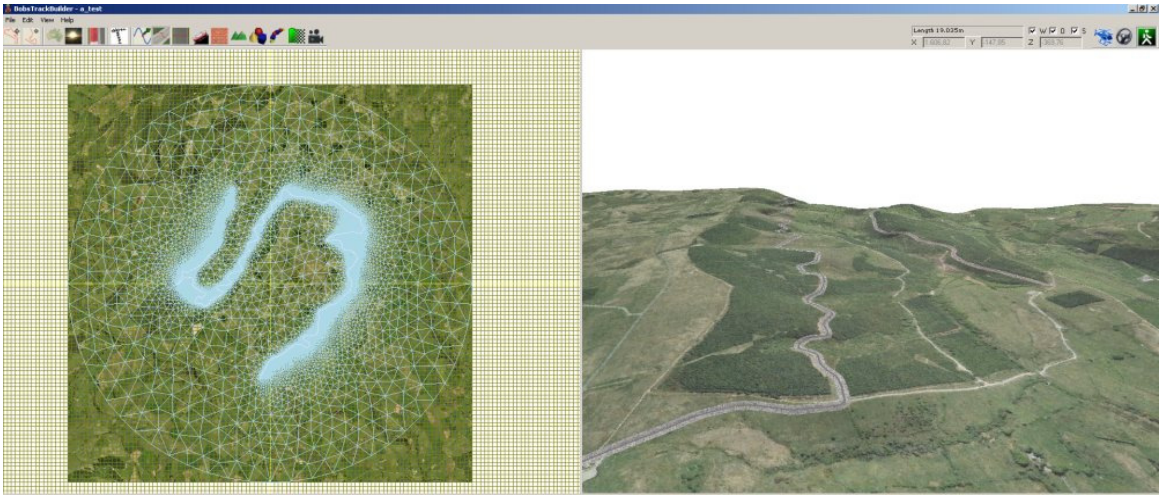
Zaxxon's method guide

Index

I SUMMARY

I.1 What is "zaxxon's method"?

Basically it is "quick" way for creating a base BTB project, based on available elevation data. Once you are used to the method you can have a BTB project ready to be tuned using BTB. You can have terrain and roads created from real data, and if you want, textured with real satellite images.



A simplified summary of the method, as long as there are several ways for doing the steps:

1. Everything starts with the kml of your route. It is converted to BTB coordinates
2. You get elevation data enough to cover your road and all your terrain
3. You give your road an elevation profile according to the elevation data you have (you can manually change it)
4. Then you define the boundaries for the terrain around your road and you create a mesh for that terrain.
5. You give elevation to your terrain using the elevation data you have
6. You create invisible walls to prevent the car from falling out of the terrain (only for RBR)
7. You split your road into several segments for efficiency
8. You split your terrain using a m x n grid (also for efficiency)

9. You create the Venue.xml, a file that can be read by BTB to start working with your project

You can also automatically include background images in your project. May be only for using them as a visual reference or you can use them as texture for your terrain zones.

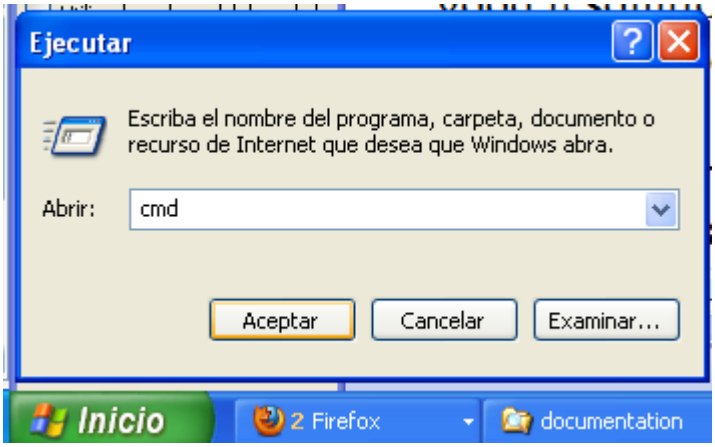
II INSTALLING THE SOFTWARE

For using the scripts you need:

1. The folder structure and the scripts
2. Octave 3.2.3
3. The graphical interface (GUI) for the scripts
4. Install of gmsh
5. WP.zip

II.1 The folder structure and the scripts

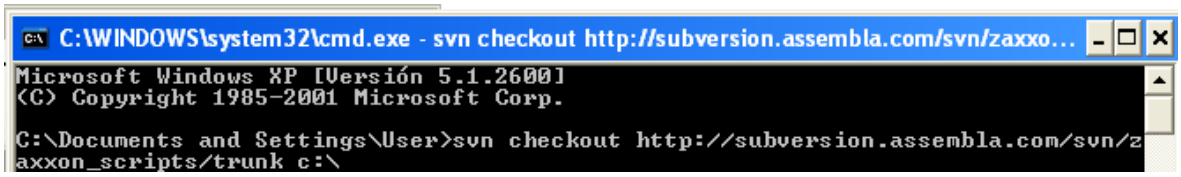
You need Subversion (SVN) to dowload the files. You can found free possibilities visiting: http://subversion.apache.org/packages.html (for example http://www.sliksvn.com/pub/Slik-Subversion-1.7.2-win32.msi from http://www.sliksvn.com/pub)

Once you have installed the Subversion client, on your Windows system open a console window: Start->Run-> "cmd"	

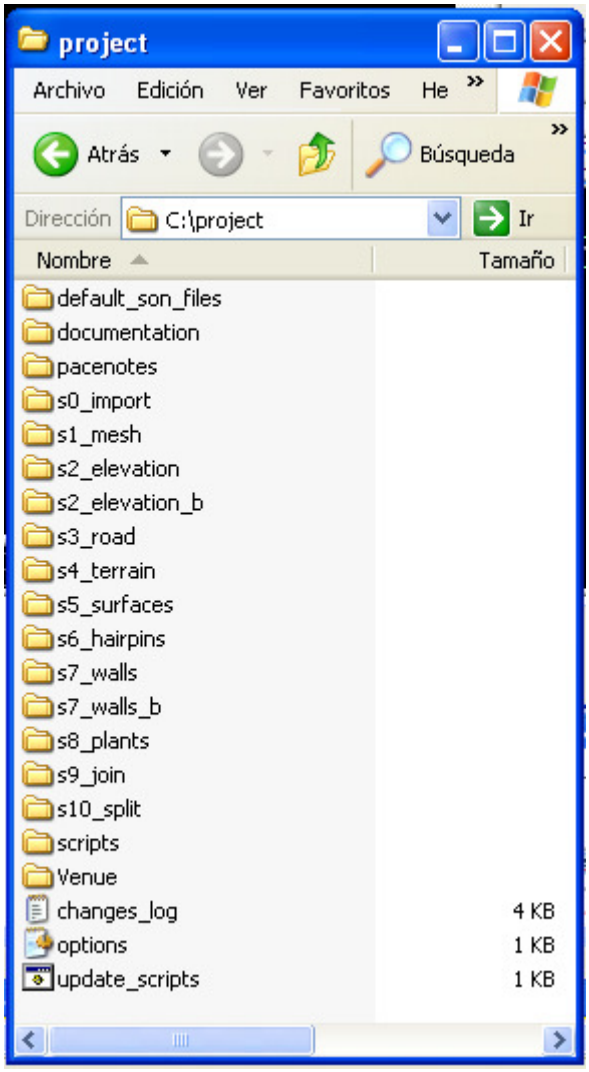
On the console window type:

```
svn checkout http://subversion.assembla.com/svn/zaxxon_scripts/trunk c:\
```

(Or you can download download_scripts.bat file and run it (double click))



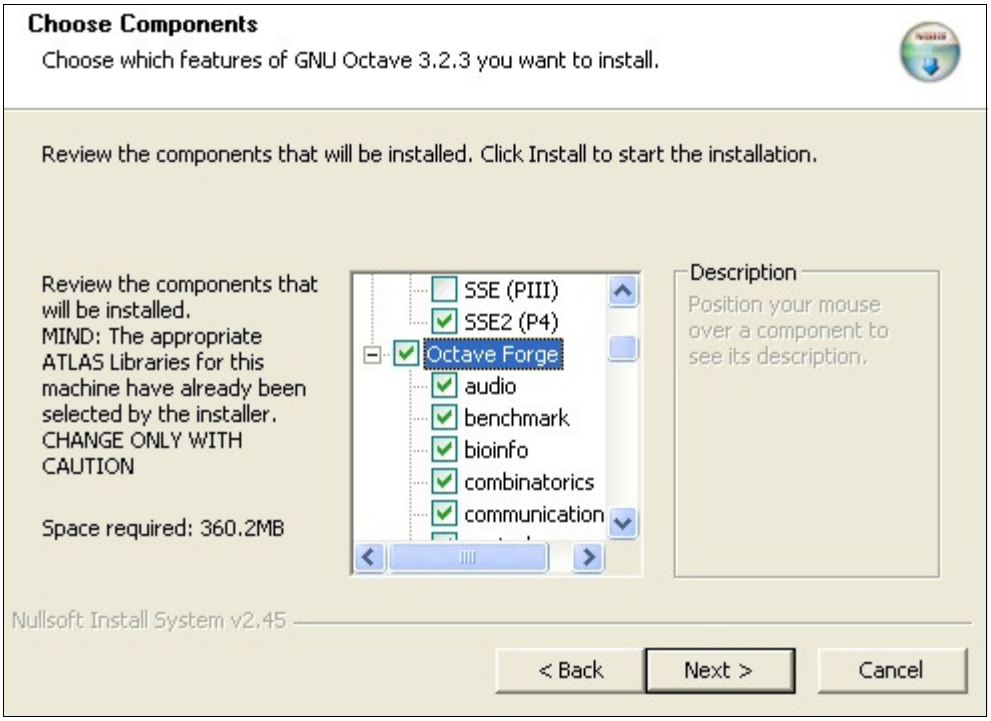
You will get a copy of all the files you need inside **c:\project**



II.2 Octave 3.2.3

You need Octave 3.2.3 (do NOT install a different version of Octave):
http://sourceforge.net/projects/octave/files/Octave_Windows%20-%20MinGW/Octave%203.2.3%20for%20Windows%20MinGW32%20Installer/Octave-3.2.3-2_i686-pc-mingw32_gcc-4.4.0_setup.exe/download

MANDATORY: In the install step **select Octave Forge libraries**



Alter installing Octave 3.2.3 run:

> pkg rebuild -auto communications

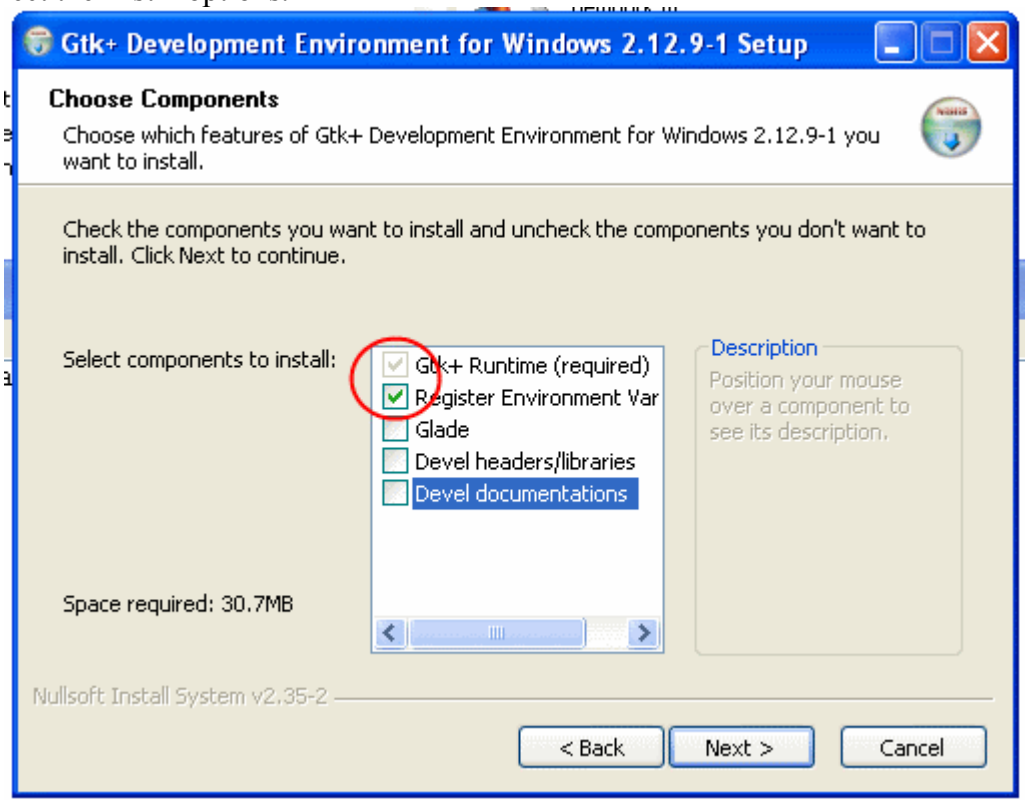
Then close Octave and open it again to start working with it.

II.3 The graphical interface (GUI) for the scripts

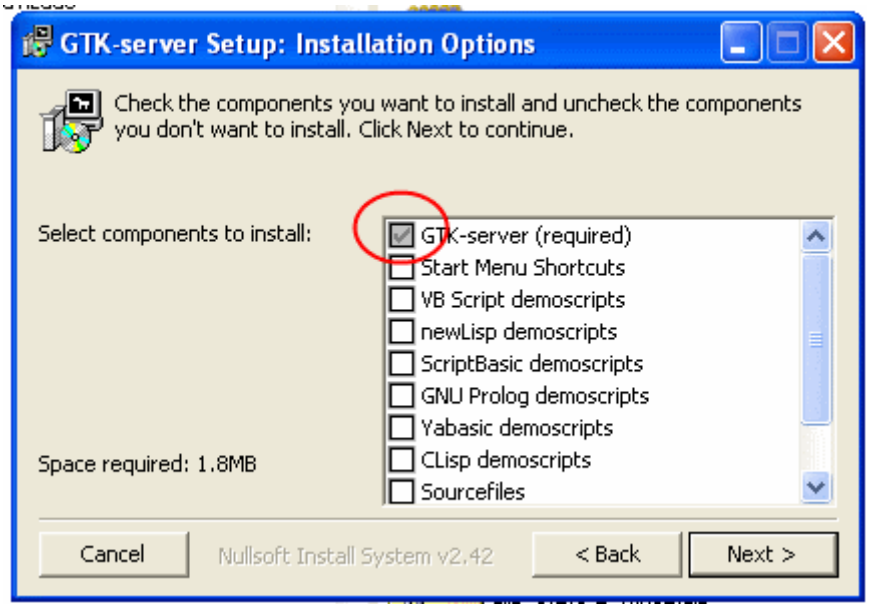
1. Install GTK (gtk-dev-2.12.9-win32-2.exe)

http://sourceforge.net/projects/gladewin32/files/gtk%2B-win32-devel/2.12.9/gtk-dev-2.12.9-win32-2.exe/download

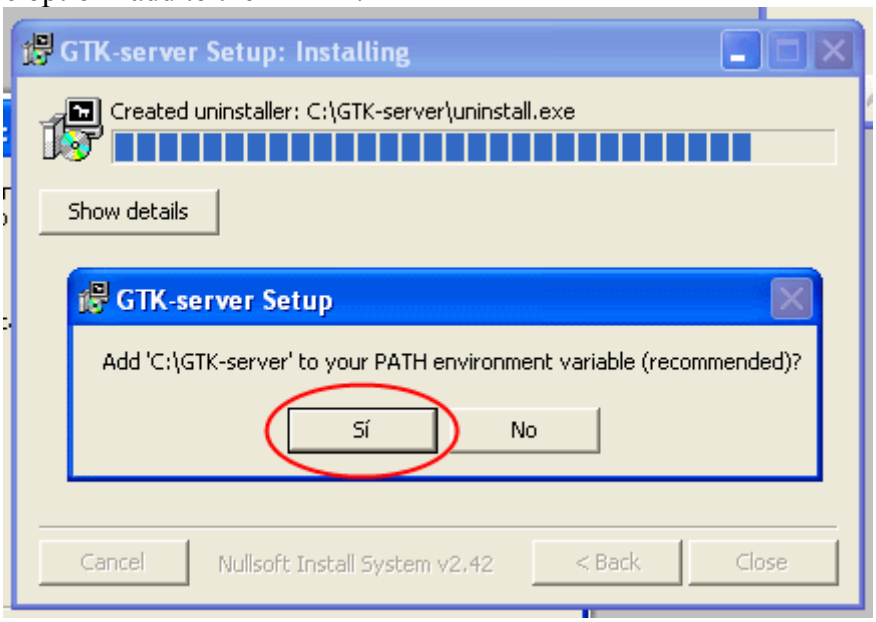
We select the first 2 options:



2. Install gtk-server (gtk-server-2.3.1-installer.exe)
http://downloads.sourceforge.net/gtk-server/gtk-server-2.3.1-installer.exe



We select the option “add to the PATH”:



II.4 Install of gmsh

Gmsh is a finite element grid generator. We will use it for creating a mesh for our roads anf terrain. You must install it. Just download and extract its content on the folder you choose.

Gmsh: http://geuz.org/gmsh/#Download

II.5 WP.zip

The created BTB project will use textures from a xpack called WP.zip. You can get XPack WP.zip from: http://www.mediafire.com/?z2tcoh4wyxj

III BEFORE USING THE METHOD

III.1 One-track or multi-track?

How many .kml files do you have for the routes of your project? If you only have one route you will have a “one-track project” and the use of the scripts will be really easy. If you have more than one track (detours, crossings, alternate routes, etc.) the process is a little bit more complicated, specially when using gmsh.

One-track

For one-track projects you have to do nothing special with the files you have in c:\project. You are ready to start using the scripts.

Multi-track

When working with several tracks the scripts are used quite the same way for all of them, each one working inside its own folder’s structure. The work we do for each of them can can be later joined, creating a project that includes different/independent tracks, like dead-end detours (aesthetical use) or real alternative driving routes.

One important idea is that there must be a main track. We will call it the FATHER. And the secondary tracks will be called the SONS.

To prepare your project to work multi-track:

1. Move the contents of c:\project to c:\project\father
2. If you have e.g 3 sons, open octave and run:

    - > addpath('c:\project\father\scripts')
    - > cd c:\project\father
    - > create_sons(3)

Now you will have the father files inside **c:\project\father**, and the files for the 3 sons in folders **C:\project\son01**, **C:\project\son02** and **C:\project\son03.** That's it: you are ready to start using the scripts.

NOTE: the sons will know they have a father and its location because inside of their folders a file called **father.txt** has been created with the contents "father". The father will know it has 3 sons and their location because inside of its folder a file called "**sons.txt**"has been created, with the contents:

son01

son02

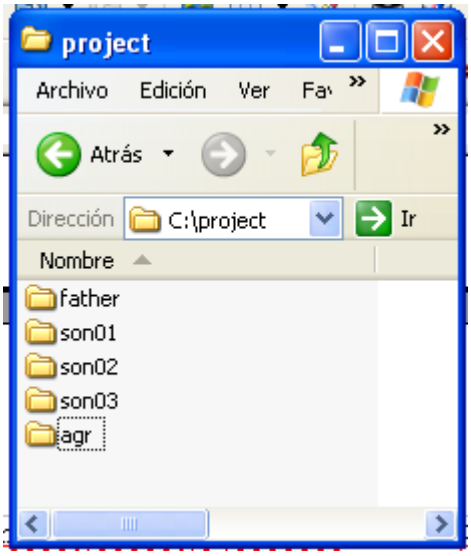
son03

III.2 Source of elevation data?

The most common sources of elevation data are Google Earth and AGR (ASCII Grid) data. The GUI is only ready to use this sources. If you want to use a different one you should read chapter V.

AGR

1) Create a folder named "agr" in the same folder where you have the "project" folder (or the father's and sons' folders in multitrack projects). E.g. if you are working one-track on c:\project, create a folder c:\agr.
2) Copy the needed .agr files to the folder you created in the step 1). Rename those files with extension .agr if needed.
3) Use the scripts as usual.



NOTE: if your .AGR files are too big the scripts may not be able to handle them. You can use script **split_agr** to split them into smaller files, as explained in section V.2

Google Earth

You can get elevation data from Google Earth (read section V.1 for more info). In that case the scripts will create a grid of points, distribute those points into dozens of files (called

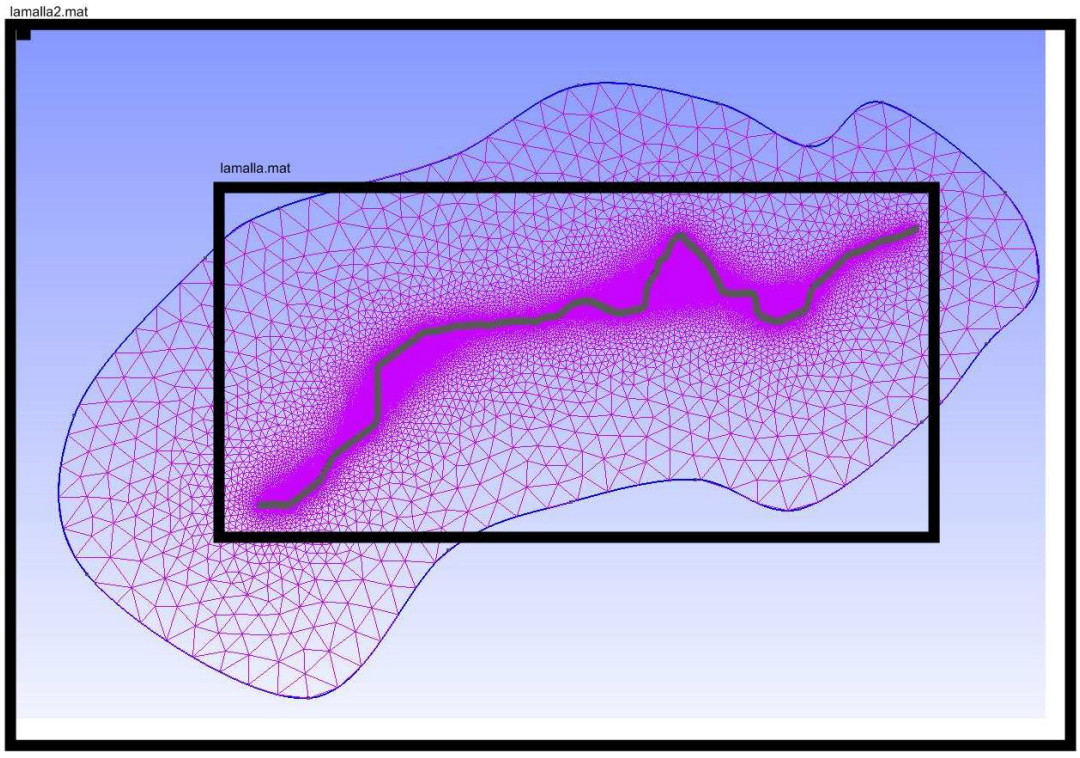
gridXXX.kml) and then ask GE to give them elevation. One file will be processed after another, either by hand if you use BTBLofty or automatically if you use python).

The scripts can work with one grid alone. But it is recommended to use 2. The reason for using 2 grids is that we want good elevation data resolution near the road but **it is useless having 10m resolution 1Km away from the roads** because we will create a mesh with big (75-250m) triangles far from the roads. Therefore we can create two grids and look for elevation data for them: one grid with good resolution at least covering the roads (father and sons) and one with worse resolution covering the whole terrain of the project.

The smaller grid will be created inside s2_elevation folder. For the bigger grid we can use a second folder called s2_elevation_b. The elevation data will be automatically copied to s4_terrain folder as file **lamalla2.mat**. The name for the smaller grid file is **lamalla.mat** ("la malla" means "the grid" in spanish language).

As was said above, the scripts will also work with only one grid (created inside s2_elevation), but as that grid must cover all the terrain of the project, if it has a good spatial resolution, the number of grid points will be high and so the size of lamalla.mat (and, if it is the case, the time needed to get elevation values from Google Earth).

