Advanced Programmer's Guide

for
VISUAL3 Rev. 2.35

Bob Haimes

December 17, 1998

DISCLAIMER: This programming interface is maturing. Attempts will be made not change anything already defined. In the future, additional options may be added to V3_GETSTATE, V3_SETSTATE and V3_GETSTRUC.

Sections marked with this change-bar are have had a major change from last release (Rev 2.30) and may require some programming modifications.

If you use these new features of VISUAL3, please report to me any feedback, in particular, any functions that seem to be missing.

License

Contents

1 Introduction

Historically, the control of the VISUAL3 visualization session has been driven by the user. Much time and effort was placed in making that interaction 'real-time'. But there are many instances when you may want the control of what is rendered on the screen to be specified by program. Examples are demonstrations, visual results form expert-systems, macros and etc.

In that VISUAL3 is an X windows application, control is driven by X events. Figure 1 shows VISUAL3's normal processing loop. In general, all X events are extracted from the X event queue and the internal VISUAL3 state information is adjusted appropriately. When all events are exhausted, any changes to state that require getting data from the application cause the appropriate routines to be called (e.g. V3SCAL, V3VECT, and etc.). After this information collection phase is done, the windows that require updating are redrawn in the data rendering phase. Then the event queue is checked for more user interaction, and this continues for the life of the VISUAL3 application.

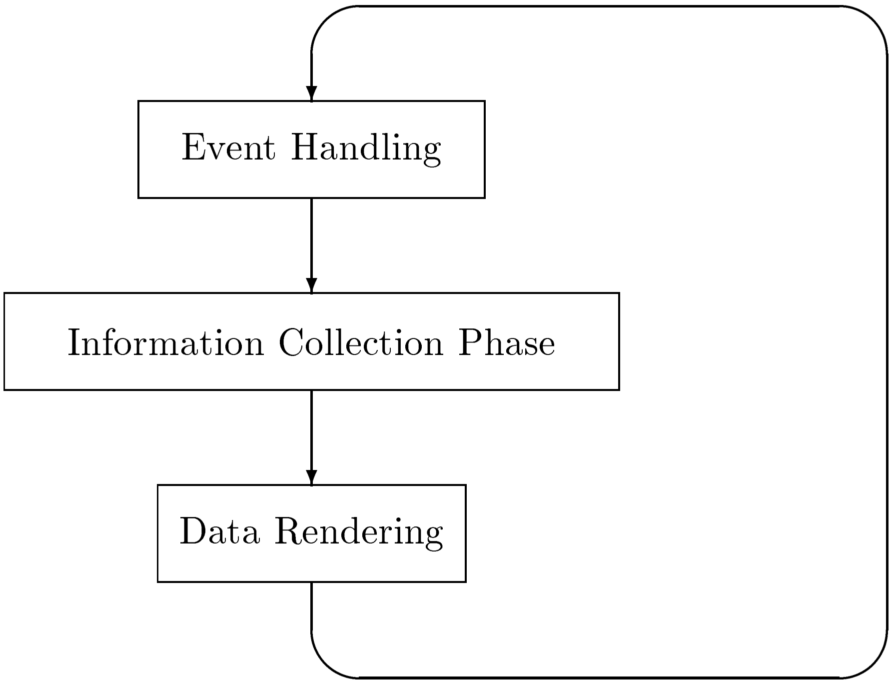

Figure 1: VISUAL3's Internal Loop

Also, in the past VISUAL3 has only exposed it's internal data to the programmer when it required data. The connectivity information or the data generated to produce cuts and other objects was hidden. This document also describes how the advanced VISUAL3 programmer can access this information.

2 VISUAL3 Event Handling

One of the goals of allowing programming control in VISUAL3 is to maintain the current action as the default. Also this must allow the programmer to either modify the effects of an event or completely take over.

Before attempting to program VISUAL3 at this level it is important to understand how the program structure has changed to allow taking control. This includes the internals of the event handling phase (that in the past has been hidden from the programmer).

The details of the VISUAL3 event handling phase can be seen in Figure 2. First, an X event is pulled of the X event queue. The event is described by a window ID - $IWIN$ (the window inwhich the event occurred), the type of event (e.g, KeyPress, ButtonPress, and etc.), the location in the window of the event, and finally any state data (e.g., which button or key was pressed). There is a special window ID flag of -1 which indicates that the event queue was empty.

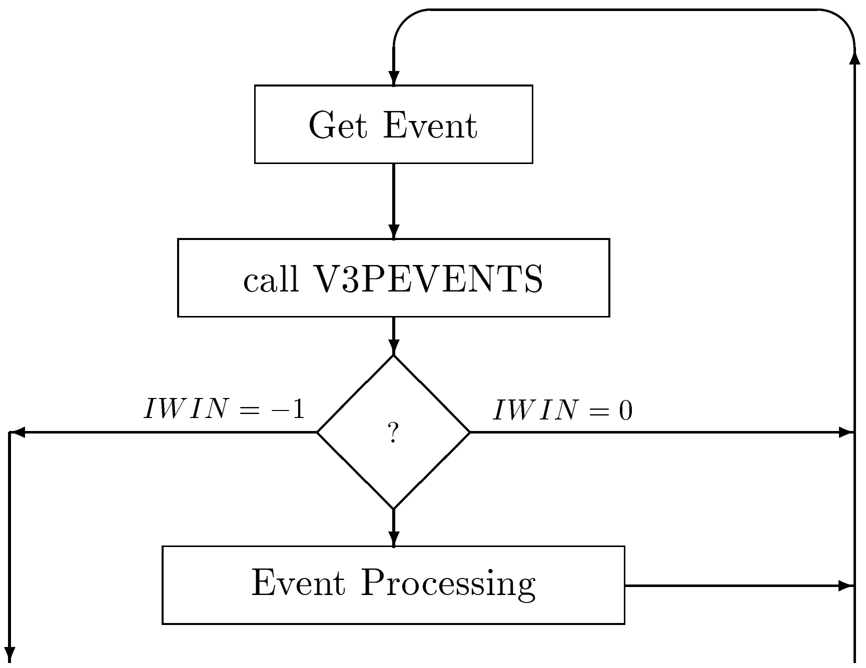

Figure 2: VISUAL3's Event Handling

Next a programmer-supplied routine V3PEVENTS is called with the event data. V3PEVENTS has the option of passing the event on (doing nothing), performing some action based on the event, changing the event (by modifying the arguments) or having the event ignored (by setting $IWIN$, the window ID, to 0). The other routines described in this document may be called within V3PEVENTS to inquire and set many of the VISUAL3 internal state variables.

Based on the value of $IWIN$ as returned by V3PEVENTS; the event processing section of VISUAL3 is called when $IWIN > 0$, the event is ignored if V3PEVENTS sets $IWIN = 0$, or this phase is terminated with $IWIN = -1$. When changing the action of VISUAL3 it is important to remember that nothing will be updated in the windows until V3PEVENTS returns with $IWIN = -1$. Also, if V3PEVENTS was called with $IWIN = -1$ and an event is constructed and passed through to the event processing section, V3PEVENTS will be re-entered (possibly with $IWIN \neq -1$), before the screen gets updated.

Examples of V3PEVENTS, that perform various functions, are contained within this distribution.

It is also possible to use VISUAL3 is a completely batch manner. In doing this no windows are open (infact the link is modified so that the graphics libraries normally required are no longer needed). Because the entire X windows method for getting events has been removed, the control of VISUAL3 is only by the procedures described in this document. The control structure is the same, but the drawing functions are NO-OPS.
It should be noted that srceen and window dumps are not possible, in that there is no graphics to capture.

A simple example with *Makefile* is provided in the distribution under the 'batch' subdirectory in the 'adv' directory.

