**What is a mesh?**

A finite element mesh of a model is a tessellation of its geometry by simple geometrical elements of various shapes (in Gmsh: lines, triangles, quadrangles, tetrahedra, prisms, hexahedra and pyramids), arranged in such a way that if two of them intersect, they do so along a face, an edge or a node, and never otherwise. This defines a so-called “conformal” mesh. Gmsh implements several algorithms to generate such meshes automatically. All the meshes produced by Gmsh are considered as “unstructured”, even if they were generated in a “structured” way (e.g., by extrusion). This implies that the mesh elements are completely defined simply by an ordered list of their nodes, and that no predefined ordering relation is assumed between any two elements.

Source: <https://gmsh.info/doc/texinfo/gmsh.html#Mesh>

More things about a mesh:



Source: <https://gmsh.info/doc/texinfo/gmsh.html#Source-code-structure>

Useful likes in that section that we may need: (still guessing how ☹)

* How to generate nodes: <https://gitlab.onelab.info/gmsh/gmsh/blob/gmsh_4_9_5/src/geo/MVertex.h>
* How to generate triangles: <https://gitlab.onelab.info/gmsh/gmsh/blob/gmsh_4_9_5/src/geo/MTriangle.h>
* **My understanding:**

We need to generate a file in mesh2 format that contains something like this:

A picture containing graphical user interface

Description automatically generated

Our “Teselado” must be the list of triangles he mentioned, it ca be 2 triangles or 100, we choose, I gues.

*Does it make sense to you?*

**Useful links for a better understanding of what we are dealing with:**

The file format we are asked should contain all these elements:

<https://gmsh.info/doc/texinfo/gmsh.html#MSH-file-format-version-2-_0028Legacy_0029>

but as per his email, ours can contain just the three elementary ones: format, nodes and elements. So we “just” need to focus on this:

$MeshFormat

*version-number* *file-type* *data-size*

$EndMeshFormat

$PhysicalNames

*number-of-names*

*physical-dimension* *physical-tag* "*physical-name*"

…

$EndPhysicalNames

$Nodes

*number-of-nodes*

*node-number* *x-coord* *y-coord* *z-coord*

…

$EndNodes

$Elements

*number-of-elements*

*elm-number* *elm-type* *number-of-tags* < *tag* > … *node-number-list*

…

$EndElements

where

version-number

is a real number equal to 2.2

file-type

is an integer equal to 0 in the ASCII file format.

data-size

is an integer equal to the size of the floating point numbers used in the file (currently only data-size = sizeof(double) is supported).

number-of-nodes

is the number of nodes in the mesh.

node-number

is the number (index) of the n-th node in the mesh; node-number must be a postive (non-zero) integer. Note that the *node-number*s do not necessarily have to form a dense nor an ordered sequence.

x-coord y-coord z-coord

are the floating point values giving the X, Y and Z coordinates of the n-th node.

number-of-elements

is the number of elements in the mesh.

elm-number

is the number (index) of the n-th element in the mesh; elm-number must be a postive (non-zero) integer. Note that the *elm-number*s do not necessarily have to form a dense nor an ordered sequence.

elm-type

defines the geometrical type of the n-th element: see [MSH file format](https://gmsh.info/doc/texinfo/gmsh.html#MSH-file-format).

number-of-tags

gives the number of integer tags that follow for the n-th element. By default, the first tag is the tag of the physical entity to which the element belongs; the second is the tag of the elementary model entity to which the element belongs; the third is the number of mesh partitions to which the element belongs, followed by the partition ids (negative partition ids indicate ghost cells). A zero tag is equivalent to no tag. Gmsh and most codes using the MSH 2 format require at least the first two tags (physical and elementary tags).

node-number-list

is the list of the node numbers of the n-th element. The ordering of the nodes is given in [Node ordering](https://gmsh.info/doc/texinfo/gmsh.html#Node-ordering).

number-of-string-tags

gives the number of string tags that follow. By default the first string-tag is interpreted as the name of the post-processing view and the second as the name of the interpolation scheme. The interpolation scheme is provided in the $InterpolationScheme section (see below).

I believe these must be useful, but still trying to guess how we can use this information ☹

And this is an example of what we need to create. It is done with two quadrangles:

$MeshFormat

2.2 0 8

$EndMeshFormat

$Nodes

6 six mesh nodes:

1 0.0 0.0 0.0 node #1: coordinates (0.0, 0.0, 0.0)

2 1.0 0.0 0.0 node #2: coordinates (1.0, 0.0, 0.0)

3 1.0 1.0 0.0 etc.

4 0.0 1.0 0.0

5 2.0 0.0 0.0

6 2.0 1.0 0.0

$EndNodes

$Elements

2 two elements:

1 3 2 99 2 1 2 3 4 quad #1: type 3, physical 99, elementary 2, nodes 1 2 3 4

2 3 2 99 2 2 5 6 3 quad #2: type 3, physical 99, elementary 2, nodes 2 5 6 3

$EndElements

This must be useful as well, but don’t know yet how to do it with our triangles ☹ but we should be generating something like this with python.

More information about the MSH file format:

<https://gmsh.info/doc/texinfo/gmsh.html#MSH-file-format>