INDEX

GiD User manual

Pre and post processing system for F.E.M. calculations.

International Center For

Numerical Methods In Engineering (CIMNE)

More information about GiD:

Developers:

- Ramon Ribó ramsan@cimne.upc.es
- Miguel A. de Riera Pasenau miguel@cimne.upc.es
- Enrique Escolano escolano@cimne.upc.es

http://gid.cimne.upc.es

INTRODUCTION

GiD is an interactive graphical user interface that is used for the definition, preparation and visualization of all the data related to a numerical simulation. This data includes the definition of the geometry, materials, conditions, solution information and other parameters. The program can also generate a mesh for finite element, finite volume or finite difference analysis and write the information for a numerical simulation program in its correct format. It is also possible to run the numerical simulation from within GiD and to visualize the results from the analysis.

GiD can be customized and configured by users so that the data required for their own solver modules may be generated.These solver modules may then be included within the GiD software system.

The program works, when defining the geometry, similar to a CAD (Computer Aided Design) system but with some differences. The most important one is that the geometry is constructed in a hierarchical mode. This means that an entity of higher level (dimension) is constructed over entities of lower level; two adjacent entities will then share the same lower level entity.

All materials, conditions and solution parameters can also be defined on the geometry without the user having any knowledge of the mesh: the meshing is done once the problem has been fully defined. The advantages of doing this are that, using associative data structures, modifications can be made to the geometry and all other information will automatically be updated and ready for the analysis run.

Full graphic visualization of the geometry, mesh and conditions is available for comprehensive checking of the model before the analysis run is started. More comprehensive graphic visualization features are provided to evaluate the solution results after the analysis run. This post-processing user interface is also customizable depending on the analysis type and the results provided.

A query window appears for some confirmations or selections. This feature is also extended to the end of a session, when GiD prompts the user to save the changes, even when the normal ending has been superseded by closing the main window from the Window Manager, or in most cases with incorrect exits.

Using this manual

This User Manual has been split into five clearly differentiated parts.

It consists of a first part, **General aspects**, where the user can find the program from basics. This helps gain confidence on how to get the maximum from the interactive actions between users and the system.

The second part, **Pre-processing**, describes the pre-processing functionality. The users will learn how to configure a project, define all its parts, geometry, data and mesh.

The third part, **Analysis**, is related to the calculation process. Although it will be performed by an independent solver, it forms part of the integrated GiD system in the sense that the analysis can be run from inside GiD.

The fourth part, **Post-processing**, emphasizes aspects related to the visualization of the results.

The fifth part, **Customization**, explains how to customize the users own files to be able to introduce and run different solver modules according to their own requirements.

Different kinds of fonts are used to help the users follow all the possibilities offered by the code:

1. `font` is used for the options found in the menus and windows.
2. `` `font' `` is used for the windows names used in the post-processing.
3. **font** is used for special references in some parts.

Sections are referenced throughout the manual with the section number, section name given in square brackets as [section] and the page number.

GiD BASICS

GiD is a geometrical system in the sense that, having defined the geometry, all the attributes and conditions (i.e., material assignments, loading, conditions, etc.) are applied to the geometry without any reference or knowledge of a mesh. Only once everything is defined, is the meshing of the geometrical domain carried out. This methodology facilitates alterations to the geometry while maintaining the attributes and conditions definitions. Alterations to the attributes or conditions can simultaneously be made without the need of reassigning to the geometry. New meshes can also be generated if necessary and all the information will be automatically assigned correctly.

GiD also provides the option of defining attributes and conditions directly on the mesh once this has been generated. However, if the mesh is regenerated, it is not possible to maintain these definitions and therefore all attributes and conditions must be then redefined.

In general, the complete solution process can be defined as:

1. define geometry - points, lines, surfaces, volumes.
    - use other facilities.
    - import geometry from CAD.
2. define attributes and conditions.
3. generate mesh.
4. carry out simulation.
5. view results.

Depending upon the results in step (5) it may be necessary to return to one of the steps (1), (2) or (3) to make alterations and rerun the simulations.

Building a **geometrical domain** in GiD is based on the following four geometrical levels of entities: points, lines, surfaces and volumes. Entities of higher level are constructed over entities of lower level; two adjacent entities can therefore share the same level entity. A few examples are given:

- **example 1:** One line has two lower level entities (points), each of them at an extreme of the line. If two lines are sharing one extreme, they are really sharing the same point, which is a unique entity.

- **example 2:** When creating a new line, what is being really created is a line plus two points or a line with existing points created previously.
- **example 3:** When creating a volume, this is created over a set of existing surfaces which are joined to each other by common lines. The lines are, in turn, joined to each other by common points.

All domains are considered in 3-dimensional space but if there is no variation in the third coordinate (into the screen) the geometry is assumed to be 2-dimensional for analysis and results visualization purposes. Thus, to build a geometry with GiD, the users must first define points, join these together to form lines, create closed surfaces from the lines and define closed volumes for the surfaces. Many other facilities are provided for creating the geometrical domain; these include: copying, moving points, automatic surface creation, etc.

The geometrical domain can be created in a series of layers where each one is a separate part of the geometry. Any geometrical entity (points, lines, surfaces or volumes) can belong to a particular layer. It is then possible to view and manipulate some layers and not others. The main purpose of the use of layers is to offer a visualization and selection tool, but they are not used in the analysis. An example of the use of layers might be a chair where the four legs, seat, backrest and side arms are the different layers.

GiD has the option of importing a geometry or a mesh that has been created by a CAD program outside GiD. At present, this can be done via a DXF, IGES,VDA,STL or NASTRAN interfaces available inside GiD.

Attributes and conditions are applied to the geometrical entities (points, lines, surfaces and volumes) using the data input menus. These menus are specific to the particular solver that will be utilized for the simulation and, therefore, the solver needs to be defined before attributes are defined. The form of these menus can also be configured for the user's own solver module, as explained below and later in this manual.

Once the geometry and attributes have been defined, the mesh can be generated using the **mesh generation tools** supplied within the system. Structured and unstructured meshes containing triangular and quadrilateral surface meshes or tetrahedral and hexahedral volume meshes may be generated. The automatic mesh generation facility utilizes a background mesh concept for which the users are required to supply a minimum number of parameters.

Simulations are carried out from within GiD by using the **calculate** menu. Indeed, specific solvers require specific data that must have been prepared previously. A number of solvers may be incorporated together with the correct pre-processing interfaces.

The final stage of **graphic visualization** is flexible in order to allow the users to critically evaluate the results quickly and easily. The menu items are generally determined by the results supplied by the solver module. This not only reduces the amount of information stored but also allows a certain degree of user customization.

One of the major strengths of GiD is the ability for the users to **define and configure their own graphic user interface within GiD**. This is done by the users, defining first, via use of graphic windows, the format for the data definition windows for pre-processing. The format that GiD must use to write a file containing the necessary data in order to run the numerical simulation program must also be defined in a similar way. This pre-processor or data input interface will thus be tailored specifically for the users simulation program, but using the facilities and functionality of the GiD system.

The user's simulation program can then be included within GiD so that it may be run utilizing the calculate menu option.

The third step consists of writing an interface program that provides the results information in the format required by the GiD graphic visualizer, thereby configuring the post-processing menus. This post analysis interface may be included fully into the GiD system so that it runs automatically once the simulation run has terminated.

Details on this configuration can be found in Chapters 16 and 17.

INVOKING GiD

When starting the GiD program from a shell or script it is possible to supply some options in the same command line. The standard UNIX command is used:

gid [-h] [-p problem] [-b batchfile] [filename] [-e anything] [-n]

All options and `filename` are optional. `filename` is the name of a problem to be opened (extension `.gid` is optional)

Options are:

- **-h** shows GiD's command line arguments.
- **-p problem** loads `problem` as the type of the problem to be used for a new project.
- **-b batchfile** executes `batchfile` as a script file (see section Batch file).
- **-e anything** can continue until the end of the line. Execute `anything` as if it were a group of commands entered into GiD.
- **-n** runs the program without any window. It is most useful when used with

the option `batchfile`.

- **-c conffile** Takes window configuration from `conffile`. This file can be generated with option See section Save configuration file.

USER INTERFACE

The user interface allows the GiD user to interact with the program. It is composed of buttons, windows, icons, menus, text entries and graphical output of certain information. The interface can be configured by the user who may use as many menus and windows as required for its application.

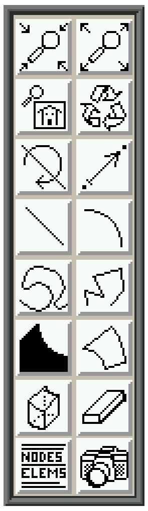
The initial layout of GiD is composed of a large graphical area, pull down menus at the top, click on menus on the right side, a command line at the bottom left and a message window above it. The project that is being run is written on the window header. The pull down menus and the click on menus are utilized for fast accessing to the GiD commands. Some of them offer a shortcut for an easier access, which is activated by clicking at the same time the keys `Control` and the letter that is displayed.

The right mouse button pressed over the graphical area, opens an on-screen menu with some visualization options. To select one of them, use the left or right button; to quit, select the left button anywhere outside the menu.

First option in this menu is called `Contextual`. It will give different options related to the current function that is being used.

A quick toolkit icons menu containing some facilities also appears on the graphical area of the window. When clicking on the icon with the left mouse button, the corresponding command is performed, whilst when clicking the right (or center, when this exists) mouse button, a short description of the command appears, which also appears when passing through the icons.

This facility is available for both actions, pre- and post-processing, although for the latter the number of functions is smaller, as some of them are inherent to the pre-processing. The window can be removed, if desired and reopened through the menu path '`Utilities` / `Graphical menu` / `Graph`'.



Icon tool box menu on Preprocess

The different icons represent, from left to right:

- **First row:**
    - Zoom in
    - Zoom out
    - Zoom frame
    - Redraw
    - Rotate trackball
    - Pan dynamic
    - Create line
    - Create arc
- **Second row:**
    - Create NURBS line
    - Create polyline
    - Create planar surface
    - Create NURBS surface
    - Create volume
    - Delete

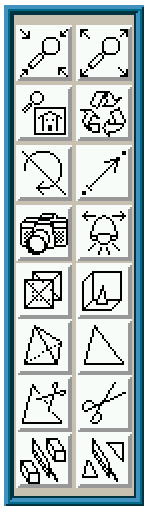
- List entities
- Print to file

For the post-processing, only the first six commands are displayed in the window, from `Zoom in` to `Pan`.

`Note:' The position of the icons depend on how the window is positioned.

If the left mouse button is pressed on `Delete`, GiD opens another window with the different entities to be deleted: `Point`, `Line`, `Surface`, `Volume` or `All types`.

If pressing the left mouse button on `List entities`, GiD opens another window with the different entities able to be listed: `Points`, `Lines`, `Surfaces` or `Volumes`.


Icon tool box menu on Postprocess

For the post-processing, the first seven commands are the same as preprocess: from `Zoom in` to `Pan` and `Print to file`.

The eighth bitmap is the `Change Light Vector` option, that allows the user to change the light direction (see section Change Light Vector).

The ninth bitmap is the `Display style` option. When cliking on it, seven more bitmaps appear, each one for every display option: Boundaries, Hidden Boundaries, All Lines, Hidden Lines, Body, Body Boundaries and Body Lines.

The tenth bitmap is the `Culling` option, that allows the user to switch on and off the front faces and/or the back faces.

The next three bitmaps allows the user to switch meshes/set/cuts on or off.

The next one allows the user to cut meshes/sets, and the last two are used to divede meshes and sets.

If windows are used to enter the data, it is generally necessary to **accept** this data before closing the window. If this is not done, the data will not be changed.

Usually, commands and operations are invoked by using the menus or mouse, but all the information can be typed into the command line.

In Windows95/NT the secondary windows appear generally in top of the main window and cannot be hiiden behind it. This mode can be changed deselecting the `Always on top` flag in the Window system menu (press second button over the windows bar to achieve it).

Mouse operations

The left mouse button is also used to make selections, selecting entities or opening a box (see section Entities selection) and to enter points in the plane z=0 (see section Point definition).

The middle mouse button is equivalent to `escape` (see section Escape).

The right mouse button opens an on-screen menu with some visualization options. To select one of them, use the left or right button; to quit, select the left button anywhere outside the menu.

First option in this menu is called `Contextual`. It will give different options related to the current function that is being used.

When the mouse is moved to different windows, depending on the situations, different cursor shapes and colors will appear on the screen.

In some windows, help is achieved pressing button-2 or button-3 over one icon.

Command line

All commands may be entered via the command line by typing the full name or only part of it (long enough to avoid confusion); case is not significant. Any command within the right side menu can be entered by the name given there or by a part of it. Special commands are also available for viewing (zoom, rotation and so on) and these can be typed or used at any time when working or from within another function. A list of these special commands is given in `View` (see section VIEW).

Commands entered by typing are word oriented. This means that the same operation is achieved if one writes the entire command and then presses `enter` or if one writes a part of it, presses `enter` and then writes the rest.

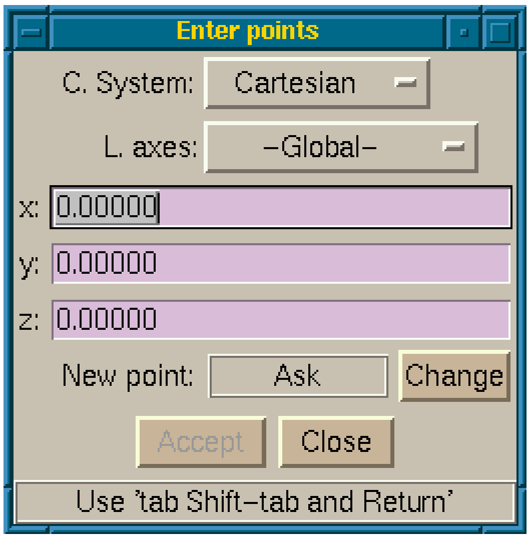
All these typed commands can be retrieved with the use of the up (to recover) and down arrows (to come back).

USER BASICS

The following features are essential to the effective use of the GiD system. They are, therefore, described apart from the pre-processing facilities section.

Point definition

Many functions inside GiD need the definition of a point to be given by the user. Points are the lowest level of geometrical entity and, therefore the most commonly used. It is, consequently, important that the user has a thorough understanding of their definition and uses. Sometimes an existing point is required and sometimes a new or an old point must be defined.



Entering coordinates window

All the options explained in this section are available through the specific window of Points Definition (see section Coordinates window). This window is accessed via the options `Utilities` and `Graphical` in the title's bar. In doing so, the user can choose not only the kind of reference system, cartesian, cylindrical or spherical, but also the system to be used, global or local and if the origin of coordinates is fixed or relative (new coordinates are referred to the last entered origin point).

In the most general case the user can enter points in the following ways:

1. Picking in the graphical window.

2. Entering points by coordinates.
3. Selecting an existing point.
4. Button `Base`.

Picking in the graphical window

Points are picked in the graphics window in the plane z=0 according to the coordinates viewed in the window. Depending on the activated preferences (see section Preferences), if the user selects a region located in the vicinity of an existing point, GiD asks whether it should create a new point or if it should use the existing one.

Entering points by coordinates

GiD offers a window for entering points in order to easily create geometries, defining fixed or relative coordinates as well as different reference systems, cartesian, cylindrical or spherical.

Coordinates of the point can be entered either in the `enter points` window or in the command line by following one of two possible formats:

1. In the format: `x,y,z`
2. In the format: `x y z`

Coordinate z can be omitted in both cases. The following are valid examples of point definitions:

```
5.2,1.0   5.2,1
8 9 2              8 9,2
```

All the points coordinates can be entered as local or global and through different reference systems in addition to the cartesian one.

1. Local/global coordinates
2. Cylindrical coordinates
3. Spherical coordinates

Local/global coordinates

Local coordinates are always considered relative to the last point that was used, created or selected. It is possible to use the commands `Utilities` and `Id` in order to make a reference to one point (see section Id). Then, to define points using local coordinates referring to the same point, use `Options` and `Fixed Relative` when entering each point. The last point selected or created before using this will be the origin of the local coordinate system. It is also possible to enter this central point by its coordinates.

The following are valid examples of defining points using local coordinates:

```
example (1):
                    1,0,0
                    @2,1,0 (actual coordinates 3,1,0)
                    @0,3,0 (actual coordinates 3,4,0)
                    2,2,2
                    @1,0,3 (actual coordinates 3,2,5)

example (2):
                    1,0,0
                    Fixed Relative (when creating the point)
                    @2,1,0 (actual coordinates 3,1,0)
                    @0,3,0 (actual coordinates 1,3,0)
                    2,2,2
                    @1,0,3 (actual coordinates 2,0,3)

example (3):
                    'local_axes_name'2.3,-4.5,0.0
```

The last example shows how to enter a point from a local coordinate system called `'local_axes_name'` (any name inside the quotation marks will fit), previously defined via the option `define local axes` (see section Local axes).

All the examples have been presented using a cartesian notation. However, cylindrical or spherical coordinates can also be used.

Cylindrical coordinates

Cylindrical coordinates can be entered as: `r<angle,z`

The z_coordinate may be omitted and angles are defined in degrees. Cylindrical coordinates can be applied to global and local coordinate systems.

The following are valid examples of the same point definitions:

```
example (1):
                    1,0,0
                    1.931852<15

example (2):
                    1,0,0
                    @1.0<30
```

Spherical coordinates

Spherical coordinates can be entered as `r<anglexy<anglez`

`Anglez` may be omitted and angles are defined in degrees. Spherical coordinates can be applied to global and local coordinate systems.

The following are valid examples of the same point definitions:

```
Example (1):
                    1,0,0
                    1.73205<18.43495<24.09484

Example (2):
                    1,0,0
                    @1.0<45<45
```

Selecting an existing point

When the user is inside a function that asks for a point, GiD can be in one of the two modes: entering a new point or selecting and old one. They can be distinguished by the cursor that will be either a cross or a box. To change from the first mode to the second one, the user must select button `join` from the right side column commands or its shortcut (Control-a). This button is then set to `No Join`. After this, select an existing point to get it. (Control-a) switches from `Join` to `No Join` and vice versa.

Options `Join` and `No join` can also be obtained from the Contextual submenu in the 3rd button menu (see section Mouse operations).

Special options `FJoin` and `FNoJoin` force GiD to change either to `Join` mode or `No join` mode independently of the previous mode.

Option Base

If button `Base` is selected (button is set on `No Base`), a point can be retrieved from any of the other modes. Then, the coordinates of this point, instead of being used, are written in the command line and can be edited before entering it.

It is possible to change the default way that GiD works with points via preferences (see section Preferences).

Option point in line

By using this option, the user can pick over a line in the graphical window. One point will be created over the line in the position where the user has picked.

Option point in surface

By using this option, the user can pick over a surface in the graphical window. One point will be created over the surface in the position where the user has picked.

Option tangent in line

By using this option, the user can pick over a line in the graphical window. One vector will be returned that is the tangent to the line in the position where the user has picked.

Option normal in surface

By using this option, the user can pick over a surface in the graphical window. One vector will be returned that is the normal to the surface in the position where the user has picked.

Entities selection

Many commands need some entities to be selected before applying them, and the method of selection is always identical. Before selecting entities, the user is prompted to decide whether to select points, lines, surfaces or volumes (in some cases this decision is obvious or it is made within the context of the option).

Within one of the generic groups (points, lines, surfaces, volumes, nodes or elements), it does not matter what type of entity is selected (for example, an arc or a spline, both line entities are selected in the same way). After this, if one entity of the desired group is selected, it is colored red to indicate its selection and the user is prompted to enter more entities. If the user selects away from any entity, a dynamic box is opened that can be defined by picking again in another place. All entities that are either totally or partly within this box are selected. Next, the user is prompted to enter more entities. If one entity is selected a second time, it becomes deselected and its color reverts to normal.

`Note:` Instead of picking twice to begin and end the selection box, it is possible to keep leftmouse pressed and move the cursor.

It is also possible to select entities by entering their label in the command line. For instance, to select the entity with number 2, the user has to input this number, `2`, in the command line. To select the entities 3 to 7, the user has to input `3:7` in the command line. Option `3:` will select all entities from number 3 to end and option `:3` will select all options from beginning to number 3.

If a layer named `lll` exists, it is possible to select all entities belonging to that layer with command: `layer:lll` . Using command `layer:` select all entities belonging to no layer.

Another way of selecting points or nodes is to write:

```
plane:a,b,c,d,r
```

Where a,b,c,d and r are real numbers that define a plane and a tolerance in the following way: `ax+by+cz+d<r`. Points close to that plane are chosen.

In some commands, another item is added to the selection group. This item, called `All`, means to select all entities levels (points...volumes) at the same time. In this case, only selection via a dynamic box is possible in the graphical window and all entities (points, lines, surfaces and volumes) in the box are selected.

To finish the entities selection, use `escape` (see section Escape).

If the option `Fast Selection` is used, entities are not colored red when selected and choosing an entity twice does not deselect it.

Caution: Use only `Fast Selection` when needing to select a large amount of entities, for example in a large mesh. The possibility of repeating entities within the selection is dangerous.

Entities belonging to frozen layers (see section Layers) are not taken into account in the selection. Entities belonging to OFF layers cannot be selected directly in the graphical window, but can be selected by giving its number or giving a range of numbers.

It is possible to add filters to the selection that, after selecting some entities, only remains selected the ones that accomplish with the filter criteria. To enter one filter write in the command line the word filter: and one option. Options are:

- HigherEntity
- MinLength
- MaxLength
- EntityType
- BadAngle

Example:

```
filter:HigherEntity=1
```

Means that only the entities that have higher entity equal to one will be selected.

It is possible to change the behavior of the selection via the Preferences (see section Preferences).

Quit

Command `Quit` is used to finish the working session. If there were changes since the last time a session was saved, GiD asks the user to save them.

Escape

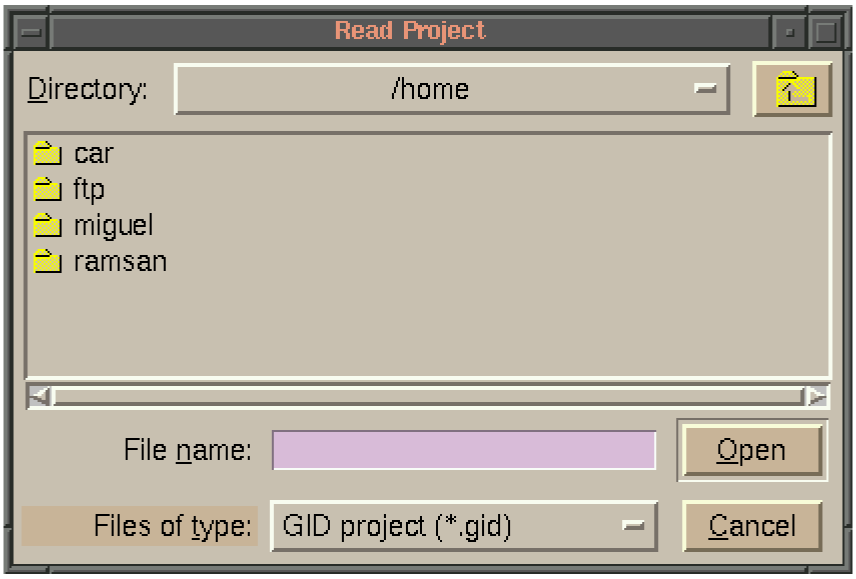
The command `escape` is used for moving up a level within the right side column menus, for finishing most commands, or for finishing selections and other utilities. This command can be applied by:

1. Middle button of the mouse.
2. Key `ESC`.
3. Button `escape` in the right side commands menu.
4. Writing the reserved word `escape` in the command line. This is useful in scripts (see section Batch file).

All above options give the same result.

Caution: `escape` is a reserved word. It cannot be used in any other context.

FILES



Browser to read and write files and projects.

GiD includes the usual ways of saving and reading saved information (`Save`, `Read`) and other file capabilities, such as importing external files, saving in other formats and so on.

New

`New` opens a new project with no title assigned.

If this option is chosen from inside a project where some changes have been introduced, GiD asks to save them before entering the new project. Next, a new problem without a title is begun.

Read

With this command, a project previously saved with `Save` (see section Save) or with `Save ASCII project` (see section Save ASCII project) can be read.

Generally, there is no difference between using a project name with the `.gid` extension or without it.

Save

`Save` a project is the way of saving all the information relative to the project: geometry, conditions, materials, mesh, etc. onto the disk.

To save a project, GiD creates a directory with its name and extension `.gid`. Some files are written into this directory containing all the information. Some of these files are binary and others are ASCII. The user can then work with this project directory as if it were a file.

User doesn't need to write the `.gid` extension because it will be automatically added to the directory name.

Caution: Be careful if changing some files manually into the **`Project`**`.gid` directory. If done in this way, some information may be corrupted.

Advice: It is advisable to save often to prevent losing information.

Save as

With this command, GiD allows the user to save the current project with another name.

When it is selected, an auxiliary window appears with all the existing projects and directories to facilitate the introduction of the project's new name and directory.

Save ASCII project

This option saves a project in the same way as the regular `save` (see section Save) but files are written in ASCII. It may be useful to copy projects between non-binary-compatible machines. GiD also allows this information to be written in a file (see section Write ASCII).

Projects saved in this way may be read with the same `read` command (see section Read).

Save ON layers

With this option, only the geometrical entities with its layer set to `ON` will be saved in a new project (see section Layers).

Note: Lower entities necessary to define the saved entities will be also saved into the new project (example: The two points extremes of a line are also saved if the line is saved).

Write ASCII

With this option a file is written containing all the information within the project. It is created in a way that is easily understood when read with an editor. This is useful for checking the information.

Note: This ASCII format is only used to check information. It cannot be read again by GiD. To write ASCII files that can be read again use the option `SaveAsciiProj` (see section Save ASCII project).

Write mesh

With this option a file is written with all the project's mesh or meshes inside. This file can be read with See section Mesh read.

Mesh read

With this option it is possible to read a mesh in order to visualize it within GiD.

It is also possible to read a new mesh and add it to the existent one. In this case, the user is prompted to keep the former one or join it to the new mesh.

The format of the file (whose name is introduced by means of the command line or directly by getting it from the auxiliary window) describing the mesh must have the following structure:

```
mesh dimension = 3 elemtype tetrahedra nnode = 4
coordinates
1 0 0 0
2 3 0 0
3 6 0 0
4 3 3 0
5 3 1.5 4
6 3 1.5 -4
7 1.5 0 2
end coordinates
elements
1 1 2 4 5 1
2 2 3 4 5 1
3 1 4 2 6 1
4 2 4 3 6 1
5 1 2 5 7 1
end elements
```

Where `elemtype` must be:

- Linear
- Triangle
- Quadrilateral
- Tetrahedra
- Hexahedra

Every element may have an optional number after the connectivities definition. This number usually defines the material type and it is useful to divide the mesh into layers to visualize it better. GiD offers the possibility of dividing the problem into different layers according to the different materials through the option `Material` (see section Layers).