3 The Use of Pointers in VISUAL3 with FORTRAN

Some of the data that gets returned from V3_GETSTRUC is in the form of pointers to blocks of memory. There are two modes that the pointers may be handled:

- %val

  A mechanism exists on all major workstation's FORTRAN (SGI, DEC, IBM, HP and SUN) to allow the pointer to be passed to a SUBROUTINE or FUNCTION and then have the memory treated as a normally declared vector or array. This is done by the VAX extension '%val(pointer)' used in the CALL or function invokation. When the pointer is passed to the sub-program by 'value', it is equivalent to passing a variable by 'reference' (the FORTRAN method). That is, in both cases, the address of the memory of interest is placed in the stack!

  In this case, the pointers are treated as INTEGER variables. This causes a problem in portability. The pointers must be treated as INTEGER*4 variables on all 32 bit machines. On 64 bit machines (DEC ALPHAs and SGIs running IRIX 6.0, or higher, in native mode) the pointers are INTEGER*8.

- POINTER statment

  The POINTER statement supported by most FORTRANs removes the portability problem. This syntax allows the coupling of a pointer with an array.

Using either mechanism, sophisticated VISUAL3 enhancements may be performed using FORTRAN. To complete this picture, two additional entries to VISUAL3 have been added to allow the FORTRAN programmer to allocate and free up blocks of memory. An example of using these memory functions can be seen in the distribution in the 'src/PLOT3D' directory.

3.1 MALLOCV3

PTR = MALLOCV3(NBYTES)

This function is equivalent to the C routine 'malloc'. It allocates a block of memory and returns the pointer to the block. On 64 bit machines, this function may need to be declared INTEGER*8.

NBYTES:i: I	The number of bytes to allocate.
PTR:o: I	The address of the block (0 is an error indicator).

3.2 FREEV3

FREEV3(PTR)

This function is equivalent to the C routine 'free'. It deallocates a block of memory. NOTE: Only free up blocks of memory that YOU allocate!

PTR:i: I	The address of the block.

3.3 REALLOCV3

PTRN = REALLOCV3(PTRO, NBYTES)

This function is equivalent to the C routine 'realloc'. It re-allocates a block of memory and returns the pointer to the new block. On 64 bit machines, this function may need to be declared INTEGER*8.

PTRO:i: I	The address of the block.
NBYTES:i: I	The number of bytes to allocate.
PTRN:o: I	The address of the new block (0 is an error indicator).

4 Structures

4.1 3D Data Structures

In general, this data is the information returned from the programmer-supplied routines, though the connectivity tables are internally generated. The structures are exposed so that multiple copies of the data need not be kept.

4.1.1 Node based

The following data is based on the 3D node space:

XYZ: R(3,NNODE)	(x, y, z)-coordinates of grid nodes.
S: R(NNODE)	Scalar function values.
T: R(NNODE)	Threshold function values.
V: R(3,NNODE)	Vector function values (Vx, Vy, Vz).
Z: R(NNODE)	Current z' values for planar or programmed cuts.
IBLANK: I(*)	IBLANK values for structured blocks. The length is the number of nodes in the structured blocks. For a case that is only structured blocks the length is NNODE. If the pointer to this array is 0, then no IBLANKing is active.

4.1.2 Cell based

The following data is based on the 3D cell space:

C: R(NCELLS)	Cell scalar function values.
CEL1: I(4,KCEL1)	Node pointers for tetrahedral cells.
CEL2: I(5,KCEL2)	Node pointers for pyramid cells.
CEL3: I(6,KCEL3)	Node pointers for prism cells.
CEL4: I(8,KCEL4)	Node pointers for hexahedral cells.
NPTET: I(8,KNPTET)	Poly-Tetrahedra strip header: **NPTET(1,n)** = the pointer to the end of the strip n, i.e. it points to the last entry in PTET for the poly-tetrahedral strip **NPTET(2,n)** = the first node in the poly-tetrahedra

NPTET(3,n) = the second node in the poly-tetrahedra

NPTET(4,n) = the third node in the poly-tetrahedra

NPTET(5,n) = the fourth node in the poly-tetrahedra

NPTET(6,n) = the last node in the poly-tetrahedra

NPTET(7,n) = first connection (face 1 of the first tetra)

NPTET(8,n) = last connection (face 2 of the last tetra)

PTET: I(KPTET) — The rest of each poly-tetrahedra, 1 node per cell. The first and last node numbers in PTET are replaced by a flag to indicate the start and end of the strip. The flag is the negative of the strip number so that if you are in a strip, you can find which strip it is.

BLOCKS: I(6,KNBLCK) — Structured block definitions:

BLOCKS(1,m) = NI

BLOCKS(2,m) = NJ

BLOCKS(3,m) = NK

BLOCKS(4,m) = cell number that terminates the block

BLOCKS(5,m) = node number that terminates the block

BLOCKS(6,m) = index into CBLOCK that starts the connectivity table for the block.

4.1.3 Domain Surfaces

The following data is based on the surface space:

NSURF: I(2,KNSURF) — NSURF(1,n) is the pointer to the end of domain surface group n, i.e. it points to the last entry in both SURF and SCEL for that group.
NSURF(2,n) is the startup drawing state (see the Appendix).

SURF: I(KSURF) — pointers to cells+faces that make the surfaces.
SURF = 8*(cell number) + face number.

SCEL: I(4,KSURF) — node numbers for surface faces. For quadrilateral faces SCEL must be ordered clockwise or counter-clockwise; for triangular faces, SCEL(4,n) must be set to zero.

TSURF: C*20(KNSURF) — titles for domain surfaces.

4.1.4 Domain Surface Edges

The following data is based on the surface edge space:

NSED: I(2,KNSED)	NSED(1,n) is the pointer to the end of the surface edge n, i.e. it points to the last entry in SED for that group. NSED(2,n) is the surface group number associated with this edge. NOTE: a domain surface may have as few a zero edges (the surface closes totally upon itself) and can have more than one edge!
SED: I(*)	pointers to nodes that make the surface edge. Each node connects to the last (except for the first entry) in a polyline structure.

4.1.5 Connectivity

The connectivity tables are used by the particle path algorithms and other tools that require neighborhood data. Any connection (cell number) that is negative indicates that there is no neighboring cell and the value is the index into SURF and SCEL for the surface face. If the pointers to all of these structures are 0, then this information has been deallocated.

CCEL1: I(4,KCEL1)	cell indices for the neighbors touching each of the four faces of the tetrahedron.
CCEL2: I(5,KCEL2)	cell indices for the neighbors touching each of the five faces of the pyramid.
CCEL3: I(5,KCEL3)	cell indices for the neighbors touching each of the five faces of the prism.
CCEL4: I(6,KCEL4)	cell indices for the neighbors touching each of the six faces of the hexahedon.
CPTET: I(2,KPTET)	The neighbor for face 3 and face 4 for each tetrahedra in the strip. Face 1 and face 2 are implicit (the last and next cells) except for the beginning and end of the strip where the connectivity can be found in the header NPTET.
CBLOCK: I(*)	A vector that contains the connectivity data (the cell index) for the husk of the block. The start index for a particular block is in BLOCKS(6,m).

face 1 - $k = 1$

first $(NI-1)*(NJ-1)$ entries indexed in a nested set of loops with j as the outer and i as the inner.

face 2 - $i = NI - 1$

next $(NJ-1)*(NK-1)$ entries indexed in a nested set of loops with k as the outer and j as the inner.

face 3 - $j = NJ - 1$

next $(NI-1)*(NK-1)$ entries indexed in a nested set of loops with k as the outer and i as the inner.

face 4 - $i = 1$

next $(NJ-1)*(NK-1)$ entries indexed in a nested set of loops with k as the outer and j as the inner.

face 5 - $k = NK - 1$

next $(NI-1)*(NJ-1)$ entries indexed in a nested set of loops with j as the outer and i as the inner.

face 6 - $j = 1$

next $(NI-1)*(NK-1)$ entries indexed in a nested set of loops with k as the outer and i as the inner.

