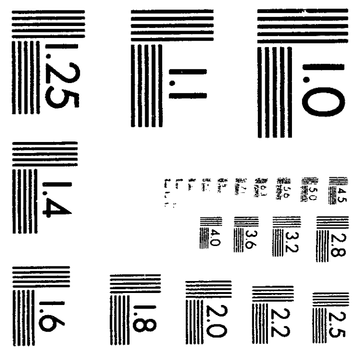
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5

1 of 1

UCRL-MA-115696

GRIZ

Finite Element Analysis Results Visualization for Unstructured Grids

User Manual

Donald Dovey
Thomas E. Spelce

Methods Development Group
Mechanical Engineering

October, 1993

Lawrence Livermore National Laboratory

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

GRIZ

Finite Element Analysis Results Visualization for Unstructured Grids

User Manual

Donald J. Dovey
Thomas E. Spelce

Methods Development Group
Mechanical Engineering

October, 1993

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

GRIZ is an interactive application for visualizing finite element analysis results on three-dimensional unstructured grids. It was developed by the Methods Development Group at Lawrence Livermore National Laboratory. The authors of the software are Don Dovey and Tom Spelce. Mark Christon wrote the video driver interface and contributed to the initial design of the program. The isosurface generation code was adapted from software written by Brian Cabral. Michael Loomis, Mark Christon and the members of the Applied Mechanics Group at LLNL provided many suggestions for improvements to the software.

1.0 INTRODUCTION

GRIZ supports interactive visualization of finite element analysis results on unstructured grids. GRIZ is a general-purpose post-processing application which is designed to work with a variety of analysis codes. Currently, GRIZ is capable of calculating and displaying derived variables for the DYNA3D, NIKE3D and TOPAZ3D analysis codes. GRIZ reads in data files in the "MDG plotfile" format.

GRIZ provides support for modern 3D visualization techniques such as isosurface display, cutting planes and display of vector data. GRIZ also incorporates the ability toanimate data over time and to store animation frames to a video disk. GRIZ is designed to utilize the capabilities of modern graphics workstations which provide hardware support for 3D graphics, thereby giving the user as much interactive performance as possible. This should make it easier for analysts to explore and interrogate their analysis results.

GRIZ uses Silicon Graphic Inc.'s GL (Graphics Library) for rendering and the Motif widget toolkit for its user interface. In order to compile and run GRIZ, both of these libraries are required. Although GRIZ is used most heavily on SGI workstations, it has been compiled and run on Sun workstations using commercial GL emulation software.

It is hoped that GRIZ will eventually replace the post-processor TAURUS for meeting analysis needs. Please keep in mind, however, that the software is still under development. GRIZ provides many capabilities which aren't present in TAURUS, but some TAURUS functionality may be missing from GRIZ. Users are encouraged to be patient and to keep a "wish list" of desired capabilities that can be given to the developers. Bug reports and suggestions for improvements to this documentation are also welcome.

GRIZ is part of a set of public domain codes developed in the Methods Development Group at LLNL. Other codes include the analysis codes DYNA3D, DYNA2D, NIKE3D, NIKE2D, TOPAZ3D, TOPAZ2D, PALM2D, MONT3D and the mesh-generation and post-processing codes MAZE, ORION, INGRID and TAURUS.

The manual is organized into chapters on operation, commands, a command summary, future development and code design.

2.0 OPERATION

This chapter gives an overview of the GRIZ program and its operation.

To run GRIZ, type the command

griz -i *filename*

where the input file name is the root name of the plotfiles. The main GRIZ window will appear on the screen.

The main GRIZ window includes a menu bar across the top, a set of view control widgets underneath, and a display area where the mesh is displayed. The controls for GRIZ fall into three principal groups. First, the menu system provides access to the most frequently-used commands, which are a subset of the full GRIZ command set. Second, the view control widgets allow the user to interactively control the mesh view. Also, clicking the mouse buttons in the mesh display area has an effect which will be explained later. Third, a command line interface is available for typing in the full set of GRIZ commands.

Menu buttons are activated by pulling down a menu with the mouse and releasing the mouse button when the desired menu selection is highlighted. The view control widgets are controlled by holding down the left mouse button while the mouse is pointing at the widget. These widgets allow the user to rotate, translate and scale the viewed mesh.

Clicking the mouse in the mesh display area while the mouse is held in one place is used for selecting nodes as well as beam, shell and volume elements. The left mouse button is used to select nodes, the middle button for shell or beam elements, and the right button for volume elements. Moving the mouse with a button held down causes the mesh view to be interactively modified. Moving the mouse in the X and Y direction with the left mouse button held down causes rotation about the Y and X axes, respectively. Moving the mouse in the X direction with the middle mouse button held down implements Z rotation, and Y movement with the middle button held down implements scaling. Mouse movement with the right mouse button causes translation in the X and Y directions. The mesh display switches to a pseudo-wireframe mode while the view is being modified with the mouse, and returns to the normal display when the mouse button is released.

To permit command line input to the program, the user should pull down the "Control" menu on the menu bar and pick the "Command" menu button. A command window will pop up on the screen. Clicking the mouse in the text-entry area of the command window will prepare the window to accept typed-in commands.

At startup, GRIZ looks in the current directory for a history file named "grizinit". A history file is simply a text file with one GRIZ command per line. If a file named "grizinit" exists, the commands in it are executed automatically at startup. If the file "grizinit" does not exist in the current directory, GRIZ looks for startup commands in a file specified by the environment variable "GRIZINIT". For example, one might use a line like the following in their .cshrc file.

setenv GRIZINIT /usr/people/smith/grizstuff/grizinit

If their is no "grizinit" file in the current directory and the "GRIZINIT" environment variable is not set, then GRIZ starts up without performing any user-specified startup commands.

To quit GRIZ, pick the "Quit" menu button at the bottom of the "Control" menu, or type in "quit" at the command line.

3.0 COMMANDS

GRIZ commands are entered in the command window (see Section 2.0 for instructions for opening this window). One command is entered on each line. A command may have zero or more arguments. When a string that includes spaces is to be provided as a single argument, the string must be enclosed in quotes. Comment lines can be included in command sequences and are demarcated with the '#' character. Anything from a '#' to the end of the line is ignored. Comments are primarily of use in history files, which are explained later. Two special types of commands appear throughout this section. First, the "**on**" and "**off**" commands are used to set and unset Boolean-valued flags. For example, "**on box**" turns on display of the mesh bounding box, and "**off box**" turns off this display option. Similarly, the "**switch**" command is used to set multiple-valued flags. For example, "**switch hidden**" switches to a hidden-line display of the mesh. The "**switch**" command can be abbreviated "**sw**". The command names are designed to be reasonably easy to remember and are short enough to be typed quickly. A command alias capability is provided for users who wish to specify an alternative command name for any command.

