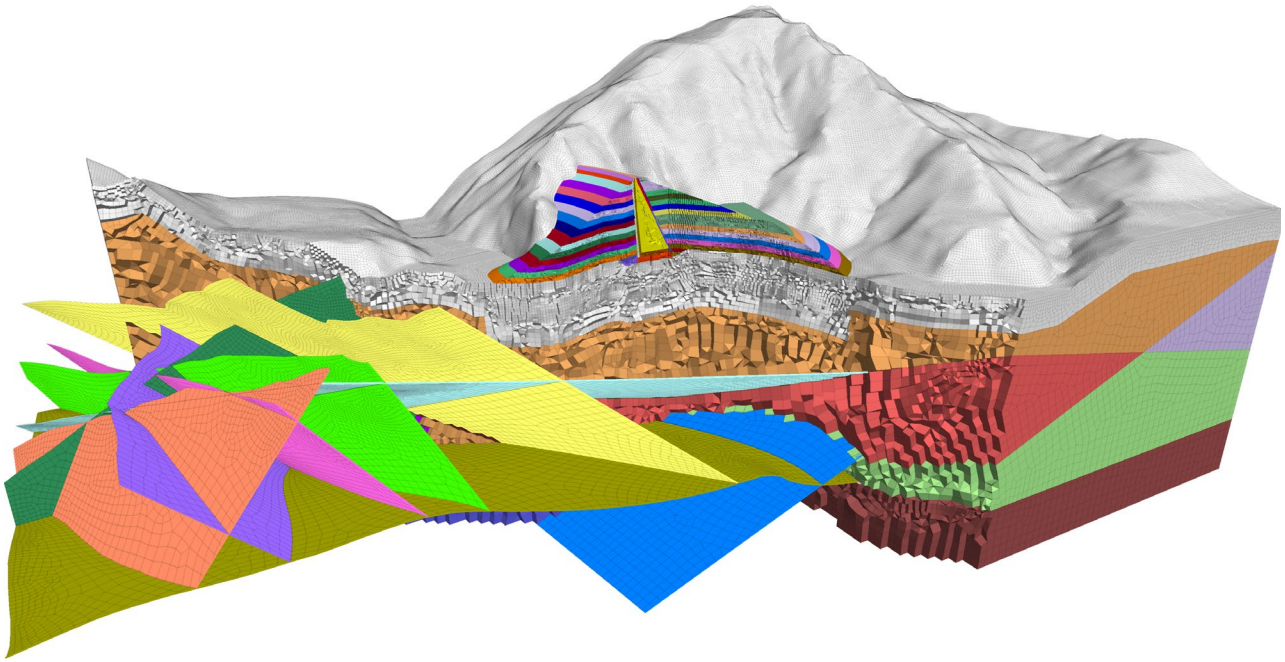
Griddle™

Advanced Grid Generation for Engineers and Scientists

Griddle 3.0 Tutorial Examples



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).

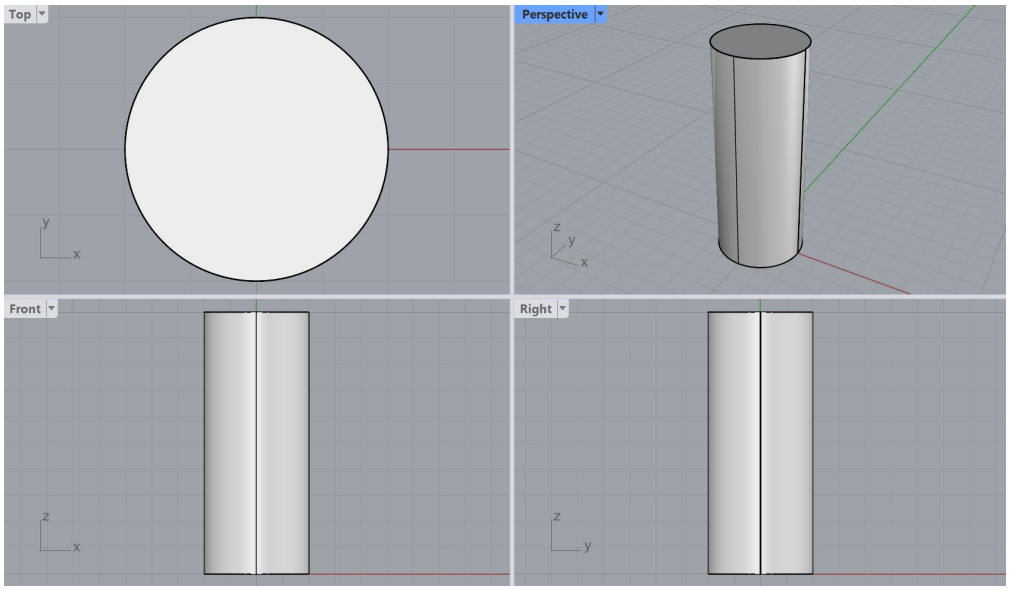


Figure 1: Solid representing a cylinder

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).

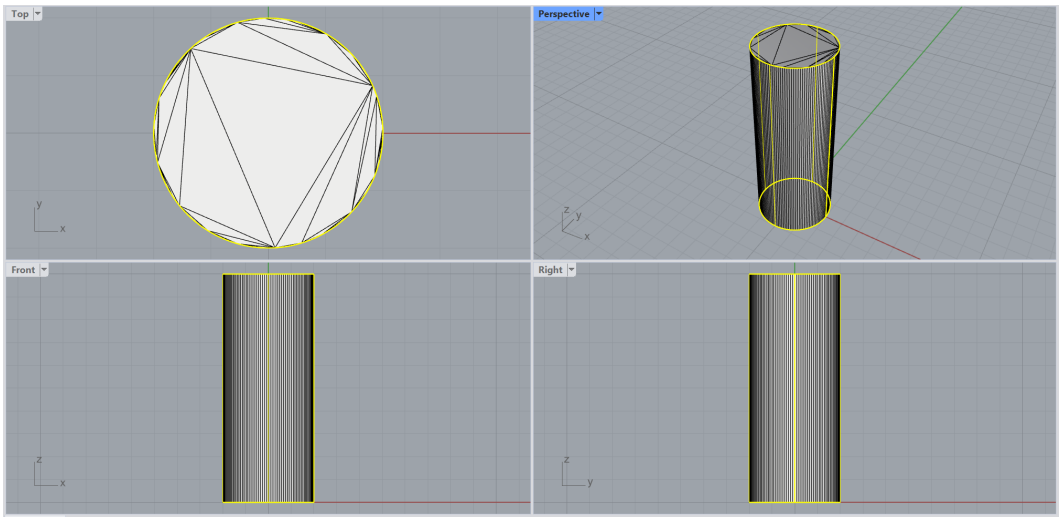


Figure 2: Original highlighted cylindrical surface and the newly created surface mesh superimposed on it.

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, *AdvancedParameters*: *ShapeQuality* = 0.65 and keep all other parameters at the default values. The resulting mesh is shown in Figure 3.

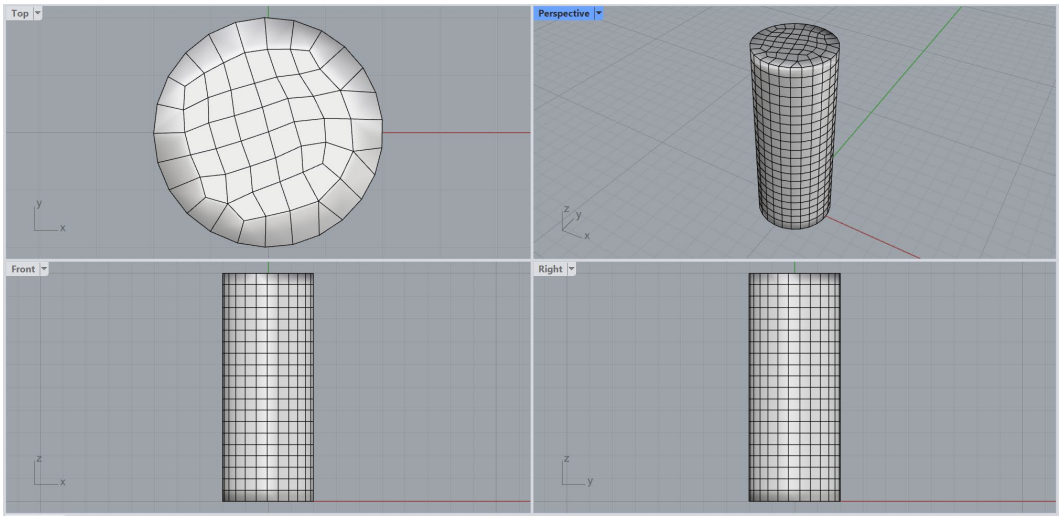


Figure 3: Quad-dominant surface mesh generated by GSurf.

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).

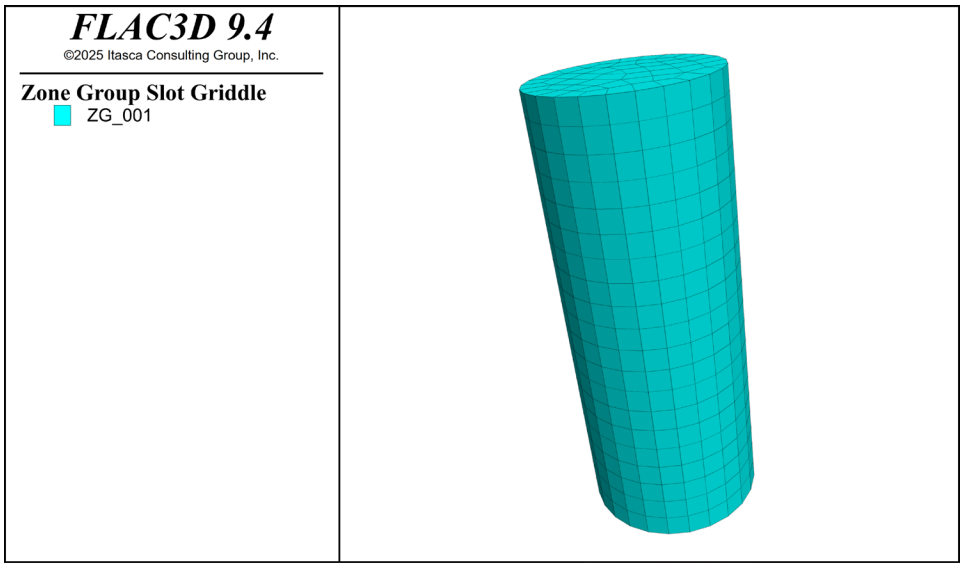


Figure 4: Volume mesh of cylinder imported into *FLAC3D*.

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).

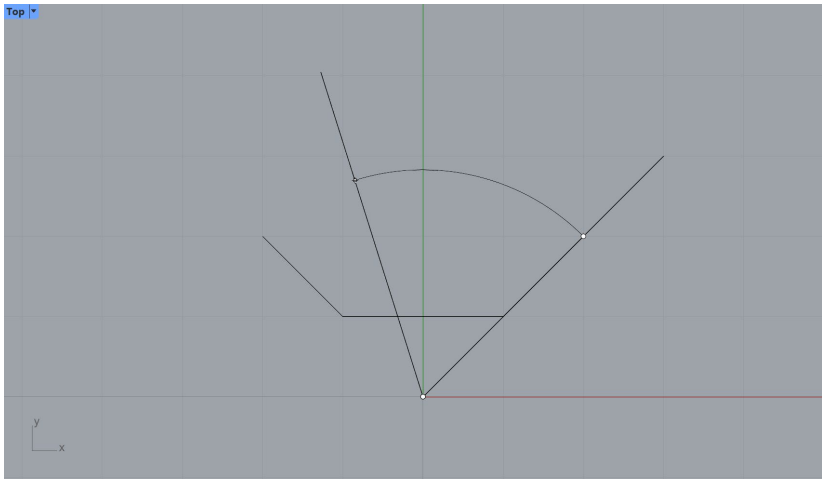


Figure 5: Creating an arc between two points.

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

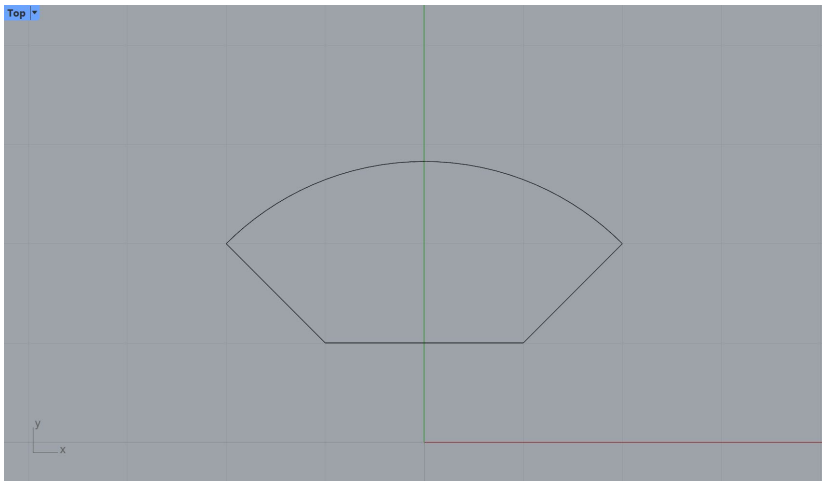


Figure 6: Single polyline consisting of three lines and an arc.

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).

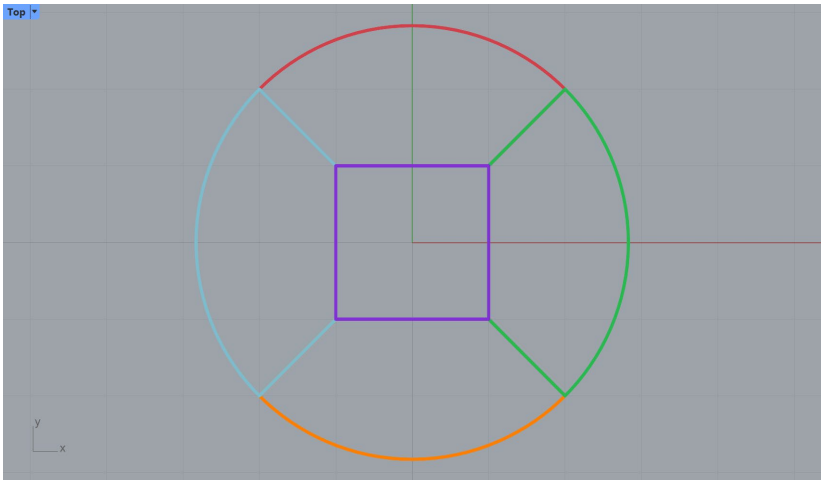


Figure 7: Five colorized polylines (some edges overlap).

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.

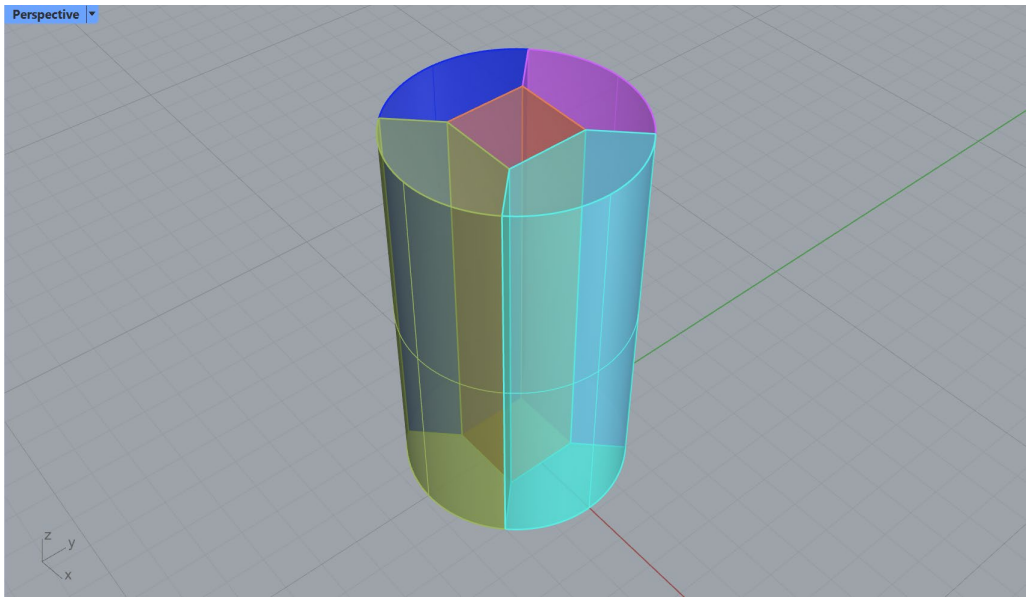


Figure 8: A cylinder subdivided into 5 solids (Ghosted view).

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).

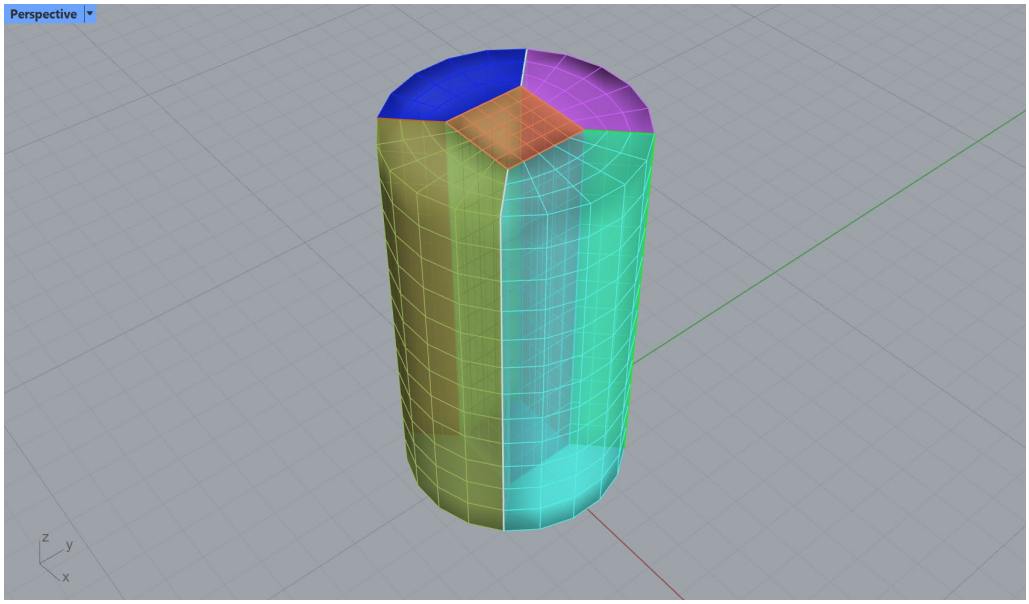


Figure 9: Surface mesh corresponding to the generated volume mesh (Ghosted view).

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.

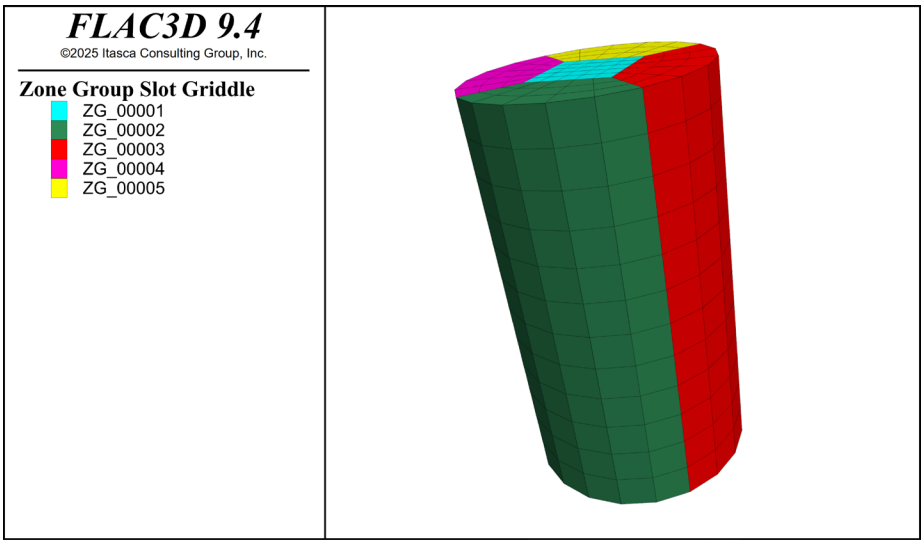


Figure 10: A cylinder subdivided into five solids.

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.

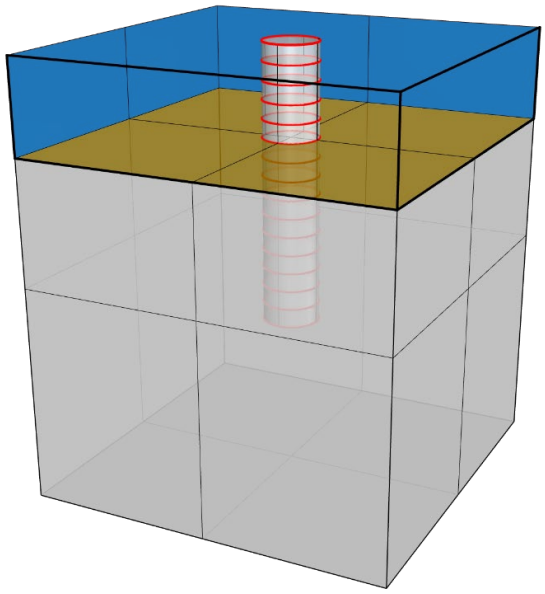


Figure 11: A general view of the model of a vertical shaft in a stratified soil.

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.

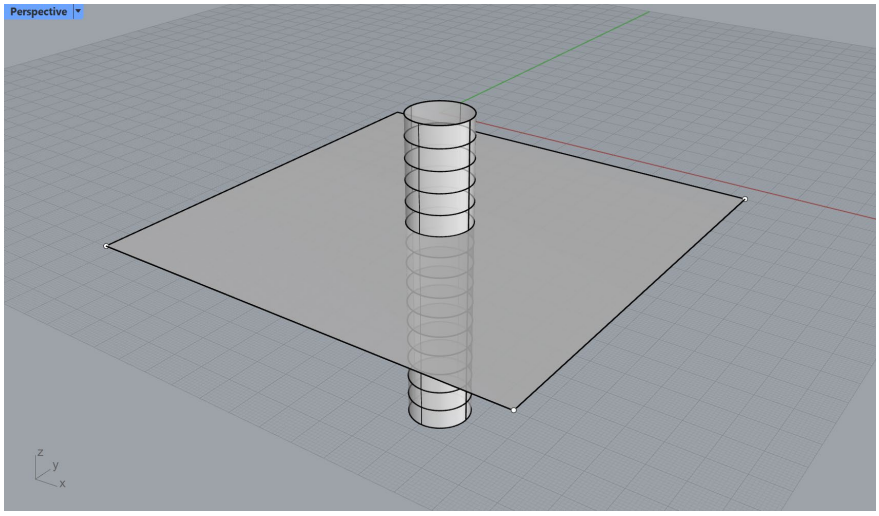


Figure 12: Construction of a shaft and a plane separating two layers of soil (Ghosted view).

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.

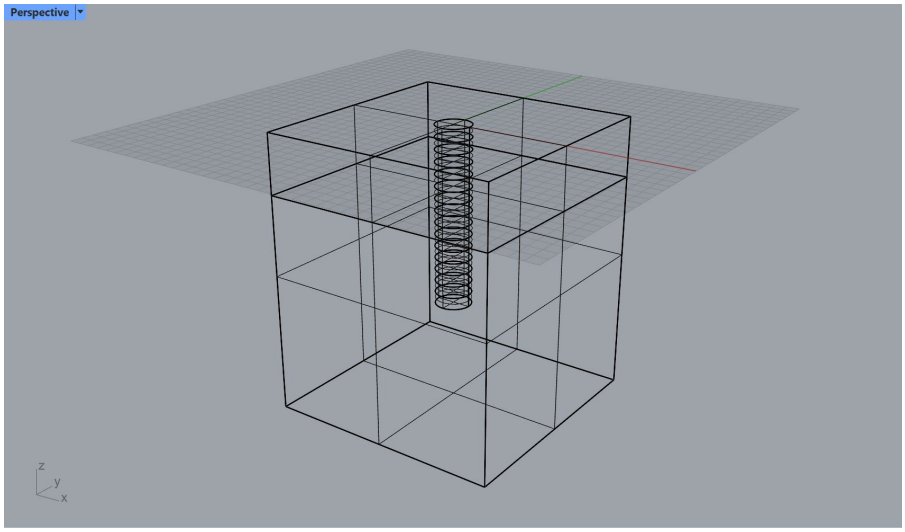


Figure 13: Complete initial geometry of the model of a shaft and two soil layers (Wireframe view).

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.

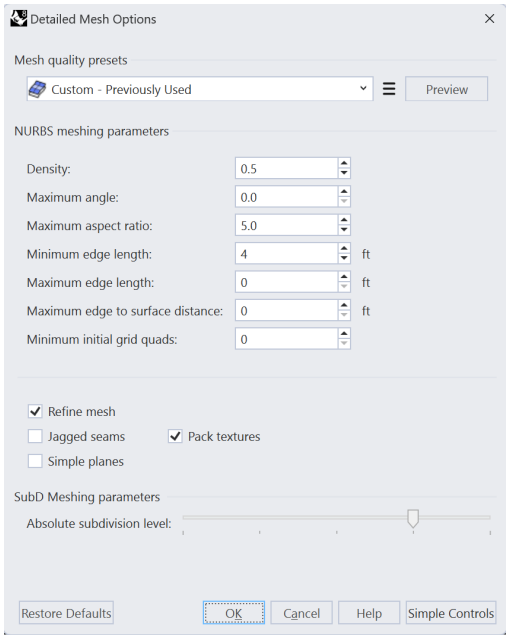


Figure 14: *Rhino* meshing dialog to create an initial mesh.

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).

