



## **GiD**

The universal, adaptative and user friendly pre and post processing system for computer analysis in science and engineering

# **GiD advanced course 2014**



# 1 Introduction

This courses are prepared to be followed using the version 12.0 of GiD.

They are divided in GiD basic courses and GiD advanced courses. We strongly recommend to make the GiD basic courses before the advanced ones.

## 1.1 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 **Windows 32** or **Linux 32**.

Click on the option corresponding to your OS (Windows or Linux), and then follow the instructions of GiD installer to install GiD into your computer.

Even if your OS is x64 bits, we will use for this course the **x32 bits version of GiD**, because some calculation module we will use later on is only available for x32 bits (dynamic libraries loaded in GiD must be compiled for the same platform of GiD).

**Note:** A x64 bits OS can run applications of x64 and x32 bits.

On web page [www.gidhome.com](http://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.

## 1.2 Registering GiD

GiD can work with no license (unregistered), but in this way user can only manage a models with a few number of nodes (about 1000)

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/password>
- Password for Compass problemtypes: <http://www.compassis.com/compass/en/Passwords>

**NOTE:** The USB memory sticks of the course are already registered with a one year password of GiD 12 (USB passwords is valid for both Windows and Linux platforms)

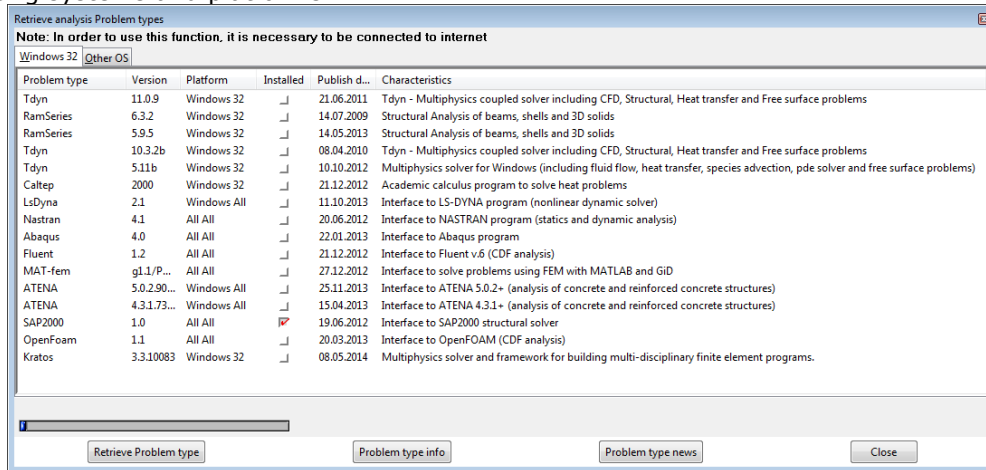
## 1.3 Installing Problemtypes

Following GiD terminology, a 'problem type' is a calculation module able to perform a simulation and which customizes GiD so as the user can apply to the model the material properties, boundary conditions and other information needed for the simulation process.

After the GiD installation process, it is needed to add the required 'problem types', to do is usually is only needed to copy the problemtypes folder inside the \problemtypes gid folder.

It is possible to manually copy these problemtypes (e.g. from the USB GiD\problemtypes folder to our current GiD) or download them from Internet

- If you don't have internet connection, you can simply copy the files from other location.
- If you have internet connection, start GiD and go to menu: **Data->Problem type->Internet retrieve**. A window with the available problem type modules will appear, splitted in the different operating systems and platforms.



There is a mark on the currently installed problemtypes.

To get some new ones, select them and click "Retrieve Problem type" to install them.

Once the problemtype has been downloaded, close the window and you can check it is present in the list of problem types installed in the **Data->Problem types** menu.

**NOTE:** The basic course requires for the 'Run a CFD simulation' chapter to install the CompassFEM (Tdyn) problemtype, that is copied inside the GiD\problemtypes folder of the memory stick (only the Windows x32 version). The example uses a coarse mesh that doesn't require any CompassFEM password.

#### 1.4 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/2014](ftp://www.gidhome.com/pub/GiD_Convention/2014), inside the corresponding folders.

# Table of Contents

<b>Chapters</b>	<b>Pag.</b>
1 Geometry importation and CAD cleaning operations	1
1.1 Description	1
1.2 Import of the model	1
1.2.1 GiD exchange preferences	1
1.2.2 Import model	2
1.2.3 Import tolerances	3
1.3 CAD cleaning operations	3
1.3.1 Visual check of geometry	3
1.3.2 Edit NURBS tool	5
1.3.3 Generate automatic test mesh	5
1.3.4 Look for gaps	6
1.3.5 Collapse model	7
1.3.6 Local collapsing	8
1.3.7 NURBS simplifying	9
1.3.8 Problematic surface	10
1.4 Create volume	11
1.5 Generate final mesh	14
1.5.1 Meshing parameters	15
1.5.2 First try in mesh generation	16
1.5.3 First sizes assignment	17
1.5.4 Local sizes assignment	17
2 Meshing advanced features	21
2.1 Rjump mesher (skip entities)	21
2.1.1 Mesh using automatic parameters	21
2.1.2 RJump mesher	22
2.1.2.1 Assign sizes	23
2.1.3 Skip specific entities	24
2.2 Chordal error	26
2.2.1 Assign sizes by chordal error	26
2.2.1.1 Mesh using automatic parameters	27
2.2.1.2 Mesh with sizes assigned	28
2.2.2 Chordal error to the whole model	29
2.2.2.1 Mesh using automatic parameters	30
2.2.2.2 Chordal error options	31
2.2.2.3 Mesh following chordal error criteria	31
2.2.3 Smoothing options	32
2.3 Force points to mesh	33
2.3.1 Mesh with forced points	34
2.4 Boundary layer mesh	35
2.4.1 Create 3D boundary layer mesh	35
3 Customization	39
3.1 Introduction	39
3.1.1 Interaction of GiD with the calculating module	39
3.1.2 Basic or Advanced	40
3.2 Basic integration, plain text	41

---

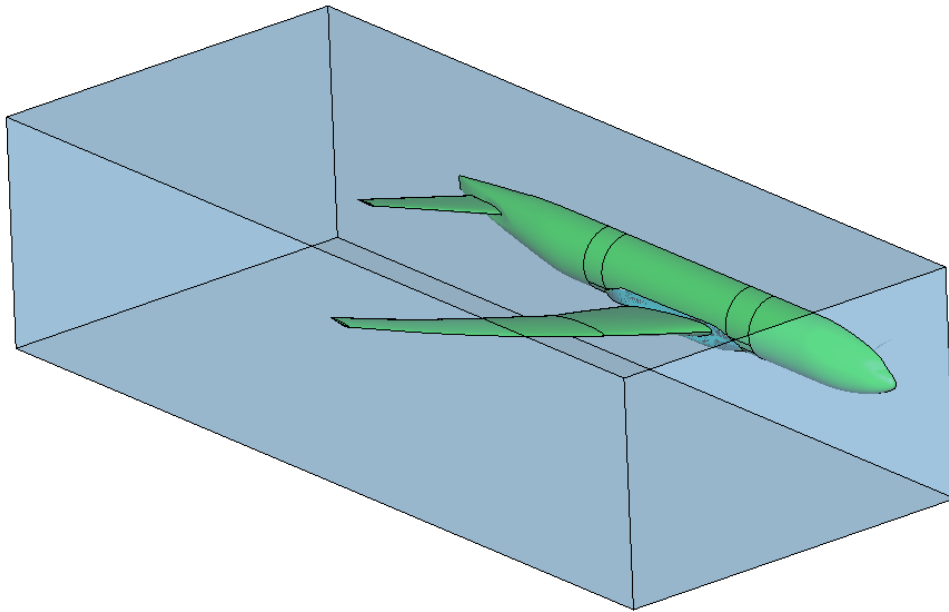
3.2.1 Creating the General File	42
3.2.2 Creating the Materials File	42
3.2.3 Creating the Conditions File	43
3.2.4 Creating the Data Format File (Template file)	44
3.2.5 Creating the Execution file of the Calculating Module	47
3.2.6 Creating the Execution File for the Problem Type	48
3.3 Advanced integration, xml complex fields	49
3.4 Extensions, tcl programing	50
3.5 Using the problemtype with an example	51
3.5.1 Executing the calculation with a concentrated weight	52
3.6 Additional information	53
3.6.1 The main program	54
4 Advanced visualization tools	63
4.1 Stereoscopic view	63
4.2 Mirror effect	64
4.3 Shadows	65
4.4 Decoration	70
5 Animations	79
5.1 Basic rules	79
5.2 Example	82
5.2.1 First observation: resolution	82
5.2.2 Second observation: perspective	83
5.2.3 Third observation: 3D	83
6 Working with large models	87
6.1 Result's cache	87
6.1.1 How it works	87
6.1.2 Example	89
6.2 Using the graphics card	92
6.2.1 Fast/safe visualization mode	92
6.2.2 Drawing methods	95
6.2.3 Example	96
6.3 Fast rotation	98
6.3.1 How it works	98
6.3.2 Example	99
6.4 Simplified view	101
6.4.1 How it works	101
6.4.2 Example	101
6.5 Textures in spheres and contour fill	104
6.5.1 How it works	104
6.5.2 Example	105
6.5.2.1 Preparation	105
6.5.2.2 Point/sphere textures	106
6.5.2.3 Contour fill textures	107



# 1 Geometry importation and CAD cleaning operations

## 1.1 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 (showed 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.

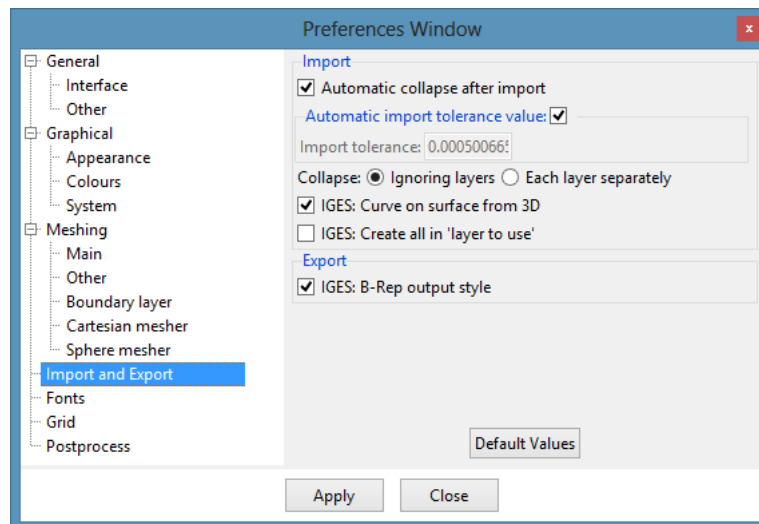
## 1.2 Import of the model

### 1.2.1 GiD exchange preferences

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 showed in figure 2 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 in 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.

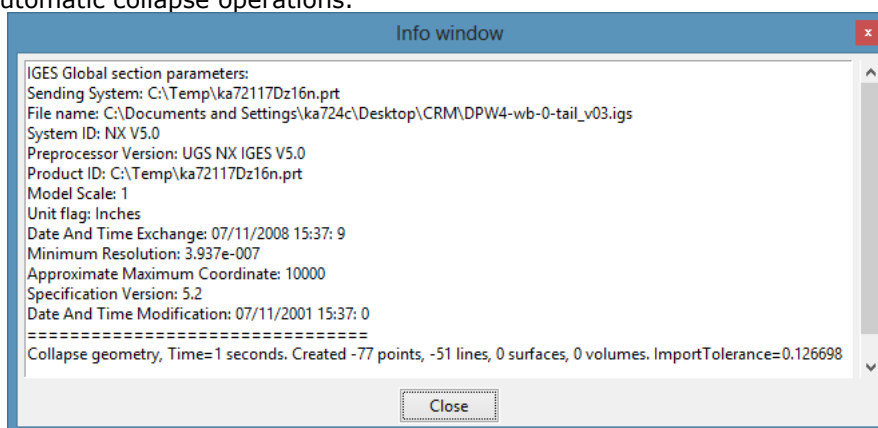
We are going to set all the options showed in the figure and click **Apply**.

### 1.2.2 Import model

The IGES file to be imported is named '**half\_aircraft.igs**' and it should be in the GiD Conference CD unit, in the folder related to **Import and Cad Cleaning Course**.

- 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 like the one showed in the figure should appear with some interesting information like the characteristics of the file imported or the tolerance used for the automatic collapse operations.

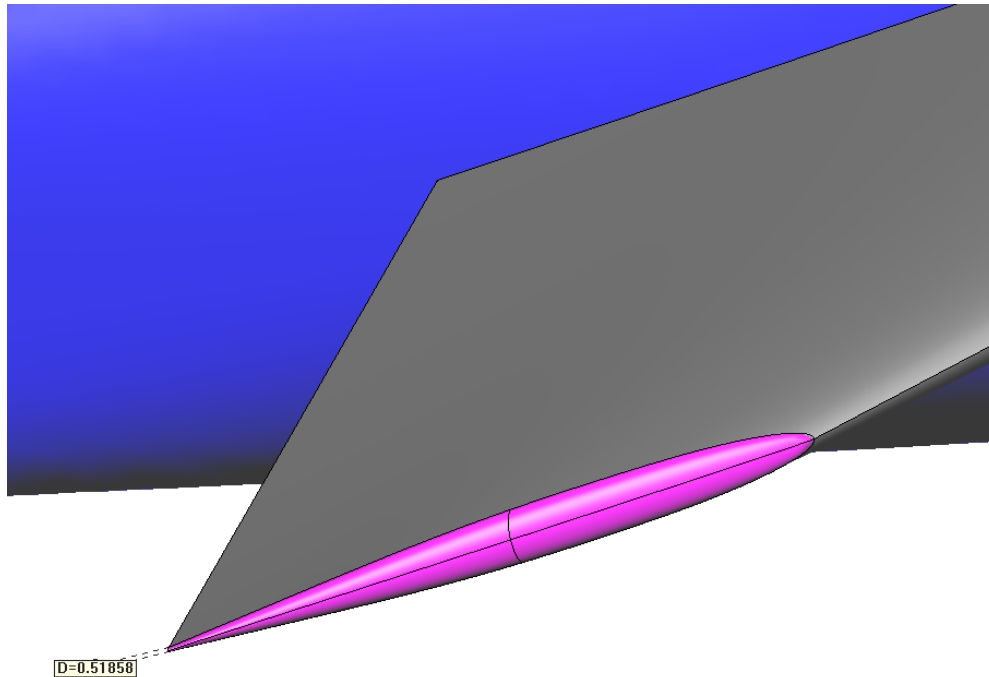


Window showing the information related to the import process.

If you close the window you should see the model imported, representing the half aircraft.

### 1.2.3 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.

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 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 showed in the previous figure, we can try for example with a 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.
- Uncheck the option **Automatic import tolerance value**.
- Enter **0.025** in the **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.

## 1.3 CAD cleaning operations

### 1.3.1 Visual check of geometry

Once we have ensured there are no gaps in the geometry, a useful step to follow when a file is imported is to check visually how it looks like. **Rotate** the model and set the **Render mode** to **Flat (View menu or mouse right button menu)**. With this initial visual scanner some possible big geometry imperfection or import error may be detected.

If some surface is not well rendered (like strange shadows in the view) or no render is done, it means that GiD has had some problem for generating the render mesh of the surface. This should point a possible error in that surface (or not).

- For this check it is good to set the **Render precision** to **1**, in the **Graphical** part of **Preferences** window. Remember to click on **Apply** before closing the **Preferences** window.

**Trick:** the GiD-Tcl procedure `"GiD_Geometry list surface unrendered"` provides the list of surfaces that were not able to be rendered.

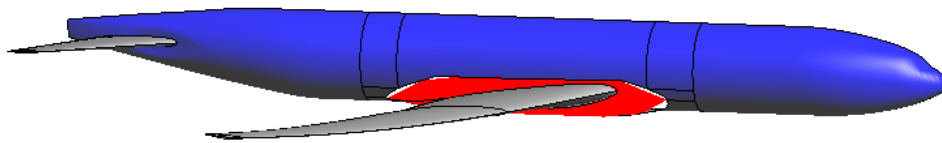
You can write this in the command line to know this list:

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

"-np-" mean that the following is not a process command, but a Tcl procedure, and "WarnWinText" is a GiD-Tcl procedure to show a modeless message window (modeless mean that the flow is not stopped. A window that stops is called modal)

In this case the messages window is empty, because all surfaces were rendered.

Like it can be seen in the figure, the render of the surface which connects the wing of the aircraft with the main fuselage seems to be 'not as fit as desired'.



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 (**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 part of Layers window).
- Send the surface to the new layer by using the **Sent to->Surfaces** option of the mouse right button menu, when the target layer is selected (the corresponding icon in the upper part of Layers window can be also used).

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, like showed in the figure.



View of the problematic surface isolated.

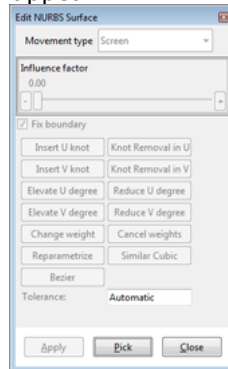
We can change the render mode between **Normal** and **Flat** to understand better the definition of the surface.

### 1.3.2 Edit NURBS tool

For checking if the surface is well geometrically defined we can use the NURBS editing tool. To make faster the visualization process, let's set the **Render** mode to **Normal** (view menu or right mouse button).

- Select **Geometry->Edit->Edit NURBS->Surface**.

The window showed in next figure should appear.



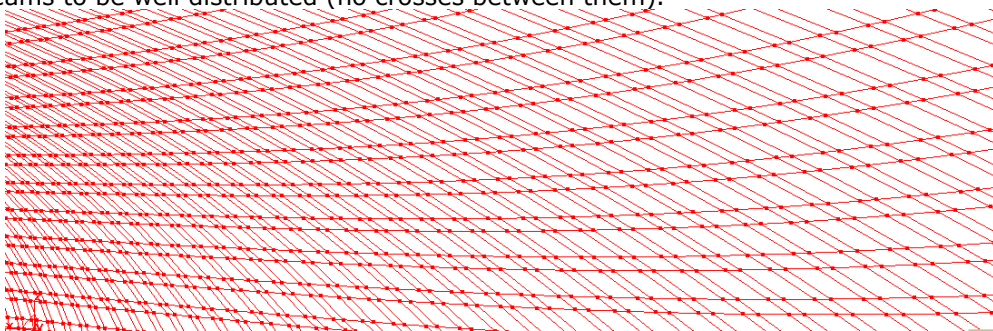
Edit NURBS surface window.

- Click on **Pick** button and select the surface. The control points of the NURBS surface are showed in red, as it can be seen in the figure.



Control points of the NURBS surface.

In this case there are a lot of control points for defining the NURBS surface, so it is needed to zoom on some area to look how their distribution look like (see figure below). The control points of this surface seems to be well distributed (no crosses between them).



Detail of the control points distribution of the surface.

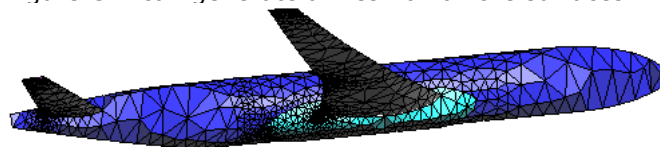
### 1.3.3 Generate automatic test mesh

Other useful check to be done when importing the geometry is to generate a first mesh on it, using the default meshing preferences of GiD. Often, some detail of the model which can be skipped in the

visual scanner is detected when looking at the mesh: possible concentration of elements, entities which cannot be meshed, etc...

- Open the **Preferences** window and click on **Default values** button in the **Meshing** part and press **Apply**.
- Then generate the mesh: Select **Mesh->Generate mesh...** and click **OK** using the mesh size proposed by GiD (in this case, **140**).

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



Render view of the automatic mesh generated.

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

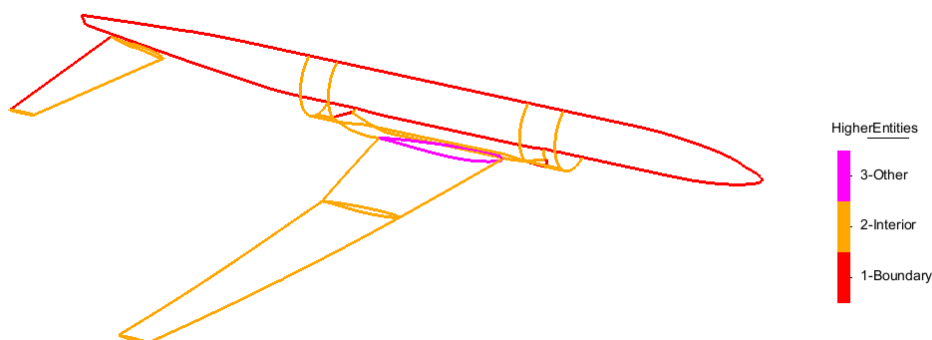
Taking into account the regions where the elements are concentrated, user may look at the size of the mesh there. If the final desirable mesh in that areas is much bigger than the one obtained, some extra geometry mesh edition may be needed to skip some geometrical detail. GiD also offers the option of skipping some line or point when meshing (**Mesh->Mesh criteria->Skip**), so user can skip some geometrical detail when meshing, avoiding the geometry edition. This is not the case of this model, because the final mesh size of the aircraft's surfaces is not so far than the smaller on the mesh obtained in this first automatic mesh generation.

#### 1.3.4 Look 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 of an entity represents the number of entities (with one more dimension) containing itself (in lines, for example, the higher entity represents the number of surfaces owning the corresponding 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 equal to 2.

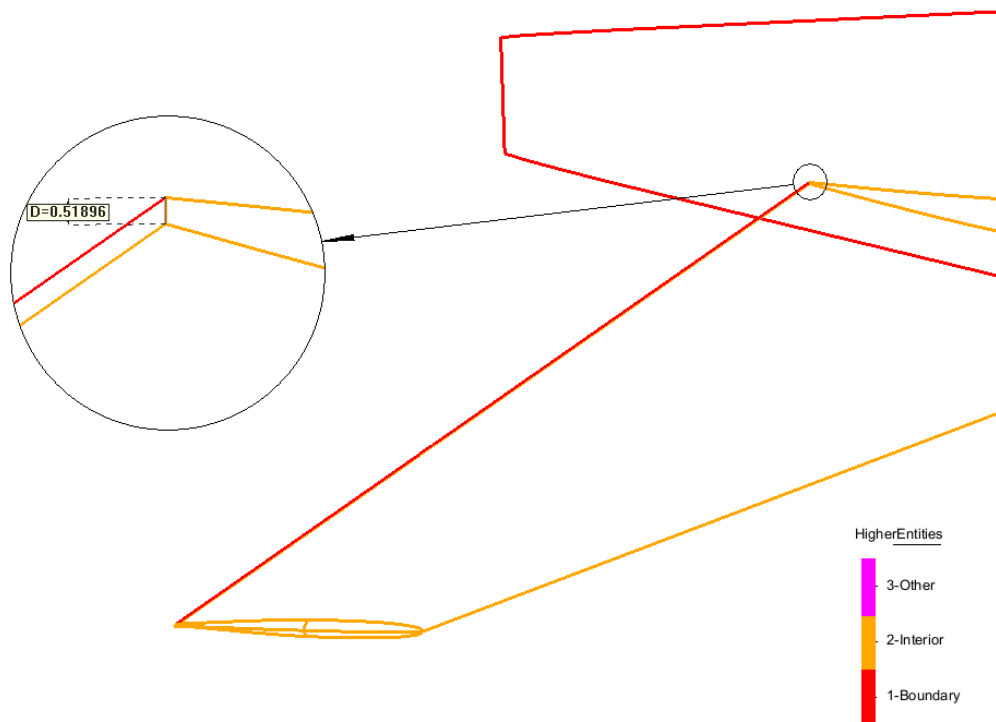
- Select **View->Higher entities->Lines**.



Higher entities of lines in the model just after the importation with tolerance = 0.025.

As it can be seen in previous figure, there are some inner lines with higher entity equal to 3, and some other with higher entity equal to 1. This 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

If we list the entities with **Utilities->List->Lines**

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

### 1.3.5 Collapse model

Taking into account the distance measured of about 0.5 units on small parts, we can try to collapse the whole model with a higher tolerance than the used in the import process, e.g. a 10% of this size to join entities closer than this value.

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

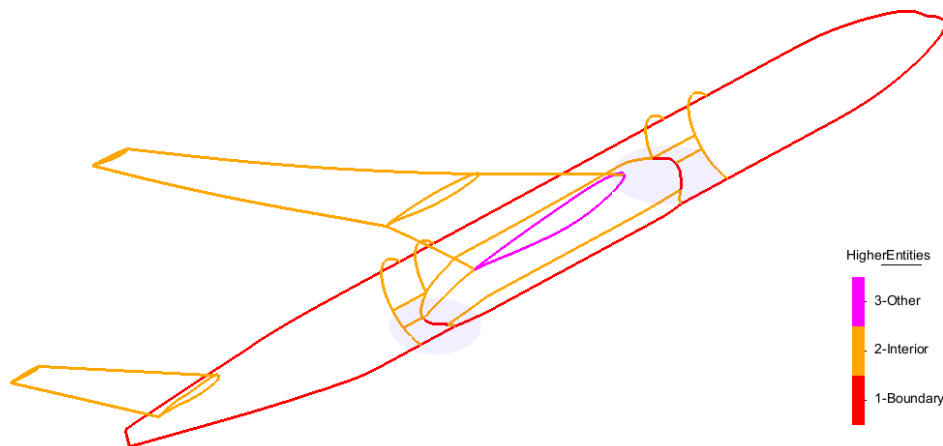
This import tolerance is used both when importing a model, and when collapsing entities of a model already inside GiD.

- 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,-5 lines ,0 surfaces, 0 volumes. Tolerance=0.05  
This mean that three points and five 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 collapsing with tolerance = 0.05.

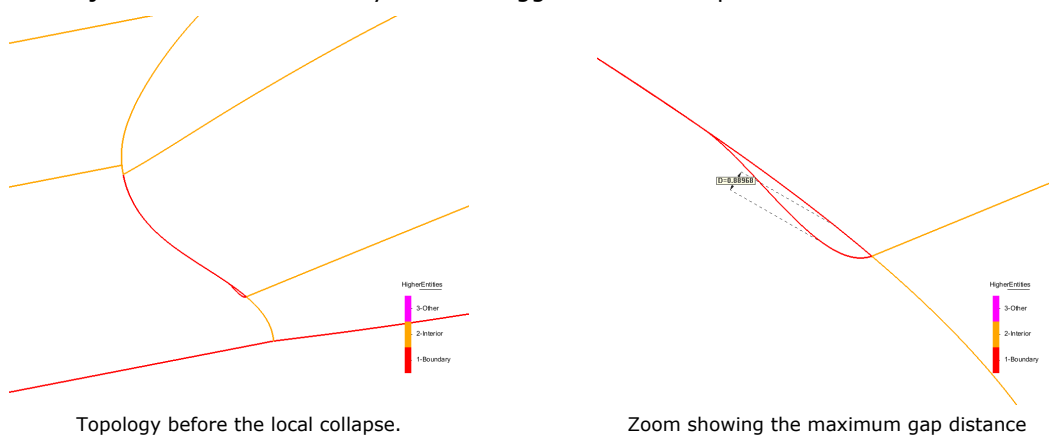
### 1.3.6 Local collapsing

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, so some important detail, such as the ending part of the wings, should 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 **Point In Line** and pick two points on the lines. This distance is about 1.0 units, to force join them it is necessary to set a bigger value as import tolerance.



Topology before the local collapse.

Zoom showing the maximum gap distance

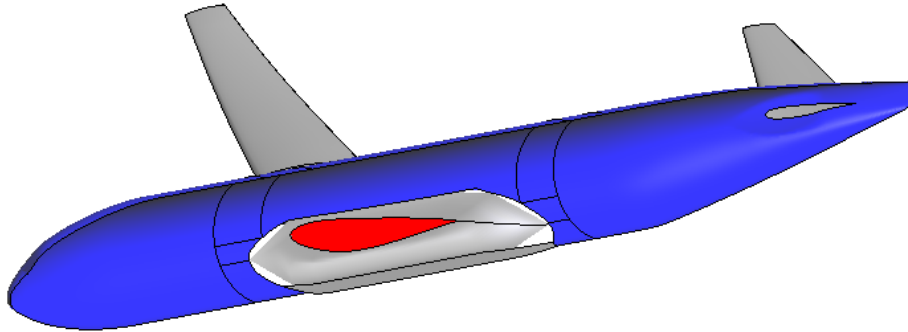
- Set the **Import tolerance** on the **Import and Export** part of **Preferences** window to **1.1**, and click **Apply** before closing the window.
- Select **Geometry->Edit->Collapse->Lines** and select the two lines which surround one of the gaps (the lines with higher entity equal to 1).

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

When finish this local collapse set the Exchange preference to the "**Automatic import tolerance value**", to avoid deleting parts accidentally collapsing with a too big tolerance.

