
Computational Model Builder Manuals

Release 5.0.0-Develop

Kitware Inc.

October 06, 2017

1	Contents:	3
1.1	ModelBuilder Application Guide	3
2	Indices and tables	41



Computational Model Builder (CMB) is a suite of applications that provide a customizable workflow for the entire lifetime of a computational simulation. Generally speaking most workflows involve

- specifying the governing equations of state of the system to be simulated, their free parameters, and other quantities required by the simulation;
- importing, constructing, or fitting a geometric model of the physical domain of the simulation;
- discretizing the geometric model into a mesh suitable for approximating functions which enable people to make a decision based on the outcome of the simulation;
- associating material properties, boundary and initial conditions, and other attributes to regions of the mesh.
- queueing the simulation (or an ensemble of simulations) for execution;
- monitoring simulation progress and inspecting results after the simulation has run;
- integrating simulation results, experimental measurements, and other knowledge in order to make a decision about how to proceed.

The first application in this suite that we are introducing is ModelBuilder, which targets the first four steps above for several major use cases. [ParaView](#) is often used for the final steps.

Contents:

ModelBuilder Application Guide

ModelBuilder is an application built on [SMTK](#) to provide engineers and scientists with a simple way to prepare simulation input decks.

User's Manual

Getting ModelBuilder

Downloading ModelBuilder

ModelBuilder can be downloaded from the [Official Site](#).

Note: ModelBuilder v4 is currently in development, but nightly builds are available.

Building ModelBuilder from Source

Because ModelBuilder has a lot of dependencies such as [opencv](#), [paraview](#), [qt4](#), [smtk](#) and so on, we separate the code base into [CMB-SuperBuild](#), and [CMB](#), where the former contains everything involved and the latter only contains ModelBuilder itself. Either way the user needs to checkout the CMB-SuperBuild repository first. In CMB-SuperBuild, ModelBuilder can be built in two different modes: developer mode and release mode. If configured in the developer mode, all the dependencies of ModelBuilder will be built, and a the config file “cmb-Developer-Config.cmake” will be generated. The user should checkout the CMB repository outside the CMB-SuperBuild and build ModelBuilder itself. Doing so separates ModelBuilder and its dependencies and enables the user to rebuild ModelBuilder easily. If configured in the release mode, ModelBuilder itself as well as everything it needs will be built, there is no need to checkout CMB repository.

See also:

[Our gitlab](#) for more instructions.

What is Model Builder?

ModelBuilder is a platform designed to integrate every process involved in the whole life cycle of numerical simulation. It combines the individual preprocessing, solver, and post-processing softwares into one application which allows

the user to create or import geometry, generate or import mesh, assign simulation parameters to the model, generate the simulation input decks for the solver, as well as view and process the computational results.

Key Concepts

In order to show how ModelBuilder works, we need to clarify a few concepts:

Model A model refers to the geometrical representation of the simulation object. It can be either 2D or 3D. In ModelBuilder, model can be created or imported. The different origins of the models determines the different operations that can be applied to them. Common operations include Save, Export, Delete, Add Auxiliary geometry etc. See [The Model Tab](#) for details.

Session ModelBuilder is capable of handling models created by a variety of applications, each of them use a different way to represent their model, the interface between ModelBuilder and the application creates the model is called a session. In other words, sessions are used to handle models of different origins. For example, the polygon session is intrinsic to ModelBuilder, which is the only session that is capable to create a model interactively in ModelBuilder; the discrete session uses VTK data structures so that it can handle models in VTK. See [The Session Operators](#) for details.

Mesh Mesh is a key component of a lot of computational methods. It is consist of vertex, edge, face, and their connectivities. It can be imported to ModelBuilder, but unlike the model, mesh currently cannot be generated by ModelBuilder itself. However, the users can link their own meshers to ModelBuilder and then generate the mesh within the ModelBuilder's framework. Mesh can be deleted, exported, interpolated etc. See [The Mesh Tab](#) for details.

Attribute Each simulation package (e.g., [Albany](#), [OpenFOAM](#), [Deal II](#)) accepts different sets of parameters. ModelBuilder calls such a set of parameters an *attribute*. Attributes consist of child *items*, i.e., individual parameters. Besides, there are conditional parameters which may or may not exist depending on the values of other items. All the parameters form a tree of items that control the simulation.

ModelBuilder can be customized for any simulation package by writing:

1. an XML description of the parameters accepted by the simulation (called a *template*)
2. a Python script (called an *exporter*) that, given particular parameter values entered into ModelBuilder GUI, generates an input deck for the simulation

More often, you will use a pre-existing template and exporter. Some of them come with ModelBuilder and some of them may be provided by your organization. Refer to [SMTK's Attribute System](#) for details.

Work Flow

The work flow of ModelBuilder can be summarized into the following steps:

1. Model: As the first step, a geometry model must be created or imported to ModelBuilder. Further operations such as grouping entities, coloring entities etc. can be applied to the model.
2. Mesh: The next step is to generate or import a mesh as described above.
3. Attributes: The last step is to assign simulation parameters to the model and export, which further breaks down to:
 - a. Load the simulation template as discussed above, and define the simulation parameters through the ModelBuilder GUI.

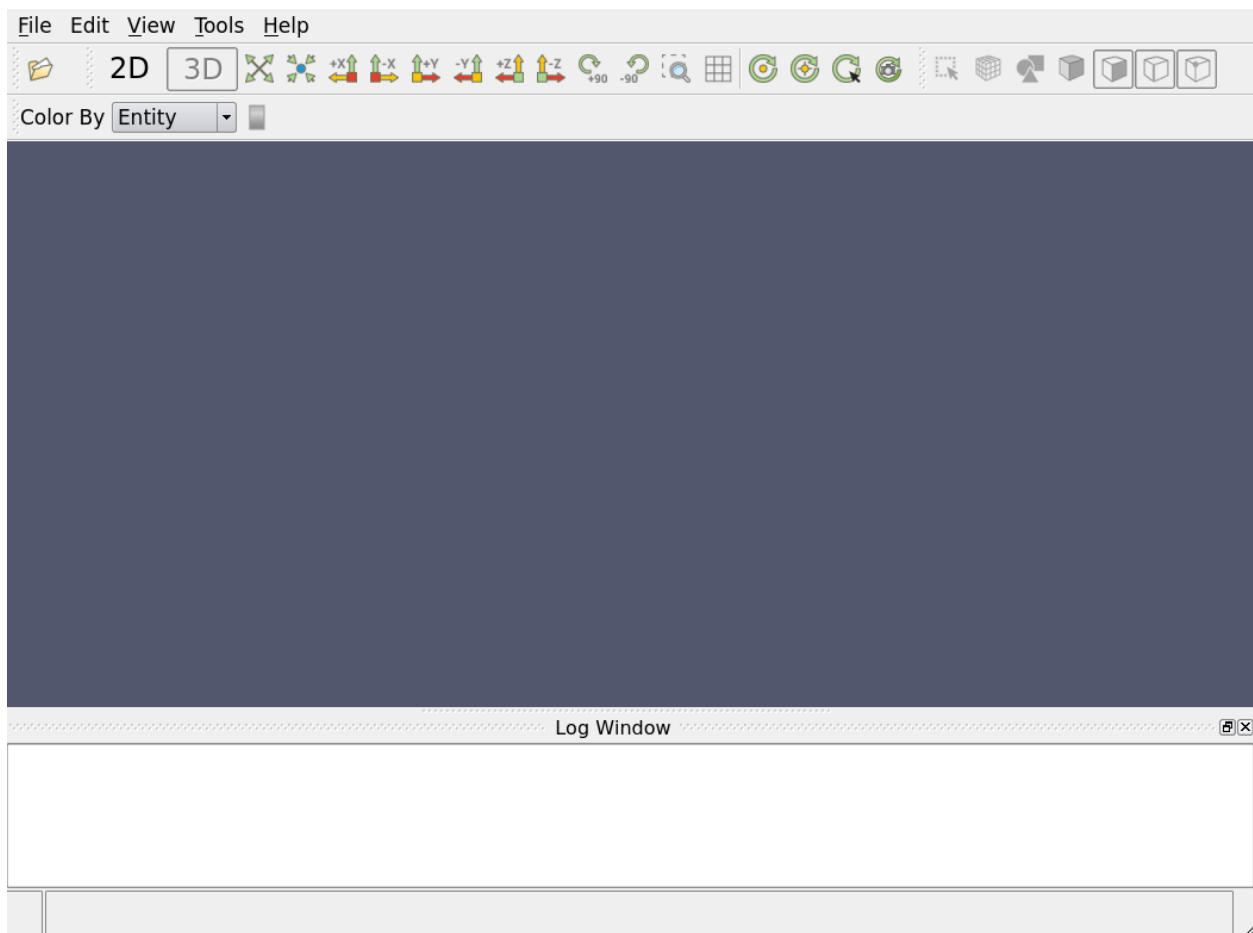
- b. Associate the parameters to the geometry regions in the model. For example, while modeling ground-water flow, once the different sediment layers are defined, they must be related, or *associated*, to a geometric portion of the earth being simulated.
- c. Export the input decks for the solver. Finally, once the simulation parameters have been specified and related to the geometry, ModelBuilder can be used to *export* an input deck for submission to a simulation.

Note that post-processing module has not been developed yet, so the work flow ends at exporting the simulation inputs.

Getting Started

This section aims to present the basic operations in ModelBuilder and familiarize the user with the logic of ModelBuilder using the following examples: `test2D.cmb` and `sample.crf`.

When opening ModelBuilder, the window will look similar to the figure below:

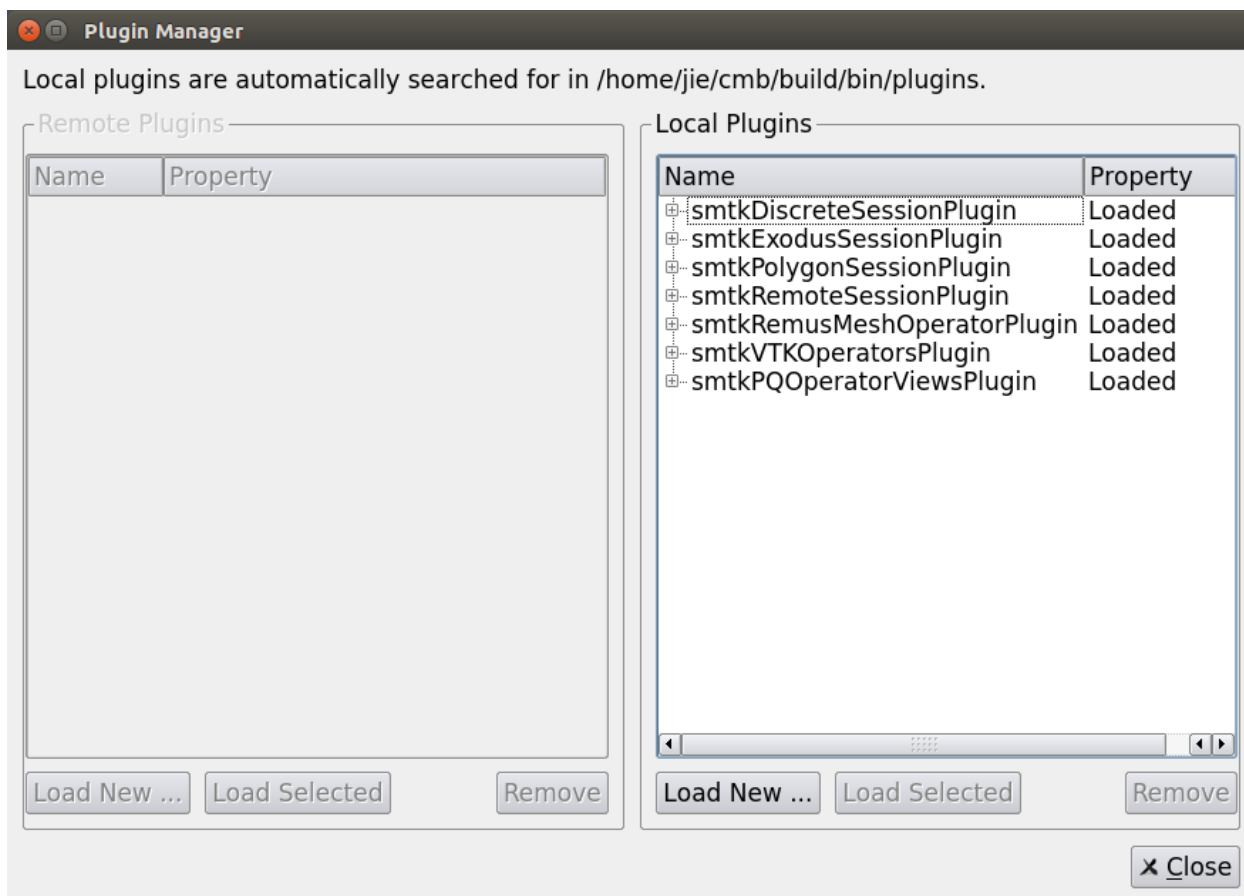
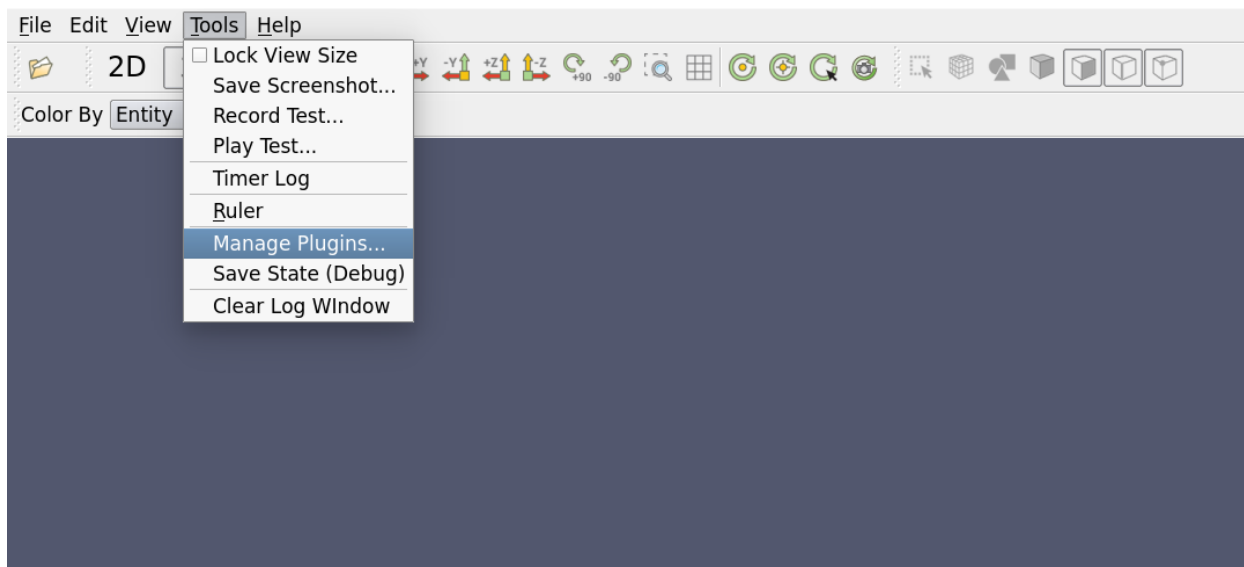