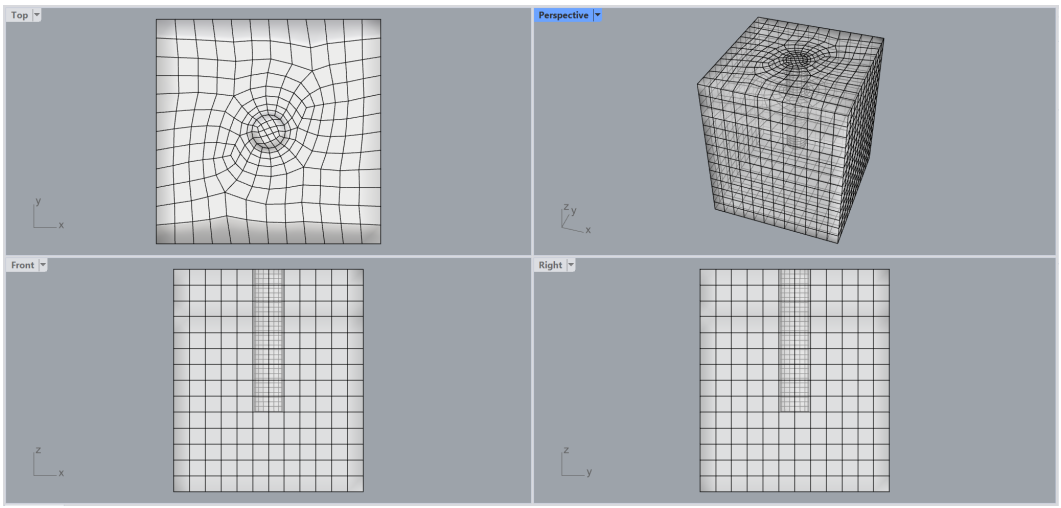


Figure 15: Remeshed model geometry using GSurf with *QuadDom* mode.

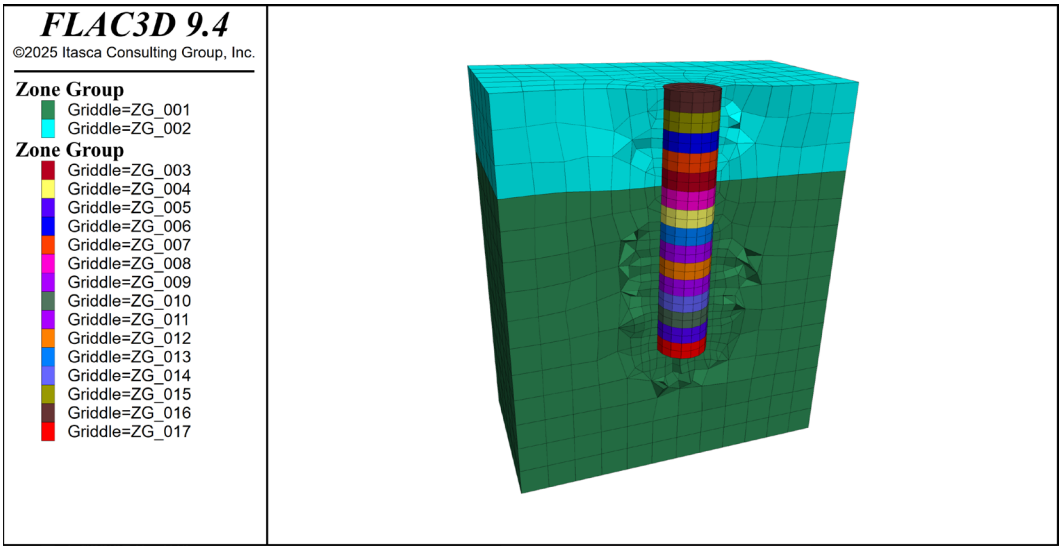


Figure 16: *FLAC3D* grid of the shaft model when using default GVol parameters.

- 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.

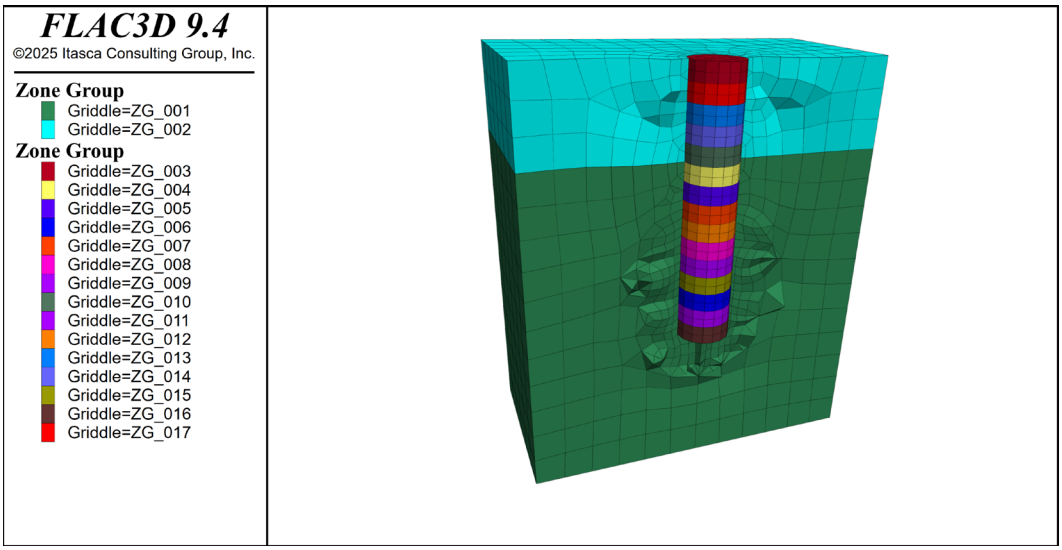


Figure 17: *FLAC3D* grid of the shaft model when using custom GVol parameters.

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.

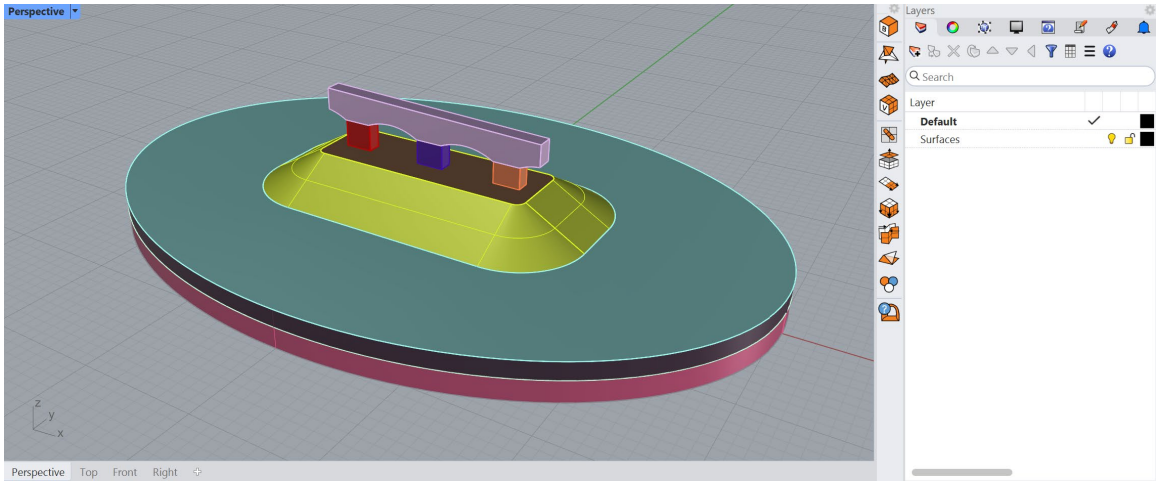


Figure 18: Model of a bridge support structure.

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).

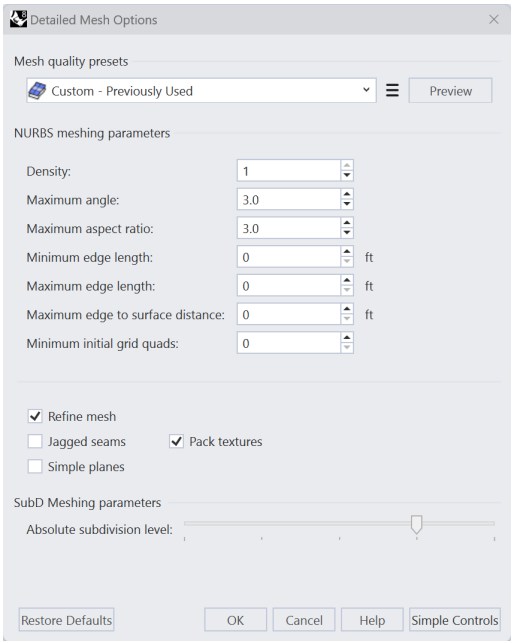


Figure 19: *Rhino* meshing dialog to create fine initial mesh.

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.

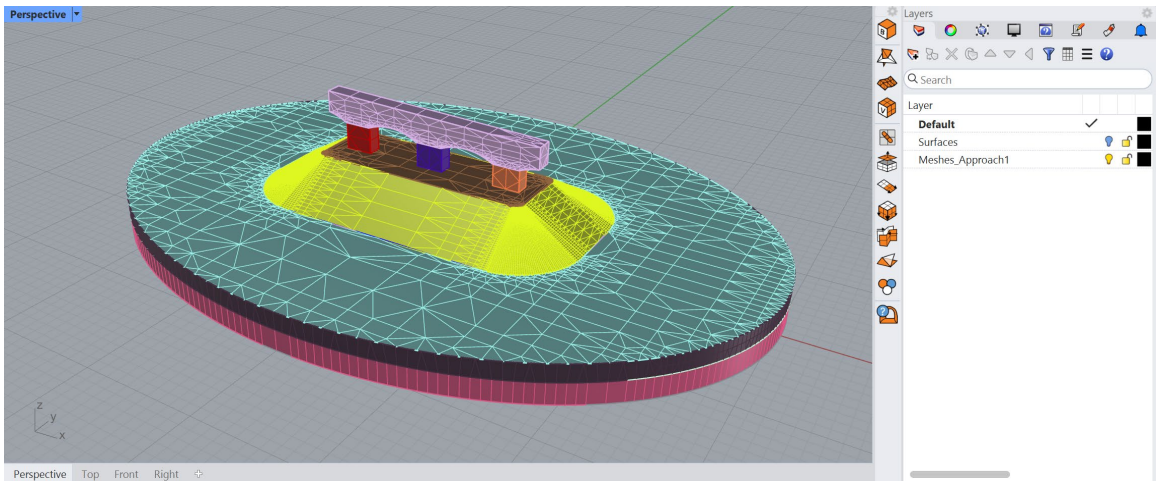


Figure 20: The initial *Rhino* mesh.

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.

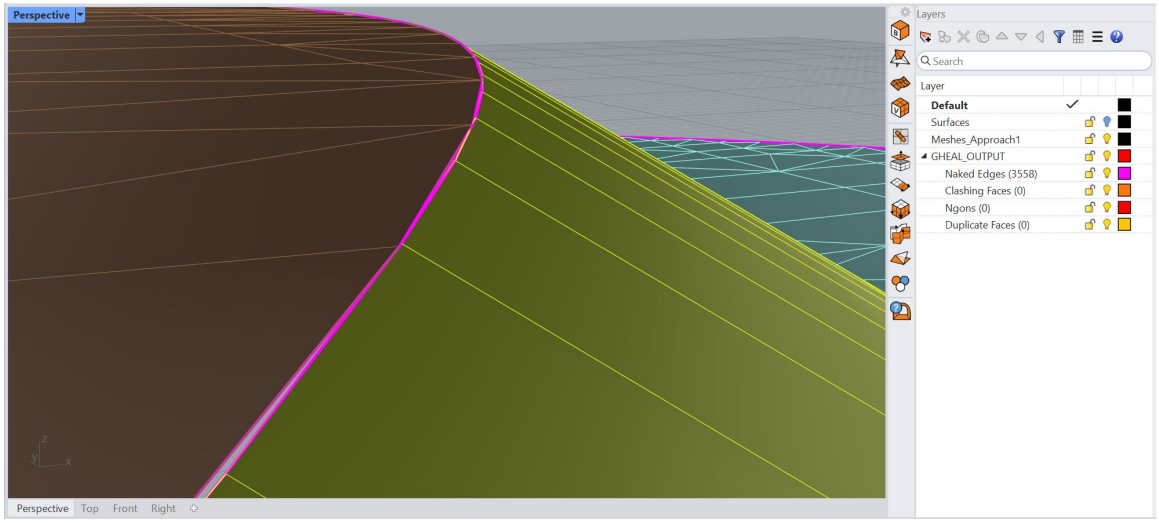


Figure 21: Mismatching mesh boundaries in the initial *Rhino* mesh.

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.

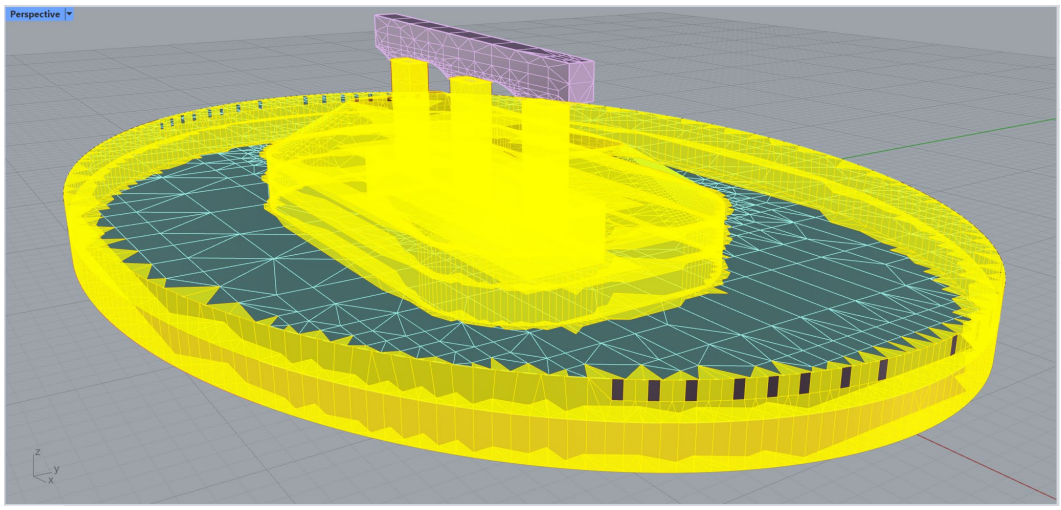


Figure 22: GInt's preview with *Tolerance* = 0. Some faces are visibly not highlighted among the surrounding highlighted faces, which is an indication of an insufficient tolerance.

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.

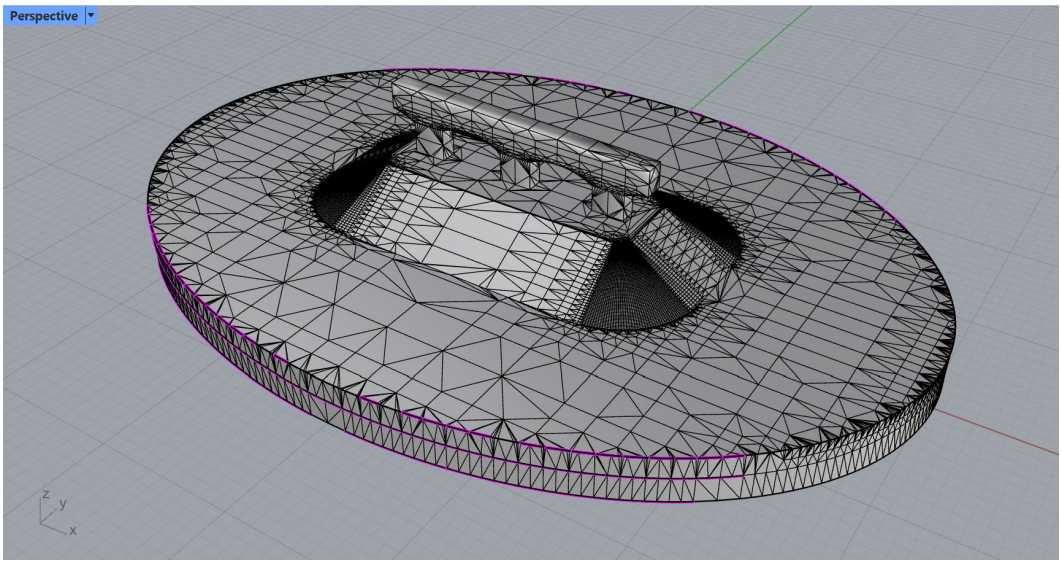


Figure 23: Naked edges (in pink) after using GInt with *Tolerance* = 0.005.

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].

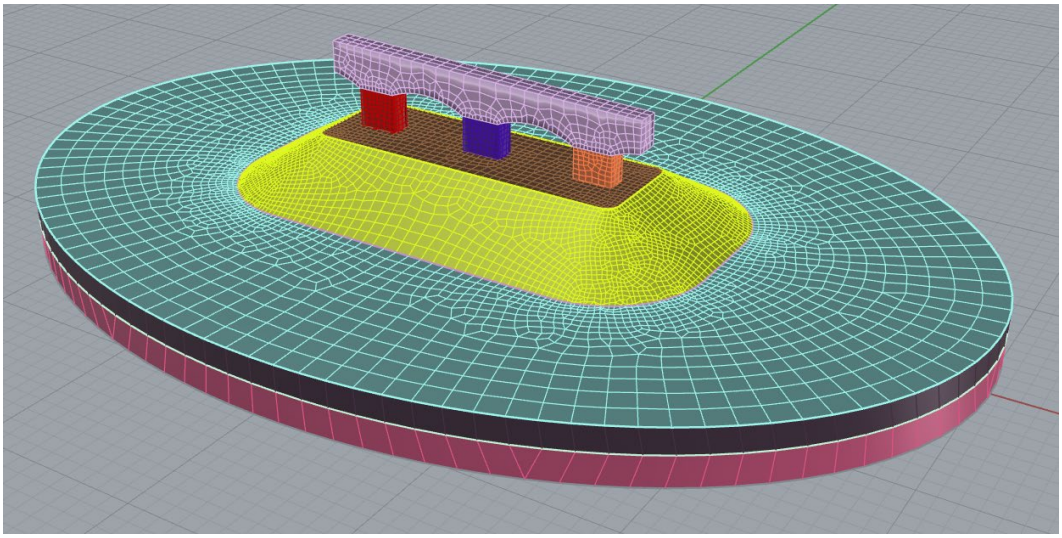


Figure 24: Remeshed surface meshes using standard approach.

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.*

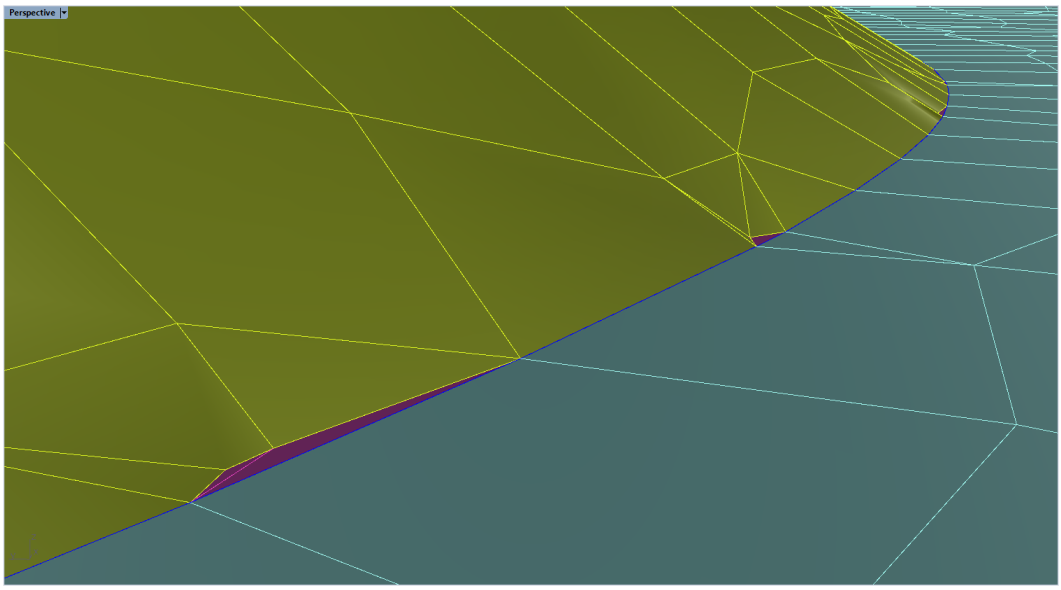


Figure 25: Slightly deformed surface meshes with approach 1.

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.

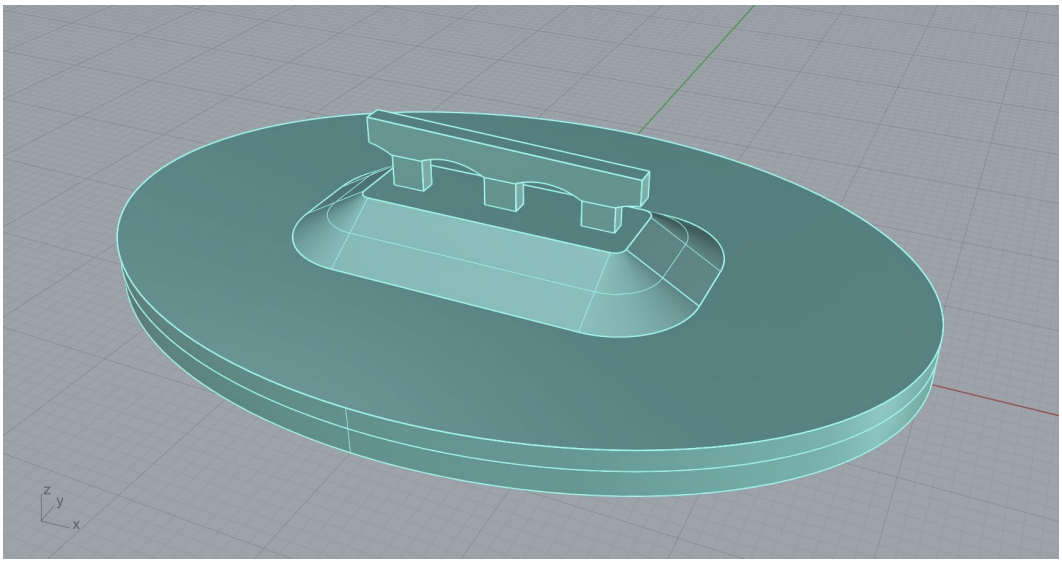


Figure 26: Single polysurface constructed by the NonManifoldMerge tool.

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”).

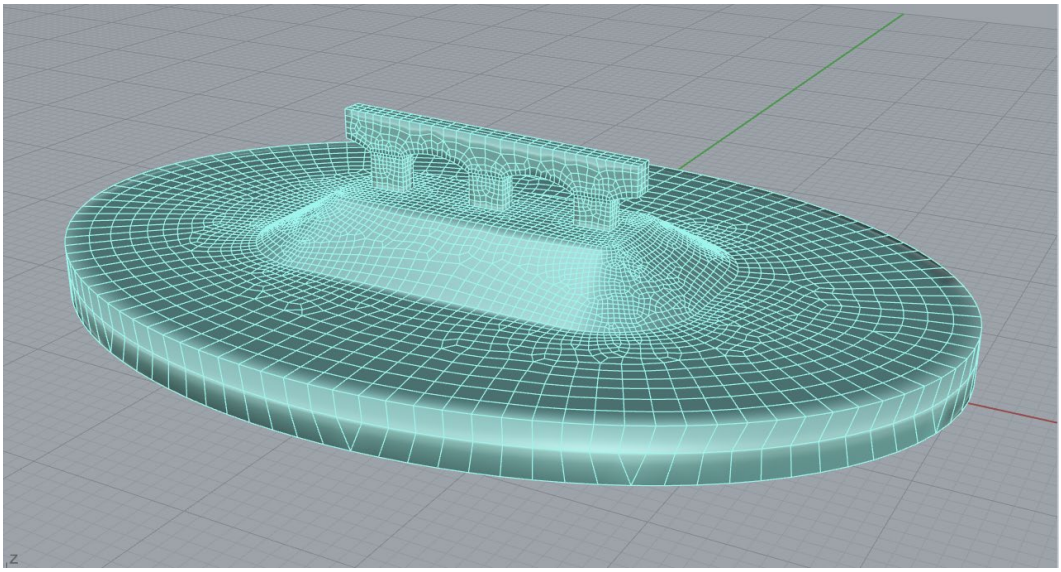


Figure 27: Remeshed surface mesh in the second approach.

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.

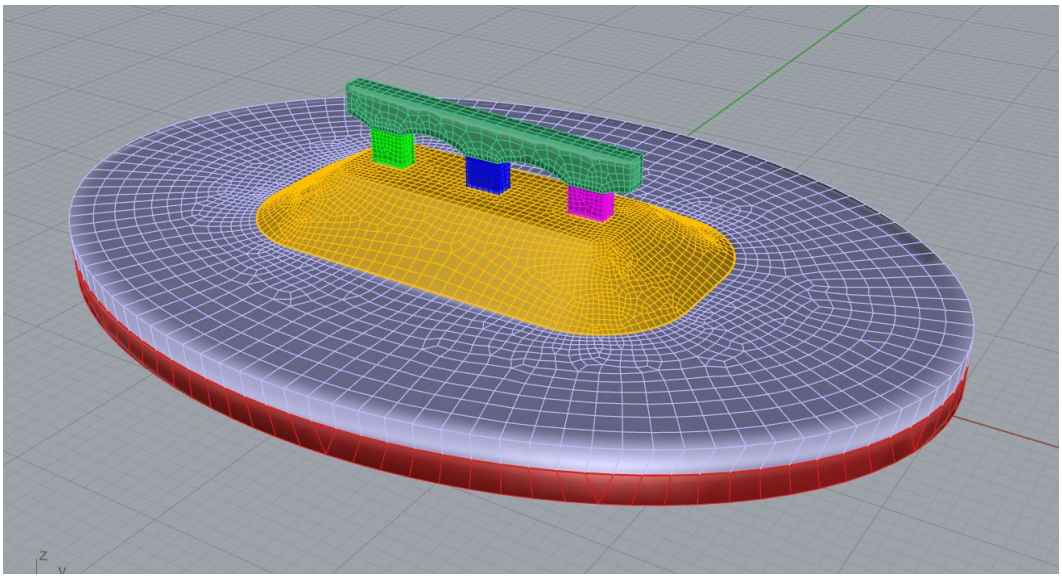


Figure 28: Mesh from Figure 27 "exploded" into sub-meshes, which are subsequently colorized.

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.

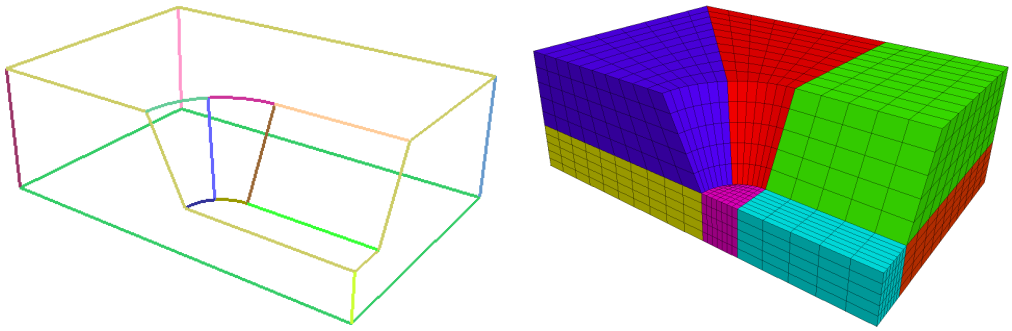


Figure 29: Reference model (left) and BlockRanger generated mesh (right).

