Rhino to Wrml STL user instructions

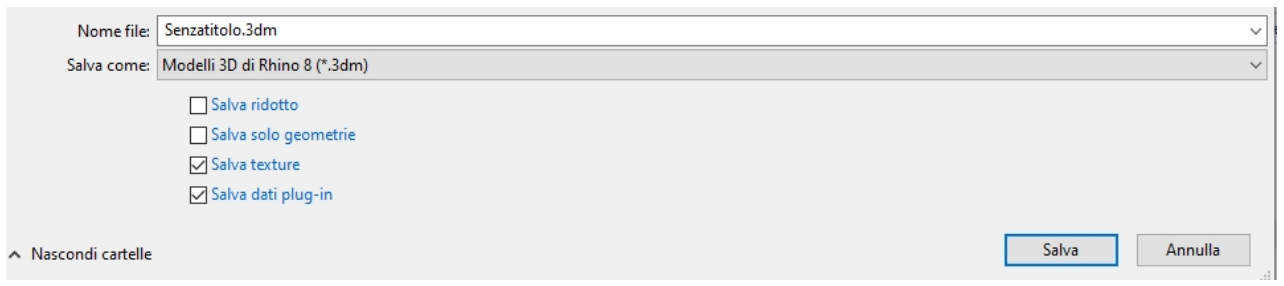
This plugin was written in Python version 2 to be compatible with Rhinoceros versions 7 onwards.

It automatically exports objects on Rhino layers, naming the resulting files in a manner consistent with the name of the source file, the layer on which the object to be exported is located, and the file version.

The aim is to speed up and automate repetitive operations that take time away from modelling and can easily generate errors.

If you launch the plugin from a new file, which has never been saved, you will be asked to save it to any local location.

An example of this is when you import objects from another software, or when you copy and paste them from a different Rhino session into an empty file.
When you start modelling from scratch in a new Rhino file, it is unlikely that you will get so far into the modelling process that you need to launch the plugin without ever having saved the Rhino file.



The starting position will be that of the last file saved by Rhino.
The STL and WRML files created will be saved in the same location as the *.3dm file.

The name you give the file is important because the objects in your file will be saved according to this name.
This will be crucial for keeping your various versions of your work organised.

"Esc" key
Some processes can cause very long or infinite loops. In this case, to stop the script from running, press the **ESC** key on your keyboard.

You have probably accidentally exploded the polysurface or mesh, generating a very high number of surfaces or polygons.

If this happens for other reasons, try to reproduce the error and understand why it happened, then send me an email, perhaps attaching your file, so I can fix it.

Creating the names of exported files:

The names of the exported files will be formed as follows

[part of the 3dm file name before the last "_"][number]["_"+ layer name]

Main rules for naming Rhino files:

You are free to name your file as you wish, but to get the most out of this script, you need to follow a few simple rules.

Special characters such as "< > : " / \ | ? * "etc. will be ignored in the name of exported files, as they are assumed to be typing errors! In many cases, the system will prevent you from entering certain characters because they would conflict with the operating system itself.

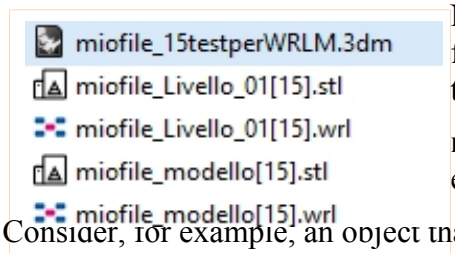
With this in mind, we can ideally divide the file name into three parts [any text

including "_" within it][_][number][any text except "_"], for example

naming the file:

myfile_15testforWRML.3dm the

following files will be saved:



Note that everything after "**_15**" is ignored in the names of the files created. This is useful for creating a copy of the Rhino file to use for exporting WRLM without "ruining" the Rhino file.

myfile_15testforWRML.3dm could be a modified copy for exporting the file **myfile_15.3dm**, which then remains intact.