Managing Plugins

In order to load a model, a correct plugin must be loaded. Clicking Tools - Mange Plugins:

a plugin manager will be shown like this:

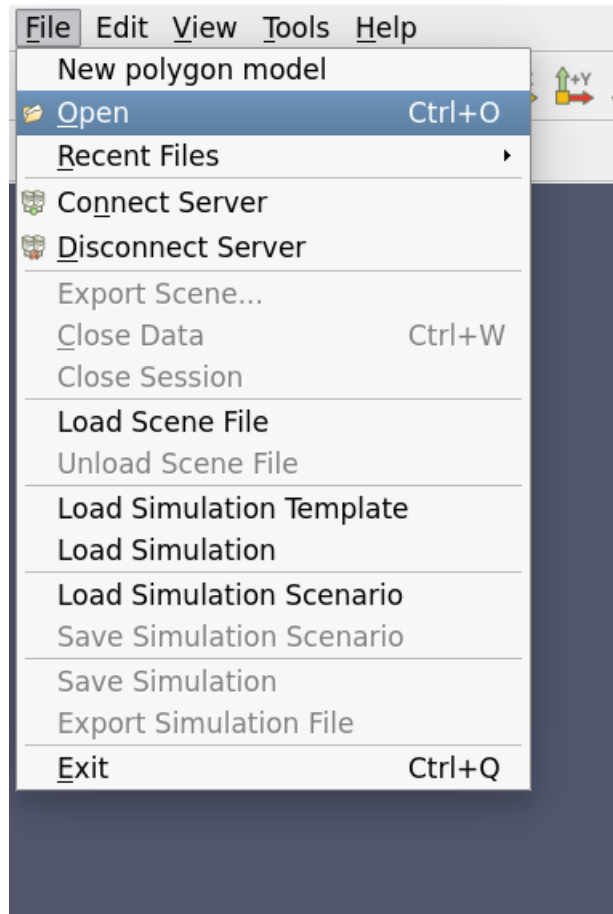
The plugins come with CMB are located in the “install/lib” directory of the cmb-superbuild. To load a new plugin, click “Load New...” and browse for a plugin. The plugins will show up in the “Local Plugins” list after being loaded




with a “Loaded” Property. In our case, the `smtkDiscreteSessionPlugin` needed by `test2D.cmb` has already been loaded by default.

Importing Model

Now that the correct plugin is loaded, we can import the model.



To import an object, click on the “Open”  icon in the File IO toolbar or go to File—Open (CTRL-O) in the main menubar.

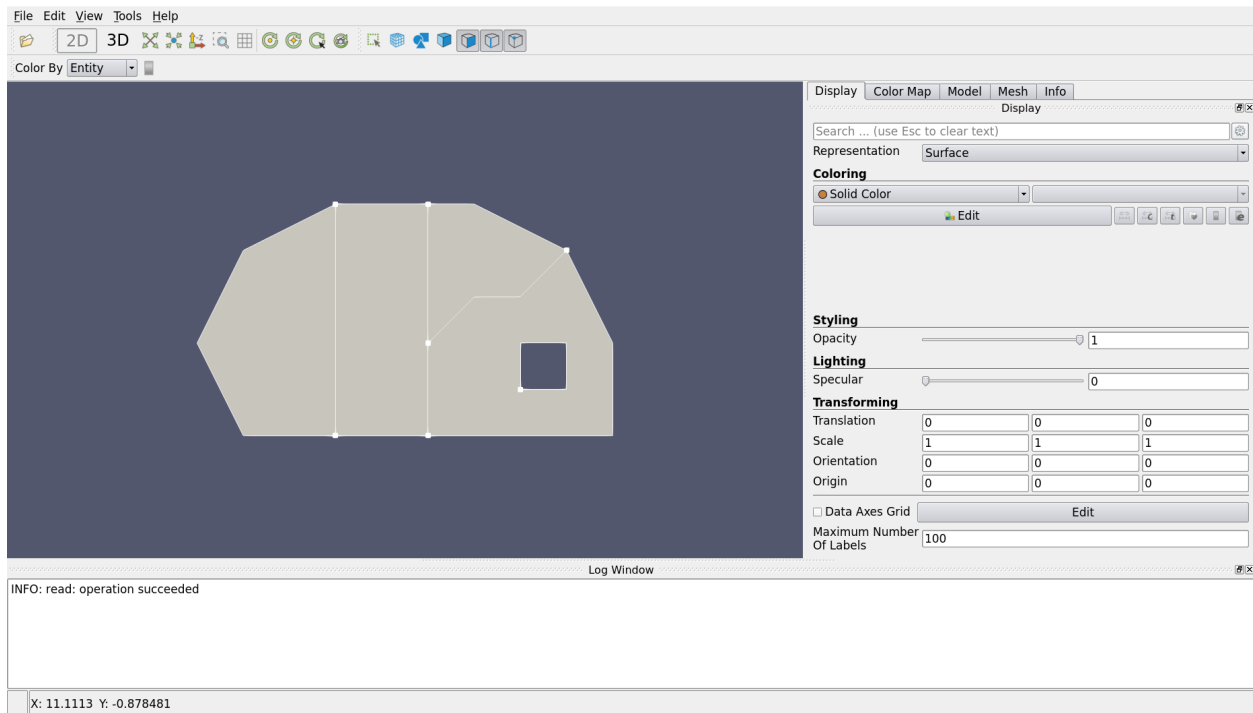
In case there are too many files in the directory, click on “Files of type” to select a certain type of file to make the target easier to find. Notice the file types shown here depend on the loaded plugins.

Browse and open the `test2D.cmb` file.

To unload all models currently in the scene, click “File—Close Data” (CTRL-W).

Window Layout

At this point, you should see a window like below:



The upper portion of the window is the tool bar which allows you to operate on the files and views, as well as the settings, tools and help.

The right middle portion of the window is menu tab which controls the model details including the geometry, mesh, visualization, and so on.

The left middle portion of the window shows the model. You can use the mouse to manipulate the camera view, and pick individual model entities.

The lower portion of the window is the log window, which displays the logs generated by the smtk server. It gives the user hints when something goes wrong.

Viewport Interaction

Camera Manipulation To rotate the current view, middle-click and drag on the viewport.

To pan the current view, left-click and drag on the viewport.

To zoom the current view, right-click and drag on the viewport (or use the scrollwheel).


Camera Adjustments For precise camera adjustments, click “View—Camera—Adjust Camera” to bring up the “Adjust Camera” window.

From this window exact numerical values of the camera properties can be entered.

Custom views can be configured here. Configurations can be loaded and saved here.

Model Interaction

There are two major methods to interact with the model entities. One way is to pick them in the entity tree under

“Model” tab; the other way is to use “Select Object”  in the viewport.

See also:


[Selection](#) for the various filters available in ModelBuilder to facilitate the viewport selection.

Right-clicking on a face in the viewport will select the face and bring up a context menu. From this menu, you can hide face, change color, and change representation (of the object).

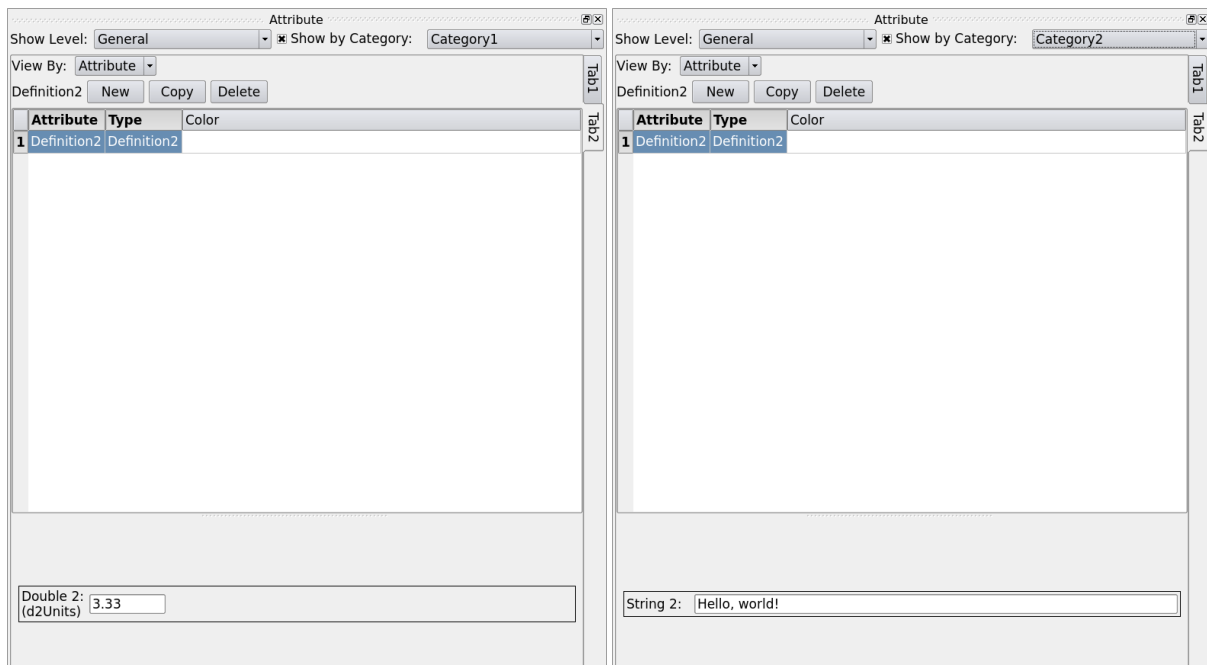
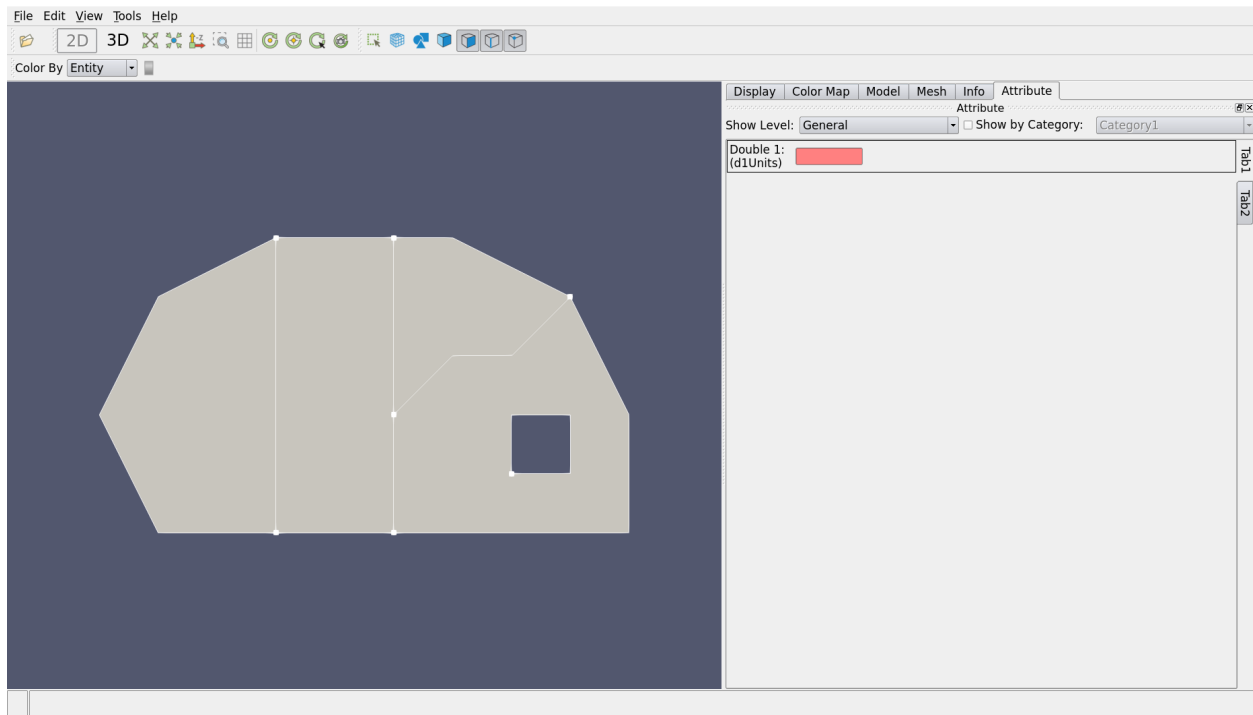


Attributes

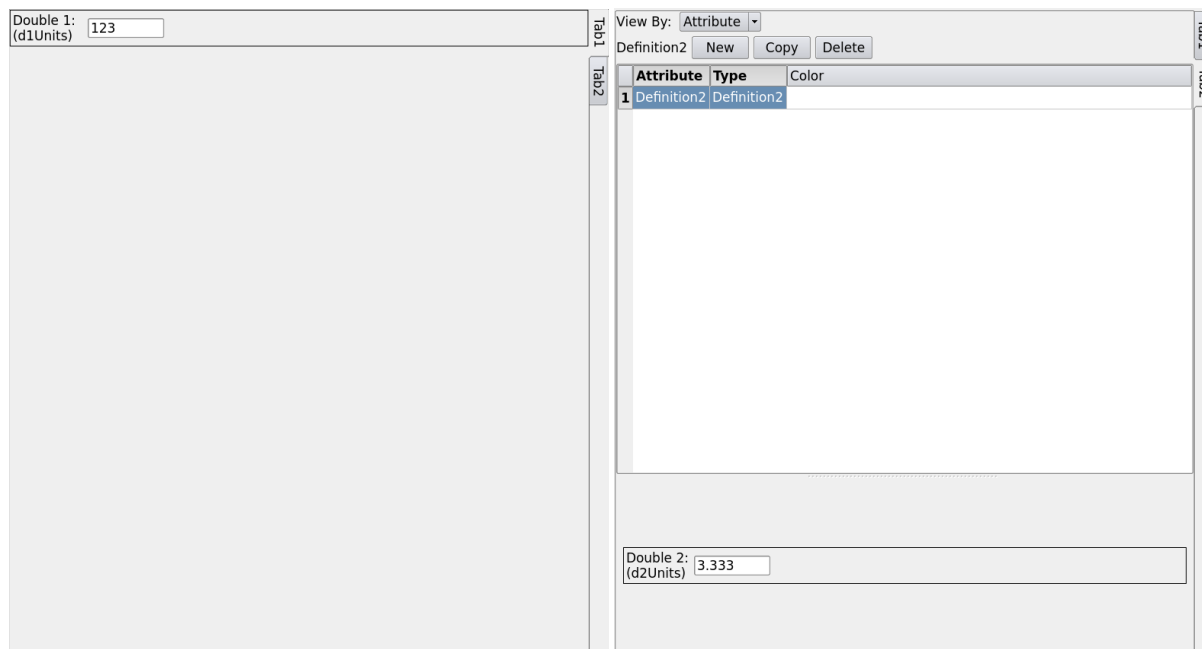


To load attributes, click the “Open”  button on the File IO toolbar and browse for the sample.crf linked above. Switch to the Attribute tab. Your program should look similar to the figure below.