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.

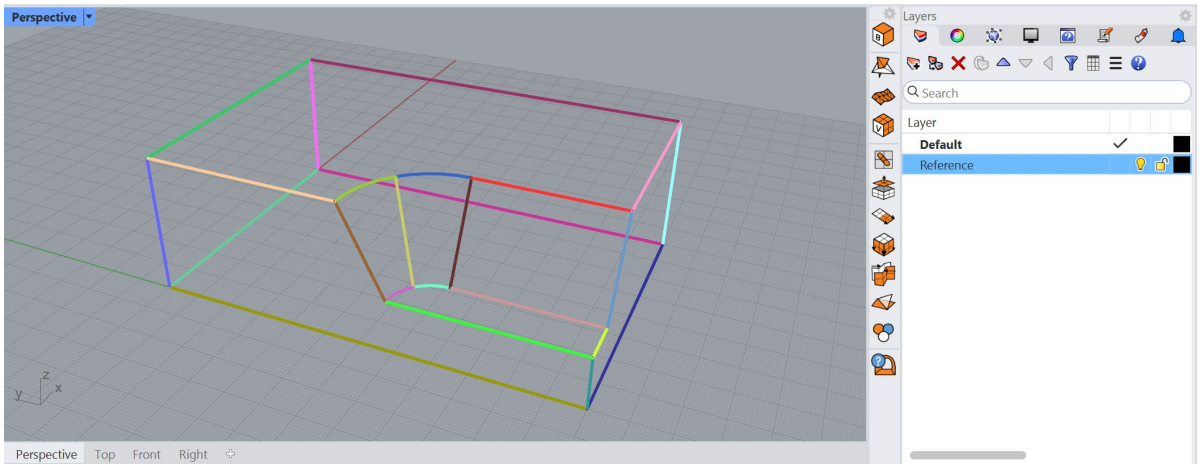


Figure 30: The reference model.

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.

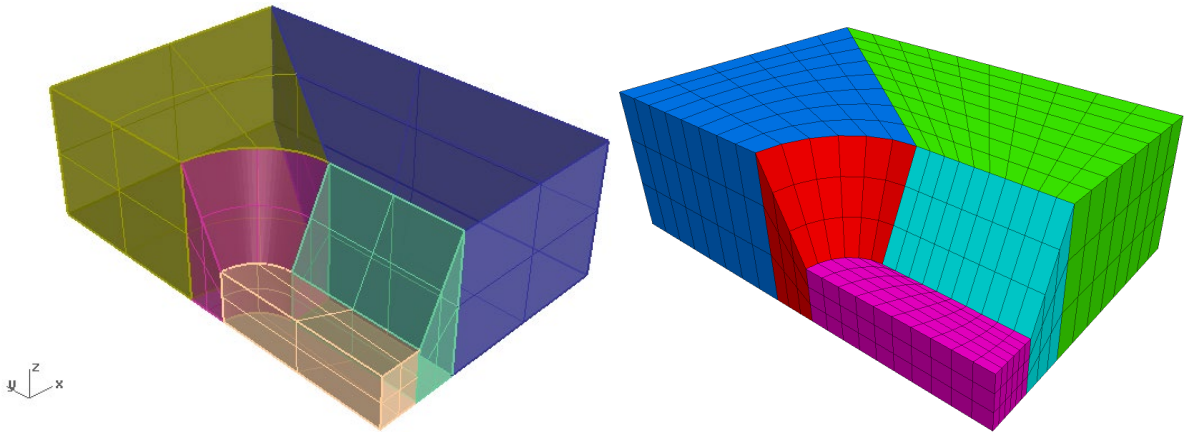


Figure 31: Solid layout resulting in five mesh blocks.

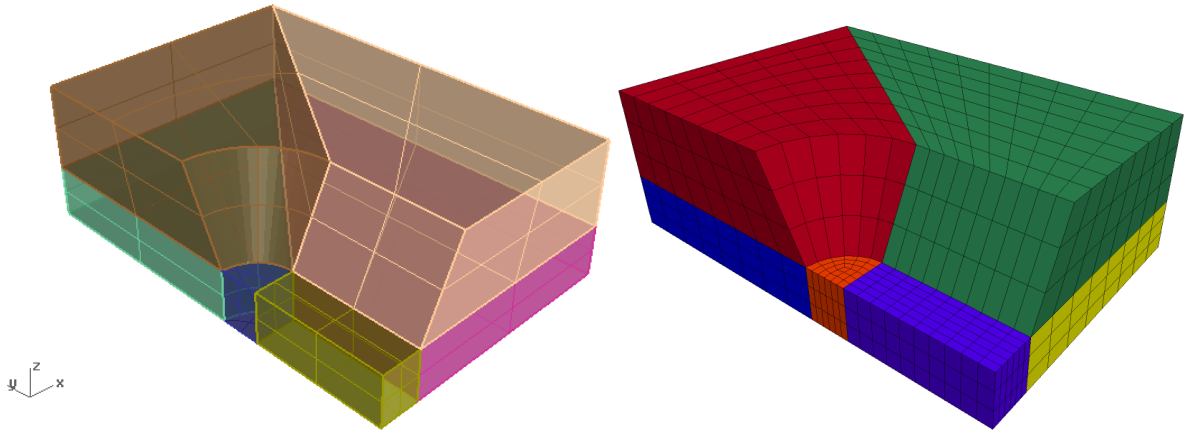


Figure 32: Solid layout resulting in six mesh blocks.

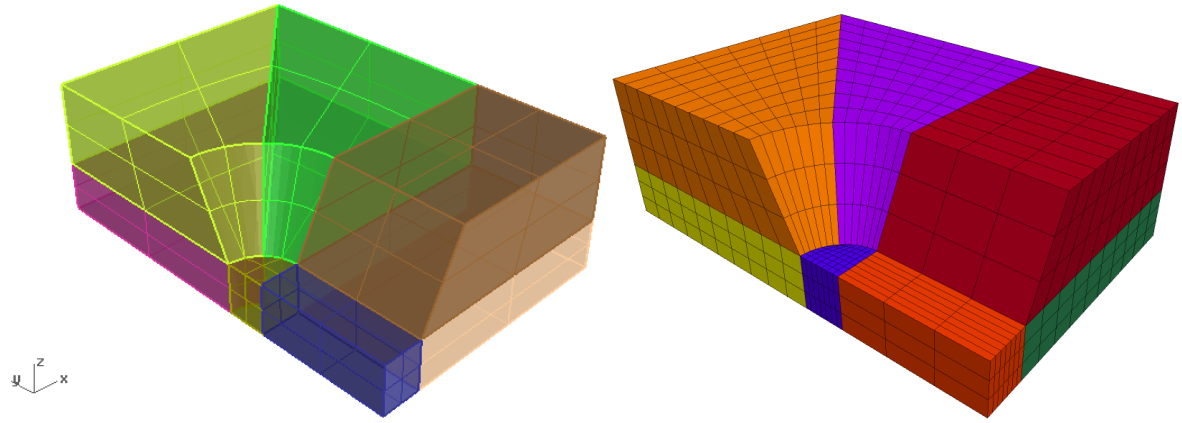


Figure 33: Solid layout resulting in eight mesh blocks.

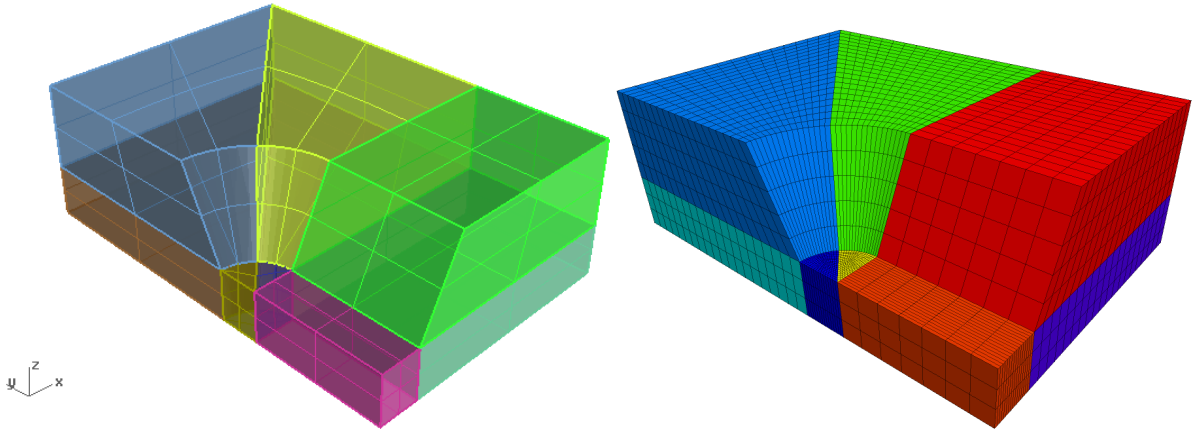


Figure 34: Alternate solid layout resulting in nine mesh blocks.

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.

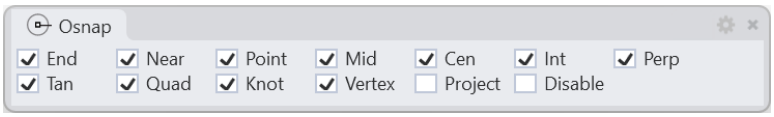


Figure 35: *Rhino* status bar.

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.

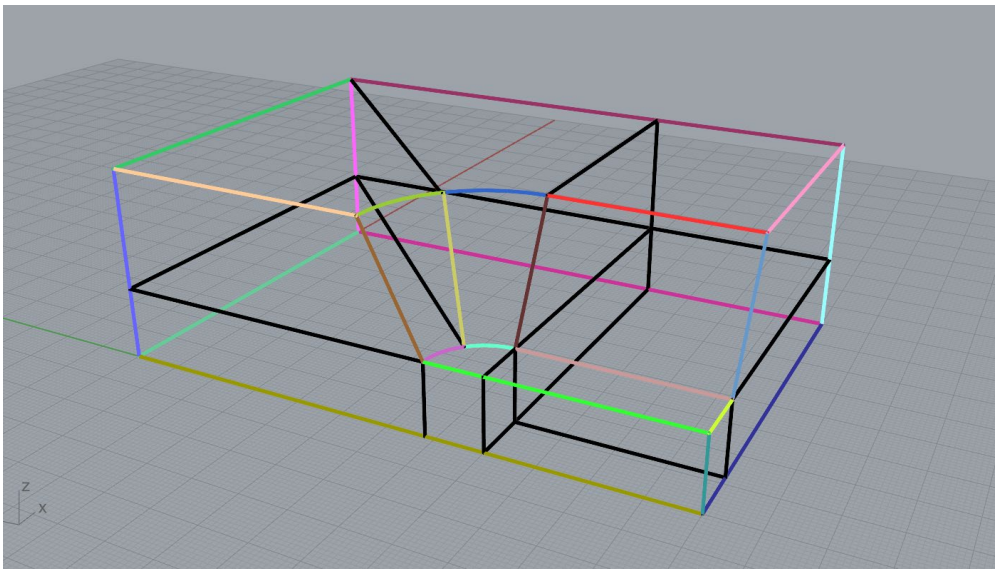


Figure 36: Creating new construction lines (in black).

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.

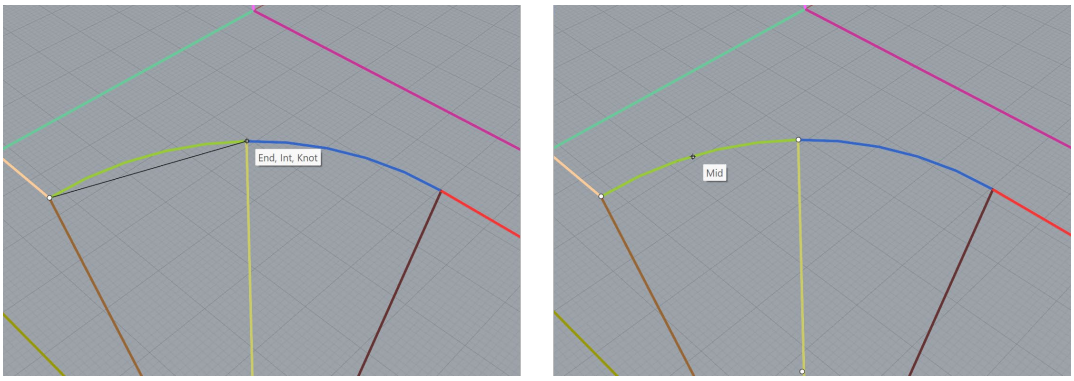


Figure 37: Building arcs.

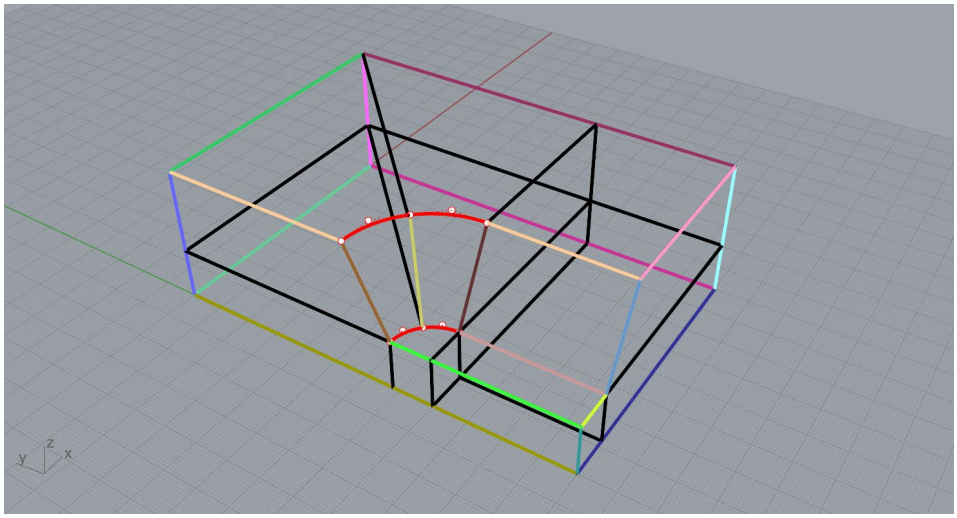


Figure 38: Red arcs retrace the original polylines at the curved section of the model.

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.

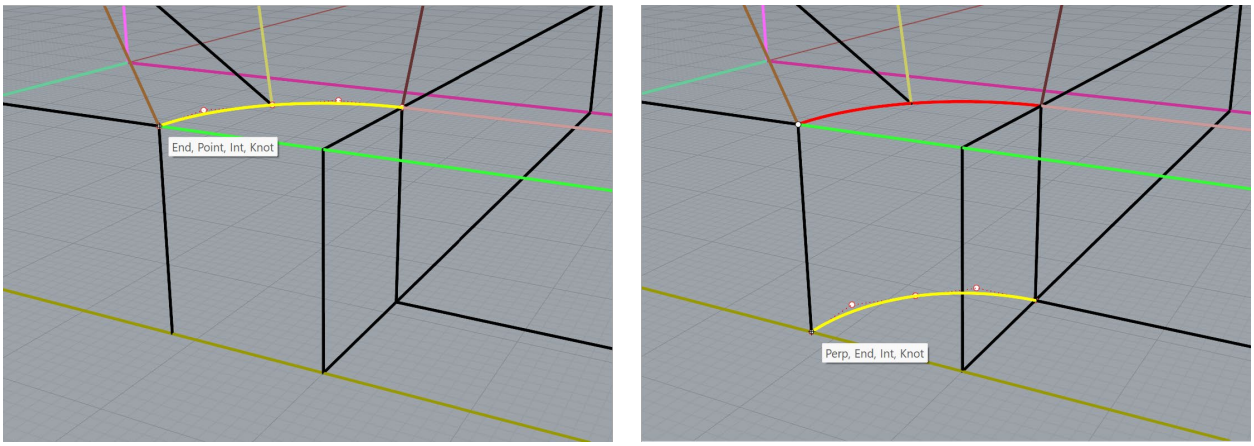


Figure 39: Copying the arcs.

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.

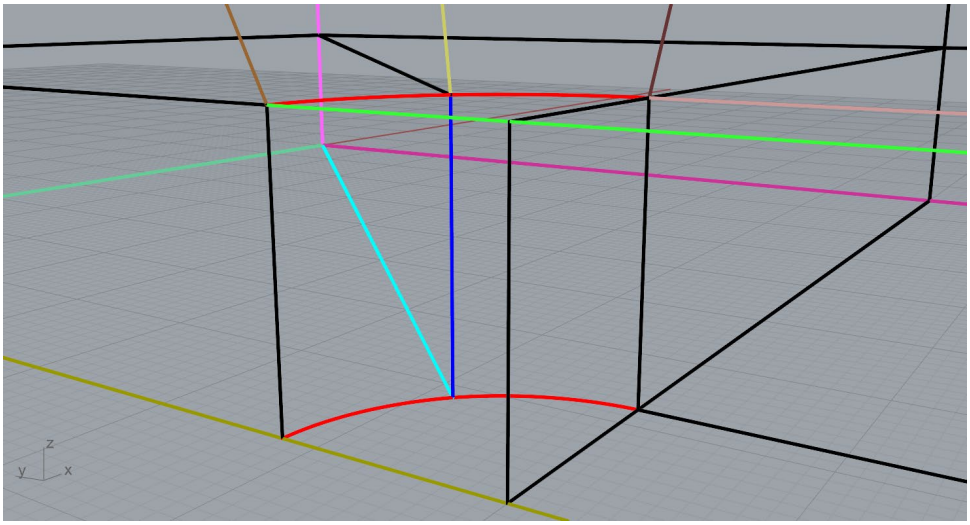


Figure 40: Connecting the middle ends of the arcs (blue lines).

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.

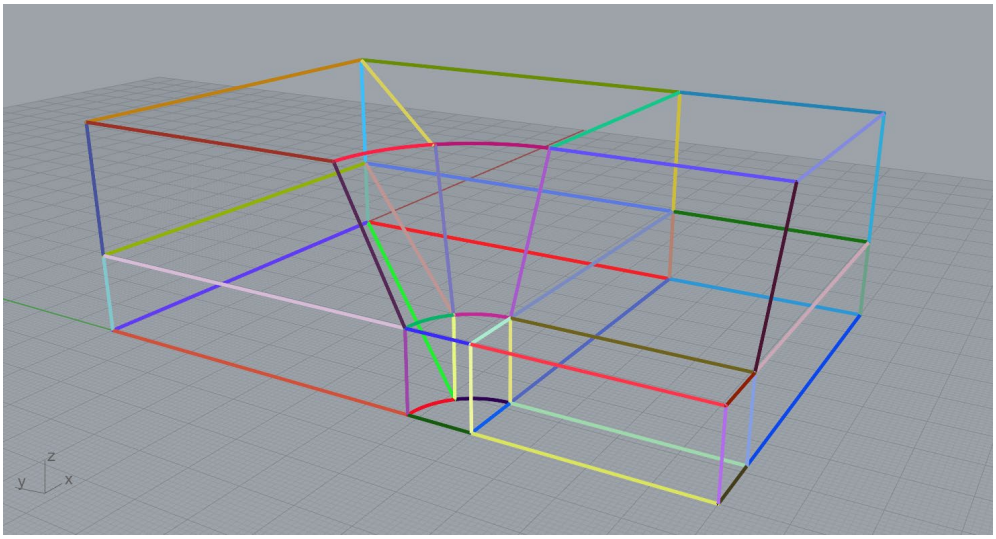


Figure 41: Model wireframe ready to be used for block creation.

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.

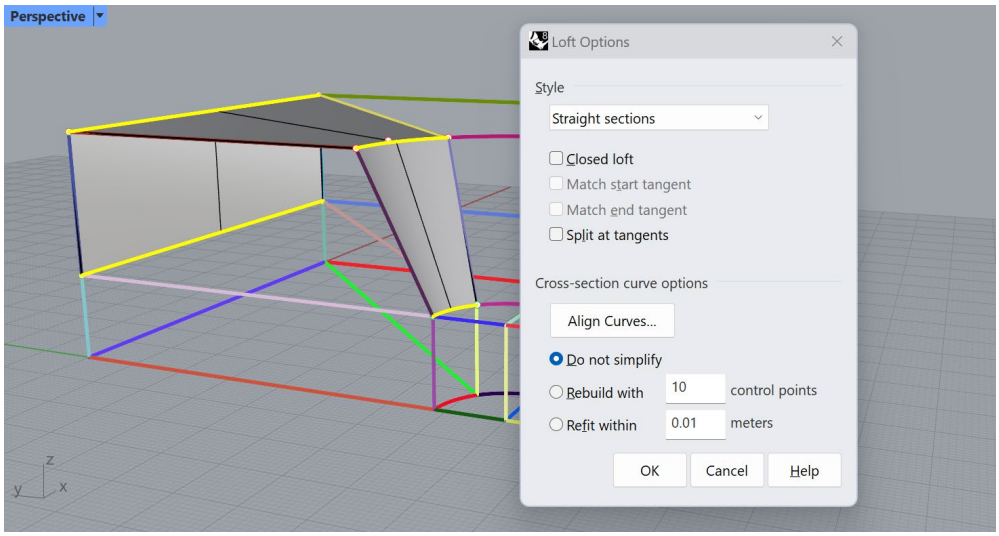


Figure 42: Building the surfaces for the first solid. Note that the grid has been hidden on this image.

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.

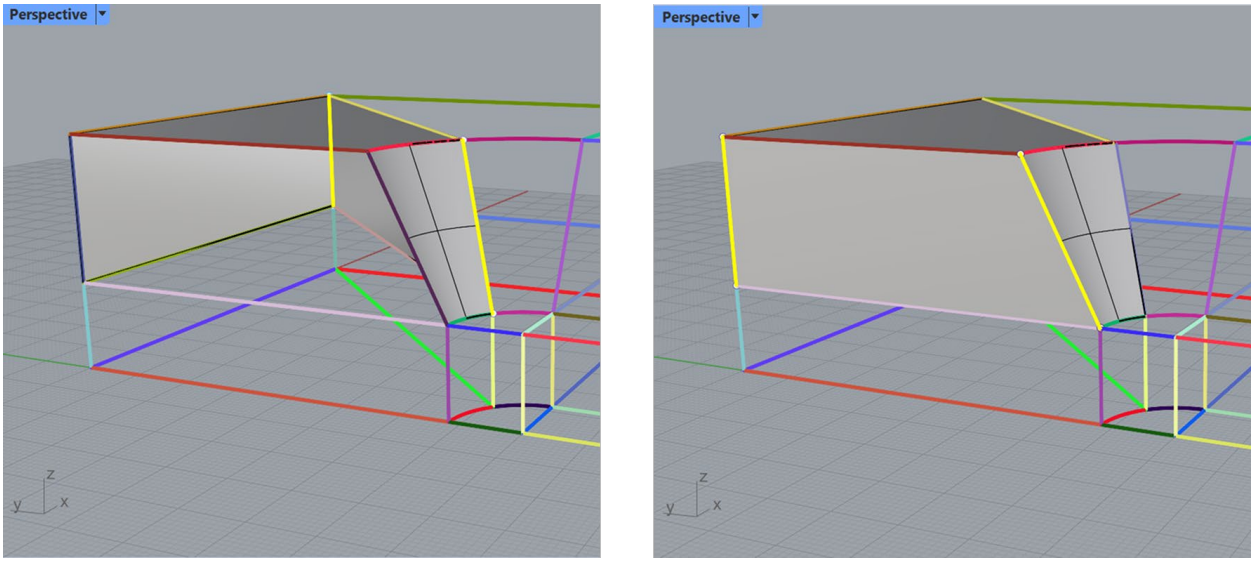


Figure 43: Building surfaces of the first solid.

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.

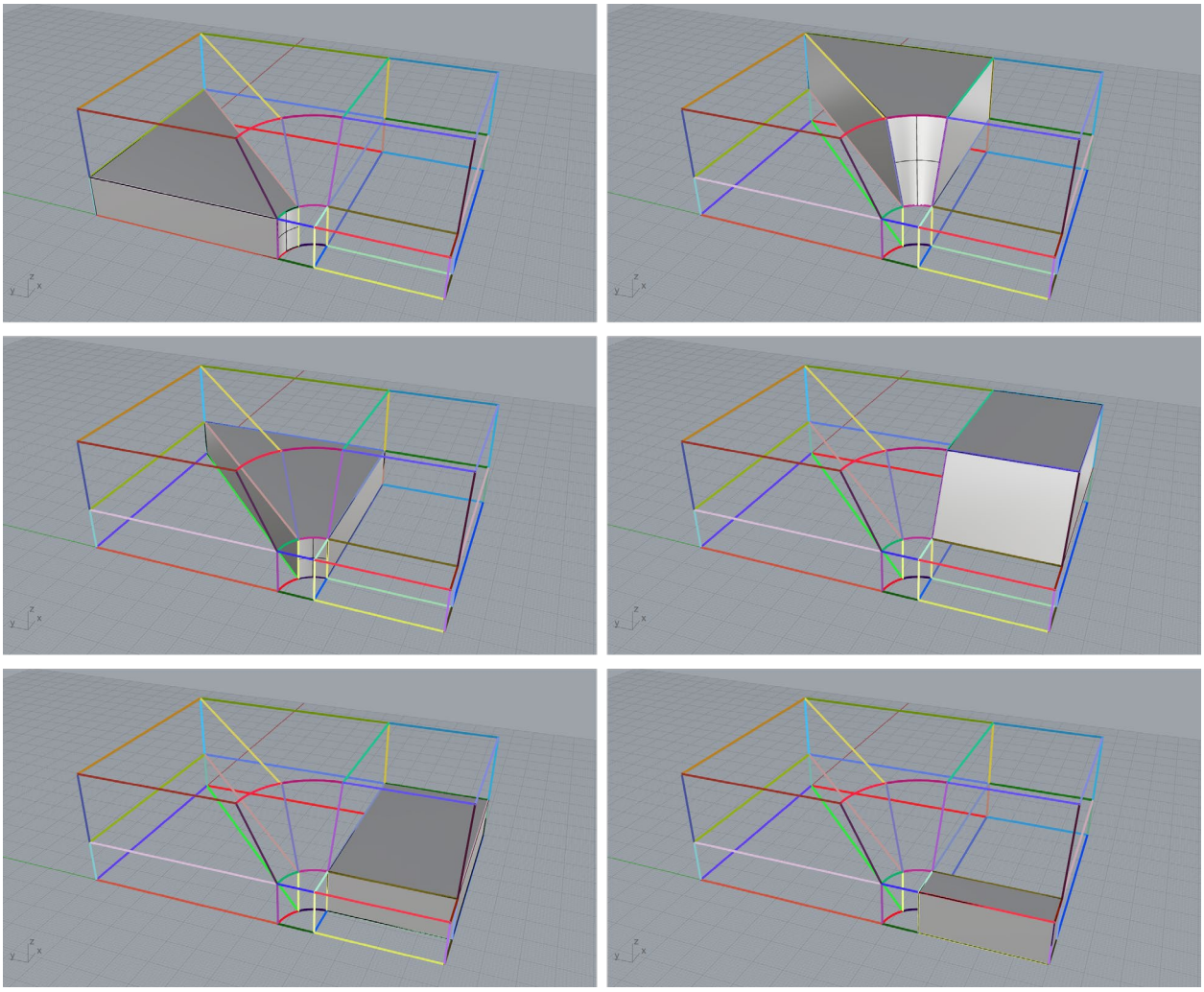


Figure 44: Building solid blocks for the model.

