

pV3 Server User's Reference Manual

Rev. 2.05

for use with the following OpenGL workstations:

Compaq ALPHAs

HP workstations (HPUX 10.20 ACE)

RedHat LINUX (Rev 6.1 or higher)

IBM RS/6000 workstations (AIX 4.3 or higher)

Silicon Graphics

SUNs (Solaris 2.6 or higher)

WindowsNT/2000

Bob Haimes

Massachusetts Institute of Technology

December 12, 2001

License

This software is being provided to you, the LICENSEE, by the Massachusetts Institute of Technology (M.I.T.) under the following license. By obtaining, using and/or copying this software, you agree that you have read, understood, and will comply with these terms and conditions:

Permission to use, copy, modify and distribute, this software and its documentation for any purpose and without fee or royalty is hereby granted, provided that you agree to comply with the following copyright notice and statements, including the disclaimer, and that the same appear on ALL copies of the software and documentation:

Copyright 1993-2001 by the Massachusetts Institute of Technology. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS", AND M.I.T. MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, M.I.T. MAKES NO REPRESENTATIONS OR WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE LICENSED SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

The name of the Massachusetts Institute of Technology or M.I.T. may NOT be used in advertising or publicity pertaining to distribution of the software. Title to copyright in this software and any associated documentation shall at all times remain with M.I.T., and USER agrees to preserve same.

Contents

1	Introduction	5
1.1	pV3 Startup	5
1.1.1	Server Startup	6
1.1.2	Environment Variables	6
1.1.3	Special Files	7
1.1.4	Notes on the Common Desktop Environment – CDE	9
1.2	pV3 Time-Outs and Error Recovery	9
2	The pV3 Server Users Interface	11
2.1	Surfaces	11
2.2	Windows	11
2.2.1	Text Window	11
2.2.2	3D Window	12
2.2.3	2D Window	12
2.2.4	1D Window	12
2.2.5	Key Window	12
2.2.6	Dialbox Window	13
2.3	Scalar Visualization Tools	15
2.3.1	Surface rendering	15
2.3.2	Planar Cutting plane	15
2.3.3	Program-defined cutting plane	16
2.3.4	Iso-surfaces	16
2.4	Vector Visualization Tools	17
2.4.1	Bubbles	17
2.4.2	Instantaneous Streamlines	18
2.4.3	Tufts	19
2.4.4	Arrows	19
2.4.5	Vector Clouds	20
2.5	Grid	20
2.6	Thresholding	20
2.7	Probes	21
2.8	Help Menus	22
2.8.1	3D Window	22

2.8.2	2D Window	23
2.8.3	1D Window	24
2.8.4	Key Window	25
2.8.5	Dials Window	26
2.9	Dialbox Functions	28
3	Multi-Disciplinary Visualization	29
4	Collaboration	30
4.1	Help Menus	31
4.1.1	Servers Window	31
4.1.2	Servers Extract Window	31
5	Post-Processing	33
5.1	tab2ps	33
5.2	img2tiff	33
5.3	img2ps	34
5.4	img2X	35
6	User Interface Differences with Visual3	36
7	Warning and Error Messages	38

1 Introduction

pV3 is the newest in a series of graphics and visualization tools to come out of the Department of Aeronautics and Astronautics at MIT. Like it's predecessors **Visual3**, **Visual2** and **Grafic**, **pV3** is a software package aimed at aiding in the analysis of a particular suite of problems. In this case it is the real time visualization of 3D large scale solutions of transient (unsteady) systems.

pV3 (which stands for parallel **Visual3**), is a completely new and different system, but builds heavily on the technology developed for **Visual3**. It has been designed specifically for co-processing visualization of data generated in a distributed compute arena. It is also designed to allow the solver to run as independently as possible. If the solution procedure takes hours to days, **pV3** can 'plug-into' the calculation, allow viewing of the data as it changes, then can 'unplug' with the worst side-effect being the temporary allocation of memory and a possible load imbalance.

pV3 provides the same kind of functionality as **Visual3** with the same suite of tools and probes. The data represented to the investigator (the 3D, 2D and 1D windows with cursor mapping) is the same. Also the same Graphical User Interface (GUI) is used.

pV3 programming is very **Visual3**-like. For the desired flexibility and the merging of the visualization with the solver, some programming is required. The coding is simple; like **Visual3**, all that is required of the programmer is the knowledge of the data. Learning the details of the underlying graphics, data extraction, and movement (for the visualization) is not needed. If the data is distributed in a cluster of machines, **pV3** deals with this, resulting in few complications to the user.

1.1 pV3 Startup

The **PVM** daemon(s) and with co-processing, the solver, must be executing. Without a **pV3** server running, every time the solution is updated, a check is made for the number of members in the **PVM** group *pV3Server* (Note: this name can be changed for multiple jobs running under the same user ID – see the Section 1.1.2 for the environment variable 'pV3_Group'). If no servers are found, no action is taken. When a **pV3** server starts (usually from an interactive xterm session on the graphics workstation), it enrolls in the specified group. The next time the solution is updated, an initialization message is processed and the visualization session begins. Each subsequent time in the solver completes a time step, visualization state messages and *extract* requests are gathered, the appropriate data calculated, collected and sent to the active server(s).

When the user is finished with the visualization, the server sends a termination message and exits. The clients receive the message, and if no other servers are running, cleans up any memory allocations used for the visualization. Then the scheme reverts to looking for server initialization, if termination was not specified at **pV3** client initialization.

1.1.1 Server Startup

The interactive server, *pV3Server*, takes three arguments at the command line (all optional). The first is the *setup* file with the default of 'pV3.setup'. The second argument is the *color* file to be used at startup (the default is 'spec.col'). The third argument is the *extract start-up* file. There is no default. An argument of '-' is a place holder, allowing the default to be used.

```
examples: % pV3Server case.setup
          % pV3Server pV3.setup bw.col
          % pV3Server
          % pV3Server case.setup - case.startup
```

1.1.2 Environment Variables

The **pV3** server uses six Unix environment variables. Some are the same as the ones used for **Visual3**. The variable 'Visual3_CP' defines the file path to be searched for color files, if they are not in the user's current directory. This allows all of the color files to be kept in one system directory.

The second variable, 'Visual_KB' is optional. This variable, if defined, must point to a file that contains alternate keyboard bindings for the special keys used by **pV3**. The file is ASCII. The first column is the key name (10 characters) and the second is the X-keysym value in decimal (use 'xev' to determine the appropriate values for the key strokes).

The third is 'pV3_TO' and should be used to change the internal Time-Out constant. If the variable is set, it must be an integer string which is the number of seconds to use for the Time-Out constant (the server's default is 60). This may be required if the time between solution updates is long. See *Time-Outs and Error Recovery*, Section 1.2.

The fourth is 'pV3-Threading'. This is used to specify how the interactive Server and the post-processing Viewer handles the handshaking between their active threads. There are two methods; (1) 'Hard' where the thread sits in a hard loop (with a thread yeild) looking for a change of state, or (2) 'Flag' where the threads use Semaphores for waiting until the state changes. The advantage of 'Hard' is interactivity, the advantage for 'Flag' is less processor time consumed. By default, this variable is set to 'Hard' for most situations with single processor workstations and 'Flag' for multi-processors.

Exceptions: ALPHA's default is 'Hard', SUN's default is 'Flag', regardless of number of processors.

The fifth is 'pV3_Group'. This is useful for differentiating multiple **PVM** jobs running under the same user ID. If this variable is set for the solver (client-side) before execution, it overrides the default client side group name *pV3Client*. The name used is the string assigned to this variable with *Client* appended. By setting this variable before server execution, it will set the server group to the variable's string with *Server* appended instead of using *pV3Server*. Only clients with the appropriate matching group name will be connected to this session.

The last is 'pV3_Warning'. If this variable is set (to anything except a NULL string) warning messages about ACK and maximum streamline segments are not reported.

1.1.3 Special Files

- Window Manager Resource File

By default, all modern Xwindows Managers allow the closing or deleting of windows by either double-clicking on the menu pull-down or selecting 'Close' or 'Quit' from the pull-down. This will abort the execution of the Server or Viewer. To avoid this, an additional window manager menu description can be added to the default information for the Window Manager. This is accomplished by specifying a user-level resource. In the distribution the following files can be found in the "servers" subdirectory; 'user.4Dwmrc', 'user.mwmrc', and 'user.dtwmrc'. These files are for SGI's default window manager, the Motif window manager and CDE's window manager, respectively. The entire resource file (for the appropriate window manager) must be copied from the system (usually something like '/usr/lib/X11/system.XXwmrc' or '/etc/dt/config/sys.dtwmrc') – if not already done. This file gets named '.XXwmrc' (where XX is 4D or m) and then the appropriate user resource file appended to the end. Note: for CDE, this file gets put in the '.dt' directory at the users top level and must be given the name 'dtwmrc'.

