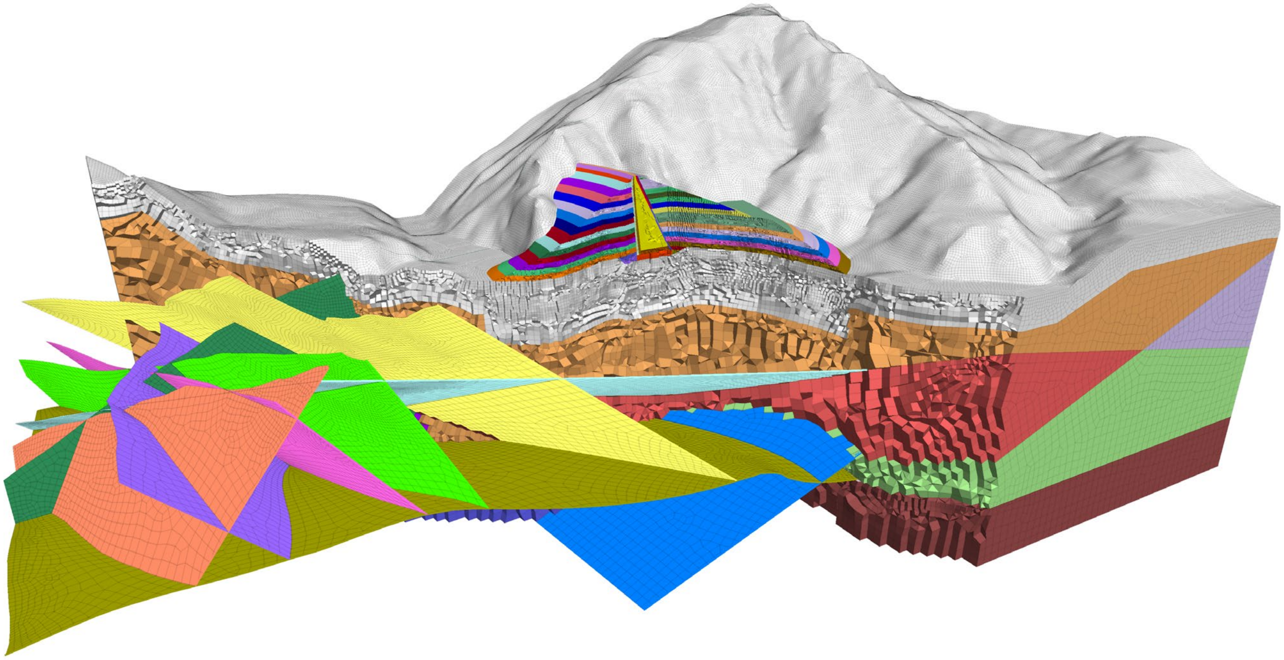
Griddle™

Advanced Mesh Generation
for Engineers and Scientists

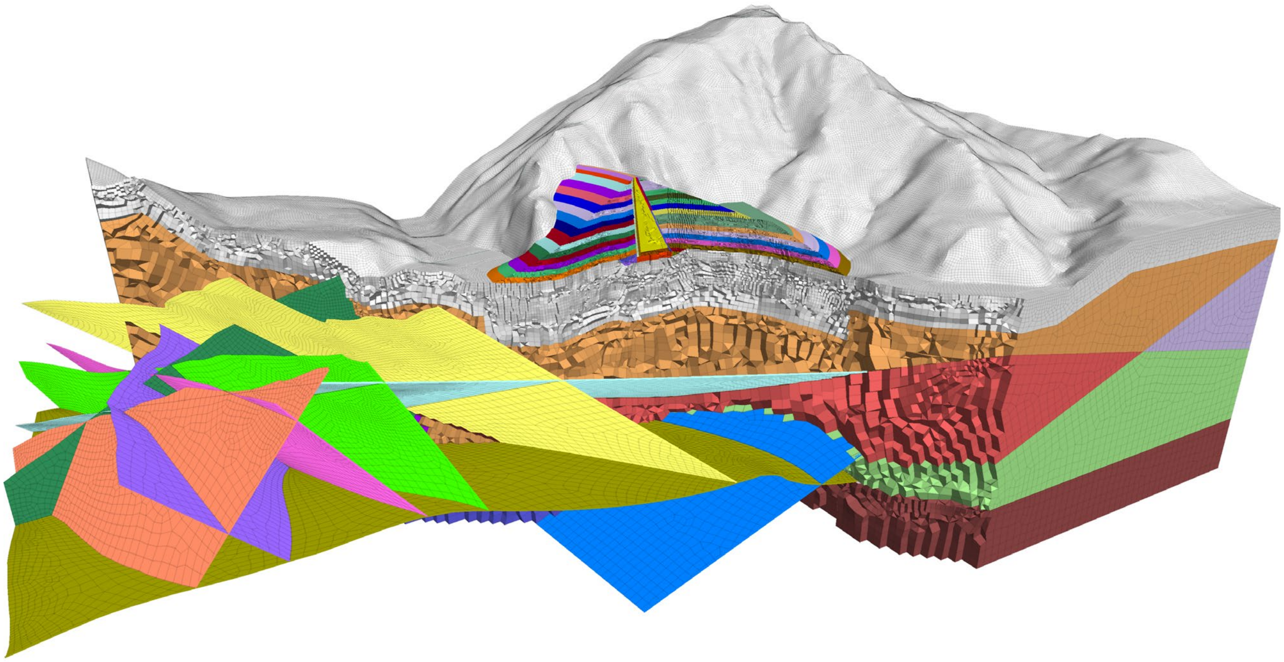
Griddle 3.0
User Manual & Tutorial Examples



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:

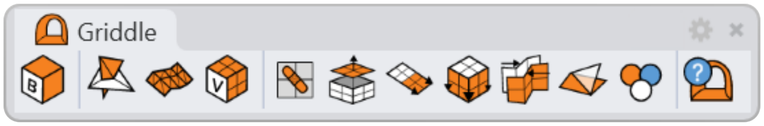
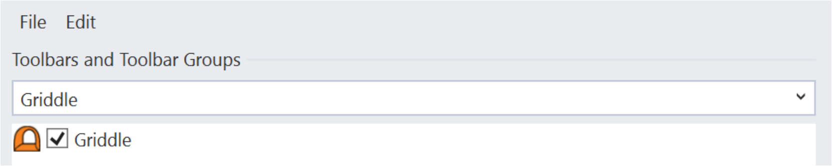


Figure 1: The *Griddle 3.0* toolbar.

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.



Toolbars and Toolbar Groups

Griddle

☑ Griddle

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
B	**BlockRanger**	Structured hexahedral volume mesher operating on solids
	GInt	Surface mesh intersector
	GSurf	Unstructured surface mesh remesher
V	**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
B	**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
V	**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

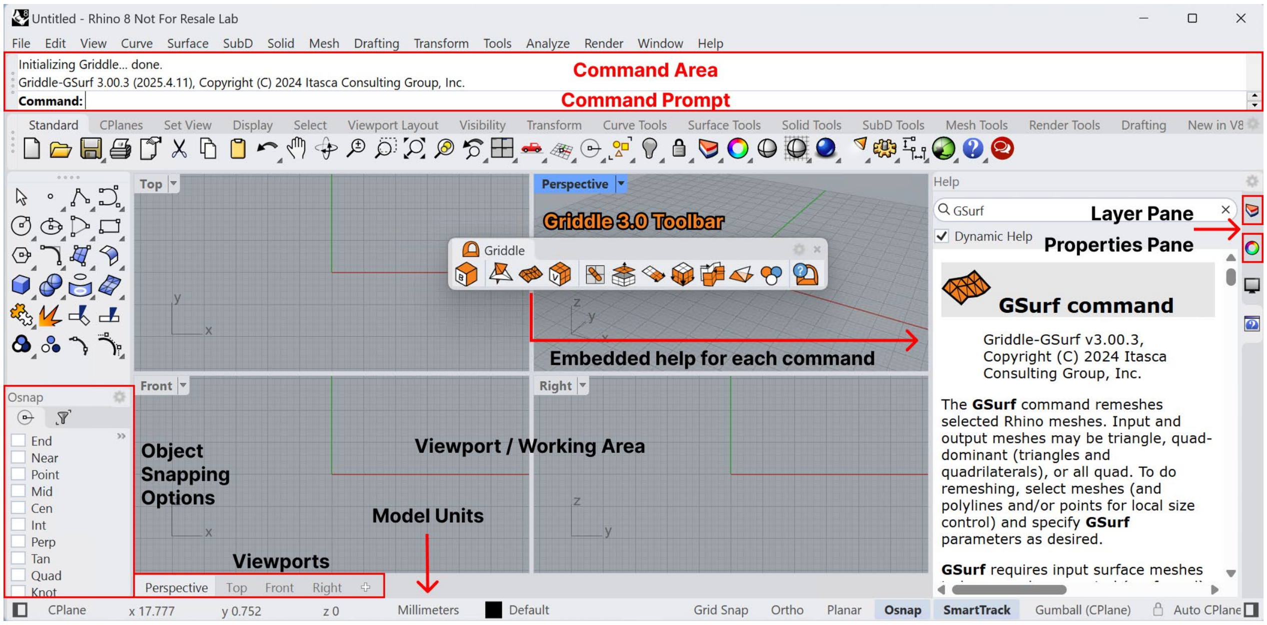
Brief Introduction to the *Rhino 8* Workspace

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 the icon from the *Griddle* toolbar issues the command **_GSurf**.



```
Initializing Griddle... done.
Griddle-GSurf 3.00.3 (2025.4.11), Copyright (C) 2024 Itasca Consulting Group, Inc.
Command:
```

GSurf command

Griddle-GSurf v3.00.3, Copyright (C) 2024 Itasca Consulting Group, Inc.

The **GSurf** command remeshes selected Rhino meshes. Input and output meshes may be triangle, quad-dominant (triangles and quadrilaterals), or all quad. To do remeshing, select meshes (and polylines and/or points for local size control) and specify **GSurf** parameters as desired.

GSurf requires input surface meshes

Untitled - Rhino 8 Not For Resale Lab
File Edit View Curve Surface SubD Solid Mesh Drafting Transform Tools Analyze Render Window Help
Command Area
Command Prompt
Standard CPlanes Set View Display Select Viewport Layout Visibility Transform Curve Tools Surface Tools Solid Tools SubD Tools Mesh Tools Render Tools Drafting New in V8
Top
Perspective
Griddle 3.0 Toolbar
Griddle
Embedded help for each command
Help
GSurf
☑ Dynamic Help
Layer Pane
Properties Pane
Front
Right
Object Snapping Options
Viewport / Working Area
Model Units
Viewports
Osnap
□ End
□ Near
□ Point
□ Mid
□ Cen
□ Int
□ Perp
□ Tan
□ Quad
□ Knot
Perspective Top Front Right
CPlane x 17.777 y 0.752 z 0 Millimeters Default Grid Snap Ortho Planar Osnap SmartTrack Gumball (CPlane) Auto CPlane

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).

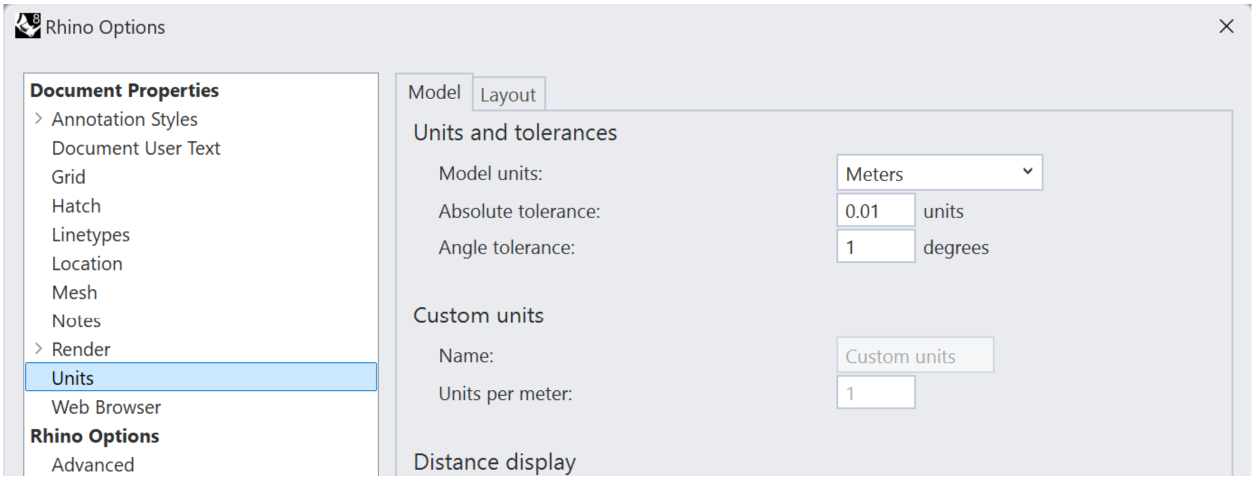
Both snapping options can be active simultaneously.

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.



Units and tolerances

Model units: Meters

Absolute tolerance: 0.01 units

Angle tolerance: 1 degrees

Custom units

Name: Custom units

Units per meter: 1

Distance display

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.

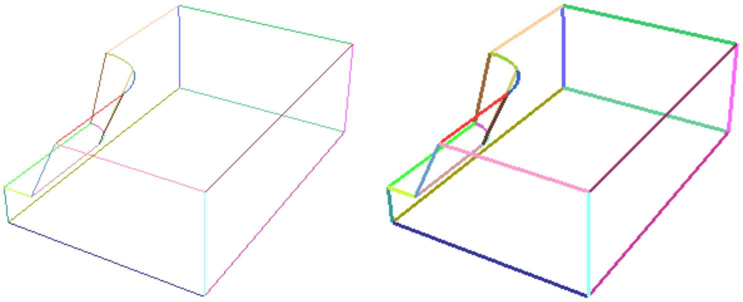
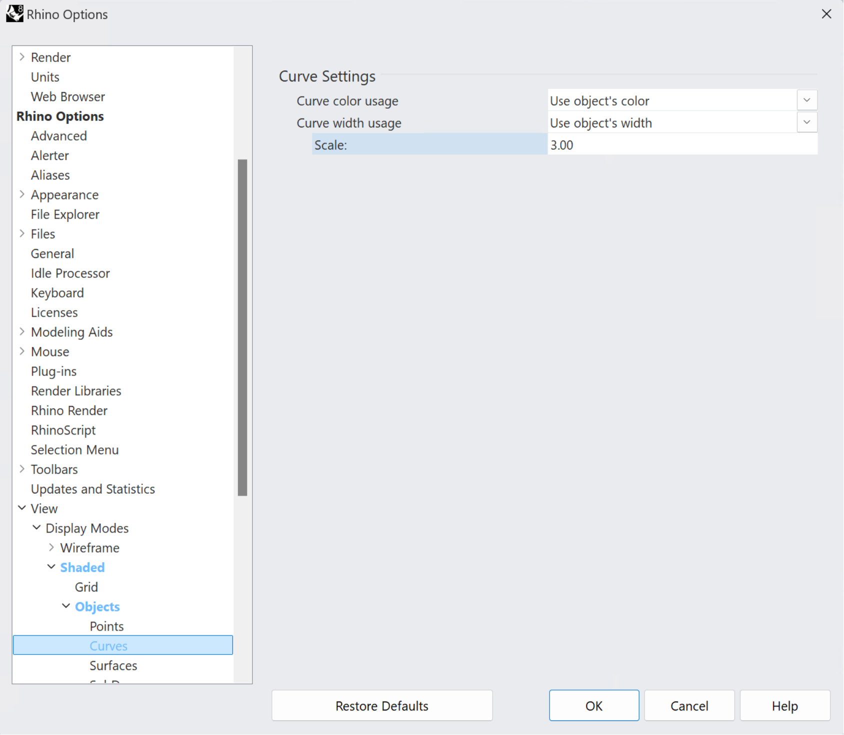


Figure 5: A regular (left) and “thick line” (right) representation.



Curve Settings

Curve color usage: Use object's color

Curve width usage: Use object's width

Scale: 3.00

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.

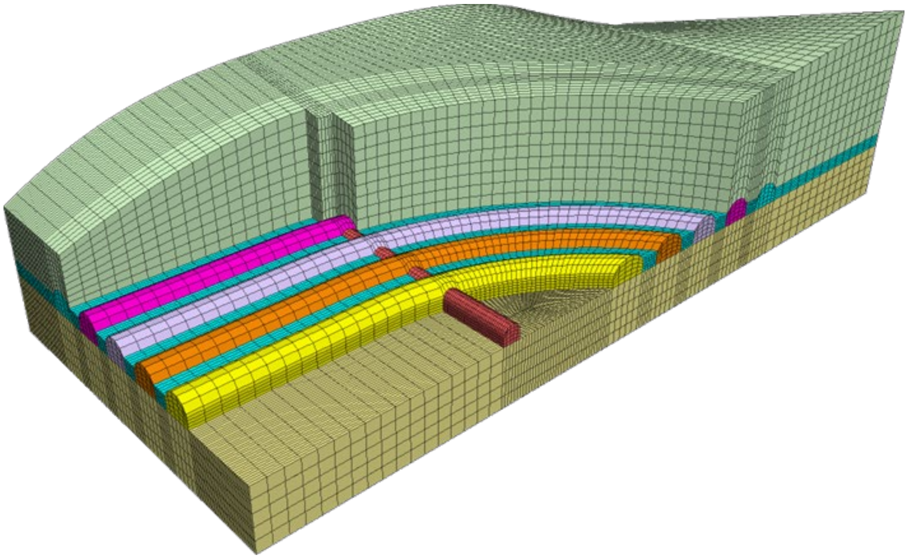


Figure 7: All hexahedral structured *FLAC3D* mesh generated by BlockRanger.

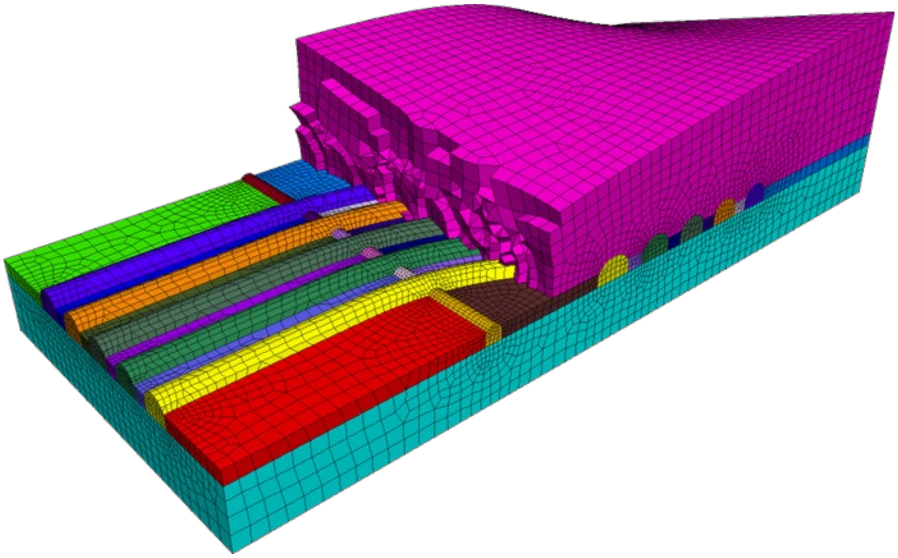


Figure 8: Hexahedral dominant unstructured *FLAC3D* mesh generated by GVol.

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.

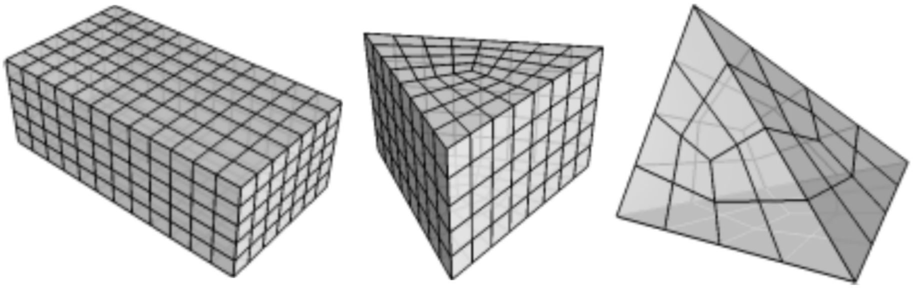


Figure 9: An example of solids supported by BlockRanger and BlockRanger generated mesh patterns for each (only exterior mesh boundaries are shown).

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.

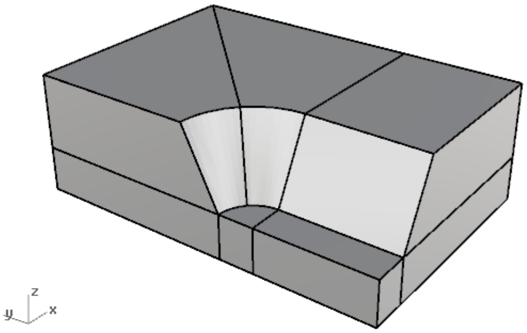


Figure 10: 3D slope example.

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*.

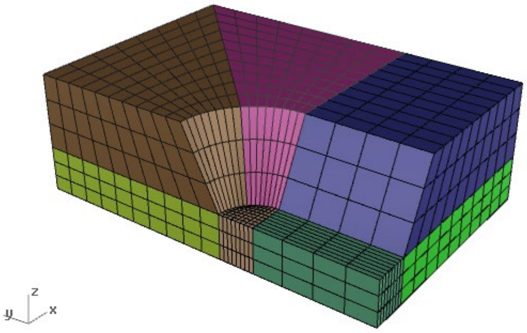


Figure 11: BlockRanger generated mesh with the default parameters.

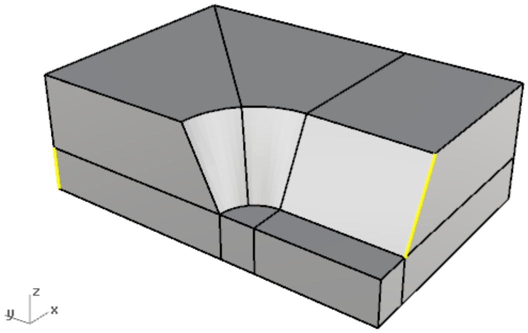


Figure 12: Duplicated edges (highlighted) with a custom number of elements specified.

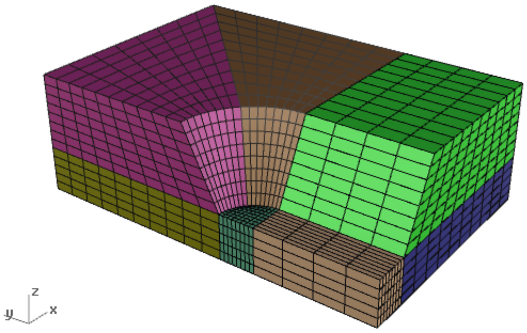


Figure 13: BlockRanger generated mesh with specified edge resolution.

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.

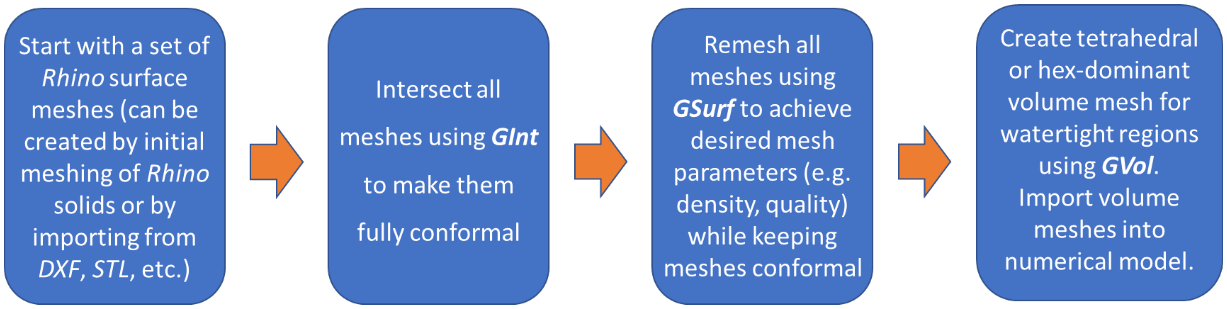
Using *Griddle* Tools for Unstructured Meshing

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.


Start with a set of *Rhino* surface meshes (can be created by initial meshing of *Rhino* solids or by importing from *DXF*, *STL*, etc.)

Intersect all meshes using ***GInt*** to make them fully conformal

Remesh all meshes using ***GSurf*** to achieve desired mesh parameters (e.g. density, quality) while keeping meshes conformal

Create tetrahedral or hex-dominant volume mesh for watertight regions using ***GVol***. Import volume meshes into numerical model.

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.

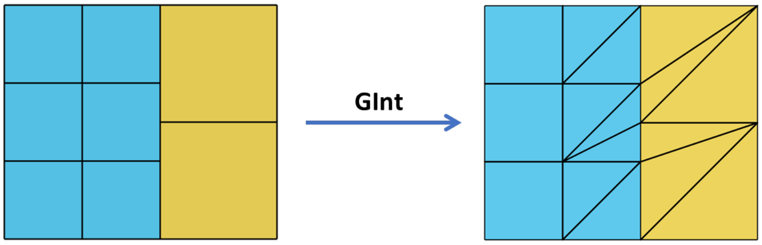


Figure 16: Example of non-conformal (left) and conformal (right) 2D meshes.

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.

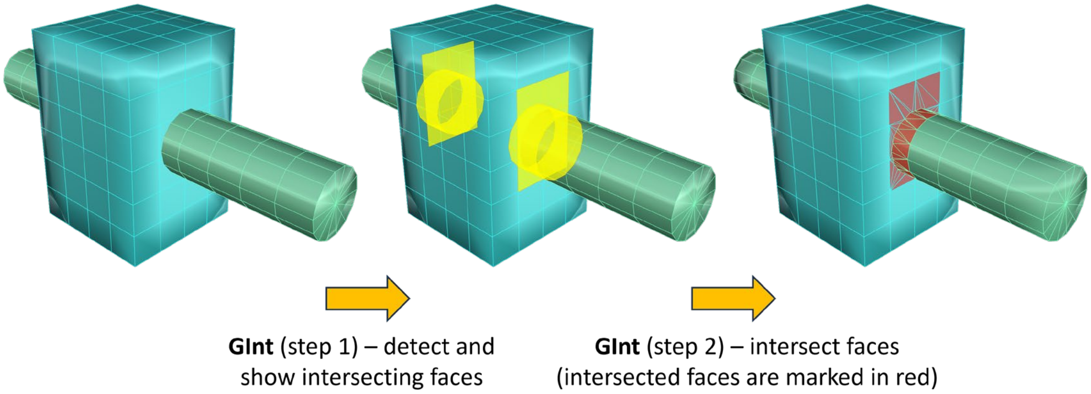

GInt (step 1) – detect and
show intersecting faces
GInt (step 2) – intersect faces
(intersected faces are marked in red)

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.

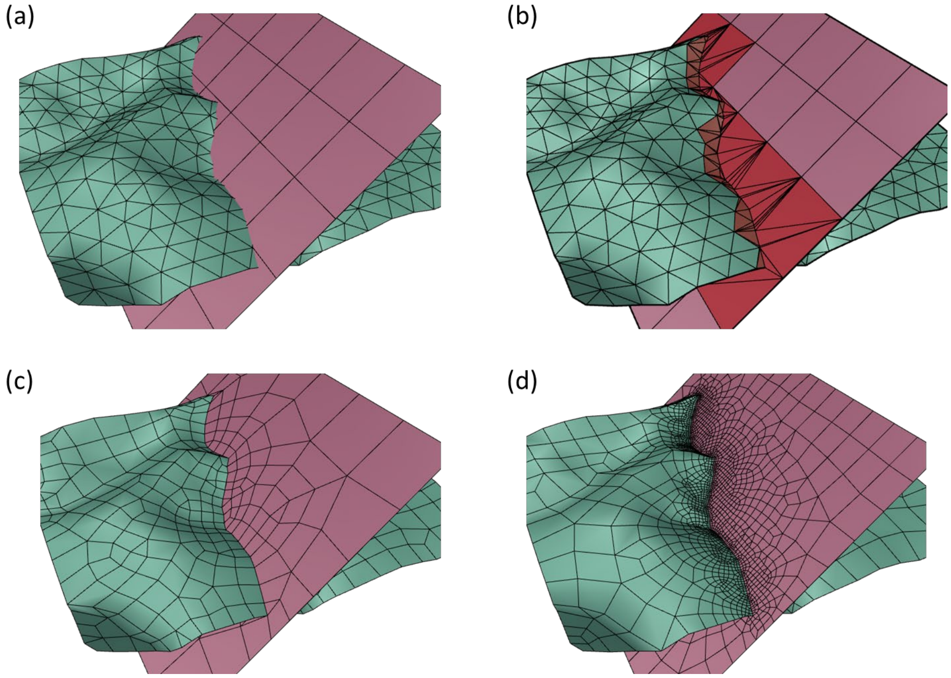


Figure 18: Examples of remeshing with GSurf without (c) and with local size control (d).

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**.

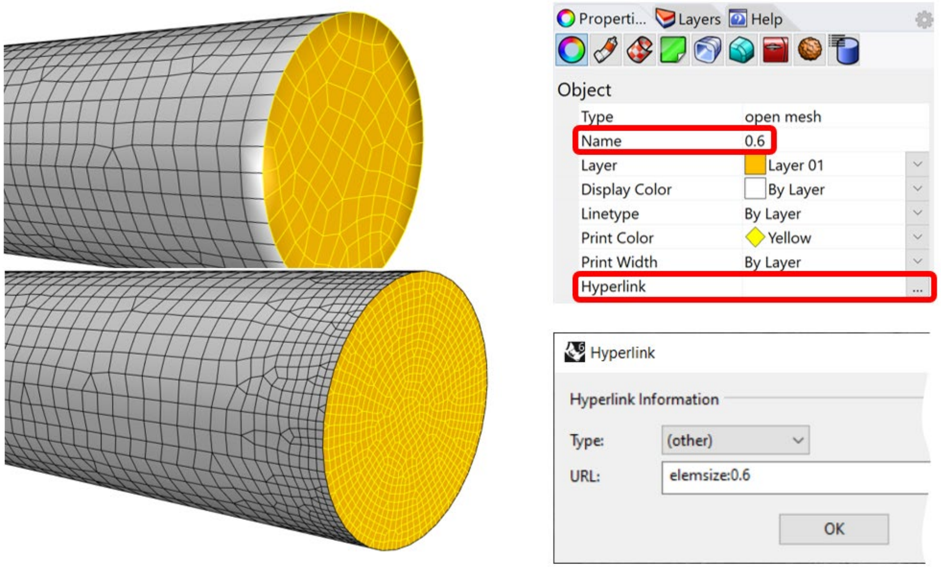
Local element size for a mesh can be specified in one of two ways:

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.



Object

Type	open mesh
Name	0.6
Layer	Layer 01
Display Color	By Layer
Linetype	By Layer
Print Color	Yellow
Print Width	By Layer
Hyperlink	...

Hyperlink

Hyperlink Information

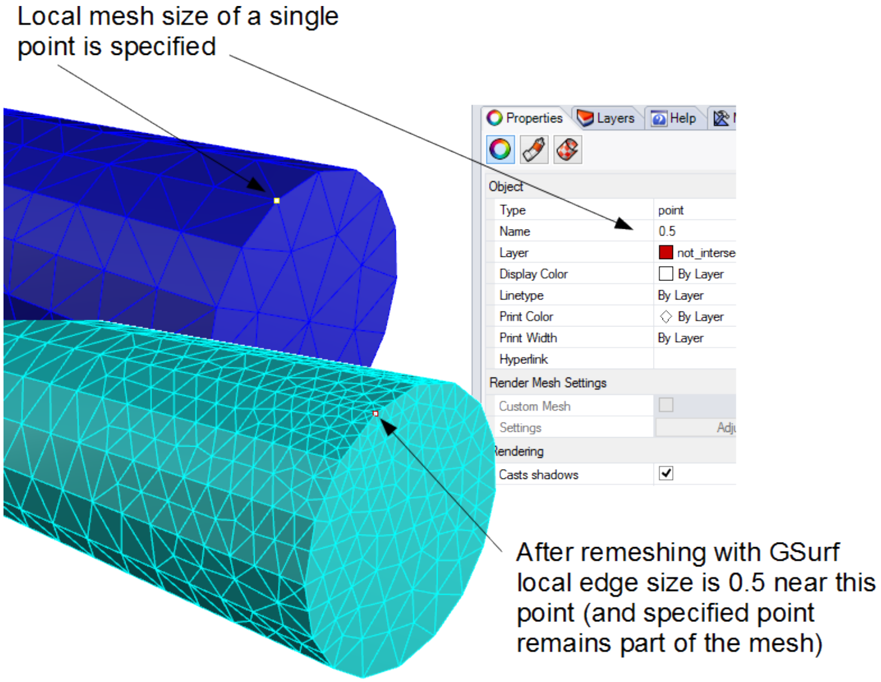
Type: (other)

URL: elemsize:0.6

OK

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.



Object
Type: point
Name: 0.5
Layer: 🟥 not_interse
Display Color: □ By Layer
Linetype: By Layer
Print Color: ◇ By Layer
Print Width: By Layer
Hyperlink:
Render Mesh Settings
Custom Mesh: □
Settings: Adj
Rendering
Casts shadows: ☑

Local mesh size of a single point is specified

After remeshing with GSurf local edge size is 0.5 near this point (and specified point remains part of the mesh)

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.

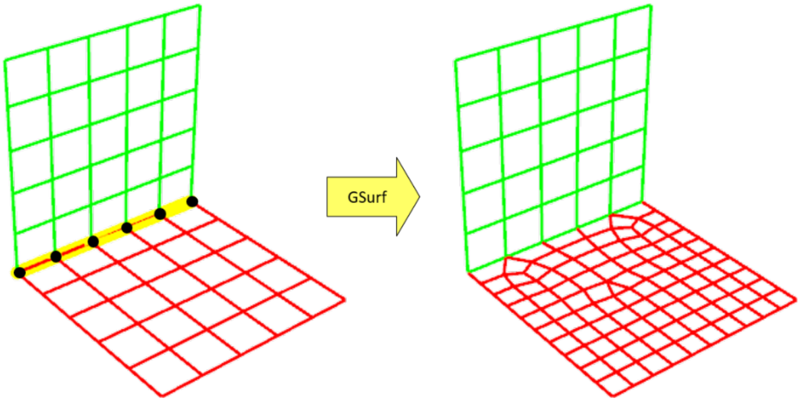


Figure 21: Specifying hard edges.

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.

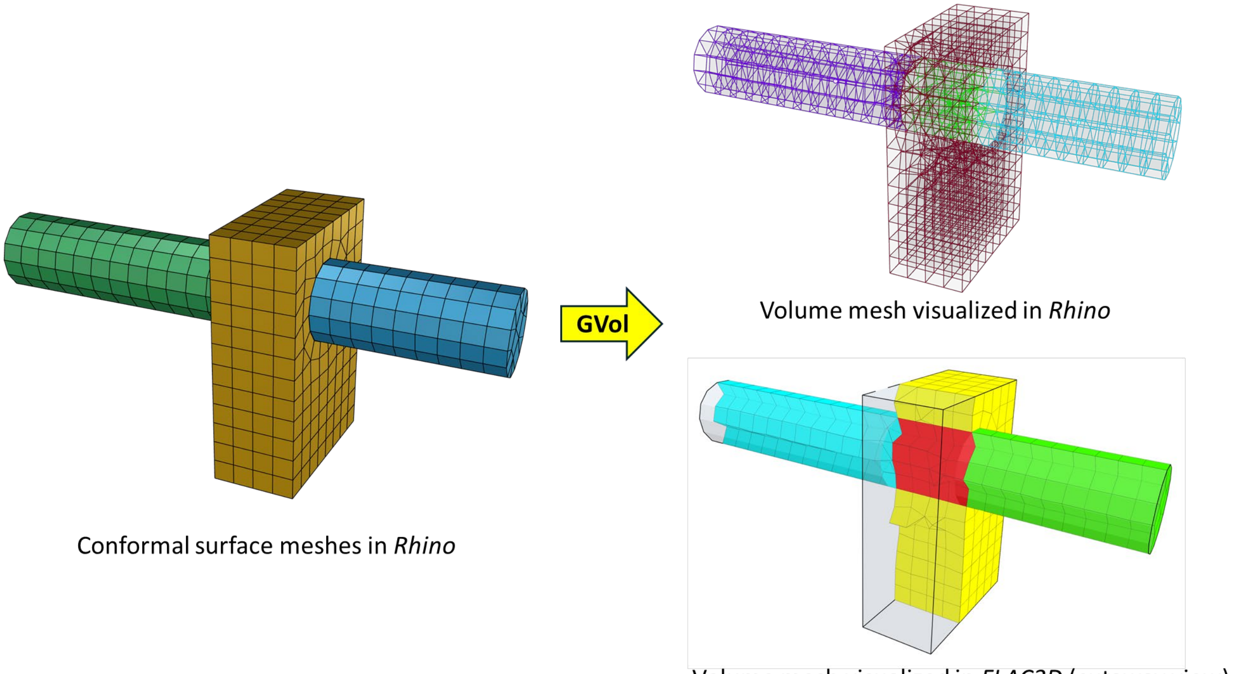
GVol – Unstructured Volume Mesher

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.



Volume mesh visualized in *Rhino*

Conformal surface meshes in *Rhino*

Volume mesh visualized in *FLAC3D* (cutaway view)

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 = …)
    - …

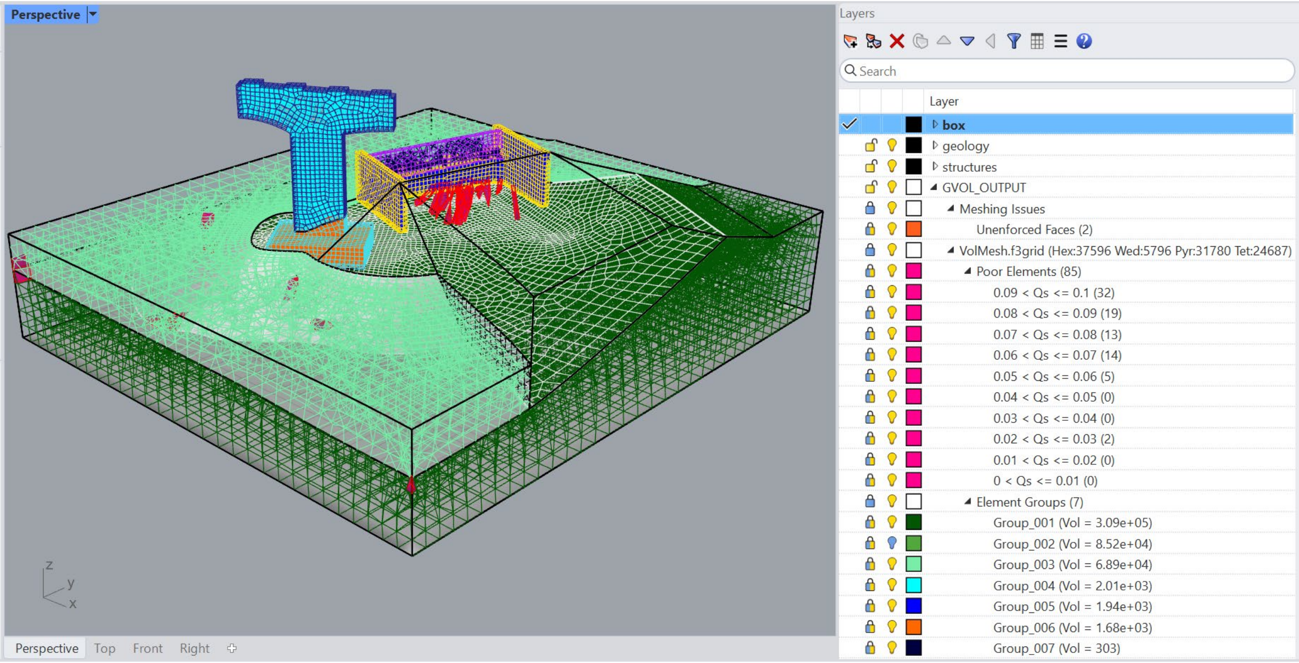
"Meshing Issues" layer provides information about possible issues **GVol** may encounter during meshing.

- "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.



```
▷ box
▷ geology
▷ structures
◢ GVOL_OUTPUT
  ◢ Meshing Issues
      Unenforced Faces (2)
  ◢ VolMesh.f3grid (Hex:37596 Wed:5796 Pyr:31780 Tet:24687)
    ◢ Poor Elements (85)
        0.09 < Qs <= 0.1 (32)
        0.08 < Qs <= 0.09 (19)
        0.07 < Qs <= 0.08 (13)
        0.06 < Qs <= 0.07 (14)
        0.05 < Qs <= 0.06 (5)
        0.04 < Qs <= 0.05 (0)
        0.03 < Qs <= 0.04 (0)
        0.02 < Qs <= 0.03 (2)
        0.01 < Qs <= 0.02 (0)
        0 < Qs <= 0.01 (0)
    ◢ Element Groups (7)
        Group_001 (Vol = 3.09e+05)
        Group_002 (Vol = 8.52e+04)
        Group_003 (Vol = 6.89e+04)
        Group_004 (Vol = 2.01e+03)
        Group_005 (Vol = 1.94e+03)
        Group_006 (Vol = 1.68e+03)
        Group_007 (Vol = 303)
```

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).

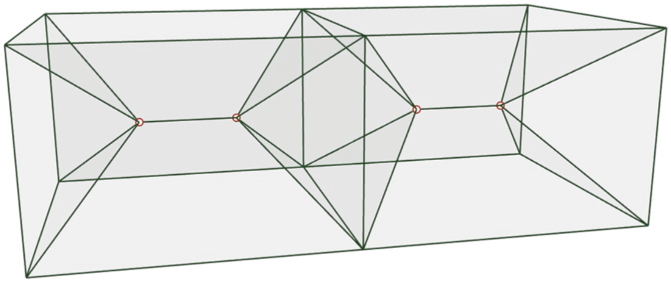


Figure 24. A volume mesh with 4 hard nodes.

- **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.

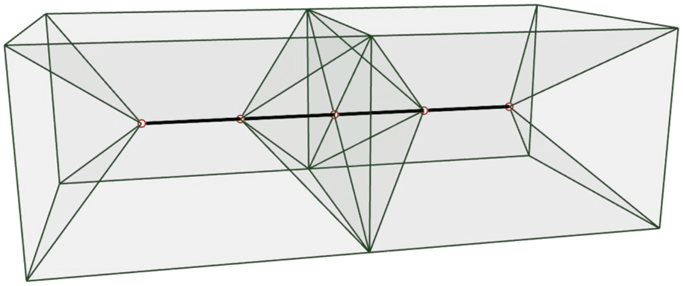


Figure 25. A volume mesh with a hard edge split into 4 segments.

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*).

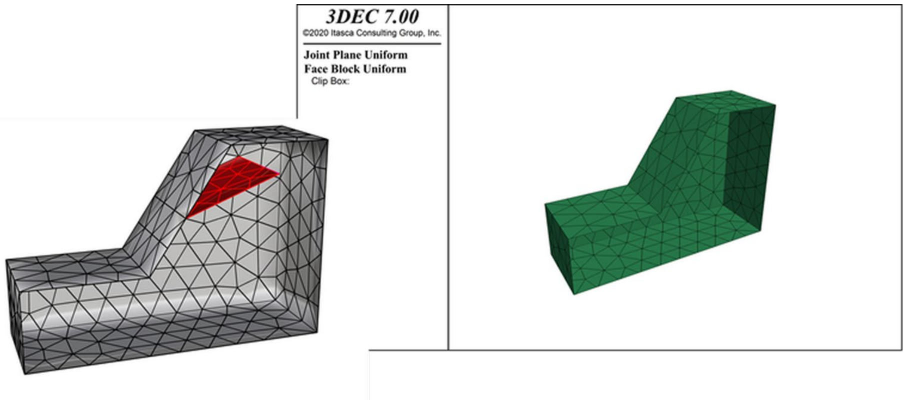


Figure 26: *Rhino* model (left) with floating surface (red) and *Joint* plot in *3DEC* representing the rigid block (right). The red surface does not become part of the rigid block (or a joint) in *3DEC*.

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:

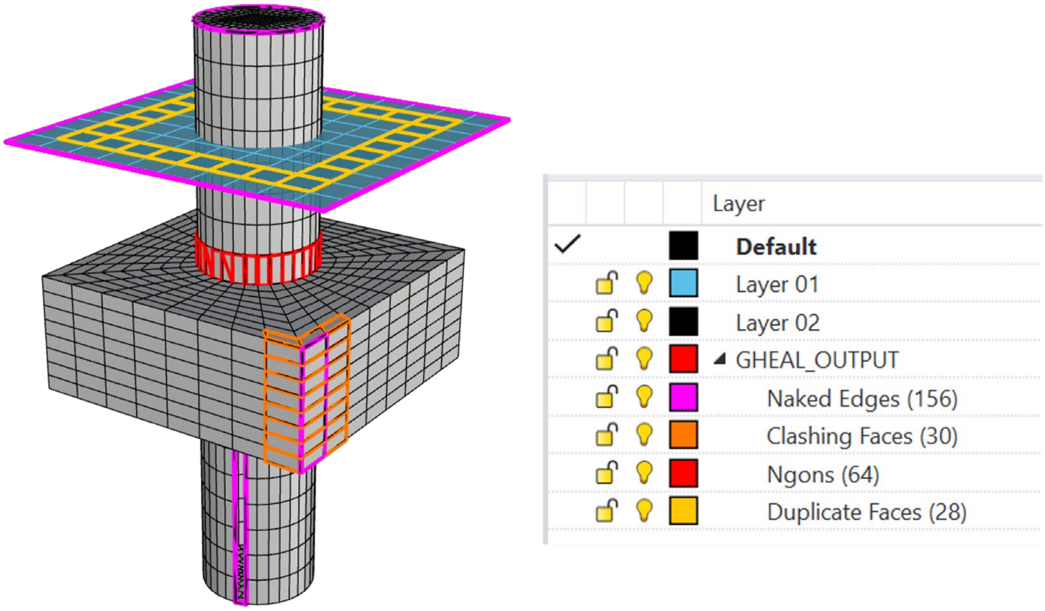
Naked edges are open face edges which belong to a single face only (not shared by connected edges). Naked edges often represent mesh boundaries and discontinuities; they can be found along disjoint and/or nonconformal faces and may not be visible (as faces may visually appear to be connected).

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.



				Layer
✓			■	**Default**
	🔓	💡	■	Layer 01
	🔓	💡	■	Layer 02
	🔓	💡	🟥	◢ GHEAL_OUTPUT
	🔓	💡	■	Naked Edges (156)
	🔓	💡	🟧	Clashing Faces (30)
	🔓	💡	🟥	Ngons (64)
	🔓	💡	🟨	Duplicate Faces (28)

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.

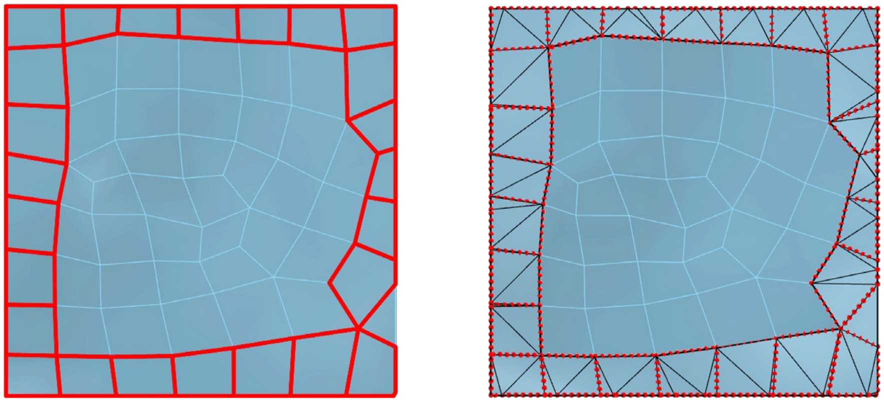


Figure 28: GHeal's function *TriangulateNgons* triangulates all faces with 5 or more sides. Red lines indicate Ngons present (left) and are outlined in by the dotted line for reference on the right.

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.

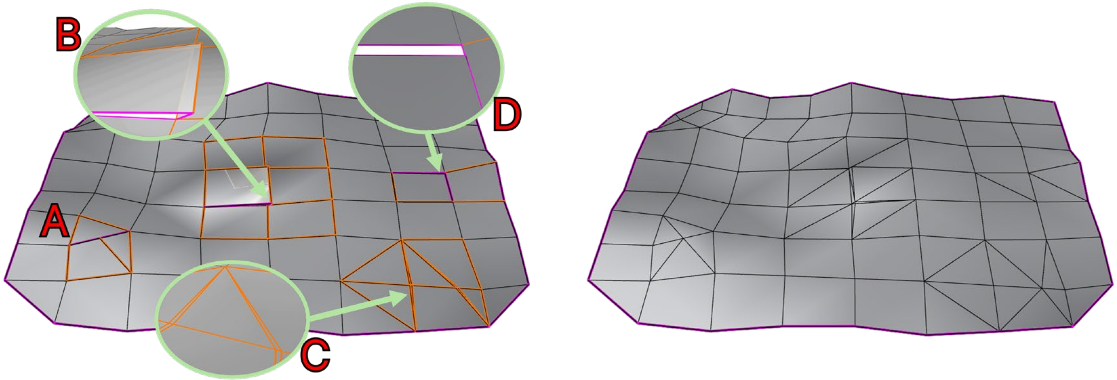


Figure 29: GHeal's function *FixClashingFaces* resolving various problems within a surface mesh.

