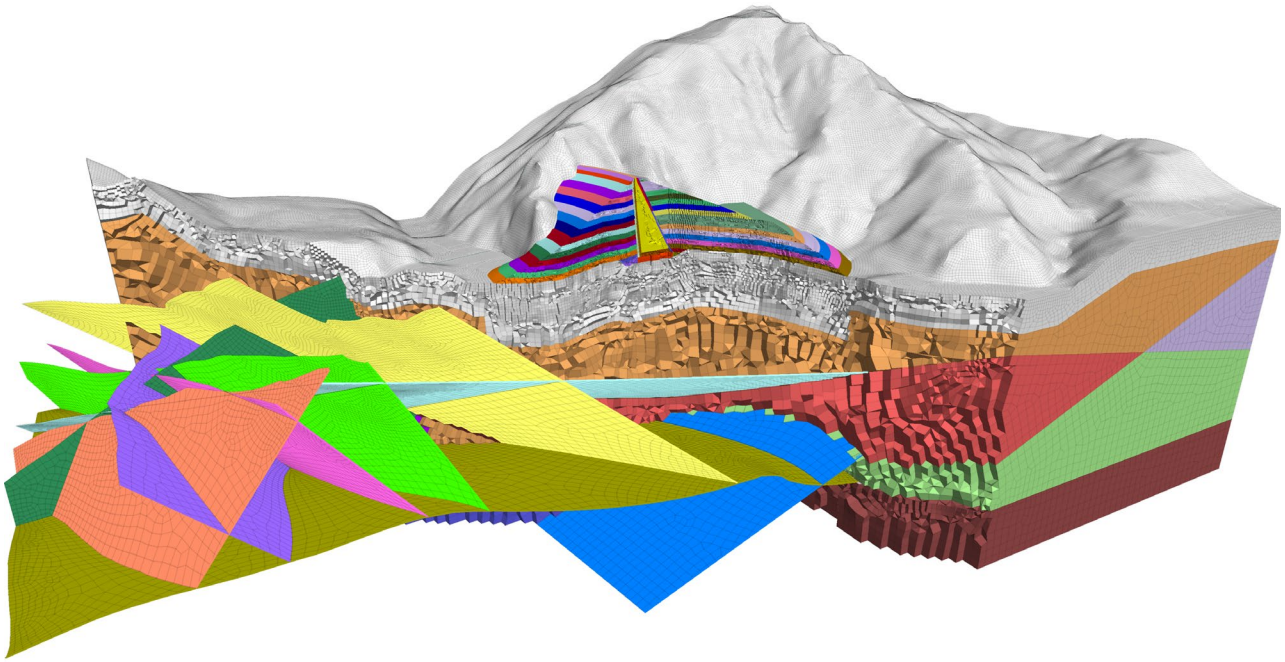
Griddle™

Advanced Mesh Generation for Engineers and Scientists

Griddle 3.0
User Manual



Itasca Consulting Group Inc.
111 Third Avenue South, Suite 450
Minneapolis, Minnesota 55401 USA

phone: (1) 612-371-4711
fax: (1) 612-371-4717
e-mail: griddle@oneitasca.com
web: www.itascasoftware.com

Contents

Introduction

This document describes the use of *Griddle* mesh generation tools in conjunction with the *Rhino* CAD system.

Griddle is a *Rhino* plugin that provides simple-to-use but powerful capabilities for advanced operations with surface meshes and tools for the generation of volume meshes (or grids). *Griddle* introduces new tools and extends the meshing capabilities within the *Rhino* CAD software, offering users both automatic and interactive approaches to mesh generation. Its integration with *Rhino* enhances meshing efficiency while leveraging *Rhino's* powerful CAD tools to create an all-in-one meshing application for both novice and experienced users alike. *Griddle* is particularly useful in geotechnical and geological engineering and in structural analysis applications; it is the recommended tool to use alongside other *Itasca Software*.

Griddle users can interact with other Itasca Software users and developers, ask questions and share experiences at Itasca Software Forum[1].

This manual contains images showing snapshots of *Rhino* windows and dialogs. The images were generated in *Rhino 8*, but analogous dialogs and windows can be found in other versions of *Rhino*.

What's new in *Griddle* 3.0

Griddle 3.0 contains several new features, improvements to existing tools, and bug fixes. A complete list of all changes and improvements can be seen here: Griddle 3.0 change log [2].

Tutorial Examples

Detailed tutorial examples of geomechanical applications are provided in *Griddle 3.0 Tutorial Examples* document available via *Windows Start Menu → Itasca Griddle 3.0*. The files for the tutorial examples can be accessed by clicking on *Griddle 3.0 Tutorial Files* link. The link opens user-writable directory `ProgramData\Itasca\Griddle300` (typically on drive "`C:`") which contains the documentation and the tutorial material. Users can directly work in this directory. A reserve copy of the same material can be found in the `Documentation` subfolder of *Griddle* installation location (typically in `C:\Program Files\Itasca\Griddle300`).

[1] Itasca Software Forum: https://forum.itascainternational.com/
[2] Griddle 3.0 change log: https://www.itascacg.com/software/downloads/griddle-3-0-update

Quick Start for *Griddle 3.0* and *Rhino*

Griddle 3.0 is a suite of tools for surface- and volume-meshing within the *Rhino* CAD system. ***Griddle 3.0* is installed as a *Rhino* plugin and is only available for the Windows versions of *Rhino 8 or newer*.**

The *Griddle 3.0* installer automatically removes all previous versions of *Griddle* installed on the machine, including those for *Rhino 5* and later.

Rhino is required to be installed before *Griddle* installation.

First Time Use After Installation

1. If *Griddle 1.0* was previously installed and you are migrating directly to *Griddle 3.0*, additional steps may be needed to remove parts of the older *Griddle* from *Rhino*[3]. Please refer to **First Time Use After Installation** section in the **Griddle 2.0 User Manual**.
2. If *Griddle* has never been installed on the user machine, or if you are migrating from *Griddle 2.0*, nothing else needs to be done after *Griddle 3.0* installation – it can be readily used. *Griddle 3.0* automatically integrates with *Rhino 8;* the *Griddle* toolbar should be visible floating within the *Rhino* workspace:

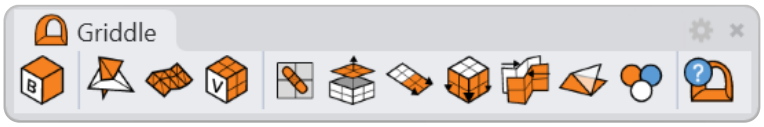


Figure 1: The *Griddle 3.0* toolbar.

If the toolbar is not visible after the installation, follow the steps below:

1. Navigate to the top menu in *Rhino* and click on **Tools** → **Options** → **Toolbars** (or run the **_Toolbar** command).
2. Click on the drop-down menu and select *Griddle*. Click the check box to activate the toolbar. This is shown below in Figure 2.

Note: the toolbar may be floating outside of the Rhino workspace.

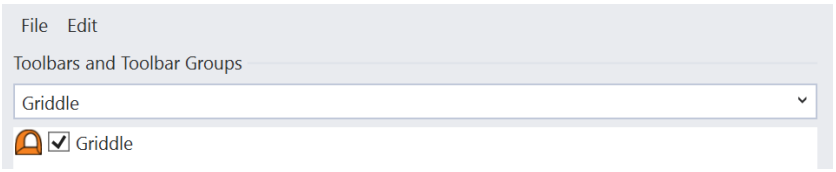


Figure 2: Activating the *Griddle 3.0* Toolbar in *Rhino 8*.

[3] *Griddle 1.0* used *Rhino* integration tools and required a multi-step installation process (separate installation of plugins for each command, toolbars, and scripts). *Griddle 3.0* operates differently – it installs all components at once. During the installation, *Griddle 3.0* attempts to remove all older versions from the system, but some parts may remain and should be removed manually.

Griddle 3.0 Components

This section provides a brief summary of *Griddle 3.0* components. Each component can be accessed by clicking on an icon on *Griddle* toolbar or by typing a command that is the same as the component name (if *Rhino* is set up to use language other than English, type an underscore before the command). A detailed description of each of these components is provided in later sections of this manual.

Table 1. *Griddle* components.

Icon	Component name, Command (English)	Description
	**BlockRanger**	Structured hexahedral volume mesher operating on solids
	**GInt**	Surface mesh intersector
	**GSurf**	Unstructured surface mesh remesher
	**GVol**	Unstructured tetrahedral/hex-dominant volume mesher
	**GHeal**	A set of tools for identifying and fixing surface mesh problems
	**GExtract**	A set of tools for the extraction of sub-meshes based on user-specified criteria
	**GExtend**	A set of tools for extending (enlarging) surface meshes
	**GExtrude**	A set of tools for surface mesh extrusion along the boundaries to create watertight domains
	**GCollapse**	A tool for collapsing (resolving and cleaning) nearly overlapping surface meshes
	**NonManifoldMerge**	*Rhino* command to merge (poly-)surfaces into a single non-manifold polysurface
	**ColorizeObjects**	A tool for assigning random colors to objects
	**GriddleAbout**	Display information about *Griddle* and check for updates

Griddle 3.0 Licensing

Griddle 3.0 requires a license to run all components with full functionality. The licenses can be provided via a local USB key (a desktop license), a network USB key (a network license) or through online Web licensing (using Itasca Web License portal).

A license can be obtained on the *Itasca Software* website: https://itascasoftware.com/products/griddle/

If a license is not present, *Griddle 3.0* operates in demonstration mode which limits *Griddle* functionality as described in Table 2.

Table 2. *Griddle 3.0* demonstration mode limits.

Icon	Component	Limitations in demonstration mode
	**BlockRanger**	Saves output volume mesh in VRML format only
	**GInt**	Only the preview of intersections is available (meshes are not intersected or changed)
	**GSurf**	• Functionality to keep meshes separate (*Output=Distinct*) is not available (all meshes will be merged in the output) • The number of elements in the output mesh is limited to 5000 • Element/mesh quality information is not available
	**GVol**	• The number of elements in the output mesh is limited to 10000 • Element and face groups information is outputted only if the original number of elements in the volume mesh is below 10000 • Element/mesh quality information is not available
	**GHeal**	Automatic mesh repair, *AutomaticHeal*, is not available
	**GExtract**	Only separation of a single surface (*Single*) is available
	**GExtend**	Not available in the demo mode
	**GExtrude**	Not available in the demo mode
	**GCollapse**	Not available in the demo mode
	**NonManifoldMerge**	Available in the demo mode
	**ColorizeObjects**	Available in the demo mode
	**GriddleAbout**	Available in the demo mode

Using *Griddle* network license

If a network USB key is purchased, follow these steps to install the required *Run-time Environment* on the license server:

1. Do not connect the network USB key to the server machine. On the machine install *Sentinel Protection Installer 7.7.1*[4] (or any newer version). Make sure that port 1947 is open for the UDP and TCP connections if using Sentinel HL (green) key and that ports 6001 and 6002 are open if using Sentinel SuperPro key.
2. Connect the *Griddle* network USB key to the server.
3. Start *Rhino* with *Griddle* on the user machine (make sure it has stable connection to the license server) and follow the steps described in the next section.

Changing *Griddle* license location

To change the type of *Griddle* license to be used or to change license location, click on the *GriddelAbout*  icon or type the **_GriddleAbout** command and then click on *Show or Change Griddle License* button. A **License Location** dialog will open (it may take a moment to open as it tests connections to the existing licenses). Users can select the desired license type in the dialog, test connection to the licenses and list license information.

If using *Griddle* network license, enter the IP address of the license server or type `localhost` to use local machine as the server.

Updating *Griddle 3.0*

Griddle 3.0 is continuously being improved. Users may expect periodic updates. *Griddle* automatically checks for available updates each time *Rhino* is closed. If a *Griddle* update is available, a message box will appear offering users to navigate to the download page.

Users may also check for updates manually by clicking on the  icon on *Griddle* toolbar or by typing the **_GriddleAbout** command. Besides displaying technical information about *Griddle* and license information, the **Griddle About** dialog notifies users when updates are available.

Users may also check for updates by navigating directly to the *Griddle* update page: Griddle 3.0 Version History[5].

The installer for *Griddle* updates removes previous versions prior to installing the update.

[4] *Sentinel Protection Installer 7.7.1* (Article KB0023089)**:** https://supportportal.thalesgroup.com/csm?id=kb_article_view&sys_kb_id=d902c13c1b48a890f2888739cd4bcbbf&sysparm_article=KB0023089

[5] Griddle 3.0 Version History: https://www.itascacg.com/software/downloads/griddle-3-0-update

Brief Introduction to the *Rhino 8* Workspace

The *Rhino* Command Area, Command Prompt, and Commands

All *Rhino* operations are performed by commands. The command area and command prompt are shown in Figure 3.

The command area is the field (generally on top) where *Rhino* displays system and command output information. Below the command area is the command prompt, where users can type *Rhino* or *Griddle* commands. The location of the command area or any other toolbar, panels, etc. can be moved by dragging them to a new location. These can either be docked to a *Rhino* window location or remain free floating such as the *Griddle* toolbar shown in Figure 3.

When a user clicks on any icon (including icons from the *Griddle* toolbar) or a menu item, a corresponding command is automatically issued and placed into the command prompt. For example, clicking on  icon from the *Griddle* toolbar issues the command **_GSurf**.

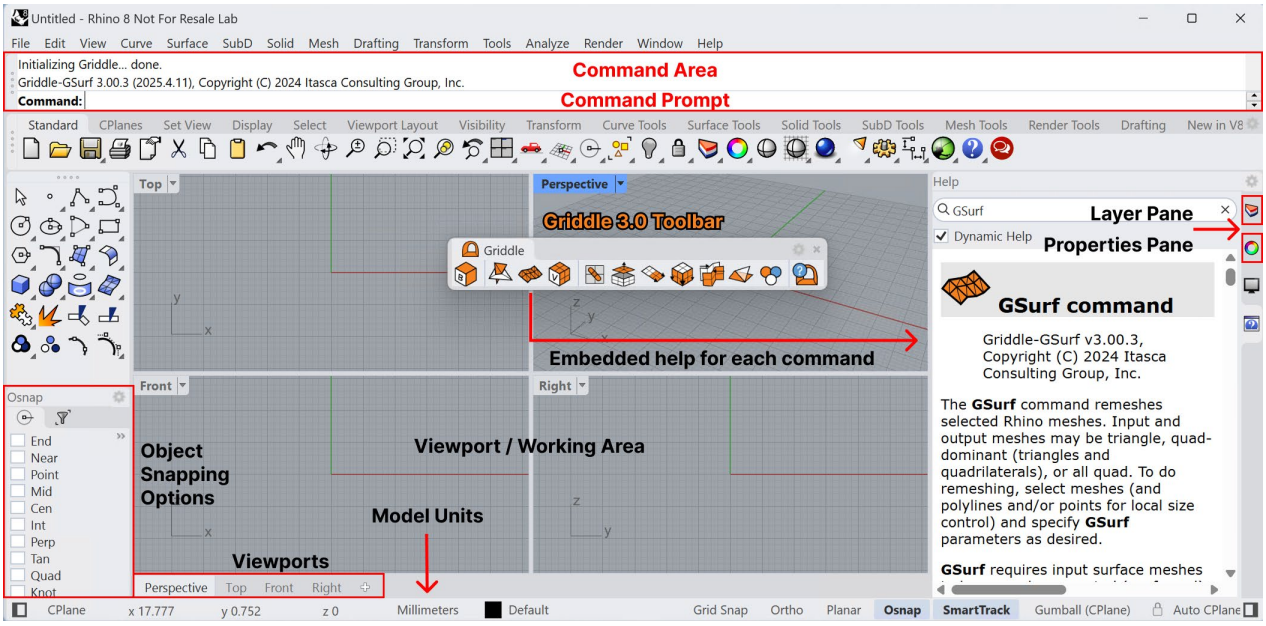


Figure 3: The *Rhino* window featuring the *Griddle* toolbar and the *Help* panel.

Another example is the command **_CommandHelp**, which opens the embedded *Help* panel. If the *Auto-Update* checkbox is selected in the *Help* panel, it will display information about the currently typed command, including *Griddle* commands, as shown in Figure 3. An alternative way to access help information is to press **F1** after entering a command (or type **_Help**) which will open a default browser with corresponding help topics.

Note that if an underscore is placed before the command name (e.g., **_Shade**), *Rhino* assumes that the user refers to the English command name (i.e., **Shade**) regardless of the locale (i.e., language and other local settings) of the operating system and *Rhino* interface.

More information on *Rhino* interface can be found in the References section (see: *The Rhino window and interface*).

Basic Controls

Basic controls for navigating the *Rhino* viewports are described in Table 3.

Table 3: Basic *Rhino* Controls

Control	Action
Select / deselect objects	Left (mouse) click
Marque select / deselect	Left click + Drag
Zoom in/out	Scroll the mouse wheel or Ctrl + Right click + Drag
Pan in the Perspective viewport	Shift + Right click + Drag
Pan in other (default) viewports	Right click + Drag
Rotate the view in the Perspective viewport	Right click + Drag
Enter / Run Previous Command	Right click

Viewports and Display Modes

Viewports are part of the *Rhino* workspace where the user manipulates models. There are four default viewports: *Perspective*, *Top*, *Front*, *and Right*, as shown in Figure 3. In the *Perspective* viewport, which represents a model in 3D perspective view, the user can rotate and pan the view (see Table 3 for controls). The other default viewports provide 2D views. The user may display several viewports simultaneously and may even create custom viewports.

Display Modes manage how the viewports are displayed to the user. This can be set by right clicking on the viewport tab or by issuing **_Shade** command (with the command, mode changes are temporary in the active viewport). Different display modes are useful in different situations. For example, the Ghosted mode will make all the objects appear translucent and would make it easier to visualize internal objects. *Rhino* defaults to the Wireframe display mode.

Layers and Properties

Objects within *Rhino* can be organized into Layers using the *Layers* panel (if needed, activate the panel using **_Layer** command). Layers can be given specific properties (name, color, parent/sub layers, etc.) to help navigate complex models. The visibility of each layer can also be toggled to show or hide the corresponding layer from the active viewport by toggling the icon next to the layer name (if the icon not visible, enable it via selection of *Columns* in the *Layers* panel).

Each *Rhino* object also has a specific set of properties associated with it, and can be viewed via the *Properties* panel. This panel can be accessed by either clicking the *Rhino* icon , shown in Figure 3, or by running the **_Properties** command. This panel will automatically refresh and show all the properties associated with the object(s) that are currently selected.

Object Snapping Options

Two types of snapping are available in *Rhino*: **Grid Snap** and **Osnap**. When **Grid Snap** is enabled, moving the mouse while it controls an object (e.g., dragging or extending an object) will cause the mouse pointer to snap to discrete positions in space corresponding to the X, Y, Z position of grid nodes.

To be able to snap to various objects and their parts, such as curves, meshes, vertices, ends, etc., **Osnap** control must be active. Click on the **Osnap** toolbutton at the bottom of the *Rhino* window to activate it or use **_Osnap** command. Next, specify which item(s) will be active for snapping (either via the command or by using Osnap panel, as shown in Figure 3).

Both snapping options can be active simultaneously.

Units and Tolerances

Rhino allows users to specify units when creating a new project. Units determine what is displayed when measuring distance, location, or angles. More importantly, units and object sizes are related to the tolerances used by *Rhino*. By default, tolerance is determined based on model units. However, users can specify custom values (see Figure 4).

Tolerances are very important when intersecting or doing Boolean and other operations with NURBS, SubD surfaces, BRep[2] objects, or when executing various *Rhino* commands on meshes. *Griddle's* **GInt** tool has its own user specified tolerance. Other tools such as **GExtend** and **GCollapse** also have a tolerance specification (depending on other function parameters). Specifying proper tolerance assures obtaining correct results when numerical or discrete operations are involved.

In all cases, make sure that the model is never too far away from the origin and, if it is, **_Move** it closer to the origin. Moving objects closer to the origin improves accuracy of operations and helps with the displaying of extra small or large objects.

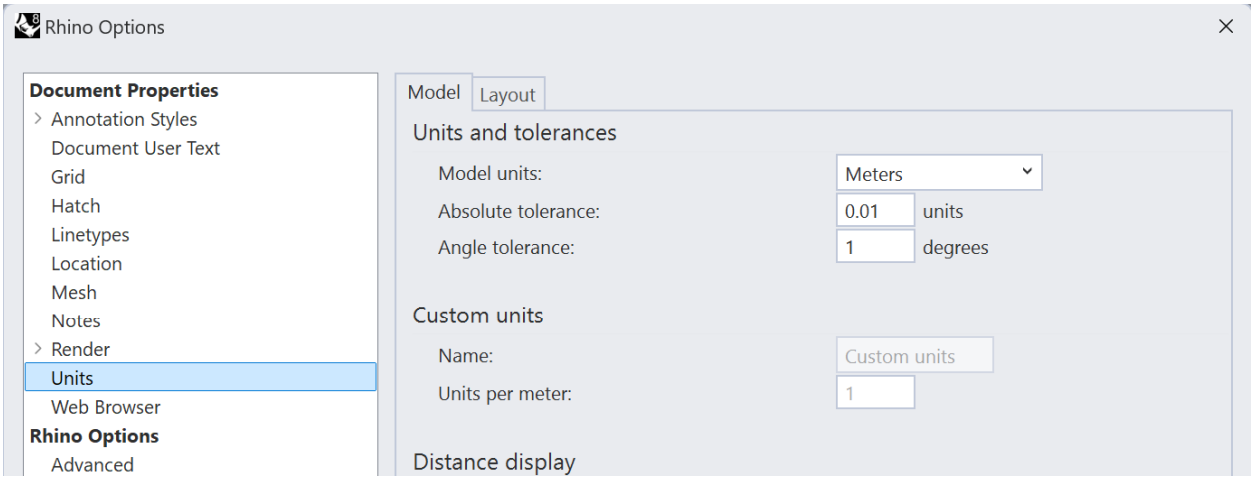


Figure 4: The *Rhino* window to specify model units and tolerances.

When starting a new project, use the appropriate *Rhino* template to specify whether the model will be using "Large Objects in Meters", "Small objects in feet", etc. The current model units can be seen at the bottom of *Rhino* window, as shown in Figure 3. After that create a model or import supported geometries/files into the project. This technique allows controlling the tolerance (instead of using an inherited tolerance when starting from an imported file).

More information on tolerances can be found in the References section (see: Understanding *Rhino* tolerances).

Orthogonal Restriction of Mouse Movement

When creating or modifying objects, mouse movement can be restricted to the X, Y and Z directions by clicking the word **Ortho** that appears at the bottom of the graphic window — or by pressing **F8**. Mouse movement may be temporarily restricted by holding down **Shift** while moving the mouse.

Thicker lines

The default representation of curves may be too thin and difficult to see in the working area. Figure 5 shows, at left, the default representation. The following procedure describes how to display thicker curves if, for example, **Shaded** display mode is used (Figure 5, right).

