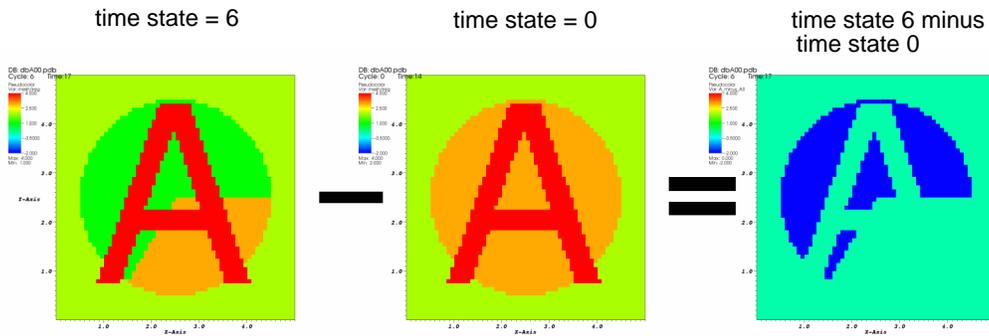


Figure 12-17: Time state 6 minus time state 0.

4.4 Plotting time derivatives

Plotting time derivatives is much like plotting the difference between the current time state and a fixed time state except that instead of being fixed, the second time state being compared is free to move relative to the current time state. To plot a simple time derivative such as the current time state minus the last time state, create an expression similar to the following expression:

```
<mesh/ireg> - conn_cmfe(</usr/local/visit/data/dbA00.pdb[-1]id:mesh/ireg>, mesh)
```

The important piece of the above expression is its use of “`[-1]id`” to specify a time state delta of -1, which means add -1 to the current time state to get the time state whose data will be used in the `conn_cmfe` calculation. You could provide different values for the time state in the `[]` operator. Substituting a value of 3, for example, would make the `conn_cmfe` expression consider the data for 3 time states beyond the current time state. If you use a time state delta, which always uses the “`d`” suffix, the time state being considered is always relative to the current time state. This means that as you change time states for the

active database using the time slider, the plots that use the `conn_cmfe` expression will update properly. Figure 12-18 shows an example plot of a time derivative.

Plot of the original variable



Plot of the time derivative

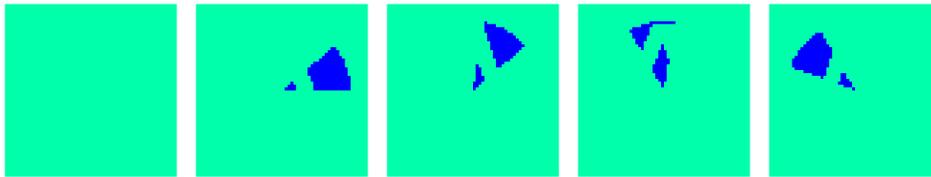


Figure 12-18: Plot of a variable and its time derivative plot

5.0 Multiple Window Operations

This section focuses on some of the common techniques for exploring multiple databases when you have multiple visualization windows.

5.1 Reflection and Translation

When you visualize multiple related databases, they often occupy the same space in the visualization window since they may have been generated using the same computational mesh but with different physics. When this is the case, you can modify the location of the plots from one of the databases in two immediately obvious ways. First of all, if you simulated the same object and it does not make use of any symmetry then you could use VisIt's Transform operator to translate the coordinate system of one of the plots out of the way of the other plot so you can look at the two plots from the different databases side by side in the same visualization window. If your databases make use of symmetry - maybe you only simulated half of the problem, then you can apply VisIt's Reflect operator to one

of the plots to show them side by side but reflected to show the entire problem. Each method has its merits.

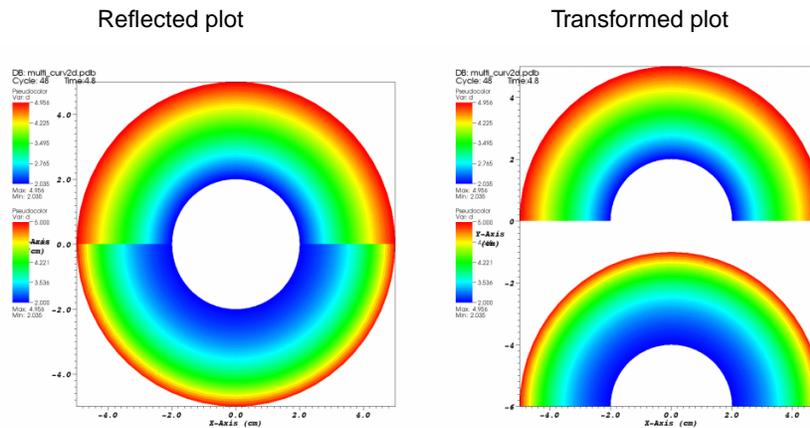


Figure 12-19: Plots side by side using the Reflect or Transform operator

5.2 Copying Windows

If you visualize multiple databases and you want to create identical plots for each database but have them placed in different visualization windows then you can either have VisIt copy windows on first reference or you can clone an existing window and then replace the database used in the new window's plots with a different database.

If you have already created multiple visualization windows, perhaps as the result of a change to VisIt's layout, then you can make VisIt copy the attributes of the active window to another visualization window when you switch active windows by enabling **Clone window on first reference** in the **Preferences Window**. To open the **Preferences Window**, choose the **Preferences** option from the **Main Window's Options** menu. This form of window cloning copies the plots, lights, colors, etc from the active window to a pre-existing visualization window when you access it for the first time. If you have already accessed a visualization window but you would still like to copy plots, lights, colors, etc from another visualization window, you can make the destination visualization window be the active window and then copy everything from the source visualization window using the **Copy everything** menu option in the **Main Window's Window** menu.

If you have no empty visualization window to contain plots for the another database, you can click the **Clone** option in the **Main Window's Window** menu to create a new visualization window with the same plots an settings as the active window. Once the new window has been created, you could visualize a new database by highlighting a new database in the **File panel's Selected files** list and clicking the **Replace** button.

5.3 Locking Windows

When you visualize databases using multiple visualization windows, it is often convenient to keep the time state and view in sync between windows so you can concentrate on comparing plots instead of dealing with the intricacies of setting the view or time state for each visualization window. VisIt's visualization windows can be locked with respect to time, view, or interactive tools. You can use the **Lock** toolbar, shown in Figure 12-20, to lock visualization windows together with respect to view and time.

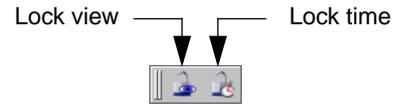


Figure 12-20: Lock toolbar

5.3.1 Locking views

If you have created plots from related databases in multiple visualization windows, you can lock the views for the visualization windows together so as you change the view in one of the visualization windows with a locked view, the other visualization windows with locked views also update to have the same view. There are three types of views in VisIt: curve, 2D, and 3D. If you have 2D plots in a visualization window, the visualization window is considered to be 2D. Locking that 2D visualization window's view will only update other visualization windows that are also 2D and vice-versa. The same is true for curve and 3D views. To lock a visualization window's view, select the **Lock->View** option from the **Main Window's Window** menu or use the visualization menu's **Popup menu** or **Toolbar**.

5.3.2 Locking time

If you have created plots from related databases in multiple visualization windows, you can lock the visualization windows together in time so as you change time in one visualization window, it updates in all other visualization windows that are locked in time. To lock a visualization window in time, select the **Lock->Time** option from the **Main Window's Window** menu.

Locking visualization windows together in time may cause VisIt to prompt you to create a new database correlation that involves all of the databases in the visualization windows that are locked in time. VisIt creates a database correlation because the visualization windows must use a common time slider to really be locked in time. If the visualization windows did not use a common time slider then changing time in one visualization window would not cause other visualization windows to update. Once VisIt creates a suitable database correlation for all windows, the active time slider is set to that database correlation in all visualization windows that are locked in time. If you alter a database correlation at this point, it will cause the time state in each locked visualization window to change. Since the same database correlation is used in all locked visualization windows, changing the time state for the database correlation changes the time state in all of the locked windows. This frees you to examine time-varying database behavior without having to set the time state independently in each visualization window.

5.3.3 Locking tools

In addition to locking visualization windows together with respect to the view and time, you can also lock their tools. This capability can be useful when exploring data that often requires the use of an operator whose attributes can be set interactively using a tool since the same tool can be used to set the operator attributes for operators in more than one visualization window. To lock a visualization window's tools, select the **Lock->Tools** option from the **Main Window's Window** menu.

Consider the following scenario: you have two related 3D databases and you want to examine the same slice plane for each database and you want each database to be plotted in a separate visualization window. You can set up separate visualization windows and slice the plots from each database independently but locking tools is easier and requires much less setup.

Start off by opening the first 3D database and create the desired plots from it. If you want to maintain a 3D view of the plots, you can clone the visualization window to get a new window with the same plots or you can apply a Slice operator to the plots. Apply a Slice operator but make sure the slice is not projected to 2D and also be sure that its **Interactive** check box is turned on. Turn on VisIt's plane tool and make sure that tools are locked. Clone the visualization window twice and for each of the new visualization windows, make sure that their Slice operator projects to 2D. There should now be four visualization windows if you opted to keep a 3D view of the data. In the last visualization window, replace the database with another related database that you want to compare to the first database.

Now that you've performed all of the setup steps, you can save a session file so you can get back to this state when you run VisIt next time. Now, in the window that still has a slice in 3D, use the plane tool to reposition the slice. Both of the 2D visualization windows should also update so they use the new slice plane attributes calculated by the plane tool.

The four visualization windows, arranged in a 2x2 window layout are shown in Figure 12-21.

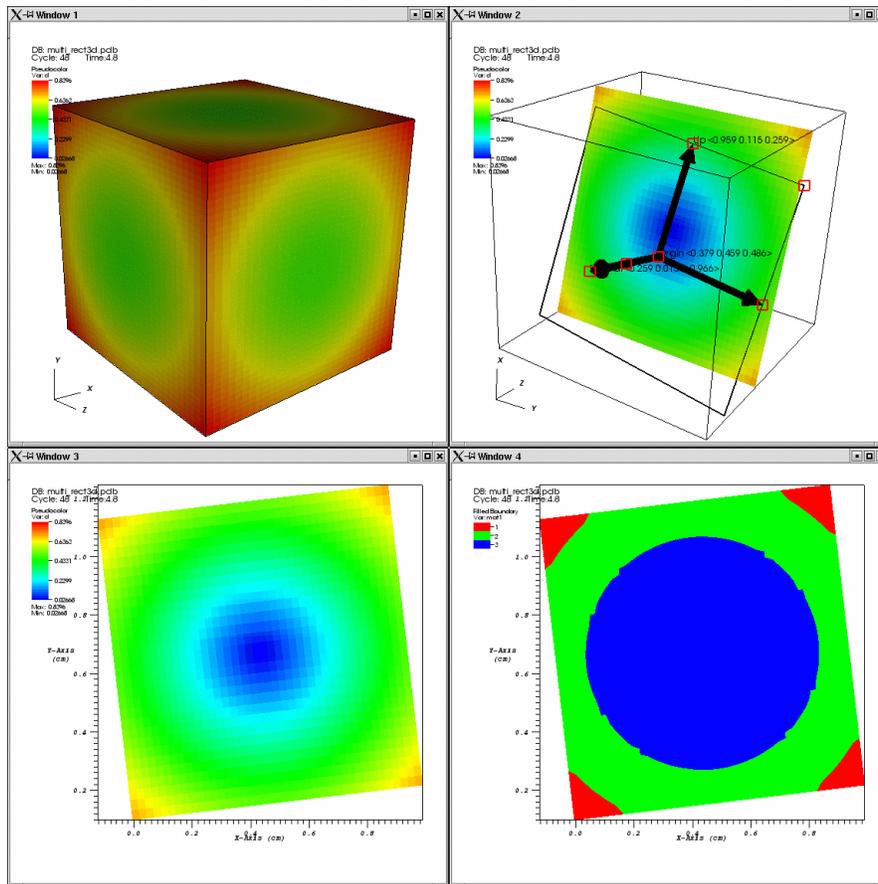


Figure 12-21: Multiple visualization windows with locked tools

1.0 Overview

Scientific simulations are almost always run on a powerful supercomputer and accessed using desktop workstations. This means that the databases usually reside on remote computers. In the past, the practice was to copy the databases to a visualization server, a powerful computer with very fast computer graphics hardware. With ever increasing database sizes, it no longer makes sense to copy databases from the computer on which they were generated. Instead, it makes more sense to examine the data on the powerful supercomputer and use local graphics hardware to draw the visualization. VisIt can run in a distributed mode that allows this exact use case. The GUI and viewer run locally while the database server and parallel compute engine run on the remote supercomputer. Running VisIt in distributed mode is almost as easy as running all components locally. This chapter explains the differences between running locally and remotely and describes how to run VisIt in distributed mode.

2.0 Distributed mode

When you run VisIt locally, you usually select files, open a database, and create plots using the open database. Fortunately, the procedure for running VisIt in distributed mode is no different than it is for running in single-computer mode. You begin by opening the **File Selection Window** and typing the name of the computer where the files are stored into the **Host** text field.

Once you have told VisIt which host to use when accessing files, VisIt launches the VisIt Component Launcher (VCL) on the remote computer. The VCL is a VisIt component that runs on remote computers and is responsible for launching other VisIt components such as the database server and compute engine (Figure 13-1). Once you are connected to the

remote computer and VCL is running, you won't have to enter a password again for the remote computer because VCL stays active for the life of your VisIt session and it takes care of launching VisIt components on the remote computer.

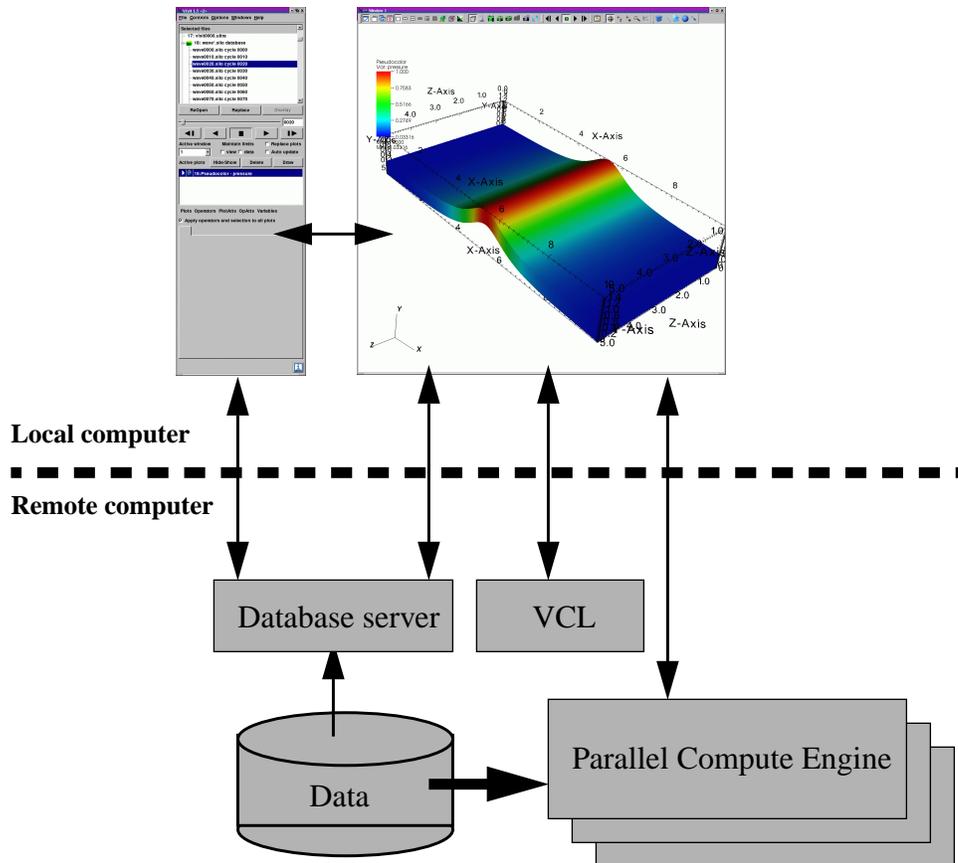


Figure 13-1: VisIt's architecture

If VCL was able to launch on the remote computer and if it was able to successfully launch the database server, the files for the remote computer will be listed in the **File Selection Window**. Add the files to be visualized to the **Selected files list** as you would with local files and dismiss the **File Selection Window**. Now that you have files from the remote computer at your disposal, you can create plots as usual.