CCEL: I(5,KSURF) — surface indices for the neighbors touching each of the three or four faces of a surface face. The fifth entry contains the number of the 3D cell that contains the face.

4.2 2D Data Structures

This data is generated by the VISUAL3 scalar tools (planar cuts, programmed cut surfaces, iso-surfaces and mapped domain surfaces). This data is exposed so that new 'probes' may be easily generated. The size of many of these arrarys (and therefore the pointers) can change during the execution of VISUAL3, so when using this data, get the current pointers before accessing the memory.

4.2.1 Node based

The following data defines the '2D node' space. Where the current number of nodes is KNODE2. '2D nodes' are the intersection of a cut or surface and 3D cell edges.

NODE2D: I(3,KNODE2)	'2D node' definitions. **NODE2D(1,n)** = first 3D node index for the edge of the cell. **NODE2D(2,n)** = second 3D node index (not used for mapped domain surfaces). **NODE2D(3,n)** = clip mask (for planar cuts only): 1 - less than xmin 2 - greater than xmax 4 - less than ymin 8 - greater than ymax
XY: R(2,KNODE2)	(x', y')-coordinates of the '2D nodes'.
DTERP: R(KNODE2)	is a number between 0.0 and 1.0 that defines the distance from the first 3D node in NODE2D (not used for mapped domain surfaces).

4.2.2 Cell based

The following data defines the '2D cell' space that collectively construct dynamic and static surfaces in VISUAL3. These surfaces are composed of KCELL disjoint triangles.

CELL: I(3,KCELL)	'2D cell' definitions. Each entry contains a '2D node' index.
ICELL: I(KCELL)	3D Cell index that refers back to the cell that produced the disjoint triangle.

4.2.3 Grid based

The following data defines the intersection of the 3D cells and the cut or surface. This is used for plotting the mesh and locating the intersection of the cut and the bounding domain (edges). The number of disjoint mesh segments is KFACE and KEDGE is the number of disjoint edge segments.

FACE: I(2,KFACE)	'2D mesh' definitions. Each entry contains a '2D node' index.
EDGE: I(3,KEDGE)	'2D edge' definitions. **EDGE(1,n)** = first '2D node' index **EDGE(2,n)** = second '2D node' index **EDGE(3,n)** = domain surface group index
CEDGE: I(KEDGE)	3D cell index associated with the edge.

4.3 1D Data Structures

The data presented here is generated by VISUAL3's stream-liner. This data is composed of the information necessary to display the streamline and optionally color it, as well as define the local curl (for ribbons) and the cross-flow divergence (for tubes).

There may be many active streamlines, each has it's own length, KSTREAM. To select a streamline use V3_SETSTATE with OPT = 11.

4.3.1 3D Streamlines

The following data items define all information associated with 3D streamlines:

CELLSL: I(KSTREAM)	3D Cell index that contains the point.
WEIGHTSL: R(8,KSTREAM)	Weights applied to the nodes of the cell to define the point. Only the first 4, 5, or 6 entries are used for tetras, pyramids, and prisms respectively. Each weight is a number between 0.0 and 1.0 and should be multiplied by the cell's nodal values to interpolate the point data. This is applied to the coordinate triads to get the point's position, the scalar values to calculate the scalar function at the point, and etc.
TIMESL: R(KSTREAM)	Integration time (from first point), for animation.
ANGLESL: R(KSTREAM)	Angle of the twist for ribbons in degrees.
DIVSL: R(KSTREAM)	Divergence. When used for Tube rendering, the size of the tube is based on a set starting size mutiplied by **e** to this power.

4.3.2 Surface Streamlines

The items below define all information associated with surface streamlines:

CELLSL: I(KSTREAM)	2D triangle index to CELL that contains the point.
WEIGHTSL: R(2,KSTREAM)	Weights applied to the nodes 1 and 2 of the triangle that defines the point. The weight for point 3 is simply one minus the two weights.
TIMESL: R(KSTREAM)	Integration time (from first point), for animation.

5 Supplied Routines and Calls

Consistant with the VISUAL3 naming convension, the routines that are part of VISUAL3 are prefixed with 'V3_', those that are supplied by the programmed start with 'V3'.

5.1 V3PEVENTS

V3PEVENTS(IWIN,TYPE,XE,YE,STATE)
This subroutine is called by VISUAL3 immediately after an X event has been pulled of the queue. This routine may perform certian functions based on the event, change the event status, have VISUAL3 ignore the event or just do nothing.

IWIN:i/o: I — The X window ID of the event or:

- **IWIN=-1** No more events to process, therefore pass the control of VISUAL3 to data collection phase
- **IWIN=0** Output only. Ignore this event and pull the next one off the queue

TYPE:i/o: I — The event type. The event types that VISUAL3 windows can generate are:

- **TYPE=2** KeyPress
- **TYPE=3** KeyRelease
- **TYPE=4** ButtonPress
- **TYPE=5** ButtonRelease
- **TYPE=12** Expose
- **TYPE=14** NoExpose
- **TYPE=33** ClientEvent

XE:i/o: I — The X pixel location in the window of the event (0 is the left)

YE:i/o: I — The Y pixel location in the window of the event (0 is the top of the window)

STATE:i/o: I — The event state. For Key events, the state is the X KeySym number (usually the ASCII code except for the special keys). For mouse button events, the state is the button number.

5.2 V3_GETSTATE

V3_GETSTATE(OPT,IVEC,RVEC,STRING)
Returns the internal VISUAL3 state. This VISUAL3 routine can be called from any programmer-supplied code.

OPT:i: I	Option set to specify what data to get
IVEC:i/o: I()	Integer data returned based on OPT (length also determined by OPT). Some OPTs control actions on input.
RVEC:o: R()	Real data returned based on OPT (length also determined by OPT)
STRING:o: C	Character data returned based on OPT

- OPT = -4 - 2D window cursor translation to 3D. This works for the Planar and Programmer defined cuts only. On input IVEC(1) must contain the 2D window X pixel value and IVEC(2) the Y pixel value. These can be the values **XE** and **YE** from **V3PEVENTS**. On output:

  IVEC(3) = 3D Cell index - 0 is an error indication.

  RVEC(1-3) = XYZ position - valid for positive IVEC(3).

- OPT = -3 - Convert Cell/Surface index to node indices. On input, IVEC(1) is the cell index (or if negative, the surface facet index). On output:

  IVEC(1) = Number of nodes - 0 is an error indication.

  IVEC(2-9) = Node indices - length filled based on IVEC(1).

- OPT = -2 - Get unsteady info:

  IVEC(1) = IOPT

  IVEC(2) = Pause State - 0 not paused, 1 paused

  RVEC(1) = Current Simulation time

- OPT = -1 - Get special key bindings. IVEC is filled with 32 integers that are the X KeySyms for the non-ASCII keys used by VISUAL3. A file 'KeyBoard' contains the default VISUAL3 KeySyms and the associated labels. These values may be modified and used by VISUAL3 if the environment variable 'Visual_KB' is set to point to the file containing the new bindings.

- OPT = 0 - Get Rev number and X window IDs:

  IVEC(1) = 1D Window

  IVEC(2) = 2D Window

  IVEC(3) = 3D Window

  IVEC(4) = Dials Window

  IVEC(5) = Key Window

  IVEC(6) = Text Window

  IVEC(7) = Comparison Window - 0 means the window is not open

  IVEC(8) = Root Window

  RVEC(1) = Revision Number

- OPT = 1 - Get the current transformation matrix for the 3D view. 12 words of RVEC are used. If 'cut-plane positioning' is on (see OPT = 3) then the transformation matrix for the plane is returned. This matrix starts at the view transformation matrix and is adjusted until the positioning is terminated.

- OPT = 2 - Get the current planar cut coordinates. 9 words of RVEC are used for 3 of the 4 corners of the cut plane in 3 space. If 'cut-plane positioning' is on (see OPT = 3) then the values returned are for the planar coordinates before the positioning began.