The following picture may be illustrative:

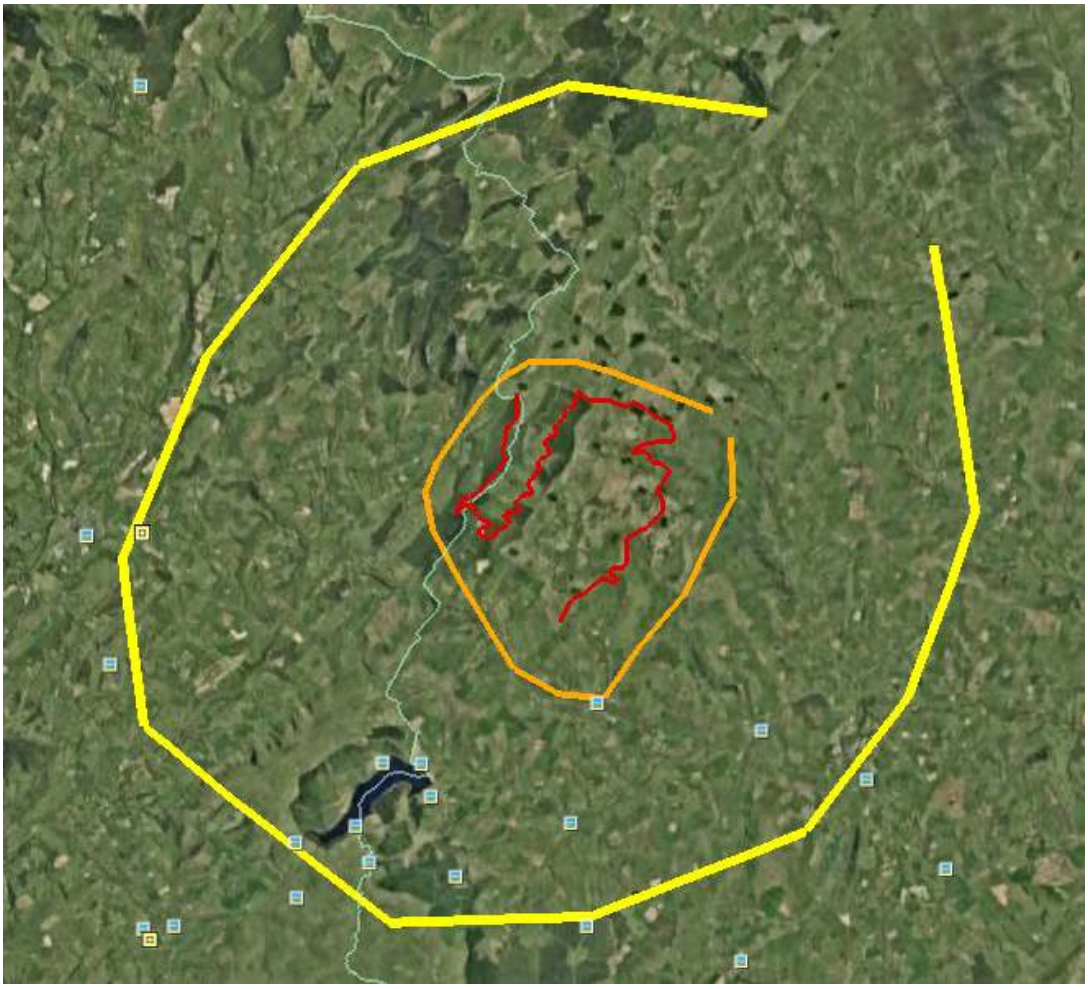


The best way to set the limits of both grids is creating routes with Google Earth. We create 2 more kmls:

Before using the method

1) "**limits.kml**" (orange route) delimiting the terrain for the main route plus all the detours. You must save it inside **s2_elevation**
2) "**limits_b.kml**" (yellow route) delimiting the terrain for all the project. You must save it inside **s2_elevation_b**

The routes don't need to be closed and the scripts will create a couple of grids with enough points to cover both of them. The big grid will cover all the terrain enclosed by the yellow route and the small grid will cover all the terrain enclosed by the orange route. Please note that the color of the route is ignored by the scripts.



IV USING THE SCRIPTS

IV.1 Multi-track project

NOTE: At this point you should have copied the .kml for the father and sons inside their respective s0_import folders. Points' elevation will be ignored.

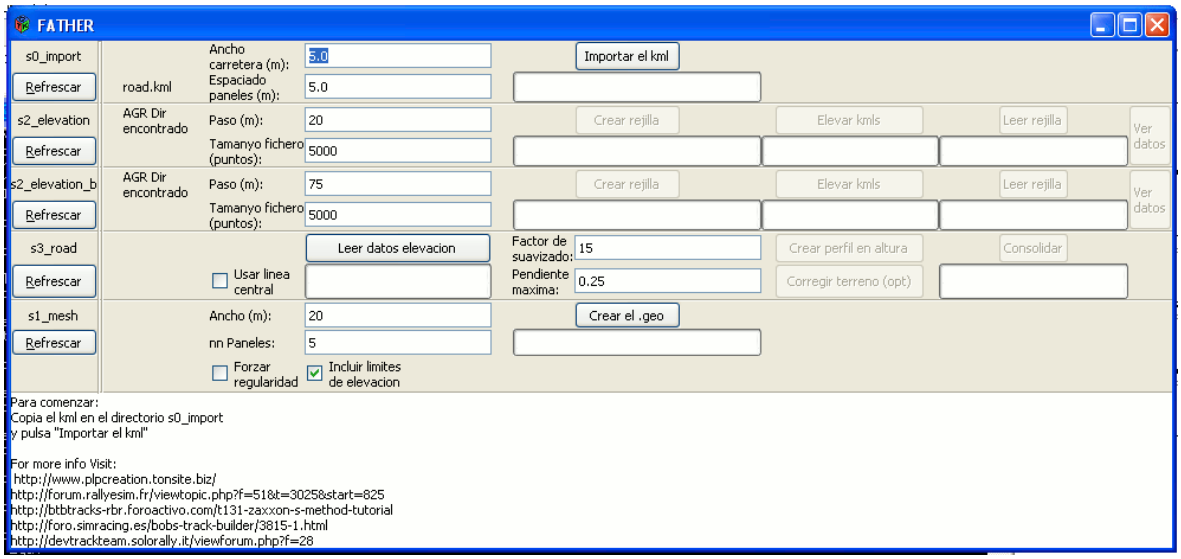
And finally we open octave, we add the script's folder to the octave's search path and run **zgui:**

```
> addpath('c:\project\father\scripts')
> cd c:\project\father
> zgui
```

zgui is the interface for the first half of the scripts (before gmsh)
zgui(1) is the interface for the second half of the scripts (when we have created anchors_carretera.msh)

NORMAL and FATHER have 2 interfaces: zgui and zgui(1). SONs only have zgui, but not zgui(1).

A new window should pop-up in our system:

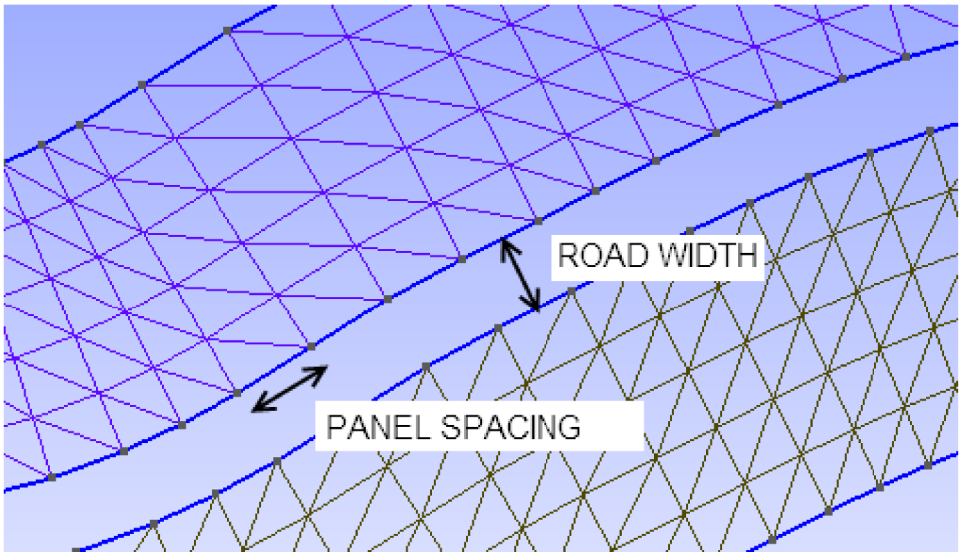


The top bar of the window should read "FATHER". And if we are using AGR data we

should see a message on screen saying “AGR Dir has been found”. In that case steps s2_elevation and s2_elevation_b should be skipped. Otherwise, if you are using Google Earth elevation data a message on screen should say that files “limits.kml” and “limits_b.kml” have been found.

s0_import

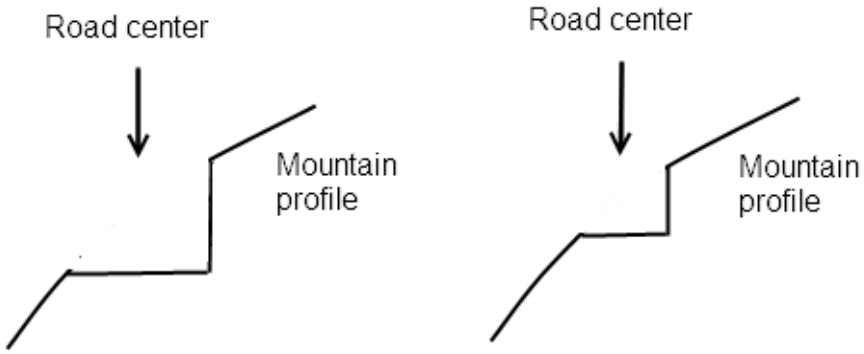
The next step is to click the “Import the kml” button. For this button we need to set a width for our road and how long it will be between the points that link the road and the terrain.



In this step each point of the .kml will be translated to a node of the BTB road.
If the step fails, you should check your kml looking for small loops (points that go backwards instead of forward) or points too close one to each other.

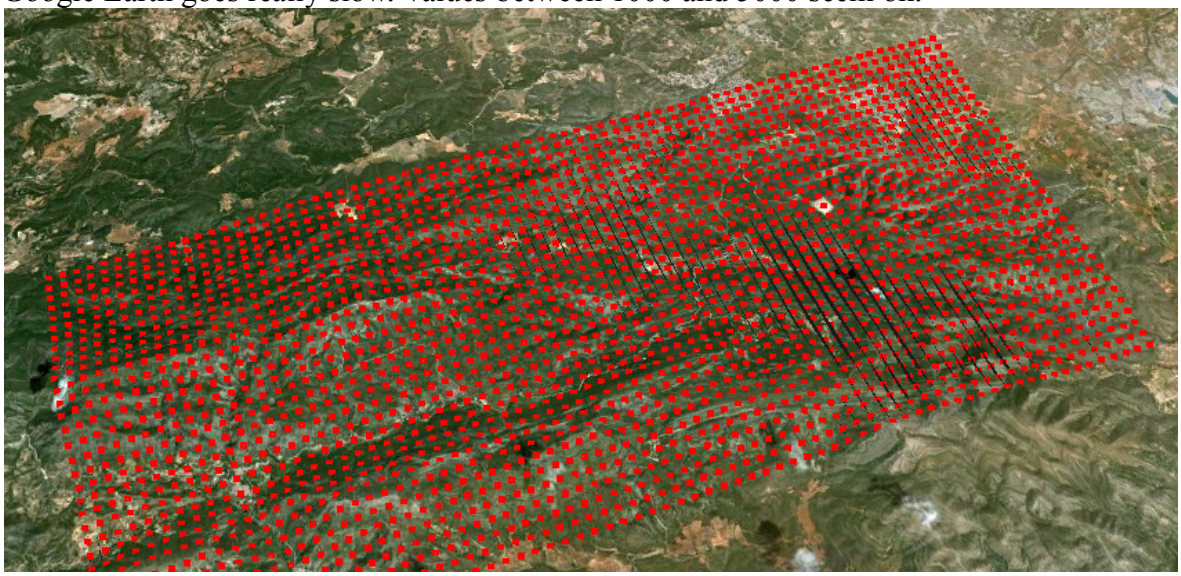
The witdth of the road is important because it affects the mesh we are creating but also

because the elevation profile of the road will take into account its width. When elevation data is not good enough a wide road will be assigned (by the scripts) a lower elevation than a narrower road. The closest value to reality seems to be the best option for the width.



s2_elevation/s2_elevation_b

The next step is getting elevation data if you are using GE as your source of data. In that case you have to set the spacing between points for your grids and click “Create the grid”. The points of the grid will be distributed into several files (called gridXXX.kml) that will be created inside s2_elevation\salida or s2_elevation_b\salida. The file size is the amount of points included in each file. 5000 is the maximum limit. If a value greater than 5000 is used Google Earth goes really slow. Values between 1000 and 5000 seem ok.

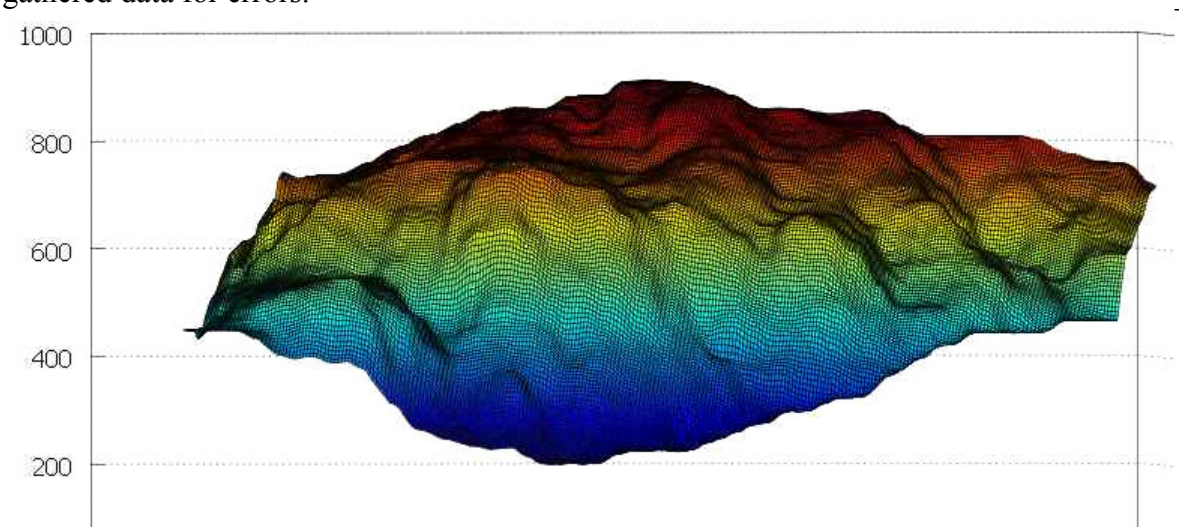


Now, if you have installed python (read section V.1) you can click on button “Raise the kmls”. Otherwise you will have to raise them using another application, as those listed in section V.1.

When the process finishes each gridXXX.kml file has been processed and a

gridXXX_relleno.kml with the same points but with altitude has been created. The next step is reading the gridXXX_relleno.kml files and creating the data matrix that the next scripts will be using. Click on the “Read grid” button. A file called lamalla.mat will be created inside the s2_elevation\salida folder.

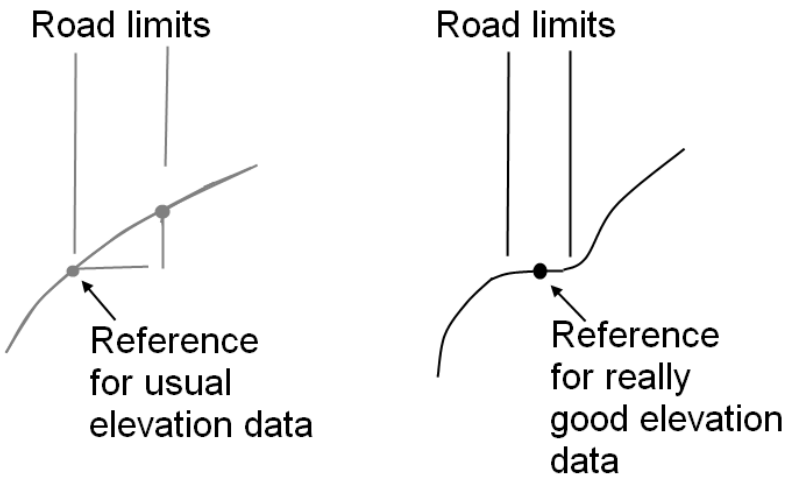
It may be a good idea at this point to click on the “View data” button so you can check the gathered data for errors.



s3_road

Now we want to give elevation to the road. For usual elevation data, the first step is getting the elevation of the terrain at the points located at both sides of the road (those points are called “anchors”). When the terrain has a different elevation on both sides of the road, the lowest of the 2 points will be used as a reference to give elevation to the road. Nevertheless, if we had extremely good elevation data we could use the elevation of the terrain at the central line of the road as reference. This last case may be the case for LiDAR data.

If you have any doubts DO NOT mark “Use the central line”.

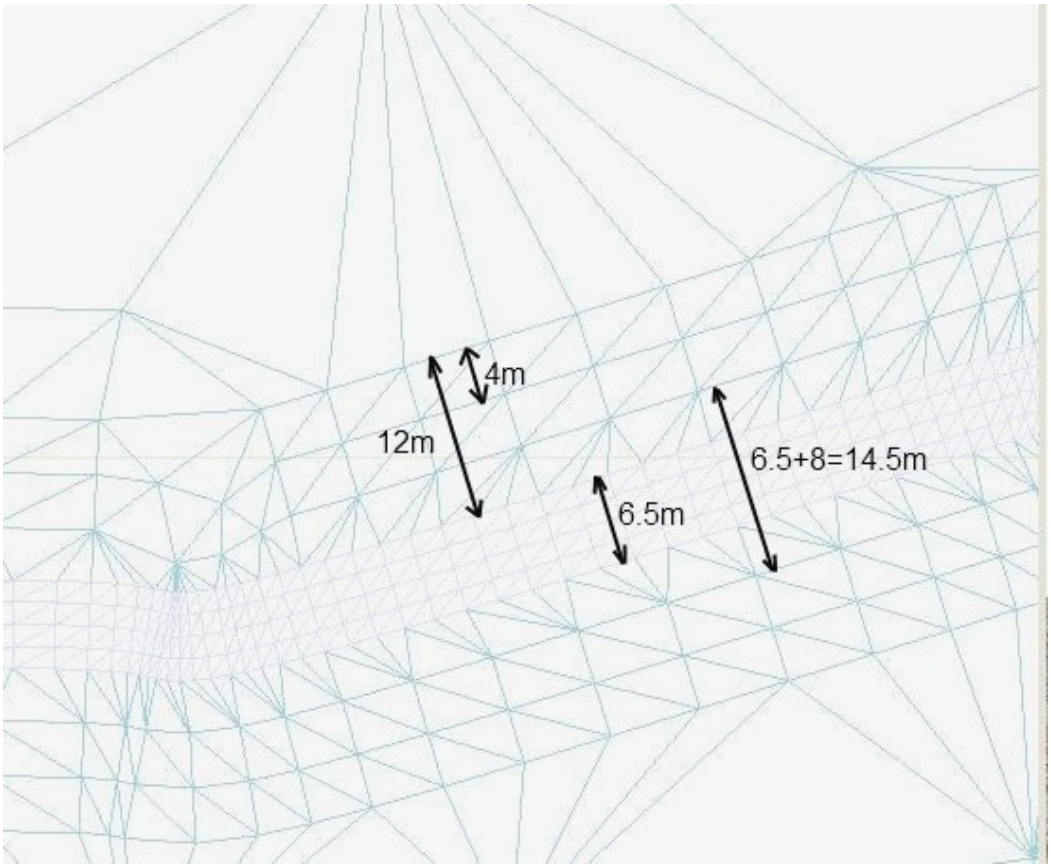


Once we have the reference elevation values along the road, we have to use them to create a smooth profile for the road. A smoothing factor S (S must be odd) will be used to create a new set of values where each elevation is the average of the elevation of the nearest S anchors. The distance taken into account when smoothing depends on the anchor's separation. The default anchor separation has a mean value of 5m. The best way to set is is trying several values (clicking "Create the elevation profile") and watch the results.The other parameter sets a maximun allowed slope for the road. 0.25 means 25%. You can set it to 1 if you don't want to use this parameter (1 would be a limit of 100% slope, and that means no limit for a normal road).

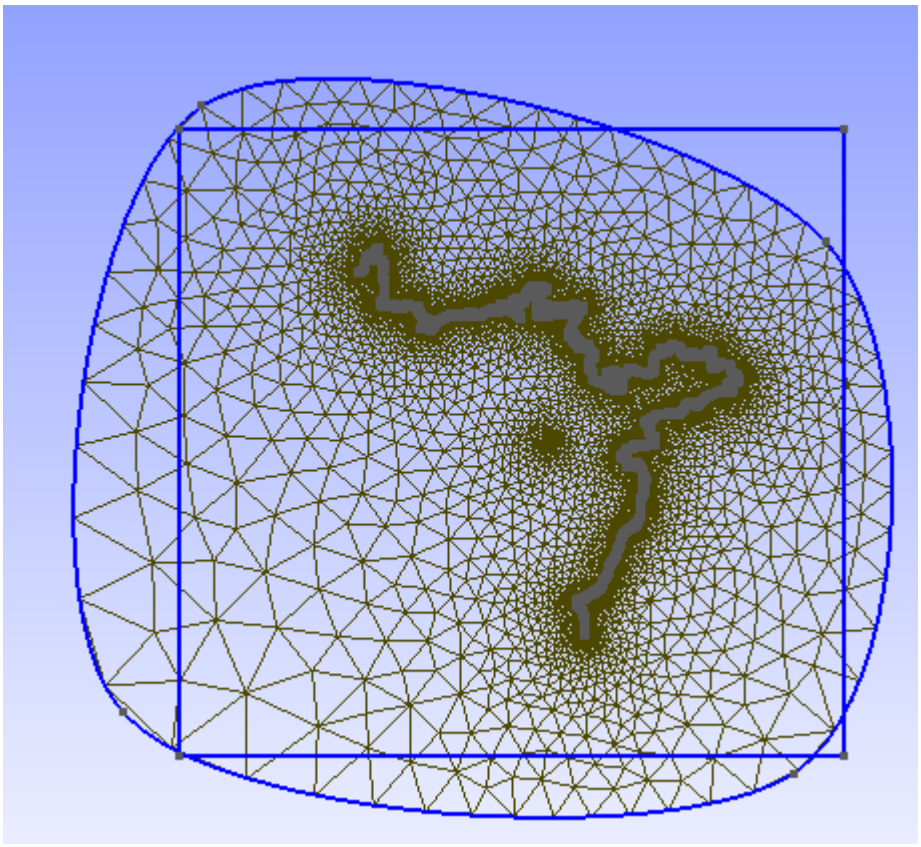
If we are content with the elevation profile proposed by the scripts we just have to press "e" in the octave console and then <ENTER>. Then we consolidate the profile clicking on the "Consolidate" button, the last one of the s3_road step. If you don't like the profile, this step allows the user to fine tune the elevation profile or even change a little the elevation data (if you are using lamalla.mat files) trying to make a better fit for your road. Read section VI for details.

s1_mesh

Once we have the final version of our road, the next step is creating the driveable terrain on its both sides. We have to set 2 parameters: the width of the driveable terrain and how many panels do we want in that width. The following picture can help you to understand the parameters: we used a width of 12m and we selected 3 panels. The mesh created by the scripts for the driveable terrain will be the base to create another mesh for the non-driveable terrain. This new mesh has to be created by the user using gmsh.



