9th GiD Convention

GiD advanced courses

GiD advanced courses

1 Geometry importation and CAD cleaning operations 4
2 Meshing advanced features 22
3 Customization 40
4 Advanced visualization tools 46
5 Working with large models 54
6 Working with terrain models 75
7 Macros 84

GiD advanced courses

Introduction

These courses are prepared to be followed using the version 14.0 of GiD. They are focused on advanced features of GiD, directed to an advanced user profile. We strongly recommend to make the GiD basic courses before these ones.

Installing GiD

To install GiD go to the GiD Convention USB unit, from now on we will assume it is '*D:*', if you have auto-run function active, the file index.html will be opened automatically, if not, please double click on 'D:\index.html'.
Choose from the installer link **MS Windows** or **Linux, 32** or **64** bits, or **macOS 64** bits, depending on your operating system (OS).
Click on the option corresponding to your OS (Windows, Linux or macOS), and then follow the instructions of GiD installer to install GiD into your computer.
Note: A x64 bits OS can run applications of x64 and x32 bits, so if you are not sure if your system is x64, please select x32.
On web page www.gidhome.com you can find two types of GiD versions:

- **Official versions of GiD** are stable versions of the program. They don't include the newest capabilities but they have passed all the validation tests.
- **Developer versions of GiD** are more modern versions of the program that have new capabilities. They have not passed the validation tests that a official version does. So, use them with care. Only download and install one of these versions if you know of a new capability very important for your needs or if you want to try the new improvements in the program.

Registering GiD

GiD can work with no licence (unregistered), but in this way the user can only manage a model with a few number of nodes (around 1,000) and surfaces.
In case you want to try using a mesh with a higher number of nodes, a free password for one month can be downloaded from the web site (or a permanent one buying a licence)

- Password for GiD: http://www.gidhome.com/purchase/passwords

NOTE: The USB memory sticks of the course are already registed with a one month password for GiD 14 (USB passwords are valid for both MS Windows, Linux and macOS platforms)

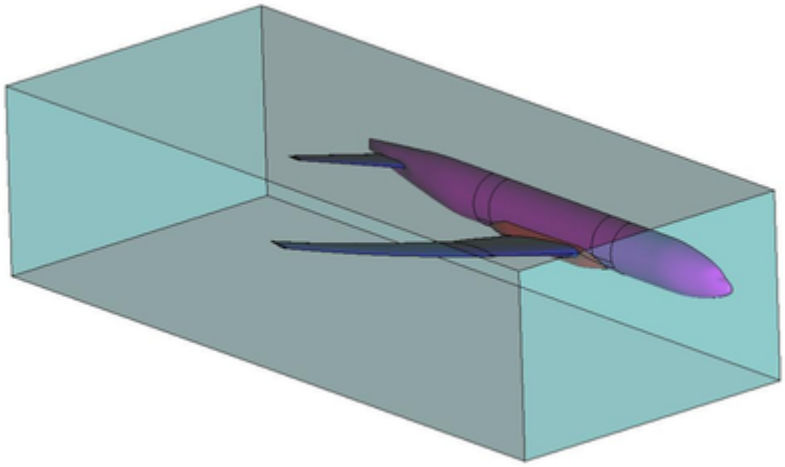
Material location

The PDF documents and most of the models of the basic and advanced courses can be found in the GiD USB memory stick.
The same PDF documents and all models used in both courses can also be found at ftp://www.gidhome.com/pub/GiD_Convention/2018, inside the corresponding folders.

Geometry importation and CAD cleaning operations

Description

This course is focused on the geometry importation and cad cleaning operations which are needed in most cases for prepare the imported model to be meshed



Final model obtained when following all the steps of the course.

We will import the geometry of half an aircraft (shown in the figure) in IGES format and we are going to follow the whole process from the importation to the meshing process of a control volume surrounding the aircraft.

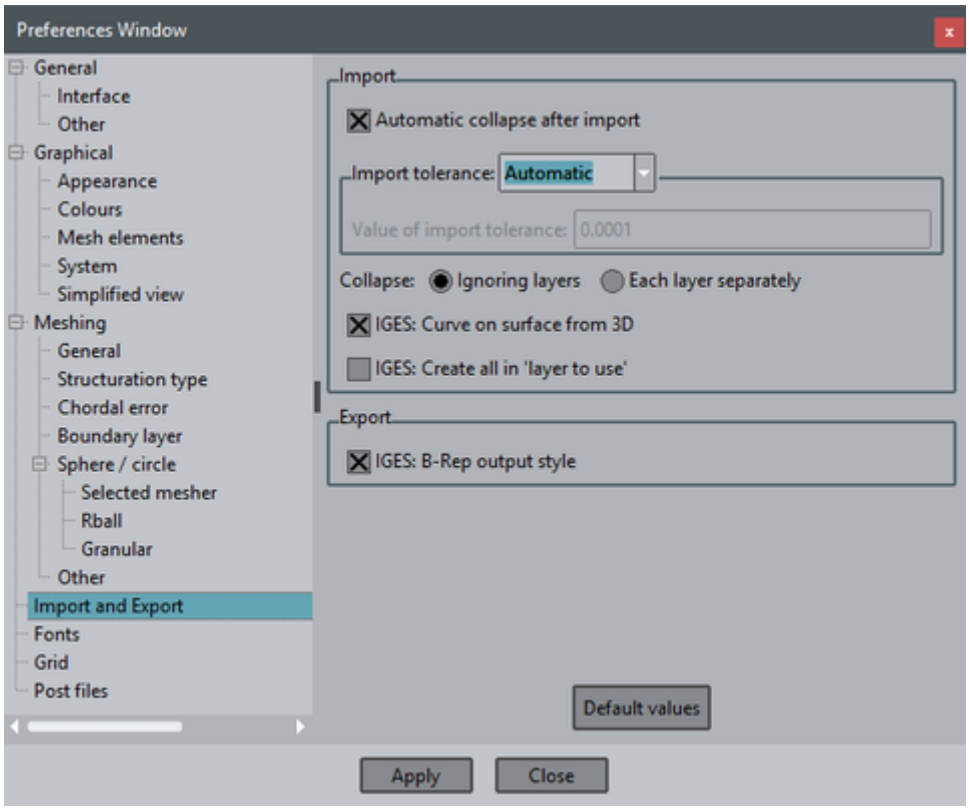
Note: the model originally was a "NASA Common Research Model" (commonresearchmodel.larc.nasa.gov)

Import of the model

GiD import and export preferences

The user can customize some parameters for the import and export of geometry. This parameters are located in the **Import and Export** part of **Preferences** window.

- Click on **Utilities->Preferences** and go to the **Import and Export** part. The window shown in figure should appear.



Import and Export preferences window.

When GiD imports a geometry (independently on the format), it automatically perform some cad cleaning and repairing operations such as check entities orientations, collapse small gaps, etc. User can avoid this automatic operations by unchecking the **Automatic collapse after import**.
If the *automatic collapse* is set, GiD can use a distance tolerance entered by the user, or can compute an automatic tolerance based on the representative distances of the model. With the option **Automatic import tolerance value** user can choose whether the tolerance is automatic or the manually specified.
We are going to set all the options shown in the figure and click **Apply**.

Import model

The IGES file to be imported is named **half_aircraft.igs** and it should be found at Material location. .

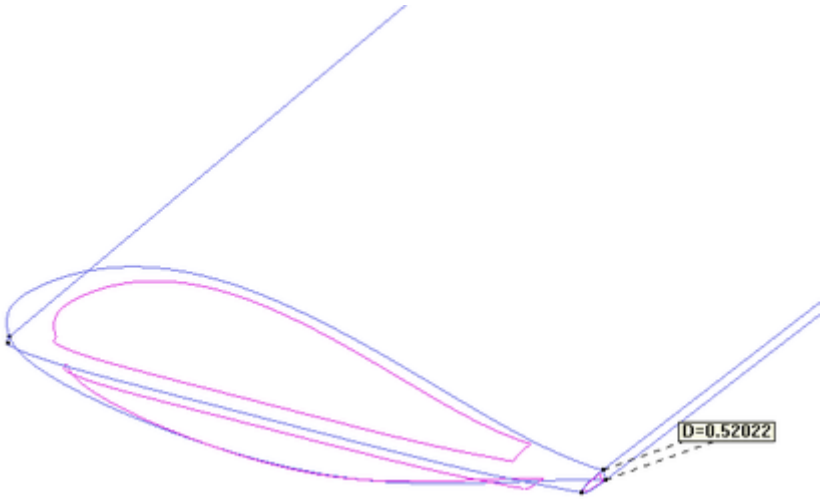
- Select **Files->Import->IGES...** and select the file **half_aircraft.igs**.

A progress bar appears showing the state of the automatic CAD cleaning operations GiD does when imports the IGES file. When the import is finished, a window should appear with some interesting information of the IGES header.

On the lower messages bar you could see the tolerance used for the automatic collapse operations, and the result of the automatic collapse on import.

Import tolerances

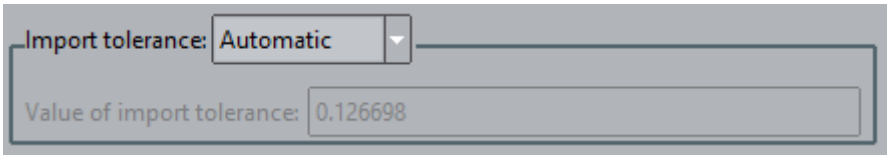
After importing a model with automatic tolerances, it is very useful to check how the model looks like, and get a minimum characteristic distance, to ensure the automatic tolerance computed by GiD does not collapse any geometric detail interesting for simulation. In this case, it should be useful to measure the distance of the ending part of the tail, as it is shown in figure.



Size of the end part of the tail.

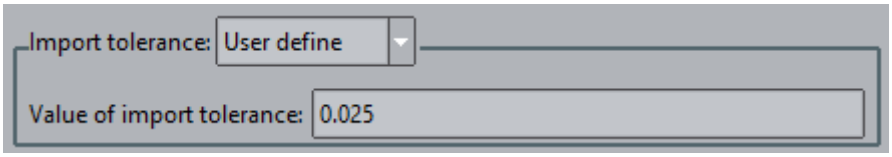
You can check it by using Utilities ->Dimension ->Create ->Distance, selecting the two points and placing the dimension label.

As it can be seen in the preferences window, now the automatic tolerance for the collapse computed by GiD is 0.126698.



Depending on the version of GiD used, this tolerance can be different, so the result of the import of the model may differ in some details.
To ensure that all the participants of the course have the same model resulting from the import, we will use a tolerance specified by the user.
Taking into account the distance shown in the previous figure, we can try for example with an *Import tolerance* of 0.025 that is smaller than the detail to preserve. For this purpose we need to import again the model:

- Select **Files->New**, and select **No** for the question **Save changes to this project**.
- Open the **Preferences** window, and go to the **Import and Export** part.
- Select from the combobox *Import tolerance:* **Used define**
- Enter **0.025** in the **Value of import tolerance** field.
- Click on **Apply** and close the **Preferences** window.



Now import again the IGES file **half_aircraft.igs** (**Files->Import->IGES...**).
Once the model is imported, save the model (this will be the model we will work with during this course). For saving the model just select **Files->Save** and choose a proper location in your computer to save the model in.

CAD cleaning operations

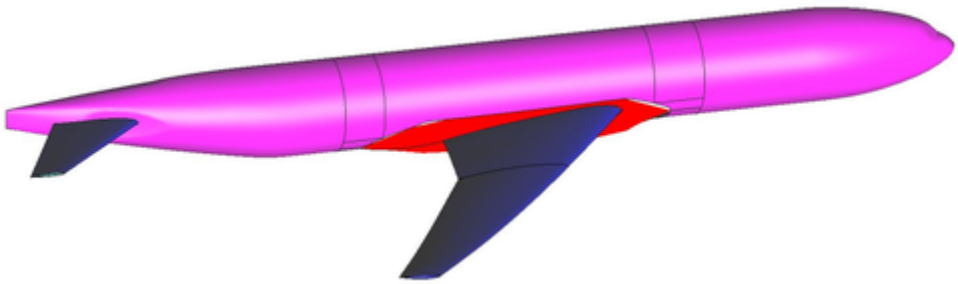
Visual check of geometry

A useful step when a file is imported is to check visually how it looks like. **Rotate** the model and set the **Render mode** to **Flat** (from the *View* menu or mouse right button menu). With this initial visual scan some possible big geometry imperfections or import errors may be detected.
If some surface is not well rendered (like strange shadows or holes in the view) or no render is done, it means that GiD had some problem to generate the render mesh of the surface. This should point a possible error in that surface (or may be not).

Trick: the GiD-Tcl procedure *GiD_Geometry list -unrendered surface* provides the list of surfaces that were not able to be rendered.
You can write this in the command line to know this list:

```
-np- W [GiD_Geometry list -unrendered surface]
```

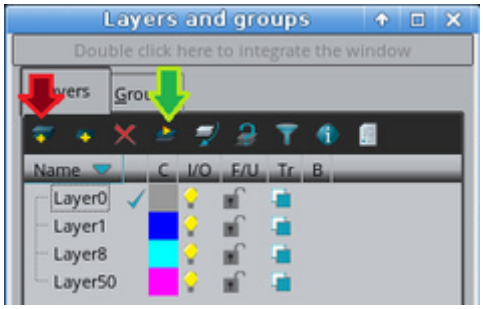
A message window will appear, but in this case it is empty, as all surfaces were rendered.
As it can be seen in the figure, the render of the surface which connects the wing of the aircraft with the main fuselage looks a bit off.



Render view of the model imported, with the surface which render is not so good highlighted.

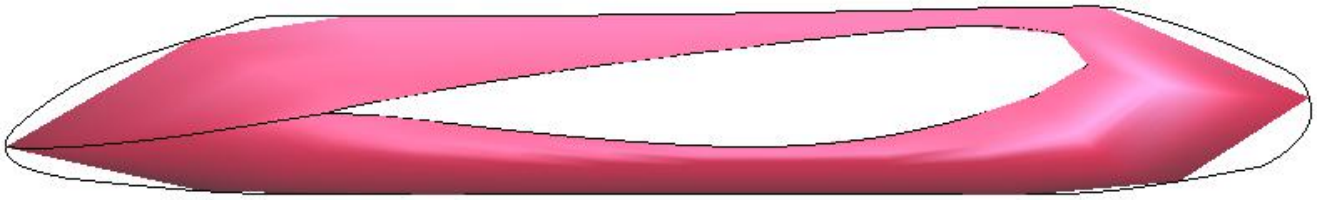
Let's check if this surface is well defined or not. A good idea is to isolate this surface in another layer to visualize it more clearly.

- Open the Layer window (Ctrl-l , (lower L) or **Utilities->Layers and groups**, or the corresponding icon  in the **Toolbar**)
- Create a new Layer (using the **New Layer** option in the mouse right button menu, or using the corresponding icon in the upper icon bar of the *Layers* window).
- Send the surface to the new layer by using the **Sent to->Surfaces** option of the mouse right button menu after the target layer is selected (the corresponding icon in the upper part of Layers window can be also used).



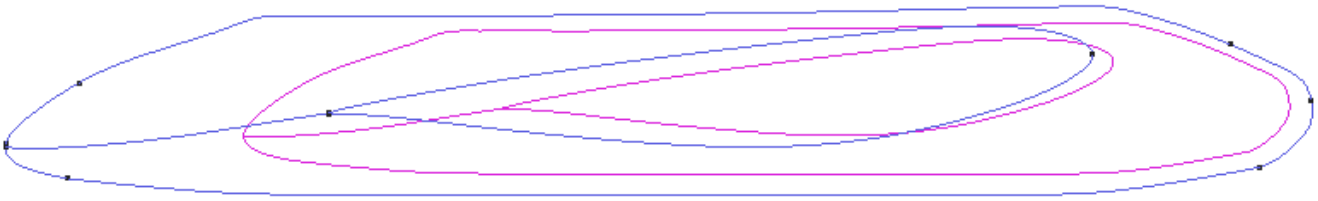
'New layer' and 'Send to' icons.

Now we can switch off all the layers except the one in which the surface is. We should be able to see just this surface, as shown in the figure.



View of the problematic surface isolated.

We can change the render mode between **Normal** and **Flat** to understand better the definition of the surface:

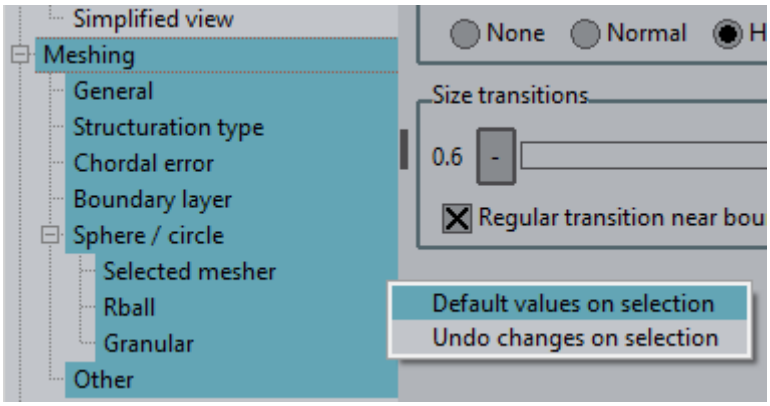


'Render normal' view

Generate automatic test mesh

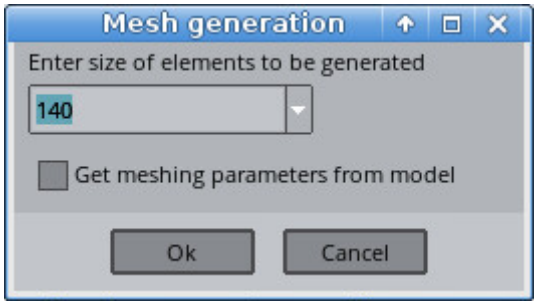
Another useful check that can be done when importing the geometry is to generate a first mesh, using the default meshing preferences of GiD. Often, some detail of the model which can be skipped during the visual scan is detected when looking at the mesh, for instance, possible concentration of elements, entities which cannot be meshed, etc.

- Open the **Preferences** window select all meshing childs and with right mouse click select **Default values on selection** from the pop-ip menu and press **Apply**.



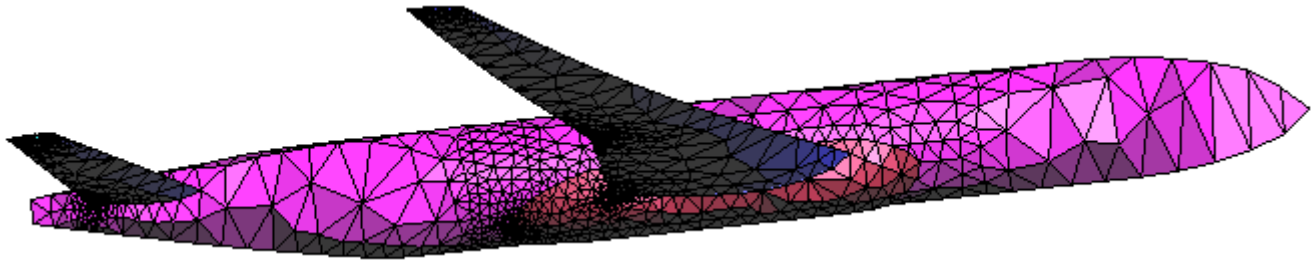
Set default values with multiple selection

- Then generate the mesh by pressing Ctrl-g or by selecting **Mesh->Generate mesh...** and click **OK** using the mesh size proposed by GiD (in this case, **140**) and unchecking the *Get meshing parameters from model* option:



General mesh size window

As it can be seen in the figure GiD can generate a mesh on all the surfaces.



Coarse mesh

In this step is not important to check the mesh quality, but to check where the elements are concentrated, and if there are surfaces which cannot be meshed.

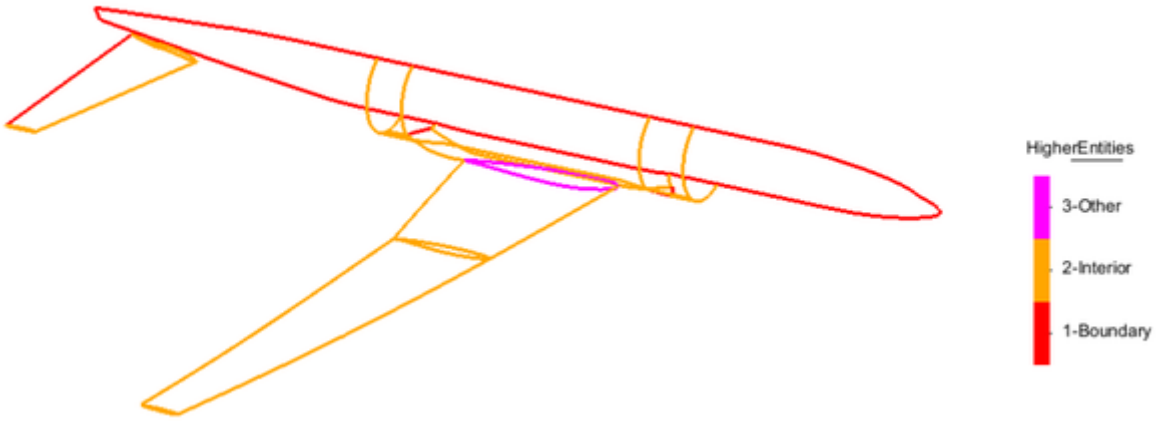
Looking for gaps

One of the typical problems when importing geometry is to have small gaps or overlapped surfaces in a model where they should not appear. An easy way of checking the topology of the model using GiD is to see graphically the **higher entity number** of the geometrical entities. The **higher entity number** of an entity represents the number of entities (with one more dimension) containing itself (for a given line, for example, the higher entity number represents the number of surfaces owning this line).
In this case, for example, as we need the geometry of the aircraft to be a boundary of a volume (the air surrounding the aircraft), we need it to be water-tight. This implies that all the inner lines of the aircraft should have higher entity number equal to 2.

- Return to geometry mode, if you are on mesh mode, by clicking at **View->Mode->Geometry**, or the switch mode icon

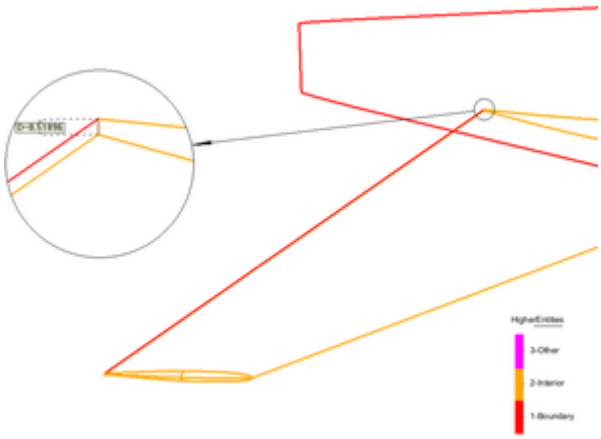
Toolbar button to switch geometry<->mesh view

- Select **View->Higher entities->Lines**.



Higher entities of lines in the model

As it can be seen in the previous figure, there are some inner lines with *higher entity number* equal to 3, and some other with *higher entity number* equal to 1. These last ones evidence the presence of some gap in the geometry.
We will study in more detail the gap of the tail (the red lines that belong to only 1 surface instead 2 as expected).



Gaps with duplicated overlapped lines.

You can check it with Utilities ->Distance or Utilities ->Dimension ->Create Distance

If we list the entities with **Utilities->List->Lines** or the corresponding icon:

Selecting a square on the red line, we obtain that there are two lines.

Collapse model

Taking into account that the measured distance on the tail is about 0.5 units, we can try to collapse the whole model with a higher tolerance than the used in the import process (remember that was 0.025), for instance, a 10% of the measured distance, to join entities closer that this value.

- Open the **Preferences** window, go to the **Import and Export** part, and change the **Value of import tolerance** to **0.05**.
- Click on **Apply** button and then you can close the Preferences window.

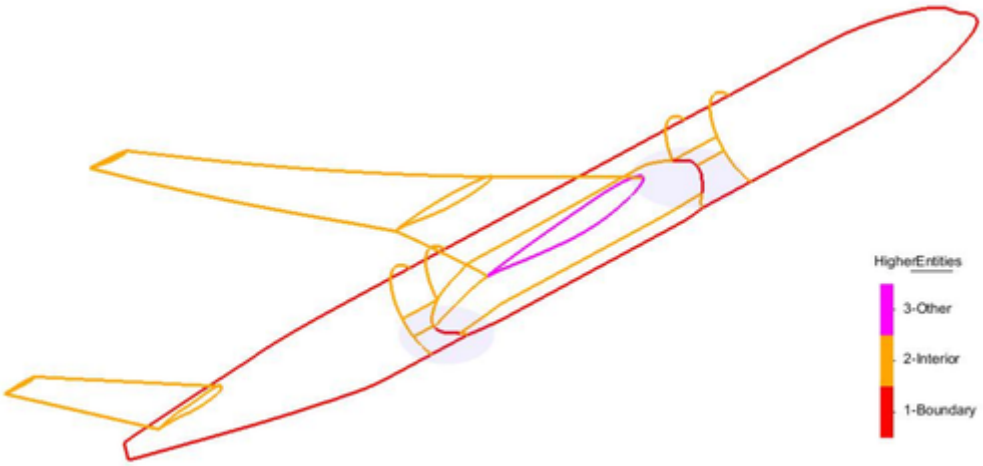
Note: This import tolerance is used both when importing a model, and when collapsing entities of a model already inside GiD.

- Select **Geometry->Edit->Collapse->model** and click **OK** on the appearing window to allow the operation of collapsing the model.

A message in the lower messages bar show this:

```
Collapse model. Created -3 points, -4 lines, 0 surfaces, 0 volumes. Tolerance=0.05
```

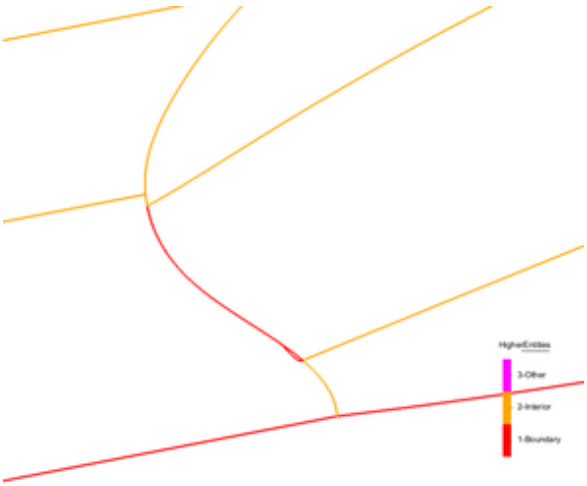
This mean that one point and three lines closer than 0.05 units were deleted.
As you can see if you view again the higher entities of lines, now the gap in the rear part of the aircraft has disappeared. Now we only have the two gaps highlighted with a shadow region in the figure.



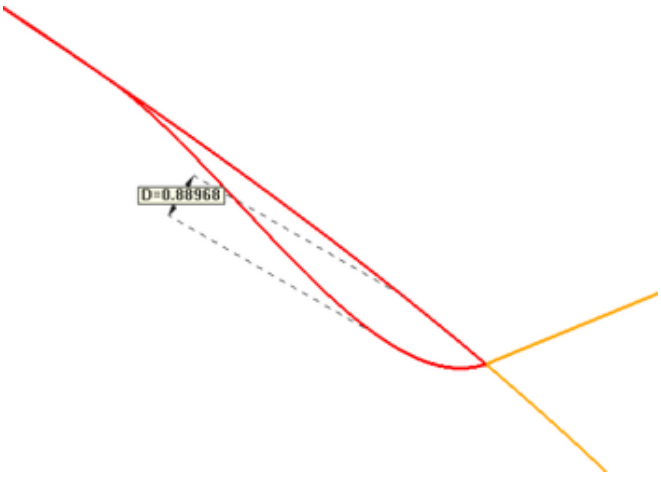
Higher entities of lines

Local collapsing

To try to repair the gaps which remains in the model, one option should be to increase more the tolerance and try to collapse again the model, but the tolerance is approaching the measured minimum characteristic size of the model, and some important details, such as the ending part of the wings, would be collapsed too, and this could be undesirable.
We are going to collapse the lines enclosing the gaps with a higher tolerance, but only these lines (not the whole model).
In the Figure it can be seen the configuration of the lines before the local collapsing.
It is possible to measure the distance between both lines with **Utilities->Distance** and selecting in the contextual mouse menu the **Point In Line** option and picking two points on the lines. This distance is about 1.0 units, to force a join between these two lines, it is necessary to set a bigger value as import tolerance.



Topology before the local collapse



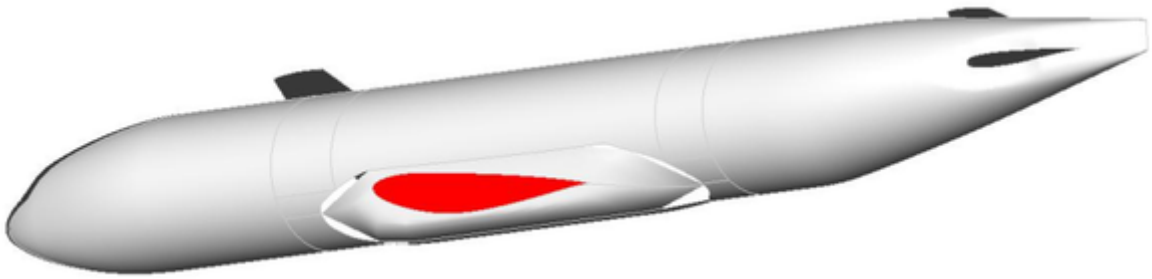
Zoom showing the maximum gap distance.
To check the distance use Utilities ->Distance *and the option* PointInLine *of the contextual menu before selecting the points.*

- Set the value of **import tolerance** on the **Import and Export** part of **Preferences** window to **1.1**, and click **Apply** before closing the window.
- Select **Geometry->Edit->Collapse->Lines** and select the two lines which surround one of the gaps (the lines with higher entity

equal to 1).
- Repeat this action on the other gaps of the model, but only selecting the wrong lines.

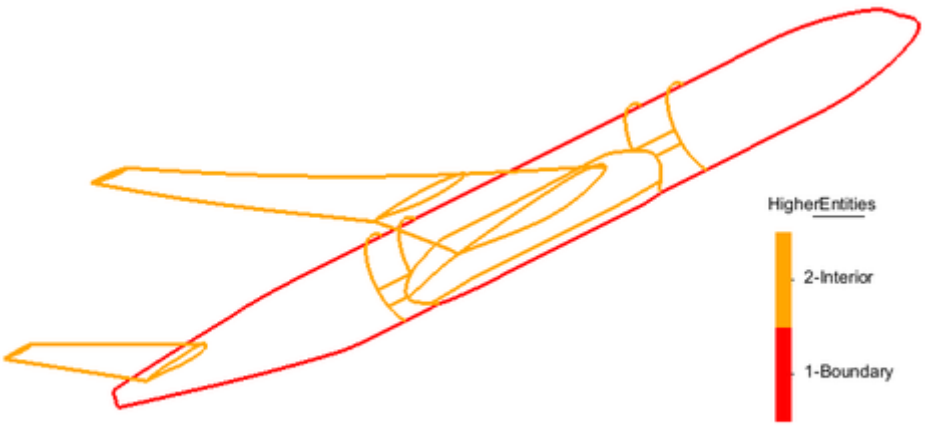
Once finished with this local collapses, in the **Import and export** child of the Preferences window set the **Import tolerance** to **Automatic**, to avoid deleting parts accidentally when collapsing with a too big tolerance.
Now, if you try to see the **higher entities** of lines, only the boundary lines of the half-aircraft should have higher entity number equal to 1. We can see also some lines with Higher entity number equal to 3. This indicates that these lines are owned by three surfaces. In this case, there is no error in the geometry, but we can delete the inner surface which connects the wing with the fuselage (see figure below), because it is not of interest for the process we want to do.

- **Geometry->Delete->Surfaces**



Surface to be deleted because it is not of interest for the simulation

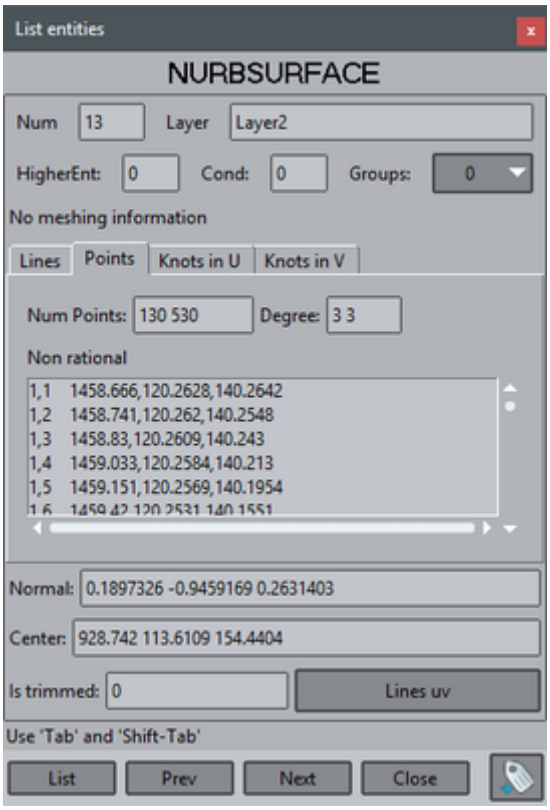
In next figure it is shown the higher entities of lines after all the cleaning operations performed until now: you can see all the lines have higher entity equal to 2, except the lines of the boundary of the half-aircraft.



Higher entities of lines after the local collapsing and after deleting the unnecessary surface

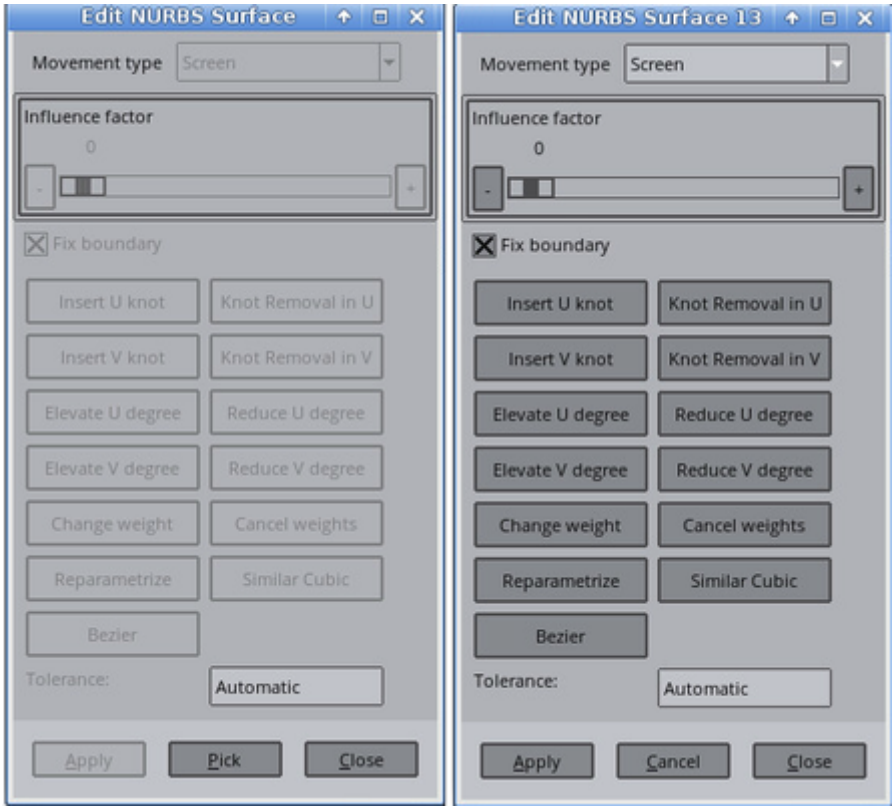
NURBS simplifying

The surface number 13 has a lot of control points for its geometrical definition. This surface is the problematic surface shown at the begining of this document.
If we list this entity with **Utilities->List->Surfaces** it take a long time to show this information window:



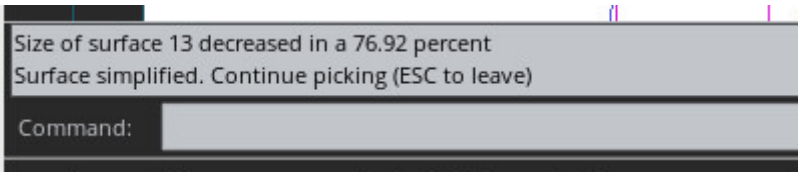
Geometrical information of NURBS surface

We can see that its control polygon has 130 x 530 = 68,900 control points. This situation will lead to a very heavy model in terms of amount of memory used to deal with it, and often this NURBS definition can be simplified without loosing accuracy for capturing the shape of the surface.
To do it open **Geometry->Edit->Edit NURBS->Surface...** and pick the surface:



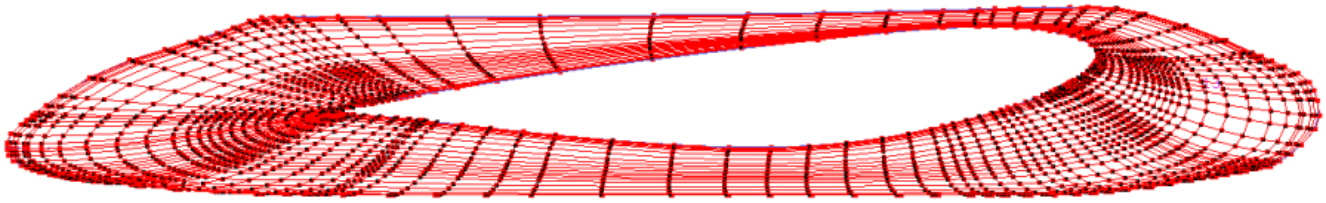
Edit NURBS window, before (left) and after (right) selecting a NURBS surface.

- When the control points are visible (after clicking **Pick** in the **Edit NURBS surface** window and selecting the surface), click on the option **Knot removal in U**. You can see that the number of control points have decreased, and in the warn line (lower part of GiD window) a message shows the memory reduction needed to store the surface.



- Click on the options **Knot removal in U** and **Knot removal in V** as many times as needed, until the message in the warn line indicates that the surface is not simplified any more (decreased in a 0.00 percent).
- Click on **Apply** to accept the NURBS simplification and Close the window.

As it can be seen in next figure, now the surface is defined with much less control points than the original one, and now it is easier to check how the control points distribution looks like.