- OPT = 3 - Get the current scalar tool status. If IVEC(1) is NOT equal to -1 when V3_GETSTATE is called, the compute to search for ZPRIME min and max is performed when appropriate.

  IVEC(1) = Current Tool :

  0 - No tool

  1 - Programmer controlled 2D window

  2 - Planar cut

  3 - Planar cut positioning on

  4 - Programmed cut

  5 - Mapped surface

  6 - Mapped surface with surface functions

  7 - Iso-surface

  8 - Programmer controlled 2D window & Iso-surface

RVEC(1) = Current ZPRIME - for 2, 4 and 7 above

RVEC(2) = XPC - for 4, 5 and 6

RVEC(3) = YPC - for 4, 5 and 6

RVEC(4) = HALFW - for 4, 5 and 6

RVEC(5) = 2D Rotation Angle in degrees - for 4, 5 and 6

RVEC(6) = ZPRIME Minimum (IVEC(1) $\neq -1$ when called) - for 2, 4 and 7

RVEC(7) = ZPRIME Maximum (IVEC(1) $\neq -1$ when called) - for 2, 4 and 7

- OPT = 4 - Lighting and Mirroring state:

IVEC(1) = Mirror status - 0 = off

1 - Mirror about X = 0.0

2 - Mirror about Y = 0.0

3 - Mirror about Z = 0.0

IVEC(2) = Lighting status - 0 = off, 1 = on

IVEC(3) = Shading level - a value between 0 (no lighting) and 255

RVEC(1-3) = Lighting vector

RVEC(4-6) = Shading background color (r,g,b) - values between 0.0 and 1.0

- OPT = 5 - Scalar state:

IVEC(1) = Binding index

IVEC(2) = Not Used

IVEC(3) = Minimum scalar function node number

IVEC(4) = Maximum scalar function node number

RVEC(1) = Minimum for display - fmin shown in the key window

RVEC(2) = Maximum for display - fmax shown in the key window

RVEC(3) = Minimum scalar function corresponding to IVEC(3)

RVEC(4) = Maximum scalar function corresponding to IVEC(4)

STRING = Scalar title - maximum of 32 characters used.

- OPT = 6 - Vector state:

  IVEC(1) = Binding index

  IVEC(2) = Minimum vector magnitude node number

  IVEC(3) = Maximum vector magnitude node number

  RVEC(1) = Vector scale - the scale factor for tufts and arrows

  RVEC(2) = Minimum vector magnitude corresponding to IVEC(2)

  RVEC(3) = Maximum vector magnitude corresponding to IVEC(3)

  STRING = Vector title - maximum of 32 characters used.

- OPT = 7 - Threshold state:

  IVEC(1) = Binding index - negative value indicates the appropriate scalar index

  IVEC(2) = Vector Cloud status - 0 = off, 1 = on

  IVEC(3) = Minimum threshold function node number

  IVEC(4) = Maximum threshold function node number

  RVEC(1) = Minimum for display - tmin shown in the key window

  RVEC(2) = Maximum for display - tmax shown in the key window

  RVEC(3) = Minimum threshold function corresponding to IVEC(3)

  RVEC(4) = Maximum threshold function corresponding to IVEC(4)

  STRING = Threshold title - maximum of 32 characters used.

- OPT = 8 - User response status:

  IVEC(1) = First mouse button action - 0 = no button press

  IVEC(2) = X pixel location for action

  IVEC(3) = Y pixel location for action

  IVEC(4) = Second mouse button action

  IVEC(5) = X pixel location for action

  IVEC(6) = Y pixel location for action

  STRING = Text answer

  The data in IVEC is valid for those operations that require snapping a line/circle or a 'rubber-band' box.

- OPT = 9 - Cell scalar state:

  IVEC(1) = Binding index

  IVEC(2) = Not Used

  IVEC(3) = Minimum function cell number

  IVEC(4) = Maximum function cell number

  RVEC(1) = Minimum for display - cmin shown in the key window

  RVEC(2) = Maximum for display - cmax shown in the key window

  RVEC(3) = Minimum cell function corresponding to IVEC(3)

  RVEC(4) = Maximum cell function corresponding to IVEC(4)

  STRING = Cell scalar title - maximum of 32 characters used.

- OPT = 10 - Current domain/static surface state:

  IVEC(1) = Surface index

  IVEC(2) = Current drawing state - see the Appendix

  IVEC(3) = Surface type :

  1 - Domain surface

  2 - Planar cut
  RVEC is filled with the 9 words (for the corners) at the time of the cut. This is like OPT = 2.

  4 - Programmer-defined cut
  RVEC is filled with the 5 words like OPT = 3.

  7 - Iso-surface
  RVEC(1) contains the iso-surface value.

  IVEC(4) = Maximum number of surfaces

  IVEC(5) = Number of domain surfaces

  STRING = Surface title - maximum of 20 characters used.

- OPT = 11 - Current streamline state:

  IVEC(1) = Current streamline index - 0 no streamline active.

  IVEC(2) = Current number of streamlines

  IVEC(3) = Particle number for streamline - a negative number indicates a surface streamline. This index is unique for the streamline and will not get reused when the line is deleted.

  IVEC(4) = Animation flag - 0 no animation, 1 - currently animating

  RVEC(1) = Ribbon rotation in degrees

  RVEC(2) = Ribbon/Tube size

  RVEC(3) = Tube Maximum size

- OPT = 12 - Current bubble state:

  IVEC(1) = Current bubble index - 0 no bubble active.

  IVEC(2) = Current number of bubbles

  IVEC(3) = Particle number for the bubble - This index is unique for the bubble and will not get reused when the bubble leaves the domain. A negative number is the indication of a surface particle.

  IVEC(4) = Cell index that contains the bubble - 0 indicates that the bubble is leaving the domain. For a surface particle this is the surface face index.

  RVEC(1-8) = Weights - Weights applied to the nodes of the cell that define the bubbles position. Only 1-4 are used for surface particles. In this case the weights are associated with the nodes that support that surface face.

- OPT = 13 - Current probe state:

  IVEC(1) = 1D plot type :

  0 - no plot

  1 - StreamLine Probe

  2 - Strip Chart

  3 - Line Probe

  4 - Edge Plot

  5 - Surface Layer Scan

  6 - Programmer-defined probe

  7 - Histogram on node scalar

  8 - Histogram on cell based scalar

IVEC(2) = Point Probe status - 0 (off) or 1 (on)

IVEC(3) = Reference line status - 0 (off) or 1 (on)

RVEC(1) = X Minimum for plot

RVEC(2) = X Maximum for plot

RVEC(3) = Y Minimum for plot

RVEC(4) = Y Maximum for plot

- OPT = 14 - Global drawing state:

  IVEC(1) = Edge Outline status - 0 (off) or 1 (on - default)

  IVEC(2) = Transparent Plane - 0 (off) or 1 (on - default)

  RVEC(1-3) = Focus - Values subrtacted from each node coordinate in 3 space before the transformation matrix is applied. This position is usually the center of the sphere that encloses the object(s). This allows applying the same transformation matrices to different data-sets.

  RVEC(4) = Object Radius - The radius of the sphere that encloses the object(s).

  RVEC(5) = Line Adjustment - The amount that lines are displaced (towards the viewer) so that the surfaces do not obscure them.

  RVEC(6) = Screen Z Clip value - Value added to screen Z to move object(s) into the front or back clipping plane.

  RVEC(7) = Feature Line Cross - The value of the cross-product between Domain Surface face normals below which the shared face is added to the Domain Surface Edge structure. Default: 0.0 - no Feature Lines, maximum value is 1.0 (show the enitire mesh).