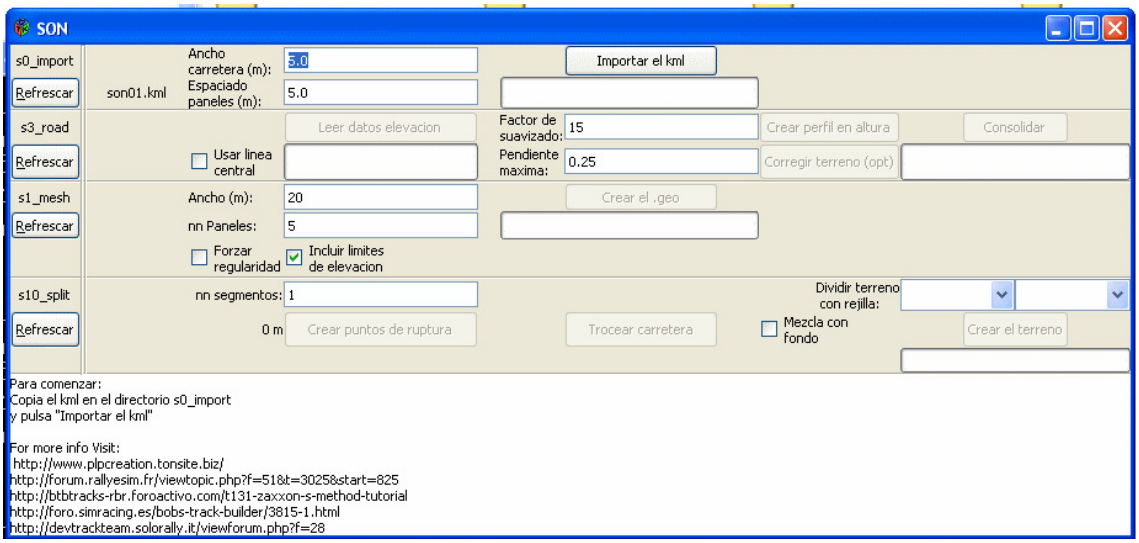
In this step it is really important to select "**Include elevation data limits**" in case you are using Google Earth as your source of elevation data. This prevents you from creating a mesh that exceeds the available elevation data limits, a mesh that can't be processed by the scripts. For example, in the following picture the mesh exceeded the limits and had to be redone to confine its boundary inside the rectangle (available elevation data limits).



Once we have processed the father (at least partially), we have to process the sons. The steps are similar but not exactly the same: the sons do not manage elevation data.

```
> cd ..
> cd son01
> zgui
```

The window is similar to the FATHER's window, but we should read “SON” on the top bar.



The son has the "s10_split" step, a simple step that splits the track into several segments for efficiency reason. Segments of length around 1Km or below seem to be ok. Greater segment lengths can make both BTB and RBR run too slow.

Sometimes the SONs are really short and using normal smoothing factors in step s3_road makes the scripts fail (e.g. If the son has only 7 anchors and we select a smoothingg factor of 13 there are not enought points to average). Selecting a smaller smoothing factor (even 1 if needed) often solves the problem.

We have to process all the sons.

Once we have processed all the sons we have to join all the tracks: the father and the sons:

```
> cd c:\project\father\s1_mesh
> join_geos
```

The output is c:\project\father\s1_mesh**joined.geo**, a file that has to be processed with gmsh to create the .msh file we need.

Processing joined.geo is not straightforward so it is not explained in this section. The output of the process should be file **joined.msh**, a gmsh mesh file with 2 Physical Surfaces: 111 for the driveable zone and 222 for the non-driveable zone. You can read more about using gmsh in section VII.

Once you have joined.msh file in s1_mesh\salida folder, you can go on with the steps, launching zgui(1) for the father.

```
> cd c:\project\father
```

```
> zgui(1)
```

s4_terrain

The options in s4_terrain step are:

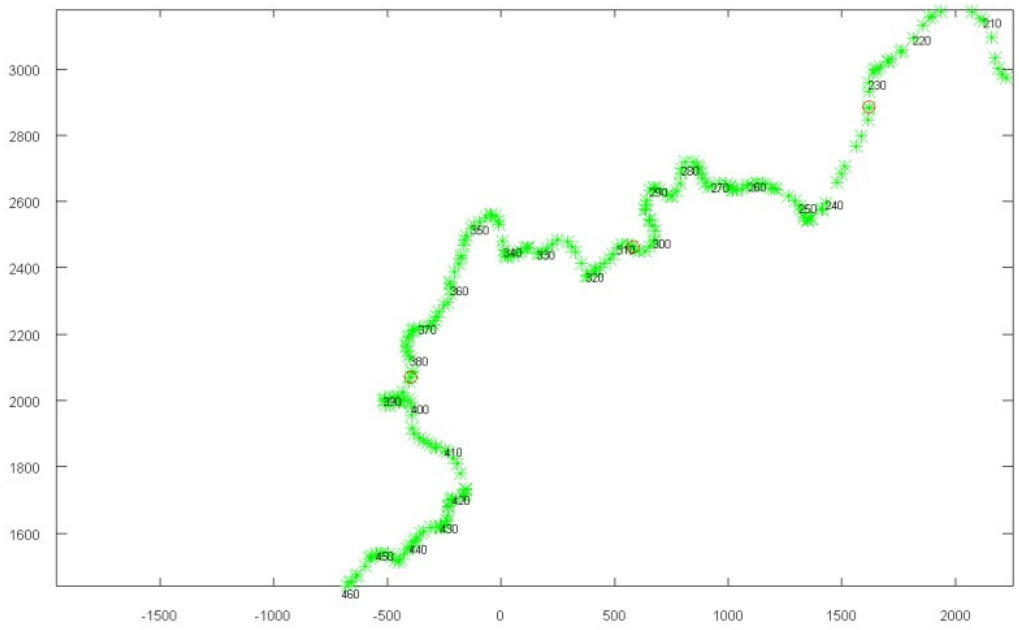
1) Create sasplanet.hlg. You have to create this file if you want to include satellite images in your project. This file is designed to be opened with SASPLANET. You can read section VIII for more info.
2) Create test.obj. You create a .3D object (format .obj) in folder s4_terrain\salida. This object can be used to check your mesh or it can be the final output you are looking for.

s7_walls

Walls are created automatically to prevent you from reaching the non-driveable zone. It is optional. It only has sense for RBR users, not for rFactor.

s10_split

First we create split points (script split_track(n), being n the number of segments we want to create). This button creates a file called '**pos_nodes.txt**" that contatins the numbering of the nodes where splitting will take place. It is also shown a graph so we can check those points on the track top view. If we want to change the split positions we just have to **edit pos_nodes.txt** and then click the next button "Split the track". The second button (calls partir_track) uses pos_nodes.txt as input so the file determines the number of segments (and where they start and finish).



Once the track has been splitted into segments (1Km segments seems to be ok for efficiency reasons), we have to split the terrain using an MxN grid. If we have background images and we are going to use them as texture for the terrain we MUST use exactly the same MxN dimensions used when creating the images. Splitting is recommended for efficiency reasons but rFactor users should kmow that too many faces in a terrain area (>1000) will make BTB fail to export the project to rFactor.

If you select the option “Blend with background”, the terrain areas will have satellite images as texture. Read section VIII for more info.

s9_join

If you want satellite images as background images in BTB, you have to type the correct folder where the .dat files can be found (Read section VIII for more info). Otherwise type a non-existant folder name. If you selected “Blending with background” option in s10_split step you MUST have background images, otherwise it is impossible to open the Venue.xml using BTB.

Finally you can add pacenotes to the father's route and finally create the s9_join\salida\Venue.xml, the final output of the process. This file needs WP.zip to be opened and also the .dds images if you are using background images.

IV.2 One-track project

Projects with only one track are a simplified version of the multi-track projects: basically the same options but you don't have to process the sons. Instead of creating joined.msh form joined.geo, you create anchors_carretera.msh from anchors_carretera.geo.

When working with only one track you should read the keyword “NORMAL”, instead of “FATHER”/”SON” in your windows.

V WORKING WITH DIFFERENT SOURCES OF ELEVATION DATA

V.1 Google Earth

It is possible to give elevation to a set of gridXXX.kml files using a Python wrapper for Google Earth COM API. You can also try to use BTBLofty or 3DRouteBuilder, but the process will be slower (and may be more expensive):

There are several ways of giving elevation to a set of gridXXX.kml files:

1) Buy 3D Route Builder. Unless you pay 15 euros, we can only process files with a maximum of 200 points
2) Use BTBLofty. It works nice in Windows XP, but doesn't work under Windows7. Even with Windows XP you have to process the files by hand: one after the other, losing a lot of your time.
3) Use raise_kml script. Basically it does the same as the 2 possibilities above. It works under Windows XP and Windows 7 and automatically processed all the gridXXX.kml files.

To use raise_kml you need to follow these steps:

Python instalation

1. Install Python 2.7 in directory C:\Python27. DO NOT INSTALL IT IN A DIFFERENT FOLDER

   http://python.org/ftp/python/2.7/python-2.7.msi
2. Install pywin32

   http://sourceforge.net/projects/pywin32/files/pywin32/Build216/pywin32-216.win32-py2.7.exe/download
3. Install pygoogleearth

   http://pypi.python.org/packages/any/p/pygoogleearth/pygoogleearth-0.0.2.win32.exe#md5=7e92b3cf1dcb4a5493aacb393a9e54a2
4. Edit file c:\Python27\lib\site-packages\pygoogleearth\geapplication.py

   changing (line 231)

   ```
   return gehelper.point_dict_from_terrain_point(terrain_point)
   ```

with:

```
return terrain_point
```

V.2 AGR Data

ASCII Grid files with the following format:

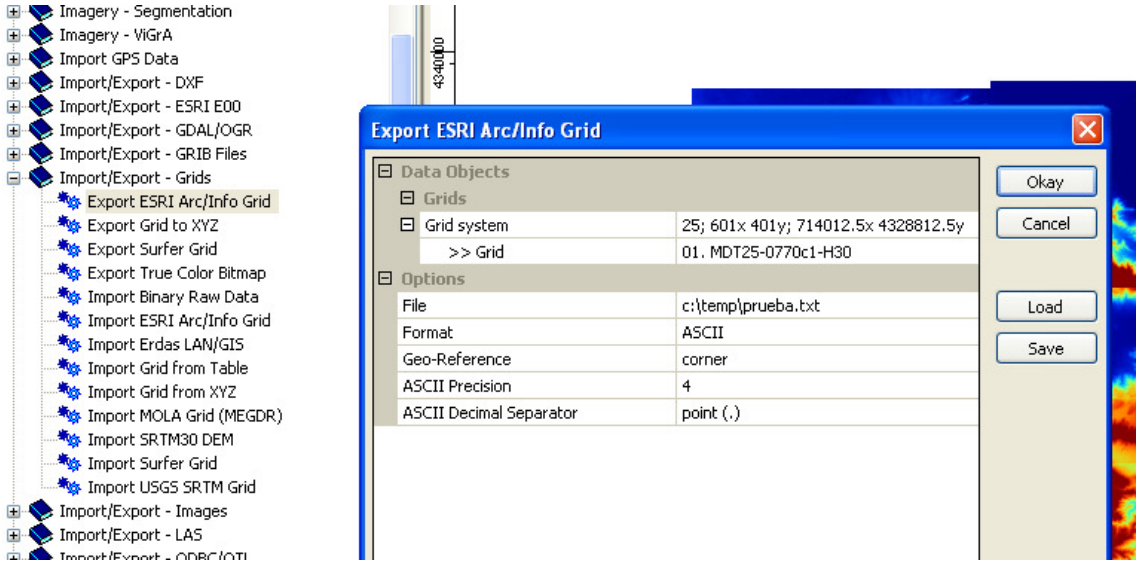
```
NCOLS 601
NROWS 401
XLLCORNER 714000
YLLCORNER 4328800
CELLSIZE 25
NODATA_VALUE -999
33.381 33.279 33.102 32.982 32.809……
```

are supported. Coordinates MUST be **UTM** (*Universal Transverse Mercator*).

Use:

4) Create a folder named “agr” in the same folder where you have the “project” folder (or the father’s and sons’ folders in multitrack projects). E.g. if you are working on c:\project, create a folder c:\agr.
5) Copy the needed .agr files to the folder you created in the step 1). Rename those files with extension .agr if needed.
6) Use the scripts as usual. Following the steps recommended on screen the s2_elevation step will be skipped as we already have elevation data.

NOTE: if you have grids with a different format you can use free software to convert them (like SAGA-GIS, import your grid ant then select *Module Import/Export - Grids \ Export ESRI Arc/Info Grid*, using ASCII format and corner geo-reference).
http://sourceforge.net/projects/saga-gis/



NOTE: .agr grids with 5 and 25m spacing can be downloaded for free for Spain. Just create an user and download MDT05 or MDT25 files.
http://centrodedescargas.cnig.es/CentroDescargas/
Using the code of the following grid, downloading is easier:
http://centrodedescargas.cnig.es/CentroDescargas/equipamiento/cuadricula_MTN50.png

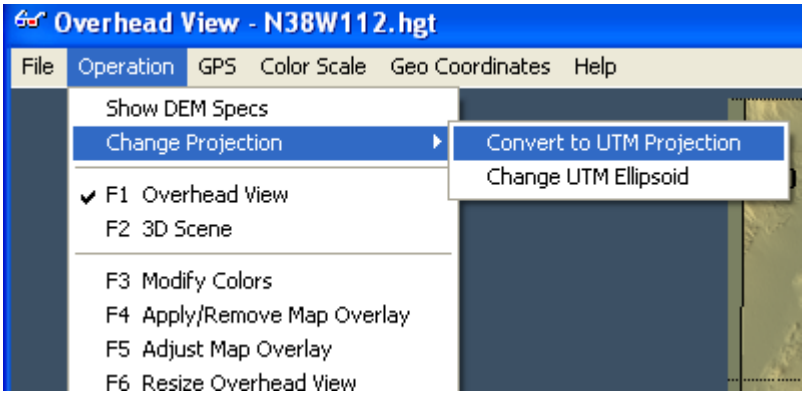
NOTE: for Catalonia, grids with 15m spacing can be downloaded for free from
http://www.icc.cat/vissir2/?lang=es_ES

just clicking on the map and choosing "otras" section. The format is exactly the same used in the cnig web, but the spacing is different. File extensions should be changed from ".txt" to "agr".

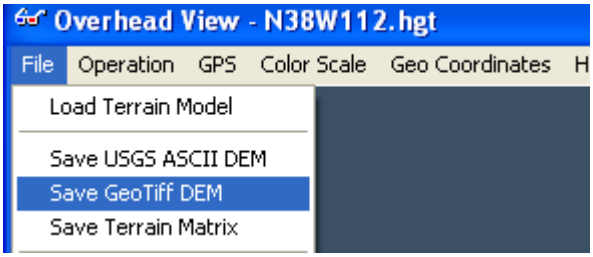
NOTE: MDT05 files may be to big for using them directly as elevation data source for the scripts. Script split_agr can be used to create smaller files from a MDT05. Example of use with file **MDT05-0667-H30.ASC** to create 5x5 files (with extension .AGR):

split_agr('MDT05-0667-H30',5)

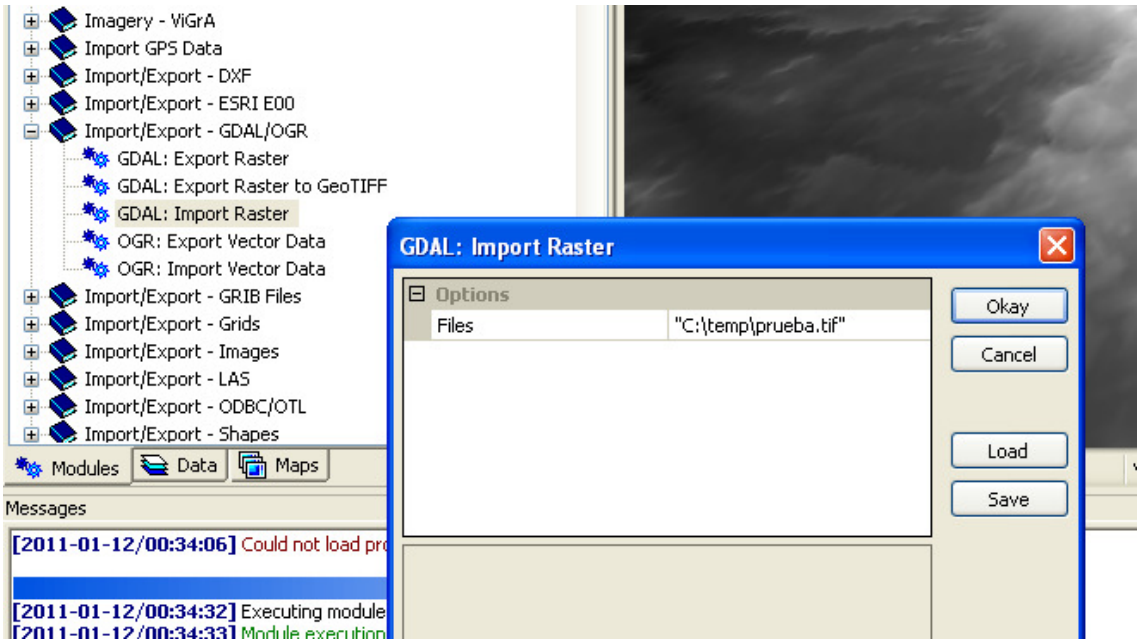
NOTE: .hgt files can be opened with 3dem to project them to UTM (Operation\Change Projection) http://www.viewfinderpanoramas.org/3dem.zip



Then saved as Geotiff dem file:



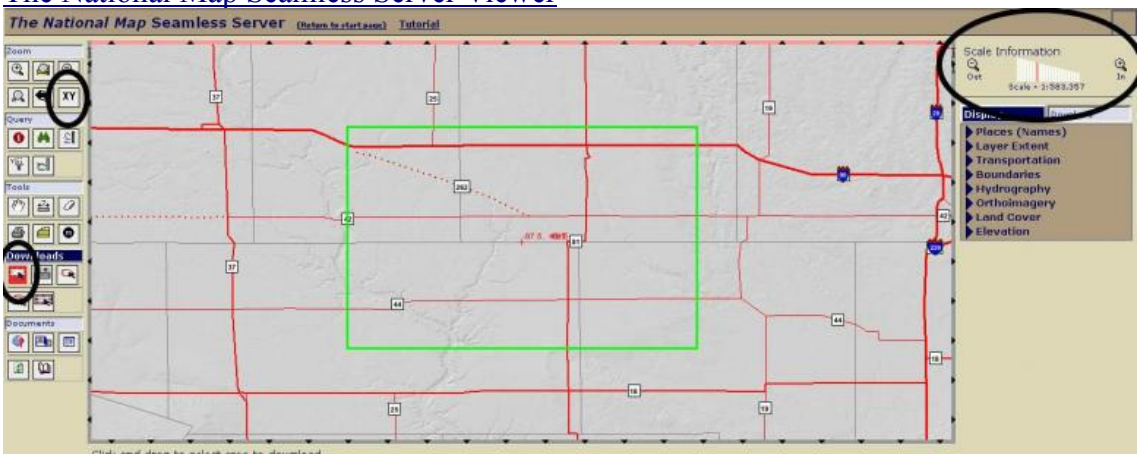
And imported with SAGA GIS:



V.3 Seamless server

This web offers for free elevation data with 10 or 30m resolution for USA and 90m (and often with missing point values) for the rest of the World.

The National Map Seamless Server Viewer



To check what kind of data they have from our zone of interest we select the tool **Zoom->XY** and write the coordinates of a point. Then we zoom in (upper right side of the screen). With **Downloads->*Rectangle*** we frame the interesting zone. If the rectangle is too big, it will turn red. In the pop-up window we click on **Modify Data Request** button. We look for the **Elevation** section and we select the data source,. For example "**National Elevation Dataset (NED) 1/3 Arc Second**"

and we choose "**GRIDFLOAT**" format. We click on "**Save Changes & Return to Summary**" button and if no problema arises a Windows with a "**Download**" button will pop up. This is the easiest an quickiest way to get good elevattion data. If we are not interested in USA tracks we can always use GE through 3DRoute Builder, BTBLofty or raise_kml.

Note:
3 arcosecond ~= 90m
1 arcosecond ~= 30m
1/3 arcosecond ~=10m

Gridfloat data can be transformed into lamalla.mat format using script **leer_gridfloat.** This script can not be used until s0_import step has completed.

Example:
> cd ..\s2_elevation
> leer_gridfloat("file.hdr","file.flt")

V.4 hgt files

Example:
> cd ..\s2_elevation
> lee_hgt("N41W009.hgt",[41 42],[-9 -8])

.hgt files data can be transformed into lamalla.mat format using script **lee_hgt.** This script can not be used until s0_import step has completed.

VI FINE TUNING OF THE ELEVATION PROFILE OF THE ROAD

There are 2 advanced actions we can do related to the elevation profile of the road:

1) Changing by hand the profile proposed by the scripts
2) Changing the terrain to fit the profile we have created

VI.1 Changing by hand the profile

There is a javascript tutorial: http://www.mediafire.com/?pdrdg6q9kp5zv3x

Run dar_altura as usual

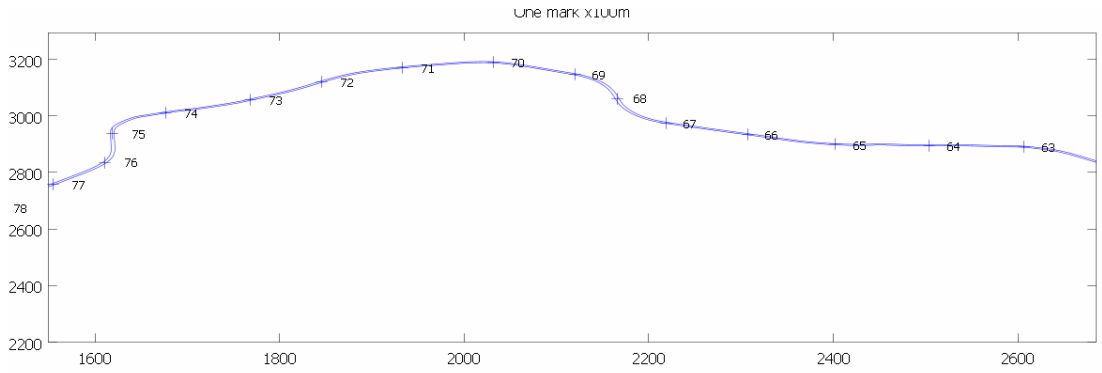
> dar_altura(15,0.3,-0.3)

```
> dar_altura(15,0.3,-0.3)
Leyendo ..\anchors.mat
Leyendo alturas_track1.mat
Leyendo el fichero retoques.txt
Cerrando el fichero
        e-_ end              n-_ new segment
```

Now **dar_altura** waits for the user to press "n" or "e" and <ENTER>

- "n" if you want to change a segment of the elevation profile
- "e" if you want to quit

But **before choosing "n", zoom the zone you are interested on**. Before zooming you probably will need to have a look at the other figure created by dar_altura. It shows the distance travelled at each point of the road (scale by 100 to get distance in meters)

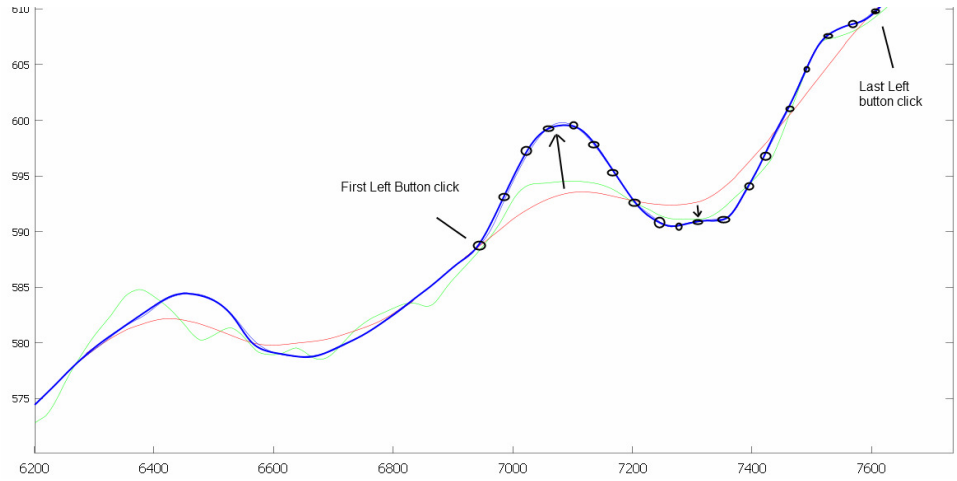


Fine tuning of the elevation profile of the road

If for example you want to change the elevation profile between position 6800m and 7600m, then you know you need to zoom that zone (select zone with right button) on the other figure. After zooming you can press “n” on the textmode window. Message on textmode window should change:

```
Leyendo el fichero retoques.txt
Cerrando el fichero
        e-_ end              n-_ new segment
n
Left click to add point. Right click to finish
```

Now you have to click on the elevation profile graph using the left button of the mouse. **From left to right**. You can use any number of points (>=3). When finished, click anywhere with the right button.