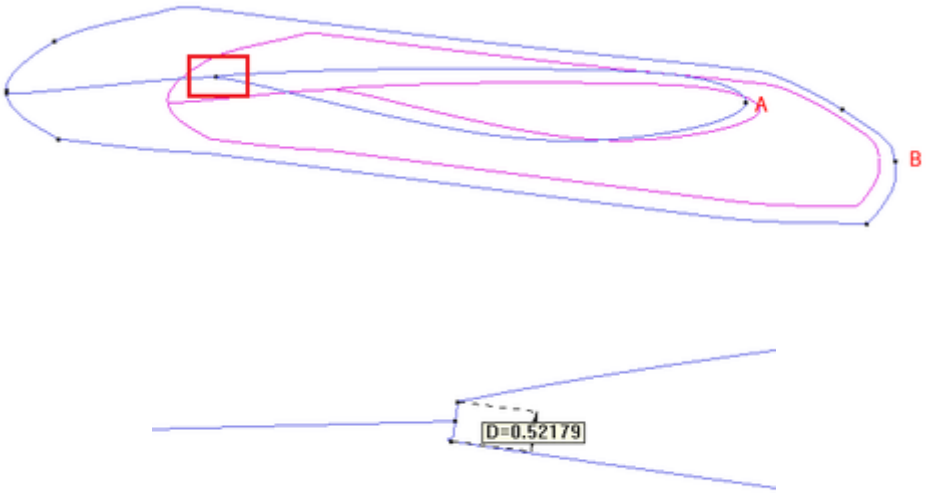
Control points of the surface once it is simplified

Listing the surface again we can see that its new control polygon has 24 x 117 = 2,808 control points. These numbers are indicative as they depend on the order of execution of the **Knot removal in U** and **Knot removal in V** operations.

Problematic surface

Also after the simplification the render mesh of the surface 13 is not very good. There are two main difficulties for the meshing algorithm of the surface:

1. There are small lines related to the mesh size (the border of the wing has about 0.5 units, and the general mesh size asked was 140 units)



The red square shows the zoomed area at the read border of the wing.

To avoid having small elements the geometry could be modified, collapsing with a size greater to 0.5 to delete the very thin surface of the back of the wing, and convert it in a single line. Then the mesh could have much less elements, but maybe the shape of this part is important for the simulation and must not be changed.

2. The surface is 'closed' because it is topologically like a ring. The parametric surface curves $v = 0$ and $v = 1$ when mapped by the parametric surface become the same 3D curve, and the inverse of this parametrization is not unique (a 3D point could be mapped in two different parametric space points).

We will try to aid the mesher by dividing the surface in two parts in the middle of the **v** parametric direction.

To do so:

- Use the tool **Geometry->Edit->Divide->Surfaces->Num. divisions**

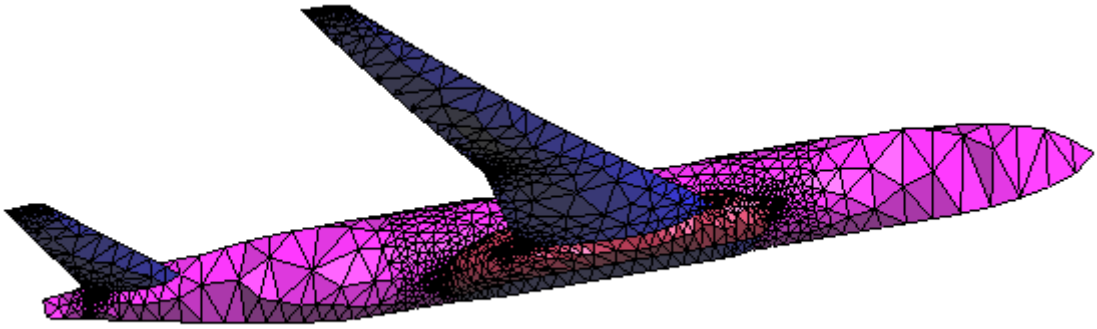
- Select the surface, click on **V sense** and enter **2** on number of divisions, **Ok**

After this division they appear some extra problem of higherentities connecting the new surfaces with the rest of the model.

Higher entities of lines

To fix the problems, switch all layers on, use the collapse lines tool locally again, and increase the tolerance as necessary to join the similar lines.

Let's mesh the model:



Coarse surface mesh

Create volume

We are going to create the control volume surrounding the half-aircraft in this section. This control volume will represent the air surrounding the aircraft.

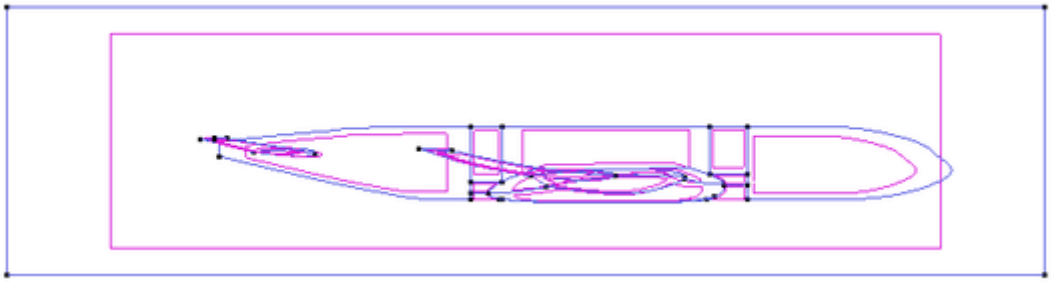
For this purpose, using the *Line* creation tool  , create the rectangle with vertex in the points:

-220,0,745
-220,0,-155
3280,0,-155
3280,0,745

Note: Remember to use **Ctrl + a** to select the last existing point to close the rectangle.
Then create a surface with this created lines as their contour lines:

- Select **Geometry->Create->NURBS surface->By contour**, and select these last created lines.

At this point, the geometrical model should look like the one shown in the figure.

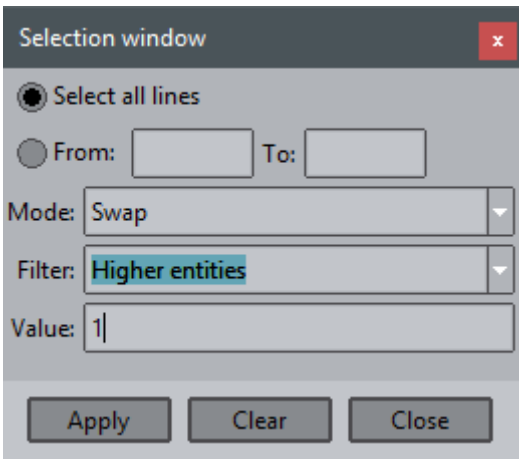


State of the model at this point

Now we should create a hole in the new rectangle, so as the inner boundary of this surface will be the contour of the half-aircraft (the lines which have higher entity equal to 1 previously).

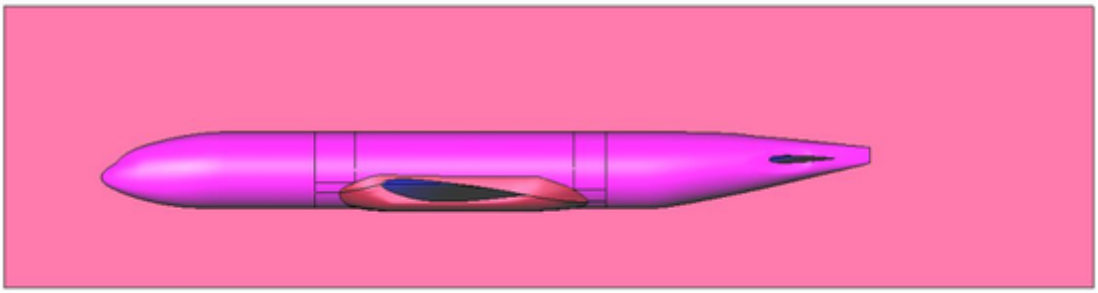
- Select **Geometry->Edit->Hole surface->With lines** and select the rectangular surface created previously.
- Then select the contour lines of the half-aircraft and press **ESC**.

Trick: To select the contour lines of the half-aircraft, you can use the **Selection window**, by selecting this option from the contextual right mouse menu.
By doing that is really easy to select all lines with higher entity 1, press apply, and unselect just the 4 lines of the rectangle:



Selection window

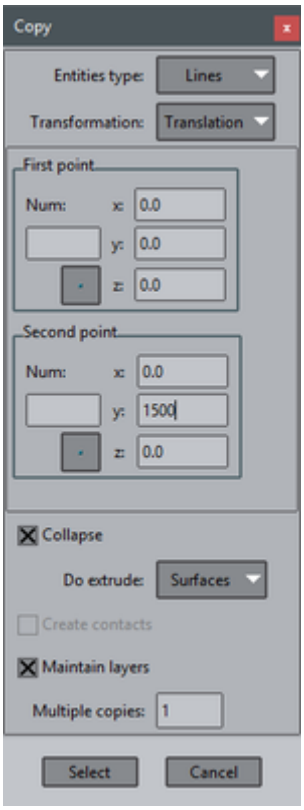
A hole in the rectangular surface should be created, like it is shown in the figure.



Rectangle holed with the lines of the plane

The hole created in the rectangular surface, using the boundary lines of the half-aircraft.
Now we are going to extrude the outer lines of the rectangle to build the control volume:

- Open the Copy window (**Utilities->Copy** or use Ctrl-C (capital C)) and set the options shown in the figure:



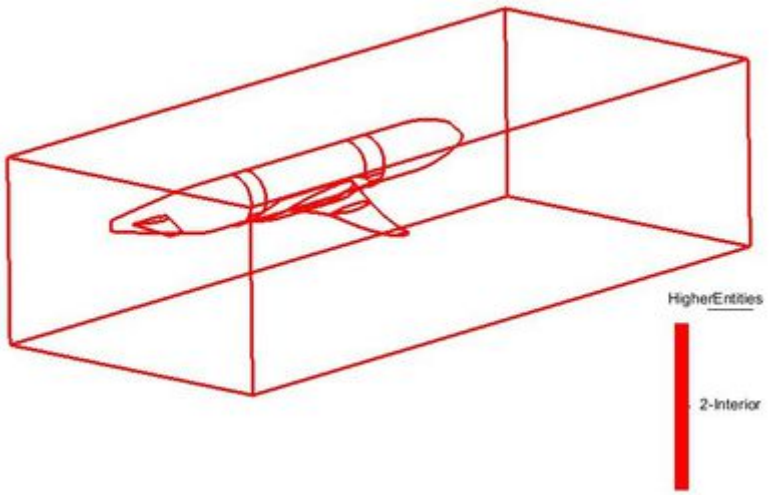
Options to be set for the extrusion of the lines

- Click on **Select,** select the four lines of the rectangle created previously, and press **ESC**. Press **Cancel** to close **Copy** window.

Now the next step is to create the surface which will close the control volume.

- Create the surface with **Geometry->Create->NURBS surface->Search**
- and click one of its four boundary lines. This tool will try to automatically find and create a surface that has this line in its boundary.

If you look at the higher entities of lines of the resulting model you should obtain that all the lines have higher entity equal to 2, which means that the patch of surfaces of the model can close a volume topologically.

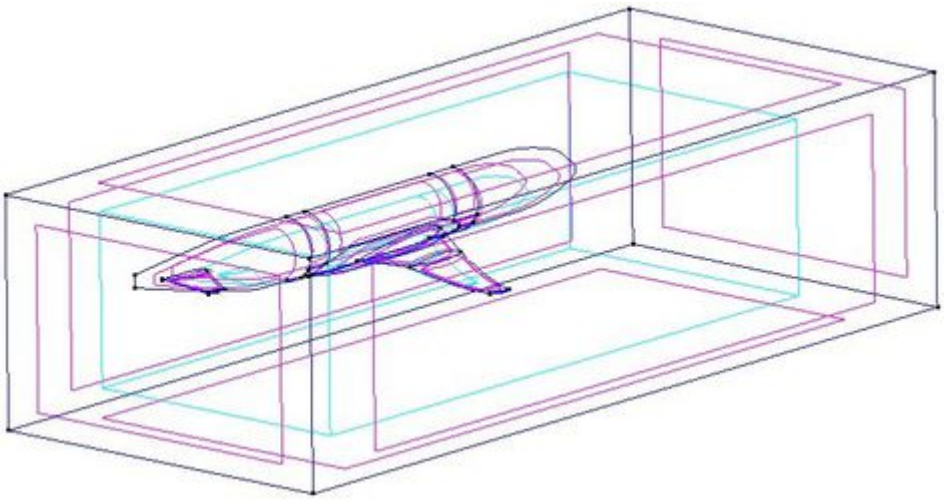


View of higher entities of lines at this point

The last step is to create the volume:

- Select **Geometry->Create->Volume->By contour**, and select all the surfaces of the model. Then press **ESC**.

Now we have finished the geometrical definition of the model, and we should have at this point the model shown in the figure (where the created volume is shown with sky-blue lines).



The geometrical model after all the import operations and the control volume creation

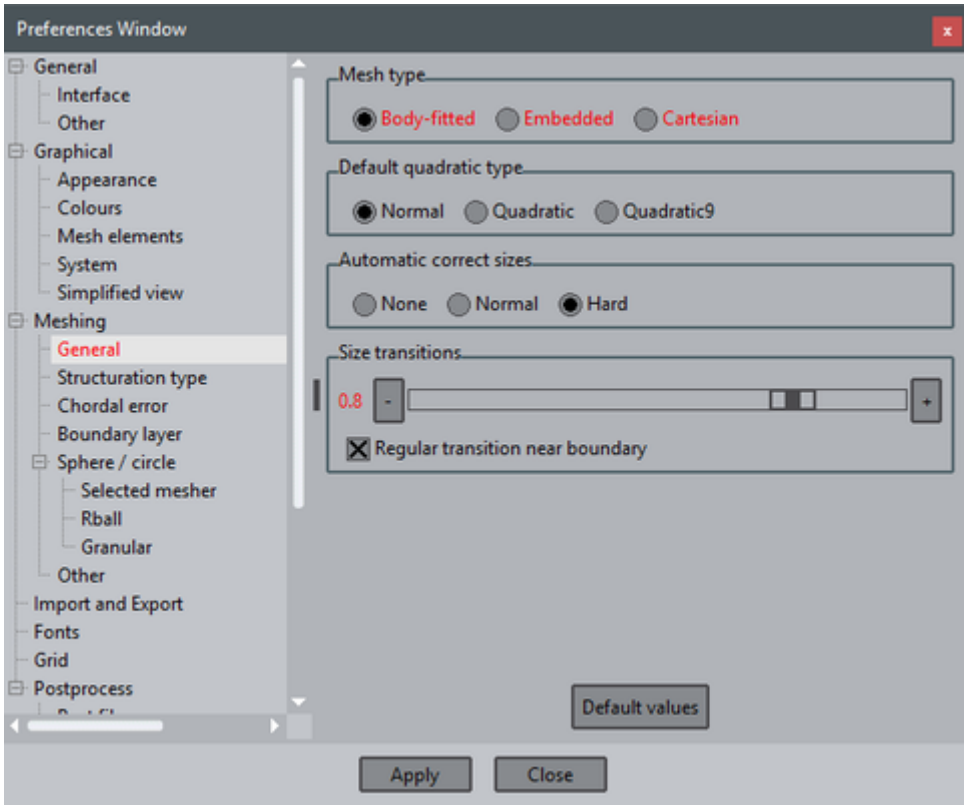
Generate final mesh

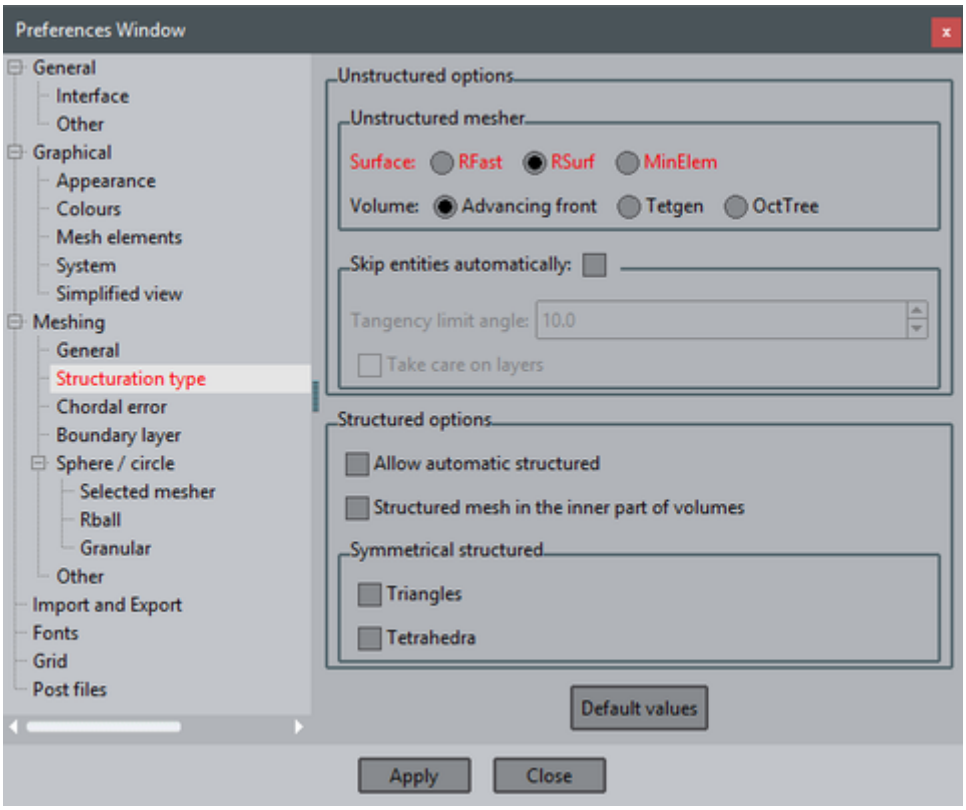
We are going to go deeper in the mesh generation process in this part. The CAD cleaning operations can be understood as all the operations needed from the moment of importing the model to the moment of runing the simulation itself. This implies to have a CAD model from which the preprocessor can generate a mesh suitable to be used in the simulation.
In this point we can establish that the geometry cleaning part has ended, and we are going to explore some of the possibilities GiD offers in the mesh properties assignment, to be able to generate the mesh.

Meshing parameters

Be sure than **Meshing->General** has **Body-fitted** mesh type.

GiD user can choose between several meshing parameters to reach the final desirable mesh for the simulation. This parameters can be checked and set in the **Meshing->Structuration type** part of the **Preferences** window (see figures).





Meshing parameters used for generating the mesh in this example

Please, set all the parameters as the ones shown in the above figures, and click **Apply**.

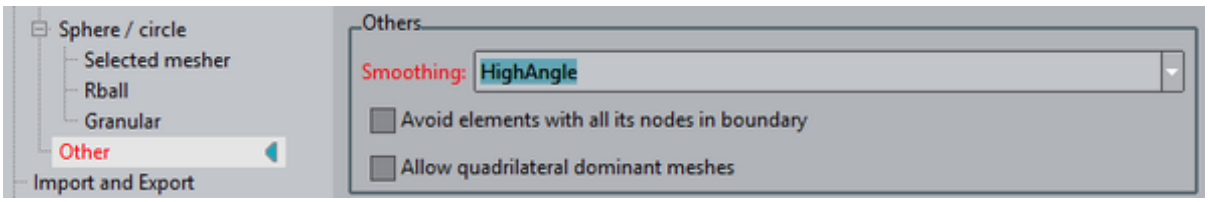
In the **GiD Help** all these variables are explained in detail. We will focus in some of them in this part, just to justify why we are usingthis ones and not others.

In the **Meshing->General** branch:

- **Mesh type:** Body-fitted is the usual kind of mesh, the mesh preserves the shape of the geometry. The other alternatives are for special solvers.

In the **Meshing->Structuration type** branch:

- **Surface mesher**: GiD uses advancing front method for the surfaces meshing, and this method can be used in two different ways: mesh in the 2D parametric space of the corresponding surface **(RFast),** or mesh the surface directly in the 3D space **(RSurf).** The **M inElem** mesher is an special mesher which generates meshes with as less elements as possible, but representing accurately the shape of the geometry according to given chordal error parameters (defined in the *Meshing->Chordar error* branch of *Preferences* wi ndow). In this case we are using **RSurf** mesher: this mesher is slower than RFast, but usualy ensures a better quality of the resulting mesh. This aspect is specially important when the surfaces are the contour of a volume, because the success in the volume meshing depends strongly on the quality of its contour surfaces' mesh.
- **Automatic correct sizes**: This is a very important parameter. If this parameter is set to **None**, GiD will only take into account the sizes assigned by the user. This may cause some problems, specially if the assignement of sizes to the geometrical entities is not very precisely done. If the parameter is set to **Normal**, GiD takes into account the sizes assigned by the user, but also applies an automatic correction. This correction basically tries to avoid very aggresive mesh size changes between close geometrical entities. GiD also tries to assign a realistic size to the entities: in some cases, the user assigns a general size (or a local size to an entity) which is much bigger than the size of the entity itself. In this cases it is impossible to generate a mesh using the size specified by the user, so GiD reduces this size. If this parameter is set to **Hard**, besides the corrections mentioned above, GiD also tries to apply a certain chordal error criteria for assigning sizes (smaller sizes to entities with bigger curvature). We will use the **Hard** option.
- **Unstructured sizes transition**: This parameter controls how the sizes transitions are done between areas where the mesh is finer and areas where the mesh is coarser: faster (values closer to 1.0) or slower (values closer to 0.0). The bigger the parameter is, the more aggresive is the transition between small elements and big ones. The correct value for this parameter depends on the simulation requirements. Now, we will use **0.8**.
- **Skip entities automatically:** These variables control whether a line or a point is skipped during the meshing process. Using this option, meshes can be generated with fewer elements than other ones because it is less dependent on the dimensions of geometrical surfaces. However, it is slower and can fail for distorted surface patches. We won't use this option now.



Smoothing: HighAngle

In the **Meshing->Other** branch:

- **Smoothing**: This parameter can be set to three different values: **Normal**, **HighAngle**, **HighGeom**. The **Normal** option performs a regular smoothing to the mesh generated. The **HighAngle** option focuses on improving the shape of mesh elements. **HighGeom** trie s to make the final mesh *closer* to the geometry, performing topological operations in order to minimize the chordal error of the elements. This last option reduces the chordal error, but has an affect on the quality of the shape of the elements. For this example we will use the **HighAngle** option.

Set the Smoothing to **HighAngle.**

First attempt in mesh generation

As it can be seen in previous figures, the default size used in the mesh generated by GiD is too coarse for any kind of simulation. With the general mesh size, provided at the moment of generating the mesh, the user has few control to generate non uniform meshes. By the way, we are going to try a first mesh generation just reducing the general mesh size.
After the mesh generation, we are going to tell GiD to give (after the mesh generation process) not only the generated tetrahedra, but also the triangles of the surfaces (by default, only the *higher dimension* elements are given). This is useful when the volume mesh could not be generated, thus allowing us to take a look on the mesh of its contour surfaces. Many times, the problem of generating a volume mesh are directly related to the bad-quality of the surfaces' mesh.

For this purpose:

- Select **Mesh->Mesh criteria->Mesh->Surfaces**, and select all the surfaces of the model. Then press **ESC**.

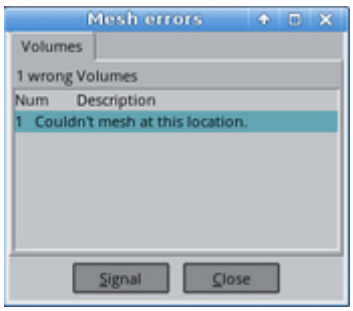
GiD has several meshing algorimths, if the selected one fails for some entity another algorithm is used.
It is possible to change this behaviour with the GiD variable *OnlyUseSelectedMesherVolumes* or *OnlyUseSelectedMesherSurfaces*.
If you write this in the *Command* line:

```
Mescape Utilities Variables OnlyUseSelectedMesherVolumes 1 Mescape
```

Then, we will detect if the advacing front volume mesher fails (otherwise the volume mesh may be generated using the *Tetgen Delaunay* me sher, with less element quality).
Now generate the mesh:

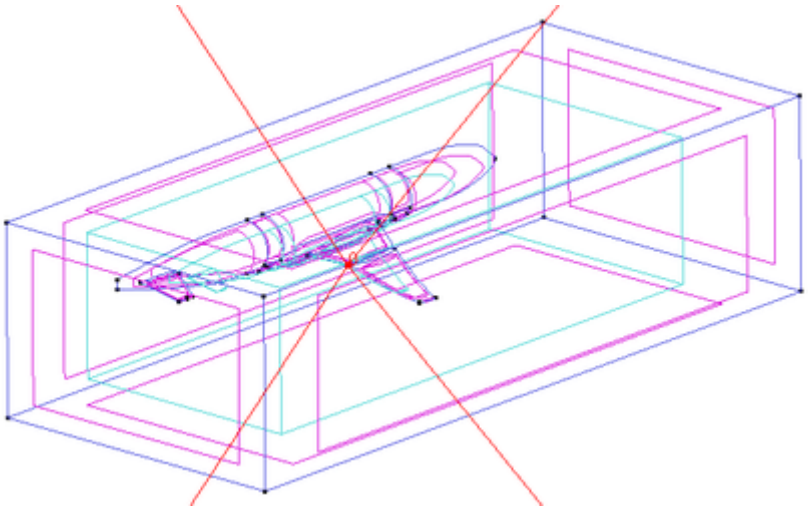
- Pres Ctrl-g or select **Mesh->Generate mesh**, erase the old mesh when asked, and enter **40** as the general meshing size and click **O K**. (This step may last some minutes).

After the mesh generation, a message appears telling that the mesh of the volume 1 cannot be generated ('Couldn't mesh at this location).



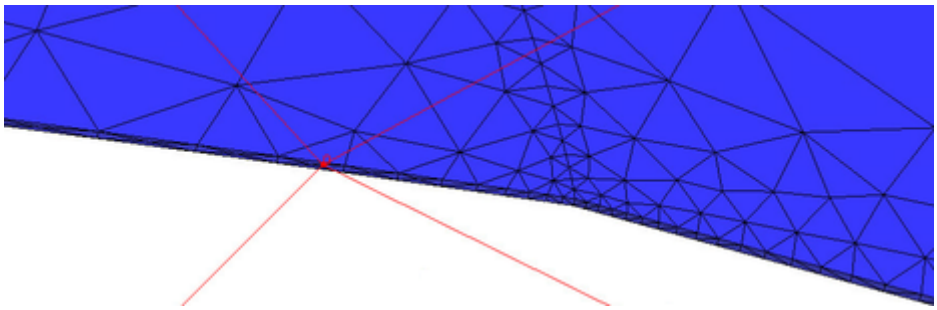
Mesh errors window

Click twice onto this message, and a big red cross will mark the problematic region of the model which made impossible the volume mesh generation (as shown in next figure).



Indication of the area where the volume mesher has found some problems

If we switch the view mode to mesh (Ctrl-m, or **View->Mode->Mesh**, or the corresponding icon  in the toolbar) and zoom in the problematic area, it can be seen that there are triangles very big in comparison to the ones in the edge of the wing (next figure). This may be the problem for meshing the volume.



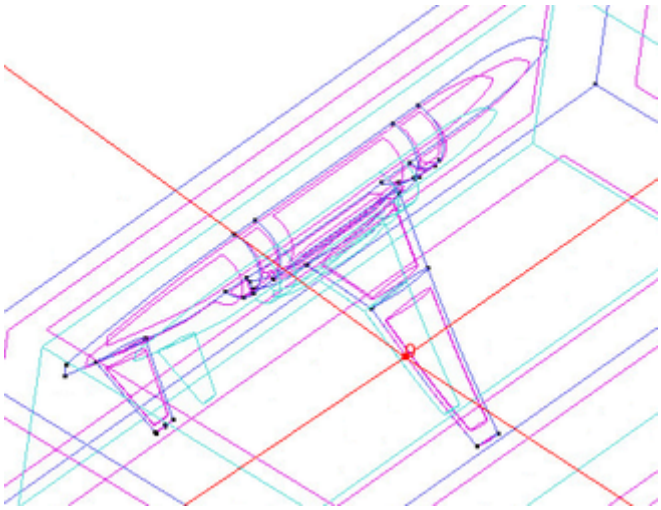
Zoomming in the problematic area signaled by the Mesh error window, in Mesh view mode.

First size assignment

One possible solution for this problem may be to mesh the whole model with a size similar to the one of the edge of the wing, but this will generate a big amount of elements, and it should be taken into account that this small size may not necessarily improve the accuracy of the simulation.
We are going to assign different sizes to the surfaces of the aircraft, the surfaces of the control volume, and the volume itself.

- Select **Mesh->Unstructured->Assign sizes on surfaces**. Enter **10** and Assign it to all the surfaces of the half-aircraft. Then press **ESC**.
- Click on **Close** to close the surface sizes assignment window.

Note that using this procedure user can assign different sizes to any kind of geometrical entity.
Now generate the mesh again (Ctrl-g or **Mesh->Generate mesh**). The mesh can not be generated again, and we see that GiD shows another part of the wing again as a problematic area. Depending on the GiD version the problematic area may be different, for instance in the tail wing.

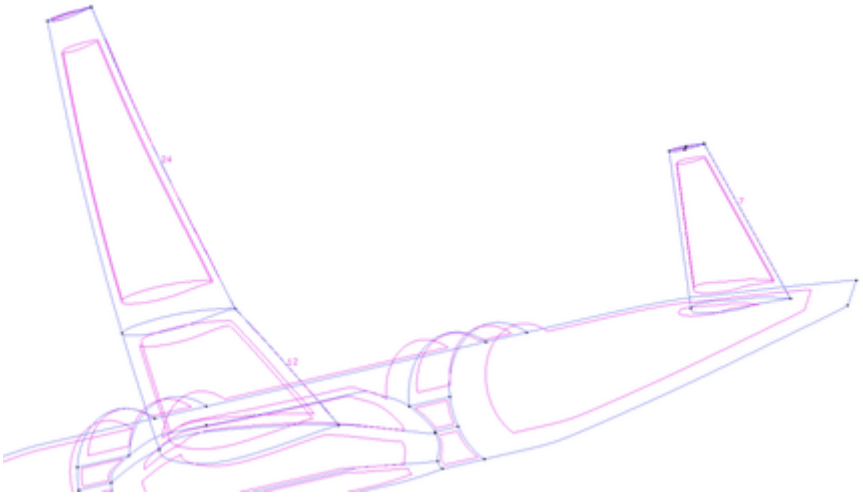


Another problematic area for the volume mesh generator.

Local size assignment

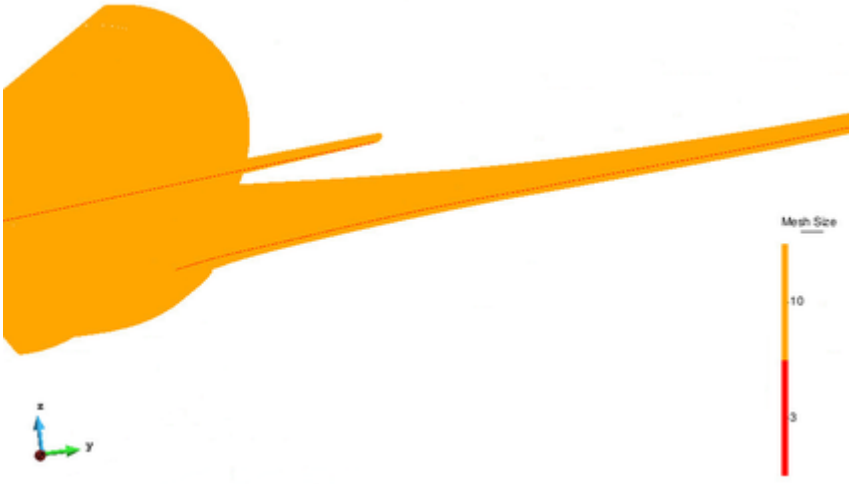
We will assign a smaller size to the surfaces of the end part of the wings. These surfaces are the number 7, 12 and 24. The id of surfaces can be checked using the **Label** option in the right mouse button menu. When the option **Label->Surfaces** is selected user can select the surfaces (with the mouse) which wants to know its number, or write down in the command line (lower part of the window) the numbers of the surfaces to be selected.

- Select **Mesh->Unstructured->Assign sizes on surfaces**. Enter **3** and select those surfaces (7, 12 and 24). Then press **ESC**.



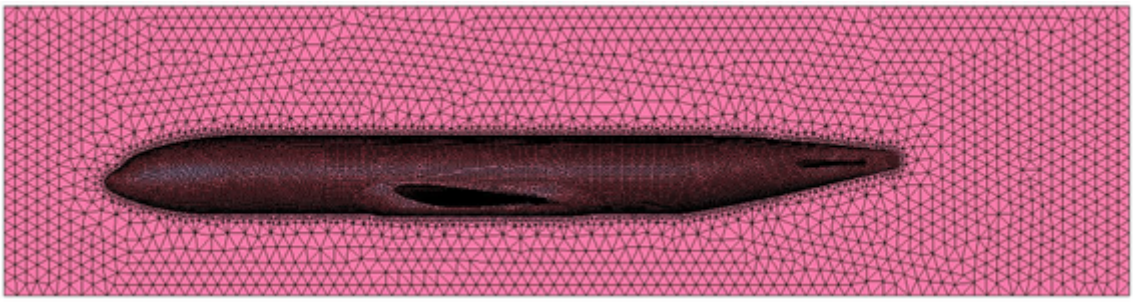
Surfaces to assign mesh sizes with label on

GiD offers many graphical tools to check which meshing properties are assigned to the geometry (size, element type, etc.). This options are accessible from **Mesh->Draw** options. You can check the sizes assigned to the surfaces by selecting **Mesh->Draw->Sizes->Surfaces:**



The red line corresponds to the surfaces with small size assigned.

You can try now to generate the mesh as before (Ctrl-g or **Mesh->Generate**), and you will see that with these sizes assigned the volume mesh can be generated.



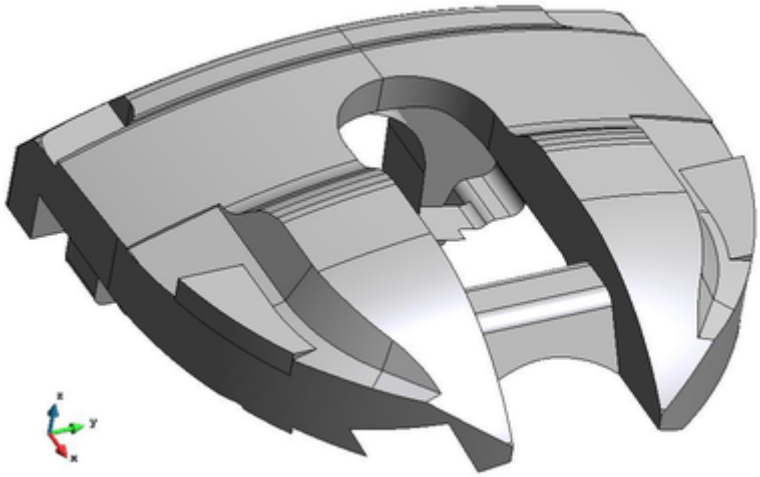
Final mesh

Meshing advanced features

This course is focused in exploring the advanced meshing features present in GiD. It is recommended to follow the basic meshing course before this one.

Skip entities

Using classical surface meshers, all the lines of the model are meshed, so the user is not allowed to generate elements larger than the surface they belong to.
Using GiD, it is possible to skip the inner lines between surfaces in contact. This option is very interesting in the cases were the geometrical definition of the model use high distorted surfaces, or very small ones in comparision with the elements size required for the simulation.
In this course we will use the model ***model_skipentities.gid***, which is a geometrical model of a mechanical part depicted in the following picture:

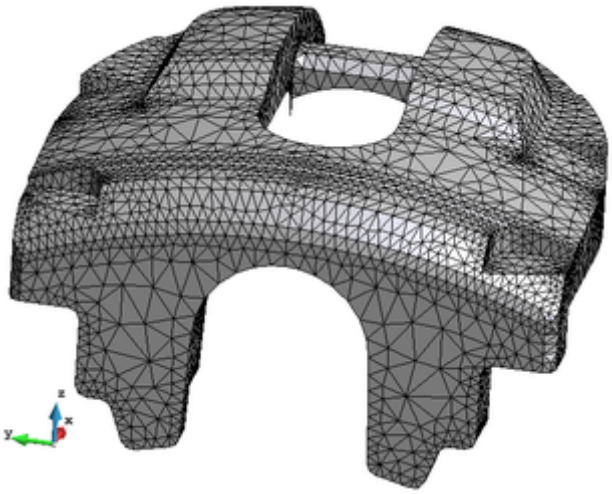


View of the model of a mechanical part used in this course.

Mesh using automatic parameters

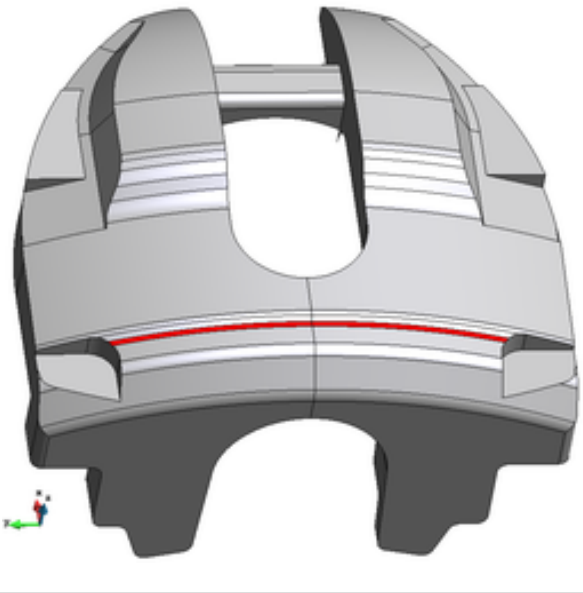
First of all we are going to generate a mesh with all the default parameters in GiD.

