Changelog

Several scripts have been slightly changed to cope with multi-track projects. The most important changes are:

- **simplificar** has been changed to make it much faster
- **mallado_regular** has been completely rewritten to allow joining several anchors_carretera.geo file. For the same reason it also creates a file called phys111.txt with the list of surfaces to be included as Physical Surface(111).
- **New scripts** for working with multi-track projects**: join_geos, addt**, **listc**
- Now most of the scripts have an alias name so they can be called easily. For example, procesar_elementstxt_mt now can be executed just writing **p_e**. Alias names are shown on screen (in "Next step" messages).
- Now the same script may behave different if it detects it is run as part of a multitrack project. "Next step" messages report the detected state for execution (Father, son or normal modes)
- **addkml** has been changed. Now it is not necessary a starting number.

Working with multiple tracks. Basic ideas

New scripts are used basically the same way as previous ones. The most important change is that you can use the scripts with several tracks, each one working inside its own folder's structure, and they can be "linked" so the output from the scripts for each track can be later joined, creating a project that includes different/independent tracks, like dead-end detours (aesthetical use) or real alternative driving routes.

There must be a main track. We will call it the FATHER. And there may be several secondary tracks, the SONS.

FATHER:

- Extract all the folders (s0_import, s2_elevation, etc) inside a directory, for example **C:\project_father**
- Copy the kml file for the main route inside s0_import folder
- Create a file C:\ project_father**sons.txt** and write down the full path of each one the sons' directory. One path per line. For example, the contents of sons.txt, if we have 3 sons, should look like:

  C:\project_01
  C:\project_02
  C:\project_03

SONS (for each additional track)

- Extract all the folders (s0_import, s2_elevation, etc) inside its directory, for example **C:\project_01**
- Copy the kml file for the route inside s0_import folder
- Create a file C:\ project_01**father.txt** and write down the full path of the father's directory. There should be only one path. For example:

  C:\project_father

NOTE

To make easy moving the folders to another place in your hard disk you can also do this:

1) Make father and sons be subfolders of the same root folder:

   C:\project\father
   C:\project\son_01
   C:\project\son_02
   C:\project\son_03

2) Then you only have to write folders name inside father.txt and sons.txt. This allows c:\project to be moved to another location in your hard disks and scripts should also work there without the need to change paths inside father.txt and sons.txt
3)

sons.txt
son_01 son_02 son_03

father.txt
father

NOTE

Script create_sons can be used to automatically create the sons' file structures

For example to create 20 sons:
> cd c:\project\father
> create_sons(20)

Or you can use a set of kmls to create the sons, just using the folder where they can be found:
> cd c:\project\father
> create_sons('c:\temp\kmls')

By default create_sons will keep the name of the kmls for the son's folders, but you can also use son01, son02, etc.

> cd c:\project\father
> create_sons('c:\temp\kmls',0)

Creating a project

Once the folder's structure for all the tracks is ready (father.txt and sons.txt have been created) we start working with the father.

1) We start with the father

> cd c:\project_father
> addpath('c:\project_father\scripts')

We use the scripts as usual (s0_import, venue, s2_elevation, s2_elevation_b, s3_road, venue and s1_mesh) **until we run mallado_regular**. After running mallado_regular for the father, we stop working with the father and we start working with the sons.

If sons.txt has been detected, the "Next step" message should always contain the word "FATHER" on the top right corner.

Please note that the grid created inside s2_elevation should be big enough to cover all the tracks: the father and the sons.

2) We go on with the sons

> cd c:\project_01

We use the scripts as usual, following the "next step" messages. Elevation data from the father will be used, so only a few steps will be done with each son (s0_import, venue, s3_road, venue, s1_mesh, s10_split).

If father.txt has been detected, the "Next step" message should always contain the word "SON" on the top right corner.

3) We finish the process with the father

> cd c:\project_father\s1_mesh
> join_geos

A file called joined.geo will be created as the combination of all the anchors_carretera.geo files, from father and sons. joined.geo must be edited with gmsh to fix overlapping meshes and create the final Physical Surfaces for driveable and non-driveable zones. How to do it is explained below.

When joined.geo is finished, we mesh it, save the mesh, we check it (open the .msh file with gmsh and check that all the surfaces are included), and then we use **trocea_malla** inside s1_mesh to go on with the process. Please note that for the new scripts trocea_malla is run inside s1_mesh, no inside s1_mesh\salida.