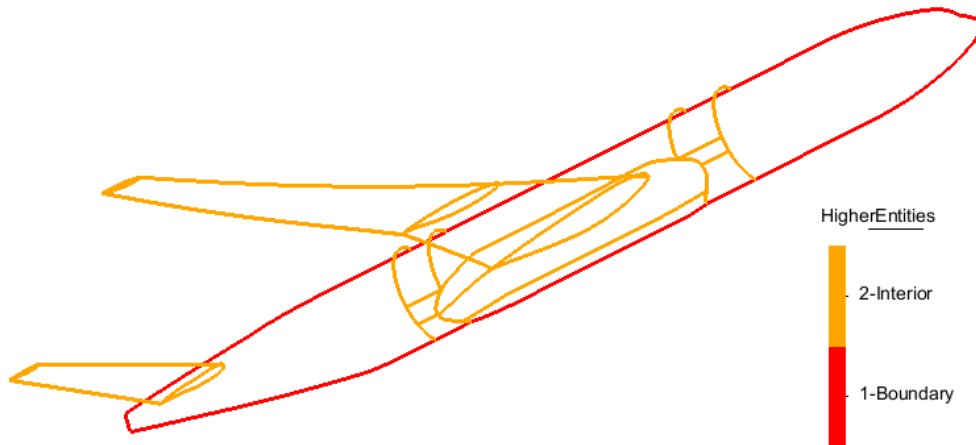
Now, if you try to see the higher entities of lines, only the boundary lines of the half-aircraft should have higher entity equal to one. We can see also some lines with Higher entity equal to 3. This indicates that this lines are owned by three surfaces. In this case, there is no error in the geometry, 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.



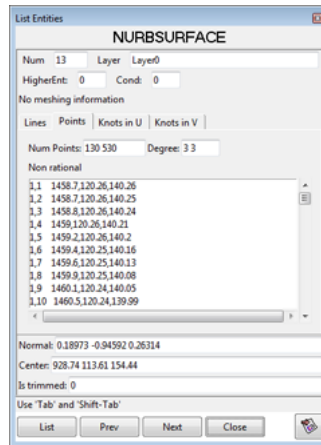
Higher entities of lines after the local collapsing.

### 1.3.7 NURBS simplifying

The surface number 13 has a lot of control points for its geometrical definition.

If we list this entity with **Utilities->List->Surfaces** it take a long time to show this information window:





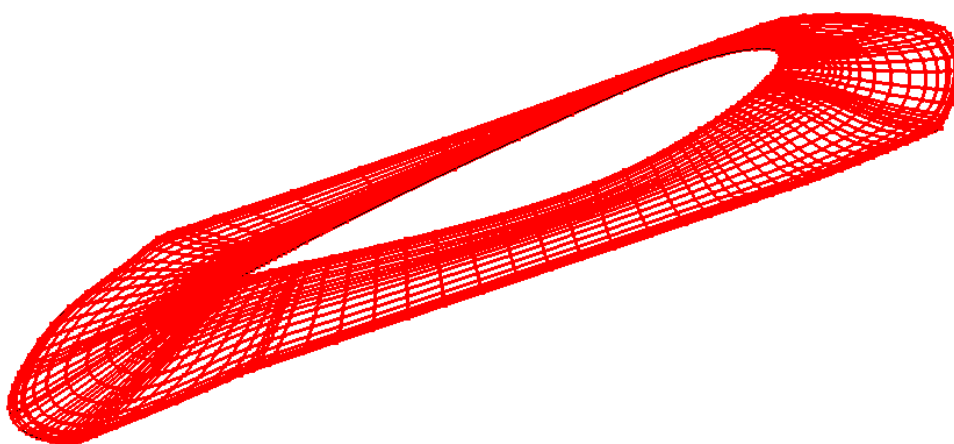
Geometrical information of NURBS surface

We can see that its control polygon has  $130 \times 530 = 68900$  control points. This situation should 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 losing accuracy for capturing the shape of the surface.

To do it open **Geometry->Edit->Edit NURBS->Surface...** and pick the 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  $27 \times 113 = 3051$  control points.

### 1.3.8 Problematic surface

Also after the simplification the render meh 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)



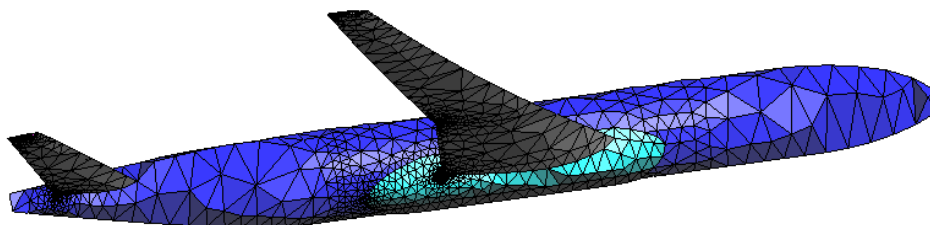
To avoid having small elements the geometry could be modified, collapsing with a size greater to 0.5 to delete the very thin surface of the back of the wing, and convert it in a single line. Then the mesh could have much less elements, but maybe the shape of this part is important for the simulation 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 dividing the surface in two parts in the middle of the  $v$  parametric direction. To do so we use the tool **Geometry->Edit->Intersection->Surface-2 points** This command really calculate the intersection of two surfaces: create the curve intersection of the surface with a virtual plane defined by two points on the surface (the plane contain the points and the vector defined by the averaged normals of the surface at these points)

Must select first the surface, and then the points with labels A, B of the previous picture (use Join to select points) then the surface will be splitted in two parts by the intersection curve. Viewing in render mode, one of this parts is not visualized because the renderer can't generate its render mesh, but if we try to generate a mesh all surfaces are able to be meshed with default settings.

Sometimes it is acceptable that a surface can not be rendered, if it can be meshed for the calculation purposes, but we must be careful because when selecting in render mode these surfaces will not be selected.



Coarse surface mesh

**1.4 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 Lines creation tools, create the rectangle with vertex in the points:

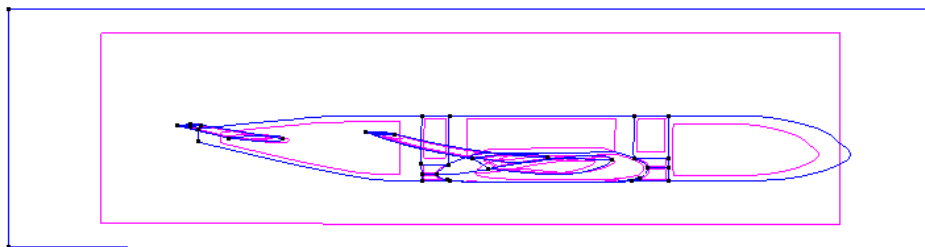
- -220 , 0 , 745
- -220 , 0 , -155
- 3280 , 0 , -155
- 3280 , 0 , 745

**Note:** Remember to use **Ctrl + a** to select an existing point.

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 showed in the figure.

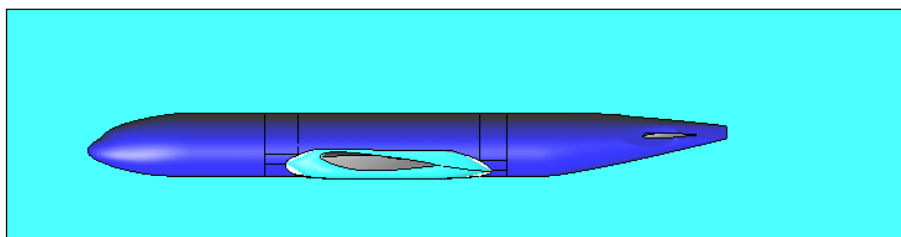


State of the model at this point.

Now we should hole the rectangle created, 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 NURBS surface** and select the rectangular surface created previously.
- Then select the contour lines of the half-aircraft and press **ESC**.

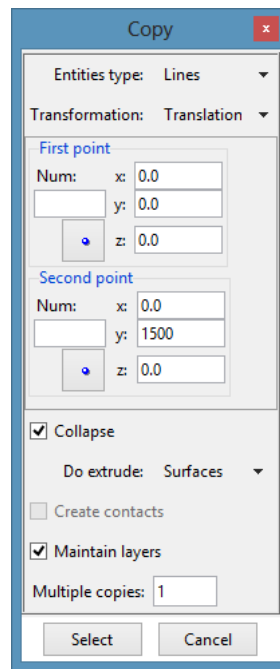
A hole in the rectangular surface should be created, like it is shown in the figure.



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**) and set the options showed in the figure



Options to be set for the extrusion of the lines.

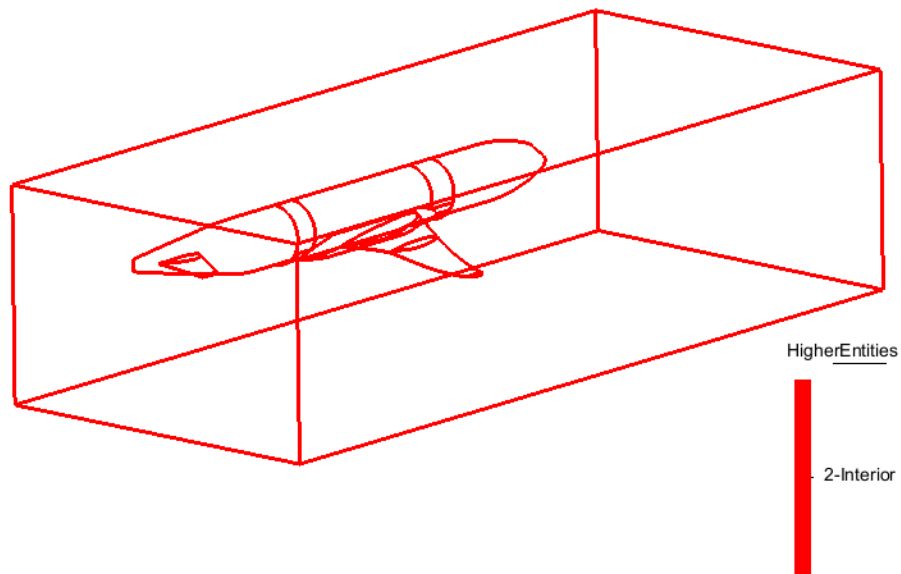
- Click on **Select**, select the four lines of the rectangle created previously, and press **ESC**. Press **Cancel** to close **Copy** window.

- Now the next step is to create the surface which should 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.

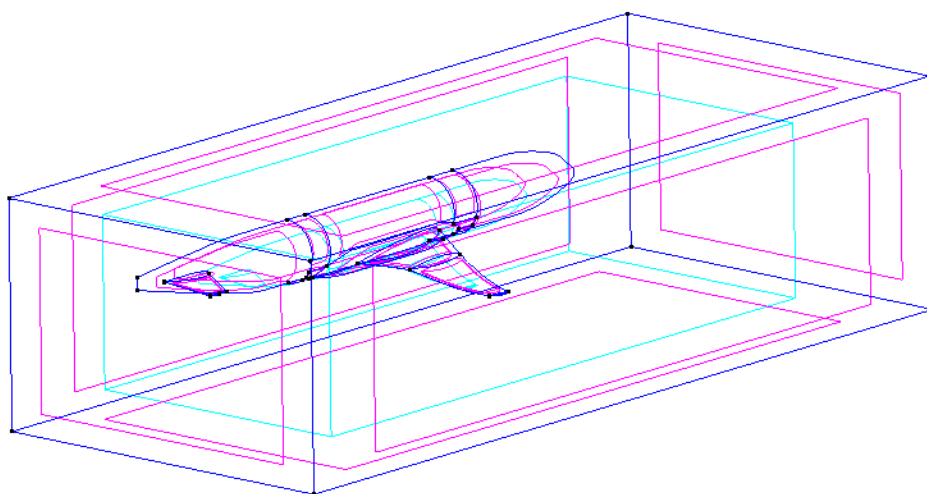


View of higher entities of lines at this point.

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**. Another option to create the volume is to check the **Contextual->Search** option in the right mouse button menu after selecting **Geometry->Create->Volume->By contour**, and then select only one surface. GiD will find the other connected ones automatically.

Now we have finished the geometrical definition of the model, and we should have at this point the model showed in the figure (where the volume is already created).



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

## 1.5 Generate final mesh

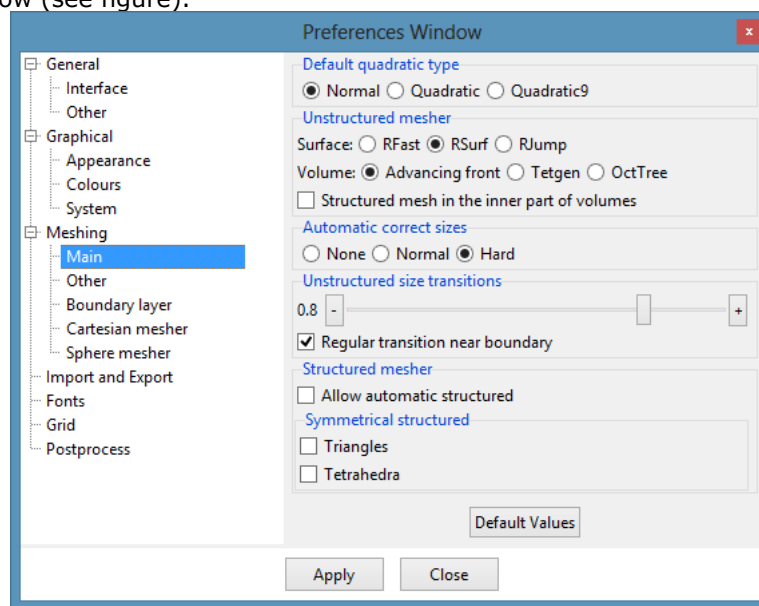
We are going to go deeper in the mesh generation process in this part. The CAD cleaning operations

can be understood as all the operations needed from the moment of importing the model to the moment of run the simulation itself. It 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 determine 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.

### 1.5.1 Meshing parameters

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->Main** part of the **Preferences** window (see figure).



Meshing parameters used for generating the mesh in this example.

Please, set all the parameters as the ones showed in the figure.

And in **Meshing->Other** set the '**Smoothing**' option to **HighAngle**

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 setting this ones and not others.

- **Surface mesher:** GiD uses advancing front method for the surfaces meshing, and it has two basic ways of using it: mesh in the 2D parametric space of the corresponding NURBS surface (**RFast**), or meshing directly in the 3D space (**RSurf**). **RJump** mesher is a special mesher which is similar to **RSurf**, but it is able to skip the inner lines between surfaces which are tangent enough in their common line. In this case we are using **RSurf** mesher: this mesher is slower than **RFast**, but it normally ensures a better quality on the resulting mesh. This aspect is specially important when the surfaces are 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 care on the sizes assigned by the user. This may cause some problem, specially if the sizes assignment to the geometrical entities is not very precisely done. If the parameter is set to **Normal**, GiD takes care on the sizes assigned by the user, but also apply an automatic correction. This correction basically tries to avoid very sharp mesh size changes between close geometrical entities. GiD also tries to assign a realistic size to the entities: in some cases, user assign a general size (or a local size to an entity) which is much more 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**, apart from 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 whether the sizes transitions between areas where the mesh is finer and areas where the mesh is coarser is faster (values closer to 1) or slower (values closer to 0). The bigger is the parameter, the more sharp is the transition between small elements and big ones. The correct value for this parameter depends on the simulation requirements. We will use now **0.8**.
- **Smoothing:** This parameter can be set between three different values: **Normal**, **HighAngle**, **HighGeom**. **Normal** option performs a regular smoothing to the mesh generated. **HighAngle** tries to focus in an improvement of the shape of mesh elements. **HighGeom** tries to make the final mesh 'closer' to the geometry performing topological changes in order to minimize the chordal error of the elements. This last option reduces the chordal error, but can affect to the quality of the shape of the elements. We will use for this example the **HighAngle** option.

### 1.5.2 First try in mesh generation

As it can be seen in previous figures, the default size mesh generated by GiD is too coarse for any kind of simulation. Using the general mesh size which user gives in the moment of generating the mesh, user have a few control for 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 force GiD to give (after the mesh generation process) not only the volume tetrahedra, but also the triangles of the surfaces (by default, only the 'higher dimension' elements are given). This will be useful, because in case the volume mesh could not be generated, we will be able to take a look on the mesh of its contour surfaces. In many times, the problems meshing a volume are related to a bad-quality 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 algorithms, if the selected one fail for some entity another algorithm is tried.

It is possible to change this behaviour with the GiD variable `OnlyUseSelectedMesher`

write this in the command line

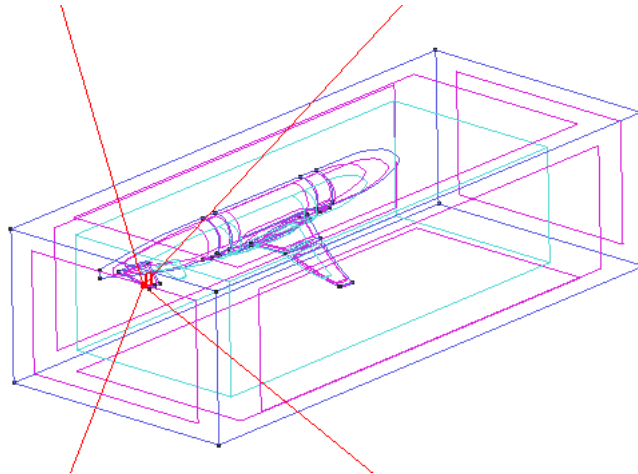
```
Mescape Utilities Variables OnlyUseSelectedMesherVolumes 1 Mescape
```

Then we will detect if the volume advancing front mesher fail (else maybe the volume mesh is generated using the Tetgen Delaunay mesher, with less element quality)

Now generate the mesh:

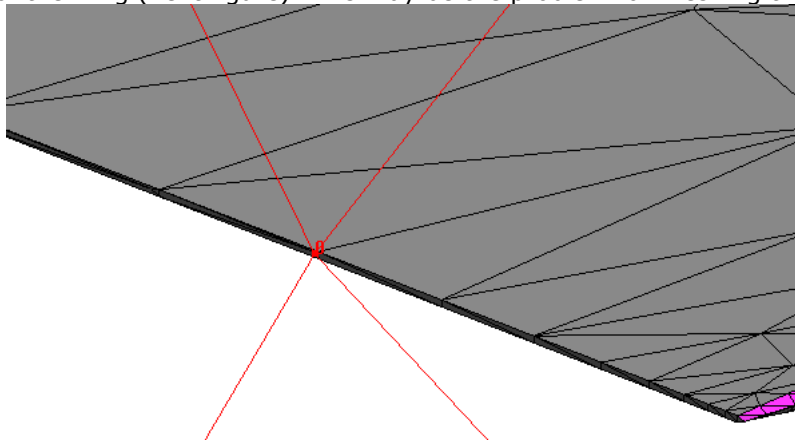
- Select **Mesh->Generate mesh**, eliminate the old mesh if needed, and enter **50** as the general meshing size and click **OK**. (This step may lasts some minutes).

After the mesh generation, a message appears telling that the mesh of the volume 1 cannot be generated ('Couldn't mesh at this location'). Click twice onto this message, and a big red cross will mark the problematic region of the model which have made not possible the volume mesh generation (like it is shown in next figure).



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

If we switch the view mode to mesh (**View->Mode->Mesh**, or the corresponding icon in the toolbar) and zoom 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.



Detail of the zone where the difference between the sizes of neighbour elements is too big for the volume mesher.

### 1.5.3 First sizes assignment

One possible solution for this problem should be to mesh the whole model with a size similar to the one of the edge of the wing, but it should give a big amount of elements, and it should be taken into account that this small size should not be necessary for improving the accuracy for 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 select 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 (**Mesh->Generate mesh**). The mesh can not be generated again, and we see that GiD shows the border part of the tail again as a problematic area.

### 1.5.4 Local sizes 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**.

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**.

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





## 2 Meshing advanced features

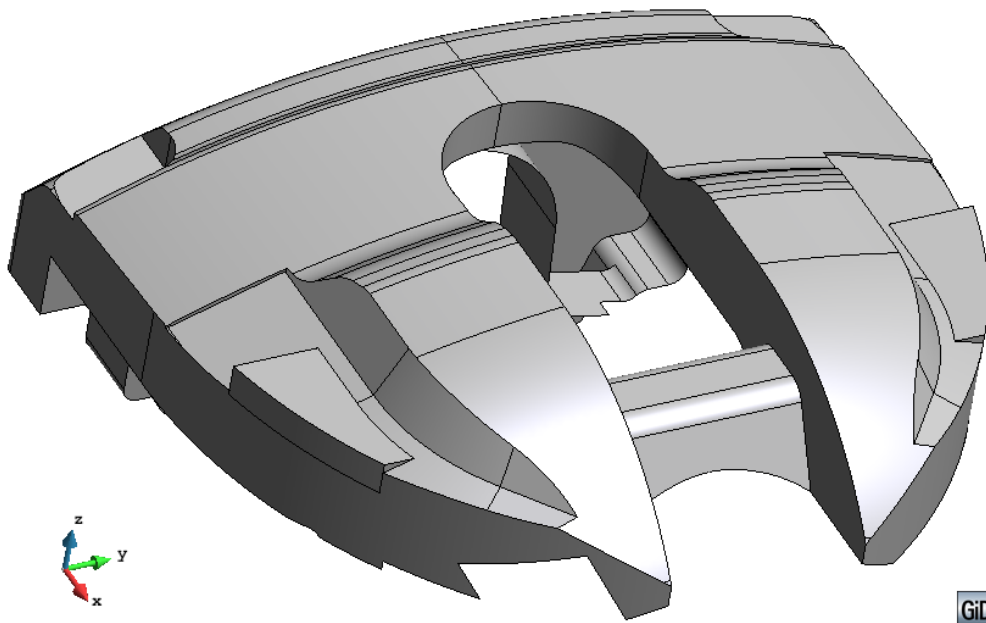
This course is focused in exploring the advanced meshing features present in GiD. It is recommended to follow the basic meshing course before this one.

### 2.1 Rjump mesher (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.

Rjump mesher is a special unstructured surface mesher inside GiD, which is able to skip the inner lines between surfaces in contact. This option is very interesting in the cases where the geometrical definition of the model use high distorted surfaces, or very small ones in comparison with the elements size required for the simulation.

In this course we will use the model '*model\_rjump.gid*', which is a geometrical model of a mechanical part showed in the following picture:

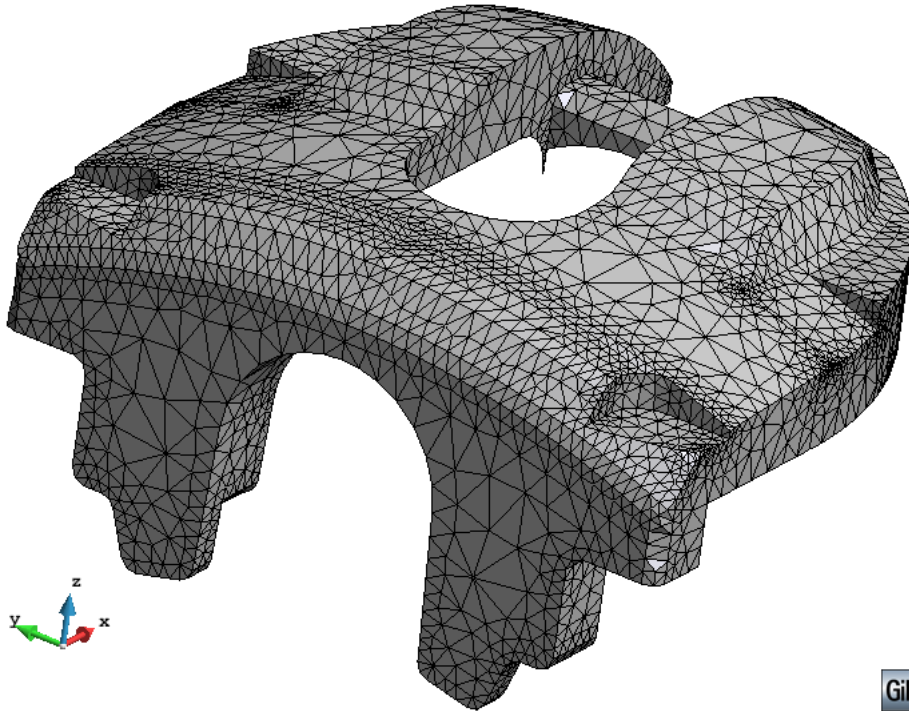


View of the model of a mechanical part used in this course.

#### 2.1.1 Mesh using automatic parameters

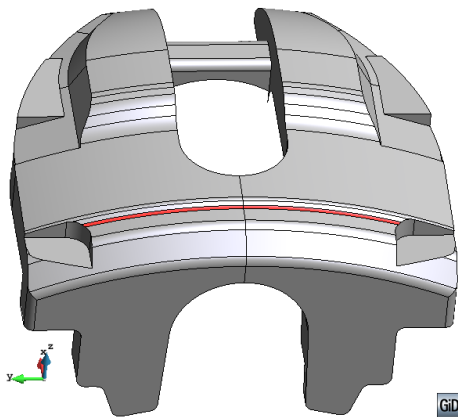
First of all we are going to generate a mesh with all the default parameters in GiD.

- Select the 'Reset mesh data' option from the 'Mesh' menu.
- Open the 'Preferences' window and set the Default values in the meshing branch. Then click on 'Apply' and close the window.
- Generate the mesh setting to 9 the general mesh size. The following mesh should be generated:

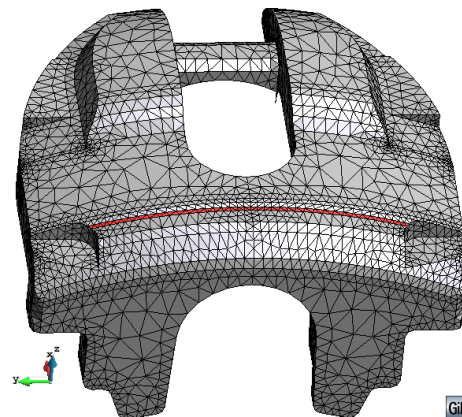


View of the mesh generated with the default meshing parameters of GiD.

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.

### 2.1.2 RJump mesher

If user wants a mesh with a more uniform element size distribution, the Rjump mesher should be set. Rjump is based in the advancing front technique meshing directly in the 3D space, and skipping the following entities 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.

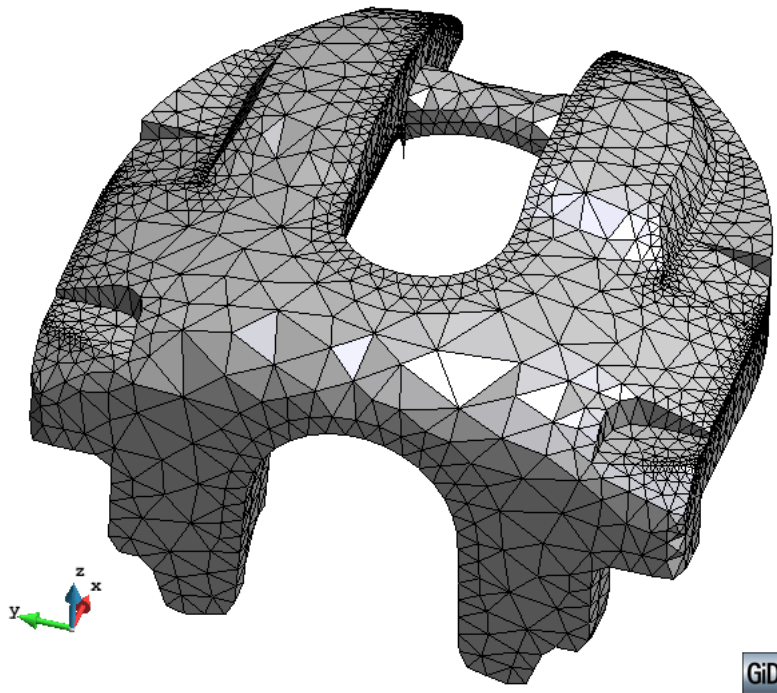
The 'tangent enough' concept is defined by a maximum angle formed by the lines (in case of points) or the normals of the surface (in case of lines). The default angle is 10 degrees.

Other criteria are applied in order to allow or not an entity to be skipped:

- Entities with some material or condition assigned are never skipped.
- Entities which are in some group different from the groups their parents are.
- 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 'Rjump' as surface mesher.
- 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 Rjump mesher.

As it 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.

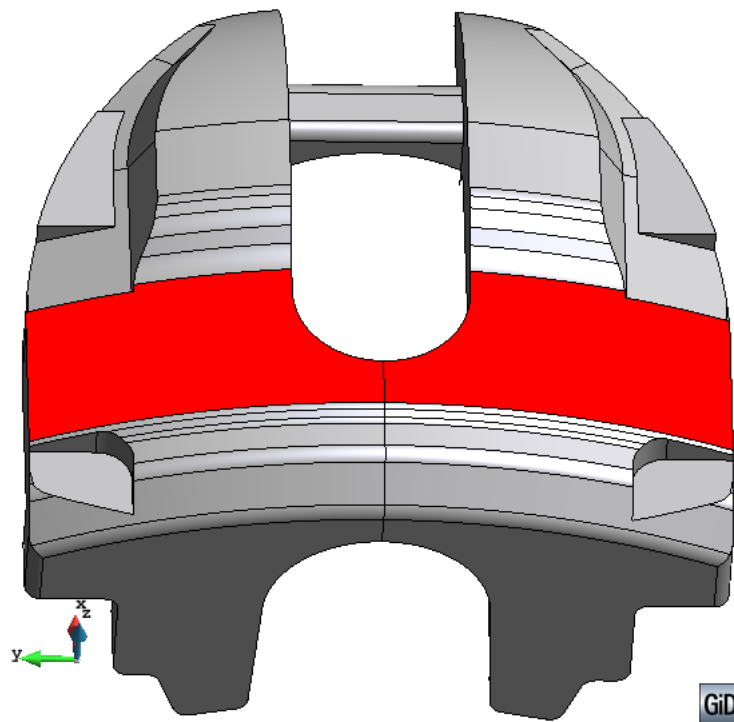
Note that the lines which form a sharp edge between the surfaces they belong to are not skipped. It has to be pointed out that the nodes of the mesh are exactly placed onto the surfaces.