The commands in this chapter are organized into the following categories: View, Rendering, Picking, Time, Results, Colormap, Visualization, Output, Miscellaneous, and Video.

3.1 View Commands

View control commands allow the user to rotate, translate and scale the mesh view.

rx ϕ	Rotate about the X axis by an angle ϕ. The X axis is parallel to the horizontal axis of the screen.
ry ϕ	Rotate about the Y axis by an angle ϕ. The Y axis is parallel to the vertical axis of the screen.
rz ϕ	Rotate about the Z axis by an angle ϕ. The positive Z axis comes out of the screen toward the viewer and is perpendicular to the screen.
tx v	Translate along the X axis by a value v.
ty v	Translate along the Y axis by a value v.
tz v	Translate along the Z axis by a value v.
scale v	Set the mesh scale value to v. Unlike the rotate and translate commands, which are incremental, the scale command sets an absolute scale value.
scalax v_x v_y v_z	Set mesh scale values for each axis individually.

rview Reset the view to the default view -- no rotation, no translation and a scale of 1. When the program starts up, it uses an initial rotation about the X and Y axis in the view parameters. As a result, the user may wish to reorient the view at start-up using this command. The view can be reset at any other time as well.

switch persp Switch to a perspective view. The default eye position is at (0, 0, 20) in the view coordinate system. The viewport extends from -1 to 1 along the X and Y axes at the point 1 in Z.

switch ortho Switch to an orthographic view. Parallel lines on the mesh will stay parallel on the screen in this view.

rnf Reset the near and far planes. In a perspective view, the near and far planes clip away parts of the scene that are in front of the near plane or behind the far plane. The near and far planes should be set just outside the mesh limits in order to obtain a good image (this is a result of the Z-buffering algorithm that the hardware uses). This command automatically resets the near and far planes in order to obtain a good image of the mesh. It may be necessary to execute this command after translating the mesh along the Z axis or scaling the mesh. It should be used when part of the mesh starts to be "eaten away" in the view.

near *v* Set the near plane position to value *v*. The current position of the near plane can be obtained with the **info** command. Near plane positioning is tricky and this command should be avoided and **rnf** used instead when possible.

far *v* Set the far plane position to value *v*. The current position of the far plane can be obtained with the **info** command. Far plane positioning is tricky and this command should be avoided and **rnf** used instead when possible.

The next four commands can be used to generate exploded views of objects.

tmx *n v* Translate material number *n* an absolute value *v* along the X axis from its original position.

tmy *n v* Translate material number *n* an absolute value *v* along the Y axis from its original position.

tmz *n v* Translate material number *n* an absolute value *v* along the Z axis from its original position.

clrtm Clear all material translations.

The rest of the commands in this section will be of interest primarily to animators.

camang ϕ Set the camera angle in the Y direction for a perspective view.

lookfr $p_x\, p_y\, p_z$ Set the location of the look from (eye) point. The default is (0, 0, 20).

lookat $p_x\, p_y\, p_z$ Set the location of the look at point. The default is (0, 0, 0).

lookup v_x v_y v_z — Set the look up vector. This vector controls the up (Y) direction of the current view. The up vector must not be parallel to the look direction, but can point in any other direction. The default is (0, 1, 0).

tfx *v* — Translate the look from point by a value *v* in the X direction.

tfy *v* — Translate the look from point by a value *v* in the Y direction.

tfz *v* — Translate the look from point by a value *v* in the Z direction.

tax *v* — Translate the look at point by a value *v* in the X direction.

tay *v* — Translate the look at point by a value *v* in the Y direction.

taz *v* — Translate the look at point by a value *v* in the Z direction.

trfr — Connect the look from point to the translation widgets. After executing this command, the translation widgets affect the look from point instead of the mesh view. The **clrtr** command can be used to restore the default behavior.

trat — Connect the look at point to the translation widgets. After executing this command, the translation widgets affect the look at point instead of the mesh view. The **clrtr** command can be used to restore the default behavior.

clrtr — Reset the translation widgets so that they control the mesh view.

3.2 Rendering Commands

The rendering controls affect the way data is displayed in the view window and the colors that are used in the display. Remember that "**on**" and "**off**" are separate commands; they are separated by a slash and shown on the same line to save space.

on/off coord — Turn on (or off) display of the mesh coordinate system.

on/off time — Turn on (or off) display of the current time at the bottom of the mesh view window.

on/off title — Turn on (or off) display of the mesh data title (which comes from the plotfile or from the **title** command) at the bottom of the mesh view window.

on/off cmap — Turn on (or off) display of the colormap.

on/off all — Turn on (or off) **coord, time, title** and **cmap** simultaneously.

on/off box — Turn on (or off) display of the bounding box of the mesh.

on/off bdrop — Turn on (or off) display of a tiled backdrop pattern behind the mesh. This option will probably not be useful for most users, but it looks flashy.

on/off edges Turn on (or off) display of the mesh edges. These aren't the true edges of the mesh, but instead are edges obtained by a simple technique. The edges are drawn in the foreground color. The mesh edges are not automatically recomputed when changing states (affects data with element deletions only). To force the mesh edges to be recomputed, turn edges off then on again.

on/off ndnum Turn on (or off) display of mesh node numbers. All numbers are drawn, so you may need to scale up quite a bit for this command to be useful. An alternative approach is to use mouse picking to determine the desired node numbers.

on/off elnum Turn on (or off) display of mesh element numbers. All numbers are drawn, so you may need to scale up quite a bit for this command to be useful. An alternative approach is to use mouse picking to determine the desired element numbers.

on/off lights Turn on (or off) display of the light positions.

The next three commands set the rendering style.

switch hidden Set the rendering style to filled polygons with their edges drawn (hidden-line drawing). This is the default.

switch solid Set the rendering style to filled polygons with their edges not drawn.

switch wire Set the rendering style to a wire-frame drawing.

The next two commands set the polygon normals for rendering.

switch smooth Smooth-shade the polygons using averaged normals at the vertices. This is the default.

switch flat Flat-shade the polygons. This may be faster than smooth-shading, but it won't look as good.

setcol fg *r g b* Set the foreground color. This is the color in which mesh lines and text are drawn. The color components are integers in the range (0-255).

setcol bg *r g b* Set the background color. The color components are integers in the range (0-255).

setcol con *r g b* Set the contour color. This is the color in which contour lines are drawn. The color components are integers in the range (0-255).

setcol hilite *r g b* Set the hilite color. This is the color used to hilite picked nodes and elements. The color components are integers in the range (0-255).

setcol select *r g b* Set the select color. This is the color used to for selected nodes and elements. The color components are integers in the range (0-255).

setcol vec *r g b* Set the vector color. This is the color in which vectors are drawn. The color components are integers in the range (0-255).

mat *n* <**amb** *r g b*> <**diff** *r g b*> <**spec** *r g b*> <**shine** *v*> <**emis** *r g b*> <**alpha** *v*>

Sets the material display properties of material number *n*. Each of the arguments in braces is optional. The arguments and their permitted ranges are outlined below.

amb: Ambient light color (each 0.0-1.0).
diff: Diffuse light color (each 0.0-1.0).
spec: Specular light color (each 0.0-1.0).
shine: Shininess (0-128, a value of 0 disables specularity). Higher values give smaller, more tightly focused specular highlights.
emis: Emissive light color, object gives off light (each 0.0-1.0).
alpha: Opacity (0.0-1.0 where 0.0 = transparent, 1.0 = opaque).