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.

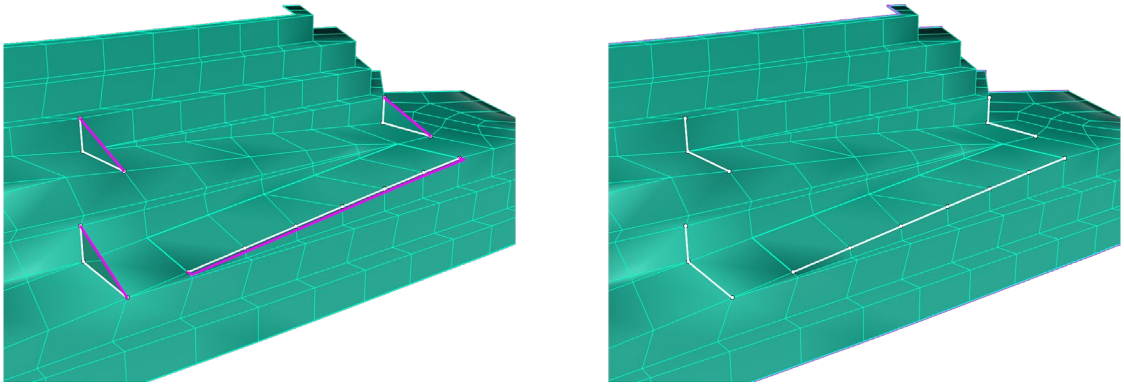


Figure 30: GHeal's function *RemoveNonmanifold* removes non-manifold pieces. Non-manifold edges are shown in white (left) and are removed upon operation completion (right).

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.*

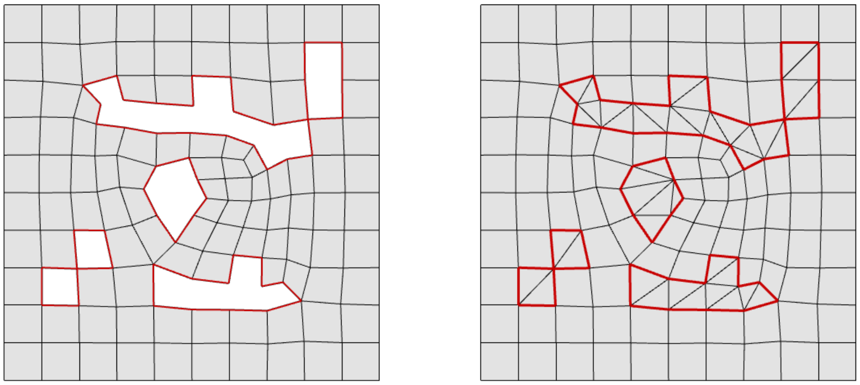


Figure 31: GHeal's function *FillHoles* fixes various holes in the surface mesh. Red lines show naked edges (left) and are provided for reference in the right image.

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.

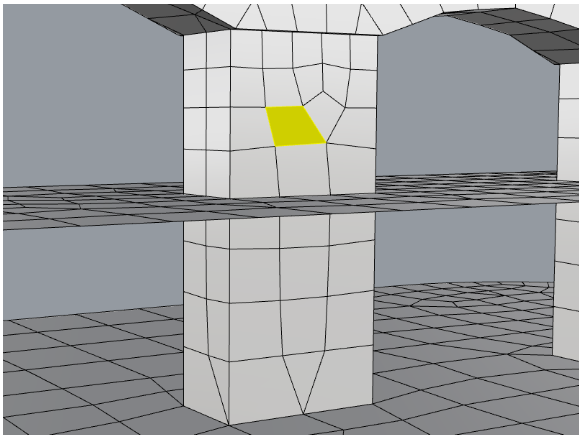


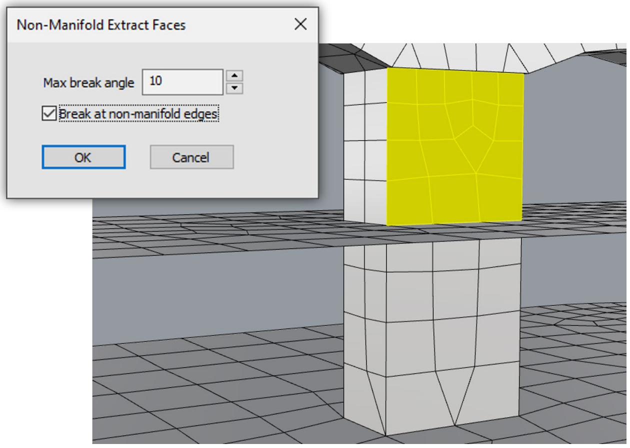
Figure 32: Selecting an initial face for GExtract's *Single* extraction mode.

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.

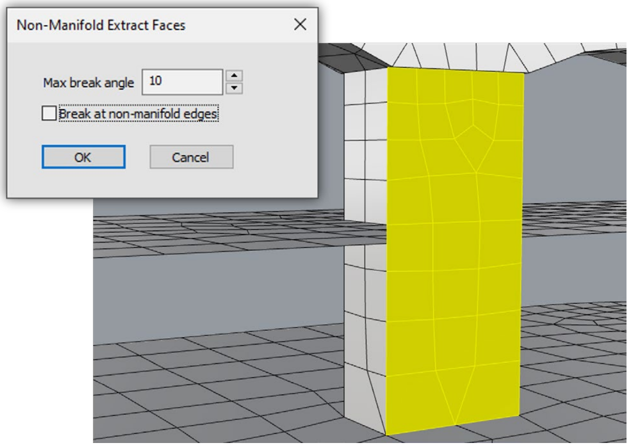


Non-Manifold Extract Faces

Max break angle 10

☑ Break at non-manifold edges

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



Non-Manifold Extract Faces

Max break angle 10

□ Break at non-manifold edges

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.

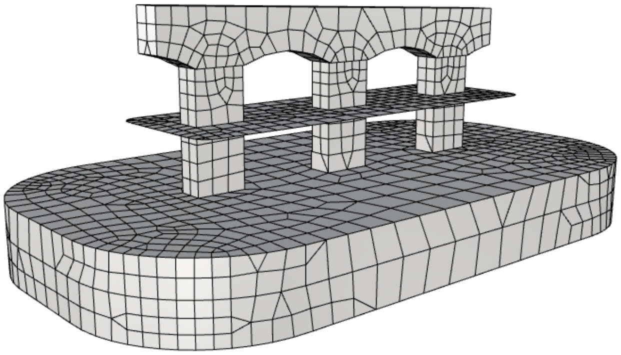


Figure 35: Initial single non-manifold conformal mesh (some internal meshes are not visible).

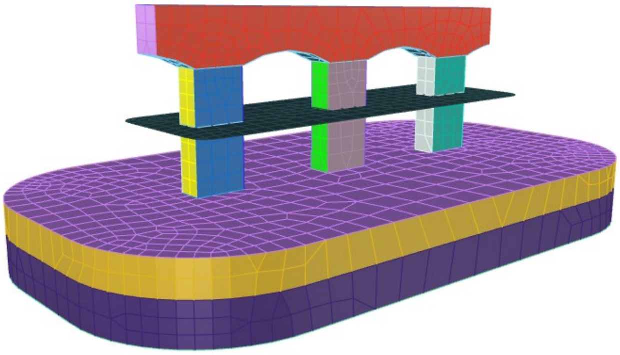


Figure 36: The result of application of GExtract in *Multiple* mode.

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.

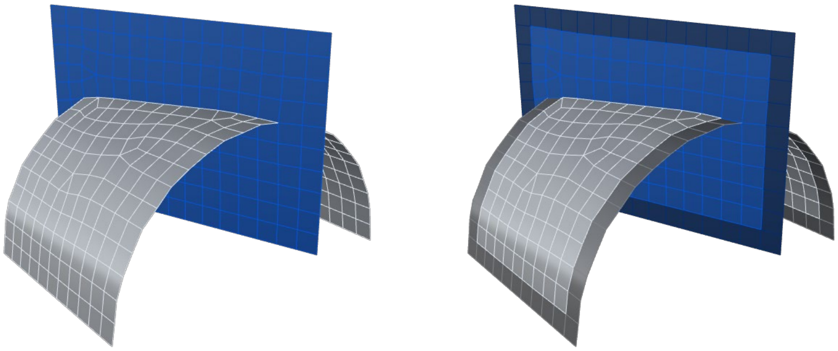


Figure 37: Extraction of boundary faces using GExtract's *Boundary* mode.

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).

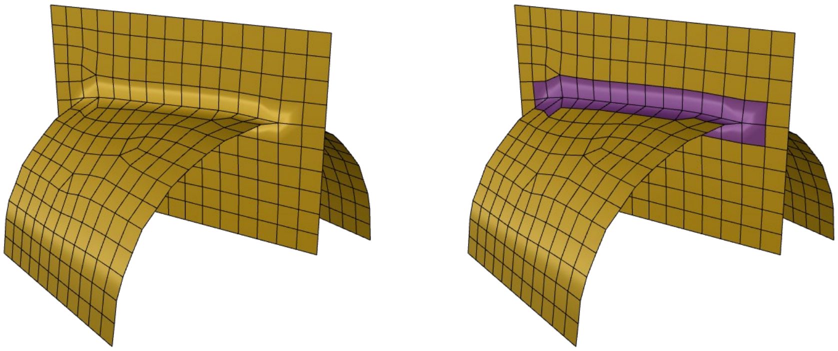


Figure 38: Extraction of faces attached to non-manifold edges using GExtract's *NonManifold* mode.

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.

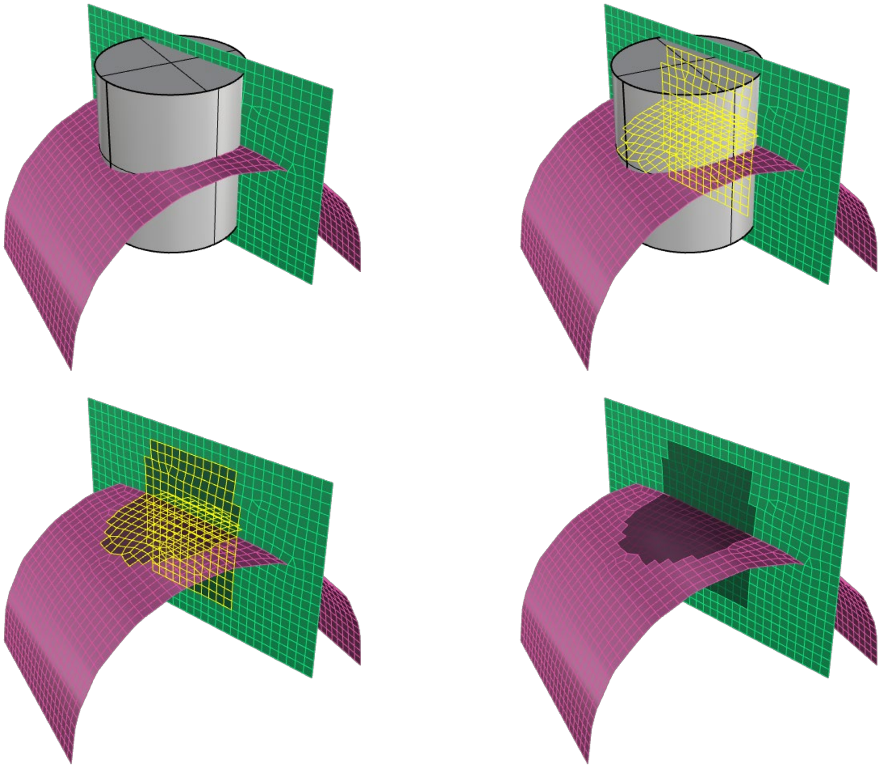


Figure 39: Process of extraction of mesh faces located within a solid by using GExtract's *WithinSolid* mode.

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.

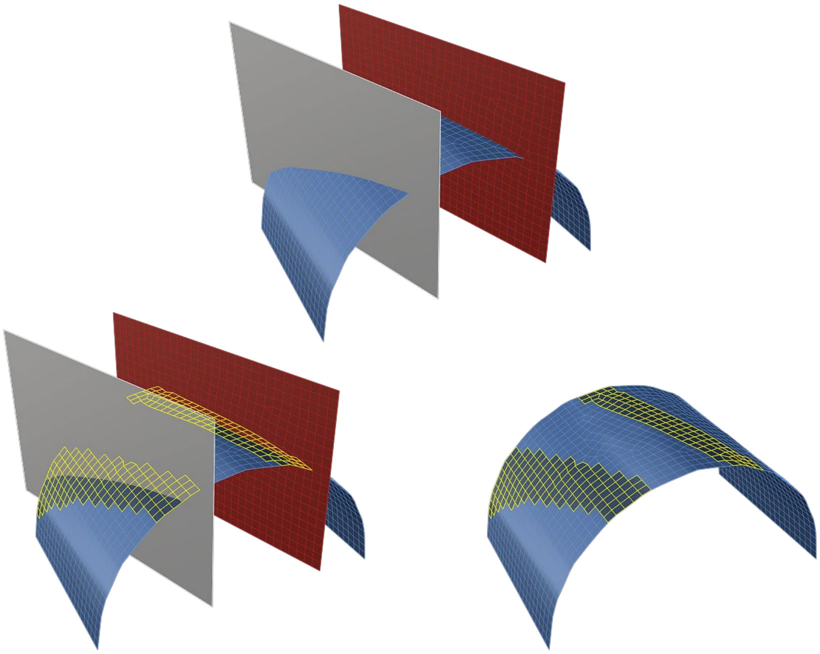


Figure 40: Extraction of mesh faces within distance of another mesh and a surface.

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.

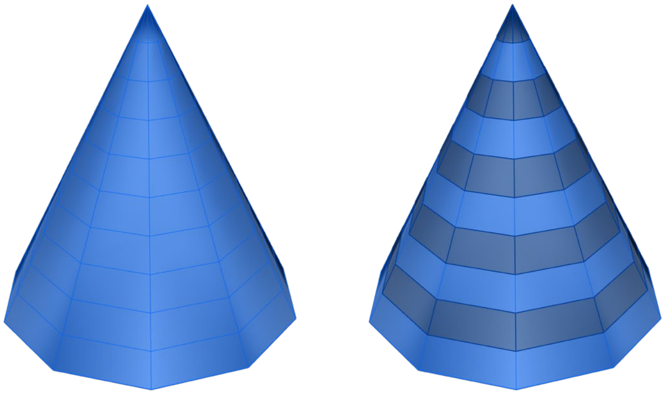


Figure 41: Extraction of mesh faces by face area. For this example, GExtract was run 5 times with increasing *Min* and *Max* area values each time to obtain the results shown on the left.

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)

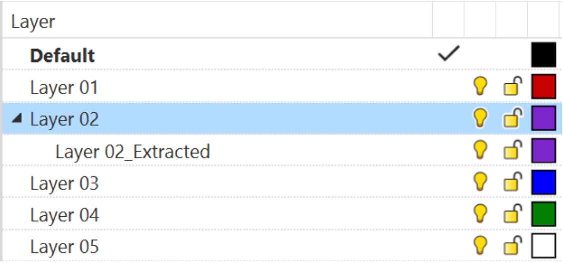
Specifies that the extracted mesh parts will be placed in the sublayers of the layers with the original meshes. Suffix "_Extracted" is added to the sublayers' names. If sublayers with such names already exist, the extracted objects will be added (appended) to them.

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.



Layer			
Default	✓		■
Layer 01	💡	🔓	🟥
◢ Layer 02	💡	🔓	🟪
Layer 02_Extracted	💡	🔓	🟪
Layer 03	💡	🔓	🟦
Layer 04	💡	🔓	🟩
Layer 05	💡	🔓	□

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.

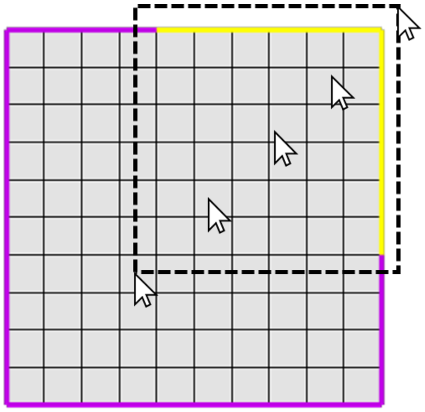


Figure 43: GExtend highlights mesh boundary in purple and allows selecting a piece of the boundary (in yellow).

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).

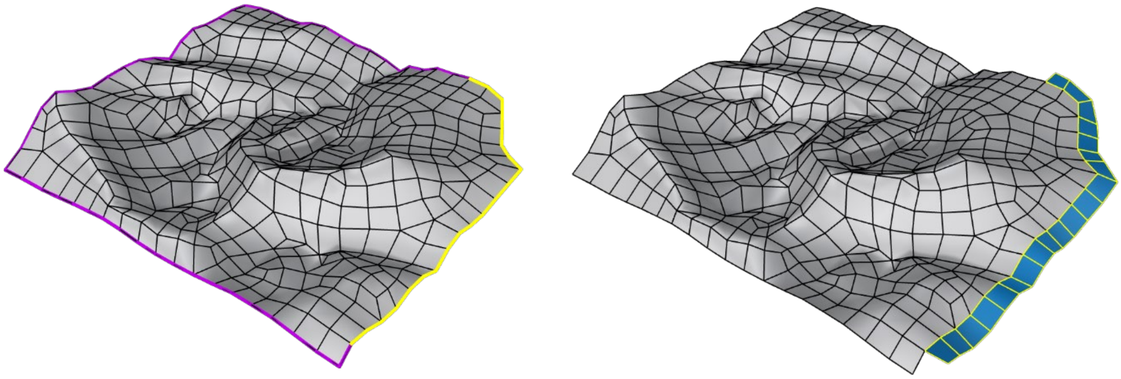


Figure 44: Extension of a mesh along a part of its boundary using *ExtendSelectedBoundary* mode.

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.

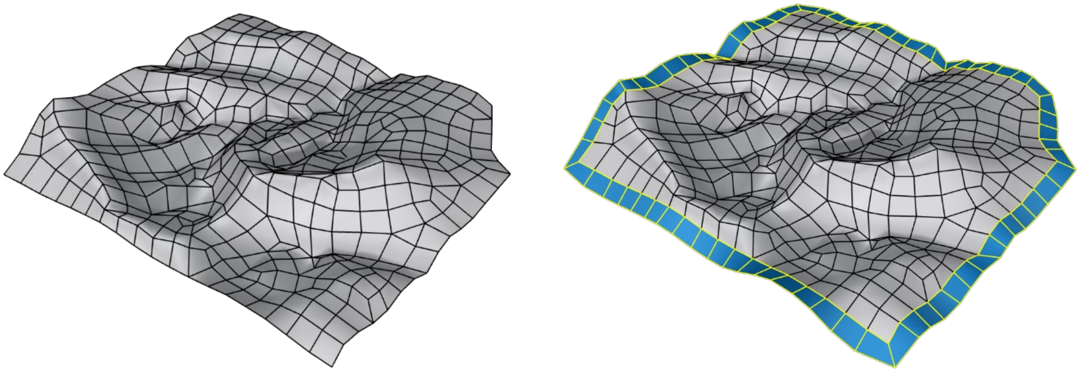


Figure 45: Expansion of a mesh along all its boundary using *ExtendAllBoundary* mode.

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.

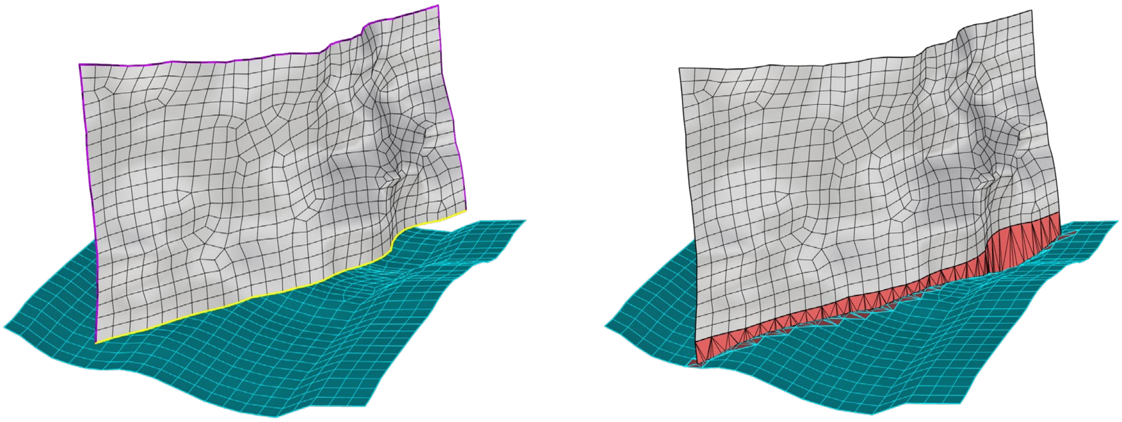


Figure 46: *Left: Source (gray) mesh with highlighted part boundary to be extended. Right: The source mesh is extended in LocalTanget direction and intersected with the target mesh.*

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.

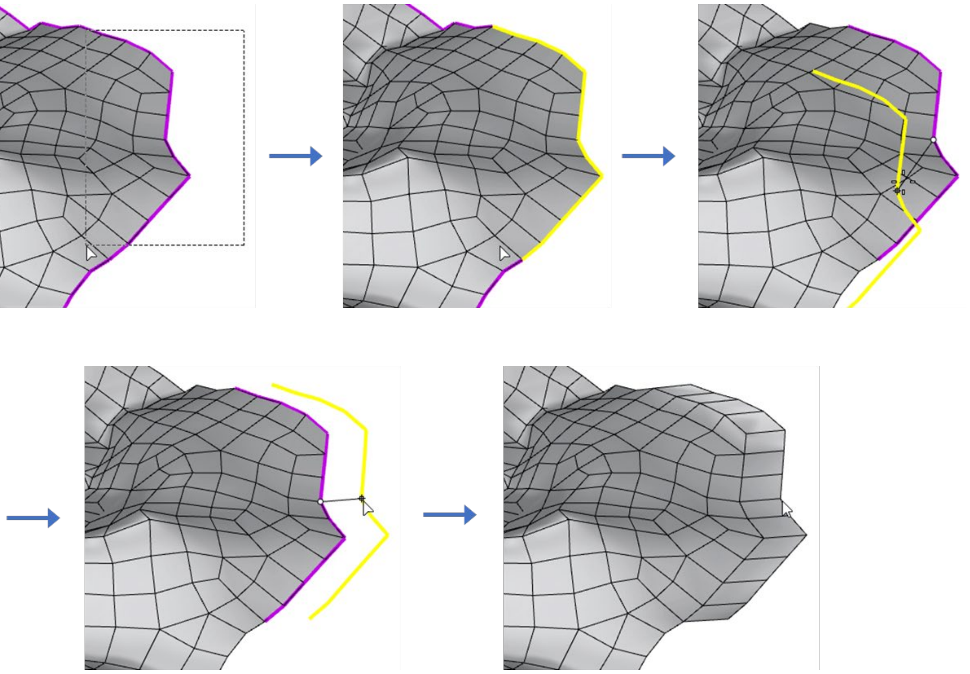


Figure 47: Process of selecting and dragging a part of a mesh boundary using *FreeExtend* mode.

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.

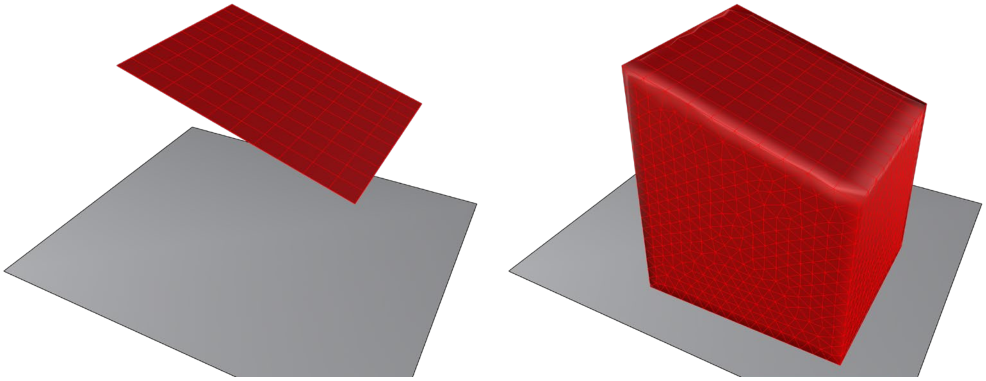


Figure 48: The initial mesh with a bounding surface (left) and the result of mesh extrusion.

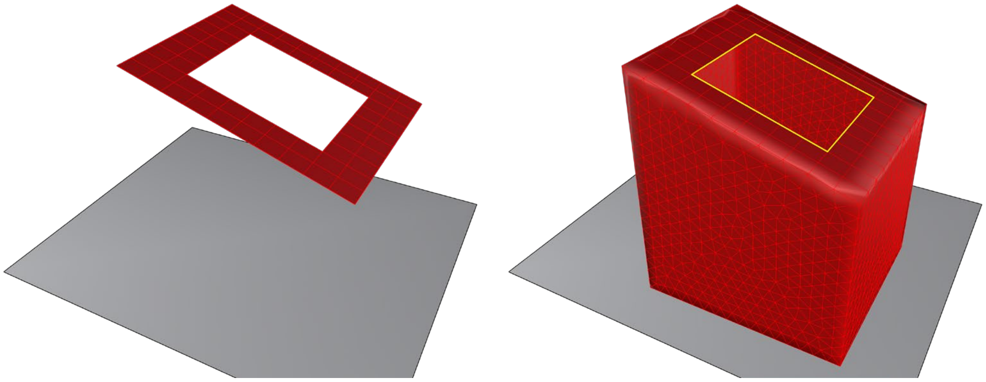


Figure 49: The result of the extrusion of *External* boundaries only when the initial mesh contains a hole. GExtrude produces an open (non-watertight) mesh domain.

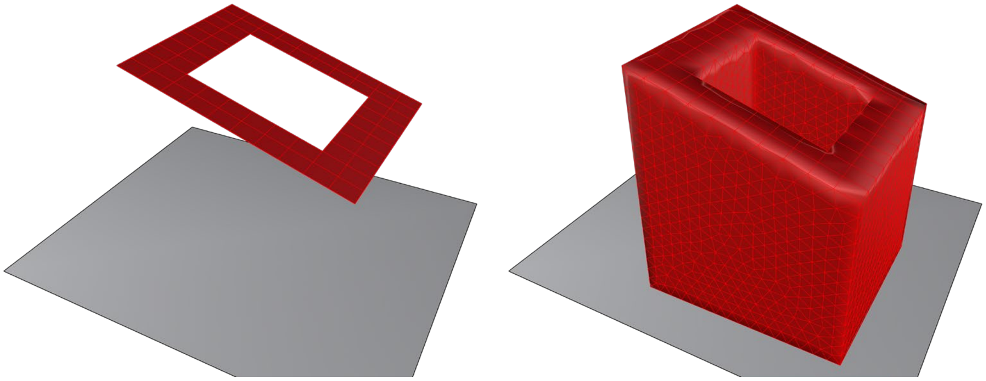


Figure 50: The result of the extrusion of *All* boundaries to form a watertight domain.

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.

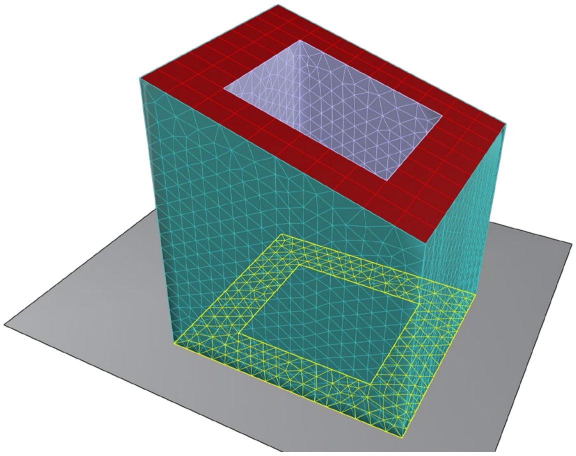


Figure 51: The result of GExtrude using the *Split* option.

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.

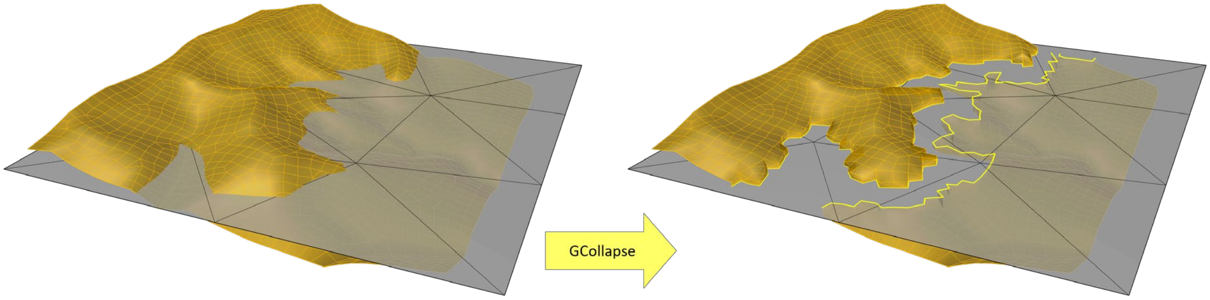


Figure 52: Collapsing a surface mesh onto a planar mesh.

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.

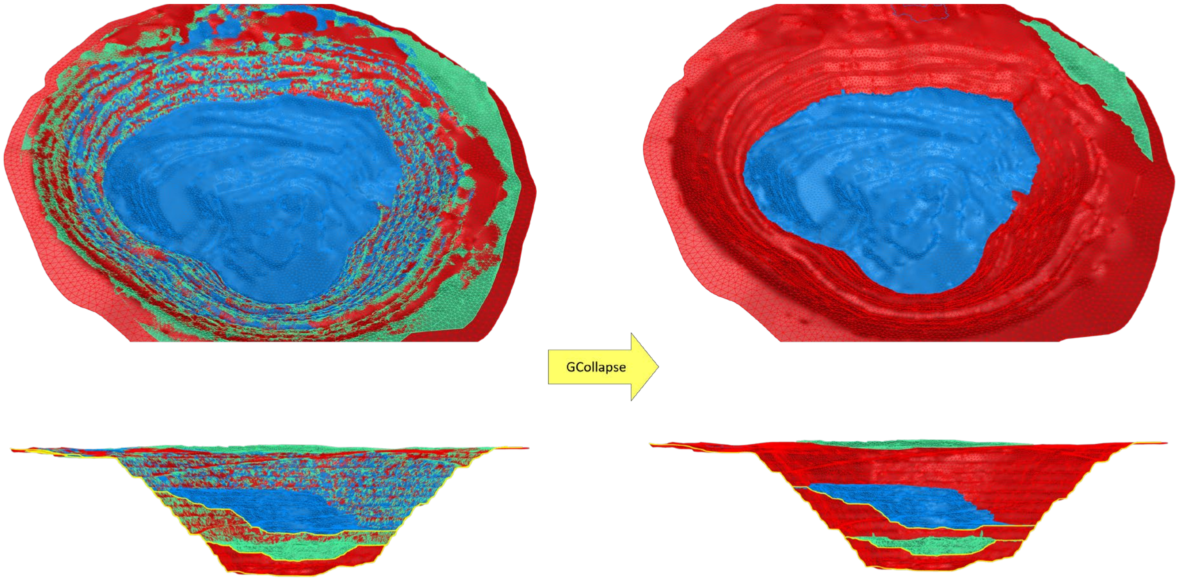


Figure 53: Resolving overlapping pit mining surfaces using GCollapse.

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).

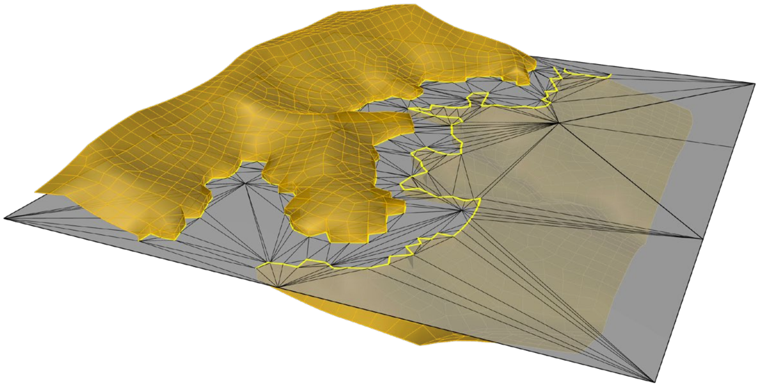


Figure 54: Intersecting the collapsed target mesh to the base mesh.

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.*

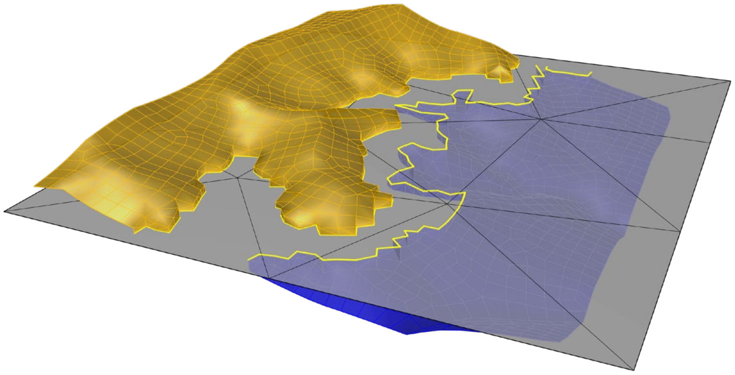


Figure 55: The target mesh split using *SplitAtIntersection*.

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)

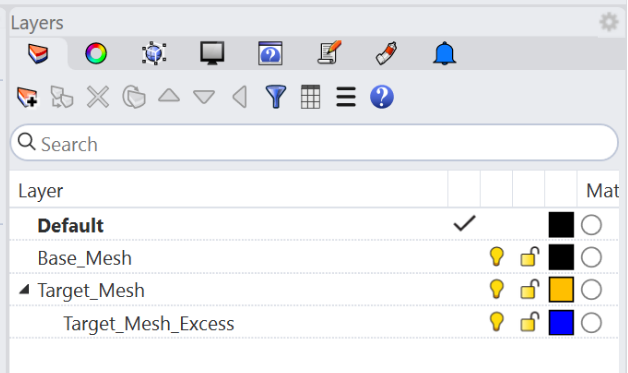
Specifies that the excess mesh pieces will be placed in the sublayer of the original mesh. If a sublayer with the name already exists, the excess mesh will be added (appended) to them.

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.



Layer					Mat
Default	✓			■	○
Base_Mesh		💡	🔓	■	○
◢ Target_Mesh		💡	🔓	🟨	○
Target_Mesh_Excess		💡	🔓	🟦	○

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*.

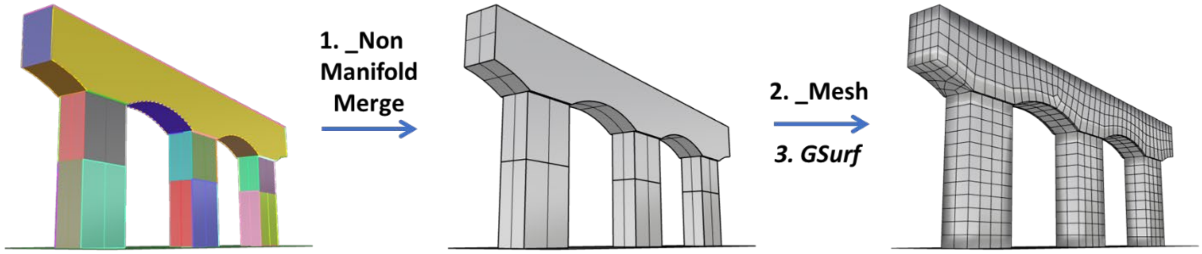


Figure 57: Three step process to create a single mesh out of multiple surfaces/polysurfaces.

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*.

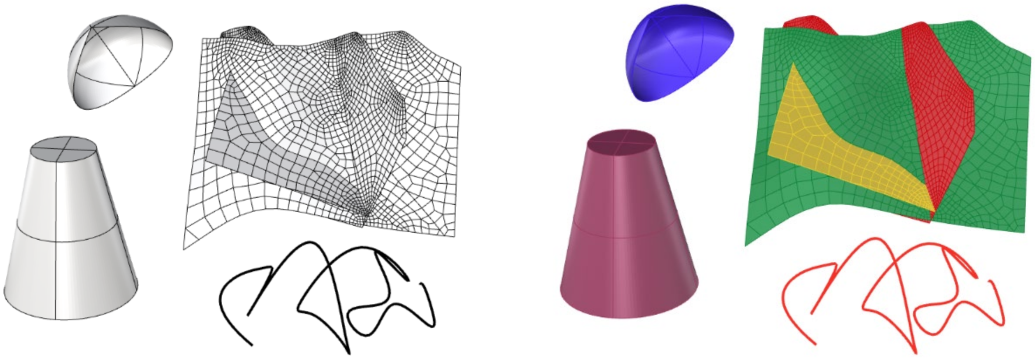


Figure 58: Various objects before and after ColorizeObjects is applied.

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

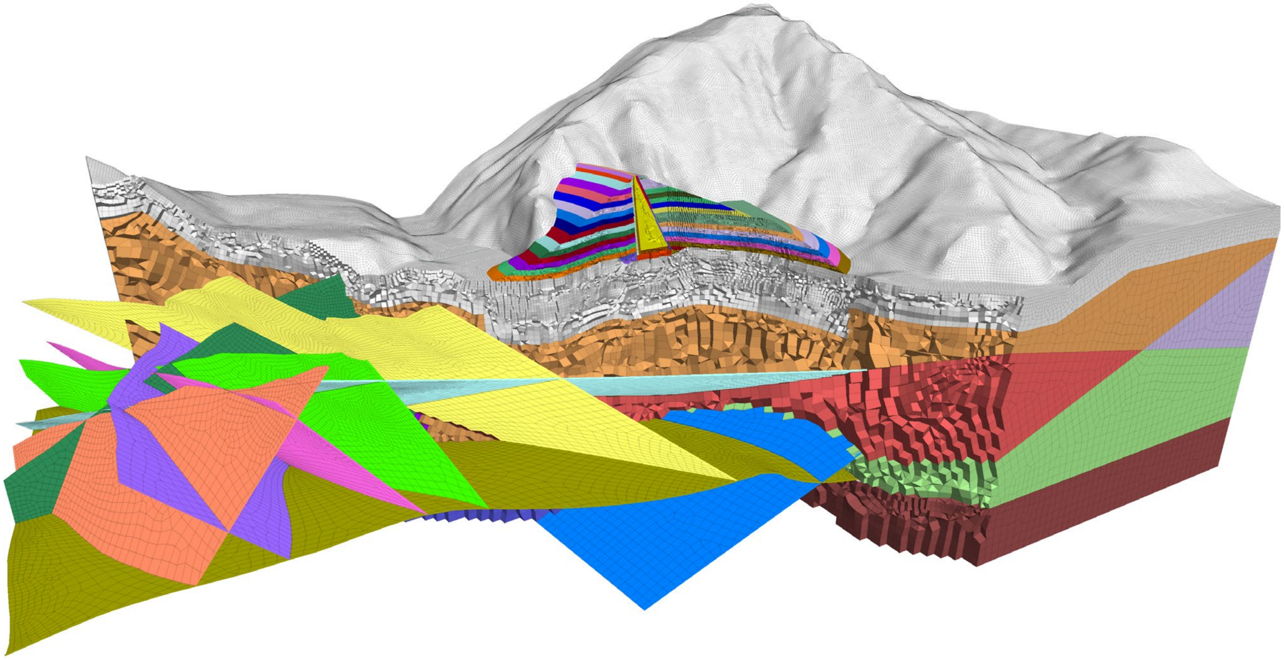
The Rhino window and interface: https://docs.mcneel.com/rhino/8/help/en-us/user_interface/rhino_window.htm

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

Griddle™

Advanced Grid Generation for Engineers and Scientists

Griddle 3.0 Tutorial Examples



©2024
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

The tutorial examples provided in this document are designed to familiarize *Griddle* users with the principles of *Griddle* operation, workflows, and typical *Rhino* commands that allow the creation of structured and unstructured volume meshes suitable for numerical analysis.

The examples presented in the tutorial are not designed to illustrate exact (or even realistic) engineering models and conditions. However, they provide the basis on which *Griddle* users can build their knowledge for working with complex engineering models.

All the examples were developed in *Rhino* 8. When working on the examples from the tutorial, periodically save the model in order to return to it if needed. *Rhino* provides a useful file saving feature, called *Incremental Save* (accessible from the Menu **File** → **Incremental Save**).

Refer to the *Griddle* User Manual to find more information about *Griddle* commands and options, and the *Rhino* help system (top menu **Help** or **F1**) for *Rhino* operations.

The files corresponding to the tutorial examples can be accessed from the *Windows Start Menu* → *Itasca Griddle 3.0* → *Griddle 3.0 Tutorial Files* link. Clicking on 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`).

Tutorial 1: A Single Cylinder

Estimated work time: 0.5 – 1 hour

This tutorial provides information on how to create unstructured and structured meshes for a simple cylinder using *Griddle*'s **GVol** and **BlockRanger** volume meshers. Before proceeding with meshing, a cylindrical geometry is created, and this process differs depending on which volume mesher will be used.

Generating unstructured mesh with GVol

1. Open *Rhino*, and when the **Templates** splash screen appears, select **Small Objects - Meters**. The same can be done by navigating to **File** → **New** from the *Rhino* top menu.
2. Save the *Rhino* project at a desired location.
3. Type the **_Cylinder** command or select the **Solid Tools** → **Cylinder** menu item. Type **0** and press **Enter** to set the location of the center of the cylinder base at the origin. Next, set the radius of the base to **2**. Enter **10** for the height of the cylinder. Press **Alt+Ctrl+E** to zoom out all viewports to the extent of all objects (or navigate to **View** → **Zoom** → **Zoom Extents All**). This completes construction of the geometry for a simple vertical cylinder (Figure 1).

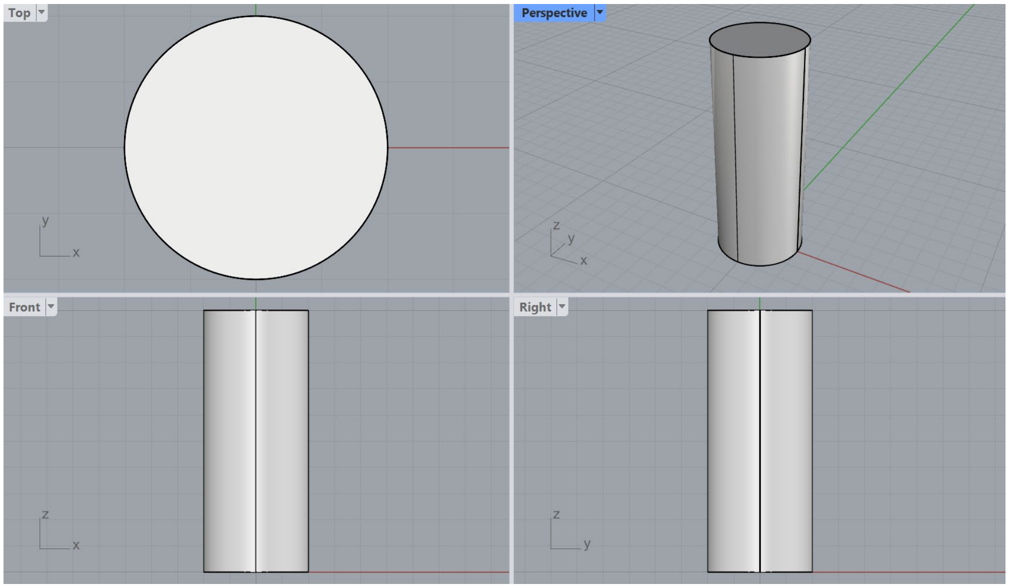


Figure 1: Solid representing a cylinder

The watertight geometry created in the previous steps can be referred to as a cylindrical solid. A solid is a closed surface that defines a volume with a clear interior and exterior. The next step is to triangulate the cylinder surface to create the initial triangular mesh.

4. Select the cylinder and type **_Mesh** in the *Rhino* command prompt. If the **Detailed Mesh Options** dialog window opens, click on the **Simple Controls** button in the lower-right corner of the window to open a simplified dialog.
5. In the simplified dialog, move the slider all the way to the right towards **More polygons** and click **OK** to create the initial surface mesh (Figure 2).

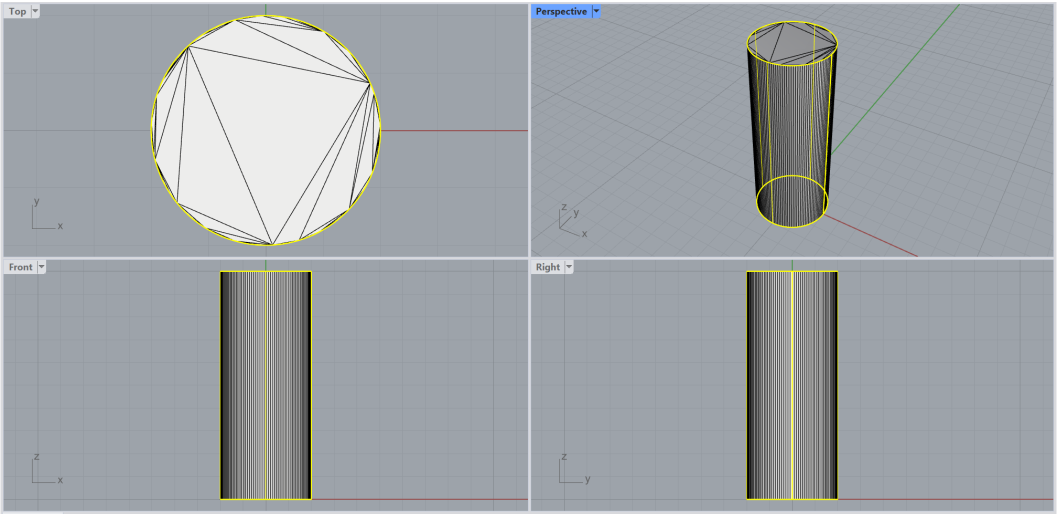


Figure 2: Original highlighted cylindrical surface and the newly created surface mesh superimposed on it.

6. While the original cylindrical surface is still highlighted (seen in light yellow in Figure 2), type the **_Hide** command to hide it, making only the surface mesh visible. This triangular surface mesh serves as input to *Griddle* tools for unstructured meshing.
7. Select the surface mesh and type **_GSurf** or click on the icon in the *Griddle* toolbar to remesh the initial triangular mesh**.** Select *Mode* = *QuadDom*, *MinEdgeLength* = 0.5, *MaxEdgeLength* = 0.5, <u>*AdvancedParameters*</u>: *ShapeQuality* = 0.65 and keep all other parameters at the default values. The resulting mesh is shown in Figure 3.

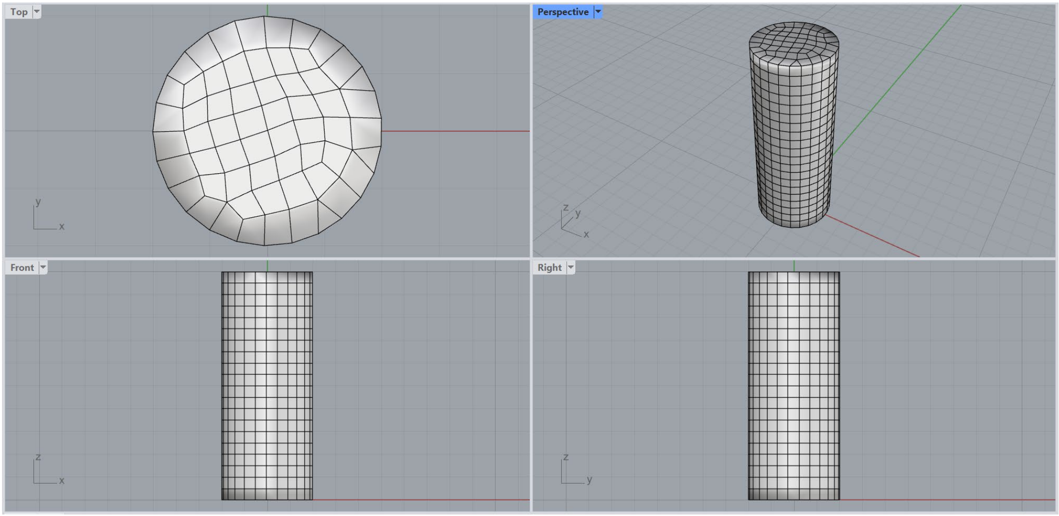


Figure 3: Quad-dominant surface mesh generated by GSurf.

8. Select the surface mesh generated in the previous step and type **_GVol** or click on the icon in the *Griddle* toolbar. Select **MeshSettings** → **Mode=HexDom**, leave all other parameters at the default settings and press **Enter** twice. After the **Save As** dialog appears, type a desired name for the output file. **GVol** will generate a conformal hex-dominant volume mesh and output it using the *FLAC3D* binary format (the default format) to the selected file. This file can be imported into *FLAC3D* using *FLAC3D*'s **File** → **Grid** → **Import from FLAC3D …** option (Figure 4).