Consider, for example, an object that requires the application of a texture on part of its surface, which is therefore divided and is no longer a closed polysurface.

I will come back to this later.

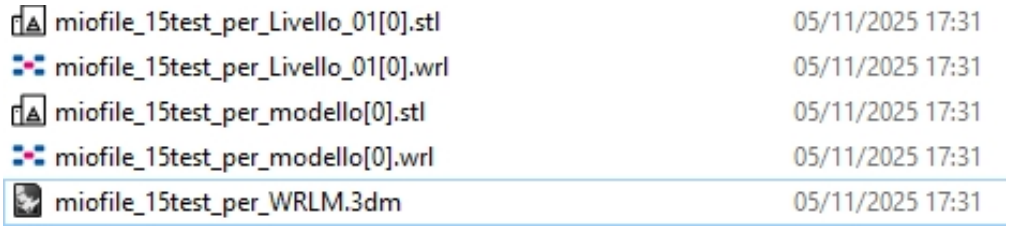
The number after the last "_" is used for the version number.

Note that the underscore used to separate the part to be ignored from the actual name is the last one on the right... therefore, the part after the number must not contain any underscore symbols.

For example, a name like this:

myfile_15test_for_WRML.3dm

The exported files will be:



The **WRML** in the **.3dm** file name after the last _ was ignored, and since there was no number from which to derive the version, "0" was added.

Creation of exported file names:

The names of the exported files will consist of

part of the .3dm file name before the last "_" + layer name + [number after "_"] in the

previous examples, the objects are present in the "**level01**" and "**model**" layers

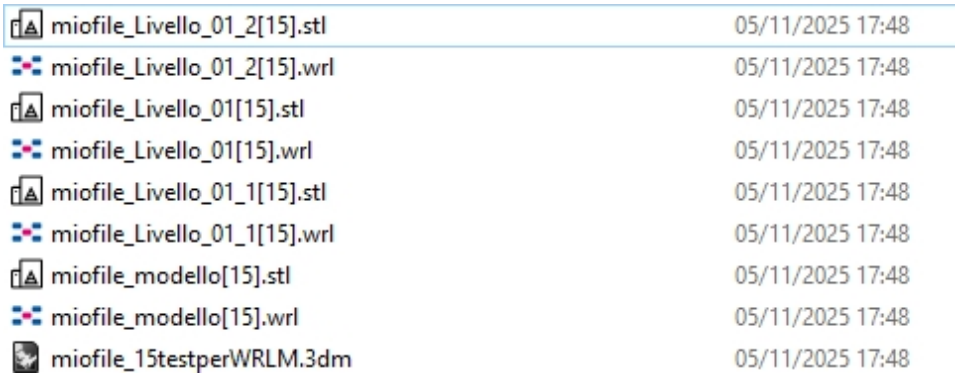
The wrml format is used for 3D colour printing as it retains the colours and textures applied to the object.

The colour of the object can be applied to the layer or directly to the object.

The colour applied to the object takes precedence over the colour applied to

the layer.

There may be multiple objects on the same layer, in which case the names of the objects exported from the layer will have a sequential number.



In the example, the "Level_01" layer has 3 objects.

Processed objects and objects ignored in the export:

The plugin will export only those objects that are visible at the time of export to wrml and stl formats, whether they are selectable or locked.