If it is necessary to enter different types of elements, every type must belong to a different mesh. More than one mesh can be entered by writing one after the other, all of them in the same file. The only difference is that all meshes except the first one have nothing between `coordinates` and `end coordinates`. They share the first mesh's points. Example: to enter tetrahedra elements and triangle elements,

```
mesh dimension = 3 elemtype tetrahedra nnode = 4
coordinates
1 0 0 0
2 3 0 0
3 6 0 0
4 3 3 0
5 3 1.5 4
6 3 1.5 -4
7 1.5 0 2
end coordinates
elements
1 1 2 4 5 1
2 2 3 4 5 1
3 1 4 2 6 1
4 2 4 3 6 1
5 1 2 5 7 1
end elements
mesh dimension = 3 elemtype triangle nnode = 3
coordinates
end coordinates
elements
1 1 2 4 1
2 2 3 4 1
3 1 4 2 1
4 2 4 3 1
5 1 2 5 1
end elements
```

Caution: After and before the sign `=` it is necessary to leave a space.

DXF read

With this option it is possible to read a file in DXF format, version 12.

The entities read are mostly all types of lines, so working with surfaces and objects should either be avoided or converted into lines.

A very important parameter to consider is related to how the points must be joined. This means that points that are close to each other must be converted to a single point. This is done by defining variable ImportTolerance (see section Preferences). Points closer together than `ImportTolerance` will be considered to be a single point. Straight lines that share both points are also converted to a single line.

User can perform one `Collapse` (see section Collapse) to join more entities.

VDA read

With this option it is possible to read a file in VDA format.

A very important parameter to consider is related to how the points must be joined. This means that points that are close to each other must be converted to a single point. This is done by defining variable `ImportTolerance` (see section Preferences. Points closer together than `ImportTolerance` will be considered to be a single point. Straight lines that share both points are also converted to a single line.

User can perform one `Collapse` (see section Collapse) to join more entities.

IGES read

With this option it is possible to read a file in IGES format, version 5.1.

GiD can incorporate files in IGES version 5.1 format, incorporating most parts of the currently recognized entities.

```
Entity number and type                            Notes
100   Circular arc
102   Composite curve
104   Conic arc                ellipse, hyperbola and parabola
106   Copious data                                forms 1, 2, 12 and 63
108   Plane (form1 bounded)
110   Line
112   Parametric spline curve
114   Parametric spline surface
116   Point
118     Ruled surface
```

```
120   Surface of revolution
122   Tabulated cylinder
124   Transformation matrix                     form 0
126   Rational B-spline curve
128   Rational B-spline surface
140     Offset surface entity
142   Curve on a parametric surface
144   Trimmed surface
308   Subfigure definition
402   Associativity instance
408   Singular subfigure instance
```

The variable `ImportTolerance` (see section Preferences) controls the creation of new points when an IGES file is read. Points are therefore defined as unique if they lie further away than this tolerance distance from another already defined point. Curves are also considered identical if they have the same points at their extremes and the "mean proportional distance" between them is smaller than the tolerance. Surfaces can also be collapsed.

Entities that are read in and transformed are not necessarily identical to the original entity. For example, surfaces may be transformed into plane,into Coons or into NURBS surfaces defining their contours and shape.

IGES write

GiD can export the geometry in IGES format (version 5.1). Points, curves and surfaces are exported and volumes are ignored.

The IGES entities generated are:

```
116   Point
110   Line
102   Composite curve
126   Rational B-spline curve
128   Rational B-spline surface
142   Curve on a parametric surface
144   Trimmed surface
```

NASTRAN read

With this option it is possible to read a file in the NASTRAN format, version 68.

GiD can read files written in NASTRAN version 68 format, incorporating most parts of the currently recognized entities:

```
Entity name                                    Notes
GRID
CBAR
```

CBEAM	translated as 2 node bars
CELAS2	translated as 2 node bars
CHEXA	8 or 20 nodes
CONM2	
CORD1C	
CORD1R	
CORD1S	
CORD2C	
CORD2R	
CORD2S	
CQUAD4	
CROD	translated as 2 node bars
CTRIA3	
CTETRA	

There are two options that can be used when reading a mesh if GiD already contains a mesh:

- a) Erasing the old mesh (`Erase`)
- b) Adding the new mesh to the old without sharing the nodes; the nodes will be duplicated although they will occupy the same position in the space (`AddNotShare`).

The properties and materials of elements are currently ignored, because of the difficulties in associating the NASTRAN file properties with the requirements of the analysis programs. The user must therefore assign the materials "a posteriori" accordingly. However, in order to make this easier, the elements will be partitioned into different layers each with the name PIdnº, where nº is the property identity number associated with the elements as defined in the NASTRAN file. Note that CELAS2 elements do not have associated property identities so these will be created by default during the reading of the file.

STL mesh read

With this option it is possible to read a mesh in the STL format.

The variable `ImportTolerance` (see section Preferences) controls the creation of new points when the file is read.

Write calculations file

If GiD runs the solver module automatically, this command is not necessary. It is however useful if the solver program is required to be run outside GiD, or to check the data input prior to any calculations.

This command writes the data file needed by the solver module.

The way this file is written must be defined previously (see section COMPILATION FILES). When testing a new problem type definition, GiD produces messages about errors within the configuration. When the error is corrected, the command can be used again without quitting the example and without having to reassign any condition or meshing again.

Batch file

Sometimes, it may be useful not to use GiD interactively. To do so, commands can be written into a file and GiD will read this file and execute the commands. These commands are the same that are used in GiD when written in the command line or using the commands in the right side commands menu.

`Example:` Many points have been digitalized and their coordinates saved in a file. These points are to be joined with straight lines to create the outline of the geometry. To do so, the file would look similar to this:

```
geometry create line
3.7 4.5 8
2 5 9
4,5,6
 ...
1 7 0.0
escape
```

A batch file can also be loaded into GiD by giving its name with option **-b** when opening GiD (see section INVOKING GiD)

Another way to read batch files to create dynamic presentations is with the `Read batch window` (see section Read batch window).

One GiD session can be registered in a batch file. This can be useful to check the batch commands or to repeat one session (see section Preferences).

Note: There are some especial commands to be added to a batch than are treated differently than regular GiD commands. Their format is one or several words after the control string ***** (five stars) and everything in one line. One example would be:

```
*****OUTPUTFILENAME filename
```

In this case, 'filename' is substituted with a real file name where all the session warnings (that which appear in the GiD messages warn line) are written. This can be useful when running GiD in batch with option **-n** (see section INVOKING GiD) and GiD output is desired.

About

This command gives some information about the program, such as the version number which is being run, the system or libraries. It also enables the user to send the supplementary information that appears in the lower part of the window, when pressing `mail`, to the program developers.

Insert geometry

This command permits to insert one previously created GiD model inside another one. Entities from the old and the new model are not collapsed.

User can perform one `Collapse` (see section Collapse) to join the old and new model.

Print to file

This option asks the user for a file name and save an image in the required format.

Accepted formats are:

- `TIFF`: Tagged Image File Format.
- `EPS`: Encapsulated postscript. Useful to insert in documents.
- `Postscript screen`: Postscript. Useful to send to a postscript printer. It is a snapshot of the screen.
- `Postscript vectorial`: Postscript. Useful to send to a postscript printer. It gives higher quality but it is only usable for small modells. Otherwise, very large files are created and it takes very long time to print them.

Read surface mesh

With this option, a mesh can be read from a file in GiD format (see section Mesh read). Elements of this mesh must be triangles or quadrilaterals. This mesh is converted by GiD in a set of surfaces, points and lines. The geometric definition of surfaces is the mesh itself, but GiD treat them as truly geometric entities. For example: this surfaces can be used as the boundary of a volume, and a new mesh can be generated over them.

User is asked for the value of an angle. An angle between elements bigger than this value, is considered to be an edge and lines are inserted over them. As a consequence, a set of boundary and interior lines are created and attached to the surfaces to mark their edges.

GEOMETRY

All available geometrical operations, generating or deleting entities and performing particular options are included in this chapter.

View geometry

This command changes from mesh visualization to geometry visualization.

Create

Generation of all the different possible geometrical entities.

Point creation

Individual points are created by entering each point in the usual way (see section Point definition). The point can later be used to join lines to it.

Caution: It is impossible to create new points joining old ones.

Line creation

To create a straight line it is only necessary to enter two points (see section Point definition) and continue entering points in order to create more lines from the first one. Every part of the total line created is an independent line.

It is important to note that when creating lines, new points are also being created (if not using existing ones).

Option `Close` joins the first point and the last point created with a straight line and finishes.

Option `Undo` undoes the creation of the last point (if new) and the last line. It is possible to continue undoing back until the first point.

Option `Number` Lets the user choose the label that will be assigned to the next created line. Program checks that this line does not exist yet.

If `Join` is chosen, it is maintained for all points until `No join` is selected.

Arc creation

To create an arc it is necessary to enter 3 points (see section Point definition) or enter a radius and the two tangent lines at the arc ends.

It is important to note that when creating an arc, new points are also being created (if not using existing ones).

An anti-clockwise arc that begins and ends on one of the first or third points defined is always created. The second point is only used as a reference and, if non-existent, it is automatically erased when the arc is created.

Option `Undo` undoes the creation of the last point (if new). It is possible to continue undoing back until the first point.

Option `By tangents` lets the user input a radius and select two lines that share one common point. An arc will be then created and the two lines will be modified to be tangent and continuous with this new arc.

To convert one arc to another one with the same center and in the same plane but with complementary angle, command `Swap arcs` can be used (see section Swap arcs).

NURBS line creation

NURBS are non-uniform rational B-splines. They are a type of curves that can interpolate a set of points. NURBS can also be defined by their control polygon, another set of points that the curve approximates smoothly.

This command can be used to create a NURBS line in the following two ways.

To create a NURBS, user enters either some interpolated points or enters the points that form the control polygon (see section Point definition). Before entering the last point of the polygon, it is compulsory to press `LastPoint`.

Option `Undo` undoes the creation of the last point. It is possible to continue undoing back until the first point.

By default, NURBS will be a cubic polynomial passing through all the points. However, this option can be changed by calling `ByControlPts`, which defines NURBS by their control polygon. This polygon is a set of points where the first and the last points match the first and last points of the curve. The rest of the points do not lie on the curve. It can be assumed that the curve approximates the points of the polygon in a smooth way. In this case, user chooses the degree of the curve that will be the degree of the connected polynomials that define the NURBS.

When defining interpolating curves, user can choose to define the tangents to one or both ends (option `Tangents`). These tangents are customizable, in the sense that they can either be defined by picking their direction on the screen or by considering an existing line as a tangent to the NURBS if it follows a previous curve (option `ByLine`). The option `Next` allows only one tangent to be defined.

In this way, it is possible to create a closed NURBS by selecting the initial point as the end one, choosing option 'tangent', 'next', and 'ByLine'.

When a NURBS has been created, all the interior points (except the first and last) are not really entity points unless they previously existed.

Option `Number` allows the user choose the label that will be assigned to the next created line. Program checks that this line does not exists yet.

To enter rational weights to the curve, command See section Edit NURBS line can be used.

Polyline creation

Polylines are a set of other lines (two, at least) of any type (including polylines themselves). Every line must share one or two of its endpoints with the endpoints of other lines.

There are two possible ways to create a polyline, either by selecting one line and searching the rest until a corner or end is reached or by selecting several lines (see section Entities selection). In the case of the latter, the order of selection is not important but all of them must join each other by sharing common points.

Polylines are drawn in green to show the difference between the other lines which are drawn in blue.

Polylines are widely used when creating 4-sided surfaces (see section 4-sided surface creation) and automatic 4-sided surfaces (see section 4-sided surface automatic creation).

When deleting a polyline, all its lines are also deleted. When exploding (see section Explode polyline), the polyline will disappear and its individual lines will appear.

It is not possible to create third level polylines: one former polyline can be included inside another, but not the new one.

Option `Number` Lets the user choose the label that will be assigned to the next created line. Program checks that this line does not exists yet.

Planar surface creation

Planar surface is an entity formed by a closed set of lines, all of them lying on the same plane. Lines must share endpoints between them.

To create a planar surface, some lines must be selected (see section Entities selection). The order of selection is not important but all of them must join each other by sharing common points and must form a closed contour. If all lines are not in the same plane the surface is not created.

It is possible to add holes to a planar surface. To do so, it is first necessary to create the outside planar surface. After this, press `Hole` button and select the created surface. Then, select lines that form every hole, one by one. Finish with `escape` (see section Escape).

If the surfaces lie on the plane `z=0`, the orientation of the surfaces will be anti-clockwise in this plane (the normal vector points towards `z` positive). Otherwise, orientation will be arbitrary. This can be checked with the DrawNormals command (see section Draw surface normals).

4-sided surface creation

A 4-sided surface is an entity formed by a closed set of four lines in the space. Its mathematical definition is a bilinear Coon's surface. The surface is totally defined by the shape of the lines, with no information about the interior. This means that, sometimes, it will be necessary to use more surfaces to obtain a good shape definition.

To create a 4-sided surface defined by three lines, it is necessary to divide one of the lines in two pieces (see section Divide). Then, the creation of a 4-sided surface is possible.

To create a 4-sided surface, several lines must be selected (see section Entities selection). The order of selection is not important, but all of them must join each other by sharing common points and they must form a closed contour. If the creation is not possible, information about the endpoints is displayed in one window.

In order to make one or more lines form parts of a polyline (see section Polyline creation), select the entire polyline as one of the lines and GiD will automatically select the piece or pieces of the polyline that are required. Using this facility, nonconforming surfaces can be created. This means creating a surface by using the entire line on one side of the polyline, and creating more than one 4-sided surface by using parts of it on the other side.

When selecting more than four lines, GiD will automatically search for all the possible 4-sided surfaces that can be created with these lines. This allows the creation of many surfaces at the same time.

The button `Automatic` is equivalent to `4-sided surface automatic creation` (see section 4-sided surface automatic creation).

If the surfaces lie on the `z=0` plane, the orientation of the surfaces will be anti-clockwise in this plane (normal vector points towards `z` positive). Otherwise, the orientation will be arbitrary. This can be checked with the DrawNormals command (see section Draw surface normals).

Option `Number` Lets the user choose the label that will be assigned to the next created surface. Program checks that this surface does not exist yet.

Caution: When creating some surfaces at the same time, it is possible that some undesired surfaces may also be created. It is necessary to check the surfaces after creation and erase the undesired ones.

4-sided surface automatic creation

Inside this option, GiD creates as many 4-sided surfaces as it can find. Every new surface will be created either in the current layer or in the layer where the maximum number of surface contour lines lie.

Caution: When creating some surfaces at the same time, it is possible that some undesired surfaces may also be created. It is necessary to check the surfaces after creation and erase the undesired ones.

NURBS surface creation

NURBS are non-uniform rational B-splines. They are a type of surfaces defined by their control polygon (one set of points that the surface approximates smoothly); one set of knots (a non-decreasing list of real numbers between 0 and 1) and, optionally, one set of rational weights.

The option `By contour` creates a NURBS by their contour lines. GiD calculates automatically the interior information of the surface so as to interpolate the boundaries smoothly.

To create a NURBS surface, some lines must be selected (see section Entities selection). The order of selection is not important but all of them must join each other by sharing common points and must form a closed contour. Number of lines must be equal or bigger than one and shape must be topologically similar to a triangle or a quadrilateral in the space, for the algorithm to work correctly. This last argument is not necessary if all the lines lie in one plane. In thi case, the surface is created as a trimmed one and problems about the shape are avoided.

`Note:` Option `No trimmed` avoid the creation of a trimmed NURBS when lines are coplanar.

To enter rational weights to the surface, use command `Edit NURBS surface` (see section Edit NURBS surface).

Option `Automatic` Creates automatically all possible surfaces with a number of sides given by the user. Every new surface will be created either in the current layer or in the layer where the maximum number of surface contour lines lie.

Caution: When creating some surfaces at the same time, it is possible that some undesired surfaces may also be created. It is necessary to check the surfaces after creation and erase the undesired ones.

Option `Number` Lets the user choose the label that will be assigned to the next created surface. Program checks that this surface does not exists yet.

Option `Trimmed` let the user select one existing NURBS surface and a set of closed lines that are inside the surface. Some of this lines may belong already to the contour of the existing surface. Some other lines may be created with an intersection with another surface. Another new surface will be created without changing the old one.

Option `Parallel lines` permit to create one surface given a set of parallel lines in the space. The new surface will interpolate all the selected lines.

To draw the isoparametric lines in u,v=0.5, check variable `Curve precision:` (see section Preferences).

Volume creation

A volume is an entity formed by a closed set of surfaces that share their lines between them.

To create a volume, some surfaces must be selected (see section Entities selection). The order of selection is not important but all of them must join each other by sharing common lines and they must form a closed contour.

If there is an error and the volume is not created, helpful information is displayed in one window.

It is possible to add holes to a volume. To do so, it is first necessary to create the outside and the interior volumes as independent ones. After this, push `Hole` button and select the outside volume. Then, select the interior volumes that form every hole, one by one. Finish with `escape` (see section Escape).

Orientation of volumes and their surfaces is automatically set to facilitate a correct meshing.

An additional feature allows the selection of surfaces that form the outer part of the volume as well as the ones that form the holes at the same time. In this case, GiD automatically recognizes the holes.

When a volume is created, GiD supplies information about its approximate size.

Automatic 6-sided volumes

This option creates all possible volumes that have 6 sides. It can be applied several times over the geometry and volumes are not repeated.

It can be useful for structured meshing (see section Structured mesh).

Contact creation

Contact surfaces or contact volumes are defined between two lines or two surfaces that are physically in the same place but with different entities as surfaces, lines and points. From a contact surface, it is possible to generate contact elements or algorithms that define special contact between two bodies.

Those surfaces that have been generated from another one, although not physically in contact, or simply, identical surfaces separated by a movement, can obtain equal meshes from this option, ensuring a one-to-one relationship between nodes.

It is necessary to choose `contact surface` or `contact volume` to create contact surfaces. The first one is between lines and the second one is between surfaces. After this, it is necessary to choose lines in the first case and surfaces in the second. GiD automatically searches possible contacts combining selected surfaces in pairs.

Contact elements are, by default, 4-node planar quadrilaterals for surfaces and 8-node hexaedra or 6-node prisms (depending on the geometry) for contact volumes.

However, the user can select 2-node lines for all cases (see section Element type).

The user can also select `no mesh` for the contact entity. This makes it possible to have exactly the same mesh in both lines or both surfaces but without any additional element.

When creating the contact volumes, GiD internally checks what surfaces occupy the same location in the space and creates the contact, therefore there is no need to specify what surfaces have to be in contact. For this reason, several surfaces can be selected at once and GiD performs the contact automatically, indicating the number of contact volumes that have been generated.

For contact surfaces, however, the two lines in contact have to be specified.

A feature in GiD is the option of creating `contact separated volumes` for surfaces that are not physically in contact.

For these separated volumes, GiD internally checks whether there exists a unique solid-rigid movement between two surfaces and creates the contact between both. There is the possibility of the existence of more than one solid-rigid movement between two of them. In this situation, GiD asks for the point image of a particular original point to define the movement and, consequently, applies the right contact.

The same considerations about element type for `contact volumes` are used.

Intersection line-line

With command `Intersect lines`, two lines must be selected. GiD searches the closest points between the two lines. If the lines do not intersect, GiD determines the distance between the two closest points, draws both points and asks for confirmation. Otherwise, the new point will be the intersection point. If confirmed, the two lines are converted into four and the second point is moved to the one in the first line and converted into a unique point (if interior points of one or two of the lines coincide with endpoints, only two or three new lines are created). Polylines cannot accept this option.

Option `No divide lines` creates the intersection point but do not modify the lines.

Caution: Second selected line cannot have higher entities. First selected line can have, but only if they are NURBS surfaces.

`Note`: This option can be used to extend one line until reaching the other line.

Intersection multiple lines

This option lets the user select several lines. Then, it tries to find as many intersection points between them as possible. Lines are divided when applicable.

Option `No divide lines` creates the intersection point but do not modify the lines.

Intersection Surface 2 points

User must select one surface and two points that lie approximately over the surface. GiD calculates the line intersection between the surface and a plane defined by the two given points and the average normal to the surface in these points.

`Note`: Planar surfaces can not be used with this option.

`Note`: See section Option point in line or See section Option point in surface can be used to define the points.

Intersection Surface lines

User must select one NURBS surface and several lines. GiD calculates the intersection between the surface and the lines. Lines will be divided in the intersection point.

Option `No divide lines` creates the intersection point but do not modify the lines.

Option `Extend` Divide lines and also extends lines until they reach the surface.

Intersection surface surface

This command create the intersection lines between two surfaces. If these lines intersect surfaces contour lines, these are divided.

Option `No divide lines` creates the intersection point but do not modify the contour lines.

Object

With this command it is possible to create several objects which are:

- Circle
- polygon
- sphere
- cilynder

- cone

Most of this commands ask for a normal to define the main plane. This can be obtained with See section Option tangent in line or See section Option normal in surface among other possibilities.

Delete

The deletion of entities can be done in two ways: at one level (`point`, `line`, `surface` or `volume`) or erasing all entities at once. A selection is made (see section Entities selection) in both cases. After pressing `escape` (see section Escape), the entities are erased.

To avoid erasing the selected entities, press `break`.

Entities which depend on higher entities, cannot be erased. For example, if a surface is created over some lines, it is necessary to erase the surface before erasing the lines.

Edit

There are some particular editing options for geometrical entities:

- Move point
- Explode polyline
- Edit polyline
- Edit NURBS line
- Edit NURBS surface
- Hole NURBS surface
- Divide
- Join lines end points
- Swap arcs
- Edit SurfMesh
- Convert to NURBS line
- Convert to NURBS surface

Move point

By using this command, an existing point is selected and moved. The new position is entered in the usual way (see section Point definition). If the new position is an existing point (when using `join`), GiD will determine the distance between the points and ask if they should be joined. If answered `yes`, both points are converted into one. Lines and surfaces joined to the point are also moved and deformed to maintain their links.

Option `More points` lets the user select more points. These points will be moved the same distance as the first one. If both endpoints of a line are moved, the line is not deformed, only transferred.

See section Join lines end points, for another method of converting two points into one.

Explode polyline

This command lets the user select lines. Lines that are not polylines or have higher entities or conditions are rejected. After confirmation, the polylines are exploded and converted into their original lines. Polylines then disappear (see section Polyline creation).

Edit polyline

The command `Edit Polyline` allows the user select lines. Lines that are not polylines are rejected. It is possible to choose several options for the polylines:

- **Use points:** When meshing this polyline, there will be at least one node at every point location that defines the polyline. These will be the endpoints of interior lines.
- **Not use points:** When meshing this polyline, the mesh generator ignores the points and therefore, the nodes will be placed anywhere. This is the default option. Nodes will only be put in the position of a point if there is a 4-sided surface over a part of a polyline (see section 4-sided surface automatic creation).
- **Only points:** When meshing this polyline, the nodes will only be placed where the geometry points are.

`Note`: If one condition is assigned to one interior point of a polyline (see section

Conditions), one node of the mesh will be placed over that point.

Edit NURBS line

Once a NURBS line is selected, it is possible to interactively edit its control points (see section NURBS line creation). Select the control points as if they were regular points and enter their new positions in the usual way (see section Point definition).

Options:

- **Insert knot:** User is asked by a knot value between 0.0 and 1.0 and this is inserted. Program checks that knot multiplicity is not bigger than order (order=degree+1). As the number of knots increase, the number of control points also increase. So, this option can be used to have more points defining the same curve.
- **Degree elevate:** With this option degree of the curve is elevated by one. The new curve will have the same shape but with more control points and knots.
- **Weight change:** A new positive weight can be introduced for any control point, except the end points.
- **Cancel weights:** All weights of the NURBS are converted to 1.0 and the curve is no more rational.
- **Reparametrize:** With the same control points a new curve is calculate that have a better advance of the knots list related to the control points.
- **Break:** Quits the function leaving the NURBS unchanged.

Edit NURBS surface

Once one NURBS surface is selected, it is possible to interactively edit its control points (see section NURBS surface creation). Select the control points as if they were regular points and enter their new positions in the usual way (see section Point definition).

Options:

- **Insert knot:** User is asked by a knot value between 0.0 and 1.0 and it is inserted. Program checks that knot multiplicity is not bigger than order. This option can be used to have more points defining the same surface.
- **Degree elevate:** With this option degree of the surface is elevated by one. The new surface will have the same shape but with more control points and

knots.

- **Weight change:** A new positive weight can be introduced for any control point, except the extremes.
- **Cancel weights:** Converts the weights of all the control points to 1.0
- **Break:** Quits the function leaving the NURBS unchanged.

`Note:` Options `Insert knot` and `Degree elevate` can be chosen for the `u` or the `v` parameters direction.

Hole NURBS surface

This option let the user select one existing NURBS surface and a set of closed lines that are inside the surface and that form a hole inside it. The lines may be created with an intersection with another surface. The hole will be added to the existing surface.

Divide

Command `Divide`, can be applied either to lines that are not polylines or to NURBS surfaces not trimmed. After selecting the entity, the division can be given in several ways:

- **Number of divisions:** line or surface will be converted in equally spaced pieces.
- **By point:** With this option one point must be selected near the entity (See section Option point in line or See section Option point in surface can be used). The entity will be divided into two entities near that point.
- **By factor:** One factor is given between 0.0 and 1.0 and entity will be divide into two where the parametric variable takes that value.

After the division, the entity disappears and new entities are created.

`Note:` In the case of surfaces it is necessary to give the division sense that can be `u` or `v`. This sense can be previously checked with the command See section Edit NURBS surface.

If the line is a polyline, the user must choose an existing interior point. The polyline will be converted into two lines that may or not be polylines.

Polyline division have an option `Angle` that permits to divide the polyline in all the points where the angle between the sublines before and after is bigger than the given one.

Caution: An interior point must belong to level one of a polyline (see section Polyline creation).

Join lines end points

With command `Join lines end points`, two lines must be selected. GiD determines the distance between the two closest endpoints, draws both points and asks for confirmation. If one of the lines is a polyline, interior points are also considered. If accepted, the points are converted into one and the lines are deformed. The new point will then take the place of the first line's point.

See section Move point, for another method of converting two points into one.

Caution: Second selected line cannot have higher entities.

Swap arcs

This command lets the user select and alter arcs. Lines that are not arcs are rejected. After confirmation, the arc is converted to a new arc with the same center and in the same plane but opposite the old one. The old arc disappears and the angle of the new arc will be supplementary to the angle of the old arc.

Caution: Arcs with higher entities cannot be swapped.

Edit SurfMesh

User must select one or several surface meshes (see section Read surface mesh). Options are:

- **Draw mesh:** Surface will be drawn as a mesh.
- **No draw mesh:** Surface will be drawn as a regular surface. With magenta lines close to the boundary lines.