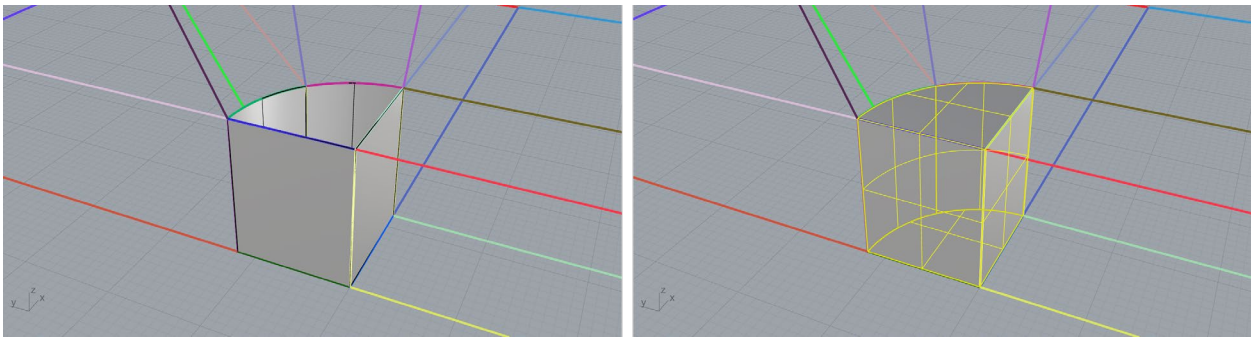


Figure 45: Building the last solid block in the model.

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.

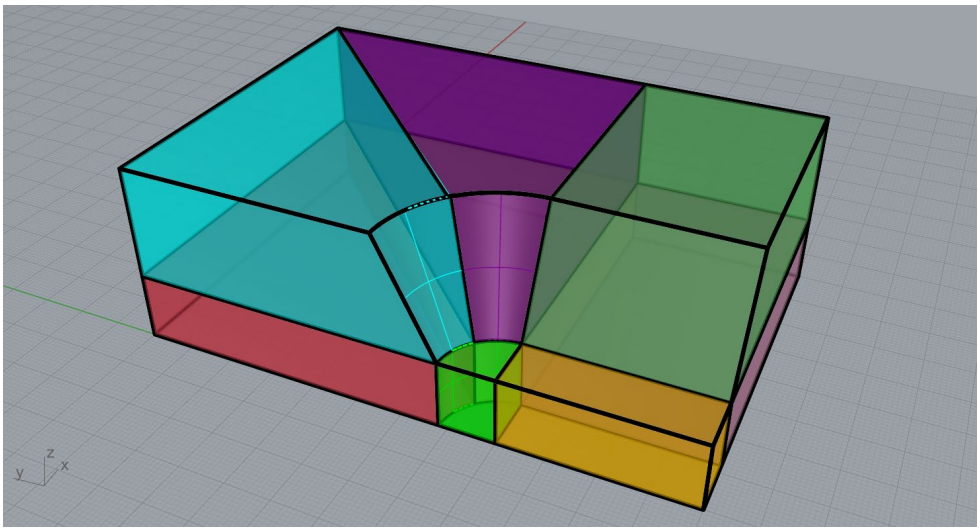


Figure 46: Solid blocks of the model (Ghosted view).

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.

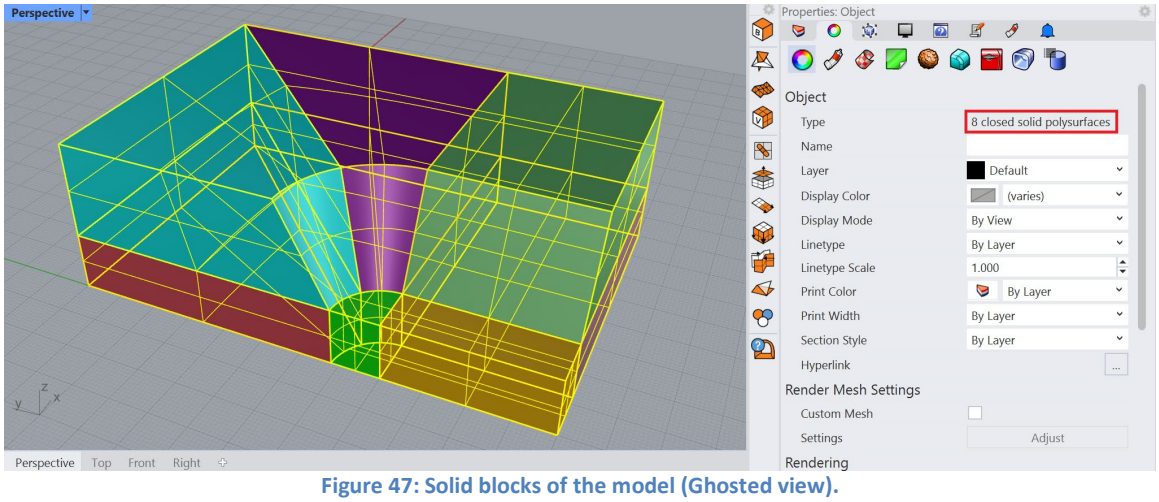


Figure 47: Solid blocks of the model (Ghosted view).

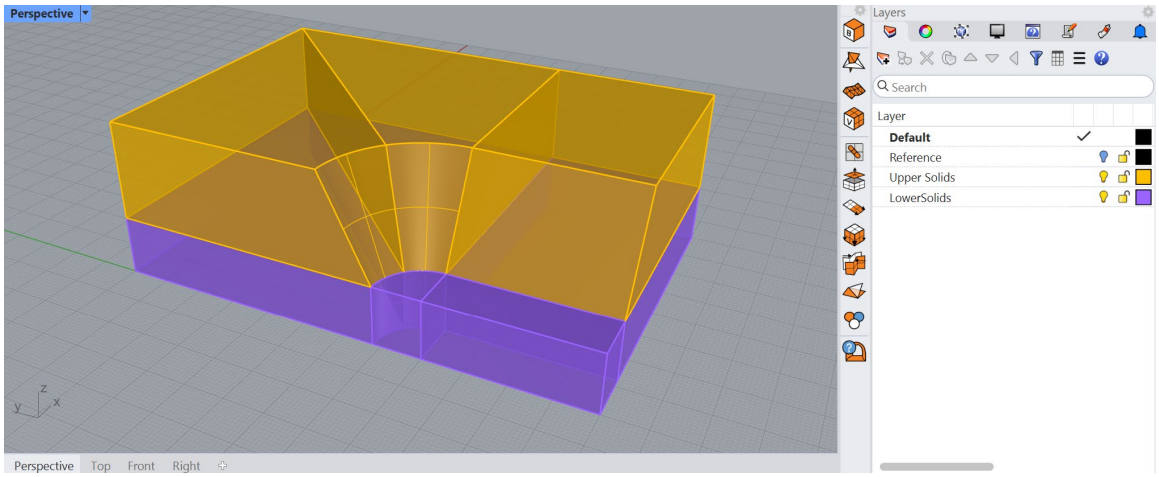


Figure 48: The model after the "Upper solids" and "Lower solids" layers have been specified (Ghosted view).

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.

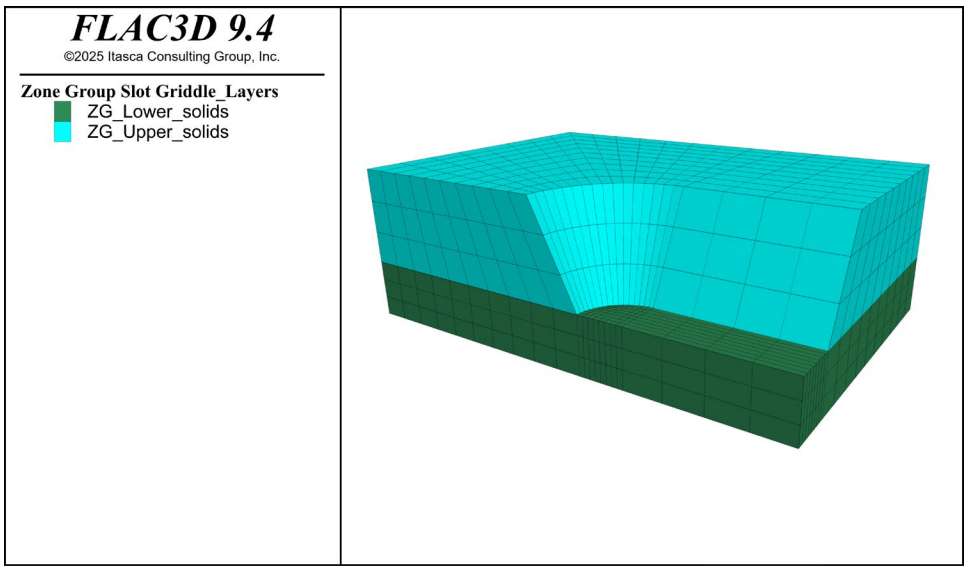


Figure 49: In *Rhino*, solids organized in layers are processed into *FLAC3D* groups in slot "Griddle_Layers".

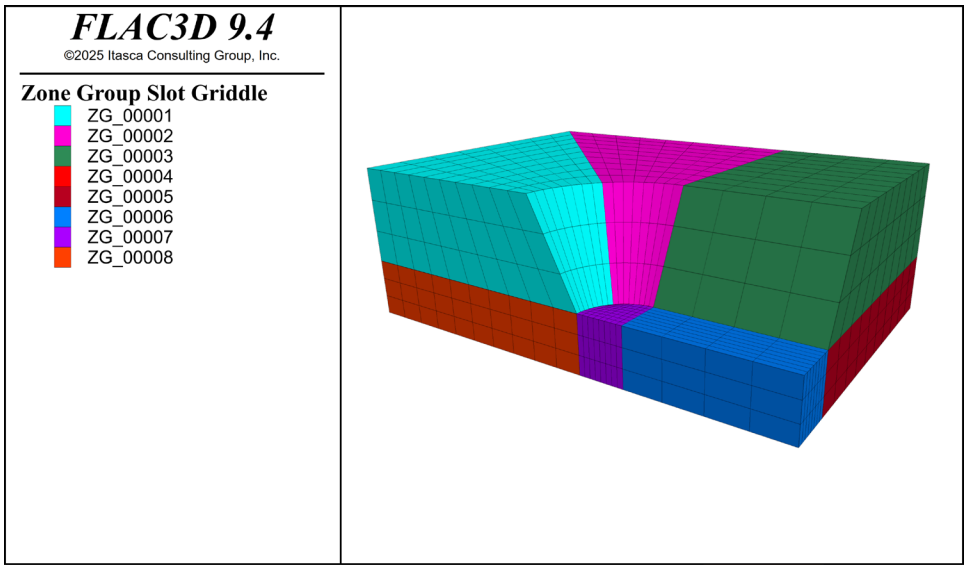


Figure 50: *FLAC3D* slot "Griddle" groups correspond to the individual *Rhino* solids.

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**.

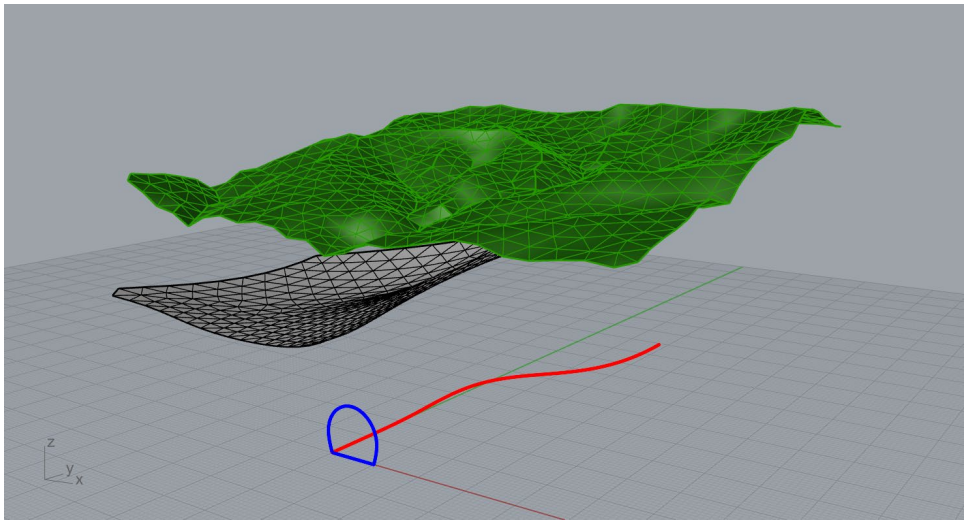


Figure 51: DXF geometry imported to *Rhino* corresponding to the tunnel profile and excavation direction.

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.

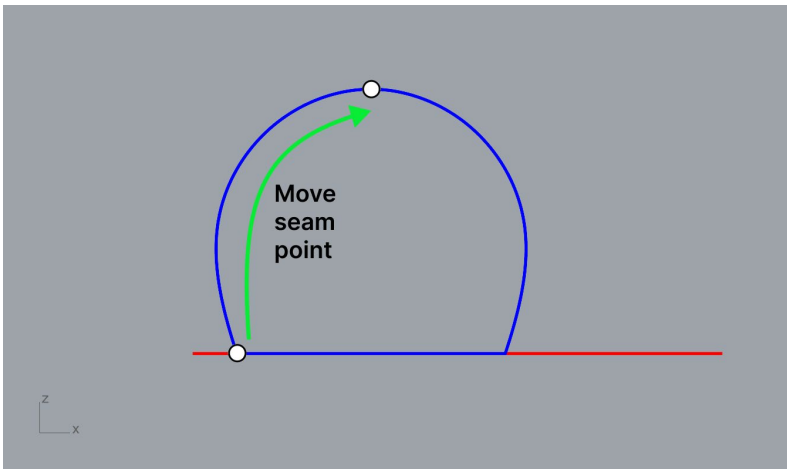


Figure 52: Drag seam point to the top of the tunnel profile.

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.

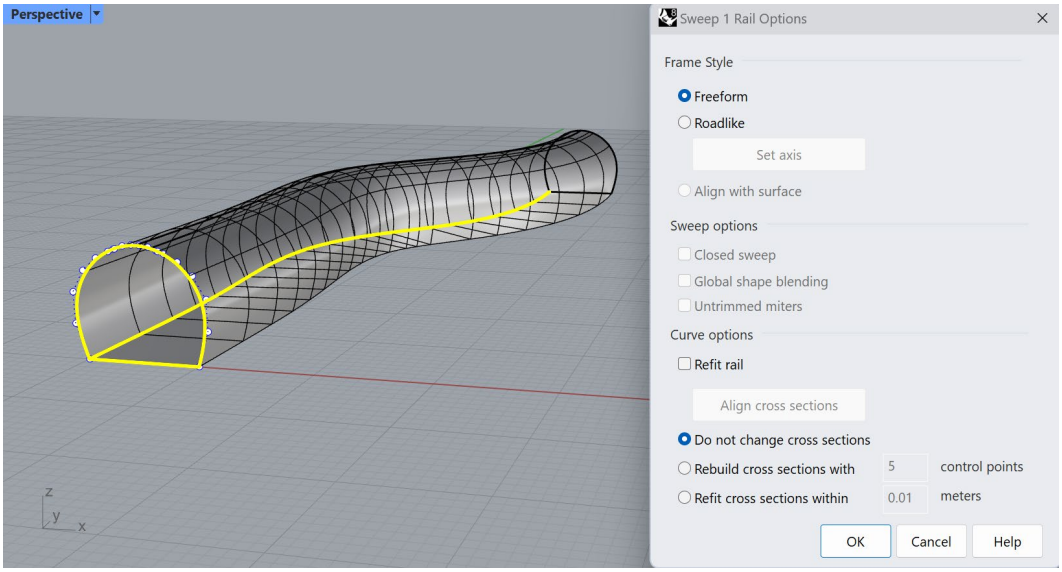


Figure 53: Creation of tunnel surface by sweeping the profile curve along the direction path.

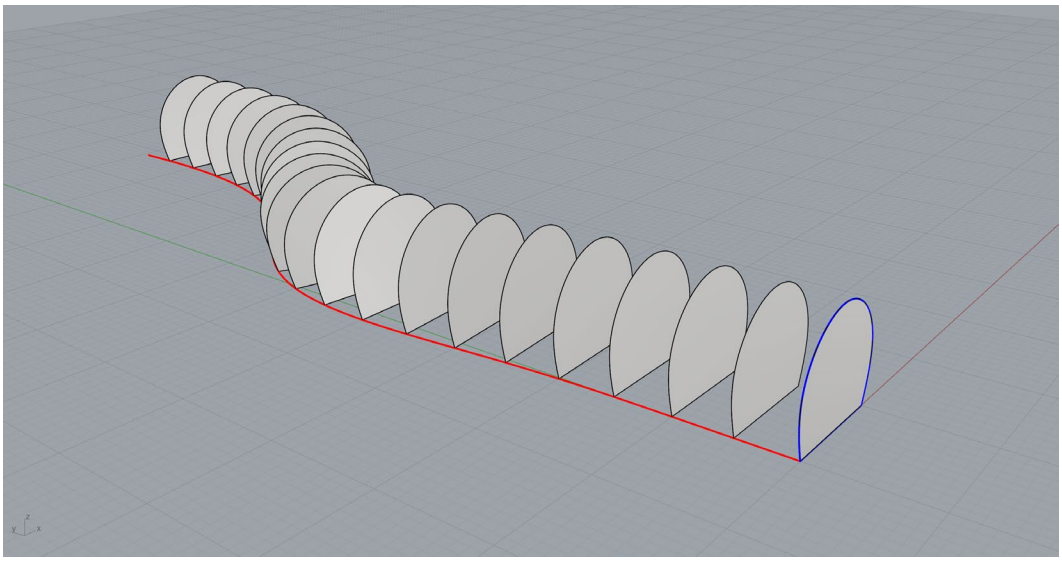


Figure 54: Creation of excavation stages by duplicating initial cross-section surface. Tunnel surface is not shown.

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.

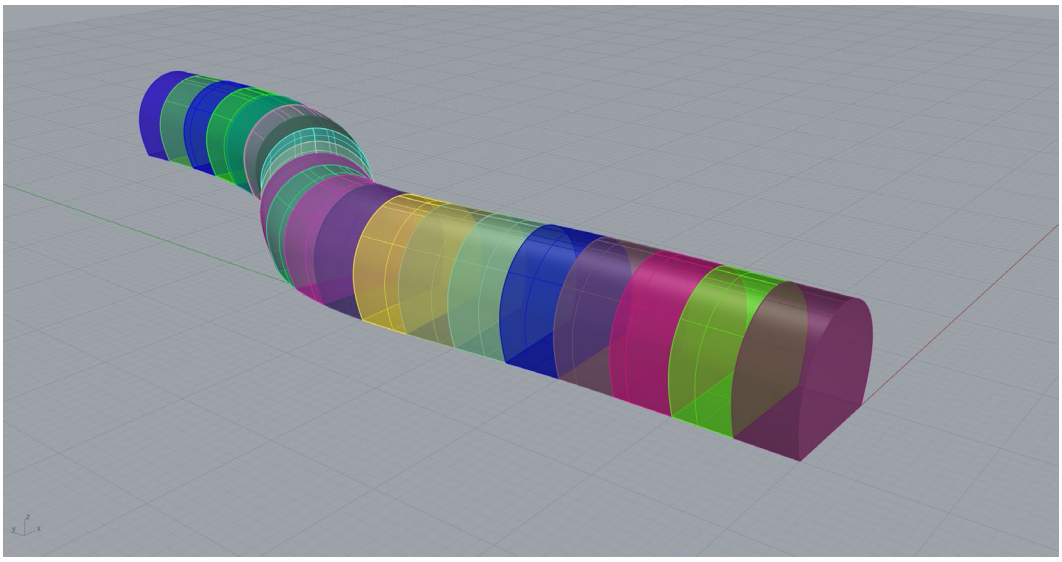


Figure 55: Watertight polysurfaces (solids) representing each excavation stage of the tunnel (Ghosted view).

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.

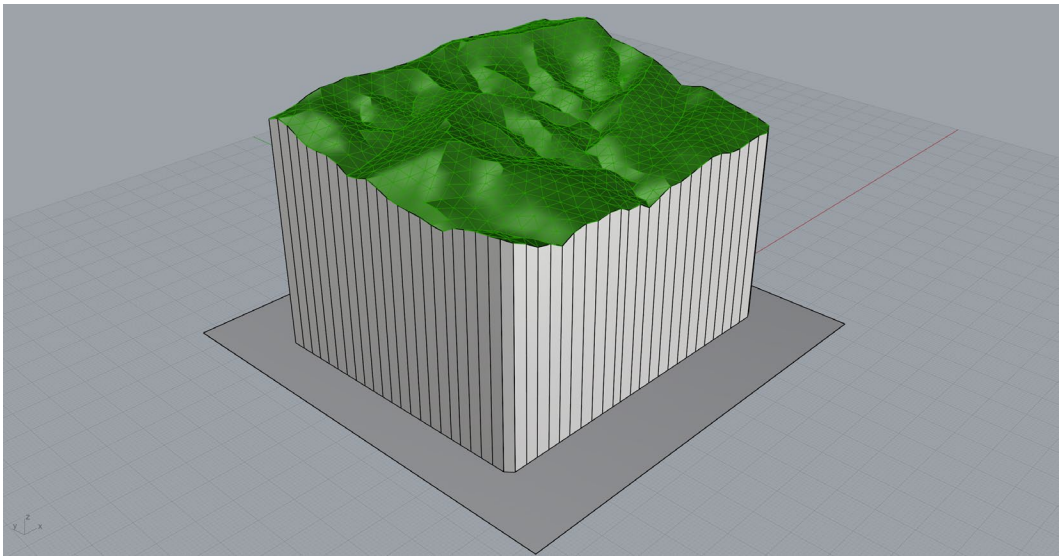


Figure 56: Extrusion of topographic surface border.

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.

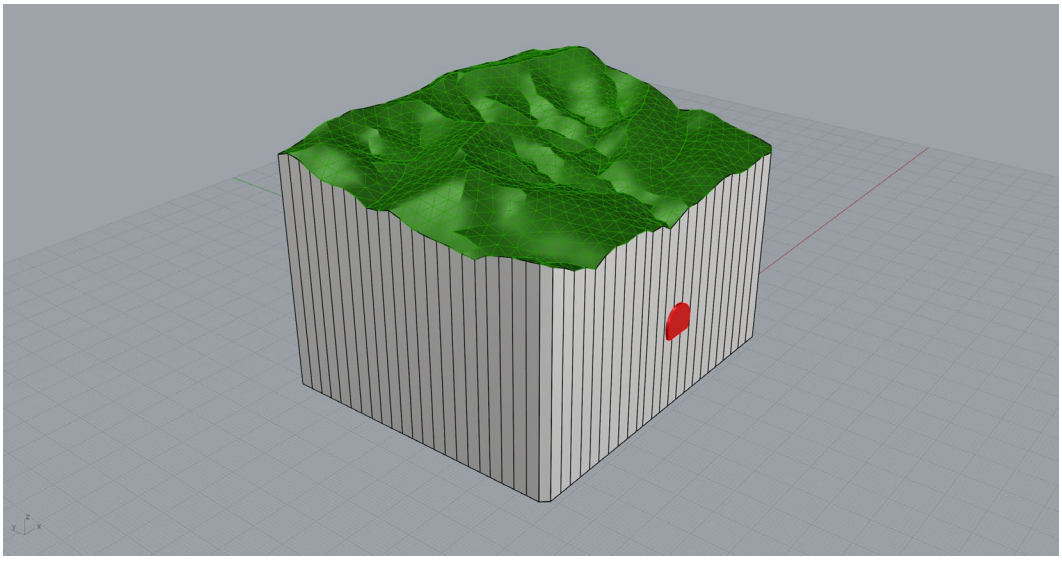


Figure 57: The front part of the tunnel extends through the modeling domain.

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  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.

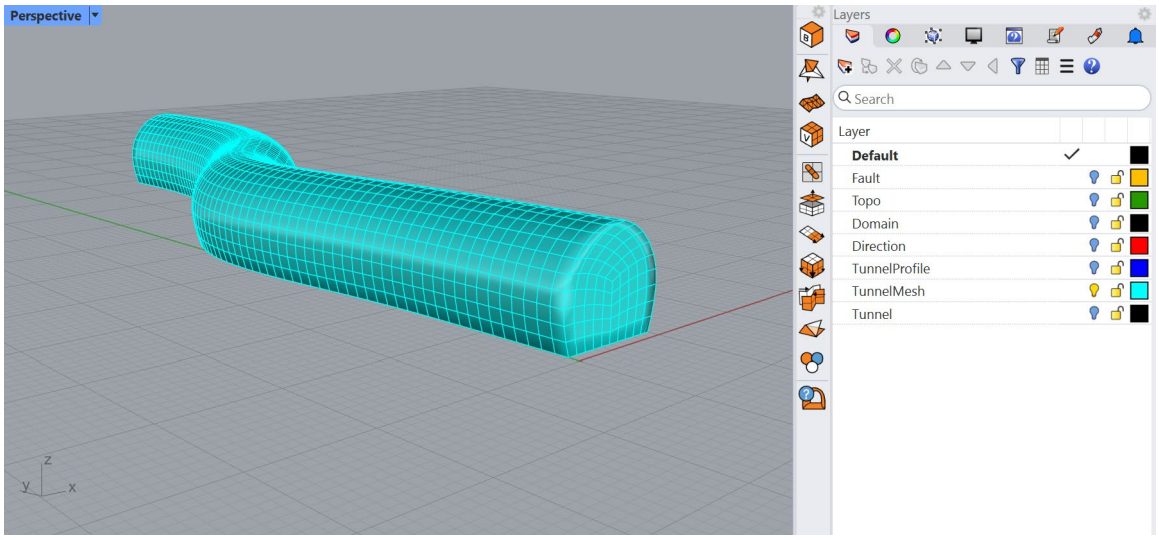


Figure 58: Surface mesh corresponding to the exterior of the tunnel's structured volume mesh.