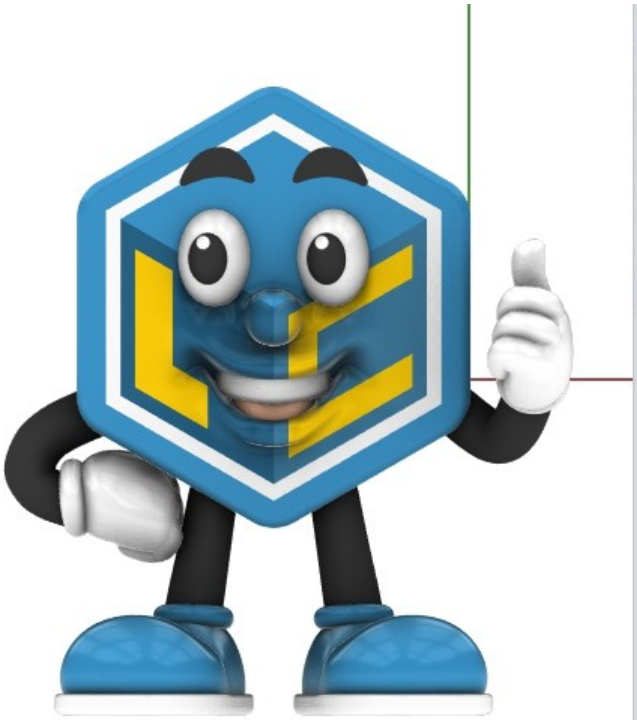
Locked but visible objects will be unlocked beforehand.

Invisible layers will be ignored along with everything inside them.

Organising files on layers for correct export

For optimal use of the plugin, it is best to organise the file by dividing the objects into layers and

applying the main colour to the layer.



Livello	Atti	Bloc	Colore	Materiale
colore giallo				/116 C (GIALLO)
blu1				/BLUSCURO
blu2				/BLUmedio
blu 3				/BLUchiaro
rosso				/GenericVray _162(rosa)#1
bianco				/White
nero	✓			**/Pantone NERO**
PIANO APPOGGIO				
LUCI				

In this example, the character model is intended for colour matching, and each part of the model that represents colour has channels inside it for the colour to pass through.
It is organised so that each part is on the level named according to its colour, so that the parts then reflect this in their names.

Running the plugin on this file will generate 7 pairs of files. The LIGHTS and SUPPORT PLANE levels are turned off and will therefore be ignored.

Personaggio3d_nero[15].wrl	05/11/2025 18:34
Personaggio3d_nero[15].stl	05/11/2025 18:34
Personaggio3d_bianco[15].wrl	05/11/2025 18:34
Personaggio3d_bianco[15].stl	05/11/2025 18:34
Personaggio3d_rosso[15].wrl	05/11/2025 18:34
Personaggio3d_rosso[15].stl	05/11/2025 18:34
Personaggio3d_blu_3[15].wrl	05/11/2025 18:34
Personaggio3d_blu_3[15].stl	05/11/2025 18:34
Personaggio3d_blu2[15].wrl	05/11/2025 18:34
Personaggio3d_blu2[15].stl	05/11/2025 18:34
Personaggio3d_blu1[15].wrl	05/11/2025 18:34
Personaggio3d_blu1[15].stl	05/11/2025 18:34
Personaggio3d_colore_giallo[15].wrl	05/11/2025 18:34
Personaggio3d_colore_giallo[15].stl	05/11/2025 18:34
Personaggio3d_15.3dm	05/11/2025 18:14

Although this export is useful for creating the various co-moulding steps, it may be useful to have a single WRLM file for printing a prototype.

To do this, simply merge all the parts that form a single 'final' object into a group.

Presence of grouped objects:

In the presence of groups, the plugin behaves differently depending on whether the grouped objects are all on the same layer or on different layers.

If the objects are all on the same layer, the group is considered as one object and will take the name of the layer with a sequential number if there are other groups or objects on the same layer.

If at least one grouped object is on a different layer, you will be asked to save it with a different name, as it is likely to be a copy.

The alternative name suggested is that of the layer highest in the layer tree.

The files **Personaggio3d_UNITO[15]**.wrl and **Personaggio3d_UNITO[15].**stl will be added to the files.

If there are groups with objects on different layers and free objects at the same time, these will have a sequential number, while for the grouped layer, you will be asked to rename it or call it with the layer name followed by a higher sequential number so as not to conflict with the previous ones.