- OPT = 15 - Dynamic surface and arrow state:

  IVEC(1) = Current drawing state - see the Appendix

  IVEC(2) = Number of tufts - the number of tufts on a side

  IVEC(3) = 3D Arrow representation :

  0 - Line segment

  1 - Line segment with colored head

  2 - Open head

  3 - Closed head

IVEC(4) = 3D Arrow Mask (default - 1):

1 - Arrows

2 - Tufts

4 - Vector Clouds

RVEC(1) = Vector scale - the scale factor for tufts and arrows

RVEC(2) = Vector scale for surface functions - the scale factor used with tufts and arrows for special surface functions

RVEC(3) = 3D Arrow head length ratio

RVEC(4) = 3D Arrow head spread ratio

- OPT = 16 - StreamLine/Bubble drawing state:

IVEC(1) = StreamLine representation :

0 - StreamLine

1 - Ribbon

2 - StreamTube

3 - Tubes with twist

IVEC(2) = StreamLine/Bubble rendering :

0 - StreamLine rendering off/Ghost bubbles off

1 - StreamLine/Bubble rendered in foreground color

2 - StreamLine/Bubble rendered using scalar color

- OPT = 17 - Key and Colormap state:

IVEC(1) = Key status :

0 - Scalar

1 - Special Surface Scalar

2 - Cell based Scalar

IVEC(2) = Number of entries in colormap

IVEC(3) = Number of entries in background colormap - between 1 and 4

- OPT = 18 - Get colormap entry in current Key colormap:

  IVEC(1) = index - set on input:

  - -2 - Foreground: used for outlines and pseudo-cursors, usually white
  - -1 - Midground: used for 1D cursor and solid surfaces, usually grey
  - 0 - Background: used for Dial and 1D window background, usually black
  - 1-on - Colormap/background colormap entry. The background colormap indices follow the colormap entries.

  RVEC(1-3) = The color (r,g,b) - values between 0.0 and 1.0

- OPT = 19 - Contouring state:

  IVEC(1) = Dynamic contour status - 0 = off, 1 = on

  IVEC(2) = Number of contour levels

- OPT = 20 - Surface Scalar state:

  IVEC(1) = Binding index

  RVEC(1) = Minimum for display - fsmin shown in the key window

  RVEC(2) = Maximum for display - fsmax shown in the key window

  STRING = Scalar title - maximum of 32 characters used.

- OPT = 21 - Surface Vector state:

  IVEC(1) = Binding index

  RVEC(1) = Vector scale - the scale factor for tufts and arrows

  STRING = Vector title - maximum of 32 characters used.

5.3 V3_SETSTATE

V3_SETSTATE(OPT,IVEC,RVEC,STRING)
Sets the internal VISUAL3 state. This VISUAL3 routine can only be called from V3PEVENTS, V3EVENTS, and V3UPDATE.

OPT:i: I	Option set to specify what internal state data to change
IVEC:i/o: I()	Integer data set based on OPT (length also determined by OPT)
RVEC:i: R()	Real data set based on OPT (length also determined by OPT)
STRING:i: C	Character data set based on OPT

- OPT = -4 - Spawn a streamline/particle at the specified starting location. A streamline is spawned for steady-state cases or when an unsteady application is 'paused'. The starting 3D cell index as well as the 3D coordinates are required. If the cell index is not known, then either use **V3_GETSTATE** with OPT = -4 or **V3_INT**. On return, **IVEC(2)** contains the streamline/particle index, a 0 indicates that the object was not spawned. The following must be filled before the call:

  IVEC(1) = 3D cell index - returns SL/Bubble index. Zero is an indication of an error.

  IVEC(2) = Direction - streamlines only:

  -1 - Upstream
  0 - Both directions
  1 - Downstream

  RVEC(1-3) = 3D coordinates

- OPT = -3 - Set a programmer defined static surface. This allows specifying internal 'surfaces' that can be controlled like any static surface. This option forces a call to the programmer supplied routine **V3STATIC** to fill in the '2D static surface' data. The following must be filled:

  IVEC(1) = KNODE2D

  IVEC(2) = KCELL

  IVEC(3) = KFACE

  IVEC(4) = KEDGE

  IVEC(5) = Drawing Mask - see the Appendix

IVEC(6) = Surface Type must be one of:

2 - Planar cut-like (with clipping) - requires RVEC

4 - Programmer defined surface-like

7 - IsoSurface-like (involves mirroring)

RVEC(1-9) - defines 3 of the 4 corners of the plane. Required for IVEC(6) = 2 only.

STRING = Surface title - maximum of 20 characters used

- OPT = -2 - Force a 2D window update. This tells VISUAL3 to do a complete redraw starting with the static objects. This is required for programmer controlled 2D window drawing.

- OPT = -1 - Force a 3D and 2D window update. This tells VISUAL3 to do a complete redraw including the static objects.

- OPT = 0 - Force a data update. This acts like an unsteady case, in that it calls V3SCAL, V3VECT and V3THRES. V3GRID is also called if IVEC(1) is greater than 1 or the case is 'grid unsteady'. This call has no effect if the application is unsteady and 'pause' is not in effect.

- OPT = 1 - Set the current transformation matrix for the 3D view. 12 words of RVEC are used. If 'cut-plane positioning' is on then the transformation matrix for the plane is adjusted.

- OPT = 2 - Set the current planar cut coordinates. 9 words of RVEC are used for 3 of the 4 corners of the cut plane in 3 space. If the planar cut is on, this plane must be in the family of planes parallel to current cut. To change the orientation, first turn off the planar cut, set a new position, then turn it back on.

- OPT = 3 - Set the current scalar tool values:

IVEC(1) = TOOL :

0 - Turn off programmer controlled 2D window

1 - Initiate programmer controlled 2D window

RVEC(1) = ZPRIME

RVEC(2) = XPC

RVEC(3) = YPC

RVEC(4) = HALFW

RVEC(5) = 2D Rotation Angle in degrees

NOTES:

(1) Use events for initiating all tools except the programmer controlled 2D window.

(2) If you don't wish to change all of the above call V3_GETSTATE(3,IVEC,RVEC,STRING) first!

- OPT = 4 - Set Lighting and Mirroring state:

  IVEC(1) = Mirror status :

  0 - No Mirroring

  1 - Mirror about X = 0.0

  2 - Mirror about Y = 0.0

  3 - Mirror about Z = 0.0

  IVEC(2) = Lighting status - 0 = off, 1 = on

  IVEC(3) = Shading level - a value between 0 (no lighting) and 255

  RVEC(1-3) = Lighting vector

  RVEC(4-6) = Shading background color (r,g,b) - values between 0.0 and 1.0. This remains the shading color even after colormap changes as long as it is different from the background color in the colormap.

- OPT = 5 - Scalar state:

  IVEC(1) = Binding index - allows the changing of the current scalar function

  RVEC(1) = Minimum for display - fmin shown in the key window

  RVEC(2) = Maximum for display - fmax shown in the key window

  STRING = Scalar title - maximum of 32 characters used (blank string indicates no change)

- OPT = 6 - Vector state:

  IVEC(1) = Binding index allows the setting of the current vector field

  RVEC(1) = Vector scale - the scale factor for tufts and arrows

  STRING = Vector title - maximum of 32 characters used (blank string indicates no change)

- OPT = 7 - Threshold state:

  IVEC(1) = Binding index - negative value indicates the appropriate scalar index

  IVEC(2) = Vector Cloud status - 0 = off, 1 = on

  RVEC(1) = Minimum for display - tmin shown in the key window

  RVEC(2) = Maximum for display - tmax shown in the key window

  STRING = Threshold title - maximum of 32 characters used (blank string indicates no change). You cannot change the title of a scalar threshold with this function.

- OPT = 8 - Enter user response:

  IVEC(1) = First mouse button action - 0 = no button press

  IVEC(2) = X pixel location for action

  IVEC(3) = Y pixel location for action

  IVEC(4) = Second mouse button action

  IVEC(5) = X pixel location for action

  IVEC(6) = Y pixel location for action

  STRING = Text answer

  The data in IVEC is required for those operations that snap a line/circle or a 'rubber-band' box.