Then you will be asked again to choose “e” or “n”.

Once you press “e” the script dar_altura exits. Inside “s3_road\salida” you can find a file called “Venue.xml” that can be opened with BTB to check the shape of the road.

Important: All changes applied are recorded in a file called “retoques.txt”. This is a plain-text file that can be edited by hand if needed. Each line of the file is composed by the number of points of a segment and the (x,y) coordinates of those points.

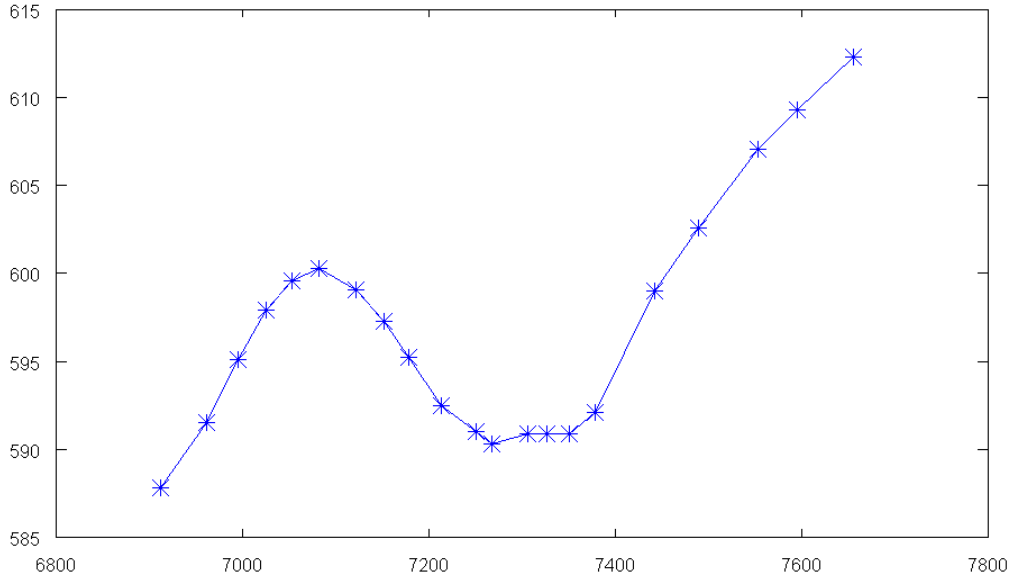
Each time dar_altura is called, ‘retoques.txt’ is read and elevation profile changes are applied. New changes will be added at the end of ‘retoques.txt’.

If you want to remove old changes, delete “retoques.txt” file. Lines inside retoques.txt are applied in the same order they appear inside the file.

Example retoques.txt line:

21 6911.3 587.8 6961.6 591.5 6994.3 595.1 7024.5 597.9 7052.2 599.6 7081.2 600.3 7121.5 599.1 7151.7 597.3 7178.1 595.2 7213.4 592.5 7249.9 591.0 7267.5 590.3 7305.2 590.9 7326.6 590.9 7350.6 590.9 7378.2 592.1 7442.4 599.0 7489.0 602.6 7553.2 607.1 7594.7 609.3 7655.2 612.3

If we plot those points we see they are the definition of one of the changes applied:

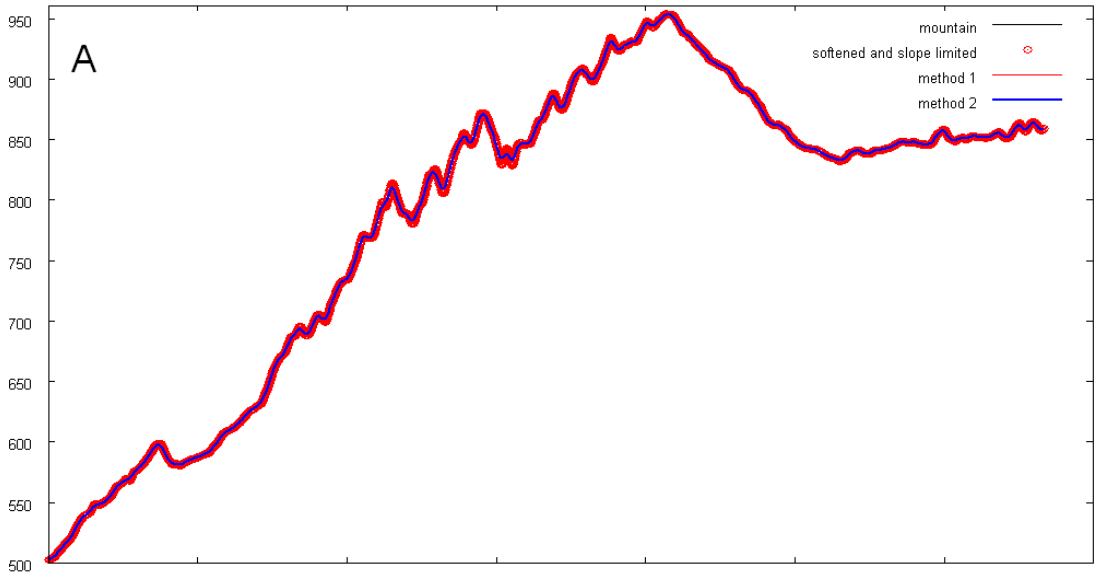


The first parameter of dar_altura is a smoothing factor. You must choose it so little manual changes are needed. It must be always odd.

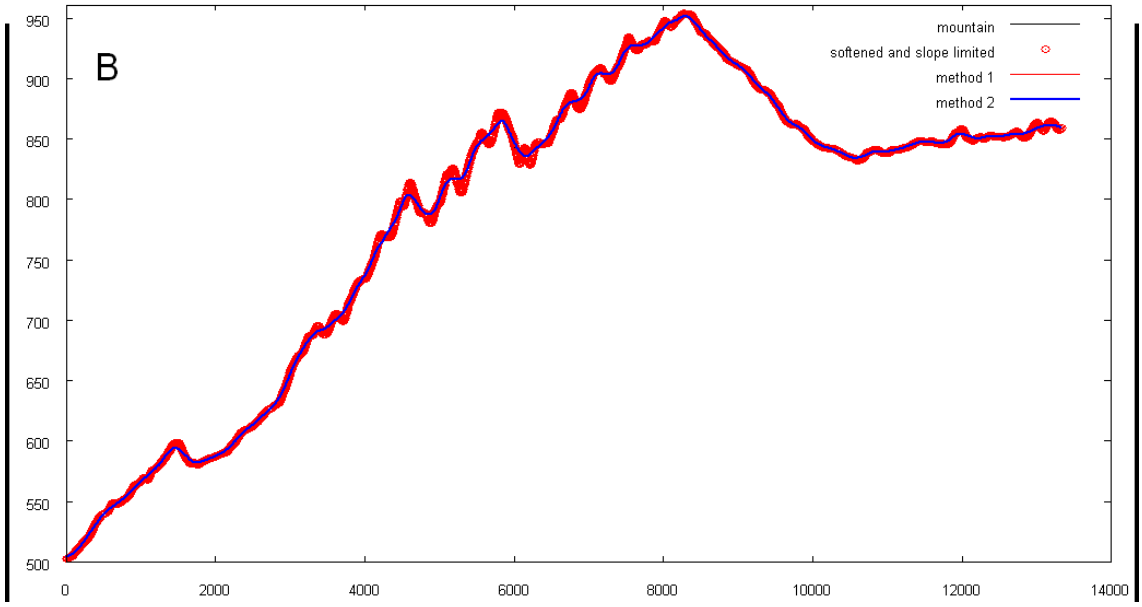
The final elevation profile will be a spline created using one elevation value each 25m. That means that *little irregularities created by manual editing may be NOT so important*. If you want you can change that distance (25m by default) setting it as the 4th parameter of dar_altura.

VI.2 Use of script “corregir (fix)”

Let’s assume we want to create a track but when we run “dar_altura” we detect elevation data is not good, and the road goes up and down in a nonsense way (see Fig A). Elevation changes are obviously caused by bad elevation data (we could check Google Earth images to confirm that fact).

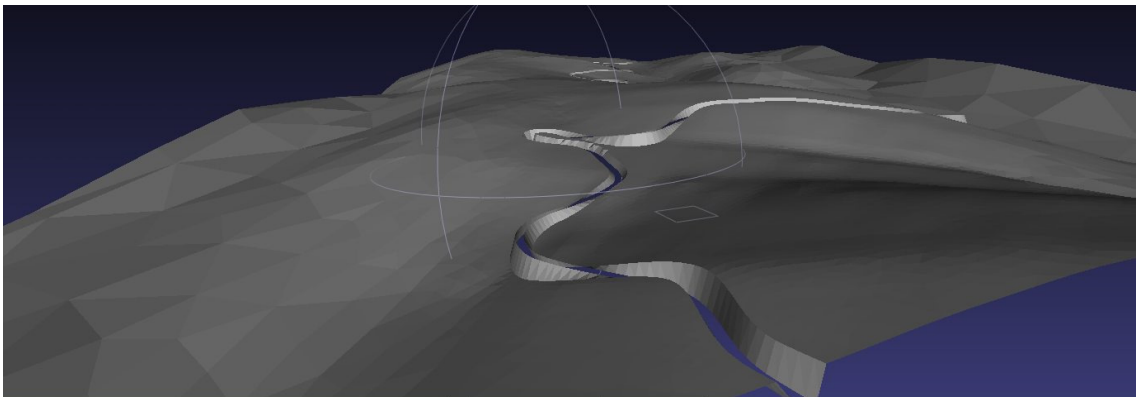


Script “corregir”, can change elevation data in the following way. First we create a softer road with dar_altura (see Fig B).
> dar_altura(53,1,-1,100)



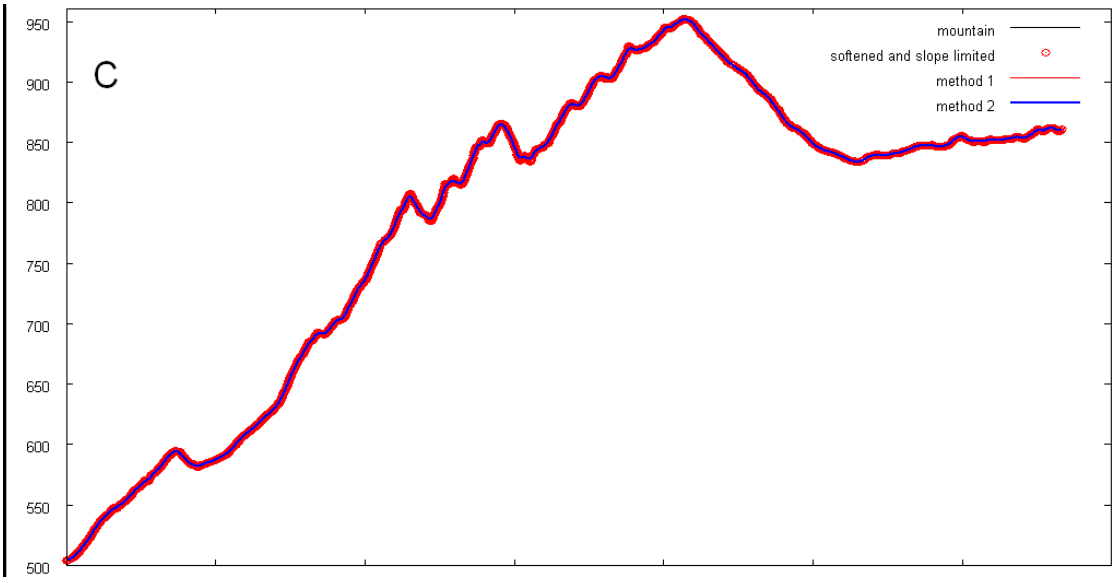
This road does not fit the mountain well. If we went on with the process using this road, we could get something like this:

Fine tuning of the elevation profile of the road



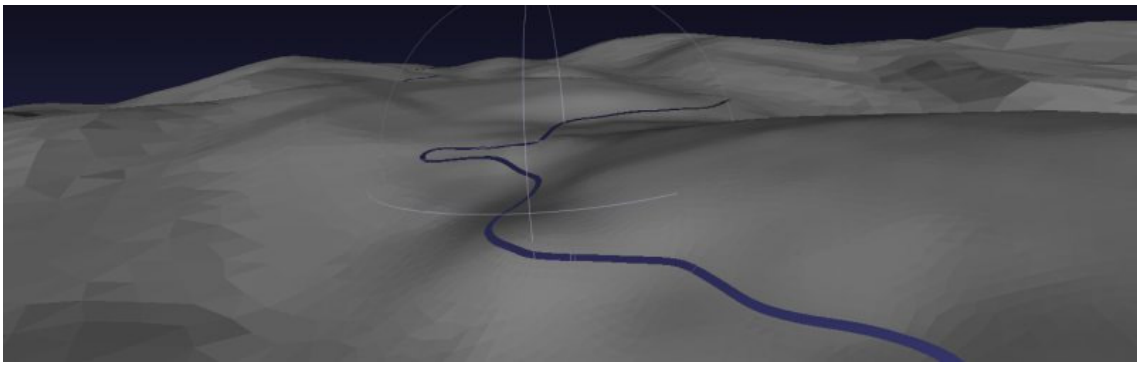
So, now we want to change the elevation data to fit our desired “soft road”:
> corregir
And we run again dar_altura to create the final road, a road that fits the new elevation data:
> dar_altura(21,0.25,-0.25)

We can see (Fig C) that now the road (blue line) is almost as soft as the desired one, and the mountain (red circles) hasn’t big elevation differences with the road. Compare differences between red circles and the blue line in Fig B and Fig C. The road is almost the same, **but “corregir” has changed elevation data to fit the road**.



Once we have a road with a elevation profile we like, we go on with the next script. The result can be as good as:

Fine tuning of the elevation profile of the road



Scripts we have run (you can use the parameters you want for them):

```
cd ..\s3_road
coge_datos
creartrack1
dar_altura(53,1,-1,100)
corregir
dar_altura(21,0.25,-0.25)
```

If you want to start the process from scratch, just run again coge_datos inside s3_road, and you have the original elevation data ready for creartrack1 and the rest of the scripts. If you don't want to use "corregir", just use dar_altura as usual.

VII USING GMSH

VII.1 What are anchors_carretera.geo and joined.geo?

anchors_carretera.geo and joined.geo are files that should be processed with gmsh. They contain:

- TerrainAnchors, the points where road and terrain faces are linked. These points are joined with straight lines.
- Help points, on both sides of the road at a distance specified as a parameter.
- Already created “Plane surfaces” for the driveable zone.

User has to define the non-driveable surface. This part is the only difficult step of the “zaxxon method”, but there are several video tutorials shown how to do it.

VII.2 gmsh basic concepts

If two points of the boundary of a mesh are joined with a straight line, both points will be part of the mesh.

If two points of the boundary of a mesh are joined with a spline, those points will not necessarily be part of the resulting mesh. gsmh will find the optimal position of the triangle nodes on that boundary.

When working with gmsh each point may have a "**characteristic length**", that is the lenght we want for the triangle sides on this point. For example, inside the file anchors_carretera.geo (it is a plain text file), the road Anchors have a characteristic length called "cl1", with a value of 20m. Nevertheless Anchor points are joined with straight lines, so triangles in touch with the road will have a side length defined by the position of the Anchors (mean of 5m). By default the scripts set the size of the mesh elements by thresholds and not by characteristic lengths (see section VII.8).

VII.3 gmsh basic controls

- Right mouse button and drag: move the project
- Ctrl + left Mouse button and drag: make zoom of a zone

- Click 1:1 text on the left-down corner: Fitting zoom
- Click Z on the left-down corner: restore top view
- Left Mouse button and drag: 3D rotate view
- Mouse wheel: zoom

VII.4 What are the limits for the non-driveable zone?

The non-driveable zone shouldn't exceed available elevation data. Otherwise scripts like procesar_nodostxt will fail. If you don't have selected the option to view the limits of the available elevation data when using the GUI and you want to see the limits on the gmsh screen, run addgrid inside s1_mesh folder:

> **addgrid(1,1)**

Where the 2 parameters are the number of horizontal and vertical divisions of the grid. This command only works if you are using lamalla.mat files, e.g. Using Google Earth as your source of the elevation data. Open the .geo again to see the changes.

Do NOT reach the limits with your gmsh meshes. Otherwise, next scripts will fail as there will be no available elevation data for some points.

VII.5 How should I create the external Boundary of the non-driveable zone?

Before meshing we should create the "Plane surfaces" to mesh. To create the plane surfaces we must have boundaries defined, and the definition of boundaries need points to be defined.

We select:

Geometry->Elementary Entities->Add->New->Point

Once the Mouse is on the desired position for the point we hold "*shift*" on the keyboard, and the coordinates get frozen. We move the mouse to the "Contextual Geometry Definitions" window and there we set coordinate Z=0 and Characteristic Length to *cl3* (cl3 is a value defined at the beginning of anchors_carretera.geo). Click Add.

Once we have defined all the points we need, we select:

Geometry-> Spline

and we define the spline shape selecting the points the spline must follow. Once the curve has been close we can define a “Plane surface” defined by the spline as its external Boundary and the driveable surface as its internal hole.

Geometry -> Plane Surface

We click on the external Boundary. If it has been correctly closed we will be asked to select the hole boundaries ("Select hole boundaries"). We must select all the segments of the driveable zone boundaries, until all the segments are colored red. At that moment the option “undo” will disappear from the top part of the window. Then we press “e” to end the definition of the “Plane surface”.

We can then mesh to check that everything is ok:

Mesh -> 2D

and all the already defined plane surfaces will be meshed.

We need at least two kinds of surfaces:

- Driveable: help points “minus” road
- Non driveable: external Boundary “minus” help points.

We can select Tools->Statistics to know how many triangles will have our mesh.

VII.6 What is the output of the process?

If we save the mesh a file called joined.msh/anchors_carretera.msh will be created. This file contains two kinds of lines:

1) Nodes. For example node #14:

14 2149.778 5113.46 0

2) Triangles (gmsh calles them elements, and they are called faces inside BTB). For example triangle 21222 belongs to the physical surface of driveable ones (111) and is defined by three nodes: 1490, 17130 and 1491:

21222 2 3 111 31672 0 1490 17130 1491

VII.7 How can I define physical surfaces?

There are 2 physical surfaces, driveable (numbered 111) and non-driveable (numbered 222). Each one of them is defined as the list of "plane surfaces" that belong to that physical surface.

The initial list of surfaces for Physical Surface 111 is the contents of file salida\phys111.txt (it is there since multitrack support was added). If your project has no additional tracks, just add the contents of phys111.txt at the end of anchors_carretera.geo:

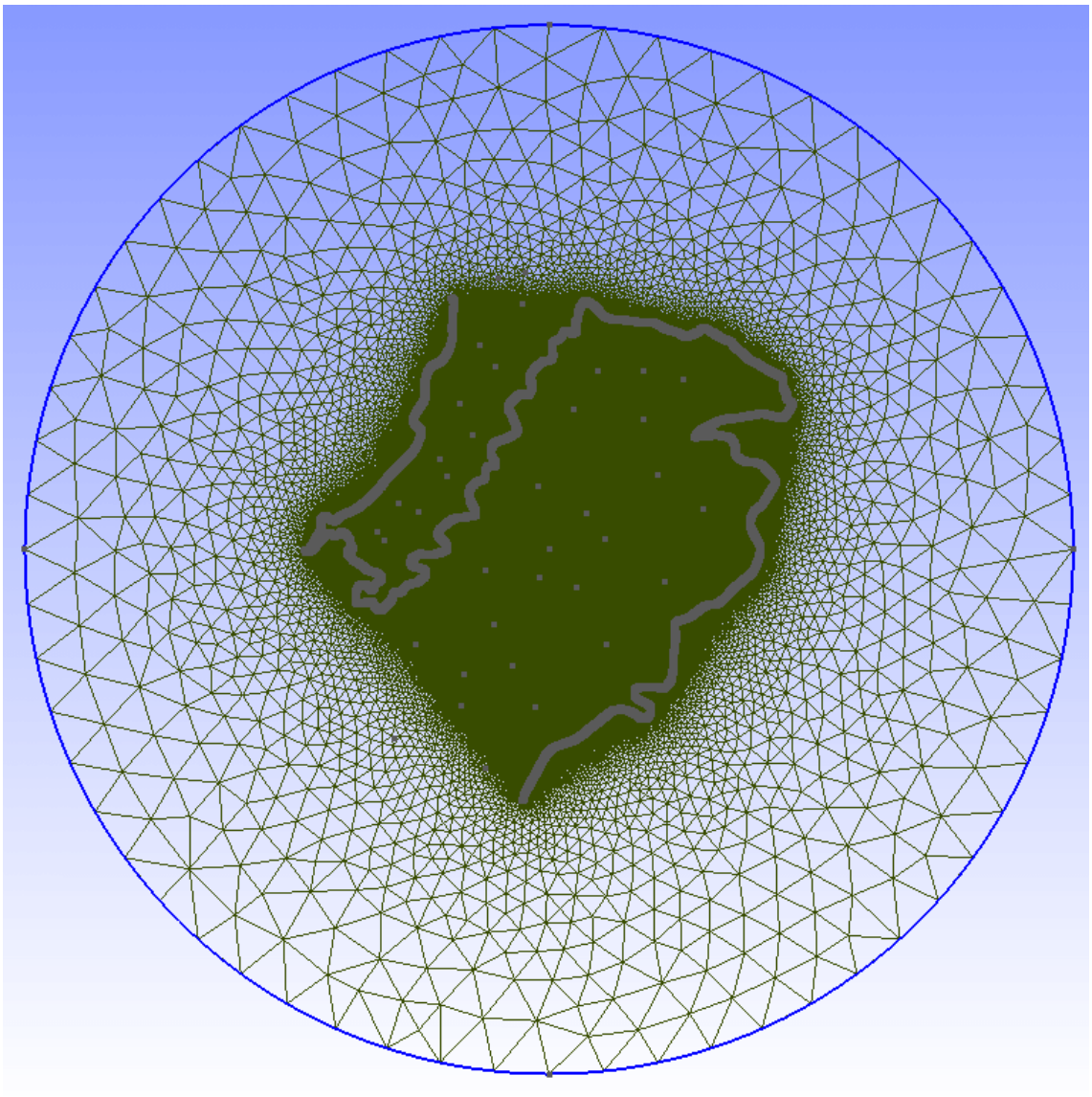
"**Physical Surface(111)= {contents of phys111.txt};**".

We add "**Physical Surface(222)={X};**" at the end of the file, where X is the code of the last Plane Surface defined inside anchors_carretera.geo (it is the non-driveable plane surface).

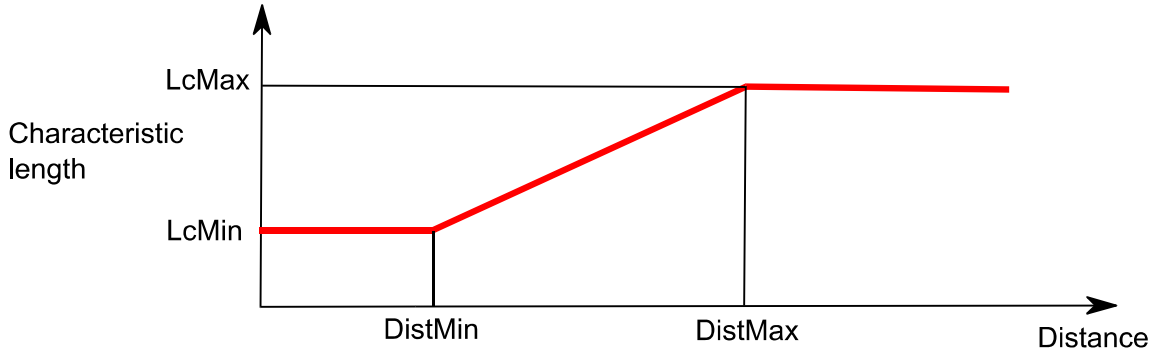
Only the plane surfaces that belong to a physical surface will be part of the .msh file, so it is important to pay attention when defining the physical surfaces. **Once you have finished meshing it is recommended to open anchors_carretera.msh with gmsh to check that all the meshes are there**. Using the Visibility tool (and there Physical Groups in the List Browser tab) may help you as you can select what physical surface you want to see.