Normally, only the first two or possibly first four arguments would be used. Only those properties that are specified in the command are modified; the other material properties are left as they were previously. For practical purposes, the diffuse component is what sets the material color. The diffuse and ambient components should be set to the same color for intuitive results. To make material 3 bright green, for example, one could use the command "**mat 3 amb 0 1 0 diff 0 1 0**".

dellit

Deletes all lights. You will need to add some lights after using this command, or you won't be able to see much.

light *n* **pos** $p_x\ p_y\ p_z$ <**color** *r g b*> <**spotdir** $v_x\ v_y\ v_z$> <**spot** *exp* ϕ>

Defines a new light or modifies the definition of a previously-defined light. The light number *n* and position are required; the arguments in braces are optional. By default, lights are treated as point light sources. The light color is used to define colored lights. The color components are values in the range (0.0-1.0). The last two arguments are used to define spot lights. Spot lights will tend to slow the rendering performance. The **spotdir** is the direction in which the spotlight is aimed. The **spot** argument *exp* is the falloff exponent (0-128) where 128 gives the sharpest possible falloff and 0 gives a constant intensity across the cone of light. The **spot** argument ϕ is the spread angle (0-90 degrees) which defines the radius of the cone. Setting the exponent to 0 and the spread angle to 180 will turn off the spotlight effect.

tlx *n v*

Translate light number *n* in the X direction by a value *v*.

tly *n v*

Translate light number *n* in the Y direction by a value *v*.

tlz *n v*

Translate light number *n* in the Z direction by a value *v*.

trlit *n*

Connect the light number *n* to the translation widgets. After executing this command, the translation widgets affect the specified light instead of the mesh view. The **clrtr** command can be used to restore the default behavior.

clrtr

Reset the translation widgets so that they control the mesh view.

hilite {**node, beam, shell** or **brick**} *n*

Hilites the specified element. For example, "**hilite shell 538**" hilites shell number 538 by drawing it in a different color and putting a label near it. Only one element or node is hilited at any time. Hiliting can be performed interactively by using the mouse to pick individual nodes, beams, shells or bricks (See **Picking Commands.**) If a result variable is displayed on the element or node, the value of the variable at the current state is shown in parenthesis.

clrhil Remove the hilite.

select {**node, beam, shell** or **brick**} *n1* <*n2*> <*n3*> ...

Selects the specified element or list of elements. The specified node or element numbers are added to those currently selected. The selected nodes and elements are used in time-history plots (see **timhis**). Selecting can be performed interactively by using the mouse to pick individual nodes, beams, shells or bricks (See **Picking Commands.**). If a result variable is displayed on an element or node, the value of the variable at the current state is shown in parenthesis.

clrsel Delete the lists of selected nodes and elements.

on/off cent Turn on or off centering of the view about the currently hilighted node or element.

invis *n1* <*n2*> ... Make the specified materials invisible. A list of material numbers is the argument. The command "**invis all**" will make all materials invisible.

vis *n1* <*n2*> ... Make the specified materials visible. A list of material numbers is the argument. The command "**vis all**" will make all materials visible.

sym $p_x\ p_y\ p_z\ n_x\ n_y\ n_z$ Add a reflection plane to the list of reflection planes. The plane is specified by a point on the plane and the normal vector to the plane.

clrsym Delete the list of reflection planes.

on/off sym Turn on (or off) reflection of the mesh geometry using the defined reflection planes.

switch symcu Reflection planes are cumulative. The initial geometry is reflected across the first reflection plane. Then the initial and reflected geometry are reflected across the second reflection plane, etc. This is the default reflection mode.

switch symor Reflection planes reflect only the original geometry, and are not cumulative.

inslp *filename* Read in polygon data in SLP format from a file. This command provides a way to display surfaces which aren't part of the mesh. Display of the surfaces is turned on and off with **on/off refsrf**.

inref *filename* — Read in a reference surface file. This file contains a list of faces of hexahedral elements. The idea is that the user wishes to treat these faces as a separate surface which can be displayed independently of the volume data. Display of the reference surfaces is turned on and off with **on/off refsrf**. The file consists of the number of faces followed by the node numbers for each face. The format is

numsrf
n1 n2 n3 n4
n1 n2 n3 n4
etc...

on/off refsrf — Turn on and off display of the reference surfaces which are created by **inslp** or **inref**.

3.3 Picking Commands

As described previously, the mouse can be used to interactively pick nodes or elements. The left mouse button is used for picking nodes, the middle mouse button is used for picking shell or beam elements (see below), and the right mouse button is used for picking volume elements. The commands in this section control the picking operation.

switch pichil — Hiliting mode. A mouse pick hilites the picked node, beam, shell or brick. (See **hilite**.) This is the default.

switch picsel — Selecting mode. A mouse pick selects the picked node, beam, shell or brick. (See **select**.)

switch picsh — The middle mouse button will be used to pick shell elements. This is the default.

switch picbm — The middle mouse button will be used to pick beam elements.

3.4 Time Commands

Time controls allow the user to move through the time states of the data.

state *n* — Move to state number *n*. The total number of states in the data can be obtained with the **info** command.

n — Move to the next state.

p — Move to the previous state.

f — Move to the first state.

l — Move to the last state.

time *t* — Move to time *t*. If the specified time does not coincide with any of the states, the program interpolates the nodal positions for the specified time. If a result is displayed, the result will be interpolated as well. However, after moving to the specified time, changing the displayed result will not work -- the new result won't be interpolated. Issuing the **time** command again will interpolate the new result.

anim — Animate the data by displaying it over time. This form of the **anim** command just displays each state in the data in sequence.

anim *n* — Animate the data by displaying it over time. This form of the **anim** command displays *n* frames between the start time and end time of the data. Interpolation is performed where necessary.

anim $n\ t_s\ t_e$ — Animate the data by displaying it over time. This form of the **anim** command displays *n* frames between the specified start time (t_s) and end time (t_e). Interpolation is performed where necessary.

stopan — Stop an animation that is in progress. The animation command is implemented in such a way that it is possible to use the view controls and type in commands while an animation is running. This command will stop a running animation at the current frame.

animc — Continue an animation from the current state, instead of starting at the first state.

maxst *n* — Set the last state of the data to state *n*. States after state *n* will be ignored. This command is useful for ignoring bad states at the end of the data.

lts — List times for states. Prints out a list of states and their times, as well as the timestep from the previous state.