- .Xdefaults

If the SGI window manager is used, '4DWm' must be told what to do with **pV3**s windows. These commands must be placed in the file '.Xdefaults'. See the file 'user.Xdefaults'. For DEC, IBM and SUN systems this information is found in the window manager's Setup File (Mwm or the appropriate file for the WM used).

pV3 requires three X fonts. The file '.Xdefaults' in the users home directory is examined for the font names and are designated "Visual*large", "Visual*medium" and "Visual*small". The sample file 'user.Xdefaults' comes with the distribution and may be concatenated to the user's '.Xdefaults' file.

The X fonts loaded on any system may be examined by the command 'xlsfonts'.

- 'Mwm' Setup File

When a user begins a session using the Motif window manager, it reads the initialization file 'Mwm' in the user's home directory. This file contains information on how to treat various classes of windows and the window focus (as well as other things). A sample file 'user.Mwm' is in the distribution and may be used as 'Mwm' or its contents concatenated to the user's 'Mwm' file.

- 'twm' Setup File

If the 'twm' window manager is used, when a user begins a session, twm reads an initialization file '.twmrc' in the user's home directory. This file defines certain key bindings and window attributes. It is necessary for correct **pV3** operation for the user to modify the standard '.twmrc' file as shown in the 'user.twmrc' file on the **pV3** distribution, or just use this file as '.twmrc'.

- Lock File

If the server is running on a multi-processor SGI workstation a file is used for the coordination of the 2 threads generated during execution. This file has the name ‘.pV3.locks’ and is open in the current directory. It should be noted that running two invocations of the **pV3** server from the same directory will NOT work. Both will use the same file for the lock and semaphore arena!

1.1.4 Notes on the Common Desktop Environment – CDE

For the **pV3** suite to work properly with CDE, the Style Manager must be used to change the default methods CDE uses for cursor/window functioning. Under the Window section “Point in Window to Make Active” must be selected. Also, “Raise Window when Active”, and “Allow Primary Windows on Top” must NOT be selected.

1.2 pV3 Time-Outs and Error Recovery

The **pV3** system was designed to be as error-free and as robust as possible. Because the client software runs closely coupled to the software generating the data, extra care has been taken to avoid causing any errors or problems related to this visualization system.

If the **pV3** server aborts, the client side software will NOT hang waiting for the completion of some message stream. In general, the server sends a series of request messages framed by an ‘end-of-requests’ message. All messages are received by the client software by either a tight loop or timed-out receive. In the tight loop, there is first a check to see if the server is still active. If so, the next message is pulled of the message queue. If there is none, the server’s presence is checked again, and so on until a message is received. Control is returned to the solver when either the ‘end-of-requests’ message is received or the server terminates (either gracefully or just disappears).

In the timed receive, an amount of time is specified for the receive. If this time is exceeded before a message is collected, it is assumed that something is wrong and the client shuts down the visualization (just as if the user terminated the session).

The type of message handling is controlled by the server’s time-out constant. This is set by the environment variable ‘pV3_TO’. If the constant is set to 0 (zero), then the client and server do not time out, and are in hard loops. If the value is some positive number, both the client(s) and server will time out if something gets hung. If the time-out constant is negative, the server will time-out after the (negated) number of seconds and the client(s) are put in a hard loops. The default is 60 (seconds).

The choice of what to use depends on the network, the type of machine(s), whether the machines are dedicated, and the cost of putting software in tight loops waiting for some response.

If you are using tight loops and **pV3** seems to hang, the only way to free things up is to abort the server. If the server seems hung, hitting *Ctrl-C* in the window that started the server should work.

NOTES:

1. Client-side time-outs are controlled by the first server that connects to a session when multiple servers have started (even if that first server has exited). The value can only be reset once all servers have terminated.
2. For SGIs: The server runs as two threads (there are two unique PIDs for the task). If for some reason the IO thread aborts, it may leave the windows up but give a prompt in the window where the server was started. This can usually be fixed by hitting 'Esc' in the 3D window or by *kill*ing the process left. If the graphics thread aborts (the windows will disappear) interrupt the IO thread by hitting *Ctrl-C*.

2 The pV3 Server Users Interface

In trying to understand the following description of the user interface, it may be helpful to run one of the demo programs found in the example directory of the distribution. It is necessary to get **PVM** going then startup one of the example clients. After the client is running the server (*pV3Server*) can be started.

2.1 Surfaces

pV3 deals with three different types of surfaces. The first category is *domain* surfaces. These are surfaces that are defined by the client program(s) during **pV3** initialization, and they typically correspond to the surfaces which bound the computational domain. A subset of this first class, are *mapped domain* surfaces, for which there is a mapping from points on the surface to an (x', y') coordinate system. This allows plotting of surface quantities in a 2D setting.

The second category is *dynamic* surfaces. These are surfaces whose orientation and position, relative to the computational domain, can be changed interactively by the user. Although there are several types of *dynamic* surfaces, only one *dynamic* surface can exist at one time. Also, a *dynamic* surface cannot be activated when a *mapped domain* surface is being plotted in the 2D window.

The third category is *static* surfaces. These are surfaces which at one time were *dynamic*, but then transferred into the surface database, along with the *domain* surfaces. These *static* surfaces are then treated in almost the same way as the *unmapped domain* surfaces.

NOTE: In **Visual3**, any *static* surface in a *grid unsteady* application deformed with the grid movement. The surface was associated with the cells and not physical space. In **pV3** the *static* surface acts like it did when *dynamic* – correctly for *grid unsteady* and *structure unsteady* cases.

2.2 Windows

The user interface is divided into six different windows. As is typical of X-window applications, the functions invoked by mouse button, keyboard or dialbox input are dependent upon the position of the cursor. Thus, different functions are available to the user depending on which window is *active*, i.e. contains the cursor. An important feature of the user interface is its help key. Pressing '?' will cause a list of the available commands for the active window to be displayed in the text window. The six different windows are:

2.2.1 Text Window

The text window is the window from which the **pV3** server (*pV3Server*) was started. It is a good idea to keep this window in the lower left corner of the screen where it will not be obscured by the other **pV3** server windows. This window is used to output various messages, including the help menus, and to input filenames, numerical values, and any other textual information.

2.2.2 3D Window

The 3D window displays data on three-dimensional surfaces. It also displays three-dimensional lines such as tufts and streamlines, and other objects that are discussed later. These objects in the 3D window can be rotated, translated and enlarged using the dialbox (or pseudo-dials – see the Dialbox Window, Section 2.2.6). All motion is relative to screen coordinates and not object coordinates. One set of key strokes allow the user to store particular viewing positions, and later restore them using the numeric keypad. The *setup* file retains this data so that it can be used at server restart or with other data-sets.

2.2.3 2D Window

The 2D window is used to display data on a *mapped domain* or *dynamic* surface (for which there exists a mapping to a (x', y') coordinate system). This 2D data may be independently rotated, translated and enlarged again using the dialbox (or pseudo-dials).

2.2.4 1D Window

The 1D window is used to display one dimensional data which is generated by various functions in the 2D window or from mapping instantaneous streamlines.

2.2.5 Key Window

This window displays the color scheme used in the 3D and 2D windows. When the cursor is in this window, there are many options available. These include the ability to interactively change the color scheme, load in a new color scheme, or change the length of displayed vectors.

There are two *traffic lights* in the left-hand portion of the Key window. The lower *traffic light* (which is red when the **pV3** server is computing, yellow when **pV3** waiting for some specific user input, and green when ready to accept new user input) is active in the left-most portion of this window. Tied to the *traffic light* color is the color of the cursor. When the cursor is yellow **pV3** is expecting mouse button presses.

The upper *traffic light* reflects major requests pending to the clients, such as a change to the current scalar field variable. Because a new request is sent out as a time-frame is being collected (and viewing the previous results) this light is also tri-state. A red light indicates that the request is being made. A yellow light means that one more iteration is required before the request is satisfied. A green light indicates no pending requests.