VII.8 Gmsh thresholds

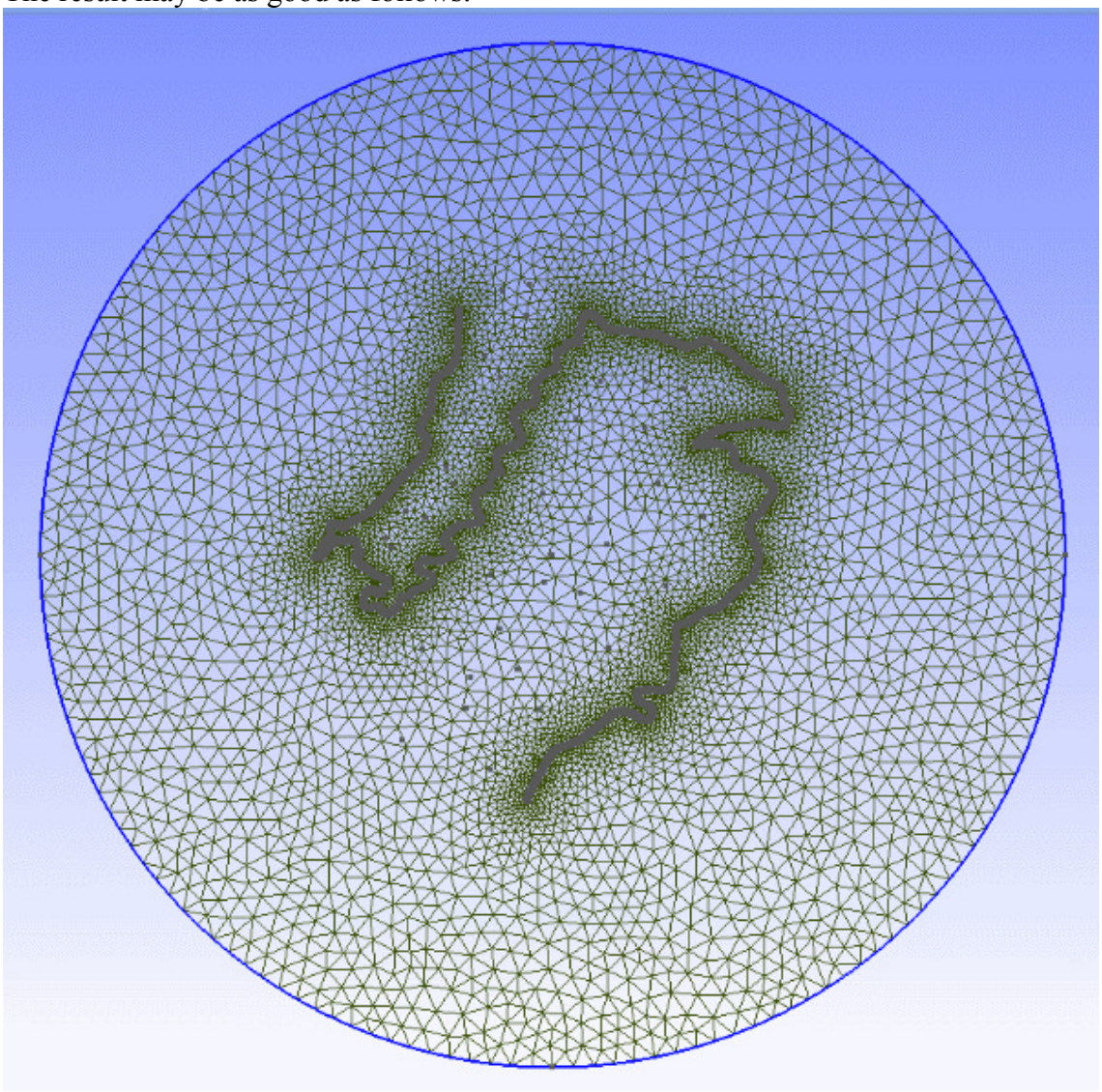
One way of controlling the triangles size with gmsh is giving each point a characteristic length (4th parameter in their definition). The problem with characteristic lengths is that gmsh creates too many triangles for U-shaped tracks (or segments of a track). For example:



Another way of controlling triangles' size is using *threshold fields*. For example you can ask gmsh to assign each triangle a size that is a function of the distance to the track.



The result may be as good as follows:



Gmsh code for using a threshold field is:

```
Field[1] = Attractor;
Field[1].NodesList = {1:3342}; //List of point of the track

Field[2] = Threshold;
Field[2].IField =1;
Field[2].LcMin = 20;
Field[2].LcMax = 2000;
Field[2].DistMin = 1;
Field[2].DistMax = 10000;
Field[2].StopAtDistMax = 0;

Mesh.CharacteristicLengthExtendFromBoundary = 0;

Background Field=2;
```

mallado_regular or join_geos scripts include (in anchors_carretera.geo or joined.geo) the code needed for using the threshold fields. It is **activated by default**. If you don't want to activate the thresholds, just open the .geo file with a text editor and search for line **If (1)** and change it to **If (0)**.

You can only activate one threshold field unless you use a Min Field (read the notes at the end of this document). The threshold that is included automatically uses all the points defined so fat in the .geo file, so all the tracks are considered for that threshold in multitrack projects.

The default parameters in the geo are set to (read s1_mesh\thresholds.txt):
LcMin = 20; //(Characteristic length at a distance LcMin and below)
LcMax = 2000; //(Characteristic length at a distance LcMax and above)
DistMin = 1;
DistMax = 10000;

NOTE:

It is also possible to change the threshold parameters within gmsh:

Select **Mesh > Define > Fields.**
In the Fields window click **Threshold**.
For example we reduce element sizes approaching the boundary limits:
1) Enter LcMax = 1000
2) Apply
3) Click 1D meshing.
Note: You should 1D before 2D each time new Threshold parameters are entered.
4) Click 2D meshing.

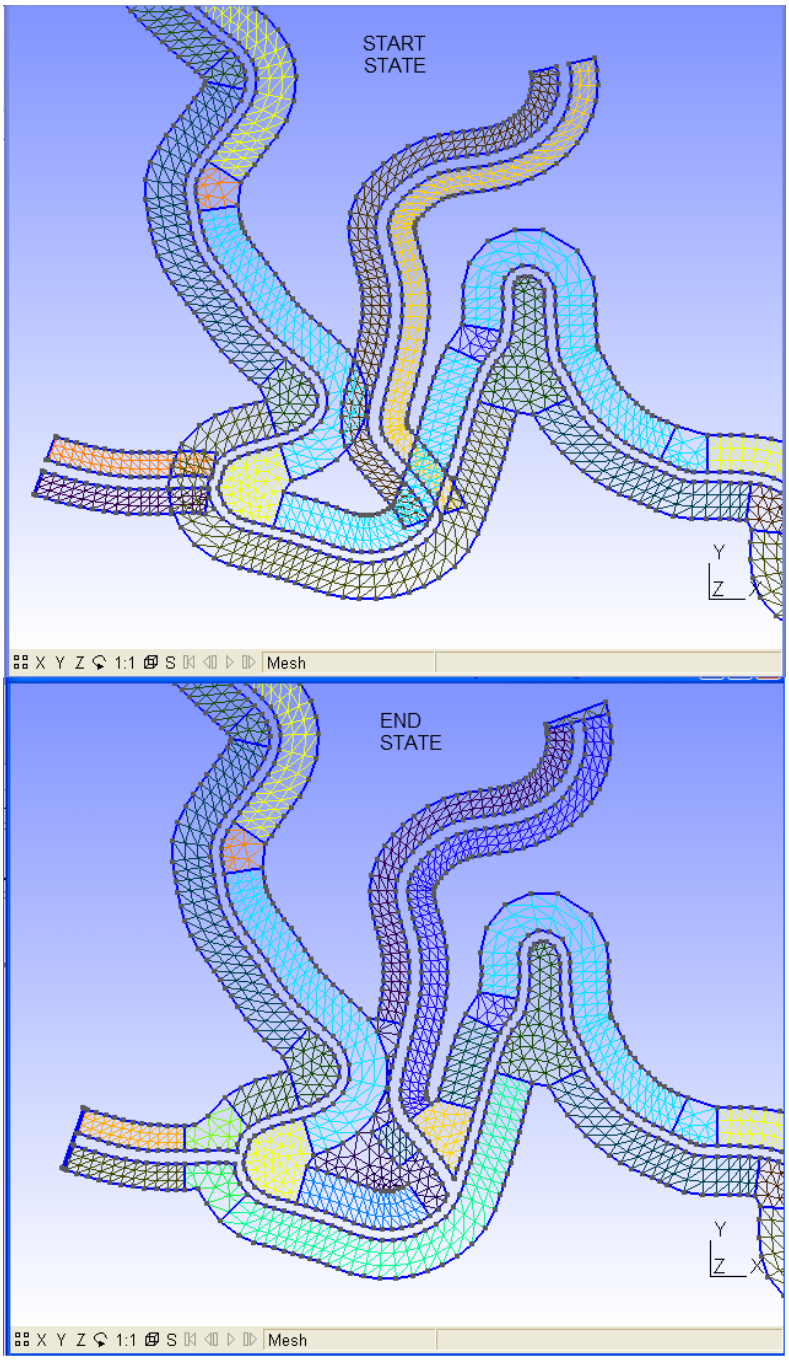
VII.9 Processing joined.geo with gmsh

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.

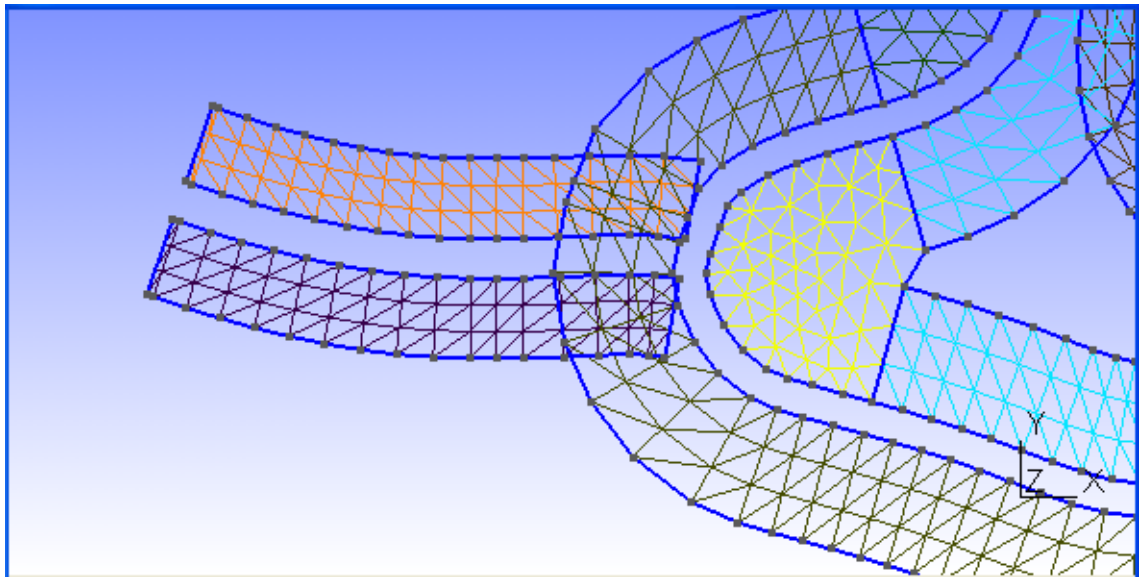
FAST WAY

Using gmsh

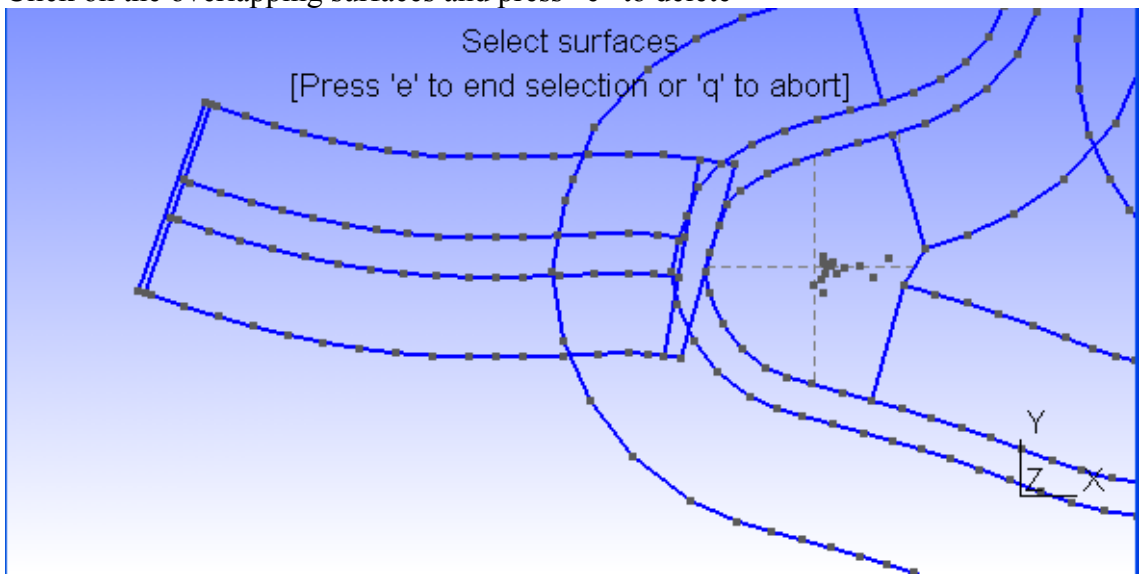
I think this is the best option:
http://btbtracks-rbr.foroactivo.com/t511-gmsh-editing#3345

EASY WAY

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

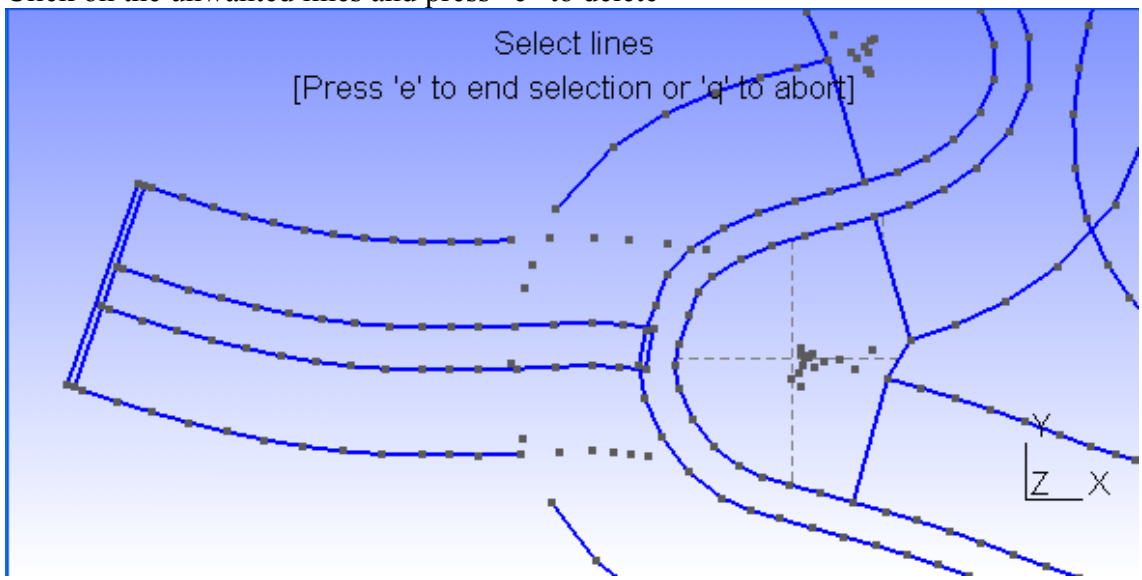


Geometry\Elementary entities\Delete\Surface.
Click on the overlapping surfaces and press “e” to delete

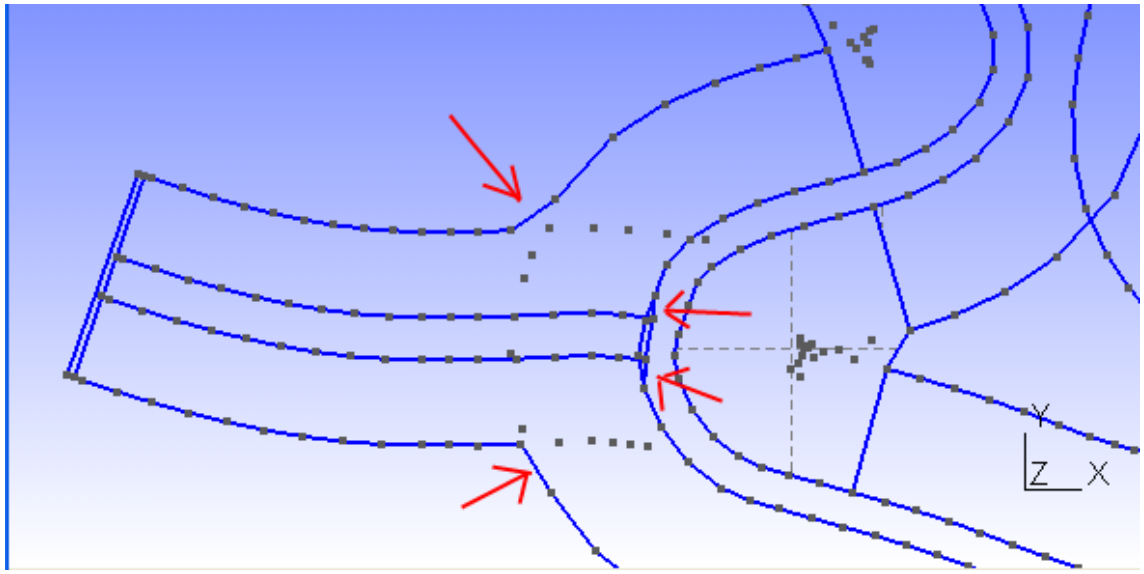


Geometry\Elementary entities\Delete\Line

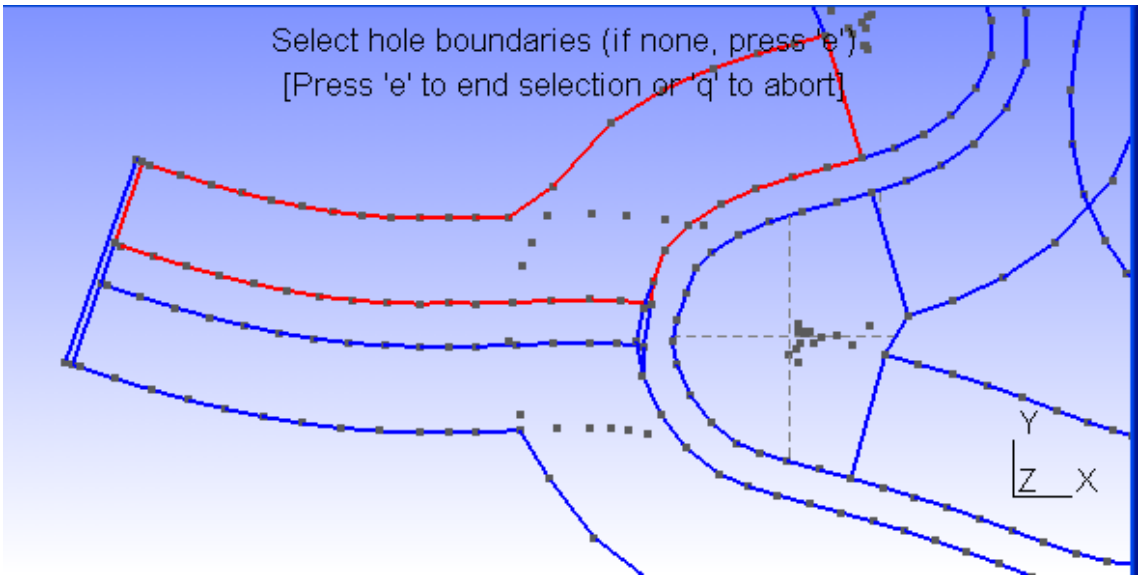
Click on the unwanted lines and press “e” to delete

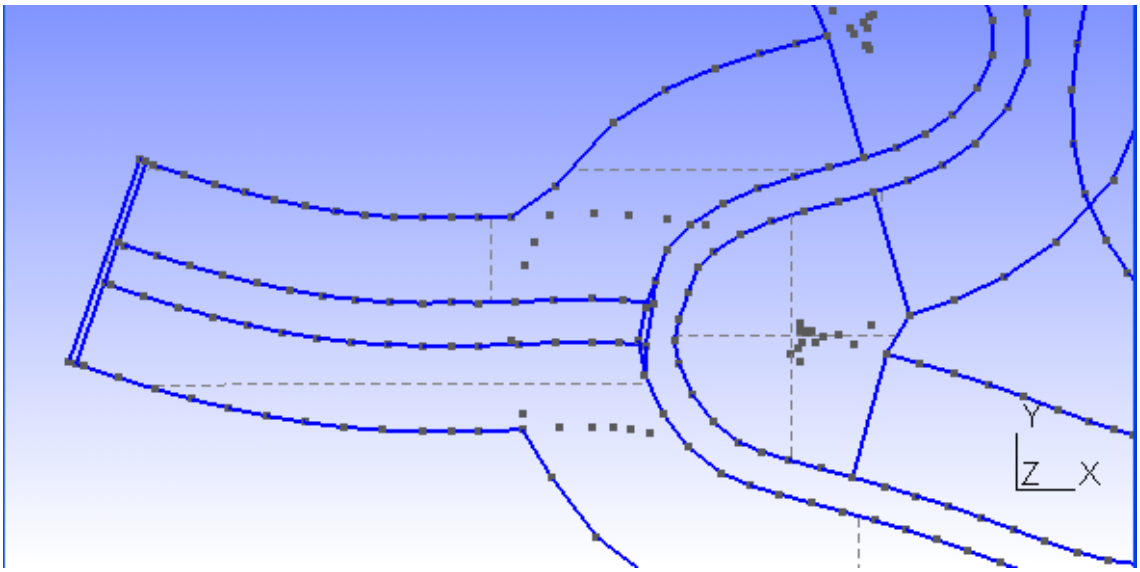


Geometry\Elementary entities\Add\New\Straight Line
Create the lines needed to close the surfaces

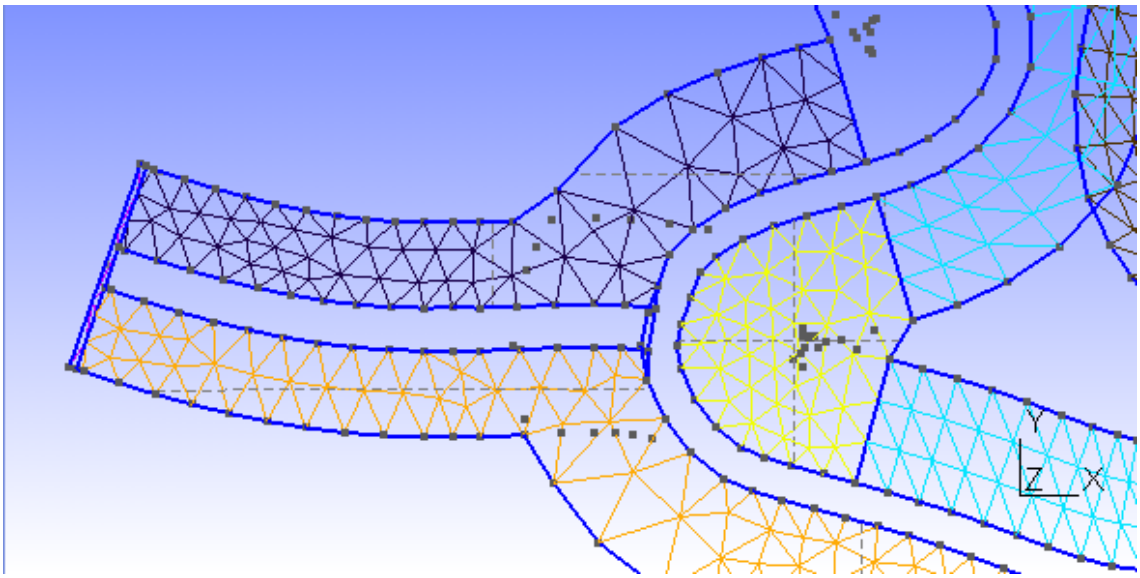


Geometry\Elementary entities\Add\New\Surface
Select the boundary of a surface and press “e”. When all the segments of the boundary are selected, option “u” disappears from screen top message. Repeat for all the surfaces needed, including the track-end surface.