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).

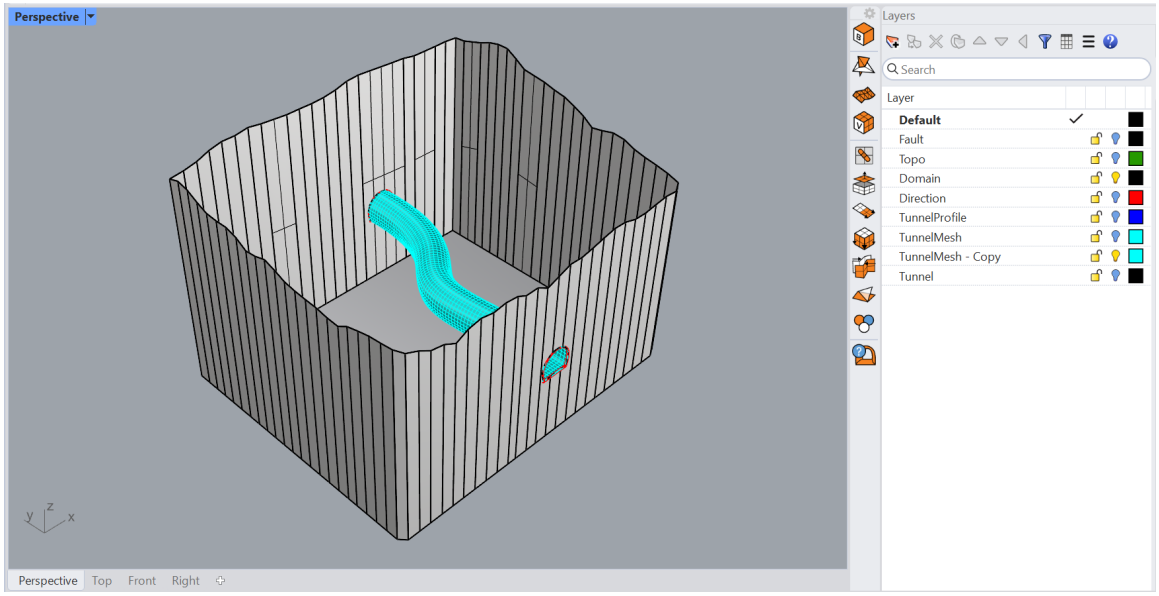


Figure 59: Removing parts of the domain at tunnel front and back.

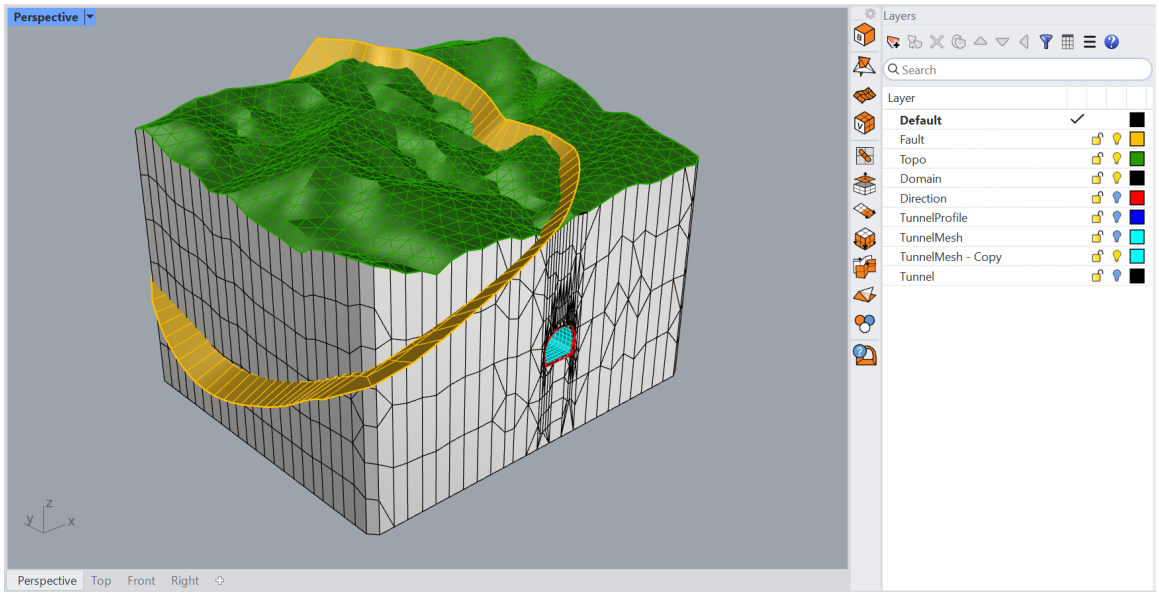


Figure 60: Extended fault mesh intersects domain and topo meshes. Note that the colors of topo and tunnel meshes and duplicated tunnel borders (curves) are changed from default.

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.

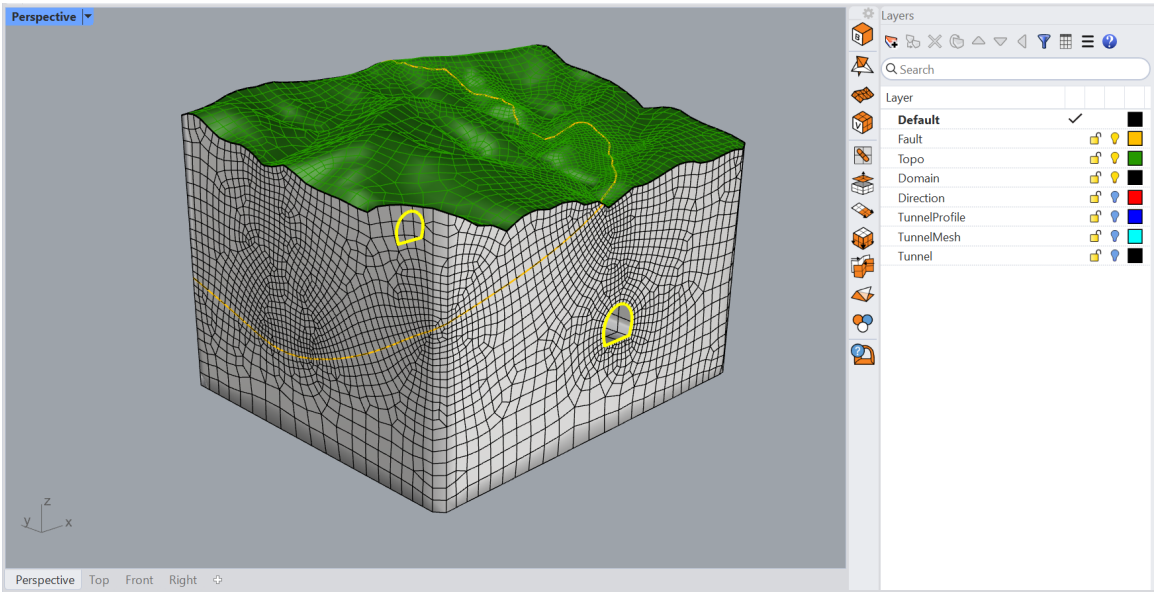


Figure 61: The result of remeshing of all meshes (except for the tunnel mesh) while preserving mesh edges along the tunnel borders (hard edges).

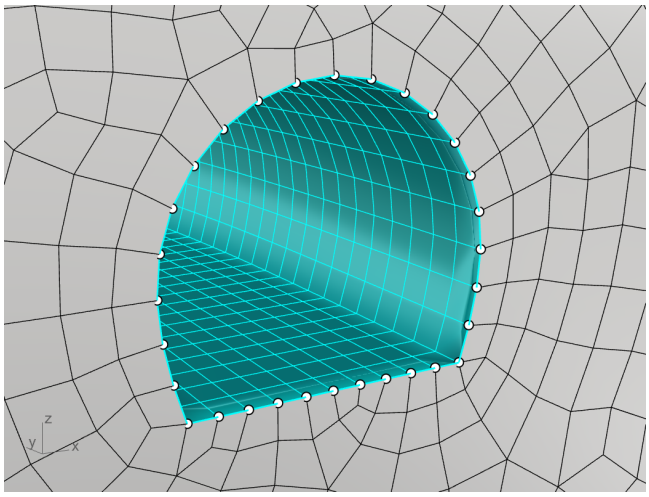


Figure 62: Close view of the tunnel front showing that tunnel and domain meshes are conformal.

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.

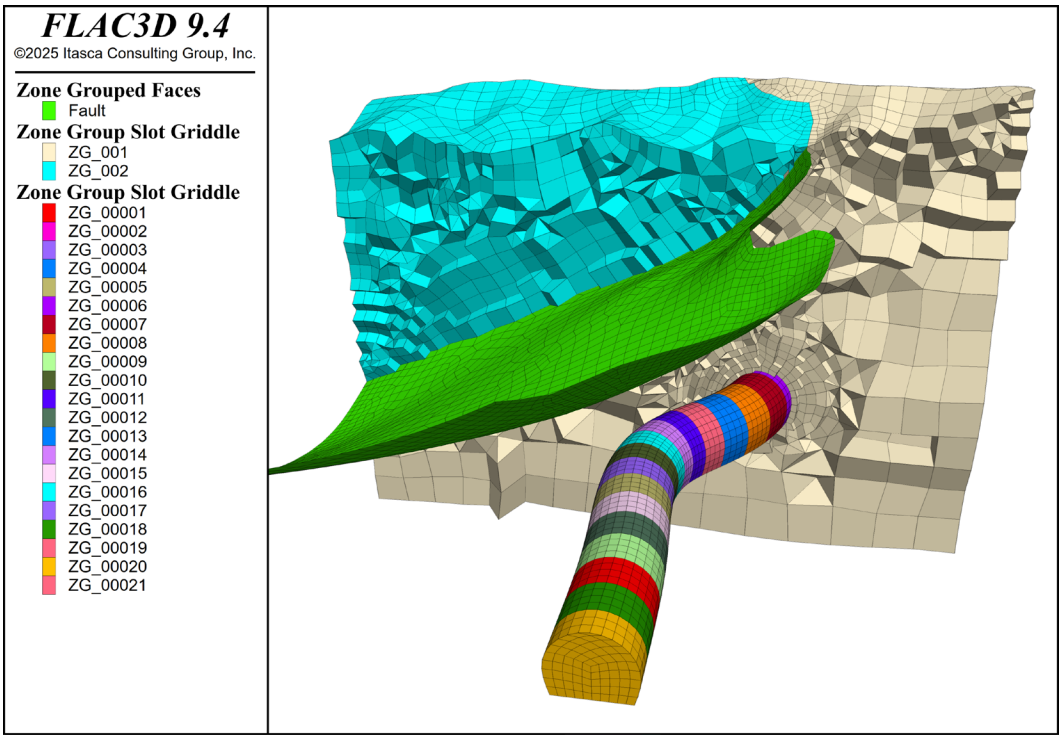


Figure 63: Structured and unstructured meshes loaded in *FLAC3D*.

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.

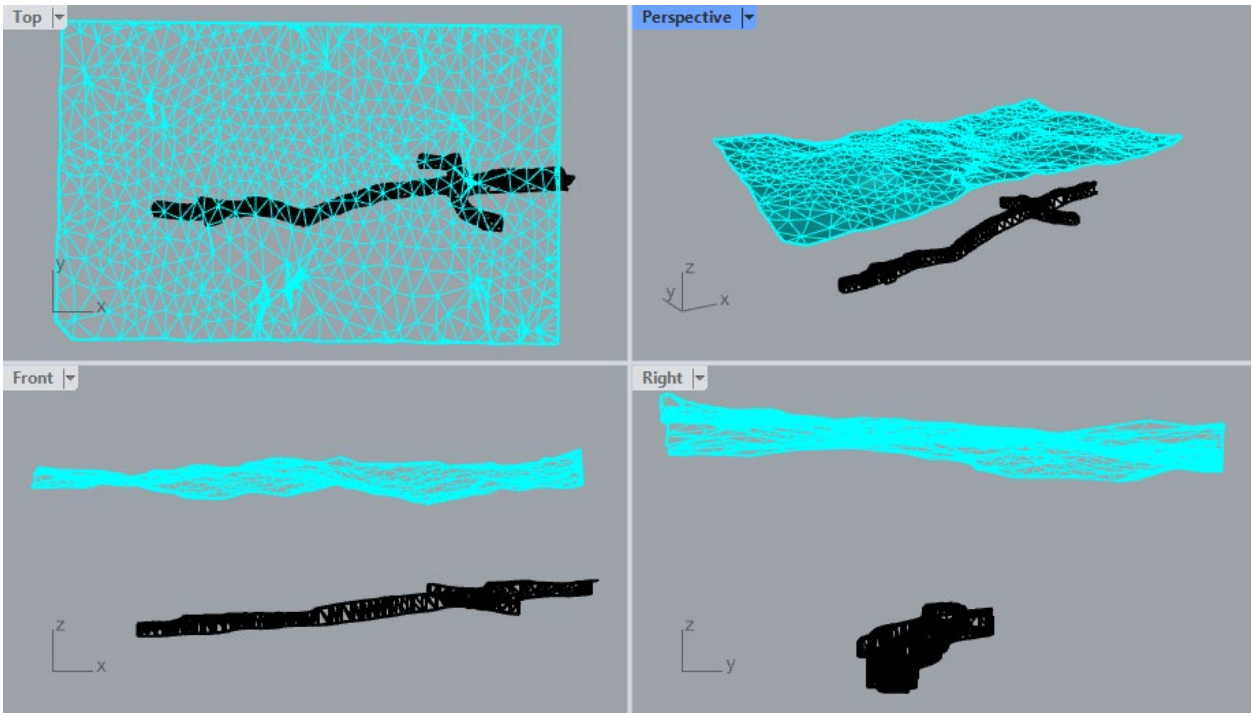


Figure 64: Drift geometry from a mine.

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**.

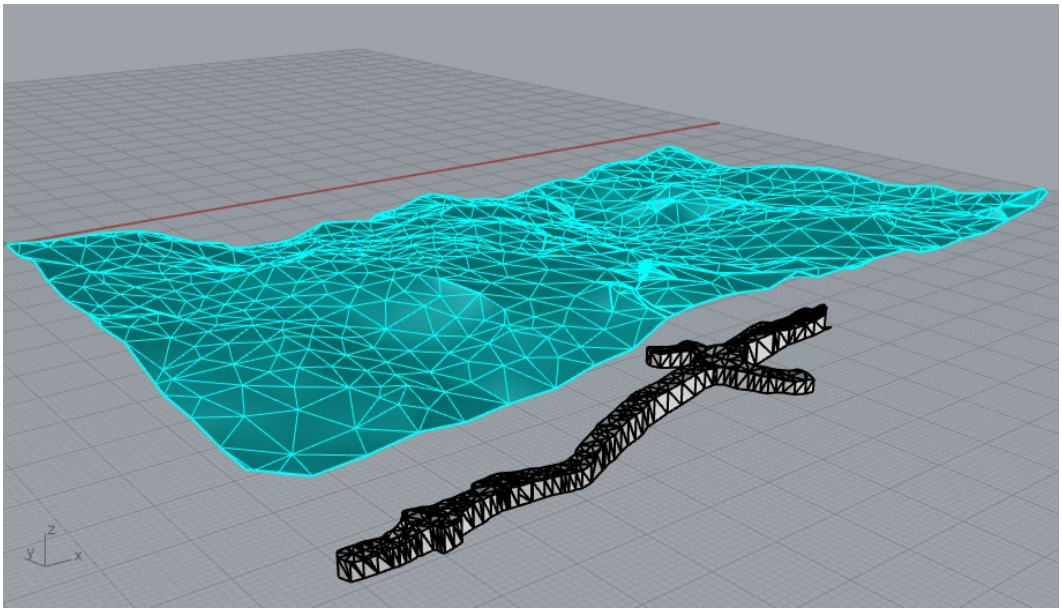


Figure 65: Drift model located near zero coordinates.

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).

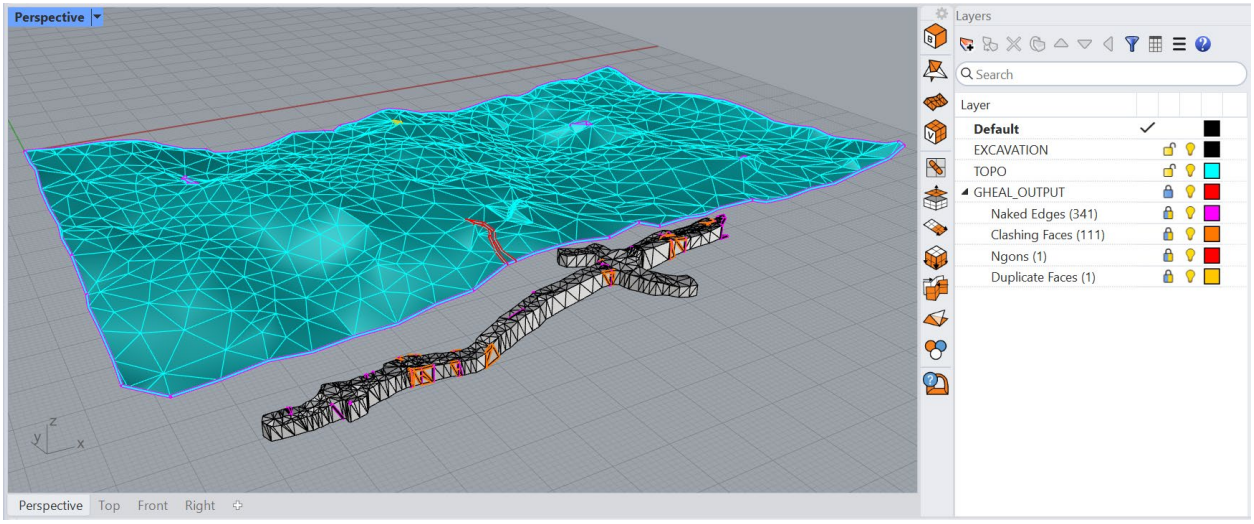


Figure 66: Displaying errors with GHeal.

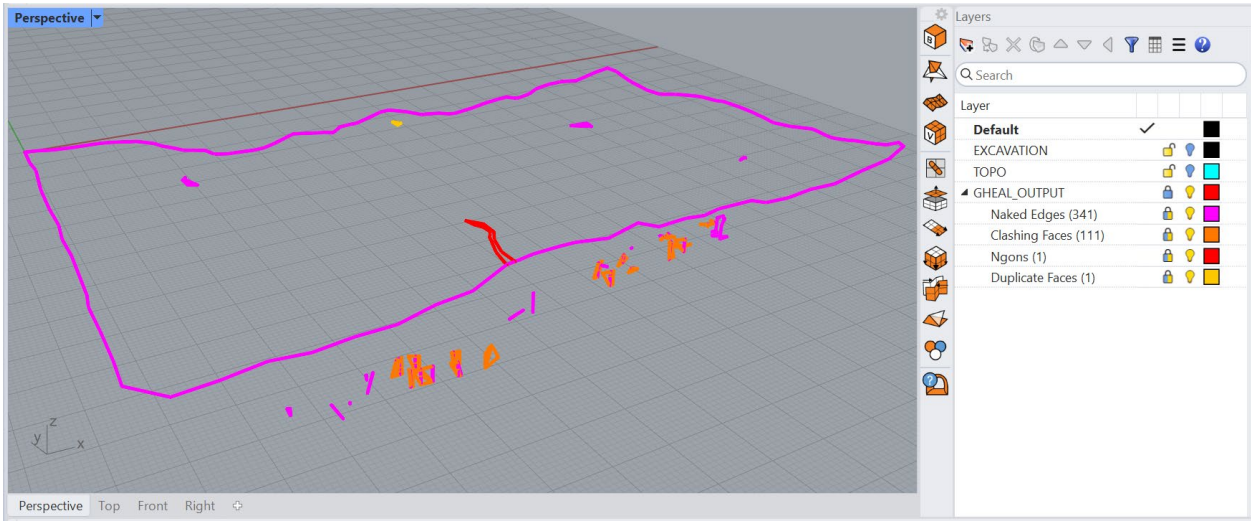


Figure 67: Displaying errors with meshes hidden.

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).

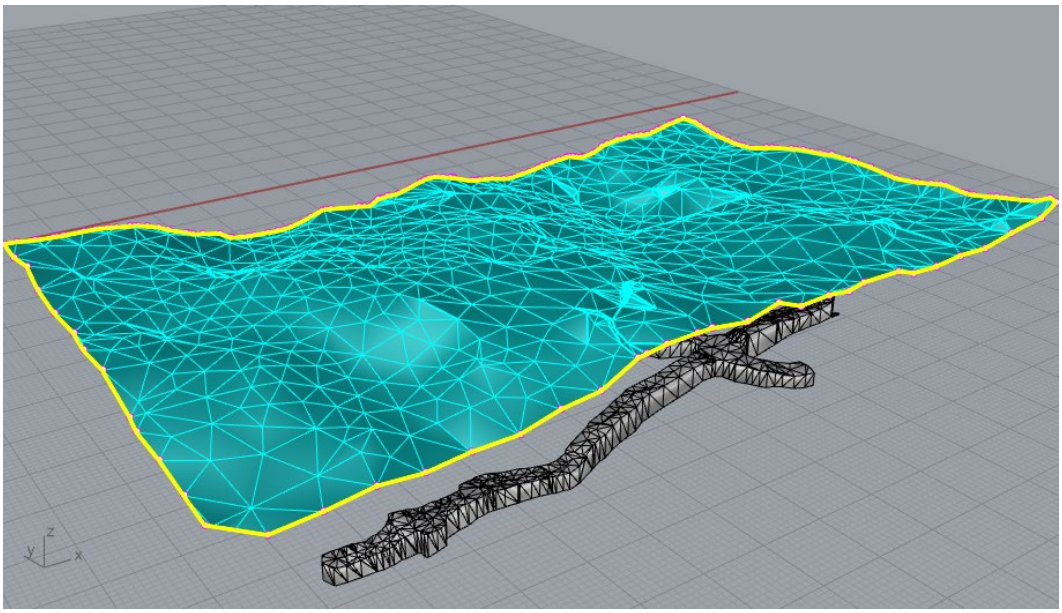


Figure 68: The fixed topo mesh with the remaining naked edges selected

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 (ensure no curves are selected), 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.

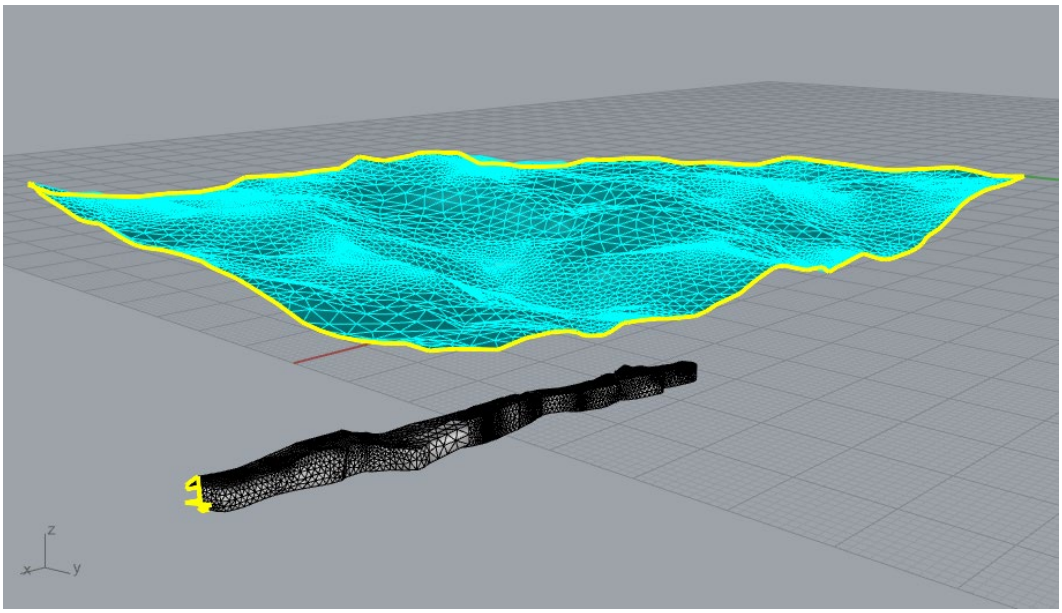


Figure 69: Remeshed and fixed meshes with remaining naked edges selected.

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.

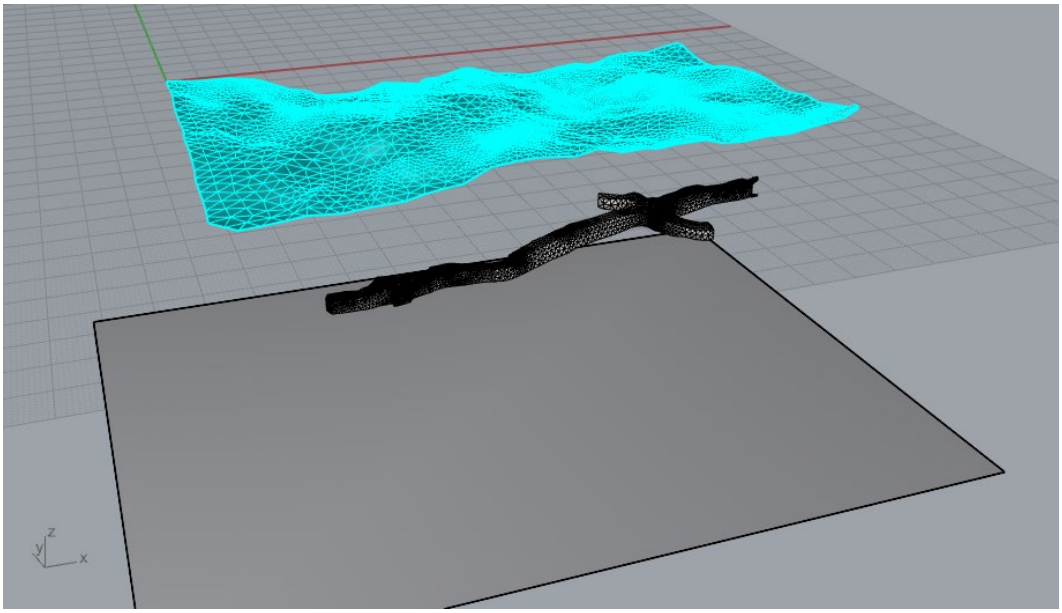


Figure 70: Creating a plane under the drift mesh.