- Select the *Reset mesh data* option from the *Mesh* menu.
- Open the *Preferences* window and set the *Default* values in the meshing branch. To set the *Default values* in all the branches of *Mes hing*, you should select all of them, click the right mouse button, and set the *Default values on selection* option. Then click on *Apply* a nd close the window.
- Generate the mesh (for instance using the shortcut **Control-g**) setting to 9 the general mesh size. The generated mesh should look like this:

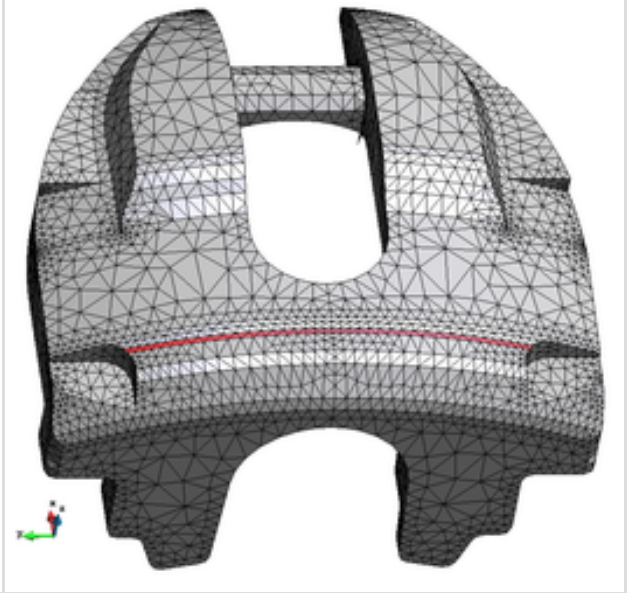


View of the mesh generated with the default meshing parameters of GiD.

It can be seen that the regions where the original surfaces are very thin make the mesher generate elements fitted inside those surfaces' contour lines. In the following figure some of these surfaces and its related elements are highlighted in red:



Some thin surfaces highlighted in red.



The elements corresponding to the thin surfaces highlighted.

Skip entities automatically

If a mesh with a more uniform element size distribution is required, the *Skip entities automatically* (*Meshing* branch in the *Preferences* windo w) option should be set. With this option, following entities will be skipped when meshing:

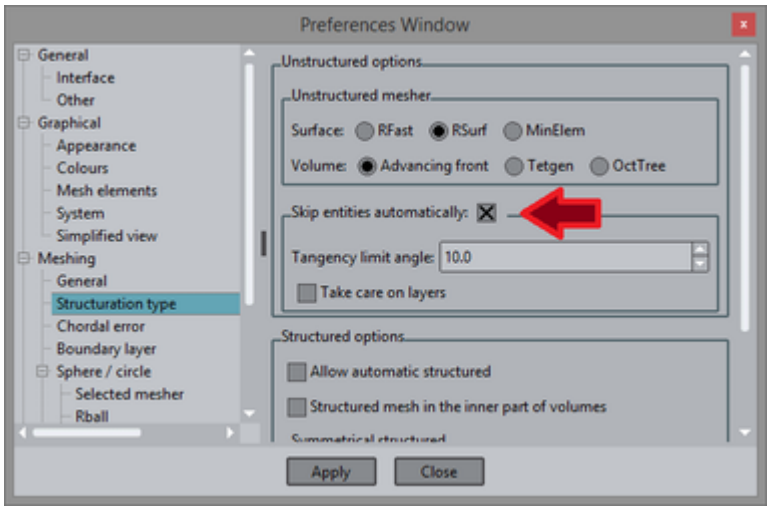
- points belonging to two lines which are tangent enough at that point.
- lines belonging to two surfaces which are tangent enough at that line.

The *tangent enough* concept is defined by a maximum angle formed by the lines (in case of points) or the normals of the surface (in case of lines). The default angle is 10 degrees, and can be set as well from the preferences window.
Other criteria are applied in order to allow or not an entity to be skipped:

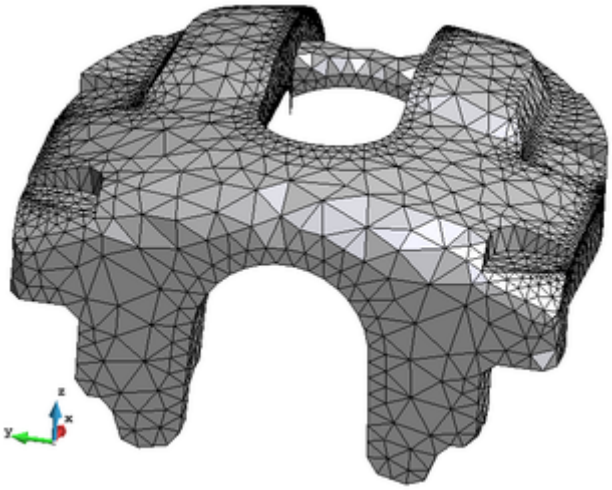
- Entities with some material, condition or group assigned are never skipped.
- Entities which parents have different groups, materials, or conditions applied are never skipped.

Let's see how it works.

- Open the *Preferences* window, and set the *Skip entities automatically* group option in the *Meshing --> Structuration type* branch from the *Preferences* window:



- Generate the mesh with the same size as before (size equal to 9). The resulting mesh should look like the following one:



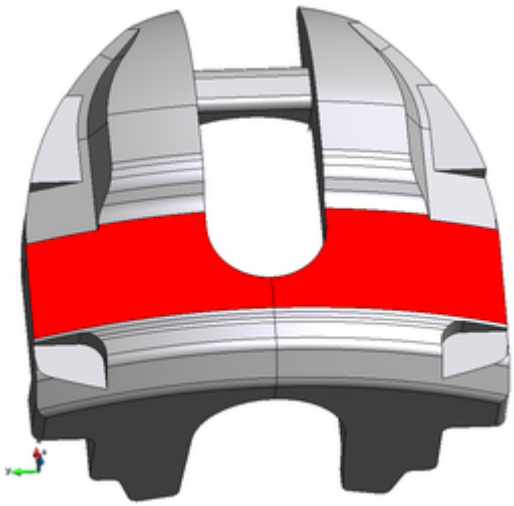
Mesh of the model using the 'Skip entities automatically' option.

As can be seen, now the elements are larger than in the mesh generated with the default options. In this case, the mesher is not forced to mesh all the lines and points of the model, so the triangles can be larger than the surfaces they belong to.
Note that the lines which form a sharp edge bewteen the surfaces they belong to are not skipped.
It has to be pointed out that the nodes of the mesh are exactly placed onto the surfaces.
Often when meshing with the *skip the entities automatically* option enabled is more difficult than using the conventional approach. In some complex geometrical models, sometimes the mesher cannot skip the desired entities, so GiD may generate the meshes without skipping any entity.

Skip entities automatically: Assign sizes

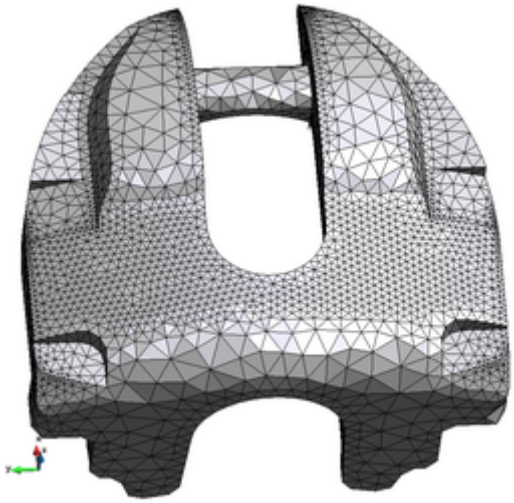
Although this option allows to generate mesh patches of surfaces toghether, user can assign different element sizes to the different surfaces or lines, and these sizes will be considered during the mesh process. Let's see it in an example:

- Assign 3 as unstructured size of the surfaces highlighted in the following figure (select *Mesh --> Unstructured --> Assign sizes on surfaces* from the top menu bar, enter 3 in the window that appears and select these two surfaces, and close the window):



Surfaces which unstructured size must be 3.

- Generate again the mesh with 9 as the general size. The resulting mesh should look like the following one:



View of the mesh skipping entities automatically and assigning a size of 3 in some surfaces

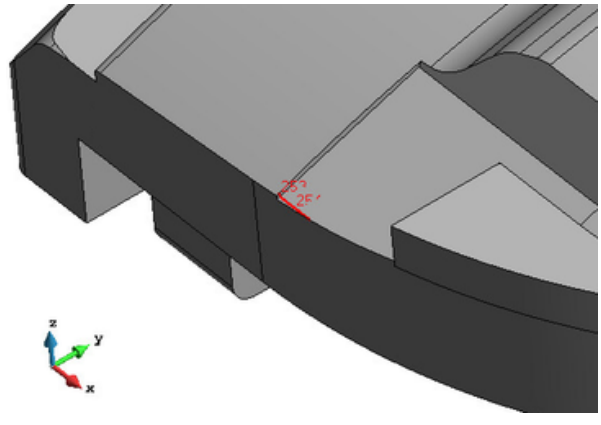
As it can be seen, the size of the elements in the region of the selected surfaces has been modified, although the lines between 'tangent enough' surfaces are still skipped.

Skip specific entities

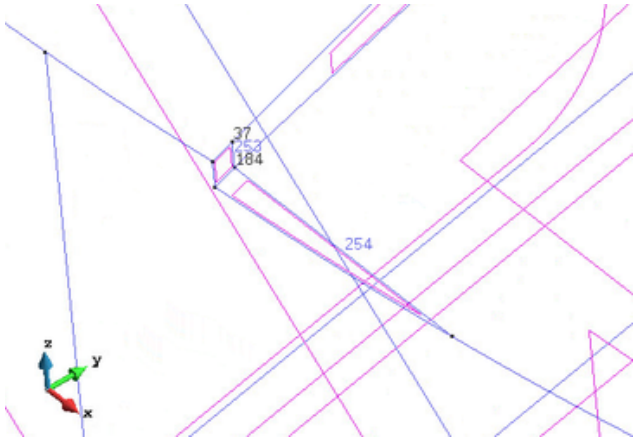
As it has been seen in the previous points, skipping entities automatically all the lines and points accomplishing the given tangency criteria are skipped. However, the user may want not to skip some entities, or skip just a specific line or point (not all of them). For this purpose the user may use the *Skip* options in the *Mesh criteria* part of the *Mesh* menu.
Accordingly to the easier way of setting the parameters for meshing, user may be interested in one of these two options:

- Skip almost all the entitites accomplishing the automatic skip criteria except some line or point. In this case, *Skip entities automatically* option must be set in the *Preferences* window, and user may select manually the entities not to be skipped (*Mesh --> Mesh criteria --> No Skip --> Points / Lines*).
- Only skip some specific entity (and don't skip the major part of entities of the model). In this case, *Skip entities automatically* must be unset from the *Preferences* window, and the entitites to be skipped must be selected manually (*Mesh --> Mesh criteria --> Skip --> Points / Lines*).

Let's see an example of the second case. Imagine we only want to skip lines number 253 and 254 and points number 37 and 184. In the following figure the region were these entities are is highlighted in red, as well a zoom of the zone is presented with the corresponding labels in this area.



Region where the entities to be skipped are, highlighted in red.

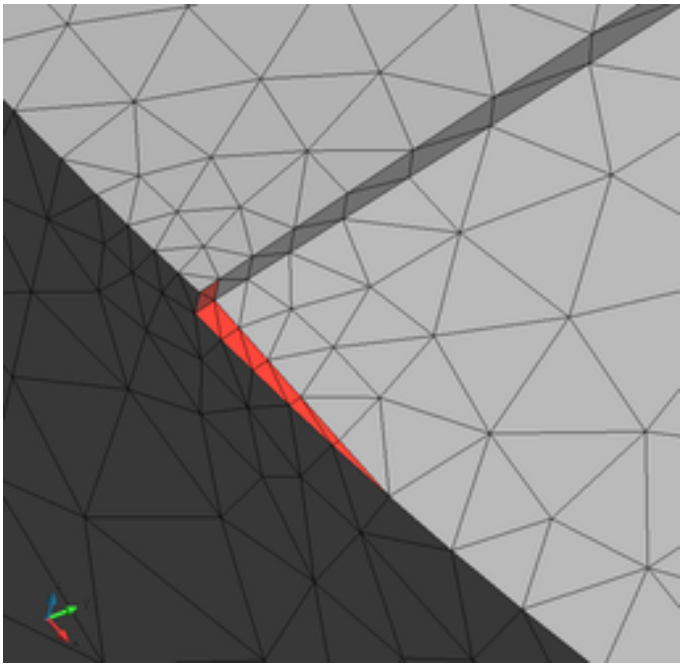


Zoom of the model to show the labels of the entities to be skipped.

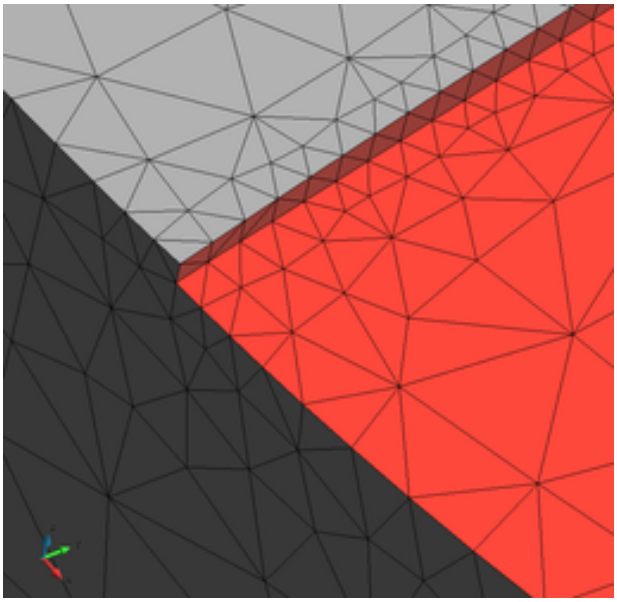
In the following figure a detail of the mesh in that region is shown, when the option *Skip entities automatically* is not set (don't skip any entity).

Let's try to skip these entities:

- Select *Mesh criteria --> Skip --> Lines* in the *Mesh* menu, and select the lines 234 and 235. Click ESCAPE to end the selection.
- Select *Mesh criteria --> Skip --> Points* in the *Mesh* menu, and select the points 37 and 184. Click ESCAPE to end the selection.
- The *Draw --> Skip entities* option of the *Mesh* menu can be used in order to see graphically the entities that will be skiped or not when meshing.
- Ensure the option *Skip entities automatically* is not set in the preferences window, and generate the mesh with 9 as the general mesh size. A zoom of the resulting mesh in the area of interest should be like the following one.



Detail of the mesh in the zone of interest without skipping entities automatically.



View of the mesh generated skipping only the selected lines and points.

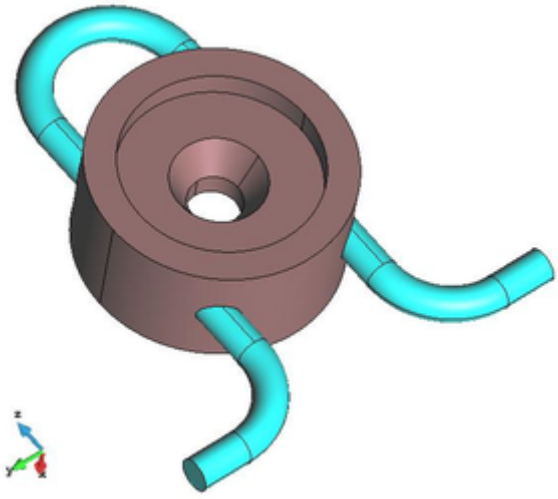
As can be seen, now the mesh is different, as the lines and points selected are not meshed (switch between geometry and mesh mode to see were the lines 253 and 254 are located with respect to the triangles).

Chordal error

The chordal error is defined as the distance between the mesh elements and the geometrical model. In planar regions, for instance, its value is always zero, but in regions with curvature the chordal error increases as the element size gets higher.
Depending on the characteristics of the model and the simulation requirements, it may be interesting to assign sizes to the geometrical entities according a given maximum chordal error, or let GiD refine the regions of the model with higher curvature in order to accomplish it.
In this section it is explained how to take care on the chordal error when generating meshes with GiD.

Assign sizes by chordal error

The option of assigning sizes by chordal error criteria is useful when the geometrical entities (lines and surfaces) have more-less the same curvature within all the entity.
By using this option, GiD computes automatically a mean curvature of the whole entity and assigns a mesh size to that entity which accomplishes with the chordal error defined considering the curvature.
In order to see an example, open the GiD model ***gid_model_basic_course.gid*** from the folder where the material of the course is.
Hereafter, a figure of this model is shown:

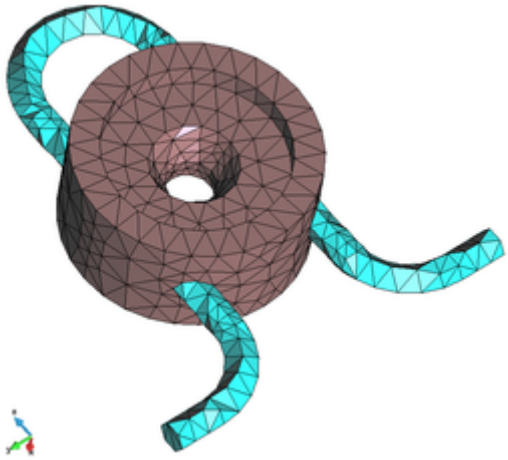


View of the model used in this part of the course.

Mesh using automatic parameters

First of all, let's see an example of mesh using the default meshing parameters of GiD.

- Select *Reset mesh data* from the *Mesh* menu, to ensure the mesh will be generated using default parameters.
- Open the *Preferences* window, and set the *Default values* in the Meshing branch. (remember to select all subbranches within the *Mesh* branch, right click and select *Default values on selection*) Then click *Apply* and close the *Preferences* window.
- Generate the mesh using 10 as general size. The following mesh should be generated:



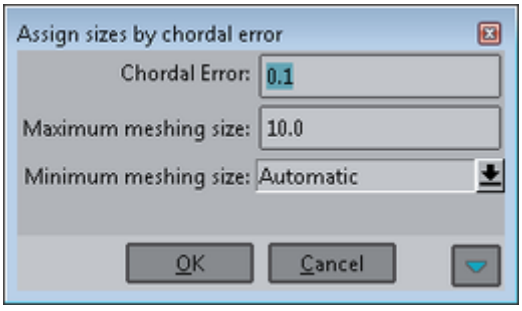
View of the mesh generated with GiD default values and a general mesh size of 10.

As it can be seen, this mesh is probably too coarse in the thin cilindric volumes, but it may be ok for the planar shapes.

Mesh with sizes assigned

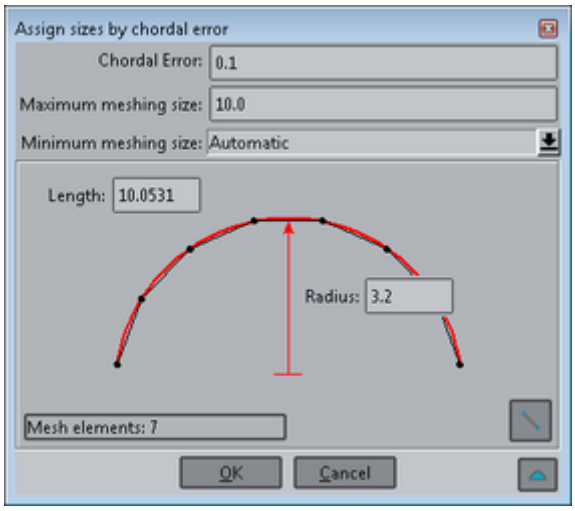
We are now going to assign better sizes to the surfaces and lines of the model, so as the final mesh size will be guided by a given chordal error criteria.

- Select the option *Unstructured --> Sizes by chordal error...* from the *Mesh* menu. The following window should appear:



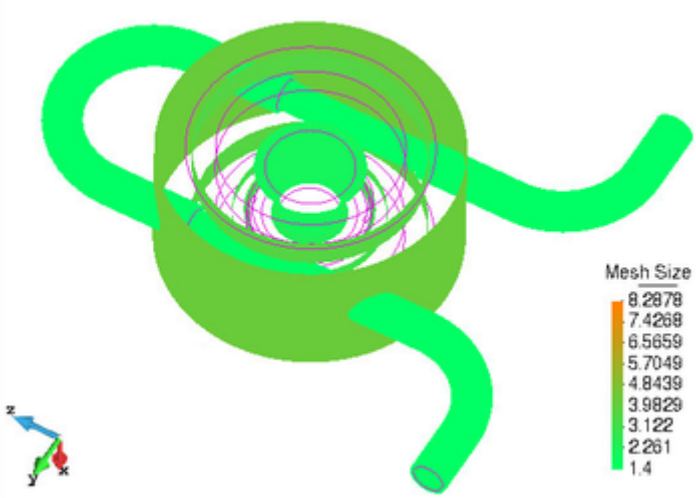
In this window user can enter the absolute value of the chordal error allowed, and the maximum and minimum mesh size. To get an idea of the corresponding element size given by a chordal error, user can expand the window by clicking on the *down-arrow button* in its right-lower part (beside CANCEL button):

- Expand the window and click on the *line-icon button* of the right-lower part of the window. Then select some curved line of the model. If you click, for instance, on the line 113, the following values appears in the window:



Now we see that with a value of 0.1 as chordal error, 7 elements will be created in the line 113.

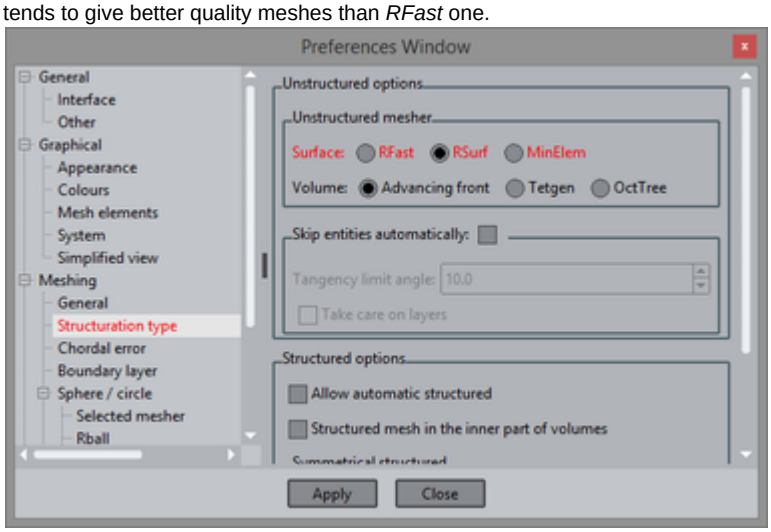
- Click *OK* in the window. Now GiD has assigned automatically a mesh size to the lines and surfaces of the model, trying to accomplish with the chordal error specified by the user.
- Select *Draw --> Sizes --> Surfaces* from the 'Mesh' menu. Then the sizes assigned can be seen onto the model as shown in the following figure:



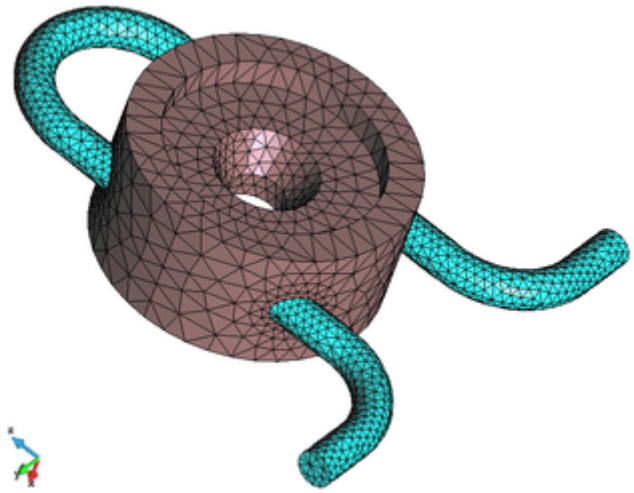
View of the sizes assigned automatically by GiD considering a given chordal error.

Note that the planar surfaces have no size assigned, as they have no curvature.

- Set *RSurf* as the unstructured surface mesher in the *Meshing --> Structuration type* branch of the *Preferences* window. This mesher tends to give better quality meshes than *RFast* one.



- Generate the mesh (*Ctrl-g*) with 10 as general size, and see the result:



Mesh generated with the sizes assigned automatically by GiD, following the maximum chordal error allowed specified by the user.

Note that this final mesh has smaller mesh size in some geometrical entities, but each geometrical entity is meshed considering a single size for the whole entity.

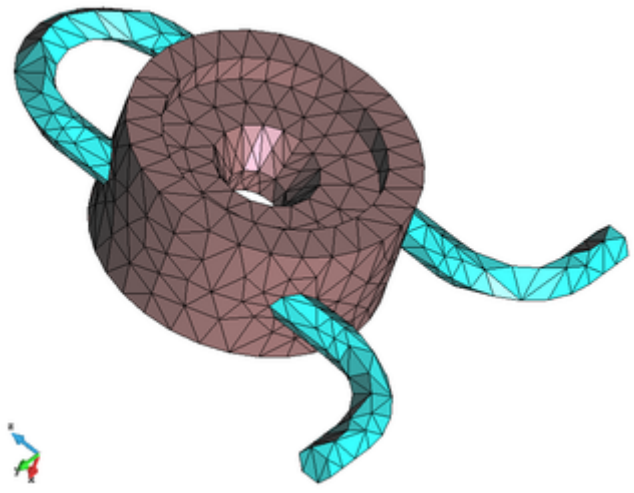
Smoothing options

In the *Meshing --> Other* branch of the *Preferences* window, user can find the *Smoothing* option, inside the Others group. These possibilities are:

- Normal
- HighAngle
- HighGeom

The first option performs a standard Laplacian-like smoothing algorithm. The second one tries to improve more the quality of the mesh elements in terms of angle.
The third option (HighGeom) is the interesting one when trying to minimize the chordal error of the mesh generated. If user set this smoothing preference, the final smoothing of the mesh will try to minimize the chordal error of the elements, but this may give a worse quality elements.
To see how this option works, let's see an exagerated example of this using the model ***gid_model_basic_course.gid*.**

- Open the model ***gid_model_basic_course.gid***.
- Select the *Reset mesh data* option from the *Mesh* menu.
- Set the *Default values* in the *Meshing --> Other* branch of the *Preferences* window. To set the Default values in all the branches of *M eshing*, you should select all of them, click the right mouse button, and set the *Default values on selection* option. Then click on *Appl y*.
- Then set *RSurf* as surface mesher in the *Meshing --> Structuration type* branch, and set *HighGeom* as *Smoothing* parameter in the *Meshing --> Other* branch.
- Generate the mesh (ctrl-g) of the model with a general size of 10. The resulting mesh should look like the following one:

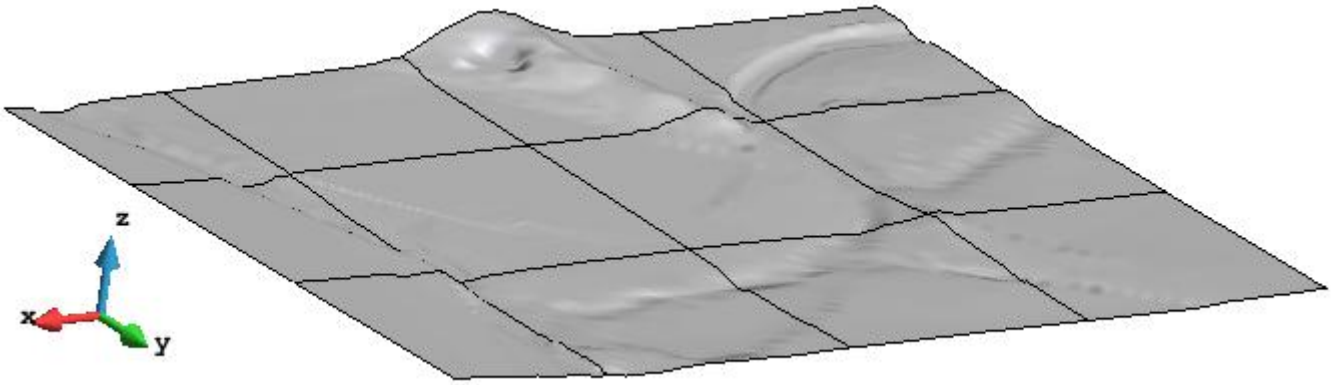


View of the mesh using a large mesh size and the HighGeom option for the smoothing.

Although the mesh has a big chordal error, comparing this mesh with the one obtained with the default parametera (see section Mesh using automatic parameters of the course), user can see that the configuration of the elements tends to follow the curvature directions of the surfaces.

Chordal error to the whole model

Sometimes, rather than assigning mesh sizes to geometrical entities (the same size assigned to the whole entity), user may need to consider a chordal error criteria considering the whole model, independently on each entity. Let's see an example.
Open the GiD model '***model_chordal_error.gid***' from the folder where the material of the course is. Hereafter, a figure of this model is shown:



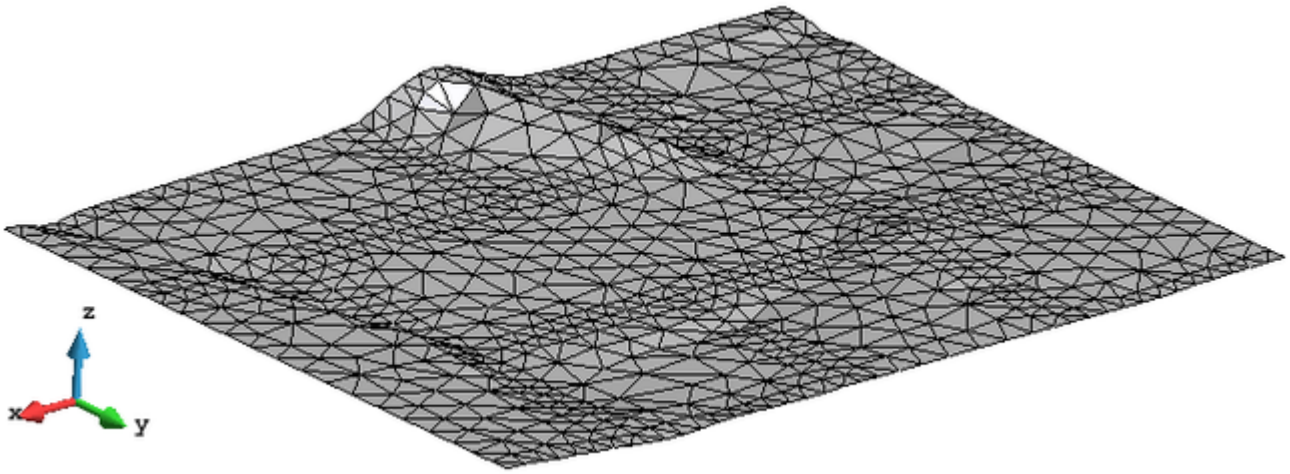
View of the geometrical model used in this course.

This model comes from a DTM (digital terrain model). One of the typical characteristics of this kind of models is the big size of the model in two directions (x and y in this example) compared with the other direction (z). This implies a specific meshing requirements in order to get the less number of elements without loosing accuracy in the geometry representation.

Chordal error to the whole model: Mesh using automatic parameters

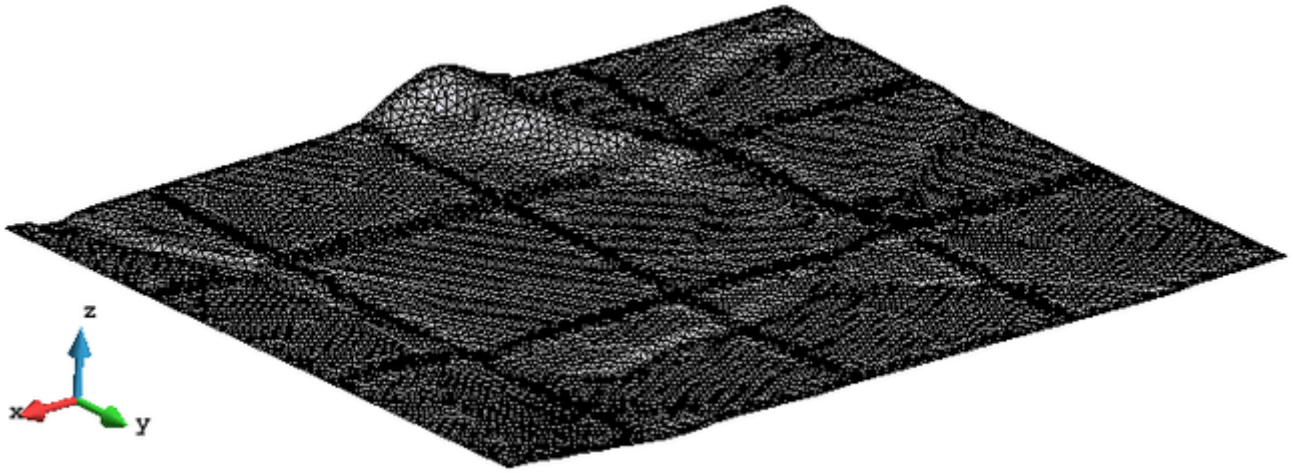
Let's have a look at the default mesh provided by GiD if meshing with the default meshing parameters:

- Set the *Default values* in the *Meshing* branch of the *Preferences* window. (remember to select all subbranches within the *Mesh* bran ch, right click and select *Default values on selection*) Then click *Apply*.
- Select *Generate mesh* option from *Mesh* menu, and generate the mesh with the automatic size proposed by GiD. The resulting mesh should look like the following one:



Mesh of the model with the automatic meshing parameters of GiD.

This mesh could look reasonable, but if we focus on some regions of the model, we see that some parts of it are not represented by the mesh because they are smaller than the element size. However, we see that these parts are localized in specific regions of the model. The rest of the model is well represented by this mesh.
We could try to generate a finner mesh of the model to capture the geometrical details, but this should lead to an excessive number of elements. Try to generate the mesh with a general size of 4. In the next figure the resulting mesh is shown:

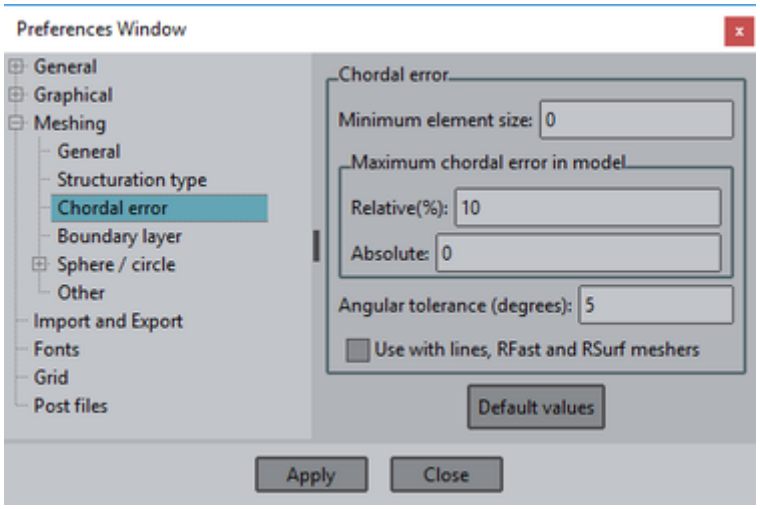


Mesh of the model with a general mesh size equal to 4.

Another option could be to assign a smaller size to the surfaces which have the details to be preserved, but it can be seen that those details do not cover the whole surface.

Chordal error options

User can control some parameters in order to apply the chordal error criteria to the whole model (not to specific geometrical entities). This parameters can be set at the *Meshing --> Chordal error* of the *Preferences* window:



View of the Meshing --> Chordal error *branch of the* Preferences *window.*

There are three parameters which controls the maximum chordal error allowed in the model:

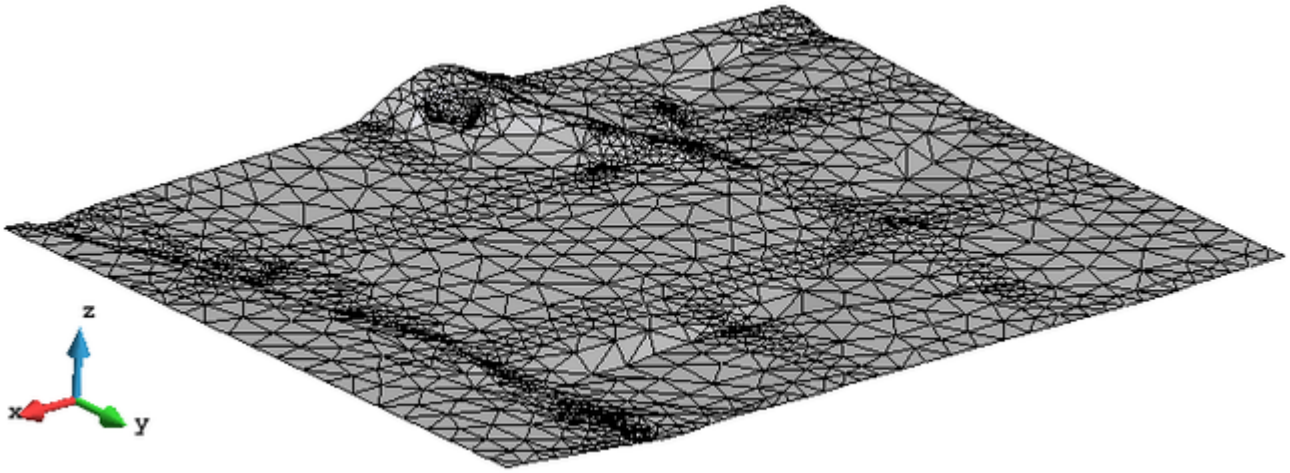
- **relative**: this value is the chordal error of an element divided by its characteristic size.
- **absolute**: this value is the chordal error itself.
- **Angular tolerance**: this value is the deviation between normals of adjacent entities.

The parameter *Minimum element size* allows the user to define the minimum size for the mesh elements. Trying to accomplish the chordal error criteria, GiD may generate too small elements in some regions, so this parameter may be interesting sometimes.
The checkbox *Use with lines, RFast and RSurf meshers* indicate that these parameters will be applied when using these meshers. Otherwise, they will only be considered in MinElem mesher (this mesher is explained in detail in section MinElem mesher).

Mesh following chordal error chriteria

Let's use a chordal error criteria in order to let GiD refine the mesh in the regions with higher curvature.