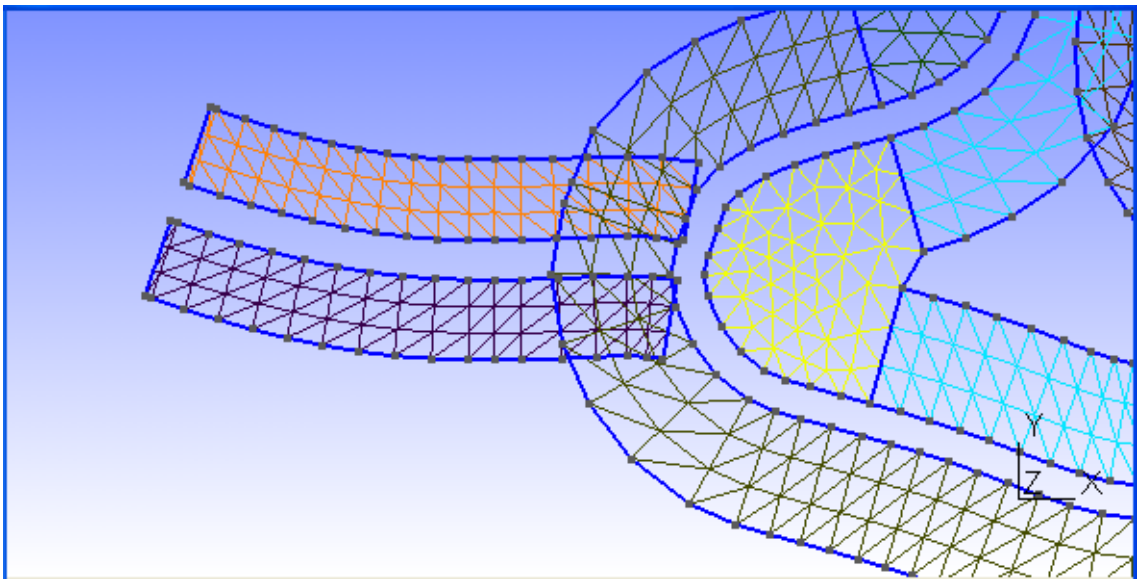
Mesh\2D
To check the resulting mesh



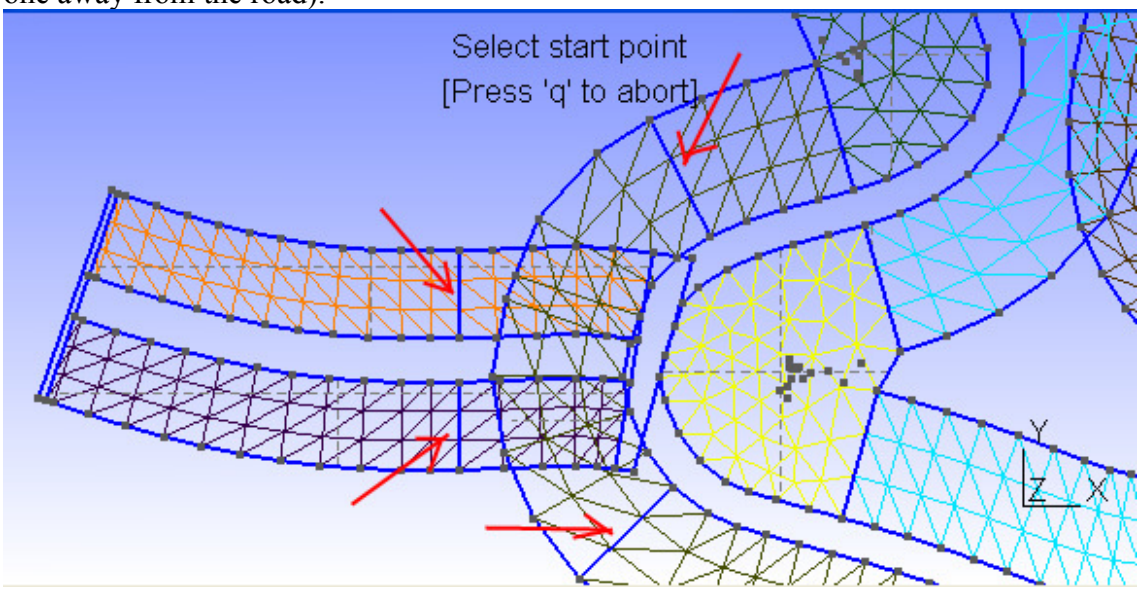
As you can see, with the “easy-way” we lose the regular pattern of the driveable mesh. The not-so-easy way tries to preserve that pattern, with a little extra effort.

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
```

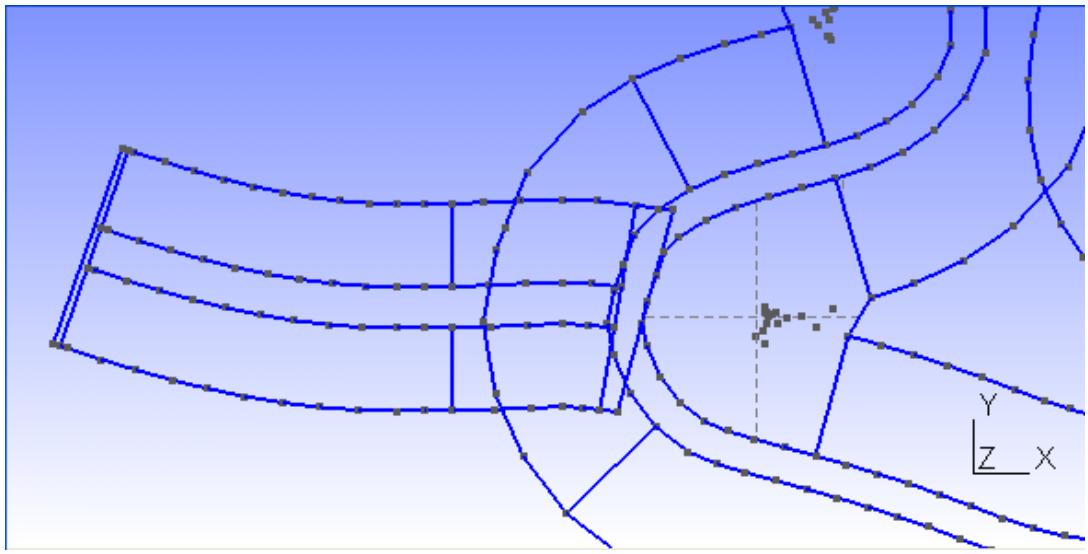
gmsh has written this

we add this

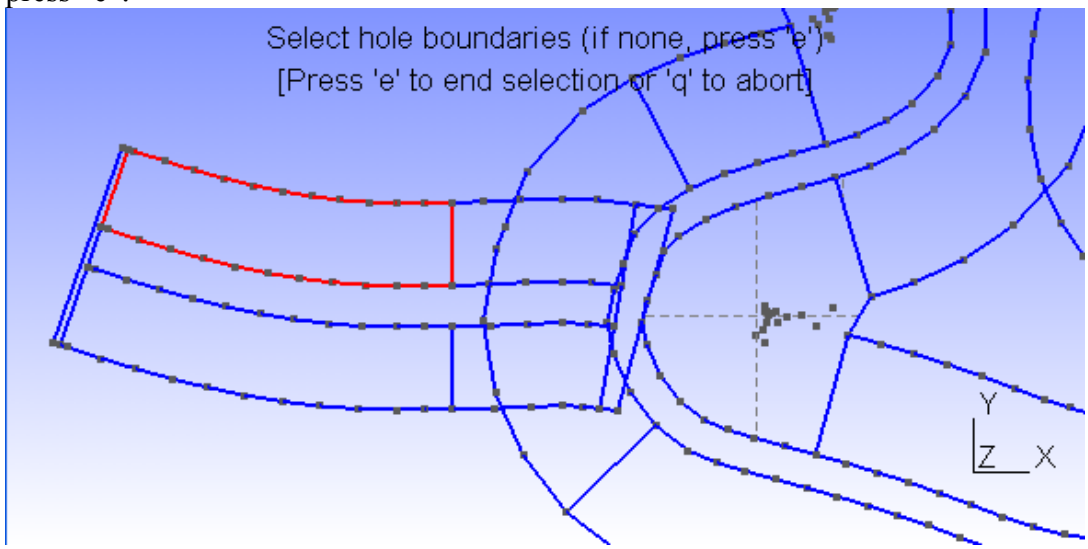
3 panels

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.

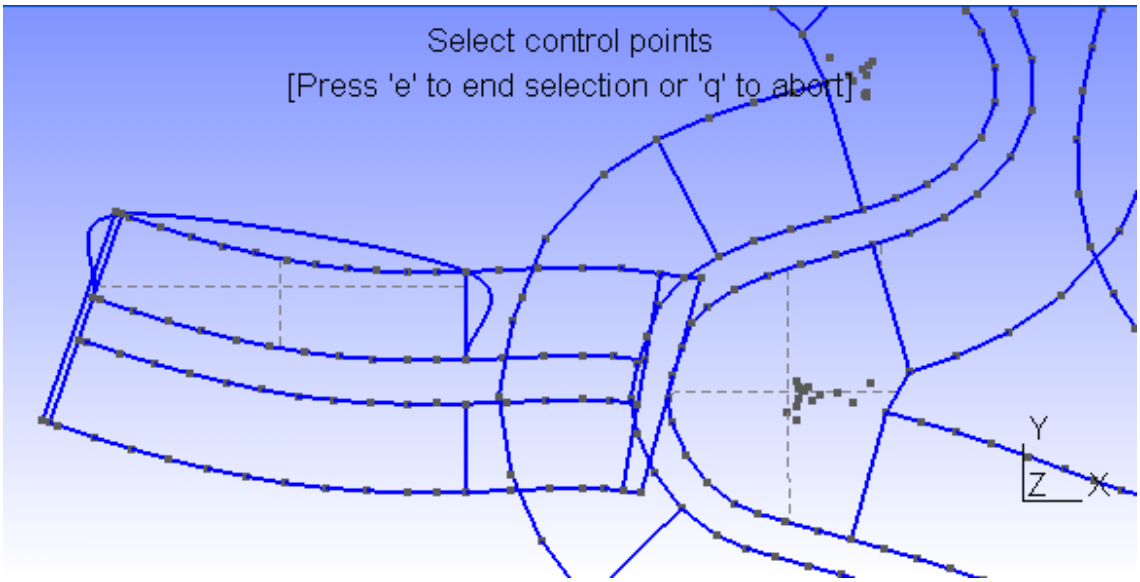
Geometry\Elementary entities\Delete\Surface.
Click on the overlapping surfaces and press “e” to delete



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”.



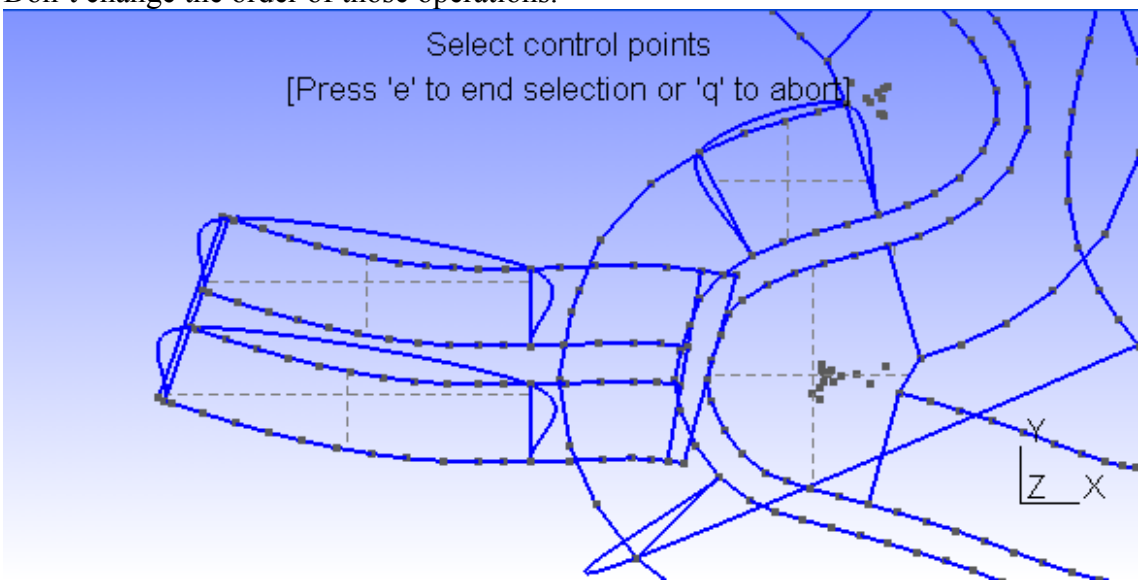
Geometry\Elementary entities\Add\New\Spline
Create a spline using the four corners of the surface you have just created, and press “e”



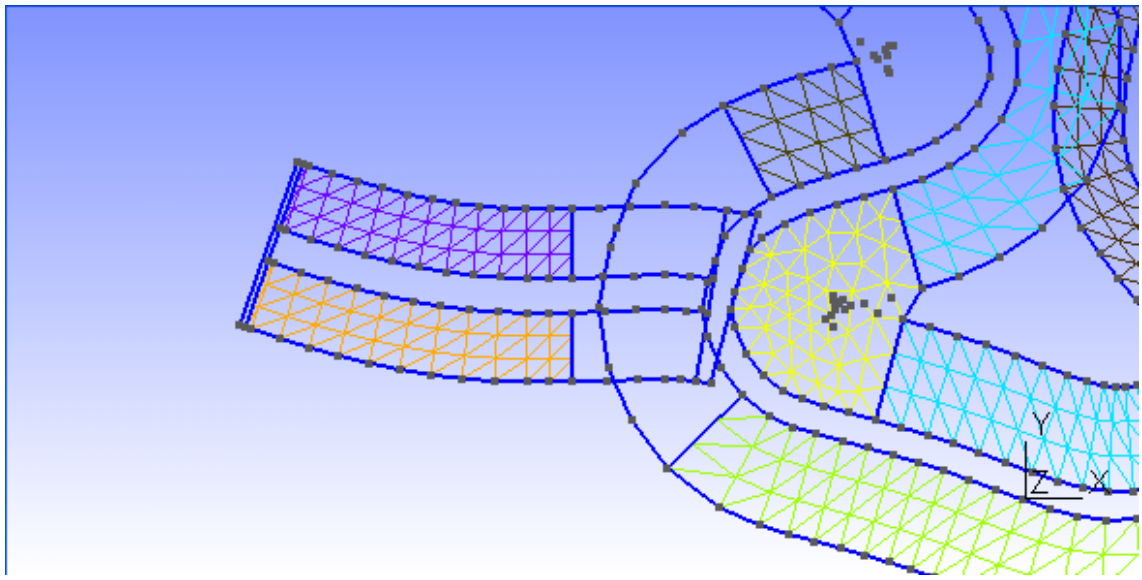
Now run script “addt” inside s1_mesh directory

```
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.



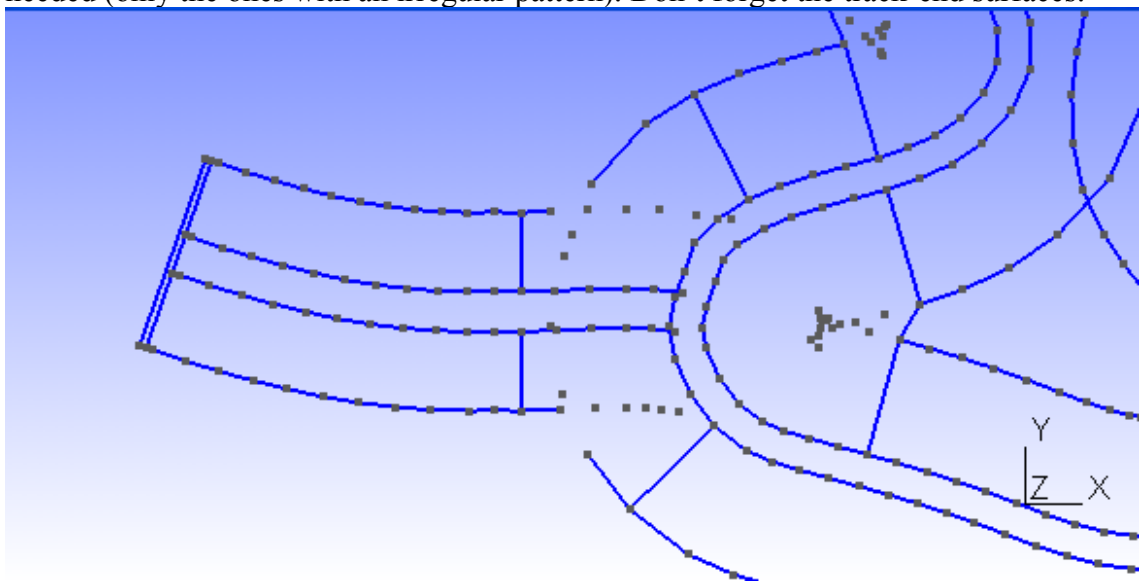
Now close gmsh, open again joined.geo with gmsh and “Mesh\2D”

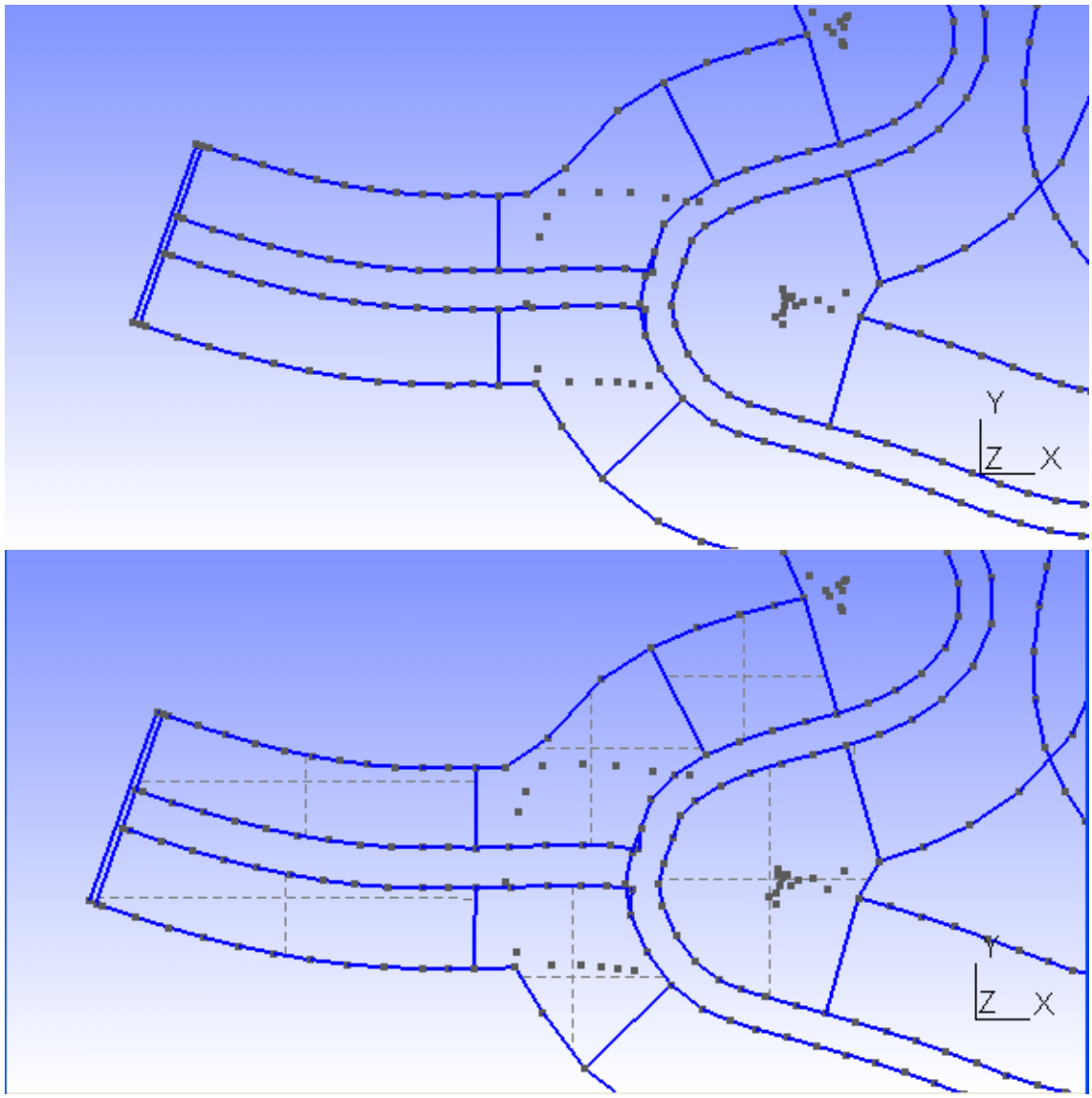


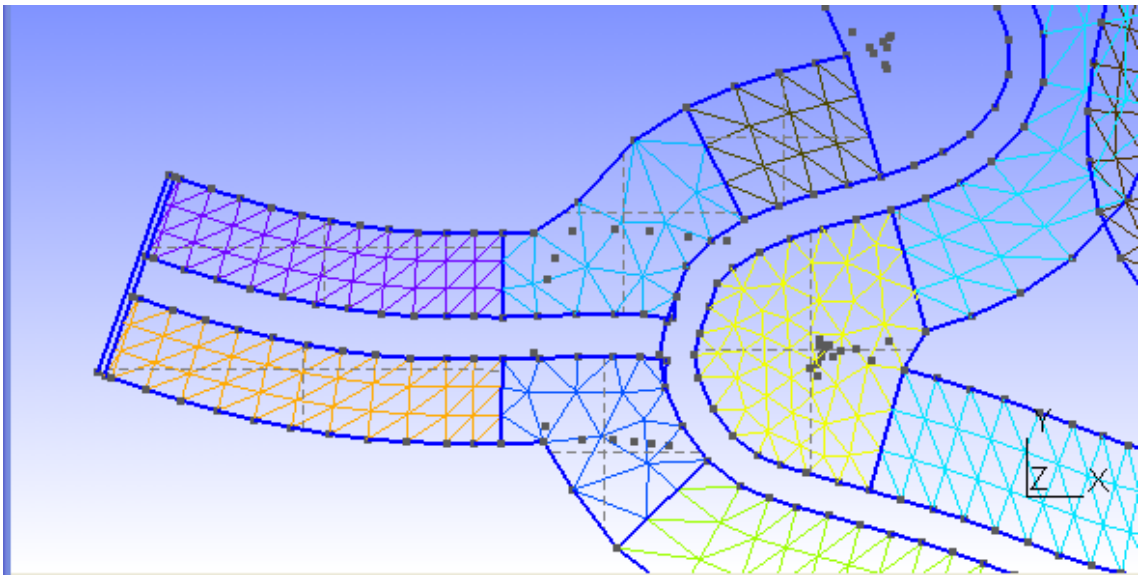
As you can see, the created surfaces have a regular pattern.

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



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).

VIII BACKGROUND IMAGES

VIII.1 Basics

The scripts can help you to include background images into your BTB project. If you want you can even use those images as the texture for your non-driveable terrain.