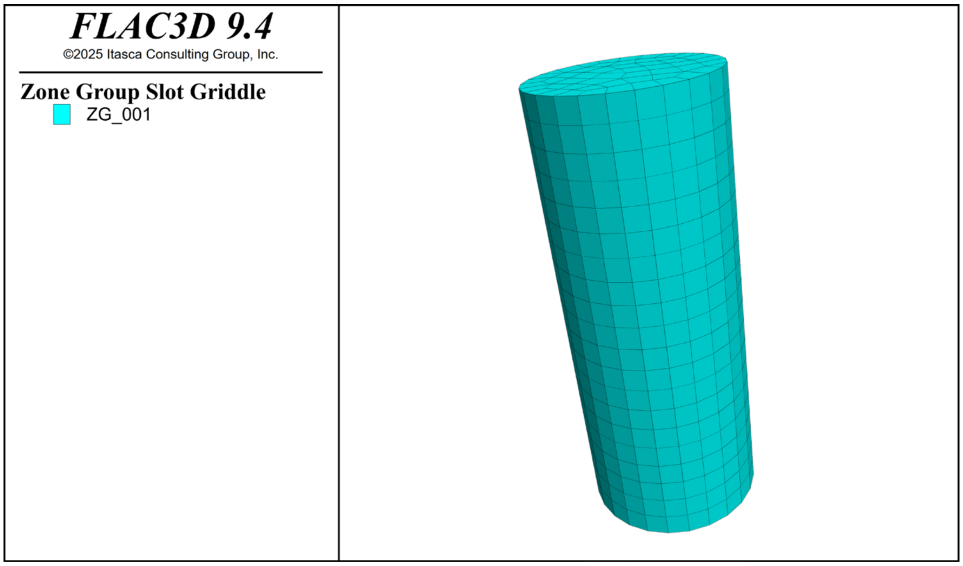


Figure 4: Volume mesh of cylinder imported into *FLAC3D*.

Although the mesh in Figure 3 and Figure 4 may resemble a structured mesh or a pure hexahedral mesh, the **GVol** output log file reports that the output mesh contains:

```
Number of elements:
  Total: 1452
  Hexahedra: 1072 (73.83% of total, 93.31% of volume)
  Prisms: 66 (4.55% of total, 3.04% of volume)
  Pyramids: 192 (13.22% of total, 2.81% of volume)
  Tetrahedra: 122 (8.40% of total, 0.84% of volume)
```

This report confirms that the output mesh is hex-dominant as it was selected in **GVol** options. Note that users may get slightly different results with regard to the number of elements of each type (this also depends on a particular version of *Griddle*).

Generating structured mesh with BlockRanger

BlockRanger is *Griddle*'s all-hex structured volume mesher, and it operates using different principles than the unstructured mesher **GVol**. As such, **BlockRanger** operates only on 4-, 5-, or 6-sided solids (not surface meshes). The cylinder created in the previous section is not such a solid as it contains only three distinct sides. Therefore, to create a structured mesh for the cylinder, the initial solid must be subdivided into several pieces suitable for **BlockRanger**. There are different ways of accomplishing this task; the most commonly used is described below.

1. Open *Rhino*, and when the **Templates** splash screen appears, select **Small Objects - Meters**. The same can be done by navigating to **File** → **New** from the *Rhino* top menu.
2. Save the *Rhino* project at a desired location.
3. Double-click on *Top* viewport to maximize it.
4. In the *Rhino* command prompt, type the following commands to create the initial curves:
   - **_Line** → *Start of line* = 1,1 → *End of line* = -1,1

- **_Line** → *Start of line* = 1,1 → *End of line* = 2,2
- **_Line** → *Start of line* = -1,1 → *End of line* = -2,2
- **_Arc** → *Center of arc* = 0 →*Start of arc* = 2,2 →*End point or angle* = -2,2
  When creating an arc, make sure that the mouse cursor stays above the previously created lines, so the arc goes through the top portion of a circle (Figure 5).

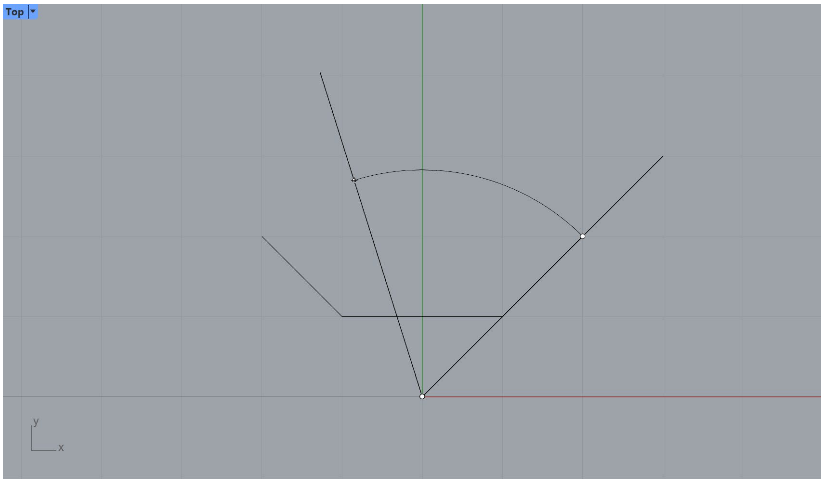


Figure 5: Creating an arc between two points.

5. Select all lines with **Ctrl**+**A** and type the **_Join** command to create a single polyline as in Figure 6.

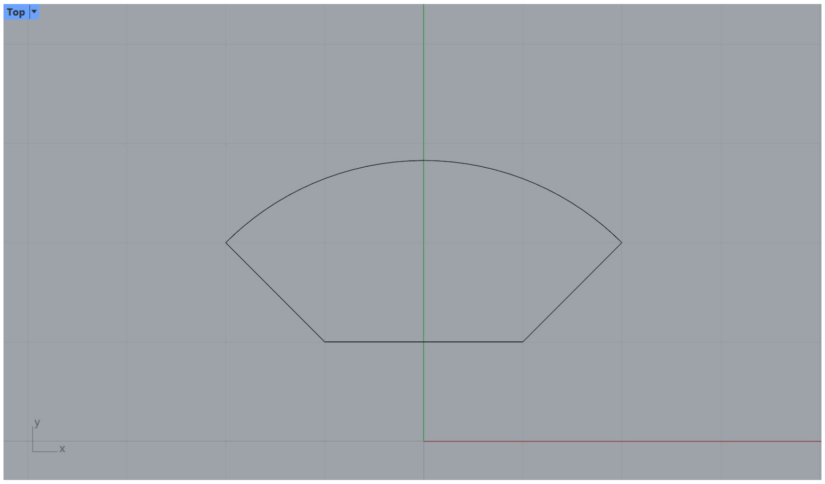


Figure 6: Single polyline consisting of three lines and an arc.

6. While the polyline is selected, type the **_Rotate** command and click on *Copy=Yes*
   - *Center of rotation* = 0
   - *Angle of first reference point* = 90 (**Enter**)
   - *Angle of first reference point* = **-**90 (**Enter**)
   - *Angle of first reference point* = 180 (**Enter**)
   - Press **Enter**
7. Type _Polyline and proceed with:
   - *Start of polyline* = 1,1
   - Next point of polyline = -1,1

- Next point of polyline = -1,-1
- Next point of polyline = 1,-1
- Next point of polyline = 1,1

8. Select all 5 objects and type **_ColorizeObjects** in the command prompt (or click on the icon in the *Griddle* toolbar). The command will assign a random color to each object (Figure 7).

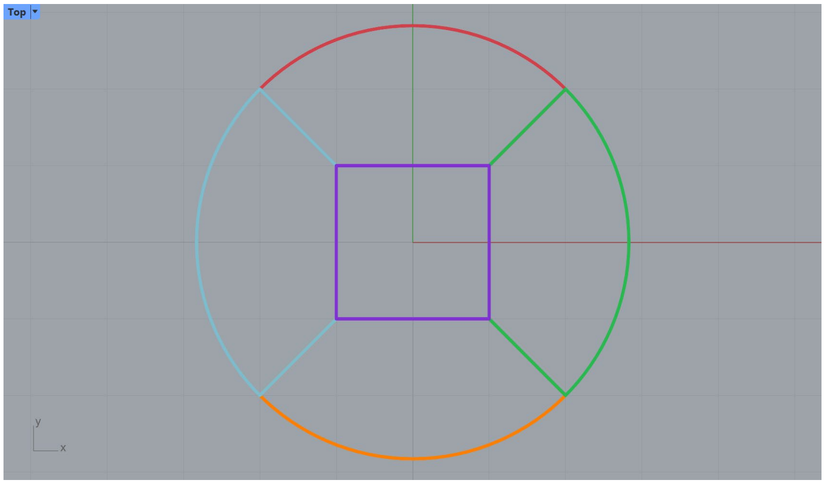


Figure 7: Five colorized polylines (some edges overlap).

9. Switch to *Perspective* viewport by clicking on the icon under the viewport.
10. Select all objects and type the **_ExtrudeCrv** command. Set *Extrusion distance* = **10** and *Solid* = **Yes**.
11. Delete the original polylines while they are selected.
12. Select all objects (should be 5 closed extrusions) and type the **_ColorizeObjects command**.

The resulting five solids represent a cylinder, but each solid is topologically equivalent to a rectangular prism. All these solids are suitable for **BlockRanger**.

13. Select all objects (or type the **_SelPolysrf** command to select only polysurfaces).
14. Type **_BR** or click on the icon in the *Griddle* toolbar and specify only *GenerateSurfaceMesh* = *BySolid* to visualize the faces on the external and internal boundaries of the solids. Set *MeshSettings* → *MaxEdgeLength* = 0.85. The output volume mesh will be outputted by default in FLAC3D binary format.

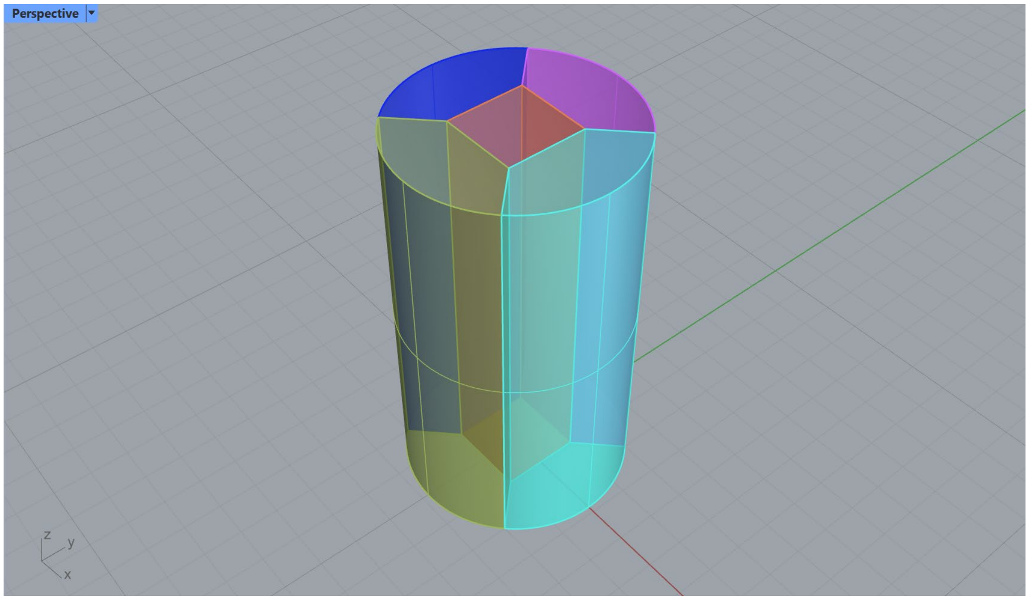


Figure 8: A cylinder subdivided into 5 solids (Ghosted view).

15. Type the **_SelPolysrf** command to select only polysurfaces and delete them.

The resulting surface mesh is provided in Figure 9 and it shows high quality quadrilateral faces on the boundaries. Note that **BlockRanger** produces a single mesh; for better representation, the mesh in Figure 9 was split into several submeshes (using **GExtract** tool) which were colorized (with **ColorizeObjects** tool).

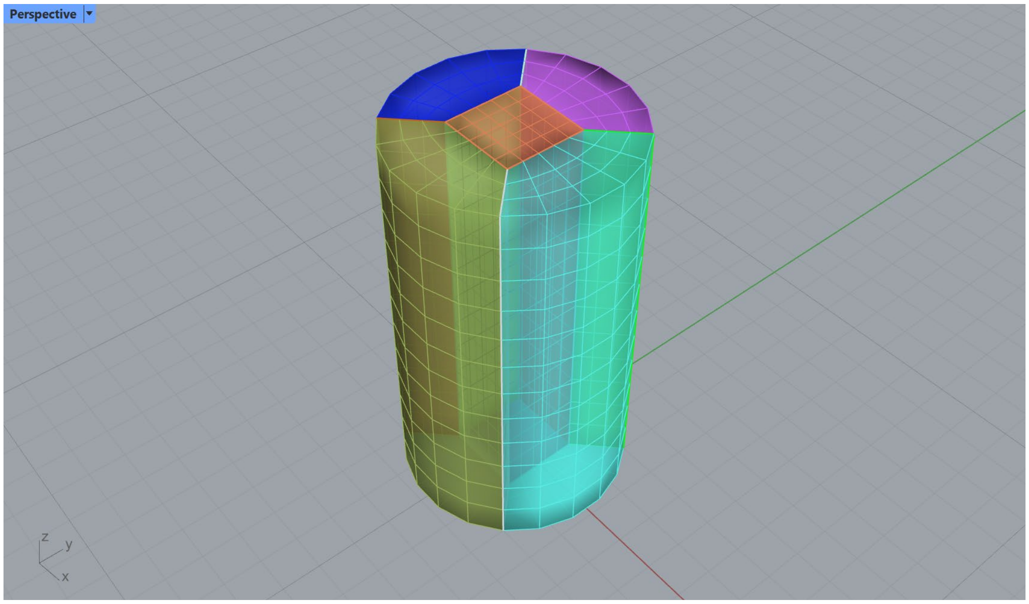


Figure 9: Surface mesh corresponding to the generated volume mesh (Ghosted view).

The output file with a volume mesh can be imported in *FLAC3D*. The zone plot will show five grouped regions composed of a structured pure hexahedral mesh (Figure 10). Note that for this simple extruded geometry, users can use the *FLAC3D Extruder* to produce similar results. However, **BlockRanger** may operate on rather complex 3D solids, which will be shown in further tutorials.

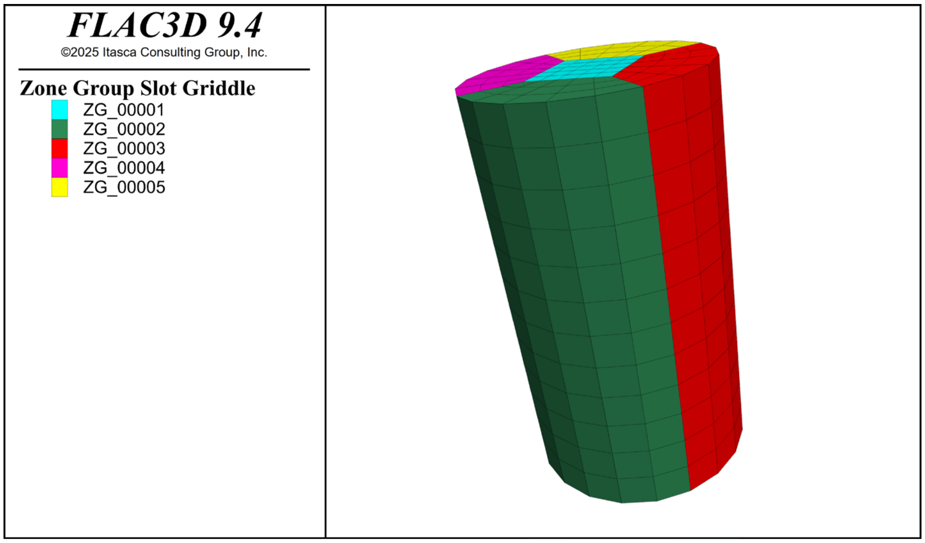


Figure 10: A cylinder subdivided into five solids.

Tutorial 2: Vertical Shaft Excavation in a Stratified Soil

Estimated work time: 0.5 – 1 hour

This tutorial shows an example of constructing the geometry and mesh for a vertical shaft (150 ft deep, diameter of 30 ft) that is to be excavated in stages. The shaft is constructed in two-layered soil with the surface separating the soil layers located at a depth of 50 ft (Figure 11). The model domain is represented as a cube 200 ft × 200 ft × 235 ft. Each excavation stage is 10 ft deep; therefore, there are 15 stages as shown by the red circular pattern in Figure 11.

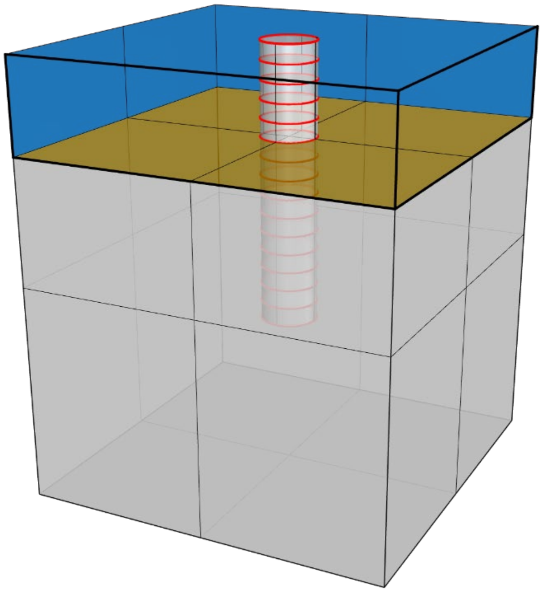


Figure 11: A general view of the model of a vertical shaft in a stratified soil.

Generation of initial geometry

1. Start *Rhino*, select **Large Objects, Feet** (similar method as step 1 in Tutorial 1) and save the *Rhino* project at a desired location.
2. Select or maximize *Perspective* viewport and switch to Shaded view.
3. Create a cylinder by typing the **_Cylinder** command. First verify Solid = **Yes** and then specify *Base of cylinder* = 0, *Radius* = 15, *End of cylinder* = -10.
4. Create a line by using the **_Line** command from 0,0,-10 to 0,0,-150.
5. Select the cylinder and type the **_ArrayCrv** command. Then select the line, click enter, and set the *Number of items* to 15 and keep O*rientation* as *Freeform*. The command will copy the initial 10 ft deep cylinder multiple times to create the desired 15 stages of excavations.
6. Type the **_SelCrv** command to select the curve and delete it.
7. Verify that 15 stages were created by selecting everything in the viewport (**Ctrl**+**A**); the command area should display the message:
   ```
   15 extrusions added to selection.
   ```
8. Create three points that will define the plane separating two layers of soil by using the **_Point** command. For the coordinates, specify:
   - -100,-100,-50

- 100,-100,-50
- 100,100,-50

9. Connect all points by a horizontal plane by typing the **_Plane** command and start dragging a plane from one of the "side" points (the first or third in the list above). The objects should look as shown in Figure 12.

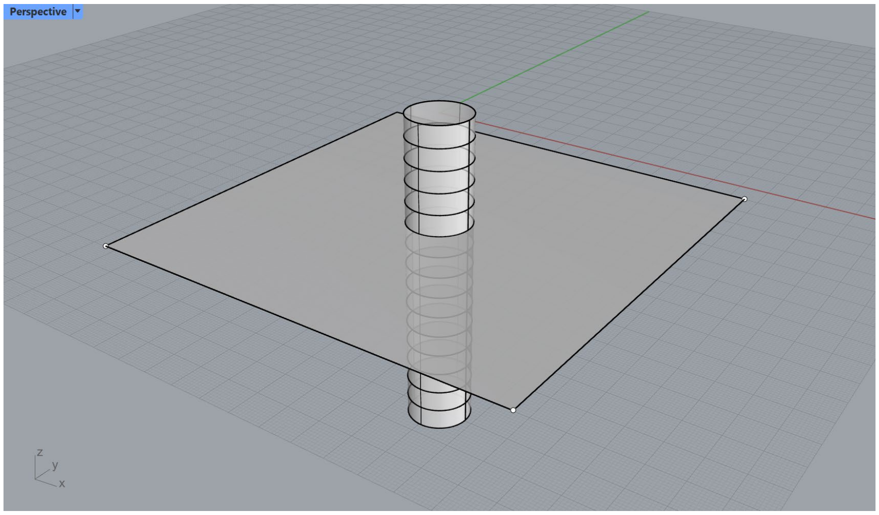


Figure 12: Construction of a shaft and a plane separating two layers of soil (Ghosted view).

10. Select and delete points using the **_SelPt** command.
11. Create a domain box by typing the **_Box** command (or select in the menu **Solid Tools** → **Box**) and click on the *Diagonal* option to define the box by 2 points. Enter **-100,-100,-235** for the coordinates of the first corner and **100,100,0** for the second corner.
12. Switch to Wireframe view and zoom to all extents if needed (**Alt**+**Ctrl**+**E**).
13. Select all objects and type the command **_NonManifoldMerge** (or click on the icon in the *Griddle* toolbar) to create a single non-manifold polysurface. This complex polysurface will serve as the initial geometry for meshing and is shown in Figure 13.

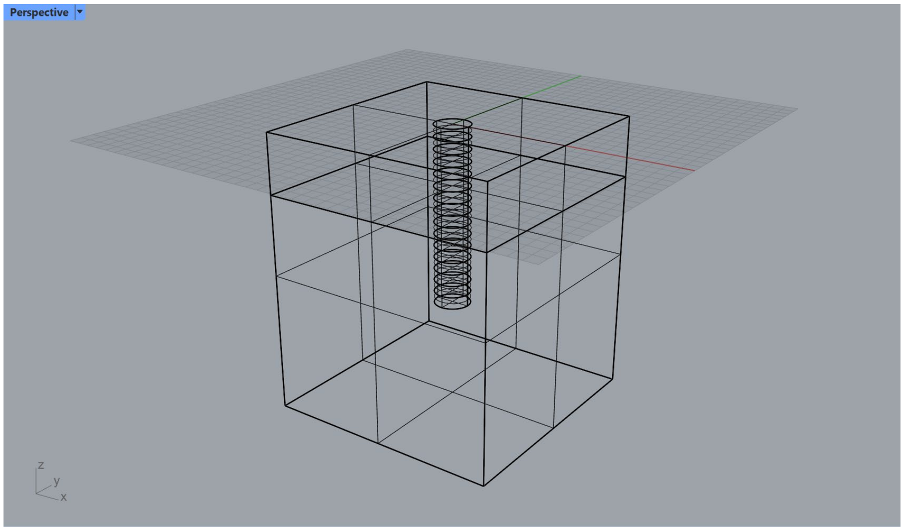
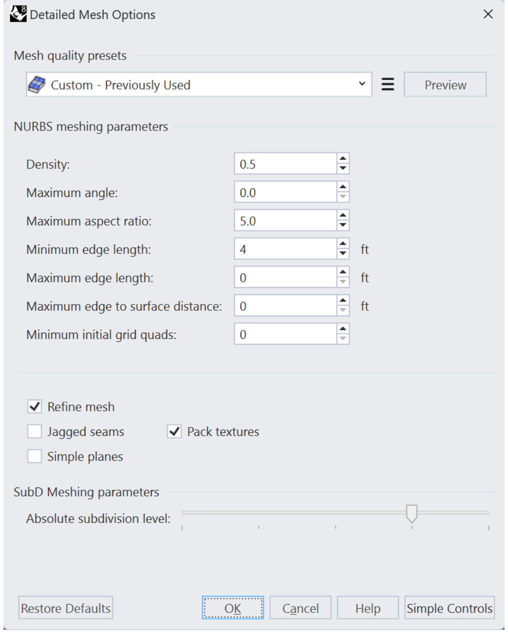


Figure 13: Complete initial geometry of the model of a shaft and two soil layers (Wireframe view).

Meshing with unstructured mesh

The geometry created in the previous step is ready for meshing.

14. Select the polysurface and type the command **_Mesh**. If a simplified dialog appears, click on the *Detailed Controls*… button and enter the values as shown in Figure 14.



Mesh quality presets

Custom - Previously Used

NURBS meshing parameters

Density:	0.5	
Maximum angle:	0.0	
Maximum aspect ratio:	5.0	
Minimum edge length:	4	ft
Maximum edge length:	0	ft
Maximum edge to surface distance:	0	ft
Minimum initial grid quads:	0	

☑ Refine mesh
☐ Jagged seams ☑ Pack textures
☐ Simple planes

SubD Meshing parameters

Absolute subdivision level:

Figure 14: *Rhino* meshing dialog to create an initial mesh.

15. While the polysurface is still selected, hide it with the **_Hide** command. If it is not selected, select it with the **_SelPolysrf** command. Only the mesh should be visible at this point.
16. Select the surface mesh and type **_GSurf** or click on the icon in the *Griddle* toolbar to remesh the initial triangular mesh**.** Select *Mode* = *AllQuad*, *MinEdgeLength* = 4, *MaxEdgeLength* = 17 and keep all other parameters at the default values. Switch back to Ghosted view and the resulting mesh should match the mesh shown in Figure 15 (note that mesh the user obtains may look slightly different on the top and middle surfaces).

17. Now the model is ready for volume meshing. To show how the volume meshing parameters influence the final mesh, two cases are considered here.

    - Select the surface mesh and type **_GVol** or click on the icon in the *Griddle* toolbar. Select *MeshSettings* → *Mode=HexDom*, leave all other parameters at the default settings and press **Enter** twice. Keep the output format as *FLAC3D* Binary.

      The resulting *FLAC3D* grid is shown in Figure 16. The grid is of good quality, but the gradation of zones between the shaft and the boundary can be slightly improved (big improvement are not possible as the domain boundaries are located close to the shaft).

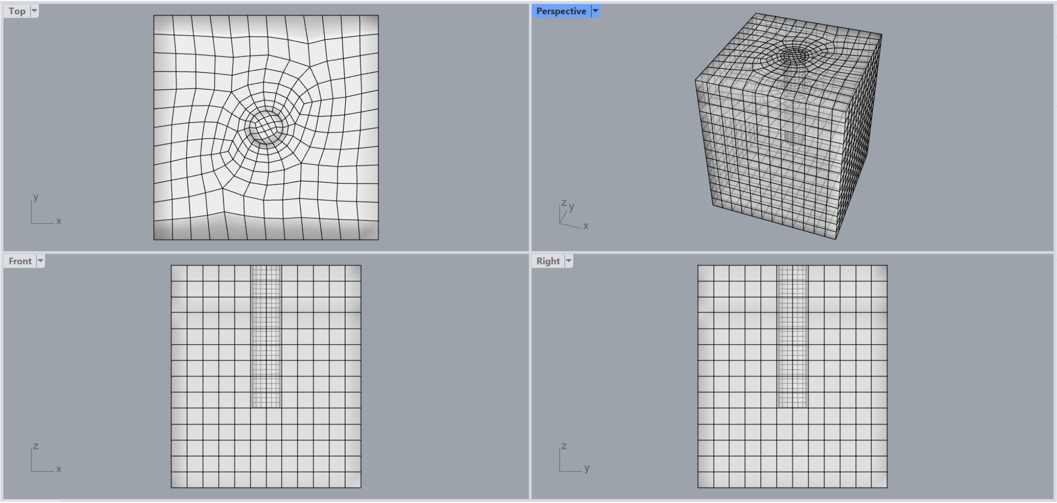


Figure 15: Remeshed model geometry using GSurf with *QuadDom* mode.

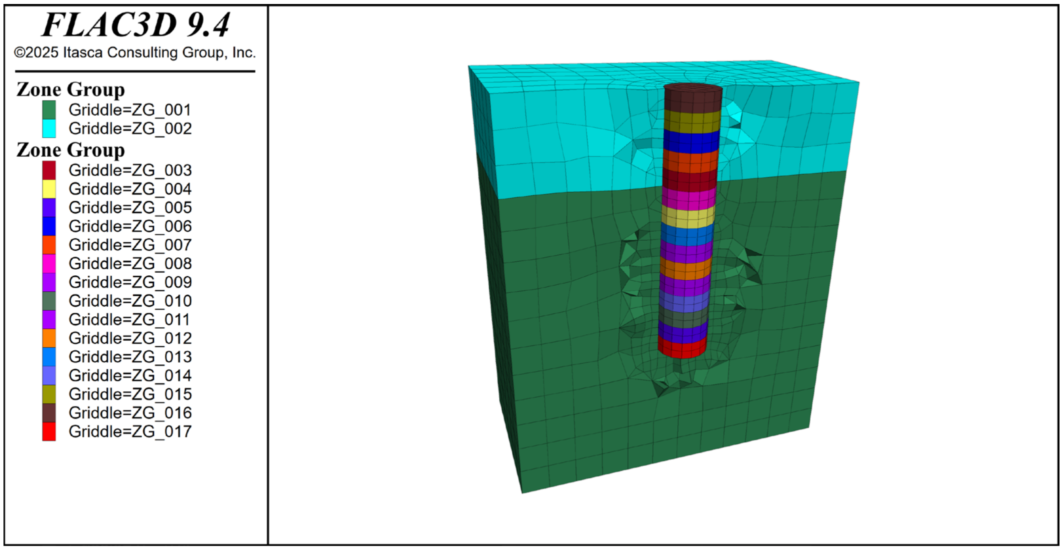


Figure 16: *FLAC3D* grid of the shaft model when using default GVol parameters.

- Create a new volume mesh using the following **GVol** parameters:
  MeshSettings → Mode = *HexDom, MaxGradation* = 10*, TargetSize* = 7*, Optimization* = 10, *ShapeQuality* = 0.6. For the explanation of each parameter, refer to the *Griddle 3.0 User Manual*. The resulting mesh is shown in Figure 17.

  The zones surrounding the shaft in Figure 17 transition from coarse to finer size in more gradual manner. Typically, such results are achieved when *MaxGradation* is set to a small value (< 0.5), but in this case, as the domain boundaries are relatively close to the shaft, a larger value of *MaxGradation* is used to force the mesher to quickly change from a large zone size on the boundary of the domain to a smaller size specified by the *TargetSize*, and then to the zone sizes on the boundary of the shaft. Use of the highest possible values for *Optimization* and *ShapeQuality* allows for improving zone quality. An alternative way to control volume mesh density is to create hard nodes (as points in *Rhino*) at desired model locations and assign an element size to them (via the hyperlink field). The volume element size around the hard nodes will be close to the specified values.

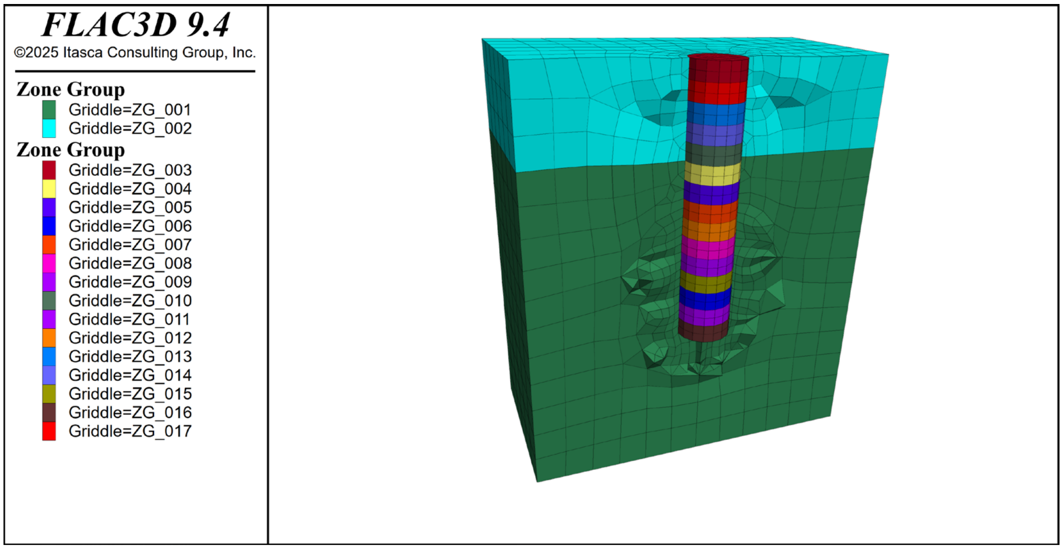


Figure 17: *FLAC3D* grid of the shaft model when using custom GVol parameters.

Excavation of the shaft can be modeled by gradual removal of the stages inside the shaft (i.e., zone groups ZG_003 – ZG_017 as in Figure 16 and Figure 17).

Tutorial 3: Meshing a Model Containing Surfaces and Polysurfaces

Estimated work time: 1 hour

This tutorial describes two meshing approaches for a model composed of analytic and freeform shapes, such as *Rhino* (poly)surfaces and solids defined by BRep (boundary representation), NURBS (non-uniform rational basis spline), or SubD (high-precision Catmull Clark subdivision surfaces).

1. Open project “T3_BridgeSupport.3dm” located in folder “TutorialExamples\3_BridgeSupport”. Create a copy of this initial project with a different name at a desired location (use **File → Save As**).
2. The model represents a part of a complex bridge support structure as shown in Figure 18. The model consists of 13 surfaces, seven polysurfaces, and two extrusions represented by BRep objects. Note how the objects are split by each other (e.g., the top sub-structure surface is split into three pieces); this is done for better meshing results.

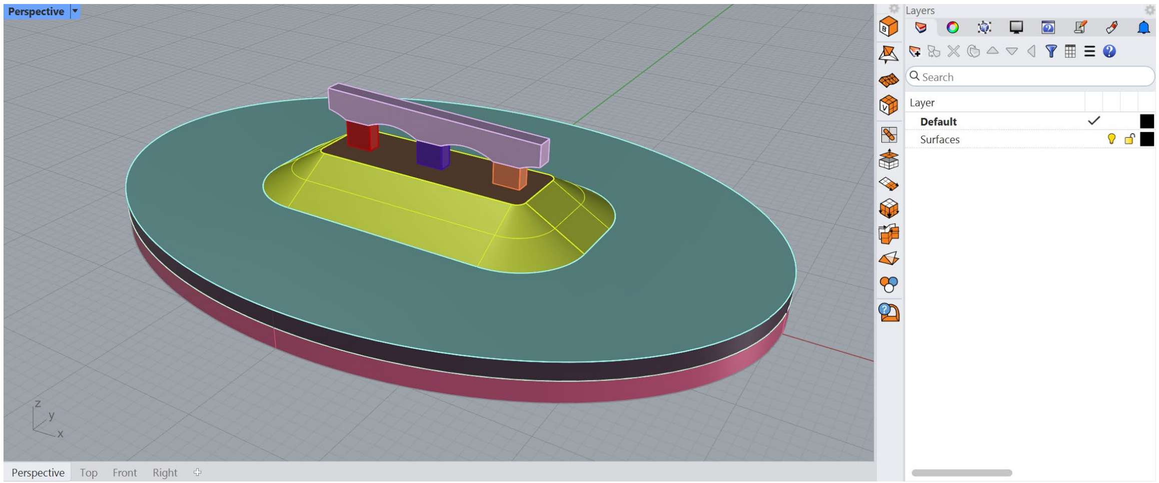


Figure 18: Model of a bridge support structure.

The typical approach to construct meshes when starting with initial (poly)surfaces such as in Figure 18 consists of the following:[1]

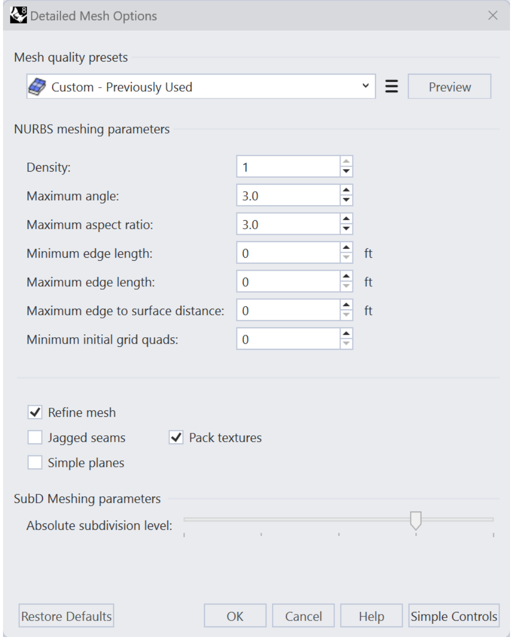
- Mesh (triangulate) all the (poly)surfaces to create initial rough meshes; if initial meshes are provided, this step is skipped.
- Intersect the initial meshes with *Griddle*’s surface mesh intersector **GInt**.
- Remesh all meshes with surface remesher **GSurf**.
- Create volume mesh with unstructured volume mesher **GVol**.

Users should follow this general approach in the majority of cases. However, in certain situations, a slight variation of this approach may be more efficient in preparing the model for volume meshing. The standard approach is described in the first sub-section of this tutorial, and potential issues are outlined. The second sub-section presents a variation of the approach, which can be applied to models similar to the one described in this Tutorial.

[1] See also *Griddle 3.0 User Manual*, Section “*Using Griddle Tools for Unstructured Meshing*”.

Mesh generation using GInt and GSurf (approach 1)

3. Maximize *Perspective* viewport and switch to Shaded view for better visibility.
4. Navigate to the *Layers* panel and create a new layer by clicking the button; name this layer “Meshes_Approach1”. If the panel is not visible, it can be opened from **Edit** → **Layers** → **Edit Layers…**.
5. Select all objects (**Ctrl**+**A**) and type the **_Mesh** command to create the initial mesh of all objects. Use detailed controls with the following settings to construct a fine initial mesh (Figure 19).



Detailed Mesh Options

Mesh quality presets

Custom - Previously Used

NURBS meshing parameters

Density:	1	
Maximum angle:	3.0	
Maximum aspect ratio:	3.0	
Minimum edge length:	0	ft
Maximum edge length:	0	ft
Maximum edge to surface distance:	0	ft
Minimum initial grid quads:	0	

☑ Refine mesh

☐ Jagged seams ☑ Pack textures

☐ Simple planes

SubD Meshing parameters

Absolute subdivision level:

Figure 19: *Rhino* meshing dialog to create fine initial mesh.

6. While all the (poly)surfaces are selected, invert the selection by typing the command **_Invert**, which will select meshes only. Navigate to the *Properties* panel (or press **F3**) and change the selected objects layer to “Meshes_Approach1”. Open the *Layers* panel and turn off the “Surfaces” layer by clicking the 💡 button.

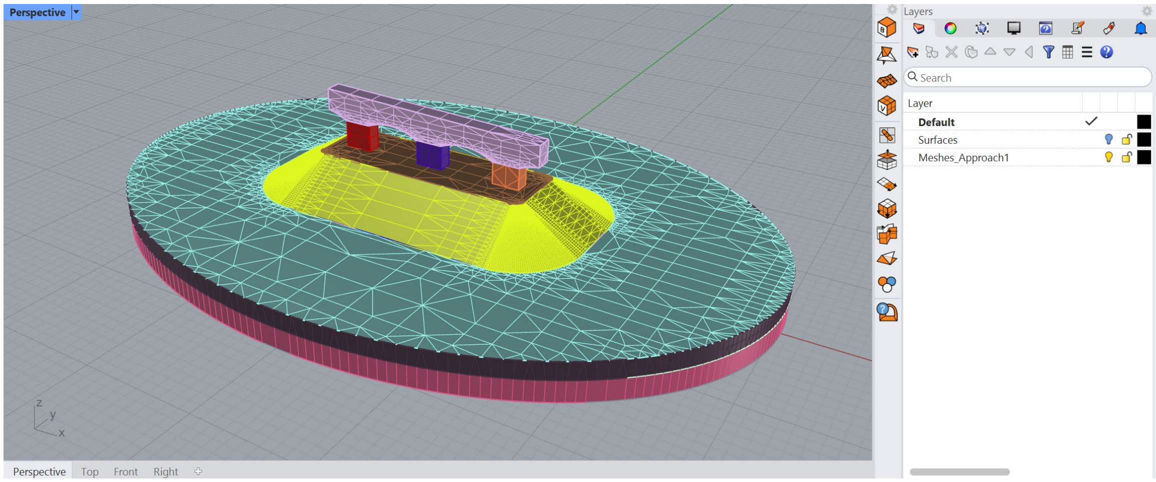
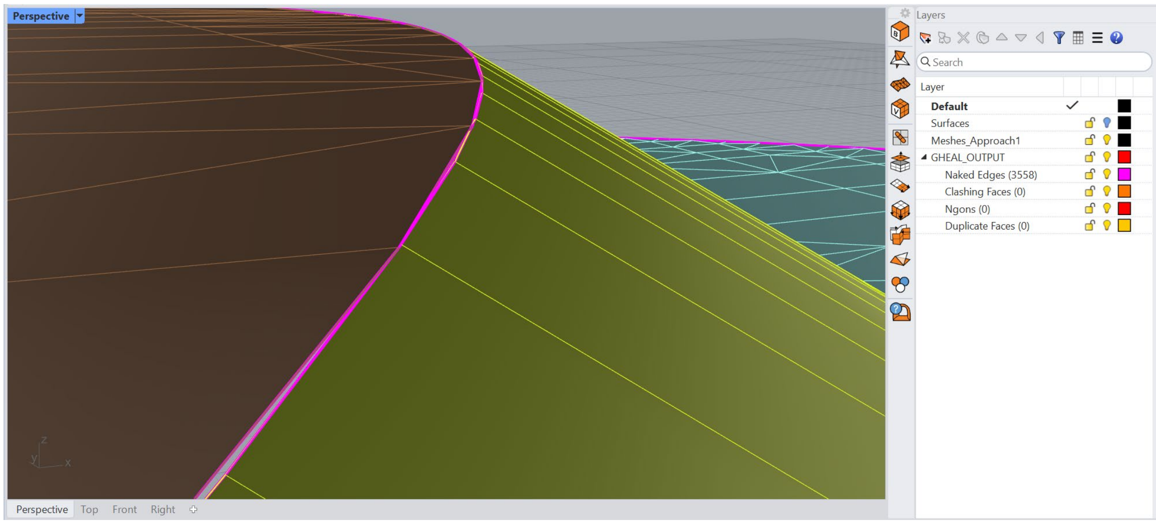


Figure 20: The initial *Rhino* mesh.

The mesh in Figure 20 is not conformal and contains gaps along the curved boundaries (Figure 21). This is due to the fact that surface meshes are a discrete representation of the initial analytic (in this case, BRep) surfaces, and the mesh for each object is created independently from other objects.



Layer
Default ✓
Surfaces
Meshes_Approach1
GHEAL_OUTPUT
Naked Edges (3558)
Clashing Faces (0)
Ngons (0)
Duplicate Faces (0)

Figure 21: Mismatching mesh boundaries in the initial *Rhino* mesh.

The gaps and non-conformal faces can be fixed by intersecting meshes using the **GInt** surface mesh intersector with properly selected tolerance. This will help creating a watertight surface mesh domain which is a required condition for volume meshing. In certain cases, however, it may be challenging to determine a proper tolerance value when intersecting many meshes. This is especially true for big models with large variations in face (element) sizes and with gaps/mismatches between neighboring faces that are of similar size as some of the mesh faces. The reason for this challenge is that a small **GInt** tolerance may not be sufficient to close the gaps and intersect all the faces to make them conformal, while a large tolerance may create undesired intersections and changes of smaller faces that fall within the tolerance. Therefore, it is recommended to do mesh intersections with minimal acceptable tolerance to avoid creating incorrect intersections. Visualization of the intersection can assist with proper tolerance selection, and users may need to try several tolerance values starting with 0 and gradually increasing it until all faces appear highlighted/intersected. This process is outlined below.

7. Type **_SelMesh** to select all surface meshes and type **_GInt** or click on the icon in the *Griddle* toolbar. For the first trial, keep *Tolerance* = 0, leave the *AdvancedParameters* as their default values, and wait for the preview to appear[2]. This preview (shown in Figure 22) identifies faces that are within tolerance and will be intersected. In this case, there are obvious faces that remain un-highlighted, which means that the tolerance is insufficient to intersect these faces. Adjust the value to *Tolerance* = 0.005, which now shows the faces highlighted, and press **Enter**.

[2] If the input meshes contain more triangular faces (quads = 2 triangles) than the limit specified in *PreviewLimit*, the preview is disabled. This value should be chosen based on the user's computers performance and will be saved in the system's registry.

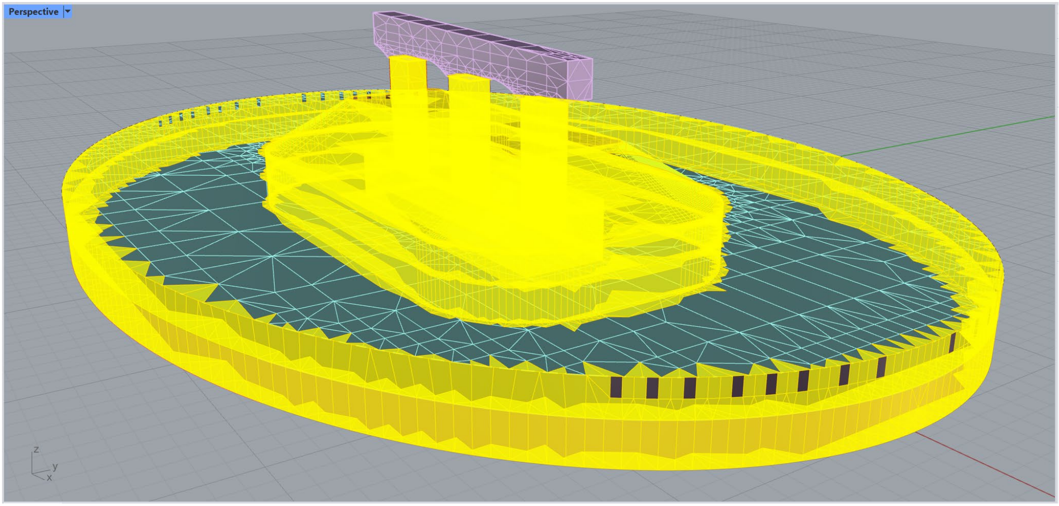


Figure 22: GInt's preview with *Tolerance* = 0. Some faces are visibly not highlighted among the surrounding highlighted faces, which is an indication of an insufficient tolerance.

8. To further check if there are any gaps or non-conformal edges/faces left, we can join all 22 meshes and check their connections. To do this, select the resulting meshes using **_SelMesh** and type **_Join** to create a single mesh. Then, check the connections for non-conformal edges by typing **_GHeal** or by clicking on the 🩹 icon in the *Griddle* toolbar. Select the *ShowErrors* option, which creates an output layer containing potential issues in the input mesh. The result – pink outlines, shows that there are still some non-conformal faces (Figure 23). These are likely small, hard to see areas that went unnoticed while looking over the preview.

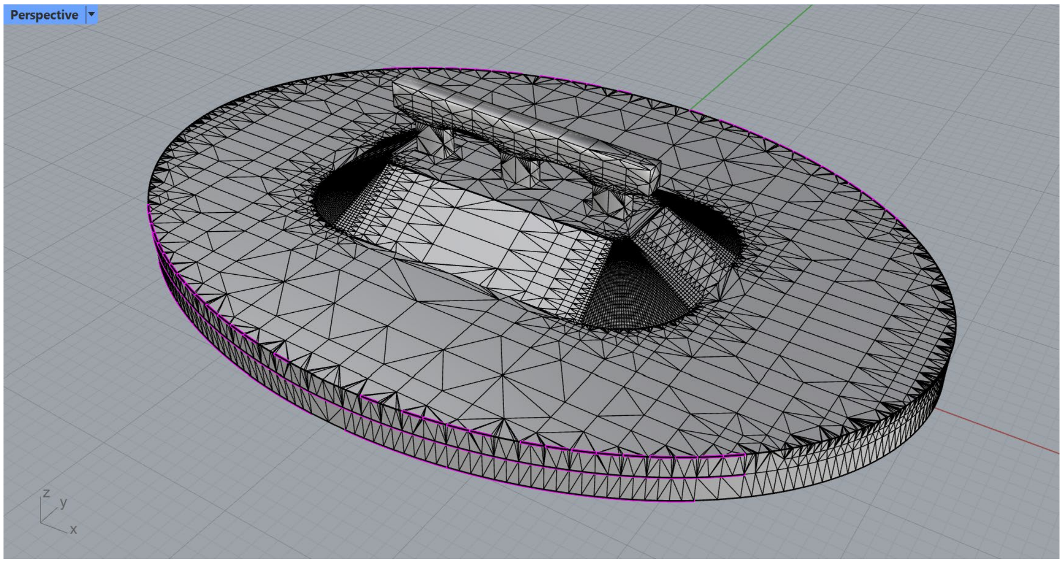


Figure 23: Naked edges (in pink) after using GInt with *Tolerance* = 0.005.

9. Undo the last three commands (**_Join**, **_GHeal** and **_GInt**) by pressing **Ctrl**+**Z** three times. Ensure the initial **_Mesh** command is not undone.

10. Repeat step 7 using *Tolerance* = 0.01 and step 8. The output mesh now doesn't contain any naked edges. Undo the last operations (**_GHeal** and **_Join**) by pressing **Ctrl+Z**.
11. Select all surface meshes and remesh with **_GSurf** ( ) using the following parameters: *Mode* = *QuadDom*, *MinEdgeLength* = 0.25, *MaxEdgeLength* = 5, *RidgeAngle* = 5, *AdvancedParameters* → *MaxGradation* = 0.06, and keep all other parameters at the default values. The resulting meshes are shown in Figure 24. These meshes look good overall, but zoomed views of the curved bottom / top embankment sections reveal (yellow-lightblue / yellow-brown meshes) that these areas contain small elements (faces) that do not fully conform to the initial shape (Figure 25). This issue is the result of independent meshing of the initial BRep surfaces and also due to using a relatively large **GInt** tolerance[3].