Within *Rhino*, Select the **Tools** → **Options** menus item. In the left pane under the **Rhino Options** title, select **View** → **Display Modes** → **Shaded** → **Objects** → **Curves**. In the *Curve Settings* pane, to the right, increase Scale (or Size) value and click **OK** (Figure 6). These changes are saved with the project.

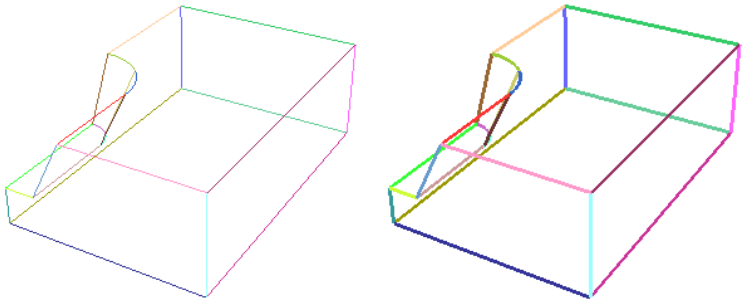


Figure 5: A regular (left) and “thick line” (right) representation.

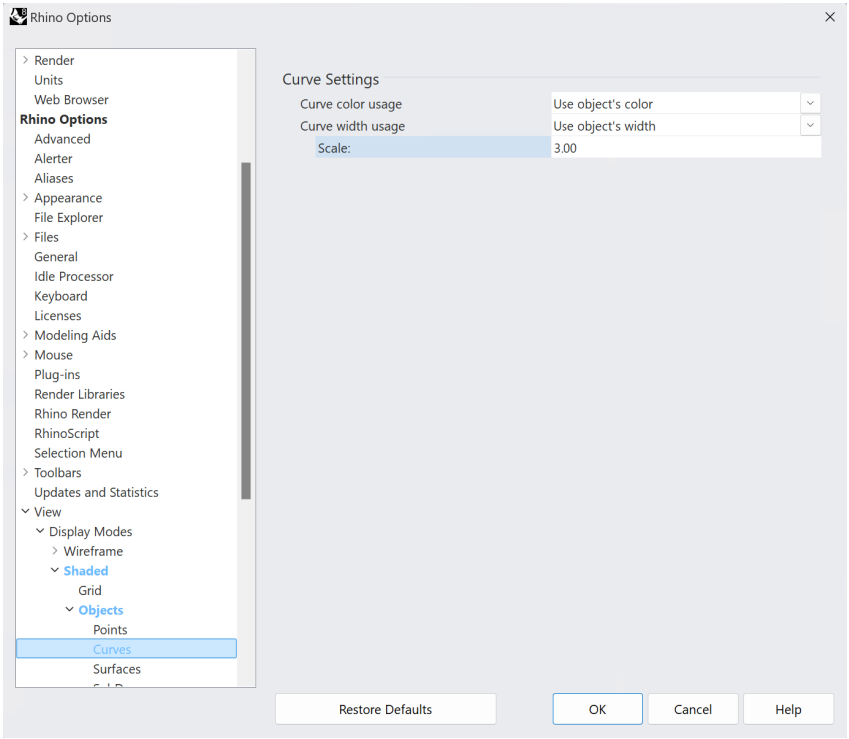


Figure 6: Setting line width to 3 pixels in Shaded View.

Construction Plane, Axes, and Background Grid

The construction plane in *Rhino* represents the local coordinate system for the viewport which can be different from the global (world) coordinate system used by *Rhino*. *Rhino’s* standard viewports come with construction planes that correspond to the viewport (and the default *Perspective* viewport uses the world *Top* construction plane). A construction plane is like a tabletop that the cursor normally moves on. The construction plane has an origin, x and y axes (represented by the dark red and green lines, respectively), and a grid. The construction plane can be set to any location and orientation in each viewport. More information about the construction plane can be found in *Rhino* documentation or by navigating to the help topics for the **_CPlane** command.

To hide or show the background grid and axes in an active viewport, press **F7**. To hide the grid in all viewports, go to **Tools** → **Options**. In the left pane of the **Rhino Options** dialog, click on **Document Properties** → **Grid**, and in the Grid properties section, uncheck *Show grid lines* and *Show grid axes*.

Avoiding Accidental Object Dragging

It is easy to inadvertently drag a selected object with the left mouse button. To avoid this, open menu item **Tools** → **Options**. In the left pane of the **Rhino Options** dialog, click on **Rhino Options** → **Mouse**, and in the *Click and drag* section, set the *Object drag threshold* to 100 pixels. Now, dragging an object requires a minimum of 100 pixels of mouse movement before it takes effect. An alternative way to reduce altering unwanted objects is by utilizing the *Rhino* Layer **Lock** tool. If a layer is locked, the user will be unable to select or alter any objects associated with it. This can be toggled with the *Rhino* icon  in the Layers Panel.

A number of other customizations and helpful tips/information about *Rhino* can be found in the *Rhino* documentation (press **F1** or navigate to **Help** menu).

Structured and Unstructured Meshes

Griddle creates structured volume meshes using **BlockRanger** and unstructured volume meshes using **GVol**. **BlockRanger** operates on solids[6] represented by BRep or NURBS objects (and not by SubD objects). **GVol** operates on sets of conformal surface meshes (structured or unstructured) which must compose watertight (closed volume) domains. After executing **BlockRanger** or **GVol**, output file(s) with volume mesh data are created. The output contains information about nodes, faces, elements, node-face connectivity, node-element connectivity, and element and surface groups[7]. A mesh from such output can be imported into numerical modeling software (e.g. *FLAC3D*, *3DEC*, *ABAQUS*, etc.)

Structured meshes are identified by regular connectivity and typically have well-shaped elements. Simple examples of structured meshes are (see examples in Figure 7, Figure 9, and Figure 11):

- a quadrilateral mesh in 2D where each internal node is joined to 4 neighboring quadrilaterals, forming a regular array of elements, and
- a structured 3D hexahedral mesh that has each internal node connected to 8 elements.

An unstructured mesh is identified by irregular connectivity. Surface unstructured meshes typically employ triangles and quadrilaterals, while an unstructured 3D volume mesh may contain tetrahedra, pyramids, prisms, hexahedra, and even more complex elements. *Griddle*'s components **GSurf** and **GVol** build triangular, quad-dominant, and quadrilateral surface meshes and tetrahedral and hex-dominant unstructured volume meshes, respectively.

Unstructured meshers typically provide an advantage of generating meshes for geometries of any complexity. Moreover, generation of an unstructured mesh is often much faster compared to the time required to build a similar model with a mapped structured mesh (for cases when it is possible to use a structured mesher).

Two examples below (Figure 7, Figure 8) illustrate the difference between structured and unstructured volume meshes created for the same initial geometry. The model includes a layered domain with several parallel curving tunnels that are intersected by another tunnel.

Figure 7 shows a fully structured volume mesh which was created by decomposing the initial geometry into hexahedral, tetrahedral, and prism-like solids within *Rhino*. After such decomposition, the solids were meshed with **BlockRanger**. Preparing solids from the initial geometry in *Rhino* takes a considerable amount of time and effort. However, the resulting mesh is a high-quality, all-hexahedral structured mesh.

Figure 8 illustrates a fully unstructured volume mesh that was generated using **GVol**. The volume mesh was created from surface meshes of tunnels and rock layers. No decomposition into simple primitive shapes/solids is needed in this case. Thus, overall model creation is much simpler and faster. The resulting mesh is a reasonable-quality hexahedral-dominant unstructured mesh.

[6] Solid is an assembly of surfaces and/or polysurfaces in *Rhino* and it has a clearly defined interior and exterior.
[7] Surface groups are not available for NASTRAN and VTK outputs. VTK also excludes element groups.

For certain models, like for the one presented in Figure 7 and Figure 8, it is also possible to create a hybrid (mixed) mesh, in which the interior of the tunnels is filled with a structured mesh (using **BlockRanger**) and the exterior volume is filled with an unstructured mesh (using **GVol**). This is possible because the tunnels have regular shape, while the overall shape of the model including tunnel surfaces is irregular. Meshes from **BlockRanger** and **GVol** need to be created separately and output into separate files. Subsequently, they can be combined into a single mesh within numerical modeling software resulting in a hybrid model mesh.

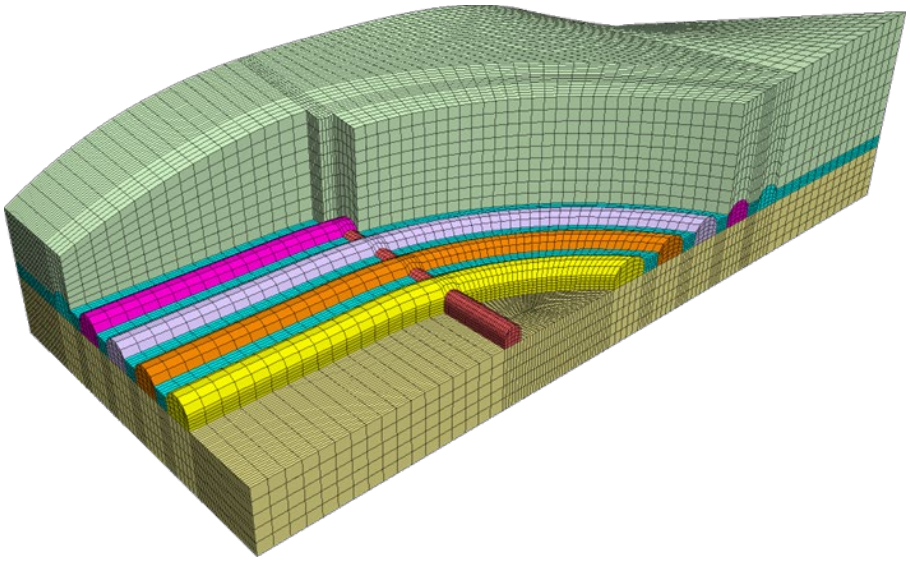


Figure 7: All hexahedral structured *FLAC3D* mesh generated by BlockRanger.

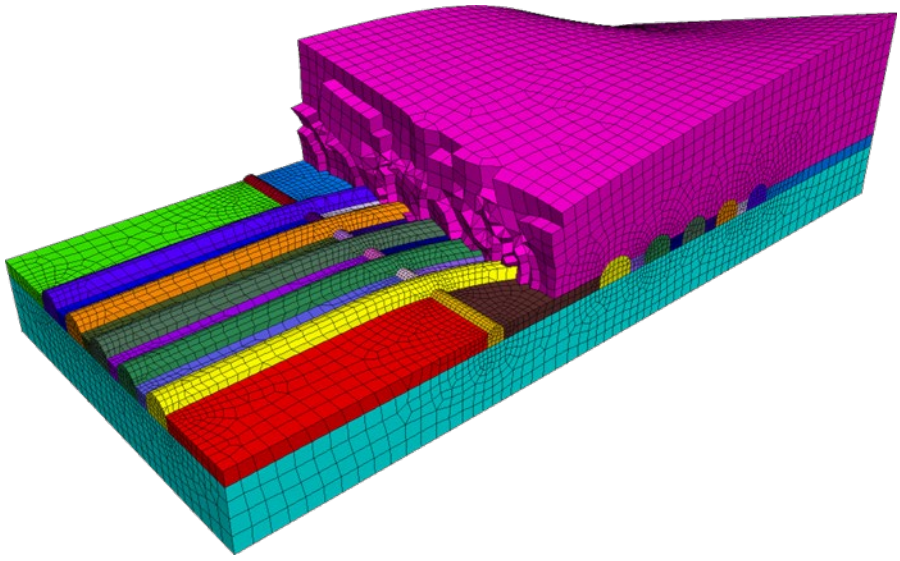


Figure 8: Hexahedral dominant unstructured *FLAC3D* mesh generated by GVol.

Using *Griddle* Tools for Structured Meshing – BlockRanger

BlockRanger is an interactive all-hex mapped mesher. The command to run **BlockRanger** is **_BR** and the icon to run it is provided in Table 1.

BlockRanger operates only on 4, 5 or 6-sided solids represented by watertight BRep or NURBS objects (not SubD). It creates a high-quality hexahedral (brick) volume mesh within such solids, which can be output in different formats for use in numerical modeling software.

Admissible **BlockRanger** solids (Figure 9) are:

- 6-sided solids (hexahedron-like) composed of 6 surfaces each bounded by 4 curves.
- 5-sided solids (prism-like) composed of 2 triangle-like surfaces bounded by 3 curves and 3 quad-like surfaces bounded by 4 curves.
- 4-sided solids (tetrahedron-like) composed of 4 triangle-like surfaces each bounded by 3 curves.

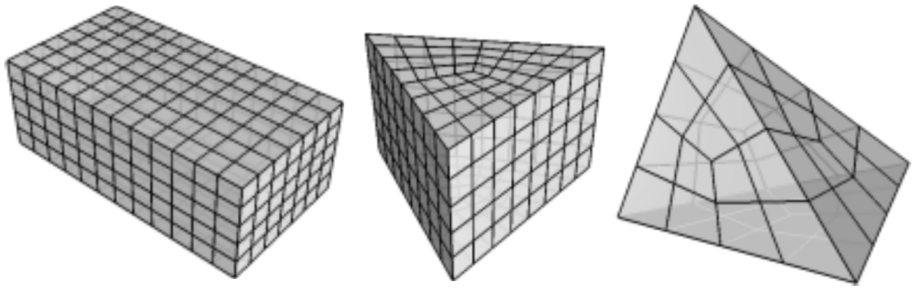


Figure 9: An example of solids supported by BlockRanger and BlockRanger generated mesh patterns for each (only exterior mesh boundaries are shown).

If the selected solids are contiguous, **BlockRanger** ensures that the resulting volume mesh maintains continuity and conformity across common block corners, edges, and faces so that no dangling nodes are created.

All solids successfully processed by **BlockRanger** are saved as grid files in a user-specified location or, if *AutoOutputName* option is specified – in the same location as the *Rhino* project. Solids that did not qualify or could not be successfully meshed remain selected.

BlockRanger Options

BlockRanger options consist of two categories: Mesh settings and Output parameters. They are outlined below. If options have predefined or default values, they are provided in square brackets.

Mesh Settings

MaxEdgeLength [*value* ≥ 0, default = *0*]

Maximum element (zone) edge length in model coordinates. If the default value of 0 is specified, this number is calculated as one tenth of the length of the longest edge in the model.

MinEdgeResolution [*value* > 0, default = 3]

Minimum number of elements across each *Rhino* solid edge. Its default value is 3.

TargetNumElements [*value* > 0, default = 3]

Use this option to reduce the maximum element aspect ratio in the grid. **BlockRanger** will initially build a grid based on the prescribed *MaxEdgeLength* and *MinEdgeResolution*, and edge subdivisions specified as an edge property. If the resulting number of elements (zones) is less than *TargetNumElements*, **BlockRanger** will improve the maximum aspect ratio of the mesh by refining it until the total number of elements exceeds *TargetNumElements*.

GenerateSurfaceMesh [= *None* (default), *ByModel*, *ByLayer*, *BySolid*]

This option specifies if surface mesh should be created on the boundaries of volume meshes. The surface mesh may be composed of external and internal boundary faces of the generated volume meshes:

- If *ByModel* is selected, the surface mesh will correspond to the external surfaces of all meshed solids combined (whole model).
- If *ByLayer* is selected, the surface mesh will correspond to the external surfaces of all meshed solids combined and the boundaries between the solids in separate layers.
- If *BySolid* is selected, the surface mesh will correspond to the external surfaces of each solid.

Note that a single surface mesh is generated containing all boundary faces and it is placed in the default layer. The surface mesh is not saved into the output file; it is generated for use within *Rhino* model only (for example, to quickly assess generated volume mesh by examining its boundaries).

OutputFormat [= *FLAC3D*, *3DEC_5x*, *3DEC*, *ABAQUS*, *ANSYS*, *NASTRAN*, *LSDYNA*, *VRML*, *VTK*]

Format in which the resulting volume grid file should be saved.

3DEC_5x format should be selected when using *3DEC* v5.0 or v5.2, while *3DEC* format should be selected for *3DEC* v7.0 or newer. When outputting to *3DEC* format, a rigid block is generated for every element. This output is equivalent to *BlockType* = *Rigid* in **GVol**. **BlockRanger** currently does not have the capability to output to *3DEC* deformable blocks format.

If the license key is not present, **BlockRanger** only allows outputting generated meshes in VRML format (see Demo limits in Table 2).

FormatType [= *Binary*, *Text*]

This option appears only if user selects *OutputFormat* = *FLAC3D* or *3DEC*. The option specifies how to save data in the output file: using binary encoding or text (ASCII). Saving and loading (reading) files in binary format is faster compared to text format but binary files are not human-readable. All other *OutputFormat* choices save in text format only.

Slots = [*Zone/BlockSlot="Griddle", Face/JointSlot="Griddle"*]

This option only appears if the user selects *OutputFormat* = *FLAC3D* or *3DEC*. It allows specifying custom slots for zones and face groups (or block and joint groups for 3DEC). As of *Griddle* 3.0, the

default names for both slots are "*Griddle*". Note that only alphanumeric ACSII symbols (0-9, a-z, A-Z) can be used in slot names.

AutoOutputName [= *N/A* (default), *UserDefinedName*]

This option specifies a string that will appear in the output filename along with other identifiers. For example, for a *Rhino* project named "RhinoProject.3dm", a *FLAC3D* output in binary format will be named: "RhinoProject_UserDefinedName_Binary.f3grid". If user specifies a string in this option, the **Save As** pop-up dialog will not appear and the output file will be named automatically as described above. If *AutoOutputName* is not specified (= "*N/A"*), a **Save As** dialog will appear, asking the user to provide the output file name and location. Note that:

- The value for this option is not saved: each time a user clicks on this option, a new string must be typed. To erase an existing string in *AutoOutputName*, click on the option and press **Enter**; this will cause a **Save As** dialog to appear when executing **BlockRanger**.
- When using this option, the output file will be saved in the same location as the *Rhino* project file. If a file with the same name already exists in that location, it will be overwritten.

This option allows running **BlockRanger** in automatic mode without user interaction (i.e., the need to type file name and press **OK** / **Cancel** in the **Save As** dialog). This is particularly useful when invoking the **_BR** command from *RhinoScript*, *PythonScript*, or other *Rhino* tools and plugins.

Local Edge Resolution Control

BlockRanger provides the ability to locally control volume mesh density by manually setting individual edge resolution. A short example explaining this is provided below. Hands-on information can also be found in the *Griddle 3.0 Tutorial Examples*, *Tutorial 4: Creating a Structured Mesh with* **BlockRanger**.

Figure 10 shows a model representing a 3D curved slope which consists of 8 solids. Figure 11 shows the resulting volume mesh after using **BlockRanger** with the default parameters.

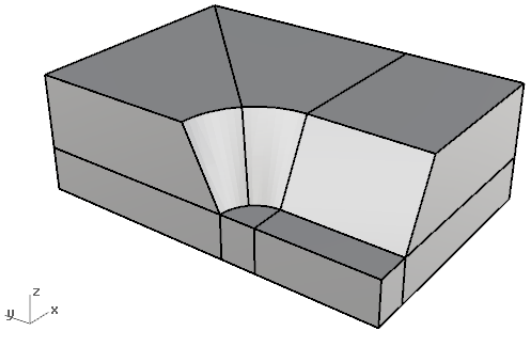


Figure 10: 3D slope example.

To be able to set the number of elements along an edge (local resolution), first use the command **_DupEdge** to duplicate an edge from a *Rhino* solid (Figure 12 shows two duplicated and highlighted edges). Select duplicated edge(s) and navigate to the *Properties* panel (or press **F3**). For this example, if numbers 8 and 5 are entered in the *Name* field for the selected edges, this will set the desired number

of elements along the edges. Now, if **BlockRanger** is run again, this time making sure to include the edges in the selection alongside the solids, the resulting grid will be more refined at the location of the duplicated edges (Figure 13). Set the option *GenerateSurfaceMesh=Yes* to visualize the resulting surface mesh in *Rhino*.

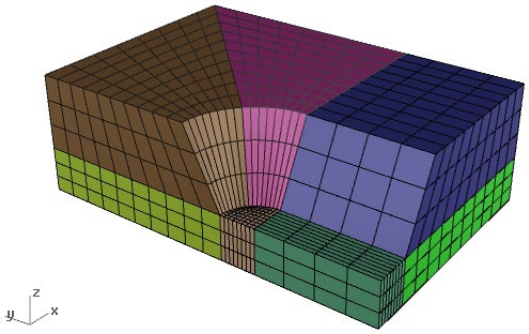


Figure 11: BlockRanger generated mesh with the default parameters.

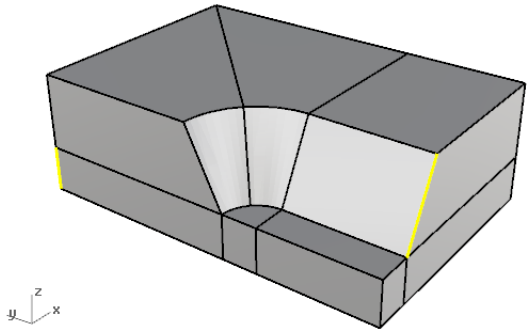


Figure 12: Duplicated edges (highlighted) with a custom number of elements specified.

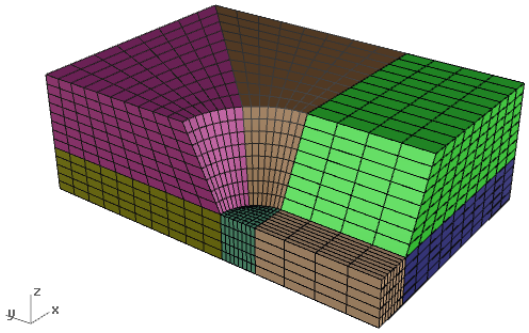


Figure 13: BlockRanger generated mesh with specified edge resolution.

Note that local edge resolution control takes precedence over automatically determined number of subdivisions per edge (based on other input parameters) only when it provides a larger number of elements for the edge (i.e., a finer mesh). Otherwise, the automatically determined number of elements is used.

Group Names Assignment

Many numerical simulation codes can associate names or groups with different parts of a volume mesh and also associate names or groups with internal and external boundaries and element faces. Such

named entities make it easier to reference parts of the model for material properties or boundary conditions assignment, creation of interfaces or contacts, or other operations.

BlockRanger requires solids as input, and with this, it can use an object name (assigned in the *Name* field in the object *Properties* panel) and layer information for groups creation in the generated volume mesh.

Group names in FLAC3D

To illustrate how solid's name and layer information are used for zone and face group assignments in *FLAC3D* output, let's assume that:

- A solid has a name set as "`SolidName`"
- A solid belongs to a layer "`SolidLayer`"
- **BlockRanger** *Slots* setting is set to *Slots*="*`ZoneSlot`*","*`FaceSlot`*"

The following zone groups and slots will be created for *FLAC3D* zones:

- Zone group "ZG_`SolidName`" in Slot "`ZoneSlot`"
- Zone group "ZG_`SolidLayer`" in Slot "`ZoneSlot`_Layers"

If a solid does not have a name assigned to it, a 5-digit numeric value will be automatically generated in the sequential order for every unnamed solid in the model producing groups:

- Zone group "ZG_#####" in Slot "`ZoneSlot`"

Here, zone groups are prefixed with "ZG_" symbols, and suffix "_Layers" is added to the slots associated with the groups which use layers information.