Convert to NURBS line

Some lines must be selected (see section Entities selection) and, after confirmation, they are converted to NURBS lines.

Lines which have higher entities above them or have conditions assigned (see section Conditions), cannot be converted. For example, if a surface is created over some lines, it is necessary to erase the surface before converting the lines.

Convert to NURBS surface

After confirmation, all the surfaces of the model are converted to NURBS surfaces.

DATA

All the data that defines the problem and that is managed in the data menus, depends on the problem type and will change for every different problem type. The following help will describe the common interfaces to all the possible data.

Data for a problem is defined by the following parameters: conditions (see section Conditions), materials properties (see section Materials), problem data (see section Problem data) and intervals data (see section Interval data) that define the problem. The conditions and materials have to be assigned to geometrical entities. It is also possible to define new reference systems (see section Local axes).

Problem type

This option permits to select between all installed problem types. When selecting a new problem type, all information about materials, conditions and other data parameters that were already selected or defined is lost.

Option `Others...` permits to enter one installed problem type or one that is in the current directory. This last possibility is useful when creating a new problem type and it cannot be installed until finished.

Note: When defining a new problem type, as this is not installed, it must be selected by other means. One possibility is to use the 'Others...' possibility in the 'problem types' menu. Another possibility is to use the following commands in the right buttons menu or written in the command line: `data defaults problemtype`.

Conditions

Conditions are all the properties of a problem (except materials), that can be assigned to an entity.

One example would be the boundary forces and displacement constraints in a solid mechanic analysis or initial velocities in a CFD analysis. Information about contact between master-slave nodes can also be considered as `conditions`.

Caution: If a mesh has already been generated, any change in the condition assignments, requires to mesh again to transfer these new conditions to the mesh. If this new generation has not been made, user will be warned when the data for the analysis is being written.

Assign condition

The condition is assigned to the entities with the given field values.

If assigning from the command `Assign Cond`, the option `Change` allows the definition of the field values. Do not forget to change these values before assigning. Option `Delete all` erases all the assigned entities of this particular condition.

Conditions can be assigned both on the geometry and on the mesh but it is advisable to assign them on the geometry and the conditions will then be automatically transferred to the mesh. If assigned on the mesh, when remeshing, conditions are lost.

Conditions that are to be attached to the boundary of the elements, are assigned to the elements and GiD searches the boundaries of the elements that are boundaries of the total mesh.

Option `Unassign` inside `AssignCond`, permits to unassign this condition. It is also possible to unassign from only certain entities.

Caution: If a mesh has already been generated, for any change in the condition assignments, it is necessary to mesh again.

Some conditions may have a behavior that depends on the type of the chosen axes. The user can choose whether to use global axes or any of the local axes previously described with the command `Local axes` (see section Local axes), or, alternatively, another option of automatic local axes calculations. In this latter case, different axes are created according to the adopted criteria of tangency and orthogonality with the geometry.

The user can apply all graphical functions (`zoom`, `rotate`, ...) and all the condition symbols are maintained in active mode in the graphical window until the user selects `escape` or presses `Finish` in the conditions window.

Draw condition

Option `Draw all` draws all the conditions assigned to all the entities. This means to draw a graphical symbol or condition number over every entity that has this condition.

If one particular condition is selected, it is possible to choose between `Draw` and one of the fields. `Draw` is like `Draw all` but only for one particular condition. If one field is chosen, the value of this field is written over all the entities that have this condition assigned.

When the condition has any field referred to the type of axes, the latter can be visualized by means of `Draw local axes`.

Unassign condition

In window mode, command `UnAssign` lets the user choose between three possibilities:

- Unassign one condition from some selected entities.
- Unassign one condition from all the entities that may have this assigned.
- Unassign all conditions from all the entities that may have them assigned.

In command mode `UnAssing`, do it for all the conditions. To unassign only one condition, use command `Delete All` (see section Assign condition).

Materials

For any problem that needs definition of materials, there is a database of existing materials that can be assigned to entities. The user can also create new materials derived from existing ones and assign them as well.

Caution: If a mesh has already been generated, for any change in the assigned materials, it is necessary to mesh again or assign the materials directly to the mesh.

Assign material

The material is assigned to the selected entities.

If assigning from a window, every time the assigned material changes, button `Assign` must be pressed again.

The user must select the entity where to assign the materials over: `line, surface` `or volume` when working in **geometry** mode or directly over the `elements` when working in **mesh** mode.

If assigning from the command line, option `UnAssignMat` erases all the assignments of this particular material.

Caution: Although a mesh had been already generated, for any change in the assigned materials, it is necessary to mesh again or assign the materials directly to the mesh.

Draw material

This option draws a color indicating the selected material for all the entities that have the required material assigned. It is possible to draw just one material type or alternatively, draw all materials. To select some of them use `a:b` and all material numbers that lie between `a` and `b` will be drawn.

Caution: When drawing materials in 3 dimensions, it may be necessary to change the viewing mode to polygons or render (see section Render) to distinguish the front and back of the objects.

Unassign material

Command `Unassign` unassigns all the materials from all the entities. For only one material, use `UnAssignMat` (see section Assign material).

New material

When using command `NewMaterial`, a new material is created taking an existing one as a base material. Base material means that the new material will have the same fields as the base one. Then, all the new values for the fields can be entered in the command line.

It is possible to redefine an existing material.

To create a new material or redefine an old one in the materials window, write a new name or the same one and change some of the properties. Then, push `Accept`.

Problem data

Problem data refers to all the data that is general to the problem. This means that it is not related to a geometrical entity and it does not change in every interval of the analysis.

It can be entered with the command `ProblemData` or in the problem data window.

If entered in a window, the data is not accepted until the button `Accept` is pressed.

This data can be entered before or after meshing.

Intervals

Intervals are a way to separate some information into several groups. The information that can be duplicated for every group can also be duplicated, if desired. When a new interval is defined, it is possible to copy all the new information about conditions, assigned to entities or not.

Therefore, the correct way to work is to define all the conditions first and afterwards create the new intervals.

Options are:

- **New:** It is allowed to define as many intervals as desired with this command. When creating a new one, user can choose whether copy the assigned conditions or not. To copy them, conditions must already be assigned (see section Conditions).
- **Current:** Chooses the current interval to use. Next assigned conditions or interval data (see section Interval data), will be considered inside this interval.
- **Delete:** Deletes one existing interval and all its related data.

These groups called intervals can be used to change some conditions or information like increment factor in an incremental analysis. It may also be useful to define some load states for the same geometry.

Interval data

This is the amount of information that is maintained different for every interval (see section Intervals).

It can be entered with the command `IntervalData` or in the intervals data window.

If entered in a window, the data is not accepted until button `Accept` is pressed.

This data can be entered before or after meshing.

Local axes

With this option, GiD allows to define new different coordinates reference systems. It is possible to write them, not only as cartesian reference systems, but also referring to as Euler angles. All user defined systems are automatically calculated and can be visualized one by one or all together.

When defining local axes, the definition mode is done through three points `3PointsXZ`, what corresponds to the origin, the X-direction and the Z-direction. The origin and the last introduced point define the Z-axis, whereas the second point indicates the side of the `x-z` plane where the point lies.

Local axes can later be used when creating a point (see section Point creation) or in some conditions that have a field related to these axes (see section Conditions).

MESHING

Generating a mesh is the process where a finite element mesh is calculated from the geometry definition. This mesh will be used for the FEM analysis in a later step. Conditions (see section Conditions) and materials (see section Materials) assigned to geometric entities will be transferred to the nodes and elements of the new mesh.

What is meshed and how, is controlled by some default options which can be changed with the commands described later.

The generation does not depend on the state of the layers at the moment of generation (see section Layers). All layers are meshed and every node and element will be assigned to the layer in which the original geometrical entity was defined.

Defaults are:

- An entity is meshed if does not belong to a higher level entity.
- A line mesh is made of two-noded elements and a surface mesh is made of non-structured triangular elements. The default for structured meshes are quadrilateral elements. Volume meshes are composed by non-structured tetrahedral elements. Structured volume meshes are composed by hexahedra.

All these default elements use linear interpolations for the unknown variables.

Generate

When everything is ready for mesh generation, select this command. If there is a previously generated mesh, GiD asks if this should be deleted, i.e., make it disappear -from memory, not from disk until the next save (see section Save).

The mesher or mesher combination, can be chosen in (see section Preferences).

Next, GiD asks for a general element size which will be applied to all lines, surfaces and volumes that do not have a previously defined one (see section Assign sizes). GiD offers to the user to default possibilities:

- Un default size automatically calculated by the program to define a coarse mesh.
- Last size given by the user in previus meshings.

User can choose one of these or enter a new one.

Size is given by the average side of the corresponding triangle or quadrilateral.

Progress in meshing is shown by one progress bar that indicates number of generated surfaces or volumes related to total number of surfaces or volumes.

Meshing can be stopped at any time by pressing button `Stop`. Sometimes it is necessary to press this button for some time to get the action done.

Mesh view

When a mesh has already been generated, this option changes the visualization to mesh viewing. The reverse, the return to geometry, is automatically performed when selecting option `View geometry` (see section View geometry) or any command related to geometry.

View boundaries

This option draws the boundaries of the mesh on the screen when chosen.

It is useful to notice adjacent surfaces with inverted normals, as in this case an edge between them appears. Analogously, it can detect meshing errors in closed volumes, as in this case there would not exist any boundary.

Boundaries for triangular or quadrilateral meshes are line-elements. Boundaries for tetrahedra or bricks meshes are triangles or quadrilaterals. This option can be useful to render a volume mesh (see section Render).

Assign sizes

Size is given by the average side of the corresponding triangle or quadrilateral.

It is possible to assign different sizes to different entities of the mesh. This means that in the vicinity of these entities, the generated elements will be approximately of that size. All the entities that have not an assigned size when meshing, take the default size. Points do not take any size if none is given to them.

To assign a size of 0.0 to an entity is the same as setting the default size. The transition between different sizes is controlled by a parameter in preferences (see section Preferences). Option `By geometry` asks the user for a minimum and a maximum size and assigns sizes to all the entities depending on the shape of the geometry. This means that smaller surfaces will have smaller elements. It will only change the existing sizes if the new size is smaller than that previously defined. Option `By cordal error` asks the user for a minimum and a maximum size and a cordal error (the maximum distance between the generated element and the real geometry). GiD assigns the corresponding sizes to all the entities to accomplish this condition. It will only change the current sizes if the new one is smaller than the previously defined. In structured surfaces, stretching is permitted. It means that if necessary, elements can have very different size along both principal directions.

Option `Correct sizes` Asks the user for a maximum size and checks all the assigned sizes in the model. It will reduce some sizes in order to control that the transitions between sizes in close entities is not too fast. It will only change the current sizes if the new one is smaller than that previously defined.

`Note:` To mesh a difficult volume trying to adjust element sizes to geometry detail, it may be useful to use `By geometry` option setting the larger size equal to the default size for meshing, and the smaller size reflecting the details (10 times smaller, for example). After, use `Correct sizes` option with equal large size.

Caution: Be careful when assigning big sizes to entities close to others where a small size has been given. It may be impossible to obtain a mesh.

Caution: When using contact elements (see section Contact creation), the same size must be used for contact and for duplicate entities.

Structured mesh

A structured mesh is defined as a mesh where all the nodes have the same number of elements around it.

The size of the elements is defined in a different way than for a non-structured mesh. In this case, the mesh is not defined by the size but by the number of elements that are required on every line. This number must be the same for all lines that are opposite to each other on each surface. When meshing volumes, this definition must be the same for opposite surfaces.

To create a structured mesh, choose `structured` and volumes, surfaces or lines. Surfaces can be 4-sided or NURBS ones. Planar surfaces meshes cannot be structured. After choosing `escape`, the number of elements per line is given. Later, lines can be selected and related lines (in option surface or volume), are added or deleted from the group. This process can be repeated as many times as necessary until all lines have a new value. By default, the generated elements will be quadrilateral or hexahedra and the lines with no numbering given will have two elements over them. All no selected lines will also have two elements by default.

Structured volumes must have 6 faces and structured surfaces must be 4-sided surfaces or NURBS surfaces.

It is possible to mix some entities with structured meshes and others with unstructured ones.

To convert a structured entity to a non-structured one, select reset (see section Reset mesh data) or assign a size to it (see section Assign sizes).

To change the default element type See section Element type.

Note 1: One NURBS surface can be structured with any number of contour lines but must have a good shape form. This means that it must have 4 big angles and the other angles must be small. With this criteria, the shape will be topologically similar to one quadrilateral.

Note 2: When assigning structured divisions to one line or with difficult topology, GiD may need to reassign some sizes to make the structured mesh conformal. It will be done automatically. If it is impossible to create compatibility between surfaces, a message is given.

Structured concentrate

By default, all partitions in one structured line have the same length. This command lets the user select one line. Then, in the graphical window the sense of the line is shown with one arrow. The user will enter a positive or negative weight. If the weight is positive the size of the elements will be smaller at the beginning of the line. If negative, the elements will be concentrated at the end of the line.

As the magnitude of the weight increases, the diference between element sizes will be bigger.

Mesh criteria

GiD provides three different criteria to generate the mesh. The `Default` option skips meshing the boundaries, that is, lines for surface meshes and surfaces for volume meshes. The `Mesh` option allows the user to choose the entities to be meshed, whilst the `No Mesh` option does the opposite.

Element type

With this command, the type of element desired is selected. It is only necessary to do this when the element type is different from the default (see section MESHING).

Types are:

- **Linear:** for lines.
- **Triangle and Quadrilateral:** for surfaces.
- **Tetrahedra and Hexahedra:** for volumes.
- **Only points:** Just for volumes. No elements are generated. Only nodes.
- **Linear, Quadrilateral and Hexahedra:** for contact surfaces.

By default, the elements are of minimum order: 3-noded triangle, 4-noded quadrilateral and so on. To increase the degree, use command `Quadratic` (pSee section Quadratic). Option `Quadratic` applies to all the elements of the problem.

The `Linear` option assures, not only the meshing of lines, but also the creation of 2-node contact lines between surfaces or volumes, if desired.

At contact surfaces, GiD elements are:

- **Linear:** 2 nodes.
- **Triangle:** 3 nodes.
- **Quadrilateral:** 4 nodes.

At contact volumes (and separated contacts), elements are:

- **Linear:** 2 nodes.
- **Prisma:** 6 nodes.
- **Hexahedra:** 8 nodes.

To decide what parts of the geometry should be meshed use command: See section Mesh criteria.

Quadratic

Option `Quadratic` applies to all the elements of the problem. If chosen, elements will be:

- **Linear:** 3 nodes.
- **Triangle:** 6 nodes.
- **Quadrilateral:** 8 nodes.
- **Tetrahedra:** not yet implemented.
- **Hexahedra:** 20 nodes.

Option `Quadratic9` is similar to option `Quadratic`, but will generate 9-noded Quadrilaterals and 27-noded Hexahedra.

Reset mesh data

This command resets, all the sizes assigned to entities. This means that all of them will be unassigned.

To unassign only some entities, assign size 0.0 (see section Assign sizes) to the entities where the default size is required.

The information about element types, mesh criteria and quadratic parameters is also reset.

Cancel mesh

If a mesh has been previously generated, this option erases the mesh and makes it disappear.

Mesh quality

This option opens a window that shows graphics about the quality of the mesh elements.

The quality criteria is the minimum angle in surface elements and the minimum dihedral angle for volume elements. This means that elements with a small angle are considered of a worse quality than the ones with bigger angles.

Edit mesh

These option let the user modify one mesh. All modification will be lost when the mesh is generated again.

Move node

By using this command, an existing node is selected and moved. The new position is entered in the usual way (see section Point definition).

Delete elements

To delete elements a selection is made (see section Entities selection). After pressing `escape` (see section Escape), the elements are erased.

The deletion of entities can be done in two ways: at one level (`point`, `line`, `surface` or `volume`) or erasing all entities at once. A selection is made (see section Entities selection) in both cases. After pressing `escape` (see section Escape), the elements are erased.

Nodes that do not belong to any element after the operation are also erased.

To avoid erasing the selected elements, press `break`.

Delete lonely nodes

After confirmation, all the nodes of the mesh that do not belong to any element are erased.

Note: This command is only necessary for meshes imported from outside because internally generated or edited meshes, have this problem implicitly corrected.

VIEW

Visualization commands change the way to display the information in the graphical window. They do not change any definition of the geometry or any other data.

Generally, they can be used within any other command without leaving it. When the visualization process finishes, the first command continues.

Zoom

Zoom is used to change the visualized size of the objects without deforming them. This only enlarges or reduces the objects and changes the orthogonal perspective of the window.

- `Zoom in`: Pick inside the graphical window. A dynamic rectangle is opened. Pick again and the visualization changes to display only the part within the defined rectangle.
- `Zoom out`: Pick inside the graphical window. A dynamic rectangle is opened. Pick again and the visualization changes so that everything in the graphical window is reduced to the size of the rectangle.
- `Zoom frame`: Choose a visualization size so as to display everything inside the window.
- `Zoom points`: User enters two points (see section Point definition), and a visualization size is chosen so as to display these two points inside the window.
- `Zoom dynamic`: Moving mouse horizontally, view is enlarged or shortened.
- `Zoom previous`: GiD goes to the previous saved zoom.
- `Zoom next`: If one previous zoom has been made, this option goes to the one after the current view in the stack list.

`Note`: Instead of picking twice to begin and end the rectangle, it is possible to maintain the leftmouse pressed and move the cursor.

Rotate

There are various ways to make a rotation in order to change the graphical view of the geometry, without changing it.

`Note`: Instead of picking twice to begin and end the rotation, it is possible to maintain the leftmouse pressed and move the cursor.

Rotate screen axes

With this option, a dynamic rotation about the screen axes is made. Screen axes are defined as:

- `X-axis`: A horizontal axis.
- `Y-axis`: A vertical axis.
- `Z-axis`: An axis orthogonal to the screen.

When entering this command, `Z-axis` is set by default and moving the mouse to the left or to the right will rotate the geometry around this axis. Picking the left mouse changes the axis. To leave this function, use `escape` (see section Escape).

To change the axes is also possible by entering the letters `x`, `y` or `z` in the command line.

To move the geometry by a fixed angle, enter the number of degrees, positive or negative, in the command line.

Rotate object axes

With this option, a dynamic rotation about the object axes is made. Object axes are the global axes in the position they are in that particular moment. i.e. the axes currently drawn in the graphical window.

When entering this command, `Z-axis` is set by default and moving the mouse to the left or to the right will rotate the geometry around this axis. Picking the left mouse changes the axes. To leave this function, use `escape` (see section Escape).

To change the axes is also possible by entering the letters `x`, `y` or `z` in the command line.

To move the geometry by a fixed angle, enter the number of degrees, positive or negative, in the command line.

Rotate trackball

With this option, a dynamic rotation is made, resembling a trackball device. It means that when picking over a geometry point with the left mouse and moving the mouse, the geometric point tries to follow the mouse pointer. This can be imagined as a ball over the graphical window which is moved with the mouse.

The left mouse button can be pressed several times to connect and disconnect the movement. To leave this function, use `escape` (see section Escape).

Rotate angle

The new position of the geometry after the rotation can be defined as the direction orthogonal to the screen via a pair of angles:

1. The **angle in the plane XY** starting from the `X-axis`.
2. The **elevation angle** from the `XY` plane.

As an example, the initial view (orthogonal to the Z-axis and with the X-axis horizontal) can be obtained with:

```
rotate angle 270 90
```

Rotate points

This option appears only in the right side commands menu.

The new position of the geometry after the rotation can be defined as the direction orthogonal to the screen via a pair of points:

1. The **target point**, the point to look at.
2. The **view point**, the point to look from.

Rotate center

The default center of rotation is defined approximately in the center of the geometry.

If it is desired to change this center, use this command and enter a point (see section Point definition). This new rotation center will be maintained until the next `zoom frame` (see section Zoom).

Rotate original view

This option, that appears only in the graphical pop-up menu, changes the view to the original one: screen orthogonal to the Z-axis with the X-axis lying horizontally and pointing to the right.

Pan

- `Two points`: With this command, the location of the geometry is displaced in the graphical window. To do this, pick two points in the graphical window.
- `dynamic`: In this case the displacement is made dynamically, moving the mouse horizontally.

Redraw

This command redraws the geometrical model in the graphical window. For those machines that include overlays, all the layers that stay underneath are not affected, so the redraw goes faster and the underneath drawings remain untouched.

Render

By using this option, the way of viewing the model changes. There are four options:

- `Normal`: This is the usual way of visualizing. The geometry and mesh are viewed including all definition lines.
- `Filled`: Solid model with no illumination and lines.
- `Flat lighting`: Solid model with flat illumination and lines.
- `Smooth lighting`: Solid model with smooth illumination (better quality).

Note: The rendering of a volume mesh requires the option `View boundaries` (see section View boundaries).

Note: Quality of visualization is controlled via preferences (see section Preferences).

Note: The light vector direction can be changed interactively with the menu option `Change Light Vector` inside the menu `Utilities` (see section Change Light Vector).

Label

With this option, the labels of the entities are drawn or not. It is possible to select some entities or to apply this option to all entities viewed in the graphical window. To select some entities, choose `select` before applying on the `Points`, `Lines`, `Surfaces` or `Volumes`. Then, select entities in the usual way (see section Entities selection).

Options are:

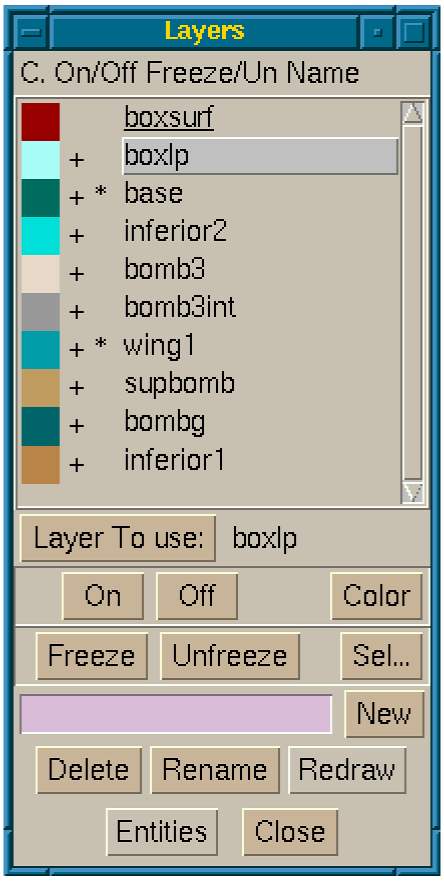
- `All`: All entities in the graphical window will have their labels drawn.
- `points,lines,surfaces` or `volumes`: If `select` is not set, all the entities in the graphical window will have their labels drawn.
- `Off`: Finish drawing labels for all entities.

Entities

With this option, it is possible to choose some of the `points`, `lines`, `surfaces or volumes` to be drawn. It is useful to make drawing faster or clearer in some instances.

Layers

Layers are a way to split a complex drawing into separate pieces. The idea is that any entity can belong to one layer or to none (an entity cannot belong to more than one layer). Then, it is possible to view only some layers and not others. It is also useful to easily select entities in the graphical window.



Layers window operations

Commands related to layers are:

- `New`: Create a new layer. This will be used as the default layer until the end of the session or until it is changed.
- `To use`: Select a layer to be used as default. All the new entities will be created within this layer. All the layers are capable of being selected.
- `On`: Entities belonging to this layer will be drawn and can be selected in the graphical window.
- `Off`: Entities belonging to this layer will not be drawn and cannot be selected in the graphical window.
- `Freeze`: Entities belonging to this layer will be drawn but cannot be selected in any way. When copying entities (see section Copy) and sharing old entities, entities belonging to frozen layers are not taken into account and not even a range of numbers (see section Entities selection) can be selected. The opposite command is `Unfreeze`, where the entities belonging to this layer can be selected, copied and shared.

- `Delete`: Delete a layer. A layer can only be deleted if it has no entities in it.
- `View`: Gives information about the layer.
- `Entities`: Moves entities to a new layer. No new entities are created. They are only moved from one layer to another. When choosing, in **geometry** mode, `point`, `line`, `surface`, `volume` or `all`, option `all` can be used only to select entities in the graphical window and change all the entities in the dynamic box to the new layer. In **mesh** mode, it is possible to choose `nodes`, `elements` or `all`. Especial option `Also lower entities` permits to send to the layer also the entities that are lower entities of the selected ones. Example: If this flag is set and one surface is selected, its lines and its points are also sent to the layer.
- `Color`: Change color to layer. This color is used when making a render (see section Render).
- `Material`: Move all entities that have a particular material number to the layer. It is useful if reading a mesh with materials from outside GiD (see section Mesh read).
- `ChangeName`: Changes the name of a layer.

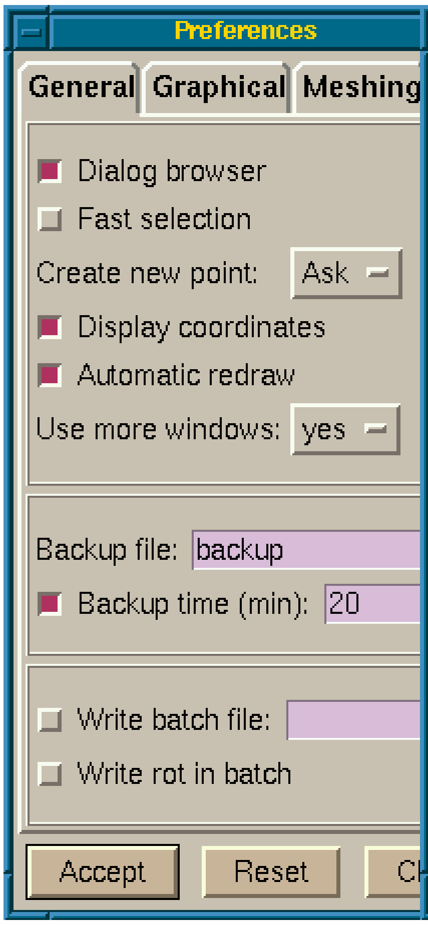
In the layer window, there exists the option `select` that allowing the selection of several meshes to perform over them an operation like: `on`, `off`, `color` and so on. This can be useful when using a large number of layers. After pressing `select` a new window appears to allow the input of a pattern that will match the layer's name. In this pattern, characters ***** and **?** are wildcards that match any characters or character, respectively, in the layer's name. So, the pattern `select` will match `select`, `selection`, `select-surface` and so on.

Important: The mesh generation does not depend on the state of the layers when the generation is performed (see section Layers). All layers are meshed and every node and element will be assigned to the layer where the original geometrical entity was.

UTILITIES

Within `Utilities` there are included some commands that provide information or allow actions over both the geometry and the mesh. Others can act over the whole project.