2.2.6 Dialbox Window

The Dialbox window serves four functions. Pressing ‘s’ switches the window between these functions.

- Pseudo-dials

The mode is to display the functions associated with each of the dials of the dialbox. Pressing the middle mouse button will switch the dialbox display between the 3D, 2D and Key windows for which the dials are active, and in each case eight dials are displayed together with labels describing the dial function (pan, zoom, rotate, etc.) when the cursor is in that window. If there is no dialbox the dials can be rotated by putting the cursor on the appropriate dial and holding down the left or right mouse button; in this case the window that is changed is the one labeled in the base of the Dialbox window. Holding down the mouse button in the center of the dial will cause faster movement. In this mode the affected window will go into *fast-drawing* mode where only the surface edges are drawn. Under many circumstances this produces more interactive positioning than using the physical dialbox.

- Surface database

The surface database lists all of the *domain* and *static* surfaces. Each entry has four small boxes which give the status of that surface’s attributes. The first box gives the rendering status (white=ON, grey=translucent, black=OFF), the second is the grid status (white=ON, black=OFF), the third is the grey surface status (white=ON with colored contours, grey=ON, black=color) and the fourth is the thresholding status (white=ON, black=OFF). The meaning of some of these terms will become clearer later when discussing the scalar visualization tools.

The user can change the attributes by pressing any mouse button when the cursor is on one of the boxes. Doing this, or pressing a mouse button on the surface label, also causes that surface to become the *active* surface. The *active* surface is highlighted in the database, and is important for certain plotting options discussed later.

NOTE: In **Visual3** it was not possible to have contouring on without the underlying surface also rendered. In **pV3**, the user can accomplish this by having the rendering box black and grey surface box white.

- Streamline database

The streamline database lists all of the streamline objects, and next to each there are four small boxes which give the status of important streamline attributes. The first box gives the rendering status (white=colored, grey=ON, black=OFF), the second is the direction (white=backward, grey=both directions, black=foreward), the third is the streamline type (white/cross=tubes with twist, white=tubes, grey=ribbons, black=streamlines) and the fourth is the particle seeding status (white=ON, black=OFF). The meaning of some of these terms will become clearer later with the discussing the vector visualization tools.

The user can change the attributes by pressing any mouse button when the cursor is on one of the boxes. Doing this, or pressing a mouse button on the streamline label, also causes that object to become the *active* streamline object. This object is highlighted in the database, and is important for certain plotting and control options discussed later.

NOTE: This is completely new for **pV3** and does not exist in **Visual3**!

- Discipline/Client database

For multi-client and/or multi-discipline cases, this database allows setting the current discipline as well as controls the visibility of all disciplines and clients. Each discipline displays 2 boxes where the clients (optionally) listed under the discipline each have a single box. If the client box is white, the client is visible. If the box is black, then the *extracts* (except of streamlines) associated with that client are not drawn. The user can change the visibility state by pressing any mouse button when the cursor is on the box.

The first box for a discipline either exposes or hides the clients associated with the discipline. If the box is black, all clients are hidden. A white box indicates that the clients are listed below. The user can change this state by pressing any mouse button when the cursor is on this first box.

The second box is a visibility flag for the entire discipline. A white box means that all clients for a discipline are visible. A black box indicates that all clients have been turned off. A grey box flags that some clients for the discipline are on and some are off. The user can turn off all clients in a discipline by pressing any mouse button when the cursor is on a white box. The user can turn on visibility for all clients by pressing any mouse button when the cursor is on a black or grey box.

The current discipline is set by pressing any mouse button when the cursor is on the title of the desired discipline.

NOTE: This is completely new for **pV3** and does not exist in **Visual3!**

2.3 Scalar Visualization Tools

The one of the data types in **pV3** is scalar data, which is simply scalar information defined at each node of the computational grid(s). **pV3** does not know anything about the data other than an associated function number and label. For example, in fluid dynamic applications, a function may have label *pressure* and a second scalar function may have label *Mach number*. An important concept in **pV3** is the notion of *active* functions, and the *active scalar function* is the function and associated label which corresponds to the scalar data currently used by **pV3**. The user can switch to a different scalar function by hitting a particular key on the keyboard which is *bound* by the **pV3** client initialization procedure. For example, key ‘p’ may be bound to the function, labeled *pressure*.

The following is a list of plotting functions available for use with scalar data:

2.3.1 Surface rendering

Gouraud-shaded (smooth color shading) surface contours of the scalar function can be rendered on any, or all, of the *domain* and *static* surfaces. Column 1 of the surface database is used to select which surfaces are to be displayed. With the cursor in the Key window, there is a variety of options to interactively change the color scheme used for the rendering. Key ‘l’ loads a new color file, while key ‘r’ restores the original color file. Two dials on the dialbox (or pseudo-dials) change the upper and lower bounds of the scalar function range spanned by the color scheme.

If the *active* surface (as defined earlier) is a *mapped domain* surface (with an associated mapping to a (x', y') coordinate system), then Gouraud-shaded contours can also be plotted in the 2D window, by pressing F5 in the 3D window. The active surface is highlighted in the surface database, but if one is unsure of which it is in the 3D window then pressing F1 will cause the rendering on that surface to blink off and on.

Another option with *mapped domain* surfaces is F6 in the 3D window, which performs a surface rendering in both the 2D and 3D windows of the active *surface scalar* function, a function that is only defined on mapped domain surfaces as specified by the programmer. The color map used to render this surface function can be viewed by toggling key ‘s’ in the Key window.

2.3.2 Planar Cutting plane

The cutting plane is a *dynamic* surface, a true planar surface cutting through the 3D field. The cutting plane is initialized by pressing F3 in the 3D window. This puts the 3D window into a special mode in which the 3D object is held fixed and the user can use the dialbox to rotate the cutting plane into the desired orientation. When ready, pressing F3 again switches off the planar movement mode and turns on the regular cutting plane mode.

Once the cutting plane is activated, it is controlled from the 2D window, meaning that it responds to keys and dials that are active when the cursor is in the 2D window. Using dials, the cutting plane can be moved from side to side, up and down, in a direction normal to the plane (using the *scan* dial) and rotated in its own plane. Function key F9 toggles (switches on and off) rendering in the 2D window, while F6 toggles rendering in the 3D window. F10 toggles the display in the 2D window of the grid defined by the intersection of the cutting plane and the 3D computational grid faces.

The cutting plane can be turned off and on using F2 in the 3D window. The cutting plane position can also be stored, like the 3D viewing position, by pressing the Ctrl key and one of the ten numbers in the numeric keypad on the right-hand-side of the keyboard. It can be restored later by pressing just the number.

2.3.3 Program-defined cutting plane

This is similar to the last option, but instead of being a truly planar surface, it is a surface corresponding to $z' = \text{const}$, where z' is a programmer-defined function (in the clients); the *scan* capability varies the value of *const* interactively. This allows the programmer to define conical, cylindrical or other surfaces not otherwise defined by the **pV3** server. The client programmer also must have defined a mapping to (x', y') coordinates so that plotting is possible in the 2D window. The program-defined cutting plane is activated by F4 in the 3D window. The other options in the 2D window are the same as for the regular cutting plane.

2.3.4 Iso-surfaces

An iso-surface is a *dynamic* surface with a uniform value of the currently active scalar variable, and is activated by F7 in the 3D window. The iso-surface value is displayed in the Key window and can be varied interactively (in the Key window) using the dialbox to *scan* the value of z' , or key 'z' to set its value, or the right mouse button to pick a value from the color key.

2.4 Vector Visualization Tools

The second **pV3** data type is vector data. This is a set of 3D vector values for each grid node. As with the scalar function, this vector data is associated with an *active* vector function, which can be changed by pressing a key that is *bound* to another vector function. The following is a list of plotting functions for vector data:

2.4.1 Bubbles

Bubbles are unsteady particle paths. This tool provides the same effect as hydrogen bubbles in experimental techniques. Bubbles are active with all **pV3** unsteady modes (unless the simulation is *paused*). A single bubble path may be spawned by simply pressing a mouse button in the 2D window (assuming seeding is off) which provides an initial point to start the integration in 3D space. If bubble coloring is on (F12 in the 3D window), the particle location will be rendered by the current scalar; if not, the current location is rendered with the default streamline color. The spheroid may also be colored by the time that the bubble was spawned. This is accomplished by hitting ‘s’ in the Key window. The time limits will probably have to be adjusted (hitting ‘f’ in the Key window).