FLAC3D face groups are generated for:

- The exterior boundaries of solids for each distinct layer containing the solids. For example: "EF_`SolidLayer1`", "EF_`SolidLayer2`", etc. in Slot "`FaceSlot`"
- The interior boundaries between the solids belonging to distinct layers. For example: "IF_`SolidLayer1_SolidLayer2`", "IF_`SolidLayer2_SolidLayer3`", etc. in Slot "`FaceSlot`"

Similar to zone groups, prefixes indicating internal ("IF_" prefix) and external ("EF_" prefix) faces are added to the face groups. Note that face groups are not generated for the internal boundaries between the solids belonging to the same layer.

Group names in 3DEC

For *3DEC 5.2*, **BlockRanger** uses the full layer name as the *3DEC* block group name and no slot is assigned. For example, the *Rhino* layer "SolidLayer" would result in a *3DEC* block group called "SolidLayer".

For the regular *3DEC* output, block groups use a 3-digit numeric value assigned in the sequential order starting from “003” for every solid in the model (values “001” and “002” are reserved in *3DEC*). Solids with the same name, if assigned, result in the same block group number. Note that layer names are not used for block grouping. All block groups are placed in the slot specified in **BlockRanger** *Slots* setting (for example, a block group “003” in slot “BSlot”).

Internal faces which separate different block groups are assigned to a joint set group consistent with blocks group number (for example, a joint group “003-004” in slot “JSlot”).

Note: **BlockRanger** replaces all occurrences of whitespace characters (blanks, tabs, etc.) in the layer and name field with an underscore character “_” to avoid string interpretation problems in *FLAC3D* and *3DEC*. Names are case insensitive, e.g. “Solid Layer” is treated as the same name as “solid layer”.

Group Names in Other Formats

BlockRanger assigns names to element groups according to *Rhino* layer names. Face or surfaces groups are not created.

Using *Griddle* Tools for Unstructured Meshing

Besides the structured volume mesher, **BlockRanger**, which operates on *Rhino* solids, *Griddle* includes tools to work with surface meshes and tools to generate fully conformal unstructured volume meshes. These tools are grouped within the *Griddle* toolbar (see Figure 1, Figure 14, and Table 1).

- **GInt** – surface mesh intersector (command: **_GInt**),
- **GSurf** – unstructured surface remesher (command **_GSurf**),
- **GVol** – unstructured tetrahedral/hex-dominant volume mesher (command: **_GVol**).

Figure 14: *Griddle* tools for unstructured meshing.

All three tools operate directly on *Rhino* surface meshes. The basic meshing workflow to generate an unstructured volume mesh is summarized in Figure 15.

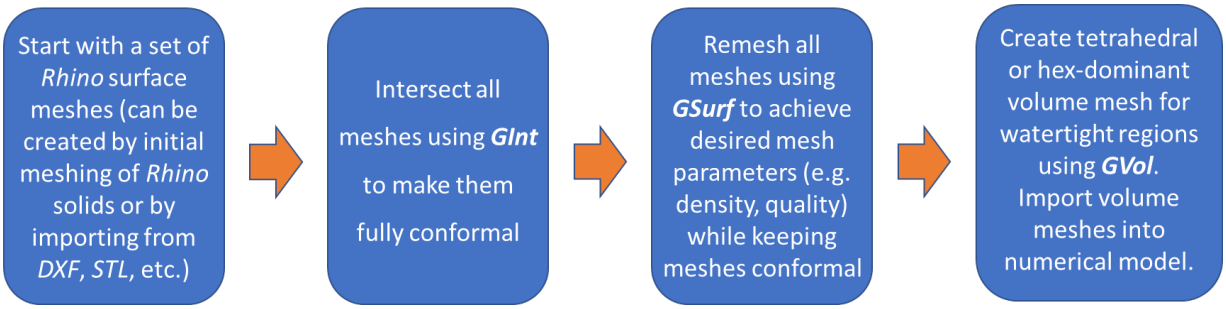


Figure 15: Typical workflow for unstructured meshing using *Griddle* tools.

Rhino has a rich set of surface meshing tools that allow users to create (e.g., by triangulation) and edit 3D surface meshes for objects represented by NURBS, BRep, or SubD surfaces and polysurfaces. These meshes, although good for machining and prototyping purposes, are usually not suitable for numerical computations. *Rhino*-generated (or imported) surface meshes often must be properly intersected and remeshed with **GInt** and **GSurf**, respectively. Once a desirable watertight set of surface meshes is obtained, it can be used (along with internal surface meshes) as an input to **GVol**, which fills the interior regions bounded by the surface meshes with hexahedral, prisms-like, pyramid-like and/or tetrahedral elements for the use in numerical simulation codes (such as *FLAC3D* or *3DEC*). The options for **GInt**, **GSurf**, and **GVol** are described below and are also accessible via the Help panel and F1-help in *Rhino* (after invoking the command).

Numerous examples describing the use of **GInt**, **GSurf**, and **GVol** to create unstructured volume meshes are provided in *Griddle 3.0 Tutorial Examples*.

GInt – Surface Mesh Intersector

Griddle's tool **GInt** intersect surface meshes to make them conformal. It is usually used in non-conformal or not-fully intersected surface meshes. Making meshes conformal is required as the unstructured volume mesher **GVol** operates on conformal surface meshes only (see the workflow in Figure 15).

- In conformal surface meshes, edges and nodes of mesh elements (faces) are fully shared between all elements connected to them.
- In conformal volume meshes, element faces, edges and nodes are fully shared between all elements connected to them.

GInt is able to produce conformal intersections within individual meshes and between separate meshes (see *IntersectionType* setting). Figure 16, left, shows a simple example of two non-conformal meshes in contact (blue and yellow meshes). After using **GInt**, the initial faces at the intersection are (randomly) split into triangular faces to create a conformal contact between the meshes. After intersecting the meshes, remeshing with **GSurf** is typically needed to improve mesh quality.

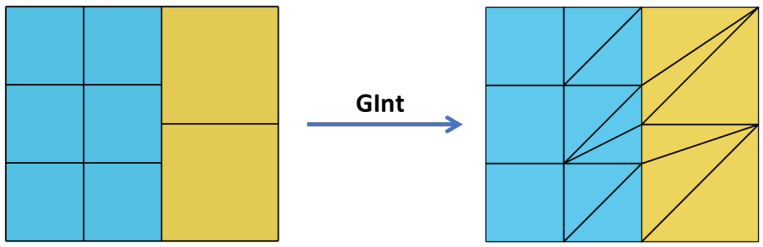


Figure 16: Example of non-conformal (left) and conformal (right) 2D meshes.

Figure 17 shows a typical use of **GInt**. When non-conformal meshes are selected, at the first step **GInt** discovers all intersecting faces within users-specified *Tolerance* and highlights them. Users may adjust tolerance to make sure that all necessary faces are highlighted and will be intersected. After this, **GInt** intersects the detected/highlighted faces, producing triangular faces within intersected mesh patches. All other mesh faces remain unmodified.

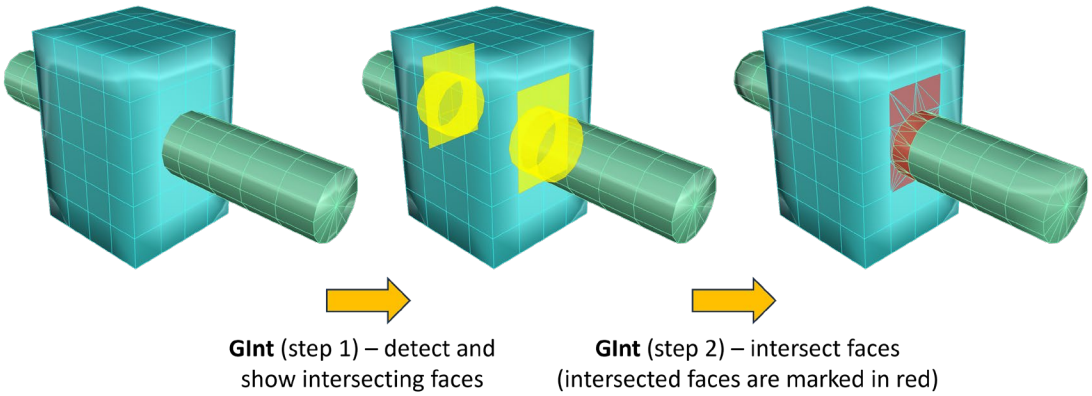


Figure 17: Example of non-conformal (left) and conformal intersecting surface meshes (right) after using GInt.

GInt operates on surface meshes with triangular and/or quadrilateral faces. If Ngons[8] are present in a input mesh, **GInt** will show a warning and will split Ngons into a set of triangular faces.

If surface meshes were originally created from *Rhino* NURBS, SubD, or BRep objects, a slightly different approach may be used to achieve a higher intersection accuracy and better final surface mesh quality. This approach is described in the *Griddle Tutorial Example 3*.

GInt Options

Tolerance [*value* ≥ 0, default = 0]

Tolerance is the most important parameter of **GInt**. It is an absolute distance (in model units) used to determine whether mesh faces intersect each other. If zero tolerance is specified, only those faces are intersected that are in exact contact with each other. Specifying a non-zero tolerance allows searching for nearly intersecting faces (within the tolerance). On the other hand, a large tolerance value may lead to undesired intersections and distortions within each mesh (e.g., small mesh faces with edges less than the tolerance will be collapsed). Users should always analyze and choose the smallest tolerance that provides all necessary intersections. Visualization of the intersecting faces can be a useful tool for this as **GInt** automatically updates the highlight of the intersecting faces when tolerance is changed.

AdvancedParameters

IntersectionType [= *BetweenMeshes*, *WithinMeshes*, *Full* (default)]

By default, **GInt** finds and intersects all faces which fall within specified tolerance (i.e., between separate meshes and within each mesh). However, if *BetweenMeshes* option is selected, only faces between distinct meshes are intersected and internal mesh face intersections are not calculated. If *WithinMeshes* option is selected, only faces within individual meshes are intersected but not between meshes.

SplitInersections [= *No* (default), *Distinct*, *Merged*]

By default, **GInt** intersects faces and keeps them merged within the original meshes. However, if option *Distinct* or *Merged* is selected, **GInt** will split intersected faces from the original meshes. If the user specifies *Distinct* option, each set of split faces will be placed in sub-layer "IntersectedFaces" of the original mesh layer. If option *Merged* is selected, all intersected faces are merged into a single mesh and placed in the default layer.

Extracting intersected faces allows for additional operations on them, for example, assigning specific mesh sizes to them to densify meshes around the intersections.

ShowIntersections [= *Yes*, *No* (default)]

[8] *A mesh Ngon is a mesh face that has more than 4 edges; it consists of more than 2 internal triangles with all interior edges invisible.*

Specifying *Yes* indicates that all intersected faces will be highlighted (in red, by the default) after the operation completes, and the highlight objects will be placed in layer "MESH_INTERSECTIONS". Users can change the highlight color by changing the layer color or turn on/off the layer to show/hide the highlight. Deleting layer "MESH_INTERSECTIONS" removes the highlight objects. Every time **GInt** is called, previous highlight objects are deleted. Note, the highlight objects are non-interactive and created for viewing purposes only; any further mesh changes may invalidate the accuracy of highlighting.

PreviewLimit [*value* ≥ 0, *initial_value* = 1e5]
If the input meshes contain more triangular faces (quads = 2 triangles) than the specified limit, the preview/highlighting of intersecting faces is disabled. This is done to increase the responsiveness of **GInt** when operating with very large meshes. The *PreviewLimit* value should be chosen based on computer performance: the value should be reduced for slower processors and can be increased for faster processors. The value is saved in the system's registry and does not need to be updated every time. To completely disable highlighting of the intersecting faces, set *PreviewLimit* = 0.

Reset resets advanced parameters to the default values.

ShowErrors option allows the enabling/disabling of Ngon warnings.

As **GInt** completes the operation, it outputs a text (ASCII) log file with information about the input and output meshes. The log file uses the name of the *Rhino* project file (e.g., “RhinoProject.3dm”) and adds “_GInt-Log.log” to it (e.g., “RhinoProject_GInt-Log.log”). The file is saved in the same directory where *Rhino* project is saved.

GSurf – Surface Remesher

Griddle's tool **GSurf** is a surface remesher designed for remeshing (or re-building) surface meshes to user-desired configuration, such as changing element size, densifying, smoothing, changing mesh type to triangular, quad-dominant, or all-quadrilateral mesh, etc.

Surface meshes are used by *Griddle's* volume mesher **GVol** as hard boundaries, from which volume meshing starts and which are preserved in the volume mesh (surface mesh faces become volume element faces). Thus, it is important to create good quality surface meshes with desired parameters before using them in volume meshing. Moreover, input to **GSurf** must be one or more conformal meshes or meshes that do not intersect at all to provide conformal output meshes suitable for **GVol**.

The example in Figure 18 shows a simple workflow for creating two conformal quad-dominant meshes densified around their intersection when starting from the original configuration shown on Figure 18a. First, the green and pink meshes are intersected using **GInt** to provide conformity between the meshes, Figure 18b. The intersected faces are highlighted in red. Next, both meshes are remeshed at once using large global element size values (specified via *Min*/*MaxEdgeLength* – see **GSurf** Options), Figure 18c. Finally, Figure 18d shows the same meshes but remeshed once more using fine local size specified around the intersection and the same large global element size (see sections below on how specify local element size). It is seen that **GSurf** creates a smooth transition between different element sizes and the gradation can be controlled.

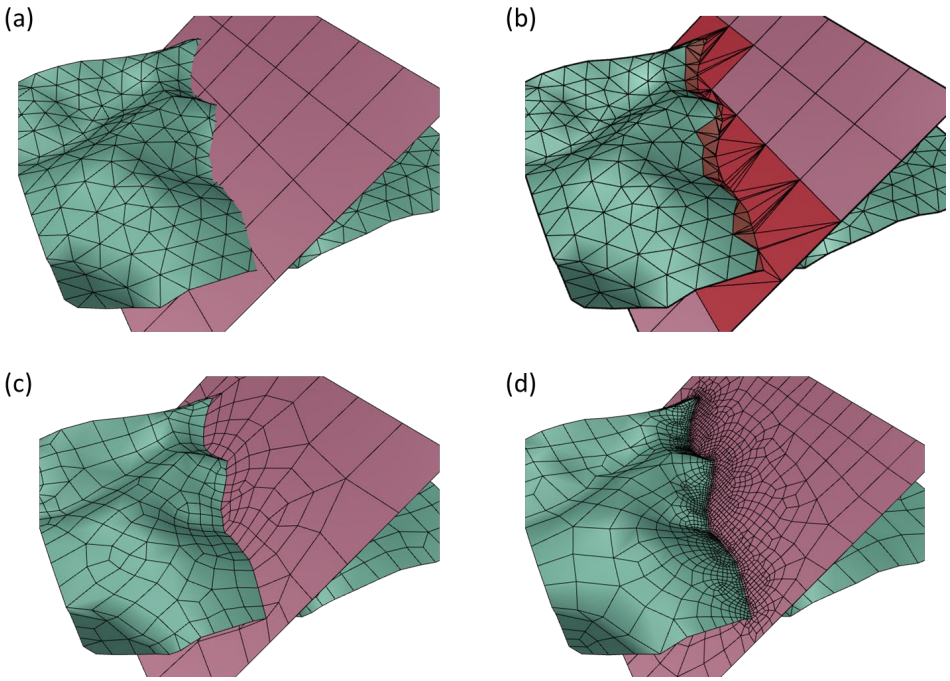


Figure 18: Examples of remeshing with GSurf without (c) and with local size control (d).

GSurf Options

Mode [*Tri* (default), *QuadDom*, *AllQuad*]

This option specifies the type of output mesh:

- *Tri* produces an all-triangle surface mesh.
- *QuadDom* produces a quad-dominant surface mesh (contains a mix of triangles and quadrilaterals).
- *AllQuad* produces a pure quadrilateral surface mesh. In certain cases, it is impossible to create an all-quad mesh and **GSurf** may show an error. In this case use *QuadDom* mode instead.

MinEdgeLength, MaxEdgeLength [*value* > 0]

These parameters control the output element size by setting minimum and maximum allowed edge sizes in the final surface mesh. To get uniform sizes, minimum and maximum edge size can be set to the same value. Edge size is specified in model units. It is important to set a non-zero minimum edge length to get a good quality output mesh.

Note that these values are not strictly enforced in the output mesh. However, **GSurf** attempts to achieve provided limits on edge size through multiple remeshing optimization passes (see *Optimization* below).

RidgeAngle [*value between* 0° *and* 90°, default = 20°]

RidgeAngle, specified in degrees, controls the level of detail (the number of ridge lines) in the resulting mesh. The angle between mesh faces sharing an edge is termed a ridge angle (angle = 0° if faces are coplanar and 90° if faces are perpendicular). Ridge lines can be traced through the surface mesh by joining edges of faces that have ridge angles greater than the specified *RidgeAngle*. Using higher value for *RidgeAngle* results in less ridge lines (less detail) included in the final mesh. A lower *RidgeAngle* results in more detail included in the final mesh. Generally, *RidgeAngle* should be kept below 45°. The default value of 20° is a good compromise between mesh size and fidelity.

AdvancedParameters

MaxGradation [*value* > 0, default = 0.1]

This parameter controls the gradation of element sizes. A value close to 0 leads to a more gradual variation of mesh size (smoother), while higher values lead to more abrupt changes in element size.

Optimization [*value between* 0 *and* 10, default = 5]

This parameter controls the optimization of the mesh. A zero value makes **GSurf** skip the optimization step; the remeshing speed is the highest in this case, but the quality may be poor. From value 1 on, the optimizer algorithm uses several techniques to improve both the shape quality and the size quality of the elements, such as node smoothing, edge swapping, node insertion, and node removal. Level 5 is usually a good trade-off between quality and speed.

QuadWeight [*value between* 0 *and* 1, default = 0.75]

QuadWeight controls the preference of quadrilaterals vs. triangles in output mesh. This parameter is used only in quad-dominant meshing mode. If *QuadWeight* = 0, quadrilaterals are never used. If *QuadWeight* = 0.5, quadrilaterals are used only when they improve the quality of the mesh. For values between 0.5 and 1, quadrilaterals are used more even if it leads to a lesser quality mesh. If *QuadWeight* = 1, the minimum number of triangles is used. Note that there is no linear relation between *QuadWeight* and the ratio of quads to triangles.

ShapeQuality [*value between* 0 *and* 1, default = 0.7]

ShapeQuality controls the trade-off between shape optimization and size optimization. The default value (0.7) gives a slightly stronger preference to the element shape quality over the size quality. For example, an elongated surface mesh patch can be remeshed using few elongated triangles or quads (if *ShapeQuality* is close to 0) or it can be remeshed with many smaller but almost perfect triangles or quads (if *ShapeQuality* is close to 1).

OutputMesh [= *Distinct* (default), *DistinctNoInpBnds*, *Merged*]

Internally, remeshing is done on individual mesh patches which are separated by "Skeleton" lines (obtained based on the original mesh boundaries) and Ridge lines (based on the *RidgeAngle*). Association between the input and output meshes is done by matching remeshed patches to the original ones.

- By specifying *Distinct* value, **GSurf** will associate output meshes with the distinct input meshes and will assign original mesh parameters to them (such as name, layer, color, etc.) All Skeleton and Ridge lines are taken into account in this case.
- When *DistinctNoInpBnds* option is selected, only Ridge lines are used to define mesh patches (Skeleton lines are ignored). This may lead to better quality remeshing but association between the input and output meshes may be poor. *DistinctNoInpBnds* option corresponds to the **GSurf** operation mode in *Griddle* 1.0.
- When *Merged* is selected, *RidgeAngle* is used to define mesh patches and no association between input and output meshes is done (a single output mesh is generated).

The patches that could not be associated with any source are placed into a layer "MISMATCHED_PATCHES".

DeleteInput [= *Yes* (default), *No*]

Specifying *Yes* in this parameter indicates that the original selected meshes will be deleted. If *No* is specified, the original meshes will remain intact.

Reset resets advanced parameters to the default values.

Additional Edge Size Control Options

In addition to the global values of min and max edge size specified in the parameters above (which are applied to all selected meshes), local overrides can be made to specify the desired element size for a surface mesh. Local values supersede the global Min and Max edge sizes values set in **GSurf**.

Local element size for a mesh can be specified in one of two ways:

1. By specifying element size via a hyperlink available through object properties (Figure 19):
   - Select a mesh
   - Open the *Properties* panel and click on the ellipsis (...) next to the *Hyperlink* property
   - In the *Hyperlink* dialog choose *Type: (other)* and type in the URL field "elemsize:*NumericValue*". Click **OK**. No spaces are allowed in the URL field.

2. By setting mesh name in the *Properties* panel to a numeric value which represents the required element edge size (see image below).

Griddle first checks if element size is specified in the *Hyperlink* field and if nothing is found, it will check the Name field for a numeric value. Specifying local element size via hyperlink is preferred for several reasons: (1) the mesh name is already set and should not be changed, (2) the mesh name may contain numeric values which could be confused with element size, (3) clean syntax.

Figure 19 shows two ways to specify local element size (*Name* field or *Hyperlink*). In this example only the yellow part of the mesh is assigned local element size, after which all meshes are remeshed with **GSurf**. The result of remeshing (bottom image) shows a much finer mesh at and around the yellow part, which corresponds to the provided element size.

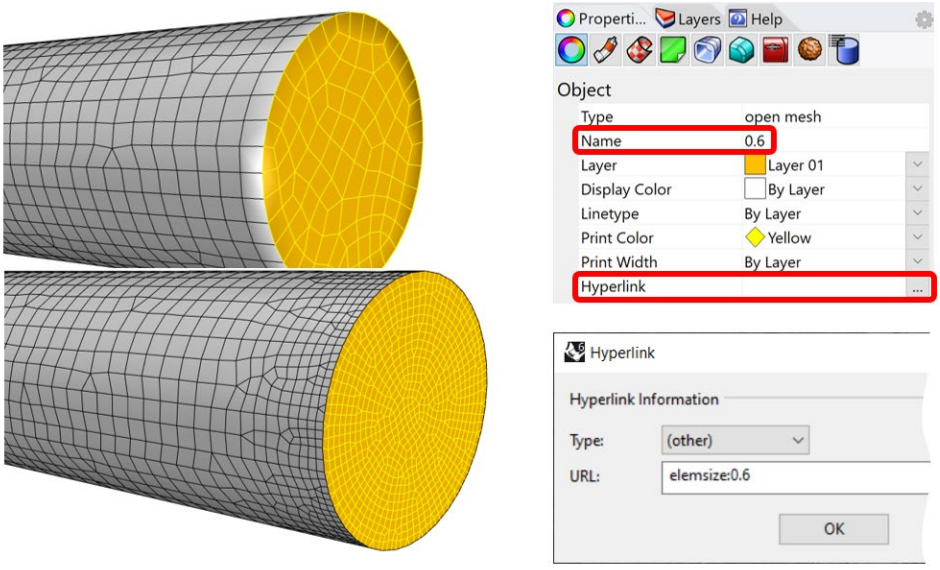


Figure 19: Two ways to specify local element size for surface meshes.