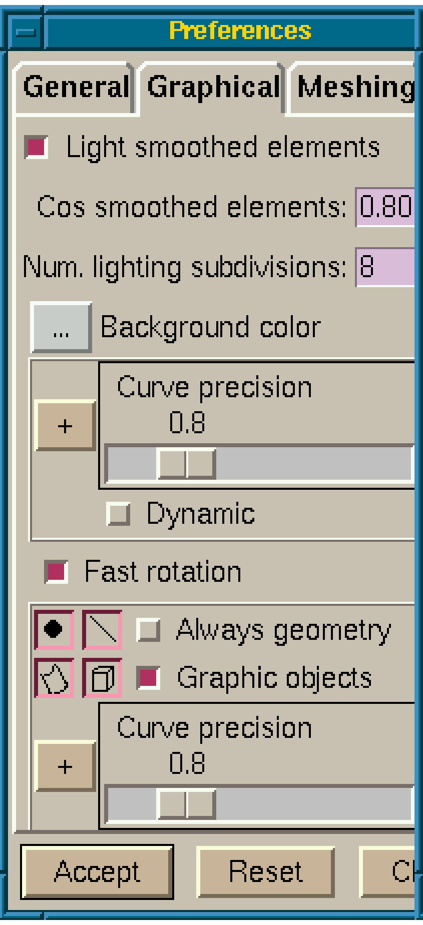
Preferences



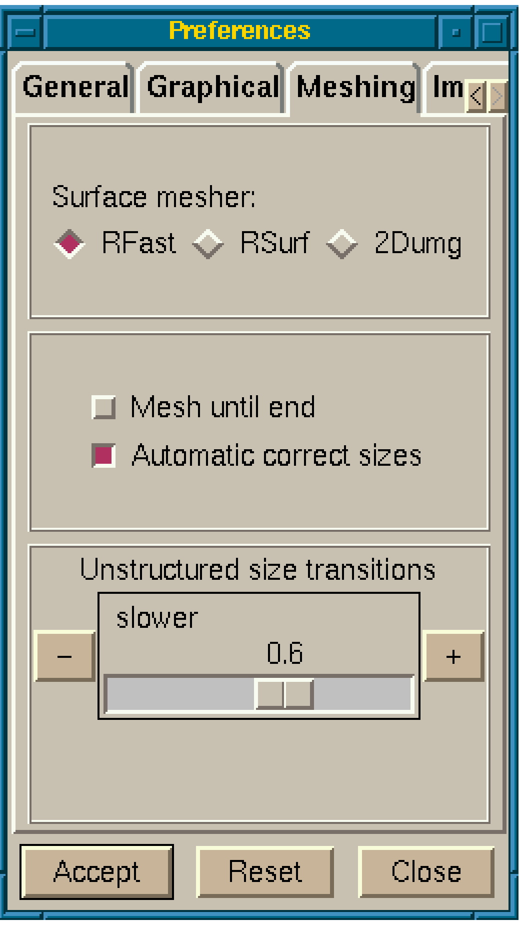
[✓ Dialog browser]
Fast selection
Create new point: Ask
[✓ Display coordinates]
[✓ Automatic redraw]
Use more windows: yes

Backup file: backup
[✓ Backup time (min):] 20

Write batch file:
Write rot in batch



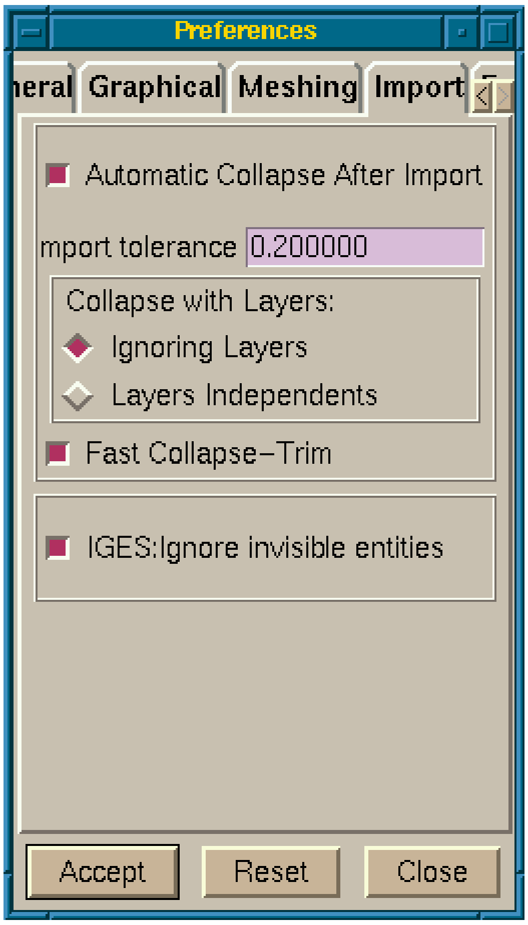
[✓ Light smoothed elements]
Cos smoothed elements: 0.80
Num. lighting subdivisions: 8
... Background color
+ Curve precision 0.8
Dynamic
[✓ Fast rotation]
Always geometry
[✓ Graphic objects]
+ Curve precision 0.8



Surface mesher:
[✓ RFast] RSurf 2Dumg

Mesh until end
[✓ Automatic correct sizes]

Unstructured size transitions
– slower 0.6 +



■ Automatic Collapse After Import

mport tolerance 0.200000

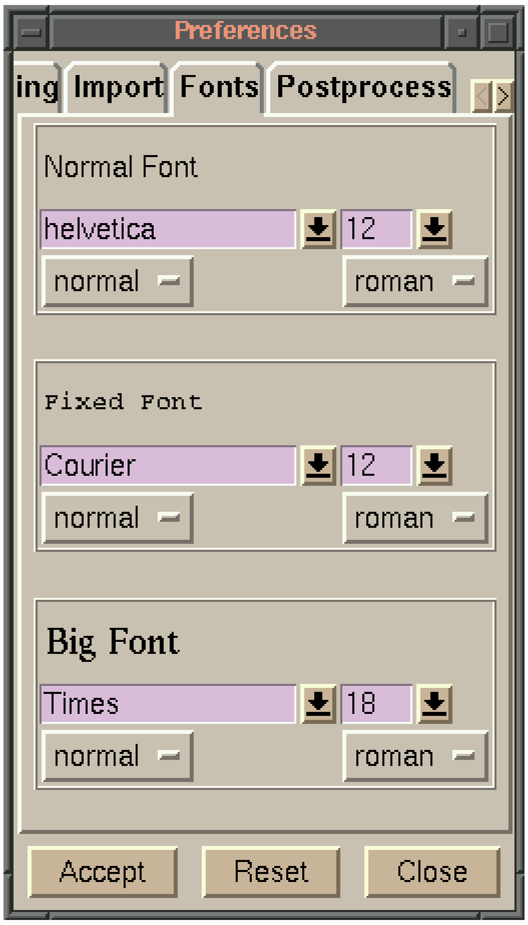
Collapse with Layers:

◆ Ignoring Layers

◇ Layers Independents

■ Fast Collapse–Trim

■ IGES:Ignore invisible entities



Normal Font

helvetica 12

normal roman

Fixed Font

Courier 12

normal roman

Big Font

Times 18

normal roman

Preferences window

There are some pre-definitions or ways of working that can be set in GiD. They can be set in two ways: via the Preferences window or with the `Preferences` command. In the following description, the different Preferences options are shown, giving their associated variables.

The first group of Preferences are used to set different ways of working with GiD:

- `Dialog browser`: If set, file names are required from the user with a browser window. If not, file names are required in the command line. Variable: `DialogBrowser`. Values: 1,0. Default is 1 (`Yes`).
- `Fast Selection`: If set, `fast selection mode` is set. If not, `normal selection mode` is set (see section Entities selection). Variable: `FastSelection`. Values: 1,0. Default is 0 (`No`).
- `Create always new point`: It alters the way how the points are entered in GiD (see section Point definition). Options are:
    - `Always`: If trying to create a new point in the vicinity of an existing one,

the new point is always created.

  - `Ask:` If trying to create a new point in the vicinity of an existing one, GiD asks the user whether to make use of the existing point or create a new one.
  - `Never:` Only allows to select existing points. It can be changed interactively when in creation point mode by setting `No join` until all points are entered.

  Variable: `CreateAlwaysNewPoint`. Respective values: 1,0,-2. Default is 0 (`Ask`).

- `Display coordinates:` If set, GiD shows the coordinates in the graphical window. Variable: `DisplayCoordinates`. Values: 1,0. Default is 1 (`Yes`).
- `Automatic redraw:` If this option is not set, redraws caused by window expositions, or by internal functions are not performed. This avoids spending a lot of time for large models. Variable: `AutomaticRedraw`. Values: 1,0. Default is 1 (`Yes`).
- `Use more windows:` Depending on the selected field, `yes`, `no` or `beginner`, some options are asked to the user inside a new window or with a message in the regular messages window. Variable: `UseMorewindows`. Values: 1,0,2. Default is 1 (`Yes`).
- `Backup file:` If this option is set and a model name is given, model is saved automatically into that model name with the frequency given in the next option. **Note:** Mesh is not saved into the backup to save time. Variable: `BackUpName`. Default name is `backup.gid` in current or temporal directory.
- `Backup time:` Number of minutes between each backup save. 0 minutes means not to save backup. Variable: `BackUpMinutes`. Default is 20 minutes.
- `Write batchfile:` If this option is set, and a filename is given, all commands used during the current session are saved into this file. It can be executed later by means of a script file with the predefined commands to be run in GiD (see section Batch file). Variable: `BatchFileToWrite`.
- `Write rotations in batch:` If this option is set, all dynamic rotations and movements are stored in the batch file. This permits to see these movements when reading this batch file with See section Read batch window. Variable: `WriteRotationsInBatch`.

The second group of preferences are used to set different ways of visualizing the model. They do not change the geometry and the model information. These preferences are:

- `Light Smoothed Elements:` If set and when rendering a mesh (see section

Render), the intersection between elements with a small angle between their normals will be illuminated as if it were a continuous solid. If not, illumination is made considering every element as planar. Variable: `LightSmoothedElems`. Values: 1,0. Default is 1 (`Yes`).

- `Number of lighting subdivisions`: When rendering the geometry (see section Render), this number sets the quality of the illumination. More divisions implies more quality but slower drawing. Variable: `NumOfLightingSubdivisions`. Default is 8.

- `cos smoothed elements`: If `Light Smoothed Elements` is set, this is the value of the cosine of the angle between the normals of the two elements. If the actual angle is less than this value, the intersection is smoothed, if not, an edge is drawn. Variable: `CosSmoothedElems`. Default is 0.8.

- `Background color`: User can choose with this option the background color of the GiD graphical window. Variable: `BackgroundColor`. Value: hexadecimal RGB char.

- `Curve precision`: This option gives the precision used to draw curves. The internal definition of curves does not change. In the preferences window, it is possible to dynamically change the drawing of a curve to test the precision. If it is set to 1.0, the drawing of isoparametric lines in u,v=0.5 are activated. Variable: `CurvePrecision`. Values: 1.0 to 0.0 from best to worst. Default is 0.8

- `Fast rotation`: If set, some options are able to be chosen. These options will apply only when rotating. Options are:

    - `points,lines,surfaces,volumes`: To draw or not to draw this type of entity when rotating. Variables: the same. Values: 0,1. Default is 0 (`No`).

    - `Always Geometry`: If set, when viewing the mesh and rotating, the geometry is drawn instead. Variable: `UseAlwaysGeom`. Values: 0,1. Default is 0 (`No`).

    - `Draw graphic objects`: If not set, when rotating the geometry, some graphical and temporal objects like normals or materials or conditions symbols are not drawn. Variable: `DrawGraphicObjects`. Values: 0,1. Default is 1 (`Yes`).

    - `Curve precision`: The same as general item `Curve precision`, but applied only when rotating. Variable: `CurvePrecision`. Values: 1.0 to 0.0 from best to worst. Default is 0.8

  Variable: `FastRotation`. Value: 0,1. Default is 0 (`No`).

The third group of preferences are meshing options:

Surface meshers can be:

- `Rfast` The most efficient in speed and reliability. With deformed surfaces can give distorted elements.
- `Rsurf` Meshes are generated directly in the space. Quality is better but it is slower and can fail for distorted surfaces.
- `2dumg` This is the only one that can generate non structured quadrilateral meshes. When trying to mesh non-structured quadrilaterals with another mesher, GiD internaly changes to it.

These generators are based on the advancing front generation mesh technique in order to improve speed and portability.

Note: GiD can internally try another mesher when one of them fails to generate the mesh for one surface.

- `Mesh until End` If this preference is set, the mesh generator will continue until end although there are surfaces or volumes that cannot be meshed.
- `Automatic correct sizes` If this preference is set, just when meshing begins, there is a correction to the meshing sizes assigned to entities so as to improve meshing.
- `Mesh transitions:` It controls whether the transitions between different element sizes are slow or fast.

The fourth group of preferences are the geometry import and collapse options:

- `Automatic Collapse After Import:` If this option is set, after reading one IGES file, one global collapse is made. If not, every surface and line will be independent from each other. Variable: `AutoCollapseAfterImport`. Default is active (1).
- `Import tolerance:` When importing a file or collapsing, the points closer than this distance are considered to be unique (see section IGES read). Lines and surfaces can also be collapsed. Variable: `ImportTolerance`. Default is 0.0001.
- `Collapse with Layers:` if set to `Ignoring Layers:` entities are collapse also if they belong to diferent layers. If set to `Layers Independents:` entities in different layers are not collapsed. (Entities belonging to frozen layers are never considered). Variable: `CollapseIgnoringLayers`. Value is 1.
- `Fast Collapse-Trim:` If this option is activated, when one collapse is made, lines with just two surfaces over them are ignored because they are usually

correct. Variable: `CollapseHigherTwo`. Value is 1.

- `IGES:Ignore invisible entities:` If this option is set, entities with 'invisible' flag in an IGES file are ignored. If it is not set, all these entities are sent to a layer with name: "Invisible". Variable: `IGESIgnoreInvisible`. Value is 1.

The fifth group of preferences are the fonts used inside GiD:

- `Normal font:` It is the font normally used inside GiD.
- `Fixed font:` This font must have the same spacing for every letter. It is used in places where this property is necessary. `Big font:` Used in some dialog boxes.

The sixth group of preferences (those corresponding to the post-processing) are explained in the facilities description of the post-processing chapter (see section POST-PROCESSING PREFERENCES).

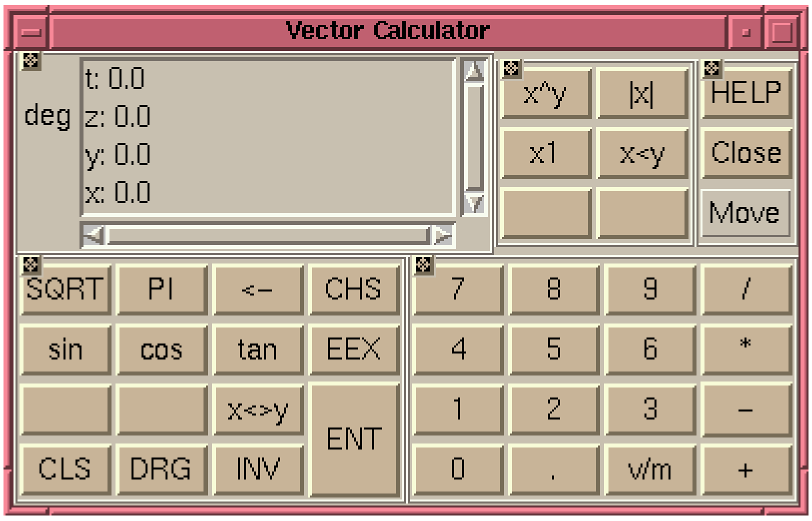
Renumber

When creating new entities, the label of the new entity will be the lowest number greater than 0 that still does not exist for this entity type. If an entity is deleted, a gap is left in the label's list. This gap would be filled with new entities but it is possible to renumber the geometry to change the former entity labels. There are no problems with materials and conditions applied to entities.

In **geometry** mode, the renumbered entities are the geometrical ones. In **mesh** mode, the renumbered entities are the meshed ones. In this case, renumbering not only fills the gaps in the labels list but also changes the node numbers so as to minimize the difference of node numbers within each element. This can be useful when the calculating module uses band or skyline storage methods.

Note: It is usually not necessary to use this command because it is automatically applied when generating the mesh.

Calculator



Vectorial calculator

This option opens a scalar and vectorial calculator. It has its own help. Try button 2 or 3 over one button.

It's possible to transfer scalar and vector and points and distances from and to the calculator and the main graphical display.

Id

This command gives the label and coordinates of an existing or new point.

Options are:

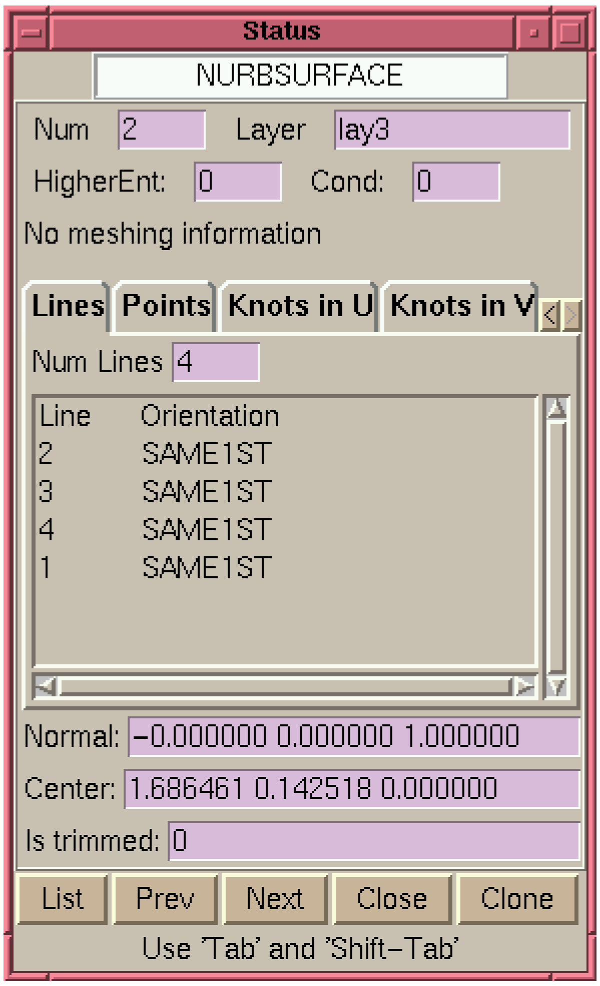
- `Line parameter:` Selecting a line and given the line parameter (t) from 0.0 to 1.0, the point belonging to that line and t is returned.
- `Surface parameter:` Selecting a surface and given the surface parameters (u,v) from 0.0 to 1.0, the point belonging to that surface and u,v is returned.

Signal

With this option, the user can select one entity (point,line surface or volume), and a pair of crossed lines in the graphical window signal the center of the entity.

They must be existing entities, except the especial case of point or nodes, where they can be existing or defined with any of the usual methods.

List



Status

NURBSURFACE

Num: 2 Layer: lay3

HigherEnt: 0 Cond: 0

No meshing information

Num Lines: 4

```
Line    Orientation
2       SAME1ST
3       SAME1ST
4       SAME1ST
1       SAME1ST
```

Normal: -0.000000 0.000000 1.000000

Center: 1.686461 0.142518 0.000000

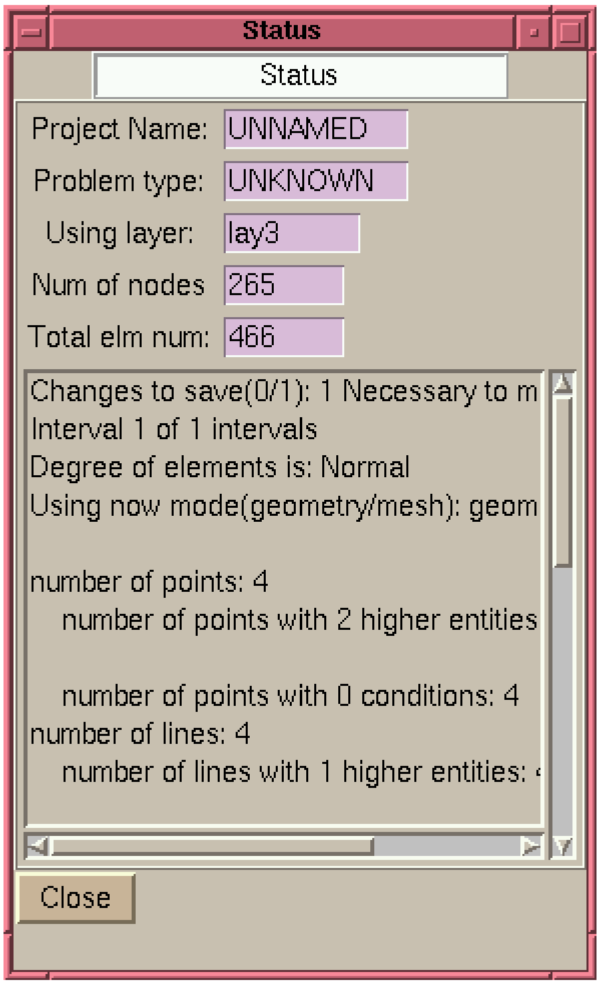
Is trimmed: 0

Use 'Tab' and 'Shift–Tab'

List entities window

Command `List` gives information about the selected entities. This information is read only.

Status



Project Name: UNNAMED
Problem type: UNKNOWN
Using layer: lay3
Num of nodes 265
Total elm num: 466

```
Changes to save(0/1): 1 Necessary to m
Interval 1 of 1 intervals
Degree of elements is: Normal
Using now mode(geometry/mesh): geom

number of points: 4
   number of points with 2 higher entities

   number of points with 0 conditions: 4
number of lines: 4
   number of lines with 1 higher entities:
```

Project status window

Option `Status` gives information about useful general data of the project.

Distance

Command `Distance` gives the distance between two existing or new points.

Draw line normals

Command `DrawNormals. Lines` draws the sense of the selected lines.

If the line lies on the plane `z=0`, apart from the sense, GiD displays the normal of the line in 2D.

Viewing commands (zoom, rotation, etc.) can be applied and the normals remain on the screen.

It is possible to swap the sense of the polylines or NURBS lines when they do not belong to higher entities.

Options are:

- `Swap some` changes the sense of the selected lines.
- `All swaped` changes the sense of all lines asking confirmation for the change.

Draw surface normals

Command `DrawNormals. Surfaces` draws the normals of the selected surfaces.

Surfaces belonging to the plane `z=0` will have, by default, its normal oriented towards the z positive. Then, they are defined as anti-clockwise surfaces in 2D.

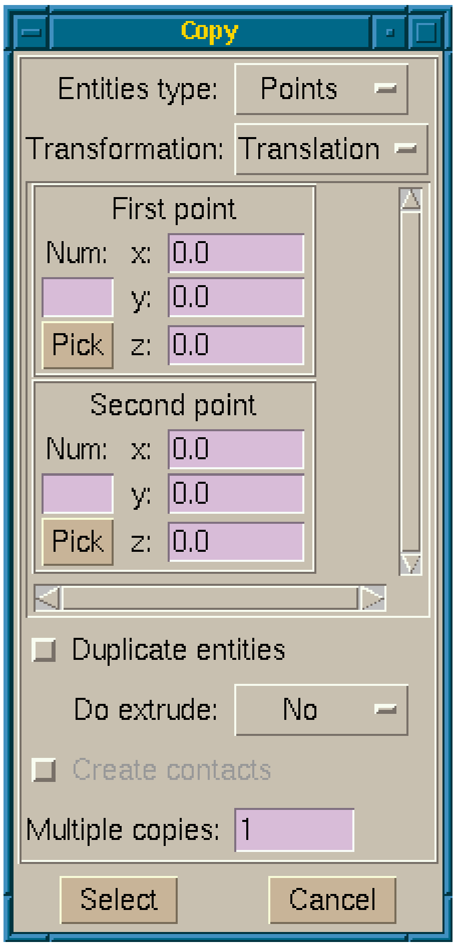
It is possible to swap the sense of the normals when the corresponding surfaces do not belong to higher entities.

Options are:

- `Swap some:` changes the sense of the normals of the surfaces after selecting them again.
- `All swaped:` changes the sense of all selected surfaces, asking for confirmation for the change.
- `Swap group:` changes normals of all selected surfaces or elements so as all of them have its normal to the same sense that adjacent entities. This sense is arbitrary but common to all adjacent entities.
- `Sel by normal:` user is prompt to enter a vector and selection is reduced to entities that have their normal in the same sense (dot product positive) as the given normal.
- `Color:` With this option set, all surfaces are draw with filled color. Their front side will be drawn with their regular color, and their back face with yellow.

Note: Volumes are correctly oriented by GiD, regardless of their surface orientation.

Copy



Entities type: Points
Transformation: Translation
First point
Num: x: 0.0
y: 0.0
z: 0.0
Second point
Num: x: 0.0
y: 0.0
z: 0.0
Duplicate entities
Do extrude: No
Create contacts
Multiple copies: 1

Copy window

`Copy` is a general function that allows the user select a group of entities and copy them with a movement that can be `translation`, `rotation`, `mirror` or `scale`. The process is:

Select the type of entities to copy: `point`, `line`, `surface` or `volume`, when being in **geometry** mode, or option `nodes` or `elements`, when being in **mesh** mode. All the lower entities that belong to the selected ones will be automatically computed. the type of movement needs to be chosen and the parameters for the movement are defined. Options are:

- **rotation:** It is necessary to either enter two points in 3D or one point in 2D. These two points define the rotation axis and its orientation. In 2D, the axis goes from the defined point towards z positive. Enter the angle of rotation in degrees. It can be positive or negative. The sense is defined by the right hand rule. In 2D, the sense is counter-clockwise.
- **translation:** It is defined by two points. Relative movements can be obtained defining the first point as 0,0,0 and considering the second point as the translation vector (see section Point definition).

- **mirror:** It is defined by three points that cannot be aligned. These points form a plane that is the mirror plane. In 2D, the mirror line is defined by two points.
- **scale:** It is defined by a center and a point. Every coordinate of the point is the scale factor for every `x,y,z` axis. Scale factor greater than one, increases size, while less than one decreases the size. It may also be negative, changing the sign of the corresponding coordinates.
- **offset:** It is defined by one positive or negative scalar magnitude. It will move each entity in the sense of its normal, the magnitude given. In 2D, normal is considered to lie in the `z=0` plane. This option only works for mesh elements.

Other available options are:

- **extrude:** This option can be set to either lines or surfaces. When a movement is selected, the copy is made and lines connecting the old and the new points are created. These lines will be either straight lines or arcs depending on the movement type. If extrude surface is chosen, NURBS surfaces connecting old and new lines will also be created. This option is not allowed when copying volumes.
- **mcopy:** Selecting this option and giving the number of repetitions, the operation selected is performed this number of times. This option is not available for `mirror`.
- **create contacts:** Creates separated contact volumes (see section Contact creation) for every copied surface. This option is only available when copying surfaces.
- **Duplicate entities:** If this option is not set, when an entity after the `copy` operation occupies the same position than an existing one that does not belong to a frozen layer, both entities are converted into one.

