MeshTV Command Line Interface Manual

December 10, 1996

Version 2.7.2

Lawrence
Livermore
National
Laboratory

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

MeshTV Command Line Interface Manual

1.0 Overview

MeshTV is an interactive program for visualizing and analyzing scientific data. MeshTV reads SILO[1] data files, which allows it to run on many hardware platforms. MeshTV handles 1D, 2D, and 3D data and provides a variety of operations on the data, enabling it to be both a visualization tool and a data analysis program. A full description of the SILO data format for MeshTV can be found in the MeshTV-SILO Manual[2], and instructions for writing SILO files can be found in the SILO User's Guide[3].

While MeshTV has a Command Line Interface (CLI), most people prefer to use MeshTV's Graphical User Interface (GUI), which is described in the MeshTV User's Manual[4]. If you are just starting MeshTV for the first time, you might want to use the GUI for awhile to get a feeling for how the program works. You might also want to read the MeshTV Getting Started Manual[5]. The MeshTV Getting Started Manual describes important MeshTV concepts and assumptions, and it also contains a tutorial for the GUI.

This manual details the various commands supported by MeshTV as they are accessed via the Command Line Interface. Users might want to use the CLI for production (batch) jobs, or to access commands which have yet to be implemented in the GUI.

---

1. A high-level, portable interface that was developed at Lawrence Livermore National Laboratory to address difficult database issues, such as different, incompatible file formats and libraries, most of which used non-standard features of the Cray compilers.
2. MeshTV-SILO Manual, To Be Released.
3. SILO User's Guide, UCRL-MA-118751
4. MeshTV User's Manual, To Be Released.
5. MeshTV Getting Started Manual, To Be Released.

2.0 Conventions

The following typographical conventions are used in the command summaries:

italic	An option or parameter to be replaced by the user with an appropriate value.
`typeface`	Command names and options. These are given to MeshTV exactly as shown.
[ ]	Arguments enclosed in square brackets are optional. The brackets are not part of the argument.
|	The “pipe” symbol indicates “or”, so if two arguments are separated by the pipe symbol, this means one or the other is appropriate.

Many of the commands use commas to separate options, as in:

```
bnd var=material, lc=red
```

These commas are optional.

Real numbers ending in zero can be entered with or without the decimal point. For example, 0 can be entered as `0`, `0.`, or `0.0`.

3.0 MeshTV Execute Line

MeshTV can be run in command line mode (without the GUI). When used in this way, the program name is **`meshtvx`** rather than **`meshtv`**. To run MeshTV in command line mode, type the following at a UNIX^{TM} prompt:

`meshtvx` *[options]*

options are:

`-banner`	Assigns banners for PostScript^{TM} output.
`-nowin`	Do not open output windows. This is useful for the production (batch) mode of operation.
`-n`	End the MeshTV prompt with a newline character. This is useful when MeshTV is run from within a script.
`-noinfo`	Do not output MeshTV informational messages.
`-nowarn`	Do not output MeshTV warning messages.
`-noerror`	Do not output MeshTV error messages.

`-q`	Quiet mode. Do not output informational or warning messages. This is equivalent to using the `-noinfo` and `-nowarn` options.
`-Q`	Really quiet mode. Do not output any messages. This is equivalent to using the `-noinfo`, `-nowarn`, and `-noerror` options.
`-ps` *rootname*	Write all plots in PostScript™ format to the file family, the first file of which will be named *rootname*0000.ps. File families are described in the MeshTV Getting Started Manual.[1]
`-rgb` *rootname*	Write all plots in SGI rgb format to the file family, the first file of which will be named *rootname*0000.rgb. File families are described in the MeshTV Getting Started Manual.
`-rps` *rootname*	Write all plots in Raster PostScript™ format to the file family, the first file of which will be named *rootname*0000.ps. File families are described in the MeshTV Getting Started Manual.
`-s` *filename*	Specify a filename from which to read commands. *filename* is a file containing MeshTV commands.
`-tif` *rootname*	Write all plots in tiff format to the file family, the first file of which will be named *rootname*0000.tif. File families are described in the MeshTV Getting Started Manual.
`-xwd` *rootname*	Write all plots in X Window Dump (xwd) format to the file family, the first file of which will be named *rootname*0000.xwd. File families are described in the MeshTV Getting Started Manual.
file	Open the SILO file named *file*.

Commands contained in a file can be input to MeshTV via the C Shell redirection facility.

`meshtvx` *[options]* **<** *command_file*

Commands generated from a program can be piped to MeshTV.

program | `meshtvx` *[options]*

4.0 MeshTV Production Mode of Operation

MeshTV can be run in a production (batch) environment where a script program, for example, supplies plot commands in a file from which MeshTV generates plots as output files of various formats. In this case, no output window needs to exist, so the `-nowin` option might be used. The `source` command invoked from within MeshTV, or the redi-

1. MeshTV Getting Started Manual, To Be Released.

rection of a command file on the execute line, can be used to supply the plot commands. The `foutput` command or the `-ps` execute line option can be used to direct plots to a PostScript™ file. Some examples of production mode usage follow.

4.1 MeshTV Example: nowin, ps, redirection

```
meshtvx -nowin -ps prob prob.silo <prob.log >/dev/null
```

Commands in file prob.log and SILO file prob.silo are used to generate plots which are written to the PostScript™ file family which starts with prob0000.ps. Plots are not sent to an output window. All MeshTV messages are redirected to the “great bit bucket in the sky” (/dev/null) rather than being sent to the user.

4.2 MeshTV Example: nowin, -n, interactive

```
meshtvx -nowin -n
1-->
foutput rootname=prob2
2-->
open prob200000.silo
Current DB is: prob200000.silo
‘2D rectilinear file’
3-->
source prob2.log
4-->
open prob200100.silo
Current DB is: prob200100.silo
‘2D rectilinear file’
5-->
source prob2.log
6-->
quit
```

Commands in file prob2.log and data in SILO files prob200000.silo and prob200100.silo are used to generate plots which are written to the PostScript™ file family which starts with prob20000.ps. Plots are not sent to an output window. The newline character ends the MeshTV prompt.

4.3 MeshTV Example: nowin, -Q, interactive

```
meshtvx -nowin -Q
1-->open prob3.silo
2-->foutput rootname=prob3
3-->source prob3.log
4-->foutput rootname=prob4
5-->source prob4.log
```

```
6-->quit
```

Commands in file prob3.log and data in the SILO file prob3.silo are used to generate plots which are written to the PostScript^{TM} file family which begins with prob30000.ps. Ditto for prob4. Plots are not sent to an output window. The prompt does not include the newline character. No warning, error, or informational messages are sent.

If invalid commands are sent to MeshTV, an error message will be returned to the standard error ("standard err"). This message will begin with the string "MeshTV Error:"

Whenever MeshTV is run, whether from the command line or via the GUI, it will output a log of the commands it received during the session. This log will be named "`%meshtv.log`" You can use this log to rerun MeshTV, either from within the code by issuing the `source` command, or by using the redirection method mentioned earlier. The only restriction is that you must rename the file, or MeshTV will attempt to read from and write to the same file, and that would cause problems.

5.0 Command Summaries

The command summaries which follow describe the usage of every command in MeshTV. Each command summary is broken into five sections — Synopsis, Arguments, Description, Examples, and See Also.

The Synopsis section contains a brief summary of the command along with its arguments. The Arguments section contains a description of each argument supported by the command. The Description section provides a more thorough explanation of what the command does. The Examples section demonstrates sample uses of the command. The See Also section refers the reader to any related commands or documents which might also be of interest.

Note that these commands can be issued via the GUI by using the Command Line window. If these commands are issued from the GUI, all text output will go to the Printer Icon Output window. If these commands are issued from the Command Line Interface version of MeshTV, output goes to the standard output ("standard out"), which is usually the shell window in which MeshTV was invoked.

If you issue these commands from the GUI, the GUI's windows will not correctly reflect the changes. For example, if you type "animate fps=5.0", the animate window will not reflect that change. In general, if you use the GUI, you should issue commands via the GUI's windows whenever possible.

addframe — Add a frame to the list of frames in an animation.

Synopsis:

`addframe` *filename*

Arguments:

filename	A SILO filename, including either absolute or relative path specifications.

Description:

Add a frame into the list of frames in an animation. The new frame will be inserted after the current frame, which can be set via the `setframe` command. Note that you can not add a frame to the beginning of an animation. To add a frame at the beginning, you must start a new animation.

For details on animation in general, please see the MeshTV Getting Started Manual.

Examples:

```
open abc1.silo
addframe /usr/people/sample/abc2.silo
addframe ../data/abc3.silo
addframe ~/data/abc4.silo
```

See Also:

`animate, nextframe, play, prevframe, setframe, stop,`
`MeshTV Getting Started Manual`

alias — Define or list command aliases.

Synopsis:

`alias`
`alias` *name*
`alias` *name* `"`*value*`"`

Arguments:

name	An alphanumeric string, starting with a letter.
`"`*value*`"`	A quoted string containing MeshTV commands.

Description:

With the `alias` command, you choose a “name” to represent a string of MeshTV commands or aliases.

The `alias` command works in much the same way as it does under the C-Shell in UNIX™, where aliases (names) represent other operations. This allows the user to effectively rename existing commands, and to define a single command which actually executes numerous other commands.

With no arguments, `alias` lists all aliases along with their values. With one argument, `alias` prints the value associated with the given *name* argument.

Examples:

```
alias clr “clear”          # Define an alias
alias clr                  # Print definition of alias
alias                      # Print definition of all aliases
alias plt “plot”
alias foo “clr; plt”
alias cplt “clear; plot”
```

See Also:

`unalias`

animate — Set animation options.

Synopsis:

`animate` *option=value [, option=value]*

Arguments:

option One of the names of an animation property. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`cache`	`geometry, off`	Cache polygons in the machine's RAM. This speeds up animations, but requires large amounts of memory.	`off`
`fps`	real number >= 0.0	The number of frames per second. Setting the rate higher than the machine can draw the images will cause frames to be skipped. 0 has a special meaning; it indicates that the animation should go as fast as the machine allows.	`0.0`

Description:

Set options for the animation. `cache` speeds up the animation significantly, but it can only be used if your machine has sufficient memory, probably a minimum of 40 MB, though the actual number might be more or less than 40 since this number depends on the size of the problem and the number of frames in the animation. `fps` provides control over the frames per second.

Examples:

```
animate fps=5.0, cache=geometry
```

See Also:

`addframe, nextframe, play, prevframe, setframe, stop`

area — Define area (lighting and color) properties for 3D plots.

Synopsis:

`area` *area_id option value [,option value, ...]*

Arguments:

area_id	An integer ranging from 0 to 15 denoting which area to define.
option	One of the names of an area property. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`ambcoef`	floating point (0. to 1.)	Ambient coefficient. Fraction of ambient light reflecting from a surface. Ambient light is light that enters the lighting calculation independent of the orientation of the surface being illuminated or the location of the eye.	`0.5`
`color`	1 <= integer <= 9	Color number from the MeshTV standard color table. (See APPENDIX B: Color Names.)	`2`
`diffcoef`	floating point (0. to 1.)	Diffuse coefficient. Fraction of non-ambient light diffusely reflecting from a surface. Diffuse reflection is an approximation of the light reflected equally in all directions from a surface. This gives a surface a dull matte appearance.	`1.0`
`speccoef`	floating point (0. to 1.)	Specular coefficient. The fraction of light from non-ambient sources specularly reflecting from a surface. Specular reflection approximates the light reflected from a surface that causes highlights on shiny objects. The intensity of specular reflection is highly dependent on the viewing angle of the observer.	`1.0`
`speccolor`	1 <= integer <= 9	Color of the specular highlight. This is taken from the MeshTV standard color table. (See APPENDIX B: Color Names.)	`1`
`specexp`	floating point >= 0.0	Specular exponent. This number represents the “shininess” of a surface. The higher the specular exponent, the shinier the surface.	`2.0`
`transcoef`	floating point (0. to 1.)	Transparency coefficient. The fraction of light transmitted through a surface.	`0.0`

Description:

The `area` command defines lighting and color properties for areas in 3D plots in the current output window. Each material in the problem has a corresponding “area”, so the first material corresponds to area 0, and any changes in lighting or color to area 0 result in lighting and color changes to the first material.

A discussion of lighting properties can be found in the MeshTV Getting Started Manual.

Examples:

```
area 1 color 3, speccoef 0.5
```

See Also:

`lightsrc, MeshTV Getting Started Manual`

block — Set the active block numbers.

Synopsis:

`block` *block# [, [begin*`:`*end | block#], ...]*
`block` *begin*`:`*end [, [begin*`:`*end | block#], ...]*
`block all`

Arguments:

block#	An integer representing a block identifier.
begin	An integer number representing the beginning of a range.
end	An integer number representing the end of a range.
`all`	Flag indicating that all blocks should be active.

Description:

The `block` command takes a comma-separated list of block numbers and/or begin-end pairs, and makes those blocks active. Only data from active blocks will be plotted.

A block is an element of a multimesh. A multimesh consists of several blocks, each of which is a separate mesh in its own right. Each block has its own variables, materials, etc.

Successive `block` commands are not cumulative, so each subsequent `block` command turns off currently activated blocks and turns on the ones that have just been specified.

Examples:

```
block 1,3,5:10           # Activate 1,3,5,6,7,8,9, and 10
block 44                 # Activate just block 44
block all                # Activate all blocks
```

See Also:

bnd — Set plotting options for the material boundary plot.