The attribute view is customized by the template file, so different templates do not necessarily have the same content. “Show Level” allows the user to present the information at different levels: in this example “General” and “Advanced” are used. Attributes can also be grouped into categories so that they can be displayed by category when too many attributes are present. In this example, the input parameter in Tab 2 is a “Double 2” under Category 1 and becomes a “String 2” under Category 2.



Two vertical tabs (Tab 1 and Tab 2) are shown on the right-hand side as examples of the different sections of the simulation inputs, as shown below:



Both Tab 1 and Tab 2 are designed to input certain parameters. But Tab 2 has a table which allows you to insert multiple instances while Tab 1 does not. These are due to the different types used in their views (coded in sample.crf).

Select File-Save Simulation to save attributes as a CRF file. To export the attributes as simulation inputs, click on File-Export Simulation Files, the export interface designed in the template that contains two analysis will be shown:

Note: To export a simulation inputs deck, a Python script is required. For a user-created template, it has to be written by the user too. For the templates that come with ModelBuilder, the Python scripts are provided.


Finally, clicking File-Close Data or press CTRL-W to close the template after saving it.

Using Simulation Template

Since often times you would need to work on an existing simulation (“simulation” refers to the files including the model), change it a little bit, apply them to a specific model and export it again for another run, we focus on this process in this section.

Loading a Simulation File

Once a model has been loaded into ModelBuilder, the simulation file can be added. Click File-Open, Load Simulation

Template or Load Simulation to load a simulation file. The “Open”  button on the File IO tool bar can be used as well.

Editing a Simulation File

Let’s use `test2D.cmb` used in the last section and `AdHSurfaceWater.crf` as an example. Suppose we want to edit the “function” tab. Click on the “function” tab on the right-hand side of the “Attribute” tab, you will see a tabular of a poly-linear function $f(x)$ by default. Enter (3, 4) in the third row.

Simulation Export Dialog

Select Analysis Type

☒ Analysis1
☐ Analysis2

☐ Enable multiple selection

Show Level: **Advanced** ☐ Show by Category:

Output file (*.bc) **BC files** **Browse**

Python script **Browse**

Cancel **OK**

New **Copy** **Delete** **Load CSV** **Add** **Remove**

PolyLinearFunction-0

☐ Use Function Expression

f(X)=

Initial Value Delta Number of Values

	x	f(x)
1	0	0
2	0	0
3	3	4
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0

Constituents

Attribute

Show Level: General ☐ Show by Category: Constituent

View By: Attribute

SolidMaterial New Copy Delete

	Attribute	Type	Color
1	SolidMaterial-0	Solid Material	

Turbulent diffusion rate

Default turbulent diffusion rate

Individual turbulent diffusion rates

+ Add Sub Group

Kinematic eddy viscosity:

Exx

Eyy

Exy

Coriolis Latitude:

Constituents Friction BoundaryCondition

Next we define a material. Clicking on the “material” tab brings up a window as the figure below:

Clicking on “New” to create a new material, physical parameters can be further specified, we use the default values here.

Associating with Model

After creating the material, we want to associate geometry to it. There is a “Model Boundary and Attribute Associations” tab in the material tab. The faces are categorized into “Current” and “Available” groups where the former means the ones associated with the current material, and the latter indicates that they are not associated to any material. The buttons in the middle moves the entities left and right.

The window titled "Model Boundary and Attribute Associations" contains two main sections: "Current" and "Available".

- Current:** A list box containing "Face1 - Model A" and "Face3 - Model A".
- Available:** A list box containing "Face2" and "Face4".
- Buttons:** Three buttons are located between the two list boxes:
 - "==" (Move from Available to Current)
 - "<==" (Move from Current to Available)
 - "<==>" (Move both ways)

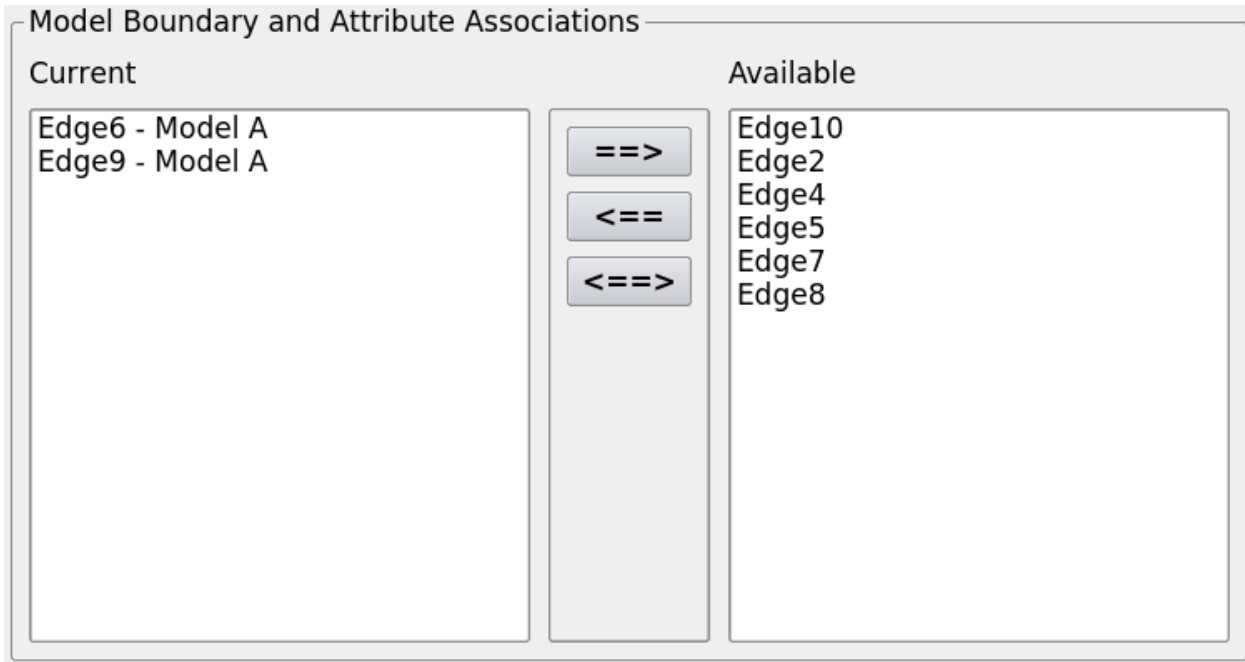
We can also assign boundary conditions to the model. Click on the “BoundaryConditions” tab, a variety of boundary conditions can be specified in the dropdown menu such as velocity, lid elevation, water depth lid and so on. In this case we create a “Total Discharge Boundary Condition” and a “Unit Flow Boundary Condition” individually.

The window shows a dropdown menu set to "Velocity BC" with "New", "Copy", and "Delete" buttons. Below is a table of boundary conditions:

	Attribute	Type	Color
1	UnitFlow-0	Unit Flow BC	
2	TotalDischarge-0	Total Discharge BC	

At the bottom, there is a "Unit flow:" label, a small icon, and a text input field containing the value "2".

Each boundary condition requires a numeric value to be specified. Similar to the material, edges can be associated to those boundary conditions we just created.



Saving a Simulation File

To save a simulation file to load in the future, go to the File menu and select “Save Simulation”. Now you are safe to exit ModelBuilder. Next time the model and simulation are opened, everything we modified will be loaded.

Note: The save process will save both the template and the values into one file. If you are expecting to configure multiple simulations from the same template, do not overwrite the original template file.

Exporting a Simulation File

Once a simulation and its attributes have been finalized, you can export by going to the File menu and selecting “Export a Simulation File”.

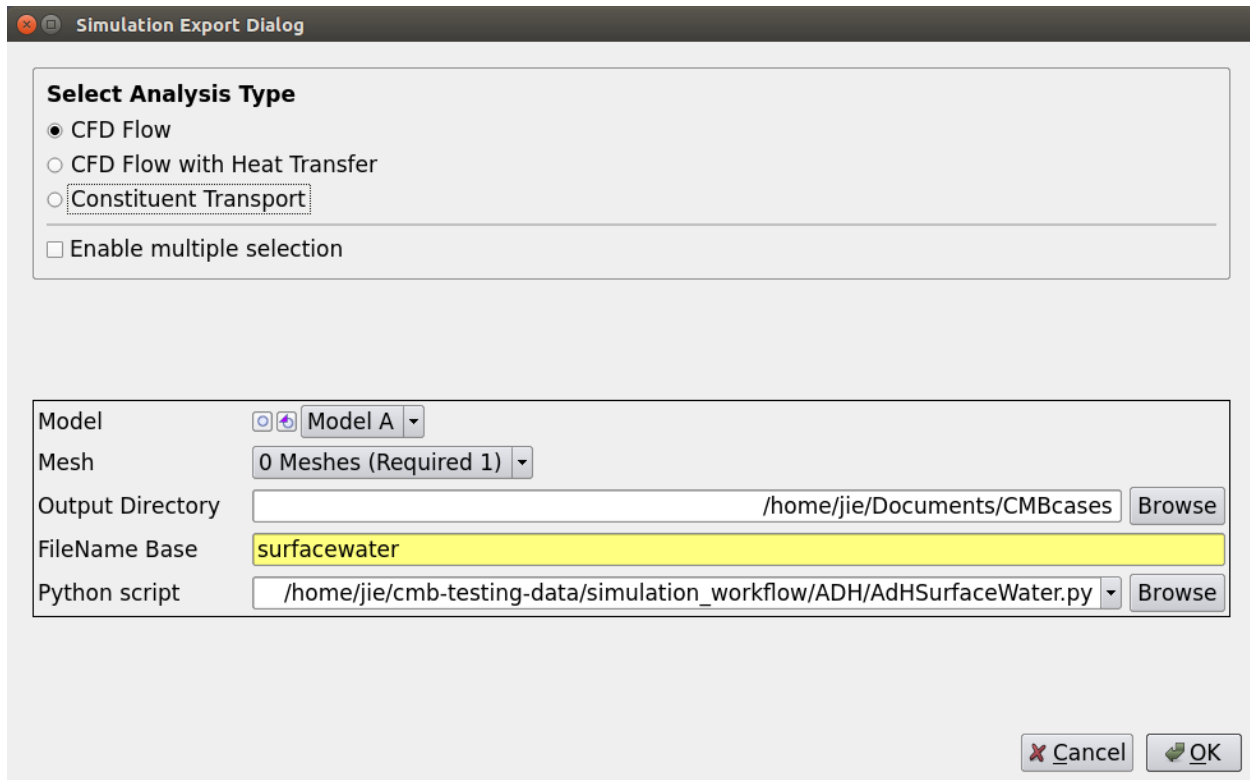
Multiple analysis can be selected for exporting. In this example we did not mesh the model so ModelBuilder will use the underlying vtk mesh as the mesh. Output directory, filename and [Python exporter](#) should be specified.

Click “OK”, you can see “surfacewater.2dm” and “surfacewater.bc” are generated in the output directory. The first file contains the geometry information of the model and the second one contains the boundary conditions. They are ready to be used in ADH solver.

Session Operators

Polygon Session

We begin our introduction to sessions with the simplest one - the *polygon session*. The *polygon session* is backed by Boost.polygon which is only available in 2D. It is named “polygon session” because the geometry is limited to combinations of polygons. This is currently the only session that allows the user to create a model in ModelBuilder instead of importing it (File - New polygon model). Next we introduce some features of this session.



Note: The features of introduced do not have to be exclusive. Some of them are shared by multiple sessions.

Model Operations A model will be automatically created when the *polygon session* starts. Right-click on the entity tree to create, close and save a model. Multiple models can be grouped by clicking Model - Create Group. Right-click on the model in the entity tree, select Model - Export edges to VTK, a window similar to this figure will pop out:

Select the target model, fill out the filename and hit apply, the edges will be exported to VTK.

Auxiliary geometry can be added to a model if necessary. Auxiliary geometry is a file that contains additional information which can be visualized on the model (for instance temperature data), or simply used as a background. A variety of data formats can be used as auxiliary geometry including .tif, .tiff, .dem, .vti, .vtp, .vtu, .vtm, .obj, .pts, .xyz. “Add an image” is a similar option to load auxiliary geometry except that it only accepts .tif, .tiff, .dem and .vti files. We will talk more about adding auxiliary geometry in [The Discrete Session](#)

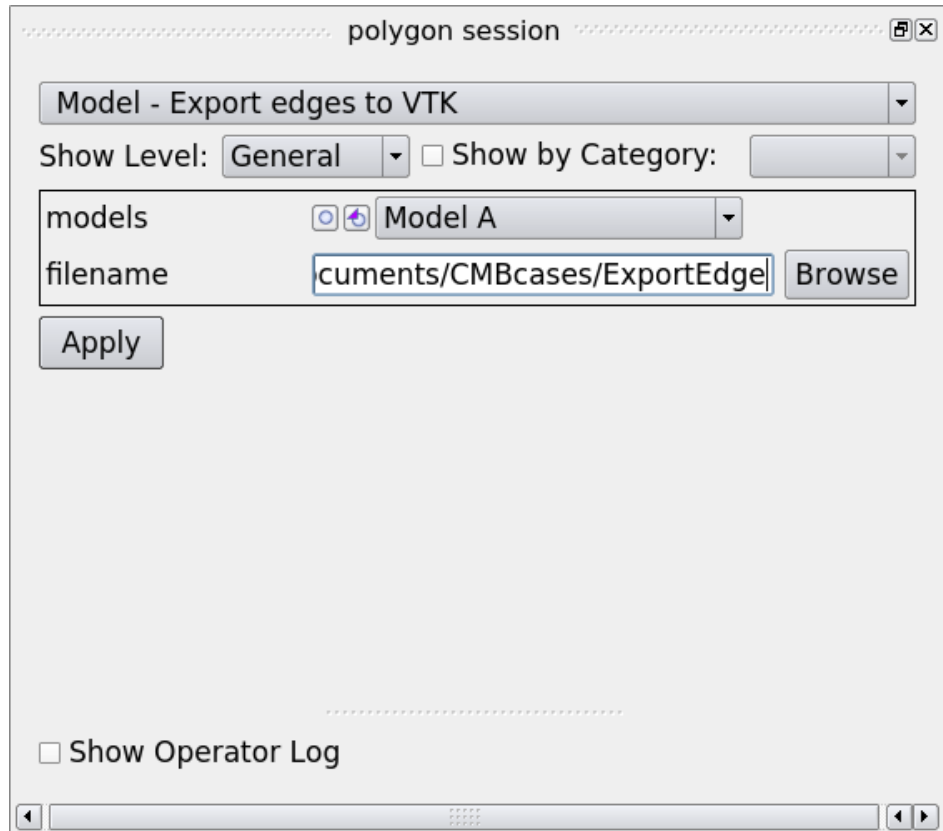
Entities can be deleted by clicking Model Entities - Delete. Sometimes it is convenient to color the model entities so that they look more distinct to each other. This can be done with [Model Entities - Assign Colors](#).