2.1 Passwords

Sometimes when you try to access files on a remote computer, VisIt prompts you for a password by opening a **Password Window** (Figure 13-2). If you are prompted for a password, type your password into the window and click the **Ok** button. If the password window appears and you decide to abort the launch of the remote component, you can click the **Password Window's Cancel** button to stop the remote component from being launched.

VisIt uses *ssh* for authentication and you can set up *ssh* so that passwords are not required. This is called passwordless *ssh* and once it is set up for a computer, VisIt will no longer need to prompt for a password. More information about setting up passwordless *ssh* can be found in **Appendix B**.

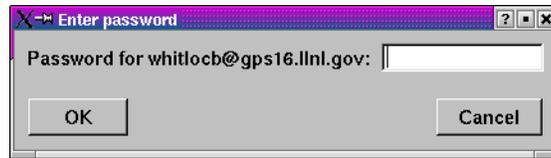


Figure 13-2: Password Window

2.2 Environment

It is important to have VisIt in your default search path instead of specifying the absolute path to VisIt when starting it. This isn't as important when you run VisIt locally, but VisIt may not run properly in distributed mode if it isn't in your default search path on remote machines. If you regularly run VisIt using the network configurations provided for LLNL computers then VisIt will have host profiles, which are sets of information that tell VisIt how to launch its components on a remote computer. The provided host profiles have special options that tell the remote computer where it can expect to find the installed version of VisIt so it is not required to be in your path. If you did not opt to install the provided network configurations or if you are at a site that requires other network configurations then you will probably not have host profiles by default and it will be necessary for you to add VisIt to your path on the remote computer. You can add VisIt to your default search path on UNIXTM systems by editing the initialization file for your command line shell.

2.3 Launch progress window

When VisIt launches a compute engine or database server, it opens the **Launch progress window** when the component cannot be launched in under four seconds. An exception to this rule is that VisIt will always show the **Launch progress window** when launching a parallel compute engine or any compute engine on MacOS X. VisIt's components frequently launch fast enough that it is not necessary to show the **Launch progress window** but you will often see it if you launch compute engines using a batch system.

The **Launch progress window** indicates VisIt is waiting to hear back from the component being launched on the remote computer and gives you some indication that VisIt is still alive by animating a set of moving dots representing the connection from the local computer to the remote computer. The icon used for the remote computer will vary depending on whether a serial or parallel VisIt component is being launched. The **Launch progress window** for a parallel compute engine is shown in Figure 13-3. The window is visible until the remote compute engine connects back to the viewer or the connection is cancelled. If you get tired of waiting for a remote component to launch, you can cancel it

by clicking the **Cancel** button. Once you cancel the launch of a remote component, you can return to your VisIt session. Note that if the remote compute is a parallel compute engine launched via a batch system, the engine will still run when it is finally scheduled but it will immediately dies since VisIt has stopped listening for it. On heavily saturated batch systems, it might be prudent for you to manually remove your compute engine job from the batch queue.



Figure 13-3: Launch progress window

3.0 Host Profiles

When VisIt launches a component on a remote computer, it looks for something called a *host profile*. A host profile contains information that VisIt uses to launch components on a remote computer. Host profiles allow you to specify information like: remote username, number of processors, parallel launch method, etc. You can have multiple host profiles for any given computer. Host profiles are most often used to specify options for running VisIt's parallel compute engine on remote computers and often the same host will have a serial host profile and several other host profiles with different parallel options.

3.1 Host Profile Window

VisIt provides a **Host Profile Window**, shown in Figure 13-4, that you can use to manage your host profiles. You can open the **Host Profile Window** by choosing **Host profiles** from the **Main Window's File** menu. The **Host Profiles Window** is divided vertically into two main areas. The top area contains a group of tabs where each tab contains the host profiles for a given computer. The top area also contains controls to create and delete host profiles. The bottom area of the window displays all attributes for the selected host profile.

3.1.1 Creating a new host profile

You click the **New profile** button to create a new host profile based on the selected host profile. If no host profile exists or if no host profile is selected, the new host profile contains reasonable default values. In any case, the host profile has a default name that you can change to make the new host profile more memorable. If the host profile is for a computer for which there are no other host profiles, a tab is created with the name of the new computer and the new host profile is added to the new tab.

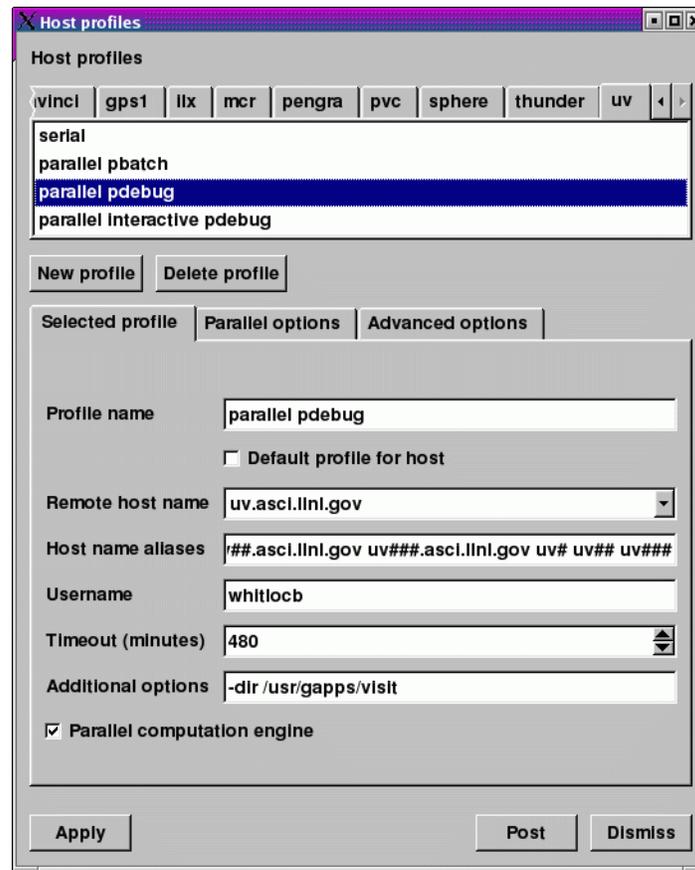


Figure 13-4: Host Profile Window

3.1.2 Deleting a host profile

If a host profile is no longer useful, you can click on it in the tabbed list to select it and then click the **Delete** button to delete it.

3.1.3 Activating a host profile

Only one host profile can be active for any given computer. When VisIt launches a remote component, it looks for the active host profile for the computer where the component is to be launched. To activate a different host profile, click the **Active profile for host** toggle button in the **Selected profile** settings. The VCL and the database server use the active host profile but VisIt will prompt you for a host profile to use before launching a compute engine if you have more than one host profile or your only host profile has parallel options set for the compute engine.

3.1.4 Setting the host name

The host name is the name of the computer for which the host profile is created. You can change the host name by typing a new host name into the **Remote host name** combo

box. You can also select an existing host name, if there are any, from the **Remote host name** combo box. If you provide a host name, you can provide the fully qualified computer name (hostname.domain.net) or just the host name. Some systems require the fully qualified name.

Some clustered systems have one overall host name but also have names for the individual compute nodes that comprise the system. The compute nodes are often named by appending the node number to the host name. For example, if the clustered system is called: *cluster*, you might be logged into node *cluster023*. When you launch a remote component, VisIt will not find any host profiles if the host name in the host profiles is: *cluster*.

To ensure that VisIt correctly matches a computer's node name to one of VisIt's host profiles, you should include host name aliases in the host profile for a clustered system. Host name aliases typically consist of the host name with different wildcard characters appended to it. Three wildcards are supported. The '?' wildcard character lets any one character replace it while the '*' wildcard character lets any character or group of characters replace it and the '#' wildcard character lets any numeric digit replace it. Appropriate host aliases for the previous example would be: *cluster#*, *cluster##*, *cluster###*, etc. If you need to enter host name aliases for the host profile, type them into the **Host name aliases** text field.

3.1.5 Setting the remote user name

The remote user name is the name of the account that you want to use when you access the remote computer. The remote user name does not have to match your local user name and it is often the case that your desktop user name will not match your remote user name. To change the remote user name, type a new user name into the **Username** text field in the **Selected profile** settings.

3.1.6 Setting the timeout

The compute engine and database server have a timeout mechanism that causes them to exit if no requests have been made of them for a certain period of time so they do not run indefinitely if their connection to VisIt's viewer is severed. You can set this period of time, or timeout, by typing in a new number of minutes into the **Timeout** text field. You can also increase or decrease the timeout by clicking on the up and down arrows next to the **Timeout** text field.

3.1.7 Providing additional command line options

The **Host Profile Window** allows you to provide additional command line options to the compute engine and database server through the **Additional options** text field. When you provide additional command line options, you should type them, separated by spaces, into the **Additional options** text field. Command line options influence how the compute engine and database server are executed. For more information on VisIt's

command line options, see **Appendix A**. Most of the host profiles that are installed with VisIt specify the expected installation directory for VisIt so VisIt does not have to be in your path on remote computes.

3.2 Setting parallel options

One of the chief purposes of host profiles is to make launching compute engines easier. This is even more the case when host profiles are used to launch parallel compute engines on large computers that often have complex scheduling programs that determine when parallel jobs can be executed. It is easy to forget how to use the scheduling programs on a large computer because each scheduling program requires different arguments. In order to make launching compute engines easy, VisIt hides the details of the scheduling program used to launch parallel compute engines. VisIt instead allows you to set some common parallel options and then figures out how to launch the parallel compute engine on the specified computer using the parallel options specified in the host profile. Furthermore, once you create a host profile that works for a computer, you rarely need to modify it.

Figure 13-5: Parallel options

3.2.1 Setting the parallel launch method

The parallel launch method option allows you to specify which launch program should be used to execute the parallel compute engine. This setting depends on the computer where you plan to run the compute engine and how the computer is configured. Some computers have multiple launch programs depending on which part of the parallel machine you want to use. The table below lists some common Operating System/Launch program pairs.

Operating System	Launch program
IRIX	mpirun
AIX	poe, interactive partition

AIX	psub, batch partition
Linux CHAOS	psub/srun, both interactive and batch
Tru64	dmpirun

In addition to choosing a launch program, you can also elect to give it additional command line options to influence how it launches your compute engine. To give additional command line options to the launch program, click the **Additional launcher arguments** check box and type command line options into the text field to the right of that check box.

3.2.2 Setting the partition/pool

Some parallel computers are divided into partitions so that batch processes might be executed on one part of the computer while interactive processes are executed on another part of the computer. You can use host profiles tell VisIt which partition to use when launching the compute engine on systems that have multiple partitions. To set the partition, check the **Partition/Pool** check box and type a partition name into the **Partition/Pool** text field.

3.2.3 Setting the number of processors

You can set the number of processors by typing a new number of processors into the **Default number of processors** text field. When the number of processors is greater than 1, VisIt will attempt to run the parallel version of the compute engine. You can also click on the up and down arrows next to the text field to increase or decrease the number of processors. If VisIt finds a parallel host profile, you will have the option of changing the number of processors before the compute engine is actually launched.

3.2.4 Setting the number of nodes

The number of nodes refers to the number of compute nodes that you want to reserve for your parallel job. Each compute node typically contains more than one processor (often 2, 4, 16) and the number of nodes required is usually the ceiling of the number of processors divided by the number of processors per node. It is only necessary to set the number of nodes if you want to use fewer processors than the number of processors that exist on a compute node. This option is not available on some computers as it is meant primarily for compute clusters. To set the number of nodes, check the **Default number of nodes** check box and type a number of processors per node into the **Default number of nodes** text field.

3.2.5 Load balancing

Load balancing refers to how well tasks are distributed among computer processors. The goal is to make each computer processor have roughly the same amount of work so they all finish at the same time. VisIt's compute engine supports two forms of load balancing.

The first form is static load balancing where the entire problem is distributed among processors and that distribution of work never changes. The second form of load balancing is dynamic load balancing. In dynamic load balancing, the work is redistributed as needed each time work is done. Idle processors independently ask for work until the entire task is complete. VisIt allows you to specify the form of load balancing that you want to use. You can choose to use static or dynamic load balancing by clicking the **Static** or **Dynamic** radio buttons. There is also a default setting that uses the most appropriate form of load balancing.

3.2.6 Setting the default bank

Some computers, if they are large enough, have scheduling systems that break up the number of processors into banks, which are usually reserved for particular projects. Users who contribute to a project take processors from their default bank of processors. By default, VisIt uses environment variables to get your default bank when submitting a parallel job to the batch system. If you want to override those settings, you can click the **Default Bank** check box to turn it on and then type your desired bank into the text field next to the check box.

3.2.7 Setting the time parallel limit

The parallel time limit is the amount of time given to the scheduling program to tell it the maximum amount of time, usually in minutes, that your program will be allowed to run. The parallel time limit is one of the factors that determines when your compute engine will be run and smaller time limits often have a greater likelihood of running before jobs with large time limits. To specify a parallel time limit, click the **Default Time Limit** check box and enter a number of minutes or hours into the **Default Time Limit** text field. If you want to specify minutes, be sure to append “m” to the number or append an “h” for hours. If you want to specify a timeout of 30 minutes, you would type: *30m*.

3.2.8 Specifying a machine file

When using VisIt with MPICH on some clustered computers, it is necessary to specify a machine file, which is a file containing a list of the compute nodes where the VisIt compute engine should be executed. If you want to specify a machine file when you execute VisIt in parallel on a cluster that requires a machine file, click on the **Default Machine File** check box and type the name of the machine file that you want to associate with your host profile into the **Default Machine File** text field.

3.3 Advanced host profile options

Figure 13-6: Advanced options tab

Host profiles contain the information that VisIt needs in order to successfully launch VisIt components on remote computers and to make sure that they are launched with enough computing resources. Most of those properties can be set using the **Selected profile** and **Parallel options** tabs of the **Host Profile Window** but there is also an **Advanced options** tab (see Figure 13-6) in the Host Profile Window that lets you specify advanced networking options to ensure that the VisIt components running on the remote computer use resources correctly and can

connect back to the viewer running on your local workstation.

3.3.1 Sharing a compute job

Some computers, notably Los Alamos National Laboratory's Q machine, place restrictions on the number of interactive sessions that a single user can have on the computer. To allow VisIt to run on computer systems that enforce these kinds of restrictions, VisIt can optionally force the database server and parallel compute engine to share the same job in the batch system. If you want to make the database server and parallel compute engine share the same batch job, you can click the **Share batch job with Metadata Server** check box on the **Host profiles Window's Advanced options tab**.