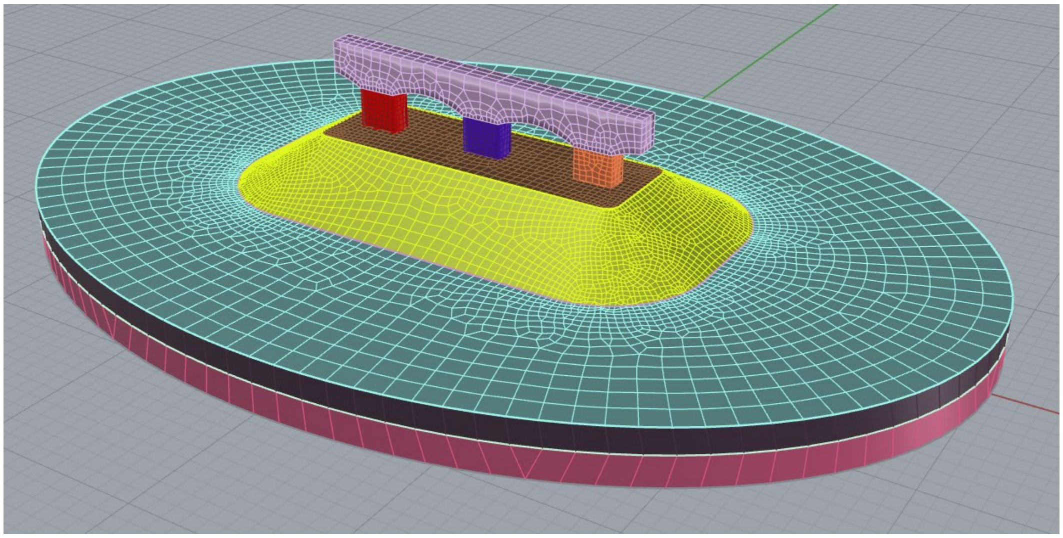


Figure 24: Remeshed surface meshes using standard approach.

Create a volume mesh using **GVol** ( ) and any desired (or default) parameters. **GVol** automatically checks input surface meshes for the presence of non-conformal faces and the process will be cancelled if any are found. It also checks input surfaces for naked edges (e.g., holes, gaps) and Ngons; if any such issues are found, an information message is shown if *ShowWarnings* is set. The message does not prevent **GVol** from running (click **Yes** to proceed or **No** to stop); it is only a reminder for users to verify that input meshes are valid. In this case, **GVol** input consists of multiple surface meshes with boundaries treated as naked edges.

[3] Another potential issue can be revealed by redoing step 11 with **GSurf** → *AdvancedParameters* → *Output* = *Merged*. The single output mesh may contain a few "clashing" faces that were introduced by **GSurf**. In this case, **GSurf** created some very small faces which fall under the model tolerance causing them to be considered "clashing" (with surrounding faces). Clashing faces are not always an indication of a geometry error, and in this case, they do not affect the ability to create a volume mesh. However, it is always recommended to fix them as much as possible (use **GHeal**'s *FixClashingFaces* function), as having very small faces in the model or nearly-intersecting / overlapping faces, would produce very small or very bad quality elements in volume mesh.

The initial error check and the message can be changed or turned off by adjusting *ShowWarnings* parameter (in **GVol**'s *MeshSettings* options). *Note that the presence of non-conformal faces will always prompt an error.*

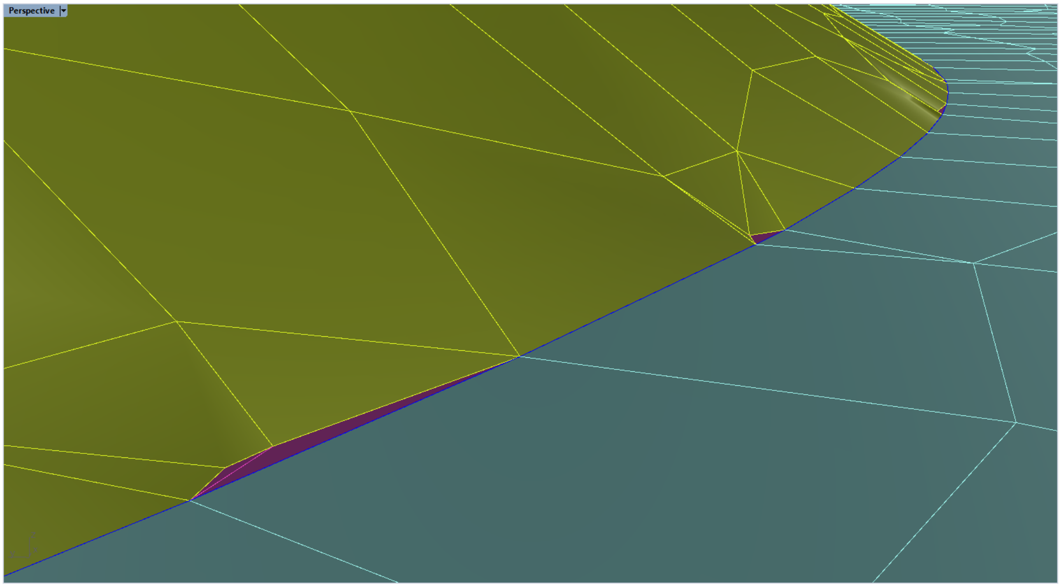


Figure 25: Slightly deformed surface meshes with approach 1.

Mesh generation using NonManifoldMerge and GSurf (approach 2)

The issues in the previous approach can be avoided if all initial (poly)surfaces are merged into a single non-manifold polysurface before meshing. For such polysurfaces, the *Rhino*'s **_Mesh** command does initial meshing in a conformal manner, and no gaps are created between mesh faces[4]. Therefore, with this approach, surface mesh intersection with **GInt** is often not needed. The drawback of this approach is that when the initial objects are merged into a single non-manifold polysurface, user-specified object information is lost (e.g., surface names, colors, etc.) However, if these parameters are not important, the approach described below may be easier, more efficient, and yield better quality meshes.

12. Navigate to the *Layers* panel and create a new layer named "Meshes_Approach2". Turn on layer "Surfaces" to show the initial surfaces.
    Delete layers (if any are present) "GHEAL_OUTPUT" and "GVOL_OUTPUT".
13. Select all objects in layer "Surfaces" (as in Figure 18) and use the tool **NonManifoldMerge** ( ) to create a single non-manifold polysurface (Figure 26).
14. Mesh the polysurface using the **_Mesh** command with the same settings as in Figure 19. Note that the default mesh settings and *Simple Controls* can be used in this case, as there will be no issues with gaps or non-conformal faces (there is no need to have a highly detailed initial mesh).

[4] For the approach to work properly, the initial surfaces must connect between the boundaries and not intersect one another.

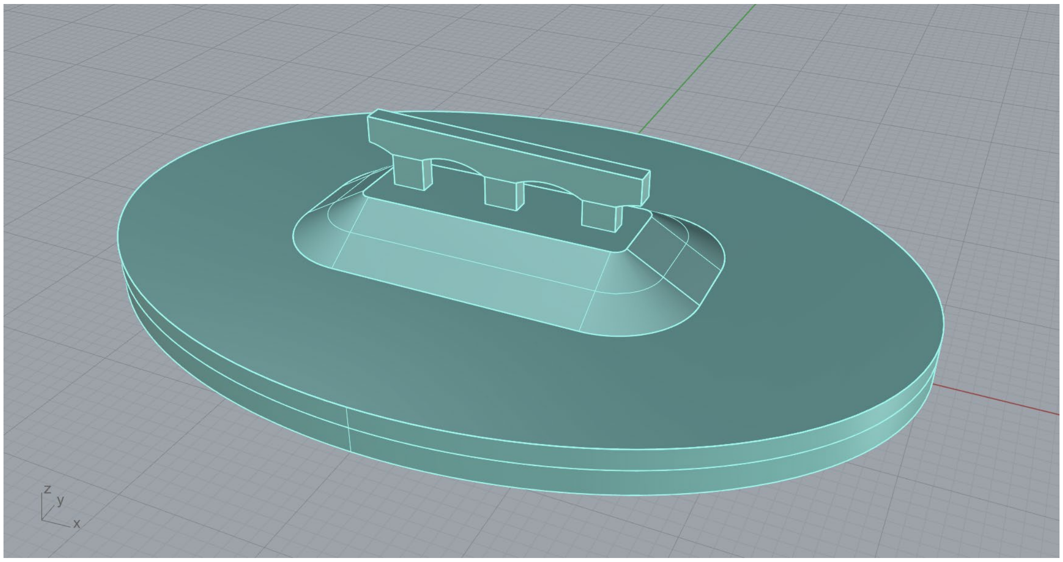


Figure 26: Single polysurface constructed by the NonManifoldMerge tool.

15. Assign layer “Meshes_Approach2” to the newly created mesh and turn off the “Surface” layer so only the mesh is present in the viewports.
16. Remesh the initial mesh with **GSurf** ( ) using the same settings as in Step 11. The quality of elements (faces) around the curved parts of the embankment is much higher if compared with the mesh in Figure 24 (or with the meshes in layer “Meshes_Approach1”).

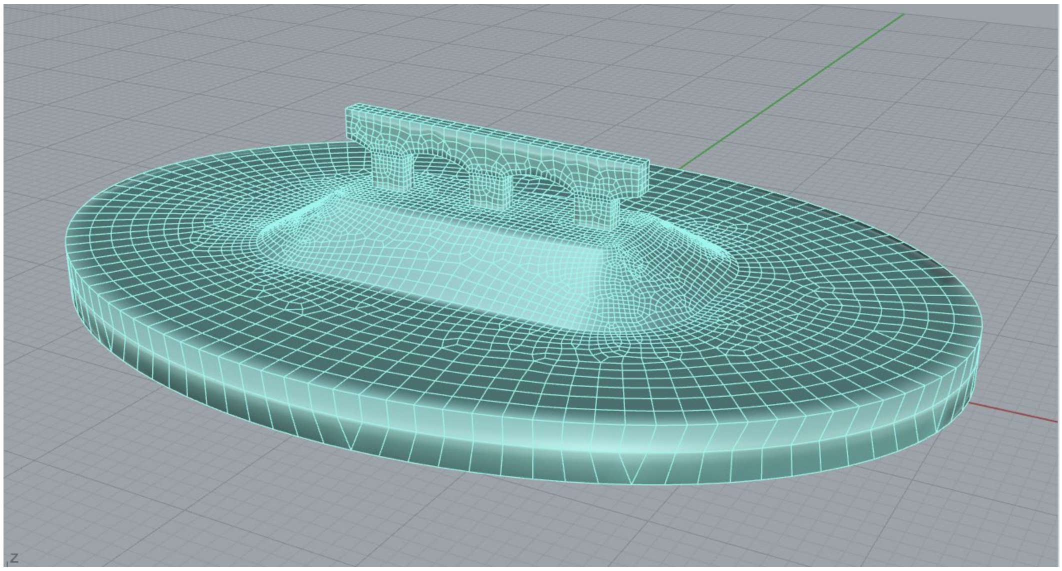


Figure 27: Remeshed surface mesh in the second approach.

17. Verify that there are no naked edges (gaps, holes) or clashing faces in the final surface mesh using **GHeal** ( ) → *ShowErrors*. There should be no curves outlining any problems in the “GHEAL_OUTPUT” layer.
18. The mesh is ready for volume meshing. Run **GVol** ( ) with any desired (or default) parameters. Due to the higher quality of input surface mesh(es), the total number of elements in the output

volume mesh is smaller in this approach than in the previous one (e.g., for Hex-dominant mesh, this approach produces in total ~41,000 elements and the first approach results in ~72,000 elements).

As mentioned previously, the issue of this approach is that all initial (poly)surfaces must be merged into a single nonmanifold polysurface before meshing. If there is a need to assign a name, color, or any other property to a specific surface mesh before volume meshing, the merged surface mesh can be easily split into sub-meshes using *Griddle*'s tool **GExtract**.

19. Select the merged surface mesh (as in Figure 27) and type the **_GExtract** command or click on the icon in the *Griddle* toolbar. Select option *Multiple* and use *MaxBreakAngle* = 180 to separate meshes only along non-manifold edges (the break angle will not have an effect as it is set to the highest possible value of 180°). Alternatively, if break angle = 89° is used, all meshes that connect at an angle > 89° will also be separated (thus, meshes that are perpendicular to each other will be separated).
20. Select all (36) meshes and use *Griddle*'s command **_ColorizeObjects** or click on the icon in the *Griddle* toolbar. This will assign a unique color to each *Rhino* object (Figure 28).
21. Now a name can be assigned to each surface mesh if there is a need to transfer surface mesh names into a volume mesh.

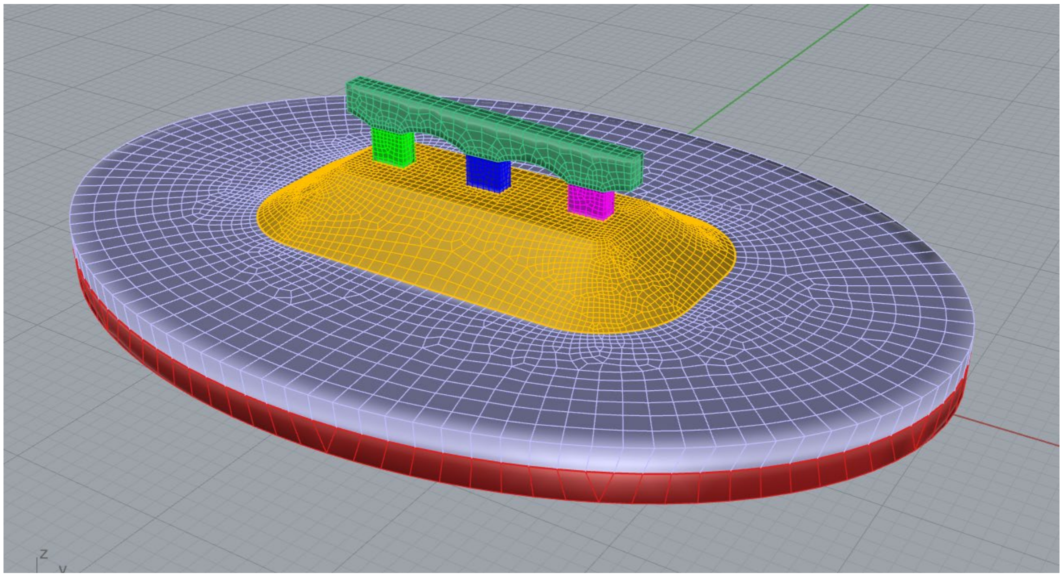


Figure 28: Mesh from Figure 27 "exploded" into sub-meshes, which are subsequently colorized.

Tutorial 4: Creating a Structured Mesh with BlockRanger

Estimated work time: 1–2 hours

This tutorial provides detailed information on how to build a block-structured hexahedral mesh from an initial CAD model provided by a DXF file. To create a mesh using **BlockRanger** (**_BR** command), an assembly of six-, five-, or four-sided *Rhino* solids that conform to the reference model must be created. The creation of such an assembly is the objective of this example. The reference model is shown in the left side of Figure 29. The final hexahedral mesh is depicted to the right. This 3D slope model has a shape similar to a bathtub; hence, it is sometimes referred to as the "bathtub" model.

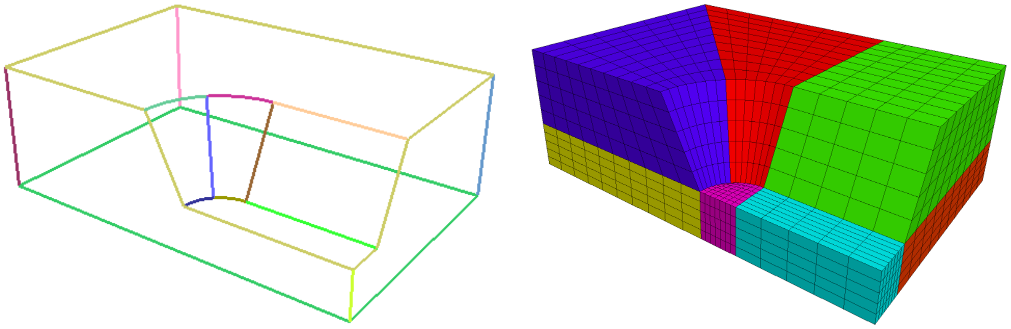


Figure 29: Reference model (left) and BlockRanger generated mesh (right).

Importing the geometry

1. Start *Rhino*, select **Large Objects, Meters** (similar method as step 1 in Tutorial 1) and save the *Rhino* project at a desired location.
2. Import the initial geometry from a DXF file by navigating to **File** → **Import** and select "T4_3DSlope.dxf" located in folder "TutorialExamples\4_StructuredMesh_SlopeModel". In the **Import** dialog, select **Model** → **Meters** and **Layout Units** → **Millimeters**, and keep other parameters at the default values.
3. Maximize *Perspective* viewport and make sure Shaded view is selected.
4. In the top menu, navigate to **View** → **Display Options…** This will open the **Rhino Options** dialog; select **View** → **Display Modes** →**Shaded** → **Objects** → **Curves** and set **Curve Width** to 4. This will display all curves thicker in the Shaded view. The initial settings can be always restored by clicking on **Restore Defaults**.
5. Navigate to the *Layers* panel as shown in Figure 30. If the panel is not visible, open it by going to **Edit** → **Layers** → **Edit Layers…** In the *Layers* panel, rename the automatically created layer "lines" to "Reference" and delete all layers except "Default" and "Reference". Ensure that all objects are now in layer "Reference" by selecting all objects and navigating to the *Properties* panel (or press **F3**).
6. Select all objects and type the command **_Move**. For the *Point to move from,* select the bottom-left corner of the model and for the *Point to move to*, type 0, and press Enter. This will move all objects and align the bottom-left corner of the model with zero coordinates. Compare the results with Figure 30.

It is always recommended to move/center objects imported from a DXF (or other formats) closer to zero coordinates as it improves the accuracy of *Rhino* operations.

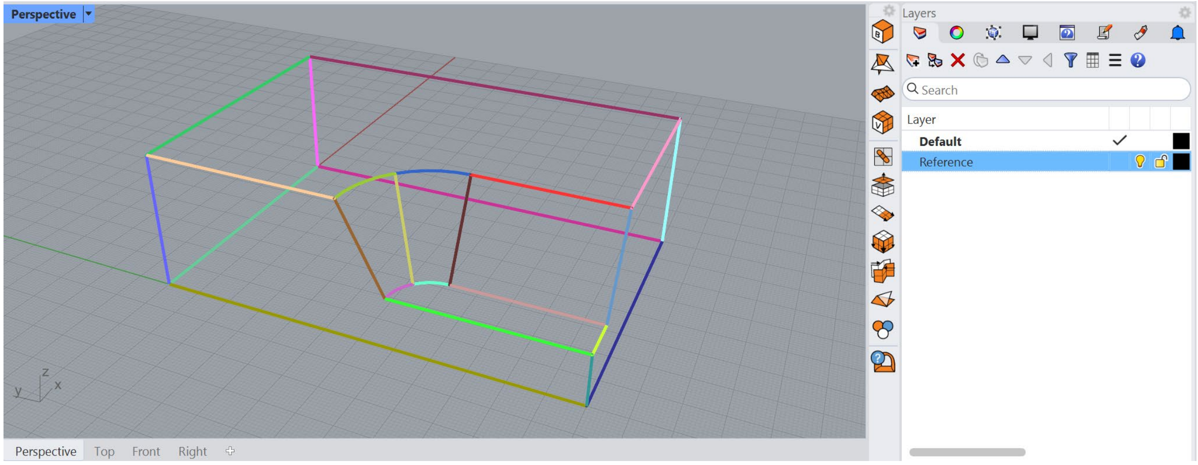


Figure 30: The reference model.

Choosing the right mesh block layout for the model

BlockRanger operates on solids and meshes each solid block individually. The present model may be decomposed into solid blocks in many ways. Some of the possible options are shown in Figure 31 through Figure 34. The arrangement of blocks depicted in Figure 33 will be used in this tutorial.

Each of the blocks shown in Figure 31 through Figure 34 satisfy **BlockRanger** requirements for solids:

- Permissible types of solids:
    - four-sided solids made of three-sided faces (a topological tetrahedron),
    - five-sided solids made of 2 three-sided faces and 3 four-sided faces (a topological triangular prism), and
    - six-sided solids made of four-sided faces (a topological hexahedron).
- Faces must be simple faces that cannot be further "exploded" into simpler faces.
- Face edges must only be simple curves that cannot be further "exploded" into simpler curves.

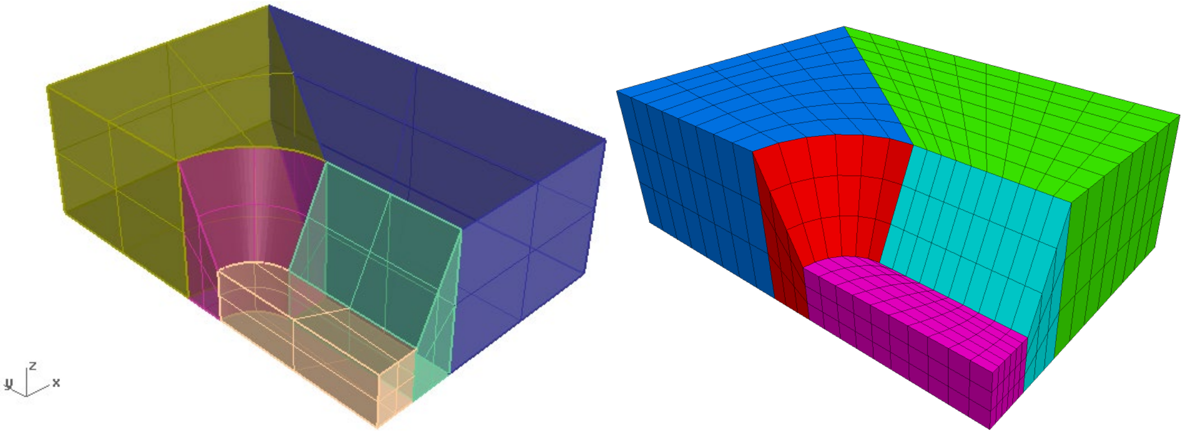


Figure 31: Solid layout resulting in five mesh blocks.

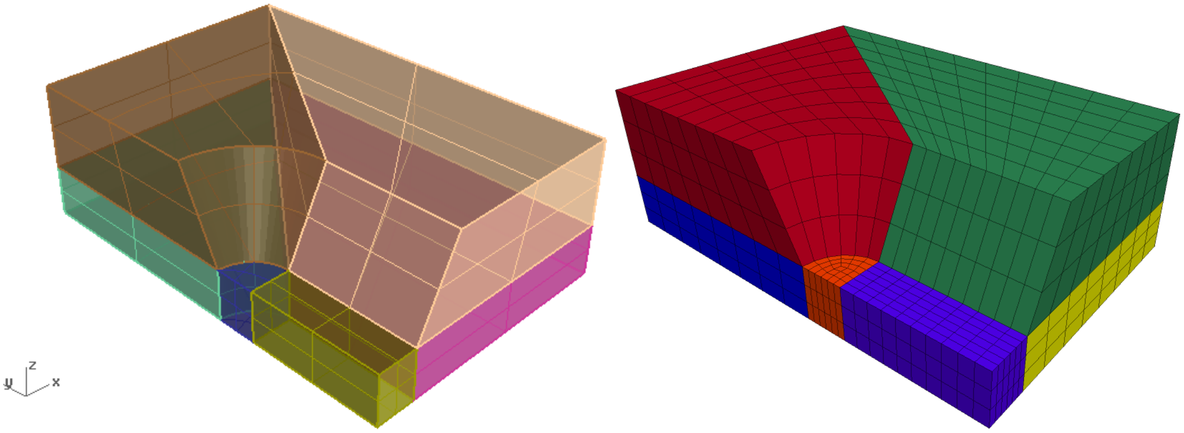


Figure 32: Solid layout resulting in six mesh blocks.

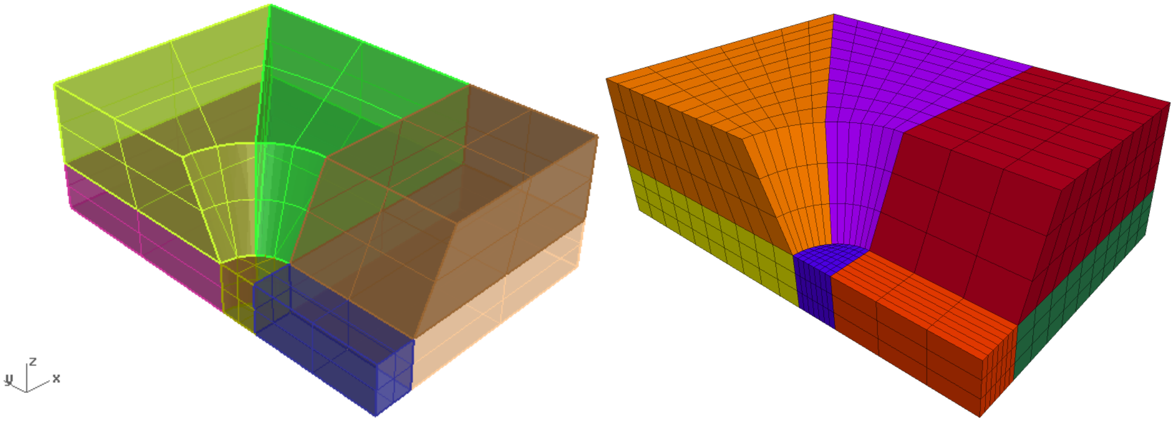


Figure 33: Solid layout resulting in eight mesh blocks.

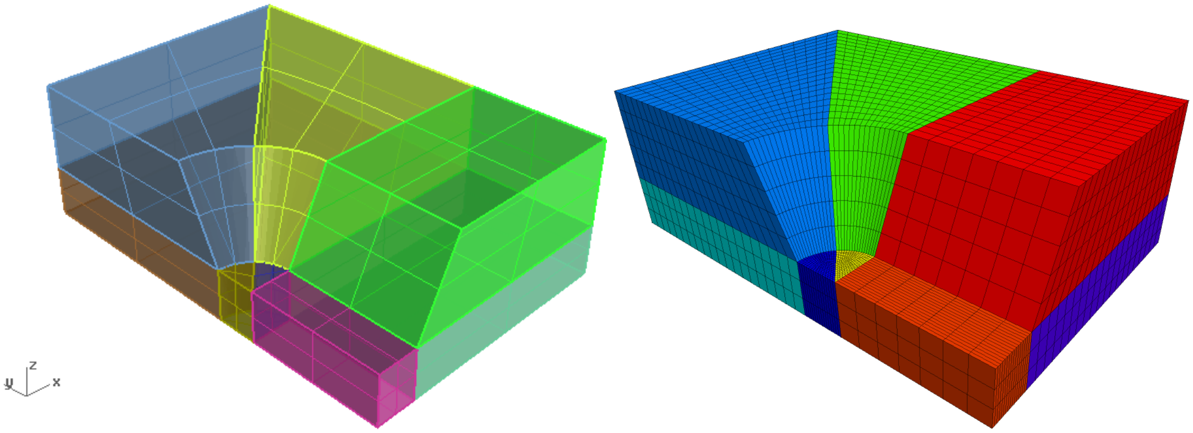
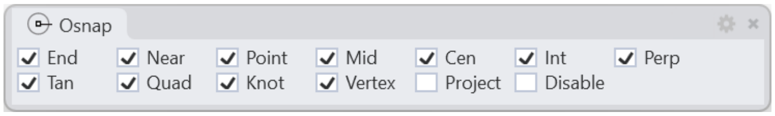


Figure 34: Alternate solid layout resulting in nine mesh blocks.

Building solids from curves

7. Activate **Osnap** (Object Snap) near bottom-left side of the screen (Figure 35). Make sure that all checkboxes designating what to snap to are checked, as shown in Figure 35.



Osnap

☑ End ☑ Near ☑ Point ☑ Mid ☑ Cen ☑ Int ☑ Perp
☑ Tan ☑ Quad ☑ Knot ☑ Vertex □ Project □ Disable

Figure 35: *Rhino* status bar.

8. Starting with the reference model presented in Figure 30, create new construction lines as shown in Figure 36. Use the **_Line** command and start by connecting nodes from the middle of the model (where the curved segments are) to the sides of the model. Pay attention to when the mouse tooltip shows “Perp” or “Perp, Int” (or other types of intersections), which means that the line under construction is perpendicular to (and intersects) another line.

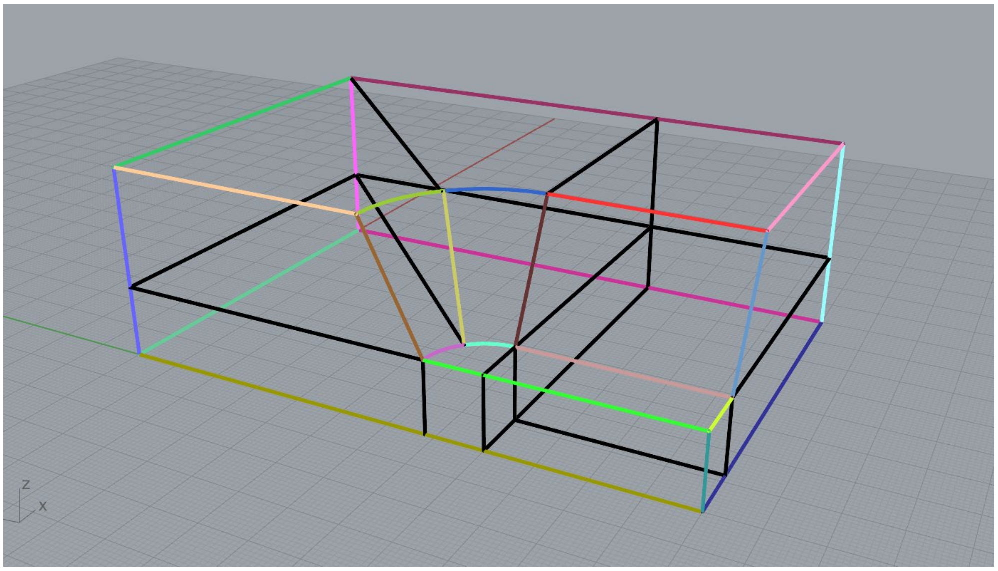


Figure 36: Creating new construction lines (in black).

The curves created in the previous step (Figure 30) are polylines, and some of them are composed of multiple linear segments that cannot be used directly to build surfaces. They must be retraced with arcs (**Curve** → **Arc**) or higher order curves (**Curve** → **Free-Form** → **Interpolate Points**) to create simpler curves with nodes at their ends only.

9. Zoom in around the curved segments in the model and create a set of new arcs. *Rhino* provides many different options to select and run different commands, and these can be explored to give the user their preferred workflow. Follow one or more of the options below to create an arc.

    - Type **_Arc** into the command prompt and click on *StartPoint*.
    - In the top ribbon menu, navigate to **Curve** → **Arc** → **Start, End, Point**.
    - Navigate to the **Curve Tools** toolbar and select on **Arc: start, point, end on arc** (⌒).

    Create separate arcs at each curved segment of the model as shown in Figure 37 and Figure 38. In total, four arcs should be created. Pay attention to when the mouse tooltip shows “End, Int, Knot” when connecting to the point in the middle of the curved segments. Such a tooltip means the mouse is at the end of a line, hit an intersection with another line, and is at a knot.

10. After the arcs are created, the initial polylines that have been retraced can be deleted.

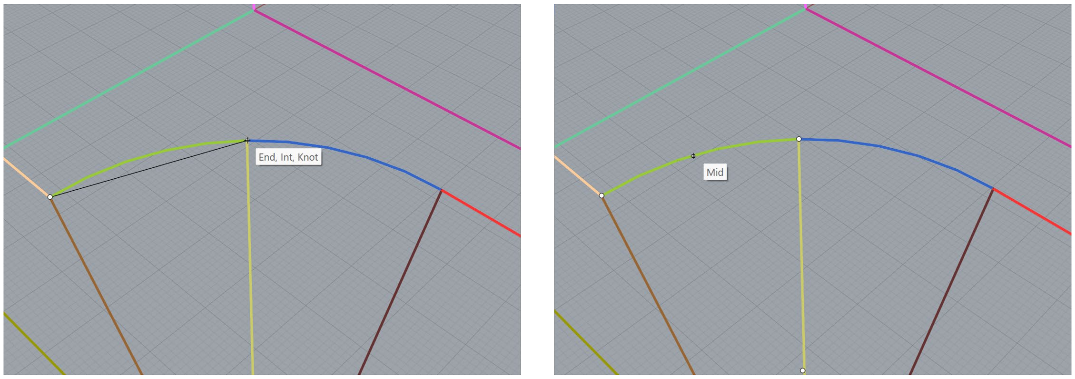


Figure 37: Building arcs.

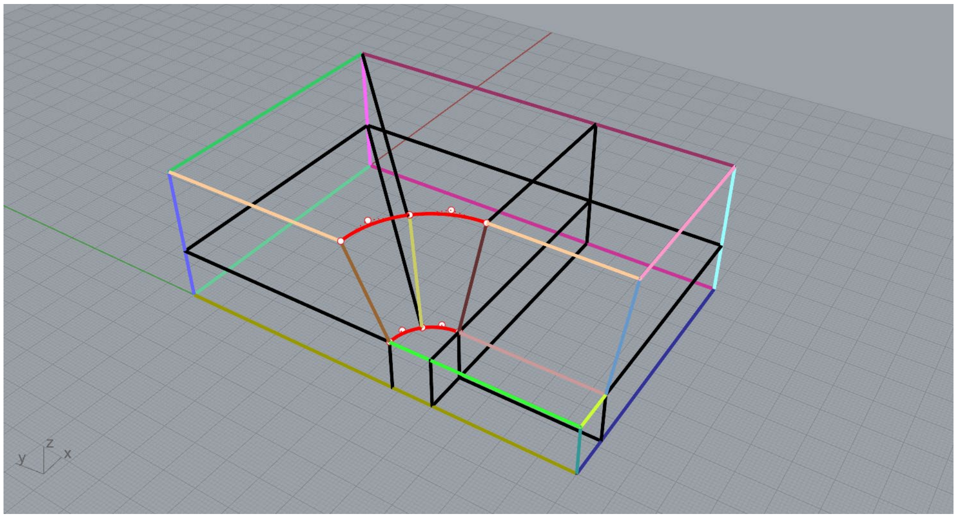


Figure 38: Red arcs retrace the original polylines at the curved section of the model.

11. While holding down the shift key, click the bottom two arcs to select them both. Run the **_Copy** command with *Vertical=Yes* and select the left corner for *Point to copy from* as shown in Figure 39. Drag the corner down until it hits the bottom line and the mouse tooltip is as in Figure 39. Click Enter again when done.

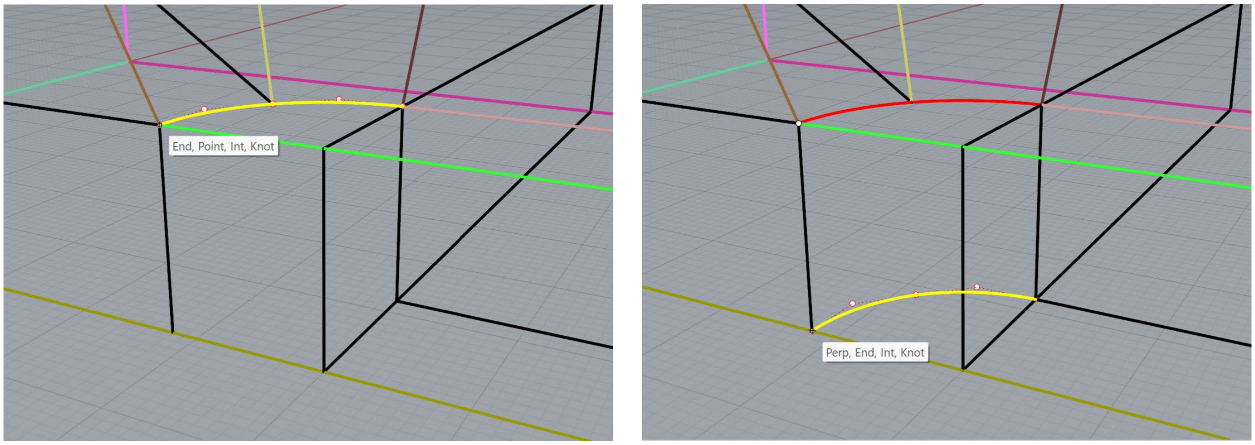


Figure 39: Copying the arcs.

12. Connect the middle nodes as shown in Figure 40 with the dark blue line. Connect the bottom end of the dark blue line to the opposite corner as shown in Figure 40 with the light blue line.

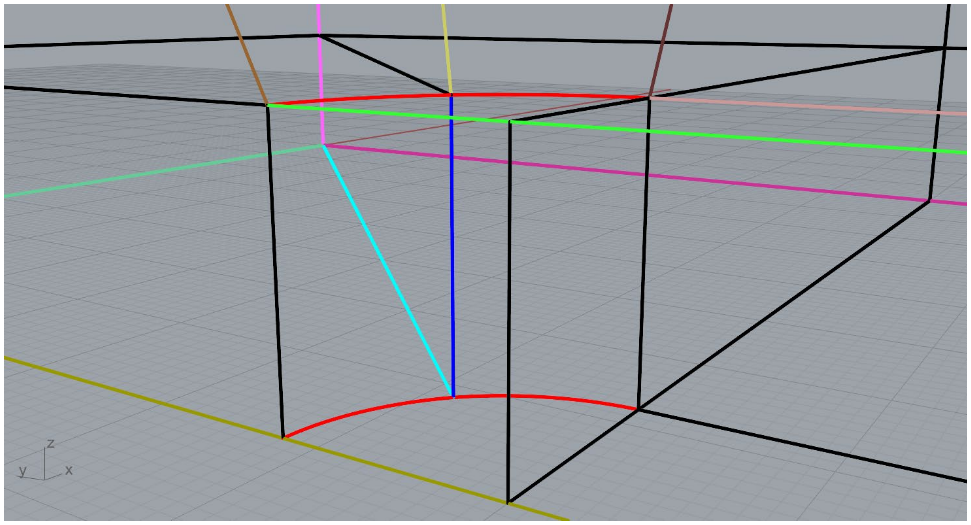


Figure 40: Connecting the middle ends of the arcs (blue lines).

13. Split all intersecting lines (mostly those from the reference model) by using the **_Split** command. For this, first select the line to split and then select a line that intersects or connects to it. For example, the tan line at the bottom of Figure 40 can be split into three pieces by the connecting black lines. The resulting wireframe should contain 56 curves (**Ctrl+A** and check the number of selected objects) and is the basis for creating solid blocks (Figure 41). Select all lines and run **_ColorizeObjects** (or press in the *Griddle* toolbar) to give each line a unique color.

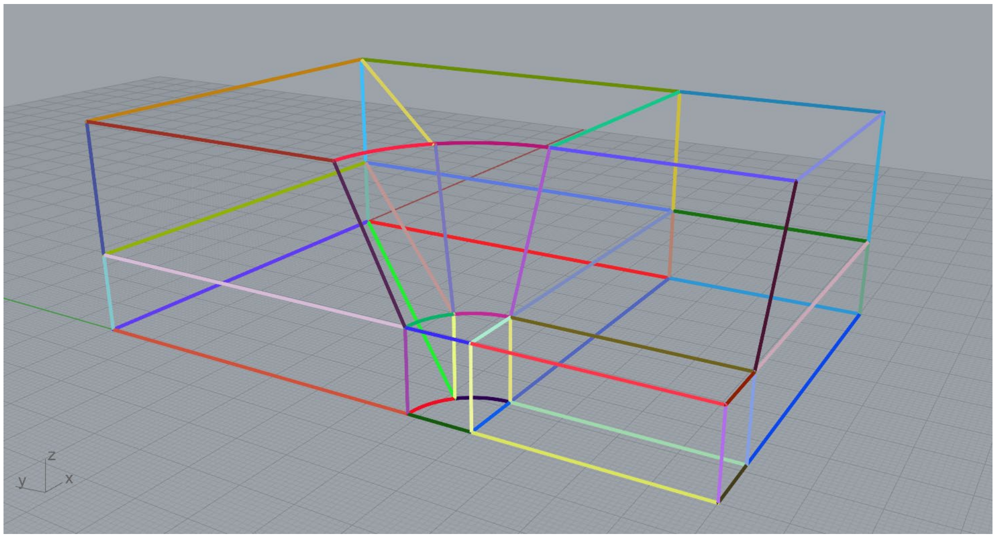
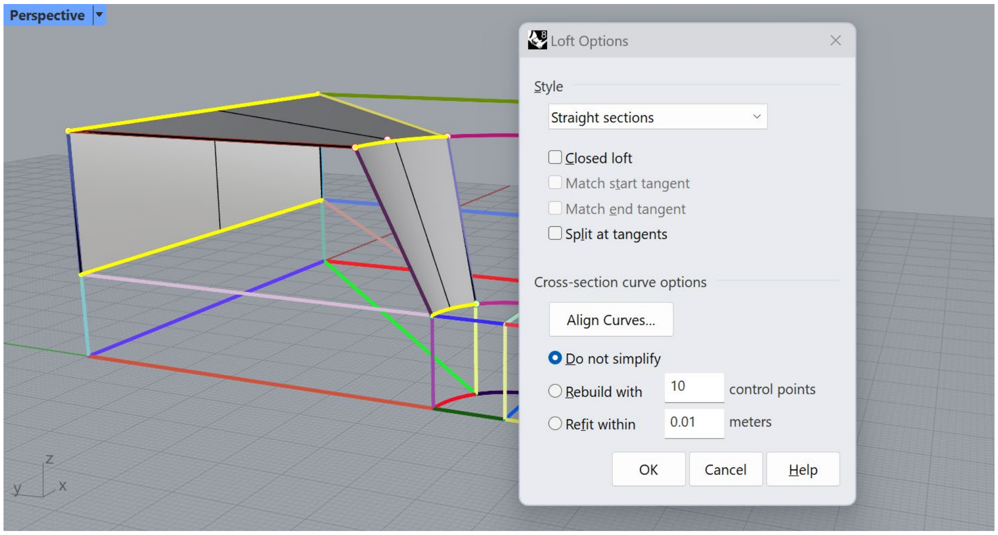


Figure 41: Model wireframe ready to be used for block creation.

14. Start creating a block at the top-left side of the model. Select the two arcs and two lines as shown in Figure 42 (selected lines are in yellow), type the **_Loft** command and use the settings shown in the right side of Figure 42. Click **OK** to build a lofted polysurface between the several curves. Alternatively, a single surface can be built between each pair of curves/lines.



Loft Options

Style

Straight sections

□ Closed loft

□ Match start tangent

□ Match end tangent

□ Split at tangents

Cross-section curve options

Align Curves...

● Do not simplify

○ Rebuild with 10 control points

○ Refit within 0.01 meters

OK Cancel Help

Figure 42: Building the surfaces for the first solid. Note that the grid has been hidden on this image.

15. Now, select only pairs of lines/curves composing the remaining surfaces of the first solid and **_Loft** them. In total, six sides must be created as in Figure 43.

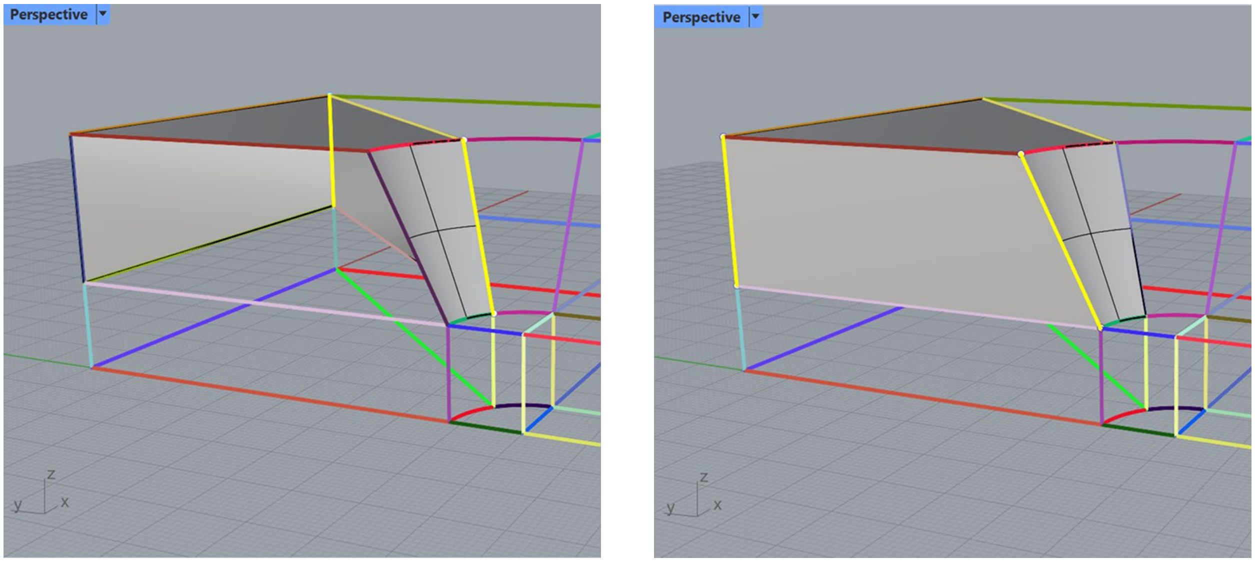


Figure 43: Building surfaces of the first solid.

16. Select all newly created surfaces and a polysurface and join them with the **_Join** command. This is the first solid block. Hide it with the **_Hide** command, as it will be easier to create other solids when nothing obstructs the view.
17. In the same manner, build all other solid blocks except for the small central block, as it requires an additional operation (described below). The stages are outlined in Figure 44.

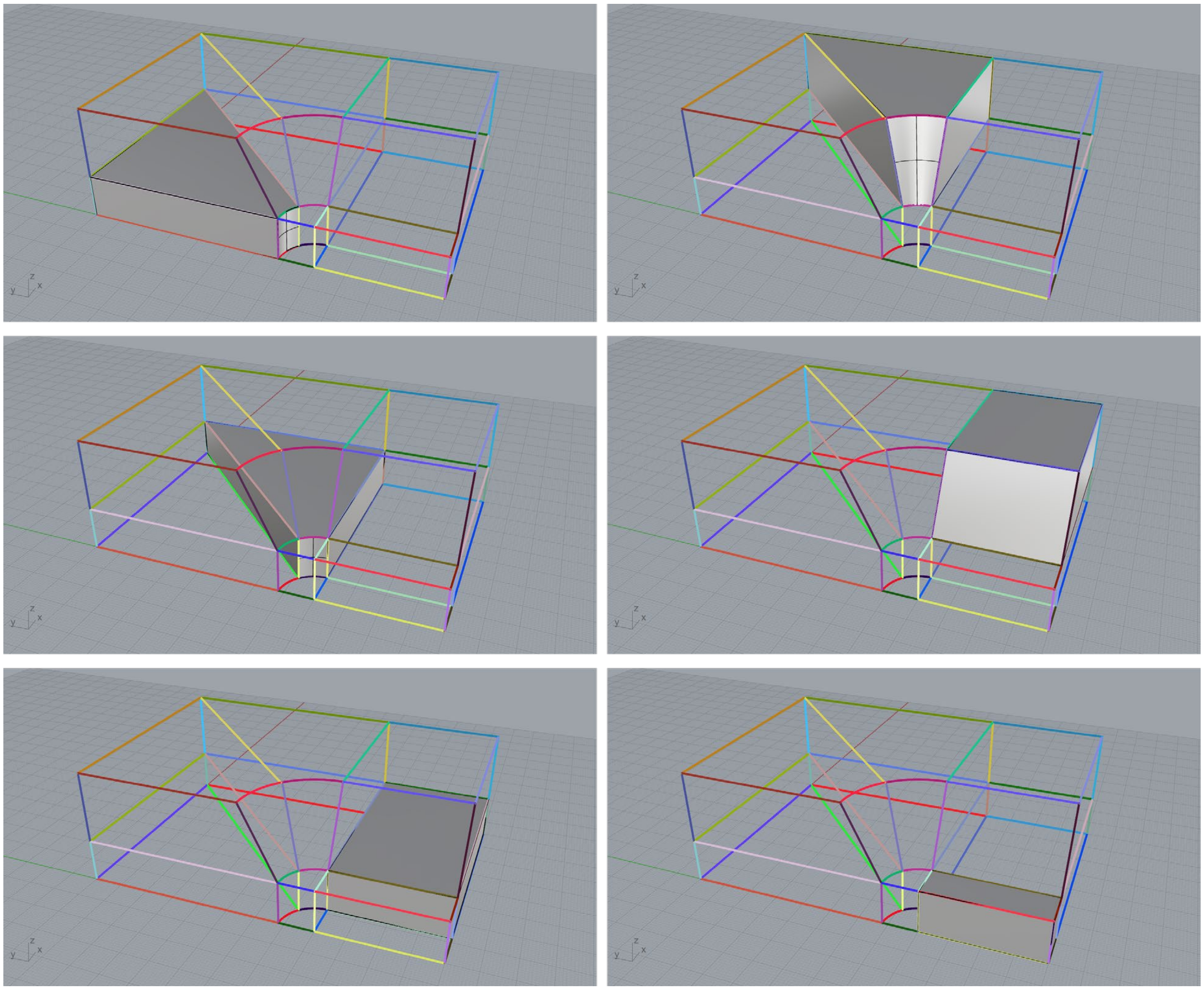


Figure 44: Building solid blocks for the model.