Synopsis:

`bnd` *option=value [,option=value, ...]*

Arguments:

option A plot option for the material boundary plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`lc`	line color	Line color. (See APPENDIX B: Color Names.)	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX C: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`pop`	See `pop` option on page 69.		`off`
`var`	variable name	Name of variable to get material data from.	`material`
`vf`	real number from 0.0 to 1.0	Volume fraction. This value isn't taken into account if the problem only has clean zones. If the problem has mixed zones, you can set the value less than 0.5 to make the boundaries encompass more, or to less than 0.5 to make them encompass less. 0.5 is the most accurate. To get a feeling for this option, try playing around with it a bit.	`0.5`

Description:

The `bnd` command sets the options used when generating a material boundary plot. Note that this command does not change any existing material boundary plots, only future ones.

Examples:

```
bnd lc=red, ls=dot, vf=.4
bnd var=myvar
```

See Also:

`matcol, materials, plot, smat`

cat — Print the contents of the given file.

Synopsis:

`cat` *filename*

Arguments:

filename	The name of the file to print.

Description:

The `cat` command operates like the cat command in UNIX[TM] C-Shell, except it only works on one file at a time. Use this command if you want to examine the contents of a short ASCII file without opening up an editor or another window. If you issue this command from within the GUI, the output will go to the Printer Icon output window, otherwise it will just print to the shell window running MeshTV.

Examples:

```
cat myfile
cat /usr/people/sample/input.file
```

See Also:

`csh, sh`

cd — Change the active directory within the current SILO input file.

Synopsis:

`cd`
`cd` *path*

Arguments:

path	A path indicating the directory to change into.

Description:

The `cd` command operates within a SILO data file in the same manner that the UNIX™ cd command operates within the UNIX™ file system. By default, the active directory is the root directory, indicated by a slash "`/`". By using the `ls` and `cd` commands, one can navigate through a SILO file. The `cd` command understands both absolute and relative pathnames, hence the path "`..`" means move up one directory level. Without an argument, `cd` changes back to the root directory.

Note that this command works on SILO files, not UNIX™ files. This means that the "~" operator in a path will not work.

Examples:

```
cd /
cd new_dir
cd ../old_dir
```

See Also:

`close, ls, open, pwd`

center3 — Reassign the center point in the active 3D window.

Synopsis:

`center3` *xnum ynum znum*
`center3 off`

Arguments:

xnum	X position for the center point. A floating point number.
ynum	Y position for the center point. A floating point number.
znum	Z position for the center point. A floating point number.
`off`	Return the center point to the default position.

Description:

This command reassigns the “center point” of the view. In other words, it picks the point toward which the “camera” or “eye” is looking. This is particularly useful before a `zoomf3` to direct the eye to the area of interest before the area is enlarged.

Examples:

```
center3 1.5, 2.0, 3.5
center3 off
```

See Also:

`winset, zoomf3`

clear — Remove all plots from the output window(s).

Synopsis:

```
clear
clear all
```

Arguments:

`all`	Flag indicating that all output windows should be cleared.

Description:

The `clear` command clears (removes) all plots from the active window, or from all windows if the `all` argument is provided. In MeshTV, all plot requests are drawn on top of any existing plots. Only an explicit `clear` command or `newplot` command will clear the window.

Note that when the `-nowin` option is used on the command line for production mode, there are no windows per se. In this case, each successive plot command overlays the previous one, and when a `savewin` or `printwin` command is issued, all plots are output. In this case, the `clear` command can be used to create separate plots. Without it, plots made in sequence will be merged together.

Examples:

```
clear
clear all
```

See Also:

`plot, newplot, redraw, winset`

close — Close the SILO file associated with the active output window.

Synopsis:

```
close
```

Arguments:

No arguments.

Description:

`close` closes the active window's SILO file. Note that issuing another `open` or `addframe` command also closes the currently open file.

Examples:

```
close
```

See Also:

`open, winset`

colordef — Redefine a color in the MeshTV standard color table.

Synopsis:

`colordef` *color_number red green blue*

Arguments:

color_number	An integer from 1 to 9 which indicates which color to replace in the MeshTV standard color table. (See APPENDIX B: Color Names.) There are 11 colors in the table, but black and white cannot be modified.
red	The red component of the color. This is a floating point percentage between 0. and 1. If any numbers are larger than 1, all are scaled so that the largest number of the rgb trio becomes one.
green	The green component of the color. This is a floating point percentage between 0. and 1. If any numbers are larger than 1, all are scaled so that the largest number or the rgb trio becomes one.
blue	The blue component of the color. This is a floating point percentage between 0. and 1. If any numbers are larger than 1, all are scaled so that the largest number of the rgb trio becomes one.

Description:

MeshTV provides 9 standard colors. These can be replaced via the `colordef` command. To create a color via an existing X Windows color name, use the `colorname` command.

This command defines a color to be used for material (`bnd`), contour (`iso`), `mesh`, filled material (`smat`), and `vec` plots. The color is comprised of red, green, and blue components, and these are specified via a floating point number between 0 and 1. If numbers larger than one are given, all are scaled so that the largest number becomes one. This means that "`colordef 1 0 0.5 1`" is identical to "`colordef 1 0 10 20`".

Note that if you are using the GUI and you issue this command in the command line interface window, the menus which allow color selection for lines will be drawn with the incorrect colors and color names, if you modify MeshTV's colors. For example, if you change color number 4, which is blue, to red, the menu entry which says "blue" (and which is shown in blue) will actually produce red.

Examples:

```
colordef 1 0.0 0.5 1.0   # Set color 1 to aquamarine.
colordef 2 0 10 20       # Color 2 is the same aquamarine.
```

See Also:

`bnd, colorname, iso, mesh, smat, vec`

colorname — Redefine a color in the MeshTV standard color table.

Synopsis:

`colorname` *color_number x_color*

Arguments:

color_number	An integer from 1 to 9 which indicates which color to replace in the MeshTV color table. (See APPENDIX B: Color Names.) There are 11 colors in the table, but black and white cannot be modified.
x_color	The name of a color as defined by the X Window System™.

Description:

MeshTV provides 9 standard colors. These can be replaced via the `colorname` command. If you want to create a color using red/green/blue color components, use the `colordef` command.

To find a list of colors defined by the X Window System, look for the rgb.txt file in your system's standard X location. On many systems, this is located in /usr/lib/X11.

This command defines a color to be used for material (`bnd`), contour (`iso`), `mesh`, filled material (`smat`), and `vec`, plots.

Note that if you are using the GUI and you issue this command in the command line interface window, the menus which allow color selection for lines will be drawn with the incorrect colors and color names, if you modify MeshTV's colors. For example, if you change color number 4, which is blue, to red, the menu entry which says "blue" (and which is shown in blue) will actually produce red.

Examples:

```
colorname 1 PapayaWhip
colorname 1 "papaya whip"
```

See Also:

`bnd, colordef, iso, mesh, smat, vec`

csh — Execute an operating system (shell) command.

Synopsis:

`csh` *command*
`csh` *“command args”*

Arguments:

command	Any shell command.
args	Arguments to the shell command.

Description:

The `csh` command is an escape to the operating system from within MeshTV. With it, you can enter any command which the operating system understands. Note that if the command has arguments, special characters (like ‘`*`’, ‘`/`’, ‘`&`’), or multiple commands separated by semi-colons, then the entire string must be quoted.

Use this command if you want to issue a UNIX^{TM} command from within MeshTV.

Note that this command is identical to the `sh` command.

Examples:

```
csh "ls -l"
csh "emacs my-command-file &"
csh "rm junk*"
```

See Also:

`listdb, sh`

ct — Set the pseudo-color plot color table for the active window.

Synopsis:

`ct` *ct_name*

Arguments:

ct_name Either the name of a built-in color table, or the name of a palette file on disk. See the table below for built-in color tables.

Built-In Color Table Names	Description
`contoured`	A simple striped color palette containing four bands.
`gray`	A white to black palette.
`hot`	A hot to cold color palette, with blue representing low values and red high values. (Default)
`rainbow`	A rainbow palette.
`xray`	The inverse of the gray palette mentioned above.

Description:

The `ct` command changes the color table which is used by the pseudo-color (`pc`) command for the active window. The built-in color tables listed above are always available. In addition, MeshTV can read raw binary palette files produced by programs such as the National Center for Supercomputing Applications (NCSA) tools. The format for these binary files is 256 sets of 3 bytes where each respective byte in the trio represents red, green, and blue for each of the 256 colors.

Each window has its own color table, although on most systems you will only see one table in use at any one time. By changing the active window, issuing the `ct` command, and then the `reset` command, the palette will change.

Examples:

```
ct gray
ct /usr/local/pals/tim.pal
```

See Also:

`pc, redraw, winset`

defvar — Define a new variable from existing variables.

Synopsis:

`defvar` *varname=expression*
`defvar` *varname*
`defvar`

Arguments:

varname	The name of the new variable being defined.
expression	An expression based on the operators and functions defined in the table below. In general, variables, other defvar variables, functions, operators, and constants can be used in *expression*. A variable referenced within *expression* can reside anywhere within the currently opened SILO file or anywhere within another SILO file. Referencing a variable by its name alone indicates it resides in the current directory of the currently opened SILO file. The complete syntax for referencing a variable is: `silo_pathname:var_pathname` `silo_pathname:` is optional. If specified, it is the UNIX™ pathname to the SILO file in which the variable resides. If `silo_pathname` contains the "`/`" character, then the entire pathname must be enclosed by single quotes so that "`/`" is not taken to be the division operator. The pathname can be relative (`../path`, for example), and ~*username* syntax is allowed. `var_pathname` is the pathname of the variable within the SILO file. If `var_pathname` is just the name of the variable, then it is assumed to reside in the current directory of the SILO file. The current directory when `silo_pathname` is specified is the root or "`/`" directory. If `var_pathname` has the "`/`" character, then the entire pathname must be enclosed by single quotes so that "`/`" is not taken to be the division operator. Because this is a SILO file path, ~*username* syntax is not allowed.

Function or Operator	**Meaning**	**Usage**
`{}`	Associate a list of expressions into a single variable. Usually used for defining a vector variable.	`{`$expr_1$,$expr_2$`[,...]}` $expr_i$ can be an expression of variables, other defvar variables, and constants that evaluate to a mesh variable.

Function or Operator	**Meaning**	**Usage**
`()`	Associative parenthesis.	Group mathematical operations, which are described in the table entry following this one.
+, −, *, /, ^	Addition, subtraction, multiplication, division, and exponentiation operators.	*expr op expr* *expr* can be an expression of variables, other defvar variables, and constants. *op* is one of the operators specified in the Operator column.
`absv`	Absolute value	`absv(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants.
`sqrt`	Square root	`sqrt(`*expr*`)` as with `absv`
`ln`	Natural log	`ln(`*expr*`)` as with `absv`
`log`	Base-10 log	`log(`*expr*`)` as with `absv`
`insel`	Index selection (i.e., reduce the size of the variable using begin, end, skip)	`insel(`*expr*, *ibeg [*`:`*iend [*`:`*iskip]]* *[*`,`*jbeg[*`:`*jend[*`:`*jskip]]]* *[*`,`*kbeg[*`:`*kend[*`:`*kskip]]]*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a quadmesh variable. The options which have "*beg*" in their name represent the beginning index. Options with "*end*" represent the ending index, and option with "*skip*" indicate the number by which to skip. X, Y, and Z are represented by the i, j, and k prefixes. The k prefix options only work in 3D plots.

`oslice`	Orthogonal slice (slice perpendicular to one of the axes)	`oslice(`*expr*`,`
		`ix=`*i* | `iy=`*j* | `iz=`*k* |
		`xx=`*x* | `yy=`*y* | `zz=`*z* |
		`px=`*xpct* | `py=`*ypct* | `pz=`*zpct*`)`
		expr can be an expression of variables, other defvar variables, and constants that evaluate to a mesh variable.
		Slicing can be done based on a node index (the `ix`, `iy`, and `iz` options), on a coordinate value (the `xx`, `yy`, and `zz` options), or by a percent value (the `px`, `py`, and `pz`.) Only one of these options is specified at a time, with the x, y, or z in the second position indicating the axis to slice along.

`aslice`	Arbitrary slice (slice through an arbitrary plane)	`aslice(`*expr*`,` *fff*`,`*fff*`,`*fff*`)` *expr* as in `oslice` The first set of three numbers defines an origin, the second set defines a normal to the plane, and the third set defines the "up" vector.
`reflect`	Reflect about one or more axes.	`reflect(`*expr*`, xmin, xmax, ymin, ymax, zmin, zmax)` *expr* as in `oslice`. One or more of the `xmin`, `xmax`, etc. must be specified. `zmin` and `zmax` are only used for 3D problems.

Description:

The `defvar` command defines new variables which can then be used in the same manner as variables from the data file. With no arguments, `defvar` simply lists all defined variables from the current MeshTV session. With one argument, `defvar` prints the current definition (if any) of the given variable to the terminal.