Often meshing with Rjump is more difficult than the conventional meshers, so in some complex geometrical models, sometimes the mesher cannot skip the desired entities. In this cases GiD generate the meshes without skipping any entity.

### 2.1.2.1 Assign sizes

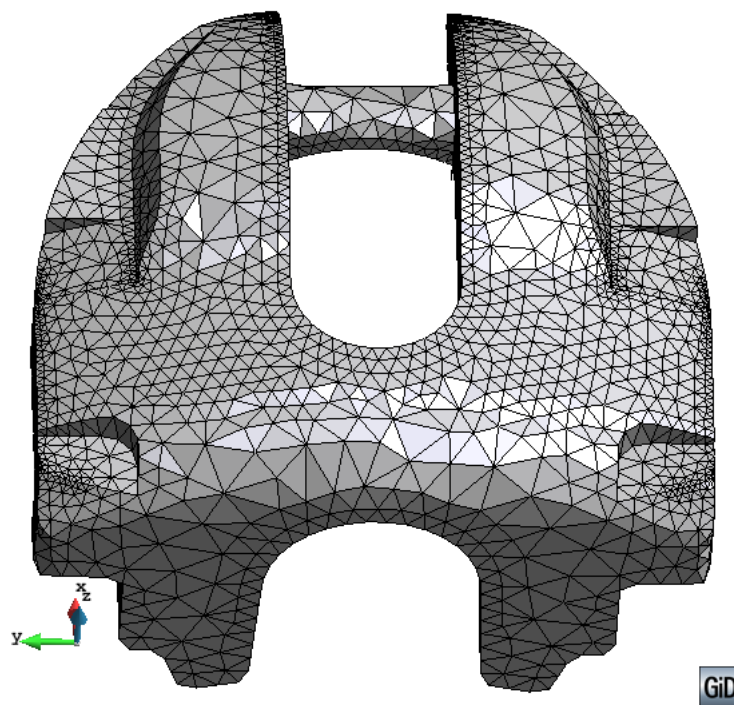
Although Rjump mesher mesh patches of surfaces together, user can assign different element sizes to the different surfaces or lines, and they will be considered. Let's see it in an example:

- Assign 3 as unstructured size of the surfaces highlighted in the following figure:



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 using Rjump mesher 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.

### 2.1.3 Skip specific entities

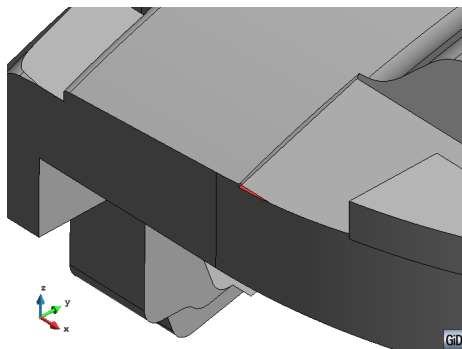
As it has been seen in the previous point, if Rjump mesher is selected, all the lines and points

accomplishing the rjump criteria are skipped. However, the user may want not to skip some entity, or skip just some specific line or point. For this purpose user may use the 'Skip' options in the 'Mesh criteria' part of the 'Mesh' menu.

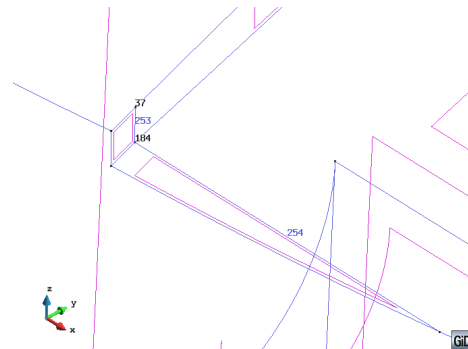
Accordingly to the easier way of setting the parameters for meshing, user may be interested in one of these two options:

- Skip almost all the entities accomplishing Rjump criteria except some line or point. In this case, Rjump mesher must be set in the 'Preferences window', and user may select manually the entities not to be skipped (Mesh criteria->No Skip->...).
- Only skip some specific entity (and don't skip the major part of entities of the model). In this case, regular mesher must be set (RFast or RSurf), and the entities to be skipped must be selected manually (Mesh criteria->Skip->...).

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 where these entities are is highlighted in red, as well a zoom of the zone is presented with the corresponding labels in this area.

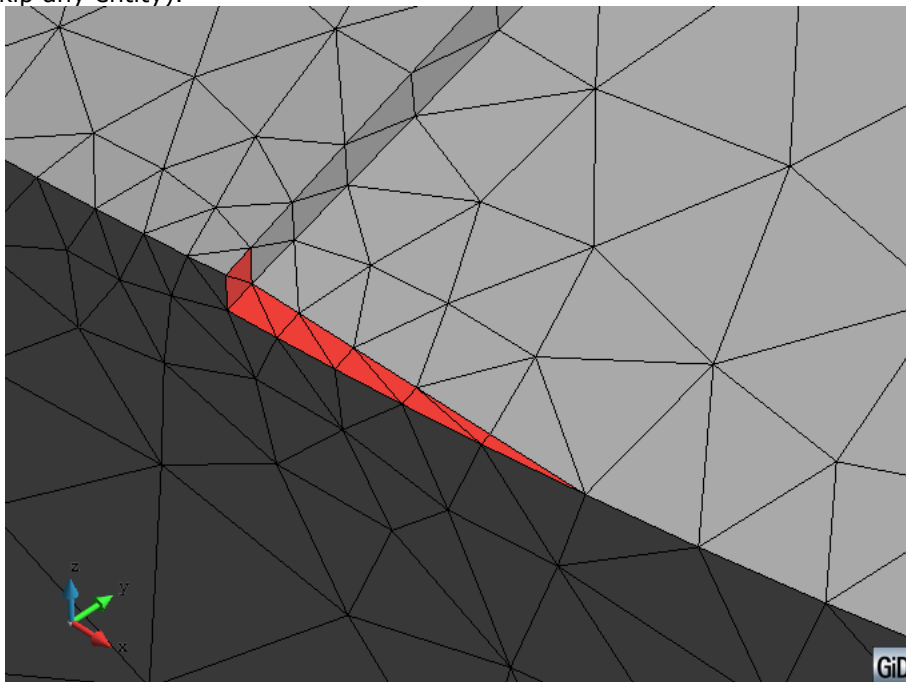


Region where the entities to be skipped are, highlighted in red.



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 default mesher was used (don't skip any entity).

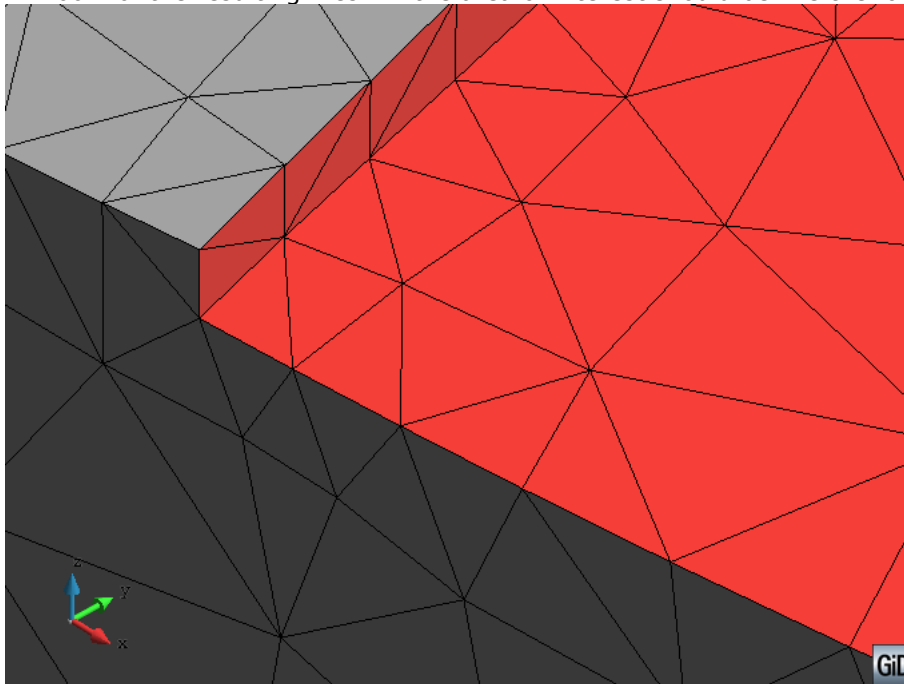


Detail of the mesh in the zone of interest using conventional mesher (don't skip any entity).



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 skipped or not when meshing.
- Set 'RFast' mesher 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.



View of the mesh generated skipping the corresponding lines and points.

As it 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 where the lines 253 and 254 are located with respect to the triangles).

## 2.2 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.

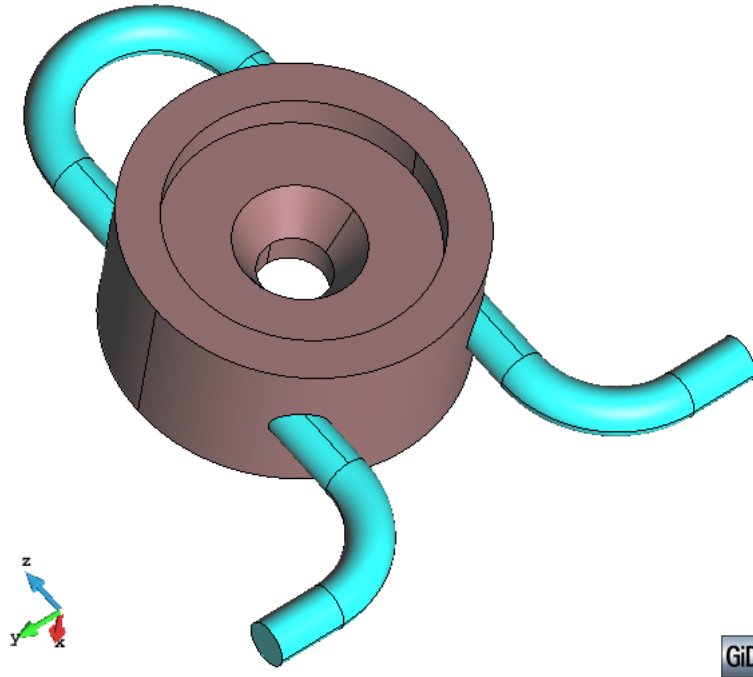
Depending on the characteristics of the geometrical model treated, it may be interesting to assign sizes to the geometrical entities according to a given chordal error, or let GiD refine the regions of the model with higher curvature in order to accomplish this chordal error.

### 2.2.1 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:

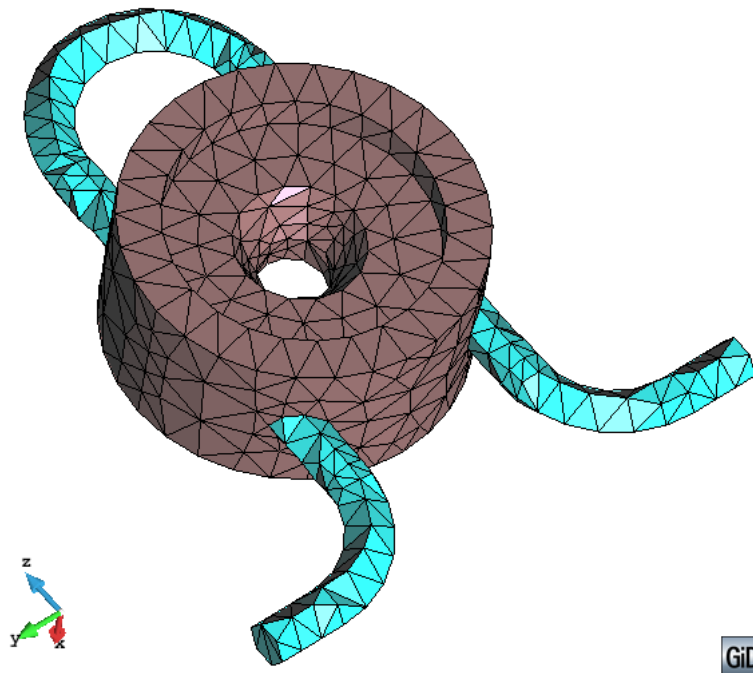


View of the model used in this part of the course.

### 2.2.1.1 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. Then click 'Apply' and close the 'Preferences' window.
- Generate the mesh using 10 as general size. The following mesh should be generated:



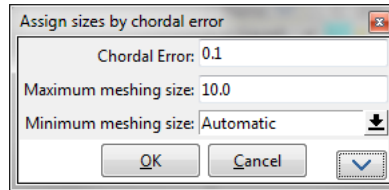
View of the mesh generated with GiD default values and a general mesh size of 10.

As it can be seen, this mesh is probably too coarse in cilindric volumes.

### 2.2.1.2 Mesh with sizes assigned

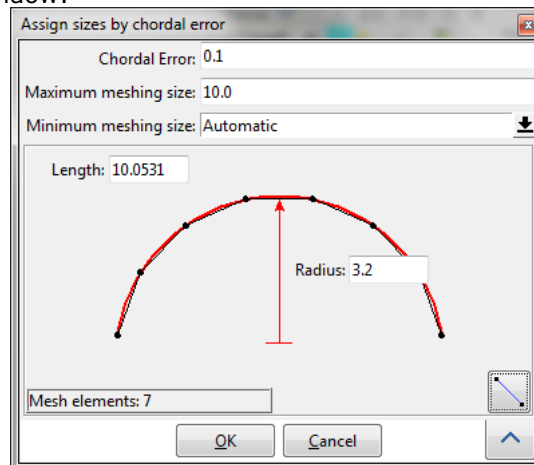
We are now going to assign better sizes to the surfaces and lines of the model, so as the final mesh size will be guided by a given chordal error criteria.

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



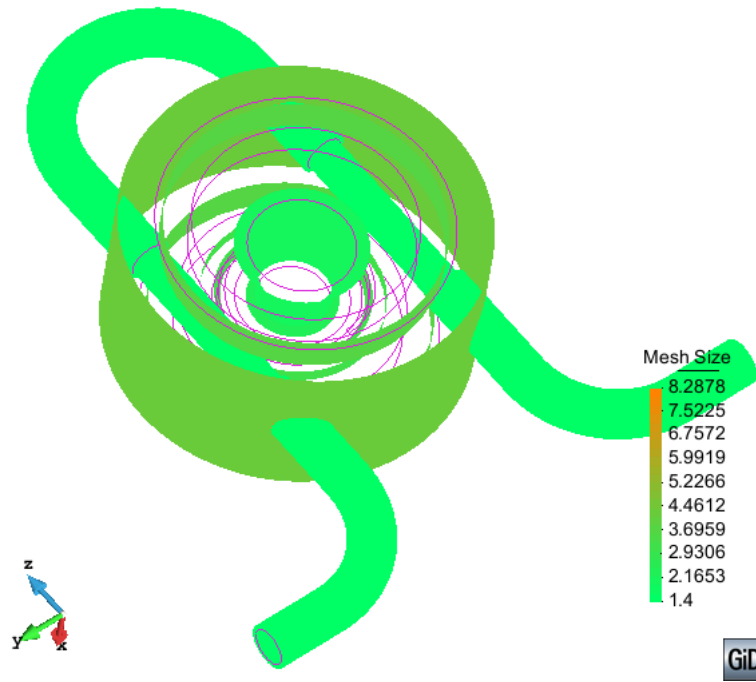
In this window user can enter the absolute value of the chordal error allowed, and the maximum and minimum mesh size desired. To get an idea of the corresponding element size given by a chordal error, user can expand the window by clicking on the button in its right-lower part (beside CANCEL button):

- Expand the window and click on the 'line' button of the right-lower part of the window, and 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.

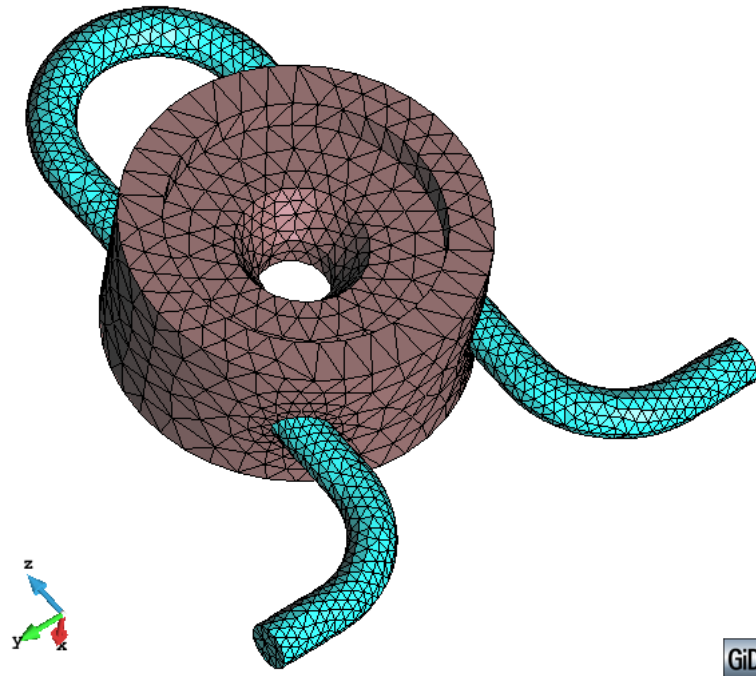
- Click 'OK' in the window. Now GiD has assigned automatically a mesh size to the lines and surfaces of the model, trying to accomplish with the chordal error specified by the user.
- Select 'Draw->Sizes->Surfaces' from the 'Mesh' menu. Then the sizes assigned can be seen onto the model as shown in the 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 branch of the Preferences window. This mesher tends to give better quality meshes than RFast one.
- Generate the mesh with 10 as general size, and see the result:



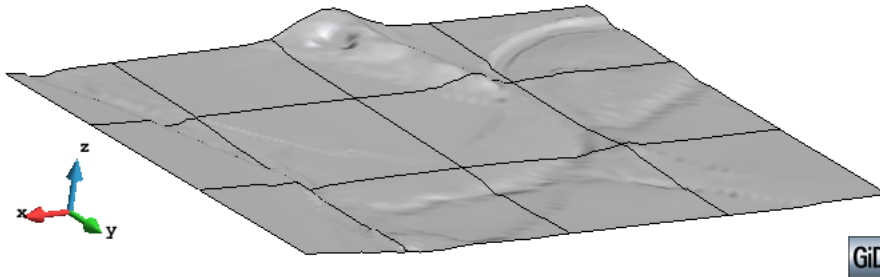
Final mesh generated with the sizes assigned automatically by GiD, following the chordal error specified by the user.

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.

### 2.2.2 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 follow a chordal error criteria to the whole model. 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:



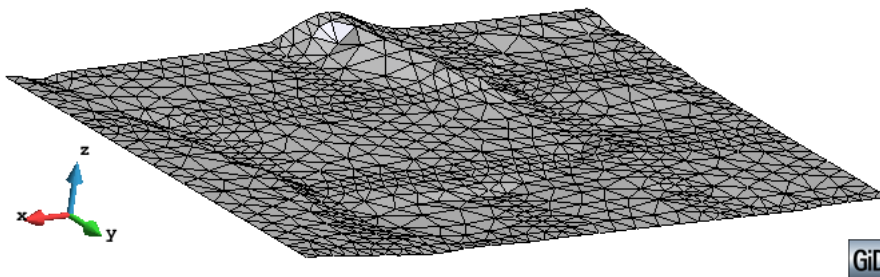
View of the geometrical model used in this course.

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.

### 2.2.2.1 Mesh using automatic parameters

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

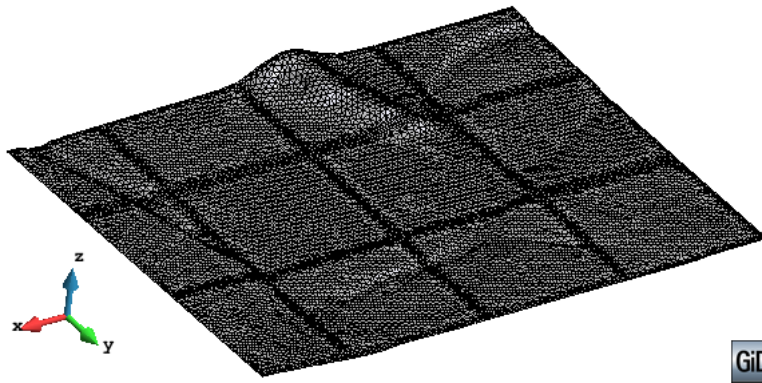
- Set the Default values in the Meshing branch of the 'Preferences' window. Then click 'Assign'.
- Select 'Generate mesh' option from Mesh menu, and generate the mesh with the automatic size proposed by GiD. The resulting mesh should look like the following one:



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 this 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:

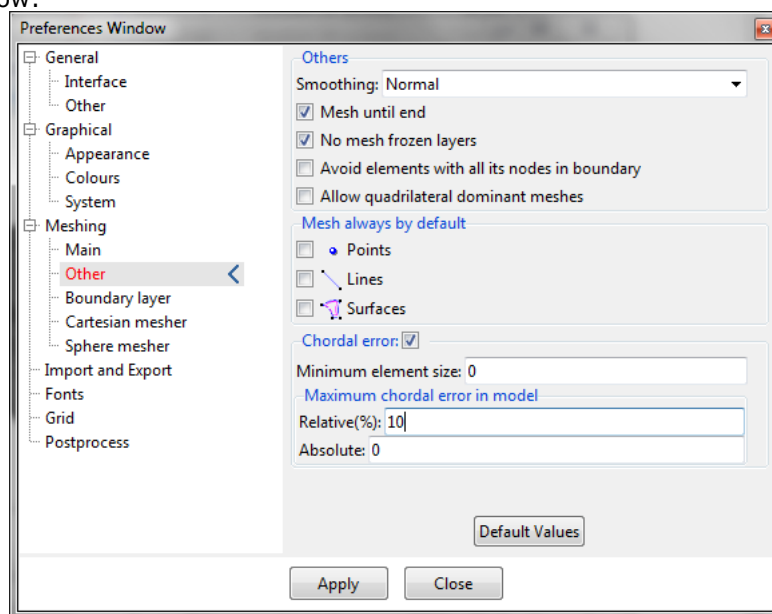


Mesh of the model with a general mesh size equal to 4.

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.

### 2.2.2.2 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->Other branch of the 'Preferences' window:



View of the Meshing->Other branch of the preferences window, where the Chordal error parameters are.

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

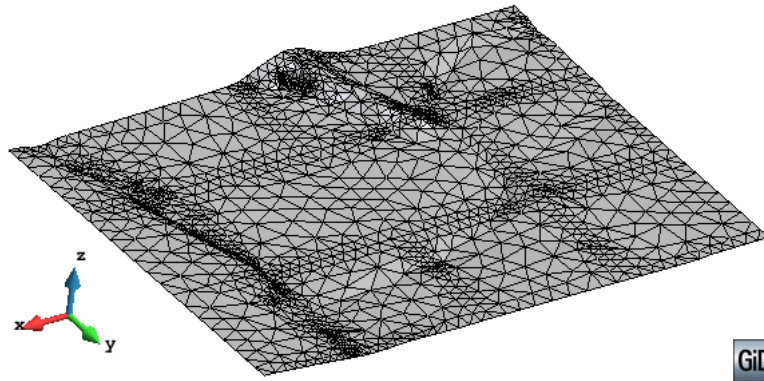
- relative: this value is the chordal error of an element divided by its characteristic size.
- absolute: this value is the chordal error itself.

A third 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, so this parameter may be interesting in some cases.

### 2.2.2.3 Mesh following chordal error criteria

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, RSurf mesher, in general, trends to represent the original geometry in a more accurate way).
- Check the 'Chordal error' option in the Meshing->Other branch, and set to 10% the relative value of the 'Maximum chordal error in model'.
- Generate the mesh of the model using 15 as the general size. The resulting mesh should be like the following one:



Mesh of the model using 10% as the maximum relative chordal error allowed.

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

### 2.2.3 Smoothing options

In the meshing part of the 'Preferences' window, user can find the 'Smoothing' options. These options are:

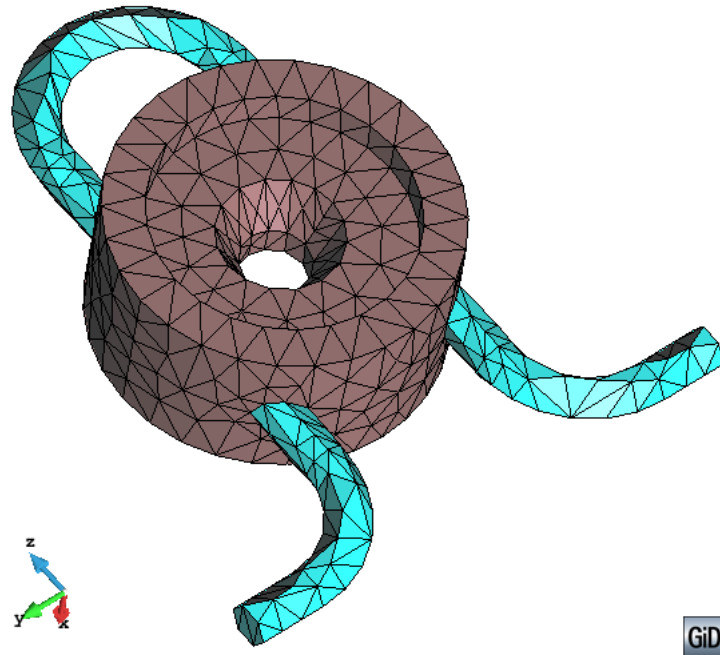
- Normal
- HighAngle
- HighGeom

The first option performs a standard Laplacian-like smoothing algorithm. The second one tries to improve 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 exaggerated 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 and click on the 'Apply' button.
- Then set 'Rsurf' as surface mesher in the Meshing->Main branch, and set 'HighGeom' as 'Smoothing' parameter in the Meshing->Other branch.
- Generate the mesh 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 as smoothing.

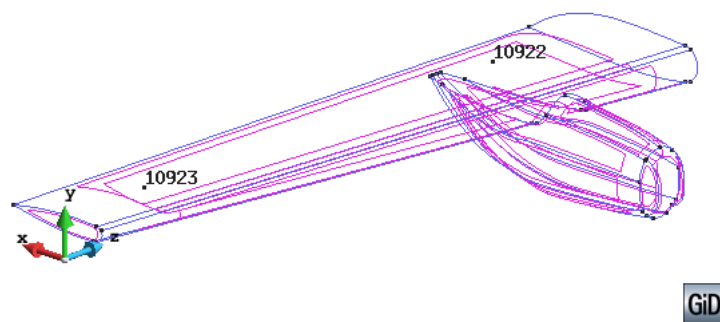
Although the mesh has a big chordal error, comparing this mesh with the one obtained with the default parameters (see section [Mesh using automatic parameters -pag. 27-](#) of the course), user can see that the configuration of the elements tends to follow the curvature directions of the surfaces.

### 2.3 Force points to mesh

The geometrical definition inside GiD follows a hierarchical approach: a line must have points in its ends, 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'`, showed in the following figure. The model represents an aircraft wing.



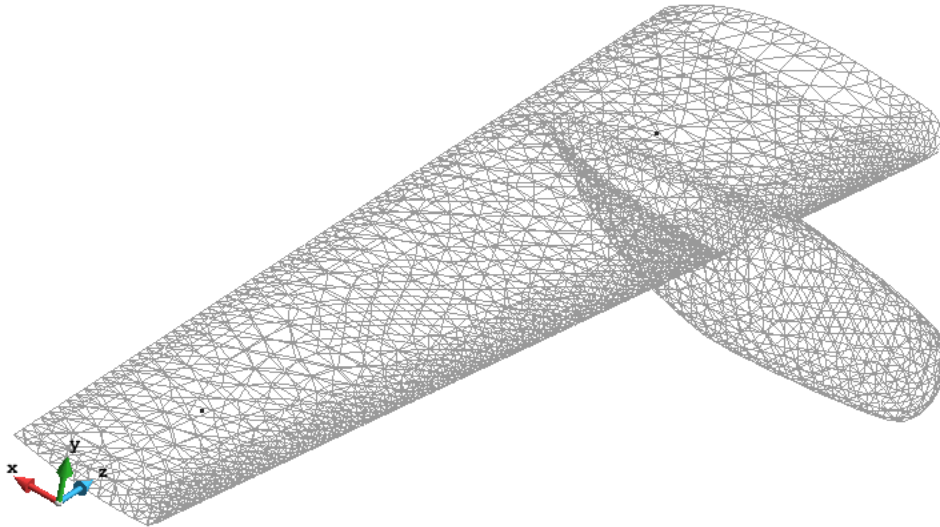
View of the model used in this course.

In the figure two points can be seen (labeled as 10922 and 10923). These points are not owing to any higher order geometrical entity (it can be seen using the View->HigherEntity functionality).