Note: Entities belonging to a frozen layer (see section Layers), are not checked when sharing old entities.

Move

This command works like the command `Copy` but moves the entities instead of copying them. The program checks automatically if some of the entities must be copied instead of moved (for this reason, these entities belong also to non-selected entities of higher level) and performs the corresponding operation.

Options like `Extrude`, `Multiple copy` and `Create contacts` are disabled for movements.

Repair

This option checks the coherence of the database information. Only use it if there are problems. When used, a window notifies repaired items and may give some warnings about incorrect entities.

Collapse

The function `Collapse` convert coincident entities or entities that are close each other into one.

The variable `ImportTolerance` (see section Preferences), controls the maximum distance between two adjacents points to be converted into one. With lines and surfaces, a maximum distance between both entities is calculated and when it is less than `ImportTolerance`, they are converted into one.

Select the type of entities: `point`, `line`, `surface` or `volume`, when being in **geometry** mode, or option `nodes` or `elements`, when being in **mesh** mode. All the lower entities that belong to the selected ones will be automatically computed. After pressing `escape`, collapse is performed.

Option `Model` performs the operation over all the entities of the model.

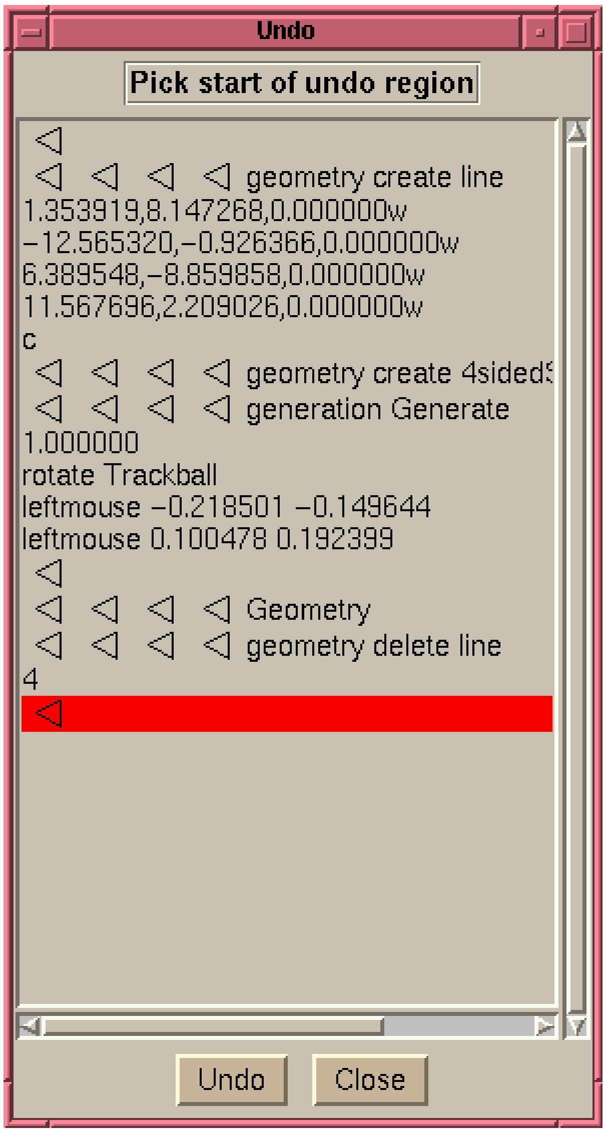
Note: Entities belonging to a frozen layer (see section Layers), are not checked when collapsing.

Uncollapse

Function uncollapse permits the user to select either lines or surfaces and duplicate boundary entities.

Typically, if two surfaces share one line as an edge, after applying this function to both surfaces, that line will be duplicated and every line will belong to a different surface.

Undo



```
◁
◁ ◁ ◁ ◁ geometry create line
1.353919,8.147268,0.000000w
-12.565320,-0.926366,0.000000w
6.389548,-8.859858,0.000000w
11.567696,2.209026,0.000000w
c
◁ ◁ ◁ ◁ geometry create 4sided
◁ ◁ ◁ ◁ generation Generate
1.000000
rotate Trackball
leftmouse -0.218501 -0.149644
leftmouse 0.100478 0.192399
◁
◁ ◁ ◁ ◁ Geometry
◁ ◁ ◁ ◁ geometry delete line
4
◁
```

Undo window

This command allows the user undo any previous commands executed since the last time the project was saved or read. To do this, the user must select in the undo window the set of commands that have to be undone.

Comments

This facility is included to add comments to the pictures obtained through GiD (see section Print to file). Three lines are allowed and with the `Comments` button the user can impose their printing on the screen to foresee mistakes or perform changes before printing.

All the comment lines have their corresponding input button and the literal text, written into the command line between double-quotes, will be the output.

Graphical

In this menu there are some options to change the windows appearance. They can be inside another window or independent or disappear. Windows are:

- **Graph menu:** This is the icons menu used to perform certain operations. Use middle or right button over an icon to get help. This option makes the window appear.
- **Right buttons:** These are the buttons that appear usually in the right side of the windows. They can perform most of the actions of the program.
- **Command line:** Inferior line where it is possible to write commands.
- **Up menu:** Menus located over the graphical window.

Note: To make the `Up menu` appear again, shortcut `Control-U` or `Control-Shift-u` can be used.

Coordinates window

This option opens a window used to enter points (see section Point definition). It can be used in any place where it is possible to enter one point.

To accept one point in this window, press `Accept` or press the `Return` key.

Option `Coordinate system` allows the user select between:

- **Cartesian:** Cartesian coordinate system.
- **Cylindrical:** User enters the radius in XY plane, the anti-clockwise angle in plane XY from x axe(theta), and the z coordinate.
- **Spherical:** User enters the radius, the anti-clockwise angle in plane XY from x axe(theta), and the angle with z axe(phi).

Option `Local axes` allows the user choose:

- **Global:** Global axes.
- **Relative center:** User defines a center of coordinates. All new points created with this window will be related to this center. In the window where the relative center is entered, user can select with button `Pick` a point from graphical window.

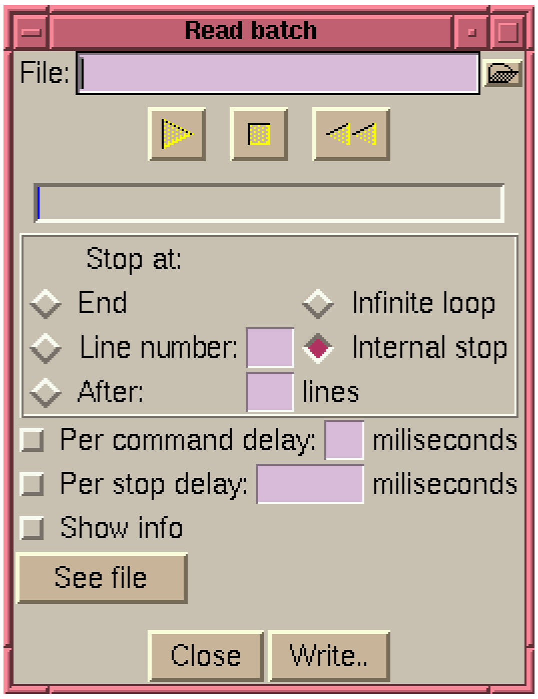
- **Relative last:** Last point entered is the relative center of coordinates.
- **Define new...:** It allows the user define a new local axes. Once defined, user can select it.
- **Any local axes:** All local axes defined with See section Local axes will be listed here. Points entered will be related to this axes.

`New point` shows the current way of entering points. Button `Change` opens the preferences window (see section Preferences) where it can be changed.

Button `Pick` allows the user select a point from the window and it is inserted in the point fields. Then it can be edited and used.

Read batch window

A batch file can be read to execute some functions (see section Batch file) or to create an animated view of these operations. This latter case can be performed with the `Read batch window`.


Read batch

File:

Stop at:

End [ ] Infinite loop [ ]

Line number: [ ] Internal stop [✓]

After: [ ] lines

Per command delay: milliseconds [ ]

Per stop delay: milliseconds [ ]

Show info [ ]

See file

Close Write..

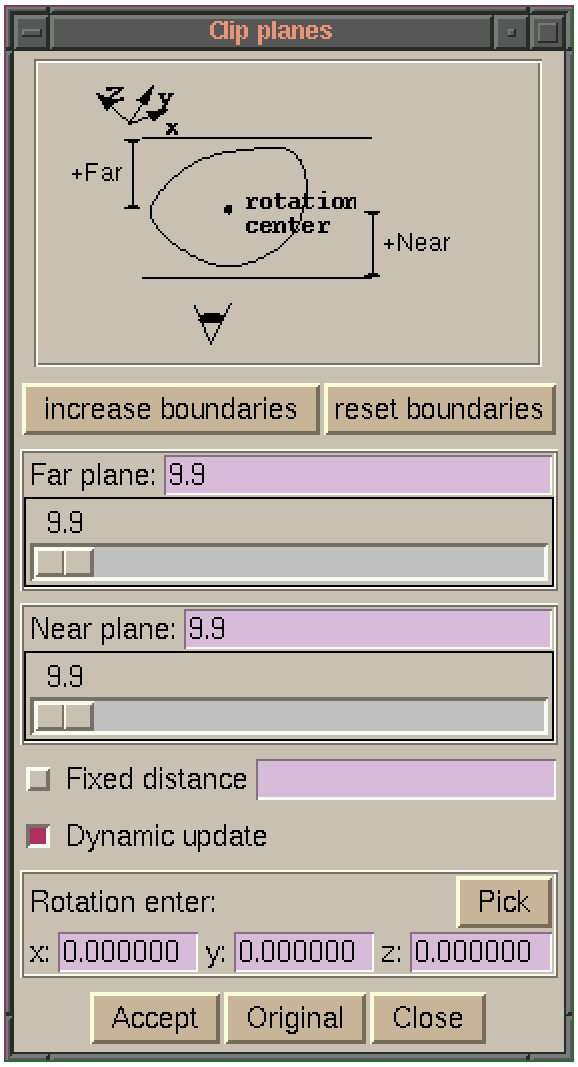
Read batch window

Once selected a file, it is possible to execute it interactively and make it stop in some interesting parts. To allow all the movements (rotations and so on) be executed in the same way that originally, option `Write rotations in batch:` must be set in preferences (see section Preferences), when creating the batch.

There are several ways to stop the running batch. One of them is to include stops in the show file section with the `Mark break` button, and selecting `Internal stop`. This marks can be saved in the batch file with button `Write...`.

Option `Show info` lets the program write all the usual messages in the GiD messages window.

Clip planes



Far plane: 9.9

9.9

Near plane: 9.9

9.9

☐ Fixed distance

☑ Dynamic update

Rotation enter:

x: 0.000000 y: 0.000000 z: 0.000000

This window allows to hide the front or the back of the view.

Clip planes is a way not to draw the geometry or mesh that is very close or very far to the viewer.

Moving `Near plane`, hides the geometry that is closer to the viewer. Moving `Far plane`, hides the geometry that is further to the viewer.

To see the changes dynamically, select `Dynamic update`. Option `Fixed distance`, allows a constant distance between both planes.

A new rotation center can be chosen or selected from the window.

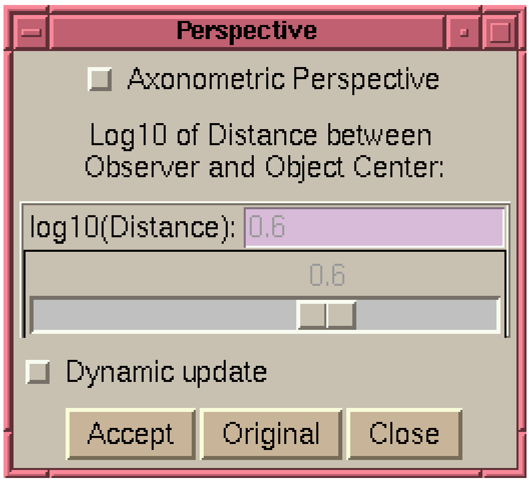
Option `Increase boundaries`, increments the ends in the bars of Far plane and Near plane.

Note: All this information is reset when performing a `zoom frame`. (see section Zoom).

Save configuration file

It is possible to save a window configuration into a file. Then if GiD is open again with option -c (see section INVOKING GiD) and this file, the windows are open in the same place and size.

Perspective



Perspective window

By default, a model is viewed inside GiD using orthogonal perspective.

With this option, it is possible to change to axonometric perspective. In this mode, the user can choose a distortion factor for the perspective. This can be updated dynamically.

Change Light Vector

With this option the user can change the **VECTOR** of the light direction interactively or just entering the **VECTOR** components on the **command line**.

Options are:

- `Default Vector`: with this option the ligth direction vector changes to its default value.
- `Change Z Sign`: here the user tells the program if the light comes from the front of the object, or from behind the object. The mouse cursor are changed arcondingly.

CALCULATE

Inside this window, the user can begin and manage the analysis of the problem. Word `process` will be used to refer the running analysis code. This word is used to show the possibility that several analysis or processes can be run at the same time.

Options for this window are:

- `Start`: Begins the process module. Once it is pressed, user can continue working with GiD as usual.
- `Kill`: After selecting a running process, this button stop its execution.
- `Output view`: After selecting a running process, this button opens a window that shows process related information such as iterations, convergence, etc.
- `Close`: Closes the window but does not stop the running processes.

A list of all the running processes is shown, with some useful information like name, starting time, etc.

The various windows will differ in accordance to the specifications of the problem (thermoelastic, impact, metal-forming, etc.), the different types of elements (2D or 3D, beams, shells, plates, etc.) and also the different requirements and specifications of the particular solver.

All the commands and facilities that are explained in this manual are generally available for the different solvers. However, there may be some commands and facilities that are only available to some of them and therefore some displays can look slightly different from those explained on these pages.

POST-PROCESSING BASICS

This chapter describes some relevant aspects of the Postprocessing step and the way to load results from a numerical analysis into GiD.

Post-processing Introduction

In the same way than GiD was able to create a geometry with its problem definition, independently of the solver module that was going to be used, GiD is also capable to read different results achieved by a particular solver, visualize the results in a universal way (i.e. contours, vectors, nodal values, cut planes, etc.) and handle sets of different kinds of elements (lines, triangles, quadrilaterals, tetrahedra and hexahedra). GiD can combine all different element types created during the pre-processing. The users can also define several analysis types like load analysis, time analysis, frequency analysis as well as their own customized analysis.

Stepping into Post-processing

This section tries to explain how post-processing information can be load into GiD.

There is no need to load a project into GiD to use its Post-processing.

Once GiD is started to solve a particular project, this project is meshed and calculated, when the option `Post-process` is selected, GiD uses the meshes generated inside GiD and only tries to read `ProjectName.flavia.res` located inside the project directory.

If there was no project loaded into GiD, or, if once in `Post-processing` it is necessary to load another post-processing information, the option `read` can be used to load this information. A `File Open Dialog Box` is shown in order to select a file. The reading process that follows this option is this:

1. First it tries to read `file_name.msh`: Volume mesh information
2. Then it tries to read `file_name.bon`: 3D Surface mesh information
3. If both of above fail, then GiD will try to read `file_name.dat`: 2D Surface mesh information
4. And, at last, it tries to read `file_name.res`: Results information

The format of these files is explained later see section Post-processing data files.

All the Post-processing features can be grouped into three categories:

- **General options**: some common options between pre- and post-processing sections of GiD.

- **Geometry options**: options that does not require numerical results to use them.
- **Results options**: options that require numerical results to use them.

These categories will be detailed in the following chapters.

POST-PROCESSING GENERAL OPTIONS

Once inside the post-processing section of GiD, almost all the visualization features and management options of the pre-processing section are available: Zoom, Rotate (Rotate screen/object axes, Rotate trackball,etc.), Pan, Redraw, Render, Label, Clip Planes, Perspective, etc.

With the `Fast-rotation` option on the `Preferences` Window (see section Preferences), GiD will use the boundaries visualization (see section Display Style), to rotate the model excluding any result. Checking on the `Always Geometry` check box, when rotating, GiD will use the pre-processing geometry information.

Inside the `Utilities` menu, the options `Variables`, `Id`, `Dist`, `Signal Entities`, `Comments`, `Print to file` have the same functionality as in Pre-processing (see section UTILITIES, see section Id, see section Distance, see section Signal, see section Comments, see section Print to file respectively), but the option `Render` adds a new feature only for the Post-processing step. Their values are:

- `Normal:` This is the usual way of visualizing. The meshes and results are viewed as they are.
- `Flat Render:` Solid model with flat illumination and lines.
- `Smooth Render:` Solid model with smooth illumination (better quality).
- `Transparent/Opaque:` Meshes are visualized with transparency (or not) so that iso surfaces inside them can be viewed easily.
- `Swap Mesh/Set/Cut Normals:` If, in render mode, some meshes appear black, with some color on boundaries, then with this option this problem will be corrected.
- `Culling:` Here the user can choose wether the `Front Faces`, `Back Faces`, `Front and Back Faces` are culled, i. e. not drawn, or `No Faces` are culled. This option is useful to llok into volume meshes, etc.
- `ChangeLightVec:` With this option, the user can change the light vector direction interactively (see section Change Light Vector).

Render options `Normal`, `Flat Render` and `Smooth Render`, display option `Transparent/Opaque` and `Culling` options can also be used from the window `View Style`.

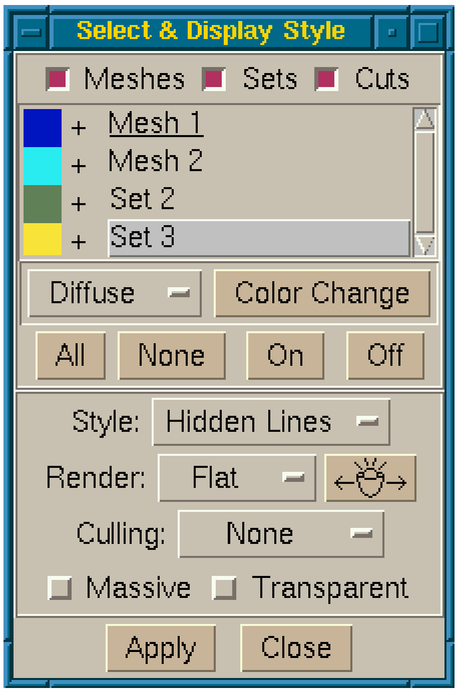
With the `Label` option, it is not only possible to see the node number, or element number, but also the nodal results values, when a particular result is visualized.

The `Pre-process` option allows the user, after a confirmation, to return to the pre-processing section of GiD.

POST-PROCESSING GEOMETRY OPTIONS

Here those post-processing options that do not involve directly display of numerical results.

There is a `View Style` window that contains the first two of them: select meshes/sets/cuts and display style.



Display Style Window

Selecting Meshes, Sets and/or Cuts, the user can switch them `On` and `Off` after pressing the `Accept` button. Also, clicking on a Mesh, Set or Cut, and pressing `Color Change` the user can change the `Ambient`, `Diffuse`, `Specular` and `Shininess` component color of the selected set, or give it its `Default` color back. The `Diffuse` component is used for all the representation, while the rest (`Ambient`, `Specular` and `Shininess`) has more sense for `Render` visualization. The button on the left of the `Render` option allows the user to change the light vector.

Near the `Culling` label, the user can choose wether the `Front Faces`, `Back Faces`, `Front and Back Faces` are culled, i. e. not drawn, or `No Faces` are culled. This option is useful to llok into volume meshes, etc.

The `Massive` option allows the user to see the elements that are inside a volume mesh, and to draw all the vectors inside of a mesh/set/cut when a `Boundaries` display style is used.

With the `Transparent` option the meshes/set/cuts will be drawn transparently or opaque.

Select Meshes/Sets/Cuts

With this options the user can switch on and off meshes, sets and cuts. Each mesh/set/cut appears colored in the window. Just clicking on them, and by using button `On` / `Off` and `Accept` the user can see how meshes, sets and/or cuts appear or disappear.

Display Style

Here the user can choose how meshes, sets and cuts should be drawn. These options are:

- **Boundaries:** All the edges of the boundaries and surface elements will be displayed, in black. An edge is the line that only belongs to one surface element, or the one that belonging to two surface elements, its faces form an angle less than a predefined one. This angle can be specified by the user in (see section POST-PROCESSING PREFERENCES) or `Results Options BorderAngle` on the `Right Buttons` menu.
- **Hidden Boundaries:** The same as before, but here the edges that are behind the meshes, sets and cuts are removed.
- **All Lines:** All the lines of surface elements are drawn. When drawing volume meshes, GiD only draws the surface/boundaries elements of this volume mesh. How GiD can draw these interior elements is explained below. Each mesh, set and cut is drawn with its own colors.
- **Hidden Lines:** The same as before, but here the lines that are behind the meshes, sets and cuts are removed.
- **Body:** The elements of meshes, sets and cuts, are drawn in filled mode, i.e. they are drawn as solid elements. Each mesh, set and cut is drawn with its

own colors. It can be very hard to recognize the shape of the meshes if no illumination is actived (see section POST-PROCESSING GENERAL OPTIONS).

- **Body Boundaries:** Same as before, but with `Hidden Boundaries` drawn too.
- **Body Lines:** Same as before, but with `Hidden Lines` drawn too.

Do Cuts

Here the user can cut and divide meshes, sets and cuts. A cut of a volume mesh results into a cut plane. The cut is done for all the meshes, despite that some of them are switched off. When cutting sets, a line set will be created. Here only those sets that are switched on are cut.

Another feature is that a cut can be deformed, if meshes are also told to do so (see section Deform Mesh). A cut of a deformed mesh, when changing to the original shape, will be deformed accordingly.

When dividing meshes/sets only those elements of the meshes/set switched on that lie on one side of a specified plane will be used to create another mesh/sets.

- **Cut Plane:** the user specifies a plane which is used to cut the meshes/sets. Several options can be used to enter this plane. With `Two Points` (the default), the plane is defined by the corresponding line and the orthogonal direction to the screen. With `Three Points`, the plane is defined by this three points. When choosing the points, the nodes of the mesh can also be used. All the volume meshes are cut, whether they are drawn or not.
- **Divide Mesh:** the user specifies a plane which is used to divide the meshes. Several options can be used to enter this plane. With `Two Points` (the default), the plane is defined by the corresponding line and the orthogonal direction to the screen. With `Three Points`, the plane is defined by these three points. When choosing the points, the nodes of the mesh can also be used. After defining the plane, the user should choose which section of the mesh should be saved by select the side of the plane to use. Only those elements that lie entirely on this side are selected. Only those volume meshes that are shown are divided.
- **Divide Sets:** the user specifies a plane which is used to divide the sets. Several options can be used to enter this plane. With `Two Points` (the default), the plane is defined by the corresponding line and the orthogonal direction to the screen. With `Three Points`, the plane is defined by these three points. When choosing the points, the nodes of the mesh can also be used. After defining the plane, the user should choose which section of the mesh

should be saved by select the side of the plane to use. Only those elements that lie entirely on this side are selected. Only those surface meshes that are shown are divided.

- **Succession:** This option is an enhancement of `Cut Plane`. Here the user specifies an axis that will be used to create cut planes orthogonal to this axis. The number of planes is also asked for.
- **Cut Wire:** Here the user can define a 'wire' tied to the edges of the elements of the meshes/sets. So when the meshes/sets are deformed, the wire is deformed too. And viceversa, if a wire is defined while the meshes/sets are deformed, when turning them into its original shape, the cut-wire is "undeformed" with them.

Cuts can also be read and write from/to a file. The information stored into the file from a 'Cut Plane' is the plane equation that defines the plane (Ax + By + Cz + D = 0), so it can be used with several models. The information stored into the file from a 'Cut Wire' is the points list of the cut-wire, i.e. the interseccion between the wire and the edges of the meshes/sets. These files are standard ASCII files. A line of a 'Cut Plane' archive contains the four coeficients of the cut defining plane separated by spaces. A line of a 'Cut Wire' archive contains the three coordinates of a point of the wire. Comment lines are allowed and should begin with a '`#`'.

An example of a 'Cut Plane' archive were three planes are stored:

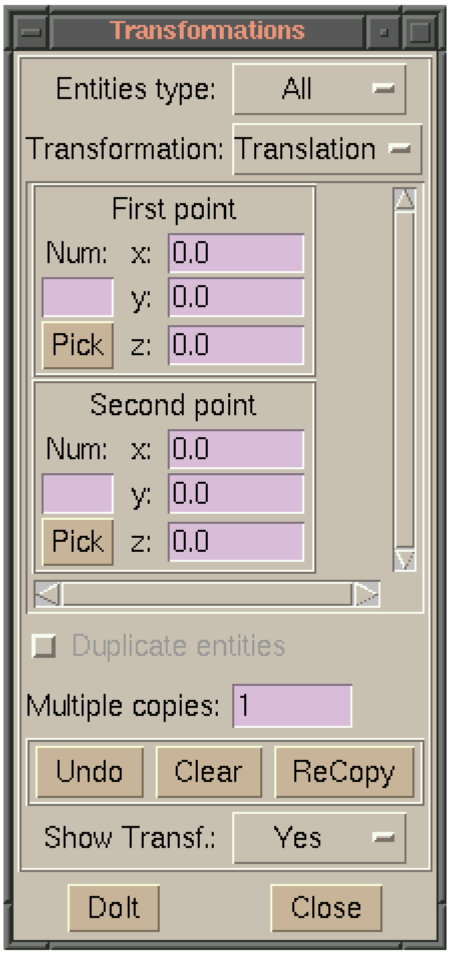
```
# planes created of a 'cut succesion'
-10.82439 0.5740206 0 51.62557
-10.82439 0.5740206 0 12.45994
-10.82439 0.5740206 0 -26.70569
```

An example of a 'Cut Wire' archive:

```
-2.444425 3.883427 2.487002
2.130787 2.762815 3.885021
0.8411534 4.458836 3.215301
4.270067 3.795048 2.037187
5.66561 3.414776 0.8219391
2.945865 3.600701 3.29012
0.4487007 3.764661 3.574121
```

Transformations

There is a `Transformations` window that allows the user to repeat the the visualization using translation, rotation and mirror transformations, so that GiD can drawn a whole model when only a part of it was analyzed:



Transformations Window

Transformations types are:

- **Translation:** It is defined by two points. Relative movements can be obtained defining the first point as 0,0,0 and considering the second point as the translation vector (see section Point definition).
- **Rotation:** It is necessary to either enter two points in 3D or one point in 2D. These two points define the rotation axis and its orientation. In 2D, the axis goes from the defined point towards z positive. Enter the angle of rotation in degrees. It can be positive or negative. The sense is defined by the right hand rule. In 2D, the sense is counter-clockwise.
- **Mirror:** It is defined by three points that cannot be aligned. These points form a plane that is the mirror plane. In 2D, the mirror line is defined by two points.