- Open the *Preferences* window.
- Set the *RSurf* as the surface mesher (as told in previous sections of the course, under *Preferences --> Meshing --> Structuration type*, *RSurf* mesher, in general, trends to represent the original geometry in a more accurate way).
- Set the *Use with lines, RFast and RSurf meshers* option in the *Meshing --> Chordal error* branch, and set to 10% the relative value of the *Maximum chordal error in model*, and an *Angular tolerance* of 5 degrees.
- Generate the mesh (ctrl-g) of the model using 15 as the general size. The resulting mesh should be like the following one:



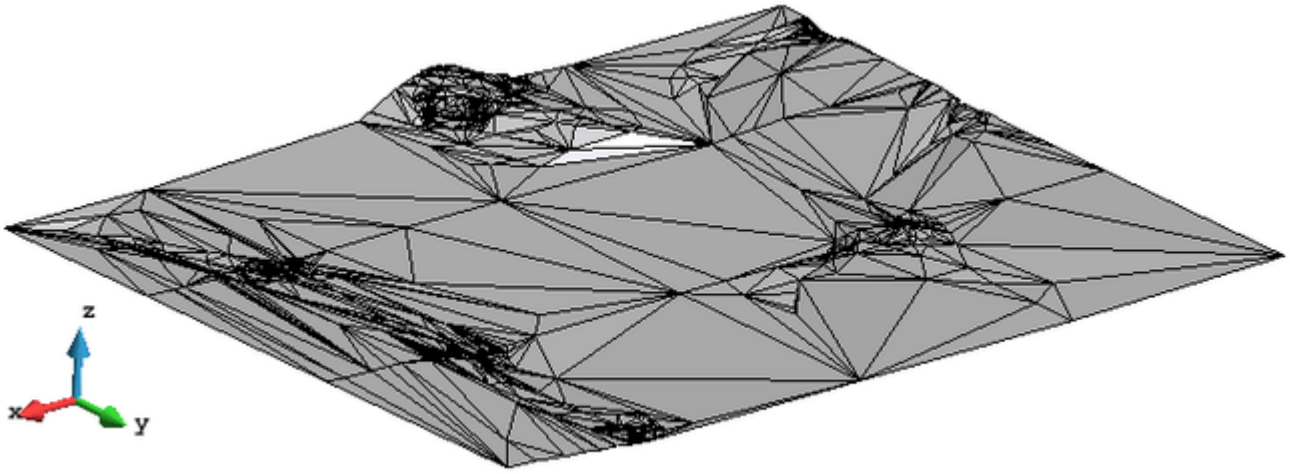
Mesh of the model using RSurf mesher and 10% as the maximum relative chordal error allowed, and an angular tolerance of 5 degrees.

As it can be seen in the figure, the regions with higher curvature have a refined mesh.

MinElem mesher

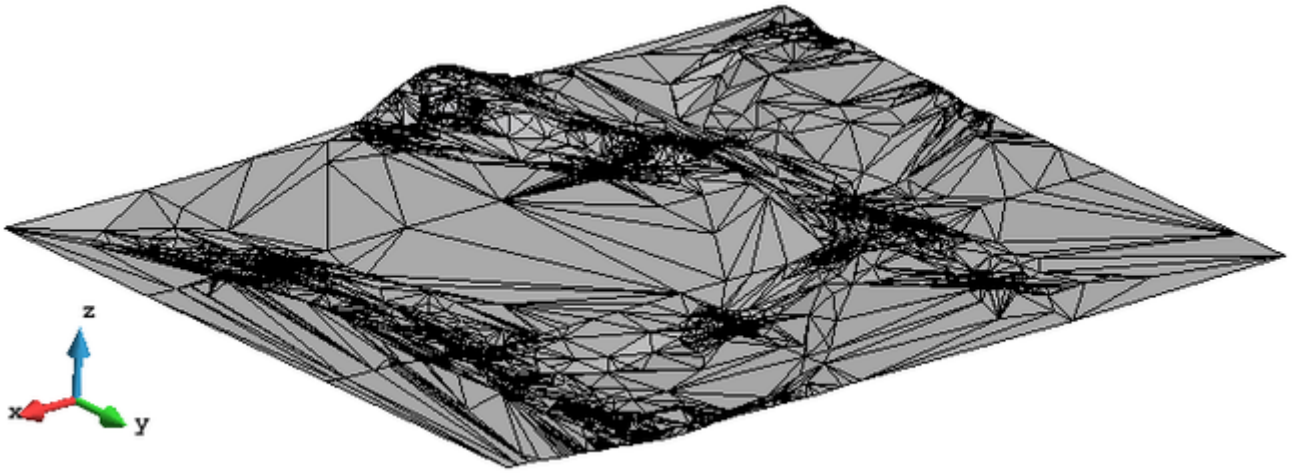
The MinElem surface mesher is designed to generate as minimum number of elements as possible representing the shape of the geometry according to the given chordal error parameters. This kind of meshes are often used for visualization purposes, or in some parts of the model where the quality of the elements is not so important. Let's see an example:

- Open the *Preferences* window.
- Set the *MinElem* as the surface mesher in *Preferences --> Meshing --> Structuration type*.
- Generate the mesh (ctrl-g) of the model using 15 as the general size (actually, this mesher does not consider the meshing size, as it only takes care of the shape of the geometry: in planar shapes, the size of the elements may be bigger than the one indicated by the user). The resulting mesh should be like the following one:



View of the mesh generated with MinElem mesher.

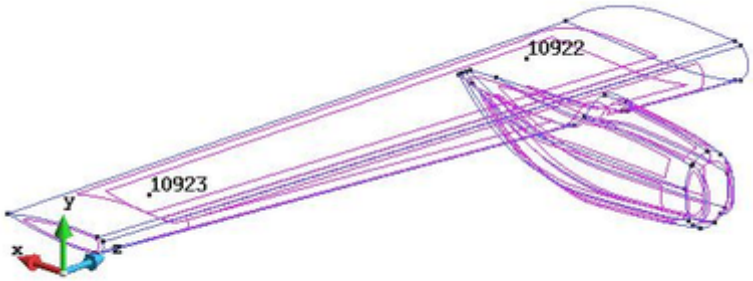
Changing the *chordal error settings* in the *Preferences* window, it can be appreciated how the mesh is adapted to the geometry depending on them. In the following Figure, for example, the resulting mesh using 2 degrees as *angular tolerance* is depicted:



View of the mesh using MinElem mesher, 10% as Relative maximum chordal error, and 2 degrees as angular tolerance.

Force points to mesh

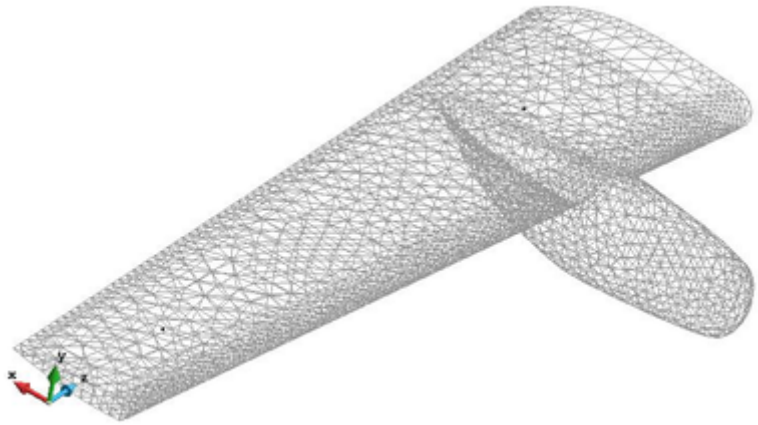
The geometrical definition inside GiD follows a hierarchical approach: a line must have points in its ends (or one point if it is a closed curve), a surface must be surrounded by lines, and a volume must have a closed collection of surfaces as a contour. This topology is automatically passed to the mesh; for instance, the nodes of a line's mesh are nodes of the mesh of the surfaces containing that line.
In some situations, user may need to force a mesh to have a node in a given position of space. GiD includes this functionality for surface and volume meshes.
In the following example we will use the model ***model_course_forced_points.gid***, shown in the following figure (*View --> Render --> Normal*). The model represents an aircraft wing.



View of the model used in this course.

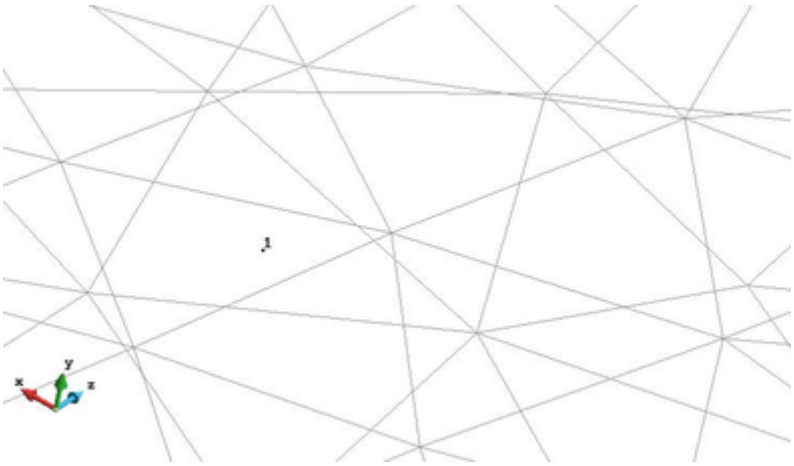
In the figure two points can be appreciated (labeled as 10922 and 10923). These points are not belonging to any higher order geometrical entity (it can be seen using the *View --> HigherEntity --> Points* functionality).
Let's generate a mesh of the model:

- First of all reset all meshing settings: open the *Preferences* window, select all subbranches within the *Mesh* branch, right click and select *Default values on selection*; then click *Apply* and close the *Preferences* window.
- Generate a mesh (ctrl-g) using 100 as the general mesh size, setting the option *Get meshing parameters from model*. This option uses most of the meshing parameters used the last time the model was meshed. The resulting mesh should be as follows:



Mesh of the model using 100 as general mesh size.

If we zoom the zone were the isolated points are, we can see that that points have generated isolated nodes, which are not connected to the surface mesh.



Zoom of the zone of the mesh where it can be seen that the node 1 (comming from the point 10923 of the model) is isolated.

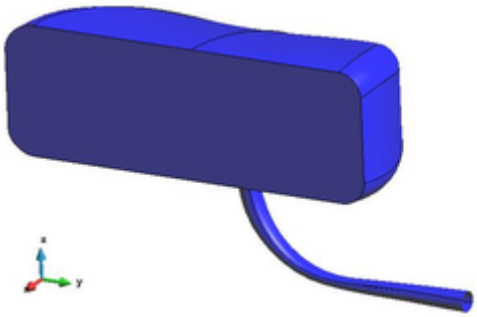
Mesh with forced points

To force that two points to own the mesh of a surface, user may follow these steps:

- Select *Mesh criteria* --> *Force points to* --> *Surface mesh* in the *Mesh* menu.
- Select the surfaces of the model and press <*Esc*> (steps to follow are indicated in the *command line*, in the lower part of GiD window).
- Then select the two points and click <*Esc*>. GiD will assign the closest surface (between the surfaces selected) to each of the points selected.
- Generate the mesh again, and you will see that now the surface mesh owns a node in each position of the points to be forced. (A useful way to check this is zoom in the area of interest and switch between *Geometry* and *Mesh* view mode, so as the position of the geometrical point can be seen.)

Boundary layer mesh

In some situations, the numerical simulation require anisotropic meshes. This kind of requirement is often associated to CFD codes, which need a big concentration of elements in some areas following the direction in which the velocity of the fluid changes rapidly (high gradients of velocity). This anisotropic meshes are typically located in the contacts between fluid and solid in Fluid Structure Interaction (FSI) problems. This particular mesh receive the name of *boundary layer mesh*, and presents layers of very stretched elements near the solid wall. This virtual layers get higher as the distance to the solid wall increases.
GiD is able to generate boundary layer meshes both in 2D and 3D. In this course we are going to generate a boundary layer mesh in 3D (volume elements) of the model ***model_course_blm.gid,*** which is a model of the rear view mirror of a competition car.



View of the model used in this course.

Switch on and off the layer named *vol* in order to see the control volume around the model.

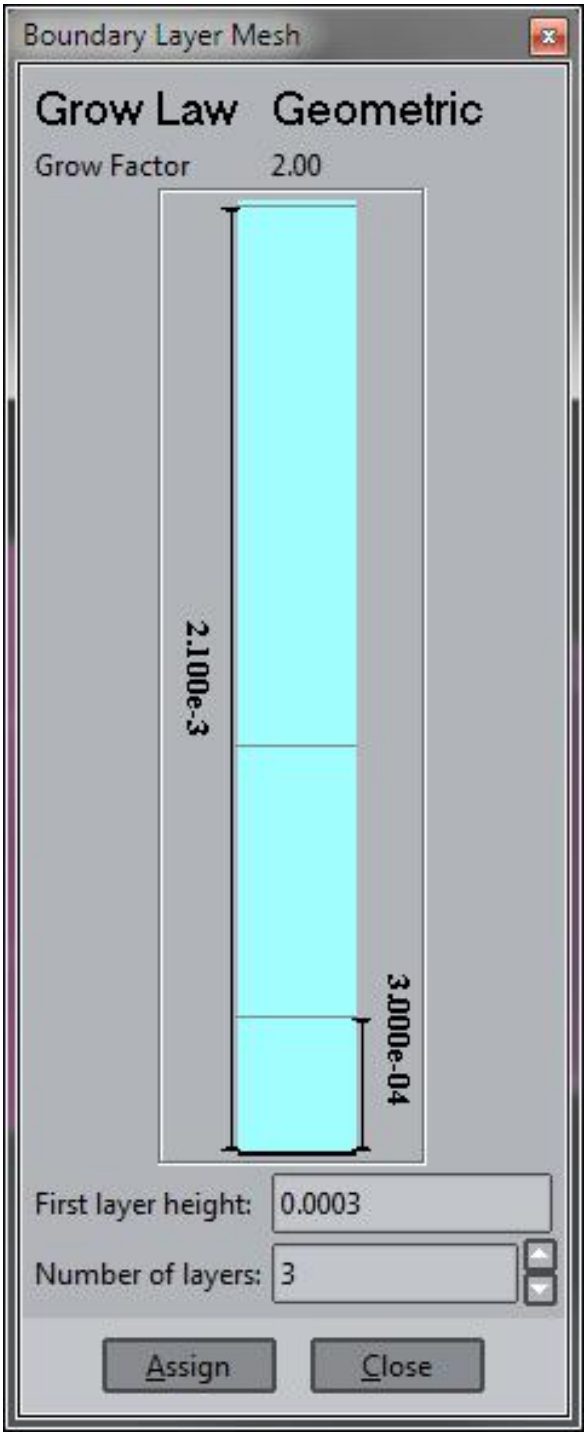
Create 3D boundary layer mesh

We are going to generate a volume mesh of the control volume (in layer *vol*), and a boundary layer mesh around the rear view mirror (surfaces in layer *mirror*).

Switch layer *vol* on again.
Select *Boundary layer --> 3D --> Assign* in the *Mesh* menu, and select the volume of the model. Then click *Escape*, and the *Boundary Layer Mesh* window should appear:

- Fill the parameters with the values shown in the figure: 0.0003 as *first layer height*, and *Number of layers* equal to 3. Press *Assign* button, and select the surfaces in layer *mirror*. Then click ESCAPE, click on the *Finish* button and close the Boundary Layer Mesh window.

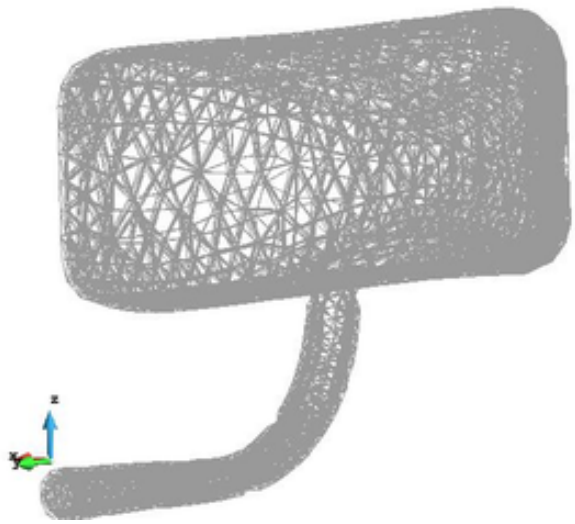
  As it can be seen, in the *Boundary Layer Mesh* window a schematic distribution of the stretching of layers of the boundary layer mesh is shown considering the parameters entered. User can change the stretching function and the growing factor in the corresponding part of the *Meshing --> Boundary layer branch* of the *Preferences* window. In this example we will use the default parameters .



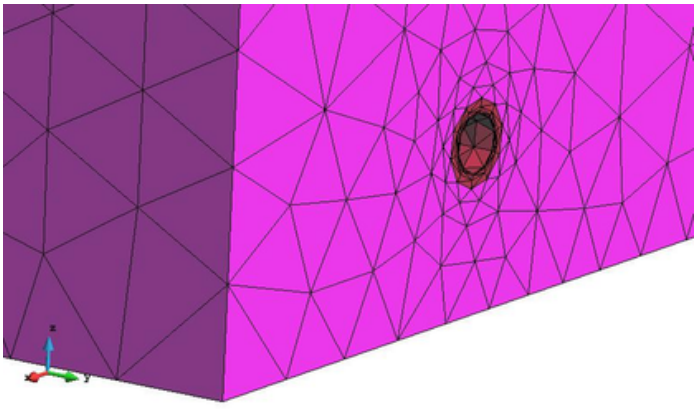
View of the boundary layer mesh window

Generate the mesh (<Ctrl>-g) with general size equal to 0.0175, and using the *meshing parameters from the model* (check the corresponding checkbox in the window appearing when selecting *generate mesh*).

The resulting mesh has the boundary layer elements attached to the surfaces in the layer *mirror*. It can be seen that a new Layer has been created inside GiD named *BLM*. This layer contains the different boundary layer meshes of the model. It is useful to see separately the boundary layer mesh from the isotropic one. If user doesn't want to get the boundary layer mesh in a separated Layer of GiD, the option can be set in the *Meshing --> Boundary layer branch* of the *Preferences* window.



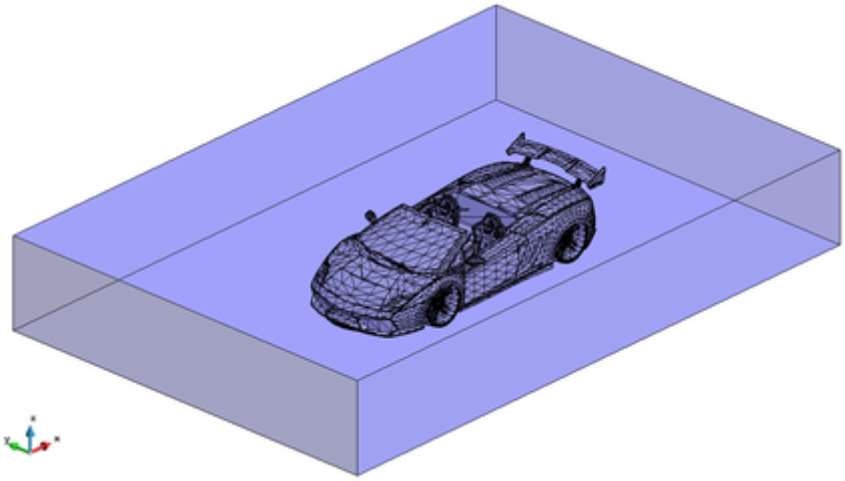
View of the boundary layer mesh (elements inside BLM//vol Layer of GiD)



Zoom of the region where the mirror gets in contact with the control volume contours.

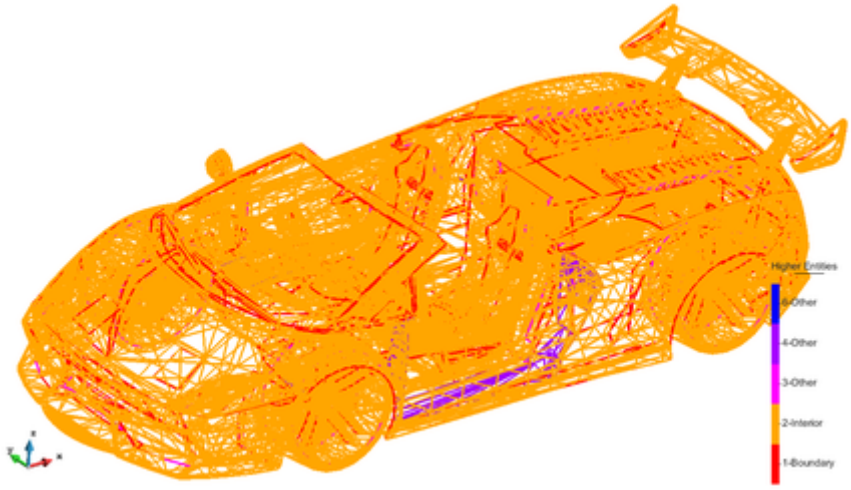
Embedded mesh options

In this section, some options of the embedded mesher are shown. The model ***gid_embedded_example_lamborgini.gid*** is used in this course, which is a geometrical model of a car inside a control volume, as shown in the following figure:



View of the model used in this course.

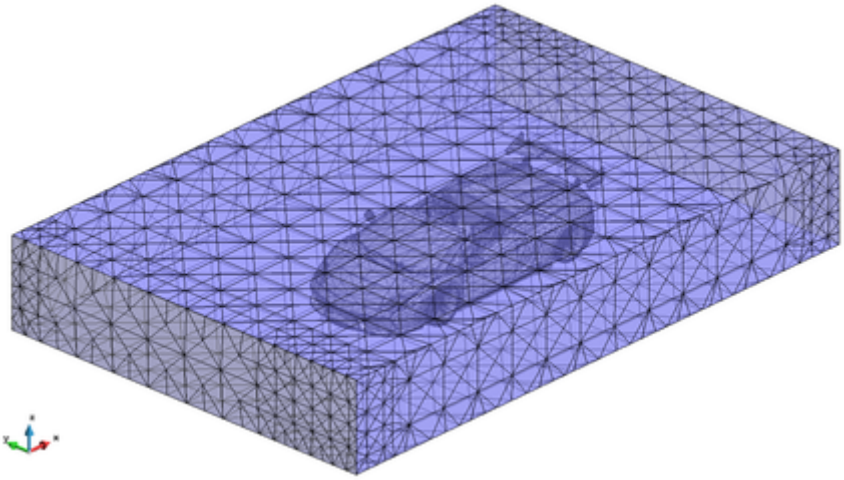
This embedded model consists in a prismatic volume (the calculation volume), and the embedded region defined by the surfaces defining the car. Note that the embedded region must enclose a region, but it is not needed to be watertight. As it can be seen in the following figure, some of the higherentities of the lines of the car are not 2 (which is the condition to be watertight):



Higher entities of the lines belonging to the surfaces defining the car.

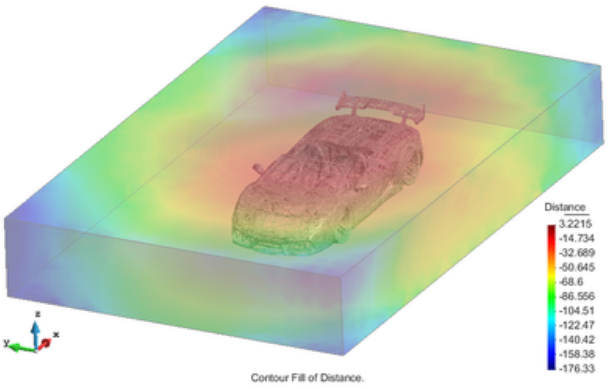
First of all, select the ***Embedded*** mesh type in the *Meshing --> General* branch of the *preferences* window, and a size transition factor of 1. in order to be faster generating the mesh.

Then, generate the mesh of the model (*Mesh --> Generate*), using a general mesh size of 20. The following mesh should be generated:

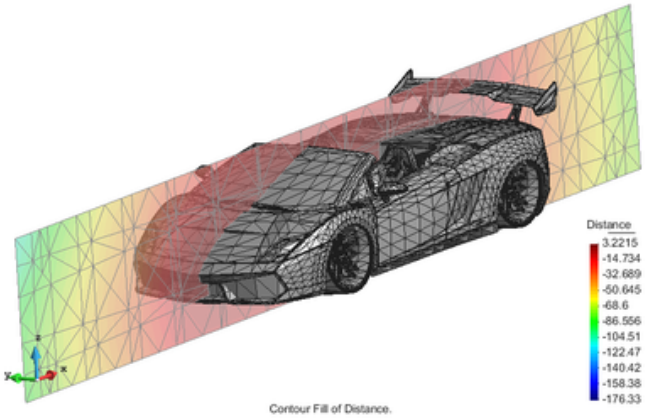


View of the mesh generated

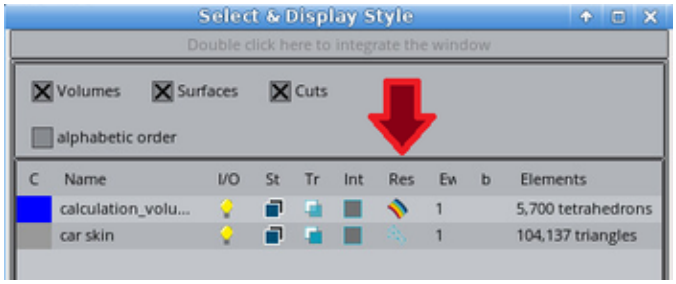
Going to the post-processing part of GiD, we can see that a result has been created automatically with the signed distances on the nodes. In the following figures the contour fill of distances on the mesh and onto a cut can be appreciated:



View of the contour fill of distances on the skin of the calculation volume mesh.

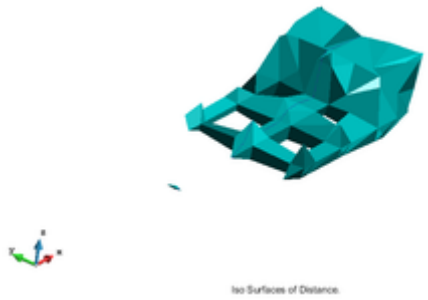


View of the contour fill of distances on a cut plane.



Remember to hide the result view on the car skin set in the Select & Display Style window:

Having a look at the iso-surface of distance equal to 0 (which may represent the skin of the car) we see that is too coarse to represent it:



View of the iso-surface of distance equal to 0 with the mesh generated with size equal to 20.

Assign mesh sizes

Assign a mesh size of 0.5 to all the surfaces of the car (*Mesh --> Unstructured --> Assign sizes on surfaces*) and generate the mesh again. Now, it can be appreciated that the mesh is finner near the skin of the car, so the iso-surface of distance 0 fits better to it:

View of the iso-surface of distance equal to 0 with the mesh generated with size equal to 0.5 in the skin of the car.

Do not consider outer elements

In most cases, the solvers working with embedded meshes don't need the elements which all their nodes are out of the embedded region. You can unset the option *Get outer elements in embedded meshes* in the *Meshing --> Other* branch of the preferences window in order not to have this elements (this option is only available if *Mesh type* is set to *Embedded in Preferences --> Meshing --> General*). If so, the generated mesh has less elements, so some memory can be saved in the simulation process.

Considering the last mesh generated, for instance, the resulting mesh goes from about 9.3 Millions of tetrahedra to 6.7Milions. You can appreciate the mesh generated is not massive by moving the clip planes (View menu) to see its inner part in render flat mode.

Customization

This tutorial takes you through the steps involved in defining a problem type using GiD. A problem type is a set of files configured by a solver developer so that the program can prepare data to be analyzed.
A simple example was chosen, and takes us through all the associated configuration files while using few lines of code. Particular emphasis is given to the calculation of the centers of mass for two-dimensional surfaces. A simple formulation both conceptually and numerically.
By the end of the example, you should be able to create a calculating module that will interpret the mesh generated in GiD Preprocess. The module will calculate values for each element of the mesh and store the values in a file in such a way as they can be read by GiD Post-process.

Introduction

Our aim is to solve a problem that involves calculating the center of gravity (center of mass) of a 2D object. To do this, we need to develop a calculating module that can interact with GiD.
The problem: calculate the center of mass.
The center of mass (X_{CM},Y_{CM}) of a two-dimensional body is defined as

$$x_{CM} = \frac{\int_S \rho \, x \, x \, y + \sum_{i=1}^{N} m_i \, x_i}{\int_S \rho \, x \, y + \sum_{i=1}^{N} m_i} \qquad y_{CM} = \frac{\int_S \rho \, y \, x \, y + \sum_{i=1}^{N} m_i \, y_i}{\int_S \rho \, x \, y + \sum_{i=1}^{N} m_i}$$

Where (x,y) is the density of the material at point (x,y) and S is the surface of the body; mi are concentrated masses applied on the point (x_i,y_i).
Each of the N elements is treated as concentrated weight whose mass is defined as the product of the (surface) density and the area of the element.

Interaction of GiD with the calculating module

GiD Preprocess makes a discretization of the object under study and generates a mesh of elements, each one of which is assigned a material and some conditions. This preprocessing information in GiD (mesh, materials, and conditions) enables the calculating module to generate results. For the present example, the calculating module will find the distance of each element relative to the center of mass of the object.
Finally, the results generated by the calculating module will be read and visualized in GiD Post-process.

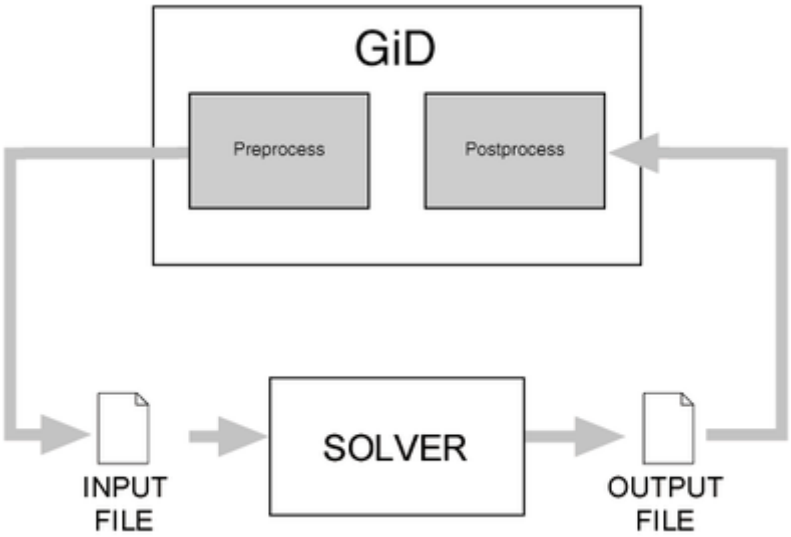


Diagram of the workflow

GiD must adapt these data to deal with them. Materials, boundary and/or load conditions, and general problem data must be defined.
The calculating module (in this example cmas2d.exe) solves the equations in the problem and saves the results in the results file. This module may be programmed in the language of your choice, 'C' is used in this example
GiD Post-process reads the following files generated by the calculating module:
project_name.post.res: results file.
Each element of the mesh corresponds to a value.
project_name.post.msh: file containing the post-process mesh. If this file does not exist, GiD uses the preprocess mesh also for postprocess.

Example: cmas2d_customlib

Let's see out example: the cmas2d_customlib problemtype. You can see it's files on the example's problemtype folder (GiD 14.0\problemtypes\Examples\cmas2d_customlib.gid)
Inside it's folder, we can find the following files:

- **cmas2d_customlib.xml**

Defines some information about the problemtype, such as the name, the minimum GiD version, a description, etc.

- **cmas2d_customlib_default.spd**

Defines the tree interface.

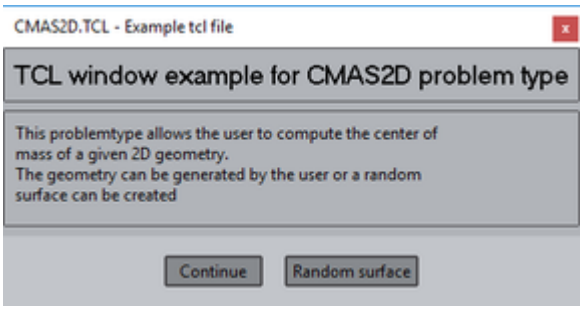
- **cmas2d_customlib.tcl**

Contains tcl code to process some GiD events, and the functions to write the input file.

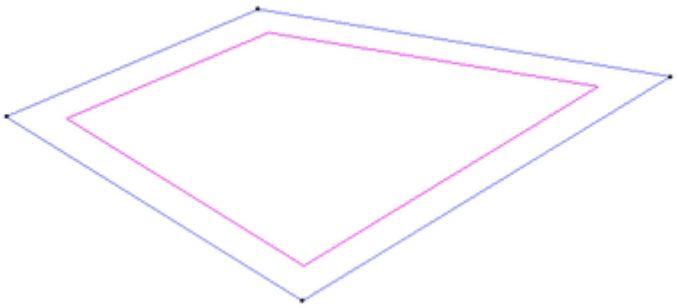
- **cmas2d_customlib.win.bat | cmas2d_customlib.unix.bat**

Script that launches the solver.

It's time to load the problemtype. Go to **Data->Problemtype->Examples->cmas2d_customlib.** The first you can see is a window like this:



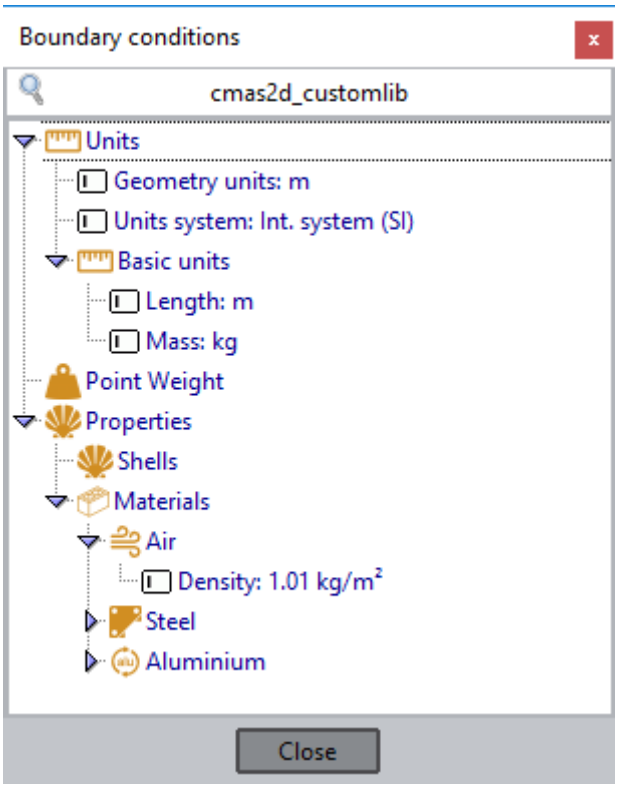
This window helps you to generate a random 4 sided surface. For this example let's click Random surface and get an auto-generated surface. You can click continue and create your own surface. This is the surface I'll work with:



After this, let's open the properties tree. Go to **Data->Data tree.**

Interface definition

In this section, we are going to prepare the interface definition document, called **cmas2d_customlib_default.spd**.
This is a file in XML format and contains all the definition of all the data necessary for the analysis.
First of all, let's see the final result:



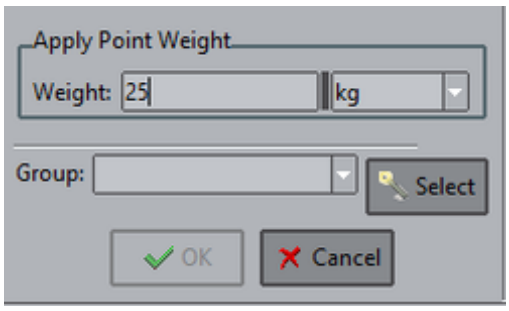
Step by step. The first we see is **Units**. It's useful to set a global criteria, such as the geometry units, or the default units system.

The next is **Point Weight**, to assign concentrated mass to a group that contains points. It's spd code is:

```
<condition n="Point_Weight" pn="Point Weight" ov="point" ovm="node" icon="constraints"
help="Concentrated mass">
 <value n="Weight" pn="Weight" v="0.0" unit_magnitude="M" units="kg" help="Specify the weight that you
want to apply"/>
</condition>
```

Let's introduce some concepts. The **'condition'** tag is used to assign properties to groups. The properties are defined in the **'value'** tags inside the 'condition'. For example, there is a property called 'Weight'. We can specify over which geometry entity will be assigned (point, line, surface and/or volume) in the **'ov'** field of the condition. For further information, check the customLib description of fields in the Customization Manual.
When we 'double click' on Point Weight, we can see this window, to assign a weight to a group, that can be created by clicking on the Select button. This will allow us to select some points in the geometry. Now assign a 25 kg point load to a point of the geometry.

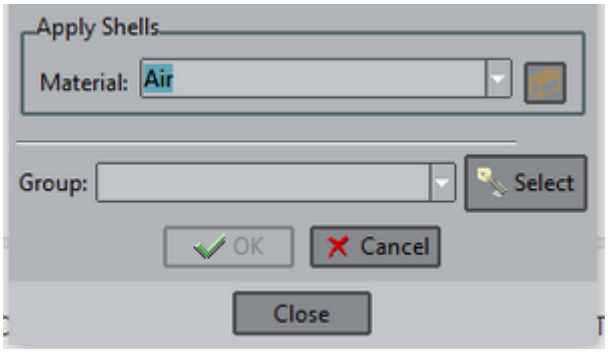
- Write '25' in the Weight field
- Click on '**select**' button and select one (or more) point.
- Press **ESC** to stop selecting. A group name will be generated.
- Click **OK**



Then we find 'Properties', a folder or 'container', that contains **'Shells'** and **'Materials'**. It's code is:

```
<container n="Properties" pn="Properties" un="Properties" icon="shells" help="Define your materials
database and apply them to the surfaces of your problem">
 <condition n="Shells" pn="Shells" ov="surface" ovm="element" icon="shells" help="Select your material
and the surfaces related to it">
  <value n="material" pn="Material" editable='0' help="Choose a material from the database"
values_tree='[GetMaterialsList]'>
   <edit_command n="Edit materials" pn="Edit materials" icon="material" proc='EditDatabaseList'/>
  </value>
 </condition>
 <include path="xml/materials.xml"/>
</container>
```

There is a 'container', another 'condition' called Shells, and a special 'value' called material. In this section, we want to assign a material from the database to a surface (see 'ov' field on the condition). NOTE: As you can see, there is an include to a file. The customLib library allows splitting the spd in different slices. You can find the materials database on that file in the problemtype folder.
By 'double clicking' on Shells, we get a window like this:



- Select a material from the list.
- Select the surface
- Press **ESC**
- Click **OK**

Your tree should look like this:

Writing the calculation files

Once created a geometry and assigned properties in the tree, it is time to write the input file.
For this example we will need to write the following information in a file:

- Number of elements and nodes of the model
- Coordinates of the nodes of the mesh
- Connectivities of the elements
- Number of materials used

- Density of each material used
- Number of point conditions
- Weight assigned to each point

Let's open the **cmas2d_customlib.tcl** file and see how are we processing the event of GiD that is called when the user wants to calculate, **AfterWriteCalcFileGIDProject**. After a few check of the environment, '**Cmas2d::WriteCalculationFile $filename**' is called (It is defined in the end of the same file).