Examples:

```
defvar                    # Print all definitions
defvar myvar              # Print the definition of myvar

defvar dnew = reflect(d,xmin,ymin)
defvar pnew = oslice(p,xx=10.)
defvar wow = insel(reflect(oslice(d*(u^2 +
          v^2),py=50.),xmin),1:150:2, 1:75)

defvar velocity = {u,v}
defvar momentum = {u,v}*mass
defvar momentum = {u*mass,v*mass}# Same as the previous line
defvar sum = a + (b - c)
defvar factor = 3.1415926/180.
defvar x = factor*sqrt(u^2 + v^2) + .25

defvar dp1 = p - '/quad/p'
defvar dp2 = p - abc.silo:p
defvar dp3 = p - '~/dir1/abc.silo':p
defvar dp4 = p - '~/dir1/abc.silo':'/quad/p'
```

See Also:

delete — Delete plots.

Synopsis:

`delete` *plot_name [, plot_name, ...]*

Arguments:

plot_name	A plot name, like `smat` or `bnd`, followed by a number which indicates the position in the list of plots. For example, if there are two `smat` plots, these are referenced as `smat1` and `smat2`. If a single plot exists, it is still referenced with a number appended, as in `bnd1`. Plot types are: `bnd`, `iso`, `label`, `mesh`, `pc`, `point`, `ribbon`, `smat`, `stream`, `surf`, and `vec`.

Description:

Delete plots of the given plot type. Note that if you want to delete multiple plots of the same type, you must either list the higher numbered plot first, or you must issue the same number. For example, if you have 3 mesh plots, you can type "`delete mesh3 mesh2 mesh1`", or "`delete mesh1 mesh1 mesh1`". This is because plots are renumbered after a deletion — if you have three mesh plots and you delete mesh2, then mesh3 will be renamed to mesh2.

Examples:

```
delete mesh3, mesh2, mesh1
delete bnd1
```

See Also:

`plot`

dist — Set options for plotting a variable vs. distance along a line.

Synopsis:

`dist` *option***=***value [, option***=***value, ...]*

Arguments:

option A distance plot option. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`lc`	line color	Line color. (See APPENDIX B: Color Names.) `off` means the line color will cycle through the MeshTV standard color table	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX C: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`outwin`	`1` <= integer <= `16`	Set the output window into which to display the distance plot. This must be different from the active window.	`2`
`var`	variable name	Name of variable to plot.	`d`

Description:

The `dist` command sets the options used when a MeshTV window is in the distance mode. By default, MeshTV will cycle through the line colors for the distance plot. If you want all plots to be the same color, you should set the line color on the command line. See `winmode` for a description of the distance mode.

The plot which displays in the output window is a plot of the chosen variable (set via the `var` option) as it varies along a line. The line is selected either by specifying endpoints, if using the `distline` command, or by drawing the line by clicking with the mouse in the active window while in the dist mode.

This command should only be issued if you have more than one output window created.

Examples:

```
dist var=d,outwin=2
dist var=foo,outwin=4,lc=red
```

See Also:

`distline, winmode, winnew, winnum`

distline — Plot a variable vs. distance along a line.

Synopsis:

`distline` *window_id* x_1 y_1 x_2 y_2

Arguments:

window_id	The identifer of the window containing the line, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.
x1,y1,x2,y2	The coordinates of the end-points of the line. These are real (floating point) numbers.

Description:

A line with end-points `(x1,y1), (x2,y2)` is plotted in the active output window. A second curve is generated from the first, and it consists of all the values of a given variable at the points along the line. The variable is specified via the `dist` command, and the resultant values are plotted versus the distances of the points from the first end-point. This curve is plotted in the output window specified in the `dist` command.

Examples:

```
dist var=d, outwin=2
distline 1 .3 .5 1.2 2.8  #Value curve goes to output win 2
```

See Also:

`dist, winnew, winnum`

end — Exit from the MeshTV session.

Synopsis:

```
end
```

Arguments:

No arguments.

Description:

The `end` command kills the MeshTV process and removes any output windows created during the session. A summary of the commands issued during the session is kept in the file "`%meshtv.log`" in the directory from which MeshTV was invoked. This log file can be used, after it is renamed, to run MeshTV with the same commands by using the `source` command from within MeshTV, or by invoking MeshTV at the UNIX™ prompt. For example, if you rename `%meshtv.log` to `file.log`, you would type the following "`meshtvx < file.log`". You must rename the file, since MeshTV will write to a file named `$meshtv.log`, and this would overwrite your commands.

Note that if you have issued an `foutput` command, and the information has yet to be written out, this command will force the information to be written.

This command is identical to the `quit` command.

Examples:

```
end
```

See Also:

`quit, source`

foutput — Send all subsequent plots to an output file.

Synopsis:

`foutput` *option=value [, option=value, ...]*

Arguments:

option An option for sending plots. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`banner`	banner string	This is a string to print to the top and bottom of a PostScript™ output file.	`""`
`rootname`	path name	Root name of output file family. Output file family members are named *pathname*0000.*filetype*, *pathname*0001.*filetype*, ...	`meshtv`
`type`	`ps, rgb, rps, tif, xwd`	The file format to save the image in. `ps` — Vector PostScript™, 2D plots only `rgb` — SGI's rgb format `rps` — Raster PostScript™ `tif` — tiff format `xwd` — xwd format	`ps`
`xres`	positive integer	The resolution in the X direction. Only applies to `rgb`, `rps`, `tif`, and `xwd` formats. If you specify `xres`, you must also specify `yres`, and the two numbers should be equal to each other.	`1024`
`yres`	positive integer	The resolution in the Y direction. Only applies to `rgb`, `rps`, `tif`, and `xwd` formats. If you specify `yres`, you must also specify `xres`, and the two numbers should be equal to each other.	`1024`

Description:

The `foutput` command causes all subsequent plots to be sent to a file family in a format specified by `type`. Only one output file family can be active at a time—plots are directed to the file specified by the most recent invocation of `foutput`. Also, invocation of `foutput` causes a new file family to be created; therefore any existing family with the same root name is overwritten.

Plots from all output windows are written to the output file family when a command is invoked that acts like a frame advance for output files; that is, a command which causes the plots in that window to be replotted (`pan3`, `rotx`, `roty`, `rotz`, `vp`, `wp`, `zoom3`) or causes the window to be cleared (`clear`, `end`, `nextframe`, `newplot`, `prevframe`, `quit`). For example, invoking the `wp` command causes the plots to be first written to the output file and then replotted with new physical window limits. Invoking the `plot` command would not cause plots to be written since execution of the command adds a plot to the window rather than causing existing plots to be redrawn or cleared.

For distance plots, two output file frames are generated, the first frame containing the distance lines, the second frame containing the distance curves.

Examples:

```
open prob1.silo
foutput rootname=prob1   # Create prob10000.ps
plot mesh
plot iso
wp 0 3 0 3               # Write mesh, iso frame.
clear                    # Write mesh, iso frame with
                         # new limits.
plot bnd
quit                     # Write bnd frame.
```

See Also:

`clear, plot, wp`

fullframe — Set the aspect ratio of the graphics window.

Synopsis:

```
fullframe on | off
```

Arguments:

`on`	Force the viewport to be square.
`off`	Allow the viewport to conform to the aspect ratio of the data.

Description:

The `fullframe` command sets the value of the fullframe parameter to either “on” or “off”. A value of “on” forces the output viewport to be square, thereby allowing the aspect ratio of the data to be other than 1:1. A value of “off” is the default, and this causes the data aspect ratio to be 1:1. This command is useful when the image being examined is long and narrow, and the viewport is too narrow to see sufficient detail.

Examples:

```
fullframe on
```

See Also:

help — Print a command list or a brief summary of one command.

Synopsis:

`help`
`help` *command*

Arguments:

command	Any MeshTV command keyword.

Description:

The `help` command without arguments lists all MeshTV commands. When the `help` command is given a command keyword as an argument, it prints a brief summary of the requested command.

NOTE: This command has yet to be fully implemented.

Examples:

```
help redraw
help
```

See Also:

history — Print the most recently entered MeshTV commands.

Synopsis:

`history` *[number]*

Arguments:

number	The number of commands to display.

Description:

The `history` command operates much like the UNIX^{TM} C-Shell's history command. Without any arguments, `history` will show the last 20 commands given to MeshTV. The number of commands shown can be changed using the optional *number* argument.

To facilitate re-execution, the sequence number is printed beside the command. To reissue a command, use the `!` command, as in the UNIX^{TM} C-Shell.

Examples:

```
history 10
history
```

See Also:

`!`

inq — Inquire about the current SILO file attributes.

Synopsis:

```
inq
```

Arguments:

No arguments.

Description:

The `inq` command prints out the attributes of the objects in the current directory of the current SILO file. Among the information provided are mesh dimensions, number of materials, time, and cycle.

Examples:

```
inq
```

See Also:

`show, SILO User's Guide`

iso — Set plotting options for the contour line/surface plot.

Synopsis:

`iso` *option=value [,option=value, ...]*

Arguments:

option A plot option for the contour plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`err`	real number `>= 0.0`		`0.0`
`lc`	line color	Line color. (See APPENDIX B: Color Names.) `off` means the line color will cycle through the MeshTV standard color table	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX C: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`level`	list of real numbers	Space-delimited set of numbers to use as actual iso level values. This conflicts with the `pct` and `nlevels` options, as explained in the Description section below.	`No default`
`pct`	iso percent levels	Space-delimited set of percentages to use for calculating iso levels. This conflicts with the `level` and `nlevels` options, as explained in the Description section below.	`No default`
`pop`	See `pop` option on page 69.		`off`
`nlevels`	positive integer	Integer number of levels, which will range uniformly from data min to data max of the chosen variable. This conflicts with the `level` and `pct` options, as explained in the Description section below.	`7`
`var`	variable name	Name of variable to plot.	`d`
`shadevar`	3D shade variable name, or “off”	Name of variable to shade surface with. Only for 3D. This has yet to be implemented.	`off`

Description:

The `iso` command sets the options used when generating a contour (i.e., iso) plot. Note that this command does not change existing iso plots, only future ones. If plotting 2D data, the iso plot creates contour lines. If plotting 3D data, the iso plot creates contour surfaces.

The levels to draw are set by either the `level`, `pct`, or `nlevels` option. If more than one of these options is set, the last option “wins”. The `nelevels` option is the default method for level selection.

To find the data value range of a variable, see the `minmax` command.

Examples:

```
iso lc=green,var=p, pct=10 25 50 75 90
iso level=0.5 1.0 2.0 5.0
```

See Also:

```
colordef, colorname, minmax, plot, pc
```

label — Set plotting options for labeling nodes and/or zones.

Synopsis:

`label` *option=value [, option=value, ...]*

Arguments:

option	A plot option for the label plot. See table below.
value	A value associated with the matching *option*. See table below.

Option	**Value**	**Value Meaning**	**Default Value**
`hjust`	`l, r, c`	Horizontal justification of labels with respect to labelled objects. `l` — left-justified. `r` — right-justified. `c` — centered.	`c`
`mesh`	mesh name	Name of mesh to label.	`physics_mesh`
`node`	`all`, one integer, two integers, three integers.	Label all nodes if “all” is the argument, else label the listed node. One integer lists a specific node. 2 numbers indicates (on collinear meshes) a logical x,y index, and 3 indicates (again only on collinear meshes) a logical x,y,z index. See the description for more detail. Note that the `type` option must have an “n” in it for this to actually plot.	`all`
`pop`	See `pop` option on page 69.		`off`
`string`	string	The string with which to label nodes and zones if labeling a single element.	`*`
`tc`	text color	Color of labels. (See APPENDIX B: Color Names.)	`foreground color`
`type`	`n, z, nz, zn`	Type of mesh object to label. `n` — label nodes. `z` — label zones. `nz`, `zn` — label both. Note that this must be set for `node` and `zone` to take effect.	`n`

Option	Value	Value Meaning	Default Value
`tht`	`0.0` < real number <= `1.0`	Height of labels.	`.025`
`var`	variable name	Name of variable whose values will be used as the labels.	`No default`
`vjust`	`b, c, t`	Vertical justification of labels with respect to labelled object. `b` — bottom-justified. `c` — centered. `t` — top-justified.	`c`
`xoffset`	x offset from the point	Place the label at some fraction of the screen width away from the point, either negative or positive.	`0.0`
`yoffset`	y offset from the point	Place the label at some fraction of the screen width away from the point, either negative or positive.	`0.0`
`zone`	`all, one integer, two integers, three integers.`	Label all zones if "all" is the argument, else label the listed zones. One integer lists a specific zone. 2 numbers indicates (on collinear meshes) a logical x,y index, and 3 indicates a logical x,y,z index. See the Description section for more detail. Note that the `type` option must have a "z" in it for this to actually plot.	`No default`

Description:

The `label` command sets the options used when generating a mesh label plot. Note that this command does not change existing label plots, only future ones.

The `type` option indicates if we are labeling nodes, zones, or both. If "n" is chosen, then `node` must also have some value. It already has a default of "all". In contrast, if "z" is chosen for the value of type, then `zone` must also be set to something, and it starts without a default. If `type` is set to "n", then any value for `zone` is ignored, and if `type` is set to "z", any value for `node` is ignored. If `type` is set to "nz" or "zn", then both `node` and `zone` must have values.