Similarly, local element size can be specified locally at a point, by creating a point (with *Rhino*'s **_Point** command) and assigning a numeric value to the point's hyperlink or name (as shown above). Such points must be coincident with the existing mesh vertices (enable vertex snapping in *Rhino* to snap new points to mesh vertices). If no element size is specified at a point, such a point will be treated as a hard node and a mesh vertex corresponding to this point will remain at the same position in the output mesh. Element size around this point will be determined automatically.

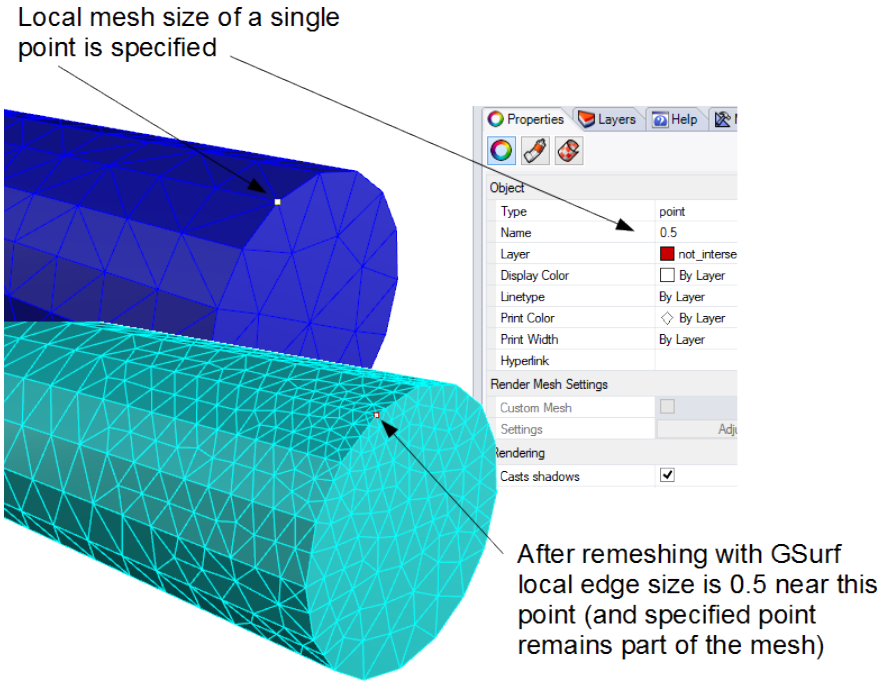


Figure 20: Specifying local element size at a point.

Hard Edges and Hard Nodes

Hard edges are edges that are preserved in a mesh during the remeshing process. Consider an example in Figure 21 (left), which shows two conformal surface meshes. If one of the meshes (for instance, the red mesh) is changed (for example, refined) while the other is unmodified, the conformity of the meshes will be broken. **GSurf** offers a way to remesh a specific or all meshes while preserving discretization and conformity along mesh edges. This can be done by creating hard edges.

1. Use *Rhino*'s **_DupBorder** command on the red mesh, which will duplicate the mesh border (i.e., will create a curve connecting only nodes on the mesh boundary). Delete unnecessary segments of the curve (use the **_SubCrv** command) leaving only those segments that connect to the green mesh (as shown in Figure 21, left).
2. Select both the red mesh and the remaining part of the curve (polyline) and call **GSurf** with desired meshing parameters (in this case, smaller edge size). **GSurf** will recognize the polyline as a hard edge and will remesh the red mesh while preserving the discretization along the hard edge.

The remeshed red mesh will be finer, and both meshes will still be conformal.

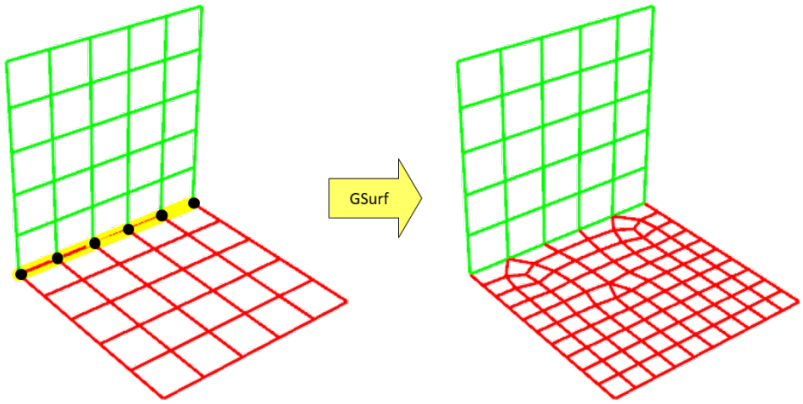


Figure 21: Specifying hard edges.

Similarly, a hard node can be defined to preserve the location of a vertex. Information about hard nodes is provided in the previous section.

Mesh and Element Quality Information; Output Log

GSurf outputs information about input and output meshes, meshing parameters, and element quality information to a log file (ASCII). The element quality information is based on shape quality analysis of each element in the remeshed mesh. The following information is provided.

- `Total area:` The cumulative surface area of all elements in the remeshed mesh in model units.
- `Element min shape quality (QS) value:` Minimum normalized shape quality among all elements in the remeshed mesh. The values of shape quality range between 0 and 1. Detailed information about shape quality calculations is provided below.
- `Max error distance:` Measure of the geometric error between the input and output (remeshed) meshes. The nodes of the remeshed mesh are located exactly on the input mesh surface. However, this is typically not the case for the centroids of new elements, and new nodes may not coincide with the initial nodes. If the input mesh surface is not flat, faces in the remeshed mesh may be some distance away from the original mesh. The `Max error distance` parameter is a measure of this maximum distance (*Hausdorff distance*).
- `Histogram QS for (All/Specific) Elements:` Provides information about distribution of shape quality (QS) values. The bins (intervals) of the histogram are of equal size.
    - `Total number of bins:` Number of bins (intervals) in the histogram. The default value is 11.
    - `Total number of counts:` Total number of counts in the histogram equal to the number of elements (all or specific type, e.g., quadrilaterals) in the output mesh.
    - `Number of larger values:` Number of hits (elements) with QS above the largest histogram bin value.

- `Number of smaller values`: Number of hits (elements) with QS below the smallest histogram bin value.
- `V max`: Maximum shape quality (QS) value among all values in the histogram.
- `V mean`: Mean shape quality (QS) value among all values in the histogram.
- `V min`: Minimum shape quality (QS) value among all values in the histogram.

Next, the information about shape quality values distribution in the histogram is provided:

```
Bin number        -- Bin boundaries --          Hits
```

Shape quality of an element QS is calculated based on the following expressions:

For triangular elements: $Q_s = 4\sqrt{3}\,S/(L_{max}P)$,

where

- S is the area of the triangle,
- L_{max} is the length of the longest edge of the triangle,
- P is the perimeter of the triangle.

For 3D quadrilateral elements: $Q_s = Q_s^{2D} Q_w$

Here, Q_s^{2D} is 2D shape quality: $Q_s^{2D} = 8\sqrt{2}\,S_{min}/(L_{max}P)$,

where

- S_{min} is the minimum area of the four triangles within the quadrilateral,
- L_{max} is the max length of the four sides and the two diagonals,
- P is the perimeter of the quadrilateral,
- and Q_w is warp quality of the quadrilateral: $Q_w = 1 - \frac{acos(min[\langle n_0,n_2\rangle,\langle n_1,n_3\rangle])}{\pi}$, where n_i is the normal at node *i* (or normal to the triangular face *i* of the quadrilateral).

The output log file uses the name of the *Rhino* project file (e.g., "RhinoProject.3dm") and adds "_GSurf-Log.log" to it (e.g., "RhinoProject_GSurf-Log.log"). The log is output to the same directory where the *Rhino* project is saved.

GVol – Unstructured Volume Mesher

GVol is *Griddle*'s unstructured volume mesher designed to create tetrahedral and hex-dominant meshes by using selected surface meshes as inputs (Figure 22). Surface meshes can be composed of triangles, a mix of quadrilaterals and triangles, or all quadrilaterals. The latter two should be used for generating hex-dominant grids. Tetrahedral grids can be generated from triangle, triangle-quad, or all-quad surface meshes (quadrilaterals are arbitrarily split along one of their diagonals into triangles).

There are two main requirements for **GVol** to be able to generate a volume mesh:

- a combination of selected surface meshes must form a watertight boundary surrounding the entire volume of interest, and
- all intersecting or connecting surface meshes must be conformal.

Surface meshes can form discrete volumes within other larger volumes; this will create separate domains each filled with volume elements (Figure 22). Surface meshes can also "float" inside the volume of interest or be partially connected to other surface meshes. All surface mesh faces, including "floating" surface meshes inside a volume, are included as "hard faces" in the final volume mesh. This means that all input surface faces will be present as the faces of elements in the resulting volume mesh. If **GVol** is unable to enforce a hard face, which may happen for nonplanar quads on the domain boundaries, such quads will be split into two triangles, and they will be reported as 'Unenforced input faces' in the log and "GVOL_OUTPUT" information layers.

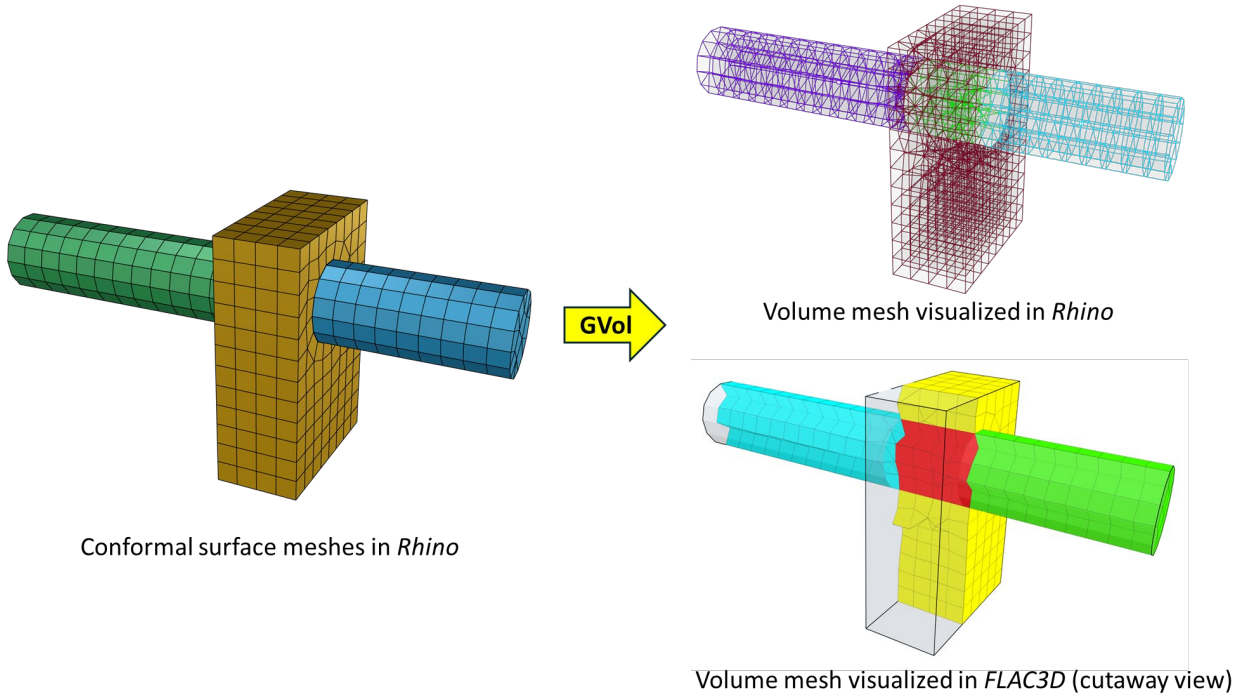


Figure 22: Volume mesh creation from surface meshes.

GVol requires input surface meshes to be properly connected. Ngons, duplicate, overlapping, or intersecting surface mesh faces are not allowed. If **GVol** produces errors, check input meshes for issues using *Griddle*'s **GHeal** tool and fix them as needed (note that remeshing may be required after fixing issues). If issues remain after using **GHeal**, consider re-intersecting surface meshes using **GInt** with higher tolerance and then remeshing all meshes again.

GVol output in *Rhino*

While **GVol** outputs volume meshes into a file of user-specified format, it also can provide important information about the generated meshes directly in *Rhino*, so users can quickly assess possible issues, the quality of generated meshes, and meshed subdomains or groups. This information is provided via a set of layers (and objects within them) visible in the *Layers* panel. An example is shown in Figure 23 and the complete layers organization as provided below:

- **GVOL_OUTPUT**
  - Meshing Issues
    - Naked Edges (count)
    - Ngons (count)
    - Nonconformal Faces (count)
    - Unenforced Faces (count)
    - Unenforced Edges (count)
    - Unenforced Nodes (count)
  - Volume mesh output filename (types and the count of elements)
    - Poor Elements (count)
      - 0.09 < Qs < 0.1 (count)
      - …
      - 0.0 < Qs < 0.01 (count)
    - Element Groups (count)
      - ZG_001 (Volume = …)
      - …

"Meshing Issues" layer provides information about possible issues **GVol** may encounter during meshing.

- "Naked Edges" and "Ngons" are reported only if *ShowWarnings* setting is set to *Yes*.
- If input meshes contain non-conformal intersections preventing **GVol** from running, outlines of such faces will be reported in layer "Nonconformal Faces".
- Entries in layer "Unenforced Faces" outline mesh faces which **GVol** was not able to use or strictly enforce in volume mesh. For example, if some input surface mesh faces (hard faces) are located outside of closed domains, they will not be used in volume meshing, therefore they are counted as unenforced faces (see more information about unenforced faces in section Mesh and Element Quality Information). "Unenforced Edges / Nodes" refer to unenforced (skipped) hard edges and nodes in volume mesh (see Hard Nodes and Edges section).
- **GVol** reports the count of each issue type.

"Volume mesh output filename" layer corresponds to the filename where volume mesh was output. This layer contains information about the volume mesh: elements type and count, information about "Poor Elements" and "Element Groups". The last two options are only provided if the user specifies to visualize mesh (see *VisualizeOutput* option below). The description of "Element Groups" is provided in the next section. For the details about the "Poor Elements", see *LowQs_Elems* option below.

Volume mesh visualization

GVol is capable of visualizing generated volume meshes and poor-quality elements directly in *Rhino* viewports. For this, the user needs to set corresponding *VisualizeOutput* settings to *Yes*. An example of the visualized meshes is provided in Figure 23. Volume meshes are rendered semi-transparent if the number of volume elements is under 1e4 or as wireframe meshes for larger number of elements. Visualized volume meshes are non-interactive – this significantly speeds up mesh rendering and allows displaying volume meshes of almost any size (up to hundreds of millions of elements). For very large volume meshes (e.g., the number of elements >> 1e7), visualization may consume significant amount of memory and increase **GVol** operation time; thus, it is recommended to set the visualization option to *No* if mesh visualization is not needed or if computer resources are limited.

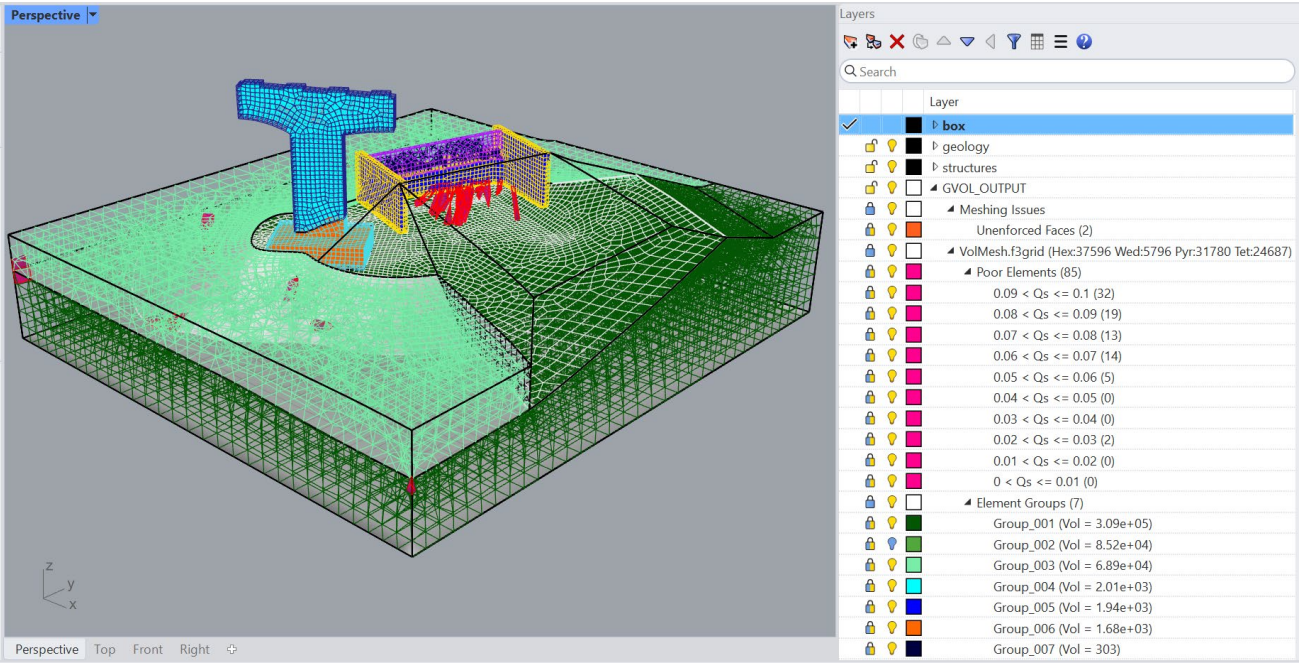


Figure 23: A model showing surface meshes, generated volume mesh, and information about element types and count, meshing issues, poor quality elements (in pink/red), and element groups. Note that some surface meshes are not shown.

- Visualized volume mesh groups are organized according to the generated element groups (from largest volume to the smallest volume) and named in *Rhino Layers* panel as shown in Figure 23.
- Visualized volume mesh groups are placed into separate layers under "GVOL_OUTPUT" ⇨ "*OutputFilename* (element count)" ⇨ "Element Groups (count)". The user is able to interact with visualized volume meshes by switching the layers on/off and/or changing their colors. Deleting layers removes corresponding visualized mesh group (but does not alter any data output to files).

- **GVol** automatically creates points at the subdomain centroids – this is to provide the ability to zoom in/out the non-interactive mesh objects (users can delete these points after unlocking corresponding layers).
- Visualized volume meshes are not saved with *Rhino* project (*.3dm) – they only exist during *Rhino* runtime. At the same time, layers with information about group names and volumes are saved (but they will be empty next time after loading a saved project).
- **GVol** deletes all previously output visualization each time it is called.

Hard nodes and edges

GVol is able to take into account isolated hard nodes and hard edges that are present within the close volume domain. Volume mesh vertices will be aligned with such hard nodes and edges (according to their discretization). In addition, the edges of connected volume elements will be aligned with the hard edges (see Figure 24 and Figure 25).

- **Hard nodes:** Assignment of a size value to a hard node will force the mesh generator to produce volume elements around the node with the overall size close to the specified value (this can be achieved when hard nodes are far enough from the domain internal and external boundaries). If no element size is assigned to a node, a volume mesh vertex will be placed at the node location, but the size of the surrounding element will be determined automatically. Element size can be assigned via object's Name property or Hyperlink field (see section Additional Edge Size Control Options in **GSurf** description).

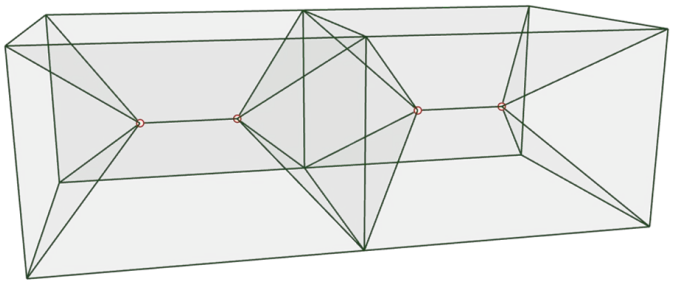


Figure 24. A volume mesh with 4 hard nodes.

- **Hard edges:** Only two types of hard edges are allowed: lines and polylines. Linear hard edges allow for further discretization according to the assigned element size (via object's Name property or Hyperlink field). Discretization will be approximate to fit the whole number of segments in the edge. Volume mesh vertices will be aligned with the ends of the hard edges and the interior nodes according to the discretization. If a hard edge is a polyline, only the ends of each segment are taken into account; element size, even if assigned, is not used for the generation of volume mesh. For either edge type, generated volume elements edges will be aligned with the edges.
  Note: Hard edges are not allowed to cut through each other nor through hard faces (surface meshes) or their edges. An error will be generated in such cases.

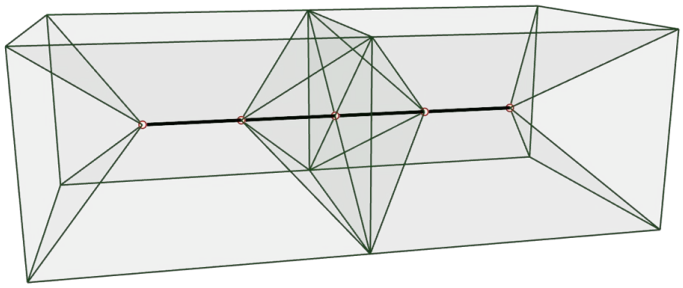


Figure 25. A volume mesh with a hard edge split into 4 segments.

GVol options

MeshSettings

Mode [*Tet* (default), *HexDom*]

This option specifies the type of the output mesh:

- *Tet* (default) produces an all-tetrahedral volume mesh. Any quadrilaterals on surface meshes are internally converted to triangles in this case.
- *HexDom* produces a conformal hex-dominant volume mesh. It is recommended for the input surfaces to be quad-dominant or quad only type meshes for this option.

MaxGradation [*value* > 0, default = 0.5]

This parameter controls the gradation of elements from the size on the boundary to the preferred size defined by *TargetSize* inside the domain (far enough from the boundaries). A value close to 0 leads to a more gradual variation of size while higher values lead to more abrupt changes in element size. *MaxGradation* has an effect only when *TargetSize* > 0.

TargetSize [*value* > 0, default = 0]

This parameter specifies the preferred element size inside the domain. The element size tends toward this value as elements get away from the boundaries. The default value of 0 indicates that the element size inside the domain is automatically determined based on the size of the elements on the boundary faces. Make sure that this parameter is meaningful and is not too small compared to the elements size on the boundary faces; otherwise, mesh generation may take a very long time as a large number of small elements will be generated inside the domain.