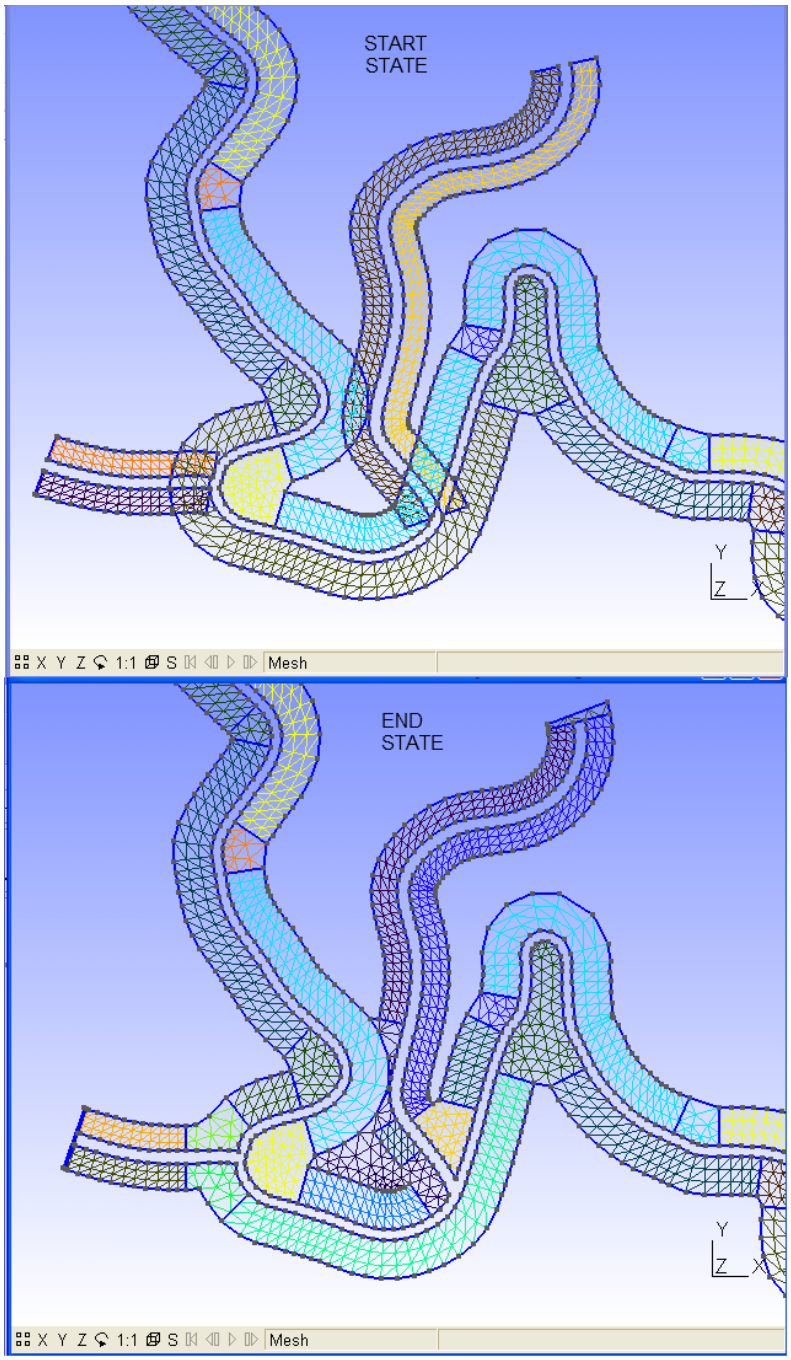
Processing joined.geo with gmsh

When we open joined.geo with gmsh, we will find that the driveable meshes of the sons overlap the father's mesh. That must be fixed by hand, using gmsh and a text editor.

The steps we will follow are:

1) Fix overlapping of driveable zones
2) Create Physical Surface(111)
3) Create non-driveable zone and Physical Surface(222)
4) Final check

1) Fix overlapping



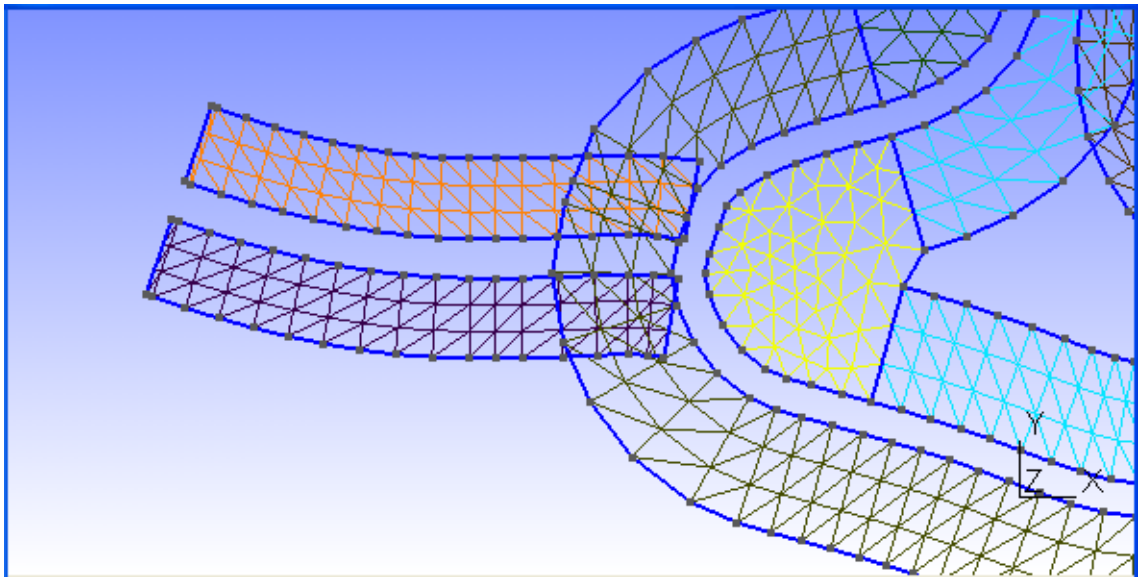
We have three options for fixing overlapping: 0) the fast, 1) the easy, and 2) the not-so-easy ways.

FAST WAY

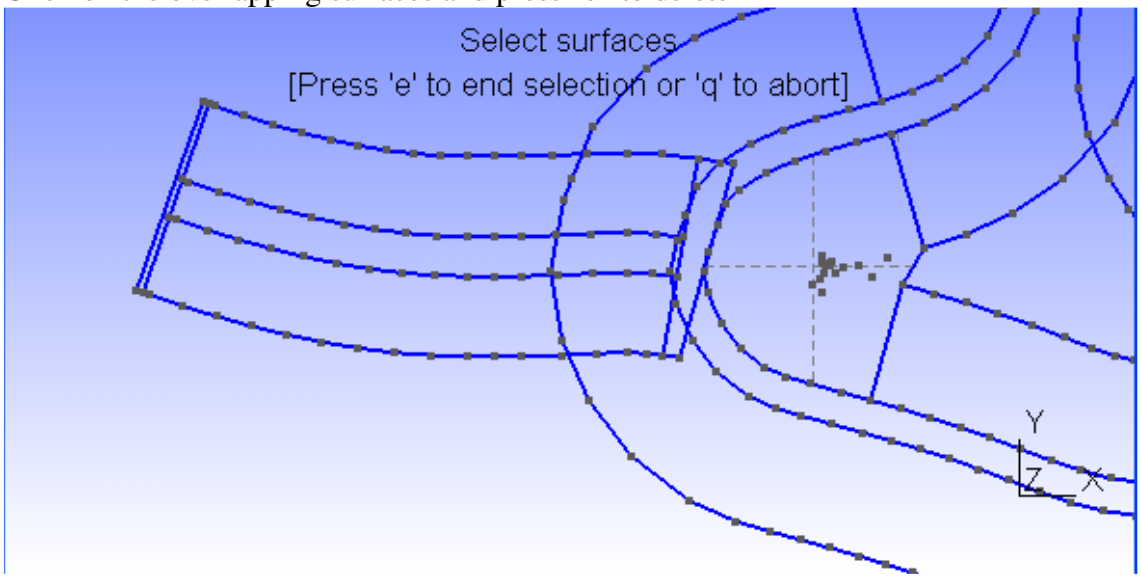
I think this is the best option:
http://btbtracks-rbr.foroactivo.com/t511-gmsh-editing#3345

EASY WAY

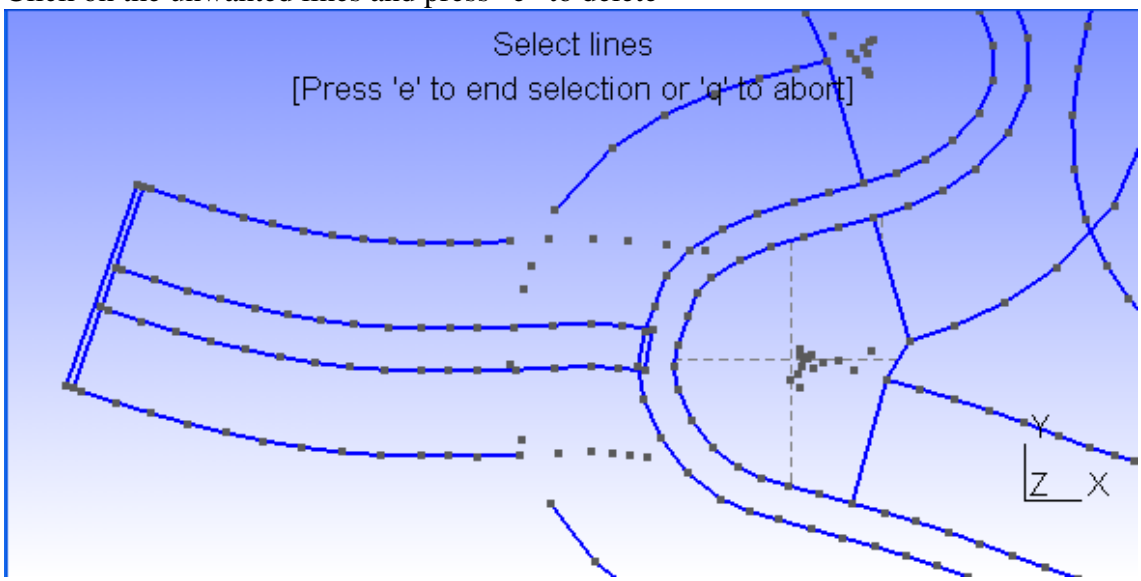
- Delete overlapping surfaces
- Delete unnecessary lines (just for having a clear view)
- Create new surfaces