TCL note: $filename is the content of variable **filename**, to define **ret** variable as "**-cancel-**" you must use "**set ret -cancel-**" and to use it "**$ret**"

First we need to do in this function is to call some initialization procedures:

To open the file for writting:
customlib::InitWriteFile $filename

To initialize the material's database, indicating wich 'conditions' have materials assigned.
customlib::InitMaterials [list "Shells"] active

Then we write some headers and to write the number of elements and nodes, we call some GiD_Info Functions:
customlib::WriteString "[GiD_Info Mesh NumElements] [GiD_Info Mesh NumNodes]"

To write the nodes and their coordinates we need to prepare the format and call WriteCoordinates:

customlib::WriteCoordinates "%5d %14.5e %14.5e%.0s\n"

As we can see, the format is prepared to write 2D coordinates (X & Y).

Next we need to write are the connectivities of the elements. For each element, we want to write it's id, it's nodes, and the material id that we assigned. In order to do this, again we prepare the parameters for the function WriteConnectivities:
set elements_conditions [list "Shells"]
set element_formats [list {"%10d" "element" "id"} {"%10d" "element" "connectivities"} {"%10d" "material" "MID"}]
customlib::WriteConnectivities $elements_conditions $element_formats

Then, the material's block. To get and write the number of materials, there is a function, GetNumberOfMaterials:
set num_materials [customlib::GetNumberOfMaterials used]
customlib::WriteString "Nº Materials= $num_materials"

And, to write the material information, again, we need to prepare the parameters to print the material's id and it's density, and call the function WriteMaterials
customlib::WriteMaterials [list {"%4d" "material" "MID"} {"%13.5e" "material" "Density"}] used

It is time to write the point weights. To get the number of nodes where we are applying the weights, we need to specify which is the condition we are writting, and call GetNumberOfNodes:
set condition_list [list "Point_Weight"]
set number_of_conditions [customlib::GetNumberOfNodes $condition_list]

And foreach node with a Point_Weight condition asigned, we need to print the node id and the asigned weight.
set condition_list [list "Point_Weight"]
set condition_formats [list {"%1d" "node" "id"} {"%13.5e" "property" "Weight"}]
customlib::WriteNodes $condition_list $condition_formats

Finally, all we need to do is to close the writting file
customlib::EndWriteFile

To test this on your example, you just need to Save your model (ctrl + s), Mesh it (ctrl + g), and calculate (F5). You can see the result of the writting process opening the file {modelname}.dat on the model folder.

Solver execution

At this point, we have generated a .dat file, containing the input data for the solver. It's time to launch it. In order to do it, we should have a .bat file (One for Windows and another for Unix based systems.
This script should delete the output files of previous executions (if exist) and launch the solver executable.

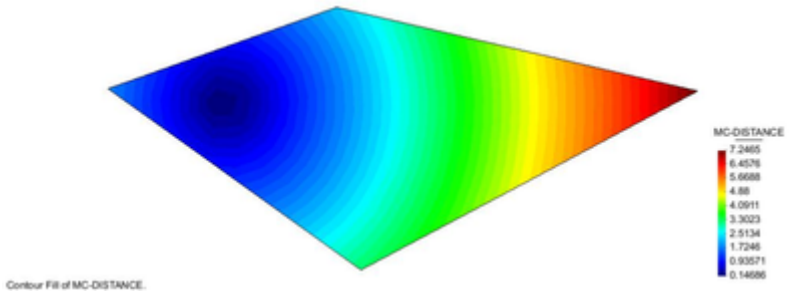
Check **cmas2d_customlib.win.bat and cmas2d_customlib.unix.bat** for more information.
The solver is located in the exec folder of the problem type. For this example, you can access to de **cmas2d.c** source file, and there are some compiled versions, for the common architectures.

Results

After the execution, the solver has generated some files:

- {projectname}.err : With the error information (If necessary).
- {projectname}.log : With some 'log' information.
- {projectname}.post.res : With the result data.

Access the .post.res file to see the format, and check the documentation about the Postprocess data files in the Customization manual.
It's time to change to postprocess and see our results. Go to **View results** in the menu, **contour fill** > **DISTANCE CENTER**
You should see something like this, but adapted to your generated geometry:



Extra: Wizard problemtype example

The current course focuses on a 'tree distribution' of the information. There is another example that implements a '**wizard distribution**', based on this tree one. You can find the wizard one on our Github site.

Advanced visualization tools

With this course you will learn how to manage the advanced visualization features present in GiD, available both in the pre and postprocessing parts.

Stereoscopic view

Model used
The model *pyramid.gid* is used in this example, which can be found at Material location.
Menu
View --> Advanced viewing settings...
Description
If you have an anaglyphic glasses you can try this option. The model can be set as an anaglyphic image in order to provide a stereoscopic 3D effect, when viewed with 2 color glasses (each lens a chromatically opposite color, usually red and cyan).
Anaglyphic images are made up of two color layers, superimposed. Since the glasses act as red and cyan filters we should be careful with the model's colors. To avoid problems we will change the contour fill color scale.

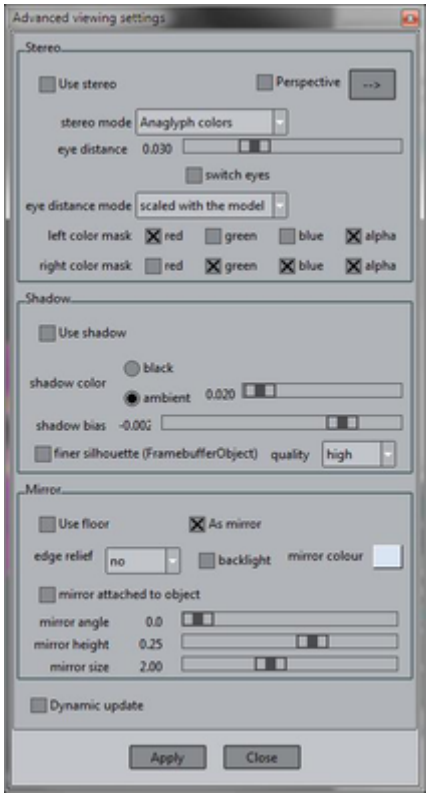
- From preprocess mode open the model
- Switch to postprocess mode
- In order to get a better view turn off the **Layer0** and **Layer8** layers

In order to test this option, first we will display a result with a given parameters in order to get a nice stereoscopic visualization.

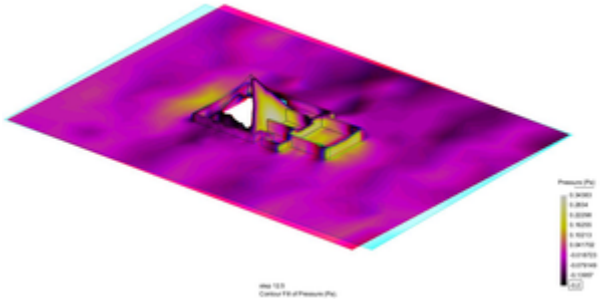
- Select the 12.5 step through **View results --> Default Analysis/Step --> RANSOL --> 12.5** or clicking on 
- Select **View results --> Contour Fill --> Pressure (Pa)**
- Select **Utilities --> Set contour limits --> Define limits...** through the menu bar or clicking on 

Choosing the first option the Contour Limits window appears. With this window you can set the minimum/maximum value used for Contour Fills.

1. Check the **Min** checkbox and change the value to **-0.2.**
2. Click on the **Apply** button and **Close** the window.
3. Open the **Preferences** window, and go to the **Postprocess --> Contour fill and lines** branch. There, set the Color map to **3D Anaglyphs 2**, and click Apply.
4. The Stereoscopic visualizatoin settings, as well as other advanced ones, are centralized in the **View --> Advanced viewing settings...** window. Open it.



5. Check the **Dynamic update** option (from the lower part of the window) in order to change the options without the need to click the Apply button.
6. Check the **Use stereo** option, and change the eye distance to the value where you can see the 3D effect properly. The image should look like this:



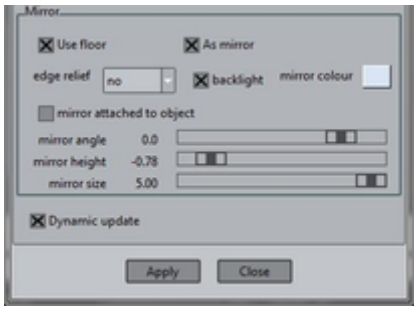
Let's restore the visualization settings for the next points of the course:

7. Uncheck the **Use stereo** option and **Close** the window.
8. Set the Default values in the **Postprocess --> Contour fill and lines** branch of the **Preferences** window.
9. Select **View results --> No Results**

Mirror effect

Model used
The model *Ferrari_Maranello_2002_visualization.gid* will be used in this example, which can be found at Material location.
Menu
View --> Advanced viewing settings...
Description
With this option enabled the model is mirrored on a ground surface, or it his ground plane can be used as floor (shadows are drawn on it).

1. From preprocess mode open the model.
2. Change the render mode selecting **View --> Render --> Smooth.** It can be seen (checking the other render modes) that the model is represented by a mesh.
3. Open the **View --> Advanced viewing settings...** window, and set the **Dynamic update** option in the lower part of it.

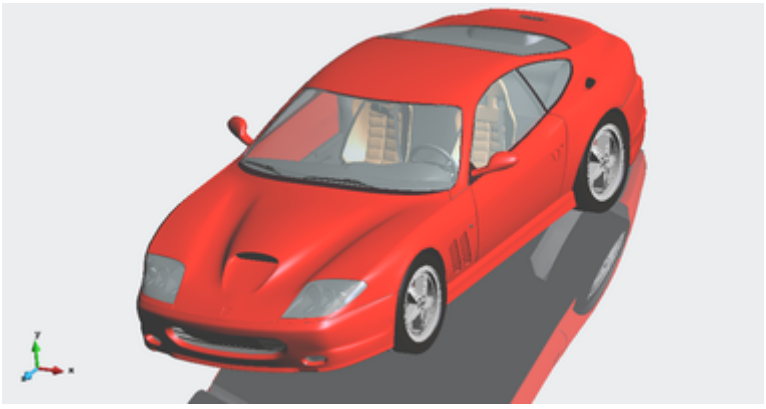


4. Look for the **Mirror** section of the window, and check the **Use Floor** option and a ground surface will appear under the model.
5. Check the **As mirror** option in order to get the model reflected in this ground surface.
6. Check the **backlight** option to get a better view

Several options can be set in order to get a better view of the reflection:

- **Mirror angle**: Changes the inclination angle of the ground
- **Mirror height**: Changes the height of the floor, relative to the object
- **Mirror size**: Changes the size of the ground

7. Play a little bit with these options until you get the desired view:



Shadows

Models used

The models *Barcelona_for_shadows.gid* and *Ferrari_Maranello_2002_visualization.gid* will be used in this example, which can be found at Material location.

Menu

View --> Advanced viewing settings...

Description

Shadows provide not only a better depth perception of the floating objects, but also provides more realism to the viewed model and results. The technique used inside GiD to create shadows is called *shadow mapping*, in which:

- first, a shadow map is created by rendering the scene from the light's point of view;
- then, the scene is rendered from the user's point of view by checking whether the pixels is in shadow or not:
  - pixels which are directly lit are drawn as always;
  - pixels in the shadow area are drawn in black or with a dimmed ambient light.

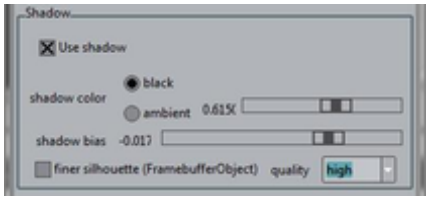
With shadows enabled the scene is rendered at least twice, and sometimes three times, on some hardware and drawing the shadowed area with dimmed ambient light.

The shadow map is created for the whole model and not just the zoomed area. To minimize the *staircase* effect on the border of the shadows, the *finer silhouette* option can be used with different qualities.

Note: using shadows will slow down the frame rate of the visualization.

Note: the shadow map is a *type* of image which is stored in the graphics card, besides the visualized mesh information; selecting a too high quality for the *finer silhouette*, may result in a very poor performance, or even crash the program, if the memory of your graphics card is too low.

Shadows can be enabled through *View --> Advanced viewing settings...* which will pop-up the window with the shadow options.



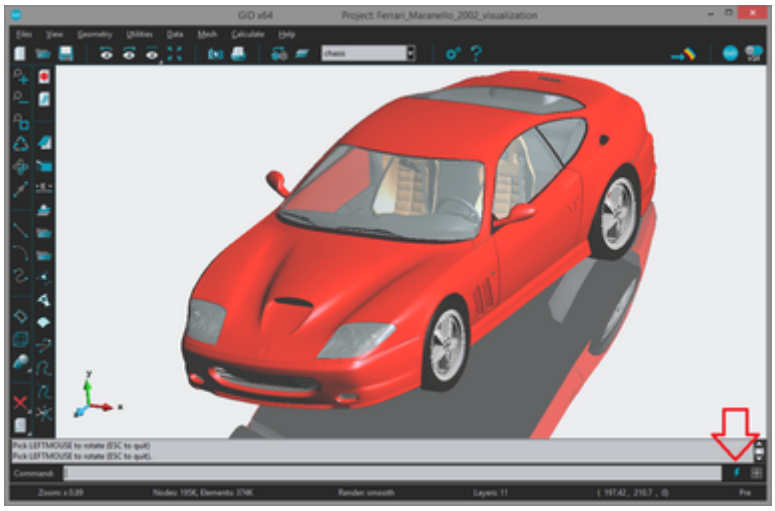
Options

- **Use shadow** enables or disables the shadows visualization;
- **shadow color** this option controls if the shadows should be black or should be drawn with a dimmed ambient light.
- **shadow bias** this options allows the user to adjust the offset between the occluder and the shadow, the default value is -0.002;
- **finer silhouette** this advanced option allows the user to control the granularity of the shadow, i.e., the resolution of the texture to be used to create the shadow. By default, i.e. deactivated, the same graphical window is used to created the shadow texture. An accelerated OpenGL 2.0, or the frame-buffer object extension is needed for this option to be used. If checked, the options are:
  - **medium** which uses a texture of 1024 x 1024 pixels to create the shadow map, using 4 MB of memory on the graphics card,
  - **high** which uses a texture of 2048 x 2048 pixels to create the shadow map, using 16 MB of memory on the graphics card,
  - **very high** which uses a texture of 4096 x 4096 pixels to create the shadow map, using 64 MB of memory on the graphics card,
  - **highest** which uses a texture of 8192 x 8192 pixels to create the shadow map, using 256 MB of memory on the graphics card.

Note: *medium* and *high* qualities may work with all king of graphics cards, even in Linux software mode, safe mode; but check the memory of your graphics card before choosing the *very high* or *highest* qualities for shadows. For these two last qualities, the graphics card should at least have 512 MB of memory.

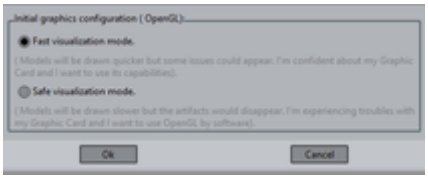
Shadows Requirements

To get the best experience the user should check if GiD uses the graphics card or not. This can be done by looking the graphics acceleration icon at the lower right of the main window (a lightning):



The blue lightning icon in the lower right part of GiD window indicates that the graphics card is used to accelerate the visualization.

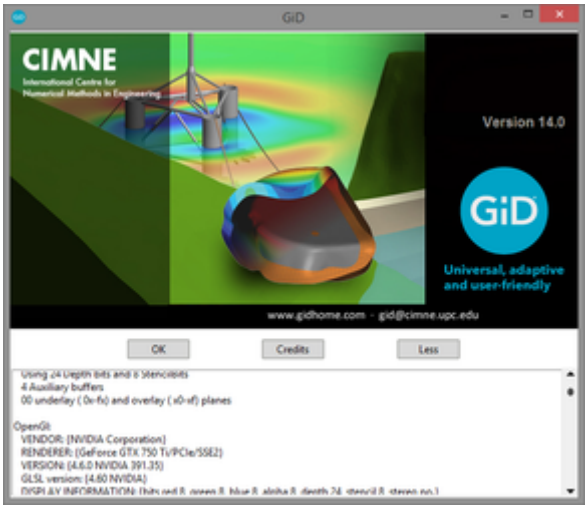
The graphics configuration can be changed by clicking on this icon, which opens the following window:



or at the *Graphical --> System* branch of the *Preferences window* under *OpenGL options*.

Window users

To use shadows GiD requires at least OpenGL version 1.5. The OpenGL version can be checked at *Help --> About*:



System information about the graphics can be found pressing the More button in the *Help --> About* window. Scrolling down, the OpenGL version can be checked.

Linux users

In Linux, the software mode of GiD uses the [Mesa 3D] library, an open source implementation of the [OpenGL] specification, which provides at least OpenGL version 2.1.

Shadows parameters

Open the *Ferrari_Maranello_2002_visualization.gid* model from the Material location. It has to be noted that the visualization results depends strongly on the graphical card of each computer, so it is common not to get the same visualizations than the ones shown in this course.

1. Select **View --> Advanced viewing settings...**

2. Enable **Dynamic update** option in the lower part of the window to avoid clicking Apply button after each parameter setting. Enable **Use shadow** option and select **black** as shadow color (you may change the shadow bias parameter in order to calibrate your visualization). When zooming, it can be appreciated that the border of the shadow shows a *staircase* effect.

Coarse shadows

Zoom where it can be appreciated the staircase effect

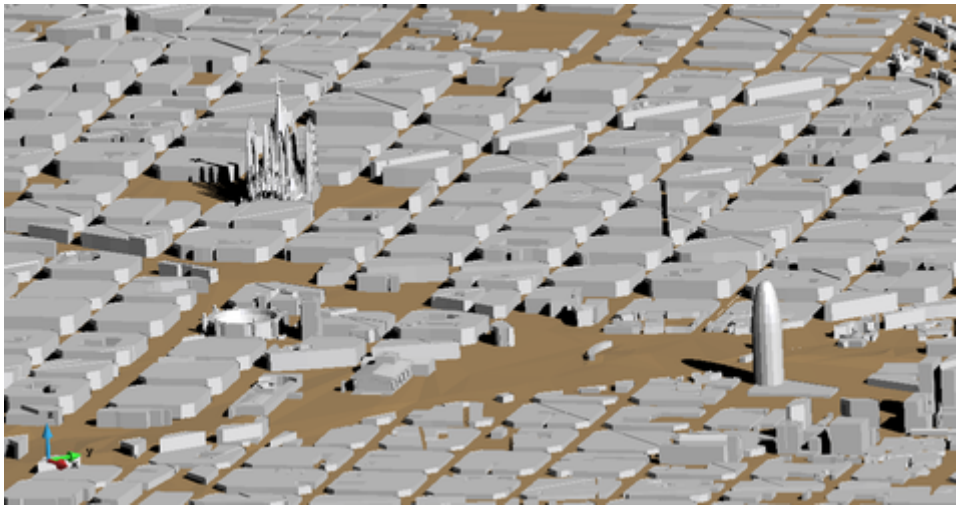
3. Enabling the **finer silhouette** option and setting different qualities it can be appreciated how the shadows visualization improves. Also, setting the 'ambient' shadow color results in a more smooth visualization.

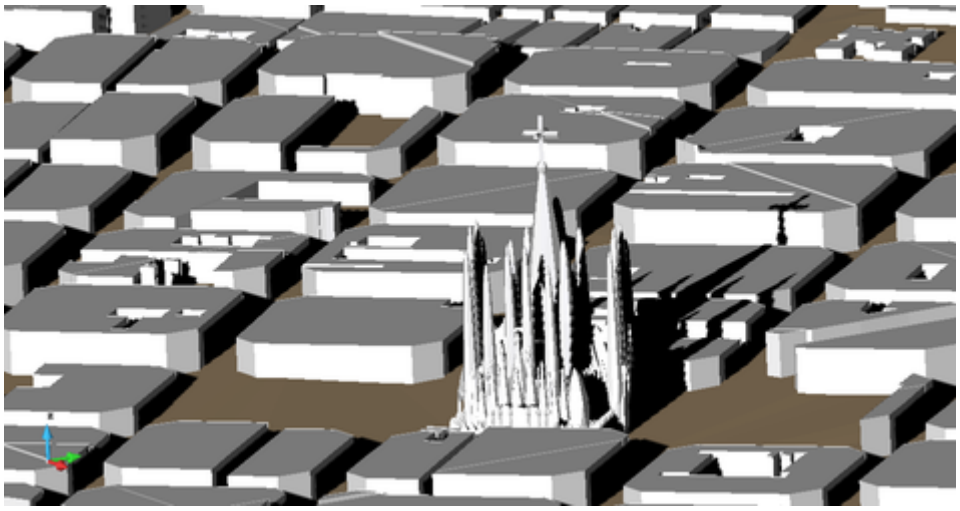
Improved shadows visualization with finer silhouette with very-high quality

Zoom view ('cristal' Layer on) of the improved shadows visualization

Shadows: Change light direction

Using the option **Render --> Change light dir** of the contextual menu, the direction of the light, and thus of the shadow, can be interactively changed.
Open the model *Barcelona_for_shadows.gid* to see the effects of this. This model represents part of the city of Barcelona, including the 'Sagrada Familia'.
Setting the shadows options and changing the light direction nice visualizations can be obtained, as the ones shown in the following figures.



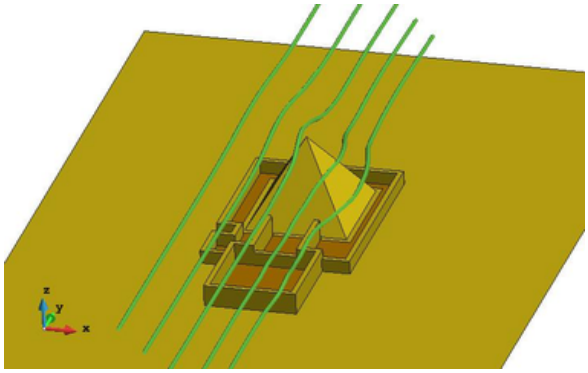


(Check that 'shadow bias' are at -0.001 and finer silhouette on)

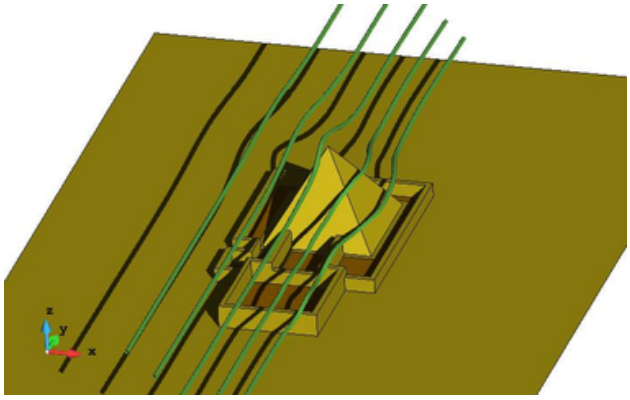
Shadows: Example with streamlines

As it has been pointed out, all the visualization features of GiD can be applied both to the pre and the postprocessing part of GiD.
Let's see an example of useful shadows to demonstrates better depth perception of the created stream lines: which ones are near the ground and which ones not.

1. Open the *pyramid.gid* model, and switch to the post-process mode in GiD.
2. Select **Files --> Import --> Stream lines...** and choose pyramid.flavia.*streams.msh* located in *pyramid.gid.*
3. Using shadows, the height of the stream lines are better perceived than without.



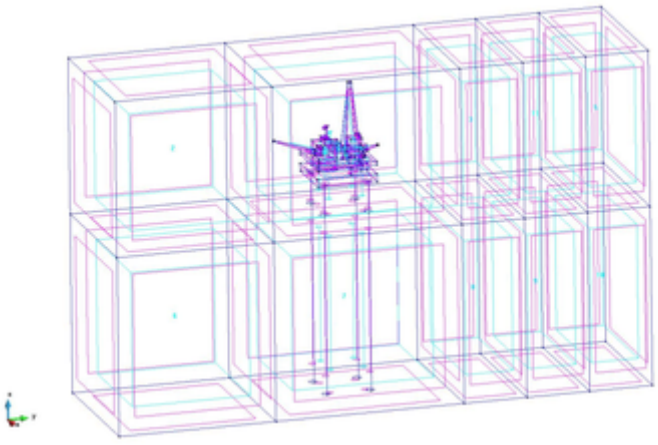
stream lines without shadows



shadows helps to perceive the relative position of the stream lines

Advanced visualization tools: Combining pre and postprocess models

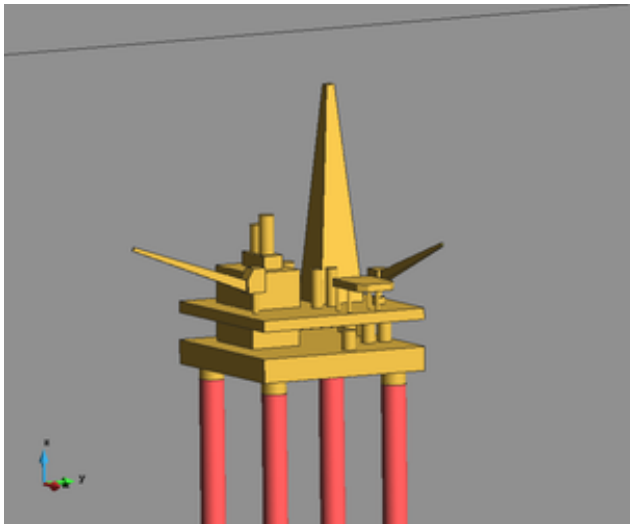
You can also combine the visualization of the geometry o preprocess mesh and the postprocess results. Here is an example.



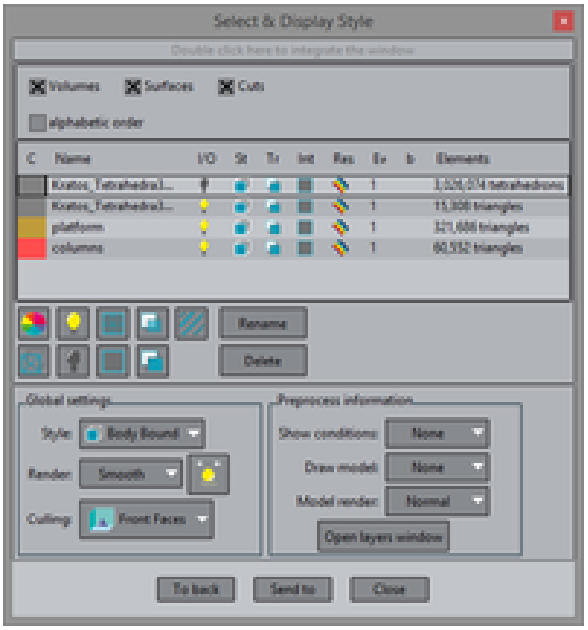
Geometry of the platform_small *project.*

This example is a simulation of waves against a oil platform. The model *platform_small.gid* can be found at Material location.

1. open the *platform_small.gid* project,
2. go to postprocess,
3. select the menu *Geometry --> Extract boundaries;*
4. select *All lines* display style;
5. create a new set by selecting elements of the platform (using the *Send to --> new set* option of the *Display style* window), change its colour and switch it off;
6. create a new set by selecting elements of the columns (using the *Send to --> new set* option of the *Display style* window), and change its colour;
7. use culling to show the interior of the volume: 

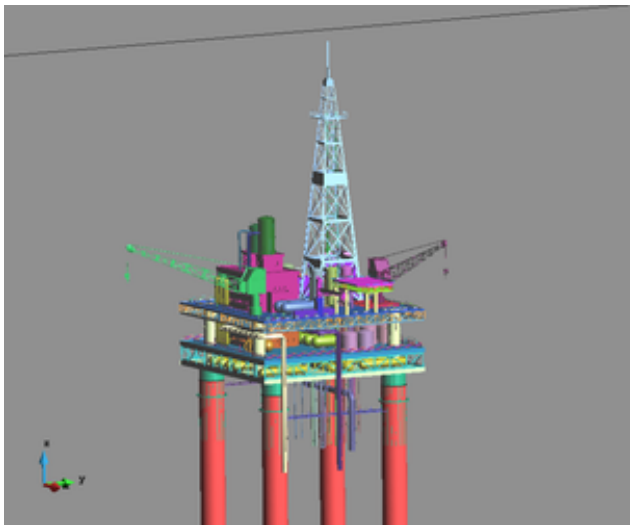


The single triangle mesh of the volume boundary has been separated in the grey box triangle set, the golden platform and the 'middle red' columns.

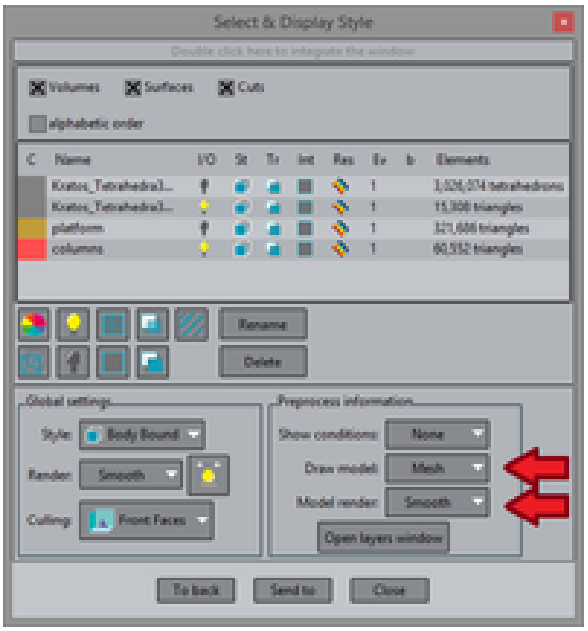


Listing the new created mesh sets. On the right, enabling the culling of the front faces.

8. switch all surface sets on and enable the culling of the font faces.
9. switch off the *platform* set, enable the visualization of the preprocess mesh, using the smooth render:

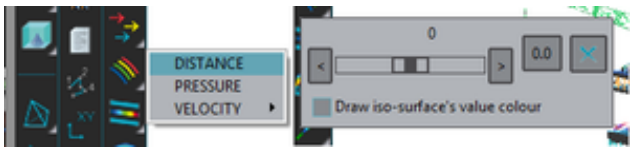


Visualizing both the preprocess and the postprocess mesh

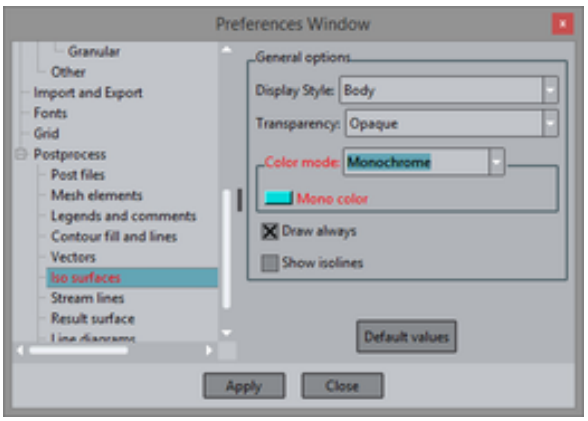


Enabling the visualization of the *preprocess mesh, smooth render*

10. Now set the 'grey box' mesh style to boundaries, and turn off the culling.
11. Do an iso-surface of DISTANCE with value 0.0, and make its color blue-sky:



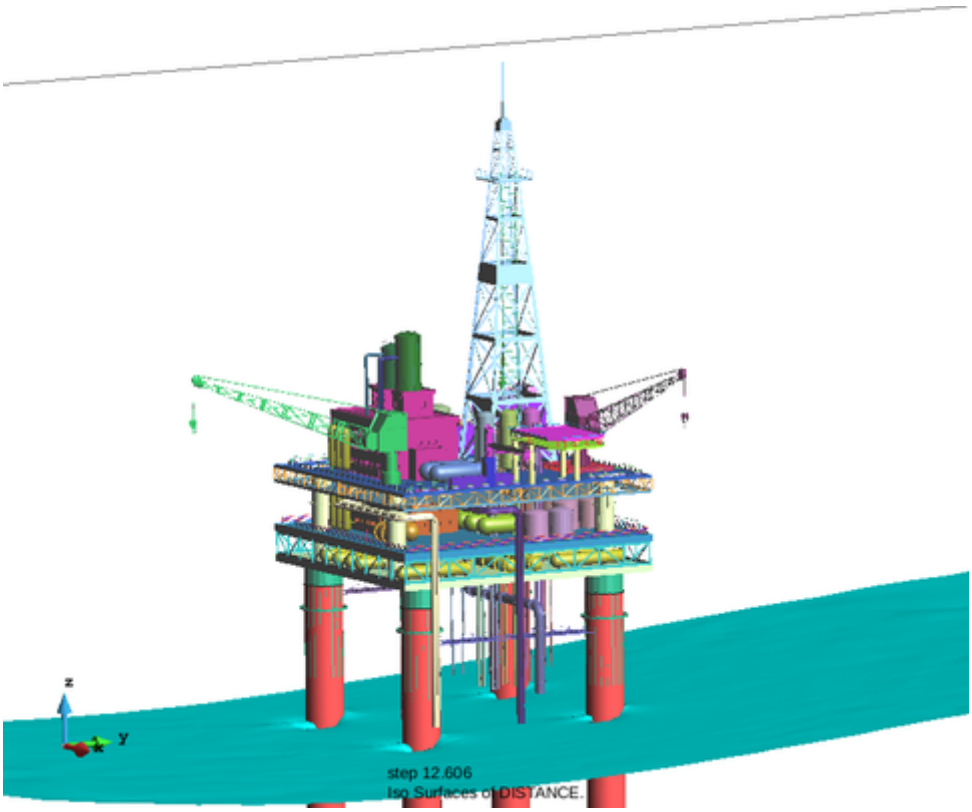
Creating the isosurface with value 0.0



Setting the iso-surface colour to blue.

12. Animate the result with the Windows --> Animate window

The achieved result is shown hereafter:

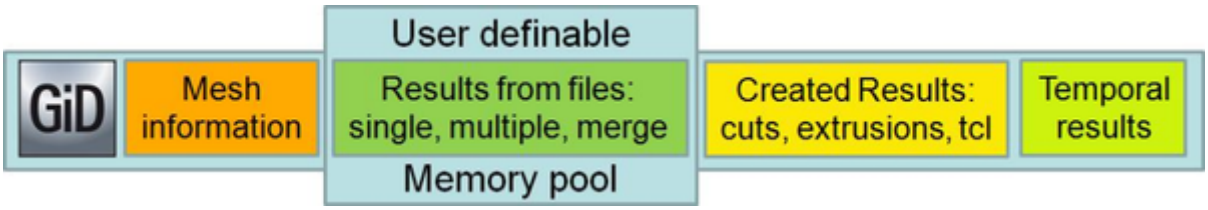


Working with large models

About Result's cache

Results' cache is a mechanism implemented in GiD which allows to analyze huge results files which do not fit in memory. The whole mesh model resides in memory but only the results used to render a contour fill, isosurfaces, etc. are loaded in memory, in a user defined pool. First this mechanism is explained in detail, including some considerations and limitations, and then an examples shows the benefits of this mechanism.

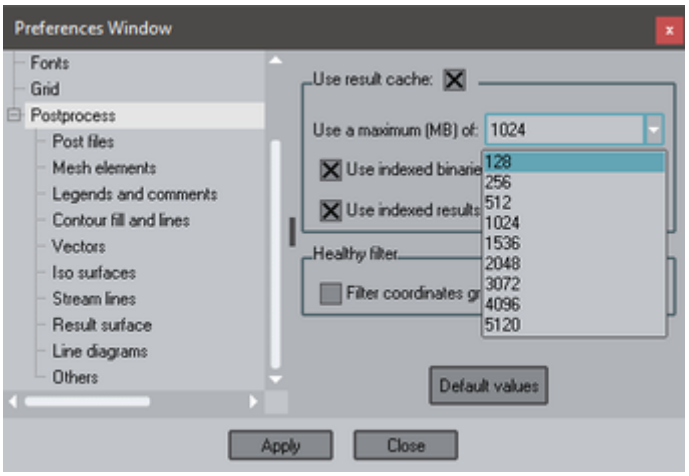
Result's cache: How it works



Results cache block diagram

The **Result's cache** mechanism allows the analysis and visualization of lots of results which, otherwise, could not be held entirely in memory.
Instead of loading all results from the file(s), a certain amount of memory, a memory pool, is reserved and used to load and unload the result values as they are needed.
When this mechanism is enabled, with the *indexed* options enabled, and a result's file is opened for the first time, GiD verifies the correctness of the file and gets some information about the results, such as minimum and maximum values, amount of memory needed, position in file; and this information is stored in the index file. The next time the same result's file is opened, only the index file is loaded, reducing the load time considerably. But no results are loaded, they are loaded only on-demand.
If the *indexed* options are off, then no index file is created or read, and so the parsing of the result's file is perfomed each time is openend, but no results is loaded in memory. They are loaded on demand.
If a result is selected to do, for instance, a *contour fill* visualization and their values are not in memory, then GiD checks if there is enough space in the memory pool and loads them. If their values are already in memory, the time-stamp of the result is actualized.
If there is not enough space in the memory pool to load the desired result, then the oldest results are unloaded, and their memory freed, until there is enough memory to load the desired result.
What's cached: Only results which are already stored in files are cached, i.e. files read with *Files --> Open*, *Open multiple* or *Merge*.
What's not cached: When cuts, extrusions are done or isosurfaces are converted to full featured meshes, the generated results are held in memory. Results created in GiD using the *Window --> Create result*, *Create statistical result* are held in memory. Also results imported using *Files --> Import* or using the *plug-in* mechanism or the TCL procedure *GiD_Result create ...* are held in memory too.
In order to cache these results, save the model, with *Files --> Export --> Post information --> whole model / HDF5 GiD post* and open it again.