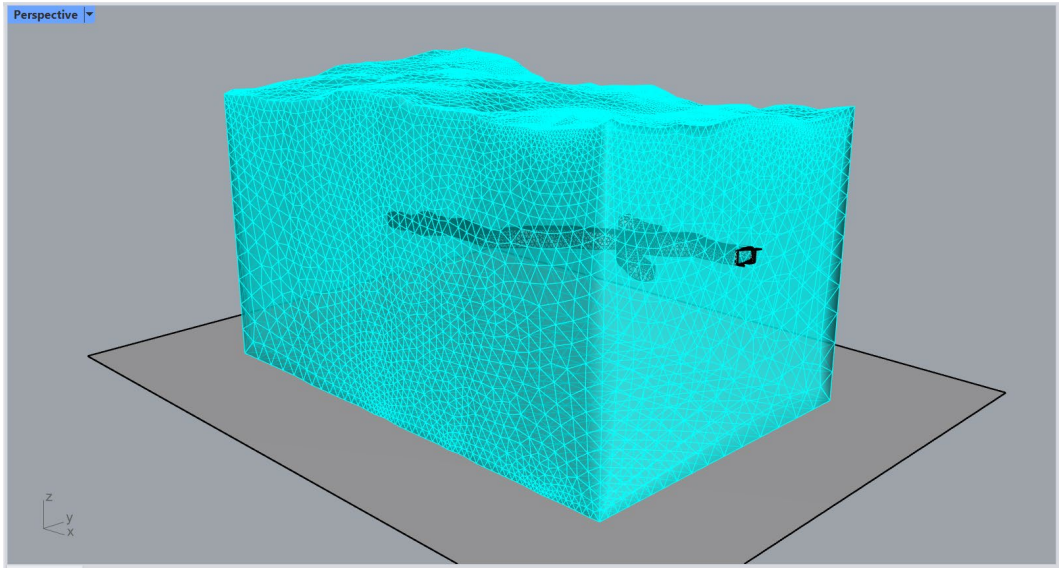


Figure 71: Creating watertight mesh domain with GExtrude command (Ghosted view).

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.

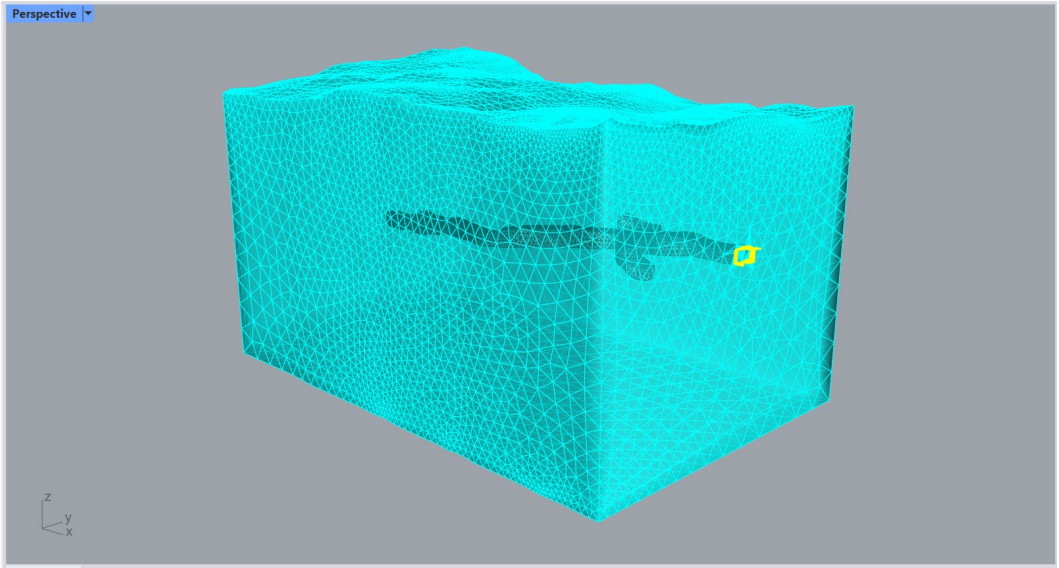


Figure 72: Meshes separated with the GExtract tool (Ghosted view).

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.

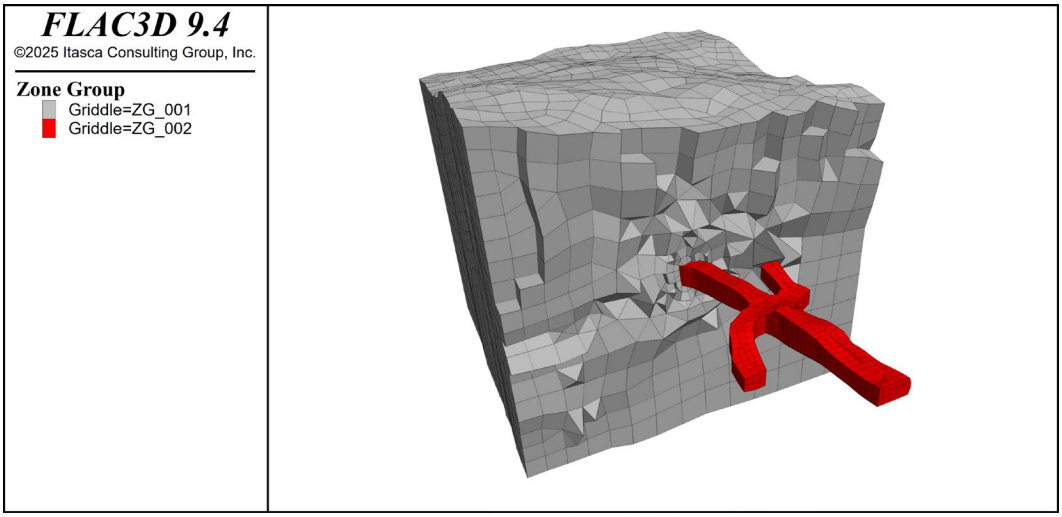


Figure 73: Final volume mesh for the model of the drift mine.

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**).

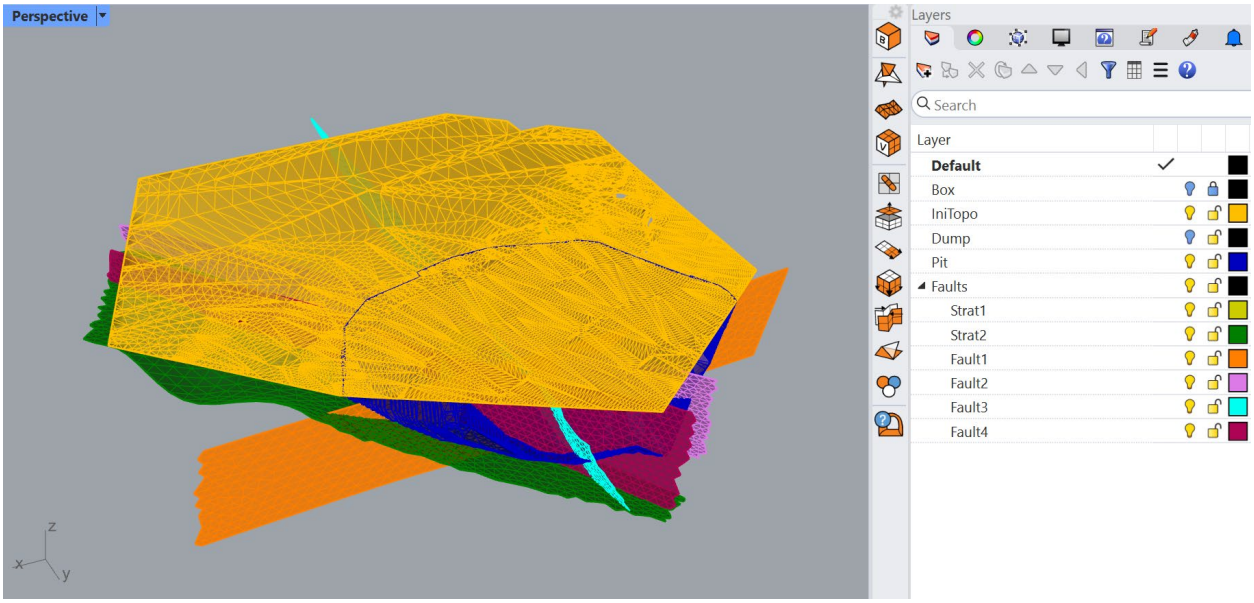


Figure 74: Initial model with topographic surface shown. Ghosted view.

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.

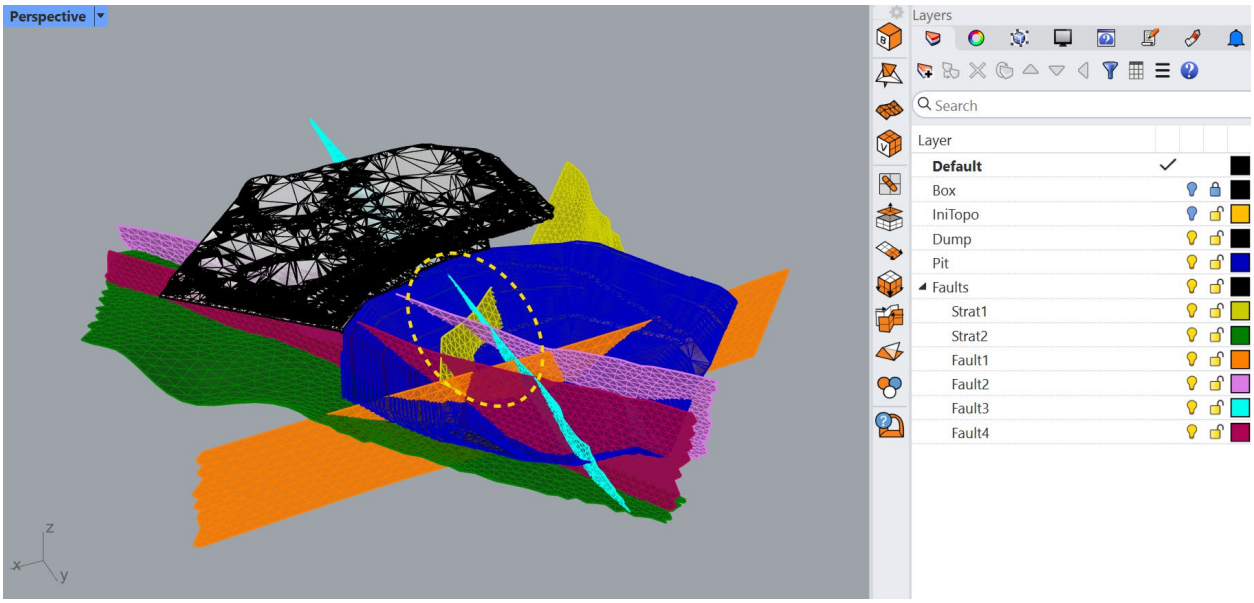


Figure 75: Initial model with large pit dump shown. Ghosted view.

- 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).

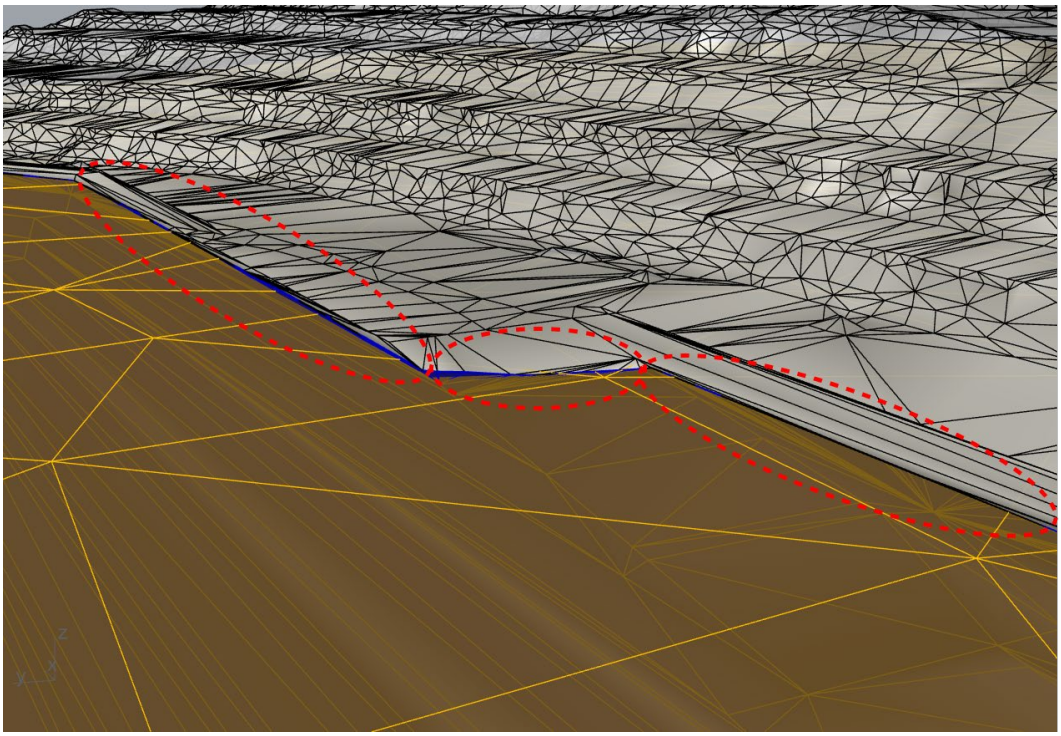


Figure 76: Zoomed-in view of the intersection between the pit, pit dump, and topo meshes. Marked regions denote some of the areas where intersections are not perfect.

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”.

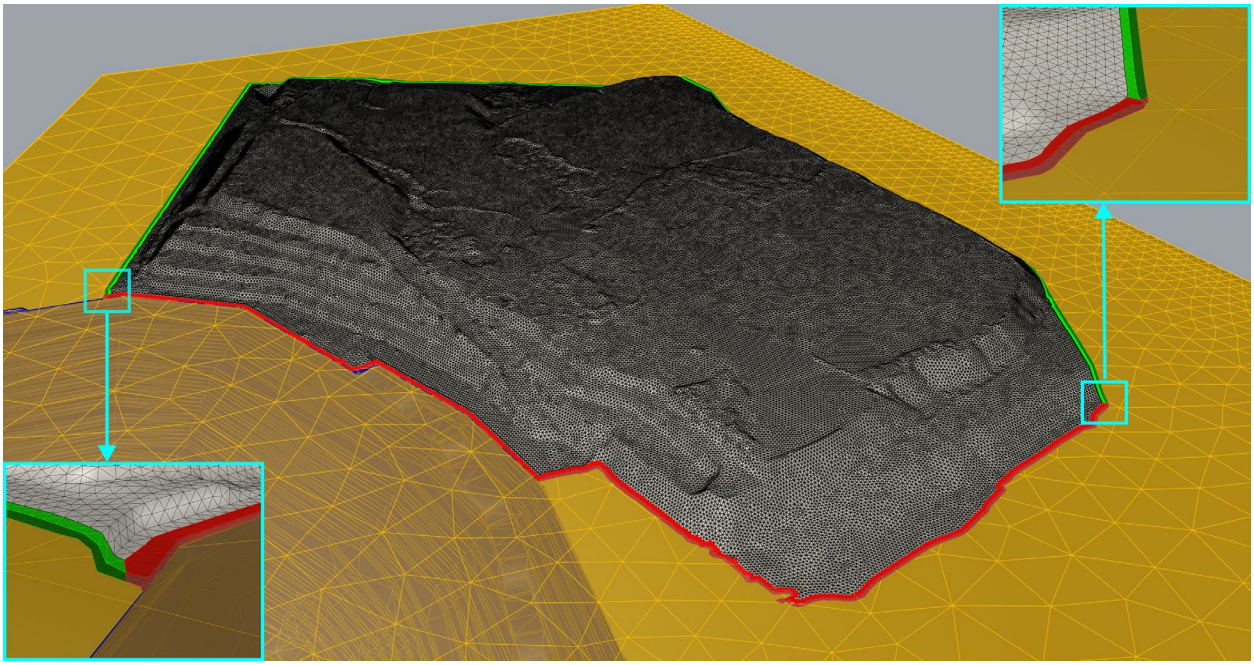


Figure 77: Boundary faces extracted from the pit dump mesh (note that colors are changed from the default gray color). Faces in red were separated and will be deleted.

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

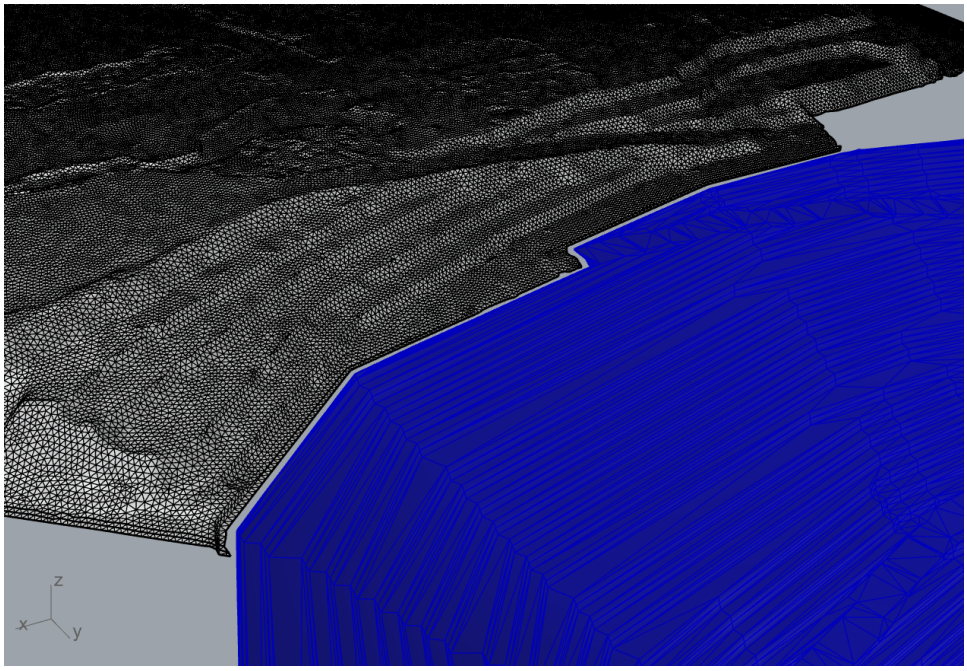


Figure 78: A gap between the pit and pit dump meshes.

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.

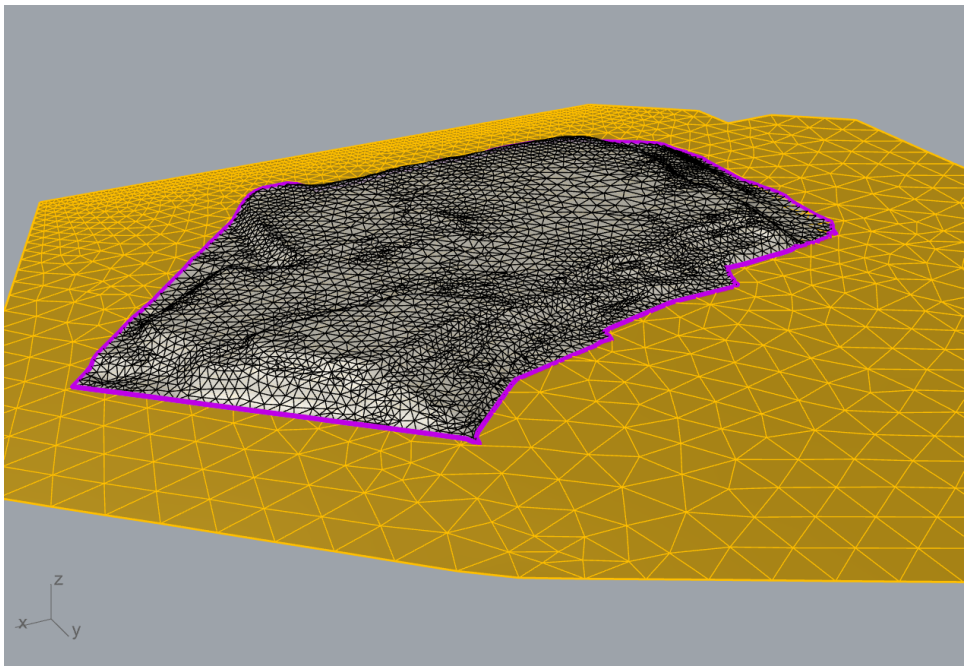


Figure 79: Selecting the mesh boundary to extend

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.

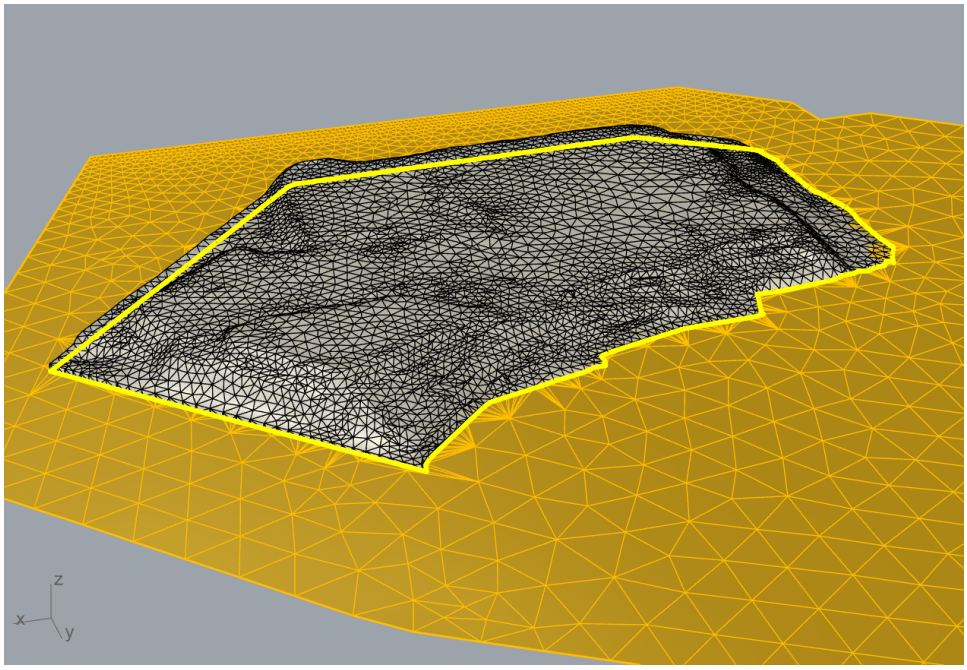


Figure 80: Extended pit dump mesh fully intersecting the topo mesh.

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].

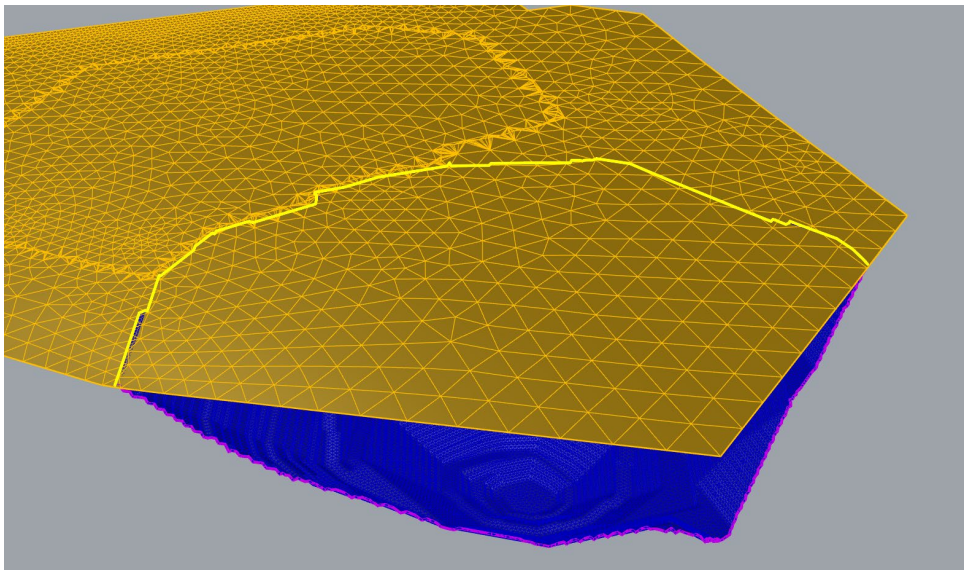


Figure 81: Selection of a piece of the pit boundary for mesh extension.

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.

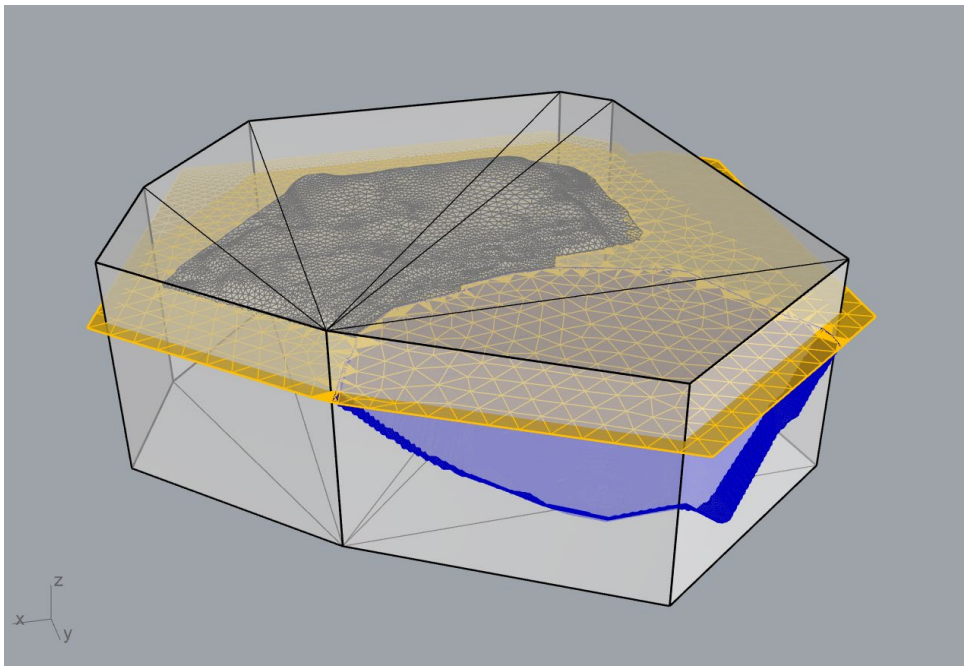


Figure 82: The box, pit, and dump meshes with the last two extended and intersected with the topo mesh.