All meshes supported by MeshTV can be labeled, but the node and zone options take different values depending on the mesh type. The values for `node` and `zone` options can be the word "all" for all mesh types. The values can also be, for certain meshes, a set of numbers. All mesh types will allow one number. If one number is given, it is assumed to be the number of the node or zone to label, starting at whatever offset (0 or 1) that was given to the data file. Quadrilateral meshes may also be given 2 or 3 numbers, depending on the dimensionality (2D or 3D) of the mesh. In the 2D case, the numbers are interpreted as offsets into the mesh, so {1,2} means one node/zone in the “x” direction, and two in the “y” direction. The 3D case is the same, with an extension in the “z” direction. The actual direction might not be exactly parallel to the “x”, “y”, or “z” axes in non-colinear problems, but it is conceptually along those axes. Note that if 2 or 3 numbers are supplied to the node or zone options, the numbers must be inside braces, as shown in the first example below.

Examples:

```
label type=nz, mesh=mesh1, tc=red, node=all, zone={1,2}
label type=z, mesh=mesh1, tht=.02, hjust=l, vjust=b zone=2
```

See Also:

```
mesh, plot
```

latitude — Set the 3D viewing parameter, latitude.

Synopsis:

`latitude` *number*

Arguments:

number	Any negative, zero, or positive real (floating point) angle in degrees.

Description:

The `latitude` command sets the viewing parameter which controls the angular distance positive or negative from the xy plane at z=0. When used in conjunction with the `longitude` command, this command allows you to set the viewing location around a 3D object. The viewing location is the place where your "eye" will be positioned.

Examples:

```
latitude 45
latitude -50.5
```

See Also:

`longitude, rotx, roty, rotz`

legend — Set the status and attributes of the various legends, which annotate plots.

Synopsis:

`legend` *legend_name* = `on`|`off` *[, legend_name* = `on`|`off`*,...]*
`legend` *legend_att = value [, legend_att = value, ...]*

Arguments:

legend_name One of the legends described in the Legend_Name table below.

legend_att One of the attributes described in the Legend_Att table below.

Legend_Name	**Meaning**
`all`	Plot all legends
`bnd`	Material boundary plot legend
`db`	Database time and cycle legend
`dist`	Distance plot legend
`iso`	Iso plot legend
`materials`	List of material numbers in a plot
`matspecies`	List of material species numbers in a plot
`mesh`	Mesh plot legend
`pc`	Pseudo color plot legend
`point`	Point plot legend
`refg`	Reference grid
`ribbon`	Ribbon plot legend
`smat`	Solid-filled material plot legend
`stream`	Stream plot legend
`surf`	Surface plot legend
`time`	Current time, date, and user legend
`vec`	Vector plot legend

Legend_Att	**Value**	**Value Meaning**	**Default Value**
`refgcolor`	line color	Grid line color. (See APPENDIX B: Color Names.) Not implemented.	`No default`

Legend_Att	Value	Value Meaning	Default Value
`refggridax`	`x, y, xy, off`	Grid axes on which to draw major grid lines. `x` — grid lines along x-axis. `y` — grid lines along y-axis. `xy` — grid lines along both. `off` — turn off grid lines.	`off`
`refglab`	`x, y, xy, off`	Grid axes on which to place tick labels. `x` — tick labels along x-axis. `y` — tick labels along y-axis. `xy` — tick labels along both. `off` — turn off tick labels.	`xy`
`refgtickax`	`b, l, bl, bltr, off`	Grid axes on which to place ticks. `b` — bottom. `l` — left. `bl` — bottom left. `bltr` — bottom left top right. `off` — turn off tick labels.	`bl`
`refgtickloc`	`out, in, outin`	Location of ticks relative to grid axes.	`in`

Description:

The `legend` command controls which legends are plotted when annotating a display, and it controls the appearance of the legends. *legend_name* and *legend_att* settings can both be invoked within a single command. A *legend_name* value of `on` means plot the legend. (On is the default for all *legend_name* values.) Note that this command does not effect existing annotations, only future ones.

Examples:

```
legend all=off
legend all=on, iso=off, bnd=off
legent refgtickloc=out, refgtickax=bltr, surf=off
```

See Also:

`bnd, dist, iso, materials, mesh, pc, point, ribbon, smat, stream, surf, vec`

lightsrc — Define lighting source properties for 3D plots.

Synopsis:

`lightsrc` *lightsrc_id option value [option value...]*

Arguments:

lightsrc_id	An integer ranging from 1 to 8 denoting which light source to define.
option	One of the names of an area property. See table below.
value	Values associated with the matching *option*. See table below.

Options	Values	Value Meaning
`ambient`	*color*	Ambient lighting from light source of color *color.* (See APPENDIX B: Color Names.) Ambient light is light that enters the lighting calculation independent of the orientation of the surface being illuminated or the location of the eye. This option can not be used with infinite, positional, or spotlight.
`infinite`	*color x y z*	Infinite light source of color *color* from point (*x*,*y*,*z*) (See APPENDIX B: Color Names.) This option can not be used with ambient, positional, or spotlight.
`off`		Turn light source off.
`on`		Turn light source on.
`positional`	*color x y z attencoef1 attencoef2*	Positional light source of color *color* from point (*x*,*y*,*z*) with attenuation coefficients *attencoef1*, *attencoef2*. (See APPENDIX B: Color Names.) This option can not be used with ambient, infinite, or spotlight.
`spotlight`	*color x y z attencoef1 attencoef2 concexp spreadangle*	Spotlight light source of color *color* from point (*x*,*y*,*z*) with attenuation coefficients *attencoef1*, *attencoef2,* concentration exponent *concexp*, and spread angle *spreadangle*.(See APPENDIX B: Color Names.) This option can not be used with ambient, infinite, or positional.

Description:

Define light source properties for 3D plots. There are two lights set by default. They are both infinite and are located at (1,1,1) and (-1,-1,-1).

Note that ambient, infinite, positional and spotlight are mutually exclusive, so a light cannot be both infinite and ambient, for example. See the MeshTV Getting Started Manual for a discussion about lighting properties.

Examples:

```
lightsrc 1 ambient red
```

See Also:

area, MeshTV Getting Started Manual

listdb — List all SILO files in the specified directory.

Synopsis:

```
listdb
listdb pathname
```

Arguments:

pathname	Any absolute or relative directory path specification.

Description:

With no arguments, the `listdb` command lists all SILO files in the current working directory. If an argument is given, `listdb` lists all SILO files in the specified directory. Files of other types are not shown. Use this command to see which files are SILO files.

Examples:

```
listdb
listdb ../data
```

See Also:

`csh, sh`

longitude — Set the 3D viewing parameter, longitude.

Synopsis:

`longitude` *number*

Arguments:

number	Any negative, zero, or positive real (floating point) angle in degrees.

Description:

The `longitude` command sets the viewing parameter which controls the angular distance positive or negative from the zero meridian. When used in conjunction with the `latitude` command, this command allows you to set the viewing location around a 3D object. The viewing location is the place where your "eye" will be positioned.

Examples:

```
longitude 23.5
longitude -45
```

See Also:

`latitude, rotx, roty, rotz`

ls — List the contents of a directory within the current SILO file.

Synopsis:

`ls` *["options"] [pathname_list]*

Arguments:

options	A letter or string of letters preceded by a dash and encased in quotes. Default is "-`dmv`". See table below.
pathname_list	One or more pathnames to list. These can be relative or absolute.

Option	**Meaning**
`-a`	List everything in file. Activates the -c, -d, -m, -r, and -v flags.
`-c`	List curves.
`-d`	List directories.
`-l`	Produce a long list containing additional information.
`-m`	List meshes (both point, quad, and UCD mesh objects. (See SILO User's Guide, UCRL-MA-118751.).
`-r`	List material (region) information.
`-v`	List all quadvars, ucdvars, pointvars, and miscellaneous objects. See the SILO User's Guide for more information about these object types

Description:

The `ls` command operates on a SILO file in the same manner than the UNIX™ ls command operates on the UNIX™ file system. The user has control via the command options over which types of objects are listed. Without any options, `ls` lists directories, meshes, and mesh variables. The `ls` command understands both absolute and relative pathnames, hence the path '`..`' means to list the contents of the parent directory. Without any path arguments, `ls` lists the contents of the current SILO directory. Because this is a SILO file path and not a UNIX™ file path, ~*username* syntax is not allowed.

When a file is opened, the active directory is the root directory, indicated by a slash "`/`". By using the `ls` and `cd` commands, the user can navigate through a SILO file.

Examples:

```
ls
ls "-v" ..
ls ../other_dir
```

See Also:

`cd, close, open, pwd, SILO User's Guide`

matcol — Assign colors to materials.

Synopsis:

`matcol` *material# color [, material# color, ...]*

Arguments:

material#	An integer representing a material.
color	One of the predefined or user-specified color names from the MeshTV standard color table. (See APPENDIX B: Color Names.)

Description:

The `matcol` command assigns colors to materials. The `bnd` plot uses these values to determine the color for each material interface, and the `smat` plot uses these values for coloring each material.

By default, colors are assigned to materials by cycling through the MeshTV standard color table. Material 1 receives color 1 (red). Material 9 receives color 9 (grey) and then material 10 receives color 1 (red) again. Use the `reset matcol` command to reset colors to their default (cycling color table) values. Note that if the pre-defined colors have been modified, you will get the modified colors.

`matcol` affects only future plots.

Examples:

```
matcol 1 red, 3 white, 5 black
matcol 2 blue
```

See Also:

`bnd, colordef, colorname, materials, smat, reset`

materials — Set the active material numbers.

Synopsis:

`materials` *material [, [begin*`:`*end* | *material], ...]*
`materials` *begin*`:`*end [, [begin*`:`*end* | *material], ...]*
`materials all`

Arguments:

material	An integer representing a material identifier.
begin	An integer number representing the beginning of a range.
end	An integer number representing the end of a range.
`all`	Flag indicating that all materials should be active.

Description:

The `materials` command takes a comma-separated list of material numbers and/or ranges, where a range consists of a begin-end pair.

Examples:

```
materials 1,3,5:10       # Activate 1,3,5,6,7,8,9, and 10
materials 44             # Activate just 44
materials all            # Activate all materials
```

See Also:

`matcol, matspecies`

matspecies — Turn material species on and off.

Synopsis:

`matspecies` *material begin*`:`*end* | *species [, material begin*`:`*end* | *species...]*
`matspecies all`

Arguments:

material	An integer representing a material identifier.
begin	An integer number representing the beginning of a range of material species.
end	An integer number representing the end of a range of material species.
species	An integer number representing a material species.
`all`	Flag indicating that all materials and species should be active.

Description:

The `matspecies` command takes a comma-separated list of material numbers and/or ranges, where a range consists of a begin-end pair.

Examples:

```
matspecies 1 5:10        # Activate species 5-10 for mat 1
matspecies 2 6           # Activate just species 6 for mat 2
matspecies all           # Activate all material species
```

See Also:

`matcol, materials`

mesh — Set plotting options for the mesh plot.

Synopsis:

`mesh` *option=value [,option=value, ...]*

Arguments:

option A plot option for the mesh plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`err`	`0.0` <= real <= `1.0`	Error tolerance for mesh decimation. This allows the user to remove polygons for faster display. The number indicates how far away from coplanar two polygons can be before they are no longer eligible to be collapsed into one polygon. A value of 0.0 will remove everything, while a value of 1.0 will touch nothing. This option only works if `type` is 1.	`0.7`
`lc`	color	Line color or off. (See APPENDIX B: Color Names.)	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX C: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`pop`	See `pop` option on page 69.		`off`
`type`	`0` or `1`	Whether the mesh should use mesh decimation. 0 indicates no decimation, and 1 indicates decimation.	`0`
`var`	variable name	Name of variable to get mesh data from.	`hydro_mesh`

Description:

The `mesh` command sets the options used when generating a mesh plot. Note that this command does not change any existing plots, only future ones.

Examples:

```
mesh ls=dash
mesh var=hydro, lc=red
```

See Also:

`defvar`, `plot`

minmax — Print a variable’s minimum and maximum data values.

Synopsis:

`minmax` *varname*

Arguments:

varname	The name of the variable to examine.

Description:

The `minmax` command prints the minimum and maximum data values of the requested variable to MeshTV’s text window. Use this when you want to set the range for the pseudo color (`pc`) plot, or when setting contour (`iso`) levels. Note that if you are using blocks, this will print the minimum and maximum data values only for blocks which are active.

Examples:

```
minmax d
```

See Also:

`block, defvar, iso, ls, pc`

nextframe — Plot the next frame in the current animation.

Synopsis:

```
nextframe
```

Arguments:

No arguments.

Description:

The `nextframe` command plots the next frame in the current animation. An animation is built through the use of the `open`, `addframe`, and `plot` commands.

Examples:

```
nextframe
```

See Also:

`addframe, animate, open, play, plot, prevframe, setframe, stop`

newplot — Clear the screen when the next plot is drawn.

Synopsis:

```
newplot
```

Arguments:

No arguments.

Description:

The `newplot` command clears the current output window, but not until the next plot is ready to be drawn. Unlike the `clear` command, which clears the output window immediately, the `newplot` command is useful for making a smooth transition from one plot to the next.

Examples:

```
newplot
```

See Also:

`clear`

open — Open a SILO file for reading.

Synopsis:

`open` *filename*

Arguments:

filename	A filename, including either absolute or relative path specifications.

Description:

The `open` command opens a SILO file and associates it with the current MeshTV output window. All successive plots produced in the current output window will obtain data from the specified SILO file.

A SILO file is typically generated by a physics simulation application, and can contain physics meshes, variables associated with those meshes, subdirectories, curves, material data, and so on.

Previously opened SILO files are closed when a new `open` command is issued. To open a file without closing previously opened files, use the `addframe` command.