Available options are:

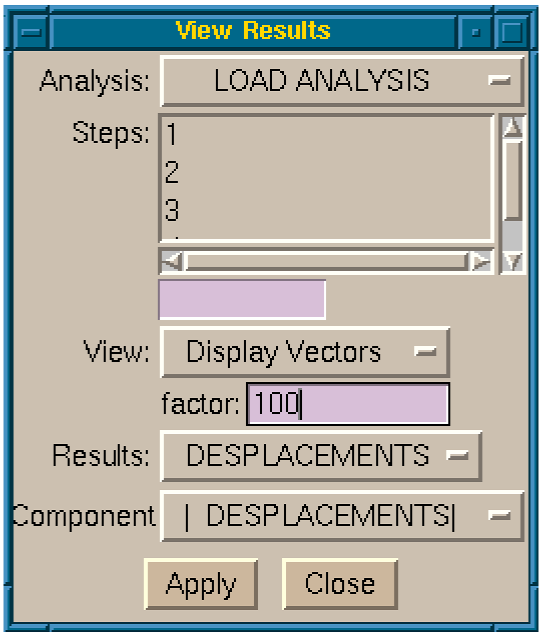
- **Multiple copies:** Selecting this option and giving the number of repetitions, the operation selected is performed this number of times.

- **Undo:** Deletes last transformation done. It can be used repeadly to clear all the transformations done.
- **Clear:** Clear all the transformations at once.
- **ReCopy:** Redo de last transformation with the new **Multiple copies:** value.
- **Show Transf.:** allows the user wether to se the original model, or the one with the transformations active.

Button **DoIt** tells the program to do the transformation selected.

POST-PROCESSING RESULTS OPTIONS

Here we'll describe those post-processing options that need results to use them. There is a window `View Results` that contains most of them: contour fill, contour lines, show minimum and maximum, display vector.


View Results

Analysis: LOAD ANALYSIS

Steps: 1 2 3

View: Display Vectors

factor: 100

Results: DESPLACEMENTS

Component | DESPLACEMENTS|

View Results Window

The possible results to be displayed can be grouped into three major categories:

- **Scalar view results:** Show Minimum & Maximum, Contour Fill, Contour Lines, Iso Surface and its configuration options.
- **Vector view results:** Mesh deformation, Display Vectors, Stream Lines (Particle Tracing)

- **graph lines:** XY plots.
- **animation:** animation of the current result visualization (now only deformation and contour fill/lines).

Almost all these results displays can be selected by using the window `View Results`. To stop viewing a result, the user can select `No Result` on this window, or `Results Geometry NoResults` on the `Right Buttons` menu.

As the results are grouped into steps and analysis, GiD must know which analysis and step is the currently selected for displaying results. The button `AnalysisSelection` is used to select the current analysis and step to be used for the rest of the results options. If some of the results view requires another analysis or step, the user will be asked for it.

Scalar View Results

Here we summarize those results views that only require a scalar value, this can be a scalar result and also a component of a vectorial or matrix result.

Show Minimum and Maximum

With this option the user can see the minimum and maximum of the chosen result. This minimum and maximum can be absolute, for all the meshes/sets/cuts, or relative (local) to the shown ones. This can be enabled or disabled through the `Right Button` menu `Results ContOptions WhichContLimits`.

Contour Fill

Contour Fill description:

This option allows the visualization of colored zones, in which a variable, or a component, varies between two defined values. GiD can use as many colors as allowed by the graphical capabilities of the computer. When a high number of colors is used, the variation of these colors looks continuous, but the visualization becomes slower unless the Fast-Rotation option is used (see section POST-PROCESSING GENERAL OPTIONS). A menu of the variables to be represented will be shown, from which the one to be displayed will be chosen, by using the default analysis and step selected (see section POST-PROCESSING RESULTS OPTIONS).

Vectors will be unfolded into X, Y, and Z components and its module. Symmetric matrix values will be unfolded into Sxx component, Syy component, Szz component, Sxy component, Syz component and Sxz component of the original matrix and also into Si component, Sii component and Siii component in 3D problems or angular variation in 2D problems. Any of these components can be selected to be visualized.

Contour Fill options:

Several configuration options can be accessed via `Results ContOptions` and through the window `preferences` such as setting fixed limits, color outliers, color ramps, and others.

- **Number Of Colors:** Here the number of colors of the Results View can be specified.
- **FixIntervals:** fixes a value, from which the zones will be drawn and a step size which is used to determinate the size of a color zone. The number of zones is calculated by adding and subtracting the step size to the fixed value, until the maximum and the minimum values are reached. If the number of zones to be drawn is greater than that available, a warning message will appear and nothing will be visualized. The chosen value and the step size must be introduced in the same line, separated by a space.
- **Which Contour Limits:** tells the program which contour limits should use when there are no user defined limits: the global ones, i.e., the absolute minimum and maximum of `All Sets`, meshes and cuts, of the current result, or only the `Shown sets`, i.e., the minimum and maximum of the currently visualized meshes/sets/cuts.
- **Fix Analysis Min/Max:** the program will search for the minimum and maximum of the result through all the steps of the analysis, and will use them to draw the results.
- **Set Minimum/Maximum Options:** Inside this group, several options for the minimum value can be defined:
    - **SetValue:** Here the user can set the Minimum/Maximum value that Contour Fill should use. Outliers will be drawn with the color defined in `Out Min Color`/`Out Max Color`.
    - **ResetValue:** Here the user can reset the minimum value, so that Contour Fill looks for that minimum.
    - **OutMinColor / OutMaxColor:** With one of `Black`, `MinColor`/`MaxColor`, `Transparent`/`Material` The user can specify how the outliers values should be drawn.
    - **MinColor / MaxColor:** With one of `Standard InverseStd Black`/`White User` the user can define the color for the minimum value for the color scale to start with. When choosing `Standard` for both `MinColor` and `MaxColor` the color scale will be the default: starting from blue (minimum) through green until red (maximum). If choosing `InverseStd` for both `MinColor` and `MaxColor` the color scale will just be the inverse of

the default: starting from red (minimum) through green until blue (maximum). With `Black` for Minimum and `White` for Maximum, the scale will be a grey ramp. The user can define its own color ramp just using `User` for Minimum and Maximum. Here, after giving the components red, green, blue and opacity, the color scale will vary linearly from the minimum component entered value to that of the maximum component value defined.

- **Color Ramp:** Specifies how should the ramp change from minimum color defined to the maximum color given by the user: `Tangent`, `ArcTangent` or `Linear`. The default is `ArcTangent`. With the option `TerrainMap` a physical-map like color ramp will be used. `RGB / HSV` tells GiD how the colors between the user defined minimum color and the user defined maximum color should change.

Note: When changing any of these options, the results view must be redone in order to take effect.

Contour Lines

Contour Lines description:

This display option is quite similar to contour fill (see section Contour Fill), but here, the isolines of a certain nodal variable are drawn. In this case, each color ties several points with the same value of the variable chosen.

Contour Lines options:

Here the configurations options are almost the same than the ones for Contour Fill (see section Contour Fill options:), with the only difference that the number given in `NumberOfColors` will be used as the number of lines for this contour lines representation.

Iso Surface

Iso Surface description:

Here a surface is drawn that ties a fixed value inside a volume mesh. A line is drawn for surface meshes. To create iso-surfaces there are several options:

- **Exact:** After choosing a result, or a result component, of the current analysis and step (see section POST-PROCESSING RESULTS OPTIONS), the user can input several fixed values and then for each given value a iso-surface is drawn.

- **Automatic:** Similarly, after choosing a result, or a result component, the user is asked for the number of iso-surfaces to be created. GiD calculates the values between the Minimum and the Maximum (these not included).
- **Automatic Width:** After choosing a result, or result component, the user is asked for a width. This width is used to create as many iso-surfaces as needed between the Minimum and the Maximum (these included) values defined.

Iso Surface options:

For Iso-Surfaces there is also a `Display Style` option, like for Meshes/sets/cuts. But with the only options `All Lines`, `Hidden Lines`, `Body`, `Body Lines`, `Transparent`, `Opaque` and for lighting `Normal` - no light, `Render` - like `Flat Render` for Meshes/sets/cuts (see section Display Style).

The colors used for iso-surfaces can also be configured with the same options as Contour Fill (see section Contour Fill options:), but having in mind that `NumberOfColors` and `OutMinColor` / `OutMaxColor` does not affect iso-surfaces.

Another interesting option is `Convert To Cuts`. With this options all the isosurfaces drawn will be turn into cuts, so that they can be saved, read, and used to drawn results over them.

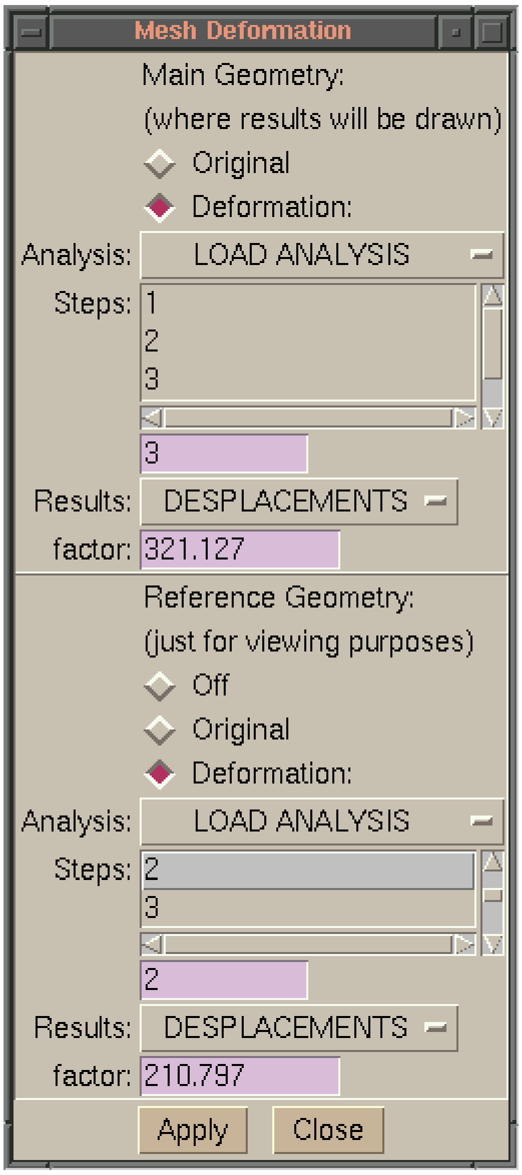
Vector View Results

Here we summarize those results views that require a vectorial or matrix result.

Deform Mesh

Meshes, sets and cuts can be deformed according to a nodal vector and a factor. When doing so all the results are drawn on the deformed meshes, sets and cuts. This is called in GiD `Main Geometry`. Thus, when the `Main Geometry` is deformed, results are also drawn deformed; and when `Main geometry` is in its original state, results are also drawn in its original state.

In GiD there is a `Deform Mesh` window to allow this. This options can also be accessed through the `Right Button` menu.


Mesh Deformation

Main Geometry:

(where results will be drawn)

◇ Original

[✓ Deformation:]

Analysis: LOAD ANALYSIS

Steps: 1 2 3

3

Results: DESPLACEMENTS

factor: 321.127

Reference Geometry:

(just for viewing purposes)

◇ Off

◇ Original

[✓ Deformation:]

Analysis: LOAD ANALYSIS

Steps: 2 3

2

Results: DESPLACEMENTS

factor: 210.797

Deformation Window

On the upper part of this window, the user can choose between the `Original` state of the `Main Geometry` and the `Deformed` state, for which a nodal vectorial result, and an anlysis and step, must be selected and a factor entered.

There is also a so-called `Reference Geometry` option. This allows to visualize meshes, sets or cuts like `Main Geometry` but NO results can be displayed over these meshes, sets or cuts. It is merely provided as a reference, to contrast several deformation or changes of the original geometry. Remember that the current Mesh `Display Style` will be used for the `Reference Geometry` and it can be only changed when redoing it.

On the lower part of the `Deform Mesh` window, this `Reference Geometry` can be configured. The user can choose between:

`Off`, so this reference visualization is not displayed.

`Original`, if `Main Geometry` is deformed and the user wants to compare it with its original state without loosing a results representation.

`Deformation`, where, after providing an analysis, step, result and factor, the user can use it to contrast two deformation states, or a deformed state and an original geometry.

Display vectors

Display Vectors description:

This option displays a menu with results from vectors and matrices (where the principal values are previously evaluated by the program). A menu of variables to be represented will be shown, from which the one to be seen should be chosen, using the default analysis and step selected (see section POST-PROCESSING RESULTS OPTIONS). Once a vector is chosen, the program will display the nodal vectors of the chosen result. The drawn vectors can be scaled interactively. The factor can be applied several times and every time it changes to the new input value.

Vectors are drawn in green, but the user can modify them to be drawn in colors according to the vector module.

When drawing a matrix result, such as the stress tensor, only a single color representation of principal values is available: blue when negatives (also drawn as >--< representing compressions), and red when positives (also drawn as <--> representing tensions).

Display Vectors options:

Vectors can be drawn in one color, `MonoColor`, or in several colors, `ColourModules`. This can be selected in the `Postprocess` pane of the `Preferences` window, or inside the `Results Options` on the `Right Button` menu.

If the user uses `ColourModules` to draw vectors, he can also select the number of colors, in addition to the number of colors of `Contou Fill`, from within the previous windows, or menu, under `VectorNumColours`.

Another configuration possibility is `VectorOffset`, also on the same locations. With this option, the user controls where the vector should be in relation of the node, ranging from 0, to tie the tail of the arrow to the node, to 1, to tie the tip of the arrow to the node.

Note: These options has no effect when viewing a matrix result.

Stream Lines

Stream Lines description:

With this option the user can display a stream line, or, in fluid dynamics, a particle tracing, in a vector field. After choosing a vector result, using the default analysis and step selected (see section POST-PROCESSING RESULTS OPTIONS), the program asks the user for a point to start with the plotting of the stream line. This point can be given in several ways:

- **just clicking on the screen:** the point will be the intersection between the line orthogonal to the screen and the plane parallel to the screen and containing the `Rotation center`,
- **joining a node:** the selected node will be used as a start point,
- **select nodes:** here several nodes can be selected to start with,
- **along line:** with this option the user can define a segment, along which several start points will be chosen. The number of points will also be asked for, including the ends of the segment. In case of just one point start, this will be the center of the segment.
- **in a quad:** the user can enter here four lines that define a quadrilateral area which will be used to create a N x M matrix of points. These points will be the start for the stream lines. When giving N x M, N lies on the first and third line, and M on the second and fourth. So, points (0, 0..N), (M, 0..N), (0..M, 0) and (0..M, N) will lie just on the lines. But in case of N=1 or M=1, this will be the center of the line, and if N = 1 and M = 1, this will be the center of the Quad.
- **intersectset** the point will be the intersection between the line orthogonal to the screen and the current viewed sets nearest the view point.

When viewing `Stream Lines`, labels are drawn to show the times at start and end points of the stream line.

Stream Lines options:

Two options can be chosen for `Stream Lines`:

- `StreamLineSize`: Here the user can select the size of the stream lines. With 0, the stream lines will be drawn as a 3 pixel wide line (render view does not affect its aspect). With a size bigger than 0, stream lines will be drawn with model related size, in square brackets a suggested value is shown (here when using a render view, stream lines are also rendered).

- `StreamLabel`: With one of `None`, `0_End` and `Ini_End` the user tells GiD whether not to drawn label; to drawn them with the convention 0 at the start of the stream line and the total time taken for the particle to travel at the end, or to drawn them with 0 at the chosen point, and - `time before` at the begin of the stream line and `time after` at the end.

At the moment these options can only be accessed through the `Right Button` menu, inside the `Results Options`.

Graph Lines

Graph Lines description:

Here the user can draw graphs in order to have a closer look to the results. Several graphs types are available: point analysis against time, result 1 vs. result 2 over points, result along a boundary line and section analysis of a results. The user can also save or read a Graph. The format of the file will be described see section Graph Lines File Format

- `PointSteps`: this graph shows the evolution of a result on a point along all the steps in the current analysis.
- `Pointgraph`: after choosing a point or points, the user can contrast a results against another,
- `Bordergraph`: clicking on a boundary, the user can see how the results varies along this boundary.
- `Linegraph`: drawing a segment on the screen, GiD display a graph with the results along this segment on the surface mesh (option under development).
- `ReadGraph`: a new graph will be created from a file.
- `WriteGraph`: the graph the user chooses will be written into an archive. When the option `All` is choosen, the user will be asked for a `prefix`. Then GiD will create a file for each Graph with the names `prefix`**-1.cur**, `prefix`**-2.cur**, `prefix`**-3.cur** and so on . . .

The user can also view the labels of the points of the graphs, not only the Graph points number, but also theirs X and Y values. If some points of the Graphs are labeled, when turning into the normal results view, the labels also appear on the results view.

Graph Lines options:

- `ShowGraphs`: here the user can switch between the graphs view and the post-processing view.
- `ClearGraphs`: to reset all the graphs and do a new start.
- `Legend2Title`: switch from a legend like graph enumeration to a title like one.
- `Grids`: here the user tells whether to drawn grids or not.
- `StyleGraphs`: the user can choose how the new graphs should look like.
- `CurrentStyle`: this option allows to drawn all the graphs with the same style.
- `Title`: here with, the user can change the title of the Graf. Enter "" to turn to the default one.
- `X_Axis`: here the user can set min and max values and divisions for the X axis, reset them, and change its Label.
- `Y_Axis`: here the user can set min and max values and divisions for the Y axis, reset them, and change its Label.

Graph Lines File Format

The Graph file that GiD uses is an standard ASCII file. Every line of the file is a point of the Graph with X and Y coordinates separated by a space. Comment lines are also allowed and should begin with a '`#`'. The title of the Graph and the labels for the X and Y axis can also be configured. If a comment line contains the Keyword '`Graf:`' the string between quotes that follows this keyword will be used as title of the graf. The string between quotes that follows the Keyword '`X:`' will be used as label for the X axis. The same is also true for the Y axis, but for the Keyword '`Y:`'.

Here an example:

```
# Graf: "Nodes 26, 27, 28, ... 52 Graf."
#
# X: "Szz-Tensiones_Nods." Y: "Sxz-Tensiones_Nods."
-3055.444 1672.365
-2837.013 5892.115
-2371.195 666.9543
```

```
-2030.643 3390.457
-1588.883 -4042.649
-1011.5 1236.958
# End
```

Animation

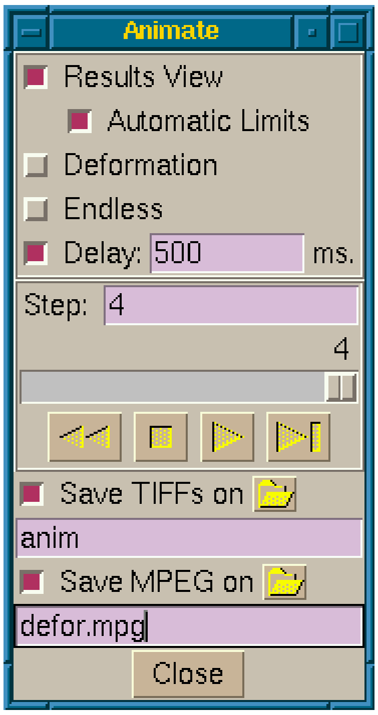
Animation Basic Tools:

The basic tools that GiD provides to create an animation are under `Utilities AnimationFile`. The animation file follows the MPEG-I standard with a frame rate on 25 fps. Windows users should have in mind that a MPEG file greater than 720x576 pixels, and created without `Delay` can not be viewed with the standard programs on Microsoft Windows (i.e. Multimedia Player, and others).

- `Start`: Here the user can create and open an animation file (in MPEG format).
- `AddFrame`: This option tells the program to save the current view to the `Start`ed animation file, keeping in mind that a frame will be shown 1/25 s. (0.04 s.).
- `End`: Here the user closes the animation file.
- `FramesPerStep`: The user tells here the size of a GOP (group of encoding frames) in frames. This option allows to reduce the size of an MPEG file when adding lots of frames (keep in mind that, when using this option, the number of total frames should be X times `FramesPerStep`).

Animate Window:

With this window a little bit of automatization has been done to create animations inside GiD. Nowadays, almost all the results visualization are animated. Only stream-lines are not automatized along all the steps of the current analysis can be done.


Animate

Results View: ✓

Automatic Limits: ✓

Deformation: ☐

Endless: ☐

Delay: ✓ 500 ms.

Step: 4

4

Save TIFFs on: ✓

anim

Save MPEG on: ✓

defor.mpg

Deformation Window

On the right of the `Step:` label, the step value is shown. On the slide bar, the step number is shown.

The four buttons under the slide bar are autoexplicative, they should 'Rewind', 'Stop', 'Play' and 'Step' the animation. Also clicking on the slide bar the animation can be stepped down or up.

Options are:

- `Automatic Limits`: here GiD searches for the minimum and maximum values of the results along all the steps of the analysis and uses them to draw the results view through all the steps.

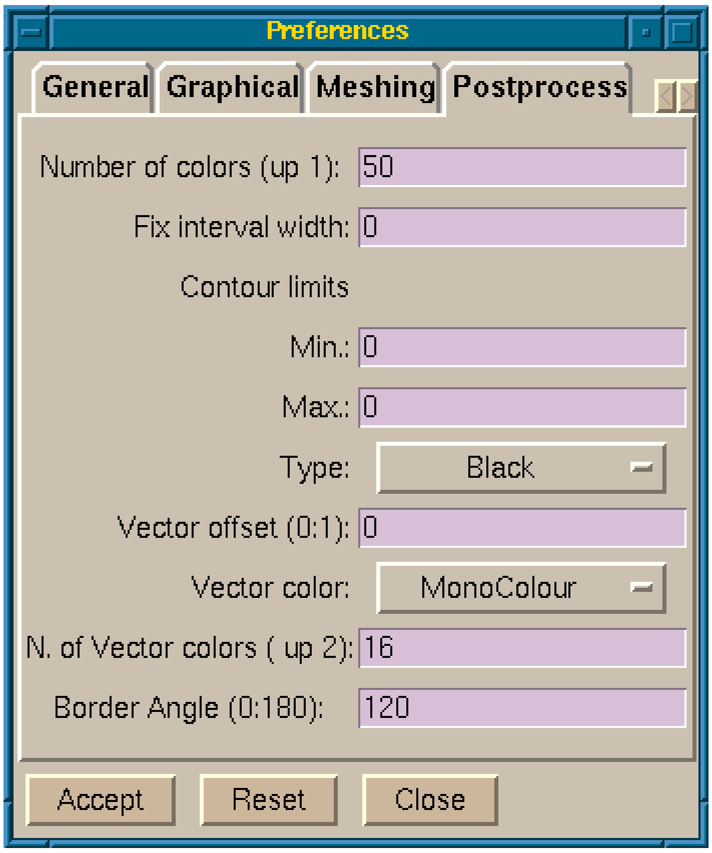
- `Endless`: to do the animation forever.

- `Delay`: here the user can specify a delay time between steps in miliseconds.

- `Save TIFFs on`: with this option the user can save a TIFF-snapshot of each step, when the '`Play`' button is pushed. Here the filename given will be used as a prefix to create the TIFFs, for instance, if the user writes `MyAnimation`, TIFF files will be created with names `MyAnimation-1.tif`, `MyAnimation-2.tif`, and so on.

- `Save MPEG on`: giving here a filename, a MPEG file will be created when 'Play' button is pushed. Windows users should have in mind that a MPEG file greater than 720x576 pixels, and created without `Delay` can not be viewed with the standard programs on Microsoft Windows (i.e. Multimedia Player, and others).

POST-PROCESSING PREFERENCES



Number of colors (up 1): 50
Fix interval width: 0
Contour limits
Min.: 0
Max.: 0
Type: Black
Vector offset (0:1): 0
Vector color: MonoColour
N. of Vector colors (up 2): 16
Border Angle (0:180): 120

Post-process preferences

Several different preference options concerning to the presentation of the results are available:

- Number of colors.
- Fixed intervals.
- Set Contour limits.
- Outside contour limits colors.
- Which Contour limits.
- Vectors offset.
- Vectors colors.

- Number of Vectors colors.
- Border angle.

`NumberOfColors` So, the number of colors allows to determine the number of colors wanted for visualization, inside the range of the graphical capabilities of the computer. A legend on the bottom-right corner of the visualization area will appear with the color of every contour (see section Contour Fill options:).

`FixIntervals` fixes a value, from which the zones will be drawn and a step size which is used to define the size of a color zone. The number of zones is calculated by adding and subtracting the step size to the fixed value, until the maximum and the minimum values are reached. If the number of zones to be drawn is greater than available, a warning message will appear and nothing will be visualized. The chosen value and the step size must be introduced in the same line, separated by a space (see section Contour Fill options:).

`SetContLimits` allows to change the maximum and minimum limits of the contours representation. All the values that lie outside this range can be drawn in black or with the minimum and maximum values colors, i.e., dark blue and red (see section Contour Fill options:).

This option allows to see the contours only over one part of the surfaces (it would be very useful, for instance, to represent a shock-wave in supersonic flows) and the rest of them are represented geometrically.

`HowContLimits` defines how to see the values that lie outside the previously defined range. If the activated option is `black`, all the values will be drawn in black. Maximum and minimum values that lie over or under the previously defined range will be drawn in the maximum and minimum definition colors (see section Contour Fill options:).

`WhichContLimits` defines which contour limits should be used when there are no user defined ones: the global ones, i. e., the minimum and maximum of all the meshes/sets/cuts of the current result, or only the local ones, i. e., the minimum and maximum of the meshes/sets/cuts that are currently visualized of the current result (see section Contour Fill options:).

`VectorOffset` points the origin of the relative vector at every node. It varies from 0, when the origin coincides with the node, to 1, when the end of the vector points to the node (see section Display Vectors options:).

`VectorColour` modules the vector coloring, for the different options from several colors to just monochrome (see section Display Vectors options:).

`VectorNumCols` defines how many colors should be used to draw the vectors in Vector Color Mode, inside the range of the graphical capabilities of the computer (see section Display Vectors options:).

`BorderAngle` gives the threshold to represent the edges between different faces if the relative angle that conforms the adjacent faces is smaller than the defined value. This can be between 0 and 180 degrees. The default is 120 degrees (see section Display Style).

BASIC CUSTOMIZATION IDEAS

When GiD is to be used for a particular type of analysis, it is necessary to predefine all the information required from the user and to define the way the final information is given to the solver module. To do so, some files are used where the conditions, materials, data, symbols and format of the resulting file for the solver are described.

Due to the vocation of GiD as general purpose pre and post processor, the configuration for the different analysis must be performed according to the particular specifications of every solver. This implies the necessity of creating specific data input files for every solver. However, GiD allows to perform this configuration process inside itself without any change in the solver and without having to program any independent utility.