To work with satellite images you need SASPLANET:

http://sasgis.ru/programs/sasplanet/SASPlanet_100707.zip

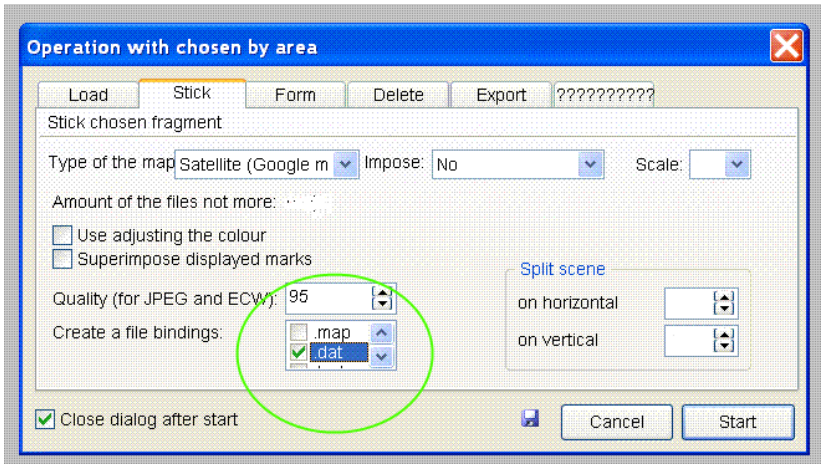
`In step s4_terrain select the option "Create sasplanet.hlg".` A file called s1_mesh\sasplanet.hlg will be created with the coordinates of a box that comprises your terrain.

Once your s4_terrain is completed you can open the file s1_mesh\sasplanet.hlg with **SASPlanet**:

Operations -> Select -> Load from file -> Load -> Start

When the process ends, save the images in a folder (c:\example_folder) with the desired splitting (5x3 grid, for example). **IMPORTANT: if you want to use the images as texture for the terrain you will have to split your terrain using exactly the same grid (step s10_split).** Use "**sat**" as the prefix for the images: they will be named sat_1-1.jpg, etc.

Operations -> Select -> Previous Selection -> Stick -> Start (YOU **MUST select .dat file bindings)**



The satellite images have been splitted and the info about the splitting coordinates is in the created .dat files.

In s10_split step you will have to say in what folder can be found the .dat files. In s10_split step the scripts (add_dat_to_geo) create a list of all the background images, with the format needed by BTB. It is called **s1_mesh\list_bi.txt**. If it exists, it is automatically inserted inside the Venue.xml when join_all is run. **If you don't want to use images in your project you should remove or rename this file before running join_all**.

Images should be named **sat_X-Y.dds** (please note that they are in .dds format) and should be located inside **My Project\XPacks\Common\Textures** (*My project* is your project's BTB folder).

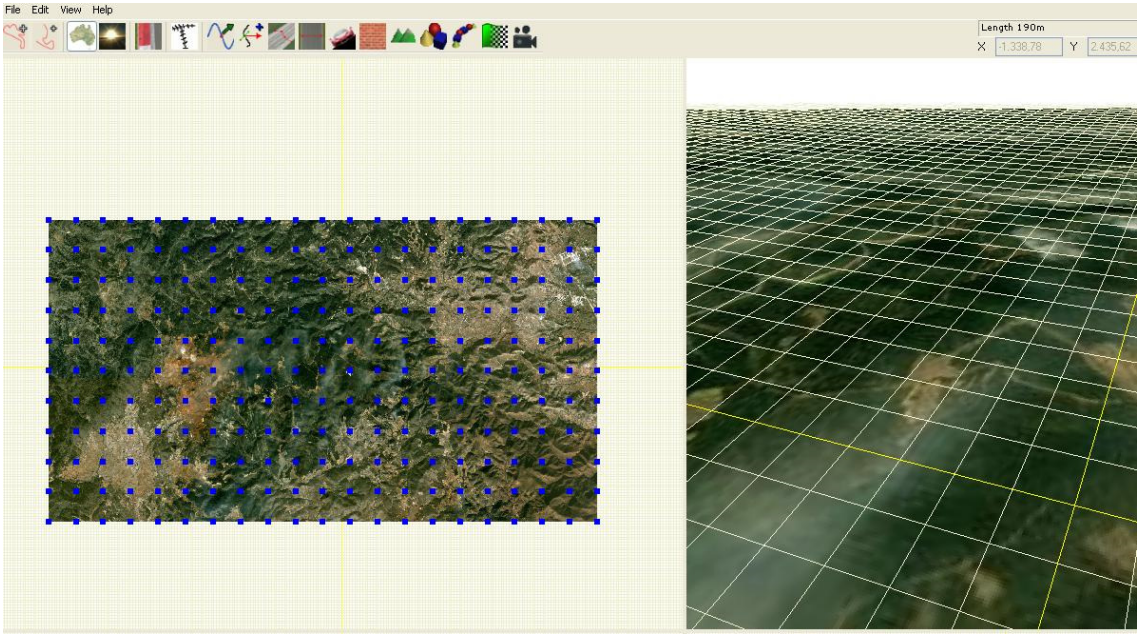
VIII.2 Blending with background images

In the creation of the BTB terrain, the scripts (procesar_elementstxt_mt) can automatically use the background images as texture for the non-driveable zone, if the terrain is splitted **using a grid of the same size as that used in SASPLANET (to create list_bi.txt)**.

For example:
procesar_elementstxt_mt(10,5,0) splits the terrain using 10x5 grid, but does NOT map background images to terrain zones. If list_bi.txt exists and the images are located inside Common\Textures folder, the project will have background images, but they will not be linked to the terrain (as their texture).

procesar_elementstxt_mt(10,5,**1**) splits the terrain using 10x5 grid, using background images for blending in terrain zones. Before opening the Venue.xml it is mandatory 1) to create list_bi.txt (giving the right location for the .dat files) before Venue.xml is created and 2) copying the images to Common\Textures folder. Otherwise BTB will refuse to open your project.

Background images



Example list_bi.txt

```
<BackgroundImages count="50">
    <BackgroundImage>
      <Path>Common\Textures\sat_1-1.dds</Path>
      <Plane>
        <Position x="-11232.375390" y="-0.5" z="5636.028398" />
        <Scale x="2100.341106" y="1" z="-2311.287859" />
        <Rotation x="0" y="0" z="0" />
      </Plane>
    </BackgroundImage>
    <BackgroundImage>
      <Path>Common\Textures\sat_1-2.dds</Path>
      <Plane>
        <Position x="-11232.375390" y="-0.5" z="3324.740539" />
        <Scale x="2100.341106" y="1" z="-2311.287859" />
        <Rotation x="0" y="0" z="0" />
      </Plane>
    </BackgroundImage>

etc.

  </BackgroundImages>
```

VIII.3 Summary

To get the images with SASPlanet you need a .hlg file with the info of your mesh boundaries.

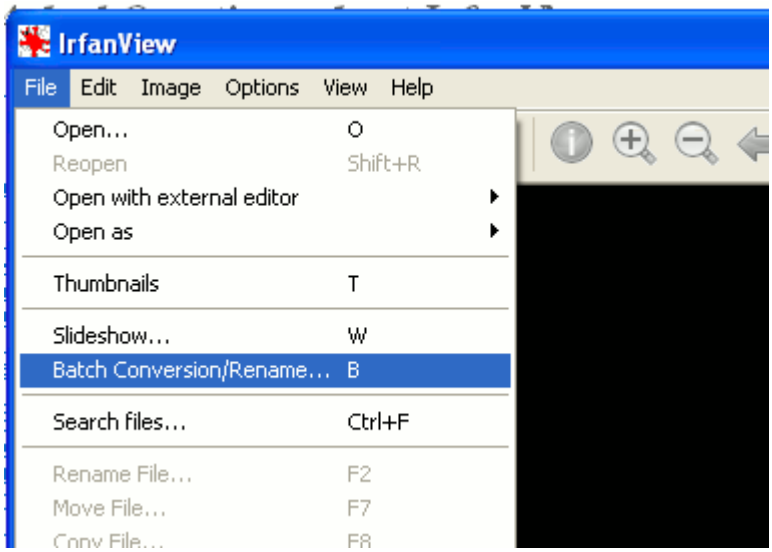
Selecting the right option on step s4_terrain (thus calling **create_hlg** script) makes that task: reads the mesh info (created from your .geo, saved as .msh and splitted with trocea_malla) and writes sasplanet.hlg

Then you open sasplanet.hlg with SASPlanet, following the steps explained in this chapter, and you get the .jpg files. For each .jpg image a .dat file has to be saved. This .dat files contain the terrestrial coordinates of the image limits. In step s10_split, a script (add_dat_to_geo) can read the .dat files and translates the terrestrial coordinates to BTB coordinates and creates a file ready to be inserted in the Venue.xml so the images are automatically placed in the background of the track. The output file is list_bi.txt. This file will be inserted in step s9_join(by script join_all) into the Venue.xml and you will get your images as background.

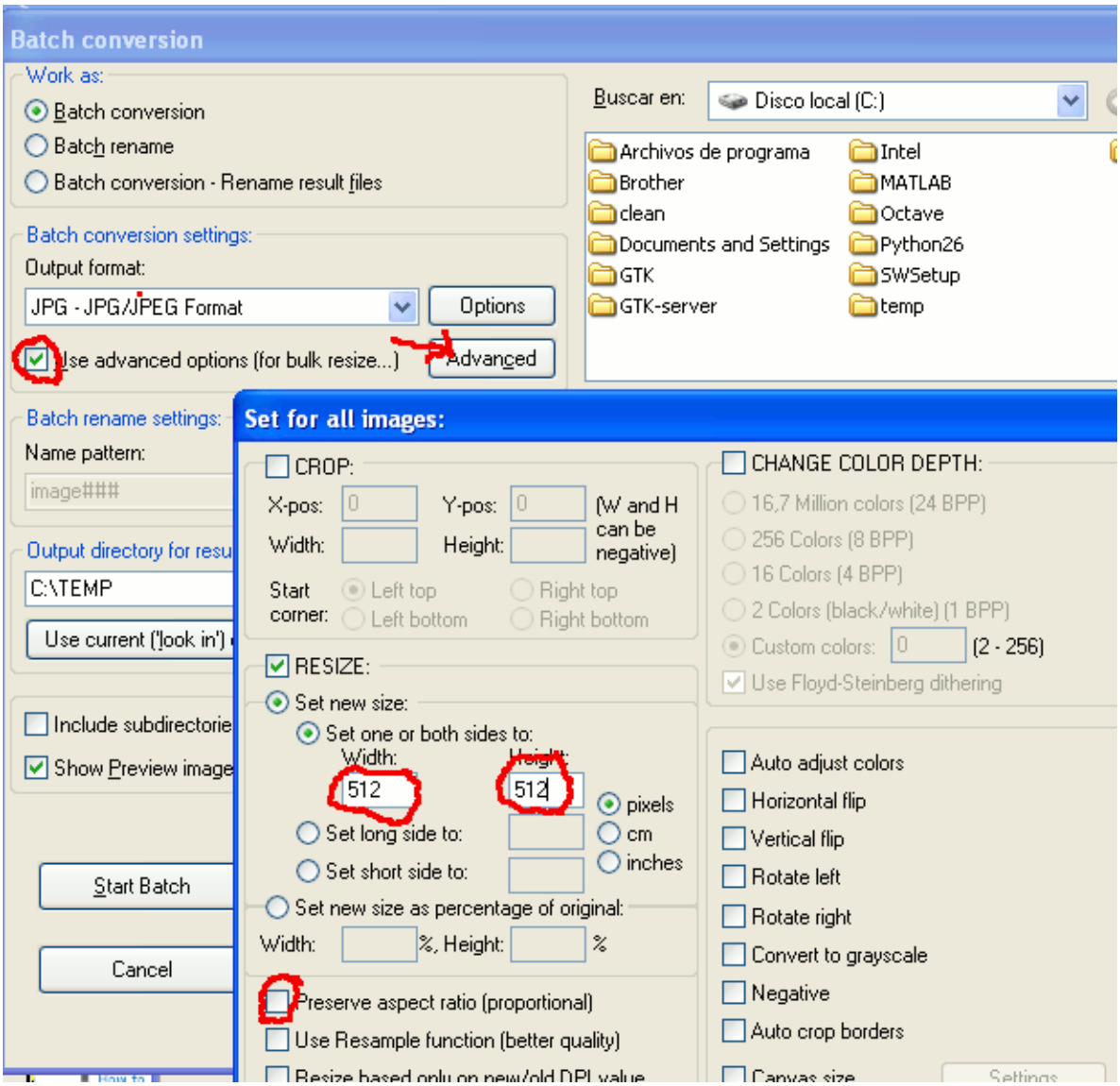
That above is a simple way to proceed. Now suppose you want to complicate things: you want a project with a terrain configured to make blending between a default texture and the background images (If that is the case, you must make sure you have created list_bi.txt by typing the right folder for the .dat files in s10_split step). Ok then, it is really easy: at the s10_split you just have to select option “Blend with background”(that calls procesar_elementstxt_mt with 3 parameters: dimensions of the matrix of images, and "1", to tell the script you want blending with background images, e.g procesar_elementstxt_mt(10,5,1)).

NOTE: I use Irfanview to batch resize .jpg with 2 power size, and DDSConvert to batch convert them to .dds. In Inrfanview I select the “batch conversion” option and then I set the dimensions desired for the images (I unmark the “preserve aspect ratio” option).

Background images

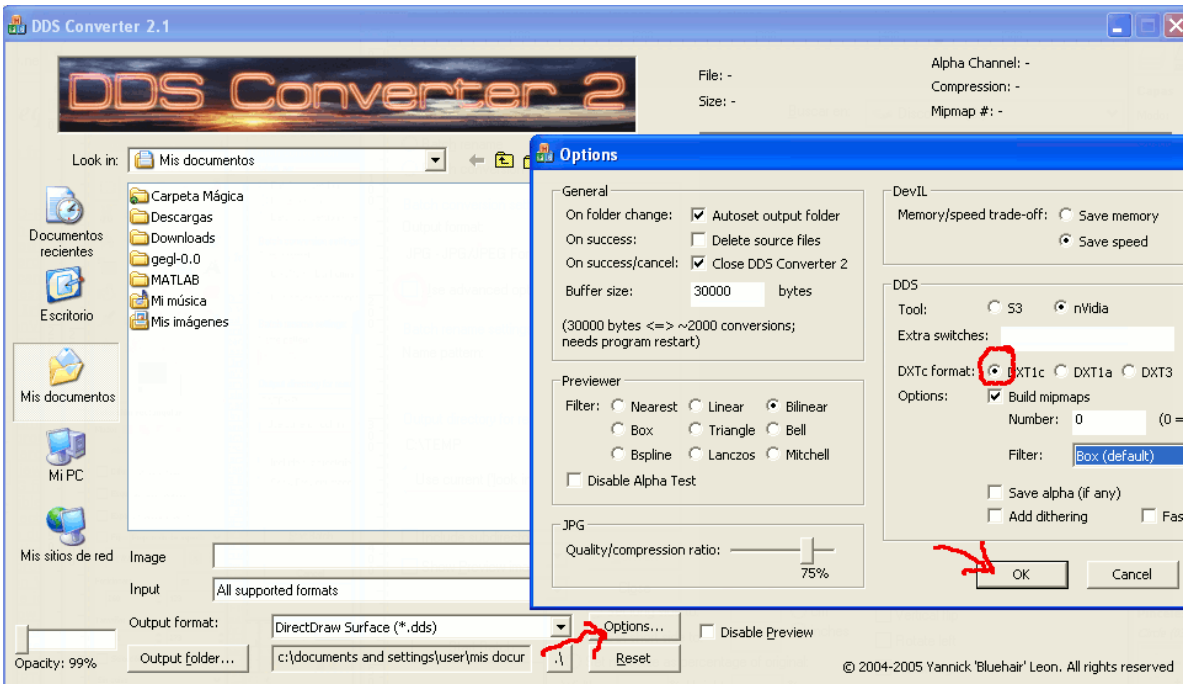


Background images



In DDSConvert I select the DXT1c format (background images shouldn't have alpha channel).

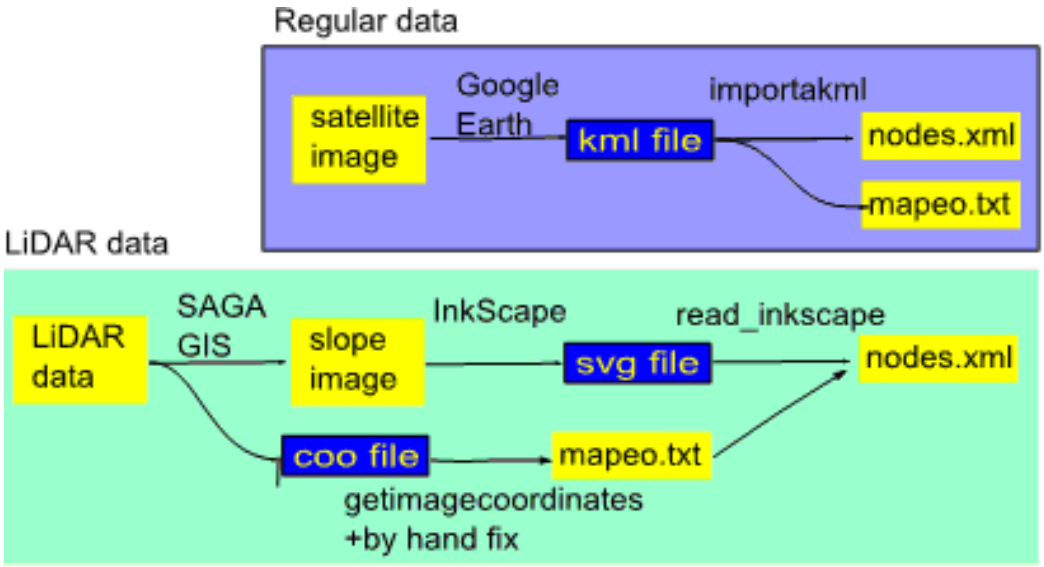
Background images



IX USING LIDAR DATA

Example of files used with InkScape (waste previous folder) and with the scripts:
http://www.mediafire.com/file/nu5hgpf9jbs4b24/waste_part1.7z
http://www.mediafire.com/file/gmy4n4xmzc5616s/waste_part2.7z

This type of elevation data has its own section on this document because they are treated in a special way. The difference with a normal project created with the scripts, is that as we have very good elevation resolution, we are **not** going to trust .kml routes created from satellite images. Instead of that **we will use elevation data to create an image** that shows us where the roads are and **we will create routes that match the roads on the image**. This way roads will be placed with high precision.



Step 1)

Use SAGA-GIS (http://sourceforge.net/projects/saga-gis/) to import .ASC files

- Import/Export - Grids\Import ESRI Arc/Info grid
- Terrain Analysis-Morphometry\Local Morphometry

If you have .las files, you can also use SAGA-GIS to import the data. For example, opening SAGA and following these steps:

- Import/Export-LAS\Import LAS Files
- Shapes\Point Cloud to Grid (1m grid)
- Grid Tools\Close gaps
- Terrain Analysis – Morphometry\Local Morphometry (slope)

And finally export the "**slope**" data using .jpg format:

- Import/Export Images\Export Image

Step 2)

Use FUSION (http://forsys.cfr.washington.edu/fusion/FUSION_Install.exe) to transform all the individual ASCII grids to DTM grid format

c:\FUSION**ASCII2DTM** N4414A3.dtm m m 1 0 0 0 N4414A3.asc

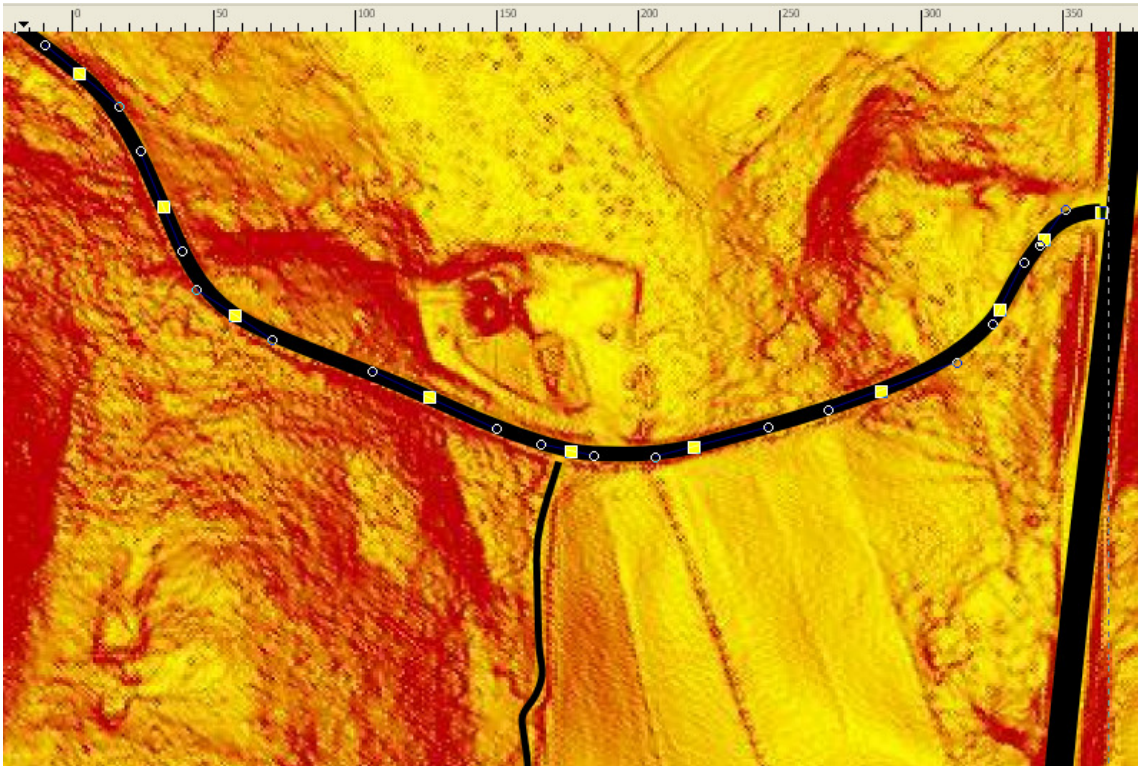
.dtm files must be copied in a folder called "lidar" placed in the same folder where we have the project folder or the father's and sons' folders in multitrack projects (e.g. if we have c:\project, then copy them to c:\lidar).

If we have a .las file I recommend filtering data to remove vegetation rebounds (example shown filters data with feet units to create 1m grid. If working with meters use 1 instead of 3.28084):

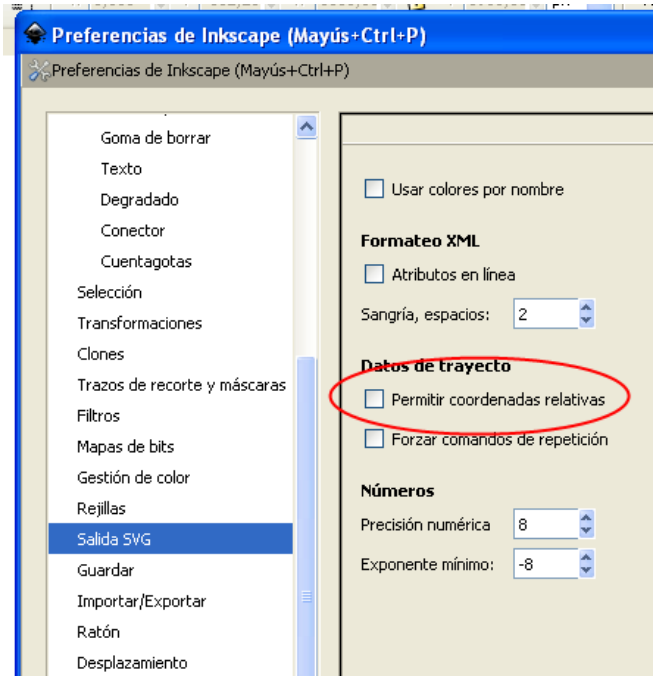
c:\FUSION\GroundFilter %1f.lda 3.28084 %1.las
c:\FUSION\GridSurfaceCreate %1f.dtm 3.28084 f f 1 0 0 0 %1f.lda

Step 3)

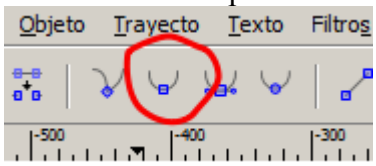
Import the **slope** .jpg with InkScape (**don't resize or move the image**) and create routes following the bright paths (low slope zones) that show where roads are. Create a path for each track in the project. Adjusting width in pixels to match the bright zones you can have a hint of the real width of the road (you may need that value later when using btb06).