Examples:

```
open abc.silo
open /usr/people/sample/abc.silo
open ../data/abc.silo
open ~/data/abc.silo
```

See Also:

`addframe, close, listdb, SILO User's Guide`

pan3 — Perform a relative pan 3D viewing operation.

Synopsis:

`pan3` *xpan ypan*

Arguments:

xpan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the X direction.
ypan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the Y direction.

Description:

The `pan3` command performs a pan operation which is relative to the current view. That is, "`pan3 .1 .2`" would move the image 10% of the window width to the right and 20% of the window height upward. Successive `pan3` commands with these same arguments would continue to pan the image. This is in contrast to the `panf3` command, which performs an absolute pan operation.

Examples:

```
pan3 .5 0.
pan3 -.2 .2
```

See Also:

`center3, latitude, longitude, panf3, zoom3, zoomf3`

panf3 — Perform an absolute pan 3D viewing operation.

Synopsis:

`panf3` *xpan ypan*

Arguments:

xpan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the X direction.
ypan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the Y direction.

Description:

The `panf3` (pan factor 3D) command performs a pan operation which is absolute, based on the original image. For example, "`panf3 .1 .2`" would move the original image 10% of the window width to the right and 20% of the window height upward. Successive `panf3` commands with the same arguments would have no effect. This is in contrast to the `pan3` command, which performs a relative pan operation.

Examples:

```
panf3 .5 0.
panf3 -.2 .2
```

See Also:

`center3, latitude, longitude, pan3, zoom3, zoomf3`

pause — Pause the execution of MeshTV and wait for a carriage-return.

Synopsis:

```
pause
```

Arguments:

No arguments.

Description:

The `pause` command pauses the execution of MeshTV. On most systems, MeshTV will wait until the user presses the Enter (Return) key. Use this to temporarily pause execution when you run MeshTV from a command input file.

Examples:

```
pause
```

See Also:

`source`

pc — Set plotting options for the pseudo-color contour plot.

Synopsis:

`pc` *option=value [,option=value, ...]*

Arguments:

option — A plot option for the pseudo color plot. See table below.

value — A value associated with the matching *option*. See table below.

Option	**Value**	**Value Meaning**	**Default Value**
`pcmin`	real number, `off`	This value is used for shading zones with the given variable (in `var`). Points at or below this minimum value will be shaded with the minimum value color. `off` indicates the actual data minimum will be used.	`off`
`pcmax`	real number, `off`	This value is used for shading zones with the given variable (in `var`). Points at or above this maximum value will be shaded with the maximum value color. `off` indicates the actual data minimum will be used.	`off`
`pop`	See `pop` option on page 69.		`off`
`type`	`cell, zone, zonesmooth`	Rendering type. `Zone` will color a zone a single color. `zonesmooth` assigns colors at the nodes and then interpolates the color between the nodes. `cell` resamples the data onto a regular mesh. `cell` has yet to be implemented.	`zonesmooth`
`var`	variable name	Name of variable to plot.	`d`

Description:

The `pc` command sets the options used when generating a pseudo-color contour plot. Note that this command does not change any existing plots, only future ones.

To determine the data value range, see the `minmax` command.

Examples:

```
pc var=d, type=zone
pc pcmin=-10.5
pc pcmin=1.0e-05, pcmax=1.0e+5
```

See Also:

`ct, iso, minmax, plot`

pick — Set options for pick & query mode.

Synopsis:

`pick` *option=value [, option=value, ...]*

Arguments:

option	A plot option for the pick plot. See table below.
value	A value associated with the matching *option*. See table below.

Option	**Value**	**Value Meaning**	**Default Value**
`pickout`	filename	A file to which to write the information which would normally be printed to the standard output. This has yet to be implemented.	`No default`
`tc`	text color	Text color. (See APPENDIX B: Color Names.)	`black`
`var`	variable name	Name of variable to print information about.	`d`

Description:

The `pick` command sets the options for when MeshTV's output window is in pick mode. This mode allows the user to get information about points by selecting points on the screen using the mouse. Node numbers (for UCD meshes), zone indices (for quad meshes), and the value of the requested variable, materials, or matspecies will be printed to the text (shell) window. (If you issue this command from the GUI, the output will go to the MeshTV Printer Icon Output window.). The mouse selections must be made within the current output window, so this command cannot be used in production mode. See the `winmode` command for more info on the pick mode.

Picking only operates on 2D plots. To do pick & query on 3D data, you must first slice the data with `aslice` or `oslice` (`See defvar`.) to get a 2D plot.

Each pick location is marked with a character which corresponds to one line of output. The default color for the pick character is black, but the color can be changed via the `tc` option.

This command also sets options for the `pickpt` command, which can be used without switching to pick mode.

Examples:

```
pick var=d
pick var=speed, tc=white
```

See Also:

`defvar`, `pickpt`, `winmode`, `winset`

pickpt — Pick information for a variable at a point.

Synopsis:

`pickpt` *window_id x y*

Arguments:

window_id	The identifer of the window containing the point, an integer ranging from 1 to the number of active windows. The maximum number is 16.
x, y	The coordinates of the point. These are floating point numbers.

Description:

`pickpt` prints the coordinates, node numbers (for UCD meshes), zone indices (for quad meshes), and the value of the requested variable, materials, or matspecies at the point (`x`,`y`) in output window *window_id*. This information is printed to the text (shell) window, unless the command is issued from the GUI, in which case the information is printed to the MeshTV Printer Icon Output window. The command also labels the point. See the description of the `pick` command for details on point labeling.

Examples:

```
pick var=d, tc=white
pickpt 1 4.3 5.6
```

See Also:

`pick, winnew, winnum`

play — Play the current animation.

Synopsis:

```
play
```

Arguments:

No arguments.

Description:

The `play` command commences playback of the current animation. Animations are created through the use of the `open`, `addframe`, and `plot` commands.

Examples:

```
play
```

See Also:

`addframe, animate, nextframe, open, prevframe, setframe, stop`

plot — Generate a plot from the current data file.

Synopsis:

`plot` *plot_list*

Arguments:

plot_list A comma-separated list of plot names. See table below.

Plot Name	Plot Description
`bnd`	Material boundary lines (for 2D data) or material boundary surfaces (for 3D data). Each line/surface is assigned a color based on its material number.
`iso`	Displays lines or surfaces of constant value. Contour lines are generated from 2D data and contour surfaces are generated from 3D data.
`label`	Displays node and/or zone numbers at the positions of a mesh's nodes and/or zones.
`mesh`	Displays the mesh.
`pc`	Displays a pseudo-colored plot. Each zone or node is assigned a color, and the entire mesh is colored according to the current pseudo-color color table. (See `ct`.) For 3D data, only the external faces are plotted.
`point`	Displays a set of points. The points can be shaded based on the value of some scalar data.
`ribbon`	Displays flow ribbons through a 3D vector field. Only works in 3D.
`smat`	Displays material boundaries in a solid-filled representation. Each zone is assigned a color based on its material number.
`stream`	Display the path of particles in a vector field.
`surf`	Displays a 3D surface of 2D data. The X and Y components are taken from the mesh, and the data variable is used as the Z component (height).
`vec`	Displays vectors based on the data variable. The length of the arrow corresponds to the magnitude of the vector, and the direction of the arrow corresponds to the direction of the vector.

Description:

The `plot` command generates the requested types of plots from the current data file. The new plots will be drawn on top of any existing plots already on the screen. If you want the plot to replace the contents of the window, first issue a `clear` command or a `newplot` command. For more information, see the command summary for each plot name.

Examples:

```
plot smat
plot iso, bnd
```

See Also:

`bnd, clear, delete, iso, label, mesh, newplot, pc, point, reset, ribbon, smat, stream, surf, vec`

point — Set plotting options for the point plot.

Synopsis:

`point` *option=value [,option=value, ...]*

Arguments:

option	A plot option for the point plot. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`lc`	marker color	Marker color. (See APPENDIX B: Color Names.)	`white`
`pcmin`	real number, `off`	This value is used for shading zones with the given variable (in `var`). Points at or below this minimum value will be shaded with the minimum value color. `off` indicates the actual data minimum will be used.	`off`
`pcmax`	real number, `off`	This value is used for shading zones with the given variable (in `var`). Points at or above this maximum value will be shaded with the maximum value color. `off` indicates the actual data minimum will be used.	`off`
`pop`	See `pop` option on page 69.		`off`
`scale`	positive real number	Number to be used as scale factor to change the size of the marker.	`1.0`
`var`	variable name	Name of variable to get point data from.	`No default`

Description:

The `point` command sets the options used when generating a point plot. Note that this command does not change any existing plots, only future ones.

To determine the data value range, see the `minmax` command.

Examples:

```
point var=radius, lc=green
point var=pmesh, pcmin=0.5, pcmax=100.
```

See Also:

`minmax, plot`

pop — Plot operation option to various commands.

Synopsis:

`pop=off`
`pop=`*option(args)*

Arguments:

`off`	This is the default.

option	A plot operation option. There are two choices, `clipp` and `reflectp`. The format for each option is:
	`clipp(`*longitude, latitude, radius*`)`
	`reflectp(xmin, xmax, ymin, ymax, zmin, zmax)`
	Any combination of the 6 possible arguments (`xmin`, `xmax`, etc.) is valid so long as there is at least one argument.

args	Arguments to the matching operation option.

For the `clipp` option, these are:

longitude	Longitude relative to object
latitude	Latitude relative to object
radius	Radius relative to object

For the `reflectp` option, the values are:

`xmin`, `xmax`	X line of reflection
`ymin`, `ymax`	Y line of reflection
`zmin`, `zmax`	Z line of reflection

`zmin` and `zmax` are only used for 3D plots. At least one argument is required, though all can be used.

Description:

Plot operations are operations that are performed on the geometry generated by the plots. Only one plot operator may be applied at a time.

The clipp operator allows the user to "prune" parts of a generated plot by defining a clipping plane. The normal of the plane is specified with a longitude, latitude, and radius, and points which lie to that side of the plane will be plotted.

The `reflectp` operator reflects the plot about the upper and lower boundaries along the various spacial directions. The user can independently control whether a reflection is done about a particular boundary. If a reflection is performed about both the upper and lower boundaries of a particular spacial direction, then the reflection about the upper boundary is done first. In addition, performing reflections about both the upper and lower boundaries results in 4 times the area/volume (not 3).

Examples:

```
bnd pop=clipp(0.0, 0.0, 1.0, -4.0)
iso pop=reflectp(xmin, xmax, ymin)
```

See Also:

```
defvar
```

prevframe — Plot the previous frame in the current animation.

Synopsis:

```
prevframe
```

Arguments:

No arguments.

Description:

The `prevframe` command plots the previous frame in the current animation. An animation is built through the use of the `open`, `addframe`, and `plot` commands.

Examples:

```
prevframe
```

See Also:

`addframe, animate, open, play, nextframe, setframe, stop`

printwin — Print the current output window on a network printer.

Synopsis:

```
printwin
```

`printwin` *option=value [,option=value, ...]*

Arguments:

option An option for printing images. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`banner`	banner string	This is a string to print to the top and bottom of a PostScript™ output file.	`""`
`capture`	`on, off`	Whether or not to get the image to save by doing a screen capture.	`on`
`invert`	`on, off`	A flag indicating if foreground and background colors should be inverted.	`off`
`printer`	printer name	The name of the printer to send the image to for printing.	`colorps`
`type`	`ps, t93, vps`	The type of the printer the image is being sent to for printing. `ps` — Raster PostScript™ `t93` — Tektronix™ t93 raster `vps` — Vector PostScript™, 2D plots only	`ps`
`xres`	positive integer	The resolution in the X direction. Only used when `capture` is off, and only applies to `ps` and `t93` formats. If you specify `xres`, you must also specify `yres`, and the two numbers should be equal to each other.	`1024`
`yres`	positive integer	The resolution in the Y direction. Only used when `capture` is off, and only applies to `ps` and `t93` formats. If you specify `yres`, you must also specify `xres`, and the two numbers should be equal to each other.	`1024`

Description:

The `printwin` command without any options prints the contents of the current output window to the network printer. With options, the `printwin` command sets parameters for future window prints, *but does not actually print the window.*

The types of printers supported are `ps` for PostScript™, and `t93` for Tektronix™ t93 raster printers.

Note that if the `capture` flag is on, the picture will be "grabbed" via a screen capture. This means that if you open another window on top of your output window and then issue the `printwin` command, you will see the second window in your printed image. The default for `capture` is `on`, because this is the quickest way to get the picture. If the capture flag is off, the picture is generated internally at 24 bits at the specified resolution.

Examples:

```
printwin printer=color6
printwin invert=on, printer=mycolor
printwin
```

See Also:

`savewin`

pwd — Print the name of the current directory in the active SILO file.

Synopsis:

`pwd`

Arguments:

No arguments.

Description:

The `pwd` command is analogous to the UNIX^{TM} pwd command, except that it shows the current directory name within the active SILO file instead of within the UNIX^{TM} file system.

Examples:

```
pwd
```

See Also:

`cd, close, csh, ls, open, sh`

quit — Exit from the MeshTV session.

Synopsis:

```
quit
```

Arguments:

No arguments.