To configure these files means to define the data that must be input by the user, as well as the materials to be implemented and other geometrical and time-dependent conditions. It is also possible to add some symbols or drawings to represent the defined conditions. It must be also defined the way that all this data must be written inside a file that will be the input file to be read by the corresponding solver.

The customization is performed with the definition of a type of problem, which will be added to those included in the system and the inclusion of some additional files (see section CONFIGURATION FILES). In other words, it implies either the creation of a directory with the type of problem's name and extension .gid. This GiD project can be located in the current working directory or in the main GiD executable directory. The first case, the creation inside the current working directory, can be useful during the development of the project. Once it is finished, it can be advisable to move the directory to the one where GiD is stored. In both cases, the series of files will be included inside the project directory. The name for most of them will be composed by the same problem's name and an extension referring to their function. Considering NAME to be the name that defines the project, the diagram of the project's configuration is the following:

```
directory:     NAME.gid
files:
```

- Configuration files

    - NAME.cnd   Conditions definitions.
    - NAME.mat   Materials properties.
    - NAME.prb   Problem and intervals data.
    - NAME.sim   Conditions symbols.
    - ***.geo   Symbols geometrical definitions.

      ...
    - ***.geo   Symbols geometrical definitions.

- Compilation files

    - NAME.bas   Information for the data input file.
    - ***.bas   Information for additional files.

      ...
    - ***.bas   Information for additional files.

- TCL extension files.

- NAME.tcl Extensions to GiD written in the TCL-TK programming language.

- Command execution files.

  - NAME.bat Operating system shell that executes the analysis process.

Files `NAME.sim`, `***.geo` and `***.bas` are not mandatory and can be added to facilitate the visualization (files `NAME.sim` and `***.geo`) or to prepare the data input for the restart in additional files (files `***.bas`).

CONFIGURATION FILES

These files generate the conditions and material properties, as well as the proper general problem and intervals data to be transferred to the mesh, giving at the same time the chance to define geometrical drawings or symbols to represent some conditions on the screen.

Conditions definitions

The file with extension's name `.cnd` contains all the information about the conditions that can be applied to different entities. The condition can adopt different field values for every entity. This type of information includes, for instance, all the displacement constraints and applied loads in a structural problem or all the prescribed and initial temperatures in a thermical analysis.

An important characteristic of the `conditions` is that they must define over what kind of entity are going to be applied, i.e., over points, lines, surfaces or volumes and over what kind of entity will be transferred, i.e., over nodes or elements.

Another important feature is that all the conditions can be applied to different entities with different values for all the defined intervals of the problem.

Therefore, a condition can be considered as a group of fields containing the name of the referred condition, the geometric entity over which it is applied, the mesh entity over which it will be transferred, its corresponding properties and their values.

The format of the file is as follows:

```
NUMBER: condition_number CONDITION: condition_name
CONDTYPE: 'over' ('points', 'lines', 'surfaces', 'volumes')
CONDMESHTYPE: 'over' ('nodes', 'elems')
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
  ...
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
END CONDITION
NUMBER: condition_number CONDITION: condition_name
  ...
END CONDITION
```

Note that this file format does not permit to put blank lines between the last line of a condition definition, `END CONDITION`, and the first one of the next condition definition.

A condition can have a field like:

```
QUESTION: field_name['#LA#'('-GLOBAL-')]
VALUE: default_field_value
```

or:

```
QUESTION: field_name['#LA#'('-GLOBAL-', '-AUTOMATIC-')]
VALUE: default_field_value
```

This type of field makes reference to the local axes system to be used. If it only has a single default value, this will be the name of the global axes. If two values are given, the second one one will reference a system that will be automatically computed for every node and that will depend on the geometric constraints, like tangencies, orthogonalities, etc. All the different user defined systems will be added automatically to these default possibilities.

All the fields must be fulfilled with words, considering as a word a character string without any blank space amongst them. The strings signaled between quotes are literal and the ones inside brackets are optional. The interface is case-sensitive, so the uppercase letters must be maintained. `Default_field_value` and different `optional_value_i` can be alphanumeric, integers or reals. GiD treats them as alphanumeric until the moment that are written to the solver input files.

The numbers of the conditions have to be consecutive, beginning with number 1.

There is no need to point out the overall number of conditions or the respective number of fields for each one. This last one can be variable for each condition.

One optional flag to add to a condition is:

```
CANREPEAT: yes
```

It is written after CONDMESHTYPE and means that user can assign one condition several times to the same entity.

Detailed example-Condition file creation

Next is an example of a condition file creation, explained step by step:

1. First, you have to create the folder or directory where all the problem type files are located, `problem_type_Example.gid/` on this case.
2. Then create and edit the file (`problem_type_Example.cnd` on this example) inside the recently created directory (where all your problem type files are located). As you can see, except for the extension, the name of the file and the directory are the same.
3. Create the first condition, which starts with the line

   ```
   NUMBER: 1 CONDITION: Point-Constraints
   ```

   where the first number is used to index the condition. Each condition will be refered with one unique number (two conditions can not share the same number). The second parameter is the name of the condition. Again, a unique condition name into this condition file is required.
4. This first line is followed by the next pair:

   ```
   CONDTYPE: over points
   CONDMESHTYPE: over nodes
   ```

   which declare over what entity is going to be applied the condition. The first line, `CONDTYPE:...` refers to the geometry, and may take as parameters the sentences "over points", "over lines", "over surfaces" or "over volumes". The second line refers to the type of condition applied to the mesh, once generated. GiD does not force to provide this second parameter, but if it is present, the treatment and evaluation of the problem will be more acurate. The available parameters for this statement are "over nodes" and "over elements".
5. Then, you'll have to declare a set of questions and values applied to this condition.

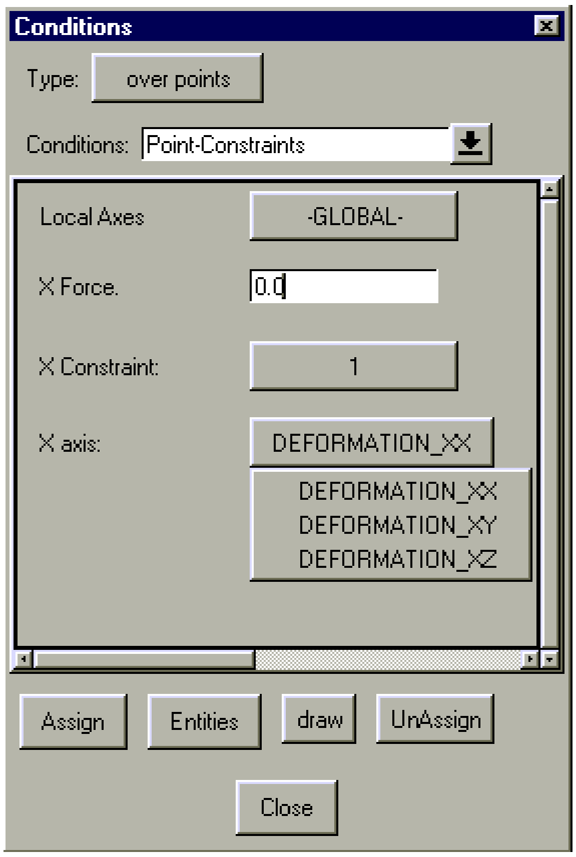
```
QUESTION: Local-Axes#LA#(-GLOBAL-)
VALUE: -GLOBAL-
QUESTION: X-Force
VALUE: 0.0
QUESTION: X-Constraint:#CB#(1,0)
VALUE: 1
QUESTION: X_axis:#CB#(DEFORMATION_XX,DEFORMATION_XY,DEFORMATION_XZ)
VALUE: DEFORMATION_XX

END CONDITION
```

After the QUESTION: word you have the choice of put:

- An alphanumeric field name followed by the #LA# statement, and then the single or double parameter.
- An alphanumeric field name.
- An alphanumeric field name followed by the #CB# statement, and then the optional values between parenthesis.

The VALUE: word must be followed by one of the optional values, if you have declared them in the previous QUESTION: line. If you don't follow this statement, the program may not work correctly. In the previous example, the `X-Force` QUESTION takes the value 0.0. Also in the example, the `X-Constraint` QUESTION includes a combo box statement (`#CB#`), followed by the declaration of the choices 1 and 0. In the next line, the value takes the parameter 1. The `X_axis` QUESTION declares three items for the combo box `DEFORMATION_XX,DEFORMATION_XY,DEFORMATION_XZ`, with the value `DEFORMATION_XX` choosen. Beware of leaving blank spaces between parameters. If in the first question, you put the optional values `(-GLOBAL, -AUTO-)`, (note the blank space after the comma), there will be an error when reading the file. Take this precaution specially in the Combo Box question parameters, to avoid inpredictable parameters.


Type: over points

Conditions: Point-Constraints

Local Axes: -GLOBAL-

X Force.: 0.0

X Constraint: 1

X axis: DEFORMATION_XX

Conditions edition window on Preprocess, showing an unfolded combo box

Materials properties

This file `NAME.mat` include originally the definition of different materials through their properties. These are base materials as they can be used as templates during the pre-processing step for the creation of newer ones.

The user can define as many materials as desired and with a variable number of fields. All the unused materials will not be taken in consideration when writing the data input files for the solver. Alternatively, they can be useful to generate a materials library.

Conversely to the case of conditions, the same material can be assigned to different geometrical entities levels (lines, surfaces or volumes) and even can be assigned directly to the mesh elements.

In a similar way as a condition was defined, a material can be considered as a group of fields containing its name, its corresponding properties and their values.

The format of the file is as follows:

```
NUMBER: material_number MATERIAL: material_name
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
  ...
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
END MATERIAL
NUMBER: material_number MATERIAL: material_name
  ...
END MATERIAL
```

If a property of a material change depending on something (one example would be one property depending on the temperature and defined with several values for several temperatures), a table of changing values may be declared for this property. When the solver evaluates the problem, it reads the values and apply the suitable property value.

The declaration of the table requires a pair of lines:

First, a QUESTION line with a list of alphanumeric values between parenthesis.

```
QUESTION: field_name:(...,optional_value_i,...)
```

This values are the name of each of the columns in the table, and the number of values declared are the number of columns.

This first line is followed by the actual data declaration: a line started with the words `VALUE: #N#` is followed by a number that indicates the quantity of elements in the matrix and, finally, the list of values.

```
VALUE: #N# number_of_values ... value_number_i ...
```

The number of values declared for the matrix, obviously, has to be the multiplication of the number of columns by the number of rows to be declared.

The usual case of aplication of this declaration is on the thermo-mechanical simulations, where the problem is exposed to a temperature variation, and the properties of the materials change for each temperature value.

All the fields must be filled with words, considering a word as a character string without any blank space amongst them. The strings signaled between quotes are literal and the ones within brackets are optional. The interface is case-sensitive, so the uppercase letters must be maintained. `Default_field_value` and different `optional_value_i` can be alphanumeric, integers or reals, depending on the type.

The numbers of the materials have to be consecutive, beginning with number 1. There is no need to point out the overall number of materials or the respective number of fields for each one. This last one can be variable for each material.

Note that in this file it's not permitted the inclusion of blank lines between material definitions neither between questions and values.

Problem and intervals data

The file `NAME.prb` contains all the information about the general problem and intervals data. The general problem data is all the information required for performing the analysis and it does not concern any particular geometrical entity. This differs from the previous definitions of conditions and materials properties, which are assigned to different entities. Example of general problem data can be the type of solution algorithm used by the solver, the value of the various time steps, convergence conditions and so on.

Within this data, the user may consider the definition of specific problem data (for the whole process) and intervals data (variable values along the different solution intervals). An interval would be the subdivision of a general problem that contains its own particular data. Typically, one can define a different load case for every interval or, in dynamic problems, not only variable loads, but also changing the various time steps, convergence conditions and so on.

The format of the file is as follows:

```
PROBLEM DATA
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
  ...
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
END PROBLEM DATA
INTERVAL DATA
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
  ...
QUESTION: field_name['#CB#'(...,optional_value_i,...)]
VALUE: default_field_value
END INTERVAL DATA
```

All the fields must be filled with words, considering a word as a character string without any blank space amongst them. The strings signaled between quotes are literal and the ones inside brackets are optional. The interface is case-sensitive, so the uppercase letters must be maintained. `Default_field_value` and different `optional_value_i` can be alphanumeric, integers or reals, depending on the type.

Conditions symbols

This file `NAME.sim` comprises different symbols to represent some conditions during the pre-processing stage. The user may define these symbols by creating, ad hoc, geometrical drawings and the appropriate symbol will appear over the entity with the applied condition every time the user asks for it.

One or more symbols can be defined for every condition and the selection will depend on the specified values in the file, which may be obtained through mathematical conditions.

Also the spatial orientation can be defined in this file, depending on the values taken by the required data. For global definitions, the user must input the three components of a vector to express its spatial direction. GiD takes these values from the corresponding conditions window. The orientation of the vector can be understood as the rotation from vector (1,0,0) towards the new vector defined in the file.

For line and surface conditions, the symbols may be considered as local. In this case, GiD does not consider the defined spatial orientation vector and it takes its values from the line or surface orientation. The orientation assumes vector (1,0,0) to be the corresponding entity's normal.

These components, making reference to the values obtained from the adequate conditions, may include C-language expressions. They express the different field values of the mentioned condition as `cond(type,i)`, where `type` (`real` or `int`) makes reference to the type of variable (independent of the letters case) and `i` is the number of the field for that particular condition.

The following is an example:

```
cond Point-Constraints
3
global
cond(int,5)
1
0
0
Support3D.geo
global
cond(int,1) && cond(int,3)
1
0
0
Support.geo
global
cond(int,1) || cond(int,3)
cond(int,3)
cond(int,1)*(-1)
0
Support-2D.geo
cond Face-Load
1
local
fabs(cond(real,2))+fabs(cond(real,4))+fabs(cond(real,6))>0.
cond(real,2)
cond(real,4)
cond(real,6)
Normal.geo
```

This is a particular example of the file `NAME.sim` where the user has defined four different symbols. Each one is read from a file `***.geo`. There is no indication of how many overall symbols are implemented. GiD simply reads all the file long through the end.

The files `***.geo` are obtained through GiD. The user may design a particular drawing to symbolize a condition and this drawing will be stored as `PROBLEMNAME.geo` when saving this project as `PROBLEMNAME.gid`. The user needs not to be concerned about the symbol size but has to keep in mind that the origin corresponds to the point (0,0,0) and the reference vector is (1,0,0). Subsequently, when these files `***.geo` are invoked from `NAME.sim`, the symbol drawing, appears scaled on the display, at the entities location.

Nevertheless, the number of symbols and consequently, the number of files ***.geo, can vary from a condition to another. In the previous example, for instance, the condition called `Point-Constraints`, which is so declared by the use of `cond`, comprises three different symbols. GiD knows that from the number 3 written below the condition's name. Next, GiD reads if the orientation is relative to the spatial axes (global) or moves together with its entity (local). In the example, the three symbols concerning point constraints are globally oriented.

Let's imagine that this condition has six fields. First, third and fifth field values express if there exist constraints along `X-axis`, `Y-axis` and `Z-axis`, respectively. These values are integers and in case they were null, the concerned degree of freedom would be assumed to be unconstrained.

For the first symbol, got from the file `Support3D.geo`, GiD reads `cond(int,5)`, or `Z-constraint`. If it is false, what means that the field's value is zero, the C-condition will not be accomplished and GiD will not draw it. Otherwise, the C-condition will be accomplished and the symbol will be invoked. When this occurs, GiD skips the rest of symbols related to this condition. Its orientation will be the same of the original drawing because the spatial vector is (1,0,0).

All the considerations are valid for the second symbol, got from the file Support.geo, but now GiD has to check that both constraints (`&&`), `X-constraint` and `Y-constraint` are fixed (values different from zero).

For the third symbol, got from the file `Support-2D.geo`, only one of them has to be fixed (`||`) and the orientation of the symbol will depend on which one is free and which one is fixed, showing on the screen the corresponding direction for both degrees of freedom.

Finally, for the fourth symbol, got from the file `Normal.geo`, it can be observed that the drawing of the symbol, related to the local orientation will appear scaled according to the real-type values of the second, fourth and sixth fields values. Different ways of C-language expressions are available in GiD. Thus, the last expression would be equivalent to put '`(fabs(cond(real,2))>0. || fabs(cond(real,4))!=0. || fabs(cond(real,6))>1e-10)`'.

Note: As previously mentioned, GiD internally creates a `PROJECTNAME.geo` when saving a project, where it keeps all the information about the geometry in binary format. In fact, this is the reason why the extension of these files is `.geo`. However, the file `PROJECTNAME.geo` is stored in the `PROJECTNAME.gid` directory, whereas these user-created files `***.geo` are stored in the `NAME.gid` directory, being `NAME` the problem's name.

COMPILATION FILES

Once the user has generated the mesh, assigned the conditions and the materials properties, as well as the general problem and intervals data for the solver, it is necessary to produce the data input files to be processed by that program. To manage this reading, GiD has the capability of creating a file called `NAME.bas` which describes the format and structure of the required data input file. This file must remain in the `NAME.gid` directory, as well as the other files already described, `NAME.cnd`, `NAME.mat`, `NAME.prb` and also `NAME.sim` and `***.geo`, if desired.

In case more than one data input file is needed, GiD allows the creation of more files by means of additional files `***.bas` (note that while `NAME.bas` creates an data input file named `PROJECTNAME.dat`, successive files `***.bas`, -where `***` can be any name-, create successive files `PROJECTNAME-1.dat`, `PROJECTNAME-2.dat` and so on), the new files follow the same rules than the ones explained next for `NAME.bas`.

These files work as interface from the GiD's standard results to the specific data input for any individual solver module. This allows to complete the process of running the analysis (see section CALCULATE) as one step more within the system.

In case of an error in the preparation of the data input files, the programmer has only to fix the corresponding file `NAME.bas` or `***.bas` and rerun the example, without having to leave GiD, recompile or, even more, reassign any data or remesh.

This facility is due to the structure of the compilation files. They are a group of macros (like an ordinary programming language) that can be read, without need of any compiler, every time the corresponding analysis file is to be written. This ensures a fast way to debug mistakes.

General description

All the rules that apply to the file `NAME.bas` are also valid for the rest of files with extension `.bas`. Thus, everything in this section will refer explicitly to the file `NAME.bas`. Any information written into this file, apart from the so-called commands, is reproduced exactly in the output file (the data input file for the numerical solver). The commands are words that begin with the character *****. If the programmer wants to write an asterisk into the file he should write ******. The commands are inserted among the text to be literally translated. Everyone of these commands returns one (see section Single value return commands) or multiple (see section Multiple values return commands) values obtained from the pre-processing. Other commands mimic the traditional structures to do loops or conditionals (see section Specific commands). It is also possible to create variables to manage some data. Comparing it to a classic programming language, the main differences will be the following:

- The text is reproduced literally, without printing instructions, as it is writing-oriented.
- There are no indices in the loops. When the program begins a loop, it already knows the number of iterations to perform. Furthermore, the inner variables of the loop change their values automatically. All the commands can be divided into three types:
    - Commands that return one single value. This value can be an integer, a real or a string. The value depends on certain values that are available to the command and on the position of the command, within the loop or after setting some other parameters. These commands can be inserted within the text and write their value where it corresponds in it. They can also appear inside an expression, what would be the example of the conditionals. For this example, the user may specify the type of the variable, integer or real, except when using `strcmp` or `strcasecmp`. If these commands are within an expression, no ***** should precede the command.
    - Commands that return more than one value. Their use is similar to that of the previously indicated commands, except for the fact that they cannot be used in other expressions. They can return different values, one after the other, depending on some values of the project.
    - Commands that perform loops or conditionals, create new variables, or define some specifications. The latter include condition or type of element chosen and also serve to prevent line-feeding. These commands must start at the beginning of the line and nothing will be written into the calculations file. After the command, in the same line, there can be other commands or words to complement the definitions, so, in an end of a loop or conditional, after the command the user may write what loop or conditional was finished.

The arguments that appear in a command are written immediately behind it and inside parenthesis. If there are more than one, they will be separated by commas. The parenthesis might be put without any argument inside. This is useful to write something just behind the command without any separation in the middle. The arguments can be real numbers or integers, meaning the word `REAL` or the word `INT` (both in upper or lowercase) that the value to which it points has to be considered as real or integer, respectively. Other types of arguments are sometimes allowed, like the type of element, described by its name, in the command `*set elem`, or a chain of characters inserted between double quotes " for the C-instructions `strcmp` and `strcasecmp`. It is also possible, sometimes, to write the field's name instead of its ordering number.

Single value return commands

When writing a command, it is generally (if not explicitly mentioned) independent of the letters case, even a mixture of uppercase or lowercase would not affect the results.

- ***npoin**, ***ndime**, ***nnode**, ***nelem**, ***nmats.** They return, respectively, the number of points, the dimension of the project being considered, the number of nodes of the element with the highest number, the number of elements and the number of materials. All of them are considered as integers and do not carry arguments (see `*format`, `*intformat`), except `*nelem`, which can bring different types of elements. These elements are `Linear`, `Triangle`, `Quadrilateral`, `Tetrahedra`, `Hexahedra`, depending on the number of edges of the element and `All`, that comprises all the possible types. The command `*nmats` returns the number of materials effectively assigned to an entity, not all the defined ones.

- ***GenData.** It must carry an argument of integer type that specifies the number of the field to be printed. This number is the order of the field inside the general data list. This must be one of the values that are fixed for all the problem, independently of the interval (see section Problem and intervals data). The field's name can also be the argument, instead, or even an abbreviation of it.

  The arguments `REAL` or `INT` to express the type of number for the field are also available (see `*format`, `*intformat`, `*realformat`, `*if`). If they are not specified, the program will print a character string. It is mandatory to write one of them within an expression, except for `strcmp` and `strcasecmp`. The numeration must start with number 1.

- ***IntvData.** The only difference with the previous one is that the field must be one of those fields varying with the interval (see section Problem and intervals data). This command must be within a loop over intervals (see `*loop`) and the program will automatically update the suitable value for each iteration.

- ***MatProp.** The only difference with the previous ones is that it must be within a loop over the materials (see `*loop`). It returns the property whose field's number is defined by the parameter of the variable.

  Caution: If there are materials with different number of fields, the user must ensure not to print non-existent fields, using conditionals.

- ***cond.** The same previous remarks apply here, although now the user must notify with the command `*set` (see `*set`) which is the condition being processed. It can be within a loop (see `*loop`) over the different intervals in case that the conditions vary for each interval.

- ***ElemsNum**, ***NodesNum**, ***MatNum**, ***ElemsMat.** These commands return, respectively, the element's number, the node's number, the material's number and the number of the material assigned to the element. All of them must be within a proper loop (see `*loop`) and change automatically for each iteration. They are considered as integers and cannot carry any argument. The number of materials will be reordered in order to begin from number 1 and will increase up to the number of materials assigned to any entity.

- ***CondNumEntities.** The user must have previously selected a condition (see `*set`). It returns the number of entities that have a condition assigned over them.

- ***LoopVar.** This command must be inside a loop and returns, as an integer, what is considered to be the internal variable of the loop. This variable takes the value 1 in the first iteration and increases a unit for each new iteration.

  The parameter `elems,nodes,materials,intervals`, used as argument for the corresponding loop, allows the program know which one is being processed. Otherwise, if there are nested loops, the program takes the value of the inner loop.

- ***Operation.** It returns the result of an arithmetical expression what should be written inside parenthesis just immediately behind the command. This operation must be defined in C-format and can contain any of the commands that return one single value. The user can force the return of an integer or a real by means of the mentioned parameters `INT` or `REAL`. Otherwise, GiD returns the type according to the result. The following are valid examples of operations:

  ```
  *operation(4*elemsnum+1)
  *operation(8*(loopvar-1)+1)
  ```

  Note: There cannot be blank spaces between the commands and the parenthesis that include the parameters.

  Note: Commands inside `*operation` do not need to put ***** at the beginning.

- ***LocalAxesNum.** It returns the identification name of the local axes system, either when the loop is over the nodes or when it is over the elements, under a referenced condition.
- ***nlocalaxes.** It returns the number of the defined local axes system.
- ***IsQuadratic.** It returns the value 1 when the elements are quadratic or 0 when they are not.

Multiple values return commands

These commands return more than one value in a prescribed order, writing them one behind another. All of them, but `LocalAxesDef`, are able to return one single value when a numerical argument giving the order of the value is added to the command. In this way, these commands can appear within an expression. `LocalAxesDef` or the rest of the commands without the numerical argument cannot be used inside expressions. Following, it is displayed a list of the commands with the appropriate description:

- ***NodesCoord.** This command writes the node's coordinates. It must be inside a loop (see `*loop`) over the nodes. The coordinates are considered as real numbers (see `*realformat` and `*format`). It will write two or three coordinates according to the problem's dimension (see `*Ndime`).
- ***ElemsConec.** This command writes the element's connectivities, i.e., the list of the nodes that belong to the element, keeping the proper sense for each case (anti-clockwise sense in 2D and depending on the standards in 3D). For shells, the user must define the sense. However, this command accepts the argument `invert` and this implies that the ordering of the nodes in quadratic elements will be inverted. The connectivities are considered as integer numbers (see `*intformat` and `*format`).
- ***GlobalNodes.** This command returns the nodes that belong to a certain element's face where a condition has been defined (on the loop over the elements). The sense is the same that the sense of the element's connectivities. The returned values are considered as integer numbers (see `*intformat` and `*format`).
- ***LocalNodes.** The only difference with the previous one is that the returned value is the local node's numbering for the corresponding element (between 1 and `nnode).`
- **LocalAxesDef.** This command returns the nine numbers that define the transformation matrix of a vector from the local axes system to the global one.

Specific commands

- ***\** To avoid line-feeding, the user must write ***\**, so, the currently used line continues on the following line of the file `NAME.bas`.
- ***#** Written at the beginning of the line, consider this line as a comment and does not write it.

These commands must be written at the beginning of a line and the rest of the line will serve for their modifiers. No additional text should be written.

- ***loop**, ***end**. They are declared for the use of loops. A loop begins with a line that starts with `*loop` (no matter the letters case, as for the rest of commands) and contains another word to express the variable of the loop. There are some lines in the middle that will be repeated depending on the values of the variable and whose parameters will keep on changing through the iterations, if necessary. Finally, a loop will end with a line that finishes with `*end`. After `*end` the user may write any kind of comments in the same line.

  The variables that are available for `*loop` are the following:

  - **elems**, **nodes**, **materials**, **intervals**, **localaxes**. They mean, respectively, that the loop will iterate over the elements, nodes, materials intervals or local axes systems. The loops can be nested among them. The loop over the materials will iterate only over the effectively assigned materials to an entity, in spite of the fact that more materials had been defined. The number of the materials will begin with number 1. If a command that depends on the loop is located outside it, the number also will take, by default, the value 1.