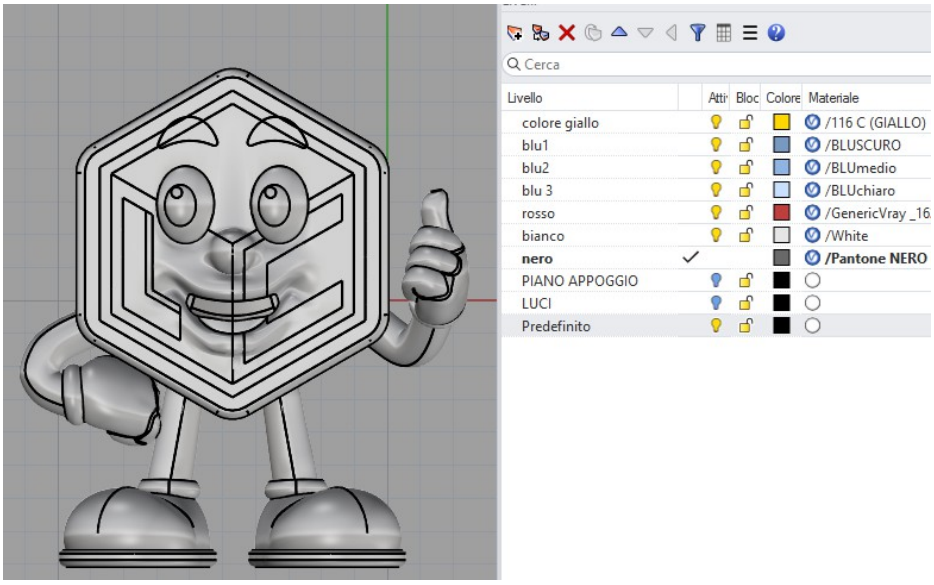
If there are multiple groups on one level, it is also necessary to distinguish whether the objects in the group are all on the same level, in which case it will take the name of the level, possibly followed by a sequential number, or whether the grouped objects are multi-level, in which case you will be asked to rename the file.
in the group are all on the same level, in which case it will take the name of the level, possibly followed by a sequential number, or if the grouped objects are multi-level, in which case you will be asked to rename the file.

In any case, care must be taken when renaming the file, as if the generated name is the same as another, it will be overwritten (a backup copy will be made, but only of the previous one!
So be careful!)

Grouping objects that are on different levels, if the colour is applied to the level, can be very useful in a phase of the work prior to the final one, where the channels for passing colours into the moulds are not yet present but you still want to obtain a single object file in wrml format for colour printing.

In this example, the model consists of a closed polysurface on the "default" layer.

I have already created the layer tree with the colour for the various parts, colour applied to the layer.



the LIBBY character into a single closed polysurface

By extracting the surfaces related to a colour and moving them to the relevant layer, grouping them into a single group and launching the plugin, a copy of the file in wrml and stl format of the joined and coloured character will be created.

Since it is a group with parts on different levels, you will be asked to rename the file as an alternative to the one proposed.

Note that it is not necessary to have a single closed polysurface to obtain a pair of files that are valid for printing and therefore usable in this preliminary version.

The various parts (colours) can be objects that overlap each other. If these are arranged on colour-related layers, or have a material applied, and are grouped together, the plugin will still export a pair of files.

In this case, however, there is a greater chance that the exported file will contain errors.

It is good practice to pass the exported files through export software, which in most cases will eliminate the errors.