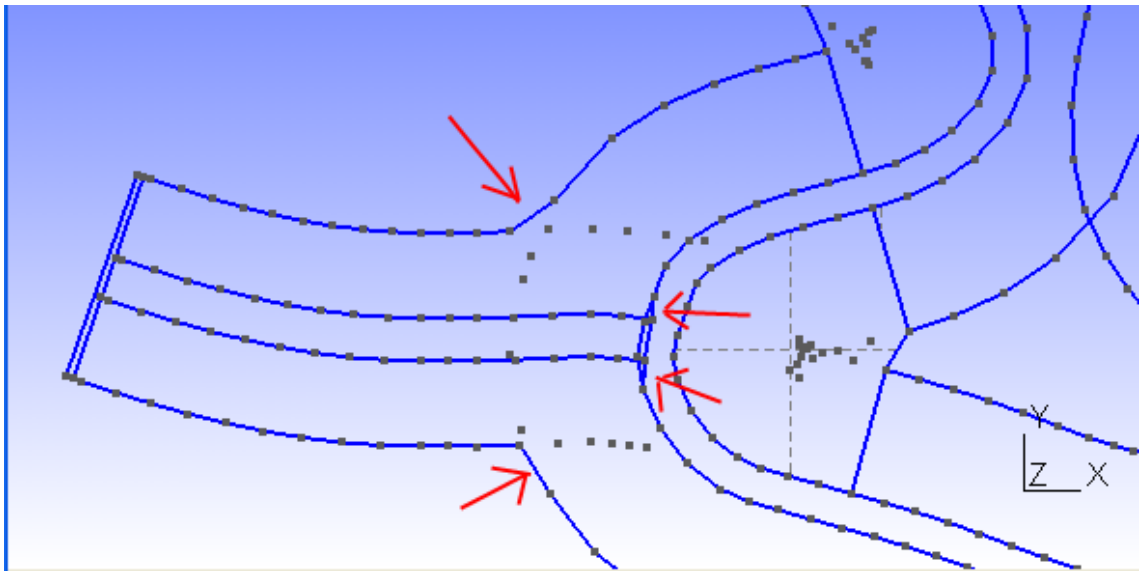
Geometry\Elementary entities\Delete\Surface.
Click on the overlapping surfaces and press “e” to delete



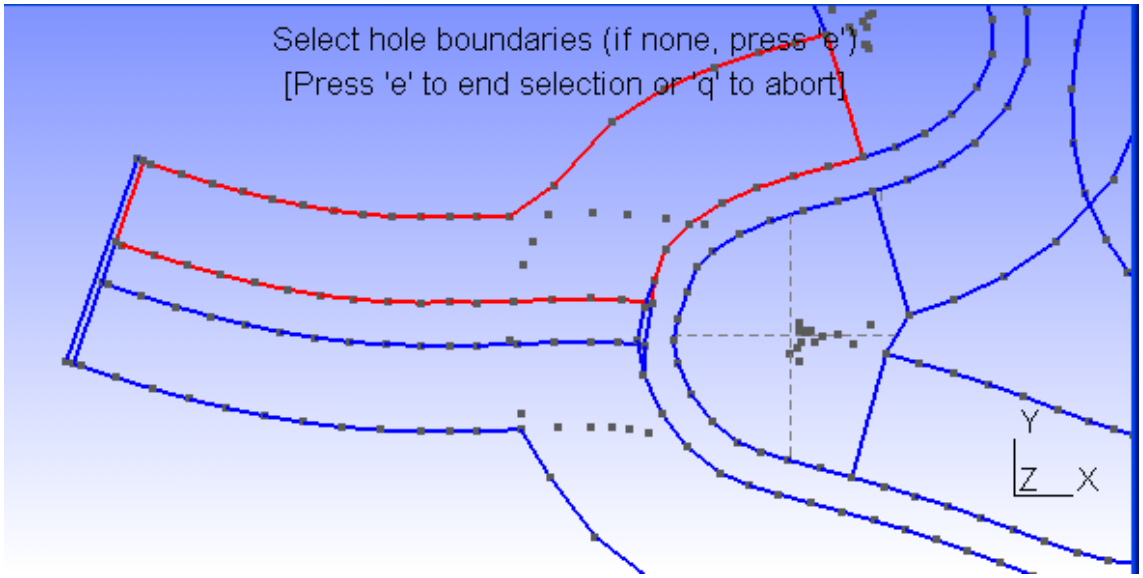
Geometry\Elementary entities\Delete\Line
Click on the unwanted lines and press “e” to delete

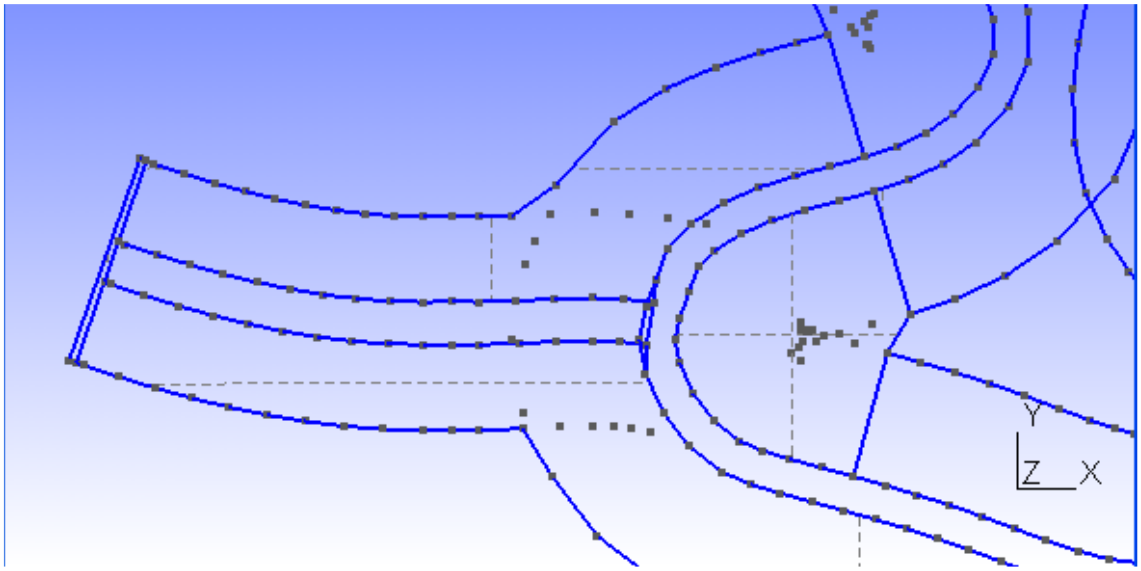


Geometry\Elementary entities\Add\New\Straight Line
Create the lines needed to close the surfaces

