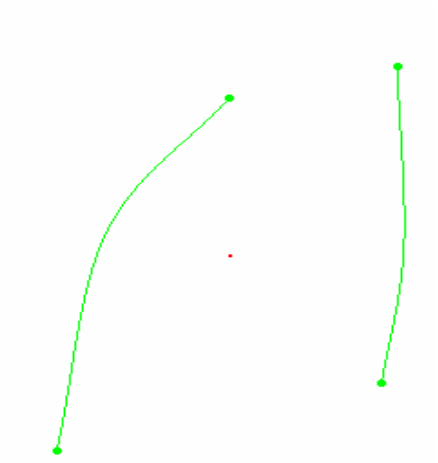
Gridgen**Training**

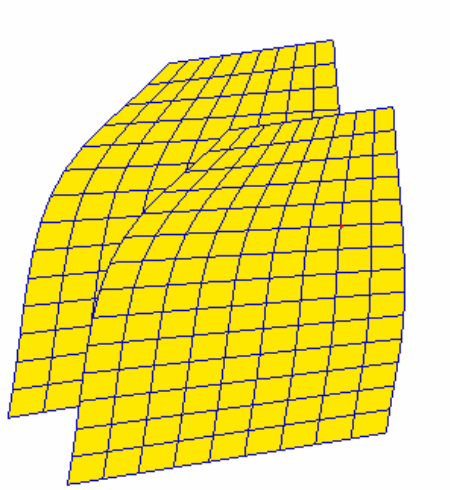
- In this session:
  - Gridgen Terminology.
  - GUI Components.
  - Gridgen Help.
  - Building 2D Structured Grids.
  - Building 2D Unstructured Grids.
  - Tutorial: “2D Bump: Basic Skills.”

What's an Entity?

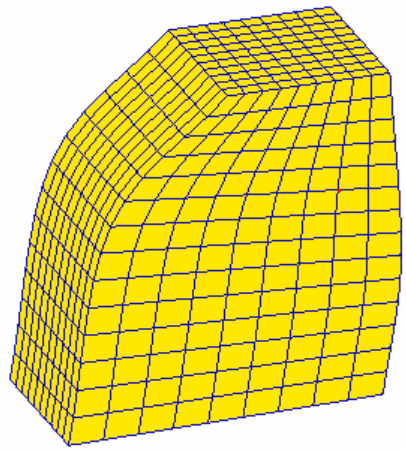
An **entity** is any individual element that makes up part of a grid or geometry.



Connectors



Domains



Blocks

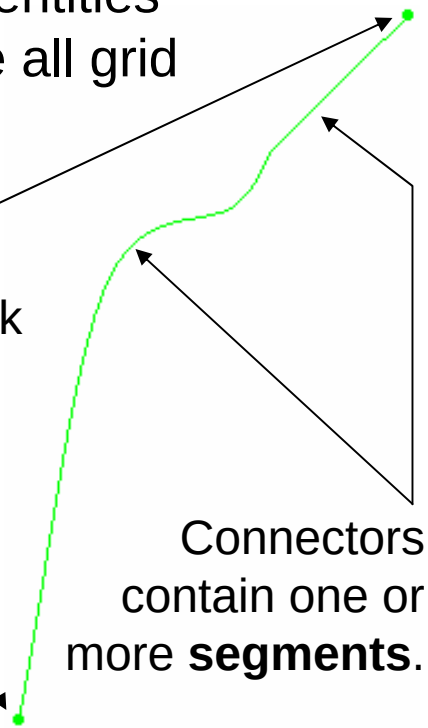
Examples of Gridgen entities

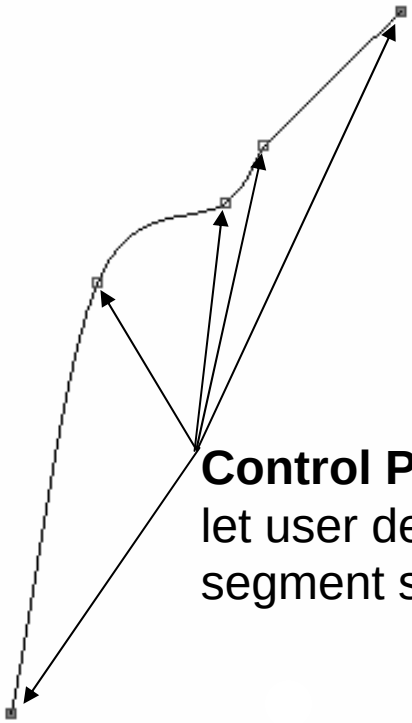
Connectors

Connectors are the curve entities that define all grid edges.

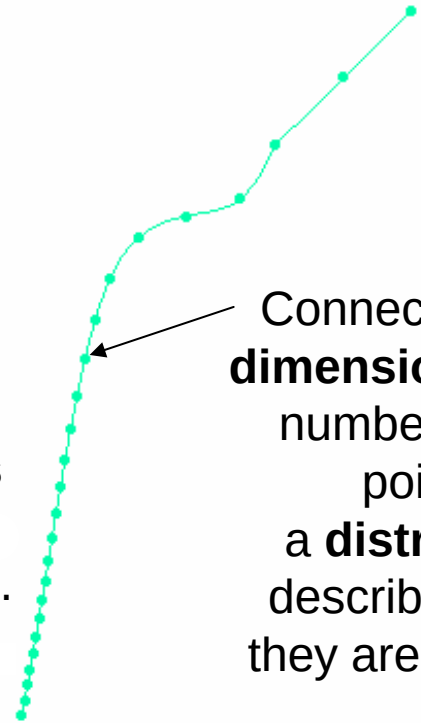
Nodes mark connector endpoints.

Connectors contain one or more **segments**.