Several bubble paths may be started from a line or circle by using key F11 in the 2D window. Similarly, a grid of bubbles may be spawned using F12 in the 2D window.

If the particle streamer is on (Tab key in the 2D window), bubbles will be continuously spawned from the current cursor location at every time step. This mimics the experimental technique of streaklines where dye is continuously injected at a spot in the flow field. With the streamer on and the boundary layer or line probe is on, particles are emitted along the line every snapshot in time (the number of particles is the last set by spawning a line of bubbles – F11 in the 2D window). And, finally if the streamer is on and the tufts are active, a grid of bubbles is seeded each time step at the tuft locations.

NOTE: Particles whose spawn position is on a surface (by starting on a *mapped domain* surface or via the Edge Plot) function differently from normal 3D bubbles. These particles remain on the surface. This is done numerically by using the projection of the local vector field onto the surface during the integration. The vector field option (and scalar field option used for coloring) is set via the ‘Volume/Surface’ toggle, see the 1D window help menu, Section 2.8.3. This allows using the current vector field or the current *special* vector function for the integration. *Special* functions are only defined at surfaces (not the entire volume).

2.4.2 Instantaneous Streamlines

Streamlines are curved 3D lines which are everywhere parallel to the local vector field. They are obtained by numerical integration of the vector field along a line starting at some chosen location. Instantaneous streamlines may only be activated for steady-state cases or when the seeding toggle is on (the key ‘|’ in the 2D window). The starting point is initially determined by use of a surface plotted in the 2D window. A point in the 2D window maps back to a corresponding point on the *dynamic* surface in the 3D window and so can be used to seed instantaneous streamlines; pressing one of the mouse buttons does this and (depending on which button is pressed) produces a streamline going upstream and/or downstream. Alternatively, key F11 (and subsequent mouse actions which are requested) defines a line or circle in the 2D window, which is then used to specify a set of streamlines in the 3D window. Key F12 initiates an object of streamlines from a regular grid of points in the 2D window, like it would spawn a grid of bubbles with the seeding toggle off.

Using the streamline database (in the Dial window), groups of streamlines can be plotted either as lines of constant color (usually white), or colored according to the value of the local *active* scalar function. In the latter case, it is helpful to enable the *grey* status for background surfaces (using column 3 in the surface database) so that instead of being rendered in color they are instead rendered in solid grey, making the colored streamlines clearer.

Each object in the streamline database can also be reset as to the direction (downstream, upstream or both) and if bubbles are to be seeded from the same locations. It should be noted that in most cases seeding particles in this manner (and with the instantaneous streamlines not rendered) is faster than doing it interactively with an active cut in the 2D window.

The streamline object database also allows each group to be drawn as a line or the following (by using column 3 in the streamline database):

- Ribbons

Stream ribbons are streamlines that have been given some width. One edge is the true instantaneous particle path, the other edge is constructed by rotating a constant length normal vector about the path tangent according to the local streamwise angular rotation rate. The result is a ribbon whose twist illustrates the streamwise vorticity of the flow.

If the streamlines are colored, the ribbon is rendered in the default streamline color (usually white), otherwise the ribbon is colored with the current scalar.

The width of the ribbon may be adjusted by using the dials when the cursor is in the Key window. A specific width may be entered by hitting ‘w’. Also, the ribbons may be rotated by using the dials with the cursor in the Key Window.

- Tubes

A tube is a streamline with a circular crossflow area. The radius of the cross-section is derived from the local crossflow divergence. The crossflow divergence measures the local crossflow expansion rate. Thus, the resulting tube displays the local expansion/compression of the current vector field.

If the streamline coloring is on, the tubes will be colored with the current scalar. Otherwise, the default streamline color is used.

The width of the tube may be adjusted by using the dials when the cursor is in the Key window. A specific width (and maximum radius) may be entered by hitting ‘w’ in the Key window. The maximum radius is useful to limit the size of the tube in stagnation regions of a flow field where the radius can become exponentially large.

- Tubes with Twist

The rotation and divergence effects can be rendered simultaneously by placing lines on the surface of tube which twist with the local rotation rate. This effectively combines the functionality of the ribbons and tubes. The final image displays the streamline, the rotation rate, the crossflow divergence, and scalar variations.

Again, if the streamline coloring is on, the tubes will be colored with the current scalar and the lines will be in the default streamline color. Otherwise, the tube is the default color and the lines are the current scalar.

As with ribbons and tubes, the rotation angle, the tube size, and the tube maximum may all be controlled from the Key window using the dial box or by the appropriate key strokes.

NOTES:

- (1) The streamline accuracy is reduced for a segment that crosses interface regions.
- (2) Invoking streamlines in a multi-client application can have a negative effect on performance. This technique is serial in nature and stalls all clients until the integrations requested are finished.
- (3) Streamlines started on a surface (by using a *mapped domain* surface or via the Edge Plot) function differently from normal 3D streamlines. These paths remain on the surface. This is done numerically by using the projection of the local vector field onto the surface during the integration. The vector field option (and scalar field option used for coloring) is set via the ‘Volume/Surface’ toggle, see the 1D window help menu, Section 2.8.3. This allows using the current vector field or the current *special* vector function for the integration. *Special* functions are only defined at surfaces (not the entire volume).

2.4.3 Tufts

Tufts are similar in concept to streamlines. A regular grid of points in the 2D window map to a corresponding grid of points on the surface in the 3D window. At the points in the 3D window, tufts are drawn which are short lines with magnitude and direction corresponding to the local vector field. At the points in the 2D window the tufts correspond to the projection of the 3D vector field onto the 2D plane. Key ‘Tab’ in the 3D window toggles tufts on and off. One of the dials in the Key window allows interactive change in the scaling parameter which relates the vector magnitude to the tuft size, and key ‘a’ allows this scaling parameter to be input from the keyboard.

2.4.4 Arrows

Arrows are the same as tufts except that they are defined only at 2D nodes (the intersection of a cut and cell edges). To emphasize that they are different, they are drawn as lines with heads in the 2D window, whereas tufts are drawn as lines with a cross base. Arrows are shown on cutting planes as well as iso-surfaces and are displayed in the 3D window as line segments. Arrows are toggled on and off by key F7 in the 2D window.

2.4.5 Vector Clouds

Vector clouds display the local vector field at nodes which meet the current threshold limits. The vector cloud technique is useful for locating interesting flow features and displaying the vector fields in these regions. Vector clouds are invoked by hitting F8 in the 3D window and are always rendered with the current scalar.

NOTE: Selecting this option (and not carefully pre-setting the thresholding limits) can produce an enormous amount of network traffic! 7 words are transmitted for each node within the system that meets the threshold criteria.

2.5 Grid

The computational grid can be displayed on any, or all, of the *static* and *dynamic* surfaces. For the *static* surfaces this is controlled through column 2 of the surface database. For *dynamic* surfaces and data plotted in the 2D window, grid display is controlled by function key F10 (in the 2D window). The grid lines that are displayed correspond to the intersection of the plotting surfaces and the faces of the computational grid(s).

2.6 Thresholding

A threshold function is another scalar function which is set, or changed, by pressing the appropriate keys. The purpose of this function, when enabled, is to restrict all *domain* and *static* surface plotting to only those parts of the surface on which the thresholding function lies within a certain range. The user can interactively vary the upper and lower threshold bounds. The user can also select, through column 4 of the surface database, the surfaces that are to be thresholded.

If the threshold function is chosen to be the same as the scalar function, then this provides a means to plot the part of a surface on which the scalar function is within certain limits. If the threshold function is chosen to be geometric (e.g. x) then this produces a dynamic *cutaway*, in which the surface is only plotted within a certain geometric volume.

The threshold function can be set in two ways. Pressing a key on the keyboard that has been defined by the programmer to be associated with a threshold function loads that functions data into the threshold array. Alternatively, pressing F9 in the 3D window loads the current scalar function data into the threshold storage.

The thresholding limits, within which plotting will be performed, can be varied interactively using dials in the Key window, or input manually using key 't'.

2.7 Probes

There are a variety of *probes* which are available when plotting in the 2D window or from streamline objects:

- Point (2D - F1)

The point probe is located at the cursor position, and returns, in the text window, the point's coordinates and the value of the active scalar and vector functions.