3.5 Results Commands

The results display commands allow the user to select the result variable to be displayed and to control various parameters associated with the results display. Results are color-mapped with regions of the highest result value drawn in the color at the top of the colormap, regions of the lowest result value drawn in the color at the bottom of the colormap, and so on.

It is important that the user keep in mind how the minimum and maximum result values of the colormap are set. A switch is provided so that either the state min/max or the global min/max can be used. The default is to use the global min/max. Because it is too time-consuming to calculate the min/max for all states at once, the global min/max is obtained incrementally. As each state is displayed, GRIZ calculates the min/max of the currently displayed result for that state, and uses those values to update the global min/max. The global min/max for a specific result is not correct until all states have been visited with that particular result displayed. One can get the global min/

max for a result without stepping through all the states or running **anim** by selecting the desired result and then typing the **globmm** command. Once the global min/max for a result variable has been obtained in either way, it will be "remembered" whenever that result is shown.

show *result* Display a result variable. Valid values for *result* are listed in Table 1.

Table 1: Result Variables

Command	Result
mat	No Result. No result is displayed. The mesh materials are drawn in different colors. This is the default.
Shared Results (Volume and Shell)	
sx	X Stress
sy	Y Stress
sz	Z Stress
sxy	XY Stress
syz	YZ Stress
szx	ZX Stress
seff	Effective Stress
ex	X Strain
ey	Y Strain
ez	Z Strain
exy	XY Strain
eyz	YZ Strain
ezx	ZX Strain
eeff	Effective Strain
press	Pressure
Volume Results	
pdev1	Principal Deviatoric Stress 1
pdev2	Principal Deviatoric Stress 2

Table 1: Result Variables (Continued)

Command	Result
pdev3	Principal Deviatoric Stress 3
maxshr	Maximum Shear Stress
prin1	Principal Stress 1
prin2	Principal Stress 2
prin3	Principal Stress 3
pdstrn1	Principal Deviatoric Strain 1
pdstrn2	Principal Deviatoric Strain 2
pdstrn3	Principal Deviatoric Strain 3
pshrstr	Maximum Shear Strain
pstrn1	Principal Strain 1
pstrn2	Principal Strain 2
pstrn3	Principal Strain 3
relvol	Relative Volume
Shell Results	
res1	M_{xx} Bending Resultant
res2	M_{yy} Bending Resultant
res3	M_{zz} Bending Resultant
res4	Q_{xx} Shear Resultant
res5	Q_{yy} Shear Resultant
res6	N_{xx} Normal Resultant
res7	N_{yy} Normal Resultant
res8	N_{zz} Normal Resultant
thick	Shell Thickness
inteng	Internal Energy
surf1	Surface Stress 1
surf2	Surface Stress 2
surf3	Surface Stress 3

Table 1: Result Variables (Continued)

Command	Result
surf4	Surface Stress 4
surf5	Surface Stress 5
surf6	Surface Stress 6
eff1	Effective Upper Surface Stress
eff2	Effective Lower Surface Stress
effmax	Maximum Effective Surface Stress
Beam Results	
axfor	Axial Force
sshear	S Shear Resultant
tshear	T Shear Resultant
smom	S Moment
tmom	T Moment
tor	Torsional Resultant
Nodal Results	
dispx	X Displacement
dispy	Y Displacement
dispz	Z Displacement
dispmag	Displacement Magnitude
velx	X Velocity
vely	Y Velocity
velz	Z Velocity
velmag	Velocity Magnitude
accx	X Acceleration
accy	Y Acceleration
accz	Z Acceleration
accmag	Acceleration Magnitude
temp	Temperature

globmm Find the global result min/max for the currently displayed result.

disable *n1 <n2> ...* Turn off the results display on the specified materials. A list of material numbers is the argument. This command can be used to look at a result on selected materials only. The command "**disable all**" disables all materials.

enable *n1 <n2> ...* Turn on the results display on the specified materials. A list of material numbers is the argument. The command "**enable all**" enables all materials.

These three commands set the strain type for strain calculations.

switch infin Set the strain type to Infinitesimal. This is the default.

switch alman Set the strain type to Almansi.

switch grn Set the strain type to Green.

switch rate Set the strain type to strain rate.

These two commands set the shell reference frame for shell result calculations.

switch rglob Set the shell reference frame to global. This is the default.

switch rstat Set the shell reference frame to local.

These three commands set the shell result surface for shell result calculations.

switch middle Calculate shell results at the middle surface. This is the default.

switch inner Calculate shell results at the inner surface.

switch outer Calculate shell results at the outer surface.

These two commands set the result min/max type.

switch mglob Use the global min/max for the displayed result. This is the default.

switch mstat Use the state min/max for the displayed result.

rzero *v* Set a result zero tolerance. Result values in the range (-*v*, *v*) are treated as zero values and are displayed in the default material color. Use **clrthr** to clear this value.

rmin *v* Set a minimum result value. Result values below or equal to this value are colormapped with the color that is in the first entry of the colormap. Use **cl-rthr** to clear this value.

rmax *v* — Set a maximum result value. Result values above or equal to this value are colormapped with the color that is in the last entry of the colormap. Use **clrthr** to clear this value.

clrthr — Clear the **rzero**, **rmin** and **rmax** threshold values.

The following commands are used to exaggerate displacements.

dscal *v* — Scale the nodal displacements by the specified scale value. The scaling is applied in all three axial directions. The command "**dscal 1.0**" will turn off displacement scaling.

dscalx *v* — Scale the nodal displacements in X by the specified scale value.

dscaly *v* — Scale the nodal displacements in Y by the specified scale value.

dscalz *v* — Scale the nodal displacements in Z by the specified scale value.

3.6 Colormap Commands

Colormap commands allow the user to modify the colormap. The colormap consists of 256 entries, each of which have *r*, *g* and *b* values in the range [0, 255]. Low result values are colormapped to entries low in the colortable, and high result values are colormapped to entries in the high end of the colortable. When a minimum result value or maximum result value has been set, the first and last entries of the colormap have a special role. Result values that exceed the min or max cutoffs are colormapped with the colors in those entries. When a min or max has been set, the first and last entries are drawn wider in the colormap display so that they can be seen easily.

ldmap *filename* — Load a text colormap. The specified text file should have 256 entries with three color component values for each entry.

ldhmap *filename* — Load an HDF colormap. The specified file should be an HDF format pallette with 256 entries.

posmap p_x p_y s_x s_y — Position the colormap on the screen. The arguments are the position of the lower left corner of the colormap and the size of the colormap in the X and Y directions. The view window extends from approximately -1 to 1 in each direction, so those coordinates can be used as a guide. When the colormap has been repositioned, the text annotations on the colormap are no longer displayed. This is mainly intended for animations.

hotmap — Load in the default hot-cold colormap.

grmap — Load in a grayscale colormap.

igrmap — Load in an inverse grayscale colormap.

invmap Invert the current colormap. The high value becomes the low value and vice versa.