Optimization [*value between* 0 *and 10*, default = 5]

This parameter controls the optimization of the mesh. A zero value makes the mesher skip the optimization step. The speed is the highest, but the quality may be poor. From value 1 on, the optimizer algorithm uses several techniques to improve both the shape quality and the size quality of the elements, such as node smoothing, edge swapping, node insertion and node removal. Level 5 is usually a good trade-off between quality and speed.

ShapeQuality [*value between* 0 *and 1*, default = 0.7]

This parameter controls the trade-off between shape optimization and size optimization. The default value (0.7) gives a slightly stronger preference to the element shape quality over the size quality. For example, an elongated volume region can be filled with few elongated elements (if *ShapeQuality* is close to 0) or it can be filled with many smaller but much higher quality shape elements (if *ShapeQuality* is close to 1).

InclHardNodesEdges [= *No* (default), *Yes*]

This option specifies whether to include isolated points and edges located within a close volume into meshing. If this option is set to *Yes*, any point or line/polyline located within the volume will be considered as a hard node/edge, and generated volume mesh will contain vertices aligned with such nodes and edges (according to the edge discretization). For details about hard nodes and hard edges see description below.

ShowWarnings

The option specifies which checks will be carried out before meshing (e.g., *none*, presence of *Ngons*, *Naked Edges*, or *All*). If any of the checks fail, a warning message will be shown.

VisualizeOutput [= *No* (default), *Yes*]

This setting contains options to visualize volume mesh and/or elements with poor quality.

VolumeMesh [= No (default), Yes]

When this option is set to *Yes*, **GVol** will visualize generated volume mesh. For the detailed description and an example, see section Volume mesh visualization.

LowQs_Elems [= *No* (default), *Yes*]

When this option is set to Yes, **GVol** will visualize the lowest quality elements in the generated volume mesh (details about element quality calculations are provided in section Mesh and Element Quality Information). Only elements with quality metric Qs under 0.1 are visualized and placed in 10 bins from 0 to 0.1 (corresponding layers are created under "GVOL_OUTPUT" ⇨ "*OutputFilename* (element count)" ⇨ "Poor Elements (count)"; layers are locked by default). As the number of low-quality elements is typically limited, the elements are visualized by using interactive mesh objects. Thus, users can select and manipulate these objects after unlocking the corresponding layers. Mesh objects corresponding to poor quality elements are saved with *Rhino* project (*.3dm).

Note that **GVol** erases all previously output visual information each time it is called. *VizualizeOutput* options are persistent and saved/loaded each time *Rhino* is launched.

OutputFormat [= *FLAC3D, 3DEC_5x, 3DEC, ABAQUS, ANSYS, LS-DYNA, NASTRAN, VTK*]

OutputFormat specifies format in which the resulting volume grid file should be saved. If the VTK option is selected, the output file name (including the path) must only consist of ASCII characters. If ABAQUS, ANSYS, or LS-DYNA options are selected, *Griddle* also outputs zone face groups as nodal components / lists, which can be used to specify FEA boundary conditions.

FormatType [= *Binary, Text*]
This option appears only if the user selects *OutputFormat* = *FLAC3D* or *3DEC.* The option sets how to save the output file: in binary or text (ASCII) format. Saving and loading (reading) files in binary format is much faster compared to the text format, but binary files are not human-readable. All other *OutputFormat* choices allow saving in text format only.

BlockType [= *Rigid, Deformable*]
This option appears only if the user selects *OutputFormat* = *3DEC*. It specifies whether each element of the output volume mesh should be represented as a rigid block (*BlockType* = *Rigid*) or as a deformable zone (*BlockType* = *Deformable*) in *3DEC*. In the latter case, watertight volumes in the model (closed volumes separated by surface meshes) will be represented as rigid blocks filled with deformable zones.

FaceGroups [= *FromAllSurf, FromNamedSurf*]
This option only appears if the user selects *OutputFormat* = *FLAC3D*. It specifies if face groups will be created from all surface meshes or only from those surface meshes which have a name assigned to them in *Rhino*. This option is useful if a model contains a large number of surface meshes but only some of them have some significance and have names assigned to them (*FLAC3D* face groups can be created only for such a subset of meshes).

Slots [= *Zone/BlockSlot="Griddle", Face/JointSlot="Griddle"*]
This option only appears if the user selects *OutputFormat* = *FLAC3D* or *3DEC*. The option allows specifying custom slots for zone and face groups (or block and joint groups for *3DEC*). As of *Griddle* 3.0, the defaults names for both slots are "*Griddle*". Note that only alphanumeric ASCII symbols (0-9, a-z, A-Z) can be used in slot names.

AutoOutputName [= *N/A* (default), *UserDefinedName*]
This option sets a string that will appear in the output filename along with other identifiers. For example, for a *Rhino* project named “RhinoProject.3dm”, a *3DEC* rigid blocks output in binary format will be named: “RhinoProject_*UserDefinedName*_Rigid_Binary.3dgrid”). If the user specifies a string in this option, the **Save As** dialog will not appear, and the output file will be named automatically as described above. If *AutoOutputName* is not specified (N/A), the “Save As” dialog will be displayed in order to obtain the output file name and location.

- Note that the value for this option is not saved: each time user clicks on this option, a new string must be typed. To erase the existing string in *AutoOutputName*, click on the option and press **Enter** without typing anything; this will cause the “Save As” dialog appear when executing **GVol**.
- The output file will be saved in the same location as the *Rhino* project file. If a file with the same name already exists in that location, it will be overwritten.
- This option allows running **GVol** in automatic mode without user interaction (i.e., the need to type file name and press **OK** / **Cancel** in the “Save As” dialog). This is particularly useful when invoking the **GVol** command from *RhinoScript*, *PythonScript*, or other *Rhino* tools and plugins.

Notes

If *OutputFormat* = *3DEC* and *BlockType* = *Deformable*, block joints will not be created along floating surface meshes located (fully or partially) within watertight volumes that will become rigid blocks (Figure 26). Such floating surface meshes (meshes with naked edges) do not split or define separate rigid blocks and currently cannot be represented as joints in *3DEC* (though, they can be represented as interfaces in *FLAC3D*).

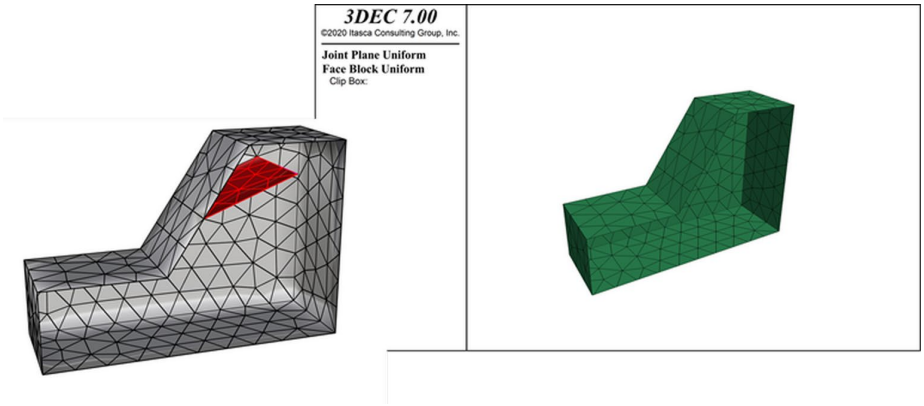


Figure 26: *Rhino* model (left) with floating surface (red) and *Joint* plot in *3DEC* representing the rigid block (right). The red surface does not become part of the rigid block (or a joint) in *3DEC*.

Mesh and Element Quality Information; Output Log

GVol outputs information about input and output meshes, meshing parameters, and element quality information to a log file (ASCII). The element quality information is based on shape quality analysis of each element in the volume mesh. Some of this information is also provided within *Rhino* interface when visualizing generated mesh.

- `Meshed volume`: The volume of all elements in the output mesh in model units.
- `Number of meshed subdomains`: Number of closed volume (watertight) subdomains present in the meshing volume.

- `Missed (unenforced) faces`: This parameter shows how many faces **GVol** was not able to use or strictly enforce in volume mesh. For example, if some input surface mesh faces (hard faces) are located outside of closed domains, they will not be used in volume meshing, therefore they are counted as unenforced faces. Furthermore, to improve volume mesh quality, **GVol** may connect two tetrahedrons (or other elements with triangular faces) to a non-planar quadrilateral hard face, which will be counted as an unenforced face.
  - `Element min shape quality (QS) value`: Minimum normalized shape quality among all elements in the volume mesh. Shape quality value ranges between 0 and 1. Detailed information about shape quality calculations is provided below.
  - `Histogram QS for (All/Specific) Elements`: Provides information about distribution of shape quality (QS) values. The bins (intervals) of the histogram are of equal size.
    - `Total number of bins`: Number of bins (intervals) in the histogram. The default value is 11.

- `Total number of counts`: Total number of counts in the histogram equal to the number of all/specific elements in the output mesh.
- `Number of larger values`: Number of hits (elements) with QS above the largest histogram bin value.
- `Number of smaller values`: Number of hits (elements) with QS below the smallest histogram bin value.
- `V max`: Maximum shape quality (QS) value among all values in the histogram.
- `V mean`: Mean shape quality (QS) value among all values in the histogram.
- `V min`: Minimum shape quality (QS) value among all values in the histogram.

Next, the information about shape quality values distribution in the histogram is provided:

```
Bin number        -- Bin boundaries --          Hits
```

Shape quality of an element QS is proportional to the minimum normalized Jacobian of the element. Element shape quality typically ranges from 0 for a degenerated element to 1 for a perfect element, however negative QS values may potentially be reported if warped elements (for which min Jacobian is negative) are present in the mesh. **GVol** typically avoids creating such elements either by splitting boundary quadrilaterals (if they are close or connected to the degenerate element) or by generating several simpler topology elements (e.g., tets, pyramids) in place of the warped element.

For tetrahedral elements: $Q_s = 6\sqrt{6}\,V/(L_{max}S)$, where

- V is the volume of the tetrahedron,
- L_{max} is the length of the longest edge of the tetrahedron,
- S is the total area of the four faces of the tetrahedron.

Note that the Jacobian of a linear tetrahedral element is equal to six times its volume.

For other 3D elements: $Q_s = \alpha Q_w^{min} \cdot J_{min}/L^3$, where

- α is the scaling factor to scale to 1 for a perfect element (different for each element type),
- Q_w^{min} is the minimum warp quality of quadrilaterals faces for elements containing such faces (see expression for Q_w in the **GSurf** Section); $Q_w^{min} = 1$ for triangular faces,
- J_{min} is the minimum of the Jacobians calculated at all the vertices (equals to the determinant of the mixed product of 3 edge vectors at each vertex); for pyramids, the minimum Jacobian at the apex is computed as the minimum of the 4 Jacobians when taking each 3 edge vectors out of 4,
- L is the square root of the sum of the square lengths of all edges (or the edges used to calculate J_{min} for pyramid).

Note that the approach to shape quality calculation in *Griddle* may differ from shape quality metrics calculation in Finite Element (e.g., *ABAQUS*, *ANSYS*, etc.) or Finite Volume (*FLAC3D*, *3DEC*, etc.) software.

Full elements quality information is output into a log file which uses the name of the *Rhino* project file (e.g., "RhinoProject.3dm") and adds the following to the filename: "_GVol" + format type (Text/Binary, if applicable) + Output Format (e.g., ".f3grid", ".inp") + ".log" (e.g. "RhinoProject_GVol_Binary.f3grid.log"). The log is output to the same directory where the *Rhino* project is saved.

Group Names Assignment

Many numerical simulation codes can associate aliases or groups with different parts of a volume mesh and also associate aliases or groups with internal and external boundaries and element faces. Such named entities make it easier to reference parts of the model for material properties or boundary conditions assignment, creation of interfaces or contacts, or other operations.

To generate a volume mesh, **GVol** requires a set of surface meshes as an input. Multiple open and closed domains may be formed by the intersections between input surface meshes, and it is not clear beforehand if sets (or subset) of surface meshes form watertight domains. For this reason, volume element groups are assigned automatically only after volume mesh generation is completed. The assignment is done in a consistent order starting from meshed domains with the largest volume and ending with the smallest volume. Input surface mesh names are transferred to volume mesh as surface/face groups.

Group names in FLAC3D

Zone groups

FLAC3D zone group names are based on the closed volume ID to which they belong. To distinguish zone groups from other group names, **GVol** assigns zone groups the prefix "ZG_" and a suffix corresponding to the closed volume ID. For example, there are 7 closed volumes identified in the model shown in Figure 23, and zones are assigned groups "ZG_001" for the largest volume through "ZG_007" for the smallest volume (corresponding to the displayed groups "Group_001" – "Group_007").

Zone groups are assigned a *FLAC3D* Slot specified in **GVol**'s "Slots" setting.

Face groups

FLAC3D face groups are assigned automatically according to the following rules:

- If zone faces belong to a surface mesh with non-empty Name field specified in *Rhino*'s object *Properties* panel, then the full string from the Name field is used.
- If zone faces belong to a surface mesh with no name, IDs of the volumes attached to the faces are used.
    - For faces on the external model boundaries, the prefix "EF_" is used followed by the volume ID to which the faces are attached, for example, "EF_001".
    - For faces on the internal model boundaries separating two volumes, the prefix "IF_" is used followed by the two volume IDs (the larger number is placed last), for example, "IF_001_002".
    - For faces on the internal model surfaces located within a single volume, the prefix "IF_" is used followed by the volume ID twice, for example, "IF_003_003".

Face groups are assigned a *FLAC3D* Slot specified in **GVol**'s "Slots" setting.

Group names in 3DEC

Block groups

3DEC block groups are represented by strings with integers. Block groups start at "001" for *3DEC* output and at "10001" for *3DEC 5.x* (due to *3DEC 5* specifics), and each new group name is increased by "1". Block groups are assigned in a consistent order starting from the block with the largest volume and ending with the smallest volume (only for *3DEC* output; *3DEC 5* block group order is random).

Block groups are assigned a *3DEC* Slot specified in **GVol**'s "Slots" setting (except for *3DEC 5x*).

Joint groups

Named joints are created for the boundaries between neighboring volumes only. If such boundary (i.e., a surface mesh) has a name assigned to it in *Rhino*'s object *Properties* panel, this name is used as the joint group name. Otherwise, joint groups are arbitrarily assigned a numeric value starting with a minimum value of "3" (this follows the logic for JointSet ID assignment in which the exterior boundaries have ID=1 and any unassigned faces get joint ID=2).

Note that "floating" surface meshes (for example, as shown in Figure 26) will not appear in *3DEC* deformable output and therefore no names would be assigned to them.

Joint groups are assigned a *3DEC* Slot specified in **GVol**'s "Slots" setting (except for *3DEC 5x*).

Group Names in Other Formats

Element groups

For the Finite Element (and other) codes, **GVol** assigns element group names belonging to separate closed volumes, as follows:

- *ABAQUS* input (**.inp**): "G10001", "G10002", "G10003", etc.
- *ANSYS* Mechanical APDL common database (**.cdb**): "1", "2", "3", etc.
- *LS-DYNA* Input (**.k**): "1", "2", "3", etc.
- *NASTRAN* Bulk Data (**.bdf**): "1", "2", "3", etc.
- *VTK*: no group assignment

Element groups are assigned in a consistent order starting from the closed domains with the largest volume and ending with the smallest volume.

Surface/boundary groups

For *ABAQUS*, *ANSYS*, and *LS-DYNA* formats, **GVol** outputs the model interior and exterior boundaries separating closed volume domains and other named surfaces as **node sets**.

If a surface mesh or domain boundary has a name assigned to it in *Rhino*'s object *Properties* panel, this name is used as the surface group name. Otherwise, surface groups are arbitrarily assigned a numeric value starting with a minimum value of "1".

Griddle Utilities for Working with Surface Meshes

GHeal – Tools for Surface Mesh Repair

GHeal is a *Griddle* utility designed to identify and fix various surface mesh issues. In automatic mode, **GHeal** removes Ngons (faces with more than 4 edges), identifies and removes duplicate mesh faces, fixes clashing (intersecting) mesh faces, removes small non-manifold mesh parts, fills mesh holes, aligns face normals. Each of these operations are referred to as **GHeal** functions. Various functions can be selected to run automatically one-after-another or functions can be executed one at a time to target specific mesh issues. If used on multiple and/or large meshes, **GHeal** identifies and fixes mesh problems more efficiently if compared to built-in *Rhino* tools or commands.

GHeal also has a *Custom* operation mode which it calls *Rhino*'s command **_MeshRepair**. This command is useful in repairing small or isolated problems, e.g., patching a single hole where the user must select the boundary of the hole. Refer to *Rhino's* help for more information about **_MeshRepair**.

GHeal Options

ShowErrors

This tool identifies various issues in selected surface meshes and creates outlines of faces with problems while assigning the outlines to specific *Rhino* layers. The reported issues are:

Naked edges are open face edges which belong to a single face only (not shared by connected edges). Naked edges often represent mesh boundaries and discontinuities; they can be found along disjoint and/or nonconformal faces and may not be visible (as faces may visually appear to be connected).

Clashing faces are mesh faces which intersect or nearly-intersect each other within the model or user-specified tolerance. In general, if any of the face components (vertices, edges, face surface) are located within the specified tolerance from any other face component or if faces are in contact or intersect, such faces will be identified as clashing. Furthermore, mesh faces (or holes) with one or more edges smaller than the specified tolerance and their neighbors are also tagged as clashing faces. This definition of clashing faces allows identifying various mesh issues: overlapping faces, intersecting faces, folded faces (Z-folds), non-conformal faces, faces smaller than the model/user tolerance, sliver holes, etc. By the default, the option uses the model Absolute tolerance (see example in Figure 4); the user can modify this value by specifying the tolerance multiplier in *AutomaticHeal* → Advanced*Parameters* → *ToleranceMulitplier*.

Ngons are mesh faces that have more than 4 edges; they consist of more than 2 internal triangles with all interior edges invisible. Ngons may look like quadrilateral faces but will not be suitable for any of *Griddle*'s operations and must be fixed. Note that some of *Rhino*'s commands produce Ngons by default (for example, mesh splitting with **_MeshSplit** command).

Duplicate Faces are duplicated mesh faces joined with the rest of the mesh. Mesh faces may be duplicated more than once but they usually appear as a single face. Duplicated mesh faces are not identified as clashing faces and must be fixed separately.

ShowErrors option creates a new layer "GHEAL_OUTPUT" containing the results of mesh error analysis. Within this layer, sublayers corresponding to various issues are created (e.g., the "Naked Edges" sublayer contains outlines of naked edges). Each sublayer displays the number of identified issues. (see Figure 27). These layers can be shown/hidden and selected to easily distinguish issues present in the mesh.

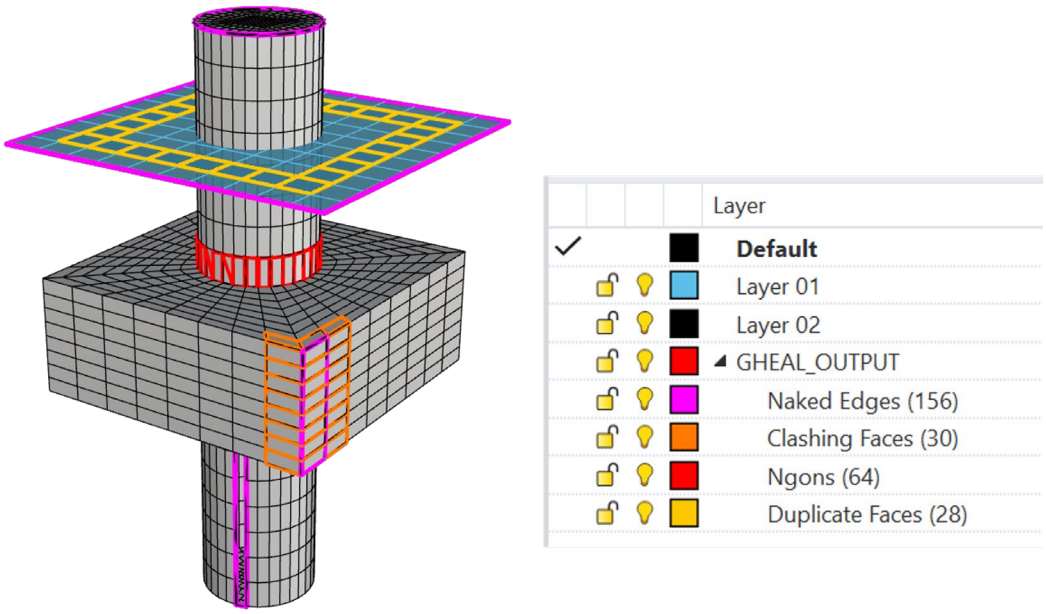


Figure 27: GHeal's Rhino Layer output

Note that surface meshes within a watertight domain can have naked edges, for example, if they are only partially connected to domain boundaries or “float” within the volume. However, the presence of naked edges on domain boundary meshes usually indicates problems that may prevent volume meshing.

CustomHeal

This option calls *Rhino*’s command **_MeshRepair**, which checks meshes and fixes problems manually (one by one). Refer to *Rhino’s* help for the description of the **_MeshRepair** command.

AutomaticHeal

AutomaticHeal mode contains two options: *IssuesToFix* and Advanced*Parameters*. *IssuesToFix* specifies types of issues to work on and *AdvancedParameters* option specifies corresponding function parameters. They are described below.

Note that once this mode is done processing, the “GHEAL_OUTPUT” layer will automatically be created or updated indicating any errors that may remain in the selected meshes.

IssuesToFix

This option specifies types of issues (functions) that **GHeal** can operate on in the *AutomaticHeal* mode. Any combination of the *issues* can be selected for **GHeal** to work on (by default all are selected). Every selected function is executed in the order it appears in the menu.

All and *Clear* options allow to select or deselect all issues in the list.

TriangulateNgons

This function splits all Ngons within the selected meshes into triangular faces. The operation is illustrated in Figure 28.

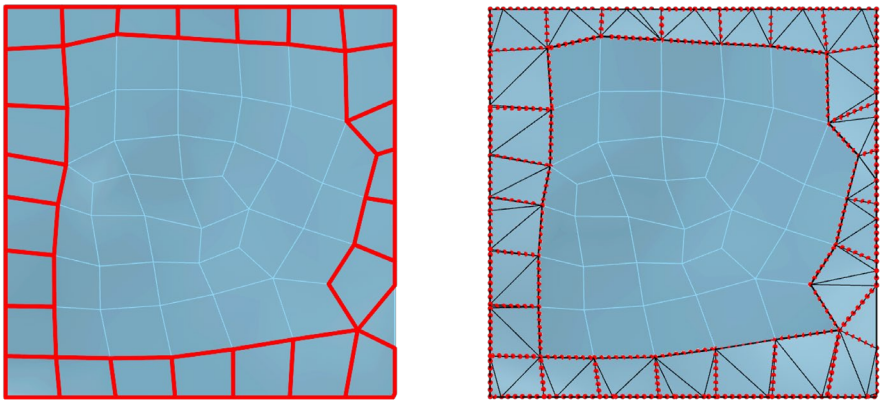


Figure 28: GHeal's function *TriangulateNgons* triangulates all faces with 5 or more sides. Red lines indicate Ngons present (left) and are outlined in by the dotted line for reference on the right.