- Strip chart (2D - F2)

The strip chart is similar to the point probe, except that instead it produces a plot in the 1D window of the current scalar function against time.

- Line (2D - F3)

When the line probe is invoked the user is asked to input two points using the mouse. These define a line in the 2D window, and the output is a plot in the 1D window showing the variation of the current scalar function along that line.

- Edge Plot (2D - F4)

The edge plot is similar to the line plot, except that in this case the line in the 2D window is the edge line closest to the cursor when this option is invoked.

- Surface Layer (2D - F5)

This option produces a line plot in the 1D window of the current scalar function along a line placed normal to an edge in the 2D window, at the edge position which is closest to the cursor. As the user moves the cursor, the normal line moves accordingly.

- Streamline Probe (Dial - '|')

The streamline probe may be started any time there are streamline objects. The current object is mapped to the 1D window. When the cursor is in the 1D window a cross-hair or disc appears in the 3D window marking the closest position on the active streamline. The size of the disc mimics the stream tube thickness. For surface streamlines the mapped cursor is a cross-hair displayed in both the 2D and 3D windows. This allows the user to both know which streamline is mapped and probe the streamline.

NOTES:

- 1) Tabular output files (visualXYZ.tab) created when this probe is active also contain the coordinate triads for the streamline.
- 2) When tubes are on, the disc size is 150% of the tube thickness.

2.8 Help Menus

2.8.1 3D Window

The help menu that is printed when one types '?' in the 3D window is as follows:

3D Window

Mouse Buttons:

m - Center View @ Cursor

Key Strokes:

~	- write visual.img File	+	- Box blow-up
F1	- Show Active Surface	F2	- Toggle Cutting Plane
F3	- Cutting Plane positioning	F4	- Toggle Program Cut Plane
F5	- Toggle Surface Display	F6	- Toggle Disp. w/Surface Fn
F7	- Toggle Iso-Surface	F8	- Toggle Vector Clouds
F9	- Set Scalar as Threshold	F10	- Animate StreamLines
F11	- Bubble Render Toggle	F12	- Bubble Color Toggle
Delete	- Delete Bubbles	\	- Time Line Toggle
Insert	- Save Dynamic Surface	^	- Shading Toggle
Tab	- Tufts Toggle	Home	- Reset View
PageUp	- Reset Clipping	PageDown	- Depth Cueing Toggle
NumPad	- Set view from position #	Ctrl-NumP	- Store view in position #
/	- Edge Outline Toggle	End	- Terminate 2D modes
	- Ribbon/Tube Toggle	Pause	- Freeze the action
!	- 3D Window Status	Esc	- Terminate pV3 Server

Comments:

- 1) At the top of the help menu in real applications there would be a list of the scalar, vector and threshold function variables and their associated keys, as defined by the application program.
- 2) A toggle is a switch that is either on or off, and so pressing the key changes it to the other status.
- 3) Clipping is similar to a geometric thresholding. It displays the part of the 3D object that is behind a plane held parallel to the screen.
- 4) In the 'NumPad' and 'Ctrl-NumP' descriptions, 'NumPad' and 'NumP' refer to one of the ten numbers on the numerical keypad on the right of the keyboard. This number is then referred to as #. This option allows the storing and recall of ten different viewing positions and any cutting planes that are active. On SGI workstations the 'NUM LOCK' light must be on for these key-strokes to be acknowledged. On some SUN keyboards, use the left-hand keypad.
- 5) Displaying the active surface, F1, will only work if the surface has some render attribute on (Box 1 in the surface database). The surface will flash on and off.
- 6) Pause is only active when there is 1 server. If 'Pause' is in effect when a second server starts, it automatically releases the clients.

2.8.2 2D Window

The help menu that is printed when one types '?' in the 2D window is as follows:

2D Window

Mouse Buttons:

l - Bubble/StreamLine going upstream
m - Bubble/StreamLine going up/downstream
r - Bubble/StreamLine going downstream

Key Strokes:

~	- write visual.img File	+	- Box blow-up
F1	- Point Probe	F2	- Strip Chart
F3	- Line Probe	F4	- Edge Plot
F5	- Surface Layer Scan	F6	- 3D Window Render Toggle
F7	- Arrow Toggle	F8	- Contour Toggle
F9	- Render Toggle	F10	- Grid Toggle
F11	- Line/Circle of StreamLines	F12	- Grid of StreamLines
Delete	- Flip X in Window	Tab	- Bubble Streamer Toggle
End	- Terminate Line Plot		- StreamLine Seed Toggle
t	- Dynamic Surf Thresh Toggle	!	- 2D Window Status

Comments:

- 1) the 'Delete' option reverses the sign of the x' -coordinate in the 2D window, effectively *turning over* the 2D window. This is helpful when the cutting plane surface you are seeing in the 3D window is the reverse side of the 2D window.
- 2) 'End' ends all plotting in the 1D window.
- 3) The StreamLine Seed Toggle allows what would spawn off Bubble(s), to add objects to the StreamLine database.

2.8.3 1D Window

The help menu that is printed when one types '?' in the 1D window is as follows:

1D Window

Mouse Buttons:

m - Set Cut Plane w/ StreamLine Probe (positioning on)
any - Seed SL/Bubble w/ Edge Plot on

Key Strokes:

r	- add Reference line	s	- Volume/Surface Fn Toggle
x	- Change X scaling	y	- Change Y scaling
End	- Terminate Line Plot	PrintScrn	- Tabular Output

Comments:

- 1) To set a planar cut perpendicular to the streamline at a given position, first turn the streamline probe on and select the appropriate streamline, then turn planar cut positioning on (F3 in the 3D window). Move the cursor in the 1D window to the correct position, and to finish, press the middle mouse button.
- 2) The *Reference line* is an additional line placed in the 1D window along with the results of a probe. This line is read from a file in the **pV3** tabular file output format and displayed in grey.
- 3) The Volume/Surface function toggle allows the specifying of what surface functions are used for integrations and rendering of surface particles and streamlines when there are *special* surface functions. This allows the choice between the *special* functions and the normal volume scalar/vector fields.

2.8.4 Key Window

The help menu that is printed when one types '?' in the Dials window is as follows:

KEY Window

Mouse Buttons:

m - Set new color at cursor position
r - Set Iso-Surface/Time value

Key Strokes:

A - Toggle 3D arrow-head type	a - (Re)Set arrow/tuft size
c - Set # of Contours	d - Discipline in Dial Window
f - (Re)Set function limits	l - Load new color file
m - Set S.L. Animations	q - Query function limits
R - Set 3D arrow-head ratios	r - Reset color scheme
s - Toggle color schemes	S - Single step (unsteady only)
t - (Re)Set thresh limits	W - Write Startup file
w - Set Tube/Ribbon width	z - Set ZPrime
! - Key Window Status	

Comments:

- 1) Option 's' allows one to toggle the display of the color schemes, between the color scheme that is used for all standard scalar rendering, the scheme that is used to display scalar surface functions, and the color map used for time rendering of particles.
- 2) The iso-surface and cutting planes correspond to a surface on which $z' = const.$ Option 'z' allows one to explicitly specify the value of this constant.
- 3) Single-stepping is like pause, but after each new data frame, the paused state is reactivated. The 'Pause' key (in the 3D window) must be used to release each time-frame.