Vertex Operations Vertices can be added using exact coordinates through “Vertex - Create” and removed through “Vertex - Demote”.

Edge Operations Edges can be created interactively:

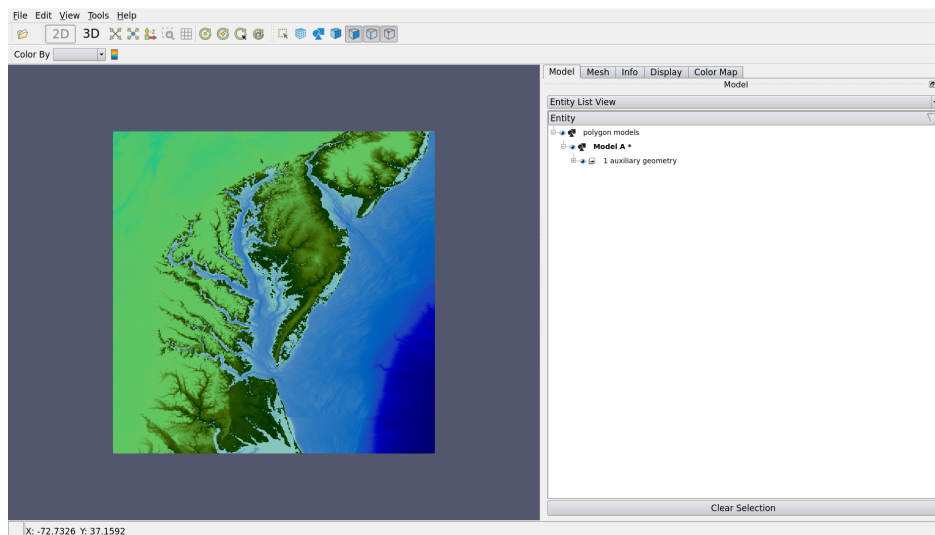
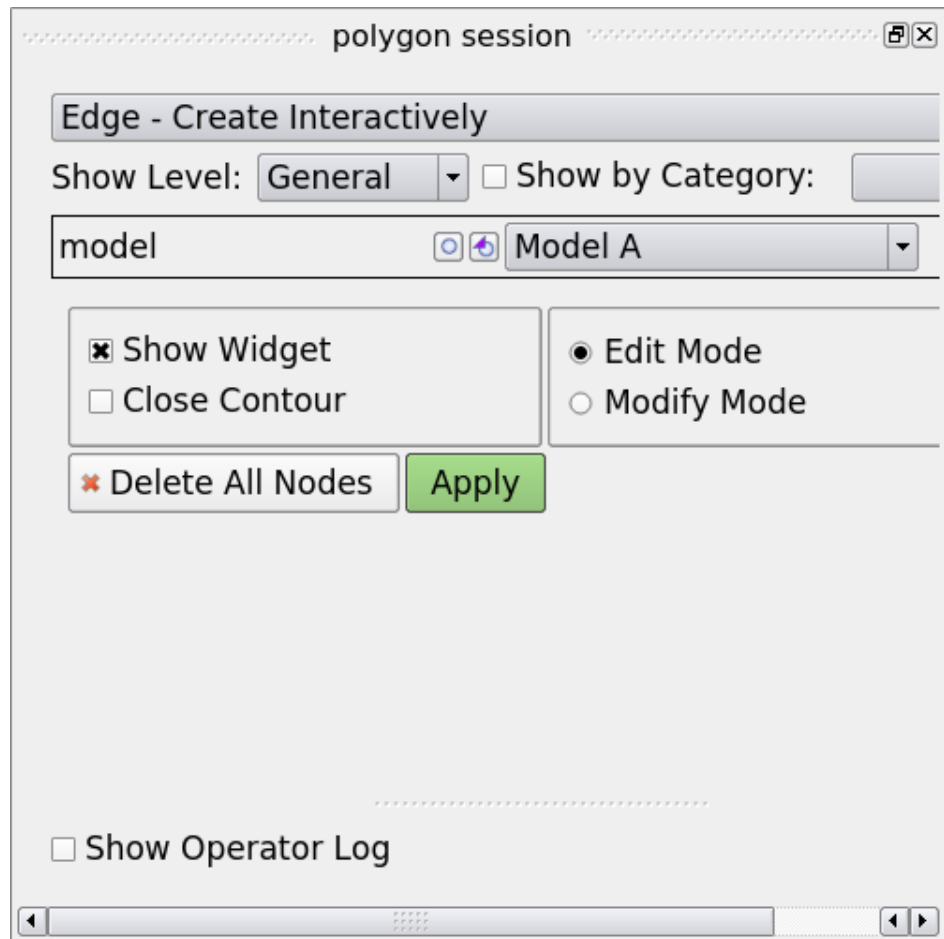
Click on Edge - Create Interactively, select the model you want to create edges for in the polygon session window:

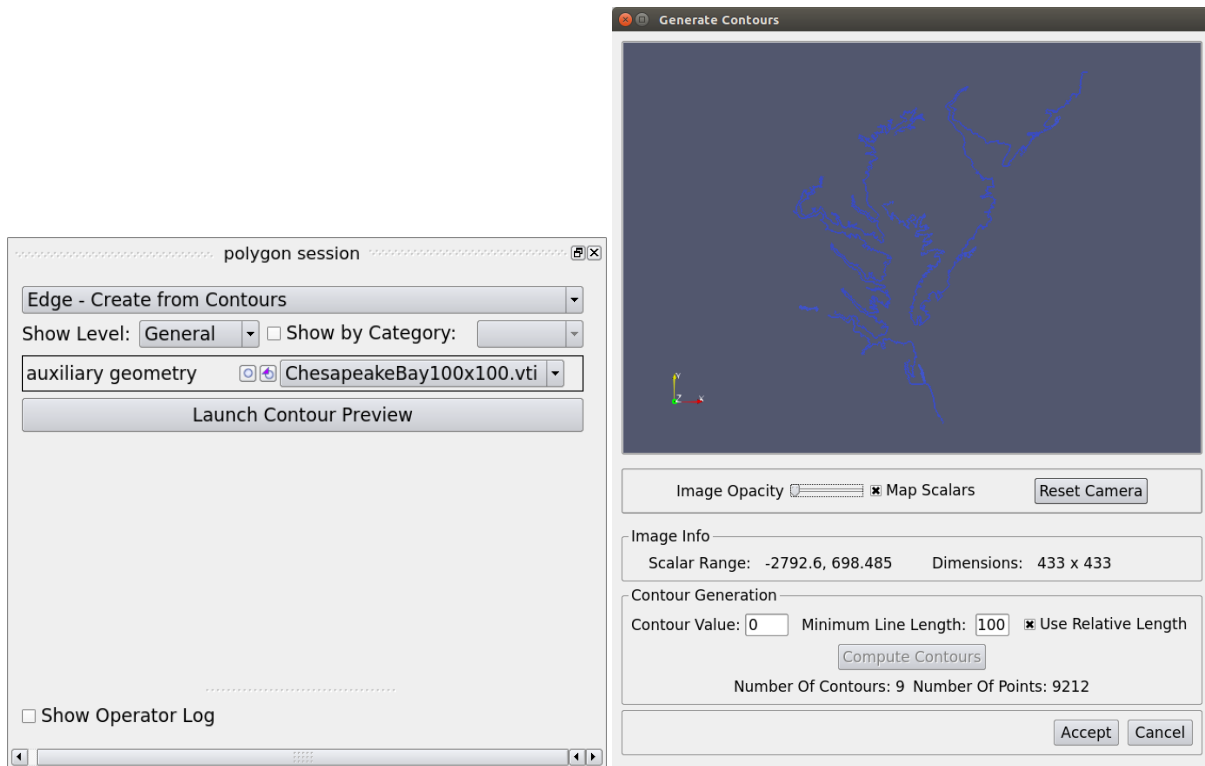
Next you can left-click in the viewport to create a vertex. A poly-linear line will be generated following the vertices created. You can click on “Close Contour” to connect the first and last vertices. Hit “Apply” to commit the changes.



If you want to create an edge with exact coordinates, use Edge - Create from Points. A tabular will be available to enter point coordinates. If vertices are already created, you can form edges through them by using Edge - Create from Vertices. If we have auxiliary geometry imported, we can even create edge from the contours. To show this, we first load an auxiliary geometry through Model - Add Auxiliary Geometry. See [The Discrete Session](#) for more instructions. Here is a figure of an auxiliary geometry:

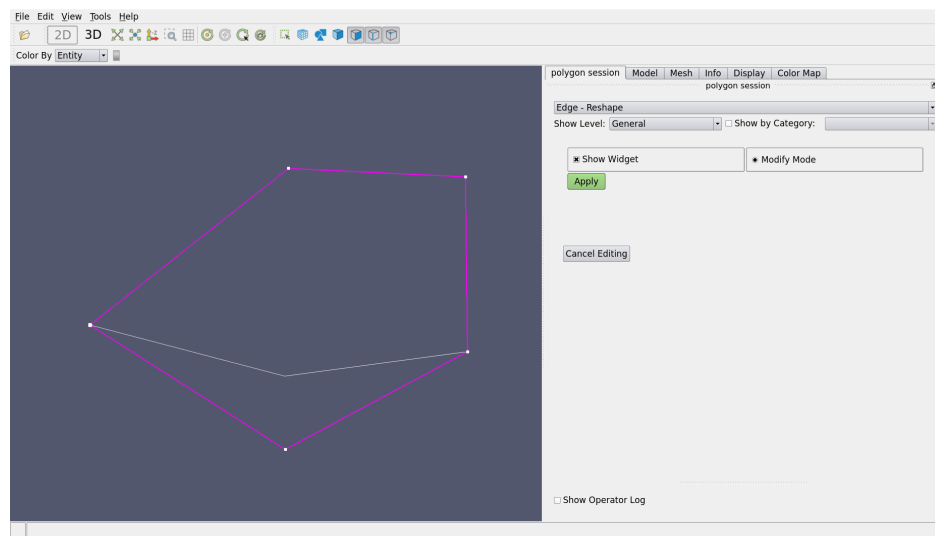
Now, right-click on the model and select Edge - Create From Contours, in the following window specify the auxiliary geometry, and click "Launch Contour Preview". A contour generator will appear, which lets you to specify the opacity, contour value and minimum line length.



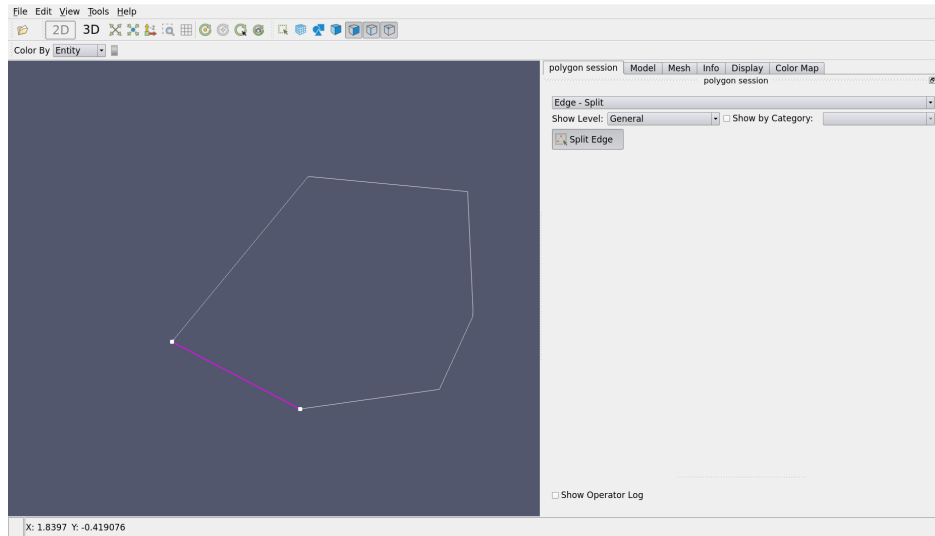


It is a good idea to turn down the image opacity in the preview, so that you can see the contour more clearly. The “Contour Value” is the data value that you want to extract at; the generated line segments that is shorter than the “Minimum Line Length” will be trimmed off. Once the contours are computed, you can find the number of connected contours and number of points on those contours in the same window. Click “Accept” if you are satisfied with the result. You will see edges and vertices created in your polygon model.

An edge can be reshaped using “Edge - Reshape”. To do so, right-click on the model tree and select “Edge - Reshape”. Then click on “select edge to edit” and box-select an edge, then drag the vertices on the screen or type in the coordinates in “Advanced” mode.

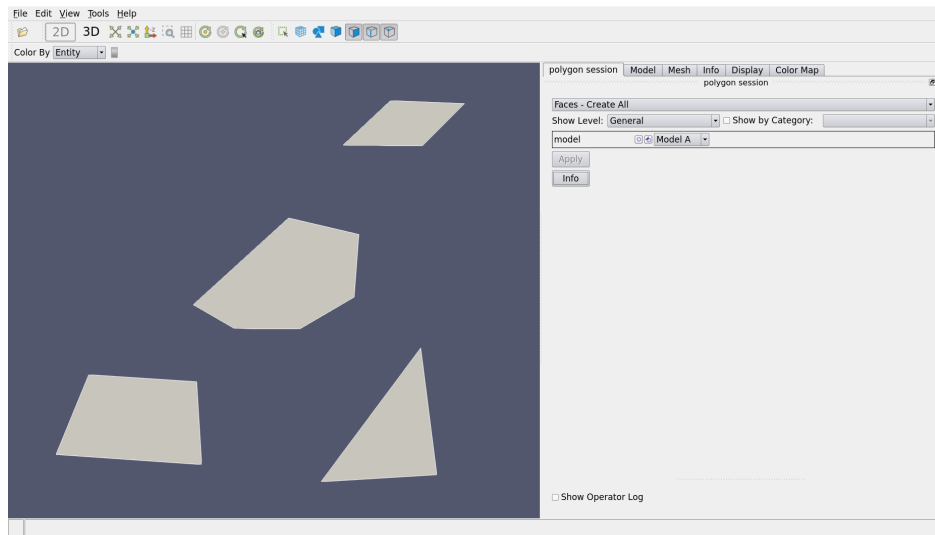


Similarly to reshape, an edge can be split interactively with a vertex through “Edge - Split”. As shown in the figure below, a closed edge is split into two:



Note: “Edge - Reshape” and “Edge - Split” can only be performed on edges with more than 2 vertices.

Face Operations Faces are created from edges. “Create from Edges” allow the user to create a face from the selected edges while “Create All” will create multiple from all the closed edges in a model.