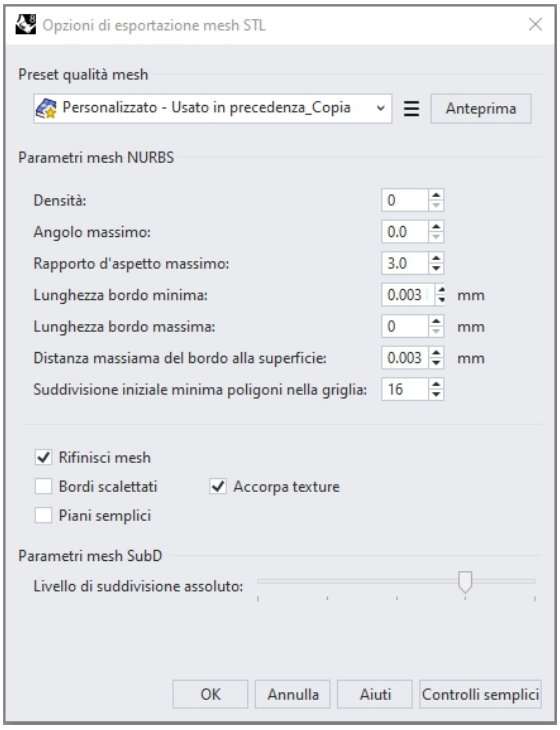
Parameters for STL and WRML export

The plugin allows the export of any geometry present on Rhino layers in wrml and stl formats, which are polygonal formats.

Objects created or imported in polygonal format are exported as they are, without reducing or optimising the number of polygons.

Objects with Nurbs geometry, surfaces, polysurfaces and SubDs are transformed into polygonal MESH during export according to these parameters:

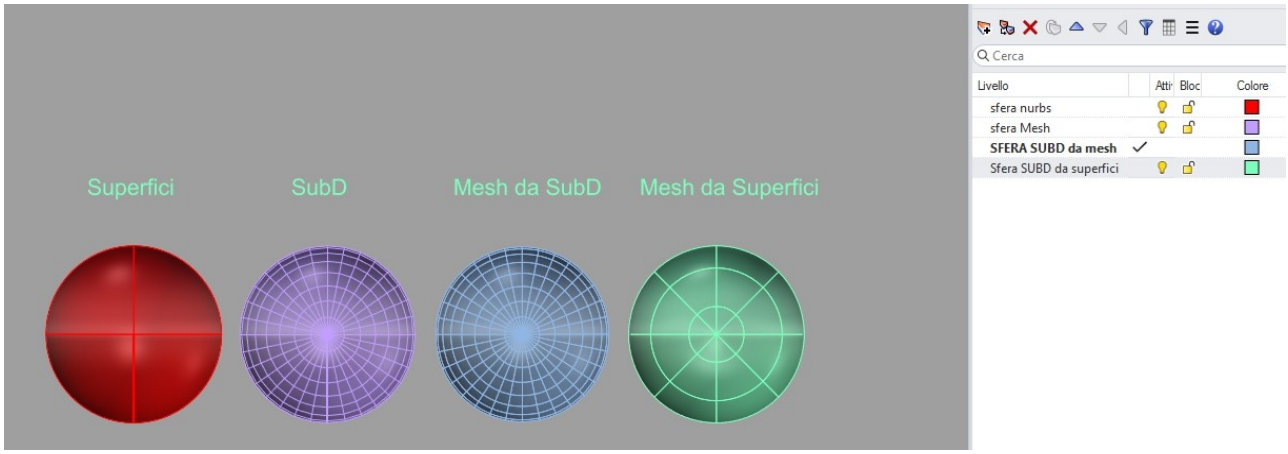
SFERE_SFERA_SUBD_da_mesh[2].wrl	06/11/2025 10:06	File WRL	31.754 KB
SFERE_SFERA_SUBD_da_mesh[2].stl	06/11/2025 10:06	Oggetto 3D	12.401 KB
SFERE_Sfera_SUBD_da_superfici_4[2].wrl	06/11/2025 10:06	File WRL	2.796 KB
SFERE_Sfera_SUBD_da_superfici_4[2].stl	06/11/2025 10:06	Oggetto 3D	1.101 KB
SFERE_sfera_Mesh[2].wrl	06/11/2025 10:06	File WRL	116 KB
SFERE_sfera_Mesh[2].stl	06/11/2025 10:06	Oggetto 3D	47 KB
SFERE_sfera_nurbs[2].wrl	06/11/2025 10:06	File WRL	116 KB
SFERE_sfera_nurbs[2].stl	06/11/2025 10:06	Oggetto 3D	47 KB
SFERE_2.3dm	06/11/2025 10:04	Rhino 3-D Model	7.759 KB



These values cannot be modified and guarantee an excellent approximation of the surface while keeping the weight of the individual file relatively low.