Result's cache options:



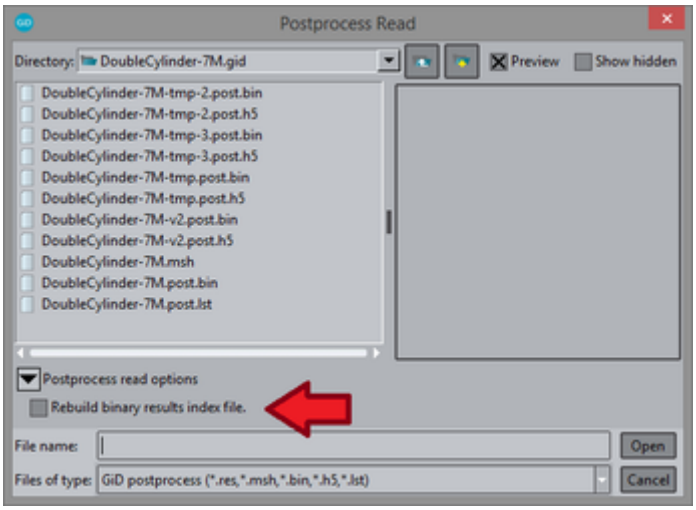
- The user can **enable the Result's cache** in the *Post files panel* of the *Utilities --> Preferences* window. If you change this setting you'll need to read the post-process file again.
- The **size of the memory pool** can also be adjusted by selecting one of the predefined memory sizes or entering the desired amount in the same entry. The size can be adjusted according to not only the amount of memory the computer has, but also the memory used by a single result. You may change this value during your session.

For instance, a mesh with one million nodes with a vector result at each step, the amount of memory needed to hold the nodal vector result

of a single step will be: 4 components (x, y, z and modulus) * 4 bytes per float * 1 million nodes = 16 MBytes of memory. If there are 100 steps in the analysis,, to get a fluid animation. the memory pool must be set to 16 * 100 = 1.6 GBytes of memory. But if the animation is of a contour fill or an iso-surface of a scalar result, then the amount of memory needed is reduced by 4, to 400 MBytes.
Other interesting options to save loading and access times from huge files are:

- **Use indexed binaries:** will speed up the access of the results on huge file, more over if the user access them randomly. It onlly applies to raw binary result files, not to hdf5 gidpost files.
- **Use indexed results information:** stored in the indices, will speed up the loading of huge results files, as the results information, except values, are already stored in the index files.It onlly applies to raw binary result files, not to hdf5 gidpost files.

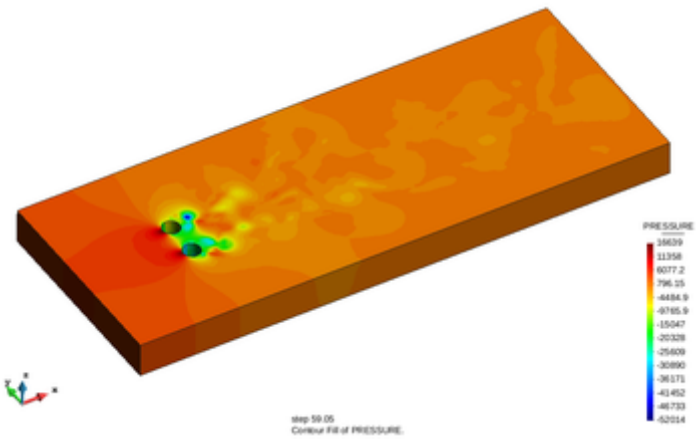
Note: The index of the results' file is automatically created for raw binary postprocess files, if it does not exists or if the results' file is newer than the index file. HDF5 GiDPost files do not need an index file. If for any reason GiD has problems reading the index file, or the information stored in this file is not actualized, the user can recreate these index file in the *File --> Open* dialog box, under the *Postprocess read options*:

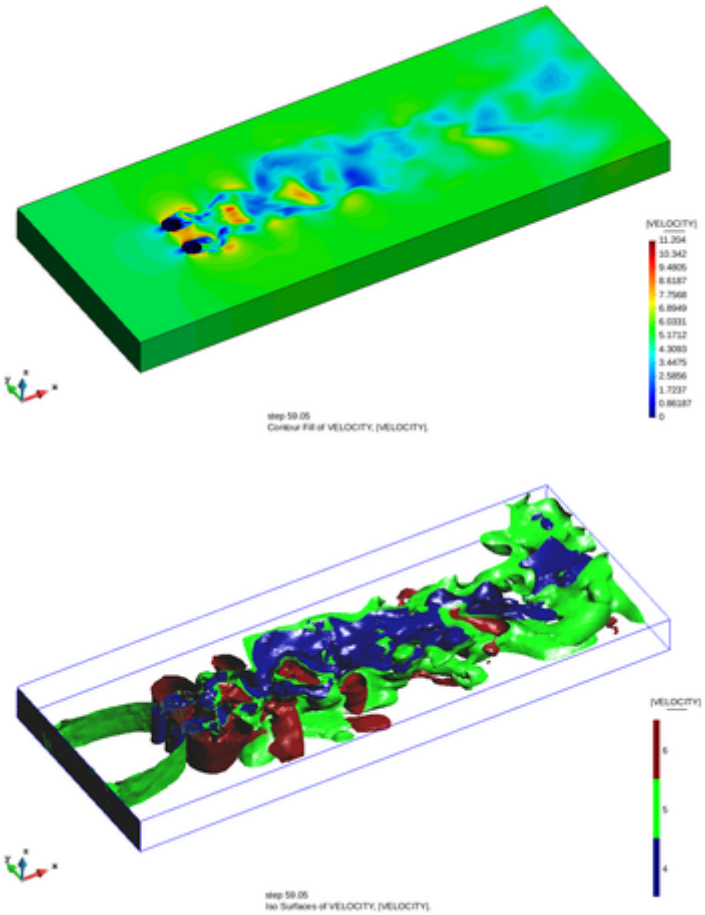


Caution

When the **result's cache** is used, the result's files remains opened, to make the retrieval of the results faster.
If a simulation is done on a cluster where the model is partitioned in 1024 pieces and 1024 separated result's files are generated, when these 1024 result's files are merged in GiD with the result's cache mechanism enabled, then the 1024 files will still be opened!
Usually in Linux the maximum number of opened files are, precisely, 1024, and some of them are already used, causing GiD to display an error when the user tries to merge these 1024 result's files.
This limit on the number of open file descriptors is one of the limits imposed by the system, like the *cputime*, *coredumpsize*, *datasize*, etc.
There are two types of limits in Linux, soft limits and hard limits. Usually the soft limits can be changed by the user, but administration privileges are needed to change the hard limits.
Sometimes soft and hard limit are not the same, and the user is able to raise the number of *open file descriptors* to its hard limit. If the user uses the **bash** shell, then the commands *ulimit -Sn* (open files soft limit) or the *ulimit -Hn* (open files hard limit) should be checked. An additional parameter can be entered to modify the limit, for instance *ulimit -Sn 2048*. In **csh/tcsh** the command is *limit [-h] openfiles*

Result's cache Example





DoubleCylinder-7M snapshots.

This example is a simulation of the flow around two 3D cylinders. The model *DoubleCylinder-7M-v2.post.h5* can be found at Material location , as it is a big model, it may not be in the USB memory stick, but in the ftp server.
Loading the model in GiD with the result's cache disabled, more than 2 gigabytes of memory will be used, making it difficult, if possible, to view it and work with it on a 32 bits platform.

To make sure the model can be loaded and handled in GiD with a 32 bits operating system, the **results cache** should be **enabled**, and the **memory pool** should be set to **128 MB** before loading the model. With this results cache size, you'll be able to read the model using less than 2 GB of memory.

If you read the *DoubleCylinder-7M-v2.post.**bin*** file:

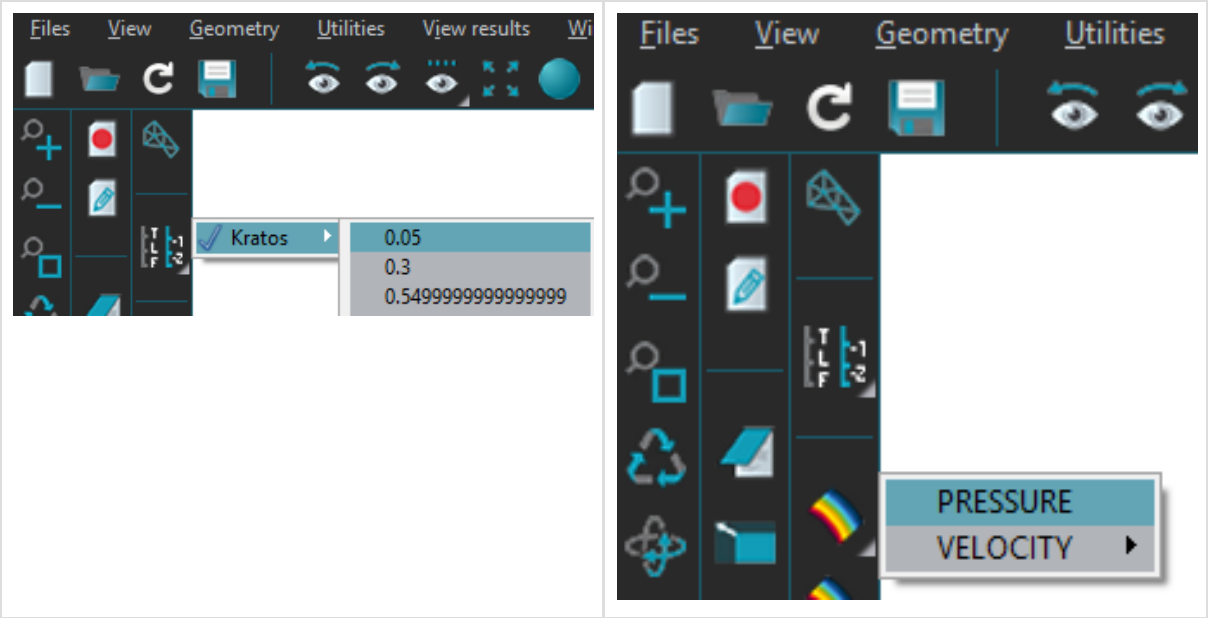
Enable the use of **indexed binaries** and **indexed results information**.
After the initial creation of the index file *DoubleCylinder-7M-v2.post.bin.idx*, the next sessions will open the model in less than 30 seconds, the time to read the mesh, create the graphical objects to visualize the mesh and read the results index.

If you read the *DoubleCylinder-7M-v2.post.**h5*** file:

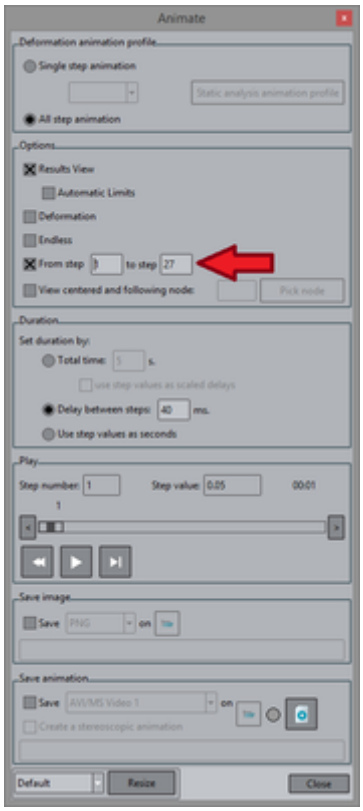
There is no need to use an index file, as the information is already inside the hdf5 file. It will take less than 45 seconds to read the mesh, create the graphical objects to visualize the mesh and get the results list.

After loading the model in GiD with these settings, in the *task administrator* can be seen that GiD has used less than 2 GBytes of memory, and is using around **700-900 MBytes** of memory, depending on the file (.bin or .h5) and GiD version (32 or 64 bits).

1. First check in the *Utilities* --> *Preferences* window that the **Result's cache** is enabled and set to **128 MB**.
2. load the Double Cylinder 7M example;
3. switch off the surface sets and let **only the volume meshes on**;
4. Select the first time step and do a Contour Fill of Pressure:



5. Open the *Window --> Animate* window, set the *Options* ***From step*** to **1** and ***To step*** to **27** and press the Play button:



6. Press the *Rewind* button and play it again.

You'll notice that the first time it takes some time to draw the result at each time-step. This is so, because it read the result from the file and then renders it. The second time the Pressure result for each time-step is already in memory and thus the animation is quickier. Check the *Ta sk manager* to see how much memory GiD is using.

- Now increase the option ***To step*** to **32** and press the *Rewind* and the *Play* button.

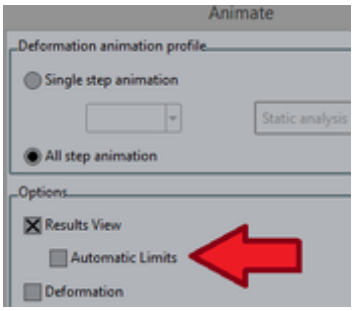
You'll notice that the first 27 animation steps are done faster and that the last 5 time-steps (from 28 to 32) is somewhat slower. This is because it read the Pressure result from file.

If you press the Rewind button and play the animation again, you'll notice that the animation is slow, i.e. not as fast as when we're doing the animation from step 1 to 27 !

This is because of the size of the Result's cache:

1. Check the **number of nodes** of the model with *Utilities* --> *status*. You'll see that the model has 1,208,206 nodes.
2. For the Pressure result, which is stored as 4-byte float, this represents around 4.6 MBytes of memory.
3. We have allocated 128 MB as results cache, which can hold 27 time-step of the pressure result.
4. As we're animating 32 time-steps, GiD keeps freeing and reading all time-steps.
5. Now **increase the result's cache to 170 MB** and do the animation from step 1 to step 32 again.
6. You'll notice that now the animation runs faster as it has more memory to store the Pressure result

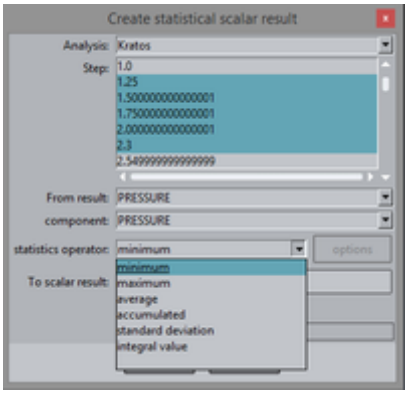
NOTE: be aware that if you use the *Automatic Limits* option of the Animate Window:



GiD will go through the Pressure result of all time-steps, i.e. will load each one of the Pressure result of each time-step, taking some time to compute the minimum and maximum Pressure value. If you already know the minimum and maximum values to set, it's recommended that you set them previously, for instance from the icon bar:

Results statistics:

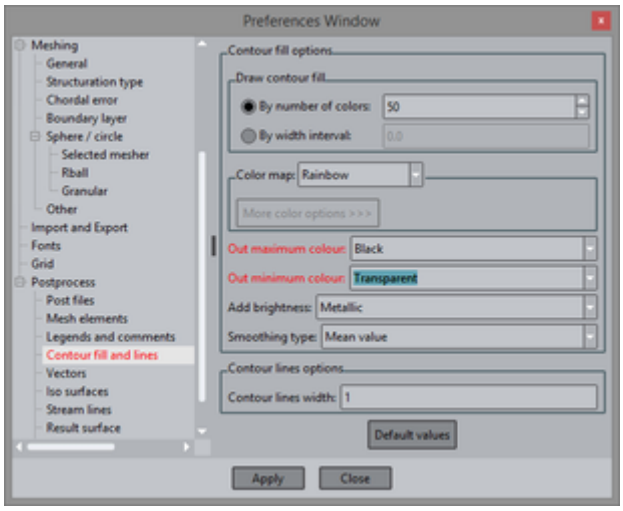
To get an approximate idea of the minimum and maximum values, we can use *Windows* --> *Create statistical results...*:



This window allows the creation of several statistical results: minimum, maximum, average, accumulated, standard deviation and integral value. These statistical results are created in the last step of the analysis.

1. Create the statistical result *PRESSURE//**minimum*** and *PRESSURE//**maximum*** between steps **1.0 and 5.05**.
2. Do a contour fill of PRESSURE//minimum and get the minimum value (**-62,235**).
3. Do a contour fill of PRESSURE//maximum and get the maximum value (**22,232**).
4. Use these values to fix the limits of the contour fill:

5. Do an animation.
6. To check how the values outside these fixed limits are drawn, go to Preferences --> Postprocess --> Contour fill and lines:

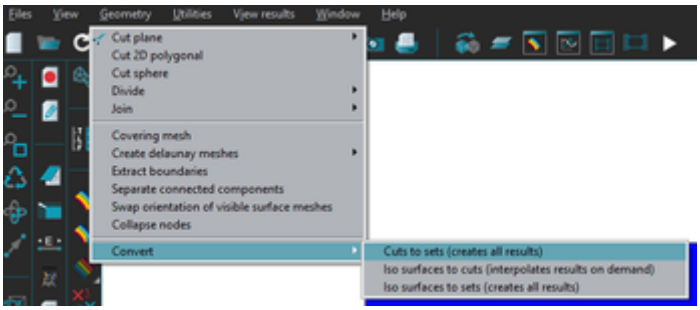


Results for cut planes are not stored in the results cache:

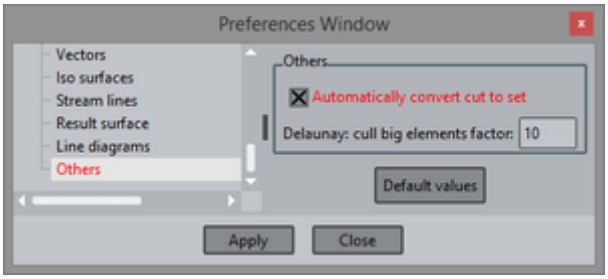
Cut planes in GiD are represented in two different ways:

- **implicit cuts**: when the the user creates a cut plane, together with the cut mesh, GiD stores interpolation information about the cut and every time the user selects a result to visualize it is interpolated to the cut plane and visualized.
- **explicit cuts**: when the user creates a cut plane, besides the cut mesh, GiD interpolates all results for all time-steps to the cut mesh, thus increasing its memory usage.

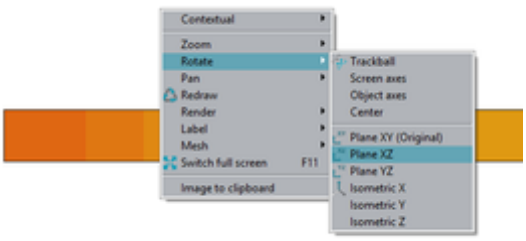
implicit cuts can always be converted to explicit cuts, using the option *Geometry --> Convert --> Cuts to sets*:



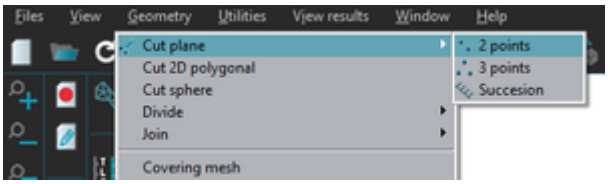
To autamatically convert implicit cuts to explicit ones, use the preference Post-*process --> Other --> Automatically convert cut to sets*: (/But for this course we'll let it unchecked, read the note below)



1. Check that the preference *Postprocess --> Others --> Automatically convert cut to sets* is disabled.
2. Check the memory GiD is using in the *task manager*: a around **900 MB**.
3. Do *Rotate --> Plane XZ:*



4. Do *Cut plane --> 2 points*, (after clicking the 1st point, you may press the *Alt key* to create a cut plane complettely horizontal):



5. Animate the result from step 1 to step 27 twice.

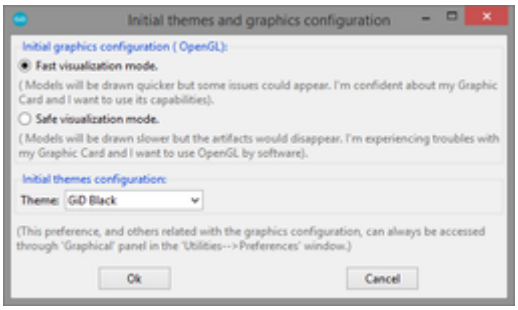
NOTE: at the moment the use of *Convert cut to sets* is **discouraged with Results cache enabled**. Only if you increase the Results Cache Memory size to hold all results in memory you'll be able to convert cuts to sets!

Using the graphics card

This section explains how GiD can use the available graphics resources and which features can be used under the selected visualization configuration.
After explaining how GiD can use the graphics capabilities, the different drawing methods are explained and their advantages. Finally an example shows the different performance numbers with several visualization and drawing settings.

Using the graphics card: Fast/safe visualization mode

Initial GiD configuration window, when GiD is launched for the first time:



- **Safe visualization mode:** GiD will render the model and mesh using only the CPU, i.e. will not use hardware acceleration of the graphics card. Using this option some artifacts may dissapear but models and meshes will be drawn slower. *In MS Windows*: the OpenGL version used is 1.1 and some features will be disabled. *In Linux:* this mode can also be enabled using the gidx script in a console or terminal window. The OpenGL version used is 3.0.
- **Fast visualization mode:** GiD will use the hardware acceleration provided by the driver and the graphics card to accelerate the visualization of the models. If some problems appear, please update your driver or use the *Safe visualization* mode.

Note: when using **Intel graphics** with Fast visualization mode it is strongly recommended to **enable** the option **Selection lines by software,** because dynamic lines, such as the ones used when performing a zoom, creating geometries or creating cut planes, are not drawn by the Intel graphics. To check which visualization mode GiD is using you can check the *card* like icon on the bottom right of the main graphical window, or check the *Help --> About* window and pressing the *About* button:

Graphics mode with the *Black theme:*

On the left, *Fast visualization mode* is enabled. On the right, GiD is in *Safe visualization mode*.

Graphics mode with the *Classic theme*:

This is an example of the text that appears on *Help --> About --> More* in **Fast** Visualization mode (**MS Windows**):

GiD internal version: 12.0 (64 bits)
Operating system: 'amd64' 'windows' 'Windows NT' '6.1'
Tcl8.6.1 Tk8.6(8.6.1)
OpenGL 3.3.0 on GeForce GTX 275/PCIe/SSE2
GLEW 1.9.0

Togl 1.7: asked for an accelerated renderer,
Asked Pixelformat description (10):
Hardware accelerated (ICD) implementation, PixelFormat 10,
DoubleBuffer: yes,
StereoBuffer,
RGBA, Composition enabled (Vista), Draw to Window, OpenGL support,
Using 24 Color bits: 8 red, 8 green, 8 blue and 8 alpha,
Using 0 Accumulation bits: 0 red, 0 green, 0 blue and 0 alpha,
Using 32 Depth bits and 8 StencilBits
0 Auxiliary buffers
00 underlay (0x-fx) and overlay (x0-xf) planes
Gotten Pixelformat description (10):
Hardware accelerated (ICD) implementation, PixelFormat 10,
DoubleBuffer: yes exchange,
RGBA, Composition enabled (Vista), Draw to Window, OpenGL support,
Using 32 Color bits: 8 red, 8 green, 8 blue and 8 alpha,
Using 64 Accumulation bits: 16 red, 16 green, 16 blue and 16 alpha,
Using 24 Depth bits and 8 StencilBits
4 Auxiliary buffers
00 underlay (0x-fx) and overlay (x0-xf) planes

OpenGl:
VENDOR: {NVIDIA Corporation}
RENDERER: {GeForce GTX 275/PCIe/SSE2}
VERSION: 3.3.0
GLSL version: {3.30 NVIDIA via Cg compiler}
DISPLAY INFORMATION: {bits red 8, green 8, blue 8, alpha 8, depth 24, stencil 8, stereo no.}
LIMITS: {
Max Texture Size: 8192
Max Elements Indices: 1048576
Max Elements Vertices: 1048576
Max 3D Texture Size: 2048
Max Cube Map Texture Size: 8192
Available video memory size: 572.379 MBytes
Maximum working number of elements: 9769 K
}

This is an example of the text that appears on *Help --> About --> More* in **Safe** Visualization mode (**MS Windows**):

GiD internal version: 12.0 (64 bits)
Operating system: 'amd64' 'windows' 'Windows NT' '6.1'
Tcl8.6.1 Tk8.6(8.6.1)
OpenGL 1.1.0 on GDI Generic
GLEW 1.9.0

Togl 1.7: asked for a generic renderer,
Asked Pixelformat description (93):
Generic implementation, PixelFormat 93,
DoubleBuffer: yes,
StereoBuffer,
RGBA, Composition enabled (Vista), Draw to Window, OpenGL support,
Using 24 Color bits: 8 red, 8 green, 8 blue and 0 alpha,
Using 0 Accumulation bits: 0 red, 0 green, 0 blue and 0 alpha,
Using 32 Depth bits and 8 StencilBits
0 Auxiliary buffers
00 underlay (0x-fx) and overlay (x0-xf) planes
Gotten Pixelformat description (93):
Generic implementation, PixelFormat 93,
DoubleBuffer: yes copy,
RGBA, Composition enabled (Vista), Draw to Window, OpenGL support,
Using 32 Color bits: 8 red, 8 green, 8 blue and 0 alpha,
Using 64 Accumulation bits: 16 red, 16 green, 16 blue and 0 alpha,
Using 32 Depth bits and 8 StencilBits
0 Auxiliary buffers
00 underlay (0x-fx) and overlay (x0-xf) planes

OpenGl:
VENDOR: {Microsoft Corporation}
RENDERER: {GDI Generic}
VERSION: 1.1.0
DISPLAY INFORMATION: {bits red 8, green 8, blue 8, alpha 0, depth 32, stencil 8, stereo no.}
<W> Emulating Front Buffer
LIMITS: {
Max Texture Size: 1024
Max Elements Indices: 2048
Max Elements Vertices: 256
Available video memory size: 0 MBytes
Maximum working number of elements: 500 K
}

Note: the *Maximum working number of elements* indicates an approximate limit of the mesh size for a comfortable interaction with the model. It depends not only on the memory of the graphics card but on the number of 3D programs, including GiD, that are running on the computer at the moment when the *Help --> About --> More* button was pressed.	This is an example of the text that appears on *Help --> About --> More* in **Safe** Visualization mode (**Linux**):
	
	GiD internal version: 12.0 (64 bits)
	Operating system: 'x86_64' 'unix' 'Linux' '2.6.32-60-generic'
	Tcl8.6.1 Tk8.6(8.6.1)
	OpenGL 2.1 Mesa 8.0.5 on Mesa X11
	GLEW 1.9.0
	
	Togl 1.7: GLX visual id 0x21, attempt 1.
	There is no overlay GLX context.
	
	OpenGl:
	VENDOR: {Brian Paul}
	RENDERER: {Mesa X11}
	VERSION: {2.1 Mesa 8.0.5}
	GLSL version: 1.20
	DISPLAY INFORMATION: {bits red 8, green 8, blue 8, alpha 8, depth 24, stencil 8, stereo no.}
	LIMITS: {
	Max Texture Size: 16384
	Max Elements Indices: 3000
	Max Elements Vertices: 3000
	Max 3D Texture Size: 16384
	Max Cube Map Texture Size: 16384
	Available video memory size: 0 MBytes
	Maximum working number of elements: 500 K
	}

Using the graphics card: Drawing methods

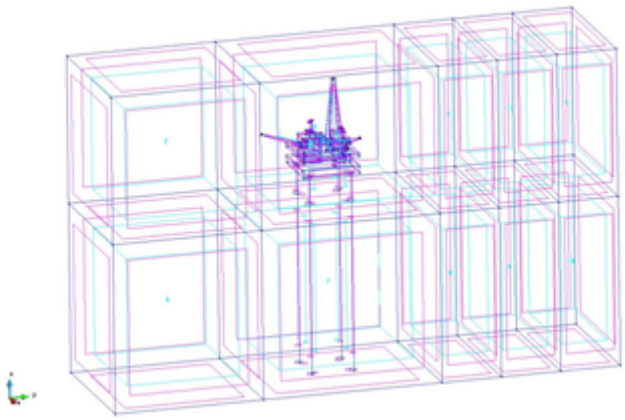
Drawing method: (can be adjusted at Utili*ties --> Preferences --> Graphical --> System*)
Several algorithms are used in GiD to draw meshes faster, but which rely on the underlying hardware at different levels. Faster methods needs good hardware graphics and good driver support, slower methods are safer and indepented of the graphics capabilities available.

- *Vertex buffer objects (VBO):* is the fastest method, but it requires OpenGL 1.5 or higher, and the *Fast visualization mode.* Extra memory of the graphics card is used.
- *Vertex array (VA):* requires OpenGL 1.1, and can work also with *Safe visualization mode* too. Extra main memory is used.
- *Display lists (DL):* requires OpenGL 1.0, basic acceleration if used with the *Fast visualization mode.* Extra memory of the graphics card is used.
- *Immediate mode (IM):* requires OpenGL 1.0, is the slowest but most robust method and requires less memory than the other methods.

Summary of features available in *Safe* and *Fast* visualization modes:

	Safe visualization mode (MS Windows)	**Fast visualization mode**
Drawing: immediate mode	saves memory / slow	saves memory / slow
Drawing: display list (DL)	a bit faster (uses main memory)	faster (uses card's memory)
Drawing: vertex array	a bit faster (uses less memory than DL)	a bit faster (uses main memory)
Drawing: vertex buffer object	No (requires OpenGL 1.5)	fastest (uses less card's memory than DL)
Effect: stereoscopy (3D)	Yes	Yes
Effect: mirror plane	Yes	Yes
Effect: render reflection	No (requires OpenGL 1.3)	Yes
Effect: shadows	No (requires OpenGL 1.5)	Yes
Effect: shiny contour fills in *Preferences --> Postprocess --> Contour fill and lines* option *Add brightness*	No (requires OpenGL 1.2)	Yes
Recommended mesh size	< 500,000 triangles	depends on the graphic's card memory available (*check Help --> About --> More*)

Using the graphics card: Example

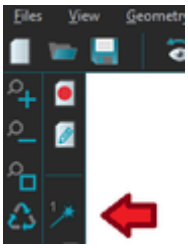


Geometry of the platform_small *project.*

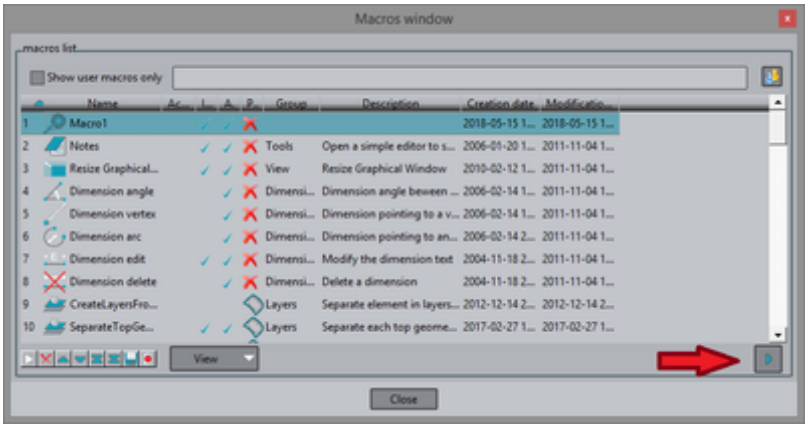
This example is a simulation of waves against a oil platform. The model *platform_small.g*id can be found at Material location.

Macro creation:

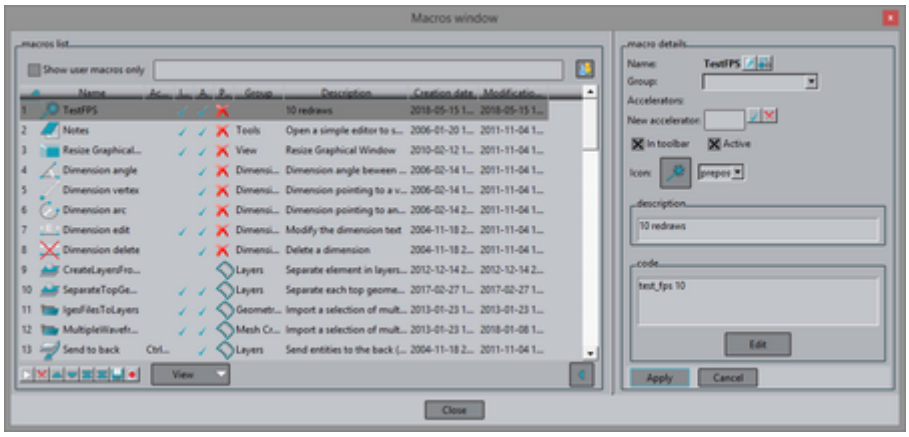
1. create an empty macro, by clicking on the Record macro icon and the Stop record macro icon , both on the macros toolbar. The new macro will appear on the macro's icon bar:



2. edit the macro with the icon on the same macros toolbar, and enter test_fps 10 in the code box:



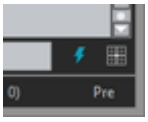
3. Edit macros window, click on the button pointed by the red arrow to expand the window and enter the code



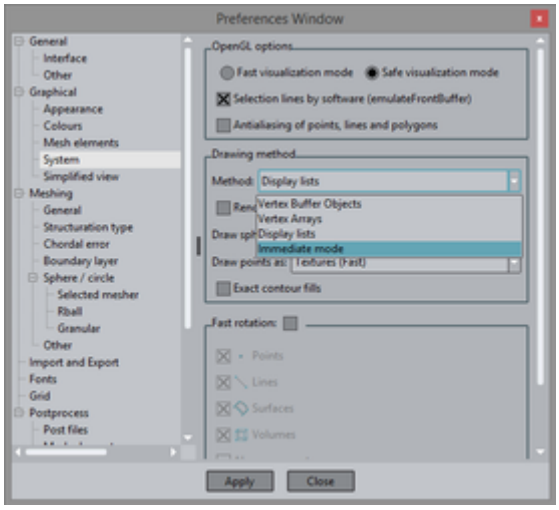
4. load the model and test the macro, a *Warning* window should appear with a text like this: *Redrawing 10 times ... done: 65.3 fps.* ;
5. don't close this Warning window, it will be used to compare the results.

Safe Visualization mode in postprocess

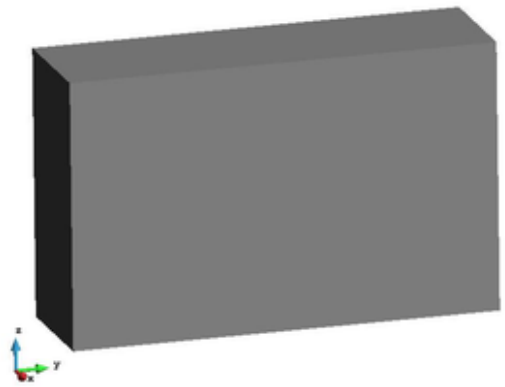
1. click on the lower right card icon to change to **Safe visualization mode;**



2. load the model again and test the macro;
3. go to **postprocess**;
4. go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Immediate;**



5. click on the newly created marco (don't close the appeared Warning window);
6. go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Display lists**;
7. click on the newly created marco (don't close the appeared Warning window);
8. go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Vertex arrays**;
9. click on the newly created marco (don't close the appeared Warning window);
10. (in MS Windows with **Safe Visualization mode** we can't use the **Vertex buffer objects** drawing method).
11. in the Warning window, four lines should appear (one for every test) with a contents like this:



Picture of the mesh in postprocess, with no result

Redrawing 10 times ... done: 103 fps. <-- macro on geometry
Redrawing 10 times ... done: 5.82 fps. <-- macro on immediate
Redrawing 10 times ... done: 8.95 fps. <-- macro on display lists
Redrawing 10 times ... done: 9.98 fps. <-- macro on vertex arrays

(These values are indicative and depend on the hardware used)

Fast Visualization mode in postprocess

12. click on the lower right card icon to change to **Fast visualization mode** (click no when GiD asks to save the model);

13. load the model again and click on the newly created macro (the geometry of the model should be viewed);
14. go to **postprocess**;
15. go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Immediate;**
16. click on the newly created marco (don't close the appeared Warning window);
17. go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Display lists**;
18. click on the newly created marco (don't close the appeared Warning window);
19. go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Vertex arrays**;
20. click on the newly created marco (don't close the appeared Warning window);
21. go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Vertex buffer objects**;
22. click on the newly created marco (don't close the appeared Warning window);
23. in the Warning window, three lines should appear (one for every test) with a contents like this:

    Redrawing 10 times ... done: 174 fps.
    Redrawing 10 times ... done: 11.1 fps.
    Redrawing 10 times ... done: 41.1 fps.
    Redrawing 10 times ... done: 40.4 fps.
    Redrawing 10 times ... done: 45.3 fps.
    (These values are indicative and depend on the hardware used)

24. compare the times obtained:

	Safe visualization mode	Fast visualization mode
geometry	103 fps	174 fps
immediate mode	5.82 fps	11.1 fps
display lists	8.95 fps	41.1 fps
vertex arrays	9.98 fps	40.4 fps
vertex buffer objects	N/A	45.3 fps

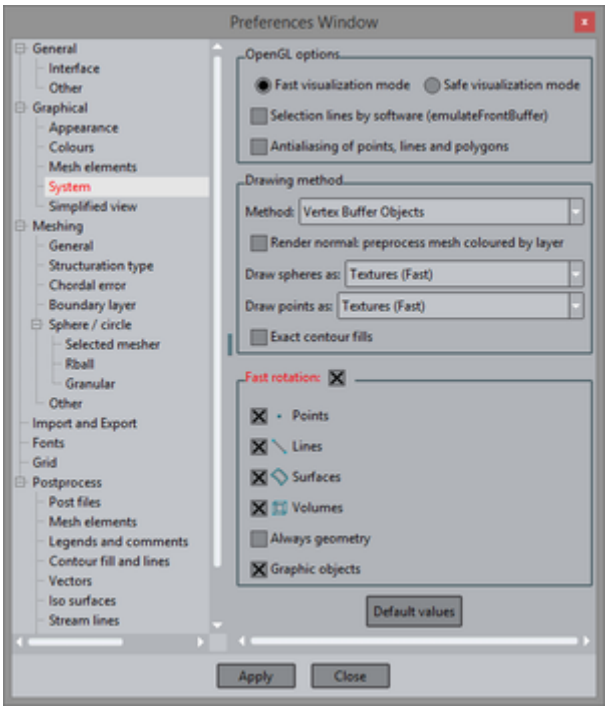
(These values are indicative and depend on the hardware used)

Note: in preprocess mode the display lists drawing method is not available.