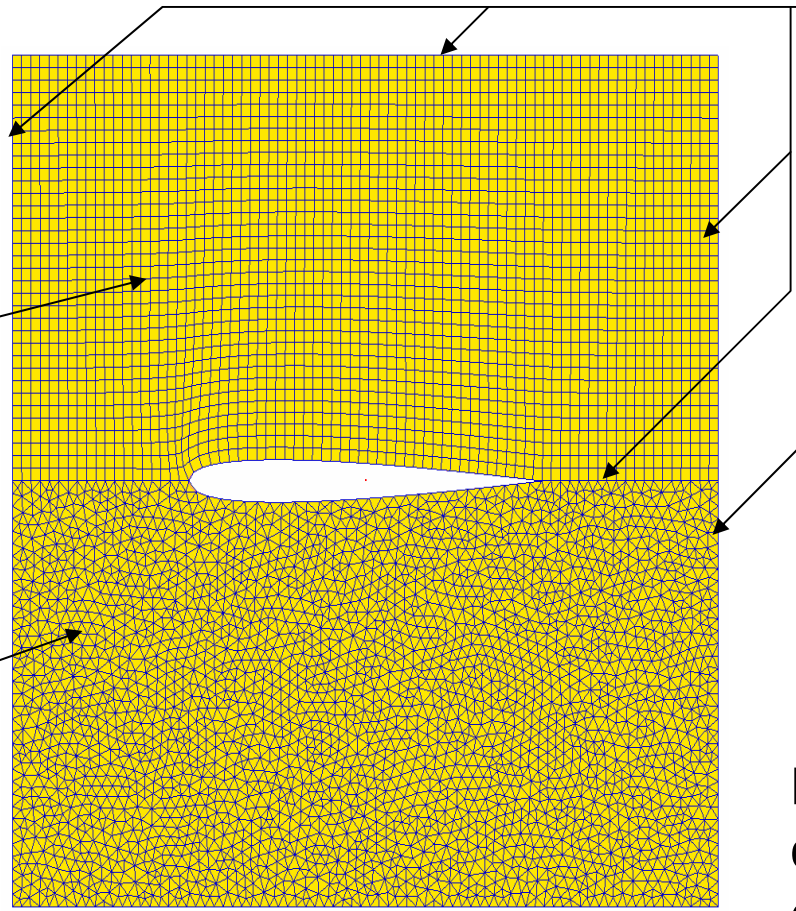
Control Points let user define segment shape.



Connectors are **dimensioned** by number of grid points with a **distribution** describing how they are placed.

Domains

A **Domain** is a surface grid entity bounded by edges.



Structured Domains have exactly four edges.

Structured Domains consist solely of quadrilateral elements.

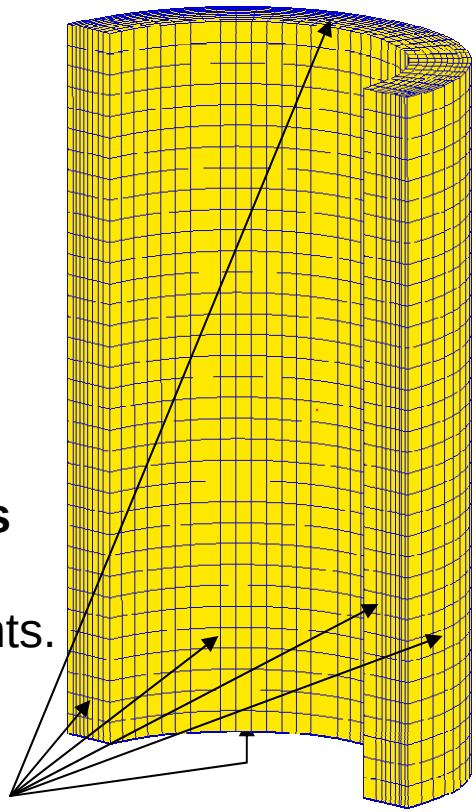
Unstructured Domains generally have one edge (more if they the encompass holes).

Unstructured Domains consist solely of triangular elements.

Each **edge** contains one or more connector(s).

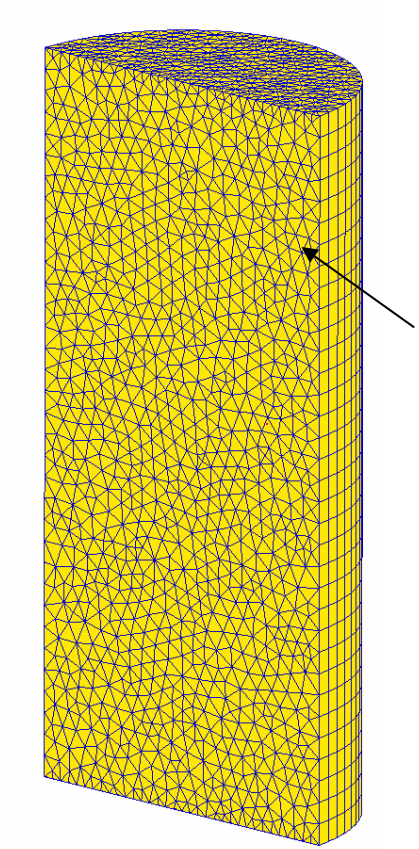
Blocks

A **Block** is a volume grid entity bounded by faces.



Structured Blocks consist solely of hexahedral elements.

Structured Blocks are bounded by exactly six faces.



Unstructured Blocks can consist of tetrahedral, pyramidal, and/or prismatic elements.

Unstructured Blocks generally have one face (more if they encompass holes).

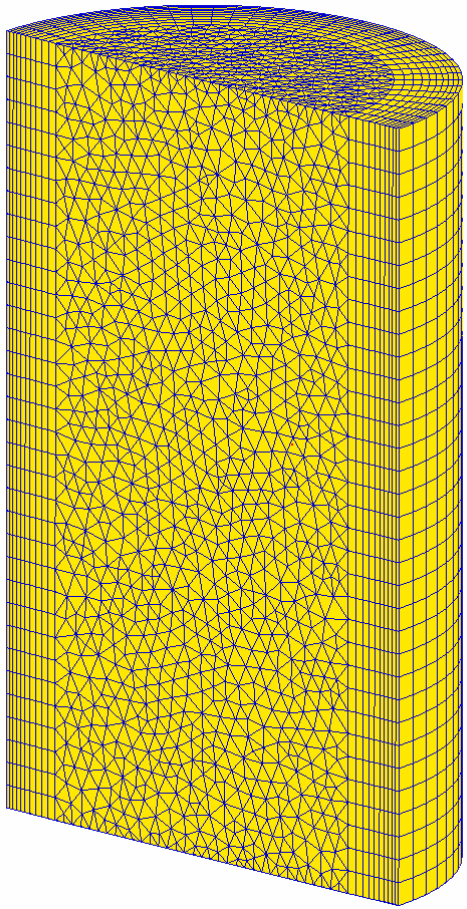
Each **face** contains one or more domain(s).

Grids

A **Grid** is composed of volume and/or surface grid blocks.

Structured Grids contain only quadrilateral and hexahedral elements.

Unstructured Grids contain only triangular and tetrahedral elements.