RemoveDuplicates

This function removes all duplicate mesh faces within selected meshes so they contain a single entry of each face. *Note that this may not result in any visual change to the mesh.*

FixClashingFaces

This function fixes clashing mesh faces within each selected mesh. Any faces within a mesh are considered clashing if they (or any of their components) intersect/overlap or are located near one another within the specified tolerance. Furthermore, any faces (or holes) with edges smaller than the specified tolerance value and their neighbors are considered clashing. By the default, *FixClashingFaces* uses the model Absolute tolerance set in *Units and tolerances* settings (Figure 4). This value can be changed (for the current instance of *Rhino* only) by specifying the tolerance multiplier in *AutomaticHeal* → Advanced*Parameters* → *ToleranceMulitplier.*

FixClashingFaces uses an iterative algorithm based on the mesh intersector **GInt**, and it intersects/splits faces locally for each mesh while trying to reduce the number of clashing faces after each internal run. It is possible that after fixing clashing faces, some will remain as the specified tolerance may be insufficient to resolve all of them. In this case, try first remeshing the meshes with **GSurf** to improve quality, then use **GHeal** → *ShowErrors*, and if necessary, repeat **GHeal** → *FixClashingFaces* (avoid significantly increasing *ToleranceMulitplier*). Note that

FixClashingFaces will not fix non-conformal and overlapping faces from different intersecting meshes. To properly intersect distinct meshes, use the mesh intersector **GInt**.

In Figure 29, the left image shows an example of 4 different types of issues identified as clashing faces: nonconformal connection between faces (A), open Z-fold (B), sliver faces (C), and sliver holes (D). These issues are resolved by **GHeal** and the results are shown in the right image.

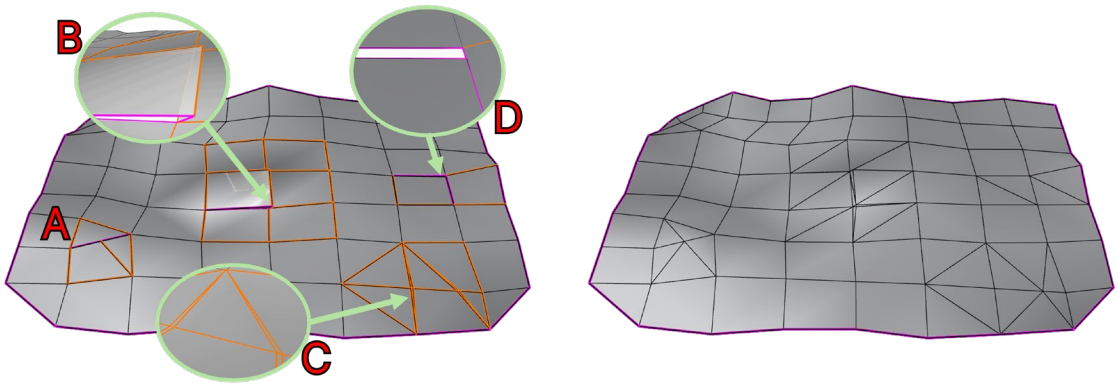


Figure 29: GHeal's function *FixClashingFaces* resolving various problems within a surface mesh.

RemoveNonmanifold

This function removes small mesh pieces which share a non-manifold connection with the rest of the mesh. This function splits meshes at non-manifold connections and uses the parameter *NonmanifoldAreaPercent* to compare the area of each split piece to the area of the original mesh. A mesh piece is removed if it consists of 3 or less faces or if the area of the piece is less than the value calculated using *NonmanifoldAreaPercent*. An illustrative example is shown in Figure 30.

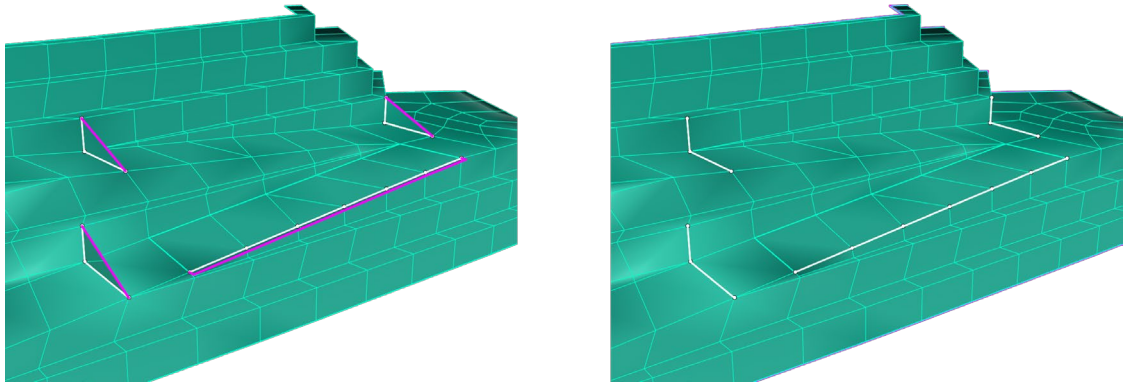


Figure 30: GHeal's function *RemoveNonmanifold* removes non-manifold pieces. Non-manifold edges are shown in white (left) and are removed upon operation completion (right).

FillHoles

This function fills holes within selected meshes (Figure 31). The function is designed for open meshes with possible holes; it treats the largest mesh boundary as the main/external boundary, and it is ignored. All other boundaries are considered to be hole boundaries. The function uses the parameter *MaxHoleAreaPercent* to compare the areas of the holes to the total area of the original mesh. A mesh hole is filled if (i) its area is smaller than the specified percentage of the entire mesh

area and (ii) the area of the hole is smaller than half of the total area of the entire mesh. Remeshing may be needed after filling large numbers of holes. *Note that for best results, FillHoles should be performed on non-disjoint surface meshes.*

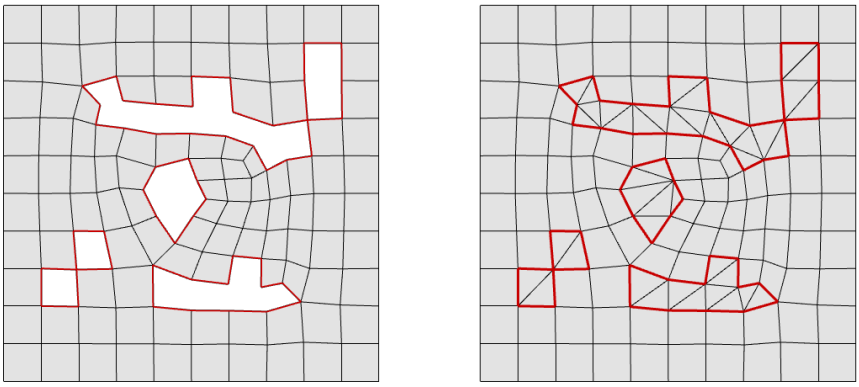


Figure 31: GHeal's function *FillHoles* fixes various holes in the surface mesh. Red lines show naked edges (left) and are provided for reference in the right image.

AlignNormals

This function aligns normals in each component of the mesh and it works on compound meshes consisting of joined/welded sub-meshes. This function is more robust than *Rhino*'s **_UnifyMeshNormals** command when working with compound meshes.

AdvancedParameters

ToleranceMultiplier [*value* 0.001-1000, default = 1]

This parameter defines the multiplier to the model Absolute Tolerance for the calculation of the current working tolerance used in the detection (*ShowErrors*) and fixing of clashing faces: *working tolerance = multiplier x modelAbsoluteTolerance.*

NonmanifoldAreaPercent [*value* 0.001-100%, default = 0.1%]

This parameter defines the limiting value as a percentage of the total mesh area which is used for identification and removal of small non-manifold mesh pieces: any non-manifold mesh piece with area under this value is removed.

MaxHoleAreaPercent [*value* 0.001-100%, default = 10%]

This parameter defines the limiting value as a percentage of the total mesh area which is used for determining if a hole should be filled (if its area is less than the limiting value).

Reset

This parameter resets all advanced parameters to their default values.

ShowWarnings

This option allows for enabling and disabling pop-up warnings messages that may occur during **GHeal** operation. If disabled, warnings will still be output to *Rhino's* command area.

GExtract – Tools for Extraction of Surface Meshes

GExtract is a *Griddle* utility that allows extracting parts of surface meshes based on user-specified criteria. As surface meshes approximating real engineering and geological structures may be rather complex, there is often a need to extract (or separate) parts of the meshes either to remove them or to assign specific properties in preparation for volume meshing. **GExtract** has eight different modes of operation to assist with that.

Multiple mesh selections are allowed for all modes (other than *Single*). This provides the user with the ability to extract from several meshes simultaneously with a single operation.

This command provides an option to place extracted meshes into sublayers of the layers with the initial meshes (see *ExtractionLayers* option). Here, the colors of the extracted meshes may be changed to easily distinguish extracted sub-meshes from the original or remaining mesh.

When the **GExtract** process is completed, all newly extracted mesh pieces will be automatically selected in *Rhino* viewports, so it is easier for the user to do other operations with them (e.g. hide, delete, move, remesh, etc.)

GExtract Options

Single

In this mode **GExtract** extracts single manifold sub-mesh containing a selected face. When using this mode, the user first must select a mesh face from which the rest of the selection will propagate (Figure 32). After that a dialog will be shown (Figure 33, Figure 34) that requests criteria for stopping selection. When the user clicks **OK**, all the selected faces will be separated as a new mesh.

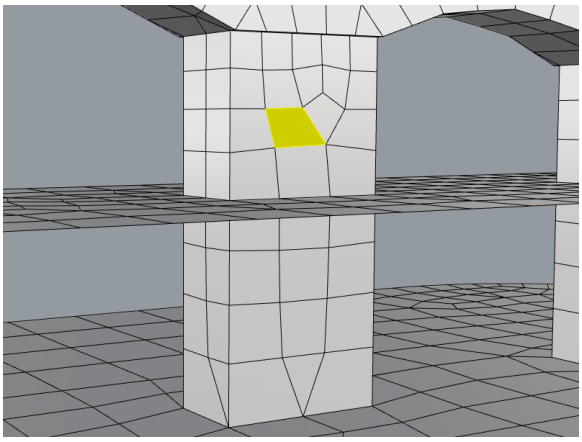


Figure 32: Selecting an initial face for GExtract's *Single* extraction mode.

The **Non-Manifold Extract Faces** dialog sets two criteria for stopping selection:

- By a break angle (*Max break angle*: 0-180°) or
- when the selection hits a non-manifold edge (*Break at non-manifold edges*).

The first option specifies the angle between joined faces (in degrees). Specifying 0° will only select joined faces that are coplanar with the face picked. If the second option is disabled, **GExtract** will propagate its selection of faces at non-manifold edges to those faces having the smallest angle with the currently processed faces.

The second option allows stopping the selection at non-manifold edges. Figure 33 and Figure 34 show the results when breaking at non-manifold edges is enabled or disabled.

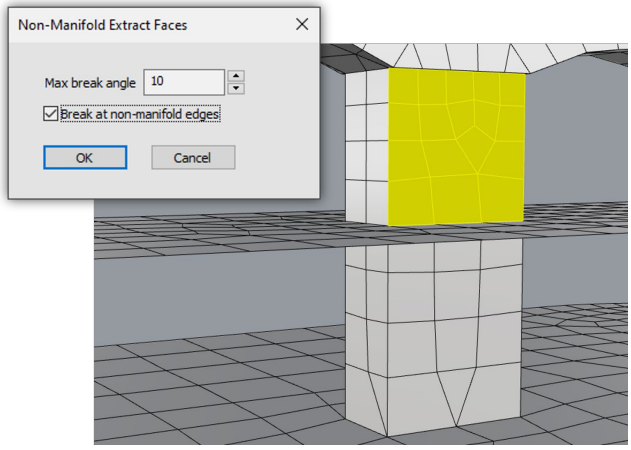


Figure 33: Face selection in the case where *Break at non-manifold edges* is enabled.

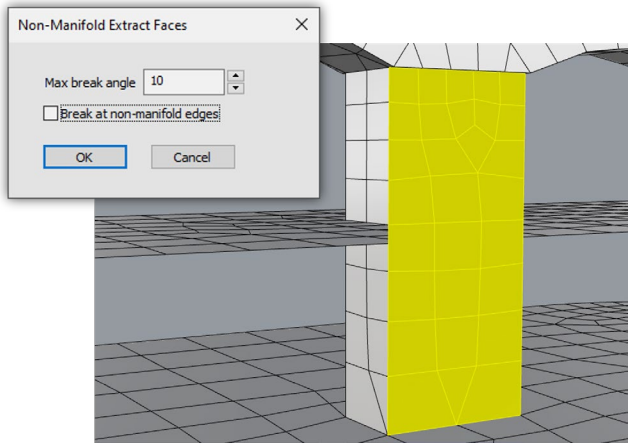


Figure 34: Face selection in the case where *Break at non-manifold edges* is disabled.

Multiple

This mode extracts all possible manifold sub-meshes from selected surface meshes. This option specifies break angle only (0-180°). The extraction of sub-meshes occurs when either angle between neighboring faces exceeds the specified break angle or if a non-manifold edge is encountered. Note that initial faces are picked automatically; the user does not need to pick any faces.

The example in Figure 35 and Figure 36 shows the use of *Multiple* mode with break angle 60°. A single non-manifold conformal mesh (Figure 35) is split into multiple manifold conformal meshes (Figure 36). Afterwards, the extracted meshes are assigned different colors using the **ColorizeObjects** tool.

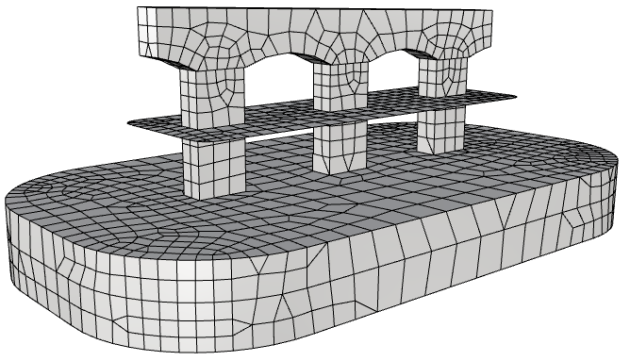


Figure 35: Initial single non-manifold conformal mesh (some internal meshes are not visible).

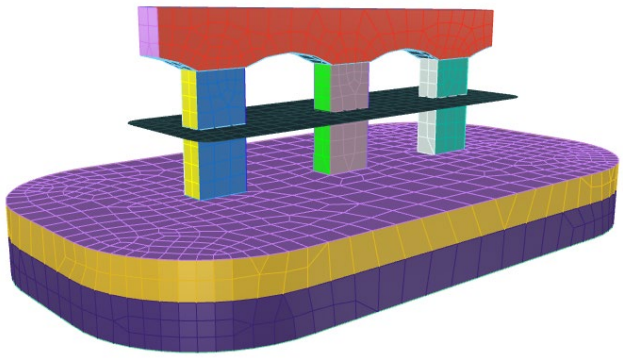


Figure 36: The result of application of GExtract in *Multiple* mode.

When using *Multiple* mode, the angle between faces is calculated using the angle between face normals. Therefore, it is important that all mesh normals are aligned to have proper mesh extractions. If mesh extraction leads to unexpected results, align mesh normals using **GHeal**'s *AlignNormals* function or *Rhino*'s **_UnifyMeshNormals** command (it may not work properly for compound/joined meshes).

Two specific cases of using **GExtract** in the *Multiple* mode deserve a special mention:

- Setting *MaxBreakAngle* = 0° "explodes" a mesh into individual faces. If a set of coplanar faces needs to be extracted, use some small break angle (e.g., 0.01°), but not 0. Note that this behavior differs from the similar option in *Single* mode.
- Setting *MaxBreakAngle* = 180° splits a non-manifold mesh into manifold sub-meshes along the non-manifold connections only (if the mesh has any); the break angle criterion is ignored. Note that this behavior differs from the *NonManifold* mode.

Boundary

This mode extracts only those faces that are attached to naked edges (i.e., mesh boundaries). An example of extraction of boundary faces from two meshes is shown in Figure 37.

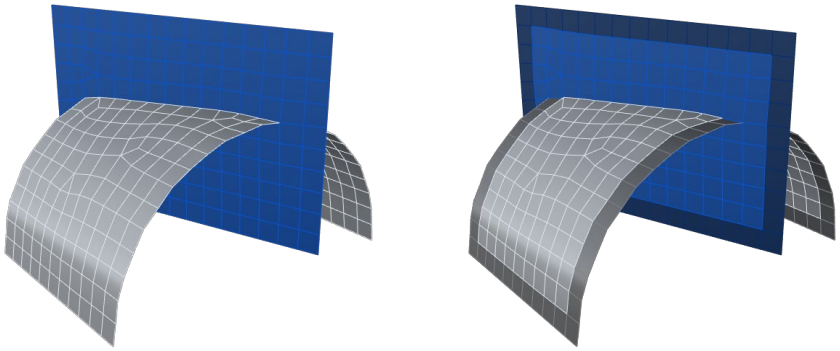


Figure 37: Extraction of boundary faces using GExtract's *Boundary* mode.

NonManifold

This mode extracts only those faces that are attached to non-manifold edges if any are present in the mesh (Figure 38). It may be used to refine mesh around non-manifold edges by using the surface remesher **GSurf** (for example, (i) assign a custom element size to the extracted mesh part and remesh everything or (ii) create hard edges along extracted mesh boundaries using the **_DupBorder** command and then remesh only the extracted mesh part).

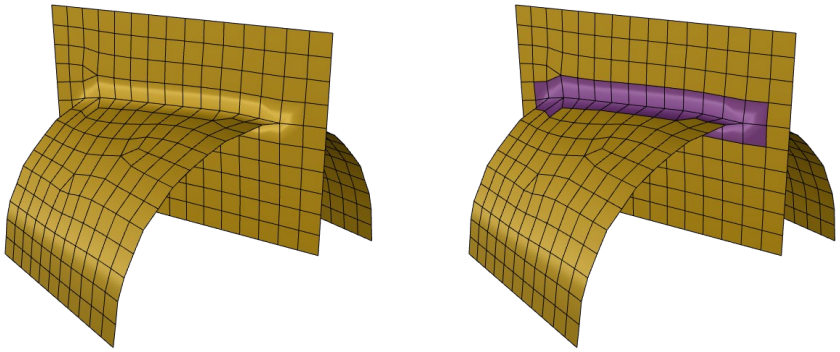


Figure 38: Extraction of faces attached to non-manifold edges using GExtract's *NonManifold* mode.

WithinSolid

This mode extracts pieces of meshes that are located within specified closed volume solids (e.g., a box, cylinder, etc.) It can operate with multiple solids simultaneously, but it is advisable to keep the solids in a separate layer(s), so they can be easily hidden while keeping the extracted meshes selected (for example, to hide the solids, switch *off* the layers containing them). This mode may be used to change element size in extracted meshes by using surface remesher **GSurf** (for example, (i) assign custom element size to extracted mesh parts and then remesh everything or (ii) create hard edges along the extracted mesh boundaries using the **_DupBorder** command and then remesh only the extracted mesh parts).

Figure 39 shows an example of extraction of sub-meshes located within a cylinder (the solid). The cylinder is represented as a BRep object (it is not a mesh). First, the user must select all meshes to process and then select the solid. After **GExtract** finishes the operation, the extracted meshes are highlighted (top-right image). If the cylinder was placed into a separate *Rhino* layer, it can be easily hidden from the view by turning off the corresponding layer. The extracted meshes stay selected (bottom-left image). At this point the user may change the properties of the selected meshes or do other operations with them.

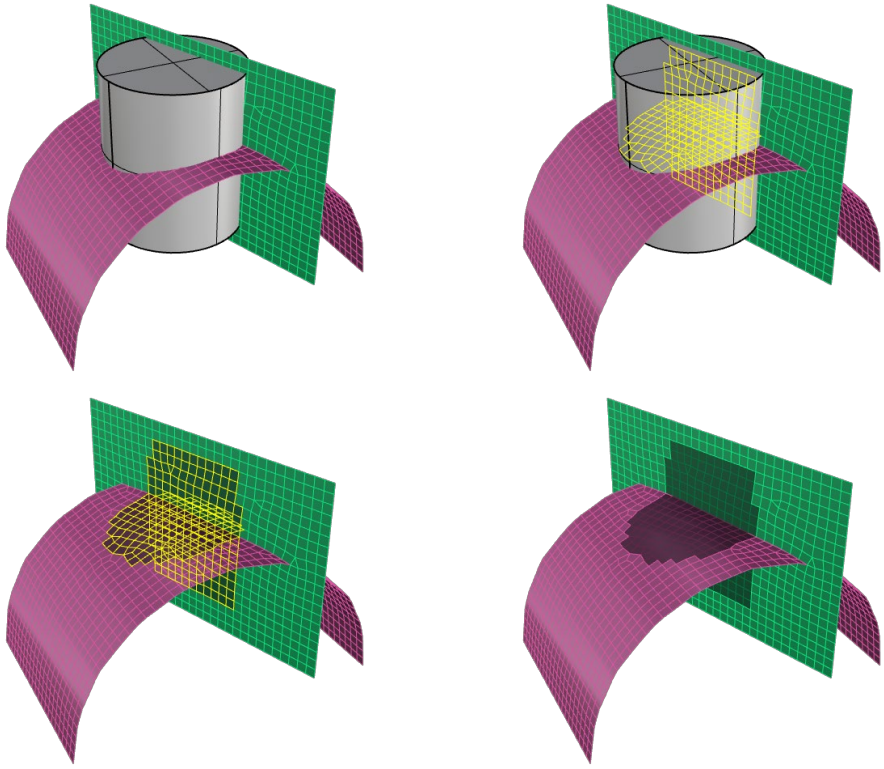


Figure 39: Process of extraction of mesh faces located within a solid by using GExtract's *WithinSolid* mode.

WithinDistance

This mode extracts only those faces of a target mesh which are located within the specified distance from selected source objects (Figure 40). In this mode, multiple objects of different types can be selected and used as sources. This includes points, point clouds, curves, poly-curves, surfaces, poly-surfaces, BRep objects, surface meshes.

The mode has two options:

Distance

This option specifies the maximum distance from target mesh faces to the nearest location on the source objects. In case the source object is a mesh, **GExtract** will automatically extract any target faces that lie along the intersection between the meshes, irrelevant of the distance (as long as it is non-zero). This prevents the possibility of large faces along the intersection going unrecognized.

Check [default = *VerticesAndCentroids*, *FaceCentroidsOnly*]

This option specifies whether distances are calculated from the source object(s) to the vertices and centroids of target mesh faces, or only to the face centroids (shortest distance is used for comparison). This option may affect the number of extracted faces.