- OPT = 9 - Cell scalar state:

  IVEC(1) = Binding index - allows the changing of the current cell scalar function

  RVEC(1) = Minimum for display - cmin shown in the key window

  RVEC(2) = Maximum for display - cmax shown in the key window

  STRING = Cell scalar title - maximum of 32 characters used (blank string indicates no change)

- OPT = 10 - Domain/static surface state:

  IVEC(1) = Surface index (-) value indicates that the numbered surface should be deleted without effecting what is the current surface

  IVEC(2) = Current drawing state - Absolute value > 16387 doesn't change the state. See the Appendix.

  STRING = Surface title - maximum of 20 characters used (blank string indicates no change).

- OPT = 11 - Streamline state:

  IVEC(1) = Streamline index - Set the current index.

  RVEC(1) = Ribbon rotation in degrees

  RVEC(2) = Ribbon/Tube size

  RVEC(3) = Tube Maximum size

- OPT = 12 - Bubble state:

  IVEC(1) = Bubble index - Set the current bubble index.

- OPT = 13 - Set probe state:

  IVEC(1) = 1D plot type :

  -1 - set scaling as defined in RVEC. A probe must be on.

  0 - turn off plot

  6 - start Programmer-defined probe

  RVEC(1) = X Minimum for plot - IVEC(1) = -1 only

  RVEC(2) = X Maximum for plot - IVEC(1) = -1 only

  RVEC(3) = Y Minimum for plot - IVEC(1) = -1 only

  RVEC(4) = Y Maximum for plot - IVEC(1) = -1 only

  Note: use the proper 'events' for initiating the other probes.

- OPT = 14 - Set global drawing state:

  IVEC(1) = Edge Outline - 0 (off) or 1 (on)

  IVEC(2) = Transparent Plane - 0 (off) or 1 (on)

  RVEC(1-3) = Focus - Values subrtacted from each node coordinate in 3 space before the transformation matrix is applied.

  RVEC(4) = Object Radius - The radius of the sphere that encloses the object(s).

  RVEC(5) = Line Adjustment - The amount that lines are displaced (towards the viewer) so that the surfaces do not obscure them.

  RVEC(6) = Screen Z Clip value - Value added to screen Z to move object(s) into the front or back clipping plane.

  RVEC(7) = Feature Line Cross - The value of the cross-product between Domain Surface face normals below which the shared face is added to the Domain Surface Edge structure. The minimum value of 0.0 indicates no Feature Lines, the maximum value is 1.0 (show the enitire mesh).

- OPT = 15 - Set dynamic surface and arrow state:

  IVEC(1) = Current drawing state - see the Appendix

  IVEC(2) = Number of tufts - the number of tufts on a side

  IVEC(3) = 3D Arrow representation :

  0 - Line segment

  1 - Line segment with colored head

  2 - Open head

  3 - Closed head

  IVEC(4) = 3D Arrow Mask (additive/orrable):

  1 - Arrows

  2 - Tufts

  4 - Vector Clouds

  RVEC(1) = Vector scale - the scale factor for tufts and arrows

  RVEC(2) = Vector scale for surface functions - the scale factor used with tufts and arrows for special surface functions

  RVEC(3) = 3D Arrow head length ratio

  RVEC(4) = 3D Arrow head spread ratio

- OPT = 16 - Set StreamLine/Bubble drawing state:

  IVEC(1) = StreamLine representation :

  0 - StreamLine

  1 - Ribbon

  2 - StreamTube

  3 - Tubes with twist

  IVEC(2) = StreamLine/Bubble rendering :

  0 - StreamLine rendering off/Ghost bubbles off

  1 - StreamLine/Bubble rendered in foreground color

  2 - StreamLine/Bubble rendered using scalar color

- OPT = 17 - Set Key and Colormap state:

  IVEC(1) = Key status :

  0 - Scalar

  1 - Special Surface Scalar - if any Surface Scalars

  2 - Cell based Scalar - if any Cell based Scalars

  IVEC(2) = Number of entries in colormap

  IVEC(3) = Number of entries in background colormap - between 1 and 4

  This forces a Key Window redraw. When changing the colors, first set the Key state (Scalar, Surface Scalar or Cell based) if not already correct, then set the new colormap entries (calls with OPT = 18) and finally (again) make this call.

- OPT = 18 - Set colormap entry in current Key colormap:

  IVEC(1) = index :

  -2 - Foreground: used for outlines and pseudo-cursors - Key independent

  -1 - Midground: used for 1D cursor and solid surfaces - Key independent

  0 - Background: used for Dial and 1D window background - Key independent

  1-on - Colormap/background colormap entry. The background colormap indices follow the colormap entries.

  RVEC(1-3) = The color (r,g,b) - values between 0.0 and 1.0

- OPT = 19 - Set Contouring state:

  IVEC(1) = Dynamic contour status - 0 = off, 1 = on

  IVEC(2) = Number of contour levels - must be greater than 1

- OPT = 20 - Surface Scalar state:

  IVEC(1) = Binding index - allows the changing of the current scalar function

  RVEC(1) = Minimum for display - fsmin shown in the key window

  RVEC(2) = Maximum for display - fsmax shown in the key window

  STRING = Scalar title - maximum of 32 characters used (blank string indicates no change)

- OPT = 21 - Surface Vector state:

  IVEC(1) = Binding index allows the setting of the current vector field

  RVEC(1) = Vector scale - the scale factor for tufts and arrows

  STRING = Vector title - maximum of 32 characters used (blank string indicates no change)

5.4 V3_GETSTRUC

V3_GETSTRUC(OPT,PTR,LEN)

Returns the internal VISUAL3 structure. This VISUAL3 routine can be called from any programmer-supplied code.

OPT:i: I	Option set to specify what data to get.
PTR:o: I	Integer (in FORTRAN) used as a pointer to the structure. In C this returns the pointer to the block of memory. A 0 (zero) indicates that the memory block is not allocated.
LEN:o: I	Length of structure.

- 3D Structures:

OPT	PTR - Pointer to	length of structure
301	XYZ	NNODE
302	S	NNODE
303	T	NNODE
304	V	NNODE
305	Z	NNODE
306	IBLANK	number of nodes in blocks

OPT	PTR - Pointer to	length of structure
311	CEL1	KCEL1
312	CEL2	KCEL2
313	CEL3	KCEL3
314	CEL4	KCEL4
315	NPTET	KNPTET
316	PTET	KPTET
317	BLOCKS	KNBLCK
318	C	NCELLS

OPT	PTR - Pointer to	length of structure
321	NSURF	KNSURF
322	TSURF	KNSURF
323	SURF	KSURF
324	SCEL	KSURF

OPT	PTR - Pointer to	length of structure
325	NSED	KNSED
326	SED	Number of ployline edge points - NSED(1,KNSED)

OPT	PTR - Pointer to	length of structure
331	CCEL1	KCEL1
332	CCEL2	KCEL2
333	CCEL3	KCEL3
334	CCEL4	KCEL4
336	CPTET	KPTET
337	CBLOCKS	len of husks
338	CCEL	KSURF

- 2D Dynamic Structures:

OPT	PTR - Pointer to	length of structure
201	NODE2D	KNODE2
202	XY	KNODE2
203	DTERP	KNODE2 (not used for domain surface mappings)
204	CELL	KCELL
205	ICELL	KCELL
206	FACE	KFACE
207	EDGE	KEDGE
208	CEDGE	KEDGE

- 2D Static Surfaces - returns the contents of the 'active' static, i.e. non-Domain, surface (the surface highlighted in the Dials database):

OPT	PTR - Pointer to	length of structure
211	NODE2D	KNODE2
213	DTERP	KNODE2
214	CELL	KCELL
215	ICELL	KCELL (only available if there are cell based scalars)
216	FACE	KFACE
217	EDGE	KEDGE

- 1D Structures:

OPT	PTR - Pointer to	length of structure
101	CELLSL	KSTREAM
102	WEIGHTSL	KSTREAM
103	TIMESL	KSTREAM
104	ANGLESL	KSTREAM (not used for surface streamlines)
105	DIVSL	KSTREAM (not used for surface streamlines)

- OPT = 1 - XEvent Structure:

PTR = *XEvent - Useful for building another Graphical User Interface (GUI) on VISUAL3. Allows passing the event information to tool-kits such as Motif. This need only be called once after the call to V3_INIT. The same structure is used for all VISUAL3 event handling.