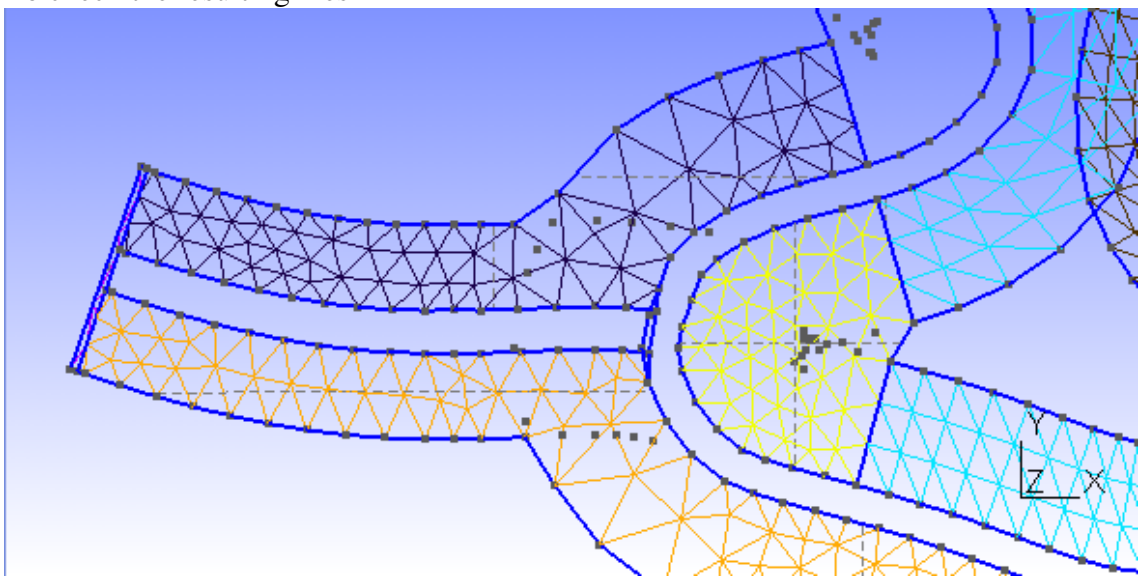


Geometry\Elementary entities\Add\New\Surface
Select the boundary of a surface and press “e”. When all the segments of the boundary are selected, option “u” disappears from screen top message. Repeat for all the surfaces needed, including the track-end surface.





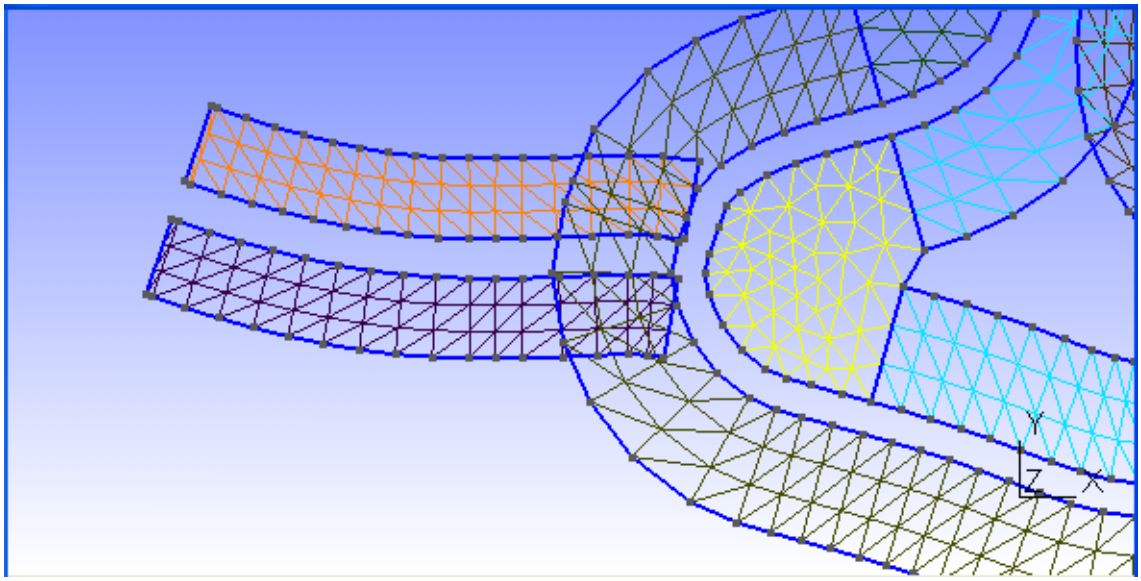
Mesh\2D
To check the resulting mesh



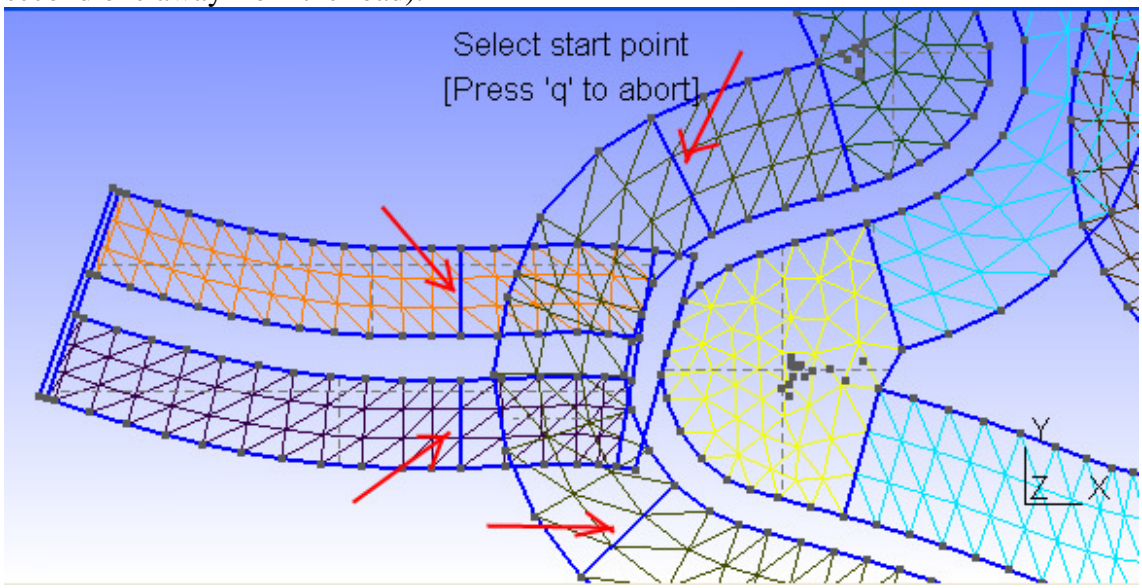
As you can see, with the “easy-way” we lose the regular pattern of the driveable mesh. The not-so-easy way tries to preserve that pattern, with a little extra effort.