Description:

The `quit` command kills the MeshTV process and removes any output windows created during the session. A summary of the commands issued during the session is kept in the file "`%meshtv.log`" in the directory from which MeshTV was invoked. This log file can be used, after it is renamed, to run MeshTV with the same commands by using the `source` command from within MeshTV, or by invoking MeshTV at the UNIX^{TM} prompt. For example, if you rename `%meshtv.log` to `file.log`, you would type the following "`meshtvx < file.log`".You must rename the file, since MeshTV will write try to write to a file named `$meshtv.log`, and this would overwrite your commands.

Note that if you have issued an `foutput` command, and the information has yet to be written out, this command will force the information to be written.

This command is identical to the `end` command.

Examples:

```
quit
```

See Also:

`end, source`

rect — Set options for plotting the integral of a variable within a rectangular box as a function of distance along the length.

Synopsis:

`rect` *option***=***value [, option***=***value, ...]*

Arguments:

option	A rect plot option. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`lc`	line color	Line color. (See APPENDIX B: Color Names.) `off` means the line color will cycle through the MeshTV standard color table	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX C: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`outwin`	`1` <= integer <= `16`	Set the output window into which to display the distance plot. This must be different from the active window.	`2`
`var`	variable name	Name of variable to plot.	`d`

Description:

The `rect` command sets the options used when the `rectline` command is issued. By default, MeshTV will cycle through the line colors for the rect plot. If you want all plots to be the same color, you should set the line color on the command line.

The plot which displays in the output window is a plot of the chosen variable (set via the `var` option) as it varies along a line. The line is selected by specifying endpoints using the `rectline` command.

This command should only be issued if you have more than one output window created.

This command only works with rectilinear meshes.

Examples:

```
rect var=d,outwin=2
rect var=foo,outwin=4,lc=red
```

See Also:

`rectline, winnew, winnum`

rectline — Plot the integral of a variable within a rectangular box as a function of distance along the length.

Synopsis:

`rectline` *window_id* x_1 y_1 x_2 y_2

Arguments:

window_id	The identifer of the window containing the box, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.
x1,y1,x2,y2	The coordinates of two opposing points of a box. These are real (floating point) numbers.

Description:

A curve is generated from the values of a given variable, integrated in strips through the box. The variable is specified via the `rect` command, and the resultant values are plotted versus the distances along the box. This curve is plotted in the output window specified in the `rect` command.

Examples:

```
rect var=d, outwin=2
rectline 1 .3 .5 1.2 2.8  #Value curve goes to output win 2
```

See Also:

`rect, winnew, winnum`

redraw — Redraw plots.

Synopsis:

```
redraw
redraw all
```

Arguments:

`all`	Redraw all output windows.

Description:

Redraw the plots in the active output window. `all` causes the plots in all windows to be redrawn. Calling `redraw` without arguments causes plots in the currently active output window to be redrawn.

Issue this command to redraw plots if they have become garbled on the screen. This command will also force some changes which apply to future plots (like changes to light sources or color tables) to apply to the current plot.

Examples:

```
redraw all
```

See Also:

reflect — Set the light reflectance method for the active output window.

Synopsis:

`reflect` *method*

Arguments:

method	The reflectance method: "`ambient`", "`diffuse`", "`specular`", or "`none`".

Description:

The reflectance method for the light source model is set to `method` for the active output window. The reflectance method determines the visual appearance of reflected light.

`ambient` adds reflected light from everywhere. It provides a kind of "background light".

`diffuse` provides a matte or "dull" finish. This is a good method for objects which have a dull surface. When an object uses this method, it is scattering light equally in all directions.

`specular` produces a highlight on the object. This is a good method for shiny objects. When an object uses this method, light is reflected where the angle of incidence equals the angle of reflection, and this produces the highlight.

`none` ignores the reflectance method, or turns off a previously selected method.

Examples:

```
reflect diffuse
```

See Also:

`lightsrc`

reset — Reset options to their default values.

Synopsis:

`reset` *[option_list]*

Arguments:

option_list A comma-separated list of options which should be reset. If no list is provided, *all* options are reset.

Option Name	Meaning
`all`	Reset all options.
`bnd`	Boundary plot options.
`ct`	Color table (palette).
`dist`	Distance command options.
`iso`	Iso plot options.
`label`	Label plot options.
`legend`	Legend options.
`matcol`	Material to color correspondence.
`material`	Material options.
`mesh`	Mesh plot options.
`pc`	Pseudo color plot attributes.
`point`	Point plot options.
`rect`	Rect command options.
`ribbon`	Ribbon plot options.
`smat`	Solid-filled material plot options.
`stream`	Stream plot options.
`surf`	Surface plot options.
`vec`	Vector plot options.
`view`	Viewing parameters (e.g., window, up-vector, etc.).
`vp`	Viewport (e.g., xmin, xmax, ymin, ymax).
`wp`	Physical window (e.g., xmin, xmax, ymin, ymax).

Description:

The `reset` command resets plotting, window, and display options back to their default values. This is a convenient way to reset plotting options (such as iso levels) or display parameters (such as the viewport) which have been changed.

Examples:

```
reset view, iso
```

See Also:

`bnd, ct, dist, iso, label, legend, matcol, materials, mesh, pc, point, ribbon, show, smat, stream, surf, vec, vp, wp`

ribbon — Set options for stream ribbons.

Synopsis:

`ribbon` *option=value [,option=value, ...]*

Arguments:

option	A plot option for the ribbon plot. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`err`	real number `>= 0.0`	A scaled error value. An error of 0.01 means that variations on the order of 1/100 the size of the mesh are filtered out through line reduction. 0.0 indicates no filtering is done.	`0.0`
`density`	positive real number	The stream density. This is a scaled parameter. It relates to the "seed point density per unit area"; however, it is scaled such that a density of 1 results in approximately 200 streams. This is so the results are the same whether the mesh is defined in cm, mm, or any other scale.	`1.0`
`maxsteps`	positive integer	Number of integration steps taken per stream.	`50`
`method`		Integration method for computing streams.	`rk2`
	`euler`	1st order Euler integration	
	`rk2`	2nd order Runge Kutta	
	`rk4`	4th order Runge Kutta	
	`eulera`	adaptive step 1st order	
	`rk2a`	adaptive step 2nd order	
	`rk4a`	adaptive step 4th order	
`numpoints`	positive integer	The number of streamlines to generate for each seed point. Two streamlines will result in a ribbon, while more than two will result in a tube.	`2`
`pop`	See `pop` option on page 69.		`off`

Option	Value	Value Meaning	Default Value
`step`	real number `>= 0.0`	Integration step size. This parameter is scaled according to the size of the mesh compared to the magnitudes of the vectors to allow the step size to give similar quality results in a wide variety of meshes. If using an adaptive method, the specified step is used only as an initial value and will be adjusted according to the curvature of the flow at each point.	`0.1`
`var`	variable name	Name of the vector variable. The variable defines pseudo-coloring as well. A 4d vector is interpreted as {x_component, y_component, z_component, colorvar}.	`velocity`

Description:

Ribbons are an extension of streams for 3D meshes. Ribbons are generated by meshing a surface between multiple streamlines which are seeded near each other. This depicts the flow through a 3D problem.

Examples:

```
ribbon numpoints=3, method=”rk2a”, maxsteps=40
```

See Also:

`pop, stream`

rotx — Rotate a 3D object about the screen's X-axis.

Synopsis:

`rotx` *angle [count]*

Arguments:

angle	A real number indicating the amount to rotate (in degrees).
count	Optional integer count parameter, indicating the number of times to rotate by `angle` degrees.

Description:

The `rotx` command allows the user to rotate a 3D object about the screen's X-axis. With the optional *count* parameter, one can rotate an object in multiple steps, thus getting a more complete picture of the object shape.

Examples:

```
rotx 45        # Rotate 45° once
rotx 10 36     # Rotate 10° 36 times
```

See Also:

`latitude, longitude, roty, rotz`

roty — Rotate a 3D object about the screen's Y-axis.

Synopsis:

`roty` *angle [count]*

Arguments:

angle	A real number indicating the amount to rotate (in degrees).
count	Optional integer count parameter, indicating the number of times to rotate by `angle` degrees.

Description:

The `roty` command allows the user to rotate a 3D object about the screen's Y-axis. With the optional *count* parameter, one can rotate an object in multiple steps, thus getting a more complete picture of the object shape.

Examples:

```
roty 45         # Rotate 45° once
roty 10 36      # Rotate 10° 36 times
```

See Also:

`latitude, longitude, rotx, rotz`

rotz — Rotate a 3D object about the screen's Z-axis.

Synopsis:

`rotz` *angle [count]*

Arguments:

angle	A real number indicating the amount to rotate (in degrees).
count	Optional integer count parameter, indicating the number of times to rotate by `angle` degrees.

Description:

The `rotz` command allows the user to rotate a 3D object about the screen's Z-axis. With the optional *count* parameter, one can rotate an object in multiple steps, thus getting a more complete picture of the object shape.

Examples:

```
rotz 45        # Rotate 45° once
rotz 10 36     # Rotate 10° 36 times
```

See Also:

`latitude, longitude, rotx, roty`

savewin — Save the current output window into an image file.

Synopsis:

```
savewin
savewin option=value [,option=value]
```

Arguments:

option	An option for saving images. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`banner`	banner string	This is a string to print to the top and bottom of a PostScript™ output file.	`""`
`capture`	`on, off`	Whether or not to get the image to save by doing a screen capture.	`on`
`family`	`on, off`	A flag indicating if the filenames should be familied.	`off`
`filename`	file name	The name (or rootname) of the file to save the image to. The proper extension (.ps, .rgb, .xwd, .tif) will be added onto the end of the filename.	`meshtv`
`invert`	`on, off`	A flag indicating if background and foreground should be inverted.	`off`
`type`	`ps, rgb, rps, tif, xwd`	The file format to save the image in. `ps` — PostScript™, 2D plots only `rgb` — SGI's rgb format `rps` — Raster PostScript™ `tif` — tiff format `xwd` — xwd format	`ps`
`xres`	positive integer	The resolution in the X direction. Only used when `capture` is off, and only applies to `rgb`, `rps`, `tif`, and `xwd` formats. If you specify `xres`, you must also specify `yres`, and the two numbers should be equal to each other.	`1024`
`yres`	positive integer	The resolution in the Y direction. Only used when `capture` is off, and only applies to `rgb`, `rps`, `tif`, and `xwd` formats. If you specify `yres`, you must also specify `xres`, and the two numbers should be equal to each other.	`1024`

Description:

The `savewin` command without any options saves the contents of the current output window into the previously specified file. With options, the `savewin` command sets parameters for future window saves, *but does not actually save the window.*

If the capture flag is on, the picture will be “grabbed’ via a screen capture. This means that if you open another window on top of your output window before the save is done, you will see the second window in your saved file. This is the quickest method for saving pictures, which is why it is the default.

If the family flag is on, MeshTV will automatically append a four digit number to the end of the rootname, and automatically increment the number each time a `savewin` is done.

The types of file formats supported are `ps` for PostScript^{TM}, `rgb` for SGI’s rgb format, `rps` for raster PostScript^{TM}, `tif` for Aldus Corporation’s tiff format, and `xwd` for the X Window System^{TM} xwd (X Window Dump) format.

Examples:

```
savewin filename=mypix
savewin invert=on, family=on
savewin
```

See Also:

`printwin`

setframe — Set the current frame of an animation.

Synopsis:

`setframe` *frame_number*

Arguments:

frame_number	A frame number in the animation.

Description:

After an animation has been stopped, a specific frame in an animation can be selected via the `setframe` command. This frame will be displayed, and it becomes the current frame.

Examples:

```
setframe 0     # Reset the animation.
setframe 10    # Go to the 11th frame in the animation.
```

See Also:

`addframe, animate, nextframe, play, prevframe, stop`

sh — Execute an operating system (shell) command.

Synopsis:

`sh` *command*
`sh` *“command args”*

Arguments:

command	Any shell command.
args	Arguments to the shell command.

Description:

The `sh` command is an escape to the operating system from within MeshTV. With it, you can enter any command which the operating system can understand. Note that if the command has arguments, special characters, or multiple commands separated by semi-colons, then the entire string must be quoted.

Use this command if you want to issue a UNIX™ command from within MeshTV.

Examples:

```
sh "ls -l"
sh "emacs my-command-file &"
sh "rm junk*"
```

See Also:

`csh, listdb`

show — Display the current settings for plotting options.

Synopsis:

`show` *option*

Arguments:

option An option name. See table below.

Option Name	Meaning
`block`	Show the active block numbers.
`bnd`	Boundary plot options.
`dist`	Dist command options.
`foutput`	Display file output options.
`iso`	Iso plot options.
`label`	Label plot options.
`legend`	Legend options.
`material`	Options for material numbers.
`matspecies`	Options for material species numbers in a plot.
`mesh`	Mesh plot options.
`pc`	Pseudo color plot options.
`pick`	Show options for pick and query mode.
`point`	Point plot options.
`rect`	Show options for rect command.
`ribbon`	Ribbon plot options.
`smat`	Solid-filled material plot options.
`stream`	Stream plot options.
`surf`	Surface plot options.
`vec`	Vector plot options.
`view`	Viewing parameters (e.g., window, up-vector, etc.).

Description:

The `show` command prints the current values for the given option to the terminal. Use this to check the current variable or other option for a specific plot.

Examples:

```
show smat
show view
```

See Also:

`bnd, ct, dist, iso, label, legend, matcol, materials, mesh, pc, point, reset, ribbon, smat, stream, surf, vec`

simclose — Close an open simulation.