Let's generate a mesh of the model:

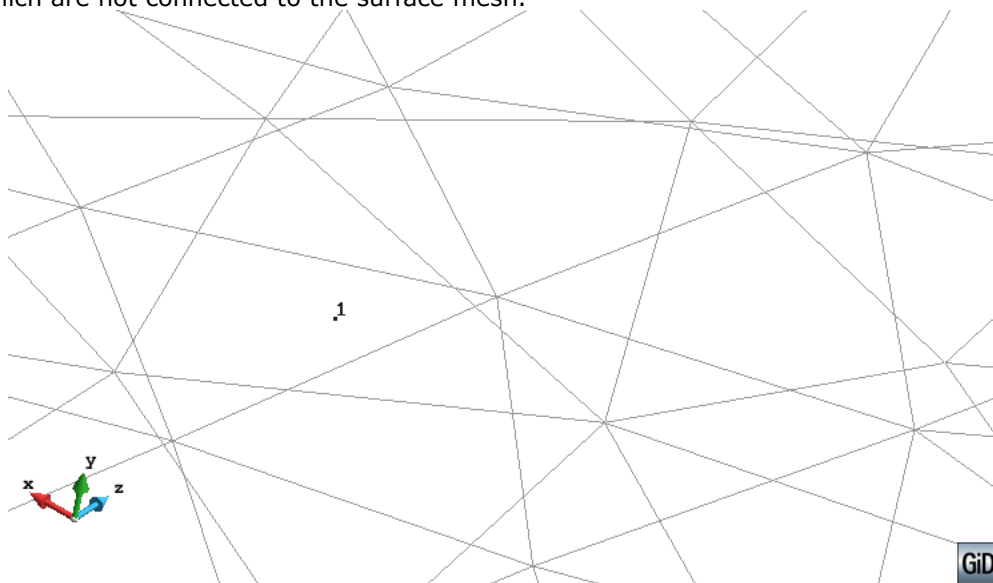


- Generate a mesh using 100 as the general mesh size, setting the option 'Get meshing parameters from model'. The resulting mesh should be as follows:



Mesh of the model using 100 as general mesh size.

If we zoom the zone where the isolated points are, we can see that those 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 (coming from the point 10923 of the model) is isolated.

### 2.3.1 Mesh with forced points

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

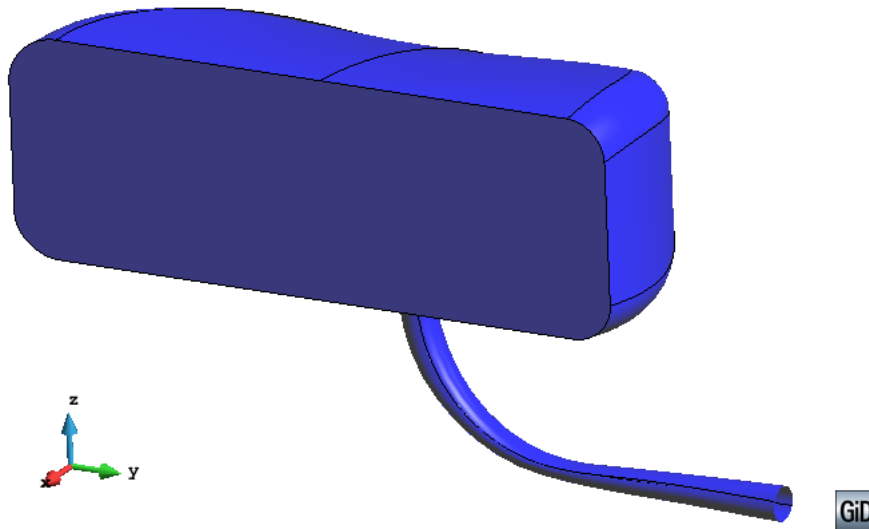
- Select 'Mesh criteria->Force points to->Surface mesh' in the 'Mesh' menu.
- Select the surfaces of the model and press ESCAPE. (steps to follow are indicated in the Warnline, in the lower part of GiD window).
- Then select the two points and click ESCAPE. GiD will assign the closest surface (between 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.

## 2.4 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 (FSI). 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.

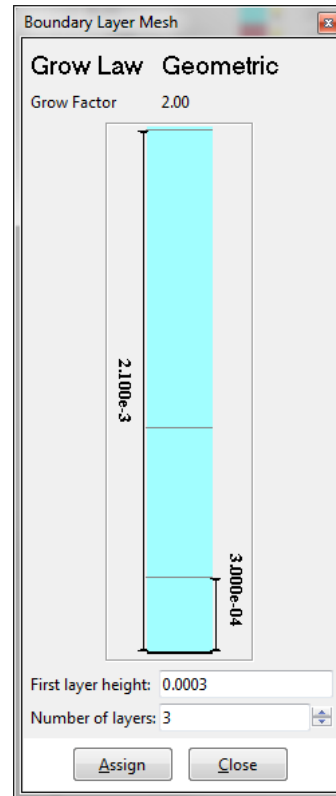
### 2.4.1 Create 3D boundary layer mesh

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 (in layer 'mirror').

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:

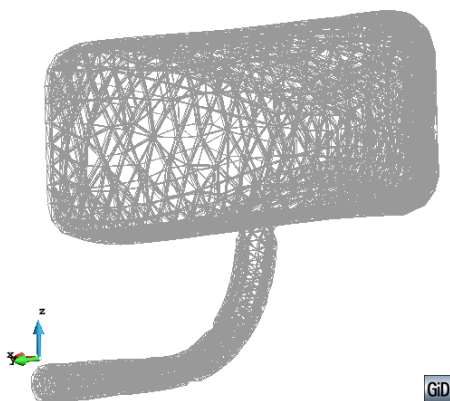
- Fill the parameters with the values showed in the figure 0.0003 as first layer height, and Number of layers==3. Press 'Assign' button, and select the surfaces in layer 'mirror'. Then click ESCAPE and close the Boundary Layer Mesh window.

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 .

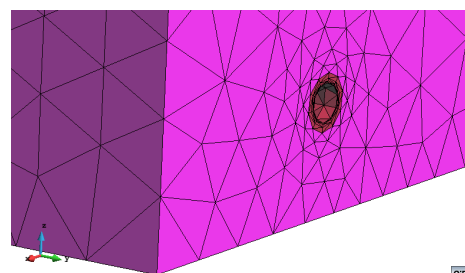


View of the Boundary Layer Mesh window.

- Generate the mesh with general size equal to 0.175, and using the meshing parameters from the model (check the corresponding option 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.





## 3 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 has been chosen which 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.

### 3.1 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

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

where  $\rho(x,y)$  is the density of the material at point  $(x,y)$  and  $S$  is the surface of the body;  $m_i$  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.

#### 3.1.1 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.

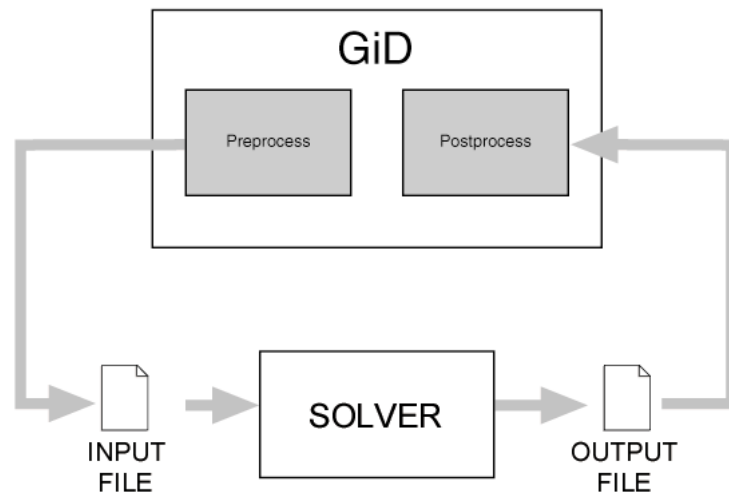


Diagram of the workflow

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.

### 3.1.2 Basic or Advanced

#### Basic Integration

Every GiD user can develop a customized preprocess module within a short time, even without any knowledge in programming languages.

Only a couple of text files should be written describing the user's problem properties (like conditions or materials) using an easy keyword system and GiD will create automatically the corresponding windows, with this keyword system which is described in the GiD documentation, all the information required for a particular problem type can be specified:

- Conditions parameters and dependencies
- Material properties
- General data and interval data
- Symbols to draw conditions nicely
- Definition of used unit system
- Other configuration parameters (version, icons, password)
- Format of the analysis file needed by the simulation executable
- Batch file to launch the calculation and, eventually, previous necessary operating GiD will create automatically windows like the shown ones allowing the end user to manage the data of the problem, assign or modify conditions, draw properties over model, etc. in a really easy way.

After the definition of the problem, GiD will write all the conditions, materials and mesh information using the previously specified format.

### Advanced Integration, CompassLIB toolkit

This toolkit is developed by Compass IS ([www.compassis.com](http://www.compassis.com)), and offers an alternative to the classical process of simulations integration in GiD.

The description of the properties and data needed by the analysis problem is performed using a XML tree (graphical part) and a Tcl file which access all the nodal and elemental information during the writing process of the analysis file. Due to the complexity of these file, a minimum of knowledge in programming languages is required.

Tcl is the scripting language used in GiD. On the other hand, this method offers a lot more possibilities, including an appealing view of the managed data, not only to the developer but also to the final user. With this toolkit, the problem and group data are always displayed during preprocess, on the left side of the graphical window.

This facilities the management of the analysis properties to the end users.

Groups are defined by their mesh elements or geometry entities, and they can be created and edited in an easy way with help of a groups edition window.

### 3.2 Basic integration, plain text

Using the Basic Problemtypes scheme

GiD configuration is accomplished through text formatted files. The following files are required:

**.prb:** configuration of the general parameter (not associated to entities)

**.mat:** configuration of materials and their properties

**.cnd:** configuration of the conditions imposed on the calculation

**.bas:** (template file) the file for configuring the format of the interchange file that mediates between GiD data and the calculating module. The file for interchanging the data exported by GiD has the extension .dat. This file stores the geometric and physical data of the problem.

**.bat:** the file that can be executed called from GiD. This file initiates the calculating module.

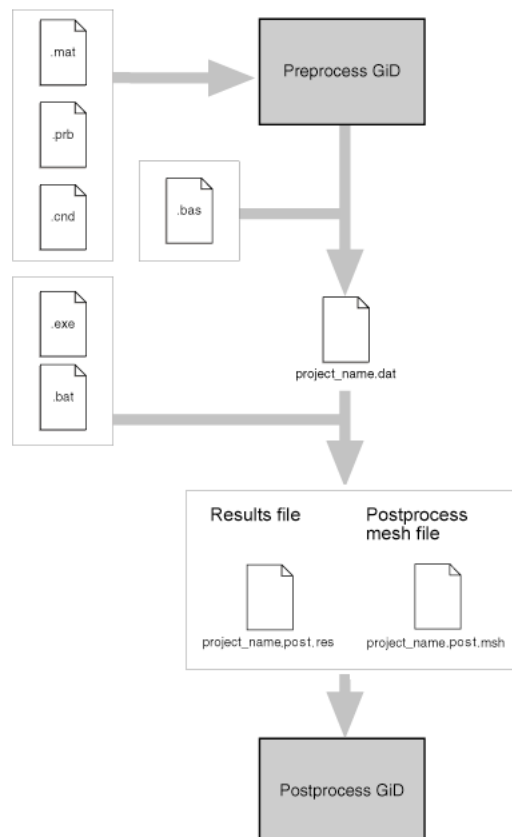


Diagram depicting the files system



### Creating the Subdirectory for the Problem Type

Create the subdirectory "cmas2d.gid". This subdirectory has a .gid extension and will contain all the configuration files and calculating module files (.prb, .mat, .cnd, .bas, .bat, .exe).

**NOTE:** If you want the problem type to appear in the GiD Data->Problem type menu, create the subdirectory within "problemtypes", located in the GiD folder □ for instance, C:\GiD\Problemtypes\cmas2d.gid

### 3.2.1 Creating the General File

Create the "cmas2d.prb" file. This file contains general information for the calculating module, such as the type of resolution algorithm chosen.

Enter the parameters of the general conditions in "cmas2d.prb" using the following format:

PROBLEM DATA

QUESTION: Name of the parameter. If the name is followed by the #CB# instruction, the parameter is displayed as a combo box. The options in the menu must then be entered between parentheses and separated by commas.

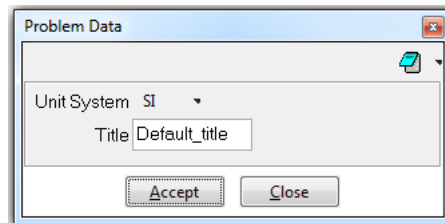
For example, Unit\_System#CB#(SI,CGS,User).

VALUE: The default value of the parameter.

...

END PROBLEM DATA

In GiD, the information in the "cmas2d.prb" file is managed in the problem data window, which is located in **Data**□**Problem Data**.



The GiD Problem Data window, for configuring of the general conditions of the cmas2d module

```
PROBLEM DATA
QUESTION: Unit_System#CB#(SI,CGS,User)
VALUE: SI
QUESTION: Title
VALUE: Default_title
END PROBLEM DATA
```

### 3.2.2 Creating the Materials File

Create the materials file "cmas2d.mat". This file stores the physical properties of the material under study for the problem type. In this case, defining the density will be enough.

Enter the materials in the "cmas2d.mat" file using the following format:

MATERIAL: Name of the material (without spaces)

QUESTION: Property of the material. For this example, we are interested in the density of the material.

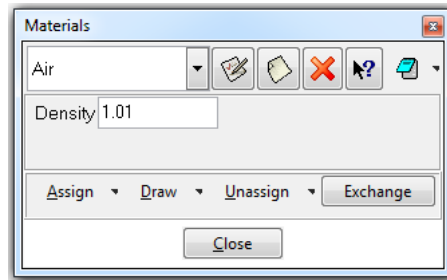
VALUE: Value of the property

HELP: A help text (optional field)

...

END MATERIAL

In GiD, the information in "cmas2d.mat" file is managed in the materials window, located in **Data->Materials**.



The GiD Materials window, for assigning materials

MATERIAL: Air  
 QUESTION: Density  
 VALUE: 1.01  
 HELP: material density  
 END MATERIAL

MATERIAL: Steel  
 QUESTION: Density  
 VALUE: 7850  
 HELP: material density  
 END MATERIAL

...

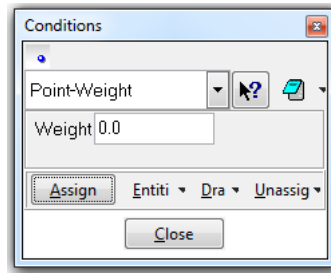
### 3.2.3 Creating the Conditions File

Create the "cmas2d.cnd" file, which specifies the boundary and/or load conditions of the problem type in question. In the present case, this file is where the concentrated weights on specific points of the geometry are indicated.

Enter the boundary conditions using the following format:

CONDITION: Name of the condition  
 CONDTYPE: Type of entity which the condition is to be applied to. This includes the parameters "over points", "over lines", "over surfaces", "over volumes" or "over layers". In this example the condition is applied "over points".  
 CONDMESHTYPE: Type of entity of the mesh where the condition is to be applied. The possible parameters are "over nodes", "over body elements" or "over face elements". In this example, the condition is applied on nodes.  
 QUESTION: Name of the parameter of the condition  
 VALUE: Default value of the parameter  
 ...  
 END CONDITION  
 ...

In GiD, the information in the "cmas2d.cnd" file is managed in the conditions window, which is found in **Data □ Conditions**.



The GiD Conditions window, for assigning the cmas2d boundary and load conditions

```
CONDITION: Point-Weight
CONDTYPE: over points
CONDMESHTYPE: over nodes
QUESTION: Weight
VALUE: 0.0
HELP: Concentrated mass
END CONDITION
```

### 3.2.4 Creating the Data Format File (Template file)

Create the "cmas2d.bas" file. This file will define the format of the .dat text file created by GiD. It will store the geometric and physical data of the problem. The .dat file will be the input to the calculating module.

**NOTE:** It is not necessary to have all the information registered in only one .bas file. Each .bas file has a corresponding .dat file.

Write the "cmas2d.bas" file as follows:

The format of the .bas file is based on commands. Text not preceded by an asterisk is reproduced exactly the same in the .dat file created by GiD. A text preceded by an asterisk is interpreted as a command.

Example:

<p><b>.bas file</b></p> <pre>%%% Problem Size %%%      -&gt; Number of Elements &amp; Nodes: *nelem *npoin</pre>	<p><b>.dat file</b></p> <pre>%%% Problem Size %%% Number of Elements &amp; Nodes: 5379 4678</pre>
--	---

The contents of the "cmas2d.bas" file must be the following:

```
.bas file
=====
General Data File
=====
Title: *GenData(Title)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Number of Elements & Nodes:
*nelem *npoin
```

In this first part of "cmas2d.bas" file, general information on the project is obtained.

**\*nelem:** returns the total number of elements of the mesh.

**\*npoin:** returns the total number of nodes of the mesh.

```

Coordinates:
Node X Y
*loop nodes
*format "%5i%14.5e%14.5e"
*NodesNum *NodesCoord(1,real) *NodesCoord(2,real)
*end nodes

```

This command provides a rundown of all the nodes of the mesh, listing their identifiers and coordinates.

**\*loop, \*end:** commands used to indicate the beginning and the end of the loop. The command **\*loop** receives a parameter.

- \*loop nodes:** the loop iterates on nodes
- \*loop elems:** the loop iterates on elements
- \*loop materials:** the loop iterates on assigned materials

**\*format:** the command to define the printing format. This command must be followed by a numerical format expressed in C syntax.

**\*NodesNum:** returns the identifier of the present node

**\*NodesCoord:** returns the coordinates of the present node

**\*NodesCoord (n, real):** returns the x, y or z coordinate in terms of the value n:

- n=1** returns the **x** coordinate
- n=2** returns the **y** coordinate
- n=3** returns the **z** coordinate

```

Connectivities:
Element Node(1) Node(2) Node(3) Material
*set elems(all)
*loop elems
*format "%10i%10i%10i%10i%10i"
*ElemsNum *ElemsConec *ElemsMat
*end elems

```

This provides a rundown of all the elements of the mesh and a list of their identifiers, the nodes that form them, and their assigned material.

**\*set elems(all):** the command to include all element types of the mesh when making the loop.

**\*ElemsNum:** returns the identifier of the present element

**\*ElemsConec:** returns the nodes of an element in a counterclockwise order

**\*ElemsMat:** returns the number of the assigned material of the present element

```

Begin Materials
N° Materials= *nmats

```

This gives the total number of materials in the project

**\*nmats:** returns the total number of materials

```

Mat. Density

```

```

*loop materials
*format "%4i%13.5e"
*set var PROP1(real)=Operation(MatProp(Density, real))
*MatNum *PROP1
*end

```

This provides a rundown of all the materials in the project and a list of the identifiers and densities for

each one.

**\*MatProp (density, real):** returns the value of the property "density" of the material in a "real" format.

**\*Operation (expression):** returns the result of an arithmetic expression. This operation must be expressed in C.

**\*Set var PROP1(real)=Operation(MatProp(Density, real)):** assigns the value returned by MatProp (which is the value of the density of the material) to the variable PROP1 (a "real" variable).

**\*PROP1:** returns the value of the variable PROP1.

**\*MatNum:** returns the identifier of the present material.

Point conditions

```
*Set Cond Point-Weight *nodes
*set var NFIX(int)=CondNumEntities(int)
Concentrate Weights
*NFIX
```

This provides the number of entities with a particular condition.

**\*Set Cond Point-Weight \*nodes:** this command enables you to select the condition to work with from that moment on. For the present example, select the condition "Point-Weight".

**\*CondNumEntities(int):** returns the number of entities with a certain condition.

**\*Set var NFIX(int)= CondNumEntities(int):** assigns the value returned by the command CondNumEntities to the NFIX variable (an "int" variable).

**\*NFIX:** returns the value of the NFIX variable.

Potentials Prescrits:

```
Node Tipus
Valor/Etiqueta
*loop nodes *OnlyInCond
*NodesNum *cond(1)
*end
```

This provides a rundown of all the nodes with the condition "Point-Weight" with a list of their identifiers and the first "weight" field of the condition in each case.

**\*loop nodes \*OnlyInCond:** executes a loop that will provide a rundown of only the nodes with this condition.

**\*cond(1):** returns the number 1 field of a condition previously selected with the \*set cond command. The field of the condition may also be selected using the name of the condition, for example cond(weight).

cmas2d.bas

```
=====
General Data File
=====
```

%%%%%%%% Problem Size %%%%%%%%%%

Number of Elements & Nodes:

```
*nelem *npoin
```

```
%%%%%%%%% Mesh Database %%%%%%%%%%
```

```
Coordinates:
```

```
Node X Y
```

```
*set elems(all)
```

```
*loop nodes
```

```
*format "%5i%14.5e%14.5e"
```

```
*NodesNum *NodesCoord(1,real) *NodesCoord(2,real)
```

```
*end nodes
```

```
.....
```

```
Connectivities:
```

```
Element Node(1) Node(2) Node(3) Material
```

```
*loop elems
```

```
*format "%10i%10i%10i%10i%10i"
```

```
*ElemsNum *ElemsConec *ElemsMat
```

```
*end elems
```

```
.....
```

```
Begin Materials
```

```
Nº Materials= *nmats
```

```
Mat. Density
```

```
.....
```

```
*loop materials
```

```
*format "%4i%13.5e"
```

```
*set var PROP1(real)=Operation(MatProp(Density, real))
```

```
*MatNum *PROP1
```

```
*end
```

```
.....
```

```
Point conditions
```

```
*Set Cond Point-Weight *nodes
```

```
*set var NFIX(int)=CondNumEntities(int)
```

```
Concentrated Weights
```

```
*NFIX
```

```
.....
```

```
Potentials Prescrits:
```

```
Node Tipus
```

```
Valor/Etiqueta
```

```
*Set Cond Point-Weight *nodes
```

```
*loop nodes *OnlyInCond
```

```
*NodesNum *cond(1)
```

```
*end
```

```
.....
```

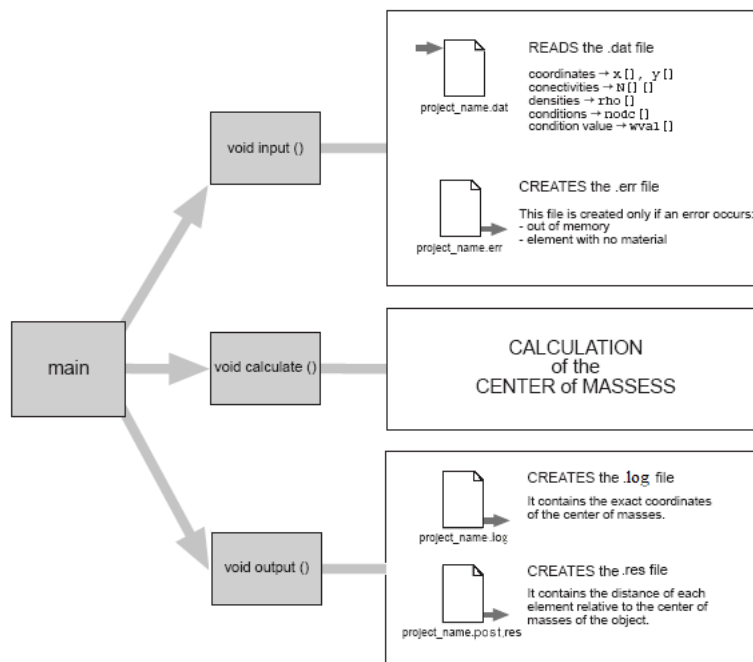
### 3.2.5 Creating the Execution file of the Calculating Module

Create the file "cmas2d.c". This file contains the code for the execution program of the calculating module. This execution program reads the problem data provided by GiD, calculates the coordinates of the center of mass of the object and the distance between each element and this point. These

results are saved in a text file with the extension .post.res.

Compile and link the "cmas2d.c" file in order to obtain the executable cmas2d.exe file.

The calculating module (cmas2d.exe) reads and generates the files described below.



cmas2d.c solver structure

**NOTE:** details of this file at [Additional information -pag. 53-](#).

### 3.2.6 Creating the Execution File for the Problem Type

Create the `cmas2d.win.bat` file. This file connects the data file(s) (.dat) to the calculating module (the `cmas2d.exe` program). When the GiD Calculate option is selected, it executes the .bat file for the problem type selected.

When GiD executes the .bat file, it transfers three parameters in the following way:

(parameter 3) / \*.bat (parameter 2) / (parameter 1)

parameter 1: project name

parameter 2: project directory

parameter 3: Problem type location directory

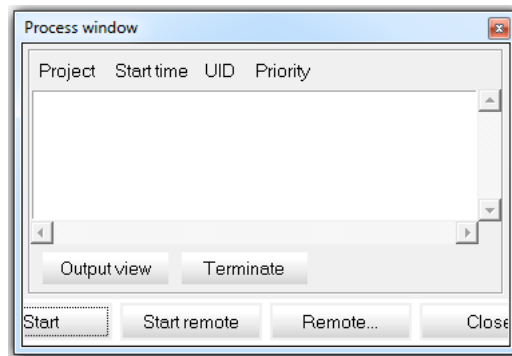


**NOTE:** The .win.bat file as used in Windows is explained below; the shell script for UNIX systems is also included with the documentation of this tutorial.

```
rem OutputFile: %2\%1.log
```

A comment line such as "rem OutputFile: file\_name.log" means that the contents of the file indicated will be shown if the user clicks Output View in **Calculate->Calculate window**.

In this example the .log file is shown. This file contains the coordinates of the center of mass.



The Process window.

```
rem ErrorFile: %2\%1.err
```

A comment line such as "rem ErrorFile: file\_name.err" means that the indicated file will contain the errors (if any). If the .err file is present at the end of the execution, a window comes up showing the error. The absence of the .err file indicates that the calculation is considered satisfactory. GiD automatically deletes the .err files before initiating a calculation to avoid confusion.

```
del %2\%1.log
del %2\%1.post.res
```

This deletes results files from any previous calculations to avoid confusion.

```
%3\cmas2d.exe %2\%1
```

This executing the cmass2d.exe and provide the .dat as input file file.

### 3.3 Advanced integration, xml complex fields

The Advanced integration using CompassLIB toolkit is based on modifying a xml file.

In order to add **conditions**, **general data**, or **units** information to the *problemtyp*, it is necessary to modify file **{PROBLEMTYPE}\_default.spd**. This is a file in XML format. This file contains all the definition of all the data necessary for the analysis.

#### The new Problem Type structure

File extension	Description	New Problem Type
name.prb	Problem and intervals data	Not used
name.cnd	Conditions definition	Used, should not be modified
name.mat	Material properties	Not used
name.bas	Information for data input file	Not used, should be void
name.tcl	Main TCL file, initialization	Used
name_default.spd	Main configuration file, XML-based	Used
scripts/writecalfile.tcl	Output description to the file for analysis	Used

Table 1: Main files that configure the new Problem Types using the Toolkit

We are not going to detail the creation of this file, but for example our conditions that was some lines inside a .cnd file:

```
CONDITION: Point-Weight
CONDTYPE: over points
```



```

CONDMESHTYPE: over nodes
QUESTION: Weight
VALUE: 0.0
HELP: Concentrated mass
END CONDITION

```

Become some lines inside the spd like:

```

<condition n="Point_Weight" pn="Point Weight" ov="point" ovm="node"
icon="constraints16" 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>

```

You can load the the problemtype by selecting the menu:

**Data->Problemtype->Examples->cmas2d\_CompassLIB**

and take a look at the code by accesing to folder: (GiD folder)\problemtypes\Examples\cmas2d\_CompassLIB.gid\

Full documentation of the parameters can be found at:

<http://www.compassis.com/downloads/Manuals/CompassLIBManual.pdf>

### 3.4 Extensions, tcl programing

Here is when some programing level in tcl is needed, when you are doing a problem type you could include the file problem\_type\_name.tcl in our case cmas2d.tcl

It is possible to modify behaibour or anything you want from GiD by modifiing this file. You have full control of GiD.

The structure of problem\_type\_name.tcl can optionally implement some event procedures that are automatically called by GiD(also can define other user-defined procedures). Their syntax corresponds to standard Tcl/Tk language.

To see all events available, go to menu: **Help->Customization**, this will open GiD customization manual, from the tree select: **tcl/tk extensions -> events procedures**

If you open the code of "(GiD folder)\problemtypes\Examples\cmas2d.gid\cmas2d.tcl" you will see an examples of how this functions are implemented. In our case we will take a look to the event:

- **InitGIDProject:** will be called when the problem type is selected. It receives the dir argument, which is the absolute path to the problem\_type\_name.gid directory, which can be useful inside the routine to locate some alternative files.

Here our procedure implemented:

```

proc InitGIDProject {dir} {
  set materials [GiD_Info materials]
  set conditions [GiD_Info conditions ovpnt]
  CreateWindow $dir $materials $conditions
}

```

GiD\_Info are information function, in this case return material and conditions information, a list of all GiD\_Info can be found in GiD customization manual, at leaf: **tcl/tk extensions -> GiD\_Info functions**

CreateWindow is a user-defined procedure created by the problemtype (you can see the code in the same file cmas2d.tcl).

In this case define and open a window with 2 buttons, almost all code is pure tcl/tk.

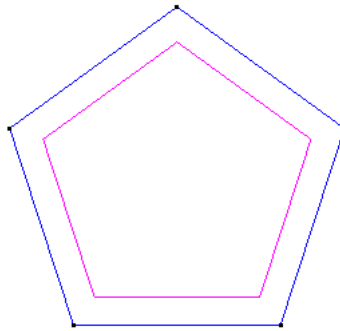
To see the result of this implementation, select the menu **Data->Problem type->Examples->cmas2d** and take a look at the windows that pops-up.