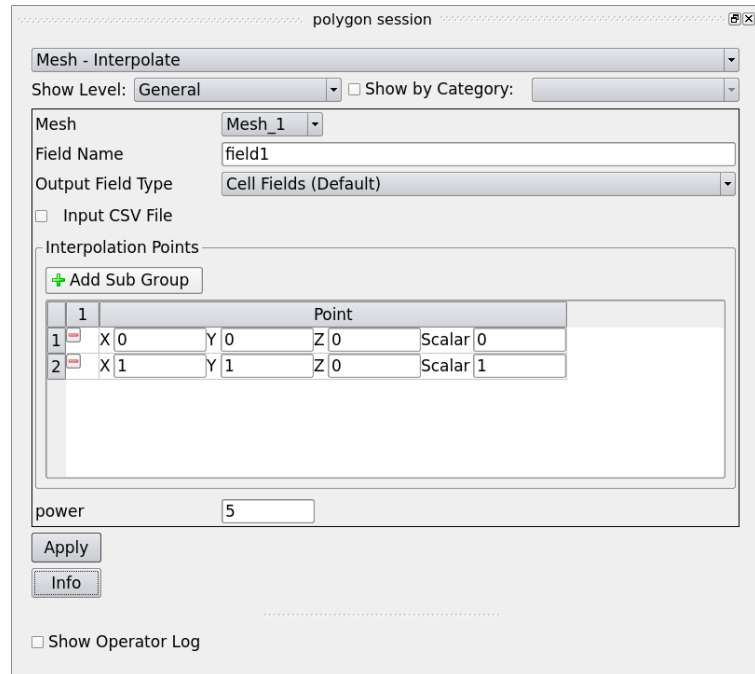


Mesh Operations If a mesh is created or loaded, it will also be shown in the model tree. Mesh can be deleted, saved or exported. However, currently ModelBuilder is unable to generate mesh, you must have an external mesher to do that.


See also:

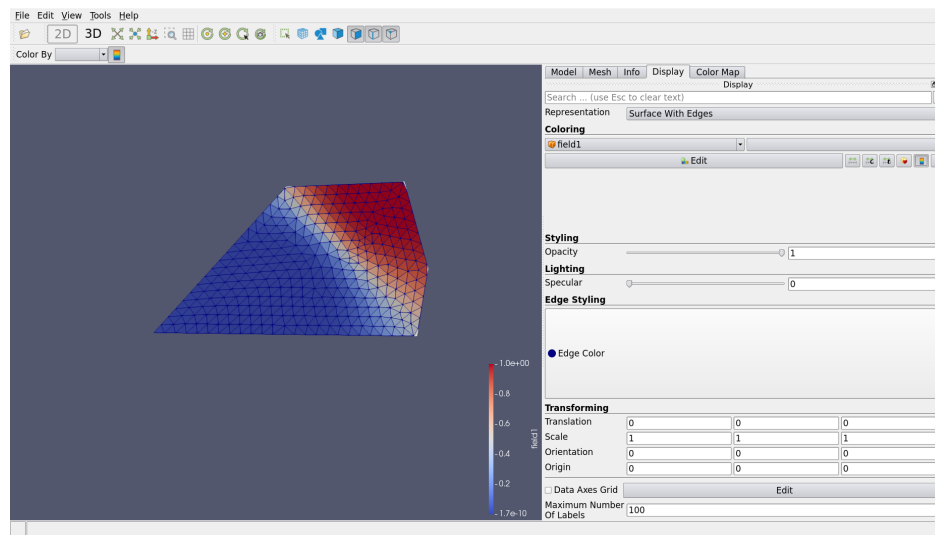
The [Mesh Tab](#) for using external mesher with ModelBuilder.

Another interesting operation available for mesh is interpolation, which creates a field variable either on cell fields or point fields. Right-click on Mesh - interpolate:



First you need to select a mesh to interpolate on, then specify a field name. After choose an output field type, you can create interpolation points where you assign the values of the field variable. In this example, two interpolation points are created, ranging from 0 to 1. The interpolation is done with [Shepard's method](#) and the “power” entry specifies the weighting power. Hit “Apply” the field variable will be interpolated on the whole mesh.

You may want to visualize the field variable. To do so, go to Display tab, choose the field variable in Coloring. You can also click on  to bring up the legend bar.

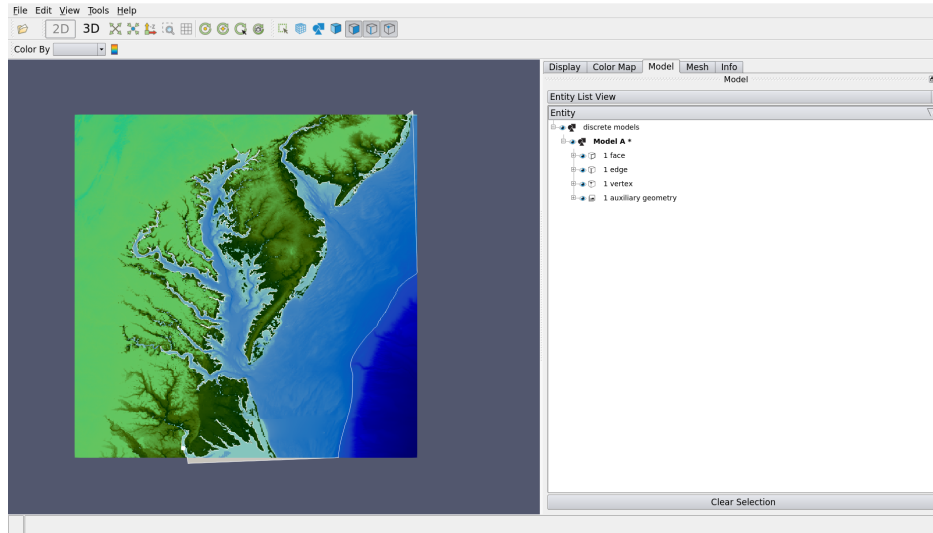


Bathymetry can be applied to mesh through Mesh - Apply Bathymetry. We save this feature for *The Discrete Session* as well.

Discrete Session

The *discrete session* is backed by the [VTK](#). VTK data structures are used to represent the geometry. The model could be either 2D or 3D. In this section, we introduce some features of the *discrete session*, however, they are not necessarily exclusive to the it.

Add Auxiliary Geometry As mentioned in *The Polygon Session*, auxiliary geometry can be added to the model. To illustrate this, open up [ChesapeakeBayContour.cmb](#), then click Model - Add Auxiliary Geometry, and select [ChesapeakeBay100x100](#). You should see both the model and auxiliary geometry on the screen.



Once we loaded an auxiliary geometry, we can do more things with it. For example, apply bathymetry. Before that, let us scale our model first, because in this particular example, the size of the model was scaled but the bathymetry data was not. Go to the Display tab and enter (200, 200, 1) in the “Scale”.

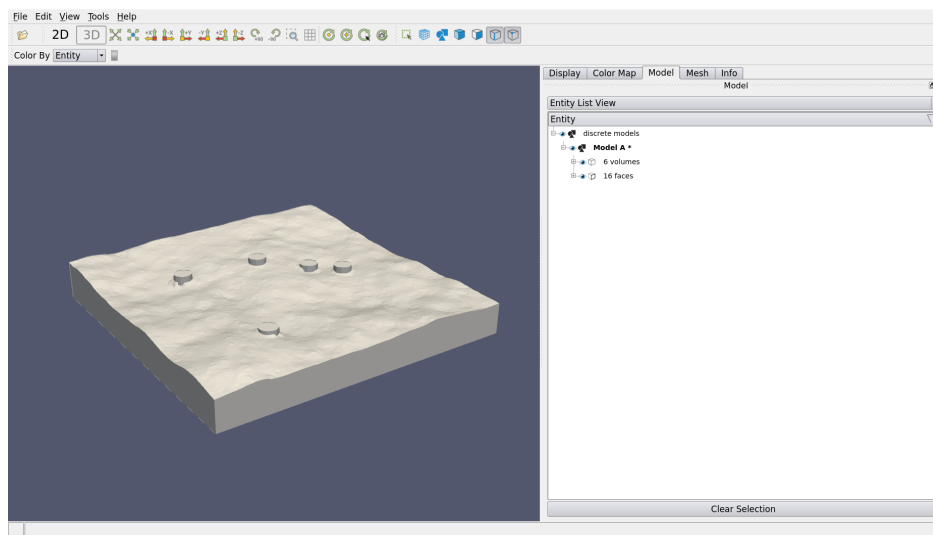
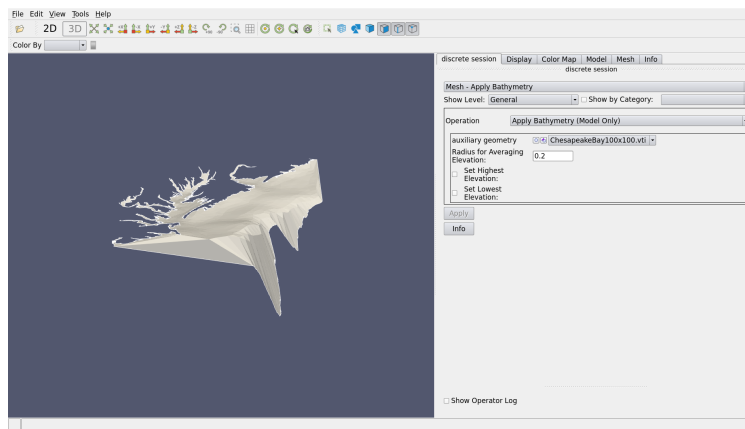
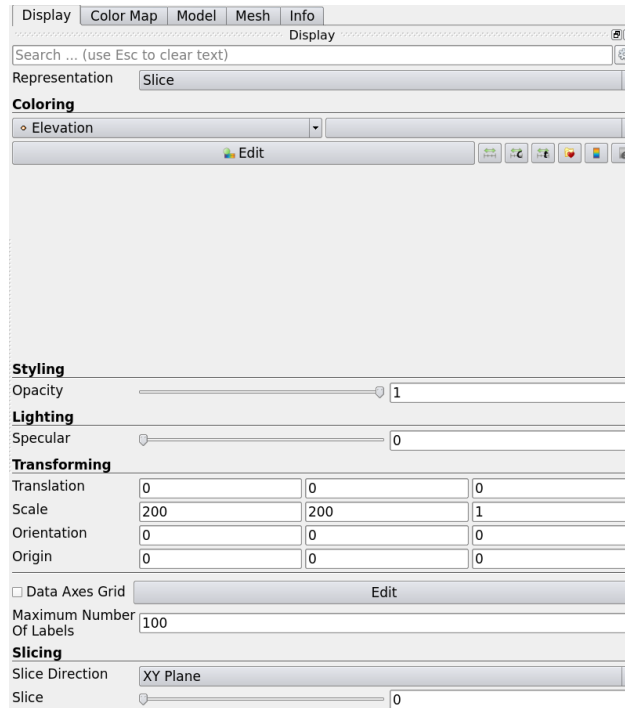
Apply Bathymetry This allows for adding depth to 2D models. Right-click on the model in the entity tree, select Mesh - Apply Bathymetry. In the following window, select the operation: bathymetry can be applied to model or mesh or both. Here we choose model only. Then specify the auxiliary geometry to be used and the radius for averaging elevation, which is the search radius of every point to look for other points to modify. Highest/Lowest Elevation can be used to set a hard cap for the generated elevation.

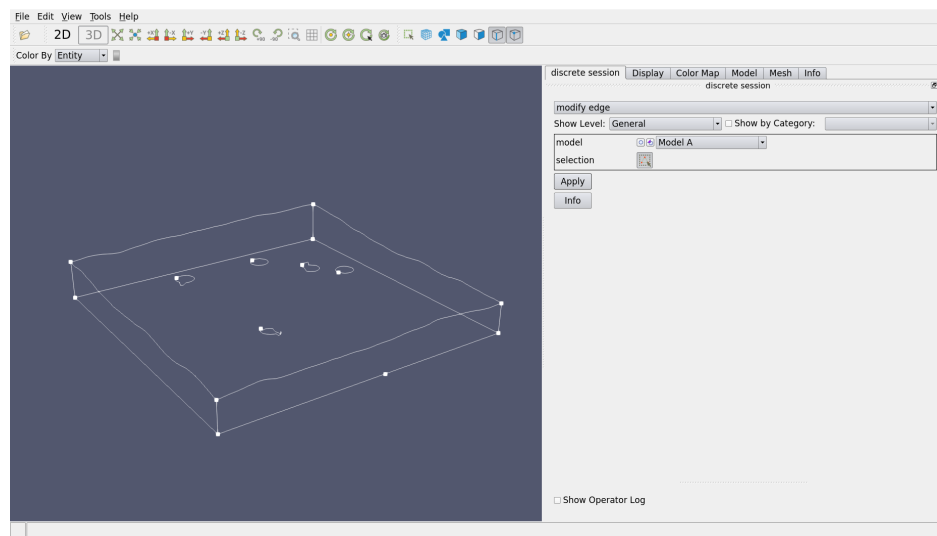
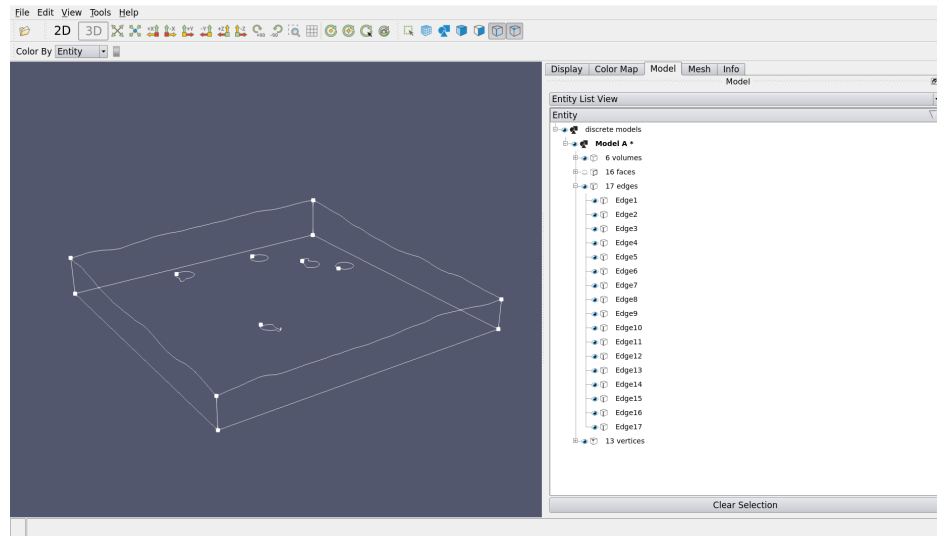
Toggle to 3D view, you can see the bathymetry is applied to the model as Z-coordinate. The “Remove Bathymetry” option reverts this elevation operation.