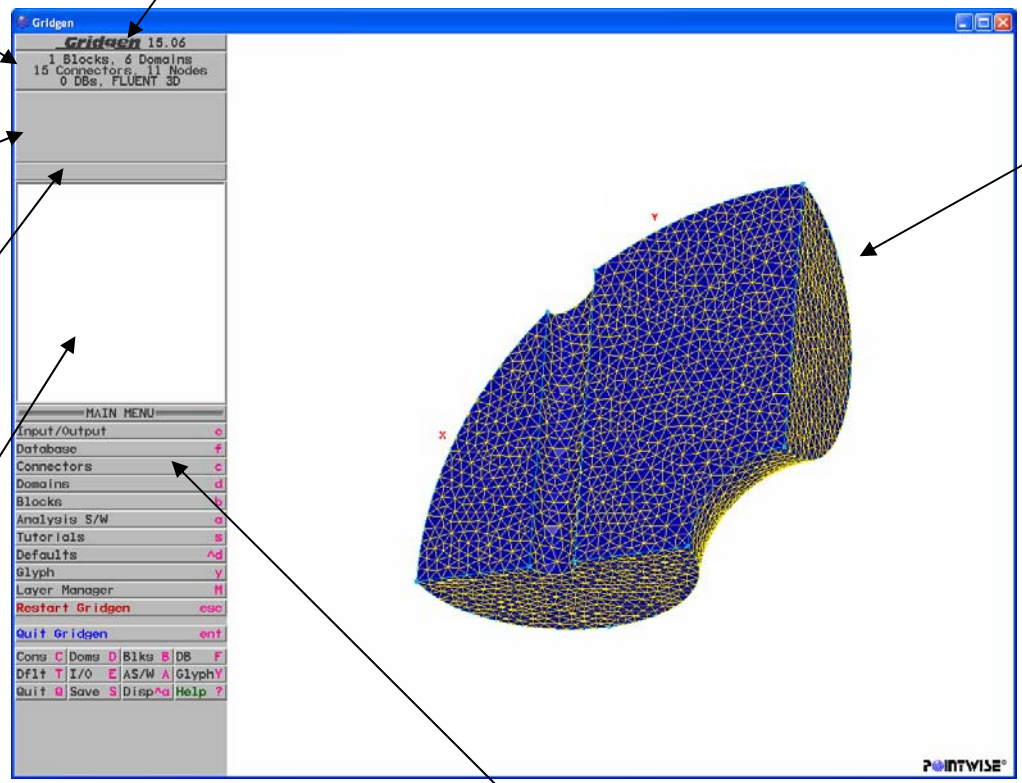


Hybrid Grids (such as that shown here) contain both structured and unstructured grid elements (quads, triangles, hexes, and tets) and can also have prismatic and pyramidal elements.

Databases

- In Gridgen, databases refer to collections of geometry information that can be used to define the shape to which the grid is to be built and typically come from computer aided design (CAD) software:
  - Analytic – curve and surface information from a CAD file.
  - Shell – discrete unstructured database surface entity, generally imported from stereolithography (STL) formatted files.
  - Network – bilinear (non-slope continuous) database surface entity, generally imported from Plot3D formatted files.
- Of course, databases are not necessary to build grids with Gridgen, but are often needed for importing complex geometries.

The Gridgen Screen



Version
Lists Gridgen version number

Status
Tallies number of grid entities

Message
Provides instructions and prompts for text input

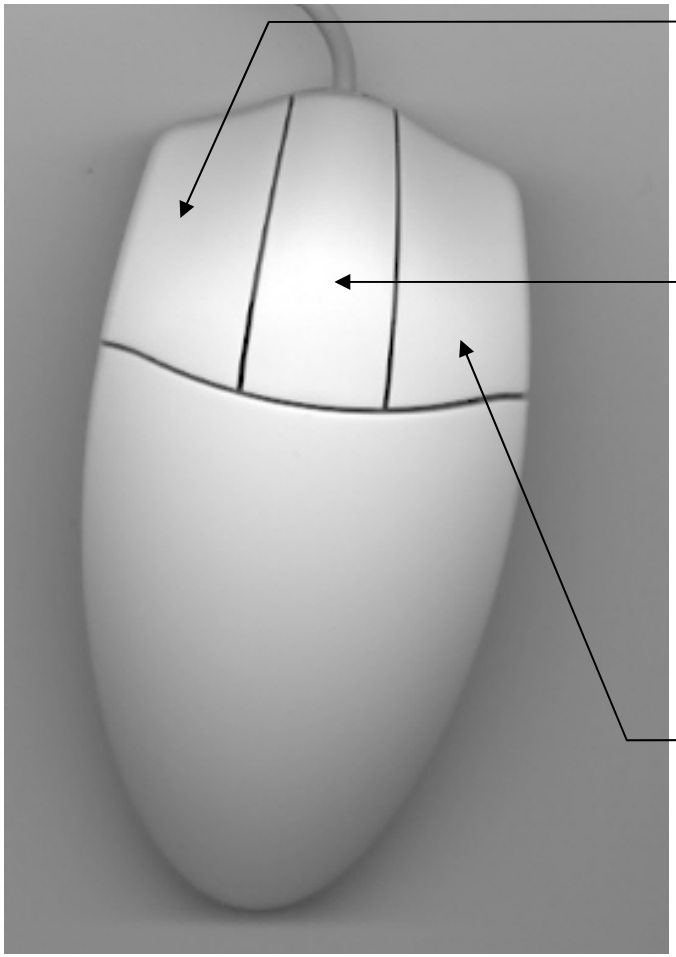
Text Input
Echoes data entered via keyboard

Blackboard
Displays context-sensitive graphical and tabular data (aka Diagnostic or Browser window)

Display
Shows graphical image of grid and database

Command Menu
Lists Gridgen Commands (aka Button Menu)

The Mouse: Pan, Zoom, and Pick



Left Mouse Button: Pan

Press and hold the left mouse button to pan the model image in the display window. While panning, the mouse cursor changes to:

Center Mouse Button: Zoom

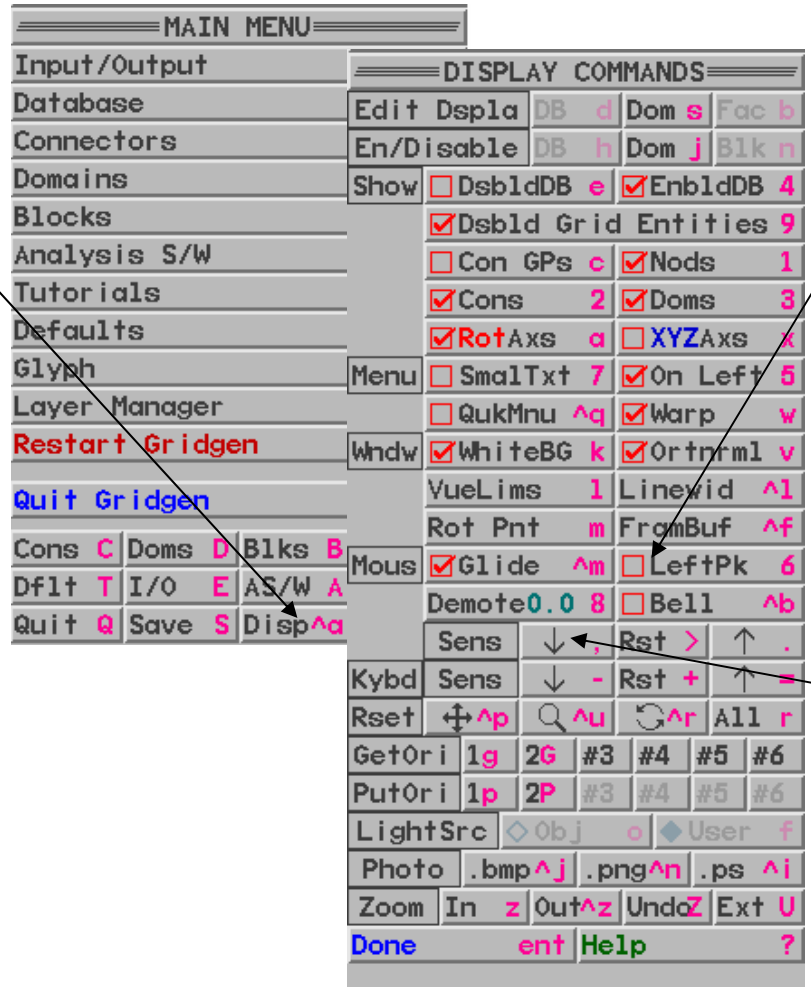
Press and hold the center mouse button to change the model image's size, pushing the mouse away from you zooms out while pulling it towards you zooms in. For mice with a center wheel, the wheel can be rolled or held down to control zoom. On two button mice, zooming is accomplished by holding both buttons down. While zooming, the mouse cursor changes to:

Right Mouse Button: Pick

Press and release the right mouse button to select commands from the command menu or pick entities in the display window or browser list. While picking, the mouse cursor changes to:

Some Mouse Options

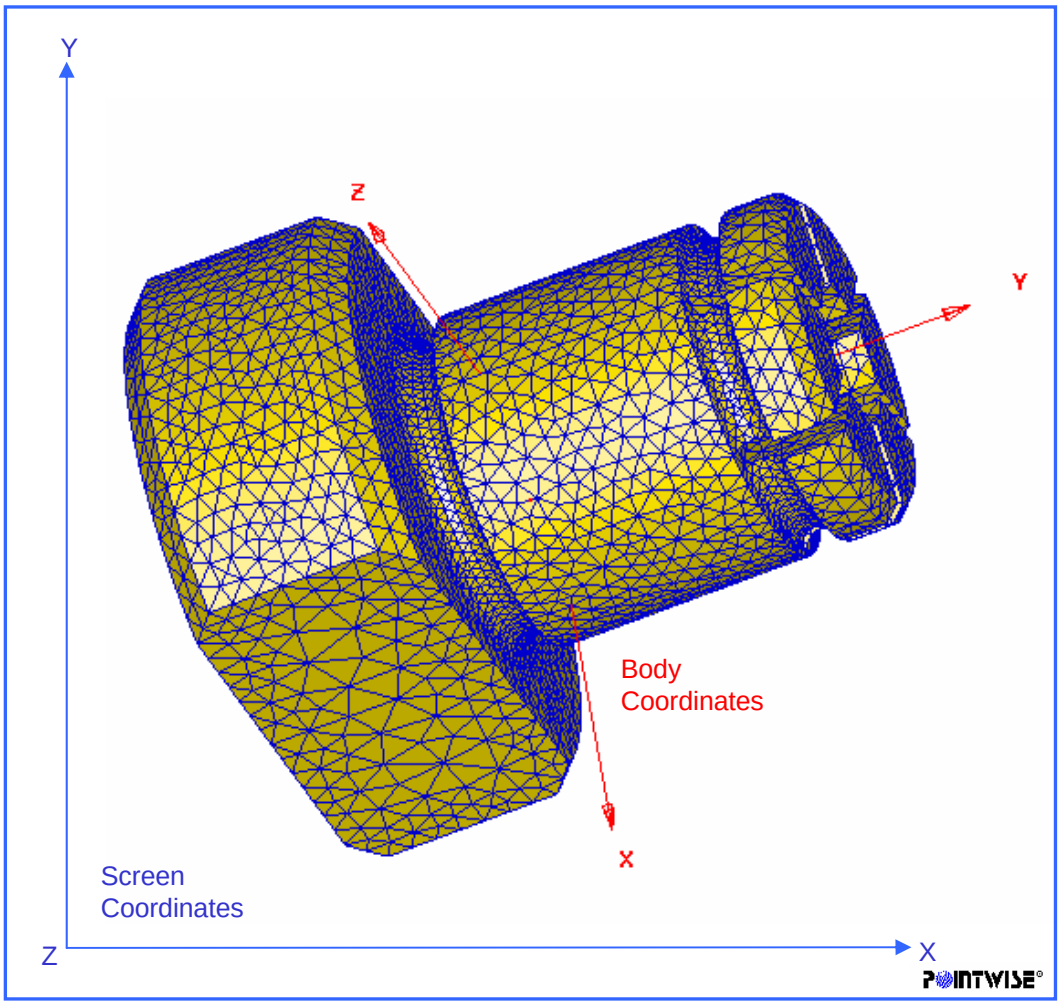
You may wish to change some of the default mouse settings. These can be accessed by selecting the Display or **Disp** submenu from Gridgen's **MAIN MENU**.



The functions of the left and right mouse buttons can be reversed, so that left is for picking and right is for panning. This can be done by selecting **LeftPk**.

When dealing with smaller geometric entities, it may also be desirable to reduce the mouse sensitivity in panning, zooming, and rotating the screen image. This can be done by clicking on the ↓ button (more clicks, less sensitivity).

Model Rotations



- Gridgen has two distinct coordinate systems: screen coordinates and body coordinates.
- In addition to panning, zooming, and picking, the mouse buttons can also be used to rotate the model screen image. This is done by combining the left and center mouse buttons with the **Ctrl** key:
  - **Ctrl** + left mouse button: screen x and y rotations.
  - **Ctrl** + center mouse button: screen z rotations.