### 3.5 Using the problemtype with an example

In order to understand the way the calculating module works, simple problems with limited practical use have been chosen. Although these problems do not exemplify the full potential of the GiD program, the user may intuit their answers and, therefore, compare the predicted results with those obtained in the simulations.

Create a surface, for example from the menu **Geometry->Create->Object->Polygon**

Create a polygon with 5 sides, centered in the (0,0,0) and located in the XY plane (normal = 0,0,1) and whit radius=1.0

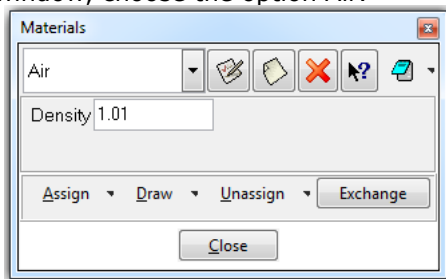


Surface used for this example

#### With Basic Integration

Load the problemtype: menu **Data->Problem type->Examples->cmas2d**.

Choose **Data->Materials**. The materials window is opened. From the Materials menu in this window, choose the option Air.



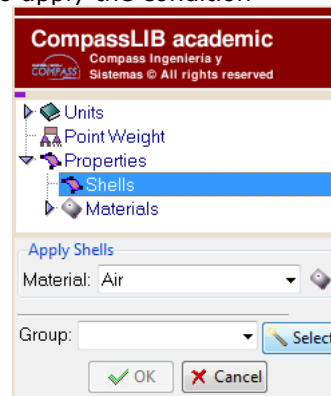
Material window

Click **Assign->Surfaces** and select the surface. Press ESC when this step is finished.

#### With Advanced Integration

Load the problemtype: menu **Data->Problem type->Examples->cmas2d\_CompassLIB**.

Choose **Data->Data(Internal)**. The tree will be displayed, go to Properties->Shells Doble click on Shells and a bellow frame will be opened to apply the condition



Apply Shell window

From the Material menu, choose the option Air. Click **Select** and select the surface. Press **End** when this step is finished.

A Group containing the surface selected will be created automatically, you can always acces to this group and change the condition applied

Choose the **Mesh->Generate** option.

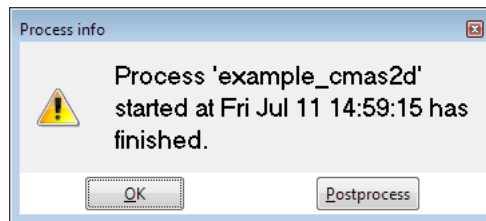
A window appears in which to enter the maximum element size for the mesh to be generated. Accept

the default value and click OK. The mesh will be obtained.

Now the calculation may be initiated, but first the model must be saved (**Files->Save**), use 'example\_cmas2d' as name for the model.

Choose the Calculate option from the Calculate menu to start the calculation module.

Wait until a box appears indicating the calculation has finished.



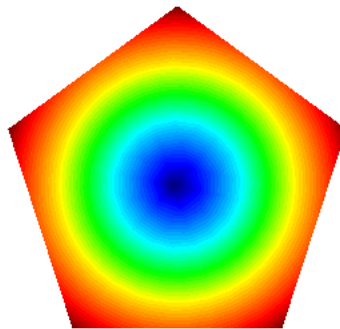
Process information box

Choose the option **Postprocess** or close the window and select from menu **Files->Postprocess**.

By default when changing to postprocesses mode no results is visualized.

Select: **View results->Contour Fill->MC-DISTANCE**.

A graphic representation of the calculation is obtained.



Visualizing the distance (MC-DISTANCE) from the center of mass of the object to each element, for an object of homogeneous material

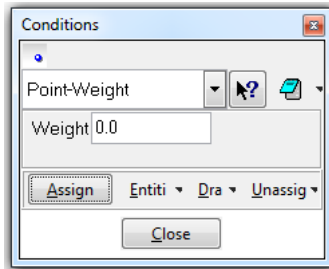
The results shown on the screen reproduce those we anticipated at the outset of the problem: the center of mass of an object made of homogeneous material coincides with its geometric center. The .log file will provide the exact coordinates of this point.

### 3.5.1 Executing the calculation with a concentrated weight

Executing the calculation for an object of heterogeneous material and subject to external point-weight. Choose the **Files --> preprocess** option (to go back to preprocess).

**With Basic Integration**

Choose the **Data->Conditions** option. A window is opened in which the conditions of the problem should be entered. Since the condition to be entered acts over points, select over points from the Type menu in the Conditions window.



The Condition Window

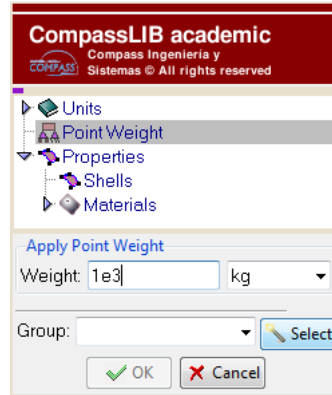
Enter the value 1e3 in the Weight box. Click Assign and select the upper corner point. Press ESC when this step is finished.

Choose **Mesh->Generate**.

A window appears in which to enter the element size for the mesh to be generated. click OK. Choose the Calculate option from the Calculate menu, thus executing the calculating module. Choose the **Files->Postprocess** option. Visualize the new results.

**With Advanced Integration**

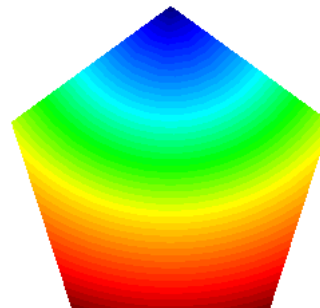
If you don't have the tree view available, choose the **Data->Data(internal)** option. From The tree double click on Point Weight.



Apply Point Weight window

Enter the value 1e3 in the Weight box. Click Select and select the upper corner point. Press End when this step is finished.

A Group containing the point selected will be created automatically, you can always access to this group and change the condition applied.



Visualization of the distance from the mass center to each element, for an object of heterogeneous material subject to point weight

Now the condition is external point-weight. As anticipated, the new center of mass is displaced toward the point under weight.

**3.6 Additional information**

**NOTE:** In this example, a code for the program will be developed in C. Nevertheless, any programming language may be used.

The code of the program that calculates the center of mass (cmas2d.c) is as follows:

The cmas2d.c file

```

#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>

#define MAXMAT 1000
#define MAXCND 1000

char projname[1024];
int i, ielem, inod, icnd;
double *x, *y;
int *N, *imat;
int nodc[MAXCND];
double rho[MAXMAT], wval[MAXCND];
int Nelem, Nnod, Nmat, Ncnd;
double x_CG, y_CG;

void input(void);
void calculate(void);
void output(void);

```

Declaration of variables and constants used in the program.

```

void main (int argc, char *argv[]) {
    strcpy (projname, argv[1]);
    input();
    calculate();
    output();
}

```

### 3.6.1 The main program

The main program is called from the `cmas2d.win.bat` file and has as parameter the name of the project. This name is stored in the variable `projname`.

The main program calls the `input()`, `calculate()` and `output()` functions.

The input function reads the `.dat` file generated by GiD. The `.dat` file contains information about the mesh. The calculate function read and processes the data and generates the results. The output function creates the results file.

```

void input () {
    char filename[1024], fileerr[1024], sau1[1024], sau2[1024];
    FILE *fp, *ferr;
    int aux,j, error=0;
    void jumpline (FILE*);
    strcpy(filename, projname);
    strcat(filename, ".dat");
    fp=fopen(filename, "r");

```

The first part of the input function links the project name with the `.dat` extension, thus obtaining the name of the file that is to be read. This file is opened in order to be read.

The `jumpline(FILE*)` function is declared. This function simply reads a line from the file that it receives as a parameter, It is used to jump lines of the text when reading the `.dat` file.

```

for (i=0; i<6; i++) jumpline (fp);
fscanf(fp, "%d %d", &Nelem, &Nnod);

```

The first six lines of the `.dat` file are jumped over since these are lines of information for the user (see `.bas` file). Then the total number of elements and nodes of the project are read and stored in the variables `Nelem` and `Nnod` respectively.

```

x=(double *) malloc((Nnod+1)*sizeof(double)); if (x==NULL) {error=1;}
y=(double *) malloc((Nnod+1)*sizeof(double)); if (y==NULL) {error=1;}
N= (int *) malloc((Nelem+1)*3*sizeof(int)); if (N==NULL) {error=1;}
imat=(int *) malloc((Nelem+1)*sizeof(int)); if (N==NULL) {error=1;}
if (error) {
    strcpy(fileerr, projname);
    strcat(fileerr, ".err");
    ferr = fopen(fileerr, "w");
    fprintf(ferr, "***** ERROR: Not enough memory. ***** ");
    fprintf(ferr, "(Try to calculate with less elements) ");
    fclose(ferr);
    exit(1);
}
for (i=0; i<6; i++) jumpline (fp);

```

Space is reserved for storing the coordinates of the nodes (pointers x, y), the connectivities (pointer N), and the materials corresponding to each element (pointer imat).

In case of error (insufficient memory), a file is created with the extension .err. This file contains information about the error and the program is aborted.

The next six lines are jumped over.

```

/* reading the coordinates */
for (inod=1; inod<=Nnod; inod++){
    fscanf (fp, "%d %lf %lf", &aux, &x[inod], &y[inod]);
for (i=0; i<6; i++) jumpline (fp);

```

The coordinates of the nodes are read and stored in the x and y variables. The node identifier indexes the tables of coordinates.

```

/* reading connectivities */
for (ielem=1; ielem<=Nelem; ielem++){
    fscanf (fp, "%d", &aux);
    for(j=0;j<3;j++) fscanf (fp, "%d", &N[(ielem-1)*3+j]);
    fscanf (fp, "%d", &imat[ielem]);
    if (imat[ielem]==0){
        strcpy(fileerr, projname);
        strcat(fileerr, ".err");
        ferr = fopen(fileerr, "w");
        fprintf(ferr, "***ERROR: Elements with no material!!! ");
        fclose(ferr);
        exit(1);
    }
}
}

```

The connectivities are read and the N variable is saved. This variable is a Nelem x 3- size table with two fields. The nodes (assumed triangles of 3 nodes) forming the element are saved in the first field. The element identifiers are saved in the second one.

All the elements are checked, ensuring that they have been assigned a material. If the identifier of the material is 0 (meaning that no material has been assigned to the element), an .err file is created containing information about the error and the program is aborted.

```

for (i=0; i<5; i++) jumpline (fp);
fscanf(fp, "%s %s %d", saul, sau2, &Nmat );
for (i=0; i<3; i++) jumpline (fp);
/* reading density of each material */
for (i=1; i<=Nmat; i++) {
    fscanf (fp, "%d %lf", &aux, &rho[i]);
    /* reading conditions*/
    for (i=0; i<4; i++) jumpline (fp);
    fscanf(fp, "%d", &Ncnd);
    for (i=0; i<6; i++) jumpline (fp);
}

```

```

    for (icnd=1; icnd<=Ncnd; icnd++) {
        fscanf (fp, "%d %lf", &nodc[icnd], &wval[icnd]);
        jumpline (fp);
    }
    fclose (fp);
}

```

Reading the remaining information in the .dat file.

The total number of materials is read and stored in the Nmat variable.

The density of each material are read and stored in the rho table. The material identifier indexes the densities.

The total number of conditions is read and stored in the Ncnd variable.

The nodes associated with a condition are read and stored in the nodc table indexed by the condition identifier. The value of the condition is stored in wval, another table indexed by the condition identifier.

```

void calculate ()
{
    double v,aux1,aux2,aux3;
    int n1, n2, n3;
    int mat;
    double x_CGi, y_CGi;
    double x_num=0, y_num=0, den=0;

```

This is the function that calculates the center of mass.

Declaration of the local variables used in calculate().

```

for(ielem=1;ielem<=Nelem;ielem++) {
    n1= N[0+(ielem-1)*3];
    n2= N[1+(ielem-1)*3];
    n3= N[2+(ielem-1)*3];
    /* Calculating the volume (volume is the area for surfaces) */
    v = fabs( x[n1]*y[n2]+x[n2]*y[n3]+x[n3]*y[n1]
             -x[n1]*y[n3]-x[n2]*y[n1]-x[n3]*y[n2] ) / 2;
    x_CGi = (x[n1]+x[n2]+x[n3])/3;
    y_CGi = (y[n1]+y[n2]+y[n3])/3;
    mat = imat[ielem];
    x_num += rho[mat]*v*x_CGi;
    y_num += rho[mat]*v*y_CGi;
    den += rho[mat]*v;
}
/* puntual weights */
for(icnd=1;icnd<=Ncnd;icnd++) {
    inod = nodc[icnd];
    x_num += wval[icnd]*x[inod];
    y_num += wval[icnd]*y[inod];
    den += wval[icnd];
}
x_CG = (x_num/den);
y_CG = (y_num/den);

```

The identifiers of the nodes of the present element are saved in n1, n2, n3.

This loop makes a rundown of all the elements in the mesh. The volume is calculated for each element. (Here, the volume is the area, provided we are dealing with 3D surfaces). The volume calculations are stored in the v variable.

The geometric center of the element is calculated (coinciding with the center of gravity) and the coordinates are stored in the x\_Cgi and y\_Cgi variables.

The numerator sums are calculated. When the loop is finished, the following sums are stored in the x\_num and y\_num variables. Finally, the result of dividing the x\_num and y\_num variables by the

den variable is stored in the `x_CG` and `y_CG` variables.

```
void output() {
char filename[1024];
FILE *fp, *fplog;
double v;
```

The `output()` function creates two files: `.post.res`, and `.log`.

The results to be visualized in GiD Post-process are stored in the `.post.res` file. It is this file that stores the data which enables GiD to represent the distance of each point from the corresponding center of mass.

The numerical value of the center of mass is saved in the `.log` file. The accuracy of this value is directly proportional to the element size.

```
/* writing log information file */
strcpy(filename, projname);
strcat(filename, ".log");
fplog=fopen(filename, "w");
fprintf(fplog, "CMAS2D routine to calculate the mass center ");
fprintf(fplog, "project: %s ", projname);
fprintf(fplog, "mass center: %lf %lf ", x_CG, y_CG);
fclose(fplog);
```

Creating the `.log` file: the `.log` extension is added to the project name and a file is created that will contain the numerical value of the position of the center of mass, which in turn is stored in the `x_CG` and `y_CG` variables of the program.

Creating the `.post.res` file. The output data (results) are stored in this file.

The format of the `.post.res` file is explained in the GiD help, see section *Postprocess data files* --> *Postprocess results format*.

```
/* writing .post.res */
strcpy(filename, projname);
strcat(filename, ".post.res");
fp=fopen(filename, "w");
fprintf(fp, "GiD Post Results File 1.0 ");
fprintf(fp, "Result MC-DISTANCE \"LOAD ANALYSIS\" 1 Scalar OnNodes ");
fprintf(fp, "ComponentNames MC-DISTANCE ");
fprintf(fp, "Values ");
for(inod=1; inod<=Nnod; inod++) {
/* distance or each node to the center of masses */
v=sqrt((x_CG-x[inod])*(x_CG-x[inod])+(y_CG-y[inod])*(y_CG-y[inod]));
fprintf(fp, "%d %lf ", inod, v);
}
fprintf(fp, "End values ");
fclose(fp);
```

In this example only a scalar result , with a single time step, is written in the `.res` file.

This is the full source code of this program:

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>

#define MAXMAT 1000
#define MAXCND 1000
char projname[1024];
int i, ielem, inod, icnd;
double *x, *y;
int *N, *imat;
```



```

int nodc[MAXCND];
double rho[MAXMAT], wval[MAXCND];
int Nelem, Nnod, Nmat, Ncnd;
double x_CG, y_CG;

void input(void);
void calculate(void);
void output(void);

void main (int argc, char *argv[]) {
    strcpy (projname, argv[1]);
    input();
    calculate();
    output();
}

void input () {
    char filename[1024], fileerr[1024], saul[1024], sau2[1024] ;
    FILE *fp, *ferr;
    int aux,j, error=0;
    void jumpline (FILE*);
    strcpy(filename, projname);
    strcat(filename, ".dat");
    fp=fopen(filename, "r");
    for (i=0; i<6; i++) jumpline (fp);
    fscanf(fp, "%d %d", &Nelem, &Nnod);
    x=(double *) malloc((Nnod+1)*sizeof(double)) ; if (x==NULL) {error=1;}
    y=(double *) malloc((Nnod+1)*sizeof(double)) ; if (y==NULL) {error=1;}
    N= (int *) malloc((Nelem+1)*3*sizeof(int)) ; if (N==NULL) {error=1;}
    imat=(int *) malloc((Nelem+1)*sizeof(int)); if (N==NULL) {error=1;}
    if (error) {
        strcpy(fileerr, projname);
        strcat(fileerr, ".err");
        ferr = fopen(fileerr, "w");
        fprintf(ferr, "***** ERROR: Not enough memory. ***** ") ;
        fprintf(ferr, "(Try to calculate with less elements) ") ;
        fclose(ferr);
        exit(1);
    }
    for (i=0; i<6; i++) jumpline (fp);
    /* reading the coordinates */
    for (inod=1; inod<=Nnod; inod++)
        fscanf (fp, "%d %lf %lf", &aux, &x[inod], &y[inod]) ;
    for (i=0; i<6; i++) jumpline (fp);
    /* reading connectivities */
    for (ielem=1; ielem<=Nelem; ielem++){
        fscanf (fp, "%d", &aux) ;
        for(j=0 ;j<3;j++) fscanf (fp, "%d", &N[(ielem-1)*3+j]);
        fscanf (fp, "%d", &imat[ielem]) ;
        if (imat[ielem]==0){
            strcpy(fileerr, projname) ;
            strcat(fileerr, ".err") ;
            ferr = fopen(fileerr, "w") ;
            fprintf(ferr, "***ERROR: Elements with no material!!** ") ;
            fclose(ferr) ;
            exit(1);
        }
    }
    for (i=0; i<5; i++) jumpline (fp);
    fscanf(fp, "%s %s %d", saul, sau2, &Nmat ) ;
    for (i=0 ; i<3; i++) jumpline (fp);
    /* reading density of each material */
    for (i=1; i<=Nmat; i++)
        fscanf (fp, "%d %lf", &aux, &rho[i]) ;
    /* reading conditions*/
    for (i=0 ; i<4; i++) jumpline (fp);
    fscanf(fp, "%d", &Ncnd) ;
}

```

```

for (i=0 ; i<6; i++) jumpline (fp);
for (icnd=1 ; icnd<=Ncnd; icnd++) {
    fscanf (fp, "%d %lf", &nodc[icnd], &wval[icnd]) ;
    jumpline (fp) ;
}
fclose (fp) ;
}

void calculate () {
double v ;
int n1, n2, n3 ;
int mat ;
double x_CGi, y_CGi ;
double x_num=0, y_num=0, den=0 ;
for(ielem=1 ;ielem<=Nelem;ielem++) {
    n1= N[0+(ielem-1)*3] ;
    n2= N[1+(ielem-1)*3] ;
    n3= N[2+(ielem-1)*3] ;
    /* Calculating the volume (volume is the area for surfaces) */
    v=fabs(x[n1]*y[n2]+x[n2]*y[n3]+x[n3]*y[n1]-x[n1]*y[n3]-x[n2]*y[n1]-x[n3]*y[n2])/2
;

    x_CGi= (x[n1]+x[n2]+x[n3])/3 ;
    y_CGi= (y[n1]+y[n2]+y[n3])/3 ;
    mat= imat[ielem];
    x_num+= rho[mat]*v*x_CGi;
    y_num+= rho[mat]*v*y_CGi;
    den+= rho[mat]*v;
}
/* puntual weights */
for(icnd=1 ;icnd<=Ncnd;icnd++) {
    inod= nodc[icnd] ;
    x_num+= wval[icnd]*x[inod] ;
    y_num+= wval[icnd]*y[inod] ;
    den+= wval[icnd] ;
}
x_CG= (x_num/den) ;
y_CG= (y_num/den) ;
}

void output() {
char filename[1024] ;
FILE *fp, *fplog ;
double v ;
/* writing log information file */
strcpy(filename, projname) ;
strcat(filename, ".log") ;
fplog=fopen(filename, "w") ;
fprintf(fplog, "CMAS2D routine to calculate the mass center ") ;
fprintf(fplog, "project: %s ", projname) ;
fprintf(fplog, "mass center: %lf %lf ", x_CG, y_CG);
fclose(fplog) ;
/* writing .post.res */
strcpy(filename,projname) ;
strcat(filename, ".post.res");
fp=fopen(filename, "w") ;
fprintf(fp, "GiD Post Results File 1.0 ") ;
fprintf(fp, "Result MC-DISTANCE \"LOAD ANALYSIS\" 1 Scalar OnNodes ") ;
fprintf(fp, "ComponentNames MC-DISTANCE ") ;
fprintf(fp, "Values ") ;
for(inod=1 ;inod<=Nnod;inod++) {
    /* distance or each node to the center of masses */
    v=sqrt((x_CG-x[inod])*(x_CG-x[inod])+(y_CG-y[inod])*(y_CG-y[inod])) ;
    fprintf(fp, "%d %lf ", inod, v) ;
}
fprintf(fp, "End values ") ;
fclose(fp) ;
free(x) ;
}

```