Synopsis:

`simclose` *[simname]*

Arguments:

simname	The name of a simulation, which follows the same rules as names for files. If this argument is omitted, MeshTV will close the currently active simulation.

Description:

The `simclose` command closes a remote computer simulation that was opened previously with the `simopen` command.

For a description of simulations, what they are and how to use them, see the MeshTV Getting Started Manual.

Examples:

```
simclose
simclose mycode
```

See Also:

`simcont, simopen, simpause, MeshTV Getting Started Manual, MeshTV SDX Manual, SILO User's Guide`

simcont — Continue a paused simulation.

Synopsis:

`simcont` *[simname]*

Arguments:

simname	The name of a simulation, which follows the same rules as those for files. If this argument is omitted, MeshTV will continue the currently active simulation.

Description:

The `simcont` command continues a remote computer simulation that was previously paused with the `simpause` command.

For a description of simulations, what they are and how to use them, see the MeshTV Getting Started Manual.

Examples:

```
simcont
simcont mycode
```

See Also:

`simclose, simopen, simpause, MeshTV Getting Started Manual, MeshTV SDX Manual, SILO User's Guide`

simopen — Open a remote computer simulation for reading.

Synopsis:

`simopen` *simname*

Arguments:

simname	A simulation name, which follows the same rules as those for files.

Description:

The `simopen` command opens a remote computer simulation which is running SILO's Scientific Data Exchange (SDX) package. MeshTV can read from the simulation just as it can read from a SILO file.

When a simulation is opened, it is implicitly paused. To continue the simulation, use the `simcont` command.

For a description of simulations, what they are and how to use them, see the MeshTV Getting Started Manual.

Examples:

```
simopen mycode
```

See Also:

`simclose, simcont, simpause, MeshTV Getting Started Manual, MeshTV SDX Manual, SILO User's Guide`

simpause — Pause execution of the current remote computer simulation.

Synopsis:

`simpause` *[simname]*

Arguments:

simname	The name of a simulation, which follows the same rules as those for files. If this argument is omitted, MeshTV will pause the currently active simulation.

Description:

The `simpause` command pauses execution of a remote computer simulation opened with the `simopen` command.

For a description of simulations, what they are and how to use them, see the MeshTV Getting Started Manual.

Examples:

```
simpause
simpause mycode
```

See Also:

`simclose, simcont, simopen, MeshTV Getting Started Manual, MeshTV SDX Manual, SILO User's Guide`

smat — Set plotting options for the solid-filled material boundary plot.

Synopsis:

`smat` *option=value [,option=value, ...]*

Arguments:

option A plot option for the material boundary plot. See table below.

value A value associated with the matching *option*. See table below.

Option	**Value**	**Value Meaning**	**Default Value**
`pop`	See `pop` option on page 69.		`off`
`type`	`all, clean`	Type of zones to plot. `all` will plot all materials, and `clean` will plot only clean zones (i.e., no zones with mixed materials.)	`all`
`var`	variable name	Name of variable to get material data from.	`material`
`vf`	`0.0` <= real number <= `1.0`	Volume fraction. This value isn't taken into account if the problem only has clean zones. If the problem has mixed zones, you can set the value less than 0.5 to make the boundaries encompass more, or to less than 0.5 to make them encompass less. 0.5 is the most accurate. To get a feeling for this option, try playing around with it a bit.	`0.5`

Description:

The `smat` command sets the options used when generating a solid-filled material boundary plot. Note that this command does not change any existing plots, only future ones. Refer to the `matcol` command for associating colors with particular materials.

Examples:

```
smat vf=.4
smat var=myvar
```

See Also:

`bnd, plot, matcol, materials`

source — Process a command input file.

Synopsis:

`source` *filename*

Arguments:

filename	The name of a file containing MeshTV commands to execute. The filename can include an absolute or relative path, though the “~” can not be used.

Description:

The `source` command operates like the UNIX[TM] C-Shell source command. It accepts a filename as input, then executes each line in that file as if a user had typed it in. Use this when you want to operate on a large set of files, but don’t want to manually execute the plotting commands.

Two potential uses for a command input file are: (1) using MeshTV in a “batch” mode, where a large number of commands, files, or both are processed without user intervention; and (2) storing a series of commands which may be too long or complex for an alias, but which the user wants to execute frequently.

Examples:

```
source batch.file
```

Sample contents of ‘batch.file’:

```
plot pc   # Assumes user already has opened a SILO file
winset 2
plot iso
winset 3
plot mesh
winset 4
plot vec
```

See Also:

`alias, csh, sh`

stereo — Set the stereo display mode.

Synopsis:

`stereo on | off`

Arguments:

`on`	Activate stereo mode.
`off`	Deactivate stereo mode. This is the default.

Description:

The `stereo` command sets the stereo viewing mode for the workstation. When “on”, the active window is enlarged to full-screen size, and a double set of images are created (one for the left eye, one for the right eye.) With the use of stereo viewers, the user can view images in 3-D.

NOTE - This option is available only on the SGI, and only on those machines with stereo viewers attached. Use of this command without the proper hardware attached will result in a “garbage” picture.

Examples:

```
stereo on
```

See Also:

stop — End the playback of the current animation.

Synopsis:

```
stop
```

Arguments:

No arguments.

Description:

The `stop` command stops the playback of the current animation. Resume the animation with the `play`, `nextframe`, or `prevframe` commands.

Examples:

```
stop
```

See Also:

`addframe, animate, nextframe, play, prevframe, setframe`

stream — Set options for the stream plot.

Synopsis:

`stream` *option=value [,option=value, ...]*

Arguments:

option A plot option for the stream plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`err`	real number `>= 0.0`	A scaled error value. An error of 0.01 means that variations on the order of 1/100 the size of the mesh are filtered out through line reduction. 0.0 means that no variations are filtered.	`0.0`
`density`	positive real number	The stream density. This is a scaled parameter. It relates to the "seed point density per unit area", however it is scaled such that a density of 1 results in approximately 200 streams. This is so the results are the same whether the mesh is defined in cm, mm, or any other scale.	`1.0`
`lc`	line color	Line color. (See APPENDIX B: Color Names.)	`white`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX C: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1`
`maxsteps`	positive integer	Number of integration steps taken per stream.	`50`
`method`		Integration method for computing streams.	`rk2`
	`euler`	1st order Euler integration	
	`rk2`	2nd order Runge Kutta	
	`rk4`	4th order Runge Kutta	
	`eulera`	adaptive step 1st order	
	`rk2a`	adaptive step 2nd order	
	`rk4a`	adaptive step 4th order	
`pop`	See `pop` option on page 69.		`off`

Option	Value	Value Meaning	Default Value
`step`	real number > `0.0`	Integration step size. This parameter is scaled according to the size of the mesh compared to the magnitudes of the vectors to allow the step size to give similar quality results in a wide variety of meshes. If using an adaptive method, the specified step is used only as an initial value and will be adjusted according to the curvature of the flow at each point.	`0.1`
`var`	variable name	Name of the vector variable. The variable defines pseudo-coloring as well. A 3d vector in a 2D mesh is interpreted as {x_component, y_component, colorvar}.	`velocity`

Description:

Streams work in 2D and 3D meshes. Streams (or streamlines) are curves in a vector field which are tangent to the flow (instantaneous flow). They are generated from a starting “seed” point, and are propagated through a vector field via various methods.

Streams are implemented for all mesh types except non-colinear (curvilinear) 3D. For unstructured meshes, only hexahedral zones are supported at this time.

The `stream` command is similar to the `ribbon` command, except that it generates lines (streamlines) while `ribbon` generates a ribbon (or tube), which is the aggregate of two (or more) streamlines.

Examples:

```
stream numpoints=3, method=”rk2a”, maxsteps=40
```

See Also:

`ribbon`

surf — Set plotting options for the surface plot.

Synopsis:

`surf` *option=value [,option=value, ...]*

Arguments:

option	A plot option for the surface plot. See table below.
value	A value associated with the matching *option*. See table below.

Option	**Value**	**Value Meaning**	**Default Value**
`boxcol`	color name `off`	Plot a bounding box in the specified color. (See APPENDIX B: Color Names.) Do not plot a bounding box.	`red`
`pop`	See `pop` option on page 69.		`off`

`surfcol`	`z`
	
	color name
	
	`off`
		Shade the surface in colors per the z-values of the surface.
		Shade the surface in the specified color and light the surface. The surface is lit from front and back light sources. (See APPENDIX B: Color Names.)
		Do not shade the surface.
			`z`

`type`	`shaded, wire, both`	`shaded` is equivalent to: `surfcol=z, wirecol=off, boxcol=red`
		`wire` is equivalent to: `surfcol=off, wirecol=black, boxcol=red`
		`both` is equivalent to: `surfcol=z, wirecol=black, boxcol=red`
			`shaded`

`wirecol`	color name `off`	Plot the surface as a wireframe in the specified color. (See APPENDIX B: Color Names.) Do not plot the surface as a wireframe.	`off`
`var`	variable name	Name of variable to plot.	`d`

Description:

The `surf` command sets the options used when generating a surface plot. Note that this command does not change existing plots, only future ones. The surface plot only operates on 2D data. The surface is shaded using the Gouraud method.

Examples:

```
surf var=p
surf surfcol=red, boxcol=off
surf surfcol=magenta, wirecol=blue, boxcol=cyan
```

See Also:

`defvar, iso, plot, pc`

unalias — Remove the definition of a command alias.

Synopsis:

`unalias` *name*

Arguments:

name	An alphanumeric string which was previously assigned a value via the `alias` command. This argument must start with a character.

Description:

The `unalias` command removes any definition previously assigned to *name* with the `alias` command. See the `alias` command summary for more information.

Examples:

```
unalias foo
```

See Also:

`alias`

vec — Set plotting options for the vector plot.

Synopsis:

`vec` *option=value [,option=value, ...]*

Arguments:

option A plot option for the vector plot. See table below.

value A value associated with the matching *option*. See table below.

Option	**Value**	**Value Meaning**	**Default Value**
`headsize`	real number > `0.0`	Size of the arrow head as a fraction of the arrow length.	`0.25`
`lc`	line color	Line color. (See APPENDIX B: Color Names.)	`white`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX C: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`max`	real number > `0.0`	Maximum length of the vector. Units are the same as the data units.	`No default`
`min`	real number > `0.0`	Minimum length of the vector. Units are the same as the data units.	`No default`
`numvec`	integer > `0`	Number of vectors to plot.The total number of nodes in the problem is divided by the number of desired vectors (this number) to get a stride. This stride is used to stride through the coordinate arrays to pick out points to put vectors on.	`400`
`pop`	See `pop` option on page 69.		`off`
`scale`	real number > `0.0`	Magnification scale. A multiplier for manipulating the vector length.	`1.0`
`var`	variable name	Name of variable to get vector data from.	`velocity`

Description:

The `vec` command sets the options used when generating a vector plot. Note that this command does not change existing vector plots, only future ones. The variable read by the vector plot is a special merged multi-value variable which may be created with the `defvar` command.

Examples:

```
vec lc=green, scale=2.5
vec var=myvelocity
```

See Also:

```
plot, defvar
```

vp — Set the 2D plotting viewport.

Synopsis:

`vp` *xmin, xmax, ymin, ymax*
`vp full`

Arguments:

xmin	The minimum value in the X direction. A real number.
xmax	The maximum value in the X direction. A real number. 0 <= xmin < xmax <= 1
ymin	The minimum value in the Y direction. A real number.
ymax	The maximum value in the Y direction. A real number. 0 <= ymin < ymax <= 1
`full`	Set the viewport to be as large as possible and still leave room for tick marks and their labels.

Description:

The `vp` command sets the 2D viewport. The `reset vp` command can be used to reset the viewport to its default values (0.25, 0.95, 0.15, 0.90). `full` might be used when creating output for presentations or publications.

`vp` differs from `wp`, though the two are similar. `vp` sets the portion of the window into which to put the plot. Thus its coordinates are in window coordinates. wp determines which part of the problem (mesh) to plot, and so its coordinates are in problem (“physical” or “world”) coordinates.

Examples:

```
vp .25, .75, .15, .65
vp full
```

See Also:

```
reset, wp
```

windeicon — Deiconify one or more windows.

Synopsis:

`windeicon` *window_list*
`windeicon all`

Arguments:

window_list	The list of windows to deiconify, where the window is referenced by a window id. A window id is an integer number ranging from 1 to the number of active windows. The maximum number of window is 16.
`all`	Deiconify all MeshTV windows.

Description:

Use this command to “open up” or deiconify MeshTV windows which have been iconified via the `winicon` command, or by some mechanism provided by the user’s window manager.

Examples:

```
windeicon 1 3 8
windeicon all
```

See Also:

`windel, winicon, winnum`

windel — Delete the active or a specified output window.

Synopsis:

```
windel
windel window_id
```

Arguments:

window_id	The window identifer, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.

Description:

If there is no argument, this command deletes the active window. If a window id is given as an argument, This command deletes the window identified by *window_id*. In either case, there must be at least one output window remaining after the deletion. Otherwise, the window is not deleted.

Note that windows remaining after a deletion are not renumbered. So if you had 5 windows open, and you deleted windows 3 and 4, you would be left with windows 1, 2, and 5.

If you delete the active window, the lowest numbered window becomes the new active window.

Examples:

```
windel
windel 3
```

See Also:

`winnew, winnum, winset`

wingrow — Make the active output window full size.