3.7 Visualization Commands

Visualization commands provide 3D visualization capabilities such as cutting planes and isosurfaces.

timhis Draw a time-history plot. The plot is drawn to the mesh display window. The currently displayed result is plotted for the currently selected nodes or elements. Any command or action which causes the window to refresh will cause the plot to go away.

thsave *filename* Generate the data for a time-history plot and save the data to the specified file. The saved data can then be used in a graph-generating application such as Gnuplot to create more sophisticated graphs.

gather {node, beam, shell or **brick}** *result1* <*result2*> ...
Gather result data for a time-history plot. Data is gathered for the specified results at the currently selected nodes or elements. Data is gathered for the selected elements of the type specified by the first argument. This command makes it possible to gather data for multiple time-history plots in one sweep through the states. If the gather command has not been used or the data needed for the **timhis** command is not currently gathered, the program automatically gathers the needed data. Time-history plots can be regenerated quickly without the expense of collecting the data again.

mth *n1* <*n2*> ... Draw a material time-history plot. The currently displayed result is plotted for the specified materials. Up to six materials may be listed. Any action which causes the window to refresh will cause the plot to go away.

thsm *sz type* Specify a smoothing filter for smoothing time-history plots. The first argument is the width of the filter in samples (defaults to 1) and the second argument is the filter type (defaults to **box**). Currently, the box filter is the only one implemented.

on/off thsm Turn on or off smoothing of time-history plots and material time-history plots. Smoothing is off by default.

Cutting plane commands.

cutpln p_x p_y p_z n_x n_y n_z
Define a cut plane. The cut plane is defined by a point on the plane and a vector normal to the plane. The cutplane is added to any previously defined cut planes.

clrcut Delete all cut planes.

on/off rough Turn on (or off) the rough cutting plane display. The "rough cut" option deletes all elements that intersect the cutting plane or are on the side of the cutting plane in which the plane normal points. This command is useful for checking for degenerately shaped elements in the interior of a volume, or for selecting nodes in the interior of the mesh.

on/off cut Turn on (or off) display of the currently defined cutplanes.

Contour and isosurface commands.

ison *n* Clear the current isovalues and create *n* isovalues which are evenly distributed between the result minimum and the result maximum.

isop *v* Add an isovalue to the isovalue list that is *v* percent of the way between the result minimum and the result maximum. The value *v* should be in the range [0.0, 1.0].

isov *v* Add an isovalue to the isovalue list that is at result value *v*. The value *v* should be in the range [result minimum, result maximum].

clriso Clear all values from the isovalue list. In order to display contours or isosurfaces, you will need to create a list of isovalues using the above three commands. At start-up, the program has 6 evenly-distributed isovalues.

on/off con Turn on (or off) display of contour curves on the surface of the mesh. Contour curves are drawn at the isovalues in the isovalue list. The contour color can be changed using the **setcol con** command. (Warning: the algorithm for generating contour curves seems to fail on some meshes and generates gibberish. This command needs to be debugged.)

on/off iso Turn on (or off) display of isosurfaces. Isosurfaces (surfaces of constant result value) are drawn at the isovalues in the isovalue list. Isosurfaces are only displayed in volume elements.

Vector result display commands. The vgrid commands can be used to succesively build up a list of points at which to display the vector result. Vectors are drawn as small line segments. When only a few vectors are displayed, a sphere is drawn at the base of each vector to show the direction.

vgrid1 $n\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$ Define *n* evenly-spaced grid points on the line segment between point one and point two. These grid points are the points at which vector results will be displayed.

vgrid2 $n_i\ n_j\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$ Define a rectangular grid of grid points between point one (lower left) and point two (upper right). Point one and two should both lie in one of the axis-aligned planes -- i.e. they should have the same X, Y or Z value.

vgrid3 $n_i\ n_j\ n_k\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$ Define a rectangular volume of grid points with point one and point two at opposite corners of a diagonal.

clrvgr Clear the list of vector grid points.

vec *rnx rny rnz* — Draw a vector result as line segments displayed at the vgrid points. The three arguments are result names from Table 3.1 which are to be used as the X, Y and Z components of the vectors.

clrvec — End the vector result display.

setcol vec *r g b* — Set the color of the vector line segments. The color components are integers in the range (0-255).

veccm — Colormap the vectors, using the vector magnitudes as the index.

vecscl *v* — Scale all the vector line segment lengths by *v*.

Particle trace commands.

prake $n\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$ <*r g b*>

Define *n* evenly-spaced initial particle positions on the line segment between point one and point two, and add them to the current list of initial particle positions. This is referred to as a "particle rake". The last three arguments are optional and specify the color of the particle rake.

clrpar — Clear the current list of initial particle positions.

ptrace $<t_0>$ $<t_{end}>$ $<\Delta t>$

Display a particle trace. Particles begin at the initial positions specified with the **prake** command. Their paths over time are calculated by integrating the velocities. The first argument is the time at which to start the particles (defaults to 0.0). The second argument is the time at which to end the particle trace (defaults to the last state). The third argument is the time-step size for integration. All three arguments are optional.

3.8 Output Commands

Output commands are for creating hardcopy or video output. The commands are listed and then discussed in more detail below.

snap *filename* — Save the image in the mesh view window to an HDF raster image file with the given name. See discussion below for more information.

outobj *filename* — Save the current mesh and result data to a polygon object file for input to Wavefront (tm). The filenames should be basename.0001.obj, basename.0002.obj, etc.

outhid *filename* — Save the current mesh polygon data to a file for input to the HIDDEN program. See discussion below for more information.

The **snap** command is used to save the screen image to a raster file in HDF format. The HDF format was developed by the National Center for Supercomputing Applications (NCSA). This raster image can be converted to color PostScript with a variety of public-domain raster image toolkits. One such toolkit is Image Tools from the San Diego Supercomputer Center (SDSC). With this toolkit, one would use the command

imconv -hdf filename.hdf -ps filename.ps.

The authors have found that conversion to postscript with this tool is not always successful for large window sizes. An alternative is to use the Utah Raster Toolkit (URT) available from the University of Utah. With the help of this toolkit, the command sequence is

imconv -hdf filename.hdf -rle - | rleflip -r | rletops -C -h 8.8 > filename.ps

These toolkits also provide the ability to resize and rotate images, to tile a picture with smaller images, and to accomplish other common image-processing tasks.

The HDF library is available via anonymous FTP from ftp.ncsa.uiuc.edu. The SDSC Image Tools can be obtained via anonymous FTP from ftp.sdsc.edu in the directory /pub/sdsc/graphics/imtools. The Utah Raster Toolkit may be obtained with anonymous FTP from cs.utah.edu, in the directory /pub.