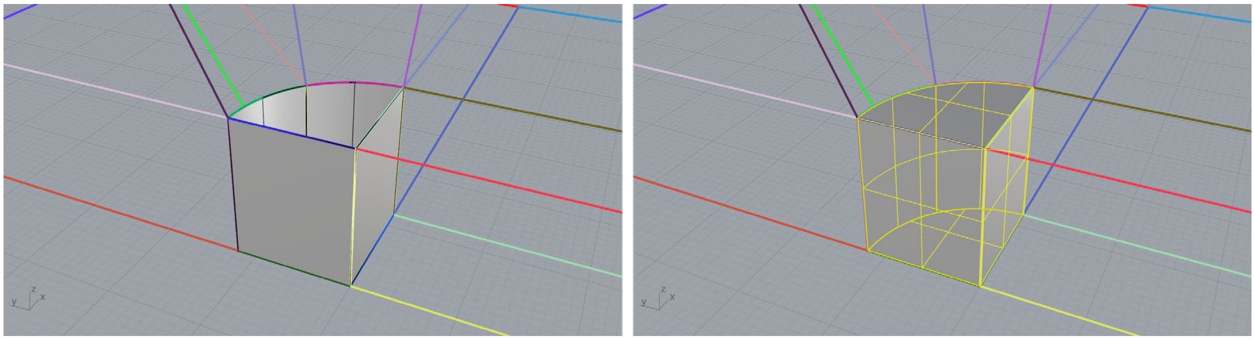


Figure 45: Building the last solid block in the model.

18. For the remaining central block, build only vertical surfaces using the **_Loft** command and join all four of them with the **_Join** command (Figure 45, left).
19. Select a single polysurface as in Figure 45 (left) and cap the holes with the **_Cap** command (Figure 45, right). The resulting solid is the last block needed before meshing.
20. Show all blocks that were previously hidden with the **_Show** command. There should be eight solid blocks as in Figure 46.

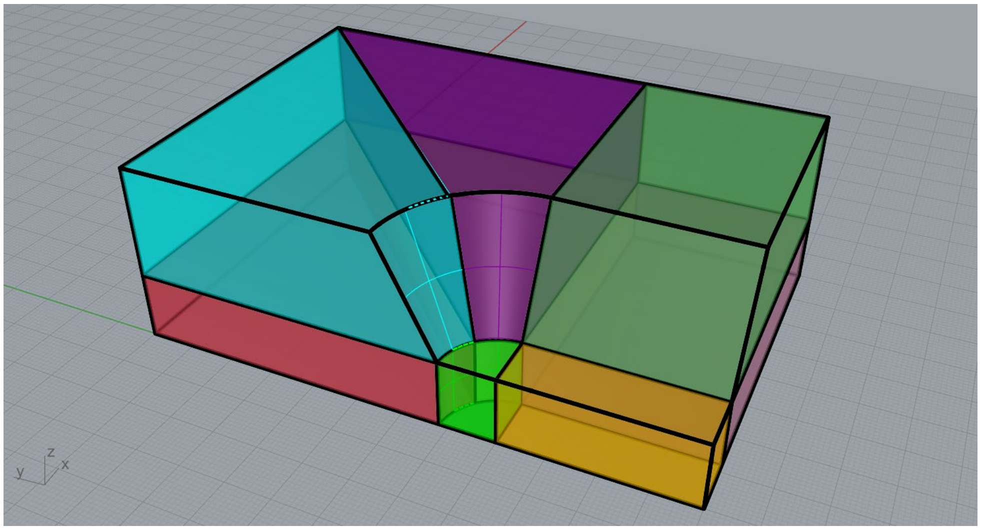
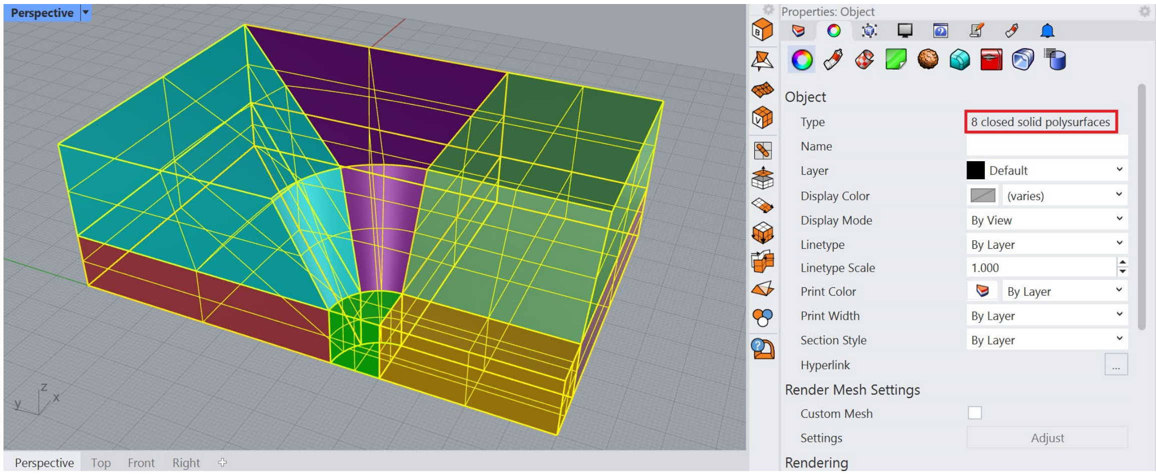


Figure 46: Solid blocks of the model (Ghosted view).

21. Select all curves by typing the **_SelCrv** command and assign layer "Reference" to them (navigate to *Properties* panel or press **F3**). Hide this layer in the *Layers* panel (press 💡 to turn it off).
22. Select all remaining objects with **Ctrl**+**A** and verify that *Rhino* reports only eight polysurfaces have been selected. If any surfaces are reported, this is an indication that they were not joined to a solid block. Find and join them so that only eight closed polysurfaces (solids) are present. While the solids are selected, navigate to the *Properties* panel (or press **F3**) and ensure that object *Type* indicates that they are closed polysurfaces (Figure 47). If the *Type* field says "varies", most likely it means that some solids are not closed (probably incorrect surfaces were created).
23. Create new layers "Upper solids" and "Lower solids" in the *Layers* panel. Assign distinct colors to these layers. Select only upper solids and assign them to the layer "Upper solids". Complete the same for the lower solids (Figure 48).

Now all solids are ready for meshing.



Properties: Object

Object
Type: 8 closed solid polysurfaces
Name:
Layer: Default
Display Color: (varies)
Display Mode: By View
Linetype: By Layer
Linetype Scale: 1.000
Print Color: By Layer
Print Width: By Layer
Section Style: By Layer
Hyperlink: ...
Render Mesh Settings
Custom Mesh: □
Settings: Adjust
Rendering

Figure 47: Solid blocks of the model (Ghosted view).

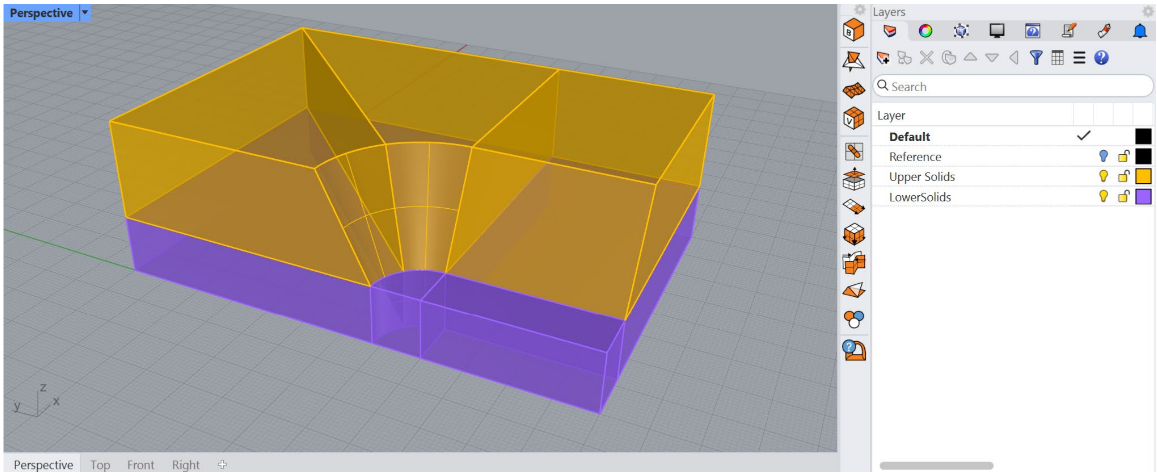


Figure 48: The model after the “Upper solids” and “Lower solids” layers have been specified (Ghosted view).

Meshing with BlockRanger

24. Select all solids and type the **_BR** command or click on the icon in the *Griddle* toolbar. Use the default **BlockRanger** parameters and save the mesh in *FLAC3D* binary format.

In the *Rhino* command area, **BlockRanger** should report: “8 solids processed, 8 solids meshed, 0 errors.” If any of the solids do not satisfy the **BlockRanger** requirements outlined previously, the corresponding solids will remain selected in *Rhino* and **BlockRanger** will report that it processed less than 8 solids.

Figure 49 shows a structured *FLAC3D* grid created by **BlockRanger**, as well as Zone groups from slot “Griddle_Layers”. Zone groups corresponding to slot ”Griddle” contain the original solid blocks and are shown in Figure 50.

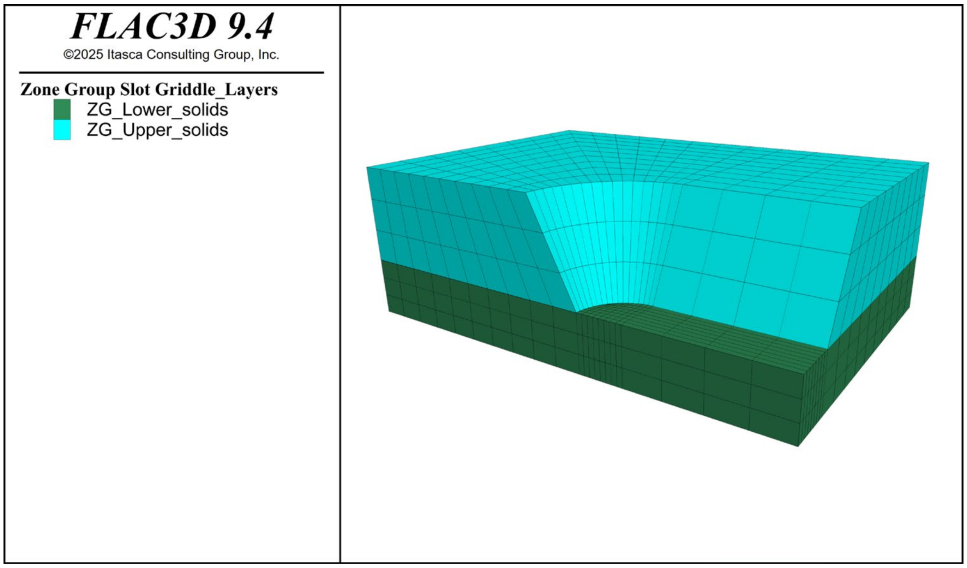

FLAC3D 9.4
©2025 Itasca Consulting Group, Inc.
Zone Group Slot Griddle_Layers
ZG_Lower_solids
ZG_Upper_solids

Figure 49: In *Rhino*, solids organized in layers are processed into *FLAC3D* groups in slot “Griddle_Layers”.

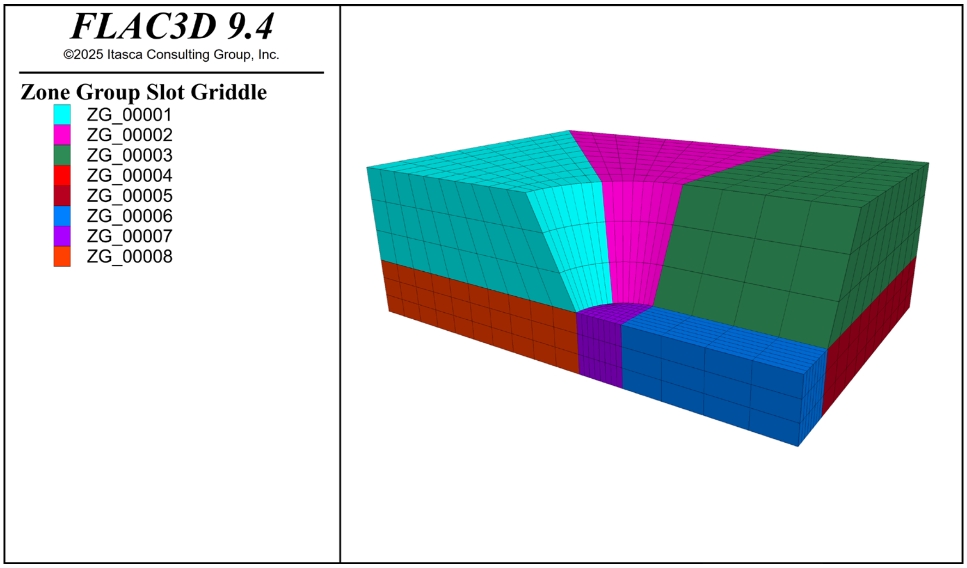


Figure 50: *FLAC3D* slot "Griddle" groups correspond to the individual *Rhino* solids.

As an additional exercise, change the **BlockRanger** meshing parameters and observe how the output mesh changes. For example, try setting *MaxEdgeLength* = 3.0 or *MaxEdgeLength* = 100.0 and *MinEdgeResolution* = 5. The simplest way to check the results is by generating bounding surface mesh with *GenerateSurfaceMesh* option. Use different choices in the option and observe how it changes the bounding surface mesh. The *ByModel* choice results in surface mesh on the boundary of the whole model only, selecting *ByLayer* adds faces on the boundaries between layers, and selecting *BySolid* adds faces on the boundaries between each solid.

Tutorial 5: Creation of a Hybrid Structured-Unstructured Mesh

Estimated work time: 1–3 hours

This tutorial describes a methodology that can be used to create hybrid volume meshes consisting of structured and unstructured meshes. This methodology ensures that both types of meshes are perfectly connected and fully conformal.

Structured meshes often provide high-quality elements and conform better to desired mesh parameters (such as element size, type). These meshes, however, can typically be created only for regular shape objects, for example, engineered or designed objects. On the other hand, unstructured meshes can fill objects of any shape and are well suited for meshing irregular geologic features. In this example, a tunnel excavation is filled with a structured mesh produced by **BlockRanger**, while the surrounding rock domain containing irregular topographic and fault surfaces is filled with an unstructured mesh.

Importing the geometry

1. Start *Rhino*, select **Large Objects, Meters** (similar method as step 1 in Tutorial 1) and save the *Rhino* project at a desired location.
2. Import the initial geometry from a DXF file by navigating to **File** → **Import** and select "T5_geometry.dxf" from folder "TutorialExamples\5_HybridMesh_Tunnel". In the **Import** dialog, select **Model** → **Meters** and **Layout Units** → **Millimeters**, and keep other parameters at the default values.
3. Maximize *Perspective* viewport, select Shaded view, and zoom out to the extent of the model by pressing **Ctrl**+**Alt**+**E**.
4. In the top menu, navigate to **View** → **Display Options…** This will open the **Rhino Options** dialog; select **View** → **Display Modes** →**Shaded** → **Objects** → **Curves** and set **Curve Width** to 4. This will display all curves thicker in the Shaded view. The initial settings can always be restored by clicking on **Restore Defaults**.

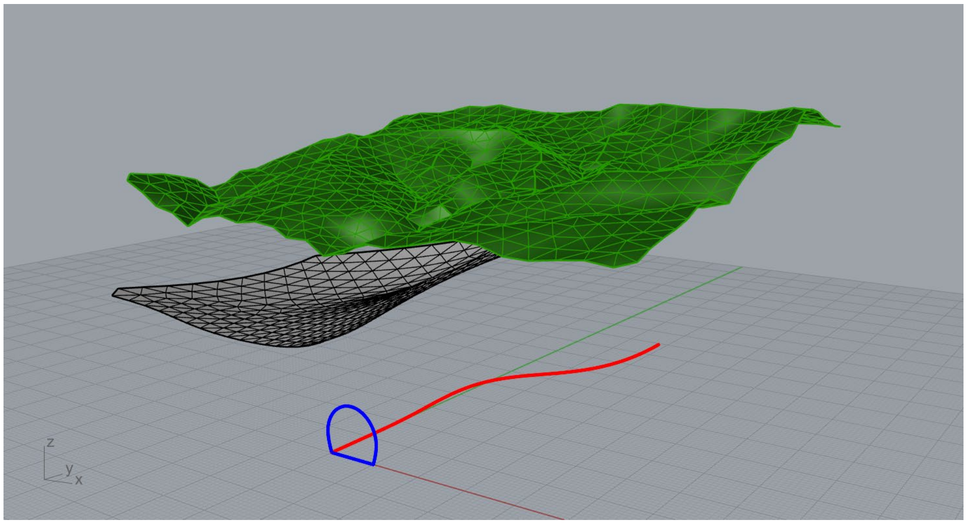


Figure 51: DXF geometry imported to *Rhino* corresponding to the tunnel profile and excavation direction.

5. Navigate to the *Layers* panel and delete all layers except for "Default", "Direction", "Topo", "Fault", and "TunnelProfile".

The imported geometry contains two meshes, one corresponding to the topographic surface and one corresponding to a large fault. Note that the fault almost reaches to the topo surface but does not intersect it. This will need to be fixed, as the fault is supposed to intersect the surface. The geometry also contains a horseshoe-shaped curve (in blue) corresponding to the tunnel profile and a red curve corresponding to the tunnel excavation direction. The tunnel must be constructed along the red curve in such a way that its cross-section is always perpendicular to the curve. Note that the front end of the red curve is already aligned with the bottom-left corner of the tunnel profile.

Construction and meshing of the tunnel with structured mesh

6. Turn off layers "Topo" and "Fault" to leave only the profile and direction curves visible.
7. Tunnel profile consists of several curves (in blue). Select all of them and type **_Join**.
8. Select the red curve (excavation direction), type the command **_Sweep1**, and select the single blue curve (tunnel profile) for the sweep shape. When offered to *Drag seam point to adjust*, drag it from the bottom-left corner of the tunnel profile to the topmost point of the profile and press **Enter** (maximize *Front* viewport for ease of dragging the seam point; this can be done while the command is active; Figure 52). It is important to move the *seam point* from the corner to create the proper geometry suitable for **BlockRanger**. For other options in the **_Sweep1** command, use those shown in Figure 53.

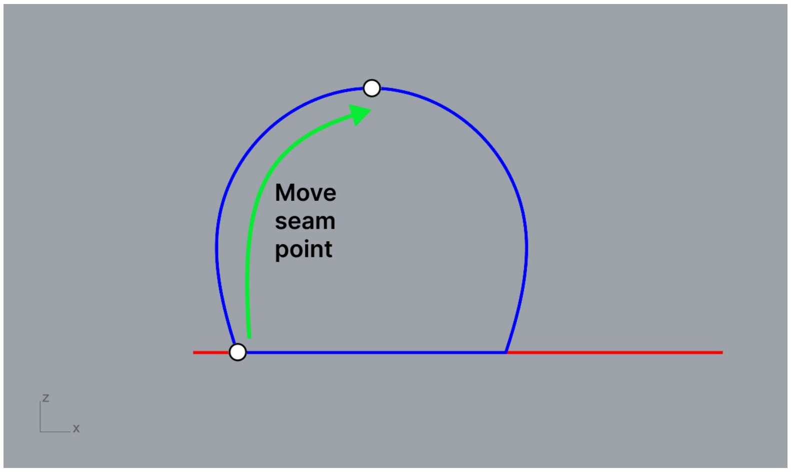
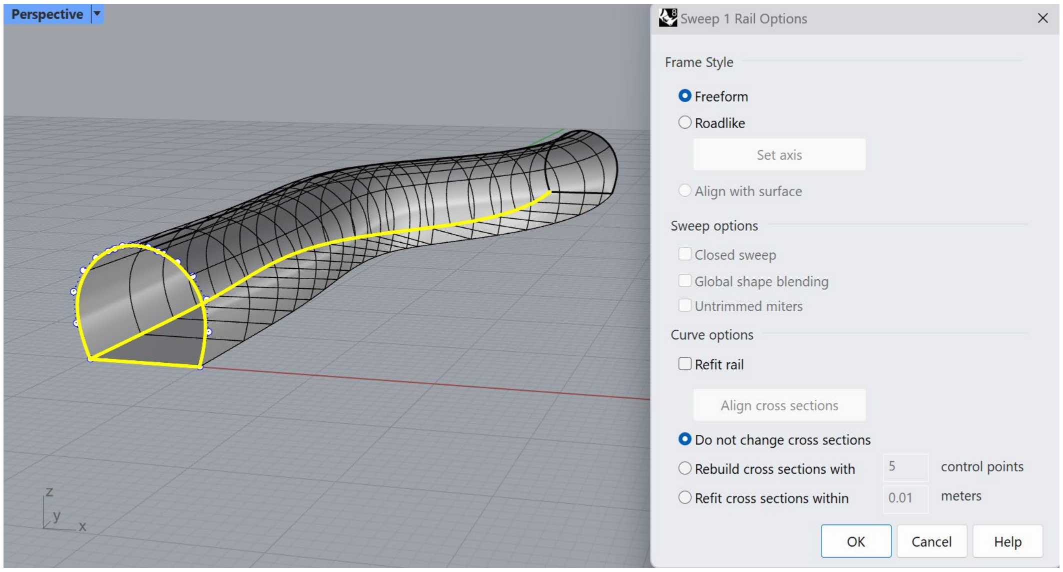


Figure 52: Drag seam point to the top of the tunnel profile.

9. Create new layer "Tunnel" and assign a newly created tunnel surface to it.
10. Select the tunnel profile curve (blue) and create the tunnel cross-section surface out of it by using the command **_PlanarSrf**.
11. Select the initial cross-section that was created in the previous step and use run the command **_ArrayCrv** to duplicate it along the direction path (red curve) with a spacing of roughly 5 m. Set *Distance between items* = 5 and *Orientation* = *Freeform*. This will create excavation stages that can be gradually removed during numerical simulation (Figure 54).

12. Turn off layers "TunnelProfile" and "Direction" as they will no longer be needed.



Sweep 1 Rail Options

Frame Style

● Freeform

○ Roadlike

Set axis

○ Align with surface

Sweep options

□ Closed sweep

□ Global shape blending

□ Untrimmed miters

Curve options

□ Refit rail

Align cross sections

● Do not change cross sections

○ Rebuild cross sections with 5 control points

○ Refit cross sections within 0.01 meters

OK Cancel Help

Figure 53: Creation of tunnel surface by sweeping the profile curve along the direction path.

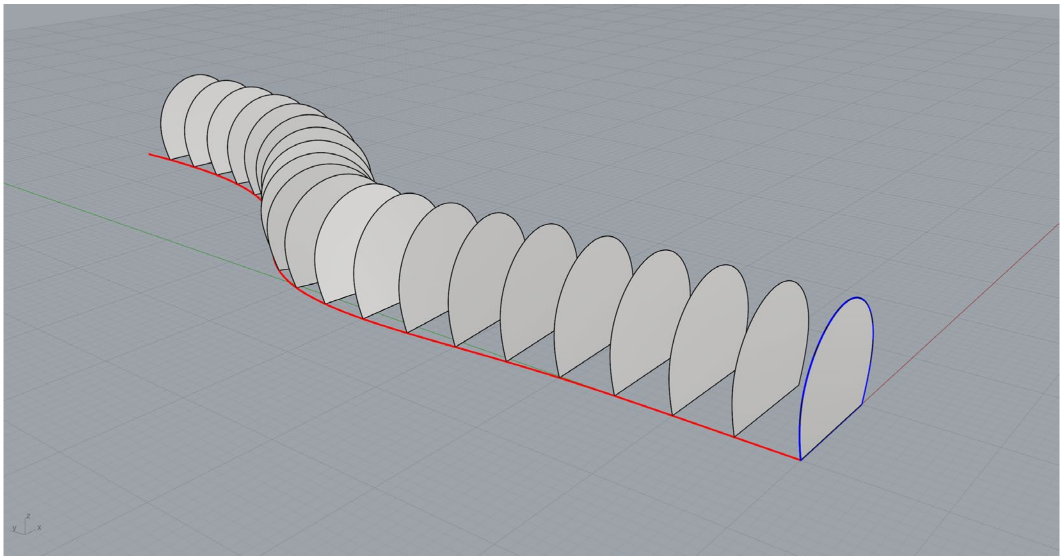


Figure 54: Creation of excavation stages by duplicating initial cross-section surface. Tunnel surface is not shown.

13. Split the tunnel surface with all the cross-section surfaces by selecting the tunnel surface only, then using the **_Split** command and selecting all other surfaces (**Ctrl**+**A**) as cutting objects. A message stating that "One polysurface split into 21 pieces." should appear in *Rhino*'s command area.
14. Turn off layer "Tunnel". Only the tunnel cross-sections should be visible. Delete all of them (21 objects).
15. Turn the "Tunnel" layer back on. Select all 21 polysurfaces and use the **_Cap** command to close the front and back of each excavation stage. A *Rhino* notification should appear that 42 caps have been created. Now each excavation stage is represented by a separate watertight polysurface (or by a solid), which can be easily meshed using **BlockRanger**.
16. Colorize all stages with *Griddle*'s **_ColorizeObjects** ( ) command (Figure 55).

Before proceeding with meshing the tunnel, some additional work is needed to trim parts of the tunnel that extend past the boundaries of the model.

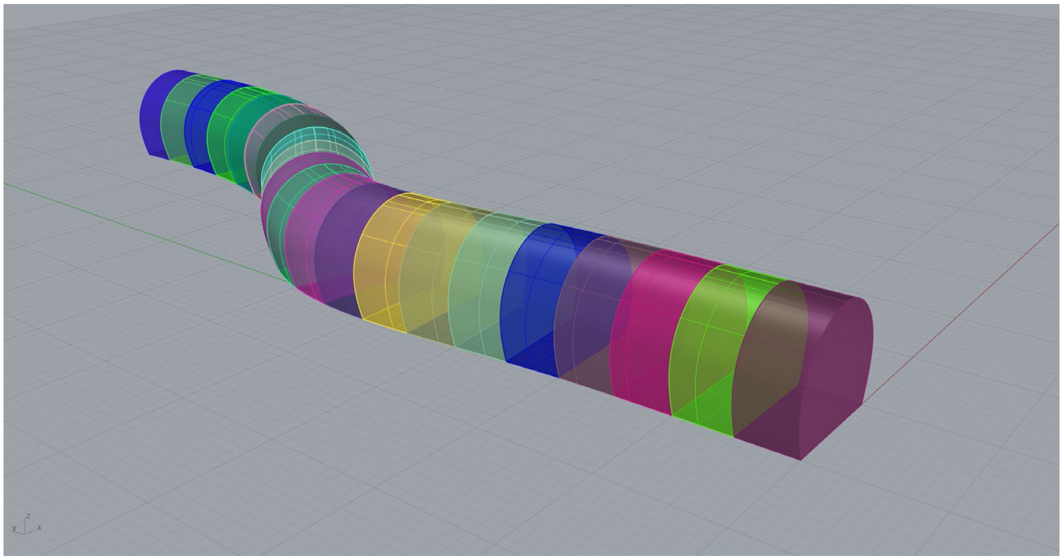


Figure 55: Watertight polysurfaces (solids) representing each excavation stage of the tunnel (Ghosted view).

17. Turn off the "Tunnel" layer and turn on the "Topo" layer to show the topographic surface mesh.
18. Create a planar surface underneath the topo mesh by using the **_Plane** command and selecting the *Center* option to specify the center of the plane and its size.
    - *Center of plane*: 0,50,-25
    - *Other corner or length*: 150
    - *Width*: press **Enter**
19. Select the topo mesh and duplicate its border using the **_DupBorder** command.
20. Create an extrusion from the border curve to the plane by running the **_ExtrudeCrv** command and selecting *ToBoundary*. This will create a polysurface that intersects the horizontal plane (Figure 56). Delete the duplicate border once finished.

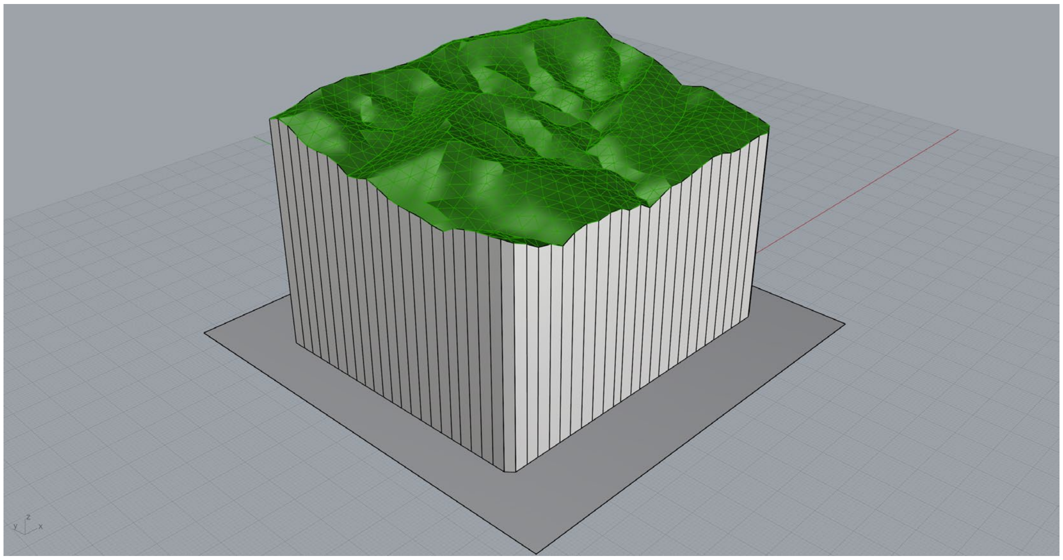


Figure 56: Extrusion of topographic surface border.

21. Select the planar surface and **_Split** it by the extruded polysurface. Delete the outer piece of the planar surface.
22. Join the bottom surface with the extruded polysurface (**_Join** command) to create a closed domain (together with the topo mesh), which will serve as the modeling domain after meshing.
23. Create new layer “Domain” and place the newly created polysurface in it (by changing the layer assignment in the polysurface properties).
24. Turn on the “Tunnel” layer and note that the front and back sections of the tunnel stick out from the domain (Figure 57). Split them by the domain and delete excessive parts.

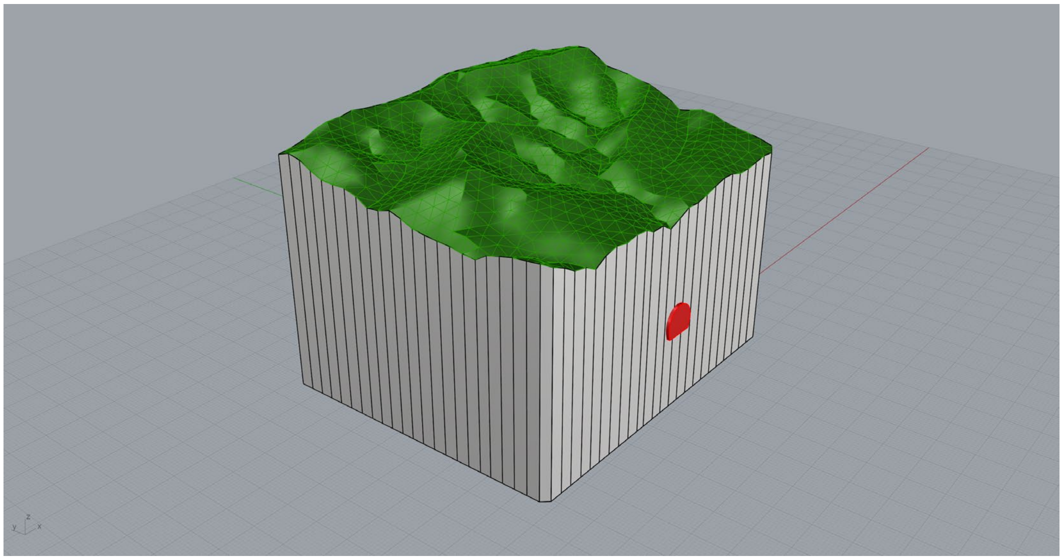


Figure 57: The front part of the tunnel extends through the modeling domain.

25. Hide layers “Topo” and “Domain” so only the tunnel stages are present.
26. Now the front and the back of the tunnel contain openings. Select the front and back tunnel stages and cap them using the **_Cap** command to create watertight solids. At this point, the tunnel is ready for meshing.
27. Select all 21 solids corresponding to the tunnel excavation stages and type the **_BR** command or click on the B icon in the *Griddle* toolbar. Use all default **BlockRanger** parameters and set *GenerateSurfaceMesh* = *ByModel*. **BlockRanger** will create a structured volume mesh for the tunnel and will output it at the user-specified location (keep *FLAC3D* binary format as output format). **BlockRanger** will also create surface mesh corresponding to the tunnel’s external surface and will output it into *Rhino* (Figure 58). This surface mesh will be used to create an unstructured volume mesh outside the tunnel.
28. Create new layer “TunnelMesh”. Change the layer of newly created surface mesh from “Default” to “TunnelMesh”. Now turn off layer “Tunnel”, leaving only one layer active as shown in Figure 58.

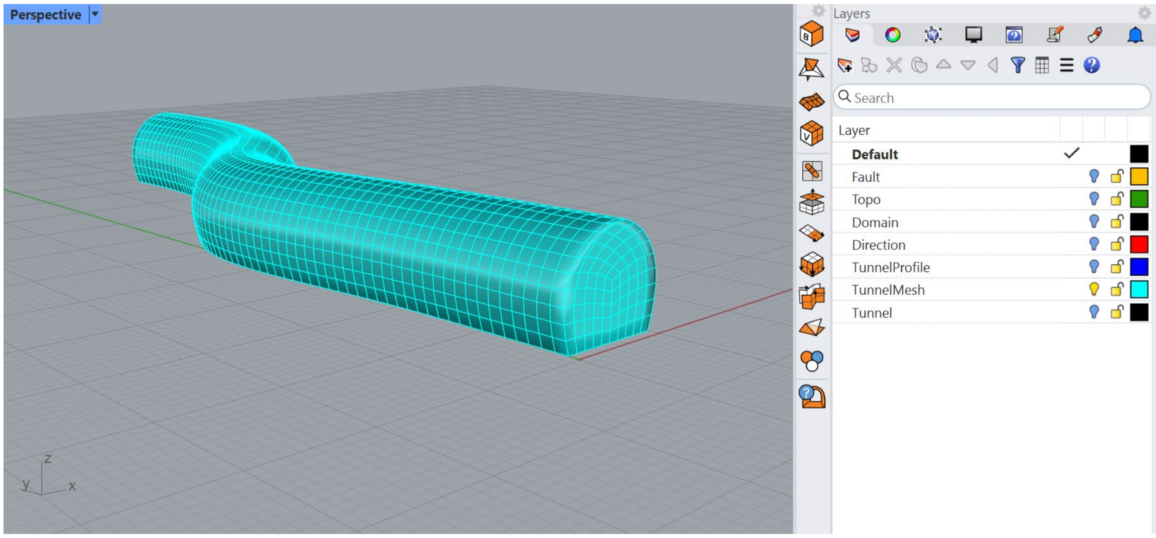


Figure 58: Surface mesh corresponding to the exterior of the tunnel's structured volume mesh.

Meshing the tunnel exterior with unstructured mesh

To create an unstructured domain mesh that is conformal with the tunnel structured mesh, bounding surface mesh presented in Figure 58 will be used.

29. First, the front and back caps of the bounding surface mesh should be removed to leave only the "shell" of the tunnel. Type the **_GExtract** command or click on the icon in the *Griddle* toolbar. Select option *Multiple* and use *MaxBreakAngle* = 85° to separate the caps. Select and delete them afterwards. The tunnel mesh should be hollow inside.
30. In the *Layers* panel, right-click on layer "TunnelMesh" and select *Duplicate Layer and Objects*. The next few steps will cause changes in the tunnel mesh, but volume meshing (the final step) requires the original mesh. Therefore, the duplicated tunnel mesh is used in the next steps. Turn off layer "TunnelMesh" and keep the duplicated layer on.
31. Select the tunnel mesh and use the command **_DupBorder** to duplicate its open boundary.
32. Turn on layer "Domain". Select the domain polysurface and split it with the duplicated tunnel boundaries by typing the **_Split** command and selecting the curves only.
33. Delete parts of the domain corresponding to the tunnel front and back to create openings for the tunnel (Figure 59). Do not delete the border curves, these will be used as a hard edge in following steps.
34. Select the domain and create the initial mesh for it using the **_Mesh** command. Use *Simple Controls* and select the middle position with the *NURBS meshing parameters* slider. While the polysurface is still selected, delete it. The initial mesh is not very good, but it will be remeshed in future steps.
35. Turn on the "Fault" layer. Note that the fault does not fully intersect the domain and the topo meshes (turn "Topo" layer on and off to see that). To extend the fault mesh to the domain boundaries, select it and type the **_GExtend** command or click on the icon in the *Griddle* toolbar. Select option *ExtendAllBoundaries*, specify *ExtendLength* = 10, and keep *MeshType* = *Merged*. The command will extend the mesh by adding new faces, and the extended mesh will fully intersect the domain and topo meshes. Turn on the "Topo" layer. The result should look similar to Figure 60.
36. Select the fault mesh and remesh it with **GSurf** () using *Mode* = *Tri, MinEdgeLength* = 3, *MaxEdgeLength* = 6, *RidgeAngle* = 30, and keep the rest of the parameters at defaults.

37. Select all objects (2 curves and 4 meshes) with **Ctrl+A** and use **GInt** to intersect all meshes. Use a tolerance of 0.001 and keep all other parameters at defaults.
38. Type **_MeshTrim** command, then select both the topo and the domain mesh as cutting object and then trim the external mesh part of the fault mesh by clicking on it and then pressing Enter to complete the command. Make sure that the main part of the fault remains inside the domain (not deleted).

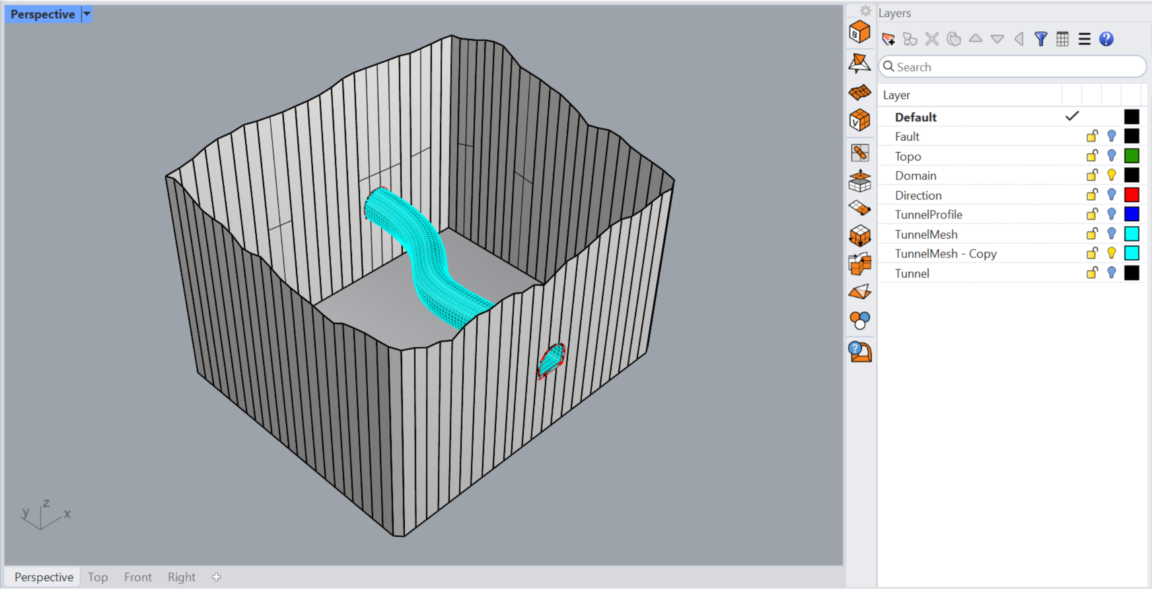


Figure 59: Removing parts of the domain at tunnel front and back.

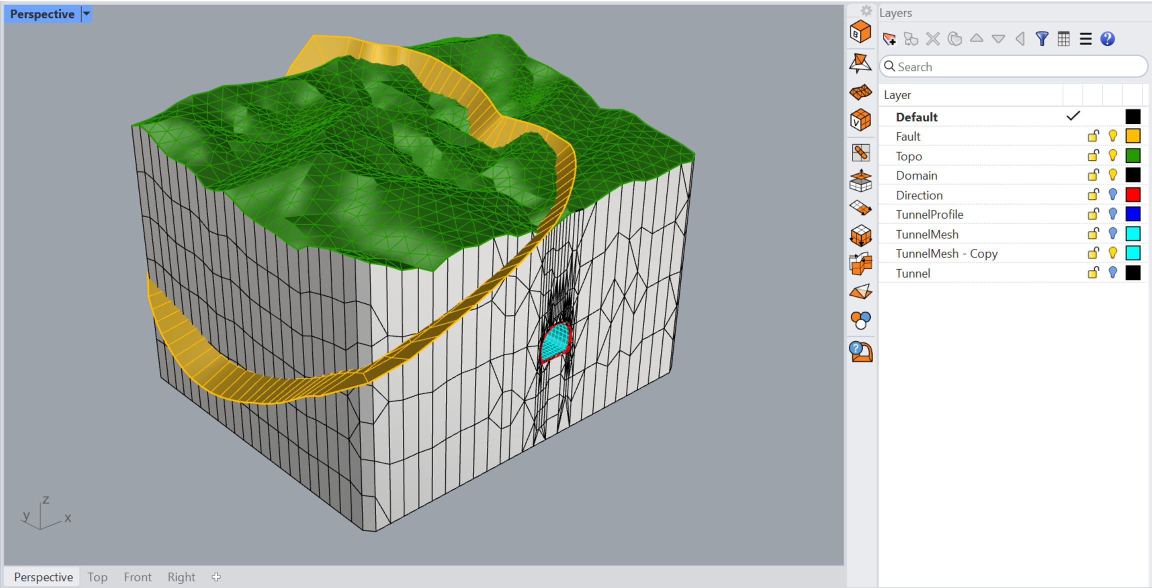
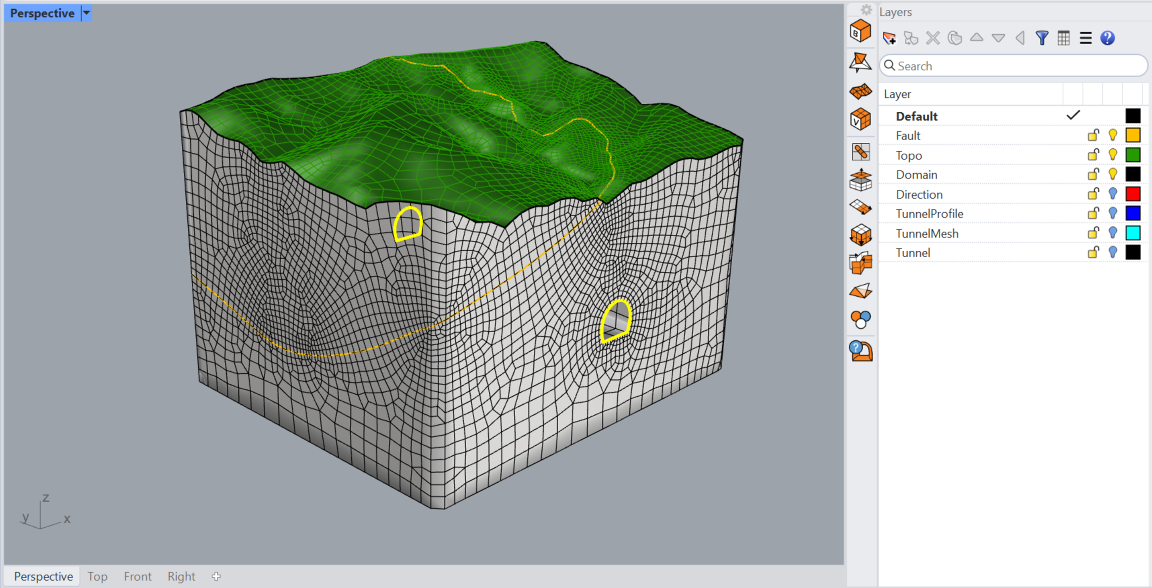


Figure 60: Extended fault mesh intersects domain and topo meshes. Note that the colors of topo and tunnel meshes and duplicated tunnel borders (curves) are changed from default.

39. Delete the copy of the tunnel mesh (blue mesh in Figure 60) or delete the whole layer containing it.
40. Select all objects again (2 curves and 3 meshes) with **Ctrl+A** and use **GSurf** () to remesh the meshes. Because the duplicated tunnel borders are selected as well (red curves in Figure 60), they will serve as hard edges to preserve conformity with the tunnel mesh. Set *Mode* = *QuadDom,*

MinEdgeLength = 1*, MaxEdgeLength* = 10*, RidgeAngle* = 40*, AdvancedParameters*: *MaxGradation* = 0.07, and keep all other parameters at default values.
After **GSurf** completes remeshing, the duplicated tunnel borders stay selected. Delete them, as they are no longer needed (Figure 61).

Turn on layer "TunnelMesh" and zoom in to the tunnel front or back (Figure 62). It can be easily seen that both tunnel and domain meshes are conformal (nodes and edges match along the border). This is due to using **GInt** ( ) to create proper initial intersections between the domain and tunnel meshes and due to using the borders of the original tunnel mesh as hard edges during remeshing.



Layer			
Default	✓		■
Fault			🟧
Topo			🟩
Domain			■
Direction			🟥
TunnelProfile			🟦
TunnelMesh			🟦
Tunnel			■

Figure 61: The result of remeshing of all meshes (except for the tunnel mesh) while preserving mesh edges along the tunnel borders (hard edges).

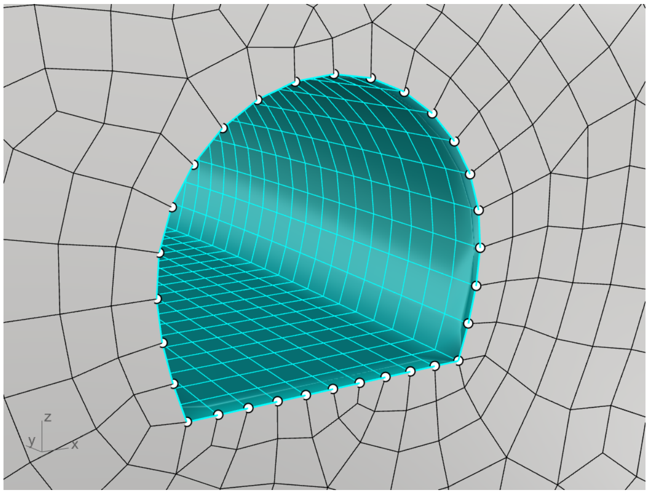


Figure 62: Close view of the tunnel front showing that tunnel and domain meshes are conformal.

41. Assign names to each of the meshes in the object *Properties* panel (click on a mesh and press **F3**), e.g., name fault "Fault", tunnel mesh - "Tunnel", etc. These names will be passed as surface names within the volume mesh.

Now the model is ready for volume meshing with **GVol** ( ).

42. Select all meshes and use **GVol** to create an unstructured volume mesh for the exterior of the tunnel. For parameters, use *HexDom* mesh and output in *FLAC3D* binary format; keep all other parameters at defaults. Save the output file at the same location as the tunnel structured mesh (do **NOT** overwrite it). Note that the volume mesh will not be created inside the tunnel this time, as the interior part of the tunnel is not a watertight domain; only the exterior of the tunnel constitutes a watertight domain.

Two meshes (grids) have been created in this example: a structured volume mesh with stages corresponding to the interior of the tunnel and an unstructured volume mesh corresponding to the exterior of the tunnel (domain). Both meshes can be loaded in *FLAC3D* (or any other code, if another format was used) and used for numerical modeling. Because the meshes are conformal and faces, edges, and nodes are duplicated along the boundary of the tunnel, *FLAC3D* can merge the meshes to avoid using the attachment logic. After importing both meshes, use the merge command:

```
flac3d>zone gp merge
--- 2610 gridpoints merged and 5160 surface faces removed.
```

The result is shown in Figure 63.

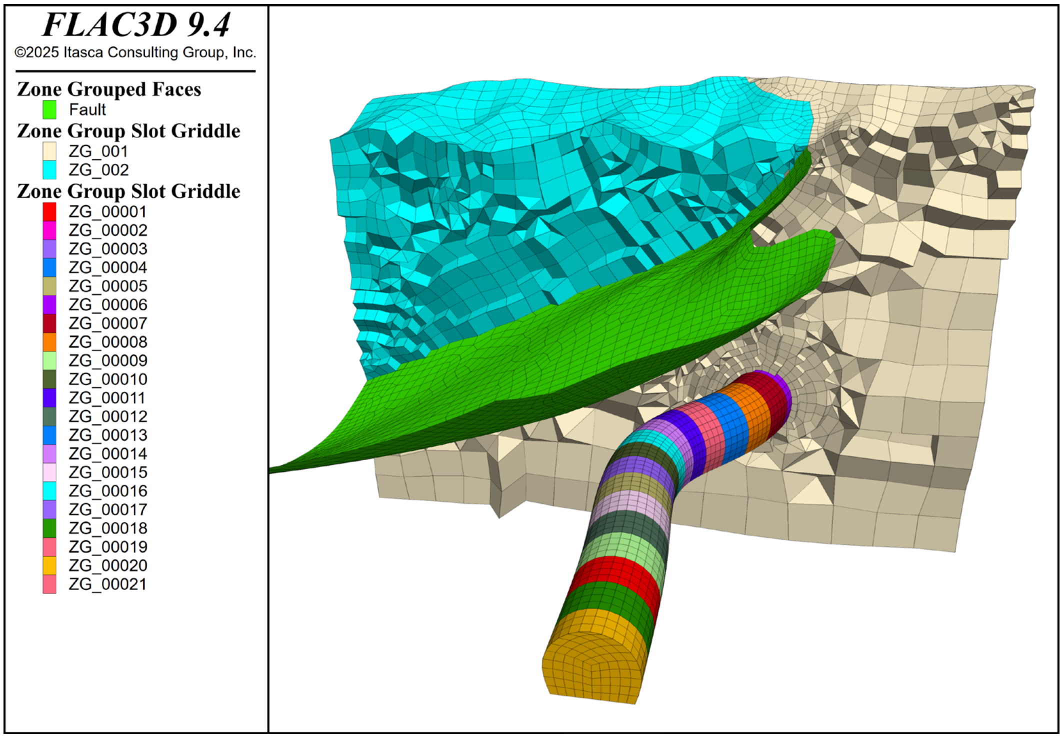


Figure 63: Structured and unstructured meshes loaded in *FLAC3D*.