3.3.2 Setting up the parallel environment

VisIt is usually executed by a script called: *visit*, which sets up the environment variables required for VisIt to execute. When the *visit* script is told to launch a parallel compute engine, it sets up the environment variables as it usually does and then invokes an appropriate parallel launch program that takes care of either spawning the VisIt parallel compute engine processes or scheduling them to run in a batch system. When VisIt is used with MPICH on some clusters, the parallel launch program does not replicate the environment variables that the *visit* script set up, preventing the VisIt parallel compute engine from running. On clusters where the parallel launch program does not replicate the VisIt environment variables, VisIt provides an option to start each process of the VisIt compute engine under the *visit* script. This ensures that the environment variables that VisIt requires in order to run are indeed set up before the parallel compute engine

processes are started. To enable this feature, click on the **Use VisIt script to set up parallel environment** check box on the **Host profiles Window's Advanced options tab**.

3.3.3 Determining the host name

There are many different network naming schemes and each major operating system type seems to have its own variant. While being largely compatible, the network naming schemes sometimes present problems when you attempt to use a computer that has one idea of what its name is with another computer that may use a somewhat different network naming scheme. Since VisIt users are encouraged to use distributed mode because it provides fast local graphics hardware without sacrificing computing power, VisIt must provide a way to reconcile the network naming schemes when 2 different computer types are used.

Workstations often have a host name that was arbitrarily set when the computer was installed and that host name has nothing to do with the computer's network name, which ultimately resolves to an IP address. This condition is common on computers running MS Windows though other operating systems can also exhibit this behavior. When VisIt launches a component on a remote computer, it passes information that includes the host name of the local computer so the remote component will know how to connect back to the local computer. If the local computer did not supply a valid network name then the remote component will not be able to connect back to the local computer and VisIt will wait for the connection until you click the **Cancel** button in the **Launch progress window**.

By default, VisIt relies on the name obtained from the local computer but if you want to specify a name instead of using the name reported by the local computer then you can use the controls on the **Advanced options** tab. To use a host name other than what the local computer returns, you can click the **Parse from SSH_CLIENT environment variable** or **Specify manually** radio buttons. If you choose the **Parse from SSH_CLIENT environment variable** option then VisIt will not pass a host name for the local computer but will instead tell the remote computer to inspect the *SSH_CLIENT* environment variable to determine the IP address of the local computer that initiated the connection. This option usually works if you have a local computer that does not accurately report its host name. If you don't trust the output of any implicit scheme for getting the local computer's name, you can provide the name of the local computer by typing its name or IP address into the text field next to the **Specify manually** radio button.

3.3.4 VisIt's ports

VisIt uses secure shell (ssh) to launch its components on remote computers. Secure shell often uses port 22 but if you are attempting to communicate with a computer that does not use port 22 for ssh then you can specify a port for ssh by clicking the **Specify port** check box and then typing a new port number into the adjacent text field.

One common case of when you might want to use a different port is if you are running VisIt on your home computer and connecting to an LLNL computer via Internet Port Allow (IPA), which allows your computer through the Lab firewall so it is possible to connect to some computers from offsite. In this type of connection, the port for ssh is often 822 so in order to successfully connect VisIt from your home desktop to an LLNL computer, you would specify port 822 in the **Advanced options** tab.

In addition to relying on remote computers' ssh port, VisIt listens on its own ports (5600-5605) while launching components. If your desktop computer is running a firewall that blocks ports 5600-5605 then any remote components that you launch will be unable to connect back to the viewer running on your local computer. If you are not able to successfully launch VisIt components on remote computers, be sure that you make sure your firewall does not block VisIt's ports. Windows XP's and MacOS X's default software firewall configurations block VisIt's ports so if you run those software firewall programs, you will have to unblock VisIt's ports if you want to run VisIt in distributed mode.

3.4 Engine Option Window

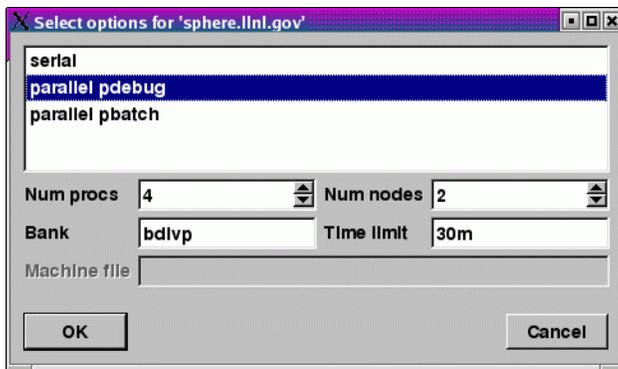


Figure 13-7: Engine Option Window

You can use **Engine Option Window**, shown in Figure 13-7, to pick a host profile to use when there are multiple host profiles for a computer or if there are any parallel host profiles. When there is a single serial host profile or no host profiles, the window is not activated when VisIt launches a compute engine. The window's primary purpose is to select a host profile and set some parallel options such as the number of processors. This window is provided

as a convenience so host profiles do not have to be modified each time you want to launch a parallel engine is run with a different number of processors.

The **Engine Option Window** has a list of host profiles from which to choose. The active profile for the host is selected by default though the another can be profile used instead. Once a host profile is selected, the parallel options such as the number of processors/nodes, processor count, can be changed to fine-tune how the compute engine is launched. After making any changes, click the window's **OK** button to launch the compute engine. Clicking the **Cancel** button prevents the compute engine from being launched.

3.4.1 Setting the number of processors

The number of processors determines how many processors are used by VisIt's compute engine. Generally, a higher number of processors yields higher performance but it depends on the host platform and the database being visualized. The **Num procs** text field initially

contains the number of processors used in the active host profile but you can change it by typing a new number of processors. The number of processors can also be incremented or decremented by clicking the up/down buttons next to the text field.

3.4.2 Setting batch queue options

Some compute environments such as large IBM computers schedule parallel jobs in batch queues. The **Engine option window** provides a few controls that are useful for batch queued systems. The first option is the number of nodes which determines the number of smaller portions of the computer that are allocated to a particular task. Typically the number of processors is evenly divisible by the number of nodes but the window allows you to specify the number of nodes such that not all processors within a node need be active. You can set the number of nodes, by typing a new number into the **Num nodes** text field or you can increment or decrement the number by clicking on the arrow buttons to the right of the text field. The second option is the bank which is a large collection of nodes from which nodes can be allocated. To change the bank, you can type a new bank name into the **Bank** text field. The final option that the window allows to be changed is the time limit. The time limit is an important piece of information to set because it can help to determine when the compute engine is scheduled to run. A smaller time limit can increase the likelihood that a task will be scheduled to run sooner than a longer running task. You can change the time limit by typing a new number of minutes into the **Time limit** text field.

3.4.3 Setting the machine file

Some compute environments use machine files, text files that contain the names of the nodes to use for executing a parallel job, when running a parallel job. If you are running VisIt in such an environment, the **Engine option window** provides a text field called **Machine file**. The **Machine file** text field allows you to enter the name of a new machine file if you want to override which machine file is used for the selected host profile. The **Machine file** text field is only enabled when the **Default Machine File** check box is enabled in the **Host profile Window's** parallel options.

4.0 Managing compute engines

VisIt can have many compute engines running at the same time. Much of the time the compute engines are those that are installed with VisIt but on occasion, simulation codes may be instrumented to act as VisIt compute engines capable of performing visualization operations on simulation data as it is created. When a simulation is used as a VisIt compute engine, VisIt can access data directly from the simulation without the need to translate data into another format or write it out to disk. When simulations are instrumented to become VisIt compute engines, they have all of the abilities of the standard VisIt compute engine and more. Specifically, simulations can accept additional simulation-defined control commands in order to make them perform actions such as writing a restart file. Since simulations offer extra capabilities over a normal VisIt compute

engine, VisIt provides different windows in order to manage them. In order to manage compute engines and check on progress, VisIt provides a **Compute Engine Window**. VisIt provides the **Simulation Window** in order to manage simulations, display their progress, and provide extra controls for the simulations.

4.1 Compute Engine Window

You can open the **Compute Engine Window**, shown in Figure 13-8, by selecting the **Compute engines** option from the **Main Window's File** menu. The main purpose of the **Compute Engine Window** is to display the progress of a compute engine as it completes a task. The window has two status bars. The top status bar indicates how much progress has been made in the overall task. The bottom status bar indicates that compute engine's progress through the current processing stage. The window also provides buttons for interrupting and closing compute engines, as well as an **Engine Information area** that indicates how many processors the engine uses and its style of load balancing.

4.1.1 Picking a compute engine

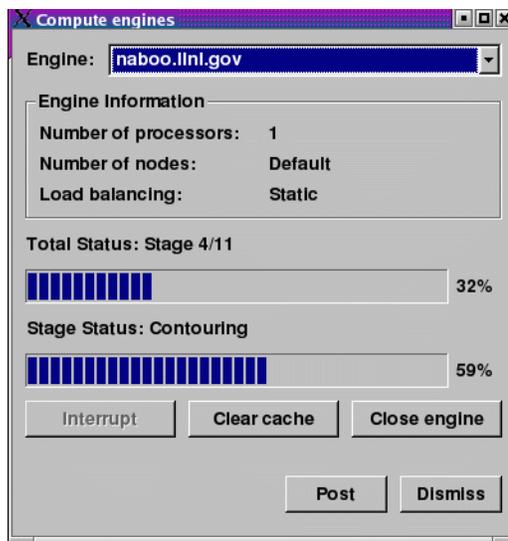


Figure 13-8: Compute Engine

mistake, you can interrupt a compute engine so it does not finish its work. When you click the **Interrupt engine** button a signal is sent to the compute engine that tells it to stop its work. The compute engine handles the interruption after it completes the current stage so there can be a small delay before the compute engine is interrupted. Any plots being generated when a compute engine is interrupted are sent into the error state and are listed in red in the **Plot list** until they are regenerated.

The **Compute Engine Window** has the concept of an active compute engine. Only the active compute engine's progress is displayed in the status bars. The active compute engine is also the engine that is interrupted or closed. To pick a new active compute engine, choose a compute engine name from the **Engine** menu. The **Engine** menu contains the names of all compute engines that VisIt is running.

4.1.2 Interrupting a compute engine

Some operations in VisIt may take a long time to complete so most computations are broken down into stages. In the event that you do not want to wait for an operation to complete, or if you realize that you made a

4.1.3 Closing a compute engine

You can close a compute engine when you no longer need it by clicking the **Close engine** button. The compute engine is closed only after you click **Yes** in a confirmation dialog window.

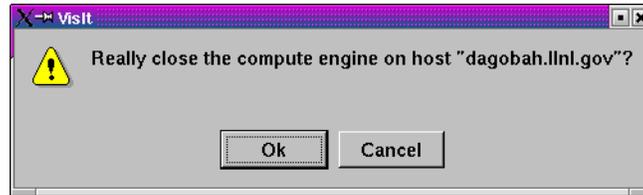


Figure 13-9: Close compute engine confirmation dialog

4.1.4 Clearing the compute engine's caches

As the compute engine processes data, it caches calculation results in case they are needed again for additional calculates. This includes meshes and variables that have been read from the database as well as the results from more complicated calculations involving expressions and operators. VisIt's compute engine periodically clears the cache of items that it no longer needs but if you want to explicitly clear the cache to free up more memory for your computer's other applications, you can click the **Clear cache** button in the **Compute Engine Window**.

4.2 Simulation Window

You can open the **Simulation Window**, shown in Figure 13-10, by selecting the **Simulations** option from the **Main Window's File** menu. The main purpose of the **Simulation Window** is to display the progress of a simulation that is acting as a VisIt compute engine as it completes its visualization tasks. The **Simulation Window** also provides buttons that cause the simulation to perform simulation-defined commands such as saving out a restart dump. The list of commands depends on the functionality that the simulation wanted to expose to VisIt when it was instrumented.

The **Simulation Window** is divided up into two main areas. The top of the window, called the **Simulation attributes** area, displays various attributes of the simulation such as its name, when it was started, the name of the computer where it is running, the number of processors, etc. Below the **Simulation attributes** area, you will find controls that are also present in the **Compute Engines Window** such as the **Interrupt** button and **Clear cache** button. The **Disconnect** button is specific to the **Simulation Window** and when you click it, VisIt will detach from the running simulation, leaving it to continue its calculation. You can later reconnect to the simulation if you want to check up on the its progress.



Figure 13-10: Simulation Window

The **Commands** area is located below the **Simulation attributes** area in the **Simulation Window**. The **Commands** area displays buttons for simulation-defined commands. When a simulation is instrumented to act as a VisIt compute engine, it publishes a list of commands that it can accept later. This allows the simulation user to provide hooks that allow the user to tell the simulation to execute certain commands like writing a restart file. Depending on the complexity of the commands that are exposed, VisIt could ultimately be used to steer the simulation as well as visualize its results.

1. Overview

In this chapter, we will discuss how to set and save user preferences. User preferences affect the default values for plots and operators as well as window properties like background color. This chapter reveals where those settings are saved and how to modify them.

2. How VisIt uses preferences

VisIt's preferences are saved into two levels of XML files that are stored in the user's home directory and in the global VisIt installation directory. The global preferences are read first and they allow the system administrator to set global preferences for all users. After VisIt reads the global preferences, it reads the preferences file for the current user. These settings include things like the color of the GUI and the initial directory from which to read files. Most of the attributes that are settable in VisIt can be saved to the preferences files for future VisIt sessions.

3. Setting default values



Figure 14-1: Make default button

Some windows have a button called **Make default** that sets the current attributes for the window as the values that will be saved to the preferences file. You must click the **Make default** button for windows that have it if you want the current settings for

the window to be saved to the preferences file. An example of a **Make default** button is shown in Figure 14-1

4. How to save your settings

To save preferences in VisIt, you need only to select **Save settings** from the **Main Window's Options** menu. When you choose this option, VisIt saves the current settings to your preferences file so you can use them the next time you run VisIt. VisIt does not automatically save your settings when you make changes to default attributes for plots, operators, or various controls windows. If a VisIt window provides a **Make default** button, the settings for that window are divided into current settings and default settings. You must click the **Make default** button to ensure that VisIt will copy the current settings over the default settings. Once you have set your default settings appropriately, save your settings.

If you want to save more than just the default attributes such as the plots in your plot list and where your windows are located, you should save a session file in addition to saving your settings. To save a session file, select the **Save session** option from the **Main Window's File** menu, and select a session file name. Once you select a name for your session file, VisIt saves out an XML description of your VisIt session that includes all of the information required to reproduce it.

VisIt saves two preference files, the first of which stores preferences for VisIt's GUI while the second file stores preferences for VisIt's viewer. When running VisIt on UNIXTM MacOS X systems, the preference files are called: "guiconfig" and "config" and they are saved in the ".visit" directory in your home directory. The Windows version of VisIt calls the preference files: "guiconfig for user" and "config for user" where user is your username and they are stored in the directory where the VisIt executable programs were installed.

If you prefer to run VisIt without reading your saved settings, you can provide the *-noconfig* option on the command line when you run VisIt. The *-noconfig* argument is often useful when you run the UNIXTM version and you get an updated version of VisIt that is incompatible with your saved settings. VisIt settings are usually compatible between different versions but this is not always the case and some users have had trouble on occasion when transitioning to a newer version. If you find that VisIt has stability problems when it starts up, provide the *-noconfig* option and then save your settings to write over any older preference files.

5. Appearance Window

You can open the **Appearance Window**, shown in Figure 14-2, by selecting **Appearance** from the **Main Window's Options** menu. The **Appearance Window** is responsible for setting preferences for the appearance of the GUI windows. You can use it to set your preferred GUI colors as well as other attributes such as orientation and style.

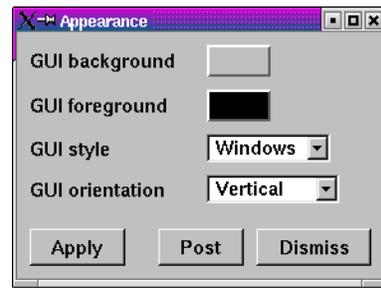


Figure 14-2: Appearance Window

5.1 Changing GUI colors

To change GUI colors using the **Appearance Window**, you need to click on the color button next to the color that you want to change. To change the background color (the color of the GUI's windows), click on the **GUI background** color button and select a new color from the **Popup color** menu. To change the foreground color (the color used to draw text), click the **GUI foreground** color button and select a new color from the **Popup color** menu.