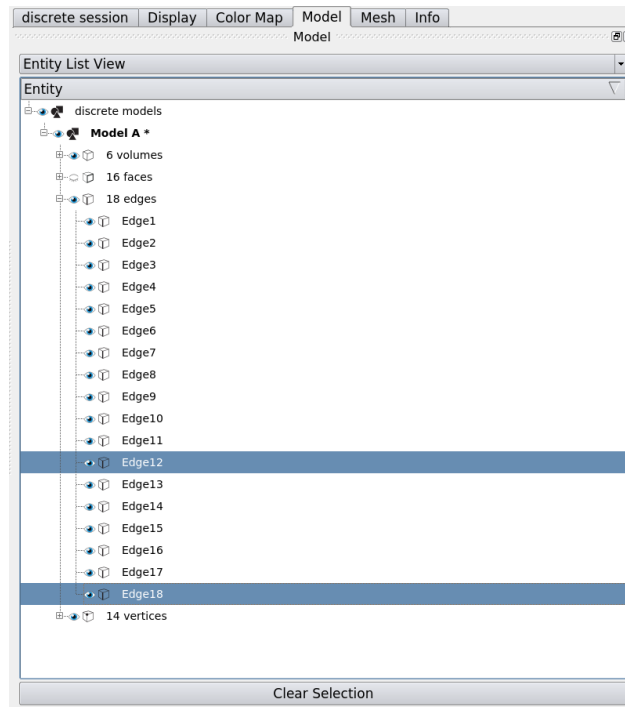
Create Edges This creates edges on the boundaries between faces. Use [smooth_surface.cmb](#) as an example, right-click on the model and select create edges, and specify the model in the next window, the edges will be automatically created at the intersections of faces. The following figures show the entity tree before and after the Create Edges operation.

Modify Edge Continue with the [smooth_surface](#) example, we introduce the Modify Edge operation. This operation can easily split an edge by adding a vertex on it. For example, box-select Edge 12 in the viewport, a new vertex will be added to where you selected the edge. Further look at the entity tree, and notice that a new Edge (Edge 18) has been created out of Edge 12.

Furthermore, if you click on “Apply” on the Modify Edge panel again, the operation will be reverted.

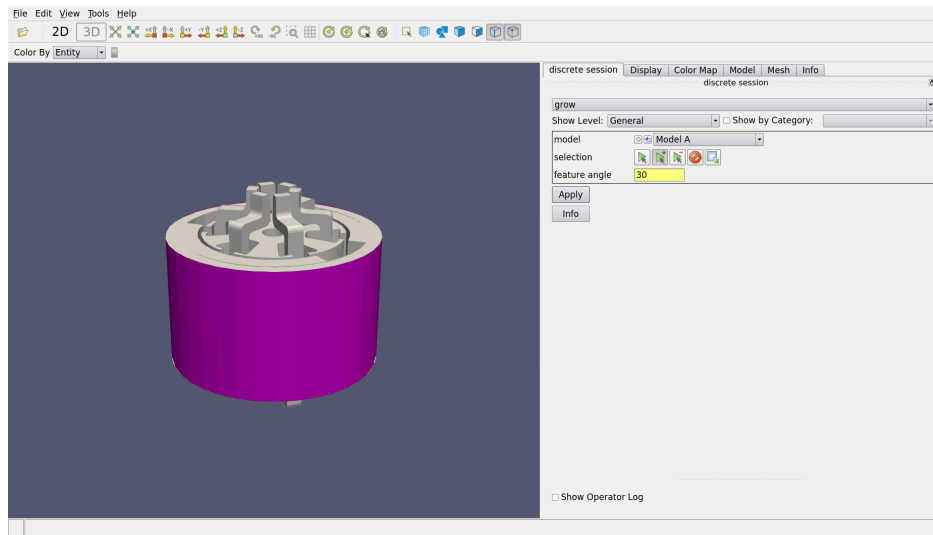






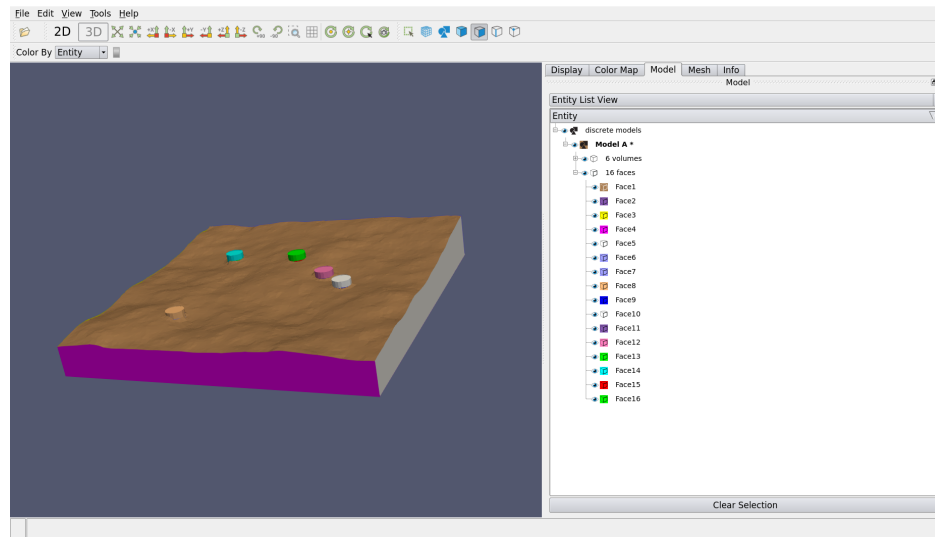
Grow Grow is used for selecting a group of adjacent faces: select one face as a seed, specify a criteria, ModelBuilder will pick the neighboring faces of the selected faces recursively, until the angle between the norms of the selected face and its neighbor exceeds the criteria. This feature only works in 3D.

For example, clicking on one small face on the outer cylindrical surface in `pmdc.cmb` in grow operation selects the whole cylindrical surface.

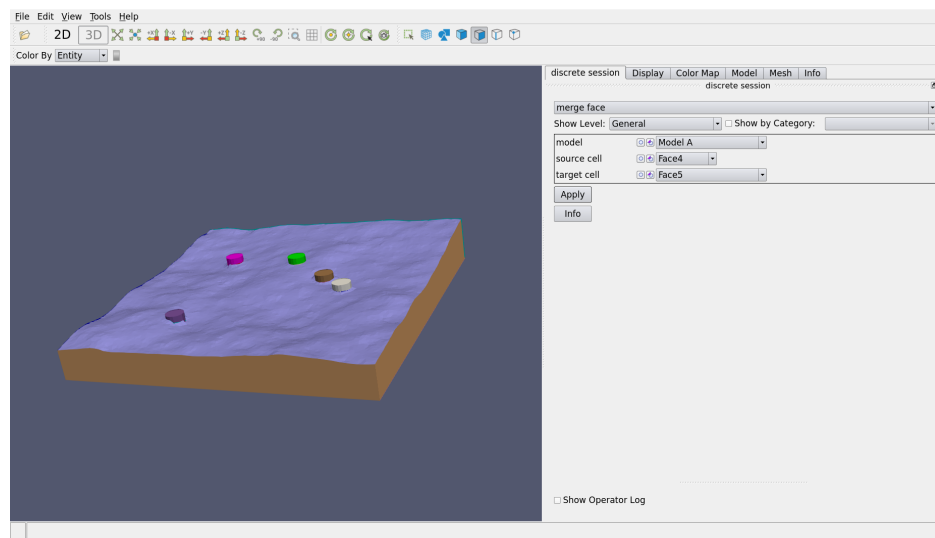


Split Face Selecting a face and splitting it using the feature angle. The mechanism is similar to “Grow” where the feature angle is used as a criteria to detect the neighboring faces.

Merge Face This operation can be used to combine adjacent faces. Let us reopen smooth_surface.cmb and color the faces in the model so that we can easily see the face identities.



Now right-click on the model in the entity tree and select “merge face”. Specify Face4 as the source cell and Face5 as the target cell and hit “Apply”. As shown in the figure below, you can see that those two faces are merged by noticing they have the same color. (or you can check the entity list)



Entity Group This operation creates a group based on the currently selected entities. In 2D models it creates groups of edges, while in 3D models it creates groups of faces. The groups can be edited through Modify Group and Remove Group.

Read This opens a new model, which is equivalent to File-Open.

Write This saves the current model.

UI Documentation

Default Toolbar Buttons

ModelBuilder by default starts with four toolbars open: File IO, Camera Controls, Selection, and Color.

File IO



Open a model or simulation template.

Camera Controls



Toggle between 2D and 3D camera interaction.



Reset the camera to the default view.



Zoom the camera view to a user-selected box and select the elements within the box.



Set the camera to the respective axis.



+90 Rotate the camera view 90 degrees clockwise.



-90 Rotate the camera view 90 degrees counter-clockwise.



Zoom the selected entity.



Show axes grid



Show a 3D orientation indicator of the camera's center.



Reset the camera center to the default.



Allow the user to manually click and define a camera center.



If checked, the camera focal point will also be changed to the rotation center when a new center is set with



Selection



Allow the user to select an objects and faces with a box



Allow the user to select meshes in the box selection



Allow the user to select models in the box selection



A filter to select volumes



A filter to select faces

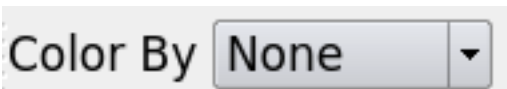


A filter to select edges



A filter to select vertices

Color



Color each entity, group, volume, attribute with a unique color



Show a legend of the colors, only valid when the model is colored by attributes

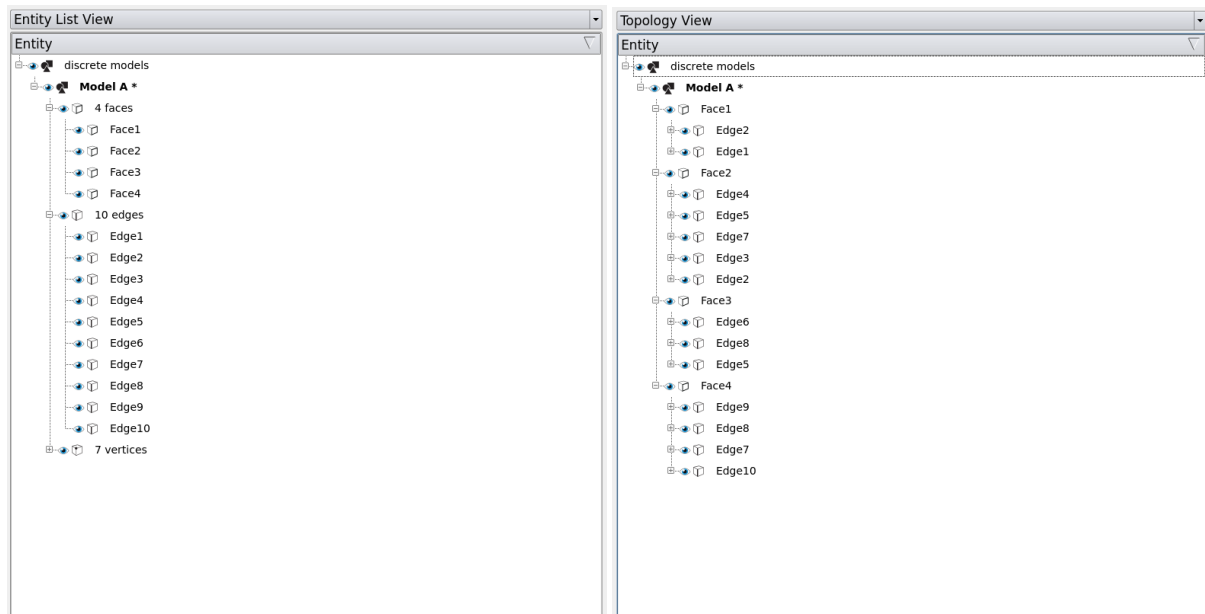
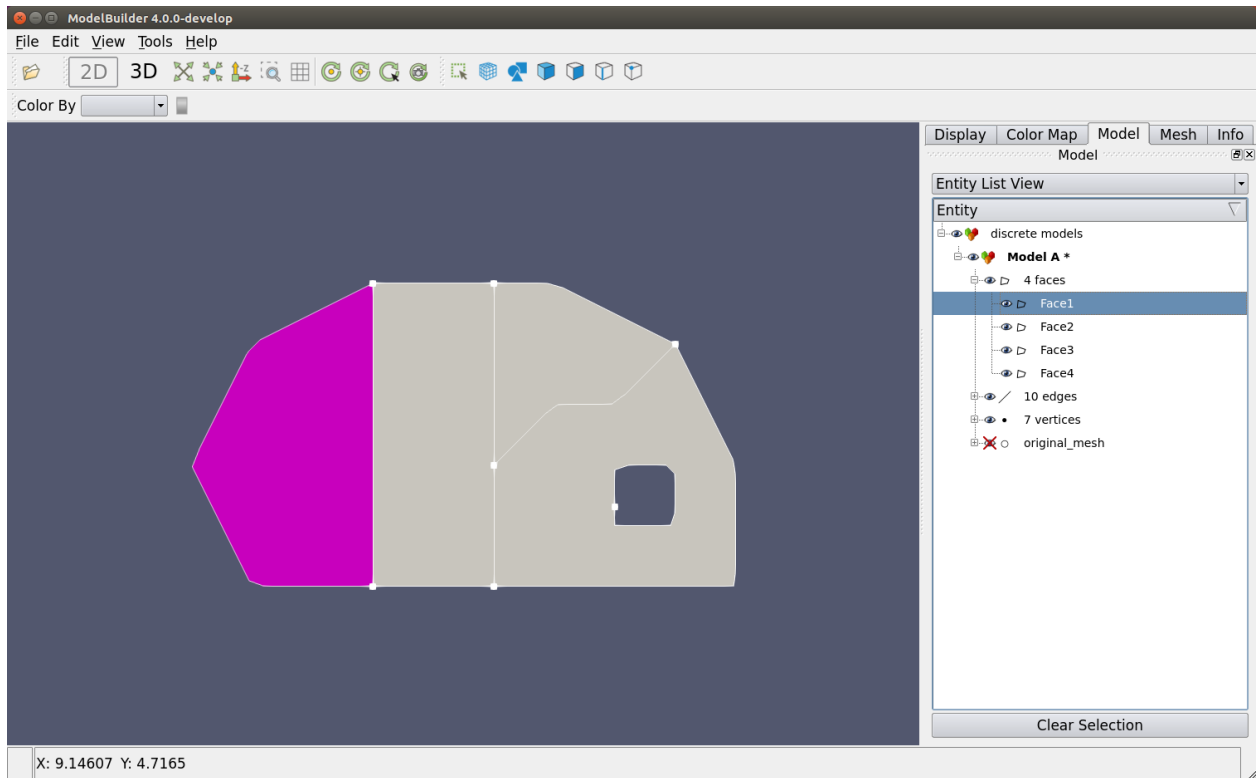
Object Properties Window

The Model Tab

The Model Tab contains a hierarchical tree of models in the scene. Multiple models can be created in one session, but a simulation should contain only one session.

When an entity is selected, either through the entity list or the viewport, it will be highlighted. Click the “Clear Selection” button or press Esc on the keyboard to clear the selection.