Model Rotations (Continued)

- For more precise rotation control, the following numpad keys can be used:
  - **7** rotates about the x screen axis.
  - **8** rotates about the y screen axis.
  - **9** rotates about the z screen axis.
  - **Num Lock** rotates about the x body axis.
  - **/** rotates about the y body axis.
  - ***** rotates about the z body axis.
- Additional rotation features can be accessed with the following “hot keys”:
  - **Page Down** used with any of the x-y-z body/screen keys results in a 90 deg. rotation about the corresponding coordinate direction.
  - **Page Up** used with any model manipulation, whether by mouse or keyboard, provides 3x acceleration of the operation.
  - **+** and **–** numpad keys reverse the direction of a keyboard model rotation.
  - **Alt** used with a numpad model rotation gives a single increment (3 deg.) rotation.

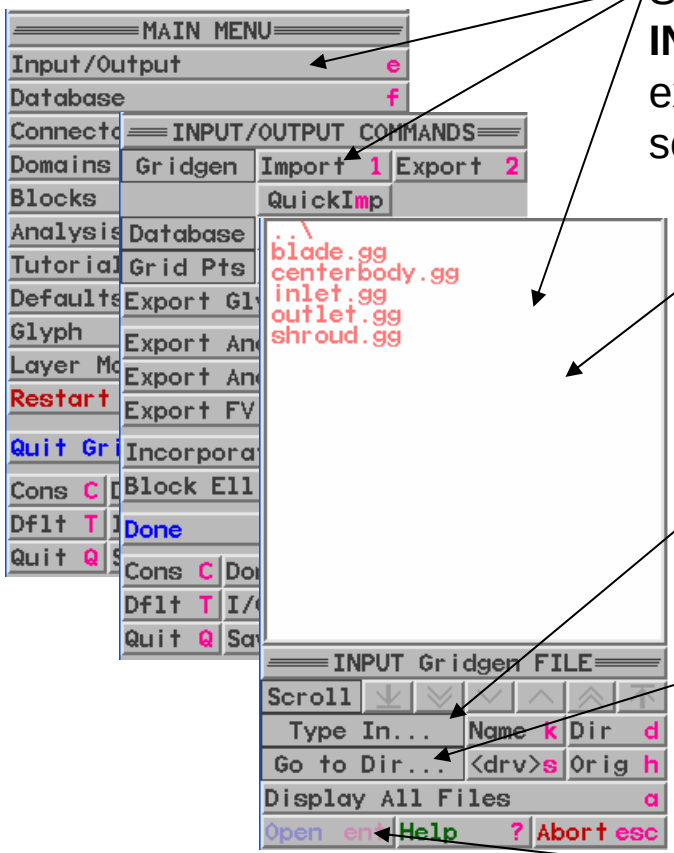
Commonly Used Hot Keys

HOT KEY	FUNCTION
Enter	Any menu button color coded Blue (Done, Save, Open, etc.)
Esc	Any menu button color coded Red (Quit, Do Not Save, Exit, etc.)
?	Online Help
x	Toggle pickable entities under cursor
r	Reset image to original view
g	Grab (save) 1st view orientation
G	Grab (save) 2nd view orientation
p	Put (restore) 1st view orientation
P	Put (restore) 2nd view orientation
F5	Zoom-in
F6	Zoom-out
z	Initiate Zoom-In box
^z	Initiate Zoom-Out box
Z	Undo last zoom command
Ctrl-Pick	Pick new rotation point

HOT KEY	FUNCTION
PageDown	90° rotations (used with rotation keys)
Alt	Single increment rotation (used with rotation keys)
NumLock	x body-axis rotation
Pad /	y body-axis rotation
Pad *	z body-axis rotation
Pad -	Reverse direction of rotation
Pad 7	x screen-axis rotation
Pad 8	y screen-axis rotation
Pad 9	z screen axis rotation
Pad 1	Restrict 3D cursor to x body-axis motion (u when database constrained)
Pad 2	Restrict 3D cursor to y body-axis motion (v when database constrained)
Pad 3	Restrict 3D cursor to z body-axis motion
Shift-u	Zoom to screen extents
Shift-Pick	Display point position and delta

Note: Hot keys for Gridgen commands are shown as pink characters on the corresponding GUI buttons.

File Browsing



Selecting **Input/Output** from the **MAIN MENU** opens the **INPUT/OUTPUT COMMANDS** submenu from which (in this example) the **INPUT Gridgen FILE** submenu has been selected.

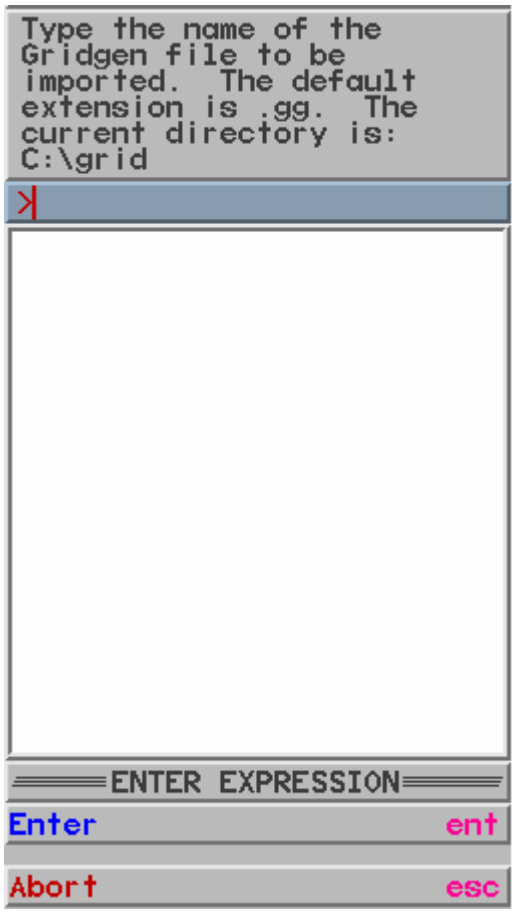
The Browser Window shows the appropriate contents of the current working directory. The mouse can be used to scroll through and select from list.

Selecting **Type in… Name** allows you to type in a filename for import or export, while **Type in… Dir** lets a new working directory to be set.

Selecting **Go to Dir… <drv>** (**/** in Unix) will provide list of available drives to change to as current working directory, while **Go to Dir… Orig** changes the working directory back to the original directory where Gridgen was opened.