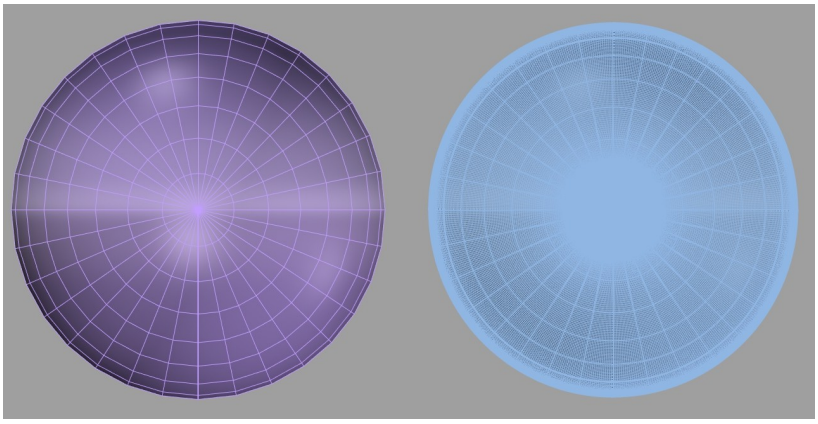
A special case may arise when exporting SUBD geometries if these are created directly by transforming MESH.

This is because the mesh created during export would be extremely dense and therefore extremely heavy.



For example: a simple sphere created as surfaces, as a mesh, transforming the mesh into SubD and transforming the surface into SudD will generate the following pairs of files:

Note the weight in KB of the sphere created from surfaces, which weighs 116 KB, identical to that of the mesh (the export process actually transforms the surface into a mesh) and that created by transforming the mesh into SubD, which weighs 31,754 KB.



If you zoom in to view the mesh created, you will see the reason for the increase in weight: the dense network of polygons created to approximate the surface (on the left is the mesh and on the right is the one generated by the Mesh>SubD geometry).

Presence of multiple geometries on the same layer.

As already explained, the plugin detects the presence of multiple objects on the same level and assigns a sequential number to avoid conflicts and therefore file overwriting.

In the example of the spheres, with three objects on the same level, we will have

SFERE_sfera_nurbs_2[2].stl and **SFERE_sfera_nurbs_2[2].wrl**

SFERE_sphere_nurbs_1[2].stl and **SFERE_sphere_nurbs_1[2].wrl**

SFERE_sphere_nurbs[2].stl and **SFERE_sphere_nurbs[2].wrl**

The highest number indicates the oldest object.

This process also occurs if the geometries are hidden and therefore not exported. In the previous example, if only the oldest sphere is visible, only **SFERE_sfera_nurbs_2[2].stl** and **SFERE_sfera_nurbs_2[2].wrl** will be exported.

Good rules to follow:

1: Create a layer tree consistent with the objects to be exported.

2: Assign the base material to the layer; any other materials or textures can be assigned directly to the object.

3: Place the objects on the correct levels so that they inherit the material. 4:

Keep only the objects to be exported visible.

Any test geometries, notes, cutting planes, infinite planes, and support geometries that are not to be exported must be hidden. As explained above regarding the sequential number assigned when there are multiple objects on the same layer, it would be advisable to

create a special layer where all the "support" geometry can be moved, which will be turned off and therefore ignored before export.

4: check the geometry to be exported: an exploded polysurface will generate as many pairs of files as there are surfaces... this could take time and mess up your working directory.