Fast rotation

Fast rotation: How it works

To enable this option just open the *Preferences* window and tgo to *Graphical --> System*:



When heavy models are loaded in GiD, it may last several seconds to draw a view of the model, for instance when drawing several millions of lines of a huge mesh. In these cases, it's very painful to interact with the model, i.e. rotate, zoom in-out, move it.
GiD offers a **Fast rotation** mode, which allows fast interaction with the model.
With this option enabled, when the user rotates the model, zoom-in or out dynamically, then GiD will change the view of the model to a simplified one, for instance instead of rotation several millions of mesh edges, only the geometry is used to rotate the model; then, when the user finished its interaction, for instance pressing the Escape button or the middle mouse button, the GiD switches back to the previous view.

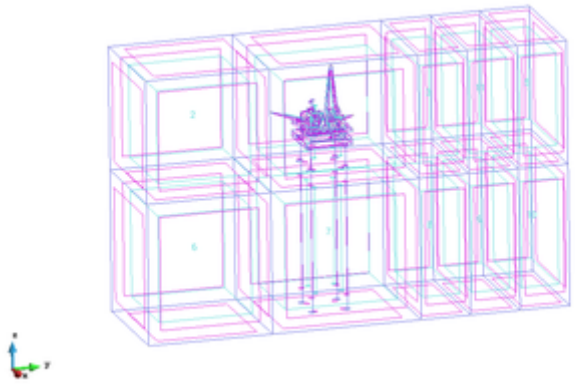
Several options can be configured in Fast rotation mode: the behaviour of these depends if GiD is in preprocess or postprocess mode:
Preprocess:

- *Points, Lines, Surfaces* and *Volumes:* to draw or not to draw each type of entity when rotating.
- *Always Geometry:* With this option set, when you view and rotate the mesh, the geometry is drawn instead.
- Draw *graphic objects:* If this option is not set, when rotating the geometry, some graphical and temporal objects like normals or materials or conditions symbols are not drawn.

Postprocess:

- *Lines:* if it's set, when rotating the model, the boundaries lines of the mesh is drawn;
- *Surfaces:* if it's set, when rotating the model, a simplified view of the mesh is drawn, depending on the options set in *Preferences --> Graphical --> Simplified view* (please, look into the *Simplified view section*).

Fast rotation: Example

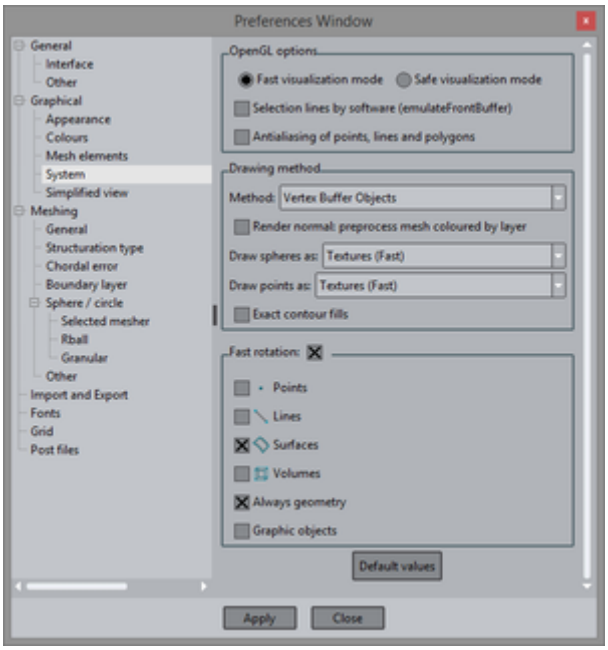


Geometry of the platform_small *project.*

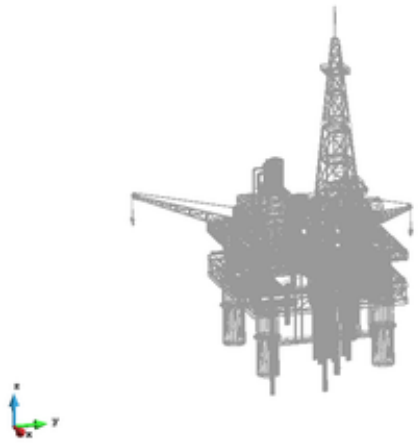
This example is a simulation of waves against a oil platform. The model *platform_small.g*id can be found at Material location.

Preprocess:

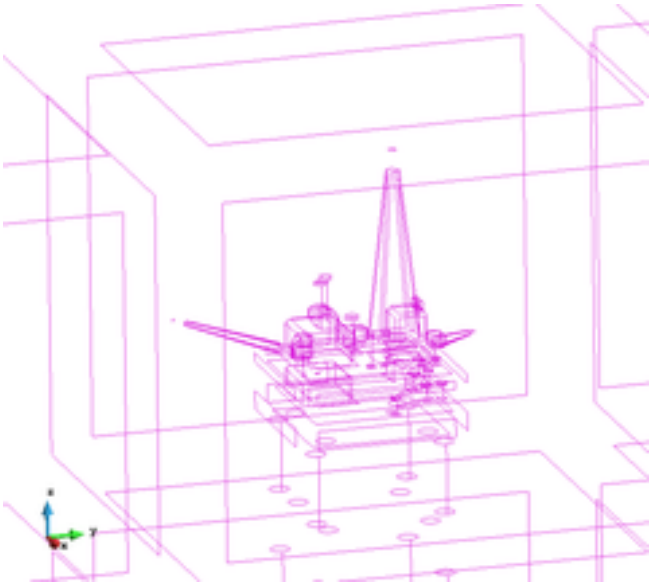
1. load the model, and change to mesh view mode, using the icon  ;
2. open the preferences window and enable the *Preferences --> Graphical --> System --> Fast rotation mode*, enabling only the surfaces and with the mode *Always geometry* set;



3. rotate the model, use the scroll wheel of the mouse to zoom in and out and check the switching between the mesh and geometry views, and back, of the model:



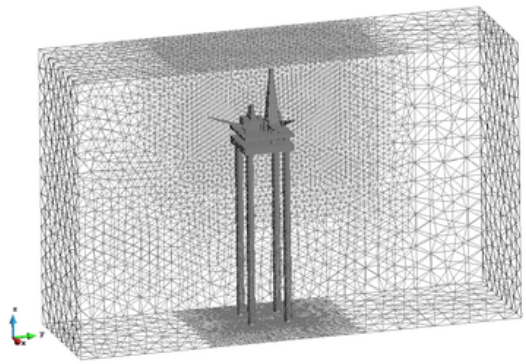
Mesh view of the platform model



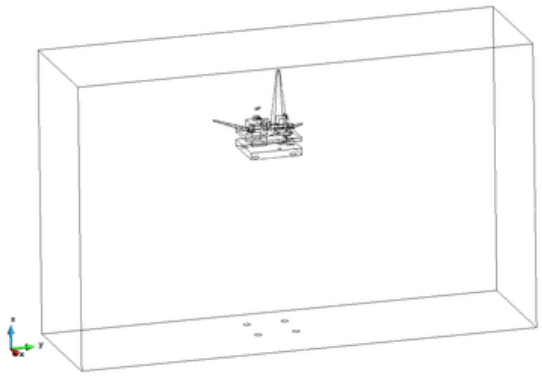
'Fast rotation' view, showing only the surfaces of the model while rotating it

Postprocess:

4. go to postprocess;
5. enable only the Lines checkbox in the *Preferences --> Graphical --> System --> Fast rotation mode* section;
6. rotate the model, use the scroll wheel of the mouse to zoom in and out and check the switching between the mesh and geometry views, and back, of the model:



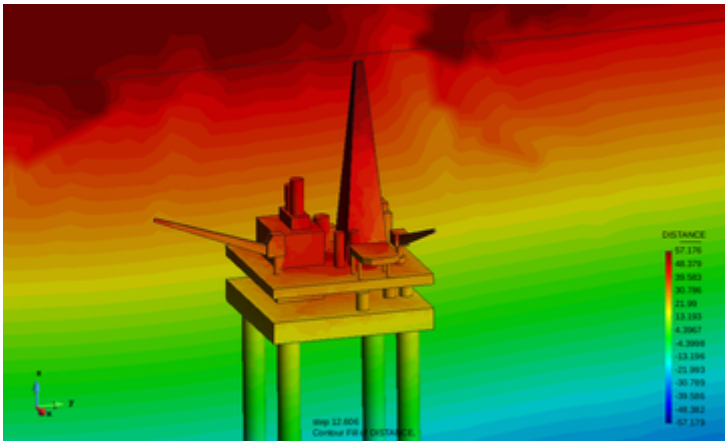
'All lines' style in postprocess mode



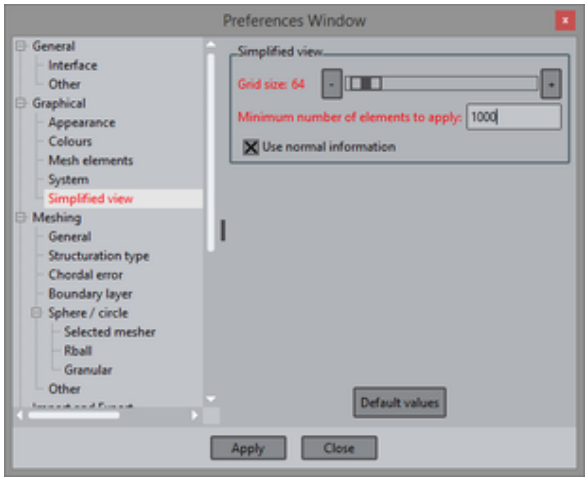
Boundaries style used in 'Fast rotation' when only the 'Lines' are enabled

7. do a *contour fill of DISTANCE* (may need to change *Display Style* to *Body* or *Body Boundaries)*;
8. do a *Culling* of the *front faces* to view the interior of the volume:

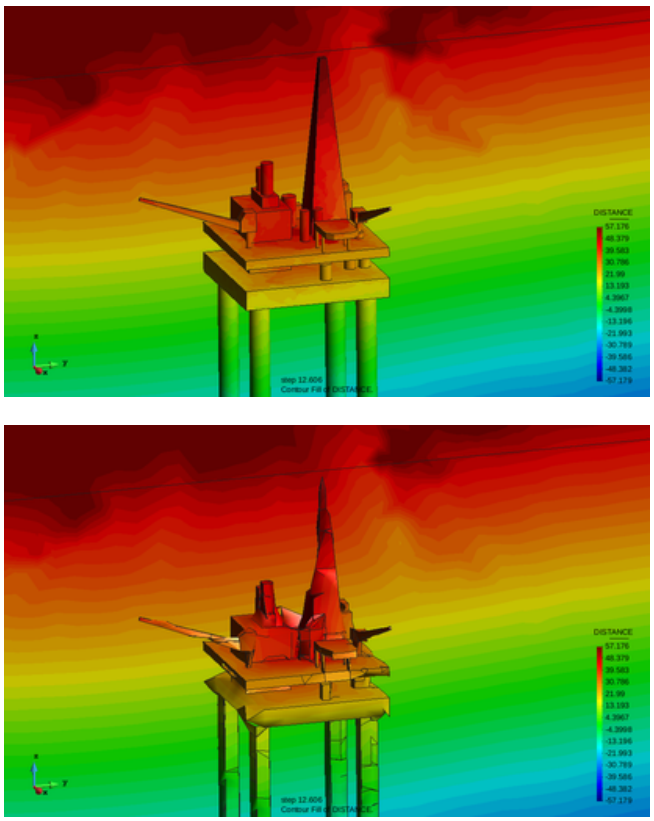
9. and zoom in to get this view:



10. switch on the 'Surfaces' checkbox in the *Preferences --> Graphical --> System --> Fast rotation mode* section;
11. in the *Preferences --> Graphical --> Simplified view* section, change the *Grid size* to *64* and the *Minimum number of elements to apply* to *1000;*
12. rotate the model, use the scroll wheel of the mouse to zoom in and out and check the switching between the mesh and geometry views, and back, of the model:



Note: When a result is displayed, the simplified view interpolates the original result into the simplified mesh.



DISTANCE contour fill on the original model above, and on the simplified view below.

13. Now disable the *fast rotation mode* in the *preferences* window.

Simplified view

Simplified view: How it works

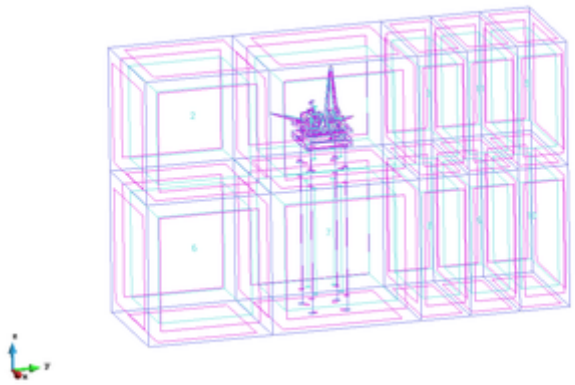
For huge meshes, which are very slow to draw, a simplified view can be used to speed up the interaction between the user and GiD. The operation will be performed on the original model or mesh, and only the visualization is simplified. The simplified mesh is calculated using a vertex clustering based algorithm (http://upcommons.upc.edu/pfc/handle/2099.1/20380) with only geometrical information. Then, each time a result is selected, it's interpolated from the original results on the simplified mesh. Simplified meshes are used if Fast rotation mode is enabled (see Fast rotation), or at user demand using the following icons:

Toolbar:  GiD is drawing the original mesh, click to change to simplified view.

Toolbar:  GiD is drawing using the simplified mesh, click to change to original mesh.

- **Grid size:** Size of the vertex clustering grid used to simplify the mesh: a smaller grid size generates a coarser mesh, a bigger grid size generates a finer mesh.
- **Minumum number of elements to apply:** minimum size of the mesh which will trigger the mesh simplification process. Bigger meshes will be simplified, smaller ones not.

Simplified view: Example

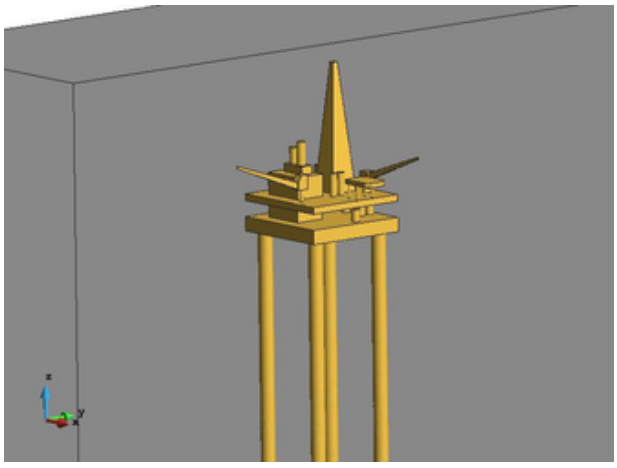


Geometry of the platform_small *model.*

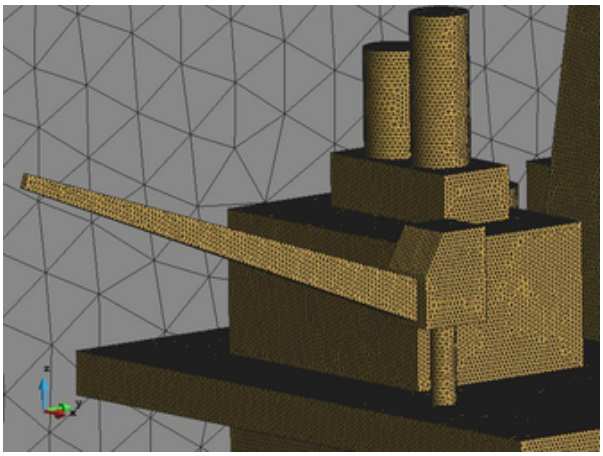
This example is a simulation of waves against a oil platform. The model *platform_small.gid* can be found at Material location.

We'll work a little bit on the model first: from the original volume tetrahedra mesh we'll create some surface meshes to represent the platform and the surrounding box.

1. load the model;
2. go to *postprocess;*
3. select the menu *Geometry --> Extract boundaries;*
4. select *All lines* display style;
5. create a new set by selecting elements of the platform (using the *Send to --> new set* option of the *Display style* window), and change its colour;

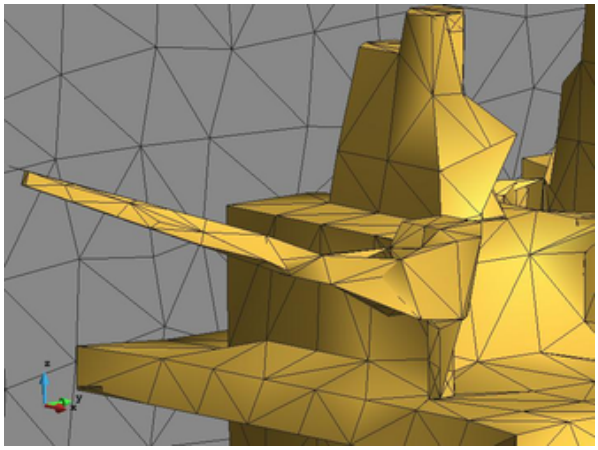


The single triangle mesh of the volume boundary has been separated in the grey box triangle set and the golden platform.

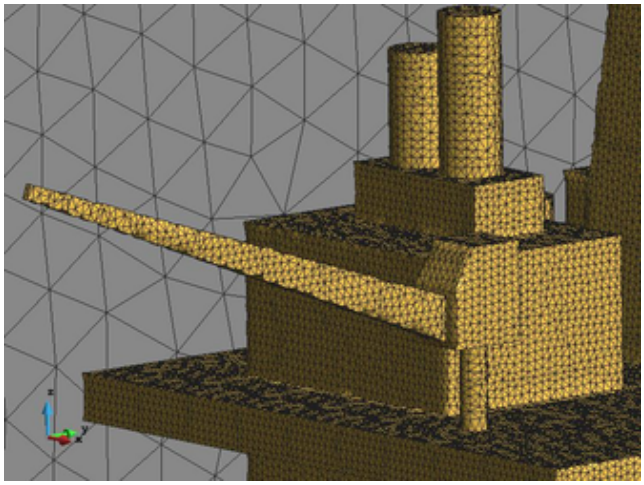


Zoom view showing the refined triangle mesh used to model the platform.

6. in the *Preferences --> Graphical --> Simplified view* section, change the *Grid size* to *64;*
7. enable the *Fast draw* mode by clicking on the  icon,
8. the icon will change to  , click on it to change back to the original full-detailed mesh view;
9. in the *Preferences --> Graphical --> Simplified view* section, change the *Grid size* to *640;*
10. enable the *Fast draw* mode by clicking on the  icon,
11. the icon will change to  , click on it to change back to the original full-detailed mesh view;
12. Note the difference in the details:



The coarse simplification uses less triangles to draw the same geometry, the 380,000 triangles are reduced to 4,500



A smaller grid size allows a better representation of the details of the model, but it uses more triangles, the 380,000 triangles are reduced to 140,000

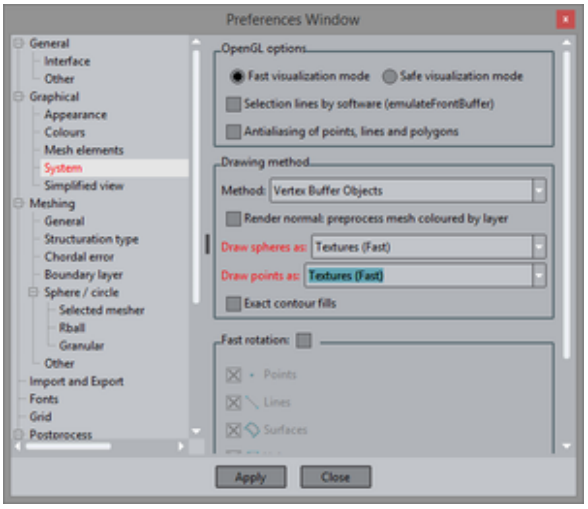
Visualizing particle models (DEM, PFEM, ...): Textures in spheres, circles and points

Textures in spheres: How it works

There are several ways to draw spheres, circles and points in GiD:

- *Mesh (slow),* using a triangle mesh to represent the sphere or point, which has between 4 and 630 triangles, depending on the detail level;
- *Textures (fast):* using images of already rendered spheres to speed up the drawing process, it's a compromise between quality and speed;
- *Square (Fastest):* the fastest way to draw spheres, use this method when speed is the most important factor.

In **Preprocess** these options are accesible through *Preferences --> Graphical --> System* inside the *Drawing method* box.



Preprocess drawing options for Points and spheres

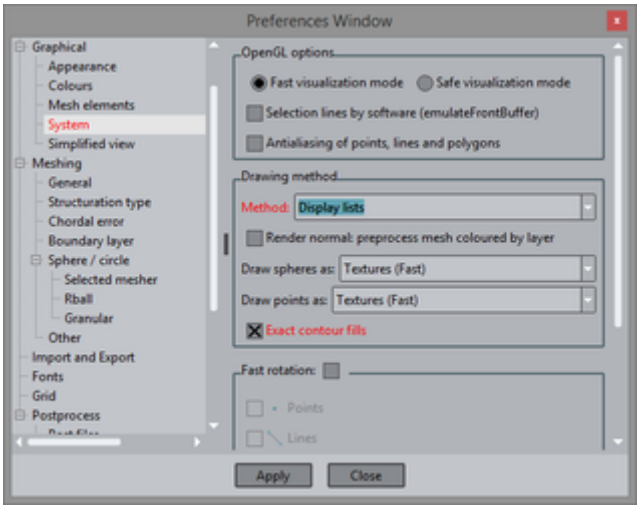
Postprocess drawing options for Points and spheres

In **Postprocess** there are other options under *Preferences --> Postprocess --> Mesh elements:*

- **Quick:** points, spheres and circles will be drawn as big dots.
- **Nice:** a triangle mesh, between 4 and 630 triangles big, is used to drawn each sphere and circles for a nicer quality.
- **Images:** selveral images can be selected and will be used to draw on the point elements, using the point coordinates as the centre of the images. For a certain point, at each redraw the next image of the collection will be used.
- **Nice detail level:** level of detail used to draw the points, spheres and circles in Nice mode. Higher detail level will draw more triangles for each sphere or circle.

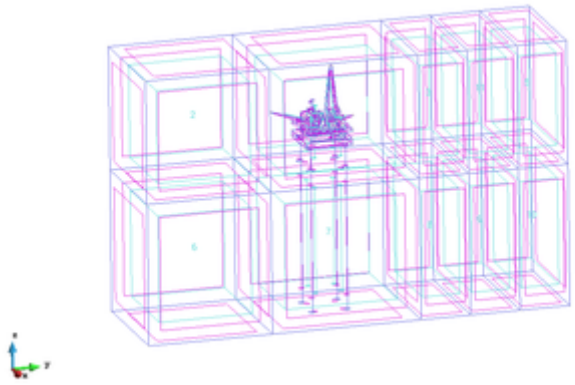
Textures in Contour Fill:

By default, a texture is also used to draw the contour fill of a result. In this case, the texture is just a colour scale, like the one used to draw the legend. This option is used by default to speed-up the render of the results.
There can be interpolation problems when there are some undefined result values for some nodes.
In *Preferences --> Graphical --> System* the *Exact contour fills* options can be used to get a more accurate representation of the colour bands in the contour fill result visualization, especifically in the extremes of the result scale:



- *Exact contour fill*s (only with *Display list* and *Immediate mode* drawing methods, only in postprocess): if set, each element is subdivided into single colour sub-elements according to the colour scale, in order to get a better quality and limit representation of the colour bands. This option is only available with Display lists or Immediate mode drawing methods.

Textures in spheres: Example



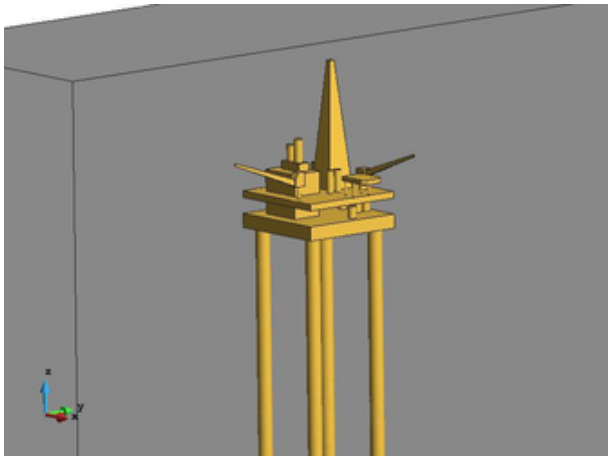
Geometry of the *platform_small* model.

This example is a simulation of waves against a oil platform. The model *platform_small.gid* can be found at Material location.

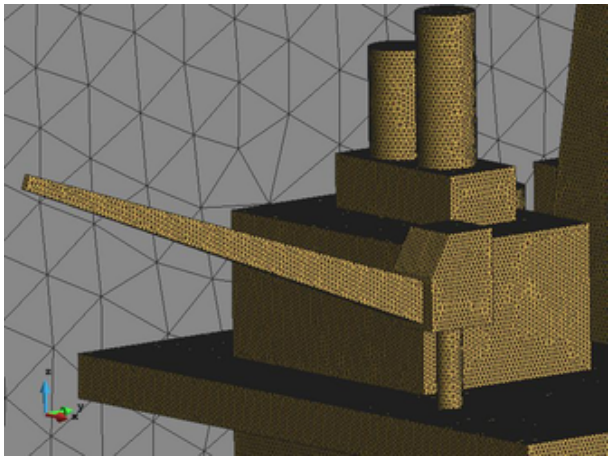
We will use the **TestFPS** macro (*test_fps 10)* created in the '**Using the graphics card: Example**' section.

As before (if not already done) we'll work a little bit on the model first: from the original volume tetrahedra mesh we'll create some surface meshes to represent the platform and the surrounding box.

1. load the model;
2. go to postprocess;
3. select *Geometry --> Extract boundaries;*
4. select *All lines* display style;
5. create a new set by selecting elements of the platform (using the *Send to --> new set* option of the *Display style* window), and change its colour;

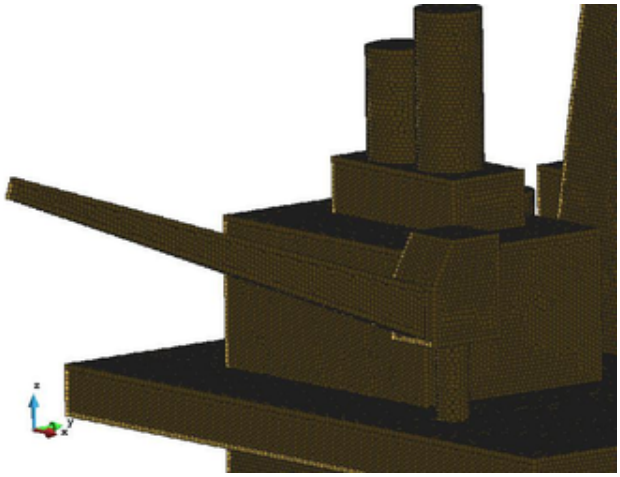


The single triangle mesh of the volume boundary has been separated in the grey box triangle set and the golden platform.



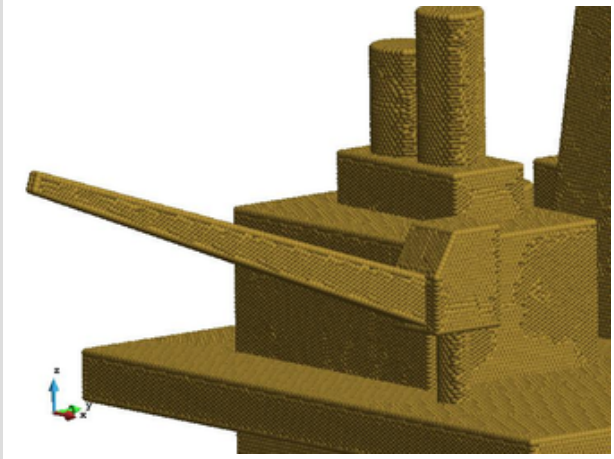
Zoom view showing the refined triangle

6. switch off all sets except the platform one;
7. test the macro, a *Warning* window should appear with a text like this: *Redrawing 10 times ... done: 20 fps.* ;
8. change the visualization style to *points;*
9. change in *Preferences --> Postprocess --> Mesh elements* , change the *Point elements options*,to *Quick*; and *Size* to 5.5;
10. test the macro;

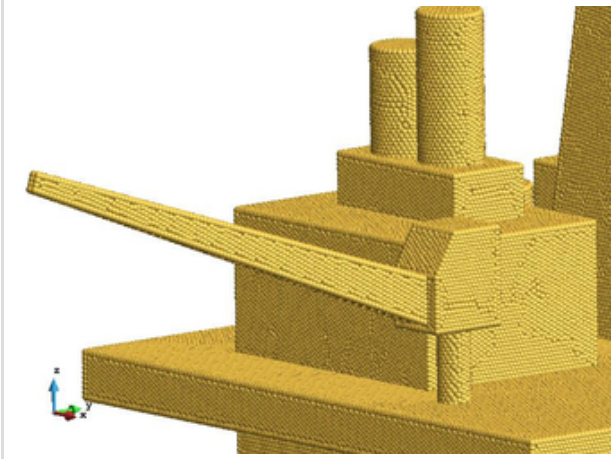


Drawing Quick points on the nodes of the platform mesh

11. change in *Preferences --> Postprocess --> Mesh elements* , change *Point elements options*, to *Images* and execute the macro;
12. change in *Preferences --> Postprocess --> Mesh elements* , change *Point elements options*, to *Nice* and set *Nice detail* to 5, and execute the macro;



Drawing points with textures on the nodes of the platform mesh



Drawing Nice points on the nodes of the platform mesh

13. Compare the times obtained:

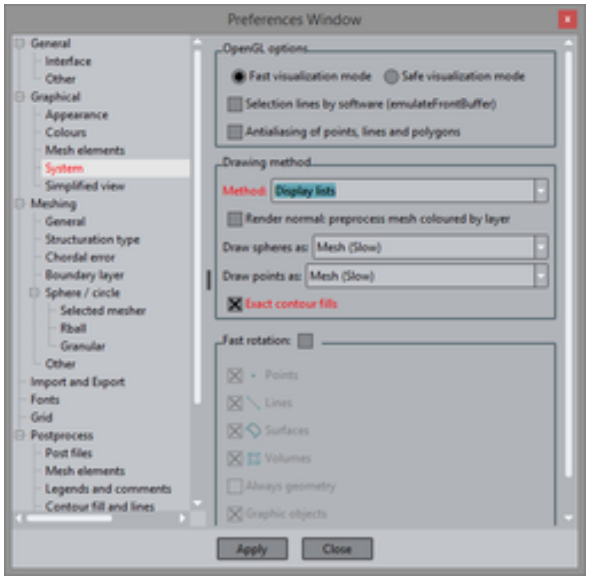
Quick points: Redrawing 10 times ... done: 35.8 fps.

Images: Redrawing 10 times ... done: 18 fps.
Nice (detail = 5): Redrawing 10 times ... done: 2.84 fps.

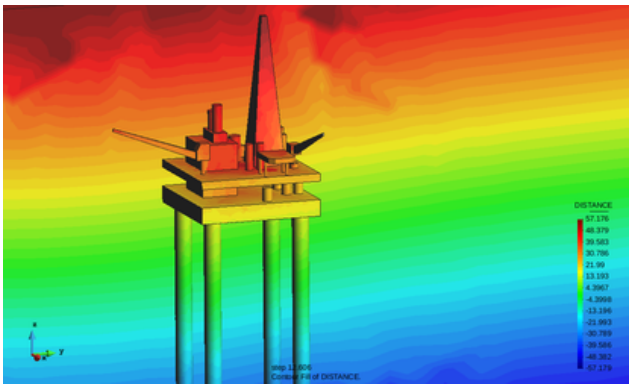
(These values are indicative and depend on the hardware used)

Textures in contour fill: Example

1. set display style to *Body Boundaries*;
2. Switch on the surface sets;
3. set culling to front faces;
4. in *Preferences --> Graphical --> System* change the *drawing method* to *Display List;*
5. test the macro, a *Warning* window should appear with a text like this: *Redrawing 10 times ... done: 38 fps.* ;
6. in *Preferences --> Graphical --> System* enable the *Exact contour fills;*



Graphical --> System preferences used in this contour fill example



View of the model used for the contour fill test, with culling of the front faces and the two surface meshes on

7. *do a Contour Fill of DISTANCE,* and execute the macro;
8. *disable the Exact contour fills* option in *Preferences --> Graphical --> System;*
9. *do a Contour Fill of |Velocity|, and execute the macro;*
10. Compare the times obtained:

Exact contour fills on: Redrawing 10 times ... done: 6.5 fps. (Exact contour fills enabled)
Exact contour fills off: Redrawing 10 times ... done: 29 fps. (Exact contour fills disabled)

(These values are indicative and depend on the hardware used)

Working with terrain models

A terrain model is a particular kind of geometry that usually is considered as z(x,y), that is, each projected point (x,y) has single value of z. Note that this kind of model is not able to represent vertical planes, nor overlapped entities in z direction.

Common ways to describe a topography are:

a) A collection of level curves (iso-lines with same value of z)

b) A cloud of irregular x,y,z points

c) A collection of triangles or TIN (triangulated irregular network)

d) The height z(x,y) on a regular grid of quadrilaterals

e) Other, like standard CAD files, using generic surfaces

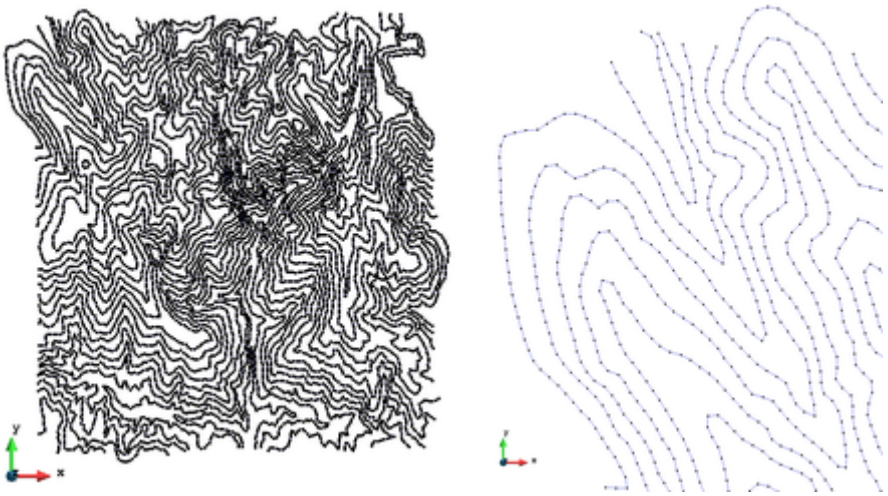
In this course, we will explore how to deal with this kind of models following some of these approaches.

Level curves

Ensure the default values of 'Automatic collapse' and 'automatic tolerance values' are set in the 'Import and Export' branch of preferences window, and import the file named '*level_curves.dxf*' using

Files->Import->DXF...

The result is a model with a collection of straight lines like this:



After importing the file, we recommend to save the model, so as we can access always to the GiD model with the file already imported.

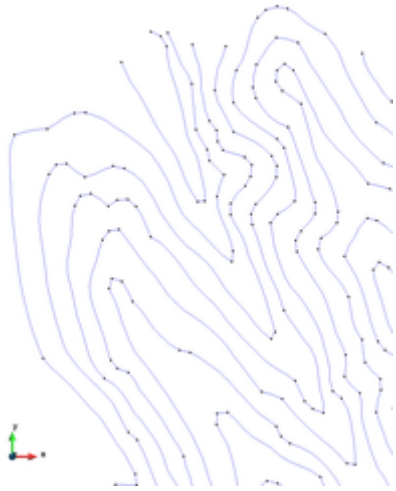
Creating separated surfaces

It is a model of lines, not of surfaces. Some lines could be used to build surfaces, but it should be a hard task.

To facilitate the work the 'near tangent' lines (with a tolerance) could be joined to reduce the number of curves of the model. To do so, use

Geometry->Edit->Join->Lines,

and select all of them. The result should look like the following figure.



To create surfaces it is needed to have a set of closed contour lines. It is possible to create extra curves to enclose regions of topologically four curves, to create surfaces defined 'from its boundary'.

Use

Geometry->Create->Line

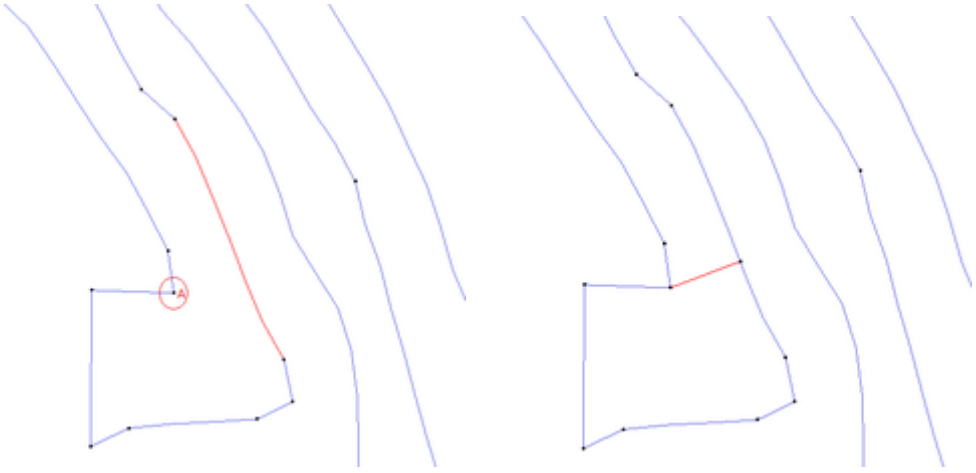
and pick two existing points, for example create these two new lines:

In some parts, there may be needed some extra point to create the desired line, it is possible to force the division of a curve 'close to a coordinate' using

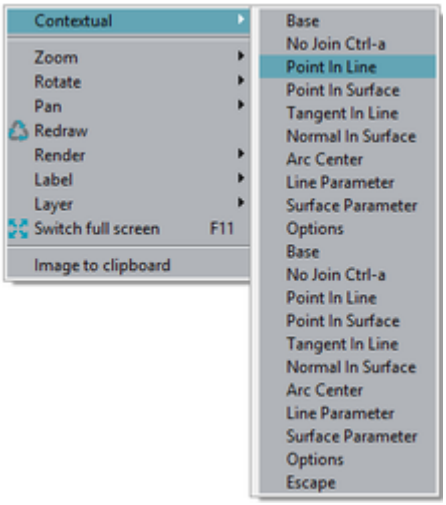
Geometry->Edit->Divide->Lines->near point

Select the existing point A, press <ESC> and then the line/s to be divided

Then, it is possible to use the new created point to create a new line:



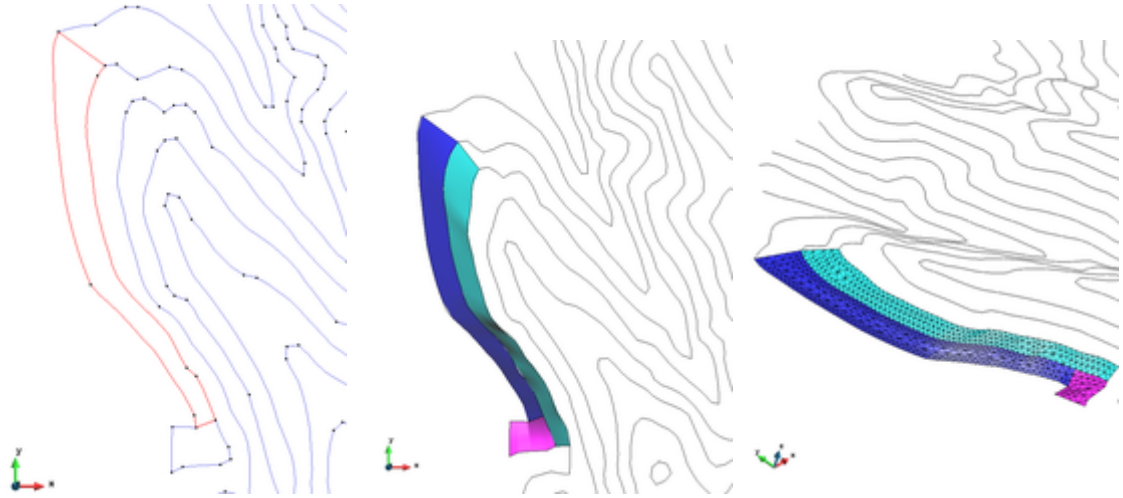
Note: when selecting the 'near point' it is possible to click the right-mouse button to open the contextual menu, and select other options, like 'Point in line' to pick a point over a curve, like the one to be divided.