There are two different views of the model tree - entity list view and topology view, which can be switched in the dropdown menu. The entity list view groups the entities based on their types, i.e., faces, edges, vertices, meshes. The topology view, on the other hand, presents the relationships among these entities by categorizing them as parents and children.

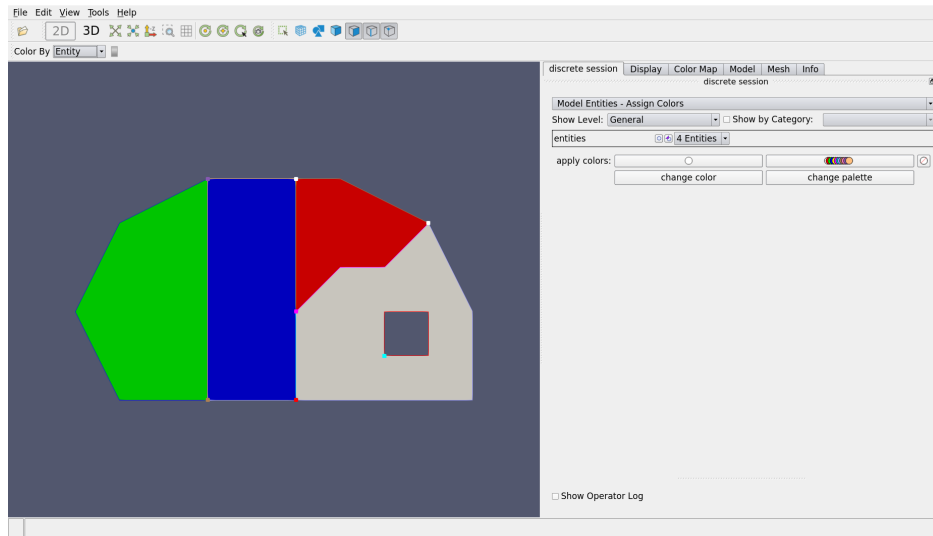


Visibility Clicking the  symbol toggles the visibility of the corresponding part.

Right-clicking Content Right clicking on any entity brings up a context menu with the operations can be applied to the specific entity. The context menu varies in different sessions as they support different operations. For example, you can create edge interactively in the *polygon session* but not in the *discrete session*.

Assign Colors A useful operation that is shared by all the sessions is assigning colors. Colors can be assigned to a specific entity or multiple entities at the same time.

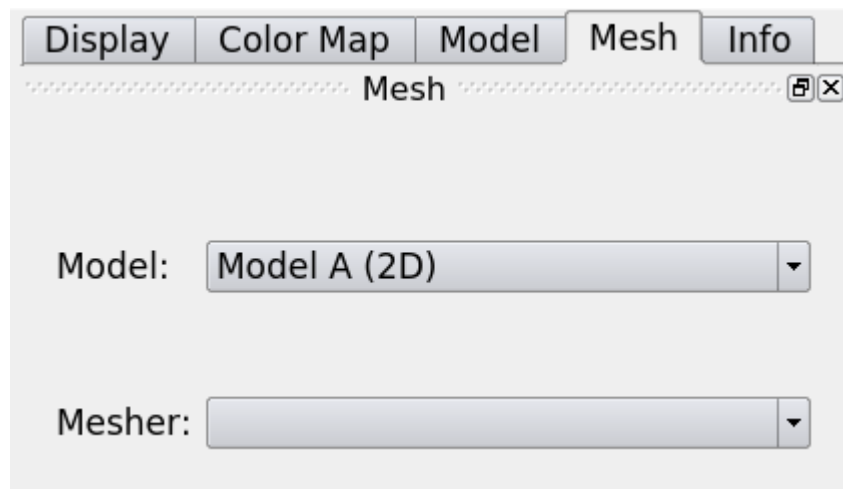
Take [test2D.cmb](#) for example. Right-click on the item in the entity tree and click on “Model Entities - Assign Colors”. You can either apply a specific color to an entity, or assign a palette to a group of entities. Here we use the default palette to color the faces.



Clear selection after you apply colors. Now every face is colored uniquely. This is particularly useful in the complicated models where you cannot tell the entities apart easily. You can turn off the colors by changing “Color By Entity” to others.

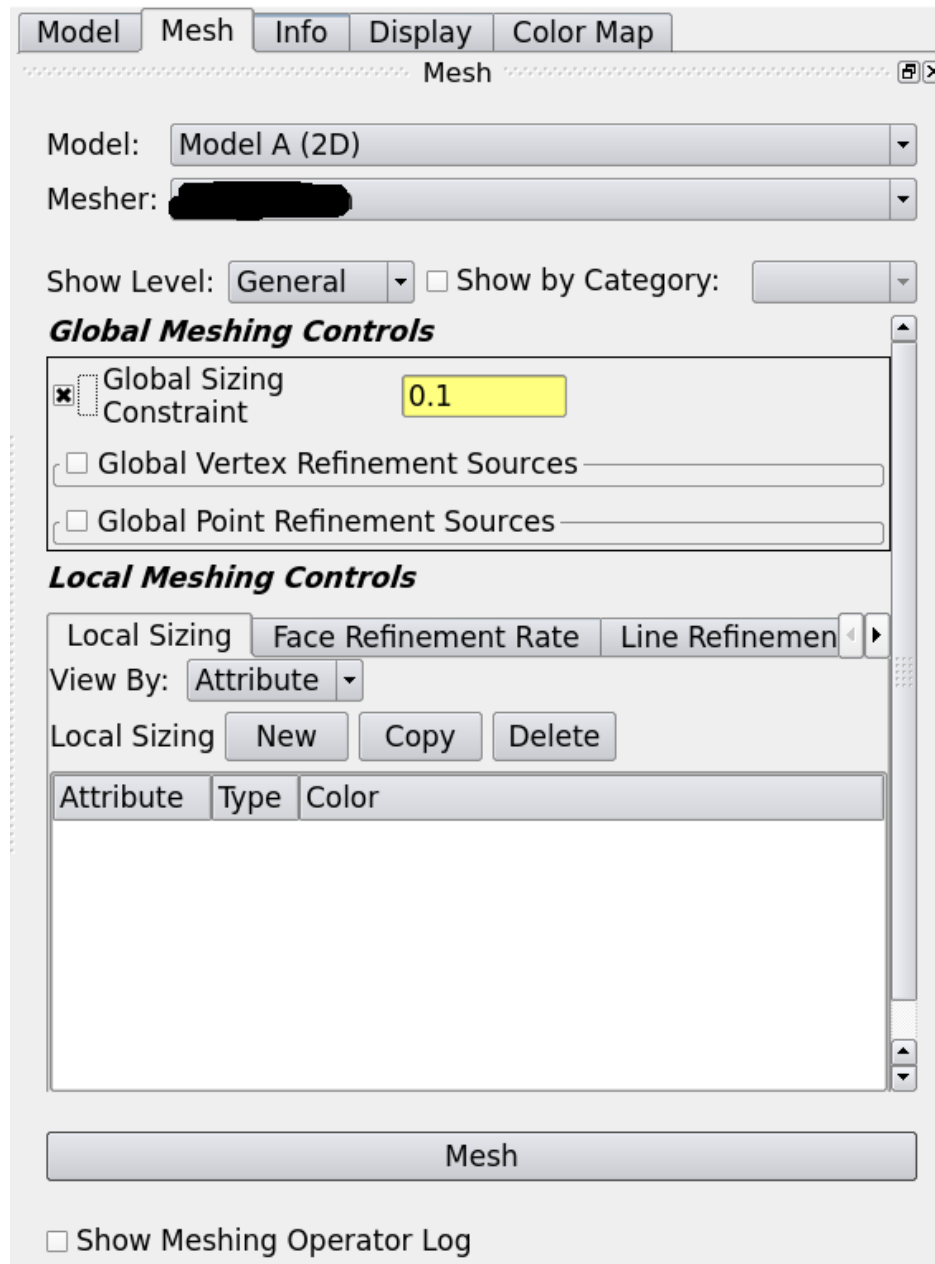
The Mesh Tab

This tab allows the user to mesh a model with an external mesher. By default the mesh tab does not have much information. The dropdown menu of “Mesher” is empty unless the user runs a mesher that is linked to ModelBuilder.



Since currently ModelBuilder is not shipped with any third-party mesher, the user has to integrate his/her own mesher into ModelBuilder.

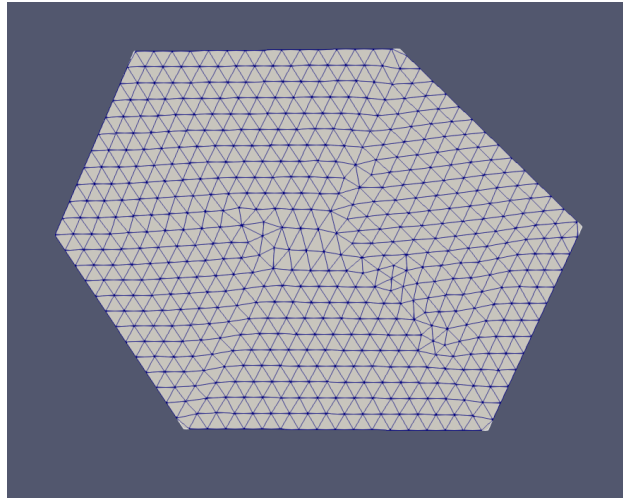
Here is an example of an external meshing package we have. We created an interface for it so that it can talk to ModelBuilder. In order to use the mesher in ModelBuilder, we must run it in the background. In our case, we run it in the terminal window. After the mesher is called, the Mesh tab shows the control options and parameters.



The menus under Mesh tab depends on the mesher and the interface program. In our example, global sizing, local sizing, refinement rate etc. can be specified. Click on “Mesh”, a 2D mesh will be generated on the model. The figure below shows the mesh generated on a random polygon model.

See also:

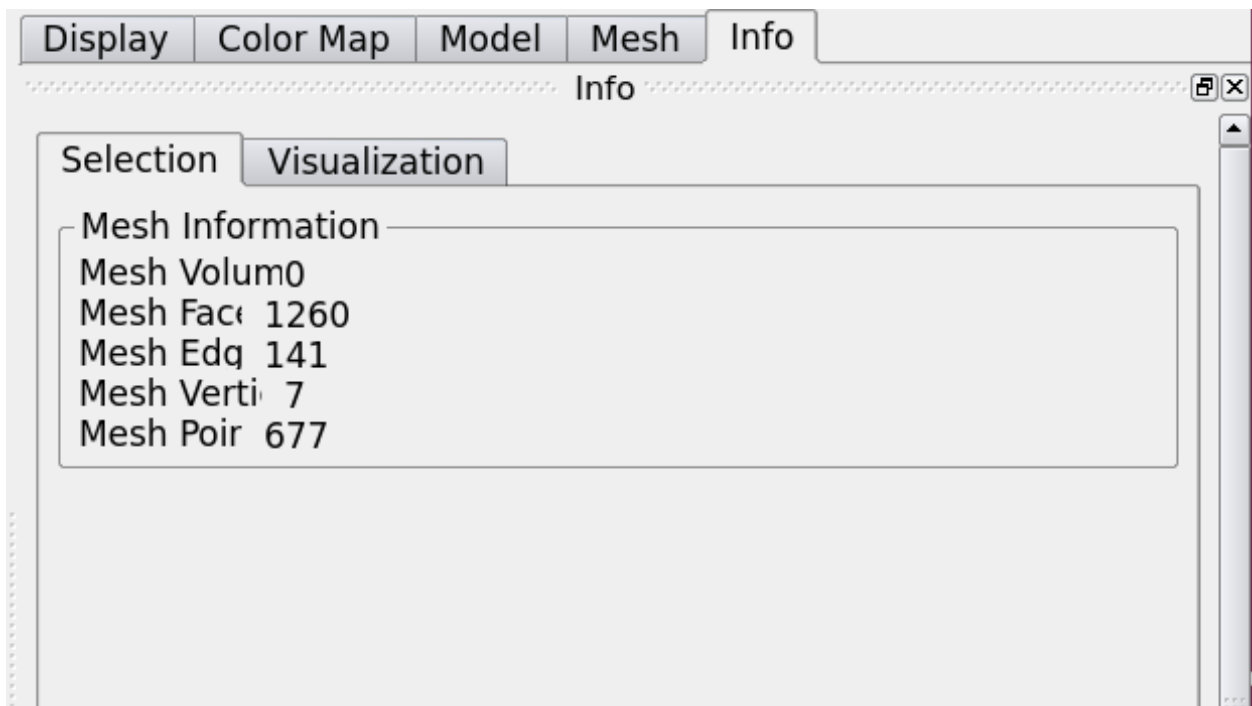
[SMTK Mesh System Reference](#) for meshing system



The Info Tab

This tab does not have configurable options; it only displays the statistical information of the model and the mesh.

There are two sub-tabs under it: “Selection” and “Visualization”. “Selection” shows the information of the selected mesh, including the number of volumes, faces, edges, vertices and points.



“Visualization” shows the rendering information of the model. “Data Hierarchy” is a hierarchical list of all the entities in the scene. Each can be individually selected to view more information. “Information” shows the overall information about the VTK dataset. The related data, for example coordinates can be viewed in “Data Arrays” and “Bounds”. Time values will be displayed if it exists.

Display
Color Map
Model
Mesh
Info

Info

Selection
Visualization

Data Hierarchy

- Multi-block Dataset

Information

Type: Multi-block Dataset

Number of Cells: 1408

Number of Points: 884

Memory: 0.15 MB

Data Arrays

Name	Data Type	Data Ranges
PointCoordinates	double	[0.0143589, 9], [0, 5], [0, 0]
Entity	string	NA

Time

Index	Value
-------	-------

Bounds

X range: 0.0144 to 9 (delta: 8.99)

Y range: 0 to 5 (delta: 5)

Z range: 0 to 0 (delta: 0)

The Display Tab

Display Color Map Model Mesh Info

Display

Search ... (use Esc to clear text)

Representation Surface

Coloring

Solid Color

Edit

Styling

Opacity 1

Lighting

Specular 0

Transforming

Translation	0	0	0
Scale	1	1	1
Orientation	0	0	0
Origin	0	0	0

☐ Data Axes Grid Edit

Maximum Number Of Labels 100

The Display Tab contains various options related to displaying the model.