NOT-SO-EASY WAY

- Delete overlapping surfaces
- Delete unnecessary lines (just for having a clear view)
- Create regular surfaces
- Create irregular surfaces



Geometry\Elementary entities\Add\New\Straight Line
Create the transversal lines where the regular pattern will stop (first point on road, second one away from the road).



Open joined.geo with a text editor and declare the created lines as transfinite. You can use a for loop to make this step faster (copy-paste the loop and change starting and end line numbers).

```
Line(99992) = {97703, 97742};^M
Line(99993) = {97685, 97724};^M
Line(99994) = {1853, 31853};^M
Line(99995) = {1845, 31845};^M

For h In {99992:99995}
   Transfinite Line(h)= 4 Using Progression 1;
EndFor
```

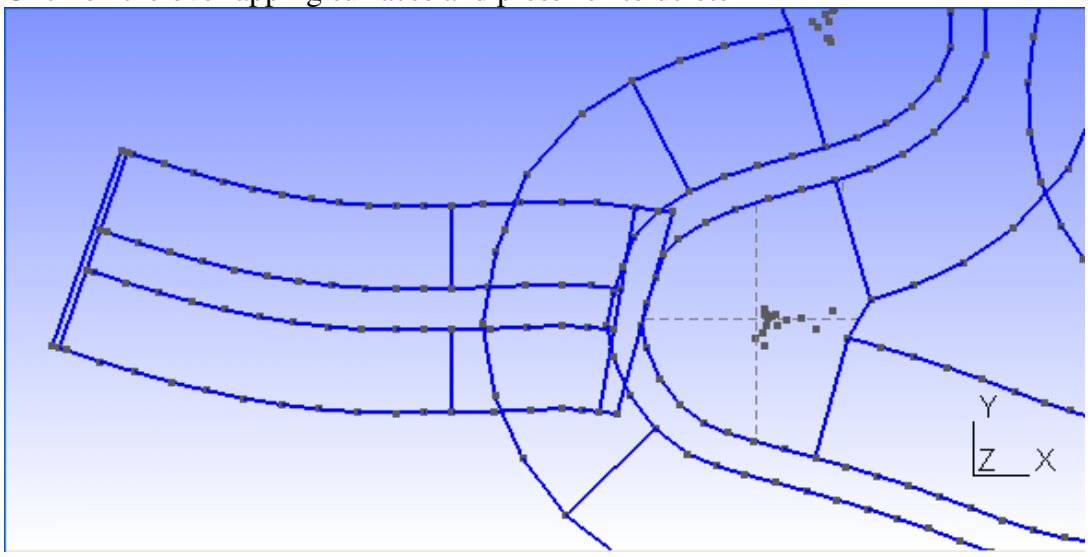
gmsh has written this

we add this

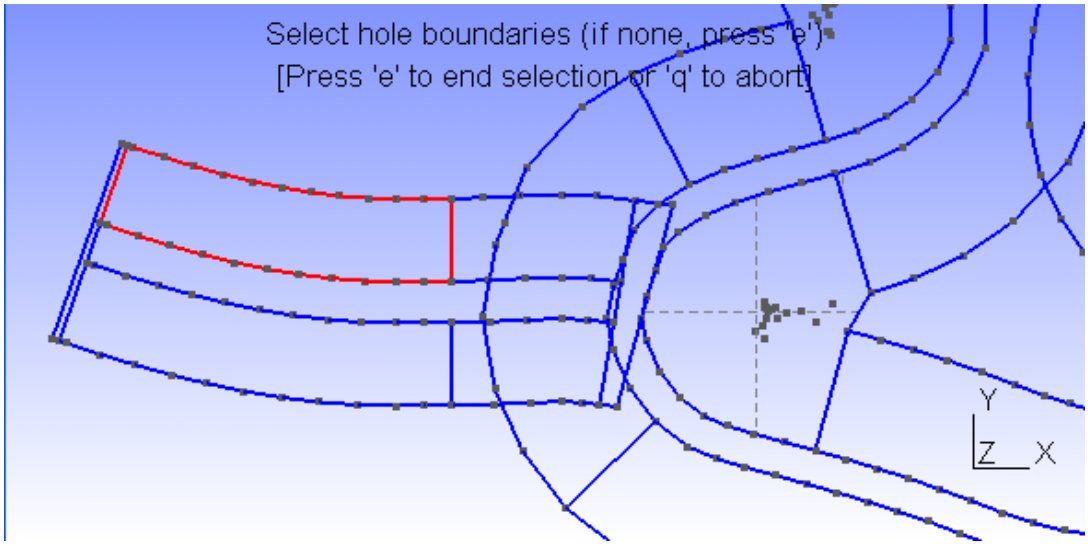
3 panels

In this example the 4 lines created had 3 panels, so they shared the “for loop”. Create this lines in groups of the same number of panels, so they can share the for loop. If you apply a wrong number of panels to a line, gmsh will refuse to create a mesh with a regular pattern for that surface.

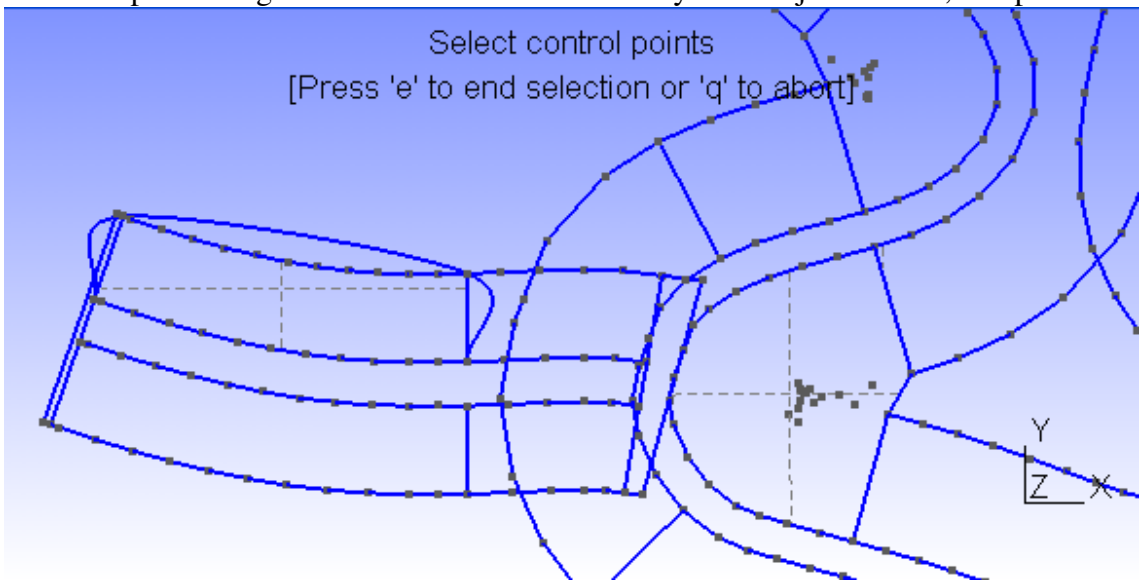
Geometry\Elementary entities\Delete\Surface.
Click on the overlapping surfaces and press “e” to delete



Geometry\Elementary entities\Add\New\Plane Surface
Select the boundary of one of the surfaces with regular pattern that need to be created, and press “e”.