VisIt will issue an error message if the colors that you choose for both the background and foreground colors are close enough that they cannot be distinguished so you do not accidentally get into a situation where the controls in VisIt's GUI become too difficult to read. Some application styles, such as Aqua, do not use the background color so setting the background has no effect unless you change to an application style like Motif, which does use the background color.

5.2 Changing GUI Style

VisIt's GUI can adapt its look and feel, or application style, to the platform on which it is running or you can make it use the application style of another computer platform. When VisIt runs on a Windows computer, it looks like a Windows application. When VisIt runs on a computer running some variant of UNIXTM, it will often look like a Motif or SGI application. When VisIt runs on MacOS X, its controls are drawn in the same fancy Aqua style that is used by other MacOS X applications.

If you prefer VisIt to use a non-default application style, you can change the application style so VisIt more closely resembles applications that run on your favorite platform. To change application styles, select a new option from the **GUI style** menu in the **Appearance Window** and click the **Apply** button. This will cause all of VisIt's windows to redraw themselves to reflect the style that you chose. The supported styles are: Windows, CDE, Motif, SGI, Platinum, Aqua, and Macintosh. The Aqua and Macintosh styles are only supported on MacOS X. Figure 14-3 shows VisIt's **Main Window** in three common application styles.

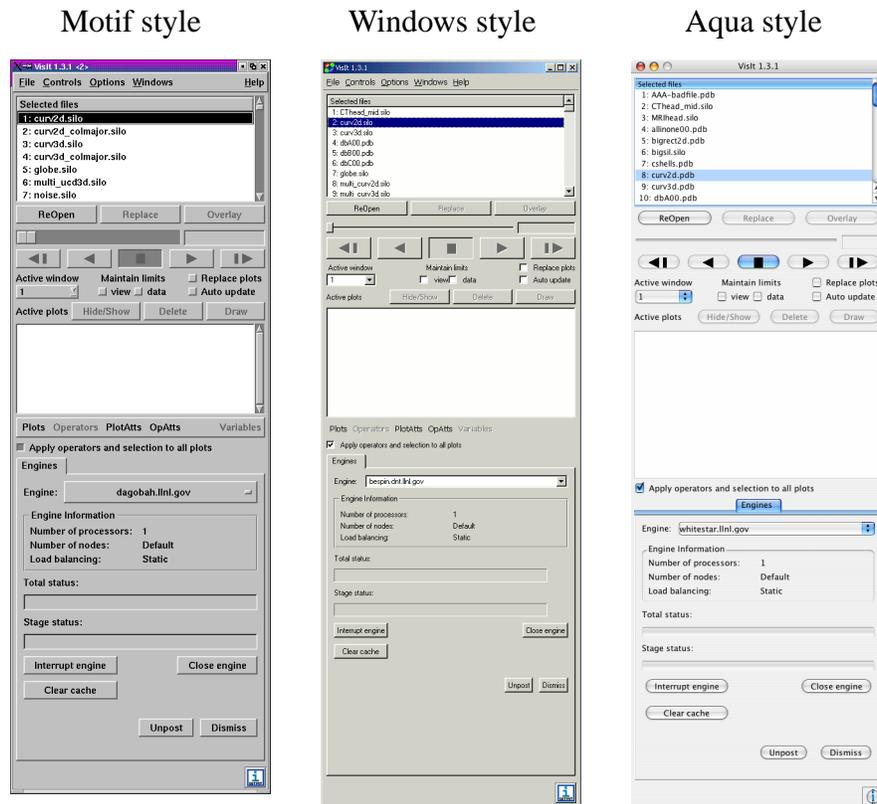


Figure 14-3: VisIt's Main Window in different styles

5.3 Changing GUI Orientation

By default, VisIt's **Main Window** appears as a vertical window to the left of the visualization windows. The default configuration often makes the best use of the display when its resolution is high enough. For displays that have less than 1280x1024 pixels, you might consider switching VisIt's window layout to a horizontal layout, which makes slightly better use of the display resolution.

When VisIt's window layout has been set to a horizontal layout, the **Main Window** is transformed into a short, wide window instead of being tall and thin. The **Main Window's** position also changes relative to the visualization windows. Instead of being located to the left of the visualization windows, the **Main Window** is instead moved so it is on top of the visualization windows, which are resized to fit the new configuration. To change VisIt's GUI orientation, select either **Vertical** or **Horizontal** from the **Appearance Window's GUI orientation** menu and click the **Apply** button. This will cause VisIt's windows to change to the appropriate configuration. Figure 14-4 shows VisIt's vertical layout and Figure 14-5 shows VisIt's horizontal layout.

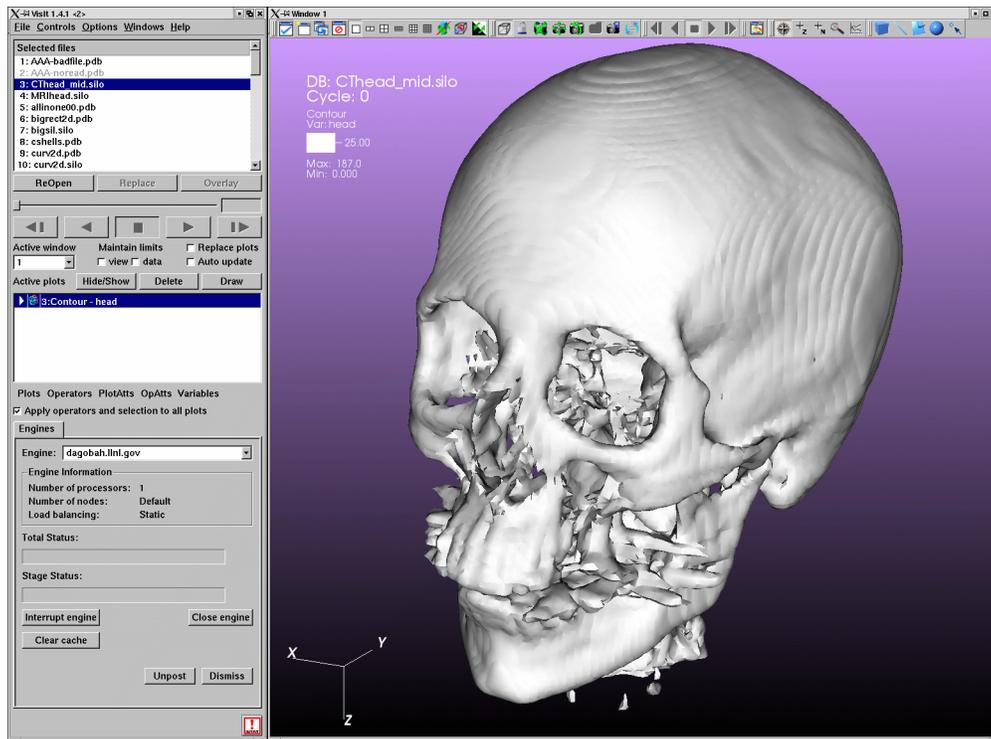


Figure 14-4: VisIt's vertical orientation.

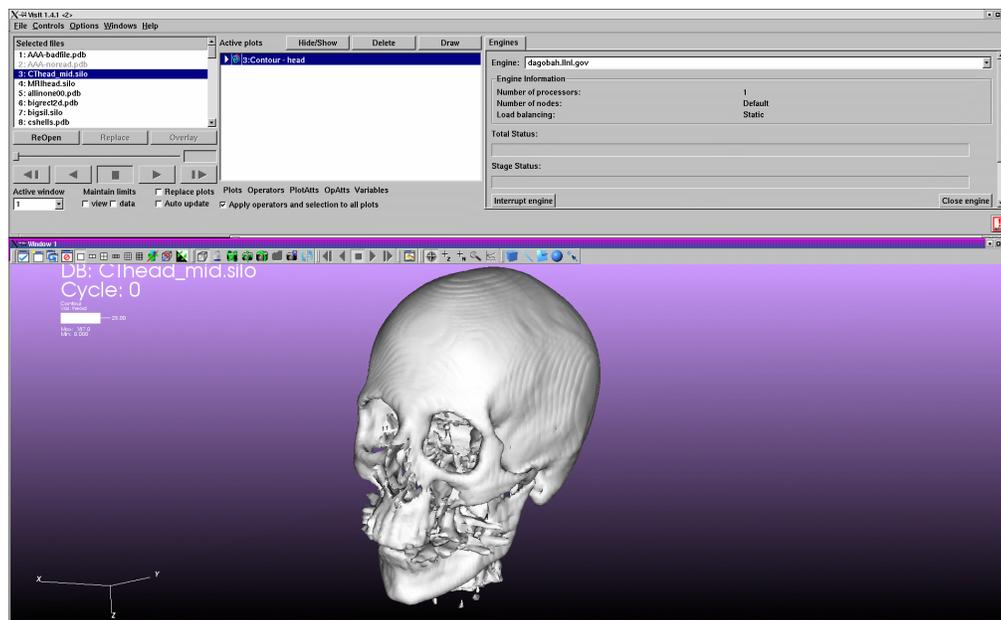


Figure 14-5: VisIt's horizontal orientation.

6. Plugin Manager Window

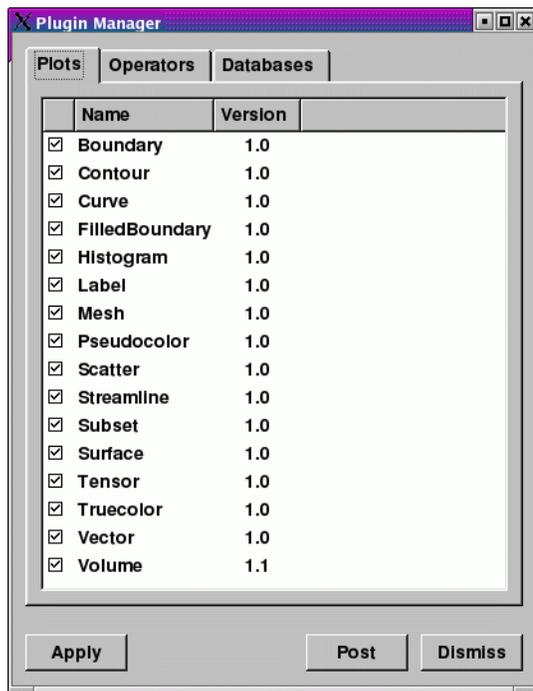


Figure 14-6: Plugin Manager Window.

mechanism that allows you to turn plugins on and off. The window has three tabs: **Plots**, **Operators**, and **Databases**. Each tab displays a list of plugins that can be loaded by VisIt. If a plugin is enabled, it has a check by its name.

You can turn plugins on and off by checking or unchecking the check box next to a plugin's name. If you save your preferences after having turned some plugins off, VisIt will not load those plugins when you next run VisIt. If you instead turned some plugins on, those plugins will be loaded when you next run VisIt.

If you have used the **Plugin Manager Window** to disable certain plot and operator plugins, they will not appear in the **Plot** and **Operator** menus. You can enable the plot and operator plugins again the controls in the **Plugin Manager Window**.

6.2 Preferences Window

The **Preferences Window**, shown in Figure 14-7, contains controls that allow you to set global options that influence VisIt's behavior. Most notably, the **Preferences Window** allows you to set various properties that affect how the **File panel** in the **Main Window** displays files. For example, you can tell the **File panel** not to show the **Selected files**

The **Plugin Manager Window**, shown in Figure 14-6, allows you to see which plugins are available for plots, operators, and databases. Not all plugins have to be loaded, in fact, many operator plugins are not loaded by default. The **Plugin Manager Window** allows you to specify which plugins are loaded when VisIt is started. The **Plugin Manager Window** is activated from the **Main Window's Options** menu after clicking on the **Plugin Manager** option.

6.1 Enabling and Disabling Plugins

All of VisIt's plots, operators, and database readers are implemented as plugins that are loaded when VisIt first starts up. Some plugins are not likely to be used by most people so they should not be loaded. The **Plugin Manager Window** provides a

list and instead you can rely on other controls for setting the active time state for databases or for selecting the active database.

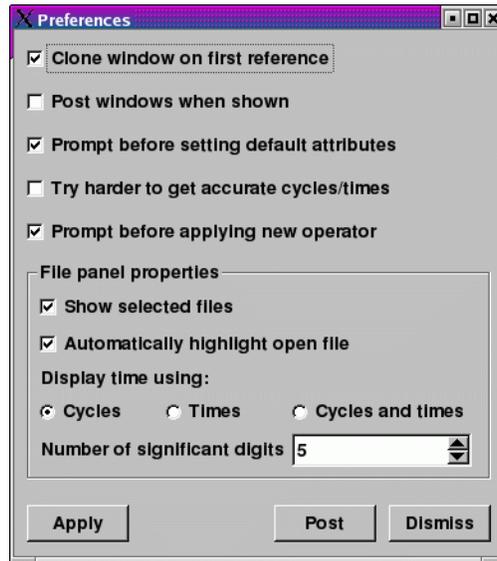


Figure 14-7: Preferences Window

6.2.1 Copying plots on first reference

VisIt's predecessor, MeshTV, copies plots to a new vis window when you set the active window if the new active window has never been accessed. Several VisIt users expressed a desire to have this functionality in VisIt so VisIt now has a **Clone windows on first reference** option in the **Preferences Window** that clones all attributes of the active window to a new window when you set the active window to a window that has never been referenced. If you don't want plots, etc to be copied to a vis window that you have not yet accessed when you make it the active vis window, be sure to turn off the **Clone windows on first reference** check box in the **Preferences window** before setting the active window.

6.2.2 Posting windows by default

When you use one of VisIt's menus to activate another VisIt postable window, such as a plot attributes window, the window manager is free to show the window wherever it likes. When you are using VisIt on a large display where the windows might pop up very far away from VisIt's **Main Window**, it is sometimes convenient to make sure that windows that can be posted to the **Notepad** area are initially posted to the **Notepad** area instead of popping up wherever the window manager puts them. To make windows post to the **Notepad** area by default when they are shown, click the **Post windows when shown** check box.

6.2.3 Showing selected files

The **File panel** is no longer required to switch between open databases so you can now turn off the **Selected files** list and its associated **Open**, **ReOpen**, **Activate**, **Replace**, and **Overlay** buttons to free up more space in the **Main Window** for other controls such as the **Plot list** or **Notepad** area. If you want to hide the **Selected files** list, open the **Preferences Window** and click off the **Show selected files** check box. When the **Selected files** list is hidden, an additional **Sources** combo box appears just above the **Plot list**. If you want to switch between different databases when the **Selected files** list is hidden, you can choose a new active database, also known as an active source, from the **Source** combo box. Selecting a new active source from the **Source** combo box is the same as highlighting a file in the **Selected files** list and clicking its **Activate** button. If you save your settings when the **Selected files** list is hidden, your future VisIt sessions will also not show the **Selected files** list. The **Main Window** is shown before and after hiding the **Selected files** list in Figure 14-8.