“Representation” changes the way the model is displayed in the viewport

3D Glyphs: show the 3D glyphs at the underlying points

Outline: display the bounding box of the model

Point Gaussian: show the underlying points as solid spheres

Points: show the underlying points as solid dots

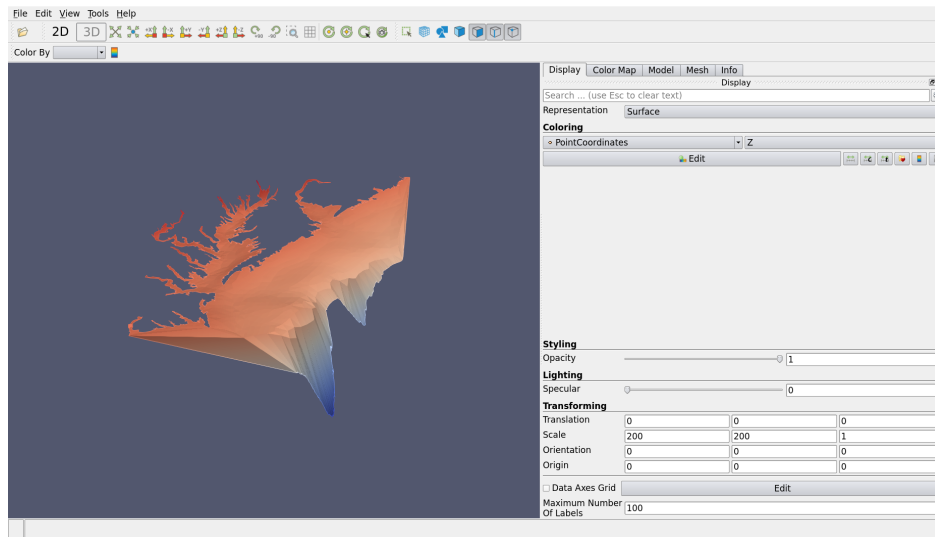
Surface: hide the underlying polygons, faces, etc.

Surface With Edges: display the surface and the underlying edges

Wireframe: display the wireframe of the underlying structures

When viewing the mesh of an object, it may be worth switching the representation to “Wireframe” or “Surface with edges”

Coloring The model can be colored by certain variables, which could not be done with “Color By” menu. For example, we can color the ChesapeakeBayContour introduced in *Discrete Session* with its Z coordinate.



Styling Opacity: control the transparency of the model

Lighting Specular: control the specularity of the model

Transforming Translation: translate the model in 3D

Scale: scale the model

Orientation: define the orientation of the model

Origin: define the center of the model

Miscellaneous Data Axes Grid: display axes grid that helps to visualize the data

Maximum Number of Labels: control the number of labels

The Color Map Tab

This tab is used tune the coloring of the model. It is populated only if the model is colored (either by a certain variable or solid color). It is useful when you want to customize the corresponding colors of the dataset.

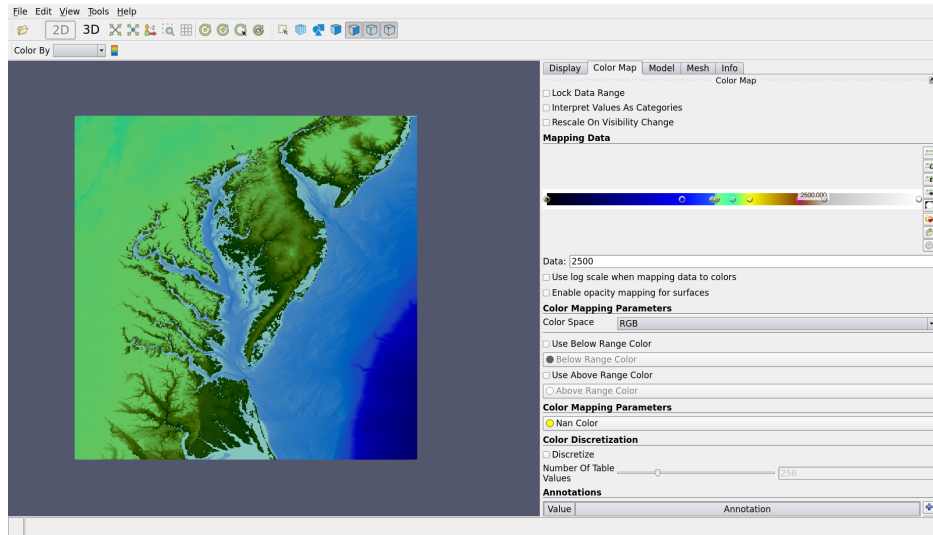
Lock Data Range: fix the data range so that it will not be adjusted by the filters

Interpret Values As Categories: switch the color functions to categorical color mode

Rescale On Visibility Change: adjust the color map with the model visibility

Mapping Data When “Interpret Values As Categories” is not checked, this section will appear for the user to manually assign a color to a certain value of the data. The user can specify as many color points as needed. Interpolation will be done for the other values.

In the figure below, color has been assigned at 8 individual data points.



To add a point, left-click on the color bar. To edit the color of a point, double-click it. To remove a point, middle-click it. Click and drag any point to move its location. Data values can be inputted into the “Data” field below the color bar. The contour will be changed accordingly after the mapping data is modified.

There are various options on the right side of the color bar including:

Rescale to data range: To scale to the color bar’s minimum and maximum values to the min/max of your dataset.

Rescale to custom range: Instead of mapping the range of color to the range of dataset linearly, you can also apply a custom range.

Rescale to data range over all time steps: For data with multiple time steps, the min/max range can be set the the dataset’s min/max range among all the time steps.

Rescale to visible range: Rescale to the range of the visible portion of the model.

Rescale to transform functions: Rescale using the transform functions.

Choose preset: Some color maps are available in ModelBulder for the user.

Save to preset: The current color map can be saved as a preset.

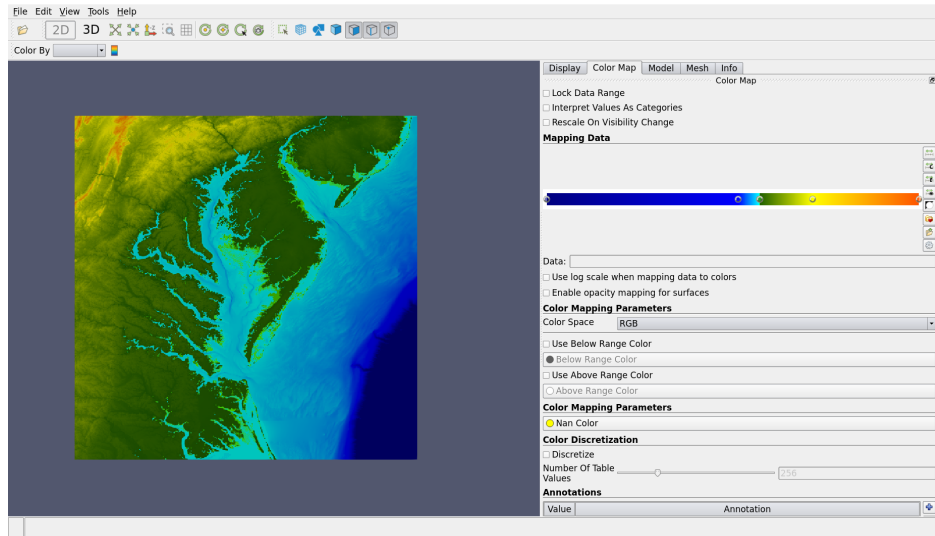
Manually edit transform function: This adds a new table below the color bar, which shows all the entries in the color map for the user to manually edit.

You can check “Use log scale when mapping data to colors” if the original datasets are scaled logarithmically.

Alpha values can be mapped onto elements as well if “Enable opacity mapping for surfaces” is checked

Here is the same dataset using another preset color map:

Color Mapping Parameters Color Space: The color space {RGB, HSV, Lab, Diverging} that the color values are mapped from. Notice how changing the color space changes the mapping:



RGB

Diverging

Use Below Range Color/Use Above Range Color: manually assign a color when the data value is below/above the range

Nan Color: default color for elements of non-numeric values

Color Discretization Discretize: whether or not to pick colors from a discrete set

Number of Table Values: the number of colors to discretize by

Annotations This is a table to annotate the color mapping.

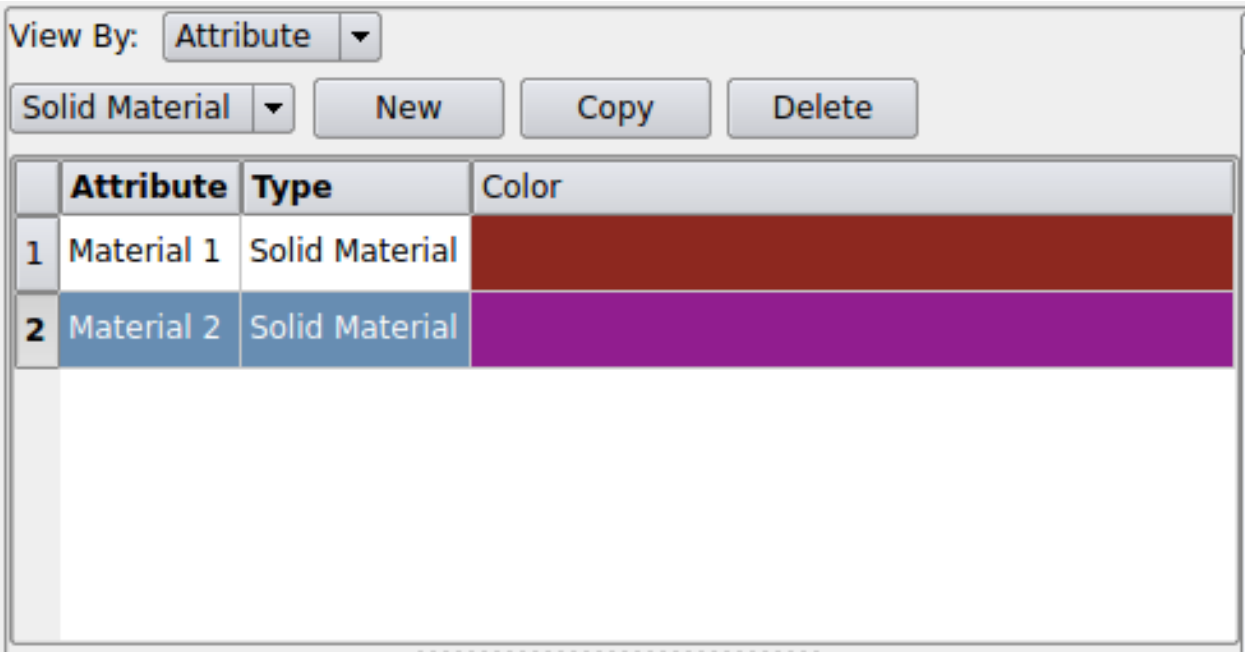
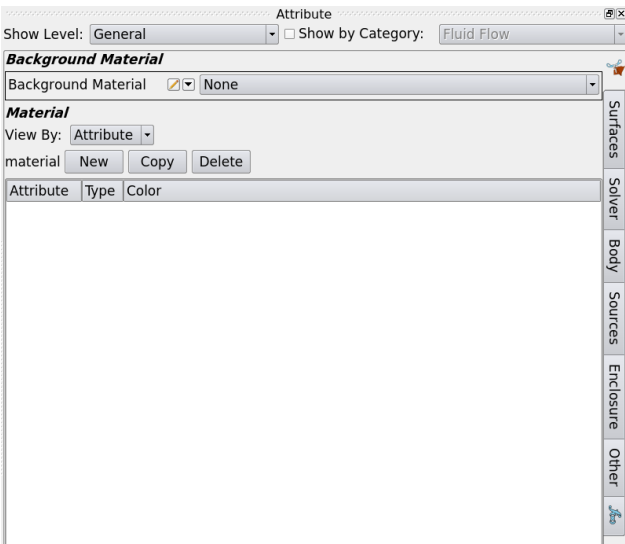
The Attribute Tab

This tab is used to specify simulation parameters and assign them to the model. It only appears when a simulation template has been loaded.

There are usually one or more tabs along the right side of the attribute tab, which separate the simulation parameters into different sections. These are defined and generated based on the loaded attribute file. For example:

“Show Level” controls how detailed the attribute tab should be. The template writer decides the level of information. This feature can be used to hide unnecessary information for certain users.

Another common feature is “Show by Category”. This is another way to categorize the simulation parameters. Unlike “Show Level”, it categorizes the parameters based on their physical meanings (it is also specified by the template writer).



Attribute Tabs There are two types of parameters: attribute and instanced. The attribute parameter can have multiple instances which allows the user to create or delete based on the simulation needs, while the instanced has only one instance.

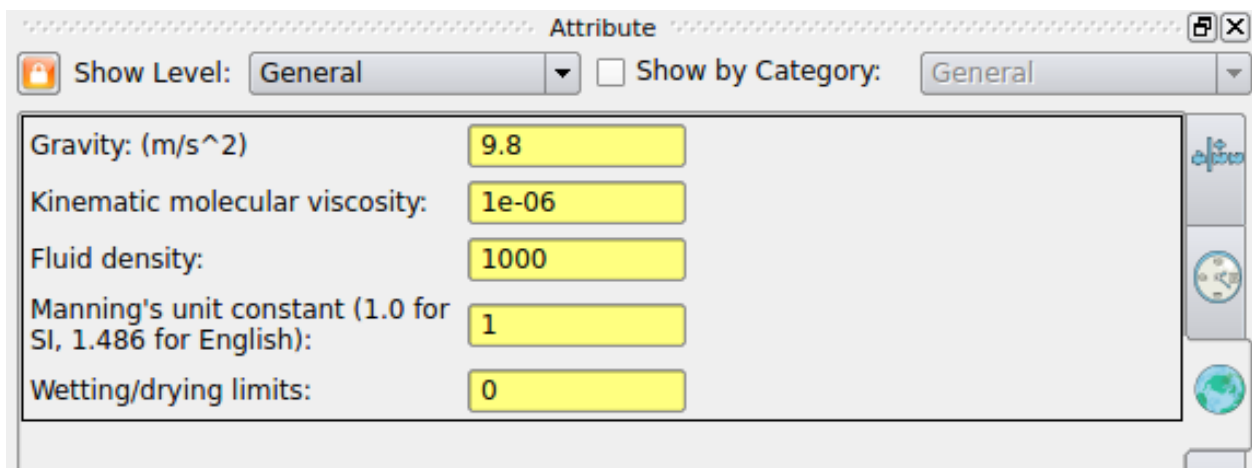
This figure shows a typical attribute, where you can add or delete materials. To add a new entry, click the “New” button. The new entry should be selected in the table. Double-click it in the “Attribute” column to edit the name. Click the “Color” column to change the representation color.

After selecting an entry, a set of configurable options should appear below the table. These can all be edited and are specific to the selected attribute entry. For attributes that can be assigned, a two-column table will be shown. Only entities that can be assigned (volumes, faces, edges, or vertices) will be shown. Moving an entity to the left side will assign the entity to the selected attribute.



Instanced Tabs Instanced tabs are used for data that has only one instance in the entire simulation for example gravity, time steps and so on.

There will usually be a tabular which lists the labels and values of the instanced parameters, as shown in the figure below:



See also:

[SMTK Template File Reference](#) for populating the Attribute Tab

Using Simulation Template for an example

Indices and tables

- *genindex*

A

Attribute Tab, [37](#)

C

Camera Manipulation, [8](#)

Color Map Tab, [35](#)

D

discrete-session, [21](#)

Display Tab, [32](#)

I

Info Tab, [31](#)

L

Loading Plugins, [7](#)

M

Mesh Tab, [30](#)

Model Tab, [28](#)

P

polygon-session, [15](#)

S

Simulations, [11](#)

T

Toolbars, [27](#)