Once file has been selected, **Open** is clicked.

File Browsing (Continued)

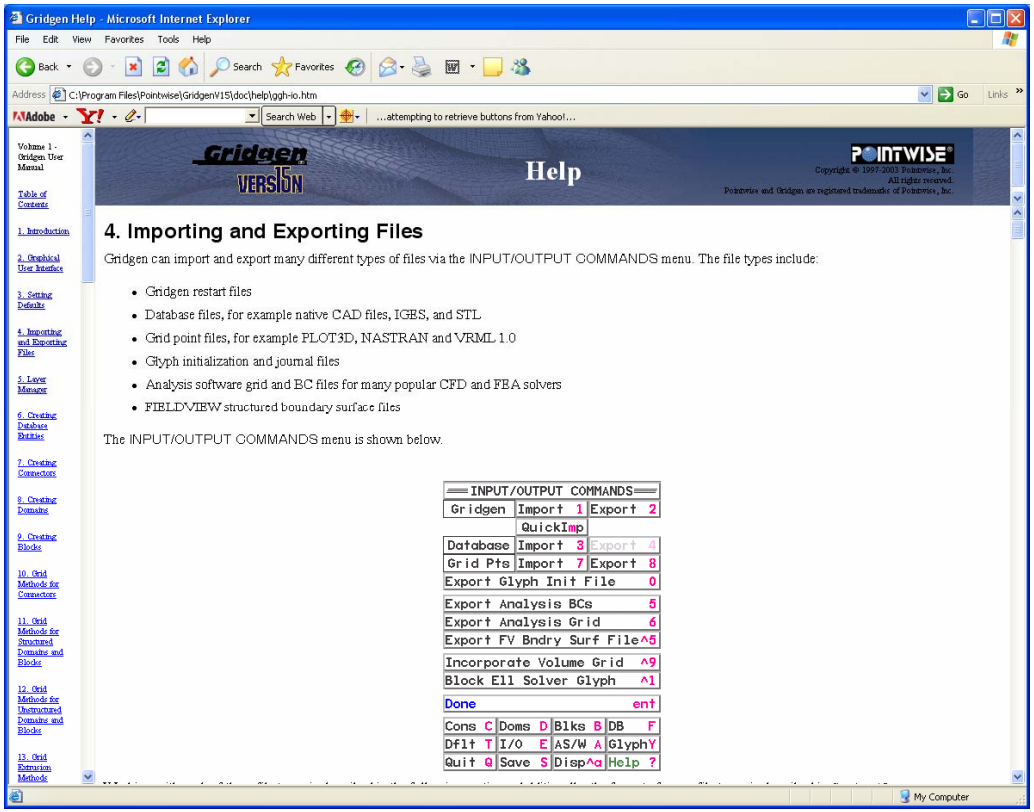


- Upon entering the **Type in… Name** mode, the text input window will echo the filename entered via the keyboard. Once the name is entered, the **Enter** key on the keyboard or **Open** button in the GUI must be clicked.
- The **esc** key can be used to exit from text input.
- No other commands or manipulations are available while in this mode.
- Glyph expressions are valid.

Gridgen File Names

- Gridgen Restart Files (.gg) contain entire grid models from control point definitions through boundary condition settings. They only reference database information.
- Composite Database Files (.dba) store, in Gridgen's native database model format, data from all database sources, as well as database entity display attributes and layer information.

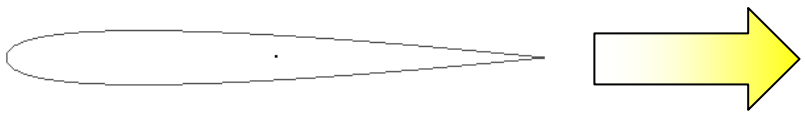
Gridgen Help

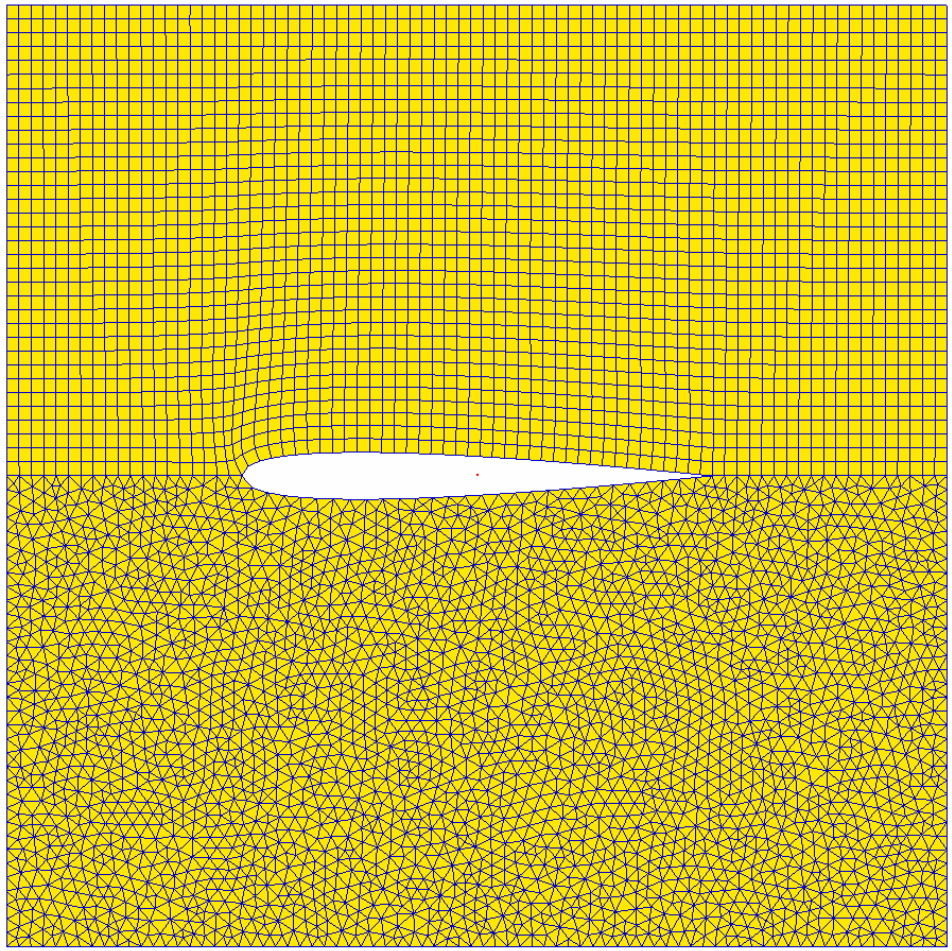


- Gridgen's html-based help can be accessed from virtually any menu within the program by either mouse clicking on the **Help** button or entering **?** From the keyboard.
- This help documentation is equivalent to the User Manual and automatically opens to the pertinent section relating to the operation or mode currently in use within Gridgen. For example, the window at the left opens when help is called upon from the opening I/O menu.