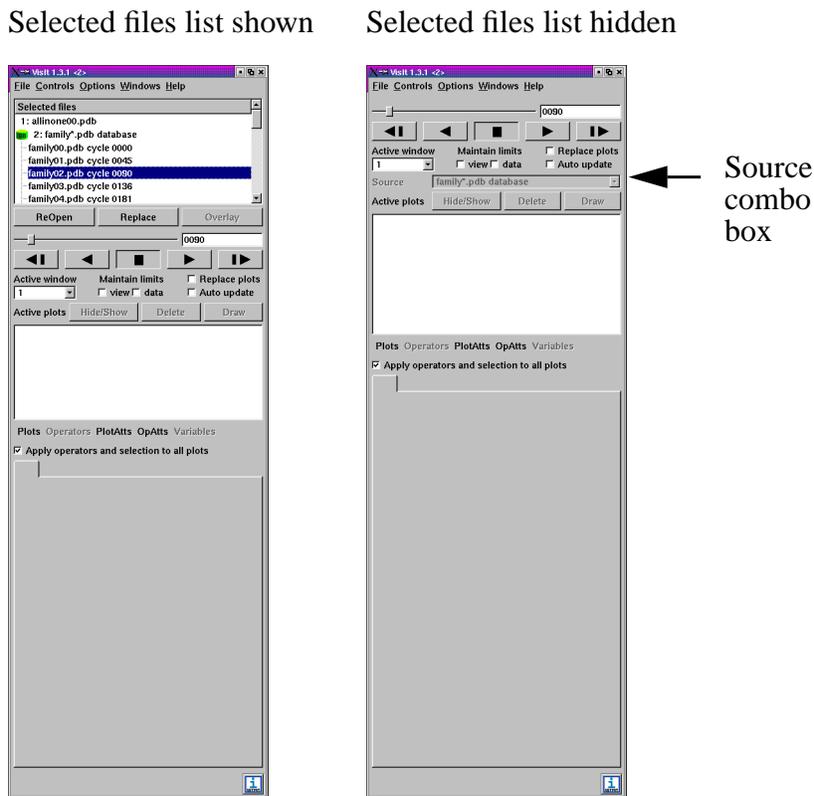


Figure 14-8: Selected files list in Main Window

6.2.4 File panel time display

The databases shown in the **File panel's Selected files** list show database cycles to indicate when databases are comprised of multiple time states. Cycles are shown by default because they can often be guessed from numbers embedded in the files that make

up each time state in the database. When you open a time-varying database that was written in a file format that natively understands multiple time states, VisIt can show cycles, times, or both in the time state's name.

If you want cycles to be shown in a time state's name, click the **Cycles** radio button. If you want times to be shown in a time state's name, click the **Times** radio button. If VisIt cannot successfully retrieve the times for each time state from your database then many of the times will have a question mark instead of a valid time. If you want to display both cycles and times in a time state's name, click the **Cycles and times** radio button. If you choose to include times in the name of a time state, you can set the number of significant digits to be shown in the time state's label. Enter a new number of significant digits into the **Number of significant digits** spin box if you want to increase or decrease the precision used in displaying the times.

Once you've chosen cycles, times, or both to be part of the information shown for each time state in a time-varying database, you can enter the selected type of time information into the **Cycle/Time** text field in order to change time states to the time state that closest matches the time state that you entered. That means that if you chose to display time in the name of a database time state, you can enter time into the **Cycle/Time** text field in order to change time states. Figure 14-9 shows what the **Selected files** list looks like when you've chosen different **File panel** time display options.

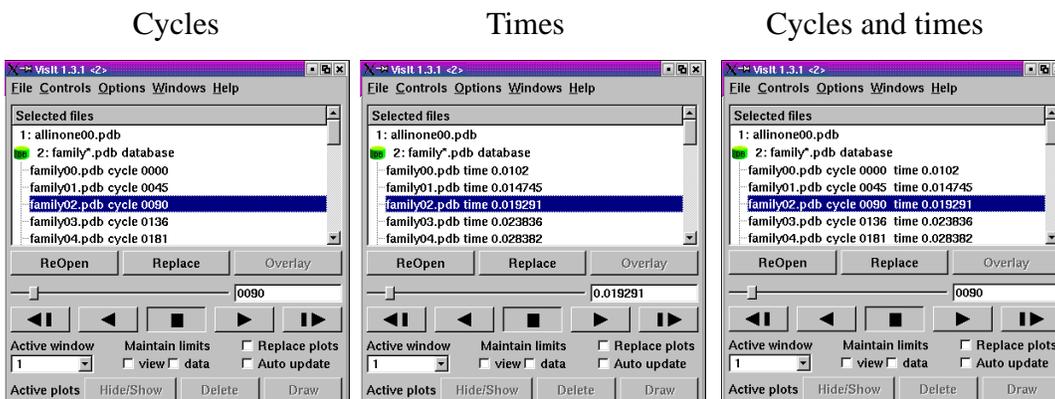


Figure 14-9: File panel display of cycles and times

6.2.5 Reading accurate cycles and times from databases

Many of the file formats that VisIt reads contain a single time state, making accurate cycles and times unavailable in VisIt's metadata for all but the open time state. Files formats afflicted with this problem (like Silo) will display zero or "?" times next to the files in the **Selected files** list when you have told VisIt to display times next to the file names. To get accurate times and cycles for these types of files, VisIt would have to open each file in the database, which can be a costly operation. VisIt does not go to this extra effort unless you turn on the **Try harder to get accurate cycles/times** option. This

option allows VisIt to display the cycles and times in the **Selected files** list and it also allows VisIt to create meaningful cycle or time-based database correlations for groups of single time state databases. Note that if you enable the **Try harder to get accurate cycles/times** option after you've opened a database, the cycles and times will not be retrieved for the open database unless you reopen it.

7. Rendering Options Window

The **Rendering Options Window** (shown in Figure 14-10) contains controls that allow you to set global options that affect how the plots in the active vis window are drawn and you can also look at information related to the performance of your graphics hardware. You can open the **Rendering Options Window** by selecting the **Rendering** option in the **Main Window's Preferences** menu. The window is divided vertically into two main areas. The top area, or **Options area**, contains controls that allow you to set rendering options which affect how your plots look in the vis window. Some topics related to how plots are drawn such as antialiasing, specular lighting, and shadows are covered beginning on page 250 in the **Making it Pretty** chapter. The bottom area, or **Information area**, displays information about how fast VisIt is able to draw graphics on your computer, as well as the size of the plots being drawn in terms of triangle count and spatial dimension.

7.1 Changing surface representations

Sometimes when visualizing large or complex databases, drawing plots with all of their shaded surfaces can take too long to be interactive, even for fast graphics hardware. To combat this problem, VisIt provides an option that allows you to view all of the plots in the vis window as wireframe outlines or point clouds instead of as shaded surfaces (see Figure 14-11). While being less visually informative, plots drawn as wireframe outlines or as clouds of points can still be useful for visualizations since you can do setup work like setting the view before switching back to a surface representation that is more costly to

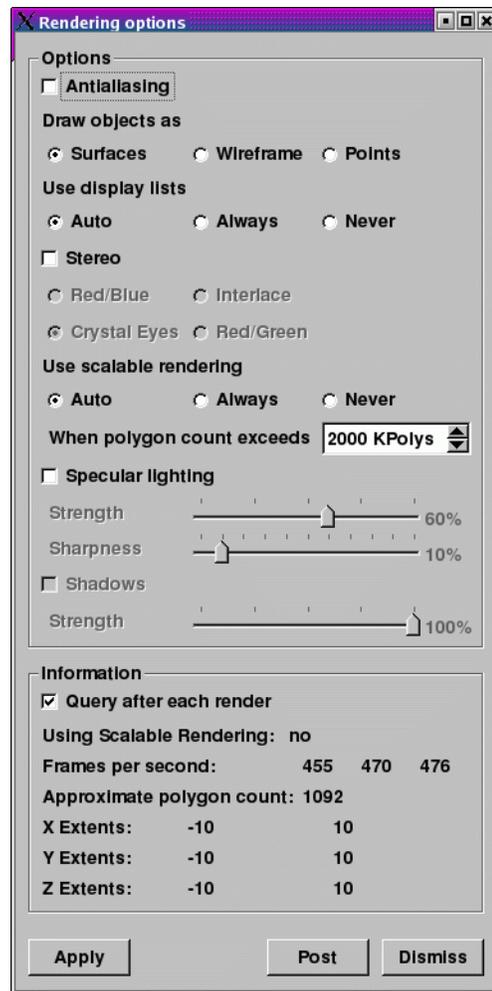


Figure 14-10: Rendering Options Window

draw. You click the **Surfaces**, **Wireframe**, or **Points** radio buttons to change the surface representation used to draw plots.

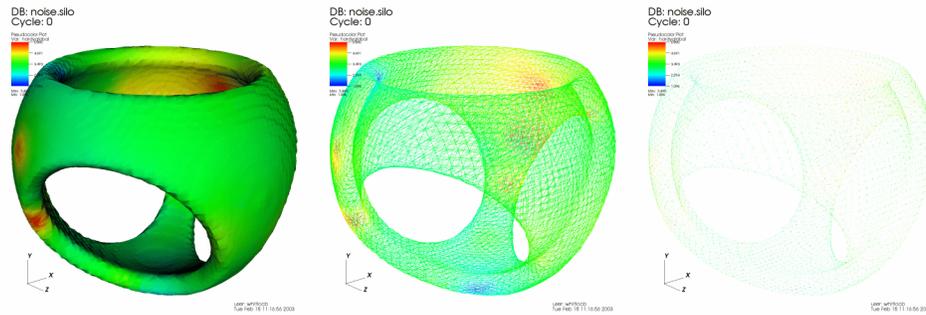


Figure 14-11: Surface representations

7.2 Using display lists

VisIt benefits from the use of hardware accelerated graphics and one of the concepts central to hardware accelerated graphics is the display list. A display list is a sequence of simple graphics commands that are stored in a computer's graphics hardware so the hardware can draw the object described by the display list several times more quickly than it could if the graphics commands were issued directly. VisIt tries to make maximum use of display lists when necessary so it can draw plots as fast as possible.

VisIt decides when to use display lists and when to not use display lists. Typically when you are running VisIt on your local workstation with plots that result in fewer than a couple million graphics primitives, VisIt does not use display lists because the cost of creating them is more expensive than just drawing the graphics primitives without display lists. When you run a UNIX™ version of VisIt on a remote computer and display the results back to your workstation using an X-server, it is almost always advantageous to create display lists for plot geometry. Without display lists, VisIt must keep transmitting the plot geometry over the network to the X-server which then uses OpenGL to draw the geometry. By default, VisIt automatically decides when to use display lists and when not to use them but you can force VisIt to either use or not use display lists. If you don't want VisIt to ever use display lists, click the **Never** radio button under the **Use display lists** options in the **Rendering Options Window**. If you want VisIt to always use display lists, click the **Always** radio button under the **Use display lists** options.

7.3 Stereo images

Stereo images, which are composites of left and right eye images, can convey additional depth information that cannot be expressed by images that are generated using a single eye point. VisIt provides four forms of stereo images: red/blue, red/green, interlace, and crystal eyes. A red/blue stereo image (see Figure 14-12) is similar to frames from early 3D movies in that it appears stereo only when using red/blue stereo glasses. Unfortunately, red/blue stereo images are not very useful for visualization because colors are lost since

most of the color ends up in the magenta range when the red and blue color channels are merged. Red/green stereo suffers similar color loss. Interlaced images alternate lines in the image with left and right eye views so that squinting makes the image look somewhat 3D. VisIt's crystal eyes option requires the use of special virtual reality goggles for images to appear to be 3D but this option is by far the best since it allows interactive frame rates with images that really appear to stand out from the computer monitor. VisIt does not use stereo imaging by default since it makes images draw slower because an image must be drawn for both the left eye and the right eye. If you want to view stereo images using the **Crystal Eyes** option, you must provide the `-stereo` command line option when you launch VisIt or clicking the **Crystal Eyes** radio button will have no effect because vis windows must be created in stereo mode. If you want to try looking at stereo images, click the **Stereo** check box and then also click one of the **Red/Blue**, **Red/Green**, **Interlace**, or **Crystal Eyes** radio buttons to choose the stereo image type.

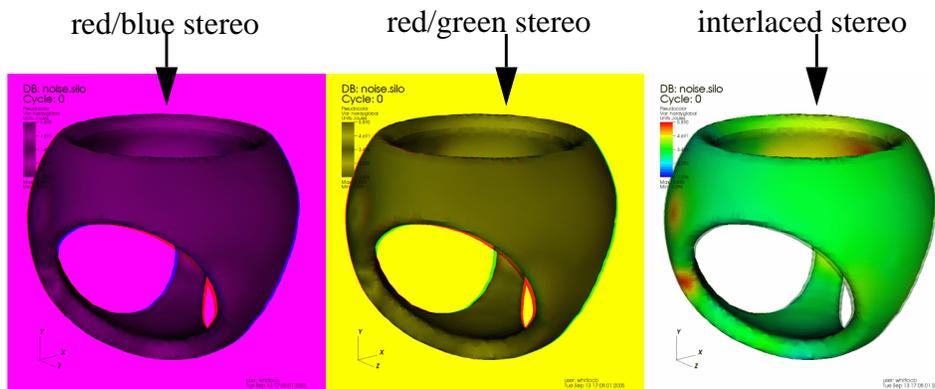


Figure 14-12: Some stereo image types

7.4 Scalable rendering

VisIt typically uses graphics hardware on the local computer to very quickly draw plots once they have been generated by the compute engine. This becomes impractical for very large databases because the amount of memory needed to store the graphics commands that draw the plots quickly exceeds the amount of memory in the graphics hardware. Large sets of graphics commands can also degrade performance when they must be shipped over slow networks from the compute engine to the VisIt's viewer. VisIt provides a scalable rendering option that can improve both of these situations by creating the actual plot images, in parallel, on the compute engine, compressing them, and then transmitting only an image to the viewer where the image can be displayed.

Scalable rendering can be orders of magnitude faster for large databases than VisIt's conventional image drawing strategy because large databases are typically processed using a parallel compute engine. When using scalable rendering with a parallel compute engine, VisIt is able to draw small pieces of the plot on each processor in parallel and then glue the image together before sending it to the viewer to be displayed. Not only has the image likely been created faster, but the size of the image is usually on the order of a megabyte instead of the tens or hundreds of megabytes needed to transmit graphics

commands, which results in faster transmission of the image to the viewer. The drawback of scalable rendering is that it is usually not as interactive as graphics hardware because each time you want to change the view or interact with the plots, round trip communication with the compute engine is required.

VisIt can automatically determine when to stop sending geometry to the viewer in favor of sending scalably rendered images. You can help VisIt decide when to use scalable rendering by either setting the scalable rendering threshold, which is the number of polygons that must exist in the set of geometry to be rendered on the viewer before the compute engine will kick into scalable rendering mode. You can set the scalable rendering threshold by entering a new number of polygons into the When polygon count exceeds spin box in the **Rendering Options Window**. The number that you enter is measured in thousands of polygons.

If you want VisIt to always use scalable rendering, you can click the **Always** radio button under the **Use scalable rendering** options. If you don't want VisIt to ever use scalable rendering no matter how many polygons are in the set of geometry that your workstation will have to draw, click the **Never** check box under the **Use scalable rendering** options. Note that if you disable scalable rendering and then you decide to plot a database with millions or hundreds of millions of cells, you risk running out of memory on your workstation or having to wait a very long time for the image to appear. In general, it is much faster to draw images of large databases with scalable rendering.

7.5 Frames per second

Frames per second refers to the number of times that VisIt can draw the plots in the vis window in the course of a second. VisIt displays the minimum, average, and maximum frame rates achieved during the last draw operation, like rotating the image with the mouse, in the **Renderer Options Window's Information area**. Some actions that force a redraw do not cause the information to update. An example of this is resizing the vis window. To make VisIt update the frame rate information after each time it draws the plots in the visualization window, check the **Query after each render** check box.

7.6 Triangle count

Triangle count refers to the number of triangles used to represent the plots in the vis window. VisIt displays the triangle count in the **Renderer Options Window's Information area**.