5: If this is essential, for example if some surfaces need to be assigned a different mapping and material from the rest of the object, the complete object must be grouped. This will be read by the plugin as a single object (see **Presence of grouped objects**)

ESC key: forced exit from the plugin

It may happen that you launch the plugin by mistake, without noticing the presence of unnecessary geometries, or that you accidentally explode a polysurface or mesh, etc. To avoid filling the working directory with thousands of files and waiting patiently for the process to finish, you can forcefully interrupt the execution of the plugin by holding down the Esc key.

Conclusions:

This plugin optimises the export of geometry from Rhino to STL and WRML formats, which are the standard formats for 3D printing and for the creation of moulds for the industrialisation of prototypes.

Saving files with a name consistent with the file name, which keeps track of the model version (if the file name is written according to a few simple rules) with the optimised generated mesh could greatly speed up your work by automating repetitive operations.

You may not need it if you don't need to create multiple versions of your work to present to your client, if you only have one object on one layer, or if you don't use Rhino professionally.

In all other cases, you will notice a significant saving in time and order in the files generated, time that you can devote to modelling and order that will allow you to share your files with less chance of errors.

By downloading the file, you will be able to test the plugin without any limitations for 30 days, after which you will need to purchase a one-year licence.

Imported files

The plugin was written in Python 2.# so that it is compatible with Rhino from version 7 onwards.

It works with Rhino, is not standalone, and can export all objects compatible with Rhino.

I normally model surfaces directly in Rhino, but it could also be useful to use it just for exporting.

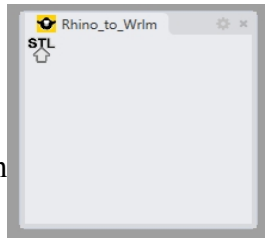
For example, files that require 3D texturing, such as animal fur, may be difficult to create directly in Rhino. You could create the base in Rhino and apply the texture using a third-party programme (such as 3DCoat), import the model back into Rhino, perhaps on a separate layer, and use the plugin for the final distribution.

Or you may be used to working directly in polygons with other software, such as Zbrush, but still need to transfer the files to Rhino to comply with the client's requirements. In this case, after importing the polygonal files that make up the parts of the model, you would use the plugin for the final export after creating the layer tree with the materials and names.

Installation

Download the folder and save it wherever you want on your hard drive, drag the **.yak** file into an open Rhino session... done!

The command bar with the button to launch the plugin should appear. Move it wherever you want and dock it to the toolbars so that find it the next time you restart Rhino.

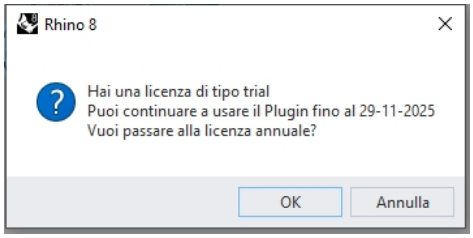


Alternatively, you can install it from Options/Plugins/Install. In th case, you will be asked for the RHP file.

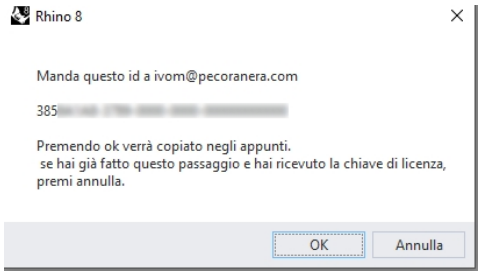
When you first start the plugin, you will be asked if you want to test the plugin or purchase the annual licence.

This message will appear when you start the plugin, but only once a day so as not to be too intrusive, with a countdown of the days remaining for the trial version.

If you wish to upgrade to the paid version, you must give your approval in the dialogue box.