Building 2D Structured Grids

Beginning with the airfoil database geometry below, the hybrid grid shown at the right will be constructed using Gridgen, starting with the 2D structured block at the top.





The “Bottom-Up”Gridgen Process

Select Analysis Software and Mesh Dimensionality

⇩

Import and Manipulate Database Geometry

⇩

Create Segments and Connectors

⇩

Dimension Connectors and Distribute Grid Points

⇩

Assemble Domains

⇩

Assemble Blocks

⇩

Setup CFD Analysis Boundaries

⇩

Export and Save Files

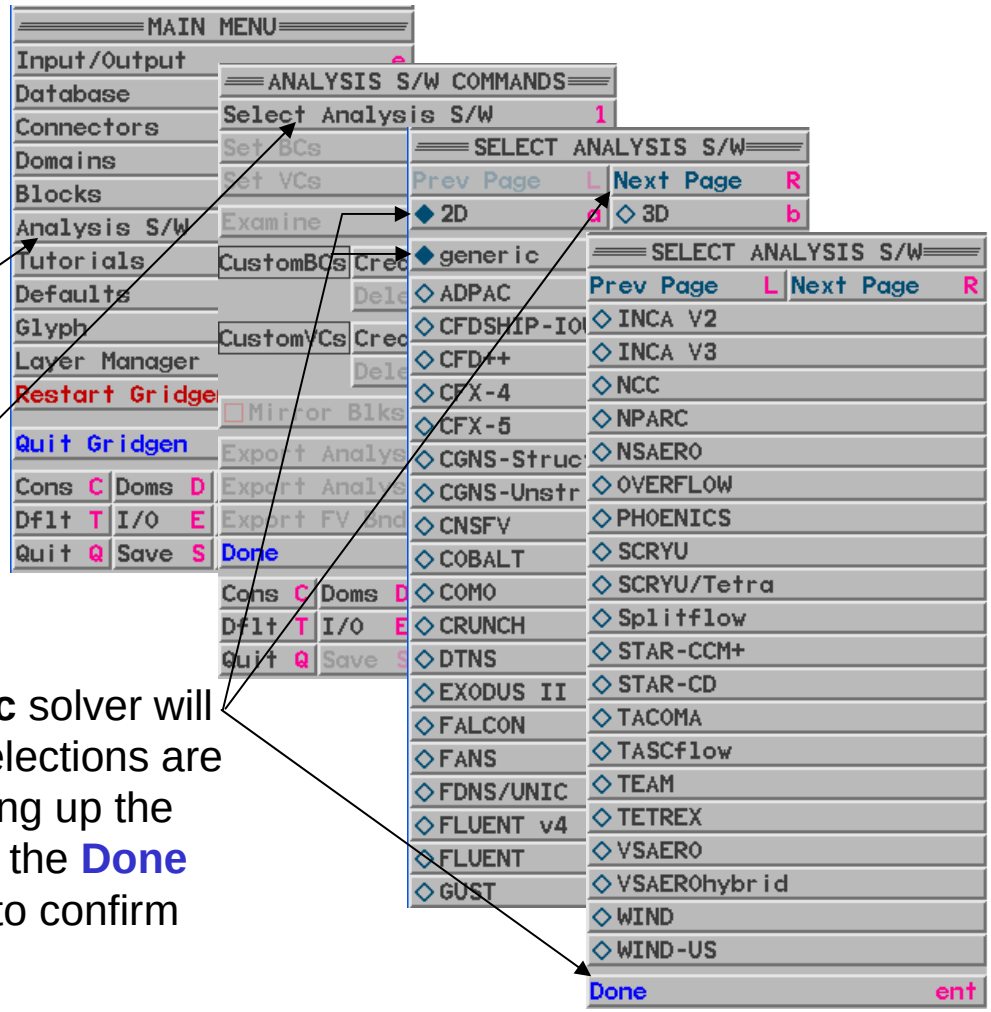
Select Analysis Software

The first step in any Gridgen project is the selection of the CFD analysis software and the dimensionality (2D vs. 3D) of the mesh being constructed.

In the **MAIN MENU**, **Analysis S/W** is chosen to bring up the corresponding commands menu.

Select Analysis S/W is then clicked, which brings up the fairly lengthy list of supported CFD packages.

For this example, **2D** and the default **generic** solver will be selected via mouse clicks. Once these selections are made, the **Next Page** button is clicked to bring up the second half of the supported software list so the **Done** button at the end of the list can be selected to confirm the choices.