```
    free(y) ;
    free(N) ;
    free(imat) ;
}

void jumpline (FILE* filep) {
    char buffer[1024] ;
    fgets(buffer,1024,filep);
}
```





## 4 Advanced visualization tools

With this course you will learn how to manage the advanced visualization features presents in GiD.

### 4.1 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.

- 1 . From preprocess mode open the model
- 2 . Switch to postprocess mode
- 3 . In order to get a better view turn off the **VLayer0** and **SLayer8** layers

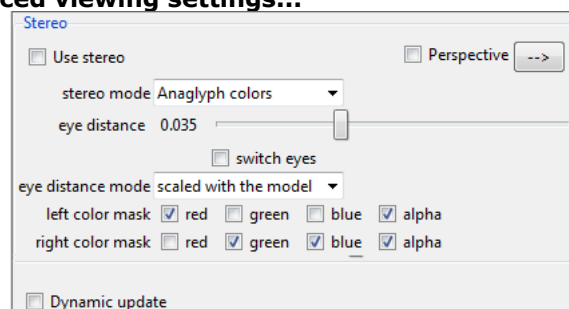
In order to test this option, first we will display a result.

- 4 . Select the 12.5 step through **View results->Default Analysis/Step->RANSOL->12.5** or clicking on 
- 5 . Select **View results->Contour Fill->Pressure (Pa)**

- 1 . Select **Options->Contour->Define Limits...** through the menu bar or clicking on 

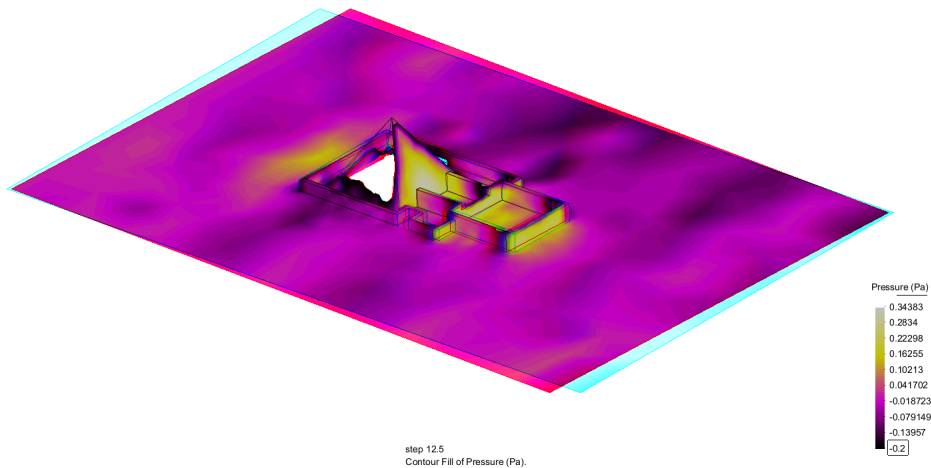
Choosing the first option the Contour Limits window appears. With this window you can set the minimum/maximum value that Contour Fill should use.

- 2 . Check the **Min** checkbox
- 3 . Change the value to **-0.2**
- 4 . Click on the **Apply** button
- 5 . **Close** the window
- 6 . Select **Options->Contour->Color Scale->3D Anaglyphs 2**
- 7 . Select **View->Advanced viewing settings...**



- 8 . Check the **Dynamic update** option in order to change the options without the need to click the Apply button
- 9 . Check the **Use stereo** option
- 10 . Set the eye distance to the value where you can see the 3D effect
- 11 . Uncheck the **Use stereo** option

- 12 . **Close** the window
- 13 . Select **Options->Contour->Reset All**
- 14 . Select **View results->No Results**



## 4.2 Mirror effect

### Model used

The model `test_mirror.gid` will be used in this example, which can be found at [Material location](#).

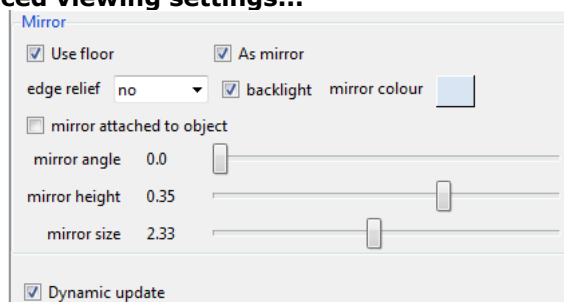
### Menu

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



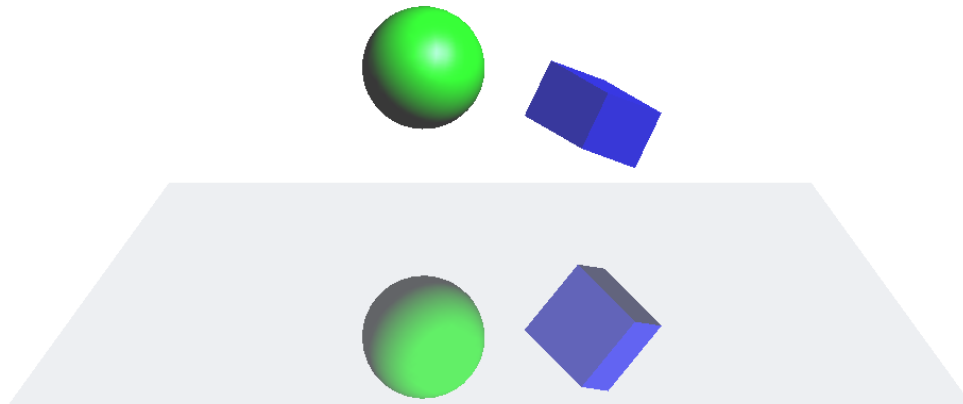
- 4 . Check the **Dynamic update** option
- 5 . Check the **Use Floor** option and a ground surface will appear under the model
- 6 . Check the **As mirror** option in order to get the model reflected in this ground surface
- 7 . Check the **backlight** option to get a better view

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

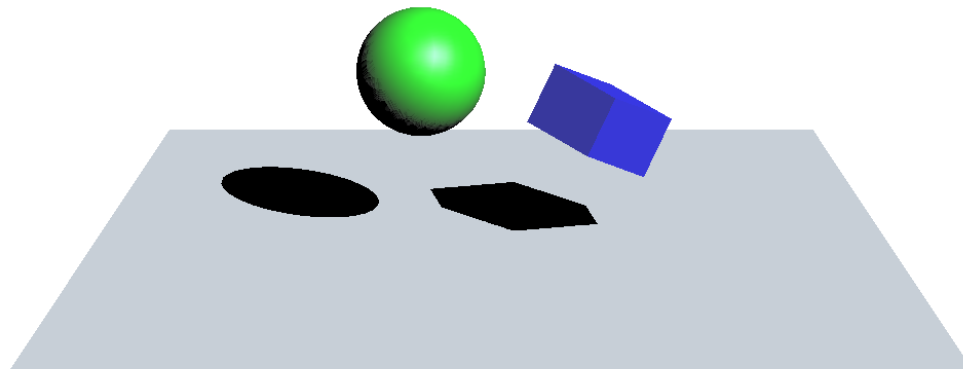
- **Mirror size:** Changes the size of the ground

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



You can also use the floor as ground floor to project the shadow of the model:

- 9 . Uncheck the **As mirror** option  
10 . Check the **Use shadow** option



- 11 . Uncheck the **Use Floor** option  
12 . **Close** the window

### 4.3 Shadows

#### Models used

The models `shadows.post.bin` and `pyramid_s.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 shadow 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.

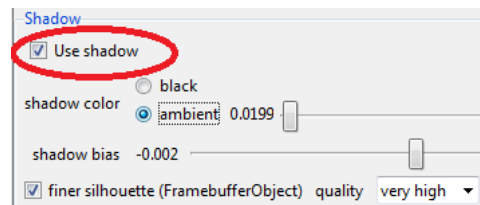
With shadows enabled the scene is rendered at least twice, and sometimes three times, on some hardware and drawing the shadowed area with dimmed ambient light.

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.



Shadow options in the advanced viewing settings window

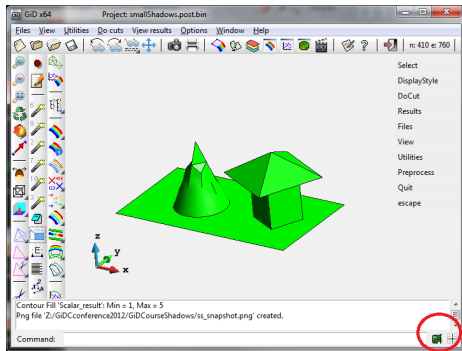
## Options

- **Use shadow** enables or disables the shadows visualization;
- **shadow colour** this option controls if the shadows should be black or should be drawn with a dimmed ambient light.
- **shadow bias** this advanced 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.

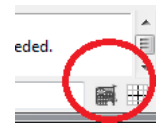
**Note:** *medium* and *high* qualities may work with all kind of graphics cards, even in Linux software mode, safe mode; but check the memory of your graphics card before choosing the *very high* or *highest* qualities for shadows. For these two last qualities, the graphics card should at least have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

## 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:

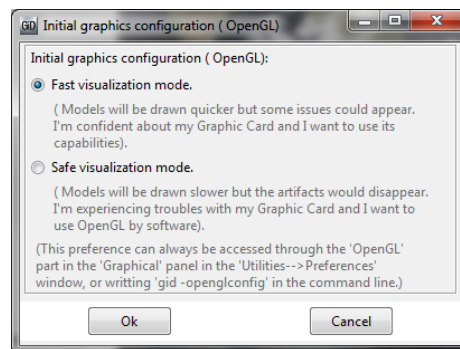


In green, the icon tells that GiD uses the graphics card to accelerate the visualization



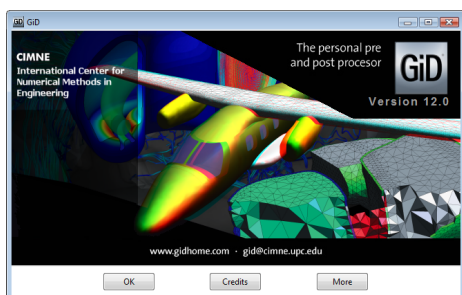
In grey, GiD does not use the graphics card, it renders in software mode instead, aka Safe mode

The graphics configuration can be changed by clicking on this icon or at the *Graphical* panel of the *Preferences window* under *Graphical system / OpenGL options*. By clicking on the icon, following window will appear:

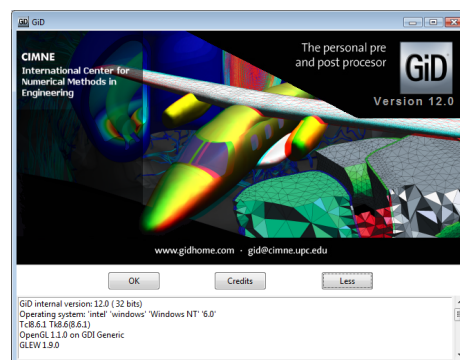


**Window users**

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



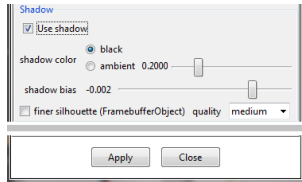
By scrolling down the messages, the OpenGL version and graphics card used can be found

**Linux users**

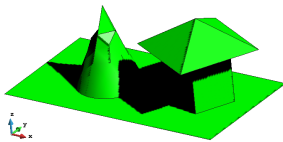
In Linux, the software mode of GiD uses the [[Mesa 3D](#)] library, an open source implementation of the [[OpenGL](#)] specification, which provides at least OpenGL version 2.1.

**Example 1**

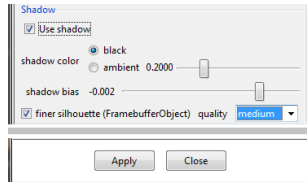
- 1 . Switch to postprocess and load **shadows.post.bin**
- 2 . Select **View->Advanced viewing settings...**
- 3 . Enable **Dynamic update** option to avoid clicking Apply button
- 4 . Enable **Use shadow** option and select **black** as shadow color. When zooming, the border of the shadow shows a *staircase* effect
- 5 . Enable the **finer silhouette** option and select different qualities to view the effect of the shadows border



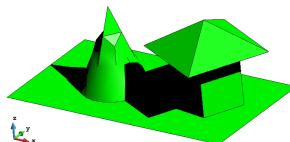
enable coarse black shadows



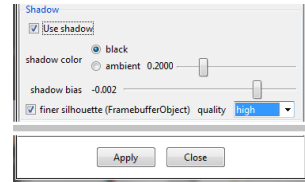
coarse shadows



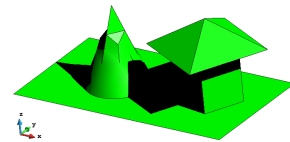
enable finer silhouette with medium quality ( 4 MB)



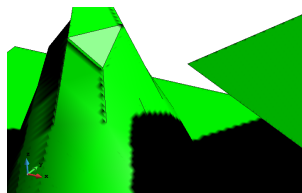
medium quality shadows



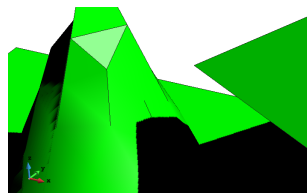
enable finer silhouette with high quality ( 16 MB)



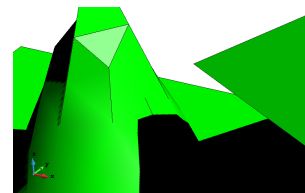
high quality shadows



coarse shadows (zoomed view)

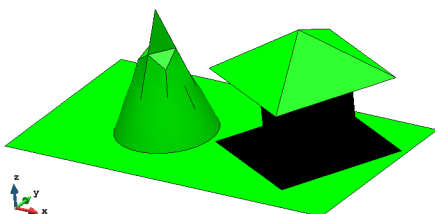


medium quality shadows (zoomed view)

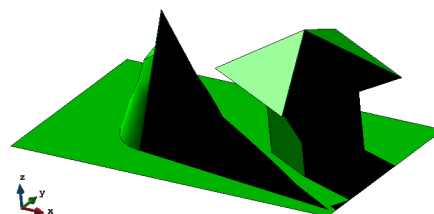


high quality shadows (zoomed view)

- 6 . 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:

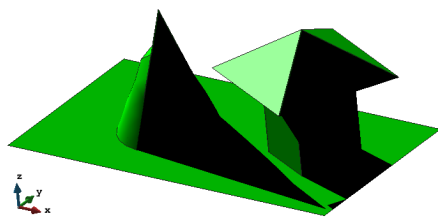


changing direction of shadows

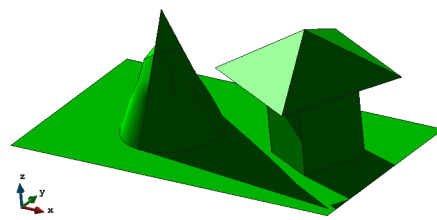


changing direction of shadows

- 7 . Select **ambient** as shadow colour: draws the model with a dimmed ambient light in the shadowed areas:

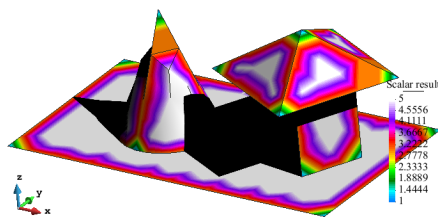


black shadows

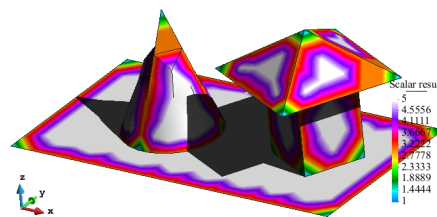


shadows under dimmed light

8 . Select some **contour fill** to the model:



black shadows

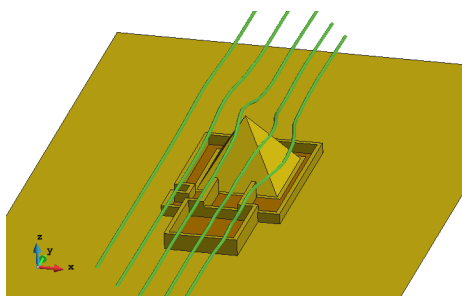


shadows under dimmed light

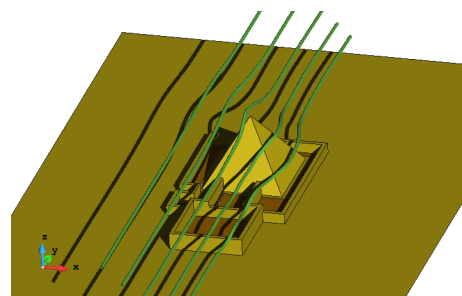
**Example 2**

Demonstrates better depth perception of the created stream lines: which ones are near the ground and which ones not.

- 1 . In preprocess load **pyramid\_s.gid** model
- 2 . Switch to postprocess
- 3 . Select **Files->Import->Stream lines...** and choose pyramid\_s.flavia.streams.msh located in pyramid\_s.gid
- 4 . Using shadows, the height of the stream lines are better perceived than without



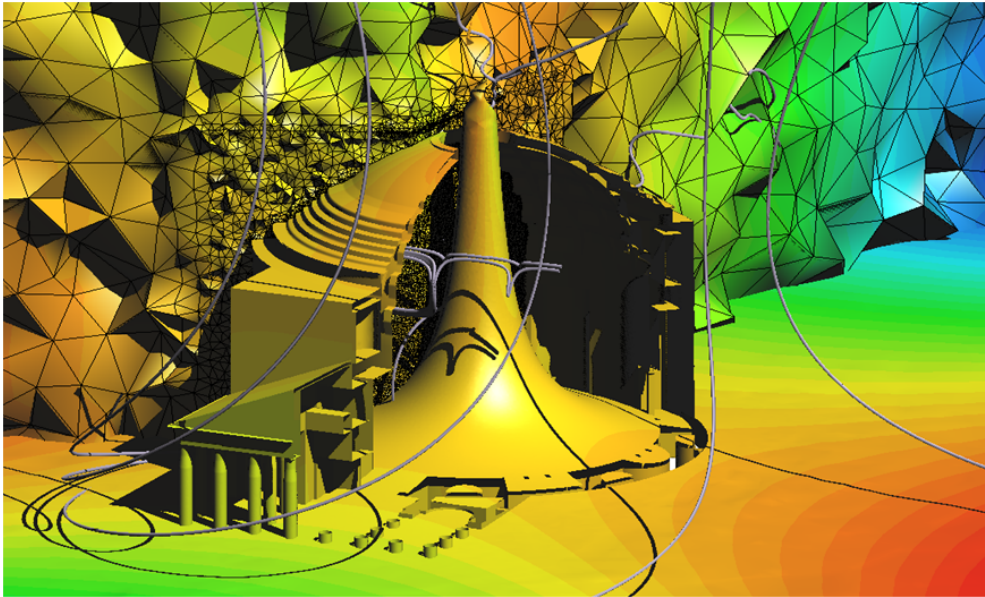
without shadows the relative position of the stream lines can not be well appreciated



shadows helps to perceive the relative position of the stream lines

**Another example**

Here is a picture of another example using shadows with stream lines:



shadows helping to appreciate stream lines correctly in space

#### 4.4 Decoration

##### Model used

The model `platform_small.gid` will be used in this example, which can be found at [Material location](#).

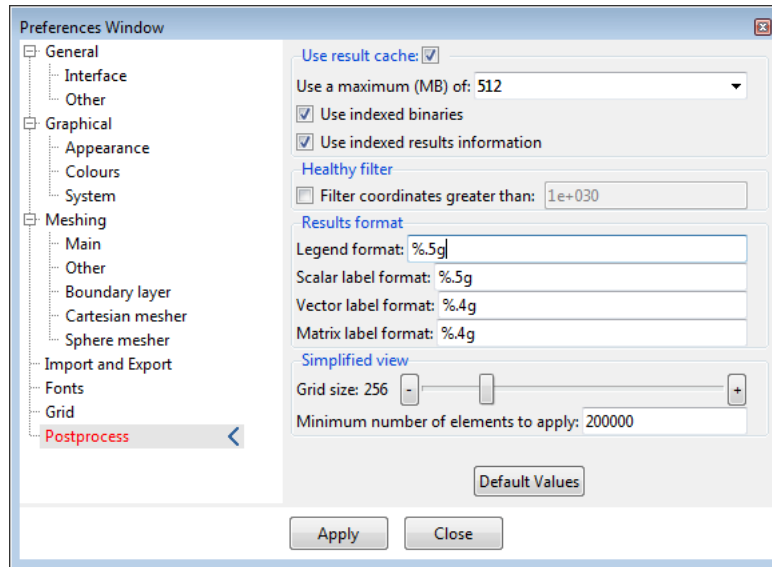
##### Description

This tutorial shows several techniques to decorate the single volume mesh of our fluid simulation:

- extract boundaries: with this option, GiD will
  - create a surface mesh with the boundary elements of a volume mesh, and
  - a mesh of lines with the boundary lines of a surface mesh;
- separate connected components:
  - if several volumes are present in a single volume mesh, then each one of them will be put in separated volume meshes;
  - each smooth surface will be put in separated surface meshes, i.e. neighbour elements which are not separated by a boundary line will be grouped together, for instance, the surface of a cube, which has six faces, will be separated in six surface meshes; but the surface of a sphere will remain a single surface mesh;
- divide by selection: selected elements will be moved to a new mesh.

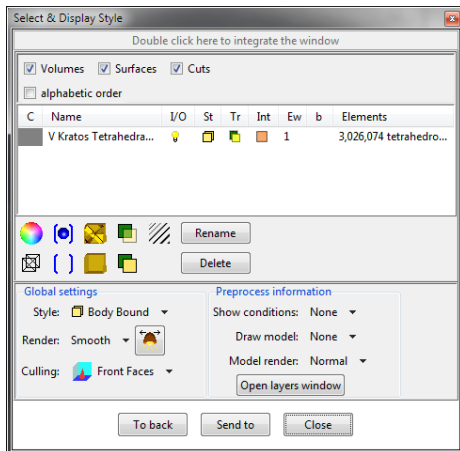
**Note:** Don't forget to enable the result's cache, if you're using a 32 bits operating system. In order to do an iso-surface animation, the required memory is:  $606,000 \text{ nodes} * 200 \text{ steps} * 4 \text{ bytes} / \text{result} = 463 \text{ MB}$

**Note:** if your computer is running a 32 bits operating system, please enable the result's cache option and set it to 512 MB. The option can be found at the **Utilities->Preferences...** , in **Postprocess** branch

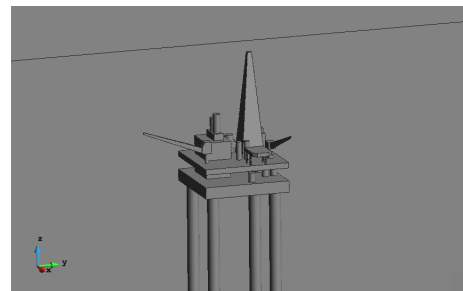


Results cache options in preferences panel

- 1 . In preprocess mode, load platform\_small.gid model
- 2 . Switch to postprocess
- 3 . Select **Window->View style...** as can be seen there is only one volume mesh, with all the information
- 4 . Select **Front Faces** in **Culling** option in order to see the interior of the volume

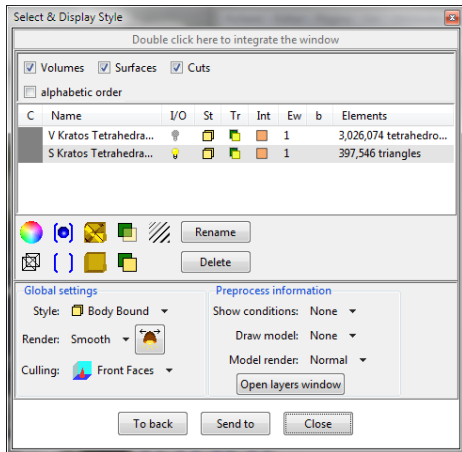


Single mesh, with culling enable to view inside the volume

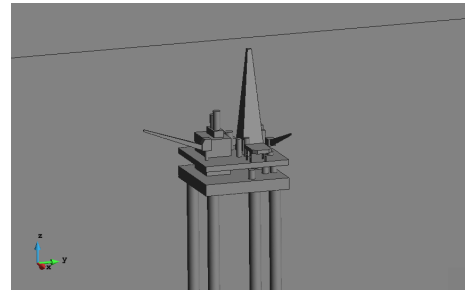


Monochrome view of the model, culling the front faces

- 5 . Select **Options->Geometry->extract boundaries** to get the boundary triangle mesh of the volume

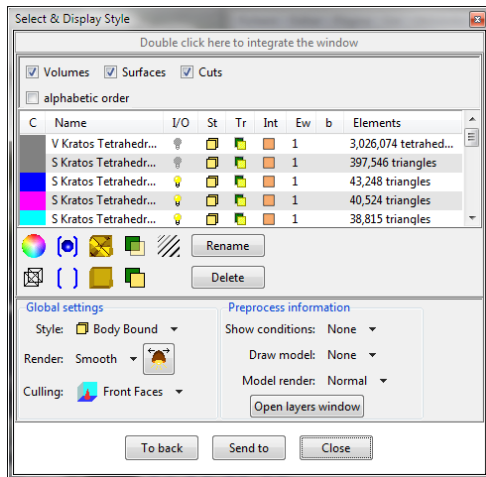


Volume and surface mesh of the model

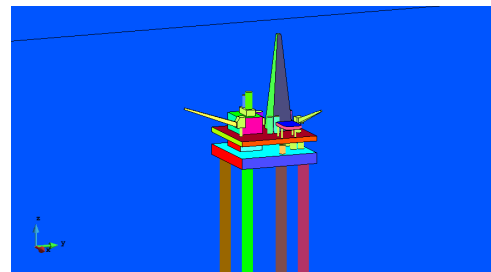


Boundary of volume mesh

6 . Then select **Options->Geometry->Separate connected components** to subdivide the single surface mesh into separate meshes limited by the boundary lines



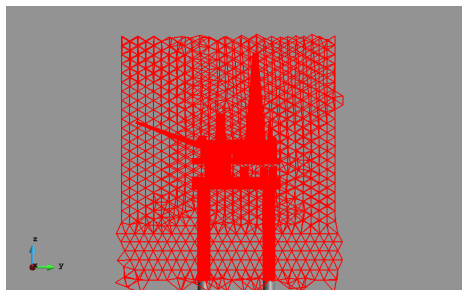
Subdivided surface mesh



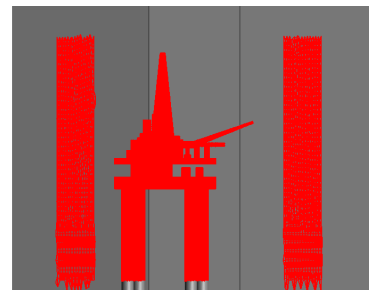
Each separated component has a different colour

There were more than 50 connected components in the model. There is another way to separate the platform from the surrounding box,

- 7 . Before trying this method select the created surfaces and press **Delete** button
- 8 . Select **Rotate->plane YZ** from the contextual menu
- 9 . Select **Do cuts->divide by selection** and select the elements next to the platform. Please notice that some more elements were selected

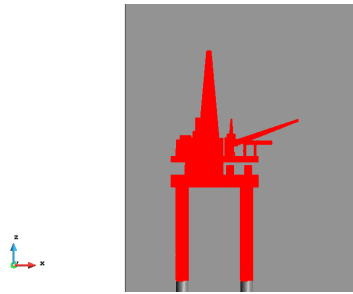


PlaneYZ view to select the elements of the platform



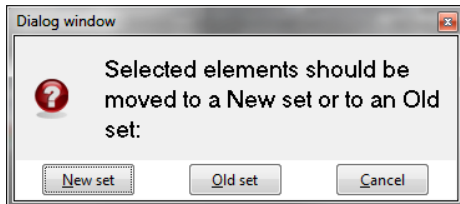
PlaneXZ-5 Some elements of the surrounding box were also selected

- 10 . Rotate the model 90 degrees and select the option **Contextual->Remove from selection** from the right button mouse menu and select the undesired elements

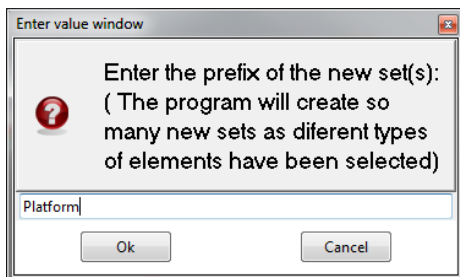


PlaneXZ to select undesired elements to be removed from the selection

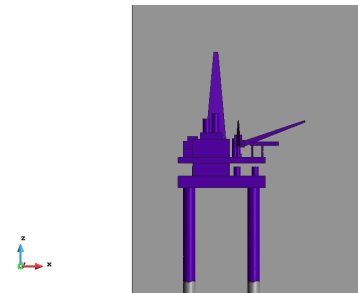
- 11 . Press **Escape** button
- 12 . A window will appear asking whether the selected elements should be moved to an existent set or to a new one. As we want a new set, we'll press the **New set** button
- 13 . Enter "Platform" and press **Ok**. A new mesh will be created with the platform elements



Selecting the new set option

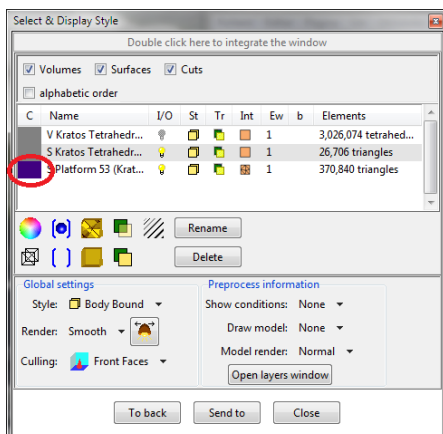


Entering the name of the new mesh

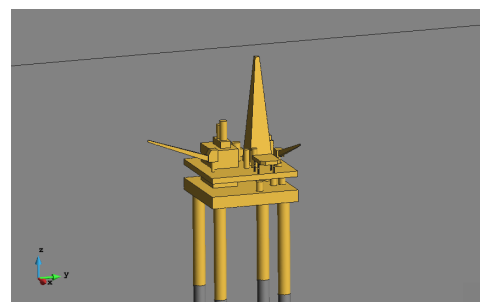


New mesh created with the platform triangles

- 14 . Now we can change the colour of the platform, just by clicking over the colour box on the left of the mesh name in the **Display style window**



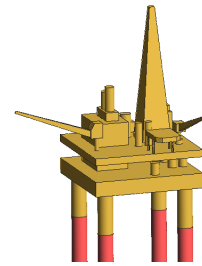
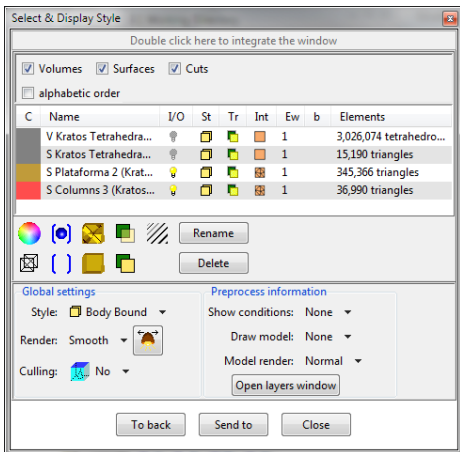
Changing colour of new mesh



Each separated component has a different colour



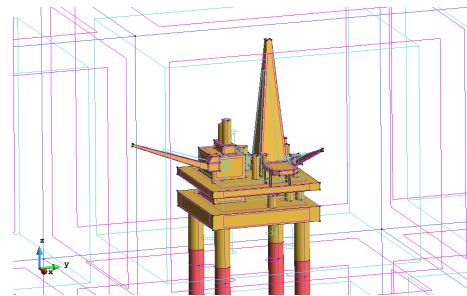
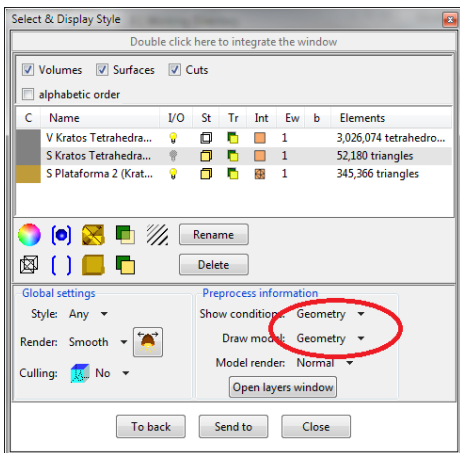
15 . Same for the columns which sustains the platform



Creating columns mesh

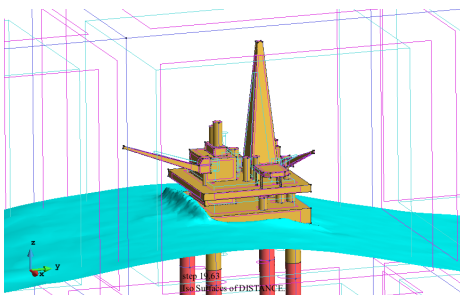
Creating and colouring columns mesh

16 . The preprocess information can also be drawn over the postprocess results, both the geometry

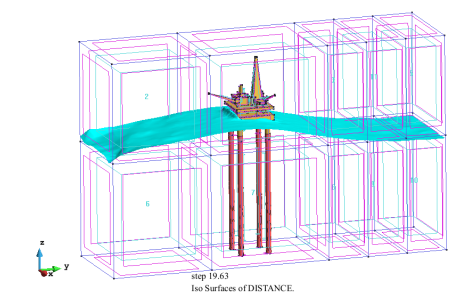


Both post-process mesh and pre-proces geometry

Selecting the pre-process geometry in normal render

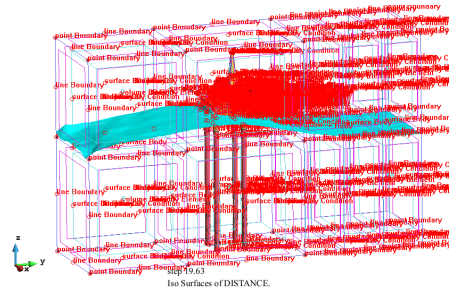
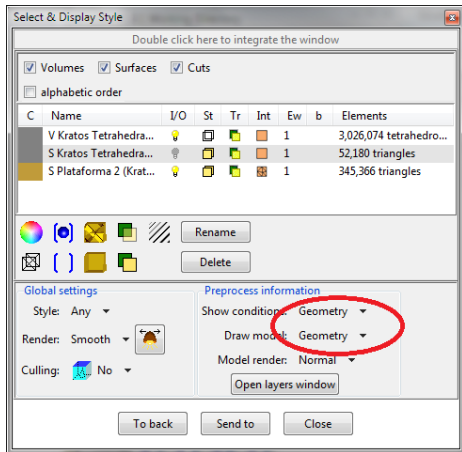


Free iso-surface over the pre-process geometry

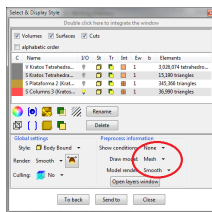


Free iso-surface over the pre-process geometry

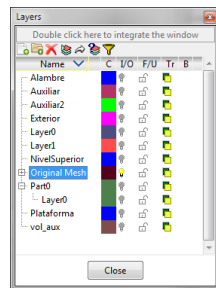
17 . With their conditions



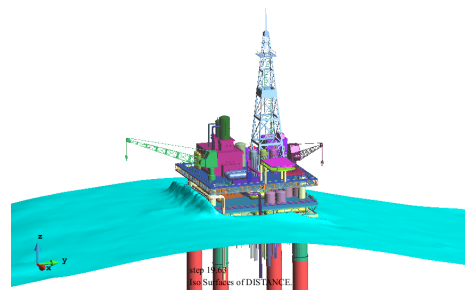
18 . And the original, full detailed model; which can also be used to animate the iso-surfaces



Selecting the original mesh in smooth render

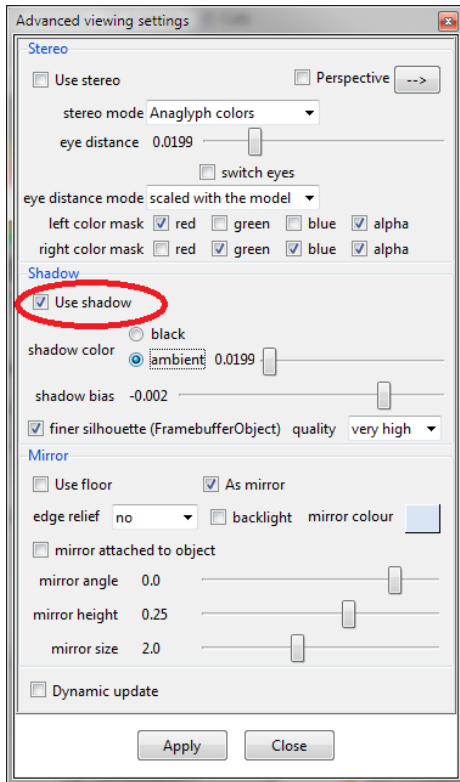


Switching on the desired layer

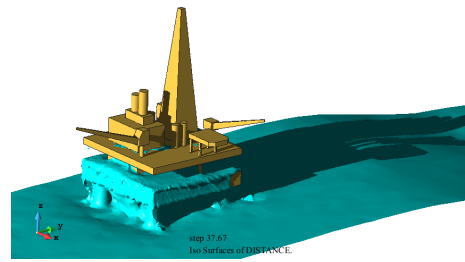


Showing the iso-surface over the full-featured original model

19 . Enabling shadows, under **View->Advanced viewing settings...**, also provides more realism to the view (the light direction can be changed by selecting the option **Render->change light dir** in the right button mouse menu)



Enabling shadows in the advanced viewing settings window



Shadows over the model, the light direction was changed





## 5 Animations

### Technique

This tutorial explains several details which should be taken into account when an animation is created for a later presentation.


Then an example is used showing tips regarding the creation of presentation videos.

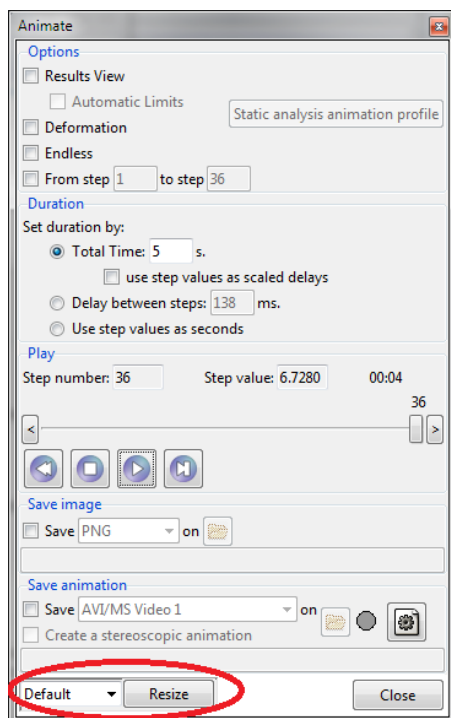
### 5.1 Basic rules

The two most important rules when creating an animation are:

- 1 which is the target of your animation, i.e. where should your video be reproduced, and
- 2 save the data used to create the video, including view, size and results, because surely the video should be recreated with small modifications.

The basic rules that should be followed when creating an animation are:

- **Target of the animation:** if the animation is to be included in a presentation to be projected, usually projectors have a resolution of 1024x768 or 1280x720, and your desktop monitor is 1680x1050 or 1920x1080.
- **Size of the window:** avoid creating videos in full-screen mode. In full-screen mode, the videos will be very big and, when the presentation reduces the resolution to fit the projector, details will be lost and text will become unreadable. Use the *Resize graphical window* macro (  ) or the *Resize* option in the Window --> Animate window or the render size control macros, which define the size of the graphical view of the model. These options are also useful to resize the graphical window between sessions. You can use this macro to guarantee the size of the graphical window when changing from pre to post or between gid sessions.



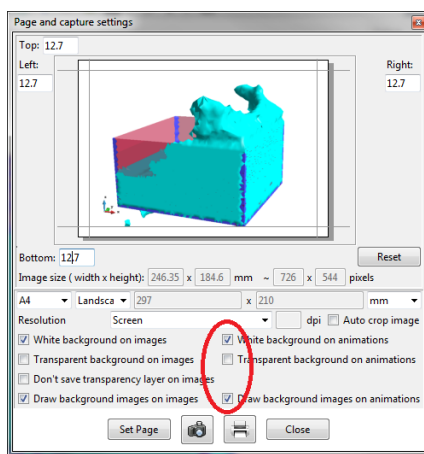
Resize button in the animation control window



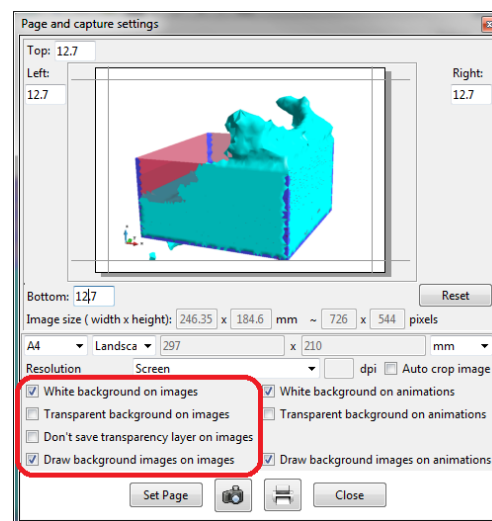
Macros to control the rendering window size

- **Views:** centre your region of interest and save the view using *View --> Save*. This way the view can be used to do different animations, or glue together different animation chunks.
- **Format:** there are several compression formats, but the recommended ones are:

- AVI/MS Video 1 : *Recommended for presentations*, although is very old, it's the most compatible and videos created with this compression can be embedded in presentations without problems and all video players can reproduce them.
- MPEG : uses the MPEG2 codec, it's also supported every where, although in MS Windows some problems can appear if the resolution used is bigger than 720x576.
- Macromedia Flash Video ( flv ) : *Recommended for webs*, uses the "screenvideo" compression, but needs a player ( like any other video ) if the animation is included in a web page.
- GIF : to include the video as picture in a web page. Bear in mind that the colour quality may be adversely affected by the drastically reduced palette, only 256 colours can be used. Try the different gif options to select the best for your animation.
- **Graphical objects**: graphical objects, like legends, comments or axes can be placed anywhere in the screen using the *Utilities --> Tools --> Move screen objects* .
- **Background**: despite the configuration of the viewed background, when creating a video the background settings are controlled through the *File --> Page and capture settings*:



Background options used to create videos



Resolution and background options used to capture images

**Note:** the *Resolution* options in the above window is very useful to create images of increased size needed for publications like posters, articles or books. This options is not used to create high-resolution videos.

- **Perspective**: use perspective for a better perception, can be enabled through *View --> Perspective...*

### Stereoscopy (3D)

Stereoscopy can be enable through the window *View --> Advanced view settings*

- **Perspective**: use perspective for a better perception, can be enabled from the same window or through *View --> Perspective...*
- **Full-screen**: if the demo is done using GiD, a better 3d experience can be achieved in full-screen mode, by pressing *F11* or *View --> Full screen*.
- **Colours**: if the stereo mode used is the Anaglyph colours, the colour palette is drastically reduced, because some colours are only viewed by one eye and not by the other producing discomfort. The appropriated colours depends on the colour filters used. If 3d colour filters used are red and cyan, most of the cases, then there is a special contour fill colour map for this mode, under *Option --> Contour --> Colour scale --> 3D anaglyphs*, if the 3d colour filters used are red and cyan.

Useful colours for red & cyan stereoscopy are greys, white, magenta and yellow.

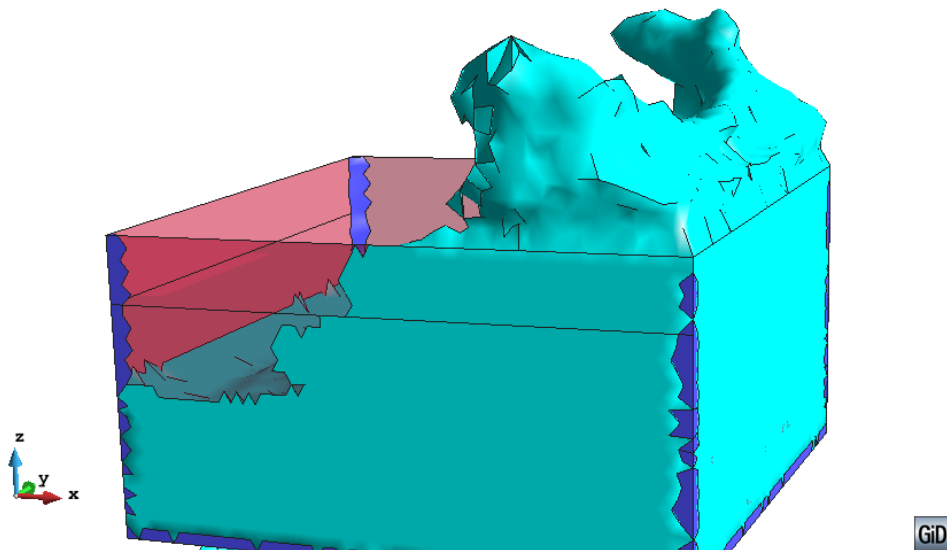
- 
- **Saving videos:** at the bottom of the *Animate window* there is an option called *Create a stereoscopic animation* which can be used to save stereoscopic animations, which can be reproduced with the proper player. In these videos, the image for the left eye and the one for the right eye are saved in the same frame, i.e. doubling the horizontal resolution of the frame, and the player splits these left and right images and alternates them in the output. Stereoscopic animations can also be created in non-stereo computers.



## 5.2 Example

### Model used

Model used in this example:

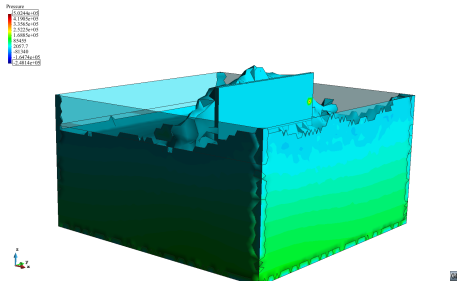


Wave generator model with a different mesh at each time-step.

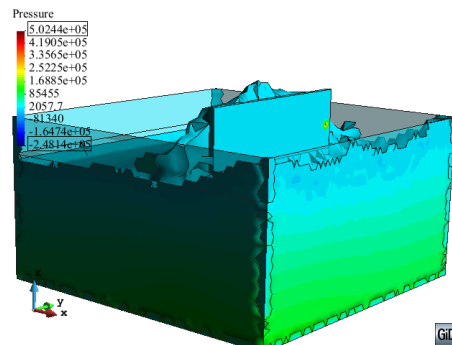
This example is a simulation of a very simple wave generator, which has different meshes at different time-steps. The file `wave_generator.post.bin` can be found at [Material location](#).

### 5.2.1 First observation: resolution

- 1 load the `wave_generator.post.bin`, file which can be found at [Material location](#);
- 2 do a contour fill of *Pressure* and maximize the window;
- 3 with *Utilities --> Tools --> Move screen objects* move the legend to the upper left corner;
- 4 now create an animation:
  - with a *Delay between steps* of 1000 ms,
  - with *Automatic limits* enabled,
  - using the *AVI/MS Video 1* compression codec,
  - and save the animation;
- 5 now reduce the size of the graphical window to 640x480 and save the animation with another name;
- 6 reproduce both animations reducing the player window to half of the screen;
- 7 it can be observed that the legend information can not be read in the first animation, but in the second;



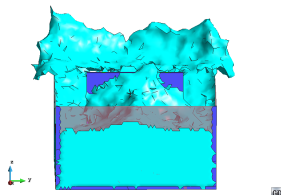
Single frame from a full screen video, in which the legend text can not be read.



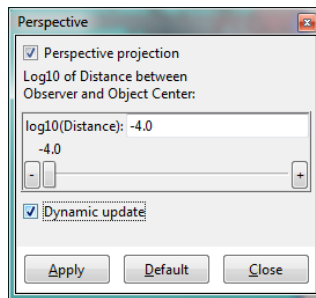
Single frame from a video with a resolution of 640x480, now the legend text can be read.

**5.2.2 Second observation: perspective**

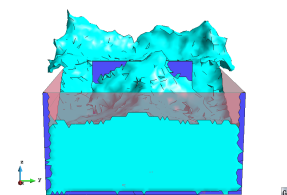
- 1 With the same example, just select *Results --> No results*
- 2 rotate the model so that the wave comes to you;
- 3 play the animation;
- 4 enable perspective;
- 5 play the animation again;
- 6 is there a difference?



Wave going away from the user with no perspective



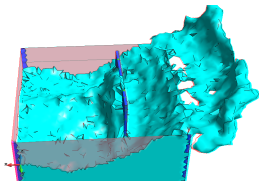
Perspective option enabled



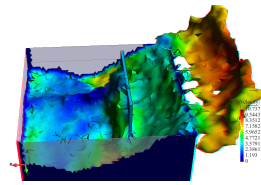
Wave going away from the user with perspective

**5.2.3 Third observation: 3D**

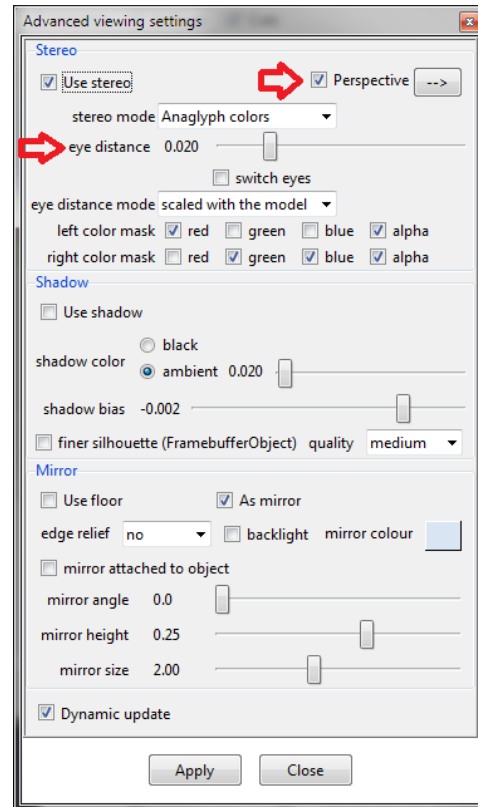
- 5.2.3.1 With the same model, open the *View --> Advanced vieweing settings ...* window;
- 2 now enable the stereoscopic view:
  - some settings may need an adjustment, like the *eye distance*, which depends on each viewer;
- 3 there is a bad *3D perception* due to the bad colours:
  - the *liquid* mesh is sky blue, and therefore only the right eye sees it and the left eye only sees a black region ( try closing one eye and then the other onr);
  - try doing a contour fill, for instance of the `|Velocity|` result, some coloured regions are better perceived than other ( try again closing one eye and then the other);
  - play with the *animate window*;



Only the right eye can see the model, the right eye only sees a black spot, producing some discomfort



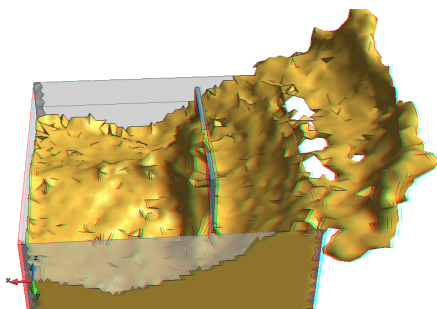
Visualizing a contour fill of |Velocity| with the standard rainbow colour map causes that some regions are viewed by one eye and not by the other, producing some discomfort



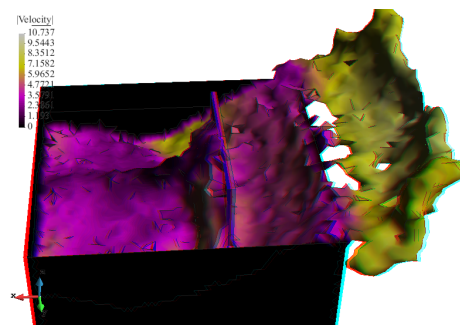
Useful stereo options

4 correct the colours:

- try choosing different colour for the meshes, like grey, yellow, or magenta;
- in the contour fill just select *Options --> Contour --> Colour scale --> 3D anaglyphs* or *3D anaglyphs 2*, this is a special colour map which uses colours visible by both eyes at the same time when the user is using the red-cyan glasses;



Meshes drawn using grey and gold like colours which can be seen from both eyes



Contour fill of |Velocity| using the 3D anaglyphs colour scale

5 play the animation again;

6 press **F11**.





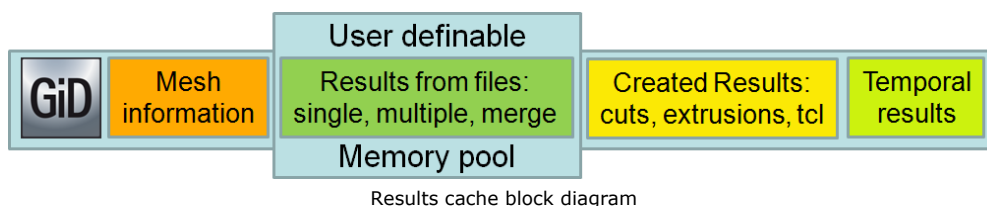
## 6 Working with large models

### 6.1 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.

#### 6.1.1 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 performed each time it is opened, but no results are 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*.

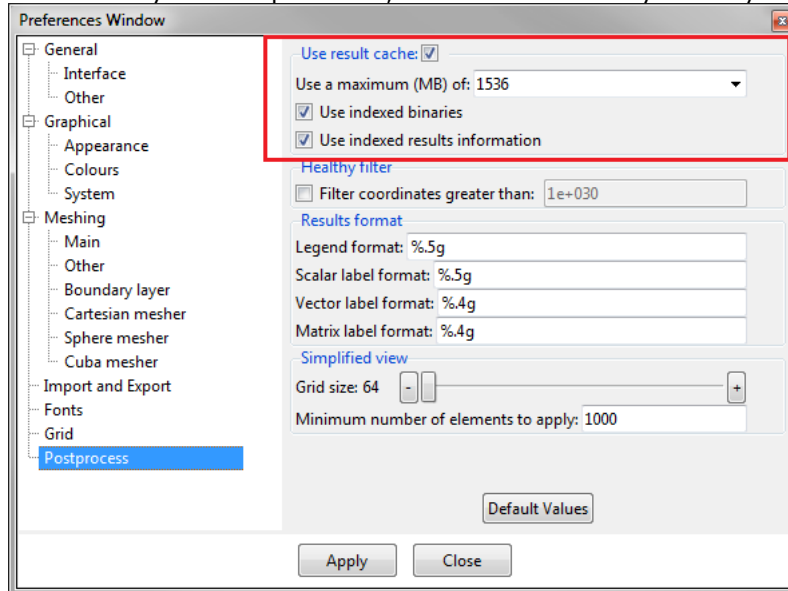
**What's not cached:** When cuts, extrusions are done or isosurfaces are converted to full featured meshes, the generated results are held in memory. Results created in GiD using the *Window --> Create result, Create statistical result* are held in memory. Also results imported using *Files --> Import* 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* and open it again.

## Options

- The user can **enable the Result's cache** in the *postprocess panel* of the *Utilities --> Preferences* window.
- 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.



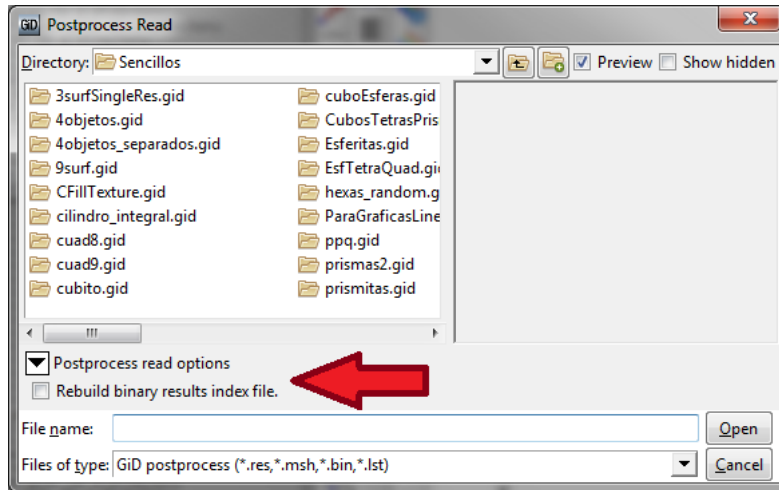
Results' cache options

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 and an animation of these vectors is desired, to get a fluid animation the memory pool must be set to  $16 * 100 = 1.6$  GBytes of memory. But if the desired 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.
- **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.

**Note:** The index of the results' file is automatically created if it does not exists or if the results' file is newer than the 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 then 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 open!

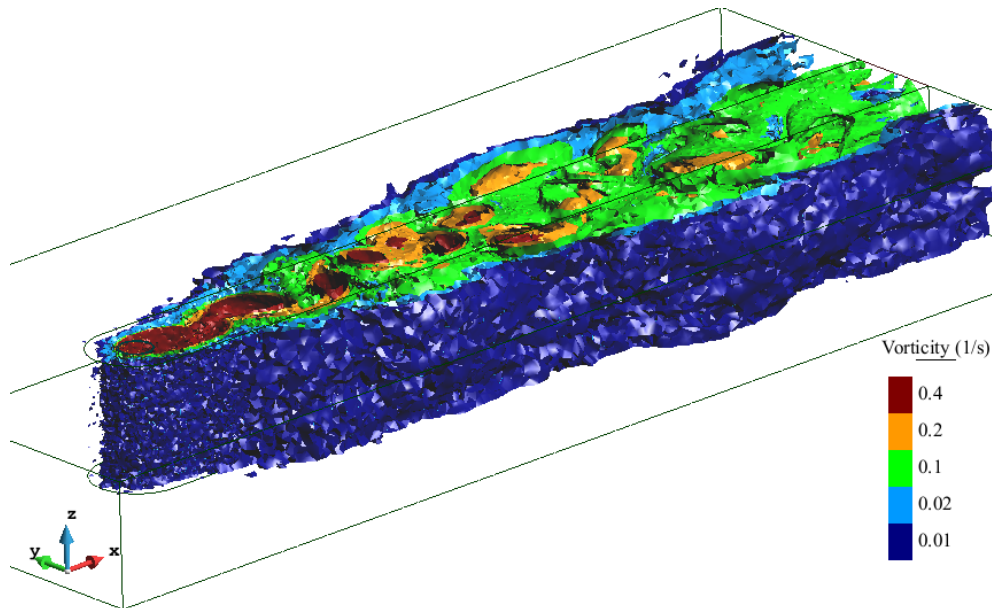
Usually in Linux the maximum number of opened files are, precisely, 1024, and some of them are already used, causing GiD to display an error when the user tries to merge these 1024 result's files.

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`

### 6.1.2 Example





3D cylinder snapshot

This example is a simulation of the flow turbulence behind a 3D cylinder. The model `3Dcilbody.post.bin` 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, around 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. Enable the use of **indexed binaries** and **indexed results information**.

With *indexed binaries* and *indexed results information* it will take around 1 minute to create the 24 MB index file `3Dcilbody.post.bin.idx`, the time to parse the whole original file. From now on, loading the same model again, or from another gid session, it will take only 15 seconds, the time to read the mesh, read the index file and create the graphical objects to visualize.

After loading the model in GiD with these settings, in the *task administrator* can be seen that GiD uses less than 1 GBytes of memory, around **450 MBytes** of memory, for the 32 bits version of GiD; and around **550 MBytes** of memory, for the 64 bits version, both in MS Windows.

- 1 load the [3D cylinder example](#);
- 2 switch off the surface sets and let **only the volume meshes on**;
- 3 check that the option *Do cuts --> Automatically convert cut to sets* is disabled;
- 4 do a cut along the model ( after clicking the 1st point, you can press the **Alt** key to snap the dynamic line to the horizontal, vertical or diagonals):

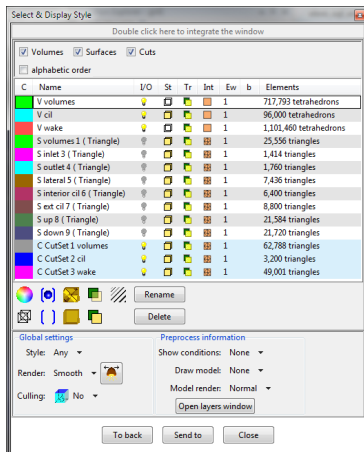


Original model

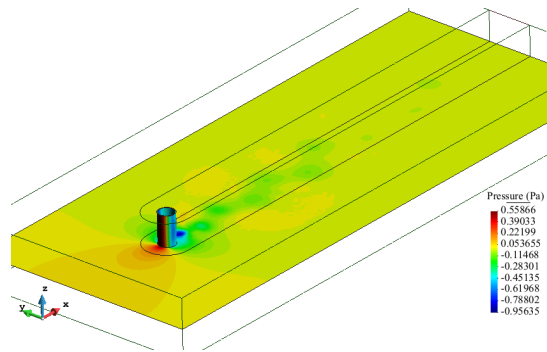


Model with cut along the domain

5 change the visualization style to "boundaries" for the volumes "volumes" and "wake", and to "body boundaries" for the volume "cil" and the cut planes:



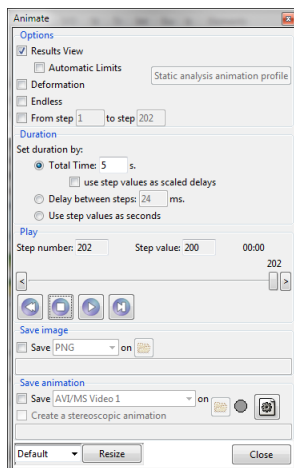
Styles applied to the different meshes



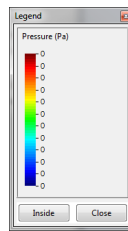
Contour fill of the pressure result at the last time step

6 select the option *View results* --> *Contour fill* --> *Pressure* ( it takes some time until GiD reads the *Pressure* result for the last step);

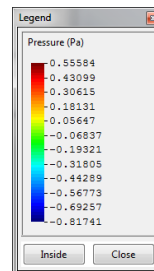
7 now open the animation window and animate the model, you'll see that the legend of the contour fill varies:



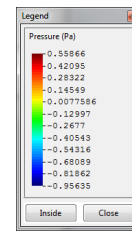
Animation window



legend of the first time step



legend of the 100th time step



legend of the last time step

8 if the **automatic limits** is checked, then GiD will read all the pressure results to get the global minimum and maximum values for all steps. This is **slow** as GiD reads the result for each step to get it's minimum and maximum values;

9 if we do an animation now we'll see that almost all steps take some time to read the result from disk and draw it, this is because the *result's cache memory pool* is too small;

10 as we have around 340.000 nodes, 202 steps, to hold the pressure result for all steps GiD requires 268 MBytes = 340.000 \* 202 \* 4 bytes per result value ( pressure is a scalar result). So now, let's increase the result's cache to 512 MB by going to the corresponding preferences and selecting the 512 MB entry in the menu, and pressing *Accept*;

11 now play the animation again: the first loop will take almost the same time as before, because the results are read in memory, but the next loops the animation is played faster, as the pressure result for all steps are in memory. By checking the *task administrator*, GiD now uses more memory than before, around **670 MBytes**;

12 now if we enable the *automatic limits* option we'll see that is faster than before.

**Note:**

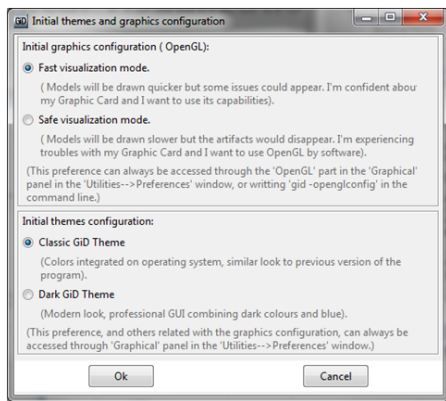
- if we convert the cut to a full featured mesh, then the results for this cut are created and **are not cached**. If you do this, you'll see in the *task administrator*, that GiD now uses almost **~2,5 GBytes** of memory.
- If this information is saved using the *Files --> Export --> Post information --> whole model'* and read again, then the results on the cut mesh will be cached as the other results too, and GiD will again use a modest amount of memory.

**6.2 Using the graphics card**

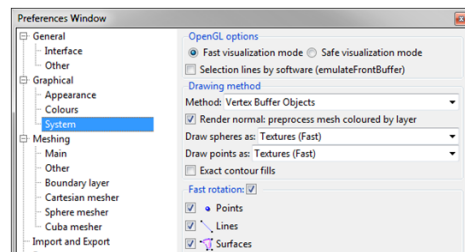
This section explains how GiD can use the available graphics resources and which features can be used under the selected visualization configuration.

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.

**6.2.1 Fast/safe visualization mode**



Initial GiD configuration when GiD is started for the first time.



Graphical configuration in the Utilities --> preferences window.

• **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 disappear but models and meshes will be drawn slower.

*In MS Windows:* the OpenGL version used is 1.1 and some features will be disabled.

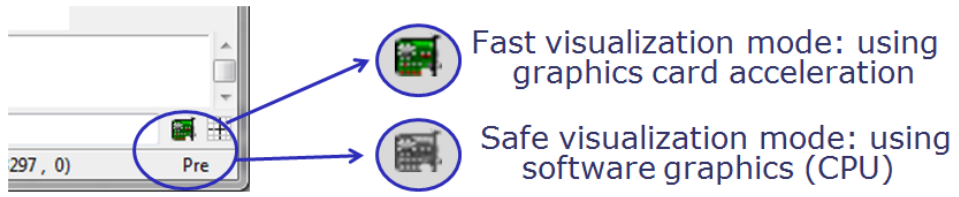
*In Linux:* this mode can also be enabled using the `gidx` script in a console or terminal window. The OpenGL version used is 2.1.

• **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:** 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:



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
}