7.7 Plot Extents

The plot extents are the minimum and maximum locations of the plot in each spatial dimension. The plot extents are the smallest bounding box that can contain the plots in the

vis window. VisIt displays the plot extents for each dimension in the **Renderer Options Window's Information area**.

1.0 Overview

In this chapter, we will discuss how to use VisIt's online help. VisIt's online help consists of release notes, copyright information, Frequently Asked Questions (FAQ), and the contents of this manual. The release notes help page lists the complete set of bug fixes and enhancements for the current version of VisIt with links to the release notes for older versions. The copyright information help page lists VisIt's copyright agreement. The FAQ help page lists commonly asked questions and the answers to those questions. Beginning VisIt users should read through the FAQ help page to find the answers to commonly asked questions. Finally, the contents of this manual are available as online help.

2.0 About VisIt

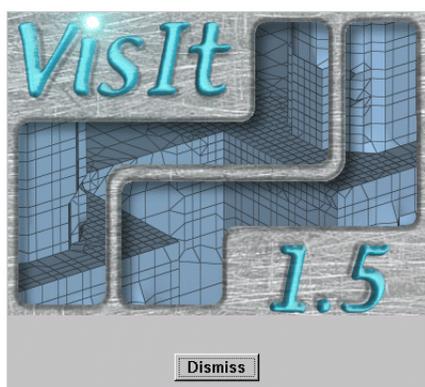


Figure 15-1: Splash screen

VisIt provides a **Splash screen** (Figure 15-1) that appears when the tool is launched. The **Splash screen** has three purposes: entertainment, displaying startup progress, and telling the user about VisIt. As VisIt launches, the **Splash screen** cycles through a handful of images that show some of VisIt's capabilities and it also tells the user what happens while VisIt is launching. Once VisIt is launched, you can look at some information about VisIt by selecting the **About** option from the **Main Window's Help** menu. Choosing that menu option displays the **Splash screen** which can be hidden by clicking its **Dismiss** button.

3.0 Help Window

VisIt's **Help Window**, shown in Figure 15-2, displays all of VisIt's online help content. You can open the **Help Window** by choosing the **Help** option from the **Main Window's Help** menu. The **Help Window** has a toolbar along the top of the window while the rest of the window is divided vertically into two main areas. The left side of the window is used to select online help pages and it is further divided with tabs for help contents, help index, and bookmarks. The right side of the window displays the content for the online help pages.

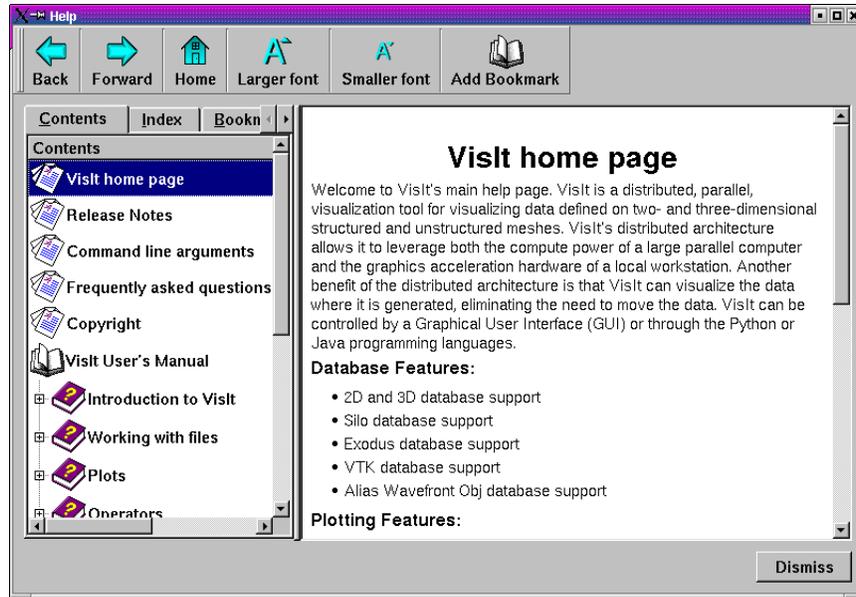


Figure 15-2: Help Window

3.1 Help Window Toolbar

The **Help Window's** toolbar exposes buttons for navigation, changing font size, and adding bookmarks. You can hide the toolbar by double-clicking on the handle located at the far left of the toolbar. The toolbar can also be moved to other parts of the **Help Window** by clicking on its handle and dragging it to the top, sides, or the bottom of the **Help Window**.

3.1.1 Navigation

The toolbar contains buttons that you can use to cycle forward and backward in the list of visited help pages. The **Back** button has an arrow icon that points to the left and the button changes the active help page to the last visited help page. The **Forward** button has an arrow icon that points to the right and it switches the help page to the page that was active before the **Back** button was clicked. If have not visited any help pages, both of these buttons are disabled. The toolbar also contains a **Home** button which has a house icon. The **Home** button displays the VisIt home page, which describes VisIt's features.

3.1.2 Changing font size

The toolbar contains two buttons that allow you to change the font size used to display online help. The **Larger font** button is distinguished by a large capital ‘A’ and a small triangle which points up. When the **Larger font** button is clicked, the font size is increased and the active help page is redrawn with the larger font. The **Smaller font** button looks similar to the **Larger font** button except that its icon’s triangle points down and its ‘A’ is smaller. The **Smaller font** button decreases the font size and immediately redraws the active help page using the new smaller font.

3.1.3 Adding a bookmark

VisIt’s **Help Window** provides the ability to create and save personal bookmarks. This allows you to easily come back to frequently-used sections of the online help. The toolbar contains an **Add bookmark** button that adds the current help page to the list of bookmarks. The **Bookmarks** tab in the left part of the **Help Window** also has this feature.

3.2 Selecting a help page

The **Help Window** has three tabs, shown in Figure 15-3, that allow a help page to be located in different ways. The first tab is the **Contents** tab and it lists all of the online help pages and allows them to be grouped into related topics. The **Index** tab lists all of the online help pages in an alphabetized list that can be searched for a particular help topic. The **Bookmarks** tab shows all bookmarked help pages which can be recalled by clicking on a bookmark.

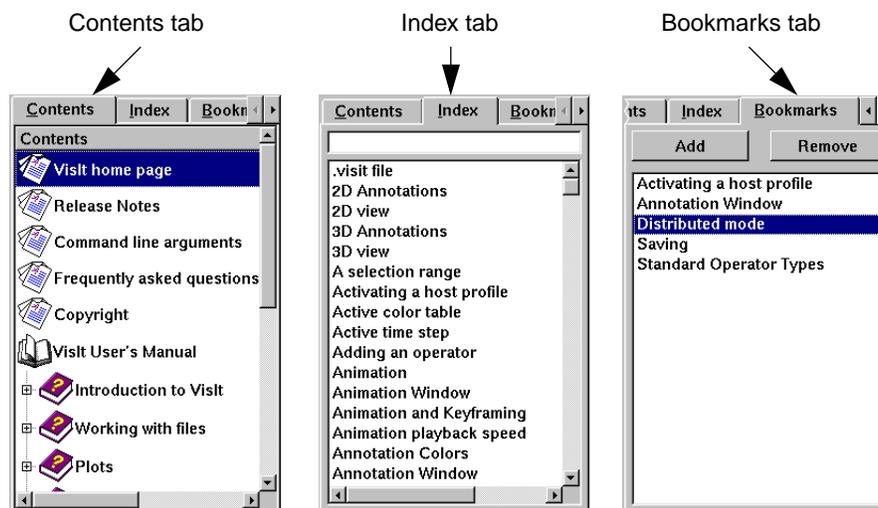


Figure 15-3: Help tabs

3.2.1 Contents tab

The **Contents** tab lists all of the online help pages and groups them into related topics which are sometimes organized in tree format. When items are organized into a tree, an entry in the list of help pages often has a book icon next to it indicating that the topic contains other help topics. When an item has a book icon, it can be opened by double-clicking on its title or by clicking the check box to the left of the title. Items that have an icon that looks like a stack of papers contain the actual help content and clicking on them displays the help page in the right half of the **Help Window**.

3.2.2 Index tab

The **Index** tab lists all of the help topics alphabetically in a single searchable list. Help topics can be selected by clicking on an item in the list or by typing a help topic into the text field above the list. As words are typed into the text field, the closest match is found in the list of help topics and the topic is displayed in the right half of the **Help Window**.

3.2.3 Bookmarks tab

The **Bookmarks** tab lists all of the help topics that have been bookmarked for further use. To view a page that has been previously bookmarked, simply click on its title in the bookmark list. To add a bookmark for the current help page, click the **Add** button in the **Bookmarks** tab or in the **Help Window's** toolbar. To remove a bookmark, click on its title in the bookmark list and then click the **Remove** button.

Appendix A **VisIt Command Line Options**

A.1 Command Line Options

The command line options are listed after the `visit` command when starting VisIt.

```
visit [options]
```

The options are listed below grouped by functionality and listed alphabetically within a group. The order in which these options are specified is unimportant; `visit -gui -beta` is the same as `visit -beta -gui`.

Arguments to visit script	Program options
<code>-cli</code>	Run with the Command Line Interface.
<code>-gui</code>	Run with the Graphical User Interface (default).
<code>-movie</code>	Run with the Command Line Interface in movie making mode where there is no window. You must also provide the <code>-sessionfile</code> or <code>-s</code> arguments when <code>-movie</code> is specified.

Additional programs	Program description
<code>convert</code>	Run the VisIt database conversion tool, which can read in a data file in one of VisIt's supported formats and write it back out in another supported VisIt database format that supports writing new files through VisIt's plugin interface. The <code>convert</code> tool is currently used primarily as a tool to convert non-Silo files into Silo files.
<code>makemili</code>	Run the <code>makemili</code> program that creates a <code>.mili</code> root file that allows VisIt to read Mili files.

Additional programs	Program description
<code>mpeg_encode</code>	Run the <code>mpeg_encode</code> software that is bundled with the UNIX versions of VisIt.
<code>silex</code>	Runs Silex, a graphical tool to browse Silo files.
<code>xmledit</code>	Runs XMLEdit, which is a graphical tool designed to aid VisIt plugin developers in designing the state objects, etc required for creating VisIt plot, operator, and database plugins.
<code>xml2plugin</code>	Runs VisIt's XML plugin generator.
<code>xmltest</code>	Runs VisIt's XML plugin tester.
<code>xml2atts</code>	Runs VisIt's XML plugin attribute generator which produces C++ code from an XML description of a state object.
<code>xml2avt</code>	Runs VisIt's XML plugin code generator for AVT components.
<code>xml2info</code>	Runs VisIt's XML plugin code generator for plugin skeleton C++ code.
<code>xml2java</code>	Runs VisIt's XML plugin attribute generator which produces Java code from an XML description of a state object.
<code>xml2makefile</code>	Runs VisIt's XML plugin Makefile generator.
<code>xml2projectfile</code>	Runs VisIt's XML plugin MSVC++ 6.0 project file generator. This component is only distributed in the MS Windows version of VisIt.
<code>xml2python</code>	Runs VisIt's XML plugin attribute generator which produces Python code from an XML description of a state object.
<code>xml2window</code>	Runs VisIt's XML plugin GUI window generator for plot and operator attribute windows.

	Version options
<code>-beta</code>	Run the current beta version.
<code>-dir <directory></code>	Run the version in the given directory. For example, <code>-dir /usr/gapps/visit/1.3.2/sgi-irix6-mips2</code> would run the sgi 1.3.2 version of visit. The <code>-dir</code> option is usually specified in a Host profile so the <code>visit</code> command does not have to be in your path on a remote computer when you run in distributed mode.
<code>-v <version></code>	Run the specified version.

	Parallel options
<code>-b <bank></code>	Bank from which to draw resources. Only applies when using a launch program that operates in a batch environment.

	Parallel options
<code>-expedite</code>	Makes <code>psub</code> and other launch programs give priority to your job when scheduling your job to run. You must have expedite privileges for this option to take effect.
<code>-l <method></code>	Launch VisIt's compute engine in parallel using the given launch program. Accepted launch programs are: <code>bsub</code> , <code>dmpirun</code> , <code>mpirun</code> , <code>poe</code> , <code>prun</code> , <code>psub</code> , <code>srun</code> , <code>yod</code> . Be sure to use the launch program that is appropriate for the computer where you want to run VisIt in parallel.
<code>-la <args></code>	Provides additional command line arguments to the parallel launch program.
<code>-n <jobname></code>	Provides the name for the job.
<code>-nn <numnodes></code>	Specifies a number of nodes to allocate to the job.
<code>-np <numprocessors></code>	Specifies the number of processors to use from the allocated nodes.
<code>-p <partition></code>	Specifies the partition to run in.
<code>-par</code>	Selects the parallel version of VisIt's compute engine.
<code>-pl <method></code>	Similar to <code>-l</code> but only launches the parallel compute engine as specified.
<code>-t <timelimit></code>	Maximum job run time.

	Window options
<code>-background <color></code>	Background color for the graphical user interface. The color can consist of either a color name or an RGB triplet. For example, the color red could be specified as <code>-background red</code> or <code>-background #ff0000</code> .
<code>-foreground <color></code>	Foreground color for the graphical user interface. The color can consist of either a color name or an RGB triplet. For example, the color red could be specified as <code>-background red</code> or <code>-background #ff0000</code> .
<code>-geometry <spec></code>	The portion of the screen to use. This is a standard X Windows geometry specification. For example <code>500x500+300+0</code> , indicates an area 500 pixels by 500 pixels, 300 pixels to the right of the top left corner.
<code>-noconfig</code>	Run without reading any configuration files. This can be useful if you run VisIt and encounter unexpected behavior on startup.
<code>-nowin</code>	Run without any windows. This option may be useful when running scripts.
<code>-small</code>	Use a smaller desktop area/window size.
<code>-style <style></code>	The style to use for the graphical user interface. One of <code>windows</code> , <code>cde</code> , <code>motif</code> , or <code>sgi</code> .

	File options
<code>-o <databasename></code>	Run VisIt and have it open the specified database.
<code>-s <scriptname></code>	Run the specified VisIt script. Note that this option only takes effect with the <code>-cli</code> option.
<code>-default_format <format></code>	Tells VisIt to use the specified database reader plugin when reading files. This is a useful option if your data files do not have file extensions, since VisIt is able to open the file on the first try instead of having to sequentially try all of its database reader plugins until one of them can successfully open the file. To make VisIt use the Silo plugin first to open files, add: <code>-default_format Silo</code> to the command line.
<code>-sessionfile <filename></code>	Run VisIt and have it open the specified session file, which will cause VisIt to restore its state to what is stored in the session file. This argument is only valid with the <code>-gui</code> or <code>-movie</code> arguments.

	Debugging options
<code>-debug <level></code>	Run with <code><level></code> levels of output logging. <code><level></code> must be between 1 and 5. A debug level of 1 provides the least amount of output logging, while a debug level of 5 provides the most output logging.
<code>-timing</code>	Run with timings. Timings are provided for the execution of each major portion of the execution pipeline on the viewer and each engine process. Timing information for launch time is also provided for the gui and viewer processes.
<code>-dump</code>	This argument causes VisIt to write VTK files for each stage of the execution pipeline so you can see the output of each VTK filter.
<code>-verbose</code>	This argument causes VisIt's CLI to print out the stages of execution for its compute engine to the console.

Appendix B **Setting Up Password-less ssh**

The following instructions describe how to set up **ssh** to allow password-less authentication among a collection of machines. These instructions apply to **ssh1**; the process for **ssh2** is different. You should first read all the instructions before proceeding. Note that these instructions only apply to computers running variants of the UNIXTM operating system.