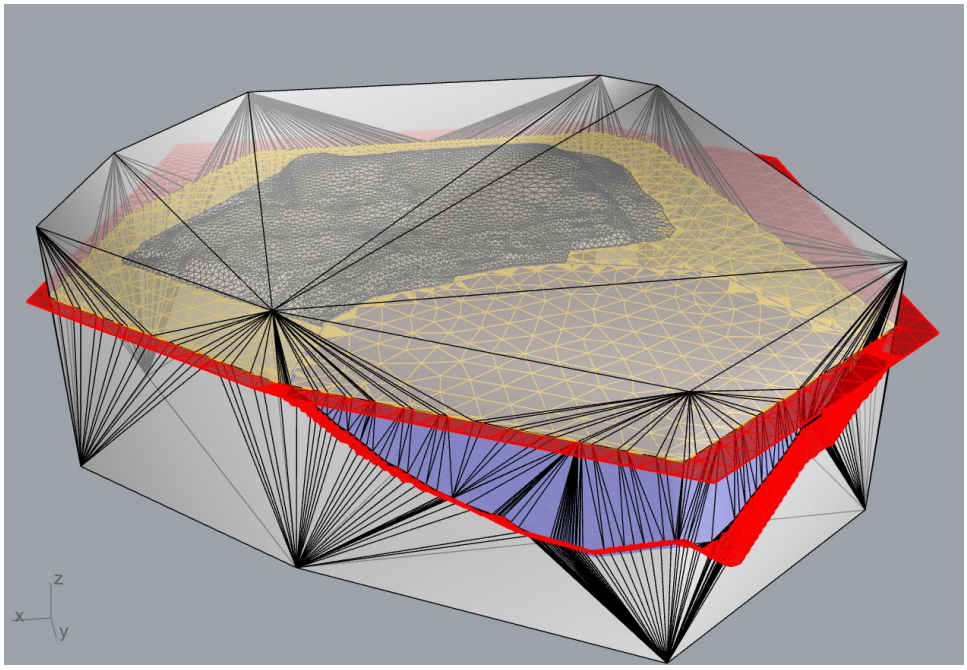


Figure 83: Trimming the topo and pit meshes.

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.

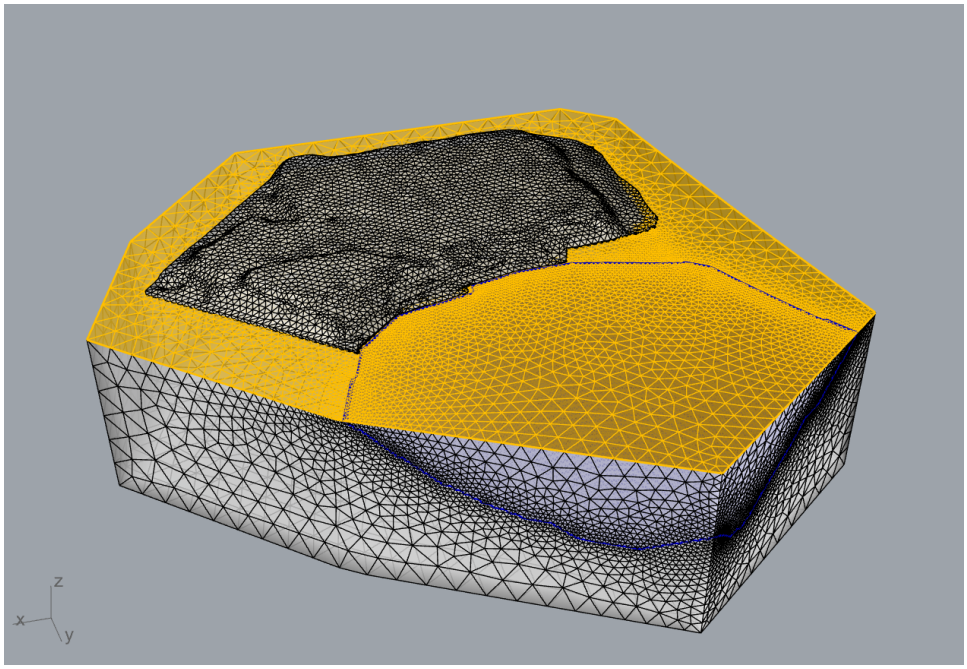


Figure 84: Intersected box, pit, pit dump, and topo meshes.

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.

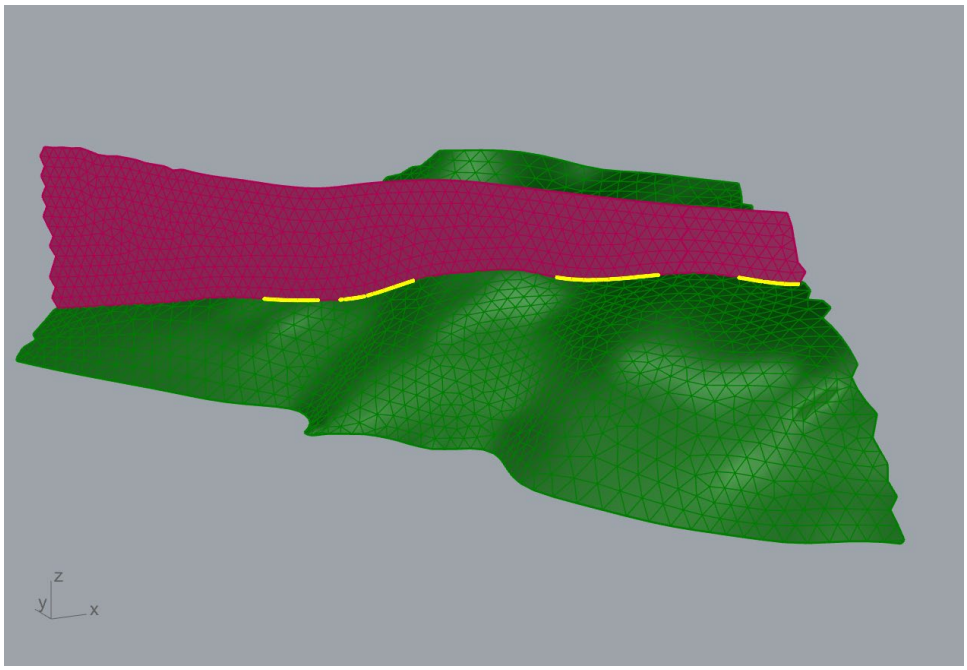


Figure 85: The _MeshIntersect command shows actual intersections (yellow lines) between "Fault4" and "Strat2" meshes. Other parts of the "Fault4" mesh do not intersect the "Strat2" mesh, leaving minor gaps.

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".

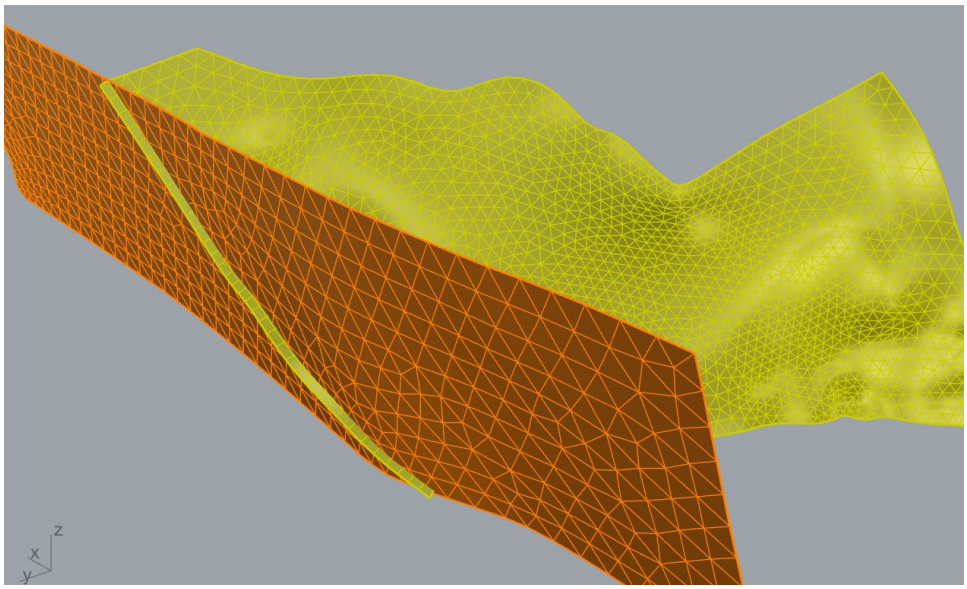


Figure 86: Extension of "Strat1" mesh along the part of the boundary closest to "Fault1".

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.

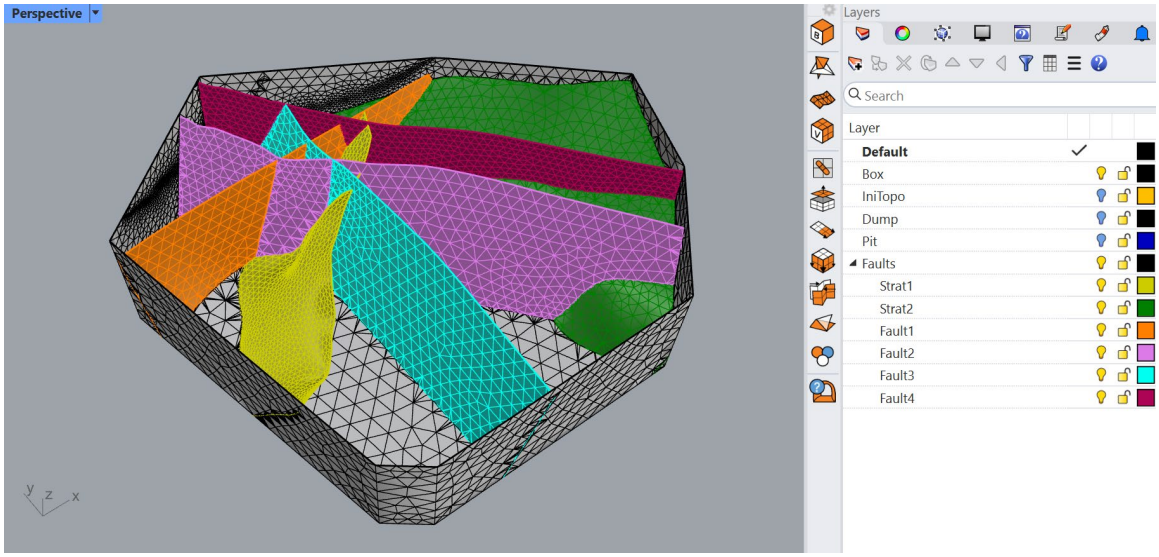


Figure 87: Intersected and trimmed "Faults" meshes.

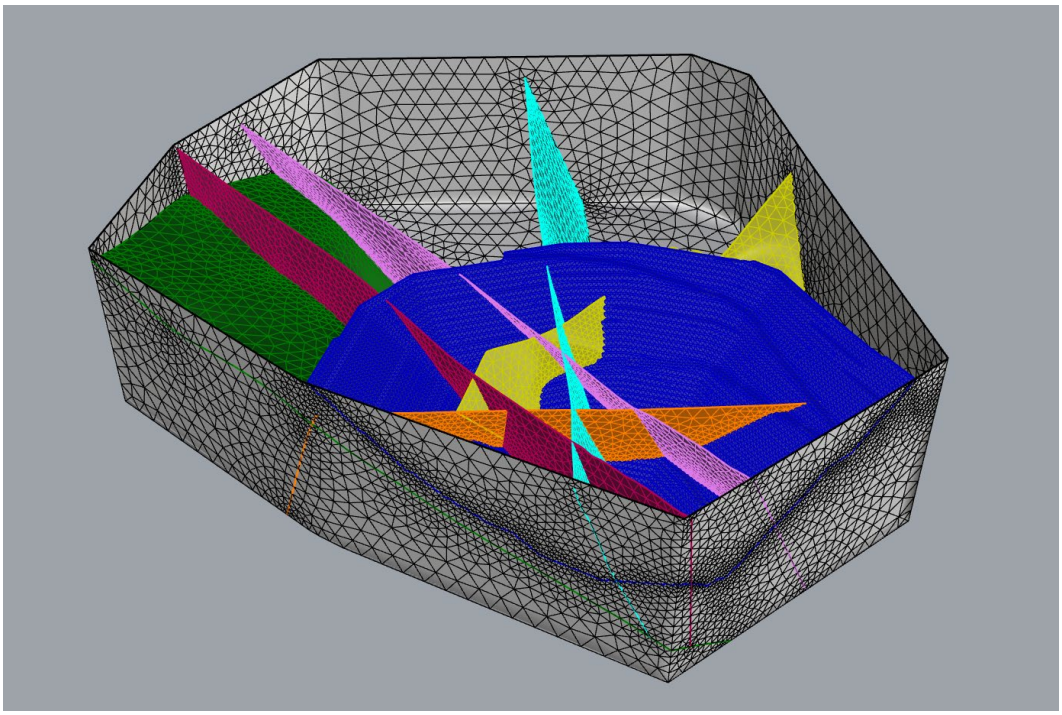


Figure 88: Final triangular surface meshes (the pit dump and topo meshes are not shown).

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.

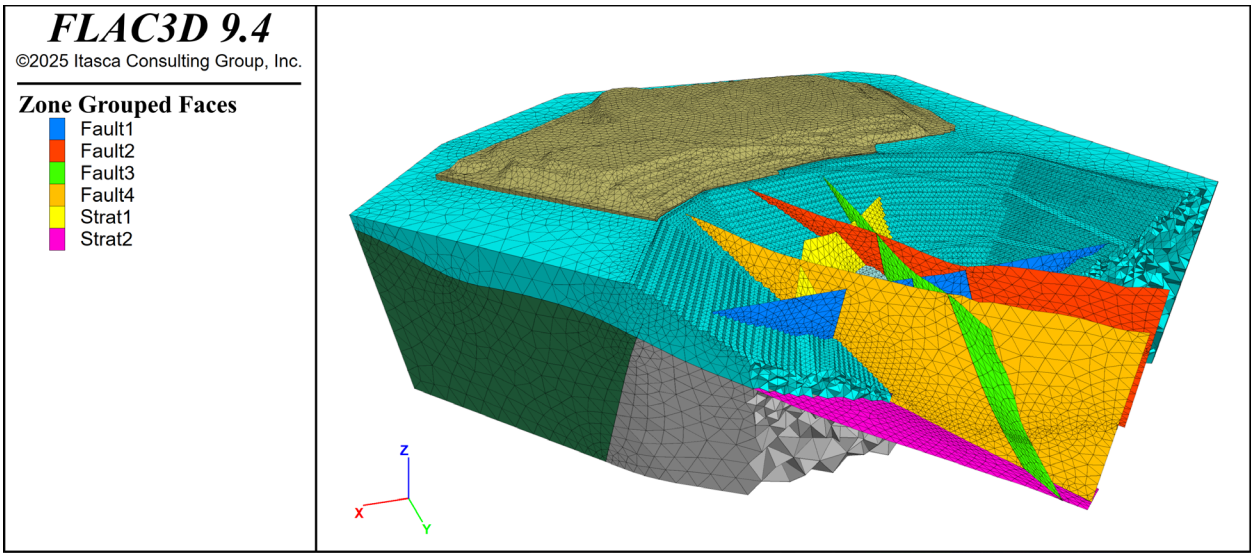


Figure 89: Tetrahedral volume mesh of open pit mine model.

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**).

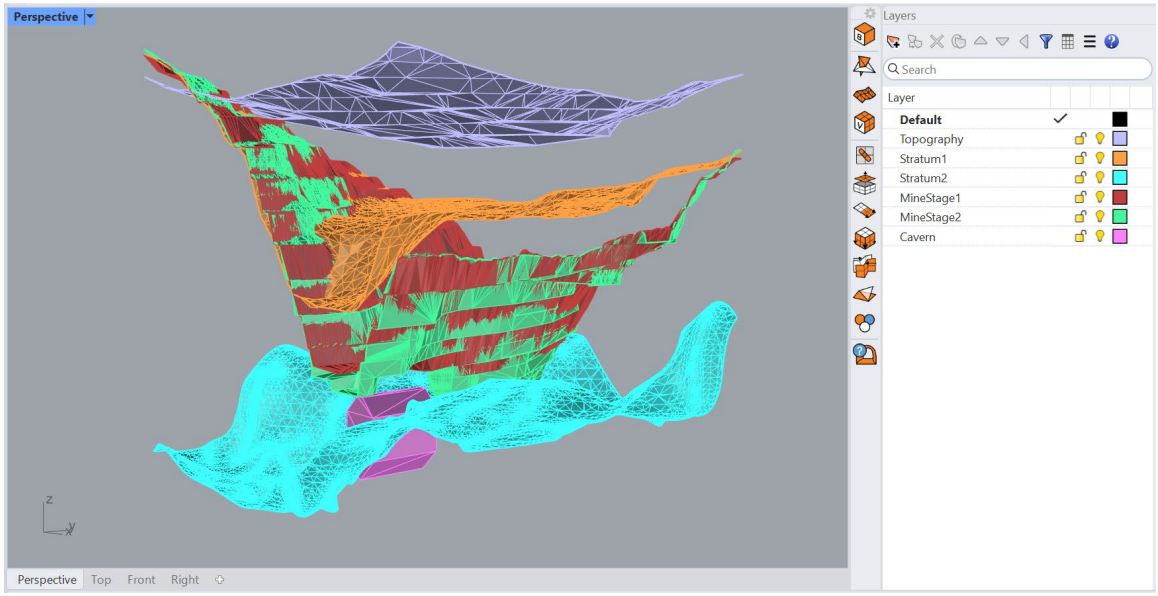


Figure 90: Initial model with various geological and mining geometries shown.

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:

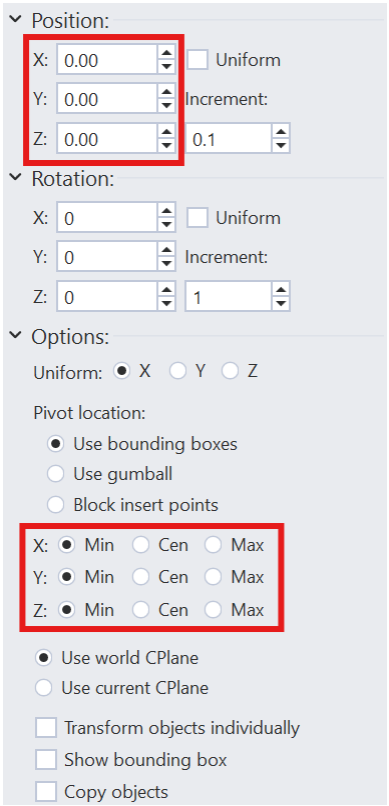


Figure 91: The Box Edit parameters used to translate the model.

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”.

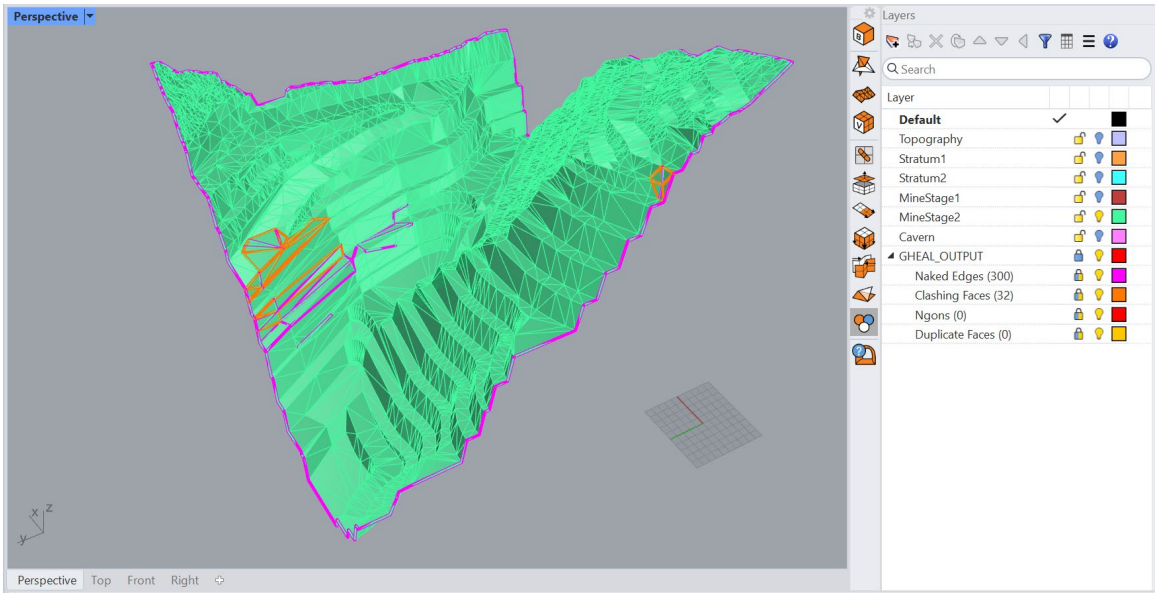


Figure 92: Naked edges and clashing revealed by GHeal in the “MineStage2” mesh.

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.

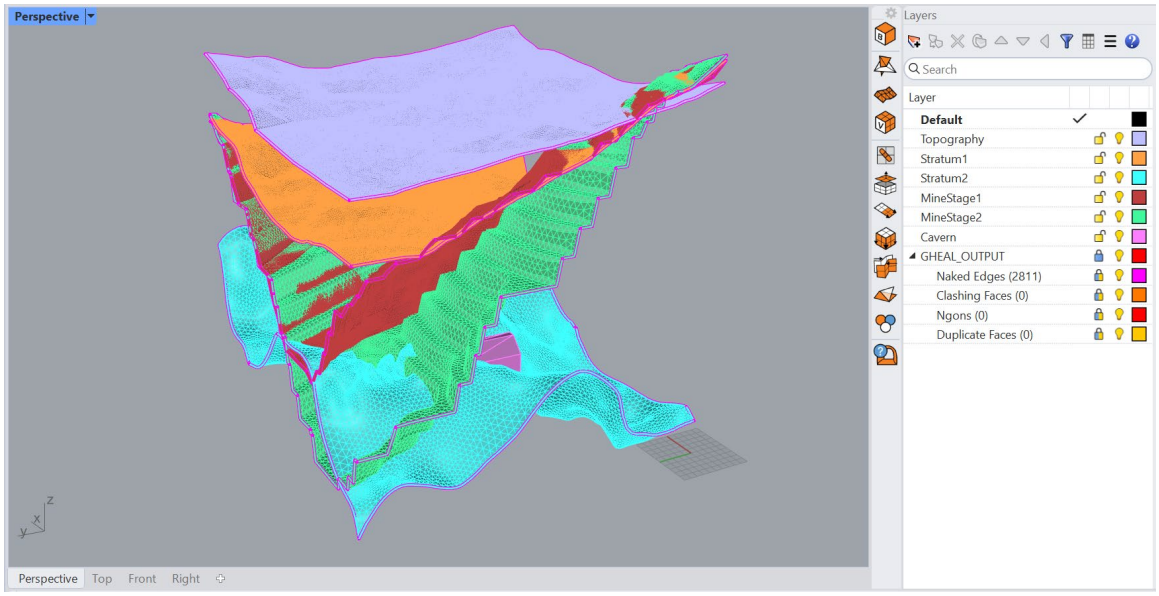


Figure 93: Cleaned-up and remeshed model meshes with GHeal output showing the presence of naked (real) boundaries only.

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.

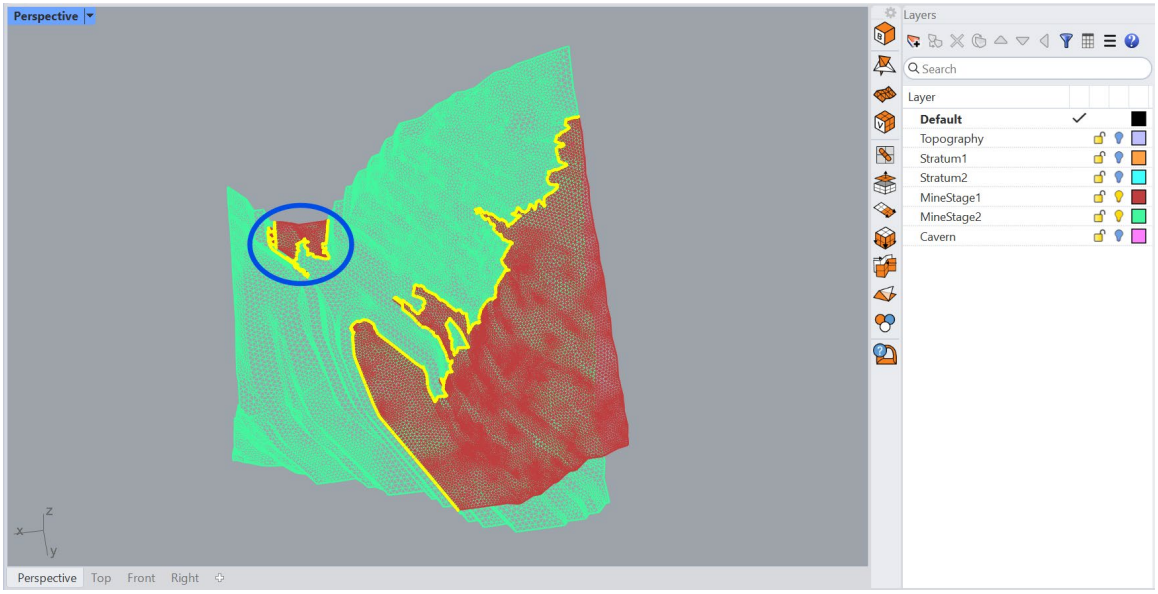


Figure 94: “MineStage2” mesh collapsed on “MineStage1” forming 3 distinct intersection curves (Wireframe view).

   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).