5.5 V3_INT

V3_INT(KC, XYZ, NNODES, NODES, WEIGHTS, NSUFG)
This routine calculates and returns the cell index containing the target position and the weights applied to the nodes of the cell to interpolate to that position. This VISUAL3 routine can be called from any programmer-supplied code.

KC:i/o: I — On input either 0 indicating that a search is to be one to find the cell or a 3D cell index to start the search. It is always best to start from a known cell that is close to the target. On output:

KC = 0 - Error position not in the volume

KC > 0 - 3D cell index containing the position

KC < 0 - (-)index of surface facet index of exiting face

XYZ:i: R(3) — Target position

NNODES:o: I — Number of nodes that support cell/facet

NODES:o: I(8) — 3D Node indices, filled to NNODES for KC > 0.
For KC < 0; 3D Node indices for the facet and:

NODES(5) = face index in the cell

NODES(6) = 3D cell index

WEIGHTS:o: R(4,8) — only filled for KC > 0.

WEIGHTS(1,n) - weight applied to the value associated with the 3D cell index in NODES to interpolate to position XYZ.

WEIGHTS(2,n) - weight applied to the value associated with the 3D cell index in NODES to compute the X derivative at XYZ

WEIGHTS(3,n) - weight applied to the value associated with the 3D cell index in NODES to compute the Y derivative at XYZ

WEIGHTS(4,n) - weight applied to the value associated with the 3D cell index in NODES to compute the Z derivative at XYZ

NSUFG:o: I — The surface group conatining the facet, only filled for KC < 0, zero otherwise.

5.6 V3STATIC

V3STATIC(NODE2D, DTERP, CELL, FACE, EDGE, ICELL)
This programmer supplied subroutine is called by VISUAL3 after a call to V3_SETSTATE with OPT = -3 has been executed. The purpose of this routine is to fill the '2D Static Structure' data.

NODE2D:o: I(3,KNODE2)	'2D node' definitions. **NODE2D(1,n)** = first 3D node index **NODE2D(2,n)** = second 3D node index **NODE2D(3,n)** = clip mask (for type = 2 only)
DTERP:o: R(KNODE2)	is a number between 0.0 and 1.0 that defines the distance from the first 3D node in NODE2D. A 2D node that coincides with the 3D node should have NODE2D(1,n) equal to NODE2D(2,n) and DTERP(n) any value between 0.0 and 1.0.
CELL:o: I(3,KCELL)	'2D cell' definitions. Each entry contains a '2D node' index defining a triangle.
FACE:o: I(2,KFACE)	'2D mesh' definitions. Each entry contains a '2D node' index that defines a disjoint line to be plotted when the grid is requested.
EDGE:o: I(3,KEDGE)	the surface outline - '2D edge' definitions. **EDGE(1,n)** = first '2D node' index **EDGE(2,n)** = second '2D node' index **EDGE(3,n)** = used internally
ICELL:o: I(KCELL)	3D Cell index for the triangle. Note: should only be filled if there are cell based scalars.

The following routines are used for drawing programmer defined objects in the 3D window:

5.7 V3DRAW3D

V3DRAW3D(ISTAT)

This programmer supplied subroutine is called by VISUAL3 at different times during the rendering phase. It is the responsibility of this routine to specify the object(s) to be plotted based on the rendering status. The object(s) are defined by calls to V3_OBJECT3D. Calls to V3_GETSTATE and V3_GETSTRUC are valid but invokations of V3_SETSTATE should be avoided.

ISTAT:i: I	Drawing status: **0** - rendering for static object(s) **1** - rendering for dynamic object(s) **2** - rendering for translucent object(s)

The options listed above reflect the manner that VISUAL3 draws the 3D scene. If the case is steady-state (or unsteady and 'pause' is in effect) and the viewing transformation matrix is NOT changing, static objects only get rendered once (until the transformation matrix is changed again). 'Snap-Shot Rendering' is used to improve drawing performance so the result of the rendering of the static objects is saved in a secondary 'pixmap' and ZBuffer. These items are used to start the scene generation for subsequent animation frames.

Dynamic objects are those that will change or move from frame to frame even if the underlying (static) object(s) are not moving. An example of this is any dynamic surface (cut plane or iso-surface) or StreamLine animation using 'blobs' to display the integration time.

Translucent objects, even if static, must be rendered last in order to get the scene to look correct. The ZBuffer must be updated with the translucent surfaces last.

If the visualization application is unsteady (and not paused), the static objects as well as the dynamic and translucent objects are rendered each time step.

5.8 V3_OBJECT3D

V3_OBJECT3D(ITYPE,ICOLOR,XYZP,RADII,COL,NC,NP)

This routine must only be used from within V3DRAW3D. It defines additional plotting for the 3D window. Multiple objects may be drawn by multiple calls to this routine from within V3DRAW3D.

ITYPE:i: I

Type of plotting primitive:

- **0** - disjoint quadrilaterals - NP must be 4 times the number of quads
- **1** - disjoint triangles - NP must be 3 times the number of tris
- **2** - polytriangle strip - NP must be the length of the strip
- **3** - disjoint line segments - NP must be 2 times the number of segments
- **4** - polyline - NP must be the length of the line
- **5** - spheroids - NP must be the number of shpere-like objects to plot

ICOLOR:i: I

The method used for coloring the object:

- **-1** - draw the object white
- **0** - draw the object in grey
- **1** - color the object from the data in COL
- **2** - the data is to be colored using the scalar field data for the node number in NC
- **3** - the data is to be colored using the actual scalar field data at the coordinates of the points specified. Each entry of NC must contain the cell number for the data point in 3 space. This option slows the rendering because each data item (and there are NP of them) needs the cell weightings computed in order to apply the coloring. Note: requires connectivity.

XYZP:i: R(3,NP)

The (x, y, z)-coordinates of the points that support the primitive.

RADII:i: R(NP)	The spheroid radius (in user coordinates). For ITYPE = 5 only. Points are plotted (and are MUCH faster) if RADII(1) is 0.0
COL:i: R(NP)	Color scaling for the data points (used only for ICOLOR = 1). The values must be in the range 0.0 to 1.0 and the scalar field color map is applied to that range to compute the color for the point.
NC:i: I(NP)	Color mapping for the data points with ICOLOR = 2 or 3. For ICOLOR = 2, the values must be in the range 1 to NNODE and for ICOLOR = 3, the values must be in the range 1 to NCELLS.
NP:i: I	Object length (the number of points)

The following routines are used for drawing programmer defined objects in the 2D window:

5.9 V3DRAW2D

V3DRAW2D(ISTAT)

This programmer supplied subroutine is called by VISUAL3 at different times during the 2D window rendering phase. It is the responsibility of this routine to specify the object(s) to be plotted based on the rendering status. The object(s) are defined by calls to V3_OBJECT2D. Calls to V3_GETSTATE and V3_GETSTRUC are valid but invokations of V3_SETSTATE should be avoided.

ISTAT:i: I	Drawing status **0** - rendering for static object(s) **1** - rendering for dynamic object(s)

Like rendering for the 3D window, the 2D window scene generation has multiple phases. Because translucency is not supported (nor does it make much sense) in the 2D window and there is no ZBuffer, the translucent phase is not implemented.

The only 2D dynamic object that VISUAL3 currently uses is the mapped cross-hair cursor.