The GRIZ software package comes with a hidden-line program named HIDDEN for generating black&white vector PostScript images of initial and deformed mesh geometry. Polygon data for the HIDDEN program is output from GRIZ by using the command **outhid filename.hid**. Then, a PostScript hidden-line drawing can be created using the command

hidden < filename.hid > filename.ps

HIDDEN implements a very simple object-space hidden line algorithm which is quite slow but usually produces good results. It is suggested that the user run the above process in the background.

Normally, HIDDEN orients the image in either portrait mode or landscape mode based on the original window size in GRIZ. If the window was larger in the X direction, a landscape image would be produced. The default orientation can be overridden with the -p flag (potrait orientation) or -l flag (landscape orientation). Giving HIDDEN a -v flag (verbose) causes it to print status information while it is generating the hidden-line image.

3.9 Miscellaneous Commands

As one might expect, miscellaneous commands do miscellaneous things. Included in these are commands for reading and writing history files and for creating command aliases.

quit Quit the program.

info Print out various information about the current state of the program. This prints out the current state number, the total number of states, the start and end times, the current time, the number of elements, the view parameters and light positions, etc. The information is printed to the window in which the program was started.

r Repeat the last command. (Minor bug: only works once after executing the **rdhist** or **h** command. This is a result of the way these commands interact.)

alias *newcom "comstring"*
Create an alias for *comstring*. Whenever *newcom* appears as a separate word in a command, it is replaced by *comstring*. The quotes aren't needed if *comstring* consists of only one word.

load *name* Load in a new dataset. The argument is the plotfile root name for the new data.

The next four commands are for creating and reading history files. History files provide a general mechanism for running GRIZ in batch mode or for executing a series of commands at once. The history mechanism can be used to quickly customize the display for a particular dataset. A history file is just an ASCII text file with one GRIZ command per line.

savhis *filename* Begin saving commands to a file designated by *filename*. All commands are saved, including menu picks and interactive control commands.

endhis Stop saving commands to a file.

rdhis *filename* Read in a command history file and execute the commands in it. (Alias: **h**).

loop *filename* Go into an endless loop, executing the commands in the specified file over and over. This command is useful for demonstrations.

title *"Title String"* Change the data title. The data title is originally read from the plotfile, but may be changed with this command.

on/off refresh Turn on (or off) the display update while a series of commands are executed. This command can be used when the mesh image takes a long time to redraw and one wants to execute several commands before having the screen update.

pause *n* — Pause for approximately *n* seconds before executing the next command. This command is intended for demonstrations.

echo *"Some string."* — Print a string to the terminal. This command is for use in history files when the user wishes to keep track of what command has just been executed.

crease ϕ — Set the edge detection angle for surface normal smoothing. The default angle is approximately 22 degrees.

on/off shrfac — There is a bug in the SGI GL drawing microcode that causes polygons drawn on top of each other to be blended, yielding ugly artifacts. In practice, this bug becomes evident when shell elements share the same nodes as the faces of volume elements. This bug can be avoided by turning on the **shrfac** flag, but at the cost of having a slower display update.

on/off splitq — Splits quadrilateral polygons into four triangles before drawing them. The purpose of this command is to reduce the artifacts that are caused by scanline interpolation of vertex colors in the rendering software. The difference will be most visible in areas where there is a rapidly-varying result value. This option makes the rendering at least four times slower.

resttl *result "Title"* — Change the title of a result variable. This is the title that will appear above the colormap and at the bottom of time-history plots.

exec *"command line"* — Execute a UNIX command in the shell.

copyrt — Display the copyright/title screen. Any subsequent command which causes the display to update will erase this screen.

3.10 Video Commands

Video support commands provide a means of using the SGI VLAN interface to a videodisc player. The video support code will not be compiled into the program unless the environment variable VIDEO_FRAMER is defined in the Makefile. The video support routines allow the user to load frames into the video frame buffer and to record those frames to videodisc.

vidsel — Select the video framer device. This command should be given once at the beginning of a recording session, before the **vidini** command is executed.

vidini — Initialize the video disc. This command searches for the first unrecorded frame on the video disc. It should be executed before any recording is done.

vidrec <*n*> — Record a single frame or *n* frames to the videodisc. The image in the mesh view window is recorded to laser disc. The integer argument is optional, and causes the frame to be recorded *n* times. Recording multiple frames is useful at the beginning and end of an animation, for example.

vidrnm <*n*> — Record a frame or *n* frames without moving the c nt image to the video frame buffer. The image in the video frame buff recorded to laser disc.

vidmov	Copy the image in the mesh view window to the video frame buffer. The frame is not recorded.
vidcb	Load a set of NTSC colorbars into the video frame buffer. The frame is not recorded.
vidti *n "Title line"*	Set a video title line, where *n* is in the range 1 to 4. These four lines are displayed with the **vidttl** command.
vidttl	Draws a screen with the video title lines in large letters, suitable for recording to videodisc. The title can be changed with the **vidti** command. Any action which causes the screen to refresh will erase this screen. This command provides a way of recording separator frames which label animation sequences on the video disc.
on/off safe	Turn on (or off) display of the "safe action" area border. The safe action area borders are five percent of the screen size and the safe title area borders are ten percent of the screen size. The "safe action" area is where all the image content should be contained -- the region outside this area may be off-screen on standard television monitors.

4.0 COMMAND REFERENCE

The primary GRIZ commands are summarized on the following pages for quick reference.

View

rx ϕ, **ry** ϕ, **rz** ϕ	Rotate view.
tx *v*, **ty** *v*, **tz** *v*	Translate view.
scale *v*	Scale view.
scalax v_x v_y v_z	Scale view axes separately.
rview	Reset view.
switch persp	Perspective.
switch ortho	Orthographic.
rnf	Adjust near/far planes.
tmx *n v*, **tmy** *n v*, **tmz** *n v*	Translate material *n* in X, Y or Z.
clrtm	Clear all material translations.