B.1 On the Local Machine

If you already have a `~/.ssh/identity.pub` file, you can skip these configuration steps. (Go to Section B.2)

```
cd
ssh-keygen
```

Accept default values by entering `<return>`. When asked for a passphrase enter a passphrase to gain a greater level of security. List the contents of the `.ssh` directory.

```
ls -l .ssh
```

The file `identity.pub` contains your public key in one very long line of text. The file `identity` contains your private key, which is non-readable data. The subdirectory `$HOME/.ssh` subdirectory must remain r-w-x permissions for the owner only. Check that this is the case.

```
ls -ld .ssh
```

B.2 On the Remote Machine

If you already have a `$HOME/.ssh/authorized_keys` file, append the contents of the local machine's `$HOME/.ssh/identity.pub` file to this file. If this isn't the case then create a `$HOME/.ssh` directory, if one does not exist, with r-w-x permission for the owner only.

```
cd
```

```
mkdir .ssh
chmod 700 .ssh
```

Check that the directory does not allow world or group access. SSH will not work if world or group access is allowed.

```
ls -ld ~
```

Copy the contents of the local machine's `$HOME/.ssh/identity.pub` file to the remote machine into the file `$HOME/.ssh/authorized_keys`. This provides the information for the remote machine to validate you. Remember that `identity.pub` contains a single long line. Either `ftp` the `identity.pub` file to the remote machine or edit it, being careful to avoid introducing carriage returns.

B.3 Completing the Process

You can then repeat the section “On the Remote Machine” for each remote machine for which you want to set up password-less ssh. You can repeat the above sections, reversing the local and remote machines, in order to allow password-less ssh to the local machine from the remote machine (i.e., copy the remote machine's `$HOME/.ssh/identity.pub` into the local machine's `$HOME/.ssh/authorized_keys` file).

Installing VisIt

C.1 Downloading VisIt

VisIt can be downloaded by directing your Web browser to <http://www.llnl.gov/visit> and clicking on the downloads link. This will take you to another Web page that lists the available binary distributions. VisIt runs on the following platforms:

- Linux (including RedHat, SUSE, CHAOS 2.0, Opteron x86_64, Altix)
- Microsoft Windows (98, 2000, ME, XP)
- MacOS X 10.3 and above
- AIX
- Solaris
- Tru64
- IRIX

Choose the distribution for the machine on which you want to install VisIt by clicking on the appropriate link. Doing so will download a VisIt binary distribution file to your computer. If you are installing VisIt on any platform except MS Windows, you must also download the *visit-install* installation script. This script installs the contents of the VisIt binary distribution and installs VisIt.

C.2 UNIX and MacOS X installation instructions

Installing VisIt on UNIXTM and MacOS X systems is done using the *visit-install* script. In order to run the *visit-install* script, make sure that you have an available shell window. On MacOS X, open the Terminal application to obtain a command line shell. Also make sure that the *visit-install* script is executable by entering the following command at the command line prompt: `chmod 750 visit-install`. The *visit-install* script has the following usage:

```
visit-install version platform directory
```

The **version** argument is the version of VisIt being installed. The **platform** argument depends on the platform where VisIt is being installed. The platform argument can be one of the following: aix, darwin, irix6, linux, linux-x86_64, osf1, or sunos5. The **directory** argument specifies the directory in which you want to install VisIt. If the specified directory does not exist then VisIt will create it.

To install an IRIX version of VisIt, the command entered at the shell prompt would look like the following:

```
visit-install 1.3.2 irix6 /usr/local/visit
```

This command will install the 1.3.2 version of VisIt in the /usr/local/visit directory. Note that when you enter the above command, the file visit1_3_2_irix6.tar.gz must be present in the current working directory.

Once you start running the visit-install script, it will prompt you to choose a network configuration. You can elect to choose no network configuration if you are installing VisIt at an institution that cannot access LLNL's computers. A network configuration is a VisIt preferences file that includes information that VisIt needs to identify and connect to remote computers in distributed mode and is included as a convenience for users of LLNL computers.

If you want to install VisIt for different platforms in the same directory, you can use the -a argument when running the *visit-install* script. For example, if you want to also install a MacOS X version of VisIt in the same visit directory as the IRIX distribution, you could enter the following command into the command shell:

```
visit-install -a 1.3.2 darwin /usr/local/visit
```

The final step in installing VisIt is adding it to your path. The best way to do this is to add it to your command line shell initialization file. If you are using C-Shell, the file to edit is called .cshrc and it exists in your home directory. It is important to add VisIt to your path because VisIt's bin directory contains the visit script that is used to run VisIt. The visit script can be run on many machines. The script determines the type machine it is running on and starts the appropriate visit component. This is crucial if you plan to run VisIt in distributed mode and do remote visualization. If VisIt was installed in /usr/local/visit, as in the above examples, the following commands will add VisIt to your search path:

```
cd  
echo "set path = ($path /usr/local/visit/bin)" >>.cshrc
```

You are now finished installing VisIt.

C.3 Microsoft Windows installation instructions

Installing VisIt on Microsoft Windows is a simple process. The binary distribution, unlike the UNIXTM distributions, is packaged as an installation program. This means that once

the binary distribution is downloaded, you can double-click on it to begin the installation wizard. Accepting the licensing agreement brings you to a screen that lets you pick the installation directory though the default installation directory should be adequate for most users. The suggested installation directory includes the version of VisIt that you are installing so you can have multiple versions of VisIt installed on the same computer.

After picking the installation directory, the wizard will prompt you for a network configuration. You can pick one of the packaged LLNL network configurations, which include information required to run VisIt in distributed mode to one of the computers at LLNL. You can also elect to have no network configuration, in which case VisIt will not know how to connect to certain computers right out of the box. Most external users will probably not want a network configuration so they can set up host profiles that make sense for their own computer network.

After picking the appropriate network configuration, the installer installs VisIt in the specified directory. Once VisIt is installed there are three ways you can launch it. The first way to launch VisIt is to select it from the VisIt program group in the Windows Start menu. You can also double-click the VisIt shortcut on the Windows desktop. Finally, you can open VisIt implicitly by double-clicking on a Silo file or VisIt session file.

C.4 Checking for new versions

A new version of VisIt is released every 4-8 weeks. You can check for new versions of VisIt on the VisIt Web site (<http://www.llnl.gov/visit>) or you can use the **Check for new version** menu option in the **Help** menu to check for new versions of VisIt. If you click on **Check for new version** then VisIt will contact the VisIt FTP site (<ftp://ftp.llnl.gov/pub/visit>) and determine whether there are any updates available. If there are updates available, VisIt will prompt you for whether or not you would like to upgrade VisIt (see Figure 15-4).

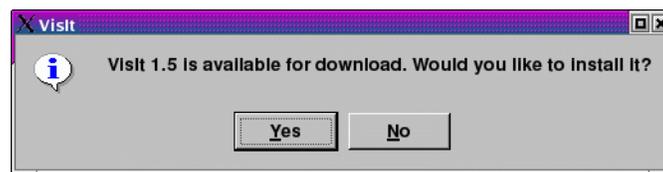


Figure 15-4: Installation prompt

If you choose to upgrade VisIt, VisIt will download the new platform-appropriate distribution file and proceed to install VisIt on your system. During download and

installation, feedback on the progress of the installation is displayed to the **Main Window's** status bar as shown in Figure 15-5.

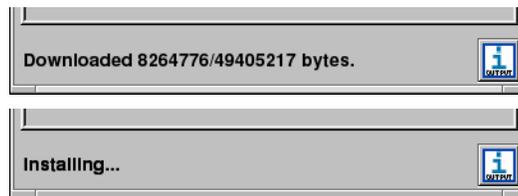


Figure 15-5: Installation status

Once the installation of the new version of VisIt is complete, VisIt will ask whether you want to migrate your current session to the new version of VisIt (see Figure 15-6). If you choose to migrate your session, VisIt will save your session, quit and restart using the saved session in the new version of VisIt.

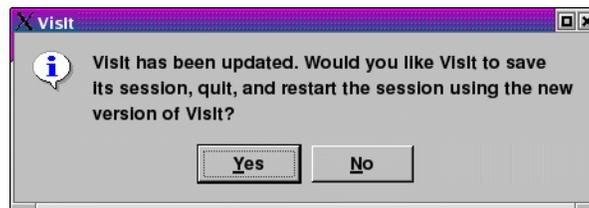


Figure 15-6: Restart prompt

Symbols

.visit file 36, 261

Numerics

2D Annotations 229
2D line annotation objects 242
3D Annotations 231

A

Activate button 35
Activating a database 281
Activating a host profile 305
Activating tools 152
active color table 245
Active plots area 41
Active time slider 283
active time step 37
active window 145
Adding a bookmark 333
Adding a database state keyframe 266
Adding a keyframe 266
Adding a new vis window 143
Adding a plot 159
Adding a view keyframe 267
Adding an operator 88, 161
additional command line options 306
Advanced host profile options 310
Alias Wavefront OBJ file format 28
Alias WaveFront Obj files 129

Altering an existing database correlation 290
ANALYZE file format 24
Animation playback mode 264
Animation playback speed 263
Animation Slider 37
Animation Text Field 37
Animation Window 263
Annotation Colors 235
Annotation Objects 236
Annotation Window 228, 230, 231, 237
ANSYS file format 24
Appearance Window 319, 320
Applying settings 21
ArcInfo binary grid file format 25
Auto update 21
Automatic database correlation 291
Automatically showing the Pick Window 220
AUX file format 24
axis labels 230, 233

B

background color 235
background style 236
Bank 313
Block number 105
BMP files 129
Bookmarks tab 334
Boundary 45
Boundary plot 44, 46
Boundary plot attributes window 46, 47
BOV file format 24
Box operator 93
Box operator attributes window 93
Box Tool 274
Boxlib file format 24
Browsing subsets 165
built-in expressions 183

C

- Cartesian coordinates 126
- Center of rotation 155
- CGNS file format 24
- Changing a SIL restriction 166
- Changing bounding-box mode 157
- Changing colors 46
- Changing directories 30
- Changing elevation height 102
- Changing filters 30
- Changing font size 333
- Changing GUI colors 319
- Changing GUI Orientation 320
- Changing GUI Style 319
- Changing hosts 29
- Changing mesh line attributes 59
- Changing plot variables 44
- Changing surface representations 326
- Changing the background style 236
- Changing the color settings 141
- Changing the color table 62
- Changing the mesh color 59
- Changing the number of streamlines 71
- Changing the opacity variable 85
- Changing the opaque color 59
- Changing the order of operators 90
- Changing the plot time range 267
- Changing the point type 59, 63
- Changing the streamline color 72
- Changing the tensor colors 78
- Changing tube appearance 128
- Changing view perspective 154
- Changing window layouts 145
- Checking for new versions 343
- Choosing a MIR algorithm 169
- Choosing an export file format 139
- Choosing movie formats 134
- Choosing movie generation method 137
- Choosing the active light 249
- Choosing the Label plot's variable 54
- Choosing the movie name 136
- Clearing a Python script from a tab 269
- Clearing cache 315
- Clearing pick points 158
- Clearing plots 158
- Clearing plots from vis windows 144
- Clearing reference lines 159
- Clip operator 94
- Clone window on first reference 296
- Cloning the active window 144
- Close compute engine confirmation dialog 315
- Closing a compute engine 315
- Closing a database 39
- CMAT file format 24
- color control point 245
- Color table 62
- color table 244, 245
- Color table window 244, 245, 246
- Command Window 268, 269
- Compute Engine Window 314
- Concise pick output 219
- Cone operator 96
- Cone operator attributes window 97
- Connecting to a running simulation 33
- Constructive Solid Geometry 116
- Contents tab 334
- Contour Plot 49
- Contour plot 44
- Contour plot attributes window 50
- Controlling image quality 85
- Coordinate system conversion 126
- Copying plots on first reference 323
- Copying window attributes 146
- Copying Windows 296
- Cosmos file format 24
- CosmosPP file format 24
- Creating a new annotation object 237
- Creating a new color table 245
- Creating a new database correlation 289
- Creating a new expression 179
- Creating a new host profile 304
- Creating a plot 42
- Creating complex subsets 167
- CSG 117
- Current Plot 61, 77
- Curve files 129
- Curve Plot 51
- Curve plot 44, 222

Curve plot attributes window 223
Curve2D file format 25
Customizing gradient backgrounds 236
Cycle database correlation 288
cycle number 37
Cylinder operator 98
Cylinder operator attributes window 98
Cylindrical coordinates 126

D

Database comparison 292
database correlation 285
database correlations 281
Database correlations and time sliders 285
database information 229
database server 29
Decimate operator 98
Decimate operator attributes window 99
Default directory 30
DeferExpression operator 100
Delete 43
Deleting a color table 246
Deleting a database correlation 291
Deleting a host profile 305
Deleting a keyframe 267
Deleting a plot 43
Deleting a vis window 144
Deleting active plots 160
Deleting an annotation object 238
Deleting an expression 180
DEM file format 25
Depth testing for 3D Label plots 55
Determining the host name 311
Digital Elevation Map 25
Displace operator 100
Displace operator attributes window 101
Displaying global node and cell numbers 219
distributed mode 301
domains 166

Drawing a plot 43, 160
Drawing internal surfaces 47, 74
Drawing only clean zones 48
Drawing points on the Curve plot 52
Dune file format 25
Dynamic lineout 221
dynamic load balancing 309

E

Editing a continuous color table 246
Editing a discrete color table 247
Elevate operator 101
Elevate operator attributes window 102
Enabling and Disabling Plugins 322
Engaging spin mode 157
Engine option window 312, 313
EnSight file format 25
Environment 303
Enzo file format 25
Equation of state database 27
ESRI Shapefile format 27
Executing a query 214
Exodus file format 25
Expanding plots 89
Export Database Window 129, 130, 138, 139
Exporting a color table 246
Exporting variables 139
Expression (Absolute value) 184
Expression (Addition) 183
Expression (Arccosine) 184
Expression (Arcsine) 186
Expression (Arctangent) 186
Expression (Array compose) 185
Expression (Array decompose) 186
Expression (Base 10 logarithm) 194
Expression (Ceiling) 187
Expression (Cell Area) 185
Expression (Cell aspect gamma) 186
Expression (Cell aspect ratio) 186

Expression (cell number) 207
 Expression (Cell shape and size) 203
 Expression (Cell shape) 203
 Expression (Cell stretch) 205
 Expression (Cell volume) 206
 Expression (Cell warpage) 206
 Expression (Condition number) 187
 Expression (Conditional) 193
 Expression (conn_cmfe) 292
 Expression (Connectivity-based common mesh field evaluation) 187
 Expression (Conservative smooth) 188
 Expression (Convert to polar coordinates) 199
 Expression (Cosine) 188
 Expression (curl) 189
 Expression (Cylindrical radius) 189
 Expression (Cylindrical theta) 189
 Expression (Degrees to radians) 189
 Expression (Diagonal ratio) 190
 Expression (divergence) 190
 Expression (Division) 184
 Expression (Effective tensor) 191
 Expression (Eigenvalue) 191
 Expression (Eigenvector) 191
 Expression (Equality) 191
 Expression (Exponentiation) 184
 Expression (External node) 191
 Expression (Floor) 192
 Expression (General distortion measure) 199
 Expression (global node number) 192
 Expression (global zone number) 192
 Expression (Gradient) 193
 Expression (Greater than or equal) 192
 Expression (Greater than) 193
 Expression (Jacobian) 193
 Expression (Laplacian) 194
 Expression (Largest angle) 194
 Expression (Less than or equal) 194
 Expression (Less than) 194
 Expression (Logical and) 185
 Expression (Logical not) 198
 Expression (Logical or) 199
 Expression (material error) 195
 Expression (material interface reconstruction volume fraction) 197
 Expression (Material volume fraction) 195
 Expression (Matrix determinant) 190
 Expression (Matrix inverse) 193
 Expression (Maximum edge length) 195
 Expression (Maximum side volume) 196
 Expression (Maximum tensor shear) 205
 Expression (Mean average) 196
 Expression (Median) 196
 Expression (Mesh coordinates) 188
 Expression (Mesh degree) 189
 Expression (Mesh skew) 203
 Expression (Mesh taper) 205
 Expression (Minimum side volume) 197
 Expression (Modulo) 197
 Expression (Multiplication) 184
 Expression (Natural logarithm) 194
 Expression (Neighbor) 198
 Expression (Node degree) 198
 Expression (node number) 198
 Expression (Not equal) 199
 Expression (Number of materials) 198
 Expression (Polar Phi) 200
 Expression (Polar Radius) 200
 Expression (Polar theta) 200
 Expression (Principal deviatoric vector of tensor) 200
 Expression (Principals of tensor) 200
 Expression (Processor ID) 201
 Expression (Radians to degrees) 201
 Expression (Random number) 201
 Expression (Recenter variable) 201
 Expression (Relative cell size) 202
 Expression (relative difference) 201
 Expression (Revolved cell volume) 202
 Expression (Revolved surface area) 202
 Expression (Round) 202
 Expression (Scaled jacobian) 202
 Expression (Shear) 203
 Expression (Sine) 203
 Expression (skew scaling) 206
 Expression (Smallest angle) 204
 Expression (Species mass fraction) 204
 Expression (Square root) 204
 Expression (Square) 204
 Expression (Subtraction) 183
 Expression (Surface normal) 205
 Expression (Tangent) 205