Tutorial 6: Mesh Clean-up and Rebuilding

Estimated work time: 1 hour

This tutorial provides an example of a model that requires initial mesh cleaning and fixing before further model operations can be done. The model contains an existing triangular surface meshes defining a drift mine and a topographic surface (Figure 64). Such surfaces often come from minimally processed field measurements by various surveillance systems, and they often consist of triangular meshes, which generally are not clean (often non-conformal, contain overlapping and duplicate faces, missing faces, etc.) Thus, they typically need to be cleaned and remeshed to produce surface meshes suitable for volume meshing.

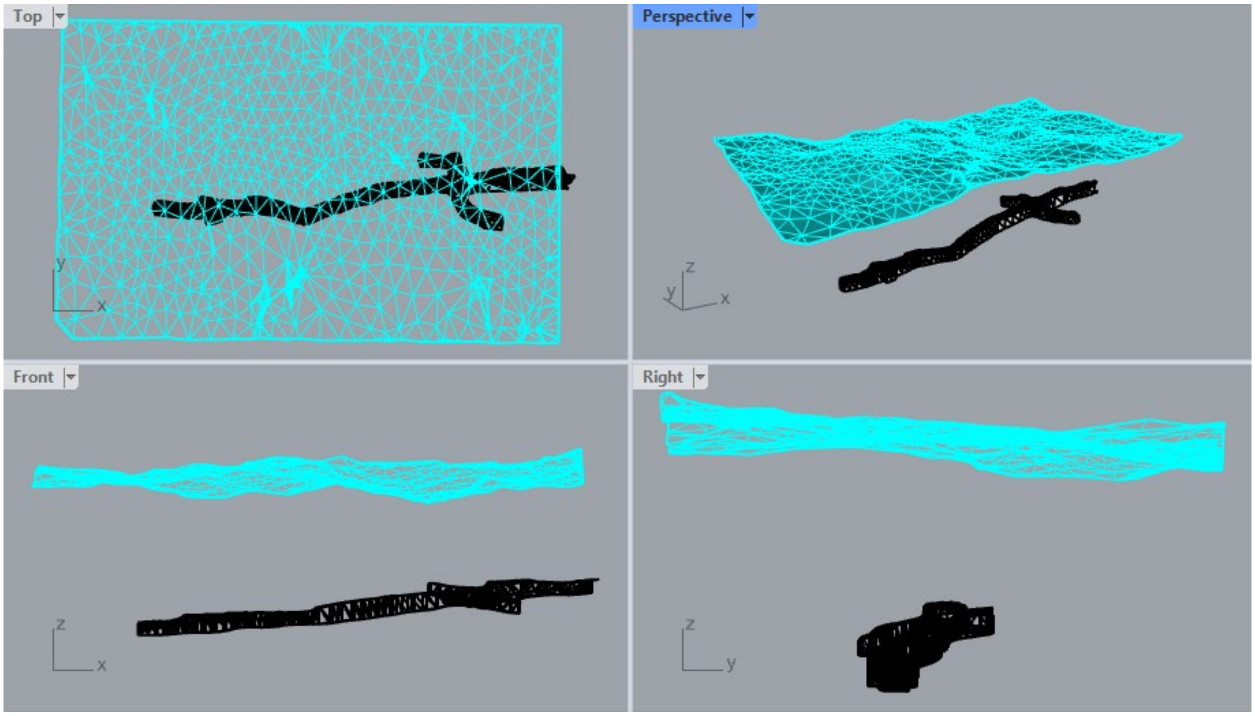


Figure 64: Drift geometry from a mine.

Project Set Up

1. Open project “T6_Drfts.3dm” located in folder “TutorialExamples\6_Mesh_Cleanup_Drifts”. Create a copy of this initial project with a different name at a desired location (use **File** → **Save As**).
2. If any of the viewport is maximized, double-click on the viewport icon to restore all four views as in Figure 64.
3. The initial model is located far from the origin and may not be visible in all viewports. Zoom out to the extent of the model by pressing **Ctrl**+**Alt**+**E**. Click on the topographic and drift excavation meshes to ensure that they are in separate layers.
4. Hover the mouse around the model and pay attention to the coordinates in the lower-left corner of the information panel: the displayed coordinates are far from the origin (zero coordinates). This may be problematic as it limits the numeric accuracy of geometric operations and the accuracy of a

subsequent numerical analysis. A good practice is to translate the model to locate it around the origin.

5. Select all objects (**Ctrl** + **A**) and type the **_Move** command. In the *Top* viewport select the top-left corner node as the point to move from and type 0 for the point to move to. Press **Ctrl**+**Alt**+**E** to zoom to the new model extents.
6. Maximize *Perspective* viewport and select Shaded view (Figure 65).
7. In the top menu, navigate to **View** → **Display Options…** This will open the **Rhino Options** dialog; select **View** → **Display Modes** →**Shaded** → **Objects** → **Curves** and set **Curve Width** to 4. This will display all curves thicker in the Shaded view. The initial settings can always be restored by clicking on **Restore Defaults**.

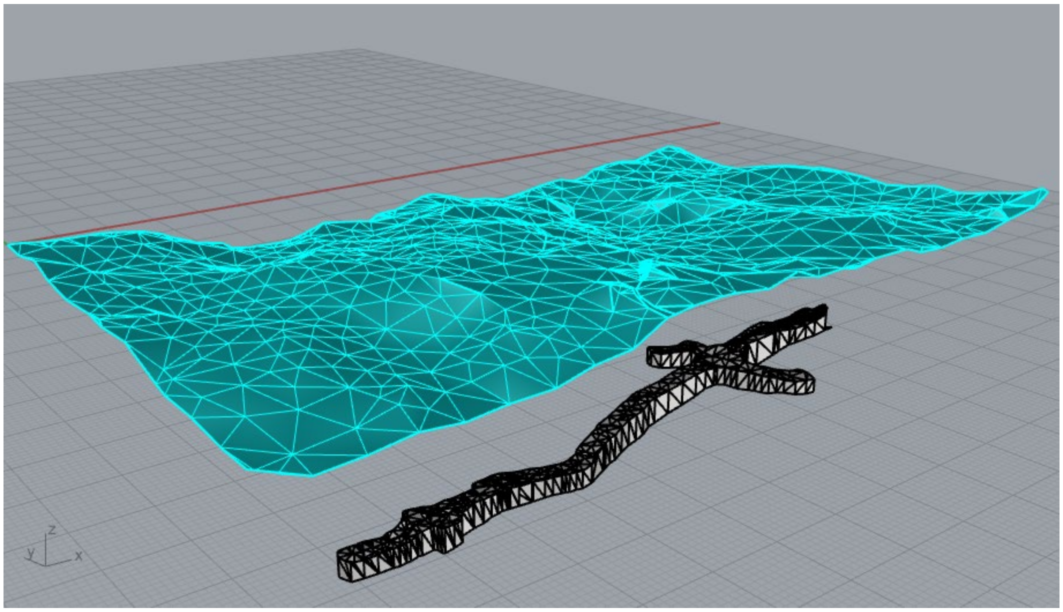


Figure 65: Drift model located near zero coordinates.

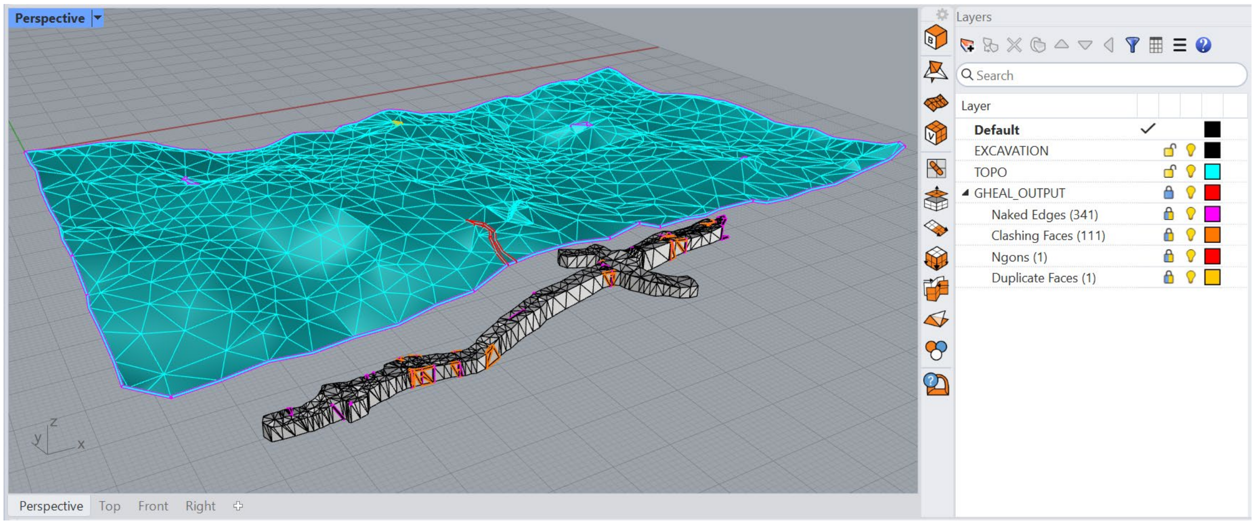
Mesh clean-up

The meshes provided in the model contain a number of issues, such as holes, nonconformal faces, duplicate faces, naked edges, etc. Such surface meshes are not adequate for creating a valid watertight mesh domain suitable for volume meshing. The meshes need to be cleaned up first.

Rhino has a few of built-in utilities which allow for checking and summarizing surface mesh issues. One such utility can be called by typing the **_Check** command (try this by selecting all meshes; a pop-up window will report a summary of various issues). Another useful command is **_MeshRepair**. However, using that utility for numerous issues may be tedious and cumbersome. *Griddle* offers the command **_GHeal**, which identifies and displays major surface mesh problems (that otherwise would prevent volume meshing) and attempts to fix numerous issues at once.

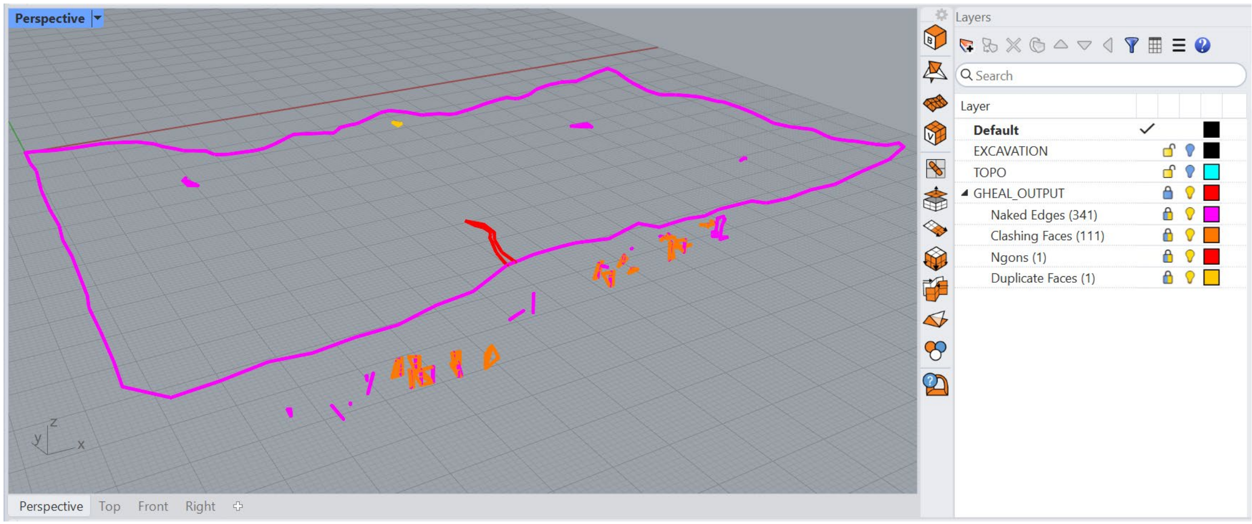
8. To check the meshes for various issues, type **_GHeal**, or click the icon in the *Griddle* toolbar, and select *ShowErrors* (Figure 66). The layer “GHEAL_OUTPUT” will appear, with sublayers containing curves outlining the corresponding issues. In this case, the layers contain many issues.

After mesh repair operations, the number of issues (and corresponding curves) will decrease, which can be checked by calling **GHeal** → *ShowErrors* again. To clearly visualize only problematic parts, layers “EXCAVATION” and “TOPO” can be turned off in the *Layers* panel (Figure 67).



Layer			
Default	✓		⬛
EXCAVATION	🔓	💡	⬛
TOPO	🔓	💡	🟦
GHEAL_OUTPUT	🔒	💡	🟥
Naked Edges (341)	🔒	💡	🟪
Clashing Faces (111)	🔒	💡	🟧
Ngons (1)	🔒	💡	🟥
Duplicate Faces (1)	🔒	💡	🟨

Figure 66: Displaying errors with GHeal.



Layer			
Default	✓		⬛
EXCAVATION	🔓	💡	⬛
TOPO	🔓	💡	🟦
GHEAL_OUTPUT	🔒	💡	🟥
Naked Edges (341)	🔒	💡	🟪
Clashing Faces (111)	🔒	💡	🟧
Ngons (1)	🔒	💡	🟥
Duplicate Faces (1)	🔒	💡	🟨

Figure 67: Displaying errors with meshes hidden.

These issues can be fixed using **GHeal** in automatic mode with the *AutomaticHeal* option (the manual mode option simply calls *Rhino*’s **_MeshRepair** command). *AutomaticHeal* allows for fixing numerous issues at once, but it is important to note that **GHeal** should be applied to problems discretely: only select to fix issues that are expected to be present. In certain cases, it is even better to fix issues one by one to get the best results. For example, first *FixClashingFaces* only, and then *FillHoles*.

In this model, there are holes and a Ngon in the topographic mesh, and there are holes and non-manifold edges in the drift excavation mesh. There are also misaligned normals in both meshes. Though for this model **GHeal** is able to fix all the issues in one pass (with all the options set to *Yes*), for better illustration of **GHeal** operations, the issues are fixed by parts in several steps.

9. Select all meshes and use **_GHeal** → *AutomaticHeal* → *IssuesToFix* and set only *TriangulateNgons*, *RemoveDuplicates*, and *FixClashingFaces* to *Yes*, and all others are set to *No.* Press **Enter** and keep all *AdvancedParameters* default. Press **Enter** again to run the command. The Ngon, duplicate face, and all clashing faces should now all be fixed, and the outlines of the remaining problems should now all be updated (including the layers).

   Now, there should be objects in the "Naked Edges" layer only. Naked edges can represent holes, "sliver" cracks/openings, non-manofold faces, and any other mesh edge that is not connected to another face (including the mesh boundary).

10. The topo mesh now only contains holes in the surface and can be resolved by running **GHeal** → *AutomaticHeal* → *IssuesToFix* and set *FillHoles* = *Yes*, and all other parameters set to *No*, and all *AdvancedParameters* kept at their default values. All problems in the surface mesh should now be resolved, and there should now only be 195 naked edges in the "Naked Edges" layer. This can also be verified by unlocking the "GHEAL_OUTPUT" layer and running **_SelCrv**; only 1 curve should be added to the selection (Figure 68).

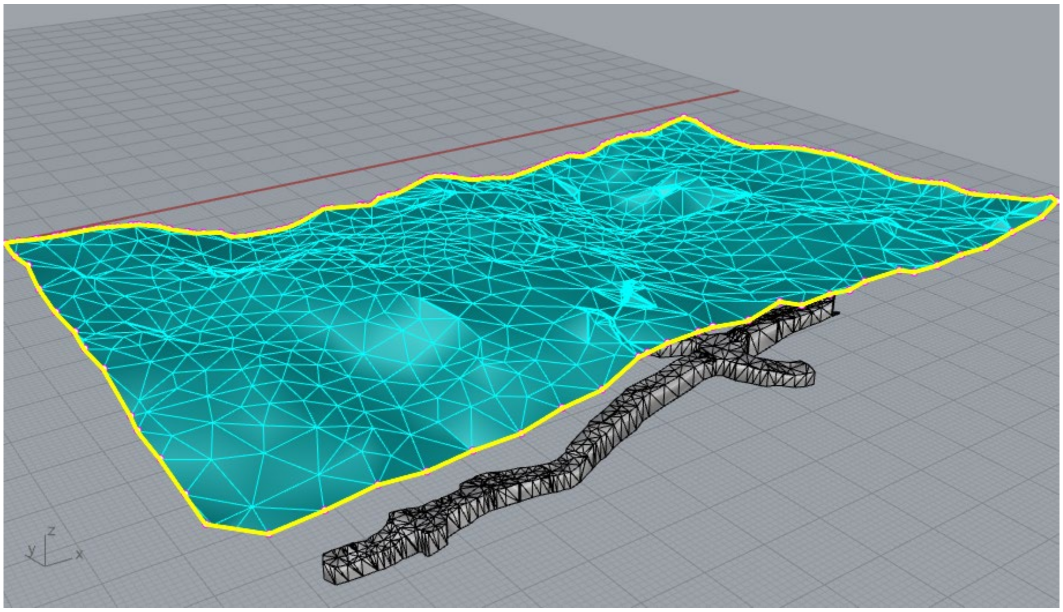


Figure 68: The fixed topo mesh with the remaining naked edges selected

11. Now select the drift mesh and run **_Check**. This window shows that there are extra naked edges and 5 non-manifold edges that remain. Resolve this issue by running **_GHeal** → *AutomaticHeal* → *IssuesToFix* with *RemoveNonmanifolds* and *FillHoles* set to *Yes*, and all other options set to *No*. Keep all *AdvancedParameters* at their default values. Output should display that 18 naked edges remain (one closed curve).
12. Select the topo and drift meshes only (<u>ensure no curves are selected</u>), and remesh them using **GSurf** ( ) with *Mode* = *Tri*, *MinEdgeLength* = 0.5, *MaxEdgeLength* = 5, while all other parameters remain at defaults. A rather fine mesh is used in this case to preserve the original mesh details.

13. Double check the meshes for issues by selecting both and running **GHeal** → *ShowErrors* one last time. The output should match that of Figure 69.

At this point, all outstanding problems are fixed and the meshes are ready to be used to construct the full model. Note that, in general, **GHeal**'s ability to fix various mesh problems significantly depends on the input mesh itself. Thus, results may vary depending on how the initial meshes are intersected and/or remeshed.

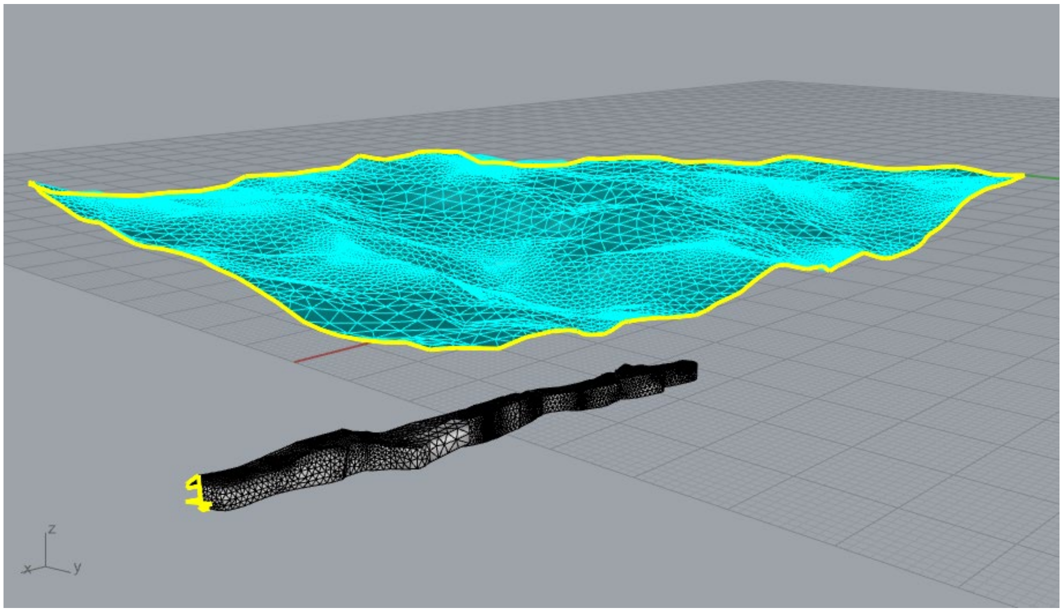


Figure 69: Remeshed and fixed meshes with remaining naked edges selected.

Building model domain and creating volume mesh

Before final remeshing and volume meshing, a watertight modeling domain must be created. This can be done using *Griddle*'s **GExtrude** tool.

14. Hide or delete "GHEAL_OUTPUT" layer so only the topo and excavation meshes are displayed.
15. Create a plane using the **_Plane** command and *Center* option. Then specify:
    - Center of plane (Deformable): 91,-55,-100
    - Other corner or length (3Point): 250
    - Width. Press Enter to use length: 200

    This will create a plane under the drift mesh as shown in Figure 71.

16. Select the topo mesh and the plane and type **_GExtrude** or click on the icon in the *Griddle* toolbar to extrude the mesh to the plane and create a watertight domain. Use the following parameters: *ExtrMeshType* = *Tri*, *Boundaries* = *External*, *MeshOutput* = *Merged*, *MinEdgeLength* = 1, and *MaxEdgeLength* = 10.
    The command will extrude the topographic mesh along its boundary until the intersection with the plane and will create side and bottom meshes. The result should look as shown in Figure 71.

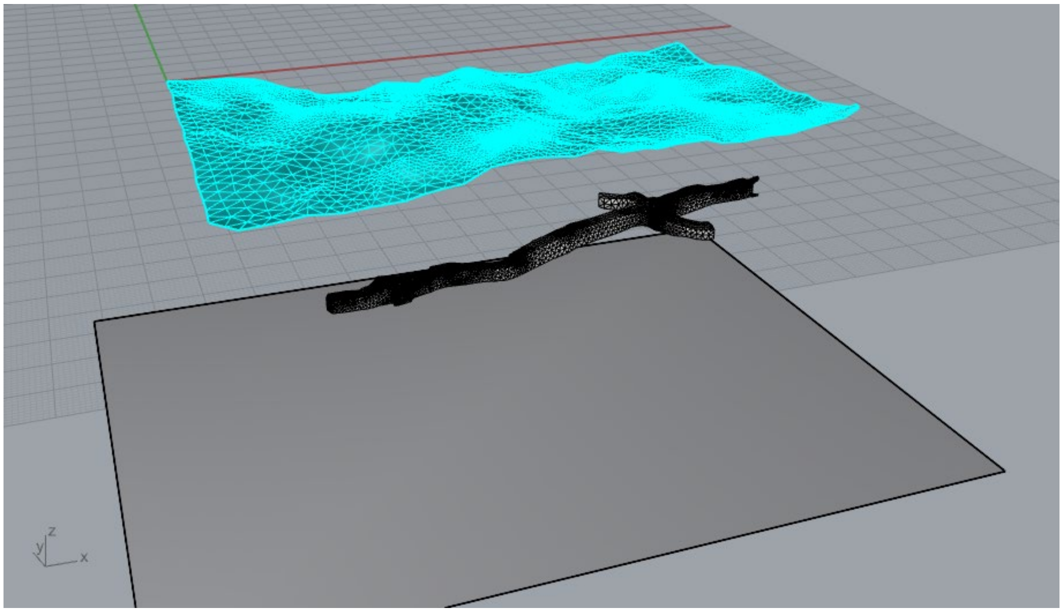


Figure 70: Creating a plane under the drift mesh.

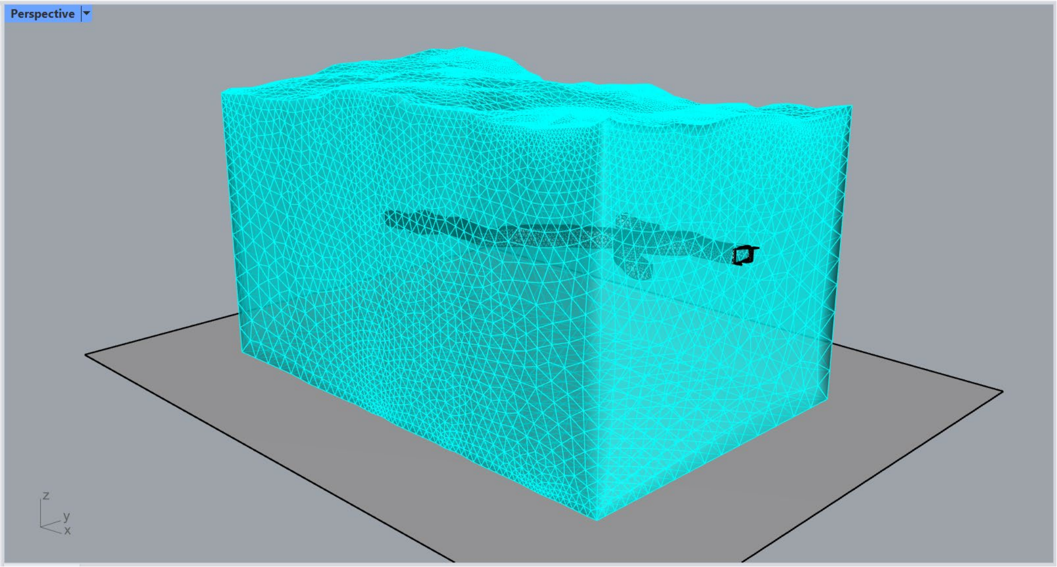


Figure 71: Creating watertight mesh domain with GExtrude command (Ghosted view).

Note that a piece of the drift mesh protrudes from the domain mesh. It can be easily separated and removed after the intersection of meshes.

17. Select and delete the plane.
18. Select all meshes (**Ctrl**+**A**) and intersect them using **GInt** with *Tolerance* = 0, and all *AdvancedParameters* as their default value.
19. Select the drift mesh and type **_MeshSplit** command, then select the domain mesh as a cutting object. After that, delete the small, separated piece of the drift mesh highlighted in Figure 72.

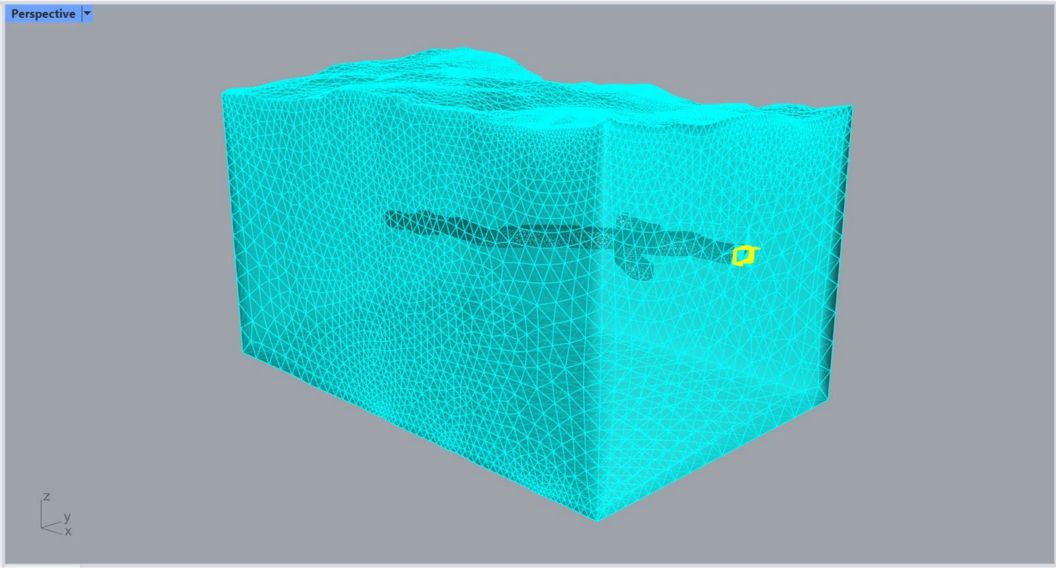


Figure 72: Meshes separated with the GExtract tool (Ghosted view).

20. Select the drift mesh and assign a custom element size in the Hyperlink field of the object properties panel (press **F3** if not visible): “elemsize:1.5”.
21. Select both meshes and remesh with **GSurf** using *Mode* = *QuadDom*, *MinEdgeLength* = 5, *MaxEdgeLength* = 10, *RidgeAngle* = 45, and *AdvancedParameters* → *MaxGradation* = 0.05; keep all other parameters at defaults. A large *RidgeAngle* is used to slightly smooth the meshes.
22. If desired, assign specific names to the drift excavation mesh and the domain mesh.
23. The surface meshes are ready for volume meshing. Create a *HexDominant* mesh using **GVol** with *MaxGradation* = 1, *TargetSize* = 7, and set the output format as desired (*FLAC3D* is used here). If a warning about the presence of naked edges appears, press **Yes** to continue with volume meshing (naked edges are present in the final model due to mesh extraction done earlier). Meshing may take some time due to the fine resolution of the topographic and drift meshes. The results of the volume mesh imported in *FLAC3D* are shown in Figure 73.

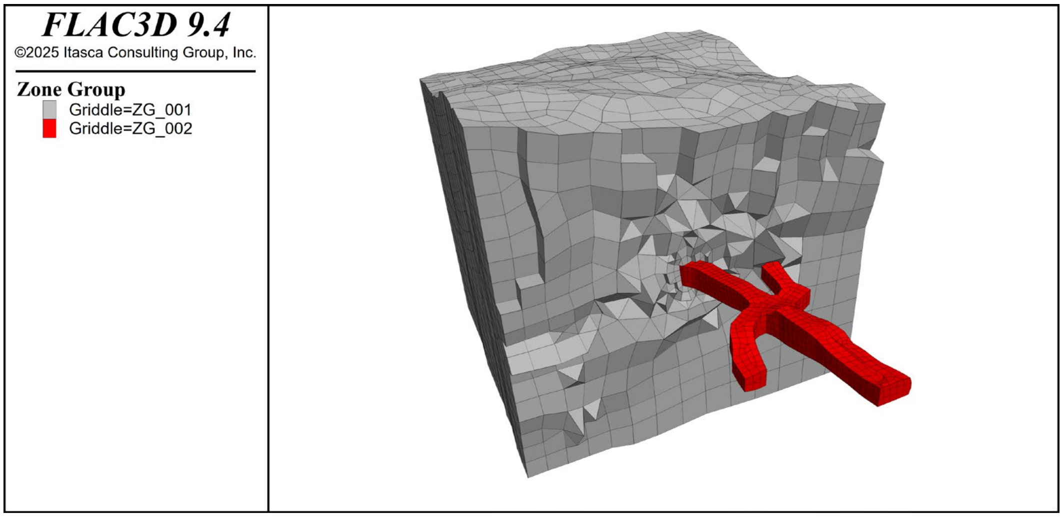


Figure 73: Final volume mesh for the model of the drift mine.

Tutorial 7: Large Open Pit Model with Multiple Intersecting Faults

Estimated work time: 1–3 hours

Griddle provides powerful tools for cleaning, preparing, and meshing very large models. This tutorial describes a workflow that can be used to prepare a large open pit mine model for volume meshing. The model contains multiple intersecting faults and stratigraphic layers. Even though the model is artificial, the initial surface meshes resemble those that are often obtained from minimally processed field data. Note that due to the complex workflow, the user may get slightly different results than those presented in the tutorial; however, this should not affect the workflow as a whole. Additionally, it is assumed that advanced **GInt** and **GSurf** parameters are initially configured to the defaults; this can be achieved by restarting *Rhino* or resetting the parameters through the option **GInt**, **GSurf** → *AdvancedParameters* → *Reset*.

Navigate to folder "TutorialExamples\7_LargeOpenPitMine", open file "T7_LargeOpenPitModel.3dm", and examine the model (Figure 74, Figure 75). Create a copy of this initial project with a different name at a desired location (use **File** → **Save As**).

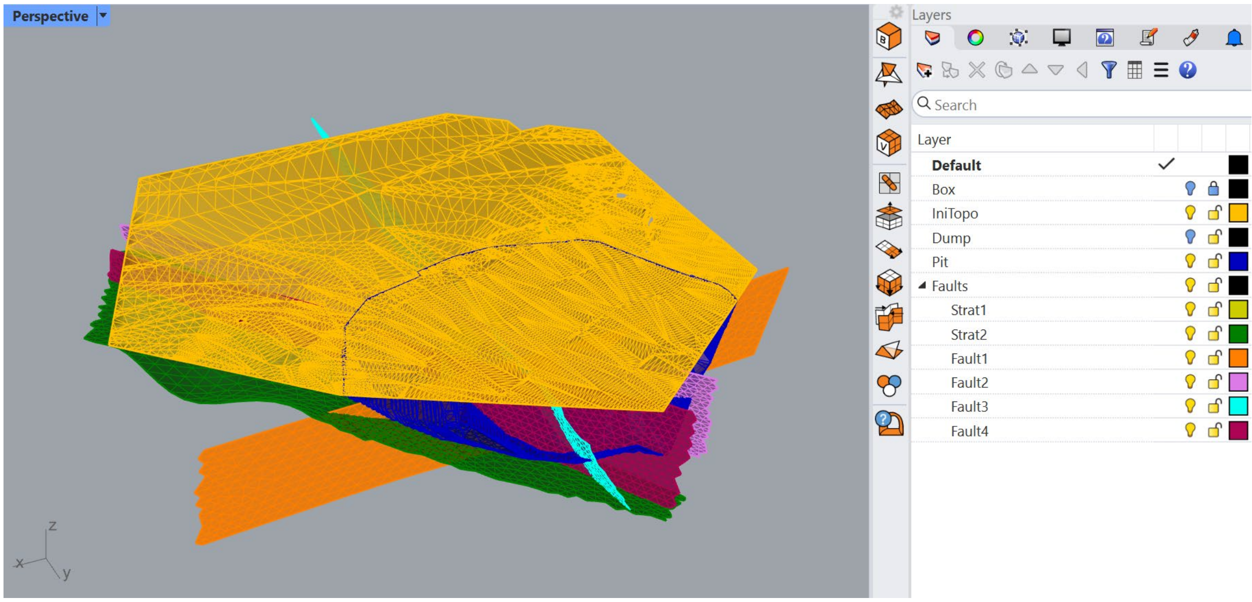


Figure 74: Initial model with topographic surface shown. Ghosted view.

Only a part of the open pit mine is provided in the model. It contains the area of interest (yellow dotted line in Figure 75) and includes the pit wall that may have the potential for instability due to several faults and a stratigraphic boundary crossing it and due to a large pit dump above the wall.

Upon examination of the initial model, the following observations can be made:

- There is a single mesh in each layer. Meshes do not have specific names (in *Properties* panel / **F3**).
- Layer "Box" is locked, and it contains a polysurface representing the desired modeling domain. The layer is locked to avoid accidental modification of objects within the layer. To enable selection and modification of the objects, unlock the layer.

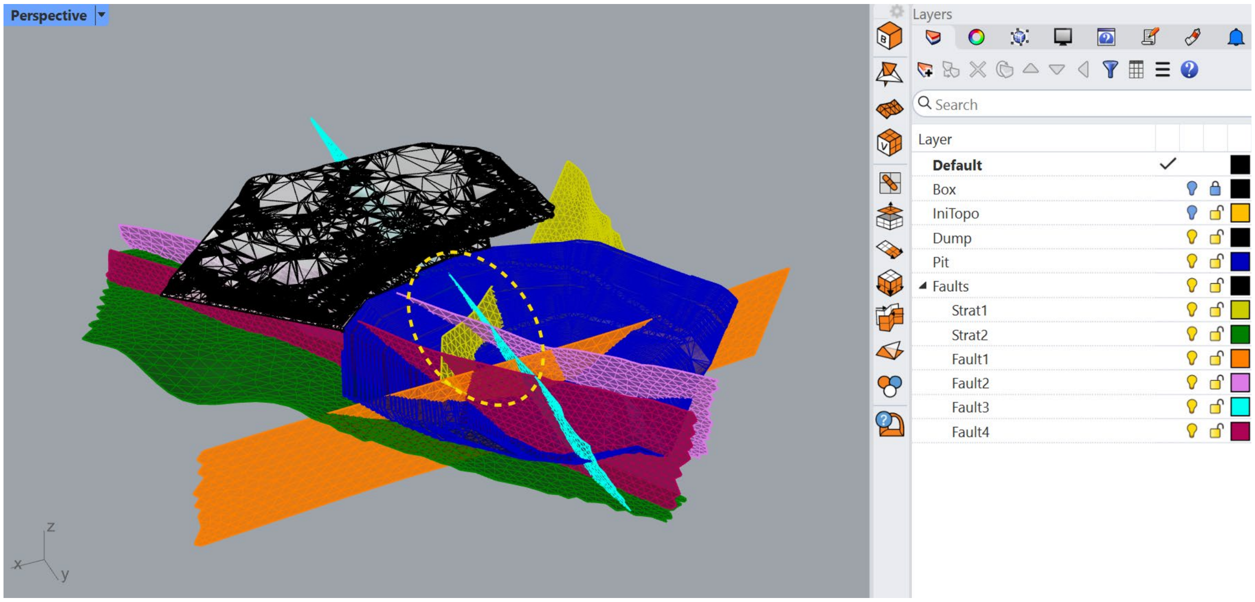


Figure 75: Initial model with large pit dump shown. Ghosted view.

- Surface meshes in layers “Pit”, “Dump”, and “IniTopo” appear to be minimally processed and contain a very large number of faces; face reduction may be needed. All other meshes appear to be remeshed and trimmed to roughly match the size of the modeling domain.
- The surface mesh corresponding to the initial topographic surface (layer “IniTopo”) visibly contains multiple holes and possibly other issues.
- None of the meshes appear to be conformal at the areas of contact or intersection.
- Some of the faults and stratigraphic boundaries seem to terminate on one another. However, it is not clear if meshes are in full contact or if there are small gaps between the meshes.
- Neither “PitDump” nor “Pit” meshes fully connect/intersect to the initial topographic surface. To verify if two meshes intersect each other, use the command **_MeshIntersect**. The command will create polylines tracing the intersecting mesh parts (do not forget to delete these polylines afterwards).
- All meshes are located away from zero coordinates but not so far as to have any effect on the accuracy of calculations. Thus, there is no need to move the model closer to the origin.

Preparation of the pit, pit dump, and topo meshes

1. Turn off all layers except for “IniTopo”. Select the topographic mesh and reduce the number of faces in the mesh using **_ReduceMesh** command. In the pop-up dialog, specify to reduce the polygon count by 50%. The command will decrease the mesh face count while minimizing geometric distortion which will automatically lead to the reduction in the number of mesh issues.
2. To make the mesh more uniform, remesh it with **GSurf** (): *Mode* = *Tri*, *MinEdgeLength* = 20, *MaxEdgeLength* = 100, *RidgeAngle* = 20°, and keep other parameters at the defaults.
3. After remeshing, some of the initial mesh problems are resolved (e.g., clashing faces). However, the mesh still contains several large holes and a small number of clashing faces. Fix these issues using **GHeal** ( ) → *AutomaticHeal* → *IssuesToFix*: *FixClashingFaces* = *Yes* and *FillHoles* = *Yes* with all other functions set to *No*. Keep all other parameters at their default values. This mesh contains

disjoint pieces (separate mesh objects that are not connected but are still one object) and **GHeal** will prompt a warning if *ShowWarnings* = *Yes*. This warning can be ignored in this situation (a small disjoint piece will be removed during **GHeal** operation). After this, ensure that there are no more holes (turn on/off "IniTopo" layer) and then delete the "GHEAL_OUTPUT" layer.

4. Remesh the topo mesh again with the same parameters used in Step 2.
5. Turn off layer "IniTopo" and turn on layer "Dump". The mesh representing the pit dump consists of a large number of non-uniform triangular faces and working with it may be difficult (in particular, remeshing). To check the number of faces in a surface mesh, use the command **_PolygonCount**; the command reports that there are 204,360 triangular polygons in the mesh.
6. There is no need for such a highly accurate representation of the pit dump, so the number of faces in the mesh can be safely reduced. Select the pit dump mesh and use the command **_ReduceMesh** to reduce the polygon count by 85%. Keep all other parameters at the defaults.
7. Turn on layers "IniTopo" and "Pit". Zoom in at any region where meshes visually intersect (e.g., Figure 76). One can find that in certain regions, the meshes overlap and protrude through one another or, conversely, do not fully connect. This is typically due to obtaining meshes from different sources and initial mesh processing (including mesh separation, intersection, and remeshing).

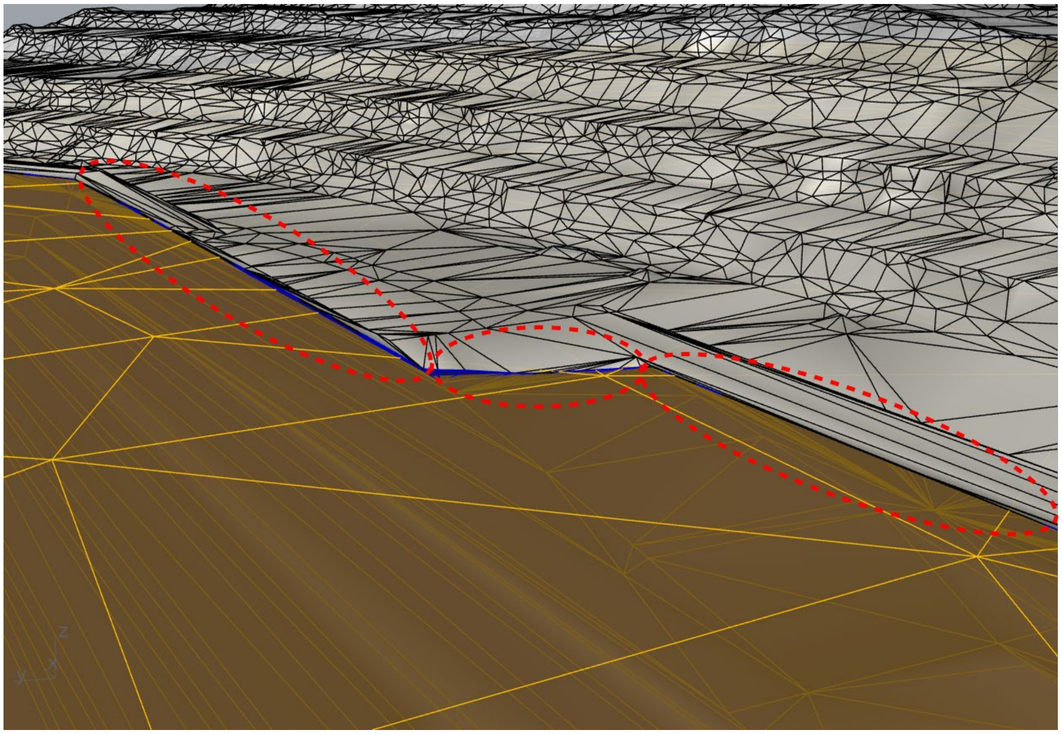


Figure 76: Zoomed-in view of the intersection between the pit, pit dump, and topo meshes. Marked regions denote some of the areas where intersections are not perfect.

Having several meshes (with rather different face sizes) overlap and/or terminate in close proximity to one another may present difficulties when using mesh intersector **GInt**. In this case, it can be challenging to find such a **GInt** tolerance that would be large enough to close all gaps and small enough that undesirable mesh deformations do not occur. Also, these nearly overlapping faces present similar challenges when remeshing and proceeding with volume meshing.

The simplest way to approach this challenge is to spatially separate meshes, so there are only two meshes intersecting along the same line, making it much easier to select a **GInt** tolerance to prepare meshes for final intersection and remeshing.

In order to separate the meshes for this model, a thin boundary layer will be extracted from the pit dump mesh along a part of its boundary. After that, the pit dump mesh will be extended downward and intersected with the topo mesh.

8. Turn off layer "IniTopo" and "Pit" layers.
9. To extract a thin boundary layer, the pit dump mesh must be remeshed such that it contains small elements (faces) along the boundary. Select the pit dump mesh and remesh it with **GSurf** ( ): *Mode* = *Tri*, *MinEdgeLength* = 5, *MaxEdgeLength* = 5, and *RidgeAngle* = 60°. Large *RidgeAngle* value is used to smooth the mesh.
10. Select the pit mesh and use the **GExtract** ( ) → *Boundary* option with *ExtractionLayer* = *CreateOrAppend*. This operation will extract a single layer of faces along the boundary of the mesh. Repeat this operation one more time to extract the next boundary layer of faces such that the two outermost "rings" of boundary faces have been separated from the dump mesh (Figure 77). Note that both "rings" will be placed in a newly created layer called "Dump_Extracted".

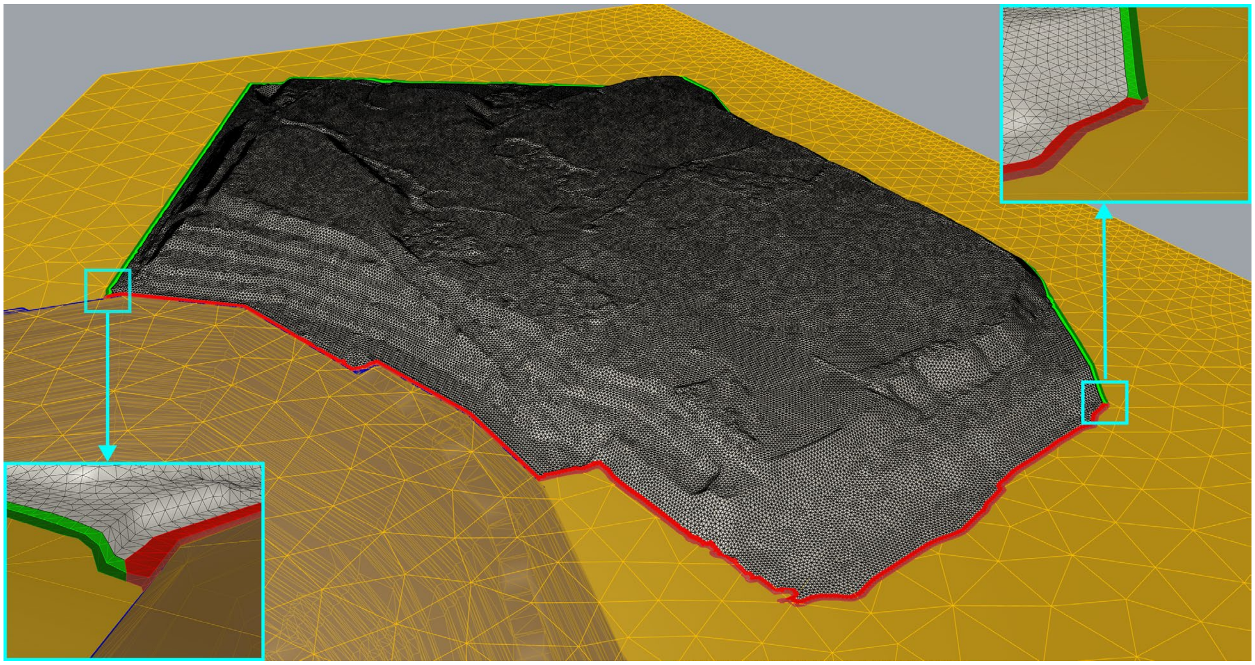


Figure 77: Boundary faces extracted from the pit dump mesh (note that colors are changed from the default gray color). Faces in red were separated and will be deleted.

11. Hide the main pit dump mesh using the **_Hide** command. The only objects that should be visible are the two thin boundary layers of the dump mesh.
12. Using the **_ExtractMeshFaces** command, extract the faces from the boundary layers that are closest to the pit boundary and those that almost overlap the topo mesh (shown in Figure 77 in red). While the command is in face selection mode, other layers can be turned on and off and specific areas can

be zoomed in/out to help with selection (use the **_Zoom** command while in selection mode). Note that faces can be removed from the selection when pressing **Ctrl**.