Rendering

on/off {coord, time, title, cmap, all, box, edges, ndnum, elnum, lights, shrfac}

switch {hidden, solid, wire}	Set the rendering style.
switch {smooth, flat}	Set rendering normals.
setcol fg *r g b*	Foreground color.
setcol bg *r g b*	Background color
setcol con *r g b*	Contour color.
setcol hilite *r g b*	Hilite color.
setcol select *r g b*	Select color.
setcol vec *r g b*	Vector color.
mat *n* **<amb *r g b*> <diff *r g b*> <spec *r g b*> <shine *v*> <emis *r g b*> <alpha *v*>**	Material display properties. Ambient light color, diffuse light color, specular light color (all 0.0-1.0), shininess (0-128), emissive light color, opacity.
dellit	Delete all lights.
light *n* **pos** p_x p_y p_z **<color** *r g b***> <spotdir** v_x v_y v_z**> <spot** *exp* ϕ**>**	Define a new light.
hilite {node, beam, shell or brick} *n*	Hilite the specified element.
clrhil	Remove the hilite.
select {node, beam, shell or brick} *n1* <*n2*> <*n3*> ...	Selects the specified elements.
clrsel	Delete the selections.
on/off cent	Turn on (or off) hilite centering.
invis *n1* <*n2*> ...	Make materials invisible.
vis *n1* <*n2*> ...	Make materials visible.
sym p_x p_y p_z n_x n_y n_z	Define reflection plane.
clrsym	Delete reflection planes.
on/off sym	Turn on (or off) reflection.
switch symcu	Reflection planes cumulative.
switch symor	Reflection planes not cumulative.

Picking

switch pichil	Hiliting mode for mouse picks.
switch picsel	Selecting mode for mouse picks.
switch picsh	Pick shells with middle mouse button.
switch picbm	Pick beams with middle mouse button.

Time

state *n*	Move to state.
n, p, f, l	Next state, previous state, first state, last state.
time *t*	Move to time.
anim	Animate data.
anim *n*	Animate data.
anim $n\ t_s\ t_e$	Animate data.
stopan	Stop an animation.
animc	Continue animation.
maxst *n*	Ignore states after the one specified.
lts	List state times.

Results

show *result*	Display a result variable.
globmm	Get result min/max.
disable *n1* <*n2*> ...	Turn off results for specified materials.
enable *n1* <*n2*> ...	Turn on results for specified materials.
switch {**mglob, mstat**}	Result min/max type.
switch {**infin, alman, grn, rate**}	Strain type.
switch {**rglob, rstat**}	Shell reference frame.
switch {**middle, inner, outer**}	Shell result surface.
rzero *v*	Result zero tolerance.
rmin *v*	Result minimum value.
rmax *v*	Result maximum value.
clrthr	Clear result threshold values.
dscal *v*	Scale nodal displacements.
dscalx *v*, **dscaly** *v*, **dscalz** *v*	Scale nodal displacements.

Colormap

ldmap *filename*	Load text colormap.
ldhmap *filename*	Load HDF colormap.
posmap $p_x\ p_y\ s_x\ s_y$	Position colormap.
hotmap	Load hot-cold colormap.
grmap	Load grayscale colormap.
igrmap	Load inverse grayscale colormap.
invmap	Invert colormap.

Miscellaneous

quit	Quit the program.
info	Print out info to terminal.
r	Repeat last command.
alias *newcom* "*comstring*"	Create an alias for *comstring*.
load *name*	Load in a new dataset.
savhis *filename*	Begin saving commands to a file.
endhis	Stop saving commands to a file.
rdhis *filename*	Read in a command history file.
snap *filename*	Save image to an HDF file.
title "*Title String*"	Change the data title.
on/off shrfac	Fix SGI polygon drawing bug.

Visualization

timhis	Draw a time-history plot.
thsave *filename*	Save time-history plot data to file.
gather {node, beam, shell or **brick}** *result1* <*result2*> ...	Gather results for time-histories.
mth *n1* <*n2*> ...	Material time-history plot.
thsm *sz type*	Select a smoothing filter.
on/off thsm	Turn on (or off) time-history smoothing.
cutpln $p_x\ p_y\ p_z\ n_x\ n_y\ n_z$	Define a cut plane.
clrcut	Delete all cut planes.
on/off rough	Turn on (or off) rough cut.
on/off cut	Turn on (or off) cut.
ison *n*	Create *n* evenly-spaced isovalues.
isop *v*	Add an isovalue that is *v* percent.
isov *v*	Add an isovalue that is at result value *v*.
clriso	Clear isovalue list.
on/off con	Turn on (or off) contour curves.
on/off iso	Turn on (or off) isosurfaces.
vgrid1 $n\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$	Add vector grid points.
vgrid2 $n_i\ n_j\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$	Add vector grid points.
vgrid3 $n_i\ n_j\ n_k\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$	Add vector grid points.
clrvgr	Clear vector grid point list.
vec *rnx rny rnz*	Draw a vector result.
clrvec	End vector result display.
veccm	Colormap the vectors.
vecscl *v*	Scale vectors.
prake $n\ p_{1x}\ p_{1y}\ p_{1z}\ p_{2x}\ p_{2y}\ p_{2z}$ <*r g b*>	Add particle trace points.
clrpar	Clear particle trace point list.
ptrace <t_0> <t_{end}> <Δt>	Display particle trace.

Output

snap *filename*	Save image to an HDF file.
outobj *filename*	Save data to Wavefront obj file.
outhid *filename*	Save data for HIDDEN.

Video

vidsel	Select the video framer.
vidini	Initialize video disc for recording.
vidrec <*n*>	Record frame(s).
vidrnm <*n*>	Record frame(s) with no move.
vidmov	Copy image to video frame buffer.
vidcb	Load colorbars into video frame buffer.
vidti *n "Title line"*	Set video title line (1-4).
vidttl	Draw video title screen.
on/off safe	Show "safe action" area border.

Etc.

near *v*	Near plane.
far *v*	Far plane.
camang ϕ	Camera angle.

lookfr $p_x\, p_y\, p_z$	Look from (eye) point.
lookat $p_x\, p_y\, p_z$	Look at point.
lookup $v_x\, v_y\, v_z$	Look up vector.
tfx *v*, **tfy** *v*, **tfz** *v*	Translate look from point.
tax *v*, **tay** *v*, **taz** *v*	Translate look at point.
trfr	Connect look from point to transl widgets.
trat	Connect look at point to transl widgets.
clrtr	Reset translation widgets.
on/off bdrop	Backdrop pattern.
tlx *n v*	Translate light *n* by *v*.
tly *n v*	Translate light *n* by *v*.
tlz *n v*	Translate light *n* by *v*.
trlit *n*	Connect light *n* to transl widgets.
clrtr	Reset translation widgets.
inslp *filename*	Read in SLP polygon data.
inref *filename*	Read in reference surfaces.
on/off refsrf	Turn on (or off) reference surface display.
loop *filename*	Loop on a history file.
on/off refresh	Turn on (or off) the display update.
pause *n*	Pause for *n* seconds.
echo *"Some string"*	Print string to terminal.
crease ϕ	Set the edge detection angle.
on/off splitq	Split quadrilateral polygons into triangles.
resttl *result "Title"*	Change the title of a result.
exec *"command line"*	Execute a UNIX command.
copyrt	Display the copyright/title screen.

5.0 FUTURE DEVELOPMENT

This chapter briefly outlines the areas where the developers of GRIZ currently plan to proceed with more development. The purpose of this list is to give users a feel for where the software is headed so that they can make suggestions with that in mind. The list is not to be interpreted as a concrete list of expected changes, since priorities will change with time.