NOTE: In the **general options** of InkScape, **SVG output** section, select that relative coordinates are **not** allowed. This way the bezier curves should contain only *M* and *C* letters, and not *m* or *c*.



May be after changing Inkscape general options existing paths still use relative coordinates, so I recommend using **Edit\Edit XML** within InkScape and make a symbolic change to the path data (add a space character and undo the change) and select "Apply"). After that operation **all the letters in the path should be uppercase: M or C**.
Path nodes should be "**s**(mooth)"-type. All the nodes can be forced to be s-type selecting all the nodes in the path (ctrl-a) and then clicking on the appropriate icon in the top bar (shift-s does it, but usually **InkScape crashes** with this operation. Save your file before you try).



Step 4)

In the .svg file you should have included **only one image**. If we have the coordinates of the image we can translate the routes to real coordinates. So for this image you have to create a .coo file containing the UTM coordinates of the image: (lasinfo from lastools, http://www.cs.unc.edu/~isenburg/lastools/download/lastools.zip, gives us that info if we use lastools, otherwise SAGA-GIS also informs of the characteristics of the data imported). Z coordinate (elevation) will not be used but the scripts require it for compatibility issues. The contents of the .coo file should be (X1, Y1, Z1, X2, Y2, Z2 must be replaced with actual values):

```
min x y z X1 Y1 Z1
max x y z X2 Y2 Z2
```

The .coo file can be used to create a first version of mapeo.txt

getimagecoordinates('fichero.coo',1).

The mapeo.txt can be fixed by hand to center the terrain (or may be change feet values to meters, if it is the case).

Step 5)

The .svg file must be processed along with the .coo of the image we used (in the example files I merged 6 images and created a .coo for the merged image):

read_inkscape ('pennsylvania.svg' ,'combinado.coo')

Before processing the .svg file I recommend opening it with a text editor and check that for every path only "s" letter is used (example: *sodipodi:nodetypes= "sssssssssssssssssssss"*). If we see other letters, like "c" we should open again the .svg with InkScape and force all the nodes of that path to be "smooth". "c" nodes at the start and end of the string are normal and should not be a problem.

The output from **read_inkscape** is a .xml file for each path included in the .svg. Those .xml

files can be renamed as nodes.xml, copied to the Venue folder of the scripts and then processed with btb06. But **create_sons** can do that step for us, as explained below. Before using the scripts mapeo.txt file has to be placed in the root folder of the project (or root folder of the father). Then we can go on with the use of the scripts the same way as if we had used importakml.

Step 6)

I download the scripts using subversion. I copy mapeo.txt in the root folder of the father.

Now **create_sons** can create the sons automatically from the .xml files located inside a folder (e.g. *create_sons('c:\temp\pennsylvania')*). It also copies the .xml files as Venue\nodes.xml for each son.

Nevertheless we can also do that task by hand:

- Use create_sons(N) in the root folder of the father to create N sons
- Copy .xml files as nodes.xml inside Venue folders of father and sons

Step 7)

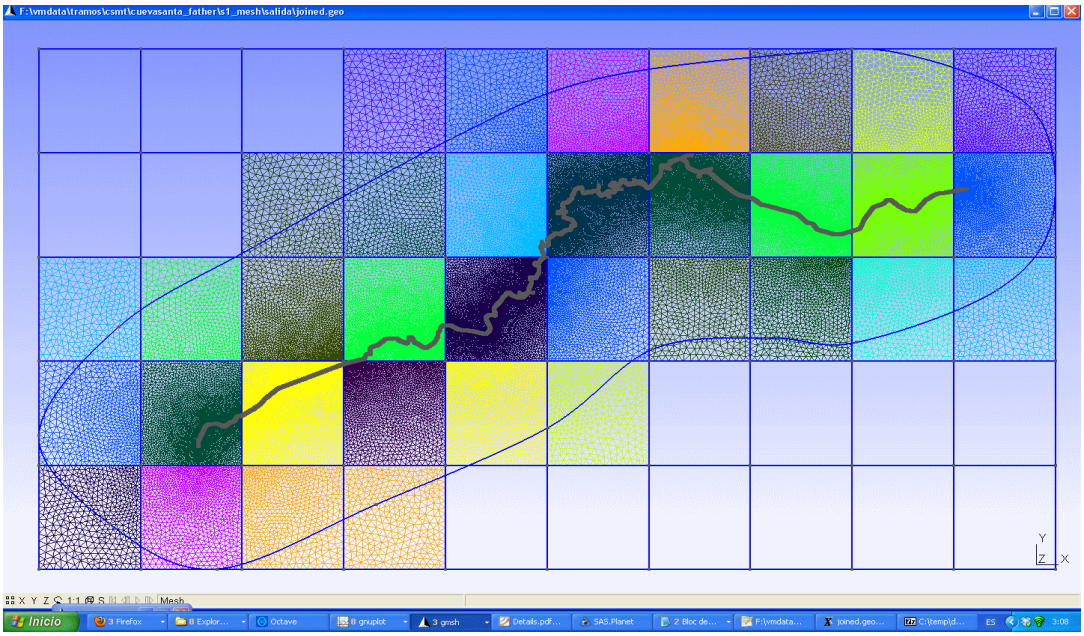
Use the scripts as usual. FUSION must be installed in C:\FUSION. FUSION will use the .dtm files to give elevation to the points when needed.
For creartrack1 it is recommended to use 1 as parameter: **creartrack1(1)**. This makes road elevation be calculated for points in the center line of the road. Otherwise road elevation profile is calculated taken into account points on the boundaries of the road.

NOTE: LiDAR data can be downloaded for a few USA states from:
http://lidar.cr.usgs.gov/LIDAR_Viewer/viewer.php

X ADVANCED USES OF THE SCRIPTS

X.1 Terrain that matches background images

You want to have terrain areas that fit exactly your background images, as I did with my Cueva Santa track. Ok, if that is the case, then first you need to insert the limits of your images into the joined.geo, the .geo you are working with. It is really simple: you run add_dat_to_geo inside s1_mesh folder and you select the option "add the grid to joined.geo". Then when you open joined.geo with gmsh and you will see the grid with the limits of the images you downloaded with SASPlanet. You have to do a little work merging those new lines with your existing terrain, creating smaller surfaces for the non-driveable zone. It is not difficult to do this step if you have a little practice using gmsh. In the image it can be seen the original mesh limits (the biggest spline curve), and the new meshes fitting the limits of the downloaded images.



Once you have your new mesh, you go on with the process as usual.

XI LIST OF COMMANDS

Script	Run it in directory
accept_mesh.m accepts anchors_carretera.msh as the definite mesh, skipping further processing with MeshLab. To be run instead of simplificar+MeshLab+juntar_mallas	**s4_terrain**
add_dat_to_geo.m Reads the .dat files created by SASPlanet while splitting a satellite image and creates a grid that can be saved as grid.geo or appended to salida\joined.geo. This script also created a list of background images (s1_mesh\list_bi.txt) ready to be included in the Venue.xml. NOTE: this script internally calls addgrid, so addgrid.hlg is overwritten.	**s1_mesh**
addgrid.m Creates a grid with .geo format. Two possibilities: **addgrid(numx,numz)** You want to view the available elevation data using a numx X numz grid **addgrid(xmin,xmax,zmin,zmax,step)** Creates a grid with the specified limits and line separation. If another parameter is added to the command, no matter its value, the list of points and lines created will be explicit (instead of using a "for loop"). addgrid creates a file called addgrid.hlg ready to be opened with SASPLanet to get the satellite images for that area.	**s1_mesh**
add_sobject.m Creates a list of SObjects to be inserted by hand in the Venue.xml. **add_sobject(num_points)** Parameter is the maximum number of points used by one SObject. Longer SObjects will be splitted.	**s7_walls_b**
addt.m Opens joined.geo and replaces the last occurrence of a Plane Surface followed by a Spline with the code to define that surface as Transinite	**s1_mesh**

btb_a_coor.m
Returns the terrestrial coordinates of a BTB point
`> [mapeo]=textread('mapeo.txt','%f');`
`> x=2380.47;`
`> z=-2350.67;`
`> [longitud altura latitud]=BTB_a_coor(x,0,z,mapeo)`
	base directory

btb06.m Creates the points in both borders of the road, where the road and the terrain will be linked (they are called anchors). Parameter is the separation between anchors on right and left side. It will affect the mesh created by mallado_regular	**venue**

coor_a_btb.m
Returns the BTB coordinates of a point given the terrestrial coordinates
`> [mapeo]=textread('mapeo.txt','%f');`
`> longit= -73.67;`
`> latit= 41.47;`
`> [x1 y1 z1]=coor_a_BTB(longit,latit,elevation,mapeo)`
	base directory

corregir.m For a given road, compares the elevation profile assigned using dar_altura and that obtained from elevation data (from lamalla.mat), and changes the terrain elevation data (lamalla.mat) to fit the elevation profile set with dar_altura. corregir also accepts a kml file as a parameter and uses its coordinates and altitude to change lamalla.mat. This could be useful if we have a kml with altitudes we trust, but dangerous as those altitudes could have an offset respect to the elevation data available. corregir('file.kml')	**s3_road**
creartrack1.m Gets elevation values for a road from its coordinates and elevation data (lamalla.mat)	**s3_road**
create_hlg Creates file s1_mesh\sasplanet.hlg (open it with SASPlanet) with the boundary coordinates (box) of anchors_carretera.msh (run trocea_malla before using create_hlg)	**s1_mesh**

create_sons Creates the folder's structure for several sons in a multi-track project. **create_sons(number_of_sons)** Creates son01, son02, etc. folders in the same folder where the father is located **create_sons('c:\temp\kmls',keep_names)** Creates one son for each kml located inside the directory used as first parameter. If keep_names is 0, folders will be named son01, son02, etc. If keep_names is 1, folders will keep the name of their respective kmls	**father's root directory**
cut_lamalla.m Reduces the size of lamalla.mat. Useful if data comes for a too big zone. cut_lamalla([xmin xmax],[zmin zmax])	**s2_elevation** **s2_elevation_b**
dar_altura.m Softens the output from creartrack1 and gives the nodes of the track their elevation and slope to fit that curve. **dar_altura(smooth_factor,pos_slope,neg_slope,step,interactive)** - smooth is a smoothing factor, the bigger the smoother - pos_slope and neg_slope are the maximum and minimum slopes allowed (1 means 45 degrees) - the final elevation profile is constructed using one point each "step" meters. Use a small value to preserve the profile's details, and a big value to smooth them. 25m is used if omitted - If interactive==0, the script doesn't give the user the option to edit the profile by hand and exists	**s3_road**
fix_project.m This script reads tracks' point coordinates contained in joined.geo and creates new files porcentajes.mat and anchors.mat for all the tracks in the project. This way may be a project where tracks and terrain are not correctly linked any more can be fixed. After running this script, all the steps from juntar_mallas to the end must be redone. This script won't work correctly if sons have been added or removed since joined.geo was created. Make a backup before using this script.	**s1_mesh\salida**

importakml.m Reads a kml file and from it creates a mapping between terrestrial and BTB coordinates. **importakml(kml_file)** All the original points of the kml will be converted to nodes of the road **importakml(kml_file,'decimate',factor)** Keeps 1 from every "factor" points of the kml as nodes of the road. For example if the kml has 100 points and factor==2, the road will have 50 nodes. **importakml_old(kml_file,tolerance)** Uses the old "approach". An *ideal* smooth road with a huge amount of nodes that follows the coordinates of the kml file (using akima splines). Finally some nodes are removed. A node is removed if removing it doesn't deviate the road more than "tolerance" meters from the "ideal path"	**s0_import**
join_all.m Final step of the process. Joins all the tracks, terrain, pacenotes and walls, creating a file called Venue.xml. To open this file good luck and WP.zip Xpack are needed.	**s9_join**
join_geos.m Joins the anchors_carretera.geo files created with mallado_regular for all the projects, creating file joined.geo inside s1_mesh\salida folder. This file should be edited with gmsh.	**s1_mesh**
juntar_mallas.m Reads i.ply, c.ply and n.ply from s4_terrain\salida and joins them in one single mesh (files anchors_contaltura.txt and elements.txt)	**s4_terrain**
leehgt.m Creates lamalla.mat from a .hgt file (1 degree x 1 degree) **leehgt(fichero,latitud,longitud)** Data extension is from latittud to latitud+1 and from longitud to longitud+1	**s2_elevation** **s2_elevation_b**

leehgt2.m
The same as leehgt, but joins 2 adjacent .hgt files
`leehgt2(file1,latit1,longit1,file2,latit2,longit2)`
`if latit1==latit2, longit1 should be <longit2`
`if longit1==longit2, latit1 should be <latit2`
	s2_elevation
	s2_elevation_b

leer_gridfloat.m Creates lamalla.mat from gridfloat file. First parameter is the .hdt and second one is .flt	**s2_elevation** **s2_elevation_b**
leetif.m Creates lamalla.mat from a geotiff file	**s2_elevation** **s2_elevation_b**

listc.m Reads salida\joined.geo and creates a file called listc.geo with the id numbers of all the Plane Surfaces created inside joined.geo after its creation (last line of joined.geo after its creation is the reference used by listc)	**s1_mesh**

make_grid.m
Creates several files containing a regular grid of points with terrestrial coordinates. Those files should be "raised" with BTBLofty or a similar application and save with a different name: grid001.kml should be saved in the same folder as grid001_relleno.kml
`make_grid(xmin,xmax,zmin,zmax,step,file_size)`
Parameters are x and z minimum and maximum values, and distance between points of the grid. Maximum file_size depends upon the application to be used. 5000 is recommended for BTBLofty.
 	
Another possibility for make_grid is creating a kml route and asking make_grid to create a grid that covers all that route:
`make_grid('limits.kml',step)`
	s2_elevation
	s2_elevation_b

mallado_regular.m Creates a terrain mesh on both sides of the road. Position of road borders (anchors) is taken from btb06 output. Besides the road a terrain of a specified width will be created, splitted in the transversal direction into the desired number of panels. Terrain width (meters) is the first parameter and the number of panels is the second one. If you want to try a regular pattern (tranfinite) for all the driveable zone, use 1 as 3rd parameter. Otherwise use just 2 parameters	**s1_mesh**
muro_pegado.m Creates walls on both sides of the road (from start to end). List of walls can be found in salida folder and should be inserted by hand inside the Venue.xml file (updating the total walls count, if needed) **muro_pegado(tam_wall,offset)** Parameters are the limit of points per wall and the displacement in meters in the outside direction from the road border (the width specified as btb06 parameter is used to compute border position)	**s7_walls_b**
msh2btb Creates a BTB terrain from file **s10_split\salida\anchors_carretera.msh**. The created terrain will not be blended with background images and it won't be connected to the roads. This command assumes you use it instead of procesar_elementstxt_mt **ply2btb(cells_x,cells_z)** Will split the terrain using a cells_x X cells_z grid	**s10_split**

pacenotes.m Gets the track shape from a driveline.ini file. Output from thius script will be used by pacenotes_2	**pacenotes**
pacenotes_a.m Gets the track shape from anchors created by btb06. Output from this script will be used by pacenotes2_a	**pacenotes**
pacenotes2.m Creates a new pacenotes.ini file using the old one and the output from pacenotes.m **pacenotes2(sensibility,distance)** Parameters are the sensibility for curve detection and the distance you want to move the pacenotes to the start of the road. 10 means 50m.	**pacenotes**
pacenotes2_a.m Creates a list of pacenotes in BTB format ready to be inserted inside the Venue.xml. Join.all looks for this pacenotes and if they exist, includes them inside Venue.xml. Parameters are the same as pacenotes2	**pacenotes**
partir_track.m Splits a track into several segments. Reads split points from pos_nodes.txt	**s10_plit**
plot_lamalla.m Plots the contents of salida\lamalla.mat as a surface.	**s2_elevation** **s2_elevation_b**
ply2btb Creates a BTB terrain from file **s10_split\salida\n.ply**. The created terrain will not be blended with background images and it won't be connected to the roads. This command assumes you use it instead of procesar_elementstxt_mt **ply2btb(cells_x,cells_z)** Will split the terrain using a cells_x X cells_z grid	**s10_split**
poner_muro.m Creates walls in the boundary between driveable and non-driveable zones. Walls are automatically included inside Venue.xml by join_all	**s7_walls**
procesar_elementstxt_mt.m Creates the terrain in BTB format from the mesh created by juntar_mallas and the output from partir_track. By default terrain is splitted using a 10x10 grid, but user can choose another grid size. **procesar_elementstxt_mt(cells_x,cells_z,do_mapping)** Will split the terrain using a cells_x X cells_z grid, and If do_mapping is 1, terrain will be created with background images blending (see add_dat_to_geo).	**s10_split**

procesar_nodostxt.m Nodes of anchors_carretera.msh mesh receive a elevation value taken from lamalla.mat, if possible, or lamalla2.mat	**s4_terrain**
process_sons.m This script processes all the sons in a multitrack project. It should be first edited to set the desired values for the parameters of the scripts called.	**base directory of father**
raise_kml.m Calls a Google Earth API to get elevation values for the gridXXX.kml files inside s2_elevation\salida folder. Output files will be named gridXXX_relleno.kml and will be ready to be processed with read_grid This script needs Google Earth and Python27 installed in the system, Read instructions for installation inside documentation folder.	**s2_elevation** **s2_elevation_b**
read_grid.m Reads the gridXXX_relleno files and created lamalla.mat, with all the elevation info collected	**s2_elevation** **s2_elevation_b**
readkml.m Translates a route from a kml file to a curve in gmsh format and BTB coordinates. Output file is written in salida folder, with the same name as input, but .geo extension. **readkml('file.kml',curve)** Second parameter can be "t", for adding straight lines, "s" for adding a spline, or "st" for adding both. Not using a second parameter means adding no curve (just points).	**s1_mesh**
readkml_bat.m Calles readkml for all the .kml files found in the specified folder. **readkml_bat('d:\folder',curve)**	**s1_mesh**
simplificar.m Splits anchors_carretera.msh in three parts that should be processed with MeshLab: intocables.ply, conducibles.ply and noconducibles.ply Also creates a folder nc_splitted with a separate .ply file for each surface of the non-driveable zone, so it is possible to simplify them individually.	**s1_mesh**
split_agr.m Splits a file with extension .ASC into several smaller files with the same format but extension .AGR. Example: `split_agr('MDT05-0667-H30',5)` splits `MDT05-0667-H30.ASC` file into 5x5=25 files with extension .AGR	**agr**
split_track.m Selects the points for splitting a track into several segments. Writes those points in file pos_nodes.txt, allowing the user to change them before running partir_track	**s10_split**

start.m Calls importakml to process s0_import\road.kml Calls make_grid with s2_elevation\limits.kml as parameter Calls make_grid with s2_elevation_b\limits_b.kml as parameter Default steps for make_grid are 25 and 75m, respectively. By default each gridXXX.kml is limited to 5000 points **start** **start(step,step_b,file_size)** Parameters are the steps in meters used by make_grid and the number of points per kml.	
terrain_noise.m Adds a random value to the elevation of the nodes of the terrain. Random value will be in the range specified. Use this script just before join_all **terrain_noise([ymin ymax])**	**s4_terrain**
trocea_malla.m Splits anchors_carretera.msh into 2 parts: list of mesh nodes (nodos.txt) and triangles (elements.txt)	**s1_mesh**
vercontorno.m Shows a contour plot using the terrain elevation data (lamalla.mat) and the road position (output from btb06)	**s2_elevation** **s2_elevation_b**

XII ADDITIONAL NOTES

XII.1 GUI

Sometimes the interface refuses to perform an action for no reason. If that happens, close the GUI, open it again and try again. If that doesn't work you can close the GUI and type the commands on the text-mode octave window. The error messages on screen will help to diagnose the problem.

If you get an error message related to iconv.dll may be copying that file from "c:\GTK\bin" to "c:\GTK-server" can solve it.

XII.2 Gmsh and multitrack

1) 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

2) **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.

3) 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).

4) 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.

5) 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.

6) 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. 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".

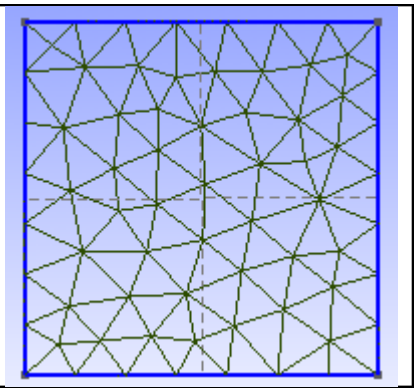
7) Don't run **listc** after creating the Plane Surface for the non-driveable zone. If you do that and use the file, remember to remover that surface from the list or you will get that terrain zone duplicated inside your project.

8) 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.

9) 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?

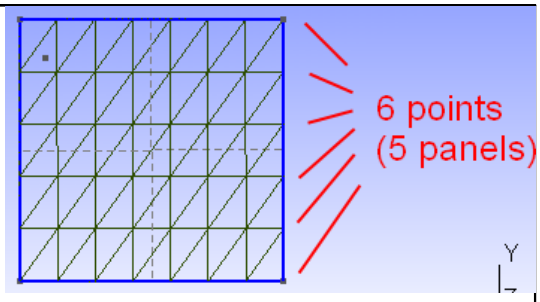
Example:

```
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.

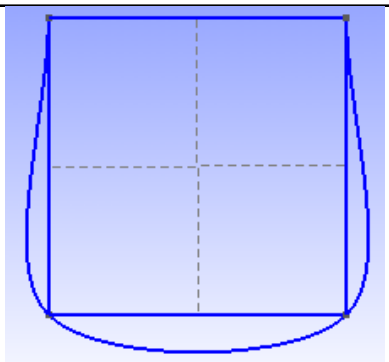
```
Point(1) = {-1, 1, 0};
Point(2) = {-1, -1, 0};
Point(3) = {1, -1, 0};
Point(4) = {1, 1, 0};
Line(1) = {1, 4};
Line(2) = {4, 3};
Line(3) = {3, 2};
Line(4) = {1, 2};
Line Loop(5) = {2, 3, -4, 1};
```



```
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.

XIII LINKS

XIII.1 Videoutorials

Install of the Grahical Interface

http://www.mediafire.com/?myip7mte8rjgpgm
(extract files and open gui.htm with a web browser. Flash player required.)

(Wink Source: http://www.mediafire.com/file/w0zmi53e8cu9nm2/gui.wnk)

2 Tracks using AGR elevation data

Videotutorial 1/3: http://www.mediafire.com/file/afdod089q47dgvc/agr_example.7z
(wink source: http://www.mediafire.com/file/da7rj6xvvdfa8p0/agr_source.7z)

Videotutorial. 2/3: http://www.mediafire.com/?8d0s63wqaqu24xh
(wink source: http://www.mediafire.com/file/kb8t9gqd93kghk9/agr_source_b.7z)

Videotutorial. 3/3: http://www.mediafire.com/file/7rspbi46lf207ya/agr_example_c.7z
(wink source: http://www.mediafire.com/file/y263vv5g4yuiwn4/agr_source_c.7z)

Parlesportes' videotutorials.

They are great, even if you don't speak french language.
Installation_et_Kmls.7z (242.43 MB): http://www.multiupload.com/OVRS2KGRAM
Simple_Route_Partie_1.7z (380.23 MB): http://www.multiupload.com/PYYRTMLNS6
Simple_Route_Partie_2.7z (581.83 MB): http://www.multiupload.com/7RN3UE7ICN
multi_route.7z (131.4 MB): http://www.multiupload.com/ZDV05IWIQ8

XIII.2 Forums

1) Méthode zaxxon in Rallyesim forum (french or english): http://forum.rallyesim.fr/viewtopic.php?f=51&t=3025
2) SISCO's forum (spanish or english): http://btbtracks-rbr.foroactivo.com/dudas-sobre-el-metodo-zaxxon-f37/
3) simracing's forum (spanish): http://foro.simracing.es/bobs-track-builder/4093-dudas-y-preguntas-del-ma-zaxxon.html
4) devtrackteam (english): http://devtrackteam.solorally.it/viewtopic.php?f=28&t=69

XIII.3 Old help files

http://www.multiupload.com/5B4KUL1ZO0
or
http://www.mediafire.com/?iuh4bp81hlnsf77