2.8.5 Dials Window

The help menu that is printed when one types '?' in the Dials window is as follows:

Dials Window (Dials)

Mouse Buttons:

l - Move Dial Clockwise
m - Change Window Mapping
r - Move Dial CounterClockwise

Key Strokes:

~ - write ImageFile of Screen	c - comparison window
d - Dial Sensitivity	s - Surface List Toggle
M - Mirror Toggle	S - Send Clients a String

Dials Window (Surface List)

Mouse Buttons: any - Select

Key Strokes:

~ - write ImageFile of Screen	c - comparison window
d - Dial Sensitivity	s - Surface List Toggle
PageUp - Move Surface List Up	PageDown - Move Surface List Down
Delete - Remove Current Surface	M - Mirror Toggle
S - Send Clients a String	V - Vec Toggle-Current Surf

Box:	1	2	3	4
black	not rendered	no grid	scalar rendered	no thresholding
grey	translucent		grey surface	
white	opaque	grid on	grey w/ contour	thresholding on

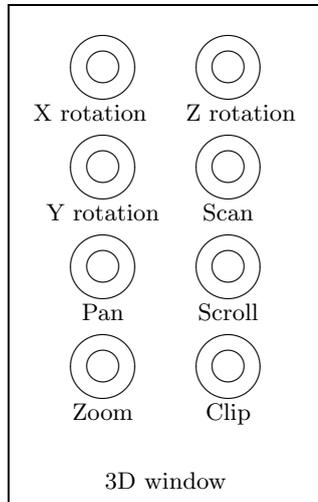
Dials Window (SL-Particle List)

Mouse Buttons: any - Select

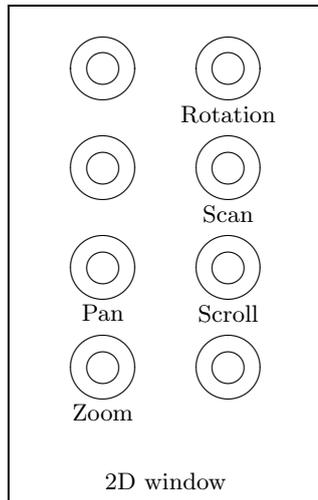
Key Strokes:

~ - write ImageFile of Screen	c - comparison window
d - Dial Sensitivity	s - Surface List Toggle
PageDown - Move SL-Part List Up	PageUp - Move SL-Part List Down
Delete - Remove Current SL-Part	- StreamLine Probe

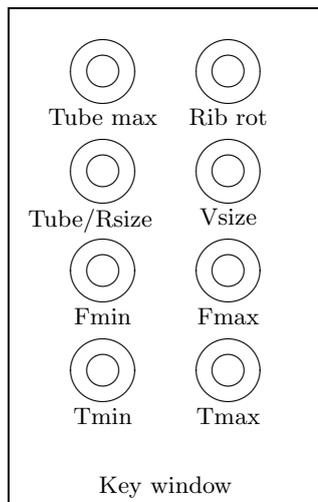
2.9 Dialbox Functions



- X rotation: rotate about X-axis
- Z rotation: rotate about Z-axis
- Y rotation: rotate about Y-axis
- Scan: move cutting plane or iso-surface
- Pan: move right/left
- Scroll: move up/down
- Zoom: enlarge/reduce
- Clip: move clipping boundary



- Rotation: rotate
- Scan: move cutting plane
- Pan: move right/left
- Scroll: move up/down
- Zoom: enlarge/reduce



- Tube max: maximum tube size
- Rib rot: ribbon rotation
- Tube/Rsize: change tube/ribbon size
- Vsize: change scaling of tufts/arrows
- Fmin: change minimum scalar function value
- Fmax: change maximum scalar function value
- Tmin: change minimum thresholding value
- Tmax: change maximum thresholding value

3 Multi-Disciplinary Visualization

Multi-disciplinary cases are defined by the client pool when a **pV3** server starts. A client defines its discipline by setting a character string when the call-back PVCLIENT is invoked at **pV3** client-side initialization. All clients belonging to the same discipline have the same field variables (i.e., scalar, vector, and etc.) and the same programmer-defined cuts. See the **pV3** Programmer's Guide.

The design of the servers (and post-processing viewer) for multi-disciplinary cases is that there be a current discipline and that while the discipline is active, all user interaction effects that discipline. This can be thought of as multiple visualization sessions (one for each discipline) with one set of output windows. This means that each discipline has its current field variables, with associated color maps and visualization state. Because data displayed in the 2D and 1D windows reflect some further examination of the 3D data, these windows only show the filtering from the current discipline (though each discipline's state can indicate output to these windows). Switching between disciplines will update the data drawn to reflect the state of the current discipline.

Control of the current discipline, as well as the discipline/client visibility is performed via the Dialbox Window, see Section 2.2.6.

The following information is 'global' and therefore not part of any discipline's state:

- Display State

There is one set of windows and they are used regardless of the discipline. This is also true for the lighting used in the 3D Window.

- View State

The viewing transformation matrix is global and not part of current discipline. This maintains a constant view as the disciplines are changed.

- Planar Cuts

The dynamic planar cut is currently the only tool that allows cross-discipline examination. The 2D window will display all the disciplines cut by the plane using their current scalar colored with their own map. If any (non-current) discipline has a *mapped* domain surface, iso-surface or a programmed cut on when the planar cut is activated these tools are turned off.

A line probe will only display the scalar along the line for the current discipline.

- Time Color Map

There is only one color map used for displaying time (on particles).

4 Collaboration

The **pV3** suite supports full collaboration for Interactive Servers as well as the ability to query what extracts a Batch Server is collecting. The collaboration is designed to be as general as possible. Each Server runs independently, collecting its own data. This insures that all Servers are currently looking at the same snap-shot of the data. The communication between Servers is only for state information.

pV3's collaboration comes in many forms:

- Servers Window

When more than one Server (either Interactive or Batch) is running an additional window is opened listing the current pool and which hosts these **pV3** Servers are executing upon. The Server executing on this workstation is listed first in white. Each other Server is color coded. When all other servers have exited, this window is closed.

- Soft Cursors

When the cursor of an Interactive Server is in either the 3D, 2D or 1D windows, it is displayed in its place using the color coded for that host.

- View Locking

The 3D view can be locked to the view matrix used on any other Interactive Server in the pool.

- Steering Control

Only one Server can have steering control (the ability to send strings to the clients). Control is usually obtained by being the first Server in the pool. If there are any Batch Servers, then no Interactive Server can get control. This is to insure that the integrity of the batch run is maintained.

The Server in control must relinquish the steering ability before any other Server in the pool can obtain the control. The exception to this is if a Batch Server starts after an Interactive Server.

Server-to-Server messages can be sent to request that steering control be given up.

- Extract Viewing & Selection

A detailed list of the extracts active within a Server can be obtained by clicking in the appropriate box in the Servers Window. This allows selecting extracts not assigned to the current session but active in the other Server. Also, the plotting attributes can be linked to the attributes assigned to the other Server.

- Dynamic Tool Locking

By linking the plotting attributes of the 'Dynamic Surface' to another Server, this allows the complete functioning of the dynamic tools to be linked to that Server.

- Pause

The *pause* function of transient cases is made inoperative when there are multiple Servers. This is to insure that the entire system does not wedge.

4.1 Help Menus

4.1.1 Servers Window

The help menu that is printed when one types '?' in the servers window is as follows:

Servers Window

Mouse Buttons: any - Select

Key Strokes:

S - Send Active Server a String

Box:	1	2	3
	View Control	Extract Window	Steering Control

Comments:

- 1) The Active Server can be selected by clicking on the title of the host in the window. It is marked by an '*' after the type of server.
- 2) The view is locked to the Active Server by clicking in the top server's first box. This will only work for Active Interactive Servers.
- 3) The extracts currently collected in a server can be viewed by clicking on the second box. This opens up a Servers Extract Window associated with the host. To close the window, again click on the (now colored) second box.
- 4) Only one server can be in steering control at a time. Control is identified by a filled third box. Control can only be obtained when the server in control releases (clicks on) the filled box. If a Batch server is running it has control, therefore no interactive server can gain control.

4.1.2 Servers Extract Window

The help menu that is printed when one types '?' in the servers extract window is as follows:

Server Extract Window

Mouse Buttons: any - Select

Key Strokes:

PageDown - Move Extract List Up PageUp - Move Extract List Down

Box:	1	2
	Link Attributes	Include Extract

Comments:

- 1) Extracts are common to both the server associated with this window and this Interactive Server if the second box is filled. The name of the extract in the window's server is listed on the left of this list, where this server's names are listed on the right. If an extract is desired that is not currently

available, click on the second (unfilled) box. Please wait a couple of updates for the request to be satisfied.

2) The extract's plotting attributes can be controlled by the attributes set in other server by clicking on the first box. This will only work is the second box is filled.

5 Post-Processing

There are two types of output files generated by **pV3** activated by the appropriate keys in different windows. These files are compatible with the post-processors supplied with **Visual3**. The first type is tabular output which is generated from the 1D window. This is an ASCII file suitable for inclusion into most line plotting or spread-sheet software. The default file name is 'visualXYZ.tab'. A post-processing program 'tab2ps' produces Postscript output.

The second file type is an image dump of the entire screen or an individual window. This file is written in a FORTRAN unformatted manner with the default name of 'visualXYZ.img'. This can be converted into Postscript (using 'img2ps') or Macintosh TIFF files (using 'img2tiff') and viewed on the screen (using 'img2X').