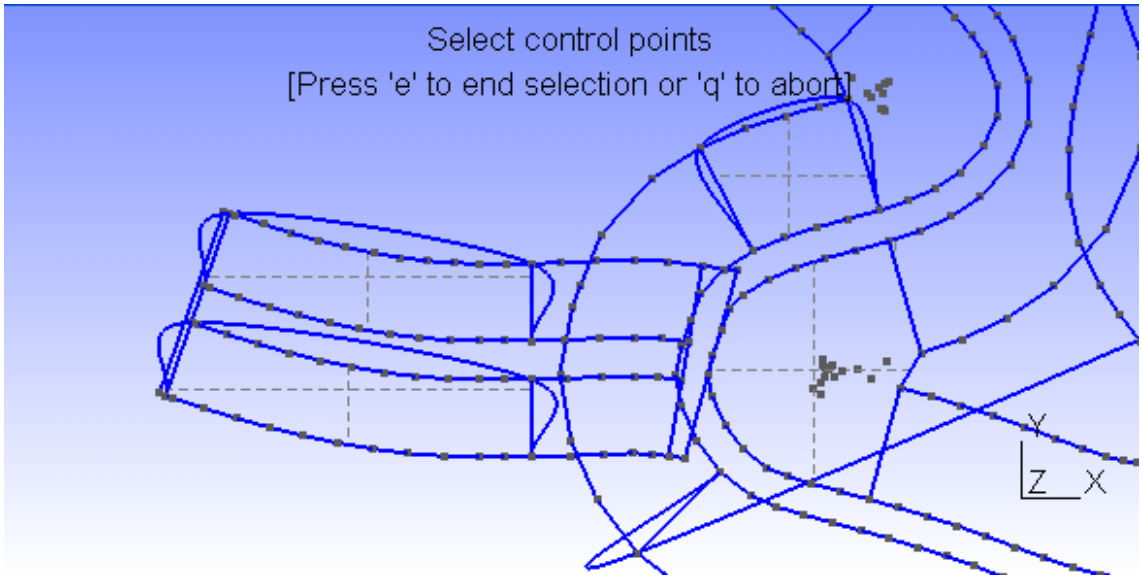
Geometry\Elementary entities\Add\New\Spline
Create a spline using the four corners of the surface you have just created, and press “e”



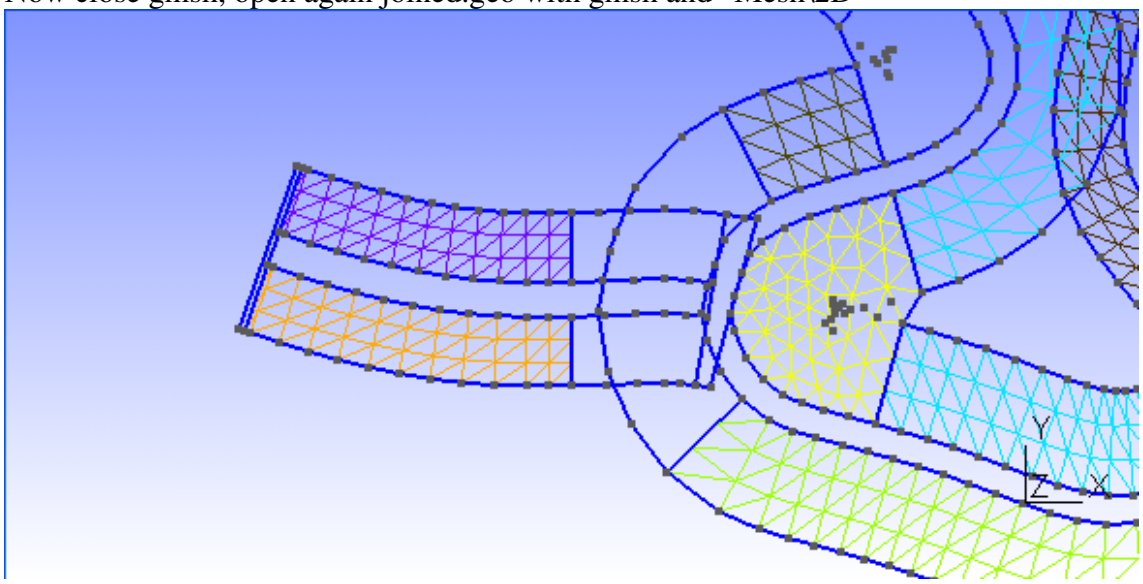
Now run script “addt” inside s1_mesh directory

```
octave-3.2.3.exe:7:f:\vmdata\tramos\tudons\tudons_father\s1_mesh
> addt
Leyendo el fichero salida\joined.geo
Cerrando el fichero
Escribiendo el fichero salida\joined.geo
Cerrando el fichero
octave-3.2.3.exe:8:f:\vmdata\tramos\tudons\tudons_father\s1_mesh
>
```

Repeat the procedure for all the regular surfaces: create surface, create spline, run “addt”. Don’t change the order of those operations.