```

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
  }
```

**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.

## 6.2.2 Drawing methods

**Drawing method:** ( can be adjusted at *Utilities --> 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.

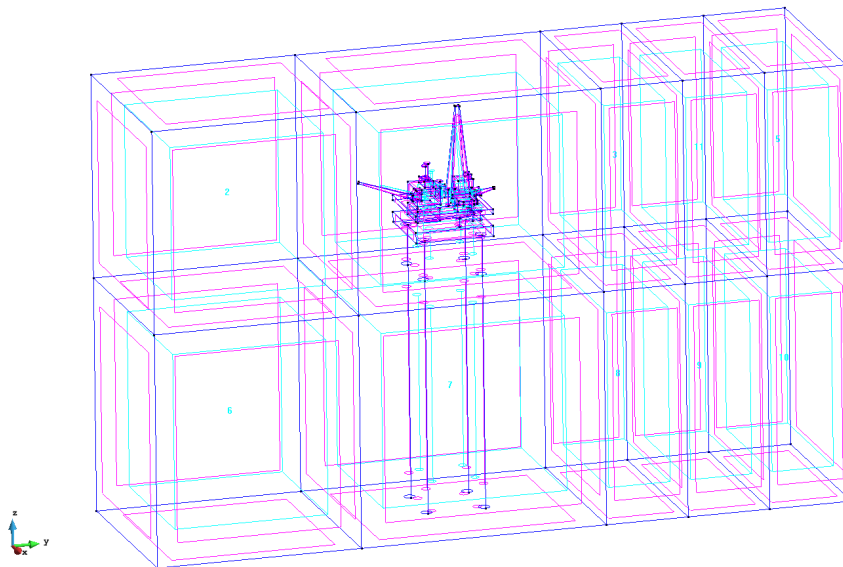
Summary of features available in Safe and Fast visualization modes:

	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 (Options --> Contour --> Bright Color)	No ( requires OpenGL 1.2)	Yes
Recommended mesh size	< 500,000 triangles	depends on the graphic's card memory available (check Help --> About --> More)

### 6.2.3 Example




#### Example

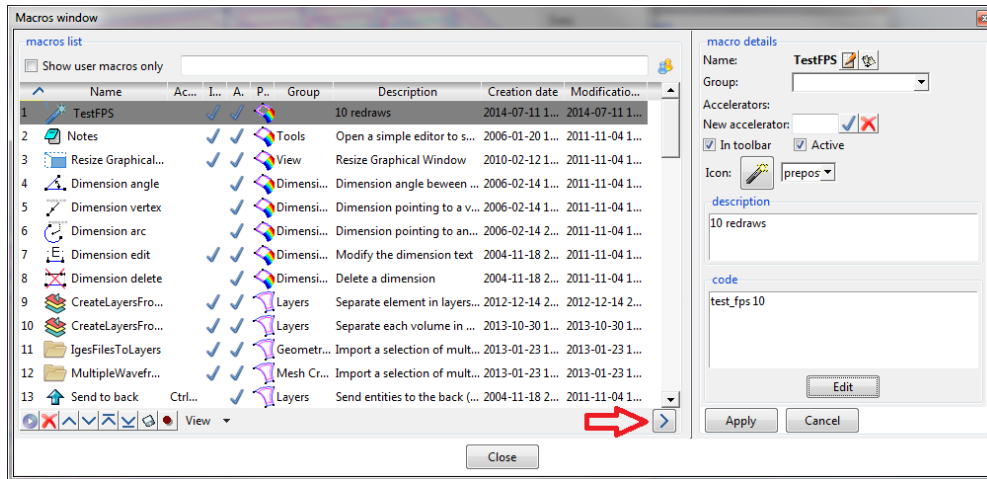


Geometry of the paltform\_small project.

This example is a simulation of waves against a oil platform. The model `platform_small.gid` 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;
- 2 edit the macro with the icon  on the same macros toolbar, and enter `test_fps 10` in the code box:

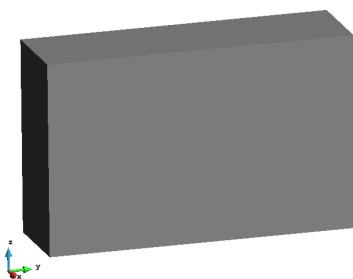


Edit macros window, click on the button pointed by the red arrow to expand the window and enter the code

- 3 load the model and test the macro, a *Warning* window should appear with a text like this:  
*Redrawing 10 times ... done: 123 fps. ;*
- 4 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 the Warning window, three 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
```

**Fast Visualization mode in postprocess**

- 1 click on the lower right card icon  to change to **Fast visualization mode** ( click no when GiD asks to save the model);
- 2 load the model again and click on the newly created macro ( the geometry of the model should be viewed);



- 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 go to the preferences window and change the option *Graphics --> System --> Drawing method* to **Vertex buffer objects;**
- 11 click on the newly created marco ( don't close the appeared Warning window);
- 12 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.
    