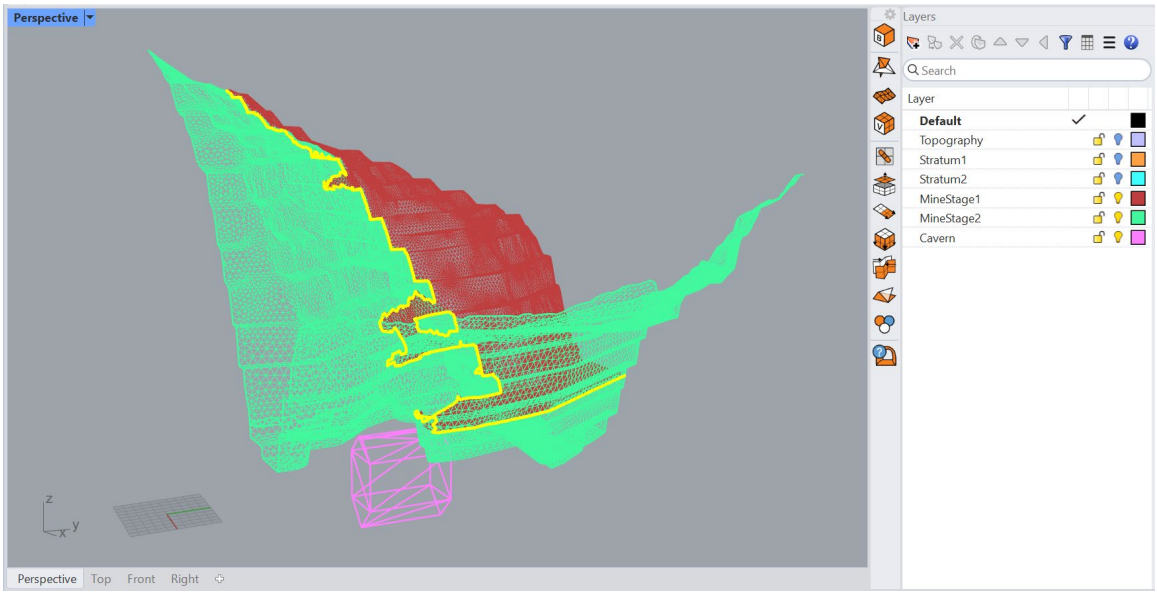


Figure 95: Collapsed and cleaned-up mining stage meshes with two complete intersections.

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.

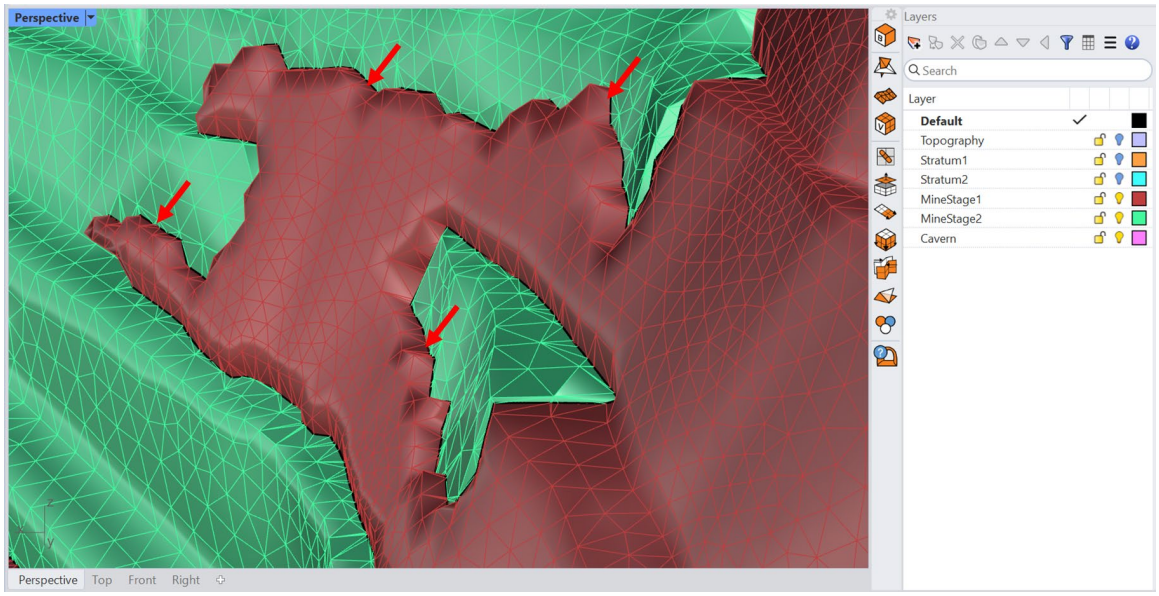


Figure 96: Zoomed-in view at the collapsed and extended faces.

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.

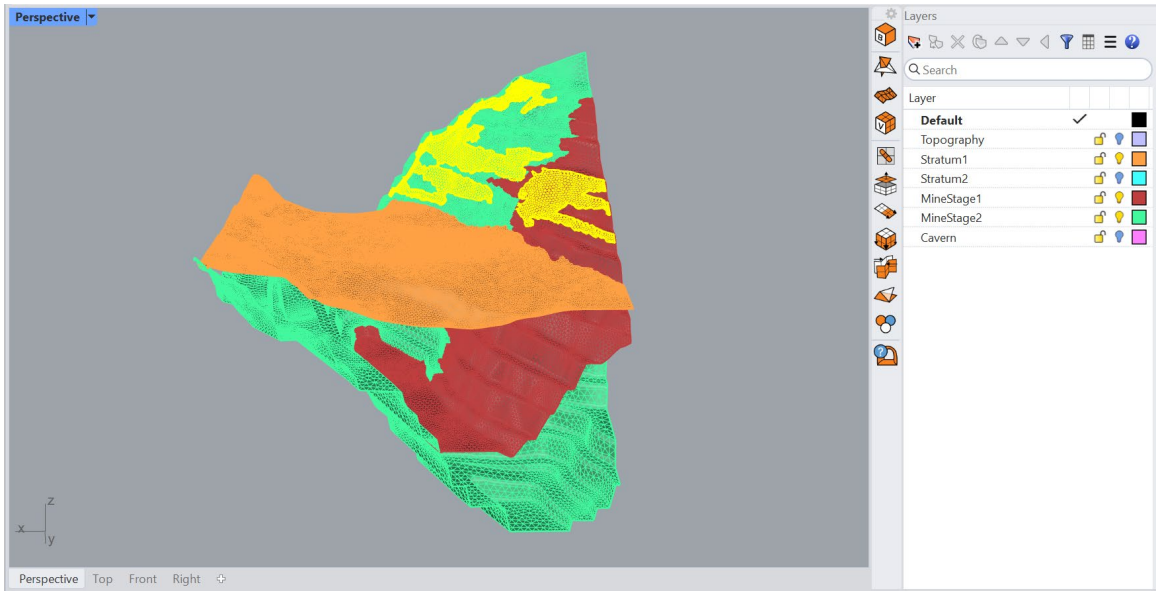


Figure 97: Result of "Stratum1" mesh collapse onto "MineStage1" and "MineStage2".

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.

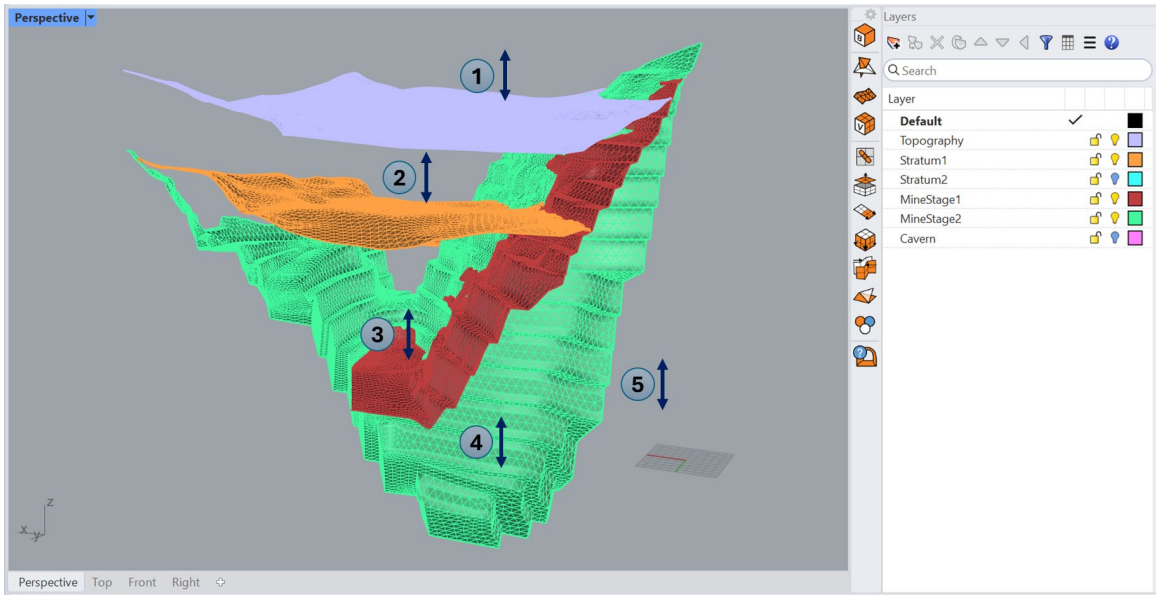


Figure 98: The model is split into 5 distinct volumetric regions as a result of GCollapse operations.

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.

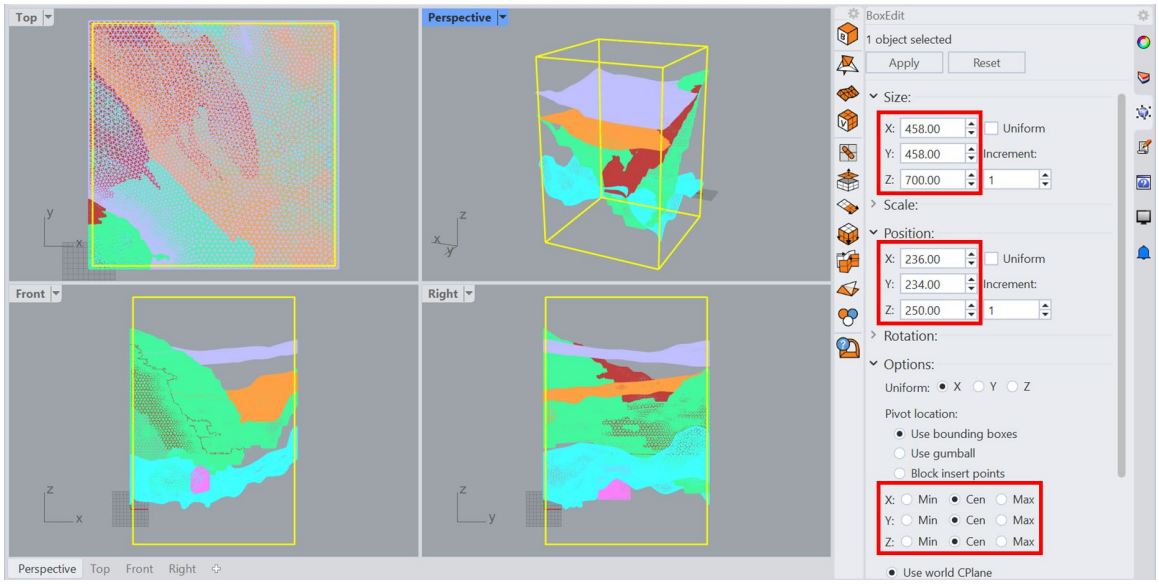


Figure 99: Model domain creation from a bounding box.

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.

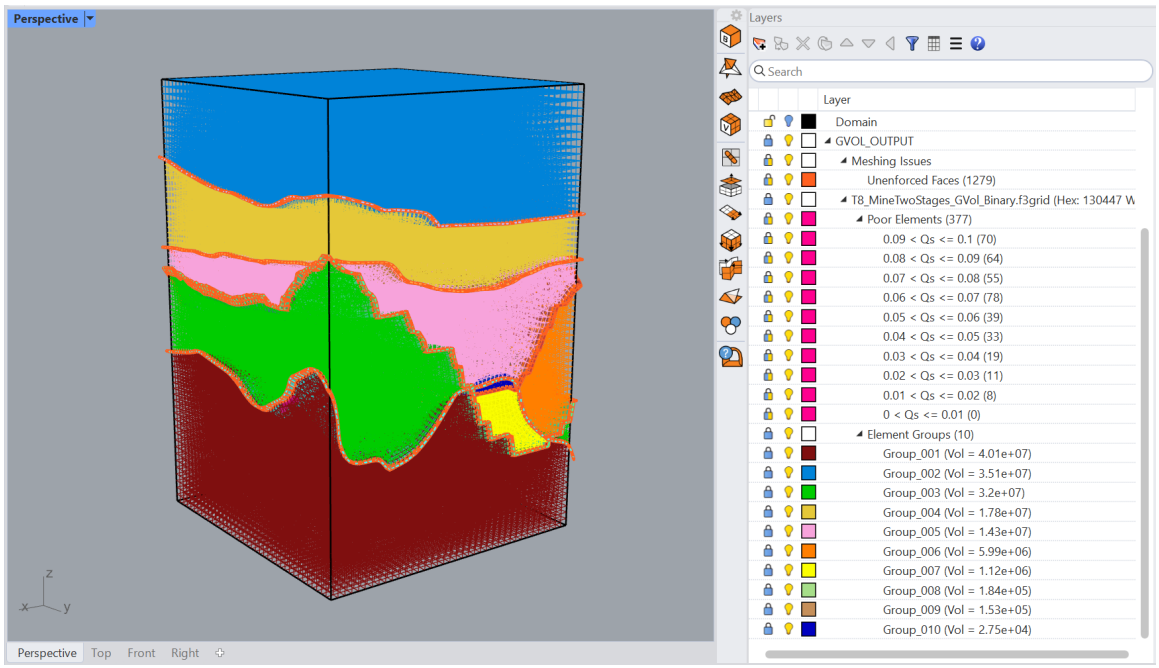


Figure 100: GVol generated mesh visualized in a *Rhino* viewport and informational output in the Layers panel. The model domain is shown with black edges.

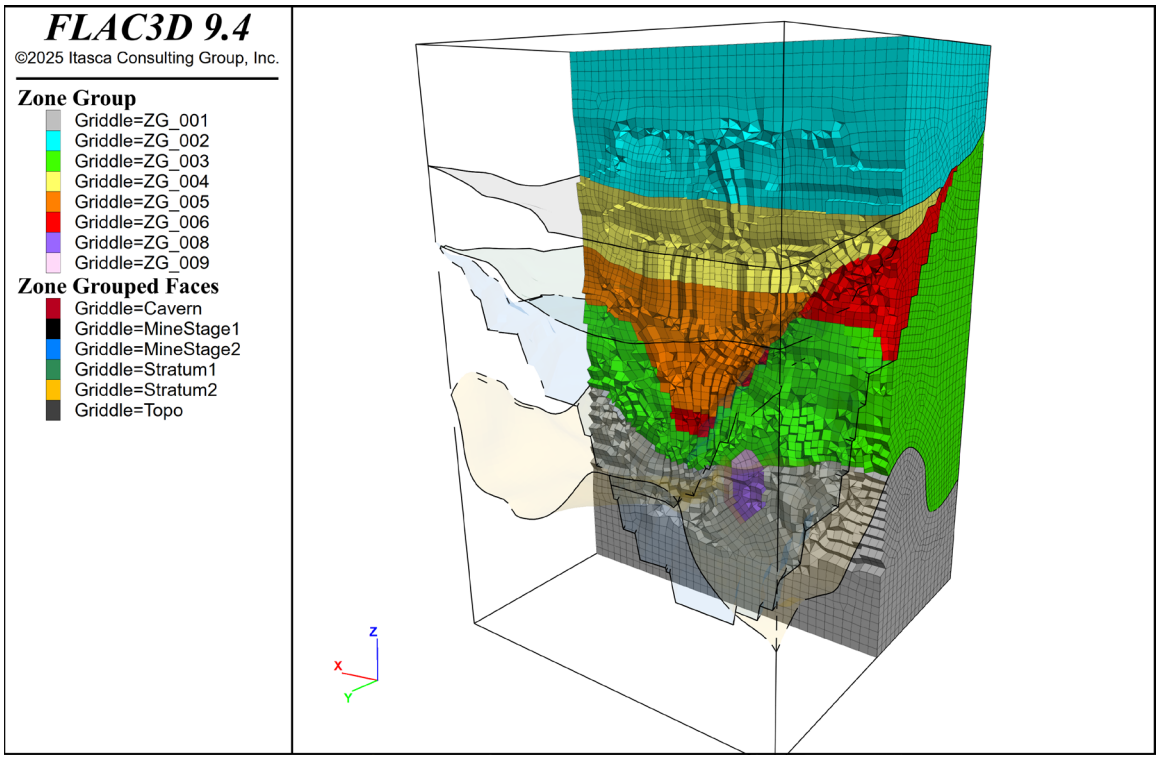


Figure 101: Cut through the model in *FLAC3D*.

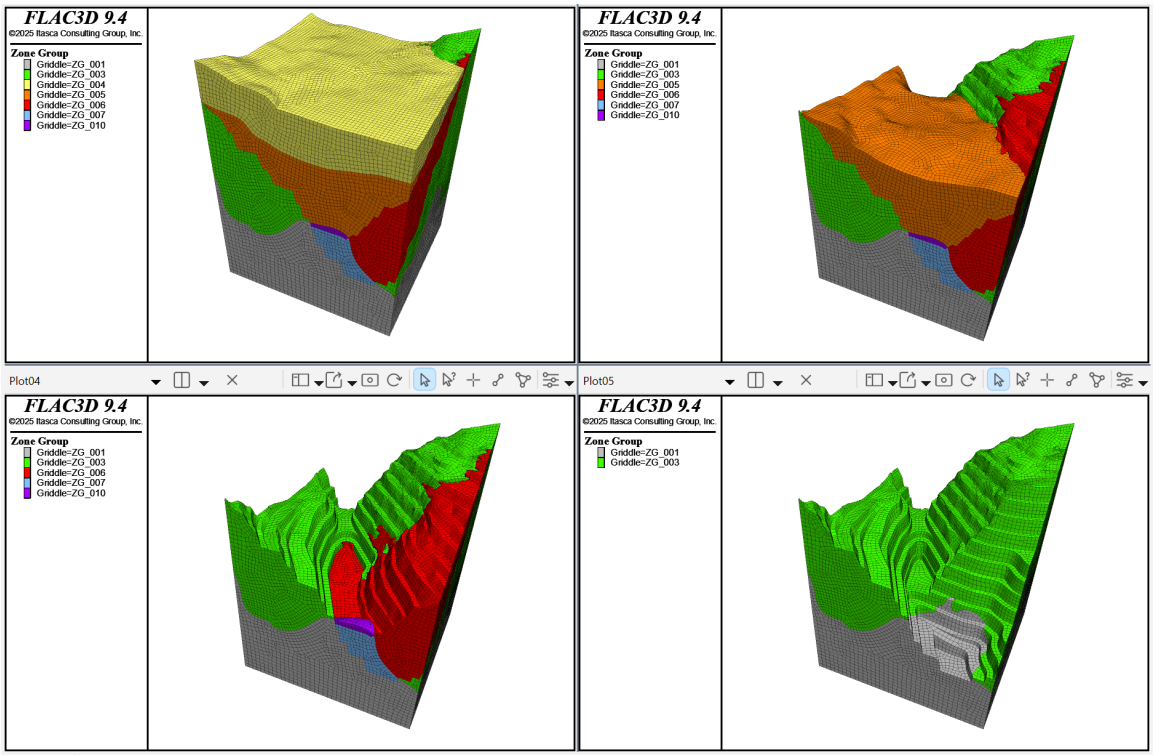


Figure 102: Different model excavation stages visible in *FLAC3D*.

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.

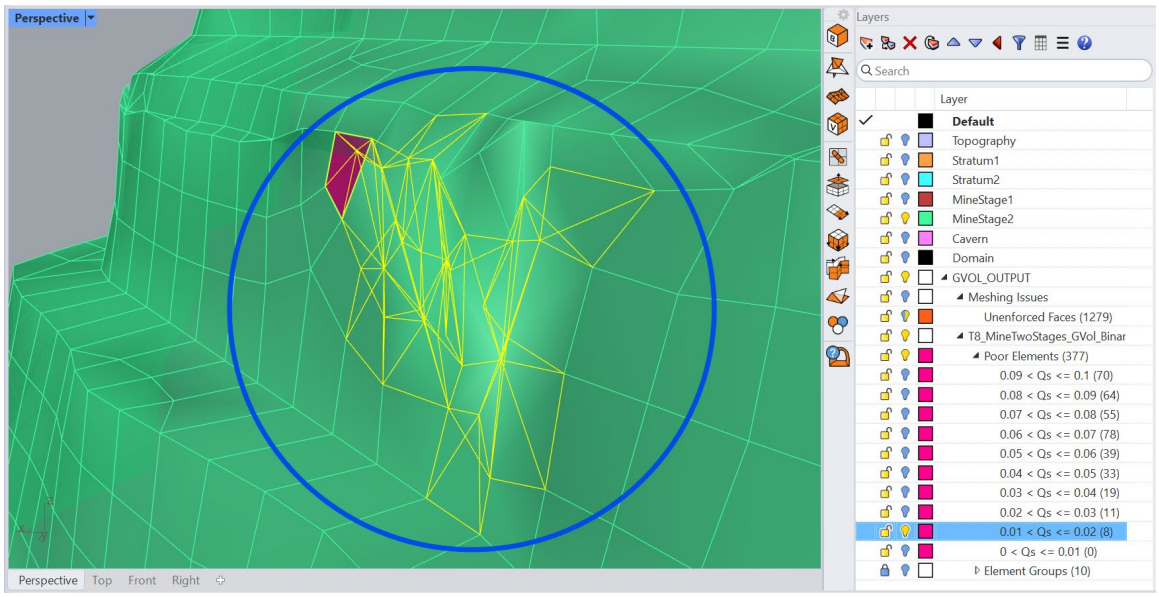


Figure 103: A big cluster of poor elements caused by the rough geometry of MineStage2 mesh.

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.

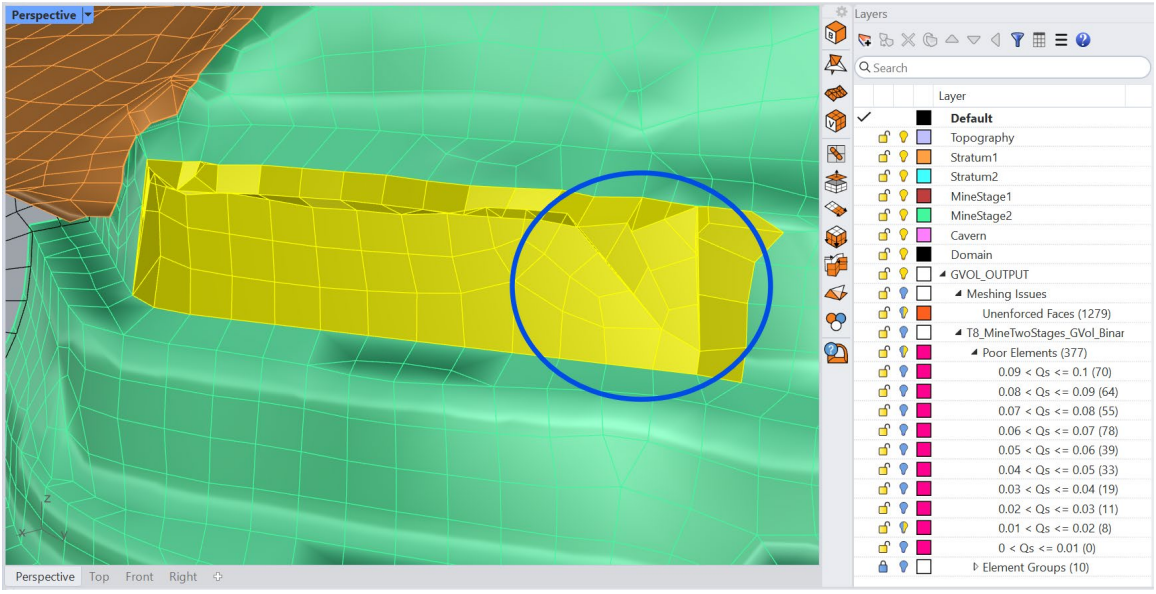


Figure 104: Poor surface mesh geometry causing bad-quality volume elements to be generated within the blue circle.

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.

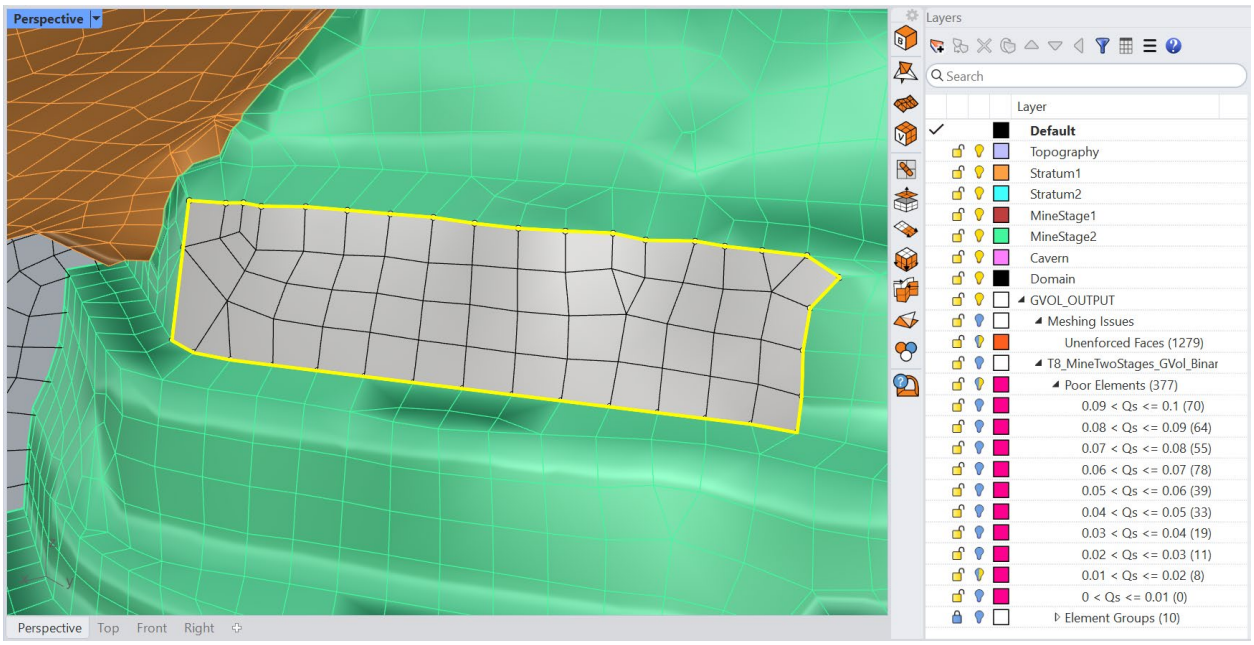


Figure 105: A rebuilt good-quality mesh patch before merging with the remaining mesh.

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.