13. After extracting faces from the layers, delete them (only faces in red in Figure 77). This will create a thin gap between the pit mesh and the dump mesh and will avoid having nearly overlapping faces between the pit dump and topo meshes.
14. Unhide the rest of the dump mesh with the **_Show** command. First select the remaining thin layers of faces and add the main pit dump mesh to the selection. Type the **_Join** command to join all meshes. Verify the mesh is in the "Dump" layer and delete the recently created "Dump_Extracted" layer. Now a small gap between the pit and the dump meshes can be clearly visible as in Figure 78 (turn on "Pit" layer to check). Make sure the mesh is in the "Pit" layer

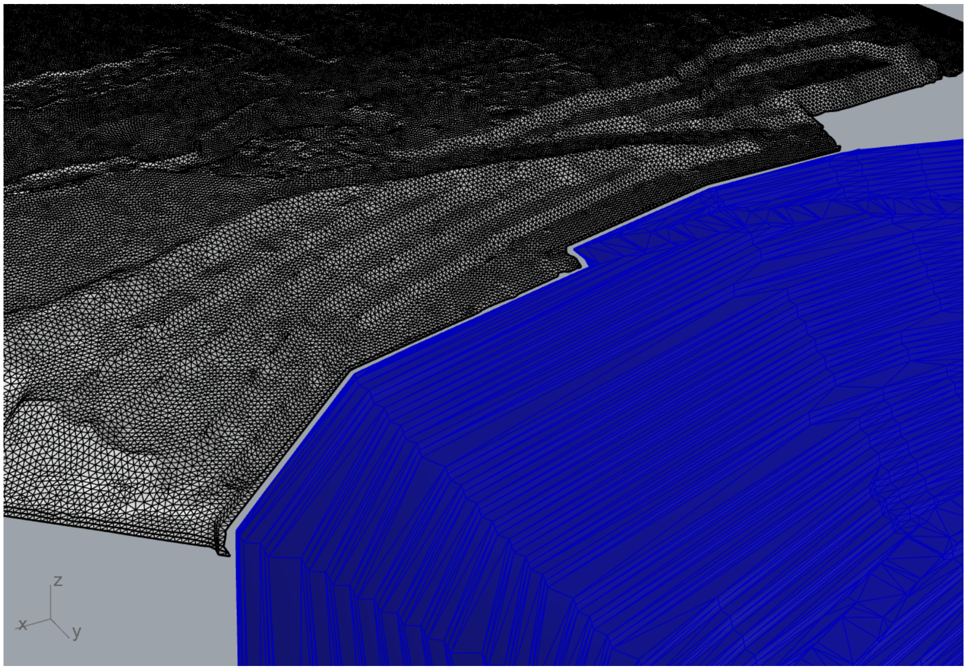


Figure 78: A gap between the pit and pit dump meshes.

15. Select the pit dump mesh and remesh it with **GSurf** (): *Mode* = *Tri*, *MinEdgeLength* = 20, *MaxEdgeLength* = 100, and *RidgeAngle* = 60°. This operation will further reduce the number of mesh faces and smooth it, making it easier to work with the mesh. Remeshing may take a minute.
16. Turn off all layers besides "Dump" and "IniTopo".
17. Use **GExtend** () → *ExtendToMesh* with the following parameters: *Direction* = *AlongVector*, *TrimExtension* = *Yes*, *MinTolerance* = 0.*001* and *MaxTolerance* = *0.01*. Select the "Dump" mesh as the target and the "IniTopo" mesh as the base. The mesh boundary will be highlighted in magenta; select the whole boundary (**Ctrl**+**A**) and press **Enter** (Figure 79). Use 0,0,0 as the start of the extension vector, and 0,0,-1 as the end to extend the mesh boundary straight down.

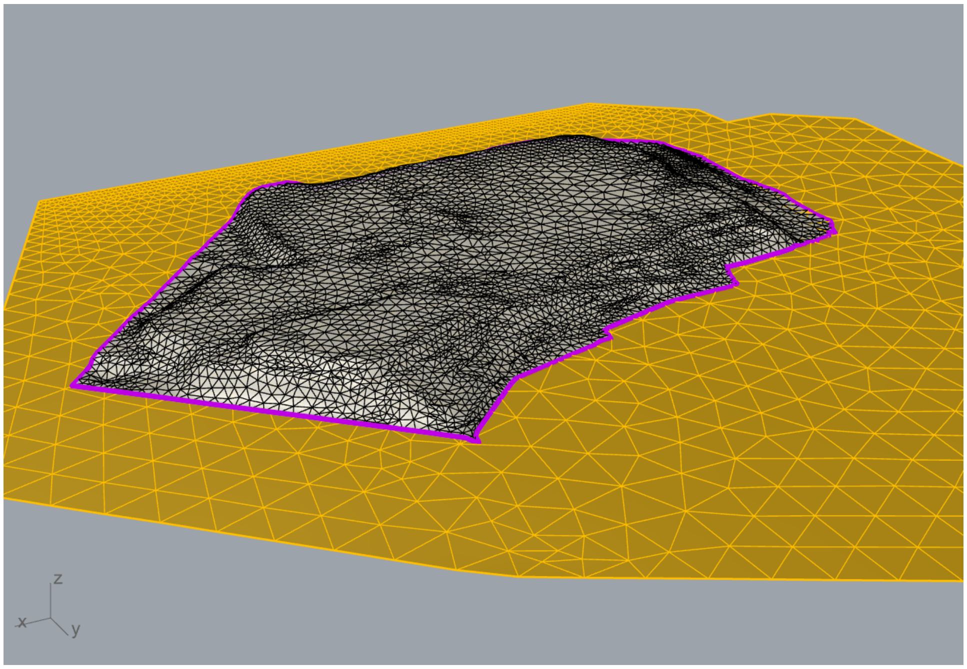


Figure 79: Selecting the mesh boundary to extend

Once the process is complete, check the intersection by selecting both meshes and using the command **_MeshIntersect**. Only one curve should be computed (Figure 80), indicating a single continuous intersection. Delete this curve when done.

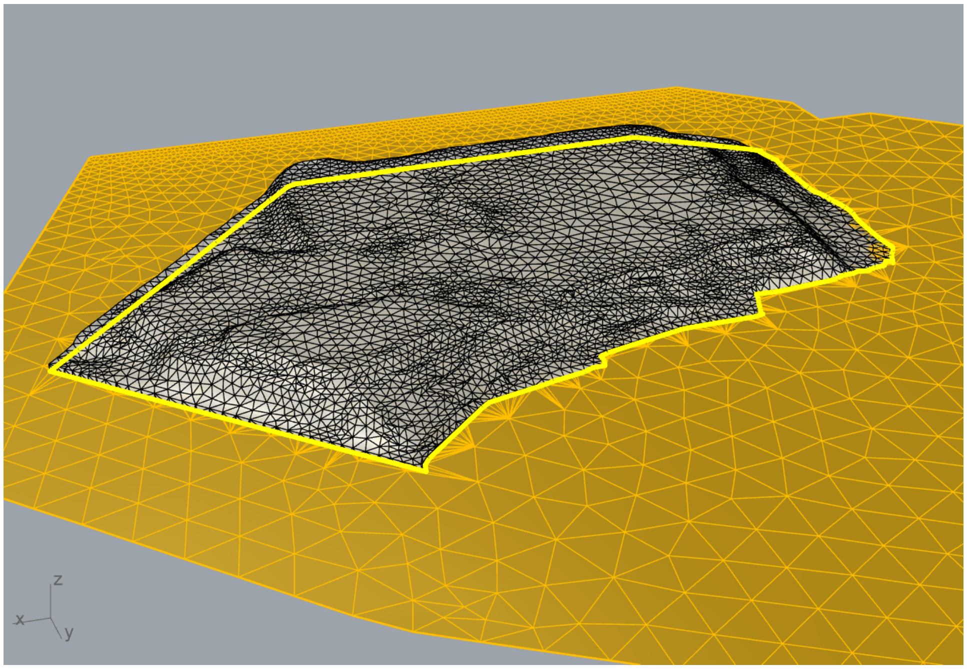


Figure 80: Extended pit dump mesh fully intersecting the topo mesh.

Now the dump and topo meshes fully intersect each other. A similar process will be used to extend a part of the pit mesh; however, the pit mesh must be remeshed first.

18. Turn on the “Pit” layer and turn off all others. Remesh the pit mesh with **GSurf** (): *Mode* = *Tri*, *MinEdgeLength* = 10, *MaxEdgeLength* = 10, and *RidgeAngle* = 20°. Select the mesh and extract the border with **GExtract** ( ) → *Boundary*, but this time with *ExtractionLayers* = *DoNotCreate*; this will result in no “_Extracted” layer being created, and the extracted piece will remain in the same layer the parent mesh lies in. Upon completion, delete the entire border while it is still selected.
19. Turn on the “IniTopo” layer. Extend the part of pit mesh boundary upward to the topo mesh using **GExtend** → *ExtendToMesh* selecting the upper bounds of the pit mesh (Figure 81) and using 0,0,0 as the start of the extension and 0,0,1 as the end to extend directly upward. Proper intersection can be checked using **_MeshIntersect** with the two meshes.
20. Navigate to the *Layers* panel and unlock layer “Box” (click on the lock icon). Turn on the layer if it is turned off. Mesh the box polysurface with the **_Mesh** command, using simple controls and the fewest number of polygons. After meshing, the polysurface is still selected. Delete it. The result is shown in Figure 82.

At this point, the pit, pit dump and topo are fully intersected with each other, and are in full contact with the box. Now the internal meshes can be easily intersected with the outer box. The approach described below intersects meshes and then deletes sub meshes using **_MeshTrim**[5].

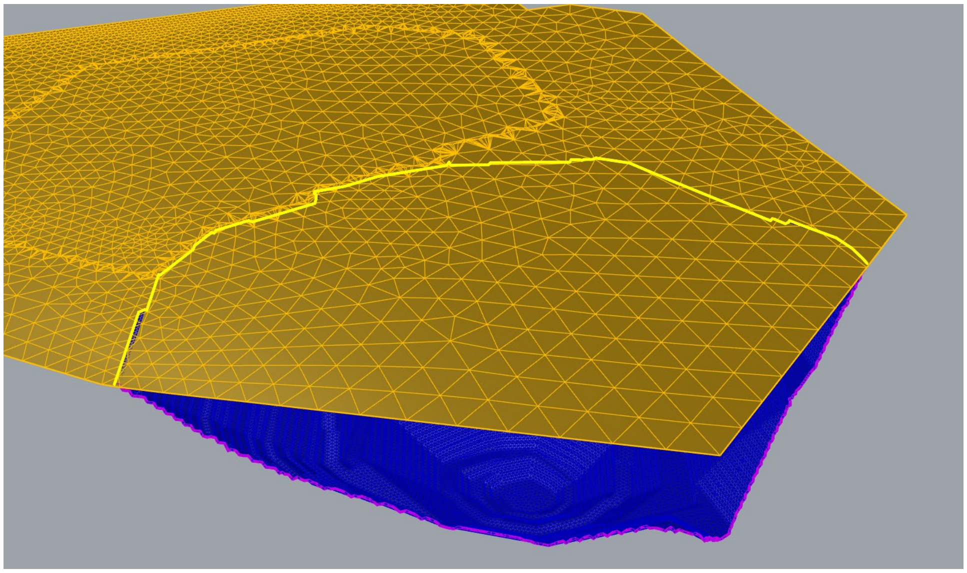


Figure 81: Selection of a piece of the pit boundary for mesh extension.

21. Turn on and select the pit, pit dump, topo, and box meshes and intersect them with **GInt** ( ) → *Tolerance* = 0 and keep other parameters at the defaults.
22. Set the view to *Shaded* and run **_MeshTrim.** Select the box as the cutting object and the outer parts of the topo and pit mesh (highlighted in red in Figure 83) as objects to trim, to remove the parts of those meshes that extrude the box mesh. Press **Enter** to exit the command when the two meshes have been trimmed.

[5] An alternative approach is to use *Rhino*’s **_MeshSplit** command.

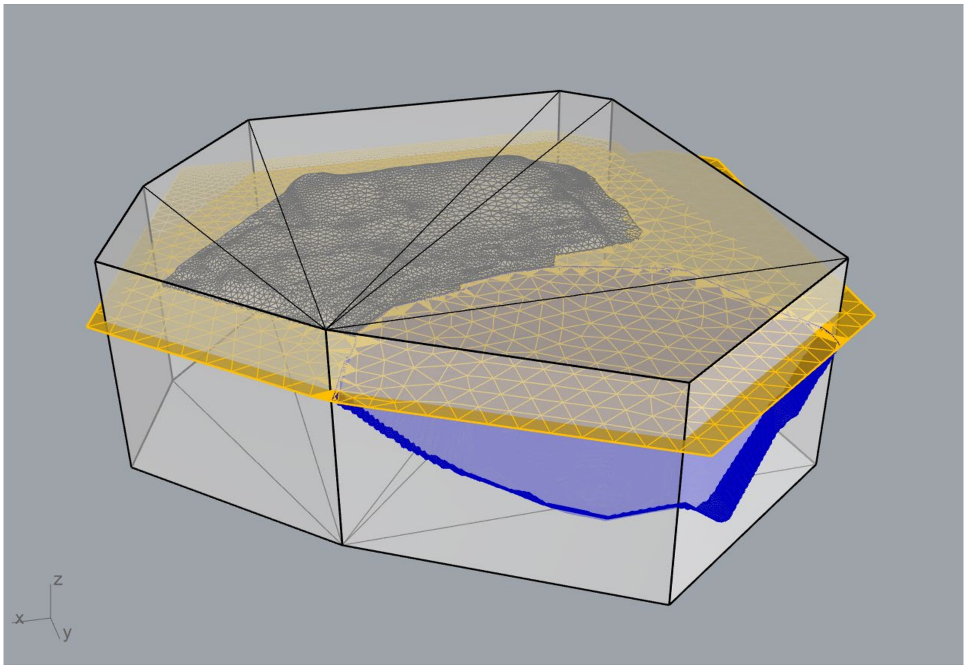


Figure 82: The box, pit, and dump meshes with the last two extended and intersected with the topo mesh.

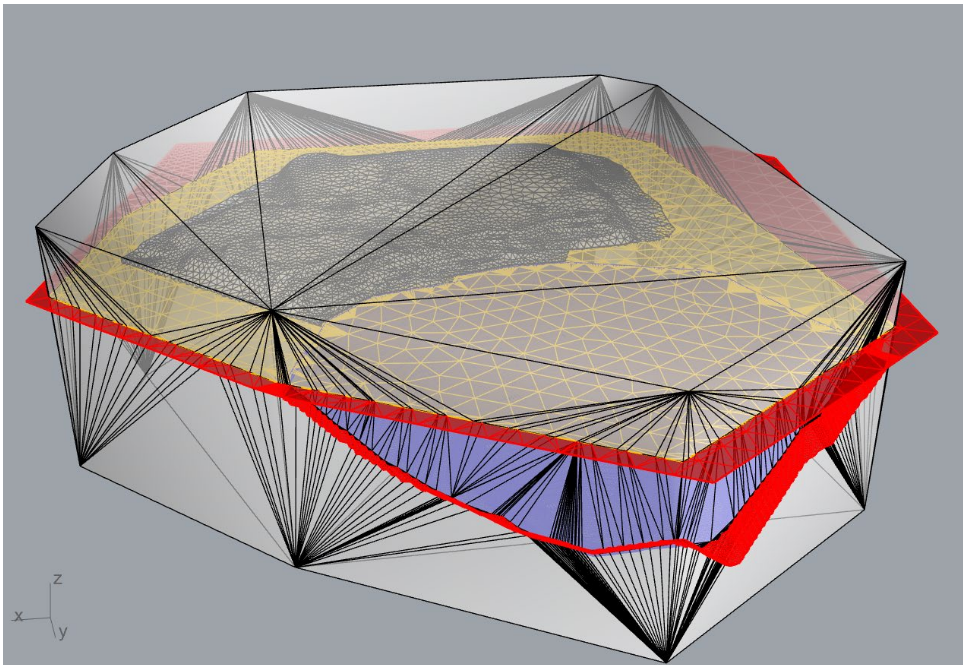


Figure 83: Trimming the topo and pit meshes.

23. Similarly, trim (or split and delete) the upper section of the box mesh by the topo mesh (see Figure 84 – it shows already remeshed meshes).
24. Verify the success of the extensions and trims by selecting all meshes with **Ctrl**+**A** (resulting in 4 meshes being added to the selection) and running **_MeshIntersect**. This should result in 3 or less curves being computed, which validates proper connections. If the results show more meshes or curves, most likely the meshes were not intersected or extracted properly; undo and repeat the last several steps according to the instructions. Delete the curves once done.

25. Select all meshes and use **GHeal** (🩹) → *ShowErrors* to check if there are any internal naked edges. Turn off layer "Clashing Faces" (they will be fixed by remeshing), unlock the layer "GHEAL_OUTPUT" and type the command **_SelCrv** to select the outlines of naked edges. The command should report that only four curves are selected. If this is the case, delete the output layer and continue. If more than four curves are present, there are holes within some of the meshes, which is likely due to incorrect mesh intersection or extraction; undo and repeat the last several steps according to the instructions.
26. All four meshes should be remeshed to make them more uniform and to simplify further work. First, assign a specific element size to the following meshes in the hyperlink field of the mesh properties panel (**F3**):
    - The pit dump mesh: *elemsize:25*
    - The pit mesh: *elemsize:15*

    Then use **GSurf** (◈): *Mode* = *Tri*, *MinEdgeLength* = 10, *MaxEdgeLength* = 100, and *RidgeAngle* = 20 to remesh all four meshes. The resulting meshes should look like that in Figure 84.

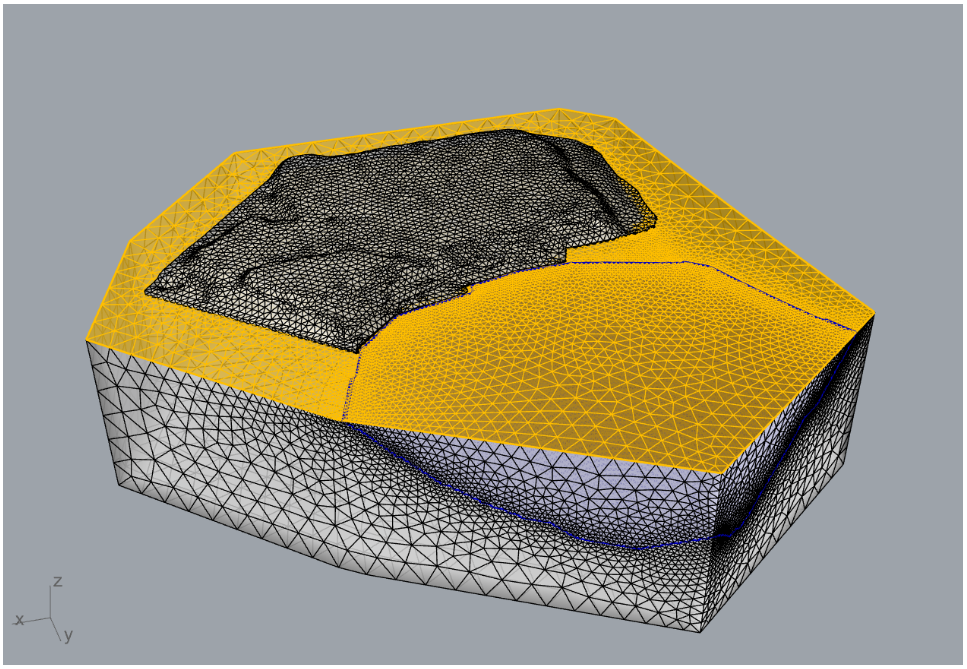


Figure 84: Intersected box, pit, pit dump, and topo meshes.

At this point, a volume mesh could be created from the surface meshes shown Figure 84. However, the model does not yet include any faults. The next section provides information about extending, intersecting, and trimming faults before including them into the model.

Preparation of the fault meshes

Turn on "Faults" layers and turn off all others. Explore the areas where meshes visually terminate on one another and/or use the **_MeshIntersect** command to see if there are full intersections between meshes. Mesh "Strat1" should terminate on "Fault1" and the rest of the faults should terminate on "Strat2". Unfortunately, none of these meshes fully intersect/terminate as they should (e.g., see Figure

85). This is a common situation when working with meshes generated from field data. As before, such meshes need to be extended and intersected, however parts of it already protrude through each other, so a more manual process is used to extend, intersect and trim the faults.

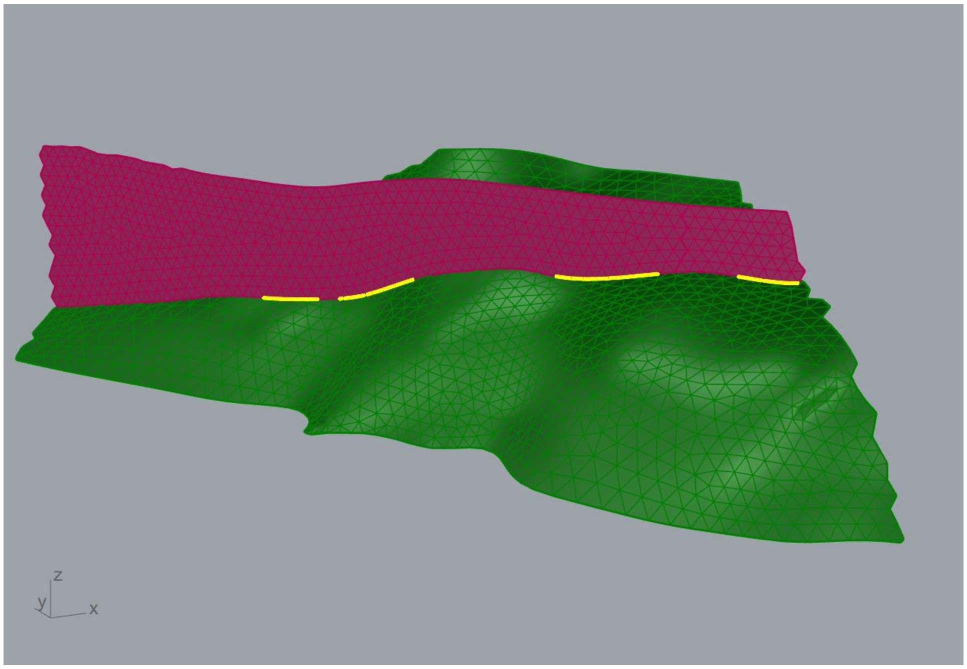


Figure 85: The _MeshIntersect command shows actual intersections (yellow lines) between "Fault4" and "Strat2" meshes. Other parts of the "Fault4" mesh do not intersect the "Strat2" mesh, leaving minor gaps.

27. Turn on only "Strat1" and "Fault1" layers and extend the "Strat1" mesh only along the edge that visually connects to the "Fault1" mesh. Use **GExtend** ()→ *ExtendSelectedBoundary*: *ExtendLength* = 20, *MeshType* = *Merged*, *Direction* = *LocalTangent*. Additional segments of the boundary can be included in the selection by clicking on them and can be removed when holding **Ctrl** and clicking. Ensure that the extended portion fully intersects and is enclosed within the "Fault1" mesh (as in Figure 86).
28. Repeat this operation to extend meshes "Fault2", "Fault3", and "Fault4" so they extend through the "Strat2" mesh. Use the same **GExtend** parameters as in the previous step. Ensure that the extended parts fully intersect and are enclosed within the "Strat2" mesh (zoom in at intersection corners).
29. Turn on layers "Box" and all layers in "Faults" and make sure all other layers are turned off. Intersect all meshes with **GInt** ( ) → *Tolerance* = 0 and keep other parameters at the defaults.
30. Using **_MeshTrim**, delete excessive parts of meshes located outside the box by using it as a cutting object.
31. View only layers "Strat1" and "Fault1". Trim the excess part of "Strat1" mesh that protrudes through the "Fault1" mesh using **_MeshTrim**.
32. Like the previous step, trim away pieces of "Fault2", "Fault3" and "Fault4" meshes which protrude through "Strat2". Do not trim "Fault1" with "Strat2".

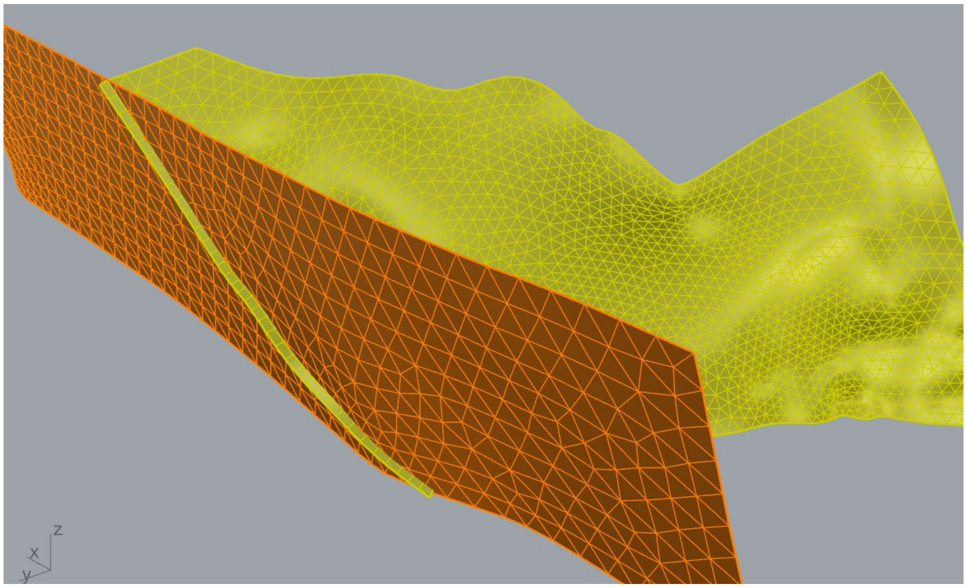
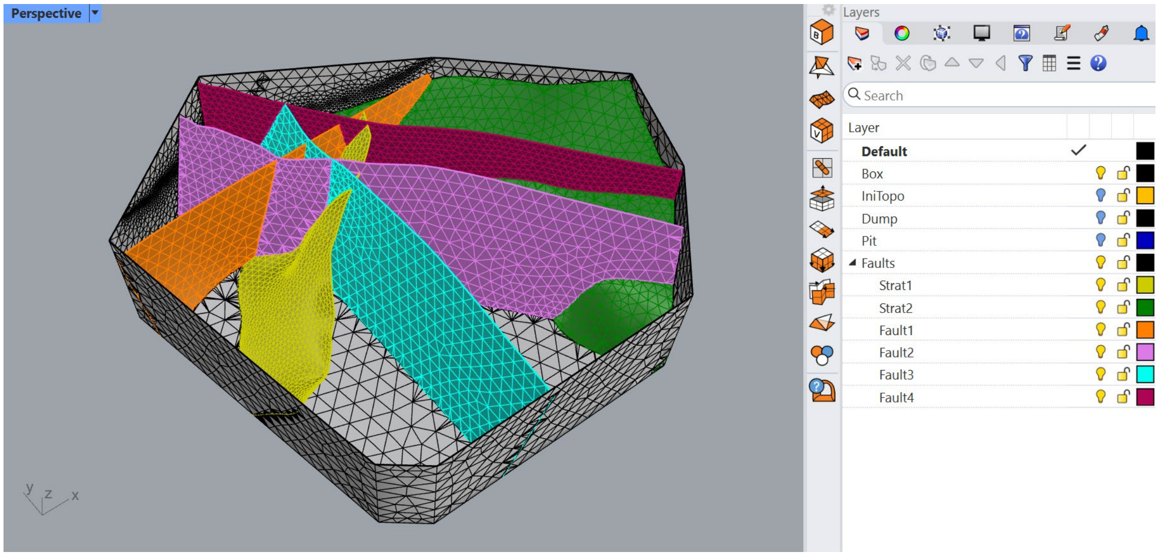


Figure 86: Extension of "Strat1" mesh along the part of the boundary closest to "Fault1".

Turn on all the "Faults" layers and the "Box". Now there should only be seven meshes, as in Figure 87 (check with **Ctrl**+**A**). All these meshes are perfectly intersected between one another and are ready to be intersected with the pit, pit dump, and topo meshes, followed by final remeshing and volume meshing.

Final mesh intersection and remeshing

33. Turn on all the layers. Select all ten meshes and intersect them using **GInt** ( ) → *Tolerance* = 0.005 and leaving all other parameters as their default values.
34. Remesh all meshes using **GSurf** ( ): *Mode* = *Tri*, *MinEdgeLength* = 20, *MaxEdgeLength* = 75, *RidgeAngle* = 15 and *AdvancedParameters*: *MaxGradation* = 0.2, *Optimization* = 8, *ShapeQuality* = 0.7, and *OutputMesh* = *Distinct*. The results are shown in Figure 88.



Layer		
Default	✓	⬛
Box		⬛
IniTopo		🟧
Dump		⬛
Pit		🟦
◢ Faults		⬛
Strat1		🟨
Strat2		🟩
Fault1		🟧
Fault2		🟪
Fault3		🟦
Fault4		🟥

Figure 87: Intersected and trimmed "Faults" meshes.

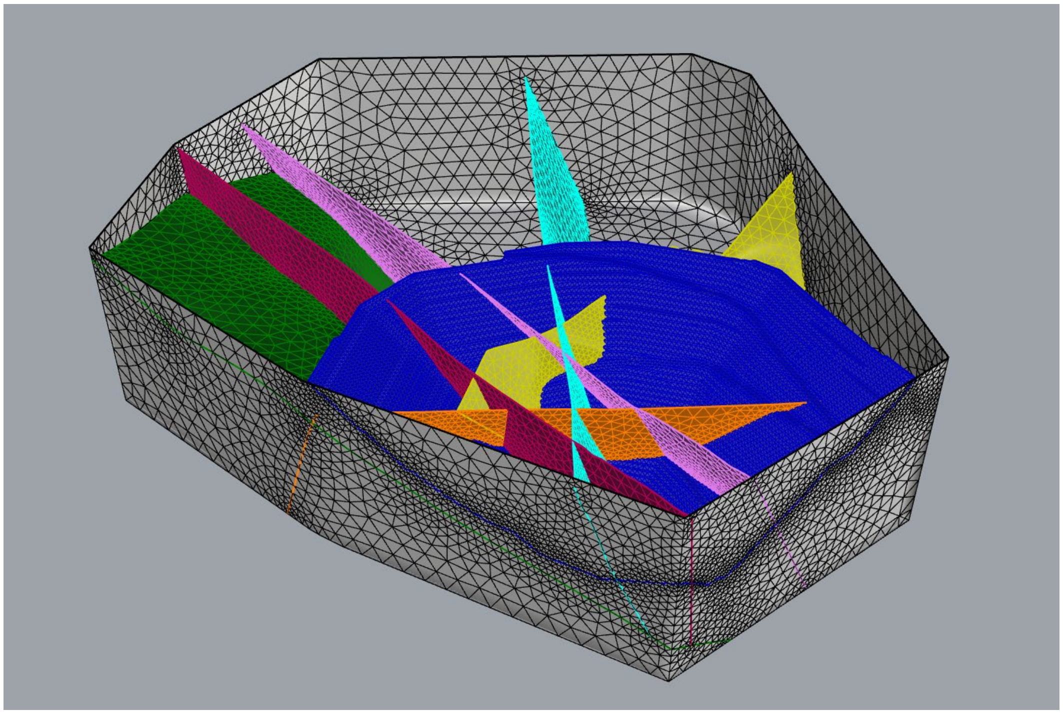


Figure 88: Final triangular surface meshes (the pit dump and topo meshes are not shown).

35. Verify that the final surface meshes do not contain any holes or clashing faces by calling **GHeal** ( ) → *ShowErrors* (select all meshes first). After that, unlock the "GHEAL_OUTPUT" layer and type **_SelCrv**. *Rhino* should report that only 10 curves are selected.

If more than 10 curves get selected, it indicates the presence of further issues. To locate these issues, hide all layers except those in "GHEAL_OUTPUT" and start analyzing the curves in each sublayer. When a sublayer is unlocked, individual curves can be selected, hidden and deleted to help investigate the problematic area. In this situation, delete the 10 largest pink curves in the "Naked Edges" layer which correspond to the outermost boundaries of the 10 meshes. Any additional curves could be investigated by selecting the curve, zooming-in to it (use **_Zoom** → *Selected*), and testing, which original mesh layer this curve corresponds to (turn on and off various mesh layers). Depending on the type of the issue, it may be fixed locally by rebuilding or re-intersecting faces or the whole mesh may need to be intersected and remeshed again with different parameters (for example, higher intersection tolerance).

36. Assign names in the meshes *Properties* panel as desired. These names will be transferred as named face groups (surfaces) to the volume mesh (currently, only for *FLAC3D* and *3DEC*). See details about group naming and surface naming in the *Griddle 3.0 User Manual*.
37. Create a tetrahedral volume mesh in *FLAC3D* binary format by running **GVol** ( ) with *Mode* = *Tet*. Keep all other **GVol** parameters at default values. Tetrahedral volume mesh generation is usually a fast process, and it is used here. In case **GVol** issues a message about the presence of naked edges, read the message and press **Yes** to continue. The final mesh is shown in Figure 89.

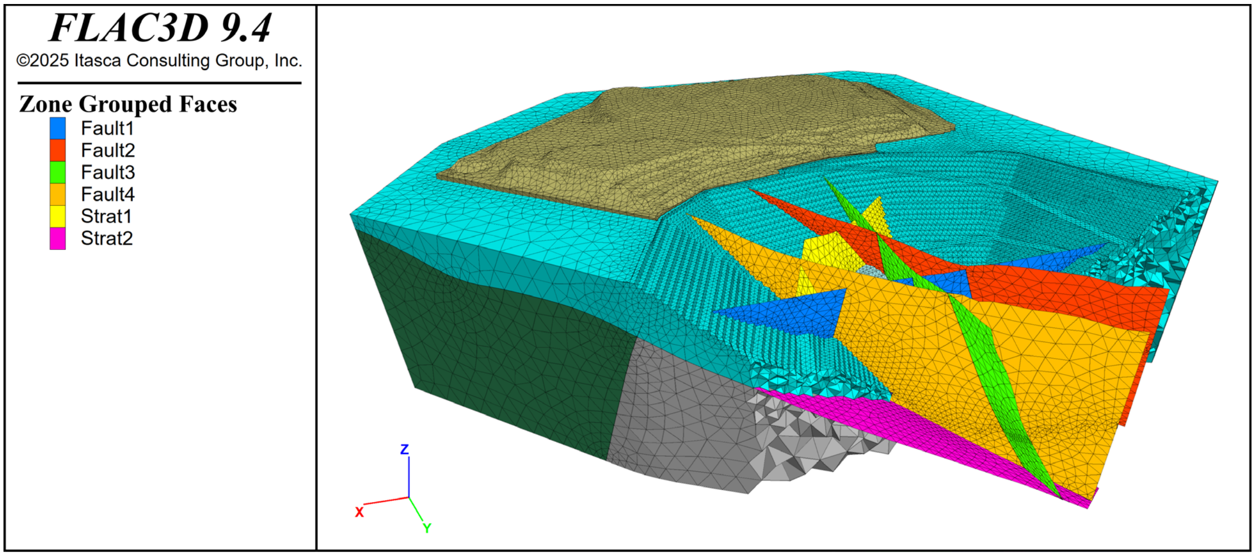


Figure 89: Tetrahedral volume mesh of open pit mine model.

As an optional exercise, remesh all meshes again using the same **GSurf** parameters as in step 34 but with *QuadDom* mesh type. Then generate a hex-dominant (HexDom) volume mesh in *FLAC3D* binary format (which may take several minutes for such a large model). Save the file under a different name than the previous file. Compare log outputs from **GVol** for both cases to see what kind and how many elements are generated.

When generating a hex-dominant volume mesh for the model, **GVol** will produce about half of the total number of elements compared to a tetrahedral mesh. However, a HexDom mesh contains a large number of undesirable pyramidal elements[6]. For such complex models, engineers sometimes generate pure tetrahedral meshes and then split each tetrahedron into several hexahedrons within *FLAC3D* to obtain a pure hexahedral grid (pure hexahedral grids yield significantly more accurate numerical results in *FLAC3D*).

Note that this model is not suitable for outputting in *3DEC* deformable blocks format as faults do not connect to the topo mesh, and, thus, they do not cut blocks within the pit or the rest of the model. *3DEC* rigid block format could be used instead, but the surface meshes will need to have much larger elements to produce a coarser volume mesh that can be loaded into *3DEC* (as *3DEC* will try to detect and create a contact between each rigid tetrahedral element/block, resulting in a very large number of block contacts).

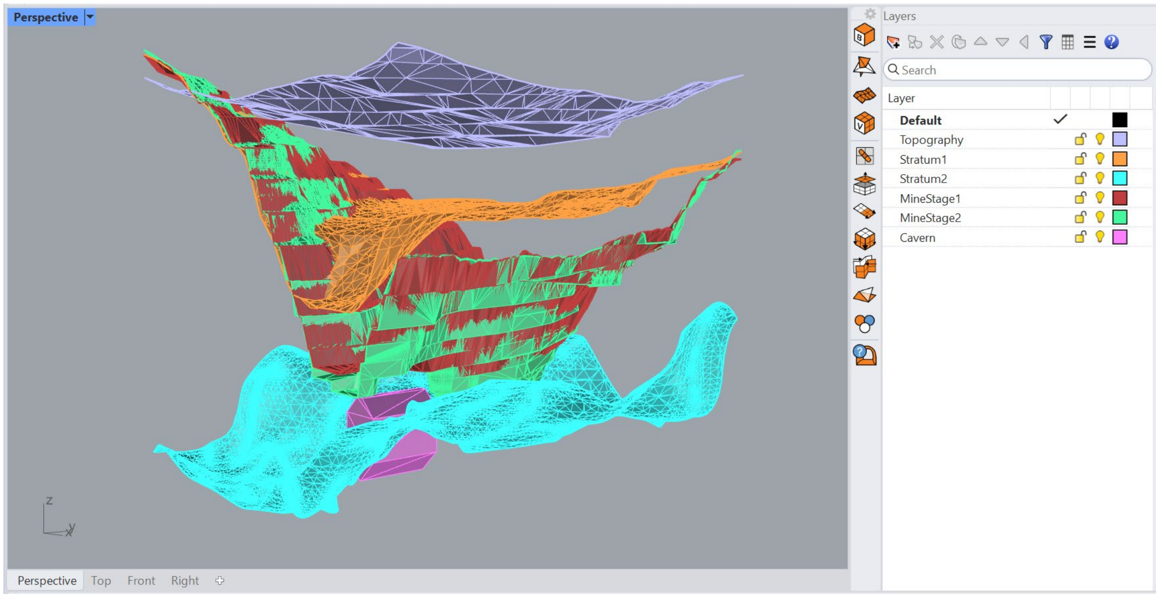
[6] *FLAC3D* grid densification becomes rather challenging if pyramids are present.

Tutorial 8: Open Pit Mine with Overlapping Mining Surfaces

Estimated work time: 1.5–3 hours

The current tutorial further illustrates *Griddle* capabilities of working with complex geometries in preparation for volume meshing. The model under consideration contains two partially overlapping mining stages, two stratigraphic boundaries, a topographic surface, and a cavern (or excavation) located underneath the lower mining surface (Figure 90). After the creation of the volume mesh and importing it into a numerical simulation software, a possible analysis goal would be the estimation of the stability of the mined surfaces/walls and the stability of the rock bridge between the lower mining surface and the cavern. The complexity of this model is in that some of the surface meshes overlap forming many small volumes between them, which would lead to difficulties intersecting all the meshes, remeshing them, and creating a volume mesh (which may end up not suitable for a numerical analysis). This model is representative of practical problems in which geometries representing various geological, designed, or measured structures may overlap.

Navigate to folder "TutorialExamples\8_MineTwoStages", open file "T8_MineTwoStages.3dm", and examine the model (Figure 90). Create a copy of this initial project with a different name at a desired location (use **File** → **Save As**).



Layer			
Default	✓		⬛
Topography	🔓	💡	🟪
Stratum1	🔓	💡	🟧
Stratum2	🔓	💡	🟦
MineStage1	🔓	💡	🟥
MineStage2	🔓	💡	🟩
Cavern	🔓	💡	🟪

Figure 90: Initial model with various geological and mining geometries shown.

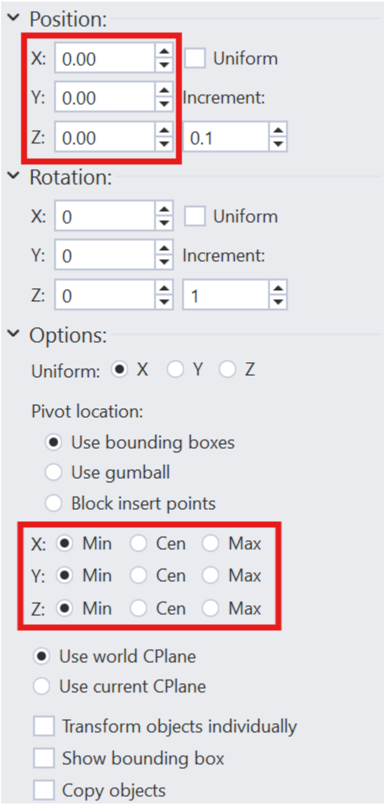
The model contains minimally processed data obtained from measurement devices (such as LIDARs) in combination with designed/planned geometries. Upon the examination of the model, the following observations can be made:

- The objects are located far away from the origin (zero coordinate) – field coordinates were used. For accurate operations in *Rhino* and *Griddle* (and consequently in numerical model software), it is important to have objects coordinates at the same level of magnitude / range as the size of the model. To check this information, select all meshes and type **_BoxEdit** command which opens the *BoxEdit* panel. In this panel, observe the Size of the objects vs. the Position: the current model has overall size in the range of hundreds of meters, while its' position is in the range of thousands of kilometers. This disparity will likely lead to the loss of numerical accuracy and to potential issues in various operations. To fix this, the whole model must be translated closer to zero coordinate, while preserving all its components and their sizes.

- Two surface meshes in the model have unreasonably high element density for the geometry they represent. These are Stratum2 and MineStage2. The command **_ReduceMesh** will have to be used to reduce (decimate) the number of faces in these meshes before further processing. Using **_ReduceMesh** is often necessary when working with initial surface meshes containing more than a few hundred thousand faces. Otherwise, *Rhino* and *Gridlle* operations may become very slow, and *Rhino* may consume large amount of computer resources.

Preparation of the model

1. Make sure that all layers are shown and no objects are hidden (use the **_Show** command). Select all the meshes (Ctrl+A), type the **_BoxEdit** command, and set the Position and other parameters shown in Figure 91 within the red boxes only:



Position:

X: 0.00 □ Uniform

Y: 0.00 Increment:

Z: 0.00 0.1

Rotation:

X: 0 □ Uniform

Y: 0 Increment:

Z: 0 1

Options:

Uniform: ● X ○ Y ○ Z

Pivot location:

● Use bounding boxes

○ Use gumball

○ Block insert points

X: ● Min ○ Cen ○ Max

Y: ● Min ○ Cen ○ Max

Z: ● Min ○ Cen ○ Max

● Use world CPlane

○ Use current CPlane

□ Transform objects individually

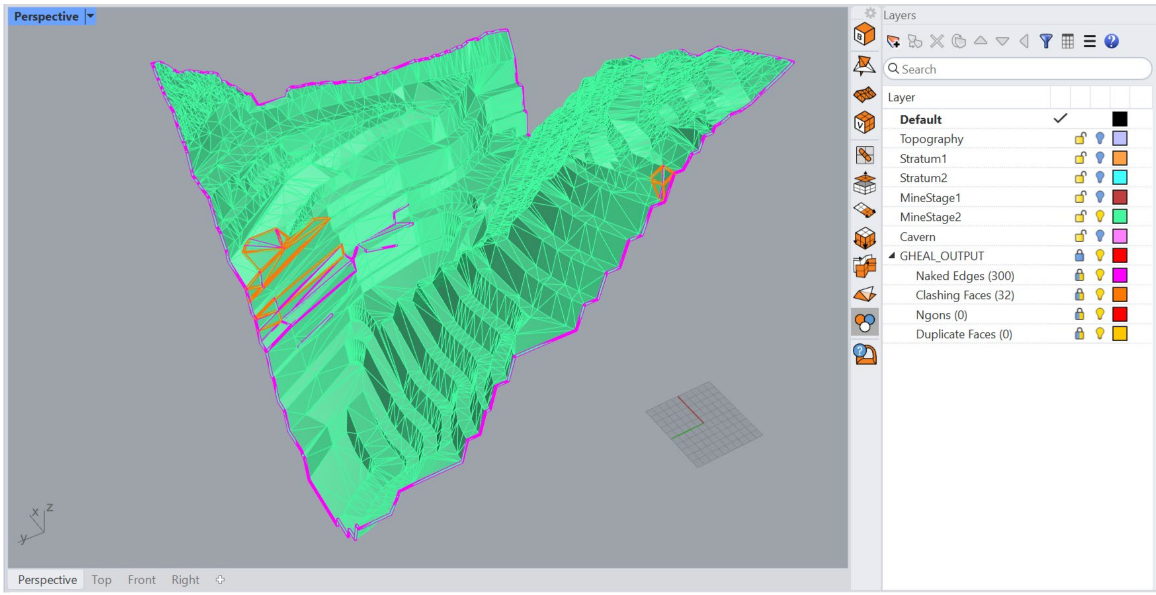
□ Show bounding box

□ Copy objects

Figure 91: The Box Edit parameters used to translate the model.

After that press on the *Apply* (edits) button. Using these parameters will translate the whole model in a way that one of the corners of a box bounding all the selected objects is at zero coordinates. The corner with smallest coordinates is used here and this is done for convenience only (in general, the users may equally choose the centroid or the corner with the maximum coordinates). An alternative way to translate objects is to use the **_Move** command.

2. Turn off all the layers except for “MineStage2”. Select the “MineStage2” mesh and reduce it by 50%. Use the **_ReduceMesh** command. In this case the command is not strictly necessary, but it provides the first pass of mesh clean-up and mesh quality improvement, while reducing the number of faces in the mesh.
3. Even before the mesh reduction, the “MineStage2” mesh contained many naked edges and clashing faces (this can be easily checked with **GHeal**→*ShowErrors*). After the mesh reduction most of these issues are still present; use **GHeal**→*ShowErrors* to reveal them (Figure 92). The issues can be fixed automatically by using **GHeal**→*AutomaticHeal*. In this case, keeping all the parameters at the default values and selecting *IssuesToFix=All*, resolves all the problems. After the operation, delete all the layers and objects in “GHEAL_OUTPUT”.



Layer	
Default ✓	■
Topography	
Stratum1	
Stratum2	
MineStage1	
MineStage2	
Cavern	
GHEAL_OUTPUT	
Naked Edges (300)	
Clashing Faces (32)	
Ngons (0)	
Duplicate Faces (0)	

Figure 92: Naked edges and clashing revealed by GHeal in the “MineStage2” mesh.

4. Use **GSurf** to remesh the “MineStage2” mesh to have better quality and more adequate size elements for further operations. Use *Mode=Tri*, *MinEdgeLength* = 4, *MaxEdgeLength* = 8, *RidgeAngle* = 30, and keeping the other parameters at the default.
5. Turn off all active layers and turn on “MineStage1” only. Use **GHeal**→*ShowErrors* to see if there are any issues with the mesh – it contains a few clashing faces. In this case it is related to the presence of a sliver face between the clashing faces outlines. As noted in *Griddle Manual*, clashing faces are reported not only for truly intersecting or overlapping faces within a mesh, but also for the faces (and their neighbors) which size is smaller than the model tolerance[7].

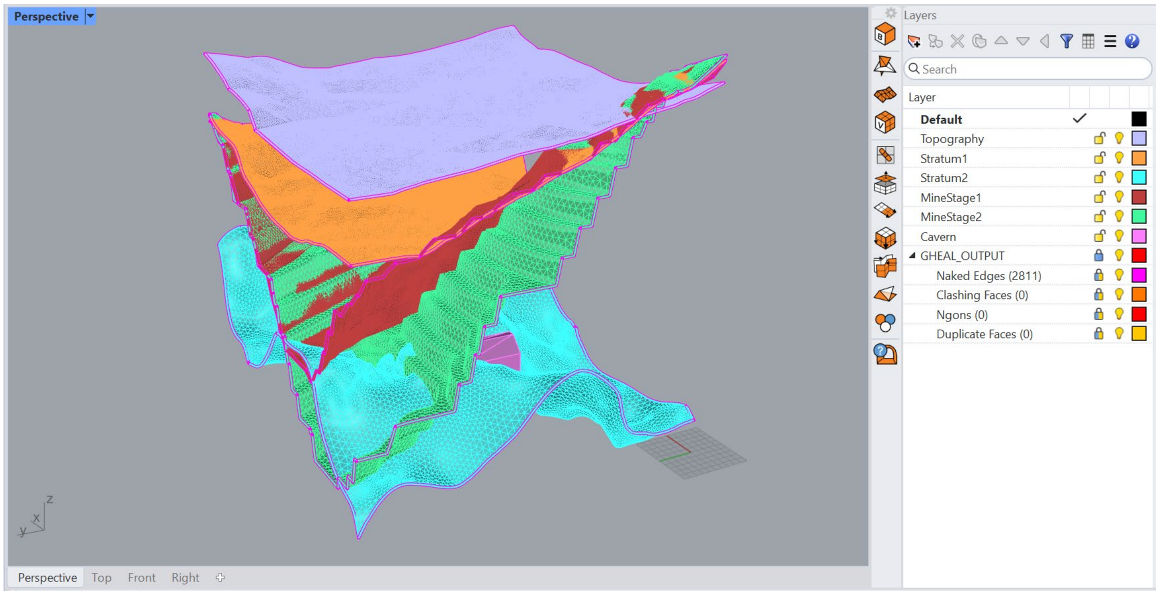
   In many cases, mesh remeshing may remove sliver faces and resolve clashing faces issues, so no further **GHeal** operations would be needed. Remesh the “MineStage1” mesh with **GSurf** using

[7] More precisely, a face and its neighbors are considered clashing if the face size (along any of the edges) is less than the model Tolerance multiplied by the *ToleranceMultiplier* set in **GHeal** → *AutomaticHeal* → *AdvancedParameters*. By the default, *ToleranceMultiplier* = 1.