- Define and support a more flexible data file format.
- Support time-history and interface force databases.
- Improve the time-history plot capability and provide Postscript output.
- Support for 2D data and 2D analysis codes. Extrusion and revolution of 2D meshes.
- OpenGL port.
- More capability for fluid dynamics visualization.
- Have a server for downloading data from a remote machine.
- Be able to connect the program to a running analysis program for updates.
- Support shell results on an arbitrary number of mid-surfaces.

6.0 PROGRAMMER NOTES

This chapter gives an overview of the design of GRIZ. The overview is intended to aid programmers who wish to modify the program to support their needs.

The code is designed in a somewhat modular fashion. The main modules are a file reader module, a module for computing derived variables, a display module, and the user interface module (see Figure 1). Each of these modules interfaces with the main program through a limited a set of routine calls. The main program stores the mesh geometry and state data, maintains flags that control the display, and performs the visualization calculations such as generating isosurfaces and computing normals.

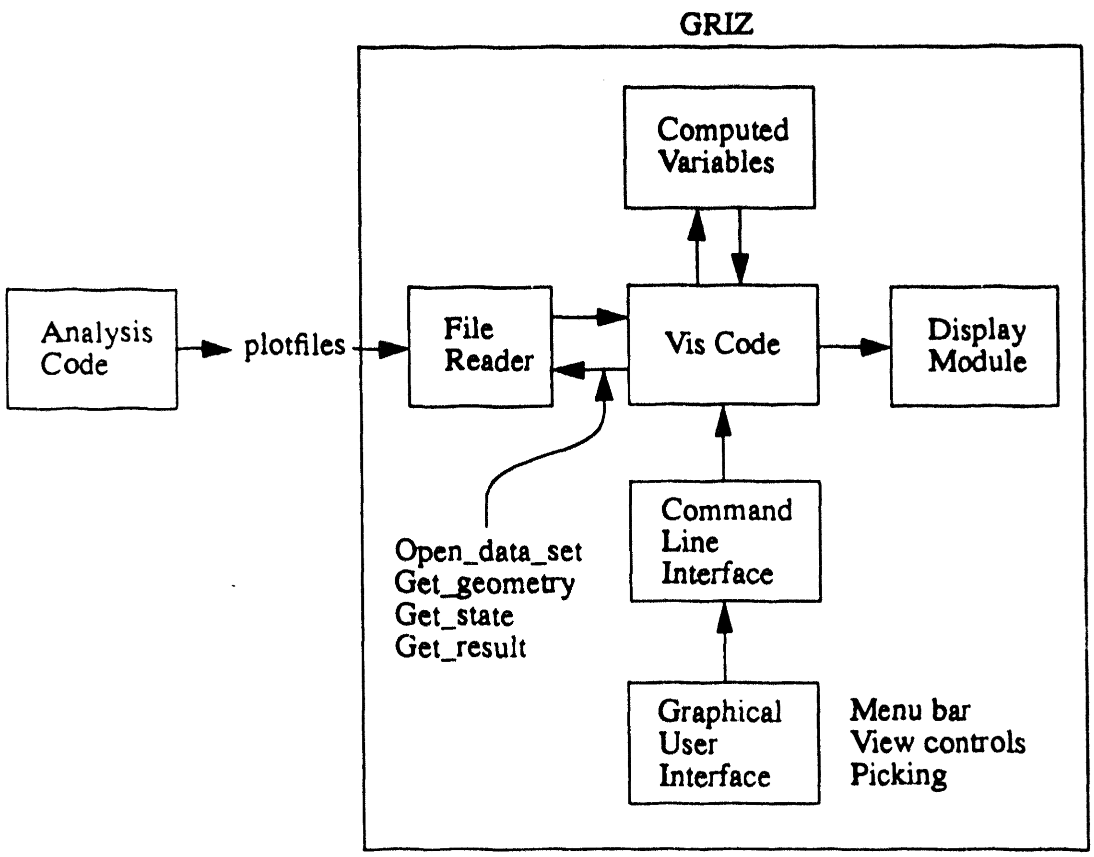

Figure 1

The file reading module provides a small set of routines which allow the main program to read data from a plotfile. Routines provided by this module include **open_file_family** (select a plotfile dataset from which to read data), **get_geom** (get the mesh connectivity and initial position), **get_state** (read in data for an individual state--this may include nodal positions and element activity data), and **get_result** (load a specific result variable at a specific state). The element activity data is optional data which controls whether elements are deleted or not at a particular state. All results are stored as scalar quantities; retrieving a vector or tensor result requires that each of its components be loaded individually.

The module for derived variables provides routines for computing derived variables from the result data stored in the data base. This module is application-specific; currently, routines are provided for computing derived variables for the DYNA3D and NIKE3D analysis codes. A table stores an ID for each result, the result title, its name as it is referred to in the command line interface, and a reference to the routine which is called to compute or load the result. Stub routines are provided for variables which can be loaded directly from the database. To include new derived variables, the programmer needs to write routines which calculate those variables and then add entries for those variables to the table.

The display module contains all display routines. This module includes the drawing routines and time-history plotting package. The module uses an immediate-mode graphics library for all of its drawing. Currently, Silicon Graphics Inc.'s GL library is used. These routines are kept in separate files and changes to the graphics library will only affect those files.

The user interface module provides the command-line and graphical interfaces. The graphical interface is based on the Motif widget set, and is layered on top of the command-line interface. Menu buttons simply call the command-line interpreter with the appropriate command to perform their actions. The view control widgets bypass the command-line interpreter for speed, but still call the command history mechanism to archive their actions if requested.

The main program is governed by a data structure which controls most of the display and visualization options of the program. This data structure also contains the mesh geometry and the data for the current state. A key component of this data structure is a table called the "face table". The face table stores each face of hexahedral volume elements. Faces that are shared between adjacent brick elements are stored only once and have one entry in the table. The face table provides an efficient means of quickly locating the external faces of the mesh, particularly when elements have been deleted or when materials have been made invisible. Flags in the face table tell whether the element on either side of a face is active at the current time.

For display purposes, averaged normals are computed at each node on the external faces of the mesh in order to give a smooth-looking surface to the mesh. It is undesirable to smooth the normals across actual edges in the mesh. Since the program has no information about where actual edges in the mesh exist, it attempts to deduce them by looking for angles greater than a critical angle between the normals of adjacent faces and the average normal at their shared node. The default critical angle is about 22 degrees, which corresponds to an angle between the adjacent faces of about 45 degrees. Obviously, this algorithm will not always do the right thing--sometimes it will smooth the normals across an edge or fail to smooth when no edge was present in the original model. The analyst who creates the mesh can avoid many of these artifacts by keeping this algorithm in mind when the mesh is designed.

INDEX

DATE FILMED

5/24/94

END