The following section describes the usage of the post-processors supplied with **pV3** and **Visual3**. It should be noted that the source and make-files for the post-processors have been included with the distribution. By making small modifications to the sources, other output devices can be supported with no changes to **pV3**.

5.1 tab2ps

tab2ps takes as an argument a tabular output file name and produces PostScript on standard output. The output may be redirected to a file or piped directly to the printer spooler.

```
examples: % tab2ps visual002.tab | lp
          % tab2ps visual002.tab > tab.ps
```

5.2 img2tiff

img2tiff takes as its first argument the image file name, and the second argument is the output TIFF file name. The TIFF file can then be transferred to a Macintosh and be used by any application that can take TIFF as input, i.e. Adobe Photoshop.

```
example: % img2tiff visual001.img output.tiff
```

5.3 img2ps

img2ps takes as the first argument the image file name. Additional arguments are options controlling the translation from TrueColor to 8-bit grey scale. The output of img2ps is PostScript and is written to standard output. The output may be redirected to a file or piped directly to the printer spooler. The options are as follows:

- cps produce color PostScript output - this is the only option that will produce color, the default and the other options produce grey scale output
- r produce a red color separation
- g produce a green color separation
- b produce a blue color separation
- cxxxxxxx color mapping where "x" is either r, g, or b.
- i inverse intensity
- 4 4-bit grey scale

examples: % img2ps visual001.img | lp
 % img2ps visual001.img -r -i > img.ps
 % img2ps visual001.img -cps | lpr
 % img2ps visual001.img -crgbrgrg | lp

NOTES:

- 1) Options -r, -g, -b, -c and -cps are mutually exclusive.
- 2) If options -r, -g, -b, -c or -cps are not selected, the color translation defaults to 3 bits red, 3 bits green and 2 bits blue (-crrrggbb).
- 3) PostScript printers with only 2 MBytes may only be able to produce hard copy from the 2D window. Use the -4 option for the 3D window. Do not attempt full screen dumps unless your printer has alot of memory!

5.4 img2X

img2X may be used to view **pV3** image file(s) on the screen. It can display as many as 10 images and also write a **pV3** image file of the entire screen. img2X may have as many as 10 arguments, each should be the name of an image file. Optionally, each image can be compressed before drawn. This is done by appending '/n' to the end of the file-name (where n is the compression factor).

examples: % img2X visual001.img visual002.img/3
 % img2X visual001.img/2

NOTES:

- 1) The compression scheme used is very simple. If n=2, every other pixel is displayed, n=3 picks every third pixel.
- 2) An image window may be closed by hitting the key 'x' in that window.
- 3) An image file of the entire screen may be generated by hitting '~' in any image window.
- 4) img2X is properly terminated by clicking any mouse button while the cursor is in an image window.

6 User Interface Differences with Visual3

The following is a list of differences that an experienced **Visual3** user should note:

- Planar Cut

During reorientation of the plane (F3 in the 3D window) cut data is plotted in the 2D window in **pV3**. This cut data may lag behind the current position as accurately shown in the 3D window. Note: this may cause some confusion, remember to turn off the re-positioning (F3 in the 3D window) when the desired orientation is reached.

- Cell Visualization Tools

pV3 does not support the plotting options for Cell Based Scalars. **Visual3** should be used to view output of these functions.

- Histogramming

pV3 does not support histogramming or many functions that requires the server to look at the entire 3D data-set.

- Line output files

pV3 does not produce line files from the 3D and 2D windows for post-processing.

- Streamlines

The methods used for dealing with streamlines in **pV3** are much more general than in **Visual3**. This includes:

- A streamline object database
- Streamline probe is initiated from the database and may contains as many lines as streamlines in the object
- Each object has independent attributes (like the surface database)
- Streamlines are active during unsteady (*nonpaused*) states
- No streamer

See Instantaneous Streamlines in the Vector Visualization Tools, Section 2.4.2.

- Bubbles

The seeding methods used for particle paths are more numerous in **pV3**. These include:

- Seeding from streamline start locations (not requiring an active cut). Allows *time lines* if the object is built from a line.
- Streamer (cursor) location
- Line/boundary layer probe locations
- Tuft locations

NOTES:

(1) *Ghost* bubbles are not plotted.

(2) Bubbles can be colored by the seed time in **pV3**. See Bubbles in the Vector Visualization Tools, Section 2.4.1.

- Surface Streamlines and Bubbles

In **Visual3**, surface integrations were initiated when a *domain* surface was mapped and seeding was accomplished in the 2D window. Once the surface was unmapped, all streamlines/bubbles were deleted. In **pV3**, seeding surface streamlines and surface bubbles follow the above rules for a *domain* surface when it is mapped. The seed points/particles are not deleted when the surface is unmapped. Also, seeding can be accomplished without mapping by using any mouse click in the 1D window with an Edge Plot active. This is VERY usefull when the surface is too complex to map to the 2D window. The control of what vector/scalar fields (*special* surface or volume) are used for the integration/rendering of the surface streamlines and particles is performed by the surface function toggle ('s' in the 1D window).

7 Warning and Error Messages

- Warning: No Time-Out Set!

The time-out constant was set to no time-out, hard loops are used for both the server and client(s).

- Warning: Visual**xxx* not defined in .Xdefaults!

The font specified is not known to the X windows system. See Special Files, Section 1.1.3.

- Warning: *xxx*Client group size = *yyy*, only *zzz* task(s) active!

The specified **PVM** client group has members that are no longer active. This is usually do to some clients exiting ungracefully.

- Warning: *xxx* clients in group but only using: *yyy*

An 'init' message was only received from *yyy* clients but the total number of active tasks in the **PVM** group is *xxx*. It is possible that a task is not calling **pV_Update** (see the **pV3** Programmer's Guide). Also, it is possible that you may have to increase the Time-Out constant. See Environment Variables, Section 1.1.2.

- Warning: Illegal Message # *xxx* from *cid*

A non-**pV3** message was received from client *cid*.

- Warning - MIRROR multi-client mismatch!

MIRROR specified in pV_Init does not match between the various clients. Mirroring is turned off.

- Warning - FLIMS multi-client mismatch!

FLIMS specified in pV_Init does not match between the various clients.

- Warning: pV3 MAX clients exceeded - set to: *xxx*

The internal maximum number of clients was exceeded. The case will run with only *xxx* clients.

- Warning: Bad Status from client: *cid status*

The client specified has had some type of problem. The data from this iteration will not be plotted. *status* is the client error code.

- Warning - New Client trying to connect! *cid*

A client is attempting to connect to a running visualization session. The request is ignored and the session continues if *cid* is greater than the maximum *cid* at startup.

- Warning: Max Segs for SL # *xxx*

The maximum number of streamline segments has been reached for the specified streamline. The integration is aborted and the streamline is displayed unfinished. This reporting can be turned off by setting the environment variable *pV_Warning*, see Section 1.1.2.

- Warning: ACK from client: *cid tool err num*

The client *cid* has experienced a problem with the request for the extract *tool*. *err* is either -97 for an allocation error or -98 indicates that the instance *num* was not found. The result is that for this client the extract will not appear. This may not even be a problem if, for example, a cut (or iso-surface) does not pass through this client. This reporting can be turned off by setting the environment variable *pV.Warning*, see Section 1.1.2.

- Warning: Double Fill from client: *cid*

The specified client has sent two data streams for this iteration.

- Warning: Clients reporting different times!

Not all clients in a time-accurate application are reporting the same simulation time.

- Warning - Pending particle/SL inserts!

A Streamline group delete was requested while the server is processing inserts. The delete request is ignored.

- Warning - StreamLine deletes pending!

- Warning - Particle delete pending!

Inserting a group of Streamlines is invalid while the server waits for the pending deletes to complete. The request is ignored.

- Warning - Another surface delete pending!

Only one surface delete can be specified during an iteration. Additional requests are ignored.

- Warning - No Expose Event!

The server waited for a window to give an X Expose event and it did not happen!

- Warning (*routine*): *xxx* can not be transfered from *cid* to *cid*

During an integration, a request has been made to continue to a task that does not exist or for some reason can not be reached.

- Warning (*routine*): Bad client id: *cid*

A non-existent task was targetted for an integration transfer.

- Warning (*routine*): DisciplineID *did* out of range.

The discipline index *did* is not valid.

- Warning (*routine*): partID *xxx* out of range.