Mode=Tri, *MinEdgeLength* = 2, *MaxEdgeLength* = 5, *RidgeAngle* = 30, and keeping the other parameters at the default. Afterwards verify with **GHeal** that all issues are gone.

Note that it is important to remesh “MineStage1” and “MineStage2” meshes separately as they overlap in some areas, and **GSurf** may produce holes when remeshing overlapping faces.

6. Turn off all active layers and turn on “Stratum2” only. Use the **_ReduceMesh** command to reduce the number of faces by 80%. Check with **GHeal**→*ShowErrors* that the mesh does not have any issues and only the naked edges on the exterior boundary are reported. Remesh it with **GSurf** using: *Mode=Tri*, *MinEdgeLength* = 3, *MaxEdgeLength* = 10, *RidgeAngle* = 35.
7. Now turn on “Stratum1” layer and remesh the mesh with **GSurf** using *Mode=Tri*, *MinEdgeLength* = 2, *MaxEdgeLength* = 4, *RidgeAngle* = 35.
   Do the same for the “Topography” mesh.
8. Basic clean-up is complete. Show all the layers and meshes and use **GHeal**→*ShowErrors* to confirm that there are no naked edges or any other issues within the meshes (the naked edges should only be present on the exterior/real mesh boundaries). The results should be similar to what is shown in Figure 93 (the number of naked edges may differ slightly from Figure 93). After this, delete “GHEAL_OUTPUT” with all its content and incrementally save the model.



Layer
Default ✓
Topography
Stratum1
Stratum2
MineStage1
MineStage2
Cavern
GHEAL_OUTPUT
Naked Edges (2811)
Clashing Faces (0)
Ngons (0)
Duplicate Faces (0)

Figure 93: Cleaned-up and remeshed model meshes with GHeal output showing the presence of naked (real) boundaries only.

Cleaning the overlapping meshes

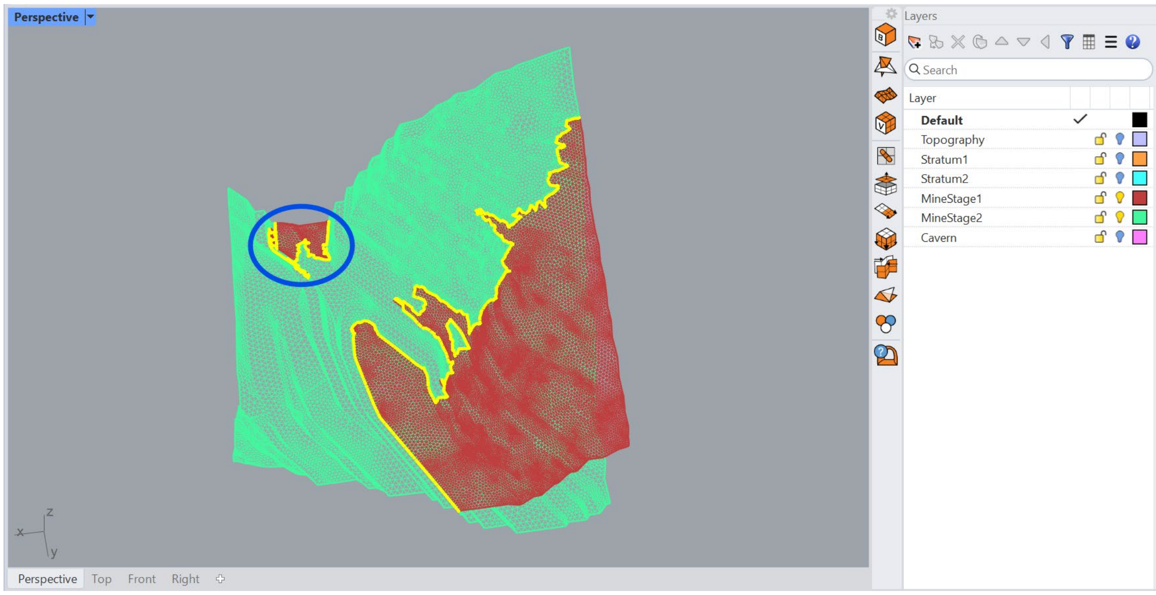
After the initial mesh operations, it is evident that “MineStage1”, “MineStage2”, “Stratum1”, and “Topography” meshes overlap in multiple regions. This geometry is not suitable for further mesh operations and volume meshing. The meshes must be separated within the overlapping regions to clearly define distinct volumes. *Griddle*’s tool **GCollapse** can be used to assist with resolving overlapping meshes. The tool requires selecting one or more base meshes and a target mesh which is collapsed to the nearest surface formed by the union of the base meshes. Users must specify the distance within which the target mesh faces are collapsed.

When more than 2 meshes overlap, it is a good idea (at least initially) to select the middle or the most important base mesh onto which other meshes may be collapsed. In this case, however, it is important to keep the lowest excavation surface (as all other will likely be removed during numerical modeling procedures). Therefore, “MineStage2” is chosen as the initial base mesh.

9. Turn all the layers off except for “MineStage1” and “MineStage2”. Use **GCollapse** tool ( ) to collapse “MineStage1” mesh onto “MineStage2”. Specify: *Distance* = 3, *IntersectAndTrim* = *Yes*, and keep all other options at defaults. Then select “MineStage1” as the target mesh and “MineStage2” as the base mesh.

   The most important parameter here is the collapse distance, which directs **GCollapse** logic to search faces of the target mesh (“MineStage1”) which fall within the specified distance from the base meshes (in this case a single base – “MineStage2”), removes them and projects newly formed boundaries onto the base mesh. *IntersectAndTrim* parameter specifies to automatically intersect and trim the projected faces, while automatically finding the most optimal intersection tolerance.

   After **GCollapse** completes, select both meshes and use *Rhino*’s command **_MeshIntersect** to create polylines tracing the complete intersections between the meshes. The command should report that it created 3 to 4 intersections. They are highlighted in yellow in Figure 94.



Layer		
Default	✓	■
Topography		🟪
Stratum1		🟧
Stratum2		🟦
MineStage1		🟥
MineStage2		🟩
Cavern		🟪

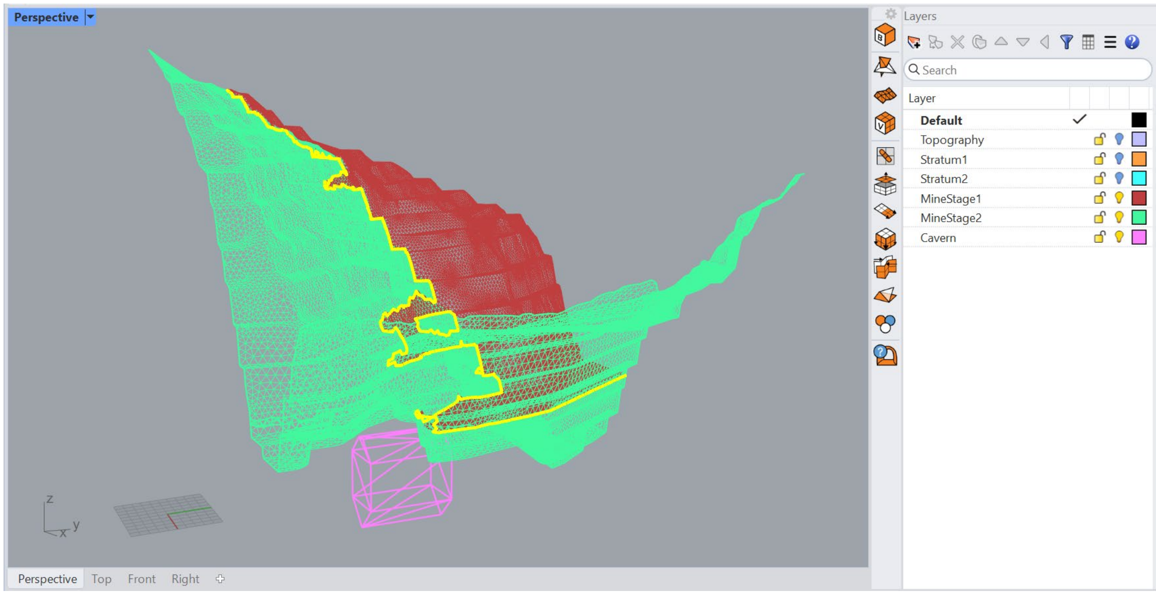
Figure 94: “MineStage2” mesh collapsed on “MineStage1” forming 3 distinct intersection curves (Wireframe view).

   Creating intersection curves (polylines) is a simple way to check if there is a complete intersection between the collapsed and base meshes. There should be one continuous polyline for every (visibly) full intersection between the meshes.

   Another important aspect to consider when using **GCollapse** tool is the average size of mesh faces vs. the collapse distance. For accuracy of the operation, it is crucial that the average target mesh face size in the overlap regions is comparable to or smaller than the collapse distance. This is why some of the meshes were remeshed with the finer mesh sizes.

10. The intersection of “MineStage1” and “MineStage2” meshes creates two volumes between them – the large one in front as on Figure 94 and the small one in the back (circled in blue). It is assumed that the main goal of this model is to assess the stability of the large portion of the pit wall and the rock bridge between the cavern and the pit, therefore the small volume in the back can be ignored as it is far enough from the cavern and does not form large-size walls.

    Select “MineStage2” mesh and use command **_SplitDisjointMesh** to separate it into two pieces, then delete the small piece. After this operation, there should be only two complete intersections between the mining stage meshes (as shown in Figure 95).



Layer		
Default	✓	■
Topography		
Stratum1		
Stratum2		
MineStage1		
MineStage2		
Cavern		

Figure 95: Collapsed and cleaned-up mining stage meshes with two complete intersections.

Figure 96 shows a zoomed-in view of one of the collapsed regions and the extended faces of “MineStage1” mesh (some are pointed at by the red arrows). These extended faces were built in the normal (orthogonal) direction to “MineStage2” mesh. **GCollapse** logic does this intentionally as after removing nearly overlapping faces, such extension technique provides a simple and reliable way of connecting meshes (and eventually creating separate closed volumes). Moreover, the shape quality of volume elements created at such orthogonal extensions will always be good as the technique avoids creation of thin wedge-like geometries.

However, it is important to stress that face extensions created in such a way may not suit all possible modeling requirements. In certain cases, especially for small-size geometries, artificially introducing a ridge by creating orthogonal face extensions may be undesirable. On the other hand, when dealing with surface meshes approximating large geologic (or other) structures, such small variations of mesh geometries do not affect the numerical modeling procedures or results, and thus the **GCollapse** approach of resolving overlapping surface meshes can be reliably used.

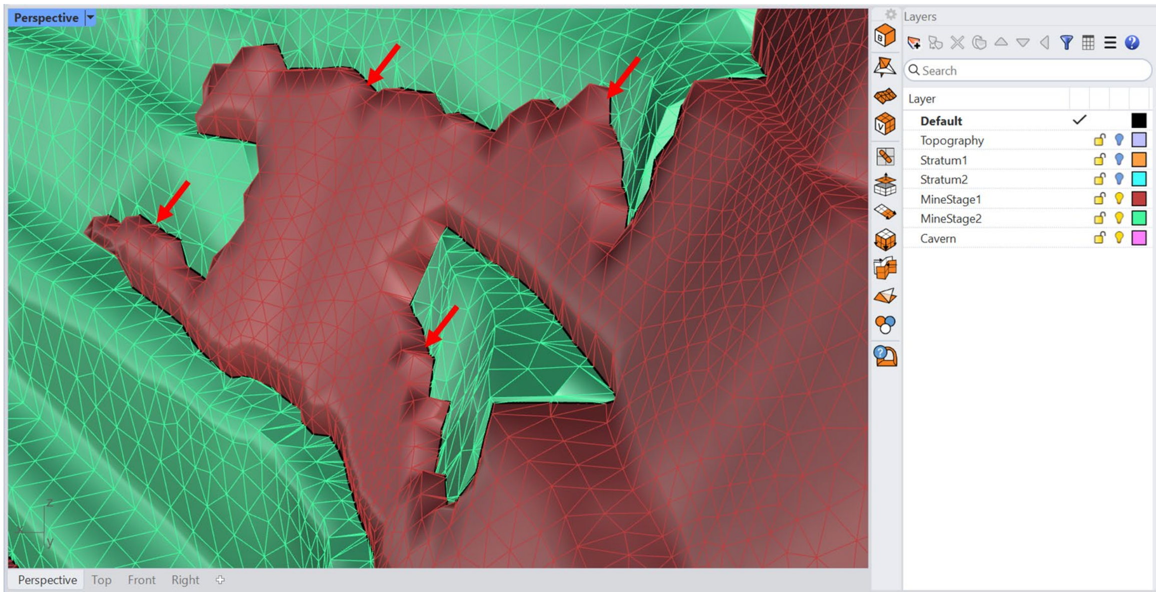


Figure 96: Zoomed-in view at the collapsed and extended faces.

11. Delete any intersection polylines that may have been created in the previous steps. Select "MineStage1" and "MineStage2" meshes and remesh them with **GSurf** using *Mode*=*Tri*, *MinEdgeLength* = 3, *MaxEdgeLength* = 10, *RidgeAngle* = 20, and keeping the other parameters at the default. This is needed as meshes are automatically intersected by **GCollapse** to provide a conformal connection, and remeshing is typically required after that to improve mesh quality.
12. As "Stratum1" mesh overlaps with the mining meshes, the former needs to be collapsed to the latter ones. Select "Stratum1" mesh and click on **GCollapse** icon ( ) in *Griddle*'s toolbar. Use the following parameters: *Distance* = 3, *IntersectAndTrim* = *Yes*, and keep all other options at the defaults. Then select both "MineStage1" and "MineStage2" as the base meshes. The result of the operation is shown in Figure 97.

    When multiple bases are selected, **GCollapse** collapses target mesh faces on the nearest surface (mesh piece) formed by the union of the base meshes. Figure 97 shows that the collapsed "Stratum1" now consists of 3 pieces located on different sides of "MineStage1" and "MineStage2". Only the largest piece is of interest, as the other two "Stratum1" pieces form small and probably artificial volumes. Use **_SplitDisjointMesh** command to split "Stratum1" mesh and delete the smaller pieces (highlighted in Figure 97).

    Select "Stratum1", "MineStage1", and "MineStage2" and remesh them with **GSurf** using same parameters as in Step 11.

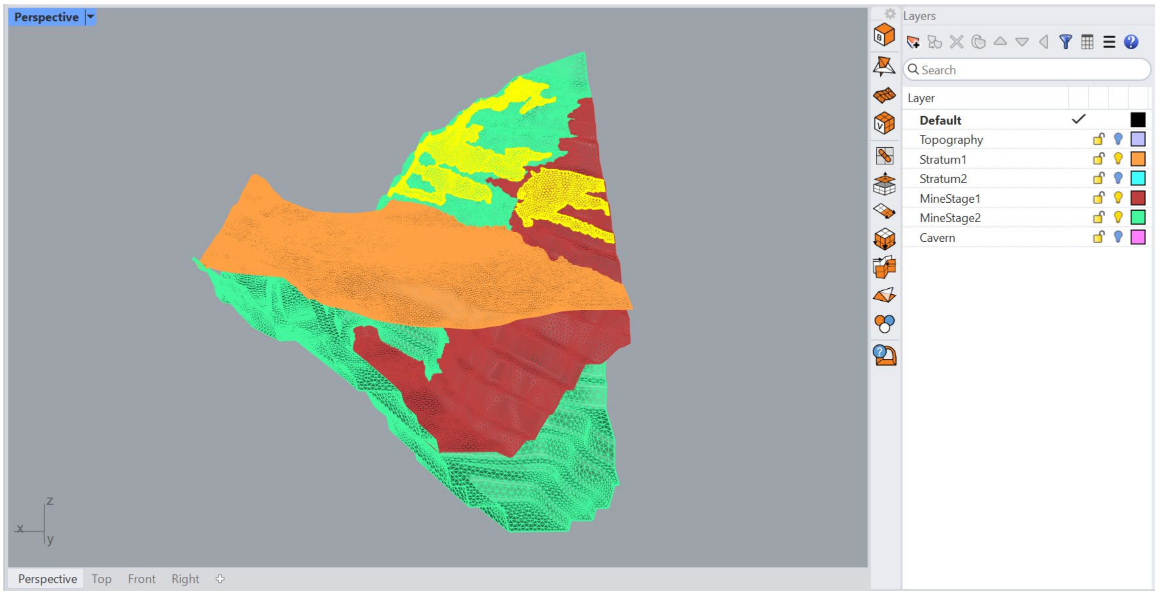


Figure 97: Result of “Stratum1” mesh collapse onto “MineStage1” and “MineStage2”.

13. Turn on layers “Topography” and collapse the topography mesh on “MineStage1” and “MineStage2” mesh following the same workflow as in the previous step but use *Distance* = 2. After the **GCollapse** completes, split the topographic mesh into 2 pieces (use **_SplitDisjointMesh**) and delete the small piece. There is no need to remesh after this step.

14. Turn on “Topography”, “Stratum1”, “MineStage1”, and “MineStage2” layers. It is seen that the meshes in these layers split the space into 5 distinct well-defined regions as shown in Figure 98. This completes mesh cleaning and separation part of the Tutorial. Incrementally save the model.

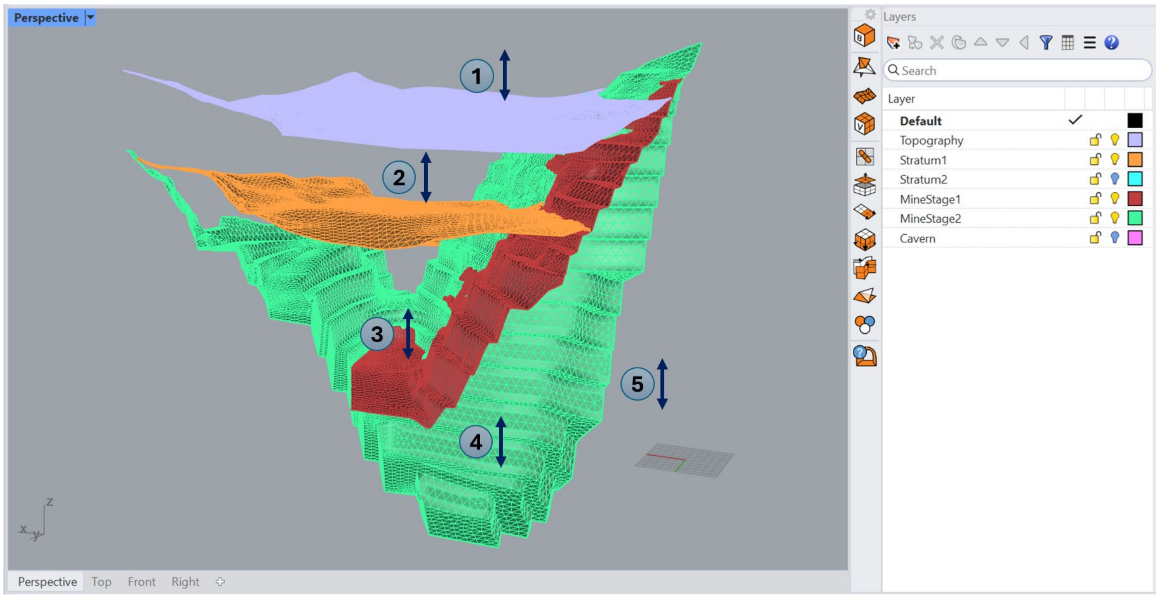
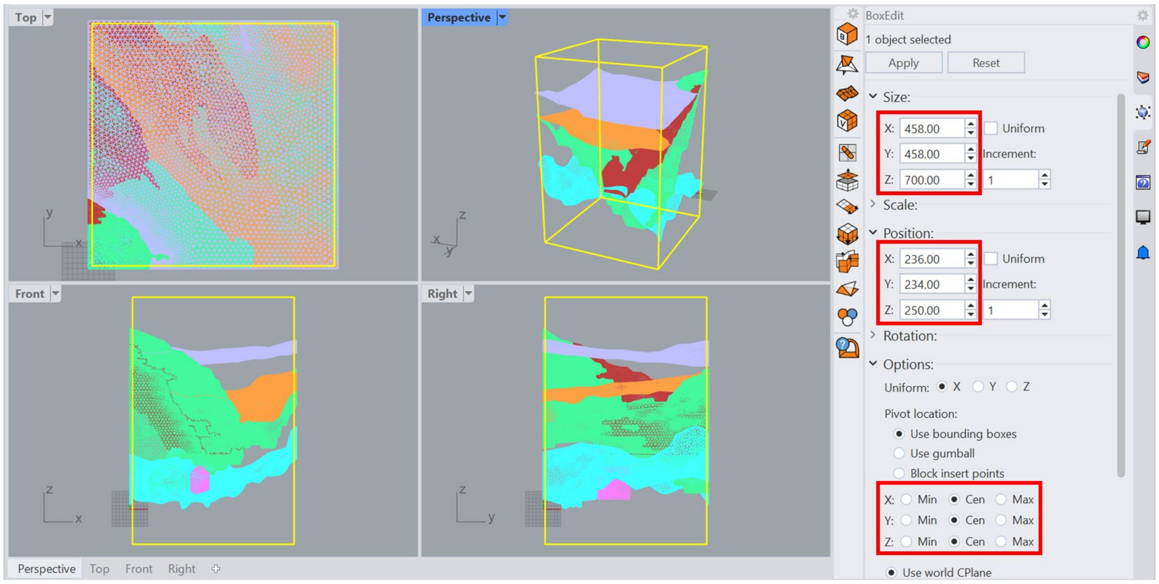


Figure 98: The model is split into 5 distinct volumetric regions as a result of GCollapse operations.

Domain creation and volume meshing

15. Turn on all the layers, select all 6 meshes and intersect them using **GInt** with tolerance = 0.01. Higher tolerance is used to merge and smooth the worst intersections between the meshes.
16. Remesh the meshes with **GSurf** using: *Mode=Tri*, *MinEdgeLength* = 4, *MaxEdgeLength* = 8, *RidgeAngle* = 30, keep other parameters at the defaults. Verify with **GHeal** that there are only 7 closed polylines composed of all naked edges corresponding to the exterior boundaries of each mesh. If there are any additional naked edges present, check where they are located. If needed, change the tolerance and or meshing parameters in the previous two steps or manually edit the issues (by removing and creating individual faces).
17. Create a new layer "Domain". Note that the model is already aligned with XYZ coordinate system in *Rhino*, and therefore, it is relatively easy to create a simple box-like domain. Select all meshes except for the "Cavern" and type **_BoundingBox** command. The command creates a tight bounding box around the selected objects. The box will be aligned with XYZ coordinate system. Specify: *CoordinateSytem* = *World*, *Cumulative* = *Yes*, *Output* = *Meshes*.

    Select the created box, change its layer to "Domain", set the name as "domain" in the Object's *Properties* panel (press **F3**, if not visible), and type **_BoxEdit** command to edit the location and the size of the box. Specify parameters shown in Figure 99 and make sure that the X, Y, and Z reference (pivot) location is set to the Center ("Cen") before any edits are done in other boxes.



BoxEdit

1 object selected

Apply Reset

Size:

X: 458.00 □ Uniform
Y: 458.00 Increment:
Z: 700.00 1

Scale:

Position:

X: 236.00 □ Uniform
Y: 234.00 Increment:
Z: 250.00 1

Rotation:

Options:

Uniform: ● X ○ Y ○ Z

Pivot location:
● Use bounding boxes
○ Use gumball
○ Block insert points

X: ○ Min ● Cen ○ Max
Y: ○ Min ● Cen ○ Max
Z: ○ Min ● Cen ○ Max

● Use world CPlane

Figure 99: Model domain creation from a bounding box.

18. Select the "Domain" mesh and remesh it with **GSurf** using: *Mode=Tri*, *MinEdgeLength* = 10, *MaxEdgeLength* = 10, *RidgeAngle* = 20, keep other parameters at the defaults.
19. Now select all 7 meshes and intersect them using **GInt** with tolerance = 0.
20. Select meshes "Topogrpahy", "Stratum1" and in the *Hyperlink* field of the Object *Properties* panel (press **F3**, if not visible), specify "elemsize:8" to enforce mesh faces to be at 8-10m in size.
21. Final remeshing is done in 2 steps which allows improving final mesh quality. First, select all 7 meshes and remesh them with **GSurf** using the following parameters: *Mode* = *Tri*,

MinEdgeLength = 8, *MaxEdgeLength* = 12, *RidgeAngle* = 30; *AdvancedParameters*: *MaxGradation* = 0.03, *Optimization* = 10, *ShapeQuality* = 0.9. After that, run remesher once more but this time select *Mode* = *QuadDom*, *AdvancedParameters*: *QuadWeight* = 0.75 and keep the rest of the parameters from the previous step. At the second step, the meshes do not need to be significantly changed, and it makes it easier for **GSurf** to build a better-quality quad-dominant mesh.

22. Select all the meshes and use volume mesher **GVol** to generate unstructured Hex-dominant volume mesh. In the main settings, set both options in *VisualizeOutput* to *Yes* and in *MeshSettings*, set *Mode* = *HexDom*, *Optimization* = 7, *ShapeQuality* = 0.8, *ShowWartnings* = *Ngons*, and keep all other parameters at the defaults. Volume meshing may take some time.

As the meshes in this model originate from real geologic and engineering data, the size of the model is moderately large, and so **GVol** meshing may take several minutes or longer depending on the processor performance. If **GVol** takes a long time, abort it (press Esc) and create a pure tetrahedral volume mesh, which is significantly faster[8].

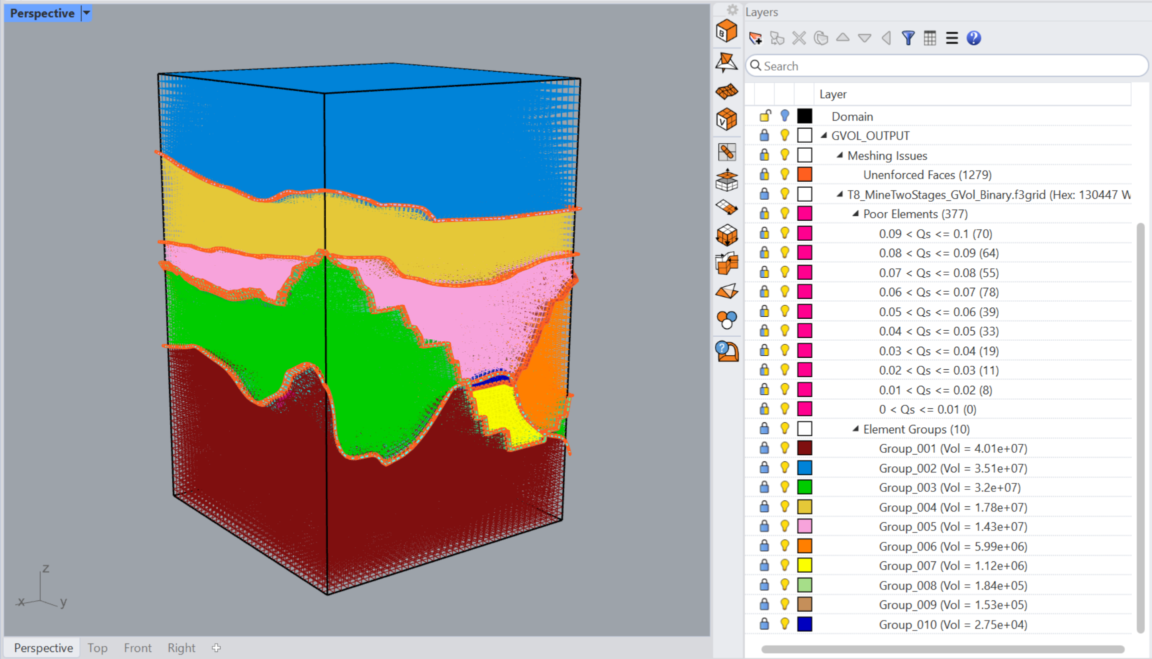
Incrementally save the model. Users may stop at this point and load the generated volume mesh into a numerical modeling software of their choice.

Note that the results presented here may differ slightly from what users obtain due to minor numerical differences and/or some inaccuracies in operations on different machines.

Generated volume mesh is visualized directly in *Rhino* alongside various information and objects provided in the Layers panel (Figure 100). In the current *Griddle* implementation, volume meshes are represented by non-interactive wireframe-style meshes. For details about volume meshes visualization in *Rhino* and some important notes, refer to *Griddle* Manual.

Figure 101 and Figure 102 show the model when it is loaded in FLAC3D. The images clearly represent the type of zones in the model and different zones groups corresponding to several mining stages.

[8] In general, it is recommended first to generate a tetrahedral volume mesh to quickly check for geometry errors – if any are found and reported in “GVOL_OUTPUT ⍰ Meshing Issues” layer, they should be fixed before attempting to create a Hex-dominant volume mesh.



```
Layers
Search
Layer
Domain
◢ GVOL_OUTPUT
  ◢ Meshing Issues
      Unenforced Faces (1279)
  ◢ T8_MineTwoStages_GVol_Binary.f3grid (Hex: 130447 W
    ◢ Poor Elements (377)
        0.09 < Qs <= 0.1 (70)
        0.08 < Qs <= 0.09 (64)
        0.07 < Qs <= 0.08 (55)
        0.06 < Qs <= 0.07 (78)
        0.05 < Qs <= 0.06 (39)
        0.04 < Qs <= 0.05 (33)
        0.03 < Qs <= 0.04 (19)
        0.02 < Qs <= 0.03 (11)
        0.01 < Qs <= 0.02 (8)
        0 < Qs <= 0.01 (0)
    ◢ Element Groups (10)
        Group_001 (Vol = 4.01e+07)
        Group_002 (Vol = 3.51e+07)
        Group_003 (Vol = 3.2e+07)
        Group_004 (Vol = 1.78e+07)
        Group_005 (Vol = 1.43e+07)
        Group_006 (Vol = 5.99e+06)
        Group_007 (Vol = 1.12e+06)
        Group_008 (Vol = 1.84e+05)
        Group_009 (Vol = 1.53e+05)
        Group_010 (Vol = 2.75e+04)
```

Figure 100: GVol generated mesh visualized in a *Rhino* viewport and informational output in the Layers panel. The model domain is shown with black edges.

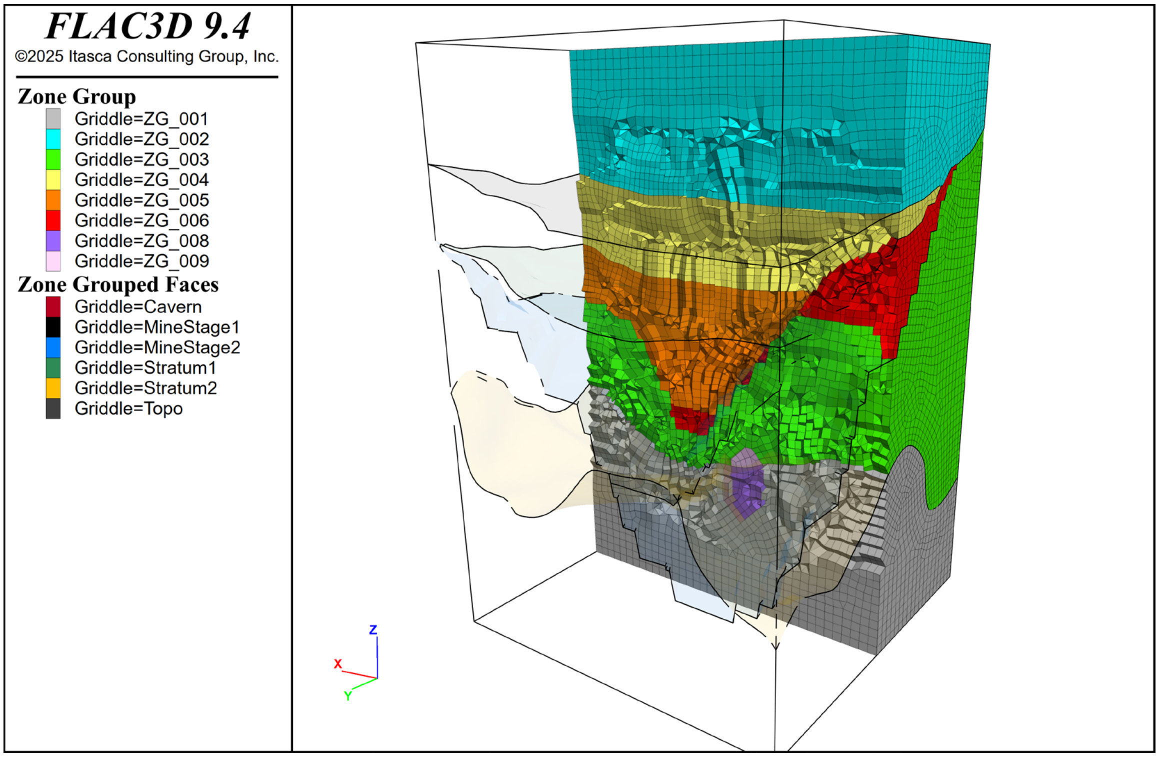


Figure 101: Cut through the model in *FLAC3D*.

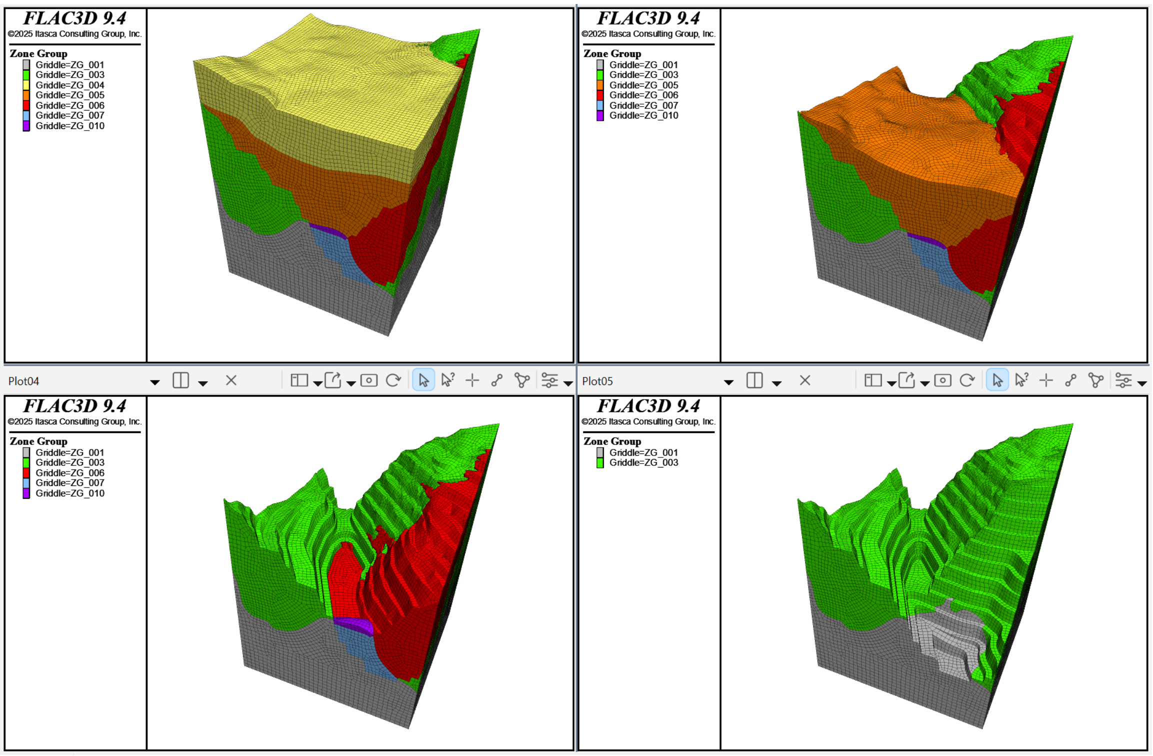


Figure 102: Different model excavation stages visible in *FLAC3D*.

Volume mesh review and further improvements

The information provided below is for reviewing only. There is no need to follow any steps presented in this section unless users would like to further explore and practice with mesh optimization techniques.

The Layers panel in Figure 100 contains various information about the volume mesh. First, take a note of the layer "Meshing Issues". It contains information about any issues encountered by **GVol** (not all issues prevent **GVol** from running). For this model, **GVol** reported having many "Unenforced Faces"[9] which mostly come from the mesh faces located outside of the meshing domain (not used in meshing). There are also a few of unenforced faces within the model corresponding to non-planar quads. This issue type can usually be ignored[10].

[9] "Unenforced Faces" are surface mesh faces which **GVol** was not able to use in the meshing process or not able to strictly enforce in the volume mesh (for example, if a non-planar quad face had to be connected to two volume elements with triangular faces at the quad junction).

[10] Surface meshes "sticking" outside of the domain box were not trimmed for this model because:

- Terminating model meshes on the domain mesh (in conformal manner) would lead to having mesh (hard) boundaries on the domain. When remeshing the domain and other meshes, such hard boundaries limit **GSurf** flexibility to rebuild the meshes while meeting all desired remeshing requirements and honoring the shape of the hard boundaries. This may occasionally lead to poor remeshing results. As the exterior mesh parts do not affect volume meshing, they can simply be ignored.
- When splitting or trimming quadrilateral faces with **_MeshSplit** or **_MeshTrim** commands, the commands create Ngons by the default (*CreateNgons* = *Yes*). If the option is set to *No*, the commands triangulate processed faces (as *Rhino* internally does mesh intersections). Either situation may not be desirable and will require an additional remeshing afterwards. Thus, it is recommended to do mesh splitting or trimming for pure triangular meshes only.

The layer “Element Groups” contains 10 sublayers corresponding to different internal subdomains (or closed volumes). The layers provide information about the group name/number and the subdomains volume. These groups correspond to various element groups in the output mesh file (for example, “Group_001” would correspond to zone group “ZG_001” in *FLAC3D* output). Groups are numbered based on decreasing subdomain volumes. Users can quickly query these groups in *Rhino* by turning the layers on and off, changing layers colors, etc. This helps identifying where the different groups are located and if there are any unexpected internal sub-domains/groups. Note that volume meshes are not interactive; they are created to help users to identify different meshed subdomains. These meshes are not saved with *Rhino* model. See more information about this in *Griddle* Manual.

The layer “Poor Elements” provides information about the worst volume elements in the model (with **GVol**’s quality metric Qs <= 0.1). These layers correspond to the last 2 quality bins in a log file generated by **GVol** and contain objects outlining poor-quality elements. The layers containing the worst quality elements (Qs <= 0.02) are the most important to pay attention to. Users can easily select the objects in the layers and zoom-in at them to investigate where the clusters of poor elements are located and what is a potential cause for their generation (very often it is due to problems with surface meshes).

Let’s investigate poor elements in the layer “0.01 < Qs <= 0.02”. Turn off the layers corresponding to the Element Groups, Meshing issues, and all the “Poor Elements” layers except for the layer of interest. Unlock all the locked layers. Right click on the layer “0.01 < Qs <= 0.02” and choose to *Select Objects* (Figure 103). A few clusters of poor elements are highlighted.

Zoom-in, for example, to one of the clusters shown in Figure 103. It contains only two bad elements with “0.01 < Qs <= 0.02” (shown in pink) but also many more poor elements from other layers/bins (shown in yellow on the image). It is seen that the cluster is located at and around the “MineStage2” mesh, and most likely is caused by poor local mesh geometry. Thus, improving mesh geometry at this location will improve volume mesh quality and will reduce the number of bad elements in several bins.

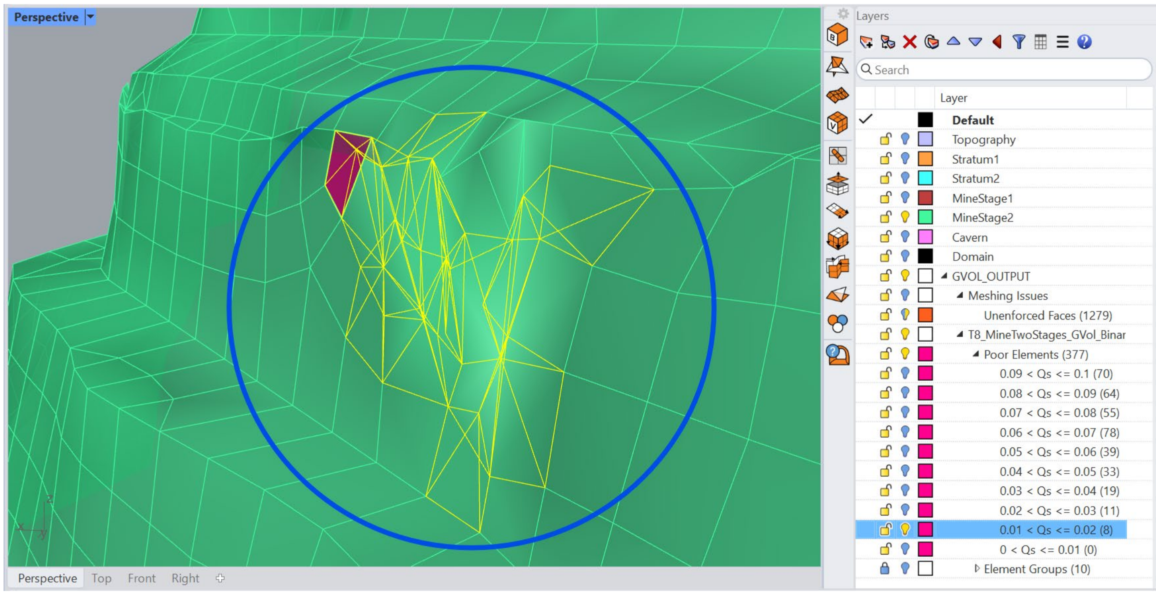
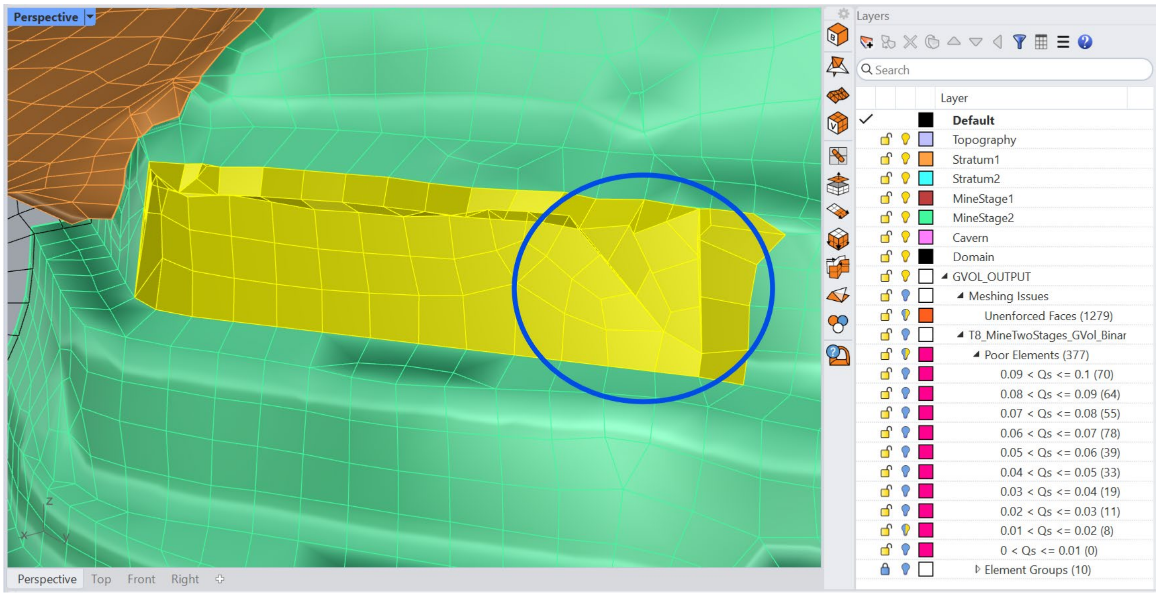


Figure 103: A big cluster of poor elements caused by the rough geometry of MineStage2 mesh.

Figure 104 shows a bad patch of "MineStage2" which contains multiple mesh folds and thin wedge-like features[11]. The blue circle in Figure 104 indicates the region where a cluster of poor volume elements is generated.



Default
Topography
Stratum1
Stratum2
MineStage1
MineStage2
Cavern
Domain
GVOL_OUTPUT
Meshing Issues
Unenforced Faces (1279)
T8_MineTwoStages_GVol_Binar
Poor Elements (377)
0.09 < Qs <= 0.1 (70)
0.08 < Qs <= 0.09 (64)
0.07 < Qs <= 0.08 (55)
0.06 < Qs <= 0.07 (78)
0.05 < Qs <= 0.06 (39)
0.04 < Qs <= 0.05 (33)
0.03 < Qs <= 0.04 (19)
0.02 < Qs <= 0.03 (11)
0.01 < Qs <= 0.02 (8)
0 < Qs <= 0.01 (0)
Element Groups (10)

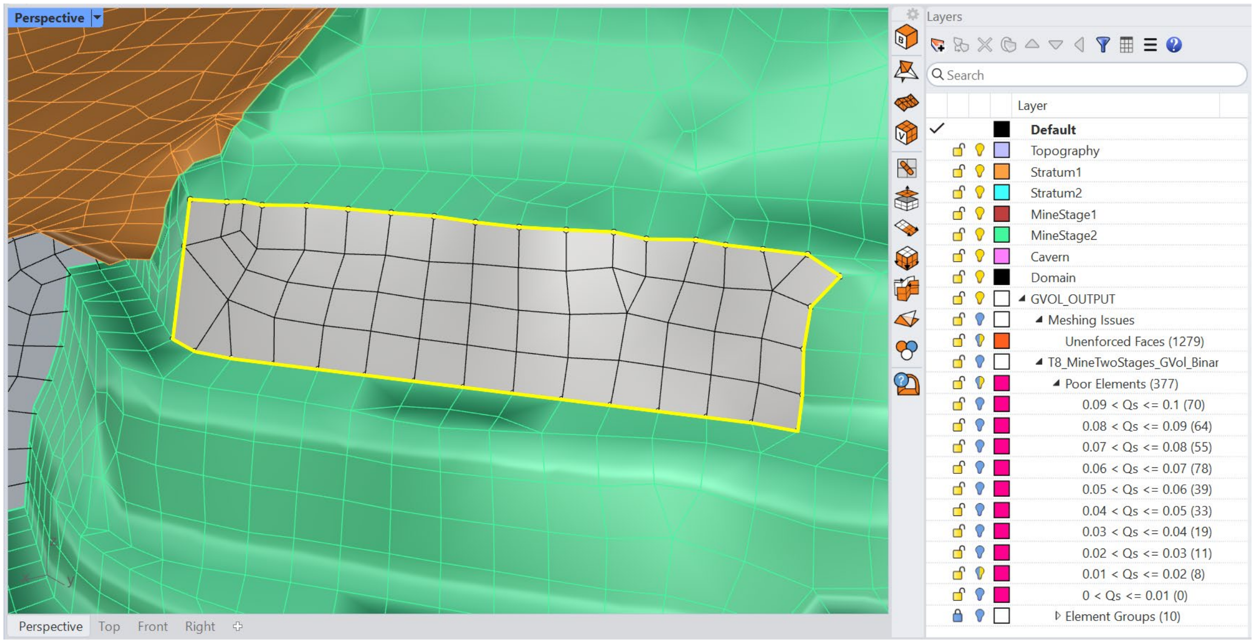
Figure 104: Poor surface mesh geometry causing bad-quality volume elements to be generated within the blue circle.

For this case, it is straightforward to fix the issue by locally rebuilding the rough mesh patch:

- Use command **_ExtractMeshFaces** to extract the highlighted faces (make sure to extract all small faces on the edges of the cluster, if any are present),
- Duplicate border (**_DupBorder**) of the extracted mesh and delete the extracted mesh,
- Make sure that the duplicated border forms a closed polyline and create an initial triangular mesh from the polyline (**_MeshPolyline**),
- Select the newly created mesh and the duplicated border and remesh with **GSurf** using the last remeshing parameters in Step 21 (the border must be selected to preserve conformity),
- Delete the border and **_Join** the updated mesh patch with the rest of the mesh.

This approach is suitable for fixing surface mesh problems locally when the rest of the mesh has acceptable quality. The results are shown in Figure 105. It is important to pay attention to any intersections of the extracted patch with other meshes: such intersections must be preserved in a new remeshed patch. This can be done by duplicating portions of intersection polylines and using them as hard edges (use **_MeshIntersect** command).

[11] The highlighted geometry represents a part of a mining bench. Rough geometries were likely produced due to rugged topographic surface, errors in measurements, and/or errors in triangulation of initial point clouds.



	Layer
✓	**Default**
	Topography
	Stratum1
	Stratum2
	MineStage1
	MineStage2
	Cavern
	Domain
	◢ GVOL_OUTPUT
	◢ Meshing Issues
	Unenforced Faces (1279)
	◢ T8_MineTwoStages_GVol_Binar
	◢ Poor Elements (377)
	0.09 < Qs <= 0.1 (70)
	0.08 < Qs <= 0.09 (64)
	0.07 < Qs <= 0.08 (55)
	0.06 < Qs <= 0.07 (78)
	0.05 < Qs <= 0.06 (39)
	0.04 < Qs <= 0.05 (33)
	0.03 < Qs <= 0.04 (19)
	0.02 < Qs <= 0.03 (11)
	0.01 < Qs <= 0.02 (8)
	0 < Qs <= 0.01 (0)
	▷ Element Groups (10)

Figure 105: A rebuilt good-quality mesh patch before merging with the remaining mesh.

The approach presented here allows for local mesh improvements based on the poor elements reported in *Rhino*. After all desired mesh fixes and improvements are done, incrementally save the model and rerun **GVol** with the same parameters as used previously.