Select control points
[Press 'e' to end selection or 'q' to abort]
Y
Z
X

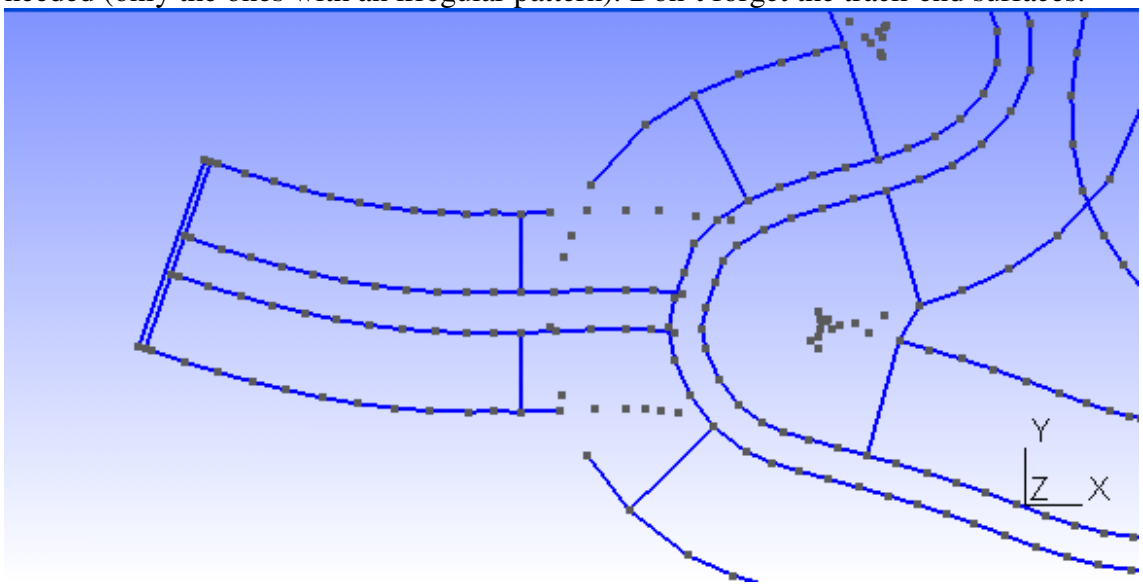
Now close gmsh, open again joined.geo with gmsh and “Mesh\2D”

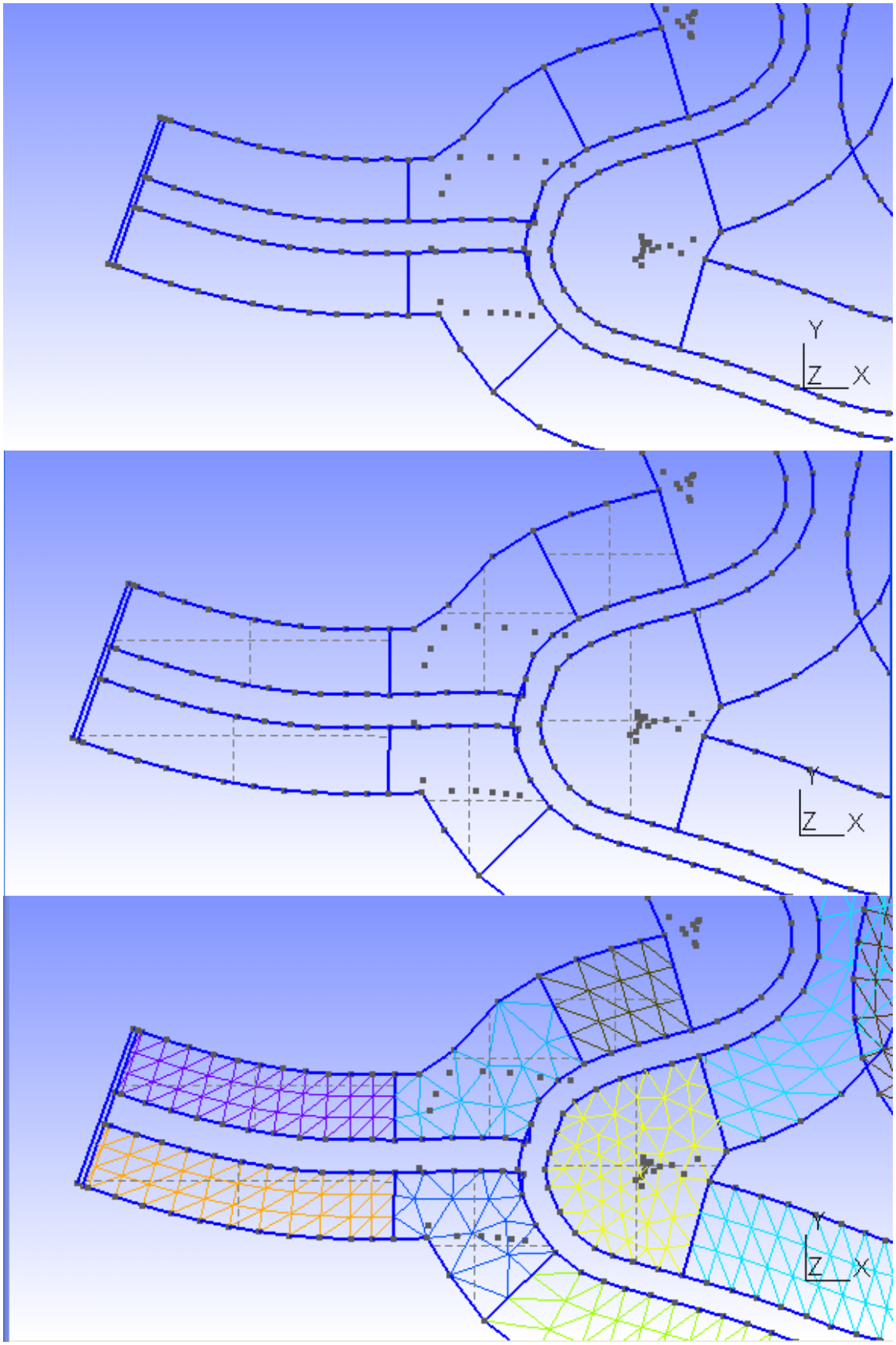


As you can see, the created surfaces have a regular pattern.

Now you can do the same steps for all the detours, or you can proceed to finish this one.

If you want to do complete all the steps for this detour, just proceed as explained in “the easy-way”: delete unwanted lines, create the needed ones and create the plane surfaces needed (only the ones with an irregular pattern). Don’t forget the track-end surfaces.




Y
Z
X
Y
Z
X
Y
Z
X

2) Create Physical Surface(111)

Script join_geos already included a list of the driveable surfaces inside joined.geo. But we have deleted some of them and created a few. If you open joined.geo with a text editor, all the Plane Surface declarations created after the text line

//******** END OF JOINED .GEO FILES**********

must be included in the Physical Surface(111) list. So copy the Physical Surface declaration to the end of the file and add all the new surfaces to the list. To help in this task, you can run a script called "**listc**" that creates a file called "**listc.geo**", with the list of plane surfaces created after the text line shown above.

```
octave-3.2.3.exe:11:f:\vmdata\tramos\tudons\tudons_father\s1_mesh
> listc
Leyendo el fichero salida\joined.geo
Cerrando el fichero
Escribiendo el fichero salida\listc.geo
Cerrando el fichero
octave-3.2.3.exe:12:f:\vmdata\tramos\tudons\tudons_father\s1_mesh
>
```

3) Create non-driveable zone and Physical Surface(222)

Non-driveable zone and Physical Surface must be created as usual: an external boundary with a hole boundary that is the outside limit of the driveable zone. When the surface is created the user has to add the Physical Surface(222) definition

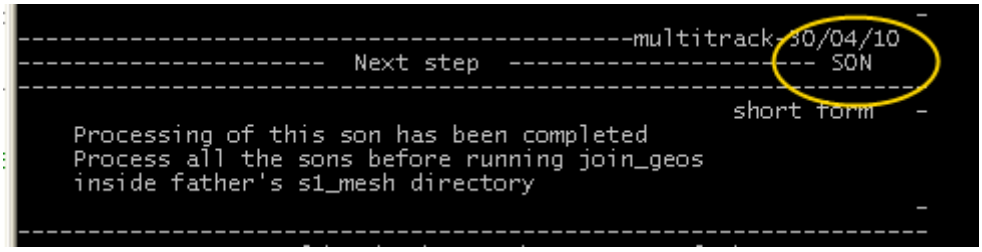
4) *Final check*

Once you think joined.geo is ready, open it with gmsh, Mesh\2D, File\Save mesh and open joined.mesh with gmsh. Check the mesh: look for all the track-end surfaces, check all the route looking for missing surfaces. If everything is ok go on with the process (trocea_malla).