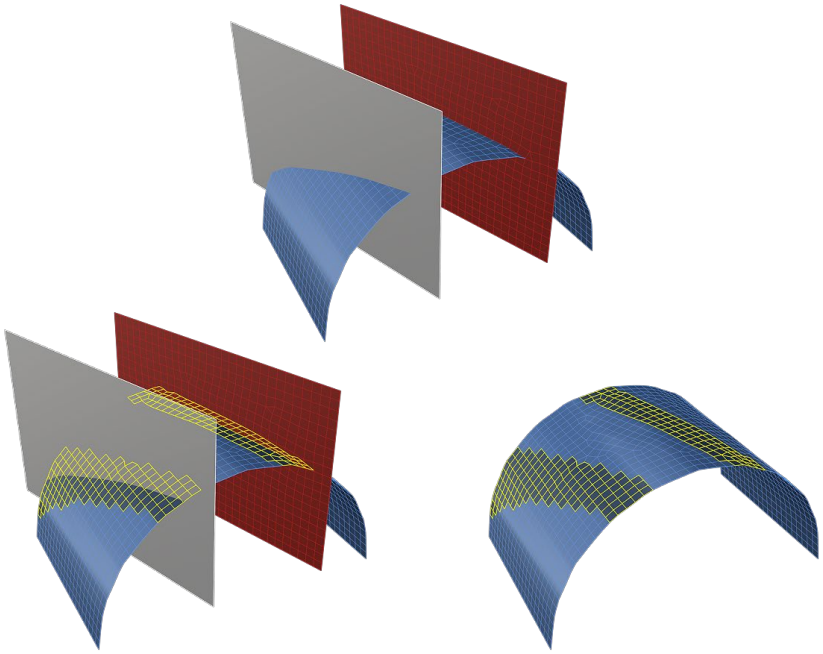


Figure 40: Extraction of mesh faces within distance of another mesh and a surface.

ByArea

This mode extracts all mesh faces from the given meshes that fall within the face area bounds (*MinArea*, *MaxArea*) specified by the user. Figure 41 shows the extraction of faces whose areas fall within the set range.

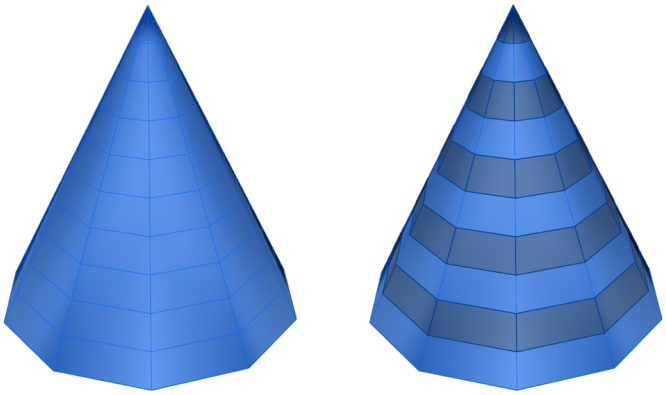


Figure 41: Extraction of mesh faces by face area. For this example, GExtract was run 5 times with increasing *Min* and *Max* area values each time to obtain the results shown on the left.

ByAspectRatio

This mode extracts only those faces that are greater than the minimum aspect ratio (*MinAspectRatio*) specified by the user. When working with quadrilateral faces, the *InclDiagonal* option toggles if diagonals are used in the aspect ratio calculations.

ShowWarnings

This parameter allows for enabling and disabling pop up warning messages that may occur (including Ngons). If disabled, warnings will still be outputted in *Rhino's* command area.

ExtractionLayers

The **GExtract** command provides an option to place extracted meshes into sublayers of the layers with the initial meshes. Each sublayer color is the same as the original layer, however the extracted meshes colors are altered so these meshes can easily be distinguished from the original or remaining meshes. Figure 42 demonstrates an example of sublayer creation when **GExtract** is executed on a mesh originally assigned to "Layer 02". The *ExtractionLayers* option has three parameters:

CreateOrAppend (default)

Specifies that the extracted mesh parts will be placed in the sublayers of the layers with the original meshes. Suffix "_Extracted" is added to the sublayers' names. If sublayers with such names already exist, the extracted objects will be added (appended) to them.

CreateOrReplace

Specifies that the extracted mesh parts will be placed in the sublayers of the layers with the original meshes. Suffix "_Extracted" is added to the sublayers' names. If sublayers with such names already exist, they will be deleted along with all the objects they contain. Then new sublayers will be created, and all extracted objects will be placed into them. *Note: this mode may result in the deletion of objects from sublayers with matching names.*

DoNotCreate

Specifies that no new layers will be created for the extracted mesh parts; these parts will be placed in the layers with the original meshes.

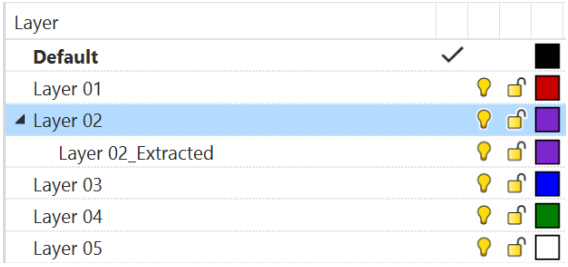


Figure 42: GExtract's "_Extracted" layer output.

GExtend – Tools for Surface Mesh Extension

GExtend is a *Griddle* utility that allows extending a surface mesh by adding new faces to it. New mesh faces are created along a selected part of the mesh boundary or along the whole boundary. After that they can be joined to the original mesh or kept as a separate mesh (the original mesh is not changed). **GExtend** has several options when extending / building new faces: (1) building new faces in the average tangent direction calculated using the orientation of the initial mesh faces at the selected part of the boundary, (2) building new faces along user-specified vector, and (3) building new faces in "free" direction (in this mode user can drag-and-drop the boundary piece on the screen to set the final position), and (4) building new faces in the direction normal to local/closest faces in a base mesh, when extending a target mesh to the base mesh.

One of the most important parameters of **GExtend** is the extension length, which specifies or proportional to edge size for newly built faces (along the extension direction). For good quality results, the extension length should be comparable to the size of faces in the initial mesh at the selected part of the boundary. If the extension length is significantly larger than the typical initial face/edge size, the extended mesh may be of poor quality or even be invalid due to possible overlapping and intersecting faces (when extending along average tangent or normal to a base).

After using the **GExtend** tool, remeshing of the extended surface mesh may be necessary to improve mesh quality and to achieve desired element size.

GExtend has four modes of operation: *ExtendSelectedBoundary*, *ExtendAllBoundaries*, ExtendToMesh, and *FreeExtend*. For the first two modes, the command extends the boundary by a given distance, while the third mode extends a mesh until intersection with another (base) mesh. The last mode allows the user to do "free" extension of the selected boundary by moving it to any location.

When extending a mesh along a part of the mesh boundary, all mesh boundaries are highlighted in purple for easier selection (Figure 43). With the boundaries highlighted, the user selects a desired region as shown in Figure 43. Additional pieces of the boundary can be added (or removed) by holding **Shift** (**Ctrl**) and using the left mouse button. Make sure to select a continuous piece of the boundary as **GExtend** operates on a single continuous segment of the boundary.

GExtend Options

ExtendSelectedBoundary

In this mode, **GExtend** extends the mesh along the selected part of mesh boundary by user specified distance (Figure 44). Three parameters should be specified for the operation to begin:

ExtendLength

This parameter specifies, in model units, the extension length. It is the same as the edge length of newly built faces in the extension direction.

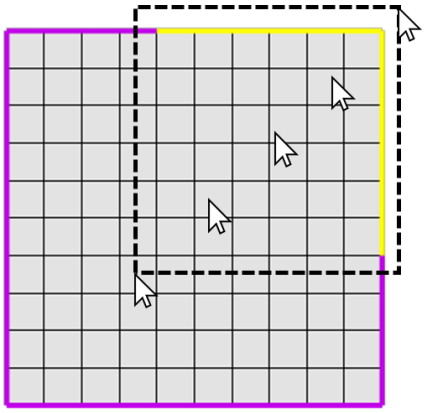


Figure 43: GExtend highlights mesh boundary in purple and allows selecting a piece of the boundary (in yellow).

MeshType [= *Merged* (default), *Separated*]

This parameter specifies if the newly built (extended) mesh patch should be merged with the initial mesh or kept separate.

Direction [= *LocalTangent* (default), *AlongVector*]

This parameter specifies the extension direction along which new faces are constructed. If *LocalTangent* is selected, new faces are constructed along average local tangent vector calculated at the nodes of the selected boundary part. If *AlongVector* option is used, the extension direction can be specified by typing the coordinates of vector's starting and ending points. Alternatively, as the command shows all mesh vertices, the user may select any point as the start and/or the end of the extension vector (in general, any two points can be chosen to specify the extension vector).

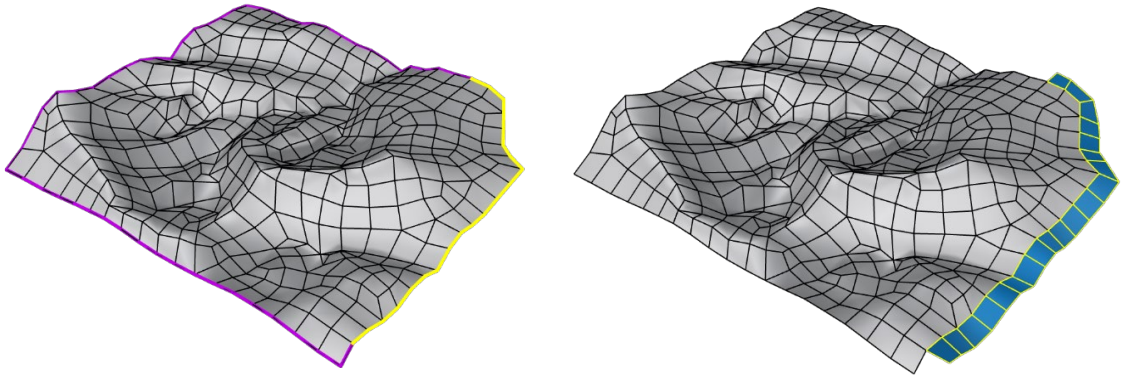


Figure 44: Extension of a mesh along a part of its boundary using *ExtendSelectedBoundary* mode.

ExtendAllBoundaries

This mode expands the initial mesh(es) along all its boundaries by a given distance (Figure 45). Multiple meshes can be selected and boundaries of each mesh will be extended. The options in this mode are the same as in *ExtendSelectedBoundary* mode.

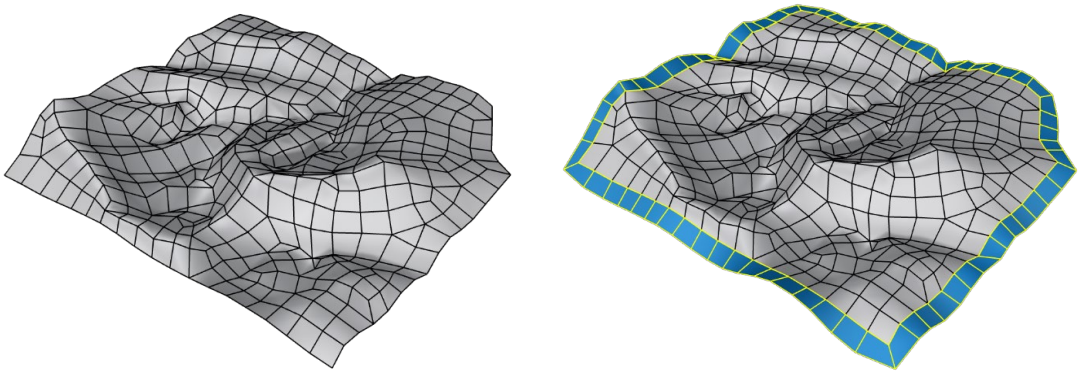


Figure 45: Expansion of a mesh along all its boundary using *ExtendAllBoundary* mode.

ExtendToMesh

In this mode, **GExtend** extends a selected (source) mesh along a part of its boundary until intersection with a target mesh. By default, the intersection of the extended piece with the target mesh is made conformal. This option allows easily closing gaps between meshes and provides conformal intersections whenever possible. The operation of *ExtendToMesh* tool is illustrated in Figure 46.

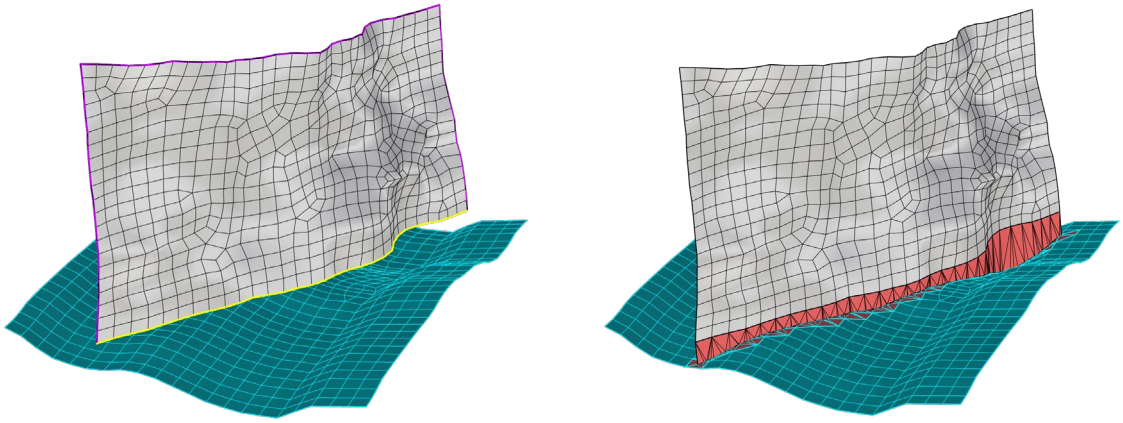


Figure 46: *Left: Source (gray) mesh with highlighted part boundary to be extended. Right: The source mesh is extended in LocalTanget direction and intersected with the target mesh.*

Direction [= *LocalTangent* (default), *AlongVector*, *NormalToTarget*]

This parameter specifies the extension direction along which new faces are constructed. For the description of the first two options, see *ExtendSelectedBoundary* → *Direction*. If *NormalToTarget* setting is selected, mesh extension towards the target mesh is done along the normal projection direction.

TrimExtension [= *Yes* (default), *No*]

This parameter designates if **GExtend** should attempt to automatically intersect the extended (source) mesh and the target mesh and then trim the extended mesh to remove any sliver faces that may appear past the target mesh. The intersection logic is similar to the one used in **GInt**, and

intersection tolerance is automatically calculated between *MinTolerance* and *MaxTolerance* values specified by the user. The logic progressively increases the value of tolerance until a single continuous polyline can be created along the intersection. In cases when multiple polylines are created or tolerance value used exceeds *MaxTolerance*, the intersector will stop and will output an error message. In such cases, the user should proceed with mesh intersection and trimming manually (e.g., by using **GInt** and **MeshSplit** commands).

MinTolerance

This is the minimum (starting) tolerance to be used in automatic mesh intersection logic.

MaxTolerance

This is the maximum tolerance to be used in automatic mesh intersection logic. If immediate tolerance value exceeds *MaxTolerance*, the mesh intersector will abort and the source mesh will be extended without conformal intersection with the target mesh.

MeshType [= *Merged* (default), *Separated*]

This option is only available if *TrimExtension* is set to *No*. It specifies if the extended mesh piece should be merged with the source mesh or kept separate.

FreeExtend

This mode extends a mesh at selected part of the boundary to a custom position by moving (dragging and dropping) the selected part (Figure 47). The movement starts at a node in the middle of the selected part and stops at a point where the user "drops" it. The extended part of the mesh is merged with the initial mesh.

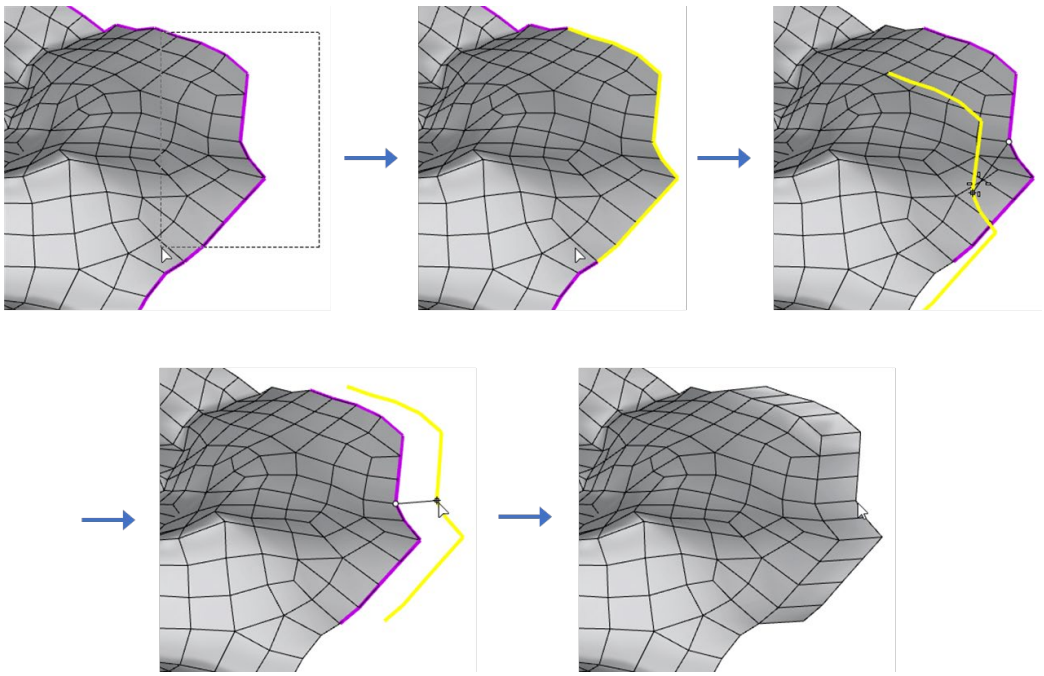


Figure 47: Process of selecting and dragging a part of a mesh boundary using *FreeExtend* mode.

GExtrude – Tools for Surface Mesh Extrusion

GExtrude is a *Griddle* utility that allows extruding a surface mesh onto a bounding surface provided by the user. The bounding surface must be a BRep or NURBS type object (for example, a surface or polysurface, but not a mesh). **GExtrude** can be used to quickly create a meshed modeling domain from a given surface mesh, for example, a topographic mesh. Depending on the input mesh and extrusion parameters, the generated domain may or may not be watertight. **GExtrude** produces the best results when the bounding surface is planar or nearly planar. It is not designed for exact mapping of the initial surface mesh onto a bounding surface, and it does not guarantee that mesh projection will exactly conform to the bounding surface if the surface is non-planar.
The extrusion is performed along the mesh boundaries, following these steps.

1. Project each node on the boundary of the initial surface mesh onto the bounding surface (using the shortest distance).
2. Connect the projected nodes to each other and to the corresponding original nodes to form side surface meshes.
3. Construct a mesh within the projected boundary on the bounding surface.
4. When extruding internal boundaries, intersect the side pieces with the bounding mesh to ensure a watertight seal.
5. Remesh the sides and the bounding mesh pieces to the specified element sizes.

GExtrude can only operate on a single mesh at a time, and the mesh cannot contain disjoint pieces, non-conformal intersections or clashing faces, or self-intersecting boundaries. **GExtrude** automatically checks for these errors.

GExtrude options

ExtrdMeshType [= *Tri* (default), *QuadDom*]

This parameter specifies the type of surface mesh used in the extruded mesh: *Tri* produces an all-triangle surface mesh, *QuadDom* produces a quad-dominant surface mesh (contains a mix of triangles and quadrilaterals). This option will not alter the initial surface mesh.

Boundaries

This parameter specifies whether to extrude the external (outermost) boundary or all boundaries (external and all valid internal boundaries, if any are present). It also limits if the bounding surface can be non-planar.

External (default)

When extruding the outermost boundary, the initial mesh must have a valid non-intersecting boundary. If there are no other boundaries present in the mesh, the extrusion will result in a watertight domain (see Figure 48). If the mesh contains internal boundaries, they will be ignored, resulting in an open mesh (see Figure 49). When **GExtrude** is set to this mode, the surface mesh can be extruded to planar or non-planar surfaces.

All

When extruding all boundaries, the initial mesh may contain holes but must not have any non-conformal edges or faces (check and fix issues with **_GHeal** command, if needed). If the initial mesh meets these criteria, the extrusion will produce a watertight domain (see Figure 50). When **GExtrude** is set to this mode, the surface mesh can only be extruded to a planar surface.

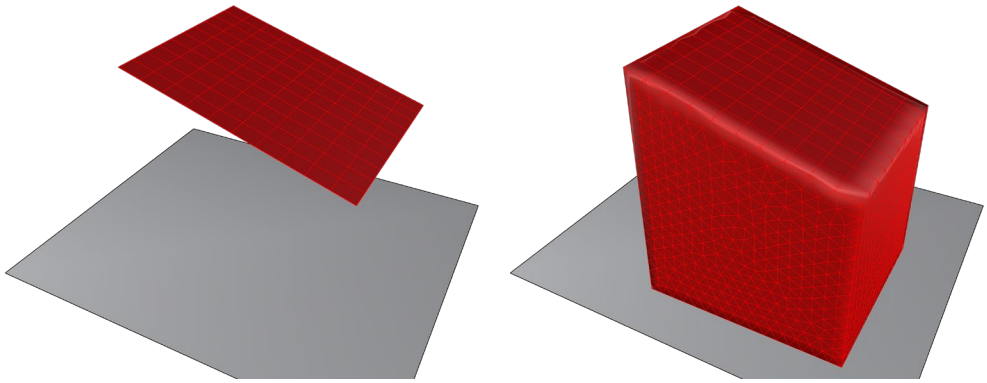


Figure 48: The initial mesh with a bounding surface (left) and the result of mesh extrusion.

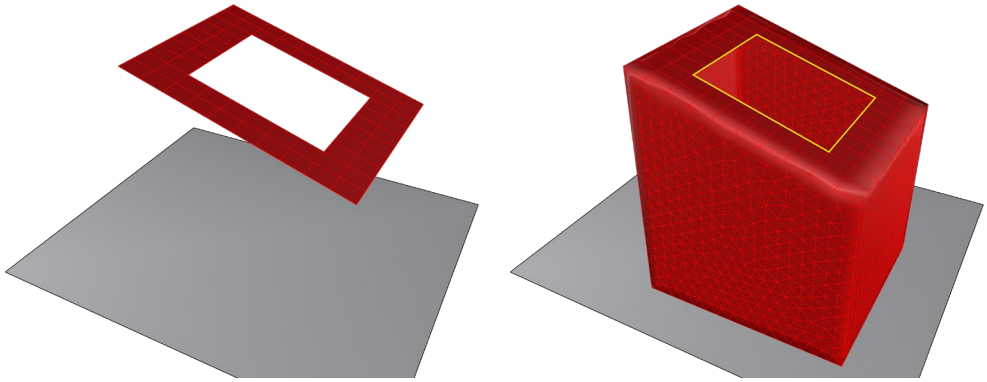


Figure 49: The result of the extrusion of *External* boundaries only when the initial mesh contains a hole. GExtrude produces an open (non-watertight) mesh domain.