An illegal particle number was encountered during a inter-client transfer.

- Warning (*routine*): partID *xxx* NO History!

A particle that has had no prior history is requesting a transfer.

- Warning (*routine*): SLXferID *xxx* out of range.

An illegal streamline number was encountered during a inter-client transfer.

- Warning (*routine*): SLXferID *xxx* NO History!
A streamline that has had no prior history is requesting a transfer.
- Warning (*routine*): SLXferID *xxx*, client *yyy*, unmark ≤ 0
An error occurred removing a streamline from the active list.
- Warning: different ZPrimes from clients!
A programmed cut has been activated and the startup value for ZPrime is different for one or more clients in a multi-client case. The resultant cut will not be continuous through the patched volume.
- Warning: Illegal Surface Type: *xxx*
The startup extract (from the startup file) has the type *xxx*. Only the values 2 (Planar), 4 (programmed) or 7 (iso-surface) are valid. The entry is ignored.
- Warning: No Startup for Surface SLs!
Surface streamlines are not supported for restart (via the startup file). The entry is ignored.
- Warning: No Startup - Extract Type: *xxx*
The startup extract (from the startup file) has the type *xxx*. This type is not supported.
- Warning: Startup File does NOT exist!
The startup file is not found, therefore **pV3** runs from the default startup state.
- Warning: Startup File too Old!
The startup file specified was written by a pre-Rev 1.20 version of the **pV3** server and this is a multi-discipline case. The server can not determine which discipline to apply the extracts. No extracts are used.
- Warning: Startup File – Discipline not Found: *discipline*
The discipline *discipline* specified in the startup is not one of the disciplines in the running simulation. No extracts are used for this part of the startup file.
- Warning - in pVSetExtract: index, exnum = *xxx yyy*
Programmed extracts have been registered, but the routine **pVSetExtract** has not been supplied. REQMASK and PLOTMASK are set to zero.

- Error - Cannot attach to lock arena!
The **pV3** server cannot set-up the lock arena to the file ‘.pV3.locks’. Is there write access in the current directory?
- Error - getting new lock!
There is some problem initiating a new lock. Try removing the file ‘.pV3.locks’.
- pV3 Error - No response from any clients!
The **pV3** clients currently running have not responded within the time-out constant.
- Error: pV3 client at different Rev than server! - id = *cid*
A client/server mismatch. You must rebuild the clients with the library that matches the server!
- Error: Invalid Startup - Check Client Revs!
The initialization messages from the client(s) have **pV3** signatures, but are invalid.
- Error: In parsing the Startup File!
This is not a fatal error but statement that there was some error in reading the startup file. The contents, from the error on, are ingored. The server continues to run.
- Error: Discipline ID Out of Range! *did*
A client has reported *did* for the discipline index which is invalid.
- Error: Client ID Out of Range! (*xxx*, 1-*max*)
A client is reporting an id of *xxx* which is not in the valid range.
- Error: Client ID Already Taken! *cid*
Client id *cid* has already been assigned. There is something wrong with the client pool.
- Error: Client OPT Out of Range! (client = *cid*) - *OPT*
Client id *cid* has specified an invalid form of unsteadyness.
- Error: Mix of pV3 Steady-state and Unsteady Clients!
You can not mix steady-state **pV3** clients with any non-zero IOPT clients.
- Error: Too Many Disciplines!
A client has started up a new discipline that requires an index greater than the number currently supported (now 8).
- Error in Allocating Client Table!
Memory allocation failed for the Client Table.
- Error in Allocating Common Region!
Memory allocation failed for the FORTRAN common region for a discipline.

- Error: No Saved Discipline COMMON Region to Load!
FORTRAN common region for a discipline was not found.
- Error: Number of Fields Mis-Match! ($xx \neq yy$)
The number of field variables returned from a client is of the wrong length. This is not fatal.
- Error Starting Thread for multi-processing!
The new thread for the visualization could not be initialized.
- Error - Client(s) have exited!
One or more clients have exited from a running visualization.
- Error in Memory Allocation!
The server has requested a block of memory and has been refused. This is usually do to the problem's size. Either wait until the workstation was fewer tasks running or find a bigger (more swap space) machine.
- Error - Time-Accurate multi-client mismatch!
- Error - NPGCUT multi-client mismatch!
- Error - TPGCUT multi-client mismatch!
- Error - NKEYS multi-client mismatch!
- Error - IKEYS multi-client mismatch!
- Error - FKEYS multi-client mismatch!
The specified pV_Init data does not match between the various clients for a case that has more than one client.
- Error during Particle Initialization!
The server has requested memory for the particle tracking and has been refused. Either wait until the workstation was fewer tasks running or find a bigger (more swap space) machine.
- Error during SL Transfer Initialization!
The server has requested memory for the streamline tracking history and has been refused. Either wait until the workstation was fewer tasks running or find a bigger (more swap space) machine.
- Singular matrix.
This occurs while processing the view transformation matrix. It is considered illegal to have a singular transformation matrix. The server should not produce this condition. It usually happens when the setup file has been corrupted. This can also happen if all the coordinate data passed to the server is identical (the same XYZ position for all nodes).
- Error - File does not exist!
The requested file does not exist.

- Error - Not a TrueColor Image!
The image file requested for the comparison window is not a TrueColor image (it is Pseudo-Color).
- Error - Image depth mismatch!
The image file requested for the comparison window has the wrong color depth.
- Error in ImageFile!
An error has occurred during the image file read.
- Error in Reference File - NOT in Tabular form!
A reference line file was specified that was not in **Visual3** .tab format.
- Discipline Error!
The discipline sub-system has not been properly initialized.
- ERROR in (SUB-)EXTRACT allocation
Memory could not be allocated for the (sub)extract subsystem.
- ERROR in REGISTRY allocation
Memory could not be allocated for the registry of programmed extracts.
- Extract NSEG ERROR: sub = *xxx*, nseg = *yyy*, max = *zzz*
A streamline sub-extract has been received with an illegal segment number. This is not fatal but something is wrong!
- Extract TID ERROR: sub = *xxx*, nseg = *yyy*, ic = *zzz*
A streamline sub-extract has been received with a *cid* for segment *yyy*, but a previous message for that segment had the client number *zzz*. This is not fatal but something is wrong!
- Extract Client ERROR: sub = *xxx*, cid = *cid*
The client tid is not in the active list. This is not fatal but something is wrong!
- Extract ERROR: type = *www*, sub = *xxx*, size = *yyy*, len = *zzz*
A sub-extract has been allocated for *yyy* words, but the message has *zzz* words. This is not fatal but something is wrong!
- ERROR: Timed out waiting for Init Hand-Shake!
- ERROR: Timed out waiting for Client Hand-Shake!
The time-out constant has been exhausted before response from the client(s). Increase the constant by the environment variable 'pV3_TO'.
- ERROR - ColorMap OverFlow!
The requested number of colors specified in the colormap file exceeds the internal colormap storage.

- ERROR - ColorMap File Error!

An error occurred parsing the colormap file.

- ERROR - Memory Allocation for Prg Cuts!
- ERROR - Memory Allocation for Key Bindings!
- ERROR: ColorMap Memory Allocation!
- ERROR: Particle Memory Allocation!
- ERROR: Tuft Memory Allocation!
- ERROR: Chart Memory Allocation!

The server has requested a block of memory and has been refused. This is usually due to the problem's size. Either wait until the workstation has fewer tasks running or find a bigger (more swap space) machine.

- KeyBoard File Does NOT Exist!

The environment variable 'Visual_KB' has been set and the file indicated does not exist.

- ERROR reading KeyBoard File!
- ERROR E-O-F in KeyBoard File!

The environment variable 'Visual_KB' has been set and there has been an error parsing the data in the file.

- pV3: ERROR pvmd Not running!

The **PVM** system has not been initiated.

- pV3: ERROR No pV3 Clients running!

The server finds no clients.

- pV3: ERROR Allocating Memory for Key Bindings (*NKEYS*)

The data structures required for *NKEYS* field variables could not be allocated.

- pV3: ERROR Allocating Memory for Particle Tracking!

The data structures required for tracking particle movements could not be allocated.

- pV3: ERROR Allocating Memory for SL Tracking!

The data structures required for tracking streamline movements could not be allocated.

- pV3: bufinfo error: *xxx, yyy*

This is not a fatal error but a receive message buffer is giving an error indication. The message is ignored.