Now we can create a new surface interpolating its boundary curves:

Geometry->Create->NURBS surface->By contour

And select the boundary curves.

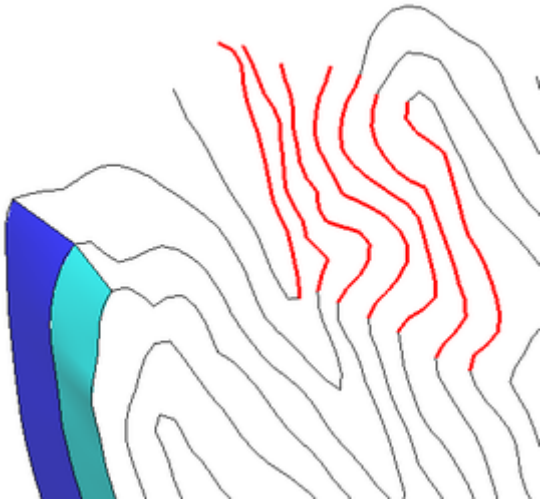


With this procedure, we can create the three surfaces of the image. The surfaces will have only C0 continuity between them (normals between them are discontinuous), but the shape could be good enough to generate a surface mesh valid for some simulations (see the mesh of the image above).

Creating surface from a family of parallel curves

Another way of create surfaces based on level curves is from a collection of 'near parallel curves'. The surface created will interpolate the curves and will have continuity in normals inside it.

For example, we can divide some curve if necessary and send the curves of the image to another layer to facilitate the selection,



We are going to smooth a little bit the shape of the curves. Prior curves must be converted to NURBS expression.

Geometry->Edit->Convert to NURBS->Line

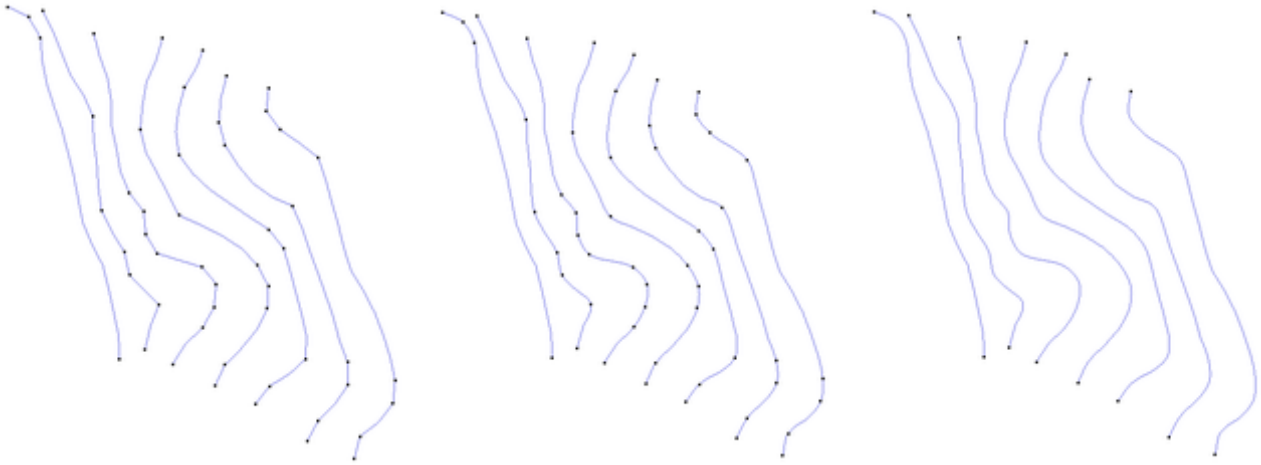
And then modify a little its shape forcing the tangency at joining points.

Geometry->Edit->Lines operations->Force to be tangent

And set a very big tolerance angle (between tangents), like 90 degrees

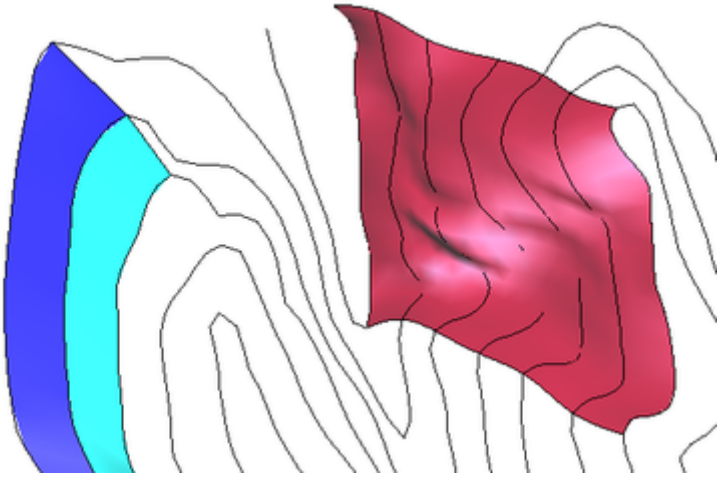
Then lines can be joined using

Geometry->Edit->JoinLines



At this point we already have the family of 'parallel' smooth curves to be used for the creation of the interpolating surface:

Geometry->Create->NURBS surface->Parallel lines



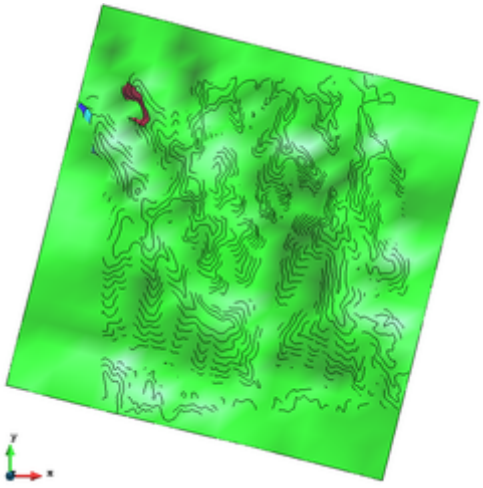
As can be seen this may be a manual and tedious procedure to re-build a surfaces model depending on the complexity of it, but in some cases this procedure may be very useful to simplify large models.

Cloud of irregular points

An interesting option is to create a single NURBS surface approximating (not interpolating) a cloud of points. It creates a rectangular shape projecting in a direction (in this case, z direction). To do so, use

Geometry->Create->NURBS surface->By points

and select all the points of the model



Note: surface sides are not parallel to XY axis, because other automatic 'main directions' are calculated

We can then use only part of the whole surface, trimming it. For instance, if we want only the rectangular region between the corners (156,1352) and (1178,128)

We can create a rectangle

Geometry->Create->Object->Rectangle

156,1352,0
1178,128,0

Then extrude it to create lateral surface that intersect with the terrain

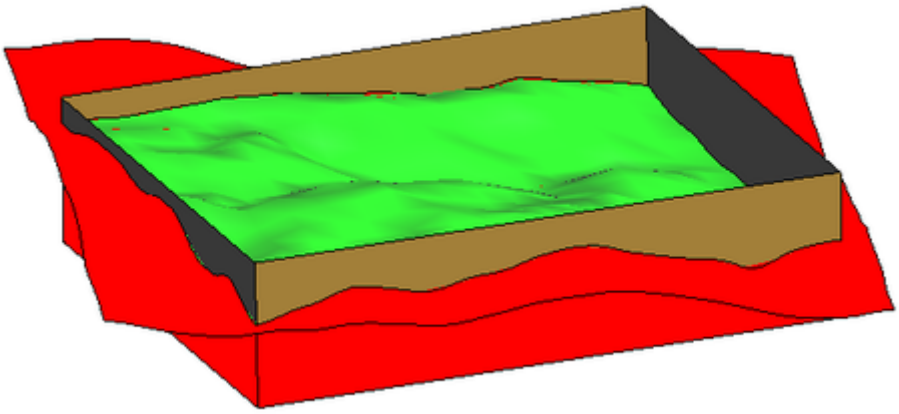
Utilities->Copy... (Lines, Translation, increment z=300, Do extrude: Surfaces)

Now, for intersecting the surfaces use

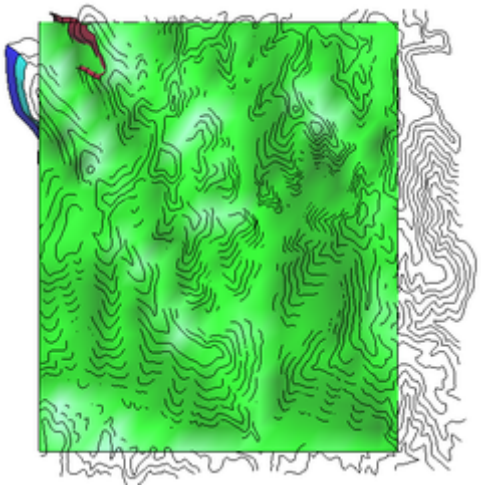
Geometry->Edit->Intersection->Surfaces

And select the surfaces

Then you can delete unwanted parts



This allow for example to create a 3D volume box for CFD simulations of the air around the topography.



TIN from points

A simple way to sample points is creating a mesh, and exporting the coordinates of the nodes as a simple text file. Create a new GiD project using

Files->New

and import another time the DXF 'level_curves.dxf' file.

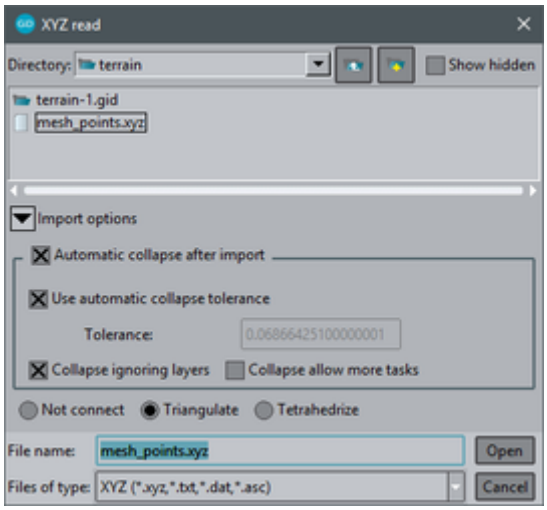
Generate a mesh (Mesh->Generate), and write the coordinates of its nodes using

Files->Export->Using template .bas (only mesh)->XYZ

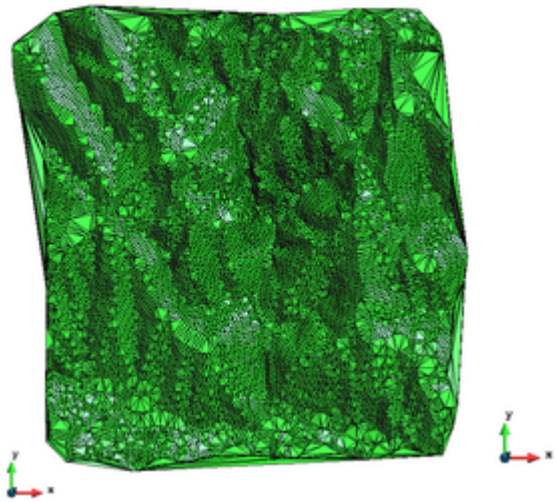
Create again a new model (*Files->New*) and import the XYZ file as geometric points

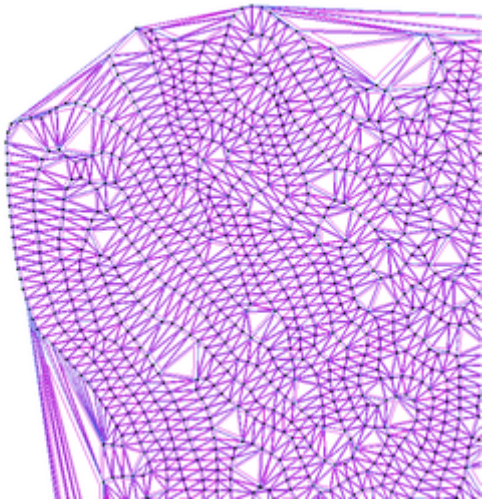
Files->Import->XYZ points...

Ensure the 'Triangulate' extra option is selected in the import options parameters, as shown in the following figure.



This will create a TIN 'Delaunay' triangulation, connecting in Z projection the provided points (note that GiD collapses automatically the nodes in the same position, so as there are no overlapping in the resulting mesh).





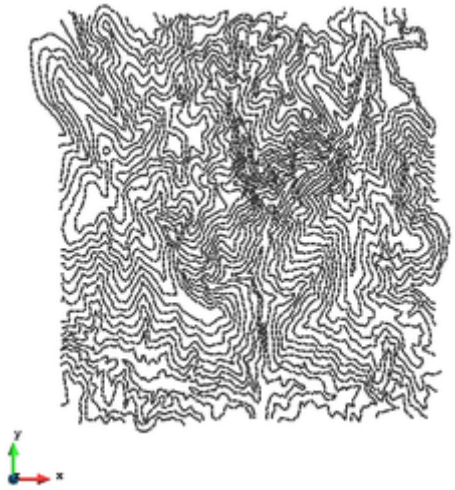
It is topologically a valid surfaces model, that interpolate all provided points, but it has a lot of triangles, some of them with small angles or short edges and this difficult other geometrical operations, and re-meshing.

Note: It is possible to use collapse of points/nodes/edges to have less triangles

Regular grid from nodes

Create again a new model (*Files->New*) and import the XYZ file but now as mesh nodes, and without connect them with triangles

Files->Import->XYZ nodes...

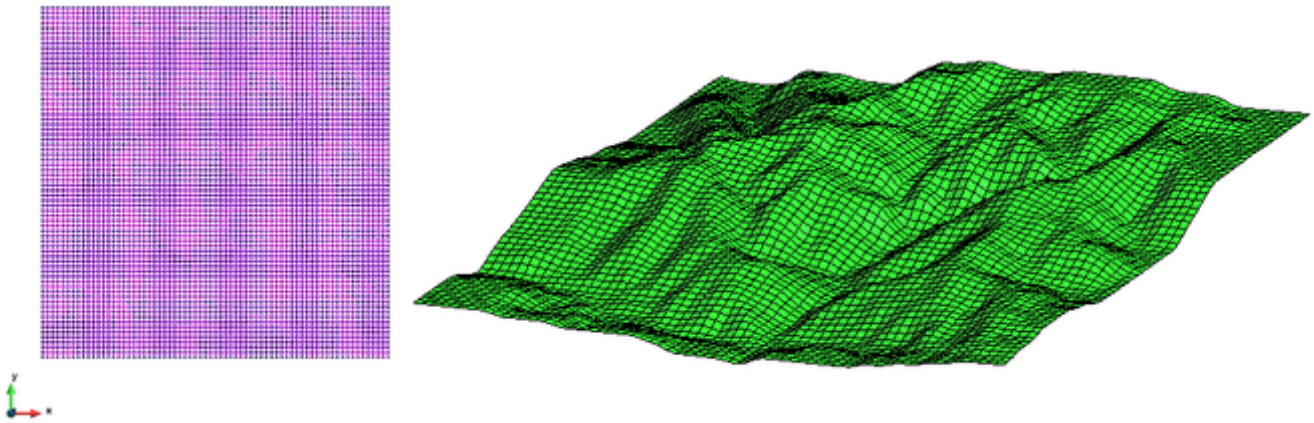


We can build a regular square grid using

Geometry->Create->Geometry from mesh->Nodes->grid surfaces

Enter a grid cell size of 20 units and select all nodes.

A regular grid of surfaces will be created, with z of points approximated from the selected nodes



Using *Nodes->grid surfaces* the z's are approximated with an averaged value based on inverse of distance to a power of close nodes.

Using *Triangles->grid surfaces* (in case the input data are triangles, not just nodes) the z's would interpolated projecting the node over the selected triangles. This option could be more expensive.

Note: A regular grid is easier to manipulate than a TIN.

Instead of create surfaces of geometry it is possible to create mesh entities, or export the grid data in a file (ARC/Info grid ASCII format, supported by most 'Geographic Information System' GIS programs). If you export a raster file using

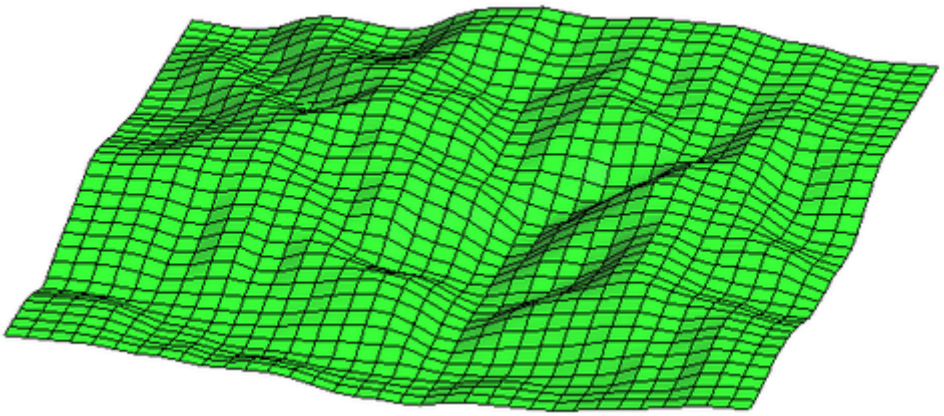
Files->Export->Nodes->Raster...

then, the raster can be re-imported with

Files->Import->Raster GDAL...

This import option supports several common raster formats like geotiff, ARC/Info ASCII or binary, png, jpg, among others (based on the GDAL library), and allows to create geometry, mesh, write as ARC/Info ASCII, and also sub-sample data for big files.

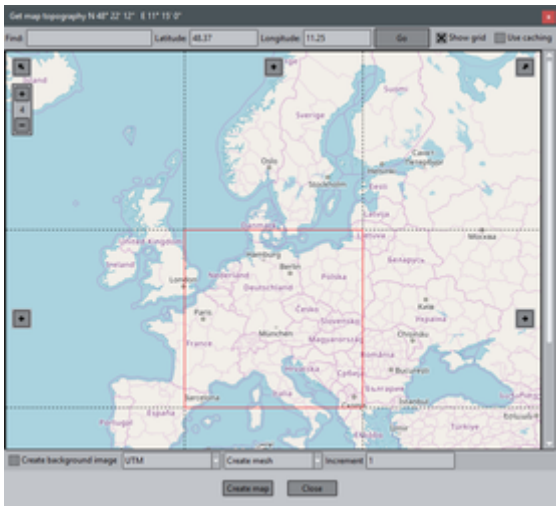
For instance, sub-sampling with increment=2 will take into account one of each 2 point data in each direction, like shown in the figure hereafter.



Grid from Internet

GiD 14 incorporates the possibility to obtain a grid from the place of the world we want using:

Files->ImportTopography...



We must enter the latitude and longitude geographical coordinates of the location we are interested in. In case we do not know its coordinates is possible to try to find them based on the name of the place,

For instance, if we enter the name 'Barcelona' and press enter, the coordinates are filled with values lat=41.3828939 lon=2.1774322 degrees

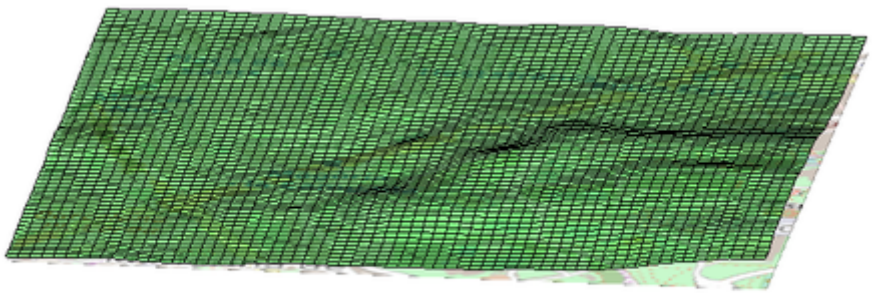
Note: If is possible to find coordinates with external tools like 'Google maps', be careful, the order of lat/lon could be reversed depending on each application

We can set the zoom level until a maximum of 14 (for a grid resolution of about 7 meters)

and press 'Create map' to create geometry/mesh/export to ARC/Info (only the central tile data, squared in red, will be imported)

If '*Create background image*' is checked then the image (from 'Open Street maps') is set at geo-referenced background image

Example using only 1/4 of data of topography of Montjuich Mountain.



Note: geo-referenced UTM topography usually require too big numbers for the coordinates, especially for the 'y' (e.g. x=427378 y=4579121), this could become a numerical problem for the graphical representation (OpenGL uses float numbers to draw, that mean about 7 significant digits), and for the calibrated tolerances of some algorithms. In case of problems a solution could be use the 'Move' tool to displace the whole model close to the 0,0 origin. It is recommendable to store somewhere (like Utilities->Tools->Notes) the offset used to recover the true coordinates if necessary.

Macros

Definition

What we call here as 'macro' is a button packed in the 'macros toolbar' (shown usually on the left side). When the user click the button a function is invoked.

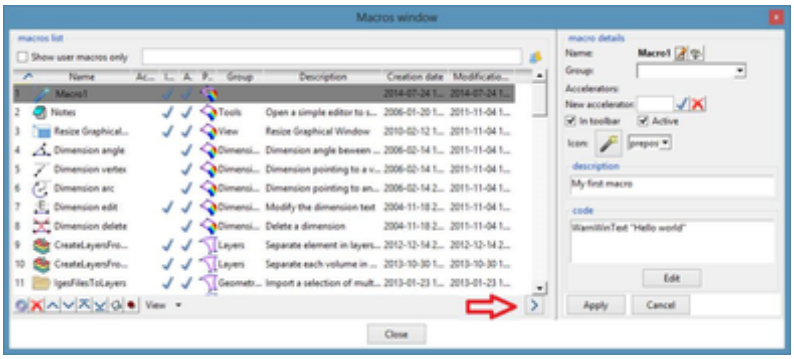
This function in this case is a Tcl scripting procedure (in GiD it is possible to define and invoke Tcl code in much other ways)

By default GiD provide some predefined macros, but the user could modify this toolbar, adding its own macros.

The definition of these macros will be stored in a user folder in a file named 'Macros.tcl'.

Creation of a macro

1. Create an empty macro, by clicking on the Record macro icon and the Stop record macro icon , both on the macros toolbar. Then a new button named 'Macro1' will appear on the toolbar and it will invoke the process commands used while recording the macro
2. Edit the macro with the icon on the same macros toolbar, and enter the Tcl code that do you want to be invoked in the code box:



Edit macros window, click on the button pointed by the red arrow to expand the window and enter the code

In this case we set as code

```
W "Hello world"
```

Remember to press "Apply" after change some macro parameter.
Then to test the macro click the new toolbar button and it will open a new window showing the message "Hello world"

Note: W is a short alias to WarnWinText, that is a procedure defined in GiD to open a window and show a message

Off course we can replace the code to do a more interesting thing.
The macro could be customized a little more:

- Setting an special icon (selecting a predefined icon or getting it from an image file)
- Specifying if the macro must be visible only in pre, only in post of both (default)
- Providing a description that will be shown as contextual help on the button.
- Defining some key accelerator to run it easily.

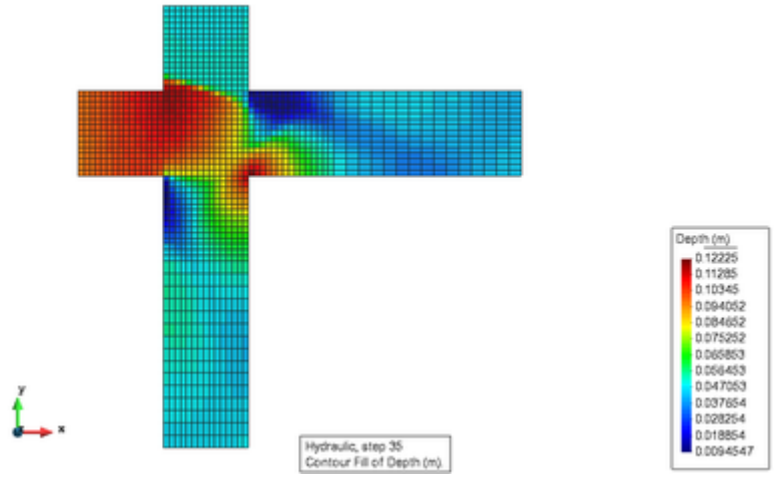
Macro to create a graph of a gauss point result

The aim of this tutorial is to show how to create a user macro to plot a graph of a gauss-point result.

A full copy of the code of this macro is provided in the file 'GraphResult.tcl'

We will use for this tutorial a GiD model named 'cross' that represents a cross street has has been calculated by the Iber solver (http://www.iberaula.es) and has some results 'on gauss points'.

Open this model and switch to post-process to see its results, e.g. a contour fill of the result named 'Depth (m)' on the last time step



In GiD postprocess it is possible to create a graph along the time of a result on a node or on an element, but the result is mapped to nodes and averaged to create a continuous field and then interpolated with shape functions to inner points.
This averaging and interpolation process modifies the 'calculated result' and the user could prefer the original unmodified value (discontinuous between neighbor elements)

This feature is not implemented, but can be achieved with a custom macro.

The things to do are conceptually simple:
1- Allow the user to select an element and specify its gauss point and the result that want to plot.
2- For each time step ask GiD for the value of this result on this location
3- Create a new graph with the list of these values.

1- Select the element, gauss point and result

To simplify the code we could assume that initially the element id and the result name is known at programming time.

In a final step we could enhance it to allow select the element picking in screen.

2- Get the list of values for all time steps

Some difficulties could appear for the developer: how to know the result of an element?

To answer this question must find in the GiD documentation: Help->Customization help->Tcl and Tk extension

GiD_Info command provide a lot of information of GiD data.

It is possible to use "*GiD_Info list_entities elements ....*" to get the information of results you want

GiD_Result is a more specialized command than can create and get information from results, we will use it.

For example, to know the value in the element number 676 of the result named 'Velocity (m/s)' of the analyis named 'Hidraulic' in the last time step=35.0 we can write this in the command line

(-np- is a GiD trick that mean that the next is a Tcl command instead of 'GiD process keywords')

```
-np- W [GiD_Result get -selection {676} [list "Velocity (m/s)" "Hydraulic" 35.0]]
```

and we will obtain something like this:

```
{Result {Velocity (m/s)} Hydraulic 35.0 Vector OnGaussPoints GPQ} {Unit {}} {ComponentNames Vx Vy . {|Velocity (m/s)|}} {676 {0.08044888079166412 -0.28458309173583984 0.0 0.295735627412796}}
```

We must parse the part of the text we are interested on, in this case we want for example to get only the result vx=0.08044888079166412

The string is formatted as Tcl nested lists, it is easy to get the apropiated item with the *lindex* Tcl command

```
set data [GiD_Result get -selection {676} [list "Velocity (m/s)" "Hydraulic" 35.0]]
set vx [lindex [lindex [lindex $data 3] 1] 0]
```

Note: Tcl commands syntax could be seen in Internet, for example at www.tcl.tk/man/tcl/TclCmd/contents.htm

lindex — Retrieve an element from a list

set — Read and write variables

proc — Create a Tcl procedure

```
#result_id : list with 3 items: { result_name analysis_name time_step }
proc GraphElementResult_GetElementResult { result_id element_id component_i gauss_i } {
    set data [GiD_Result get -selection [list $element_id] $result_id]
    set num_components [expr [llength [lindex $data 2]]-1]
    set pos_gauss [expr $gauss_i*$num_components+$component_i]
    set value [lindex [lindex [lindex $data 3] 1] $pos_gauss]
    return $value
}
```

GiD_Result is a GiD-added Tcl command, the rest (*expr, llength*) are Tcl standard commands.

$x returns the value of the variable named x (similar to 'set x')

We must use this procedure to get the values for all time steps, with this other procedure, that return an item with two sublists, one for the time steps and another for the result values on these steps.

```
proc GraphElementResult_GetResultAllTimeSteps { analysis_name result_name element_id component_i gauss_i
} {
    set time_steps [list]
    set values [list]
    if { [lsearch [GiD_Info postprocess get all_analysis] $analysis_name] == -1 } {
        error "Analysis '$analysis_name' doesn't found"
    } else {
        set all_time_steps [GiD_Info postprocess get all_steps $analysis_name]
        foreach time_step $all_time_steps {
            set result_id [list $result_name $analysis_name $time_step]
            set value [GraphElementResult_GetElementResult $result_id $element_id $component_i $gauss_i]
            if { [string is double -strict $value] } {
                lappend time_steps $time_step
                lappend values $value
            }
        }
    }
    return [list $time_steps $values]
}
```

GiD_Info is a GiD-added Tcl command, the rest (*list, lsearch, error, foreach, string, lappend, return*) are Tcl standard commands.

3- Create a graph with the values

There is an special Tcl command to handle postprocess graphs, like create a new graph
GiD_Graph create <graph_name> <label_x> <label_y> <x_values> <y_values> <x_unit> <y_unit> ?<graphset_name>?

We want to create a new graph with the extracted information, checking that its name doesn't exists with this procedure

```
#get an unused graph_name, adding a suffix -2, ...-3 if base_name already exists
proc GraphElementResult_GetUnusedGraphName { base_name } {
    set graph_name $base_name
    if { [GiD_Graph exists $graph_name] } {
        for {set i 2} {$i < 10000} {incr i} {
            set graph_name $base_name-$i
            if { ![GiD_Graph exists $graph_name] } {
                break
            }
        }
    }
    return $graph_name
}
```

and this procedure create the new graph

```
proc GraphElementResult_NewGraph { analysis_name result_name result_component element_id gauss_i } {
    if { $result_component == "" } {
        set component_i 0
        set label_y $result_name
    } else {
        set data [GiD_Result get -selection [list $element_id] [list $result_name $analysis_name 0]]
        set component_i [lsearch [GiD_Info postprocess get cur_components_list $result_name]
$result_component]
        if { $component_i == -1 } {
            error "result_component '$result_component' doesn't found"
        } else {
            set label_y $result_component
        }
    }
    set graph_values [GraphElementResult_GetResultAllTimeSteps $analysis_name $result_name $element_id
$component_i $gauss_i]
    if { [llength [lindex $graph_values 0]] } {
        set graph_name [GraphElementResult_GetUnusedGraphName "Evolution element $element_id"]
        GiD_Graph create $graph_name Time $label_y {*}$graph_values "Sec" "" ""
    }
}
```

When creating the new graph it is interesting to show it, opening the graphs window, this can ben done using this procedure

GidUtils::OpenWindow GRAPHS

The procedure is defined in the file scripts\dev_kit.tcl, that is a file of special interest for Tcl developers because include common tools.
Now it is possible to create a graph for the analysis name Hydraulic, result name "Detph (m)" for the element number 676 (using the first component in case of more than one, like vectorial, and the first gauss point in case of more thant one) with something like this

-np- GraphElementResult_NewGraph Hydraulic "Depth (m)" 0 676 0
-np- GidUtils::OpenWindow GRAPHS

but this is not comfortable for a final user.
To invoke our code we will add a new button to the 'macros toolbar', and when pressing the button a menu will be shown with the current results to be selected, and the the user must select in screen the elements to be plotted.

4- Enhance the GUI

The next procedure creates a new Tk menu widget and add to it items for the current results with the procedure *AddGraphsResultsToMenu* (t hat is defined somewhere in the GiD scripts to do this task), and then shows the menu on the right of the widget provided as parameter (e.g. our toolbar button)

```
proc GraphElementResult_ShowMenuResults { w } {
    set r_menu .gid.mGraphGaussResult
    if { ![ winfo exists $r_menu]} {
        menu $r_menu
    }
    #trick, must prefix with [namespace current] because macros are defined in the
"toolbarmacros::macrospace" namespaceto not pollute the global names
    AddGraphsResultsToMenu $r_menu [namespace current]::GraphElementResult_SelectedResult
Point_Evolution
    set x [expr {[winfo rootx $w]+[winfo width $w]}]
    set y [winfo rooty $w]
    tk_popup $r_menu $x $y
}
```

When selecting a menu option it will invoke the procedure *GraphElementResult_SelectedResult* that will do the task we want: allow the user selection of elements and create its graphs.

```
proc GraphElementResult_SelectedResult { args } {
    lassign $args result_name result_component
    set current_analysis [GiD_Info postprocess get cur_analysis]
    set current_step [GiD_Info postprocess get cur_step]
    set header [GiD_Result get -info [list $result_name $current_analysis $current_step]]
    if { [lindex [lindex $header 0] 5] != "OnGaussPoints" } {
        W "This tool is only valid for elemental gauss-points results"
    } else {
        set gauss_i 0 ;#set hardcoded the first gauss point of the element (can allow user selection in
case of more than one)
        set element_ids [GidUtils::PickEntities Elements multiple]
        if { $element_ids != "" } {
            foreach element_id $element_ids {
                GraphElementResult_NewGraph $current_analysis $result_name $result_component $element_id
$gauss_i
            }
            GidUtils::OpenWindow GRAPHS
            GiD_Redraw
        }
    }
}
```

The procede *AddGraphsResultsToMenu* verify that our result is one defined on gauss points, else a message is raised

GidUtils::PickEntities is a GiD Tcl procedure defined in 'dev_kit.tcl' that allow a interactive selection of entities.
Finally we create the graphs with *GraphElementResult_NewGraph* , show its window and *GiD_Redraw* force a redraw the screen to show the selected elements in the normal way.

Finally we must change the tcl code of our macro button to invoke GraphElementResult_ShowMenuResults, but this procedure require the Tk button name to place the menu over them.

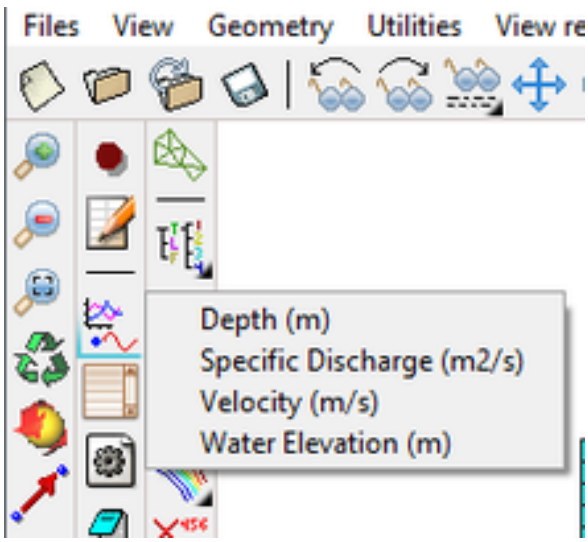
This is a problem because we don't know its name (the macros toolbar doesn't provide the button name to its command). We define an auxiliary procedure that returns this name in a tricky way (getting the name from the stack of Tcl procedures with the *info level* Tcl command).

```
#trick to be able to show the menu on the right of a macro button (because macros doesn't provide the
button name to its command)
proc GraphElementResult_FindParentButton { } {
    set command [info level -1]
    set b ""
    set w .gid.bitmapsMacrosToolbar
    set inum 0
    while { [winfo exists $w.w$inum] } {
        if { [winfo class $w.w$inum] == "TButton" && [$w.w$inum cget -command] == $command } {
            set b $w.w$inum
            break
        }
        incr inum
    }
    return $b
}
```

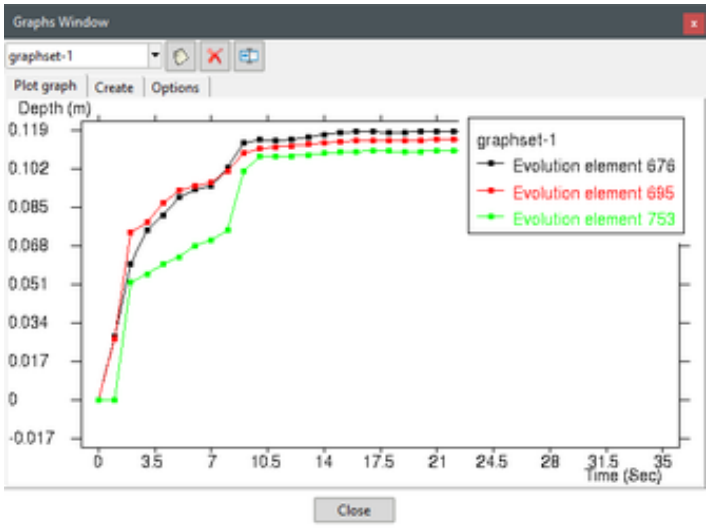
and this is the code that must be finally invoked by our macro:
GraphElementResult_ShowMenuResults [GraphElementResult_FindParentButton]

Note: the macro code must include this line and all required procedures that are not defined in another place.

This is the final aspect of this macro button and its result (selecting the Deph result for the elements 676, 695 and 753.



Menu of results to plot



Graph of the result of three selected elements