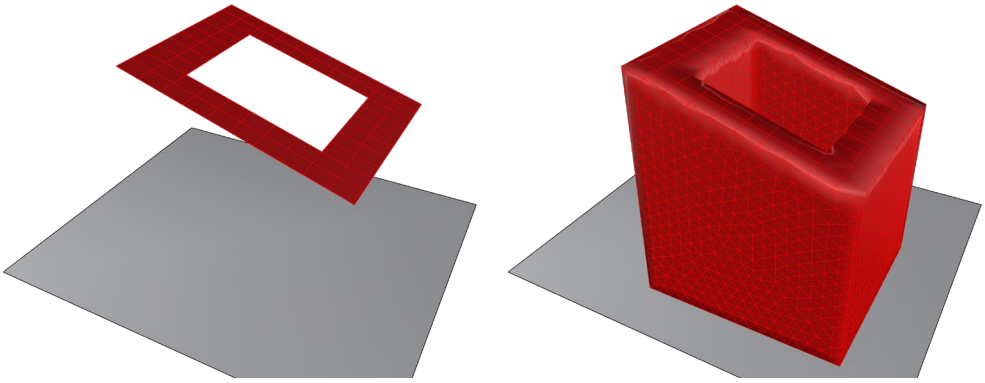


Figure 50: The result of the extrusion of *All* boundaries to form a watertight domain.

MeshOutput [= *Merged* (default), *Split*]

This parameter specifies if the extruded mesh will be merged with the initial mesh or kept separate. Figure 51 demonstrates an extrusion with this parameter set to *Split*. Note: The **ColorizeObjects** tool is used here to help visualize the separated mesh surfaces.

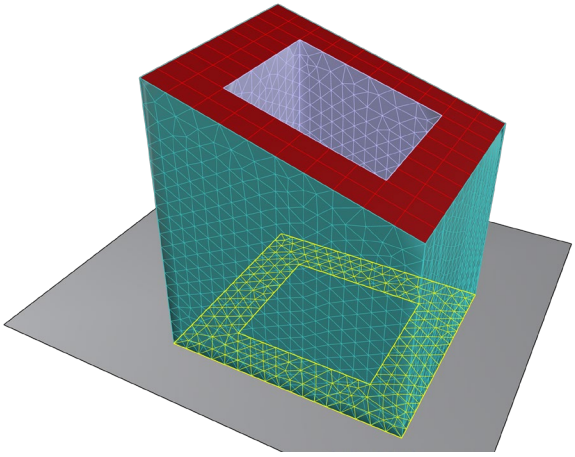


Figure 51: The result of GExtrude using the *Split* option.

MinEdgeLength, MaxEdgeLength

These parameters control the resulting minimum and maximum edge size in the final surface mesh. To get uniform sizes, minimum and maximum edge size can be set to the same value. Edge size is specified in model units. It is important to set non-zero minimum edge length to get a good quality output mesh.

ShowWarnings

This parameter enables and disables pop up warning messages that may occur during the extrusion. This includes the detection of disjoint mesh pieces and invalid/open boundaries. If disabled, warnings will still be output to *Rhino's* command area only.

GCollapse – Tools for Overlapping Surface Meshes

GCollapse is a *Griddle* utility that resolves overlapping or nearly overlapping meshes by collapsing target mesh faces onto one or more base meshes. Target mesh faces found within a specified distance from any base meshes are first removed and newly created target mesh boundaries are extended to the closest base mesh faces. If option *IntersectAndTrim* is specified, **GCollapse** will automatically intersect the extended boundaries with the base meshes to provide a conformal connection. Figure 52 shows a simple example of this utility.

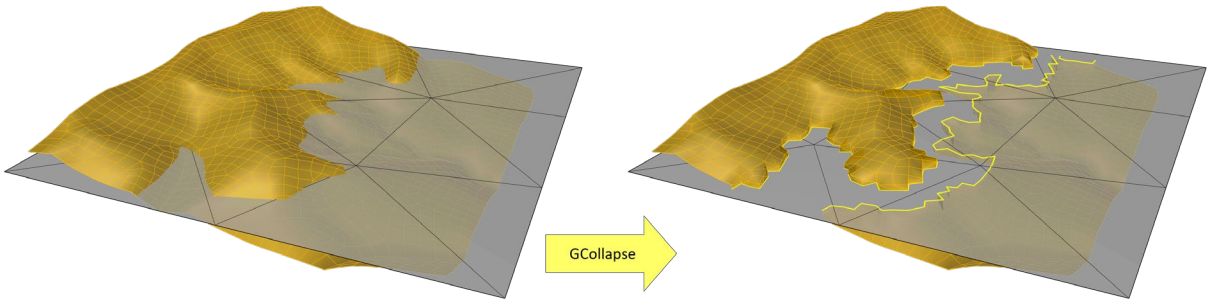


Figure 52: Collapsing a surface mesh onto a planar mesh.

Figure 53 shows **GCollapse** performed on an open pit mine model with three partially overlapping mining surfaces (mining stages). After selecting a base mesh (in this case, the deepest red mesh), **GCollapse** allows collapsing / separating the remaining meshes in order to create well defined volumes between them, which is necessary for proper volume meshing. This example requires only two operations with **GCollapse**: (1) collapse the green (target) mesh onto the red (base) mesh and remesh the results to the desired parameters, and (2) collapse the blue (target) mesh onto the red and green meshes (bases) followed by another remeshing operation to improve surface mesh quality. This process can be repeated on as many meshes as necessary.

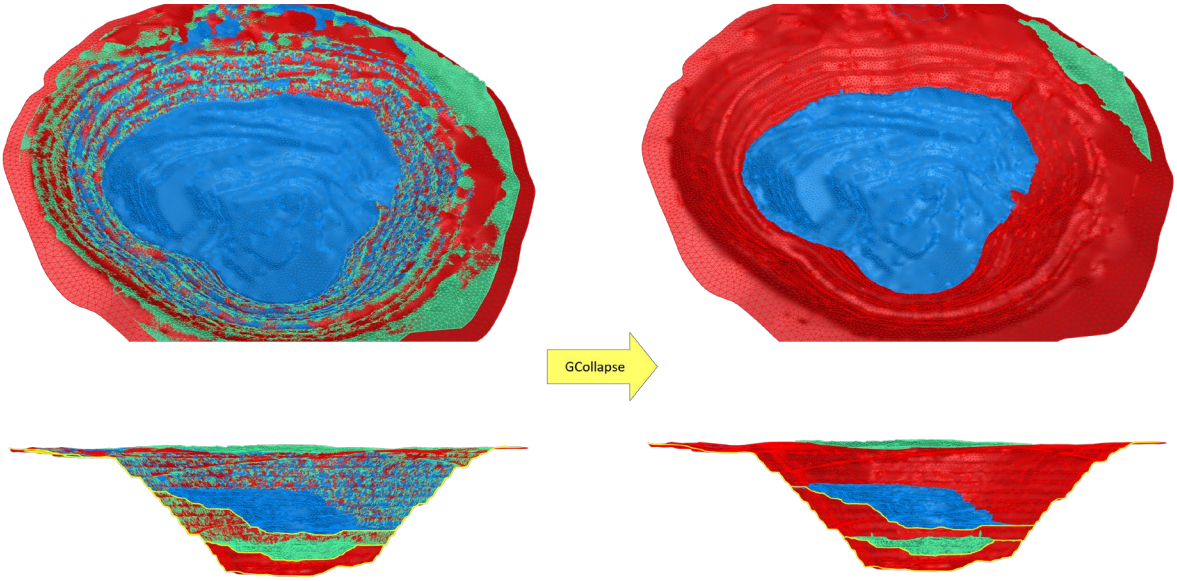


Figure 53: Resolving overlapping pit mining surfaces using GCollapse.

After using the **GCollapse** command, remeshing may be necessary to improve the mesh quality or to achieve a desired element size.

GCollapse requires the base meshes to contain triangular faces only (as the operation can only be accurate if the base mesh faces are planar). If any base mesh contains quadrilateral faces and/or either base or target meshes contain Ngons, **GCollapse** will show a warning and will split those faces and Ngons into a set of triangular faces.

GCollapse options

Distance [*value* > 0]

This parameter specifies the maximum distance from any base mesh within which target mesh faces are collapsed. If there is a true intersection between the target and any base mashes, **GCollapse** will automatically collapse any intersecting target faces. This prevents the possibility of large faces along the intersection going unrecognized.

IntersectAndTrim [= *Yes* (default), *No*]

This parameter toggles the ability to automatically intersect and trim the extended target mesh faces with the base meshes, resulting in all meshes being conformal (e.g., as in Figure 54). The intersection algorithm is similar to the one used by **GInt**. The intersection tolerance is automatically searched within the range specified by the user in *AdvancedParameters* until a value that provides the best intersection results is found. This value produces the smallest number of continuous (potentially multiply connected) intersection polylines when compared to all other tolerance values within the specified range. In rare cases no full intersection can be found, especially if the tolerance range is very restrictive (try increasing the tolerances range).

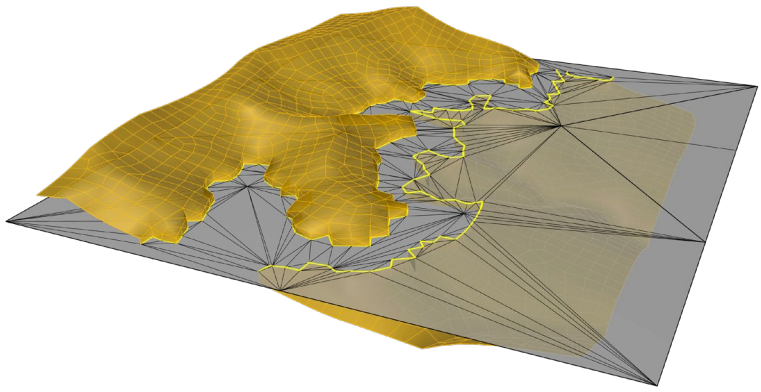


Figure 54: Intersecting the collapsed target mesh to the base mesh.

SplitAtIntersection [= *Yes*, *No* (default)]

This parameter allows splitting the target mesh into two separate meshes if there is a full intersection with the base mesh (see Figure 55). It requires a valid *DirectionVector* to be set. *Note: this functionality requires using a single base mesh only. If this option is turned on, the user will not be able to select multiple bases.*

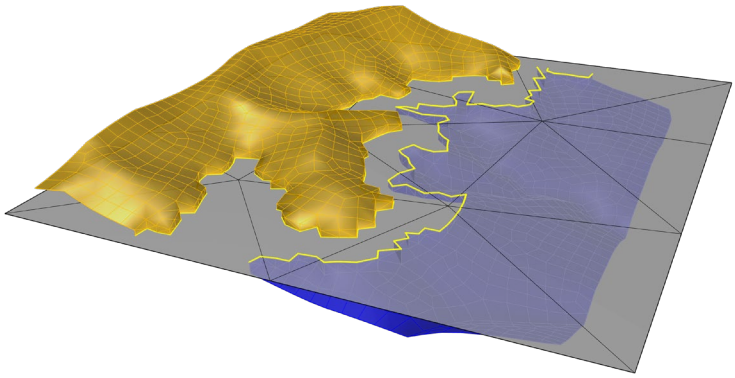


Figure 55: The target mesh split using *SplitAtIntersection*.

DirectionVector [= *Downwards* (default), *Upwards*, *Custom*]

This parameter is used to properly define the sides of collapsed and split target mesh pieces in relation to the base mesh (i.e., which target mesh pieces will be on one side of the base mesh, and which are on the other). As base meshes may have a random orientation in space and complex shape, it is up to the user to define the direction of splitting. This option is only available if *SplitAtIntersection* is turned on.

ExcessLayer

GCollapse provides an option to place the split target mesh pieces into sublayers of the layer with the initial meshes. This sublayer will preserve all parameters of the input mesh, however the excess mesh color will be altered. This is done to easily distinguish the split mesh pieces.

The sublayers use the name of the original layer and adds the suffix "_Excess". This layer may contain a part of the target mesh that may be considered unnecessary or redundant (i.e., an extraction of material). The *Excess* layer is determined based on the *DirectionVector*, where the piece on the origin side of the vector (relative to the base) is deemed to be excess. *Note: this functionality requires a full intersection between the base and the target mesh.*

Figure 56 demonstrates the *Rhino* layer output if this option is enabled. This option is only available if *SplitAtIntersection* is turned on.

CreateOrAppend (default)

Specifies that the excess mesh pieces will be placed in the sublayer of the original mesh. If a sublayer with the name already exists, the excess mesh will be added (appended) to them.

CreateOrReplace

Specifies that the excess mesh pieces will be placed in the sublayer of the original mesh. If a sublayer with the name already exists, it will be deleted along with all the objects it contains. The new sublayer will be created, and the excess mesh will be placed into it. *Note: this mode may result in the deletion of objects from sublayers with matching names.*

DoNotCreate

Specifies that no new layers will be created for the excess mesh pieces – they will be placed in the layers with the original mesh.

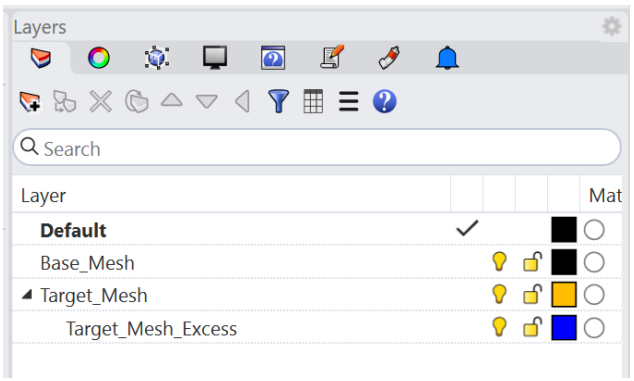


Figure 56: GCollapse's "_Excess" layer output.

ShowWarnings

This parameter allows for enabling and disabling pop up warning messages that may occur during the operation. This includes the detection of Ngons or quadrilateral faces, deletion of the full target mesh, improper extension/intersections/splits, and checks for an invalid *DirectionVector*. If disabled, warnings will be outputted in the *Rhino* command area only.

AdvancedParameters

Check [= *VerticesAndCentroids* (default), *FaceCentroidsOnly*]

This parameter specifies whether distances are calculated from the base meshes to the vertices and centroids of target mesh faces, or only to the face centroids (shortest distance is used for comparison). This option may affect the number of collapsed faces.

MinTolerance [*value* ≥ 0, default=0.001]

This parameter specifies the minimum (starting) value to be used for intersection tolerance search in the automatic mesh intersection logic used when *IntersectAndTrim* is turned on.

MaxTolerance [*value* ≥ 0, default=0.1]

This parameter specifies the upper limit for intersection tolerance search in automatic mesh intersection logic used when *IntersectAndTrim* is turned on.

MinAreaPercent [*value* ≥ 0, default=1%]

This parameter allows for additional cleanup of the target mesh after removing nearly overlapping faces (within the specified distance). The face removal may cause some parts of the target mesh to become completely disjoint from the remainder of the collapsed mesh. The logic compares the areas of the disjoint mesh pieces to the total (original) area of the target mesh and deletes the pieces which areas are below the specified limit. If it is important to keep all pieces unaltered, set the area to 0 percent.

Reset

This parameter resets all advanced parameters to their default values.

Joining Non-Manifold Surfaces

NonManifoldMerge tool calls the corresponding *Rhino* command to join several manifold or non-manifold surfaces and polysurfaces into a single non-manifold polysurface[9].

Starting with surfaces or polysurfaces that fully connect along one or more edges (as in Figure 57), a single polysurface can be created with the **NonManifoldMerge** tool. This polysurface can later be meshed using **_Mesh** command resulting in a fully conformal non-manifold mesh with all parts of the mesh being properly attached, as opposed to the case of meshing multiple surfaces separately. Now the mesh intersection step (as in Figure 15) may be skipped and the user may continue with remeshing the initial mesh (with **GSurf**) and then creating a volume mesh (with **GVol**).

This procedure avoids problems related to using improper tolerance during the mesh intersection step (**GInt**) and thus obtains accurate results faster. However, the drawback of this approach is that the output surface mesh is a single object (it can be split into parts using **GExtract**). A simple example of such a procedure is shown in Figure 57 and a detailed example is provided in *Griddle 3.0 Tutorial Examples*.

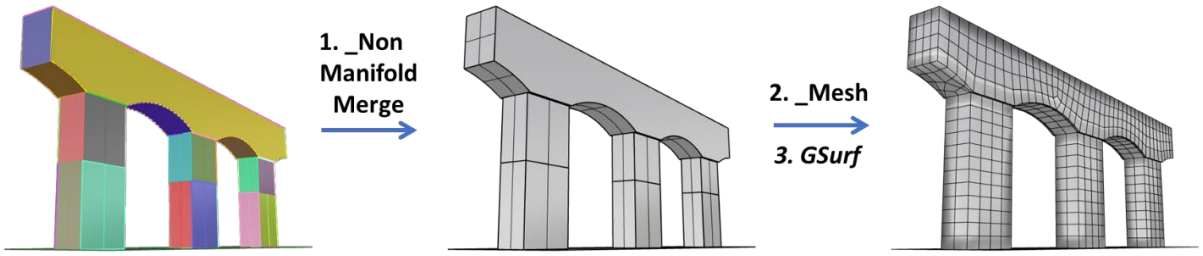


Figure 57: Three step process to create a single mesh out of multiple surfaces/polysurfaces.

Note that in certain cases *Rhino*'s **_Mesh** command may not generate initially conformal mesh from a single complex polysurface created with **NonManifoldMerge** command. This may happen in cases when the **NonManifoldMerge** is applied to (poly-)surfaces that cut through each other rather than connecting along the edges (as in Figure 57).

[9] In this context, manifold means that a surface edge is shared by at most two surfaces. In contrast, consider two intersecting planar surfaces represented by a single polysurface object. The edge along the intersection line is connected to 4 sub-surfaces. Thus, the whole polysurface object is non-manifold.

Colorize Objects

In general, all newly created objects in *Rhino* get the color of the active *Rhino* layer. To differentiate objects from each other by color, select several objects in *Rhino* and click on the **ColorizeObjects** tool to assign different (random) colors to each selected object (Figure 58). To revert to a colorization **by layer**, select objects, open object *Properties* panel (or press **F3**), and in the item labelled *Display Color*, select *Display Color → By Layer*.

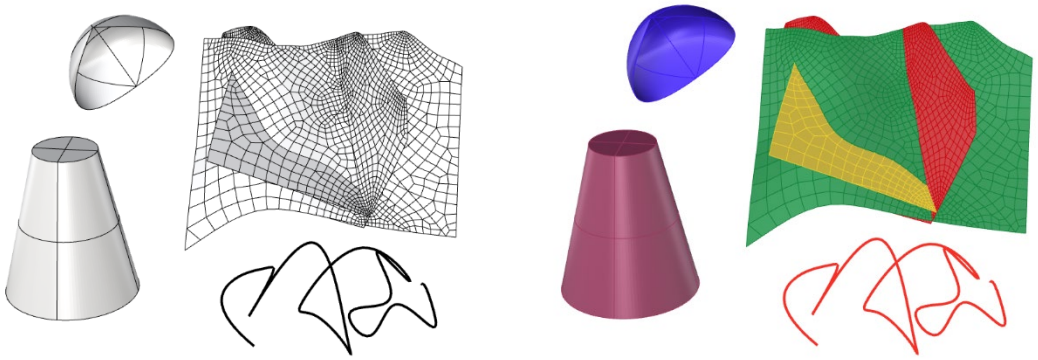


Figure 58: Various objects before and after ColorizeObjects is applied.

Importing *Griddle* Volume Meshes into 3rd party software

Griddle's volume meshers **BlockRanger** and **GVol** output volume meshes into a text or binary file in the format corresponding to the selection of numerical simulation software, such as *FLAC3D*, *3DEC*, *ABAQUS*, etc. The list below provides general tips on importing mesh files into corresponding software (note that workflow in the users' version of the software may differ from what is present below):

- *FLAC3D*: Navigate to menu *File* → *Grid* → *Import from FLAC3D…* and open the mesh file (`*.f3grid`) or use the "`zone import`" command.
- *3DEC* 7.0 or newer: Navigate to menu *File* → *Grid* → *Import from 3DEC…*
  or File → *Grid* → *Import from Griddle…* and open the mesh file (`*.3dgrid`)
  or use the "`block import`" command.
- *3DEC* 5.2: Navigate to menu *File* → *Open Item…* and select *Data File: Call*. Then locate and open the mesh file (`*.3ddat`) containing commands to generate the grid.
- *ABAQUS*: Workflow is provided for *Abaqus/CAE*. Navigate to menu *File* → *Import* → *Model* → *Abaqus Input File (*.inp,…)* and open the mesh file (`*.inp`)
- *ANSYS*: Workflow is provided for *ANSYS Workbench*. Importing depends on how the volume mesh is generated:
  - For **GVol** generated meshes, navigate to menu *File* → *Import* and select the mesh file (`*.cdb`).
    Within the *Project Schematic* window, double click on *Setup* to open the *External Model* window. Within the *Properties of File* pane, uncheck the checkbox "*Check Valid Blocked CDB File*".
    Navigate back to the *Project Schematic* window and right-click on *Setup* and select *Update*, which will import the mesh.
  - For a **BlockRanger** generated mesh, first run script from *ANSYS Workbench* on the output file by navigating to menu *File* → *Scripting* → *Run Script File*. Within the `Scripts` folder, select the `ConvertAnsysFileToCdb.py` script and open the mesh file (`*.cdb`) to convert it from the old-style format to a blocked CDB file.
    Once the file is converted, navigate to menu *File* → *Import* to import the converted CDB file. Within the *Project Schematic* window, right-click on *Setup* then *Update*, which will import the mesh.
- *LS_DYNA*: Workflow is provided for *LS-PrePost*. Navigate to menu *File* → *Import* → *LS-DYNA Keyword File*, select *All Files* in the file filter, and open the mesh file (`*.lsdyna970`).
- *NASTRAN*: Workflow is provided for *MSC-Patran*. Navigate to menu *File* → *Import* → *Source: MSC.Nastran Input* and open the mesh file (`*.bdf`).
- *VTK*: use *File* → *Open* menu and select a vtk file with generated volume mesh.

References

Itasca Consulting Group Inc. (2023) *FLAC3D — Fast Lagrangian Analysis of Continua in Three Dimensions*, Ver. 9.0. Minneapolis: Itasca. https://docs.itascacg.com/itasca940/flac3d/docproject/source/flac3dhome.html

Itasca Consulting Group Inc. (2024) *3DEC — Three-Dimensional Distinct Element Code*, Ver. 9.0. Minneapolis: Itasca. https://docs.itascacg.com/itasca940/3dec/docproject/source/3dechome.html

Itasca Software Forum: https://forum.itascainternational.com/

Griddle 2.0 User Manual: https://www.itascacg.com/software/griddle-manual-tutorials

The Rhino window and interface: https://docs.mcneel.com/rhino/8/help/en-us/user_interface/rhino_window.htm

Understanding Rhino Tolerances: https://wiki.mcneel.com/Rhino/faqtolerances