Synopsis:

`wingrow`

Arguments:

No arguments.

Description:

The `wingrow` command causes the active window to be “full” size. Full size is the size of a window when only one window is opened, as with the `winnew` command. An example of use might include issuing a `winnum 4` to generate 4 windows, and then issuing a wingrow to make the first window full size again.

Examples:

`wingrow`

See Also:

`winnew, winnum, winshrink`

winicon — Iconify one or more windows.

Synopsis:

`winicon` *window_list*
`winicon all`

Arguments:

window_list	The list of windows to iconify, where the window is referenced by a window id. A window id is an integer number ranging from 1 to the number of active windows. The maximum number of window is 16.
`all`	Iconify all MeshTV windows.

Description:

Use this command to "close" or iconify MeshTV windows. The windows iconify to the desktop rather than the MeshTV Icon Box window.

Examples:

```
winicon 1 3 8
winicon all
```

See Also:

`windeicon, winnew, winnum`

winmode — Set the cursor mode of the active output window.

Synopsis:

`winmode` *mode*

Arguments:

mode	One of: ‘pick’, ‘dist’, ‘rect’, or ‘vplane’

Description:

The `winmode` command sets the cursor mode of the current output window for 2D plots. (For 3D objects, the cursor is in virtual-sphere mode, and is unable to change to any of the modes listed above.) The default window mode for 2D objects is vplane (virtual-plane), but the mode can be changed to any of the above listed modes.

When a window is in `pick` mode, the cursor turns into a cross-hair. When you click the mouse in a plot, information about the selected point will be written to the command line window. This information will vary depending on the kind of plot chosen. See the `pick` command for more information.

In `dist` (distance) mode, the cursor is an arrow. If you drag the cursor with the left mouse button down, you will see a line between the point where you initially clicked and the point where your cursor currently resides. When you release the left mouse button, the line will draw from your beginning point to your ending point. See the `dist` command for more information.

In `rect` (zoom) mode, the mouse can be used to select a rectangular region of interest which is then enlarged to fill the viewport. This is essentially a zoom. (See `zoom3`.)

In `vplane` (virtual plane) mode, the left and middle mouse buttons have different functions. The left mouse button can be used to pan (move) the image by depressing the button and dragging the cursor across the window. Release the button when the image is in the position you want. The middle mouse button can be used to zoom in on the image either by a small amount if clicked once, or by a series of small amounts by holding down the middle mouse button. If you want to zoom by large amounts, you should select the rect mode. The middle mouse button in combination with the ALT key depressed can be used to zoom out on the image either by a small amount or a series of small amounts as described earlier.

Examples:

```
winmode pick
winmode rect
```

See Also:

`dist, pick, zoom3`

winnew — Open a new output window.

Synopsis:

`winnew`

Arguments:

No arguments.

Description:

The `winnew` command opens another output window. This window becomes the active window.

Examples:

```
winnew
```

See Also:

`windel, winnum, winset`

winnum — Set the number of output windows.

Synopsis:

`winnum` *number*

Arguments:

number	One of: 1, 4, 9, or 16

Description:

The `winnum` command sets the number of output windows used by MeshTV. Depending on the window manager used, the windows are either automatically placed on the screen for you, or you have to place each window manually. Windows, if they are arranged automatically, are arranged in rows and columns.

When you have multiple output windows, the command "`winnum 1`" will resize the current active window to be full-screen. The other windows will iconify and be placed into the MeshTV Icon Box window, which is below the output window area. The command "winnum 4" will create four windows so that they fit in the output window area. If there were already more than 4 windows, the first 4 will resize and the remaining will be placed into the MeshTV Icon window. If there were less than 4, windows will be added until 4 exist.

The `winnum` command sets the size of the windows created. New windows are created at the same size as current windows. For example, "`winnum 4"` produces 4 windows in a 2X2 grid. A subsequent call to `winnew` would produce a new window which was the same size as windows in a 2X2 grid.

Here is a list of the grid sizes:

winnum 1	1X1
winnum 4	2X2
winnum 9	3X3
winnum 16	4X4

Examples:

```
winnum 4
winnum 1
```

See Also:

`windel, winnew, winset`

winset — Set the active MeshTV output window.

Synopsis:

`winset` *number*

Arguments:

number	The window identifer, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.

Description:

The `winset` command tells MeshTV which output window is active. Only one output window is active at any one time. Commands for setting options, databases, plot selections, and so forth apply only to the active window. Each output window has its own set of options which must be set separately.

Examples:

```
winset 2
```

See Also:

`windel, winnew, winnum`

winshrink — Make the active output window normal size.

Synopsis:

```
winshrink
```

Arguments:

No arguments.

Description:

The `winshrink` command causes the active output window to be shrunk to “normal” size. This command should be issued after a `wingrow` command has been issued, and it will return the window to the size it was when it was created.

Examples:

```
winshrink
```

See Also:

```
wingrow
```

winzoom — Zoom on the plots in a specified output window.

Synopsis:

`winzoom` *window_id* x_1 y_1 x_2 y_2

Arguments:

window_id	The window identifer, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.
x1, y1	The (x,y) coordinates of the upper left-hand corner of the zoom rectangle. These are real (floating point) numbers which are problem coordinates.
x2, y2	The (x,y) coordinates of the lower right-hand corner of the zoom rectangle. These are real (floating point) numbers which are problem coordinates.

Description:

The plots in the output window specified by `window_id` are replotted using the zoom rectangle as the limits of the plot, thereby effecting a zoom. This command works only for 2D plots.

Examples:

```
winzoom 1 -4.0 4.0 2.0 2.0
```

See Also:

`winmode, zoom3, zoomf3`

wp — Set the 2D plotting window via problem coordinates.

Synopsis:

`wp` *xmin xmax ymin ymax*

Arguments:

xmin	The minimum value to use in the X direction. A real number.
xmax	The maximum value to use in the X direction. A real number. *xmin* < *xmax*
ymin	The minimum value to use in the Y direction. A real number.
ymax	The maximum value to use in the Y direction. A real number. *ymin* < *ymax*

Description:

The `wp` command sets the 2D plotting window using problem (physical/world) coordinates. This command allows the user to view a subset of a mesh or plot. The `rect` mode of the `winmode` command performs the same function, but allows the user to use the mouse for window selection.

`wp` differs from `vp`, though the two are similar. `vp` sets the portion of the window into which to put the plot. Thus its coordinates are in window coordinates. wp determines which part of the problem (mesh) to plot, and so its coordinates are in problem (“physical” or “world”) coordinates.

Examples:

```
wp 0 10 0 10
wp 0. 15. 10. 40.
```

See Also:

`vp, winmode, wp3`

wp3 — Set the 3D plotting window via problem coordinates.

Synopsis:

`wp` *xmin xmax ymin ymax zmin zmax*

Arguments:

xmin	The minimum value to use in the X direction. A real number.
xmax	The maximum value to use in the X direction. A real number. *xmin* < *xmax*
ymin	The minimum value to use in the Y direction. A real number.
ymax	The maximum value to use in the Y direction. A real number. *ymin* < *ymax*
zmin	The minimum value to use in the Z direction. A real number.
zmax	The maximum value to use in the Z direction. A real number. *zmin* < *zmax*

Description:

The `wp3` command sets the 3D plotting window using problem (physical/world) coordinates. This command allows the user to view a subset of a mesh or plot.

Examples:

```
wp3 0 10 0 10 0 10
wp3 0. 15. 10. 40. 15. 15.
```

See Also:

`wp`

zoom3 — Set the 3D relative zoom factor.

Synopsis:

`zoom3` *factor*

Arguments:

factor	Any positive real number, indicating the multiplication factor applied to the object before displaying.

Description:

The `zoom3` command enlarges or reduces the current picture by the given factor, *relative to the current picture size*. That is, each `zoom3` command operates on the current view system, thus providing a cumulative effect. As an example, each successive use of the command "`zoom3 2`" would double the size of the displayed picture.

Examples:

```
zoom3 2        # Size doubles.
zoom3 2        # Doubles size again.
zoom3 .5       # Size shrinks to half the current size.
```

See Also:

`pan3, panf3, winmode, zoomf3`

zoomf3 — Set the 3D absolute zoom factor.

Synopsis:

`zoomf3` *factor*

Arguments:

factor	Any positive real number, indicating the multiplication factor applied to the object before displaying.

Description:

The `zoomf3` command enlarges or reduces the current picture by the given factor, *relative to the original picture size*. Because the command does not provide a cumulative effect, multiple invocations of the command "`zoomf3 2`" would not alter the size of the displayed picture after the first invocation.

Examples:

```
zoomf3 2        # Size doubles.
zoomf3 2        # Size doesn't change.
zoomf3 .5       # Size shrinks to half original size.
```

See Also:

`pan3, panf3, winmode, zoom3`

! — Re-execute a MeshTV command.

Synopsis:

`! !`
`!` *number*
`!` *string*

Arguments:

!	Re-execute the last command.
number	An integer number indicating the sequence number of the command to re-execute.
string	A string of characters indicating the initial letters of the command to re-execute. The last command which began with the characters in *string* will be re-executed.

Description:

The `!` command is modeled after the similarly named UNIX^{TM} C-Shell command. It allows you to re-execute MeshTV commands quickly and easily. Command selection can occur in three ways. First, by indicating the sequence number of the command (which can be retrieved by using the `history` command, or by looking at the sequence number built into the MeshTV command prompt). Second, by indicating the first few characters of the command to re-execute. The final way is an abbreviation for re-executing the last command issued to MeshTV, accessed with the command "`! !`".

Using this command can save typing time.

Examples:

```
!!        # Re-execute the last command
!20       # Re-execute command number 20
!pl       # Re-execute the last command beginning with 'pl'
```

See Also:

`history`

— MeshTV comment character.

Synopsis:

#
text

Arguments:

text	The text for the comment.

Description:

The character '`#`' is MeshTV's comment character. On a given input line, any text following a '`#`', till the next newline, will be ignored. This feature is typically used to annotate command input files which could be used to run MeshTV in batch mode.

Examples:

```
plot bnd        # Generate a boundary plot.
```

See Also:

`source`

APPENDIX A MeshTV Tutorial

The following tutorial illustrates some of the basic concepts and uses of the command line interface to MeshTV.

A.1 Before You Can Start

Before starting on the tutorial, you will need the following:

- A SILO file, preferrably "rect2d.silo" which can typically be retrieved at `/usr/local/meshtv/data/rect2d.silo`.
- Access to the executable `meshtvx`. If when you type `meshtvx` you get the message:

  `meshtvx: Command not found.`

  then `meshtvx` is either not on your machine or not in your search path. Contact your system administrator for assistance.

A.2 Typographical Conventions

For the purpose of this tutorial, the following typographical conventions will be used:

`typeface`	Items printed in typeface should be entered verbatim.
1. normal	Numbered items printed in normal font are merely annotation and should not be entered.

A.3 Tutorial

1. Begin the command line interface to MeshTV.

   ```
   meshtvx
   ```

2. Open a SILO file.

```
open /usr/local/meshtv/data/rect2d.silo
```

3. List the contents of the file (shows the names of all variables and meshes.)

```
ls
```

4. Plot a mesh.

```
mesh var=quadmesh2d
plot mesh
```

5. Clear the output window.

```
clear
```

6. Make a contour plot of density.

```
iso var=d
plot iso
```

7. Change the contour levels.

```
iso level={.1 1. 2.}
clear; plot iso
```

8. Change the contour levels by percent.

```
iso pct={10 25 50 75 90}
clear; plot iso
```

9. Generate a pseudo-color contour plot of pressure.

```
pc var=p
clear; plot pc
```

10. Find the min and max values of pressure.

```
minmax p
```

11. Set the range for the pseudo-color plot.

```
pc pcmin=.5, pcmax=1.5
clear;plot pc
```

12. Use the pick and query facility. Use left mouse button to select points.

```
winmode pick
pick var=p
```

13. Plot a mesh in another color on top of current plot.

```
mesh lc=red
plot mesh
```

14. Change the window limits (using physical/problem dimensions).

```
wp 0 5 0 5
```

15. Reset the physical window limits.

```
reset wp
```

16. Change to four output windows.

```
winnum 4
```

17. Change the active window.

```
winset 2
```

18. Generate a material-boundary plot.

```
bnd var=mat1
plot bnd
```

19. Change the active window and generate a solid-material plot.

```
winset 3
smat var=mat1
plot smat
```

20. Generate a distance plot of density. Use the left mouse button to select the two endpoints of a line (within window 3).

```
winmode dist
dist var=d, outwin=4
```

21. End this session of MeshTV.

```
end
```

APPENDIX B Color Names

The following table contains all of the color names understood by MeshTV. These color names can be used for line color attributes, text color attributes or one of the 3D lighting attributes. See the command summaries for more details.

TABLE 1 MeshTV Standard Color Table

Color Numbers	Color Names
1	red
2	green
3	blue
4	cyan
5	magenta
6	yellow
7	orange
8	violet
9	grey
	black
	white
	off (used by some commands to indicate automatic cycling of colors)

The last three entries, black, white, and off cannot be modified by the user.

APPENDIX C Line Styles

The following table contains all of the line styles understood by MeshTV. These line styles can be used by many plots, such as the `bnd`, `dist`, `iso`, and `mesh` plots, to name a few. See the command summaries for more details.

TABLE 2 MeshTV's Line Styles

Line Styles
dash
dot
dotdash
solid