For a program controlled 2D window, the programmer must perform a forced update (V3_SETSTATE OPT = -2/-1) for a static draw or should render everything during the dynamic phase.

5.10 V3_OBJECT2D

V3_OBJECT2D(ITYPE,ICOLOR,XYW,COL,NP)

This routine must only be used from within V3DRAW2D. It defines additional plotting for the 2D window. Multiple objects may be drawn by multiple calls to this routine from within V3DRAW2D.

ITYPE:i: I	Type of plotting primitive: **0** - disjoint quadrilaterals - NP must be 4 times the number of quads **1** - disjoint triangles - NP must be 3 times the number of tris

2 - polytriangle strip - NP must be the length of the strip

3 - disjoint line segments - NP must be 2 times the number of segments

4 - polyline - NP must be the length of the line

5 - points - NP must be the number of dots

ICOLOR:i: I — The method used for coloring the object:

-1 - draw the object white

0 - draw the object in grey

1 - color the object from the data in COL

XYW:i: R(2,NP) — X and Y window values for the points that make up the drawing primitive. The plotting in the 2D window for X ranges from -1.0 (left side) to 1.0 (the right side of the window). The Y range is from -1.0 (bottom) to 1.0 (the top of the window).

COL:i: R(NP) — Color scaling for the data points. Used only for ICOLOR = 1. The values must be in the range 0.0 to 1.0 and the scalar field color map is applied to that range to compute the color for the point.

NP:i: I — Object length in points

The following routines are used for Programmer-defined probes:

5.11 V3PROBE

V3PROBE(FLAG,XAXIS,YAXIS)

This programmer supplied subroutine is called by VISUAL3 at the end of the rendering phase if the programmer-defined probe is activated. It is the responsibility of this routine to specify the line(s) to be plotted and return the axis annotation. The line or lines are defined by calls to V3_LINE. Calls to V3_GETSTATE and V3_GETSTRUC are valid but invokations of V3_SETSTATE should be avoided.

FLAG:i: L	logical flag
	.TRUE. first call after probe has been activated
	.FALSE. subsequent calls
XAXIS:o: C*32	X-Axis label for the plot
YAXIS:o: C*32	Y-Axis label for the plot

Note: Currently the only way to initiate the programmer-defined probe is by a call to V3_SETSTATE with OPT = 13 from V3PEVENTS or V3EVENTS.

5.12 V3_LINE

V3_LINE(X,Y,NP)

This routine must only be used from within V3PROBE. It defines a line to be plotted in the 1D window. Multiple lines may be drawn by multiple calls to this routine from within V3PROBE. Auto-scaling is performed on the first line for the first call to V3PROBE after the probe has been started. Afterwards user or programmed events can control the scaling.

X:i: R(NP)	X values for the line
Y:i: R(NP)	Y values for the line
NP:i: I	Line length

5.13 V3INIT

V3INIT(NPIX1,NSX1,NSY1, NPIX2,NSX2,NSY2, NPIX3,NSX3,NSY3)
This routine should not be confused with V3_INIT. It is called from within V3_INIT before the windows are open. It allows the customized selection of window sizes and placement as well as allowing any advanced programming that should only be performed at initialization (such as memory allocation).

NPIX1:i/o: I	The size of the 1D window. Enters with VISUAL3s default size.
NSX1:i/o: I	The suggested X pixel location (in the root window) for the start position of the 1D window. Enters with VISUAL3s default location.
NSY1:i/o: I	The suggested Y pixel location (in the root window) for the start position of the 1D window. Enters with VISUAL3s default location.
NPIX2:i/o: I	The size of the 2D window. Enters with VISUAL3s default size.
NSX2:i/o: I	The suggested X pixel location (in the root window) for the start position of the 2D window. Enters with VISUAL3s default location.
NSY2:i/o: I	The suggested Y pixel location (in the root window) for the start position of the 2D window. Enters with VISUAL3s default location.
NPIX3:i/o: I	The size of the 3D window. Enters with VISUAL3s default size.
NSX3:i/o: I	The suggested X pixel location (in the root window) for the start position of the 3D window. Enters with VISUAL3s default location.
NSY3:i/o: I	The suggested Y pixel location (in the root window) for the start position of the 3D window. Enters with VISUAL3s default location.

6 Portability

6.1 FORTRAN Programming

For the most part FORTRAN source that runs on one workstation implementation of VISUAL3 will work on others. See the Appendix for 64 bit machines.

NOTES:
(1) Some FORTRANs have their own MALLOC and FREE calls. For portability it is advised to use VISUAL3's routines (documented in this manual). This is because the calling sequences differ from machine to machine.
(2) Getting command line arguments is different on HP machines (IGETARG) and the other workstations (GETARG).

6.2 C Programming

Unfortunately VISUAL3 is not source code compatible for C across all workstations. This has to do with supporting FORTRAN.

The C programmer that wishes to have their VISUAL3 application running on many platforms should be aware that:

- main program

  On HP, IBM and SUN workstations, normal C conventions apply. For DEC and SGI ports the name of the main program must be 'MAIN__' and on all others the name is 'MAIN_'.

- getting command line arguments

  On HP, IBM and SUN the normal C mechanism to get command line arguments may be used. On the other vendors machines, the FORTRAN method must be used.

- routine names

  For HP and IBM ports, all VISUAL3 entry points are the FORTRAN names in lower-case. On all other platforms, the external entries are lower-case with an underscore ('_') appended to the end.

See the file 'wsdepend.h' in the demo subdirectory of the distribution for a method to avoid these problems. This include is used with 'cprism'.

A Drawing State Mask

The following are additive (or-able) so that the proper surface attibutes can be specified:

1 - Render - Surface rendering on

2 - Grid - Mesh display on

4 - Grey - Surface colored with grey (Domain and static surfaces only). If this is the only bit set, the surface outline will be plotted.

8 - Threshold - Surface is thresholded according to the threshold function and limits

16 - Contour - Contour lines are plotted on the surface

32 - 3D Render - Plot surface also in the 3D window (Dynamic surfaces only)

64 - Arrows - Arrow drawing on

128 - Tufts - Grid of tufts on (Dynamic surfaces only)

8192 - Cell Rendering - Render with cell based scalar (Not with threaholding on)

Note: For domain and static surfaces, take the drawing state specified and negate it to produce a surface with the translucent attribute.

B 64 Bit Machines

This programming interface assumes that pointers and integers are of the same length. This is not the case for 64 bit workstations (DECs ALPHA and SGIs running is 64 bit mode). For these machines, integers occupy 4 bytes but pointers are 8 bytes. Therefore, any pointer must be stored as an INTEGER*8 (or cast as a *long* in C, if required). This requires changes to the following routine interfaces:

B.1 V3PEVENTS

V3PEVENTS(IWIN,TYPE,XE,YE,STATE)

IWIN:i/o: I*8	The X window ID of the event or −1
TYPE:i/o: I	The event type.
XE:i/o: I	The X pixel location in the window of the event
YE:i/o: I	The Y pixel location in the window of the event
STATE:i/o: I	The event state.

B.2 V3_GETSTATE

V3_GETSTATE(OPT,IVEC,RVEC,STRING)

OPT:i: I	Option set to specify what data to get
IVEC:i/o: I*8()	Integer data returned based on OPT
RVEC:o: R()	Real data returned based on OPT
STRING:o: C	Character data returned based on OPT

B.3 V3_SETSTATE

V3_SETSTATE(OPT,IVEC,RVEC,STRING)

OPT:i: I	Option set to specify what internal state data to change
IVEC:i: I*8()	Integer data set based on OPT
RVEC:i: R()	Real data set based on OPT
STRING:i: C	Character data set based on OPT

B.4 V3_GETSTRUC

V3_GETSTRUC(OPT,PTR,LEN)

OPT:i: I	Option set to specify what data to get
PTR:o: I*8	INTEGER*8 (in FORTRAN) used as a pointer to the structure.
LEN:o: I	Length of structure.