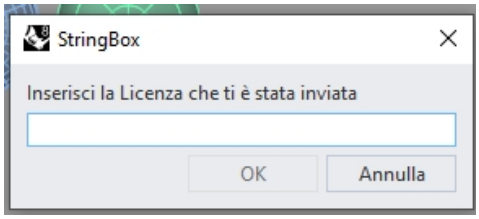
A new box will open:



By pressing OK, your PC/Mac ID will be copied to the clipboard.

Send an email to:ivom@pecoranera.com with proof of your purchase. Paste the ID from the clipboard and enter your name. By entering "annual licence request" in the subject field, you will help me to organise my incoming emails.

Once your purchase has been verified, I will send you back the personalised serial key that you will need to enter in the box accessible by pressing "Cancel" in the previous box.



From this moment on, you will be able to use the plugin for one year.

If you have arrived here by mistake, pressing cancel will allow you to continue using the trial licence for the remaining time.

Command line commands.

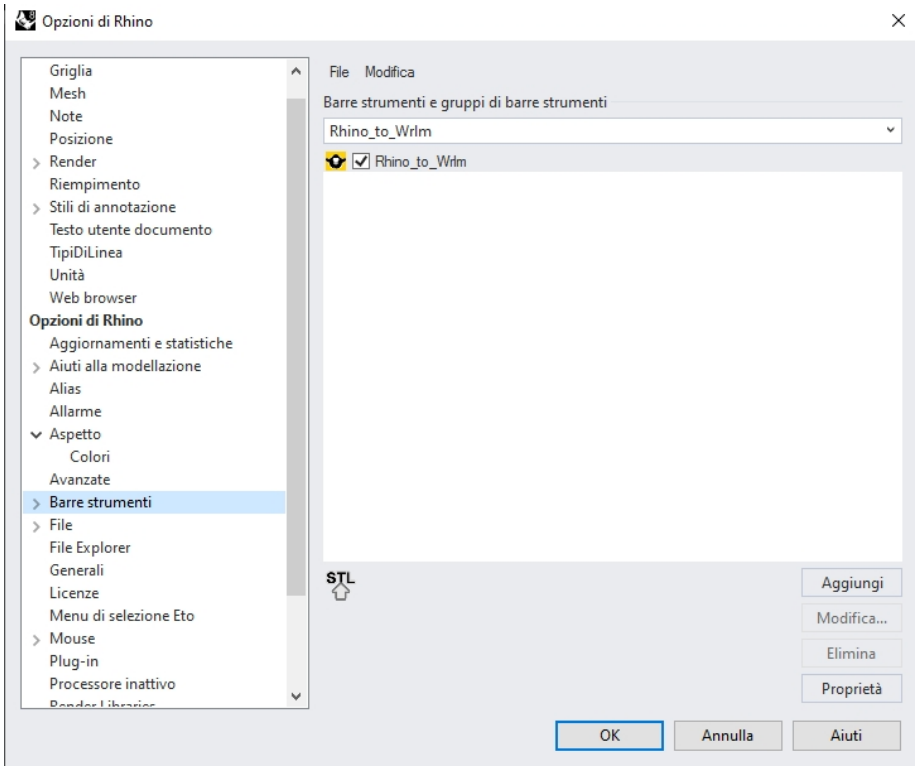
Like any operation that can be performed in Rhino, you can start the plugin directly from the command line. This is useful if the plugin's command bars have disappeared, which is not impossible if you use free bars rather than fixed ones.

Type "**_RhinoToWrmlStl**" or "**RhinoToWrmlStl**" (actually, just type the first few letters, as there is auto-complete) and press Enter to start the plugin.

This is also a reliable way to check whether the plugin is running or has been uninstalled.

If for some reason the button to start the plugin has disappeared and you are sure that it is installed, to make it reappear, open:

options/appearance/application bars/RhinoToWrml



Check the box and the button will reappear.

Uninstalling:

type "package manager" in the command line, select RhinoToWrml in the "installed packages" window and click on uninstall.