Expression (Trace of tensor) 206
Expression (Unary negation) 183
Expression (Vector cross product) 188
Expression (Vector dot product) 190
Expression (Vector magnitude) 195
Expression grammar 180
Expression Window 177, 179
Expressions 177
ExternalSurface operator 100, 103

F

File extensions 23
File grouping 32
File Information Window 38
File pane 324
File Panel 23, 34, 38
File panel 17, 322, 325
File panel display of cycles and times 325
File panel time display 324
File Selection Window 23, 31, 36
FilledBoundary 45
FilledBoundary plot 44
FilledBoundary plot attributes window 46, 47
Finding materials with low volume fractions 171
FLASH file format 25
Flying through plots 150
Forcing material interface reconstruction 172
foreground color 235
Formatting labels 56
Frames per second 329
Freeform control 84
Fullframe mode 155, 254

G

Gaussian control 85
GDAL library 25
General Annotations 228
Geographic Information Systems 25
geometry file formats 129
Geometry smoothing 47, 60, 63, 75
GIS 25
Global Lineout Window 223
gradient background 236
gradient style 236
grid lines 230, 232
Group 31
Group number 105
Grouping files 31
Growing layers 111
GUI orientation 320

H

HDF5 file format 28
Heads on the vector glyph 81
Help Window 332, 333
Help Window Toolbar 332
Hide/Show 43
Hiding a plot 43
Hiding active plots 160
Hiding all axes 232
Hiding an annotation object 238
Hiding toolbars 151
Histogram plot 44, 52
Histogram plot attributes window 53
horizontal layout 320
host profile 304
Host Profile Window 304, 306
hot points 273

I

Image annotation objects 243
Image file formats 26
image files formats 129
image volumes 26
Index Select operator 103
Index select operator attributes window 104, 105
Index tab 334
Interactive mode 224
Interactors window 150
Internet Port Allow (IPA) 312
Interrupting a compute engine 314
InverseGhostZone operator 105
Inverting colors 158
Inverting the clipped region 96
Isosurface operator 106
Isosurface operator attributes window 107
Isovolume operator attributes window 109

J

JPEG files 129

K

Keyframe time 266
Keyframing mode 265
Keyframing Window 264
KullLite file format 26

L

Label Plot 54
Label plot 44, 54, 55
Label plot attributes window 56, 57
Labelling subset names and material names 57
Lasnex PDB file format 27
Launch progress window 303
Light color and brightness 250
Light type 249
Lighting 62, 77
Lighting edit mode 248
Lighting Window 248
Limits 61, 77, 83
Line tool 275, 276
Lineout 148
Lineout mode 149, 220
Lineout Operator 223
Lineout operator 109
Lineout operator attribute window 224
Lineout operator attributes window 224
Lines file format 26
Load balancing 308
Locking time 162, 297
Locking tools 298
Locking views 154, 162, 258, 297
Locking vis windows together 147

M

Main Window 17, 21, 34
Maintain data 260
Make default 317
Making a stereo movie 135
Making all cells visible 106
Making longer streamlines 68
Making smoother looking streamlines 68
Managing operators 87

Managing plots 41
Managing the selected file list 31
Material Reconstruction Options Window 169, 171
Merge operator 109
Mesh color 59
Mesh Plot 58
Mesh plot 44
Mesh plot attributes window 58, 59
Mesh plot opaque modes 58
Mesh variable 179
Mili file format 26
MIR 169
Mixed variables 173
Movie progress dialog 138
Moving a keyframe 267
Moving the ThreeSlice operator 123
Moving toolbars 151
Multiple time sliders 283

N

NASTRAN bulk data file format 26
Navigate 148
Navigate mode 148
Navigation 332
Navigation styles 150
NETCDF file format 26
Notepad 17
Number of samples 85

O

octant 113

OnionPeel operator 109
OnionPeel operator attributes window 111
Opacity 46, 62, 74
Opacity variable 85
Opaque color 59
Opaque mode 58
Opening a file 35
Opening a file on a remote computer 36
Opening a time varying database 36
Opening files 35
Original Data 61, 77
Output Indicator 21
Output Window 21, 30
OVERFLOW file format 26
Overlay 38
Overlaying a database 38

P

Padded index database correlation 286
Password Window 302
PATRAN neutral file format 26
PDB file format 27
Pf3D simulation 27
Pick 148
Pick mode 149, 216
Pick Window 218, 220
Pick window 158
Picking a compute engine 314
Picking an output directory for saved files 130
Picking over time 220
Pipeline caching 263
Pixie3D file format 27
plane tool 276
Playing animations 37
plot 41
Plot Extents 329
Plot legends 229
Plot list 41, 42, 43, 89
Plot Manager 17

Plot2D file format 27
 Plot3D file format 27
 Plotting surface normals 100
 Plotting the difference between two databases 292
 Plotting time derivatives 294
 Plotting values relative to the first time state 293
 Plugin Manager Window 88, 92, 322
 plugins 322
 PNG files 129
 Point size 48, 59, 62, 63, 74
 Point Tool 278
 Point type 48, 74
 Point3D file format 27
 Popup color menu 46, 50, 73
 popup menu 148
 Ports 311
 Positioning a light 249
 Positioning the slice plane 118
 Positioning the slice plane using the Plane Tool 120
 Positioning the slice sphere using the Sphere Tool 122
 Posting a window 19
 Posting windows by default 323
 PPM files 129
 Preferences Window 322, 324
 Printer Window 129, 140, 141
 Project operator 111
 Projecting the slice to 2D 97, 120
 Pseudocolor plot 44, 60
 Pseudocolor plot attributes window 61
 Queries (compactness) 209
 Queries (cycle) 209
 Queries (eulerian) 209
 Queries (Integrate) 210
 Queries (Kurtosis) 210
 Queries (L2Norm Between Curves) 210
 Queries (L2Norm) 210
 Queries (Lineout) 210
 Queries (Max) 210
 Queries (Min) 210
 Queries (MinMax) 211
 Queries (Moment of inertia) 211
 Queries (NodeCoords) 211
 Queries (NodePick) 211
 Queries (Number of cells) 211
 Queries (Number of nodes) 211
 Queries (Pick by cell number) 212
 Queries (Pick by node number) 212
 Queries (Pick) 211
 Queries (Revolved surface area) 212
 Queries (revolved volume) 212
 Queries (Skewness) 212
 Queries (spatial extents) 213
 Queries (Spherical compactness factor) 213
 Queries (time) 213
 Queries (variable sum) 213
 Queries (volume) 213
 Queries (Watertight) 213
 Queries (weighted variable sum) 214
 Queries (zone center) 214
 Query 207
 Query Over Time Window 215
 Query window 207
 Querying over a time range 215
 Querying over time 214
 Quitting VisIt when there are multiple user interfaces 270

Q

Queries (2D area) 209
 Queries (3D surface area) 209
 Queries (area between curves) 209
 Queries (Centroid) 209

R

Raster Postscript 129
ray-casting 82
Reading accurate cycles and times from databases 325
Recentering the view 154
Reference line labels 225
Reflect operator input mode 113
Reflecting plots 114
Reflection limits 114
Refreshing the file list 33
Remove Recent Paths Window 33
Removing all operators 161
Removing half of a plot 94
Removing one quarter of a plot 95
Removing operators 90
Removing the last operator 161
Rendering Options 250
Rendering Options Window 251, 326
Reopen 37
Reopening a database 37
Replace 37
Replacing a database 37
Resample operator 115
Resample operator attributes window 116
Resampling onto a rectilinear grid 116
Resampling surfaces projected to 2D 117
Resetting the view 154
Resizing the box 93
Restricting the number of labels 55
Returning node information 219
Returning zone information 220
Revolve operator 114
RGB files 129
Rotation 125

S

SAF file format 27
Samples per ray 86
SAMRAI file format 27
SAR file format 27
Save movie wizard 129, 133, 134
Save settings 318
Save Window 129, 130, 131
Save window 133
Saving and reusing views 155
Saving binary geometry files 132
Saving curves 223
Saving images with screen capture 132
Saving session 268
Saving stereo images 132
Saving the Command Window's Python scripts 269
Saving tiled images 133
Scalable rendering 328
Scalar Mesh Variable 179
Scale 126
Scale point size by variable 59, 63
Scaling 50, 107
Scaling an input variable 66
Scaling the data 62, 76
Scatter Plot 64
Scatter plot 44
Scatter plot attributes window 64, 65, 67
Scatter plot wizard 65
seed for OnionPeel 110
Selecting a help page 333
Selecting a plot 43
Selecting a variable 65
Selecting an annotation object 238
Setting a selection range 104
Setting an input variable's role 65
Setting axis label properties 230, 233
Setting axis titles and units 231, 234
Setting background and foreground colors 235
Setting batch queue options 313
Setting colors 83
Setting contour colors 50

Setting curve attributes 223
 Setting curve color 52
 Setting grid line properties 230, 232
 Setting how cells are removed 93
 Setting image resolution 132
 Setting isosurface levels 106
 Setting Limits 50, 107
 Setting line style and line width 51
 Setting lineout endpoints 224
 Setting opacities 84
 Setting operator attributes 91
 Setting paper format 141
 Setting parallel options 307
 Setting plot attributes 43
 Setting point properties 48, 67, 74
 Setting streamline appearance 71
 Setting the active time step 37
 Setting the active window 145
 Setting the center of rotation 258
 Setting the colors 67
 Setting the cylinder's endpoints 98
 Setting the default bank 309
 Setting the file type 131
 Setting the histogram calculation method 53
 Setting the histogram data range 52
 Setting the host name 305
 Setting the isosurfacing variable 108
 Setting the machine file 313
 Setting the minimum and maximum values 66
 Setting the number of bins 53
 Setting the number of contours 49
 Setting the number of frames 265
 Setting the number of lineout samples 224
 Setting the number of nodes 308
 Setting the number of printed copies 141
 Setting the number of processors 308, 312
 Setting the number of vectors 79, 81
 Setting the output mesh type 124
 Setting the parallel launch method 307
 Setting the partition/pool 308
 Setting the pick variable 218
 Setting the plot axis type 233
 Setting the printer destination 141
 Setting the radius 98
 Setting the remote user name 306
 Setting the save file name 131
 Setting the streamline source 69
 Setting the tensor scale 78
 Setting the threshold variable 124
 Setting the time curve's destination window 215
 Setting the time parallel limit 309
 Setting the time state 262
 Setting the timeout 306
 Setting the type of graph 53
 Setting the variable range 124
 Setting tick mark properties 230, 232
 Setting up the parallel environment 310
 Setting vector appearance 80
 Shadows 252
 Sharing a compute job 310
 Showing curve labels 52
 Showing internal zones 58
 Showing node and zone numbers 55
 Showing selected files 324
 SIL 164
 SIL restriction 164
 Silo file format 27
 Simplifying heavily mixed cells 171
 Simulation interface library 33
 Simulation Window 314, 315
 Skew factor 62, 77
 Slice operator 118
 Slice operator attributes window 118
 Smooth operator 120
 Software rendered images 86
 solid background 236
 Species 173, 174
 Specifying a machine file 309
 Specifying the slice cone 97
 Specular lighting 251
 Spherical file format 27
 sphere tool 278
 SphereSlice operator 121
 SphereSlice operator attributes window 122
 Spherical clipping 96
 Spherical coordinates 126
 Splash screen 331
 ssh 303
 static load balancin 309
 Stereo 132
 Stereo images 327

STL file format 27
STL files 129
streaming movie 135
Streamline Plot 67
Streamline plot 44
Streamline plot attributes window 69, 71
Stretched index database correlation 287
subset categories 165
Subset Inclusion Lattice 164
Subset Plot 72
Subset plot 44, 72, 74
Subset plot attributes window 73, 74
Subset Window 163, 165, 166
Surface and Wireframe modes 75
Surface color 76
Surface plot 44, 75
Surface plot attributes window 76
surface_normal expression 100
Switching window modes 151
Symmetric Tensor 179

Time slider 283
Time slider annotation objects 240
time varying database 36
timeout 306
Toolbar 150
transfer function 82
Transform operator 125
Transform operator attributes window 125, 126
Translation 126
Triangle count 329
Truecolor plot 44, 79
Try harder to get accurate cycles/times 325, 326
Tube operator 128
Turning a light on 249
Turning database information on/off 229
Turning multiple sets on and off 168
Turning off incident nodes and cells in pick output 219
Turning off pick markers for new pick points 219
Turning off species 175
Turning off the triad 232
Turning plot legends off globally 229
Turning user information on/off 228

T

Tails on the vector glyph 81
TecPlot file format 27
Tensor plot 44, 78
Tensor plot attributes window 78
Tensor variable 179
Tetrad file format 28
Text annotation objects 238
TFT file format 28
ThreeSlice operator 122
Threshold operator 123
Threshold operator attributes window 124
tick marks 230, 232
TIFF files 129
Time database correlation 288
time derivative 294
time format string 241
Time limit 313

U

ULTRA 129
Undo view 149, 154, 258
UNIX C-Shell 30
UnPost 19
Use background 59
Use foreground 59
User information 228
user information annotation 228
user preferences 317
Using axis alignment buttons 256
Using display lists 327
Using Resample with CSG meshes 117

Using the Decimate operator 99
Using the Displace operator 101
Using the Elevate operator 102
Using the GUI and CLI to design a script 269
Using the Isovolume operator 108
Using the main menu 19
Using the Revolve operator 115
Using the Smooth operator 121
Using view commands 256

V

Variable 44, 54
Variable centering 60
VCR buttons 37
Vector Mesh Variable 179
Vector plot 44, 80
Vector plot attributes window 81
Vector scaling 81
View recentering 258
Viewing status messages 21
virtual database 31, 32
VisIt architecture 15
visit -movie 270
Vista file format 28
ViSUS file format 28
Volume 82
Volume plot 44, 82, 83, 85, 86
Volume plot attributes window 83, 84
volume transfer function 82
volume-rendering 82
VTK file format 28
VTK files 129

W

window layout 145
window modes 148
Wireframe mode 47, 74
Wireframe properties 76
Wireframe view 51

X

Xmdv file format 28, 139
XML 317

Z

Zoom 148
Zoom interactor settings 149
Zoom mode 148