  After the command `*loop`, if the variable is `elems` or `nodes`, the user may include a modifier `*all` or `*OnlyInCond`. The first one signifies that the iteration is going to be performed over all the entities, whereas the second implies that the iteration will only take place over the entities that accomplish the concerned condition. This condition must have been previously defined with `*set cond`. By default, it is assumed that the iteration will affect all the entities.

- ***if**, ***else**, ***endif**. These commands create the conditionals. The format is a line which begins with `*if` followed by an expression between parenthesis. This expression will be written in C-language syntax, may contain any of the single value return commands, will not begin with `*` and its variables must be defined as integers or reals (see `*format`, `*intformat`, `*realformat`), with the only exception of `strcmp` and `strcasecmp`. It can include relational as well as arithmetic operators inside the expressions. The following are valid examples of the use of the conditionals:

```
*if((fabs(loopvar)/4)<1.e+2)
*if((p3<p2)||p4)
*if((strcasecmp(cond(1),"XLoad")==0)&&(cond(2)!=0))
```

The first example is a numerical example where the condition is accomplished for the values of the loop under 400, while the other two are logical operators where the condition is accomplished when `p3<p2` or `p4` is different from 0 in the first case and when the first field of the condition is called `XLoad` (with this particular writing) and the second is not null in the second case.

If the checked condition is true, GiD will write all the lines until finding the corresponding `*else`, `*elseif` or `*endif` (`*end` is equivalent to `*endif` after `*if`). `*else` or `*elseif` are optional and require the writing of all the lines until the corresponding `*endif`, but only when the condition given by `*if` is false. If `*else` or `*elseif` exist, it must be written between `*if` and `*endif`. The conditionals can be nested among them.

The behavior of `*elseif` is identical to the behavior of `*else` with the addition of a new condition:

```
*if(GenData(31,int)==1)
  ...(1)
*elseif(GenData(31,int)==2)
  ...(2)
*else
  ...(3)
*endif
```

In the previous example, the body of the first condition (written as `1`) will be written into the data file if `GenData(31,int)` is 1, the body of the second condition (written as `2`) will be written into the data file if `GenData(31,int)` is 2 and if none of both is true, the body of the third condition (written as `3`) will be written into the data file.

Note: A conditional can also be written in the middle of a line. To do this, the user will begin another line to write the conditional by means of the command **`*\`**.

- **`*for`.** The syntax of this command is equivalent to `*for` in C-language:

```
*for(varname=expr.1;varname<=expr.2;varname=varname+1)
```

The meaning of this statement is the execution of a controlled loop, since `varname` is equal to `expr.1` until it is equal to `expr.2`, with an incremental value of 1 for each step. `varname` is any name and `expr.1` and `expr.2` are arithmetical expressions or numbers whose only restrictions are to express the range of the loop.

- **set.** This command has three purposes:
    - `*set cond`. To set a condition.
    - `*set elems`. To indicate the elements.
    - `*set var`. To indicate the variables to use.

  For all of them it is not binding the use of lower case, so `*Set` will also be valid in all the examples.

  In the case of the conditions, GiD allows the combination of a group of them via the use of `*add cond`. When a specific condition is about to be used, it must be defined first and afterwards, this will be used until the definition of another one. If this feature is performed inside a loop over intervals, the corresponding entities will be chosen. Otherwise, the entities will be those referred to the first interval.

  This is done in this way because when the user indicates to the program that a condition is going to be used, GiD creates a table that allows to know the number of entities over which this condition has been applied. It is necessary to specify if the condition takes place over the nodes or over the elements to create the table.

  So, a first example to check the nodes where there displacement constraints exist can be:

```
*Set Cond Volu-Cstrt *nodes
*Add Cond Surf-Cstrt *nodes
*Add Cond Line-Cstrt *nodes
*Add Cond Poin-Cstrt *nodes
```

  that allows the user to apply the conditions directly over any geometric entity.

  There are some modifiers available to point out particular specifications on the conditions.

  If command `*CanRepeat` is added after `*nodes` or `*elems` in `*Set cond`, one entity can be several times in the entities list. If command `*NoCanRepeat` is used, entities will be just once in the entities list. By default, `*CanRepeat` is off except for the case of one condition that have the `*CanRepeat` flag already set. A typical case not to use `*CanRepeat` would be:

```
*Set Cond Line-Constraints *nodes
```

In this case, when two lines share one endpoint, instead of two nodes in the list, only one is written. A typical case to use `*CanRepeat` would be:

```
*Set Cond Line-Preassure *elems *CanRepeat
```

in this case, if one triangle of quadrilateral has more than one face in the marked boundary, we want this element to appear several times in the elements list, one for each face. Other modifiers are used to inform the program that there are nodes or elements that can satisfy one condition more than once (for instance, a node that belongs to a certain number of lines with different prescribed movements) and that have to appear unrepeated in the data input file, or, in the opposite case, that have to appear only if they satisfy more than one condition. These requirements are achieved with the commands `*or(i,type)` and `*and(i,type)`, respectively, after the input of the condition, where `i` is the number of the condition to be considered and `type` is the type of the variable (integer or real).

For the previous example there can be nodes or elements in the intersection of two lines or maybe belonging to different entities where the same condition had been applied. To avoid the repetition of these nodes or elements, GiD has the modifier `*or`, and in the case that two or more different values were applied over a node or element, GiD only would consider one, being this value different from zero. The reason for that can be easily understood looking at the following example. Considering the previous commands transformed as:

```
*Set Cond Volu-Cstrt *nodes *or(1,int) *or(2,int)
*Add Cond Surf-Cstrt *nodes *or(1,int) *or(2,int)
*Add Cond Line-Cstrt *nodes *or(1,int) *or(2,int)
*Add Cond Poin-Cstrt *nodes *or(1,int) *or(2,int)
```

where `*or(1,int)` means the assignment of that node to the considered ones satisfying the condition if the integer value of the first condition's field is different from zero (`*or(2,int)` means the same assignment if the integer value of the second condition's field is different from zero). Let us imagine that a zero in the first field implies a restricted movement in the direction of the X-axis and a zero in the second field implies a restricted movement in the direction of the Y-axis. If a point belongs at the same time to an entity whose movement in the direction of the X-axis is constrained, whereas its movement in the direction of the Y-axis is released and to an entity whose movement in the direction of the Y-axis is constrained, whereas its movement in the direction of the X-axis is released, GiD would join both conditions in that point, appearing as a fixed point in both directions and as a node satisfying the four expressed conditions would be counted only once.

The same considerations explained for adding conditions through the use of `*add cond` apply to the elements with the only change of putting `*add elems`. Moreover, sometimes it can be of interest to remove sets of elements from the assigned ones to the specific conditions. This can be done with the command `*remove elems`. So, for instance, GiD allows combinations of the type:

```
*Set Cond Dummy *elems
*Set elems(All)
*Remove elems(Linear)
```

to indicate that all dummy elements apart from the linear ones will be considered, as well as:

```
*Set Cond Dummy *elems
*Set elems(Hexahedra)
*Add elems(Tetrahedra)
*Add elems(Quadrilateral)
*Add elems(Triangle)
```

The format for `*set var` differs from the syntax for the other two `*set` commands. Its syntax is as follows:

```
*Set var varname = expression
```

where `varname` is any name and `expression` is any arithmetical expression, number or command, having the latter to be written without ***** and having to be defined as `Int` or `Real`. The following are valid examples for these assignments:

```
*Set var ko1=cond(1,real)
*Set var ko2=2
*Set var S1=CondNumEntities
*Set var p1=elemsnum()
```

- ***intformat**, ***realformat**, ***format.** These commands explain how the output of different mathematical expressions will be written into the analysis file. The format is a line which begins with the corresponding command (independently of the use of capital letters) and continues with the desired writing format expressed in C-language syntax between double quotes **"** signaling the start of a field by the percentage symbol **%**.

  `*format` can mix integer and real definitions, but it only affects the following line, whereas the integer definition of `*intformat` and the real of `*realformat` remain unchanged until another definition via `*intformat` and `*realformat`, respectively, is provided. The following are valid examples:

```
*Intformat "%5i"
*Realformat "%10.3e"
*format "%10i%10.3e%10i%15.6e"
```

EXECUTING AN EXTERNAL PROGRAM

Once a correct file (or files) for the analysis program can be written, it may be interesting to run it directly from inside GiD (see section CALCULATE).

To do so, it is necessary to create a PROBLEMNAME.bat in the problem type directory. This must be a shell script that can contain any type of information and that will be different for every operating system.

If this script already exists, when choosing `Start` in the calculations window, GiD will automatically write the analysis file inside the example directory assigning the name `EXAMPLENAME.dat` to this file (if more files, names EXAMPLENAME-1.dat ... are used). Next, this shell script will be executed. GiD will assign to this script 3 arguments:

- **1st argument:** EXAMPLENAME
- **2nd argument:** c:\a\b\c\EXAMPLENAME.gid
- **3rd argument:** c:\a\b\c\PROBLEMNAME.gid

Among other utilities, this script can move or rename files and execute the process until it finishes.

The information about what is displayed when the user presses `Output view:`, is also given here. The way of doing so is to include a commented line in the script of the form: `# OutputFile: $1.log` or `rem OutputFile: %1.log` where `$1.log` means to display in that window, a file whose name is: `EXAMPLENAME.log` The name can also be and absolute name like `output.dat`. If this line is omitted, when the user presses `Output view:`, nothing will be displayed.

Note: This file must have the executable flag set (see UNIX command `chmod`) in UNIX systems.

Two examples are given of easy scripts to do so. One of them is for Unix machines and the other one is for ms-dos or Windows.

- **UNIX example:**

```
#!/bin/csh
set basename = $1
set directory = $2
set ProblemDirectory = $3

cd $directory

#    OutputFile: $1.log

rm -f $basename.flavia.res

$ProblemDirectory/myprogram $basename
```

```
mv $basename.post $basename.flavia.res
```

- **MS-DOS example:**

```
set basename=%1
set directory=%2
set ProblemDirectory=%3

cd %directory%

rem    OutputFile: %1.log

del %basename%.flavia.res

%ProblemDirectory%/myprogram %basename%

move %basename%.post %basename%.flavia.res
```

TCL-TK EXTENSION

There might exist a TCL-file in the problem type directory, whose goal is to extend GiD with the TCL-TK programming language. This is called PROBLEMNAME.tcl, where PROBLEMNAME is the name that defines the problem.

With this language it is possible to add new windows or new functionalities to the program.

The structure of `PROBLEMNAME.tcl` is composed of two possible processes:

```
proc InitGIDProject { dir } {
...body(1)...
}
proc EndGIDProject {} {
...body(2)...
}
```

Note: Notice the location of the brackets.

In this diagram, `InitGIDProject`, will be called when a new project is about to be read and `EndGIDProject`, will be called when this project is about to be closed. The syntax corresponds to TCL-language and each process should have four arguments in the following manner: **proc**, **name**, **args** and **body**.

The first argument corresponds to the word `proc`, which points out the beginning, the second one corresponds to the process name (written as in the example), the third one contains the different arguments relative to each process and finally, the fourth one contains the body with the different instructions and statements and also expresses the end of the process.

For the first process, the argument `dir` is the absolute path to the

PROBLEMNAME.gid directory, which can be useful inside the routine to locate some alternative files.

Not even this word is required for the second process, so, the simplest way to prepare the start and finish of the solving processes in GiD is:

```
proc InitGIDProject { dir } {
}
proc EndGIDProject {} {
}
```

Post-processing data files

Post-processing data files are ASCII files, and can be separated into two categories:

- 3D Data Files: `ProjectName.flavia.msh` for volume mesh information and `ProjectName.flavia.bon` for surface mesh information.
- 2D Data Files: `ProjectName.flavia.dat` for 2D mesh information.

and, also, `ProjectName.flavia.dat` for results information. All of these files are compatible with the old FLAVIA/SOLAVIA files. If a project is loaded into GiD, when changing to `PostProcess` it will only look for `ProjectName.flavia.res`. Mesh information files are only used when there is no mesh information from `PreProcess`, or when it is asked to read them. A drawback of the last option is that each mesh information file can only handle one type of element.

- **ProjectName.flavia.msh**: The first file, which is named `ProjectName.flavia.msh`, should contain the information relative to the 3D volume mesh. It contains the nodal coordinates of the 3D mesh, its nodal connectivities and the material of each element. The nodal coordinates must include those on the surface mesh. If no material is supplied, GiD takes the material number equal to zero.
- **ProjectName.flavia.bon**: The second file, which is named `ProjectName.flavia.bon`, should contain the information about 3D surface sets. It can be used to represent boundary conditions of the volumetric mesh and additional surfaces (for instance, sheets, beams and shells). At least, all the mesh points supplied in `ProjectName.flavia.msh` should be present in `ProjectName.flavia.bon` at the beginning of the file.
- **ProjectName.flavia.dat**: This file contains information about 2D meshes. And only can be used if none of the two above are used. It should specify nodal coordinates of the meshes, its connectivities (elements) and, if desired, its material number (if not specified, GiD takes to be 0).

- **ProjectName.flavia.res**: The third file, which is named `ProjectName.flavia.res`, must contain the nodal variables. GiD allows the user to define as many nodal variables as desired, as well as several steps and analysis cases (limited only by the memory of the machine).

The files are created and read in the order that corresponds with the natural way of solving a finite element problem: mesh, surface definition and conditions and finally, evaluation of the nodal results. The format of the read statements is normally free, i.e. it is necessary only to separate them by spaces. Thus, the users can modify the files with any format, leaving spaces between each field and can also write out the results with as many decimals as desired. In case of error, the program warns the user about the type of mistake found.

The choice of the ASCII files is based in their portability from other computers or mainframes, where the problem analysis can be made. Binary files are not portable between different machines because of the different data storage format of each machine. (However, they are not compulsory. GiD reads all the information directly from the pre-processing files in order to gain efficiency, whenever possible).

It is always useful to check the mesh or test the results, thus, the format of the three files is to be explained.

File ProjectName.flavia.msh

Set 1: Header

The total number of lines in this set is 6. All of them are free lines for any use. This will be the case of the first five lines, which may have an information role, informing about the project name, current version, as well as extra comments that can seem useful to add. Although they can be skipped, they are kept as a particular option inside GiD (comment lines) and as an utility to comment some additional information, like the type of project, equations, conditions and others.

Note: It is advisable, as it occurs in different solver modules used by GiD, that the sixth line explains the contents of the seventh line.

Set 2: General mesh data

The total number of lines in this set is 1, composed by at least 3 integers, the 4th integer is optional:

`n_3D_mesh_elements n_3D_mesh_points n_element_type [ last_node]`

where:

- `n_3D_mesh_elements` = number of mesh elements.

- `n_3D_mesh_points` = number of mesh points.
- `n_element_type` = type of elements.
- `last_node` = number of the last node and required if nodes are not between 1 and `n_3D_mesh_points`.

The third parameter is used by the program to recognize what kind of finite element is being used. To do this in a standard way, GiD considers the following finite element types:

- number 1 corresponds to a hexahedra with eight nodes.
- number 3 corresponds to a tetrahedra with four nodes.

Set 3: Free line for any use

The total number of lines in this set is 1, which is a free line for any use, though most modules inside GiD write here the word 'Coordinates' to point the meaning of the following lines.

Set 4: Coordinates

The total number of lines in this set is `n_3D_mesh_points`, one for each nodal point, composed by 1 integer plus 3 reals numbers:

```
i x_coord[i] y_coord[i] z_coord[i]
```

where:

- `i` = node number.
- `x_coord[i]` = x_coordinate of the node number `i`.
- `y_coord[i]` = y_coordinate of the node number `i`.
- `z_coord[i]` = z_coordinate of the node number `i`.

All the points of the meshes of the domain have to appear in this file.

Set 5: Free line for any use

The total number of lines in this set is 1, which is a free line for any use. The same comments used for set number 3 are valid here, with the change of including the word 'Connectivities' instead of 'Coordinates'.

Set 6: Connectivities

The total number of lines in this set is `n_3D_mesh_elements`, composed by 1 integer plus `n_nodes/element` integers and 1 optional integer more:

`j node[j][1] node[j][2] ... node[j][n_nodes/element] mat[j]`

where:

- `j` = element number.
- `node[j][1]` = node number 1 for the element number `j`.
- `node[j][2]` = node number 2 for the element number `j`. ...
- `node[j][n_nodes/element]` = last node number for the element number `j`.
- `mat[j]` = material index of the element number `j`.

The nodal connections must follow some specifications, so, for each tetrahedral element with four nodes, the rule is that the first three nodes that form a triangular face must be so sorted in order to define a normal which points towards the semi space containing the fourth node.

The vector `mat[j]` holds the material index of the element number `j`.

File ProjectName.flavia.bon

Set 1: Header

The total number of lines in this set is 6. All of them are free lines for any use. All the comments relative to the header of `ProjectName.flavia.msh` remain valid for the current file `ProjectName.flavia.bon`.

Note: It is advisable, as it occurs in different calculation modules included in GiD, that the sixth line explains the contents of the seventh line. **Set 2: General boundary data**

The total number of lines in this set is 1, composed by at least 3 integers, the 4th integer is optional:

`n_bound_elements n_bound_points n_element_type` [`last_node`]

where:

- `n_bound_elements` = number of boundary elements.
- `n_bound_points` = number of boundary points.
- `n_element_type` = type of elements.
- `last_node` = number of the last node and required if nodes are not between 1 and `n_bound_points`.

For the third parameter, GiD considers the following finite element types:

- number 7 corresponds to a triangle with three nodes.
- number 9 corresponds to a quadrilateral with four nodes.
- number 11 corresponds to a line with two nodes.

Set 3: Free line for any use The total number of lines in this set is 1, which is a free line for any use, though most modules inside GiD write here the word 'Coordinates' to point the meaning of the following lines.

Set 4: Coordinates The total number of lines in this set is `n_bound_points`, one for each nodal point, composed by 1 integer plus 3 reals:

```
i x_coord[i] y_coord[i] z_coord[i]
```

where:

- `i` = node number.
- `x_coord[i]` = x_coordinate of the node number `i`.
- `y_coord[i]` = y_coordinate of the node number `i`.
- `z_coord[i]` = z_coordinate of the node number `i`.

All the points of the domain have to appear in this file, what includes all the mesh points introduced in `ProjectName.flavia.msh` at the beginning. Once all the volumetric mesh had been introduced, it is possible to add surfaces that belong to a boundary of the domain but do not belong to a volumetric mesh and by this reason they will not appear in `ProjectName.flavia.msh` and only in `ProjectName.flavia.bon`.

Set 5: Free line for any use The total number of lines in this set is 1, which is a free line for any use. The same comments used for set number 3 are valid here, with the change of including the word 'Connectivities' instead of 'Coordinates'.

Set 6: Connectivities The total number of lines in this set is `n_bound_elements`, composed by 1 integer plus `n_nodes/element` integers and 2 optional integers more:

```
j node[j][1] node[j][2] ... node[j][n_nodes/element] set[j]
```

where:

- `j` = element number.
- `node[j][1]` = node number 1 for the element number `j`.
- `node[j][2]` = node number 2 for the element number `j`. ...
- `node[j][n_nodes/element]` = last node number for the element number `j`.

- `set[j]` = number of set to which the element number `j` belongs.

The vector `set[j]` allows to distinguish groups of elements in different sets. It applies, for instance, in the case of defining the different conditions that the element fulfills.

File ProjectName.flavia.dat

Set 1: Header

The total number of lines in this set is 6. All of them are free lines for any use. The first five lines, which may have an information role, informing about the project name, current version, as well as extra comments that can seem useful to add. Although they can be skipped, they are kept as a particular option inside GiD (comment lines) and as an utility to comment some additional information, like the type of project, equations, conditions and others.

Note: It is advisable, as it occurs in different solver modules used by GiD, that the sixth line explains the contents of the seventh line. **Set 2: General mesh data**

The total number of lines in this set is 1, composed by at least 3 integers, the 4th integer is optional:

`n_2D_mesh_elements` `n_2D_mesh_points` `n_element_type` [`last_node`]

where:

- `n_2D_mesh_elements` = number of 2D mesh elements.
- `n_2D_mesh_points` = number of 2D points.
- `n_element_type` = type of elements.
- `last_node` = number of the last node and required if nodes are not between 1 and `n_2D_mesh_points`.

The third parameter is used by the program to recognize what kind of finite element is being used. To do this GiD considers the number of nodes that the finite element type uses. So,

- number 2 corresponds to a line with two nodes.
- number 3 corresponds to a triangle with four nodes.
- number 4 corresponds to a quadrilateral with four nodes.
- number 6 corresponds to a triangle with six nodes.
- number 8 corresponds to a quadrilateral with eight nodes.

- number 9 corresponds to a quadrilateral with nine nodes.

Set 3: Free line for any use The total number of lines in this set is 1, which is a free line for any use, though most modules inside GiD write here the word 'Coordinates' to point the meaning of the following lines.

Set 4: Coordinates The total number of lines in this set is `n_2D_mesh_points`, one for each nodal point, composed by 1 integer plus 3 reals:

```
i x_coord[i] y_coord[i]
```

where:

- `i` = node number.
- `x_coord[i]` = x_coordinate of the node number `i`.
- `y_coord[i]` = y_coordinate of the node number `i`.

All the points of the domain have to appear in this file, what includes all the mesh points introduced in `ProjectName.flavia.msh` at the beginning. Once all the volumetric mesh had been introduced, it is possible to add surfaces that belong to a boundary of the domain but do not belong to a volumetric mesh and by this reason they will not appear in `ProjectName.flavia.msh` and only in `ProjectName.flavia.bon`.

Set 5: Free line for any use The total number of lines in this set is 1, which is a free line for any use. The same comments used for set number 3 are valid here, with the change of including the word 'Connectivities' instead of 'Coordinates'.

Set 6: Connectivities The total number of lines in this set is `n_2D_mesh_elements`, composed by 1 integer plus `n_nodes/element` integers and 2 optional integers more:

```
j node[j][1] node[j][2] ... node[j][n_nodes/element] set[j]
```

where:

- `j` = element number.
- `node[j][1]` = node number 1 for the element number `j`.
- `node[j][2]` = node number 2 for the element number `j`. ...
- `node[j][n_nodes/element]` = last node number for the element number `j`.
- `set[j]` = number of set to which the element number `j` belongs.

The vector `set[j]` allows to distinguish groups of elements in different sets. It applies, for instance, in the case of defining the different conditions that the element fulfills.

Note: The numeration of quadratic elements is linear and not hierarchical, i.e. nodes should be specified counterclockwise, without jumping internal nodes.

File ProjectName.flavia.res

This file is a complete list of the dumped results, where each result will be organized as follows:

Set 1: Header. Results description

The total number of lines in this set is 1, composed by 1 character string, 1 integer, 1 real, 1 optional character string what depends on the first integer, plus 3 integers:

`descr_menu load_type step_val [load_desc] data_type data_loc desc_comp`

where:

- `descr_menu` = results title that will appear on the menus (maximum 15 characters without any blank spaces inside).
- `load_type` = type of analysis effectuated to obtain this result: 1 - time analysis (Time Step). 2 - load analysis (Load Step). 3 - frequency analysis (Frequency). 4 - user defined analysis (User Step).
- `step_val` = number of steps inside the analysis.
- `load_desc` = description, without any blank spaces inside, of the analysis that will appear on the menus. This field must only be specified when the analysis is defined by the user (`load_type` = 4).
- `data_type` = kind of results: 1 - scalar. 2 - vector. 3 - matrix.
- `data_loc` = position of the data: 1 - on the nodes. 2 - on the Gauss points.
- `desc_comp` = specification of the existence of a description of each component that will be displayed as a menu's button: 0 - no description (inside GiD, the program itself creates the description for the corresponding components). 1 - there will be a description, without any blank spaces inside, of the components, with one component per line.

Set 2: Description of the components

The description of each one of the result's components, without any blank spaces inside, should be described here if needed, one per line. The number of lines will be as follows:

- One line if it is a scalar.

- Three lines if it is vector.
- Six lines if it is a matrix.

This description will appear in different menus to select the variable to be displayed at each stage.

Note: GiD also supports 2D results types, so description components can be two for vectors, and three or four for matrix and plane strain analysis, respectively.

Set 3: Results

The total number of lines in this set is the total number of points if `data_loc` = 1 or the total number of elements multiplied by the number of Gauss points per element if `data_loc` = 2. The definition of the results is itemized below.

- If it is a scalar. Each line is composed by one integer plus one real number: `i result[i]` where:
    - `i` = node or Gauss point number.
    - `result[i]` = value of the result on the node or Gauss point number `i`.
- If it is a vector. Each line is composed by 1 integer plus 3 reals: `i result_x[i] result_y[i] result_z[i]` where:
    - `i` = node or Gauss point number.
    - `result_x[i]` = value of the x_component of the result on the node or Gauss point number `i`.
    - `result_y[i]` = value of the y_component of the result on the node or Gauss point number `i`.
    - `result_z[i]` = value of the x_component of the result on the node or Gauss point number `i`.
- If it is a matrix. Each line is composed by 1 integer plus 6 reals: **Note:** GiD also supports 2D results types, so vector type results can be specified only by its `x_component` and `y_component`. `i result_Sxx[i] result_Syy[i] result_Szz[i] result_Sxy[i] result_Syz[i] result_Sxz[i]` (this ordering corresponds to the numeration through the diagonals of the upper triangular matrix that gives a maximum speed optimization) where:
    - `i` = node or Gauss point number.
    - `result_Sxx[i]` = value of the xx_component of the result on the node or Gauss point number `i`.

- `result_Syy[i]` = value of the yy_component of the result on the node or Gauss point number `i`.
- `result_Szz[i]` = value of the zz_component of the result on the node or Gauss point number `i`.
- `result_Sxy[i]` = value of the xy_component of the result on the node or Gauss point number `i`.
- `result_Syz[i]` = value of the yz_component of the result on the node or Gauss point number `i`.
- `result_Sxz[i]` = value of the xz_component of the result on the node or Gauss point number `i`.

Note: GiD also supports 2D results types, so matrix type results can be specified only by its `xx_component`, `yy_component`, `xy_component` and the fourth optional `zz_component` for plain strain matrices.

To include the Gauss points in the results, they must be treated as if they were a type of result, but:

- they must be inserted at the beginning of the file,

- even though the structure does not vary, the headers meanings at set number 1 change.

Set 1: Header. Gauss points

The total number of lines in this set is also 1, but it is composed always now by one character string, one integer, one real plus three integers:

```
descr_menu load_type step_val data_type data_loc desc_comp
```

where:

- `descr_menu` will not be used.
- `load_type` = 0, to indicate that they are Gauss points.
- `step_val` = number of Gauss points per element. This must be constant for the whole geometry.
- `data_type` = indication whether the Gauss points will be described below, or they must be calculated by GiD, thus, its listing can be dropped: 0 - the Gauss points will be the ones which are described below, 1 - the program must calculate the Gauss points.
- `data_loc` does not matter, but it must be specified.

- `desc_comp` does not matter, but it must be specified.