NOTES:

1) May be you want to **delete the documentation folder for the sons** to save disk space

2) If you don't see the **key word SON or FATHER on the "Next step message"**, don't go on with the process until you correct the problem



3) **Crossings** can be processed easier if they are treated as **2 separate branches**.So when creating the kml files for them just create 2 kmls instead of 1.

4) There is a script called **process_sons** that can be used to process all the sons if there are a lot of them and they share parameters like road width or mesh width and number of panels. File scripts\process_sons.m can be easily edited to fit your needs. To use this script run first "cd C:\project_father". Please note that this script uses the first .kml file it finds inside each s0_import directory (so save there only one .kml file).

5) When you create or delete an object with gmsh, this program annotates the change immediately inside the .geo file. If you want to undo steps, you can close gmsh, open the .geo file with a text editor and remove the unwanted operations.

6) With gmsh you can't delete a point that belongs to a line. You can't delete a line that belongs to a surface. If you want to remove a line that belongs to a surface, first remove the surface.

7) Can I have a "island" of non-driveable terrain, completely surrounded by driveable terrain? I believe poner_muro will fail if that situation is given. May be MeshLab has also problems to cope with that. I will try to change poner_muro as soon as I can to handle that situation. For the moment, you can create that surface as driveable and later with BTB split the terrain and change the properties. Nevertheless in that case no walls would be automatically created for that "island".

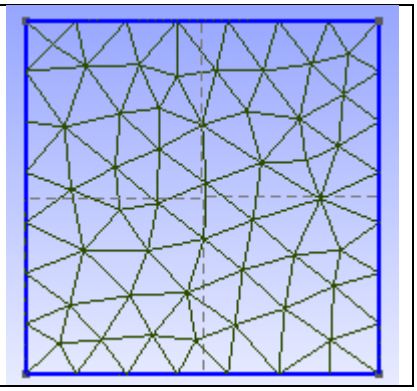
8) Don't run list after creating the Plane Surface for the non-driveable zone. If you do that and use the file, remember to remover that surface from the list or you will get that terrain zone duplicated inside your project.

9) I have processed joined.geo and now I want to add another track to the project, can I do that without redoing all that work done with gmsh? I think it is possible. Start by doing a backup. Then remove the line with text "**END OF JOINED .GEO FILES"** from joined.geo and insert at the final part of the file the anchors_carretera.geo created for the new son. Try to open it with gmsh. If it doesn't work, ask me.

10) What is the mission of script addt? Why do I create a spline, if I don't want a spline? Let's make clear that point. How do we create a surface with a regular pattern?

Example:

```
Point(1) = {-1, 1, 0}
Point(2) = {-1, -1, 0}
Point(3) = {1, -1, 0}
Point(4) = {1, 1, 0}
Line(1) = {1, 4}
Line(2) = {4, 3}
Line(3) = {3, 2}
Line(4) = {1, 2}
Line Loop(5) = {2, 3, -4, 1}
Plane Surface(6) = {5}
```

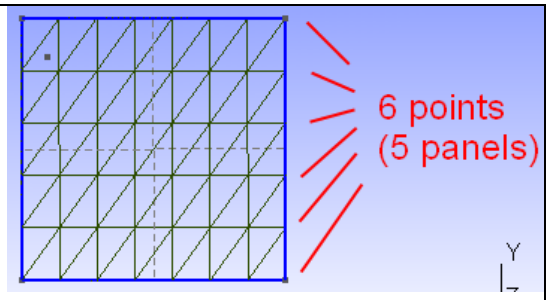


If we want to create a horizontal regular pattern, we must use Tranfinite lines (setting the number of points on them) on left and right lines and also declare the surface as Transfinite.

```
Point(1) = {-1, 1, 0};
Point(2) = {-1, -1, 0};
Point(3) = {1, -1, 0};
Point(4) = {1, 1, 0};
Line(1) = {1, 4};
Line(2) = {4, 3};
Line(3) = {3, 2};
Line(4) = {1, 2};
Line Loop(5) = {2, 3, -4, 1};

Transfinite Line(4)= 6 Using Progression 1;
Transfinite Line(2)= 6 Using Progression 1;

Plane Surface(6) = {5};
Transfinite Surface(6)={1,2,3,4};
```

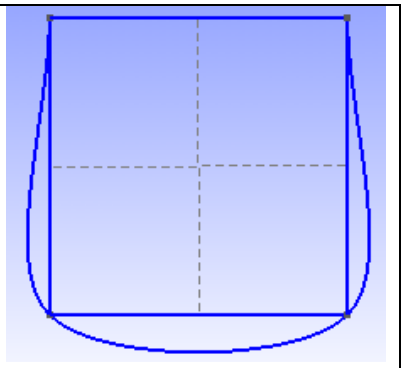


Script "addt" (add transfinite) looks for the last occurrence of "Plane Surface" on joined.geo and if after the Plane Surface there is a spline declaration, it **changes the Spline for a Transfinite Surface** declaration using the same number as the plane surface and the same points as the spline list. For example, if the contents of joined.geo is:

```
Point(1) = {-1, 1, 0};
Point(2) = {-1, -1, 0};
Point(3) = {1, -1, 0};
Point(4) = {1, 1, 0};
Line(1) = {1, 4};
Line(2) = {4, 3};
Line(3) = {3, 2};
Line(4) = {1, 2};
Line Loop(5) = {2, 3, -4, 1};

Transfinite Line(4)= 6 Using Progression 1;
Transfinite Line(2)= 6 Using Progression 1;

Plane Surface(6) = {5};
Spline(7)={1,2,3,4};
```



and we run “addt”, joined.geo would be changed to have the same contents and that shown on the previous code (that with the Transfinite Surface declaration).

That is the reason why the order of execution should always be: create the surface, create the spline and run “addt”. If you forget to run “addt” you can always open joined.geo later with a text editor, search for “Spline” and change it by hand.

If we don’t want to use “addt”, we can create the Plane Surface with gmsh, and then without closing gmsh open joined.geo with a text editor, add the Transfinite Surface declaration by hand, writing the numbers of the corner points of the plane surface that we want to give a regular pattern, and save the file. Next time we mesh we would see the result.