```

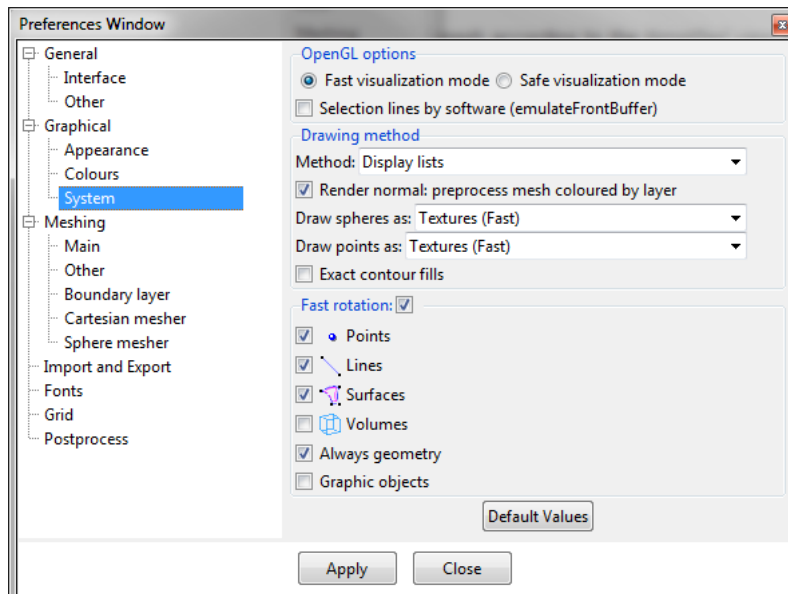
- 13 compare the times obtained:

	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

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

### 6.3 Fast rotation

#### 6.3.1 How it works



Fast rotation enable in the preferences window

When heavy models are loaded in GiD, it can take 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 has implemented a **Fast rotation** mode, which allows fast interaction with the model.

When this option is 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 rotating 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.

When Fast rotation is enabled: several options can be configured, the behaviour if these depends if GiD is in preprocess or postprocess mode:

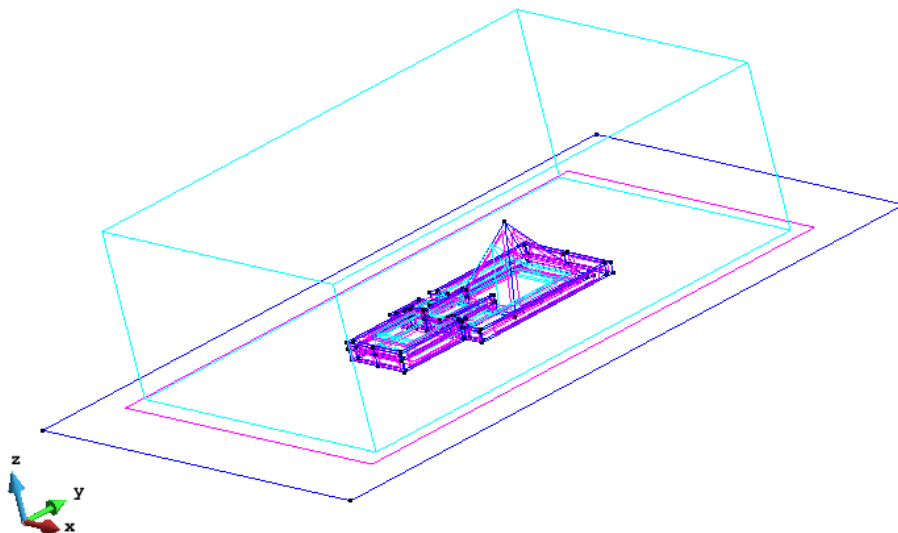
**Preprocess:**

- *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 boundaries lines of the mesh is drawn;
- *Surfaces:* if it's set, when rotating a simplified view of the mesh is drawn if the mesh has more elements than the limit specified in the Preferences-->Postprocess-->Simplified view section.


### 6.3.2 Example



Geometry of the pyramid calculation example

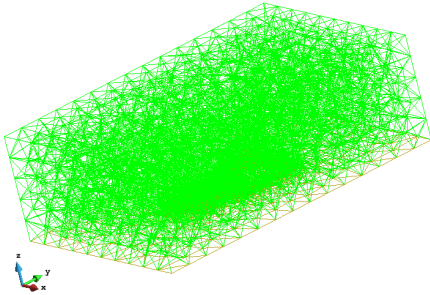
This simulation is the air flow around a pyramid, calculated in the basic tutorial example, look at [Run a CFD simulation](#). The model `pyramid.gid` can be found at [Material location](#).

**Preprocess:**

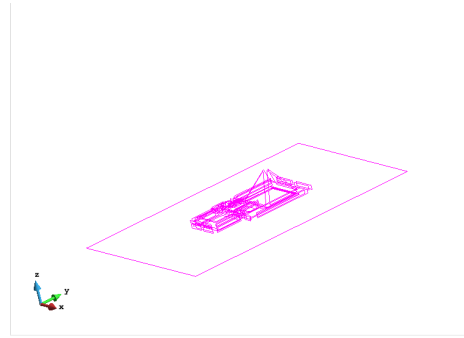
- 1 load the model, and change to mesh view mode, using the icon ;
- 2 open the preferences window and enable the *Preferences --> Graphical system --> Fast rotation*

*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:



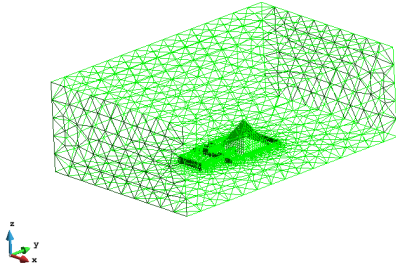
Mesh view of the pyramid model



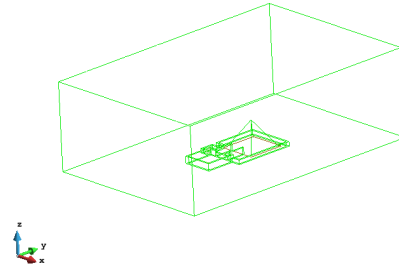
'Fast rotation' view, showing only the surfaces of the model while rotating it

### 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:

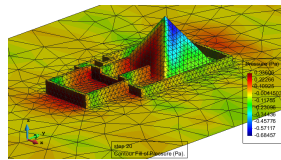
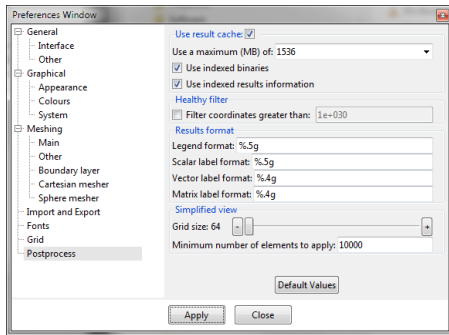


'All lines' style in postprocess mode

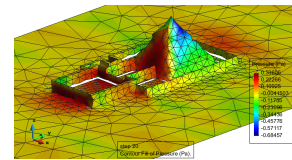


Boundaries style used in 'Fast rotation' when only the 'Lines' are enabled

- 7 do a contour fill of pressure;  
 8 switch on the 'Surfaces' checkbox in the *Preferences --> Graphical system --> Fast rotation mode* section;  
 9 in the *Preferences --> Postprocess --> Simplified view* section, change the *Grid size* to 64 and the *Minimum number of elements to apply* to 1000;  
 10 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:



Original view of the pyramid Pressure result



'Fast rotation' view of the pyramid model


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

## 6.4 Simplified view

### 6.4.1 How it works

#### Simplified view

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 -pag. 98-](#)), or at user demand using the following icons:

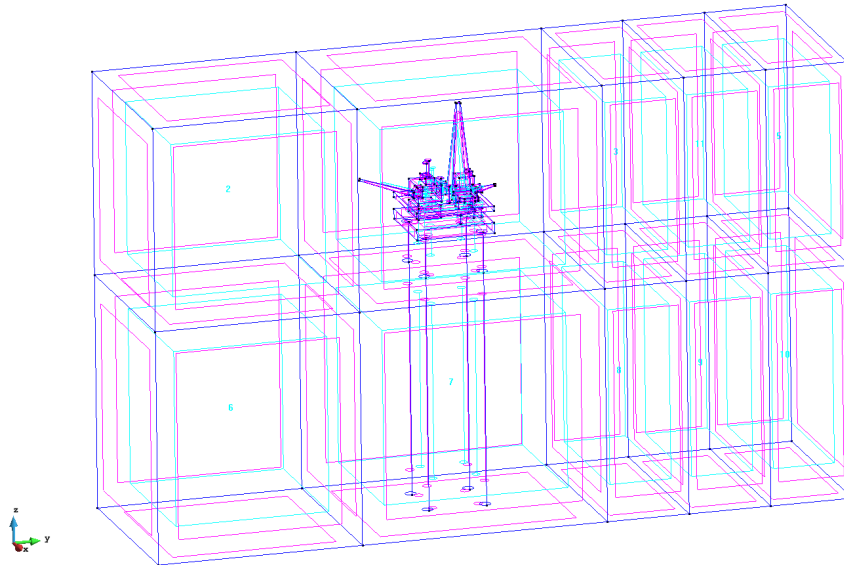
**Toolbar:**  GiD is drawing the original mesh, click to change to simplified view.

**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.
- **Minimum number of elements to apply:** minimum size of the mesh which will trigger the mesh simplification process in Fast rotation is enabled ( see [Fast rotation -pag. 98-](#)). Bigger meshes will be simplified, smaller ones not.

### 6.4.2 Example

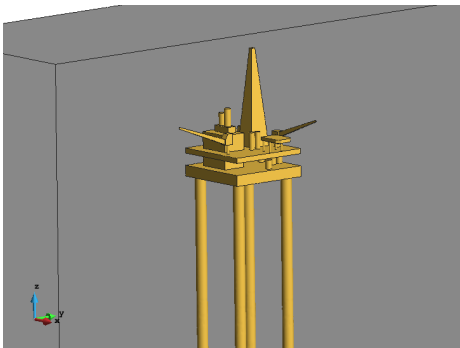
#### Example



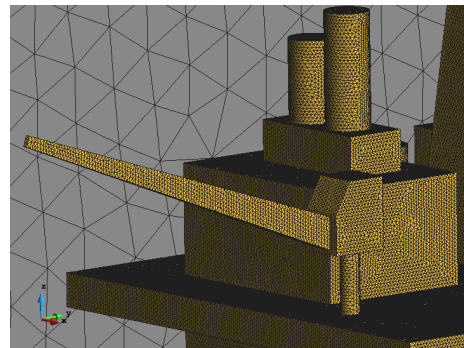
Geometry of the platform\_small model.

This example is a simulation of waves against a oil platform. The model `platform_small.gid` can be found at [Material location](#).





- 1 load the model;
- 2 go to *postprocess*;
- 3 select *Options* --> *Geometry* --> *Extract boundaries*;
- 4 select *All lines* display style;
- 5 create a new set with the elements of the platform, and change its colour;

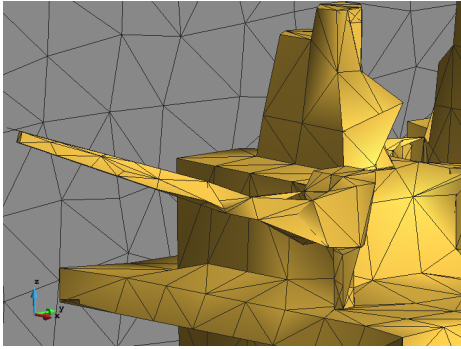


The single triangle mesh of the volume boundary has been separated in the grey box triangle set and the golden platform.

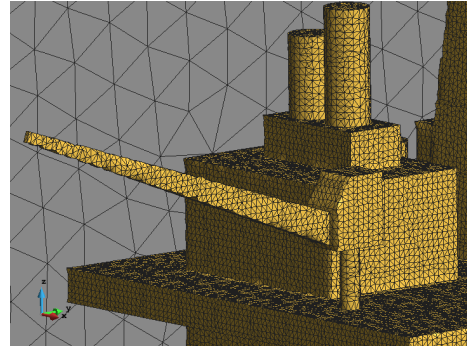


Zoom view showing the refined triangle mesh used to model the platform.

- 6 in the *Preferences* --> *Postprocess* --> *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* --> *Postprocess* --> *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

**6.5 Textures in spheres and contour fill**

**6.5.1 How it works**

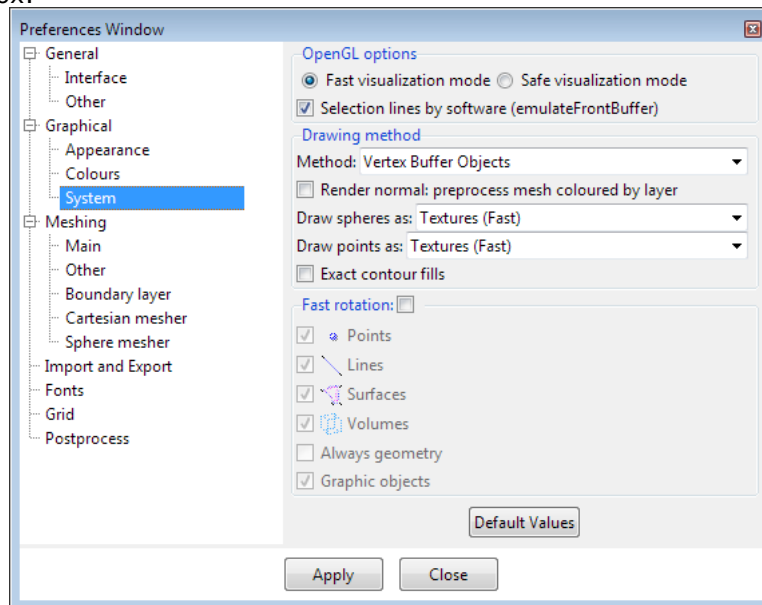
There are several ways to draw spheres and points in GiD:

*Nicely*, using a triangle mesh to represent the sphere or point, which has between 4 and 630 triangles, depending on the detail level;

*Textures*: using images of already rendered spheres to speed up the drawing process;

*Square / Quick*: the fastest way to draw spheres, use this method when speed is the most important factor.

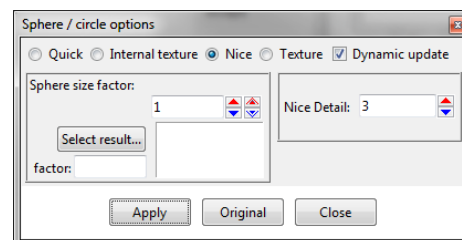
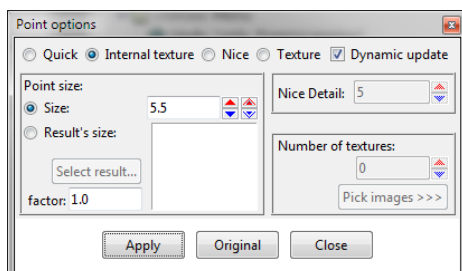
In **preprocess** these options are accesible through *Preferences --> Graphical --> System* inside the *Drawing method* box:



Preferences window: Graphical --> System

- Under *Draw spheres as* and *Draw points as*, the speed and quality of the mesh element type sphere and point can be configured, as there are sevel methods to draw spheres and points:
  - Textures (Fast)*: an image is used to drawn spheres / points. it's a compromise between quality and speed.
  - Mesh ( Slow)*: a triangle mesh, between 4 and 630 triangles big, is used to draw each sphere / point for a nice quality.
  - Square ( Fastest)*: draws spheres / points as a square. This is the fastest drawing method for point and sphere elements.

In **Postprocess** these options are under *Options --> Geometry --> Point options* and *Sphere options*:



- **Quick:** points, spheres and circles will be drawn as big dots.
- **Internal texture:** an image is used to draw points, spheres and circles; it's a compromise between quality and speed.
- **Nice:** a triangle mesh, between 4 and 630 triangles big, is used to draw each sphere and circles for a nicer quality.
- **Texture:** several 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.
- **Point/Sphere size factor:** factor to be applied to the points, spheres and circles radii to increase the size of the drawn elements.
- **Nice detail:** 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.

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.

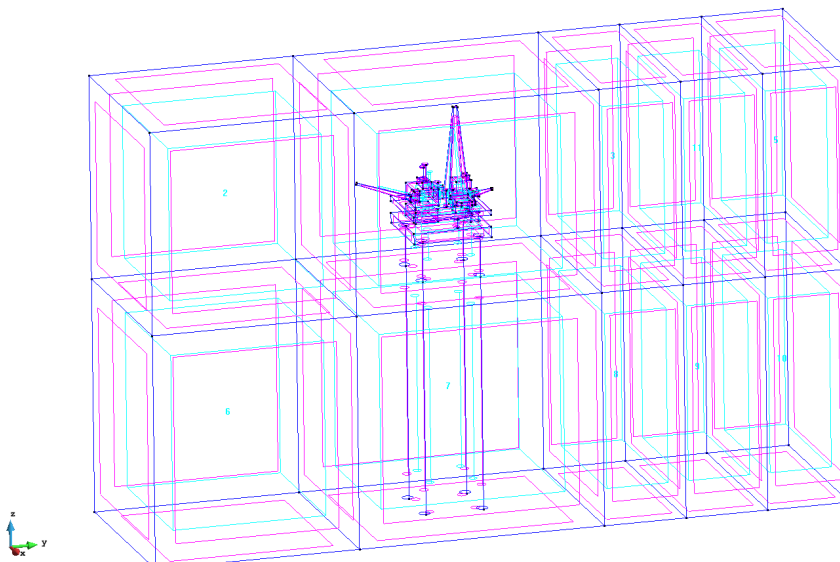
There can be interpolation problems when there are some undefined result values for some nodes.

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, especially in the extremes of the result scale:

- *Exact contour fills* ( 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.

## 6.5.2 Example

### 6.5.2.1 Preparation




Geometry of the platform\_small model.

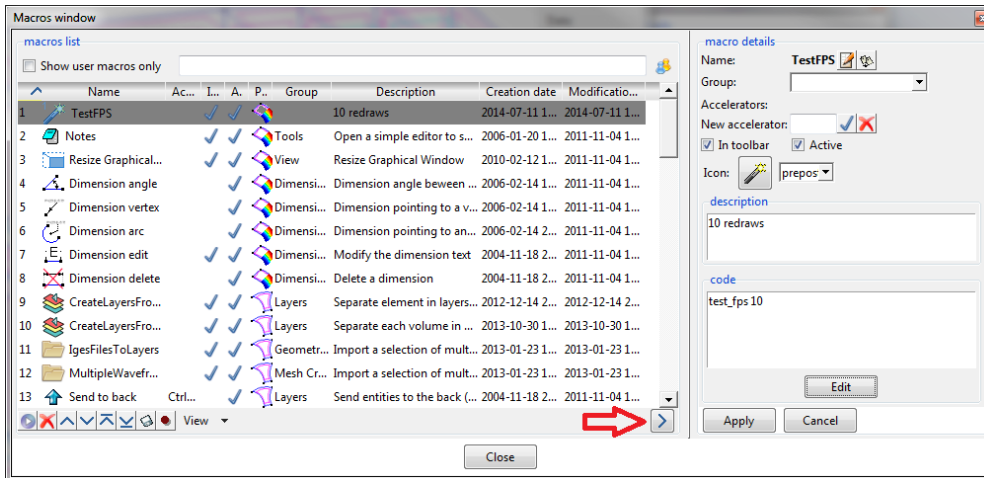
This example is a simulation of waves against a oil platform. The model `platform_small.gid` can be found at [Material location](#).

#### Macro creation:

- 1 create an empty macro, by clicking on the Record macro icon  and the Stop record macro icon

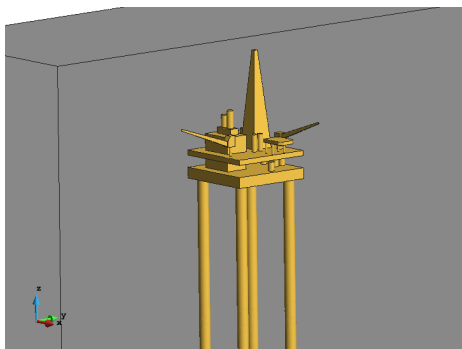


- , both on the macros toolbar;
- 2 edit the macro with the icon  on the same macros toolbar, and enter `test_fps 10` in the code box:

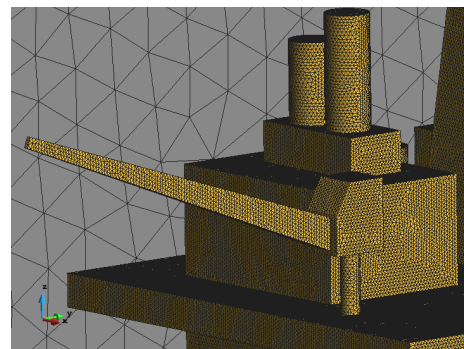


Edit macros window, click on the button pointed by the red arrow to expand the window and enter the code

- 3 load the model;
- 4 go to postprocess;
- 5 select *Options --> Geometry --> Extract boundaries*;
- 6 select *All lines* display style;
- 7 create a new set with the elements of the platform, and change its colour;



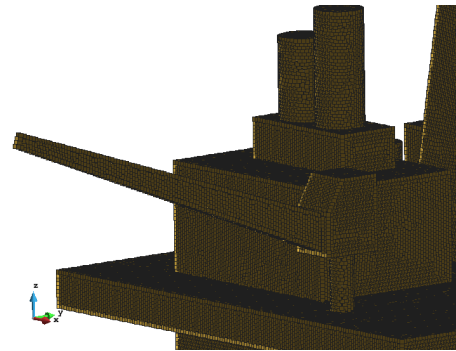
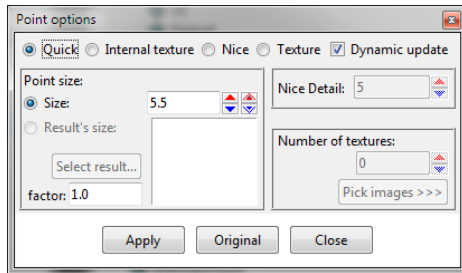
The single triangle mesh of the volume boundary has been separated in the grey box triangle set and the golden platform.



Zoom view showing the refined triangle

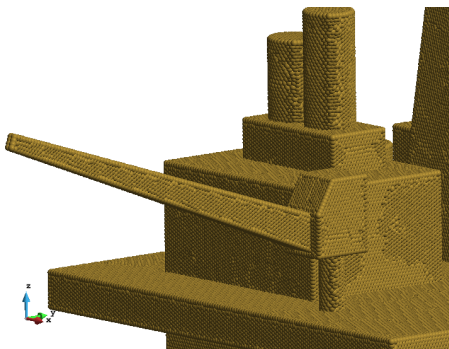
### 6.5.2.2 Point/sphere textures

- 1 switch off all sets except the platform one;
- 2 test the macro, a *Warning* window should appear with a text like this: *Redrawing 10 times ... done: 20 fps. ;*
- 3 change the visualization style to *points*;
- 4 change in *Options --> Geometry --> Point options* to *Quick*; and *Size* to 5.5;
- 5 test the macro;

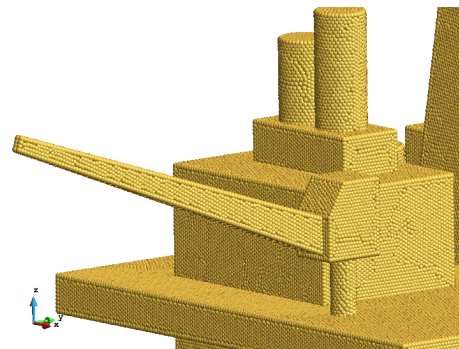


Drawing Quick points on the nodes of the platform mesh

- 6 change in *Options* --> *Geometry* --> *Point options* to *Internal texture* and execute the macro;  
 7 change in *Options* --> *Geometry* --> *Point 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

- 8 Compare the times obtained:

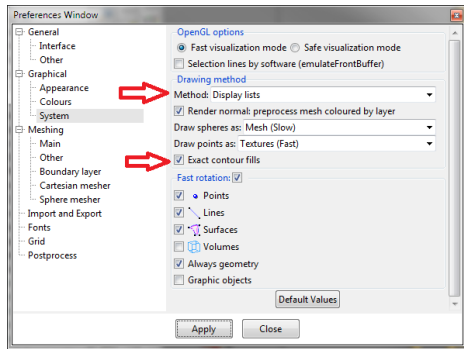
*Quick points*: Redrawing 10 times ... done: 35.8 fps.

*Internal texture*: Redrawing 10 times ... done: 18 fps.

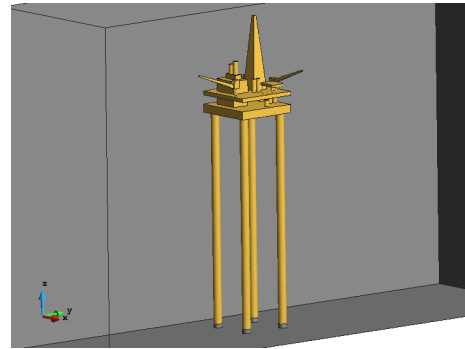
*Nice ( detail = 5)*: Redrawing 10 times ... done: 2.84 fps.

### 6.5.2.3 Contour fill textures

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

- 6 do a Contour Fill of  $|Velocity|$ , and execute the macro;
- 7 disable the Exact contour fills option in Preferences --> Graphical --> System;
- 8 do a Contour Fill of  $|Velocity|$ , and execute the macro;
- 9 Compare the times obtained:
  - Exact contour fills on: Redrawing 10 times ... done: 6.5 fps.
  - Exact contour fills off: Redrawing 10 times ... done: 29 fps.