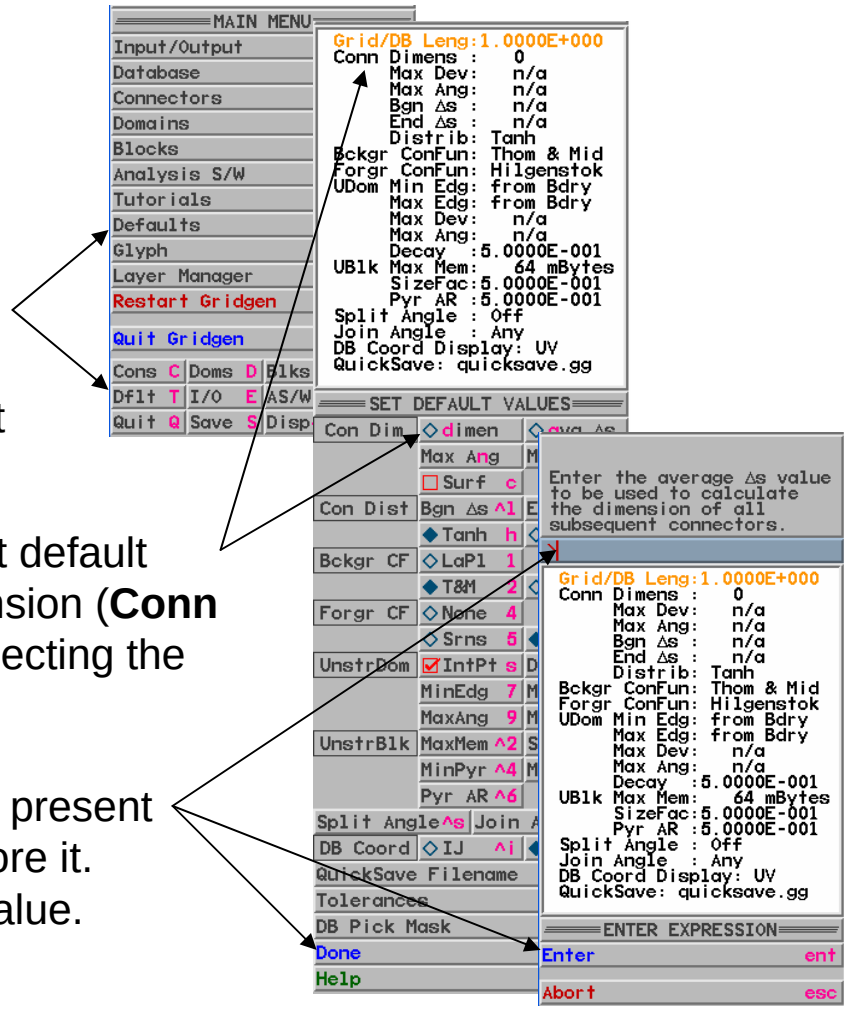
Set Defaults

Although not essential, setting default values for some items, such as connector dimension, can save time during the mesh construction process.

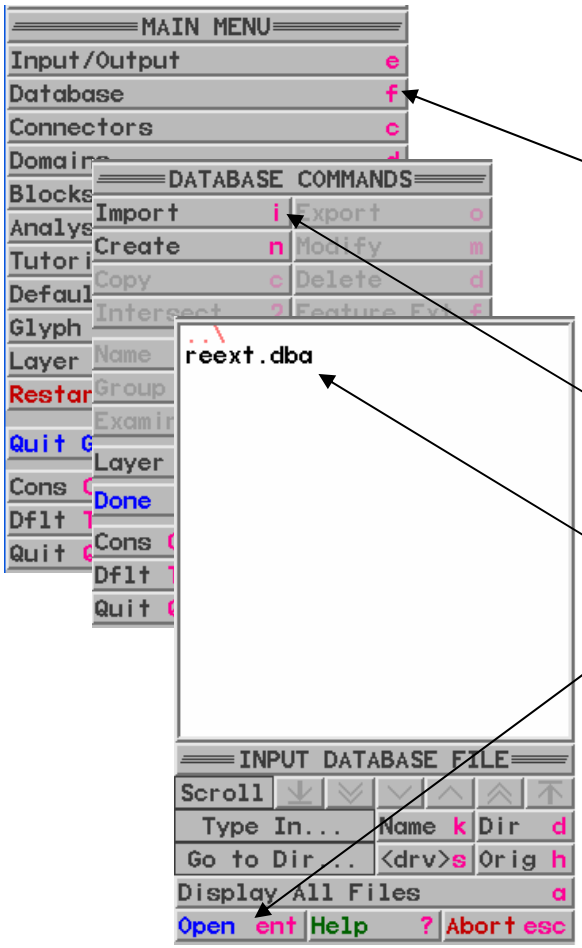
In the **MAIN MENU**, **Defaults** is chosen to bring up the defaults menu. Note: Another way to bring up the defaults menu is to select the **Dflt** quick key.

The blackboard window shows the current default settings, e.g., the default connector dimension (**Conn Dimens**) is 0. This can be changed by selecting the appropriate button below.

This activates the text input window. For the present example, type in 36 and then hit **Enter** to store it. The blackboard window will show the new value. Click **Done** to finish setting defaults.



Import Database Geometry



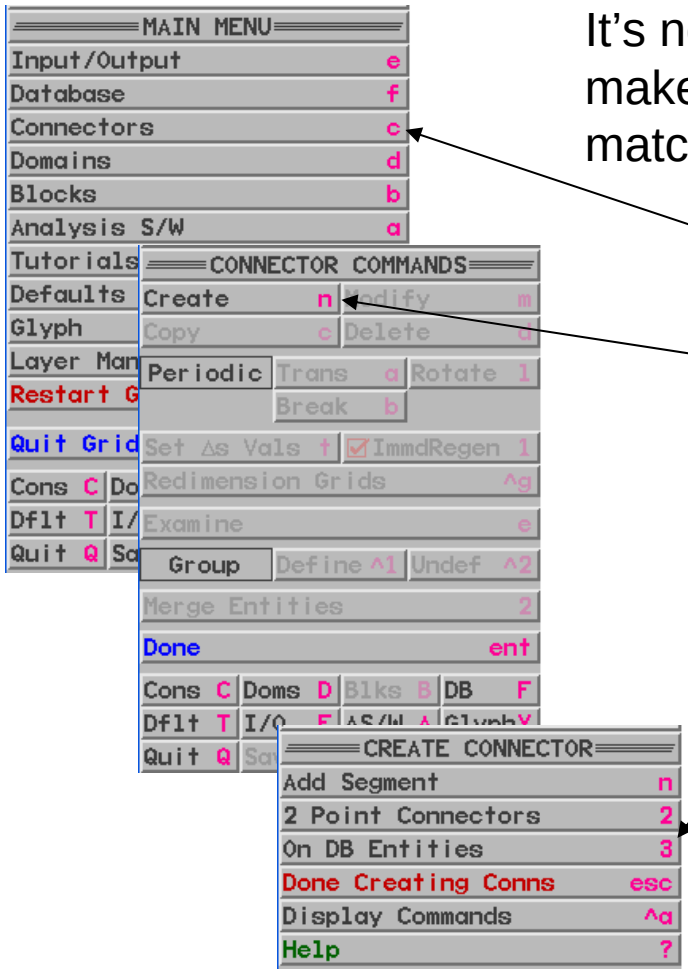
Next, the database file containing the airfoil geometry will be imported.

In the **MAIN MENU**, **Database** is chosen to bring up the corresponding commands menu. Note: As an alternative, the **f** hot key on the keyboard could be used.

Import is then selected to bring up the **INPUT DATABASE FILE** submenu.

The browser window shows the available database (.dba) files in the current working directory. The desired file can then be picked with the mouse (changing the name from pink to black in color). Click on **Open** to read the file into Gridgen and then **Done** in the **DATABASE COMMANDS** menu to return to the **MAIN MENU**.

Create Connectors



It's now possible to start building the connectors that will make up the edges of the grid. The first connector will match the upper contour of the airfoil.

In the **MAIN MENU**, **Connectors** is chosen to bring up the corresponding commands menu.

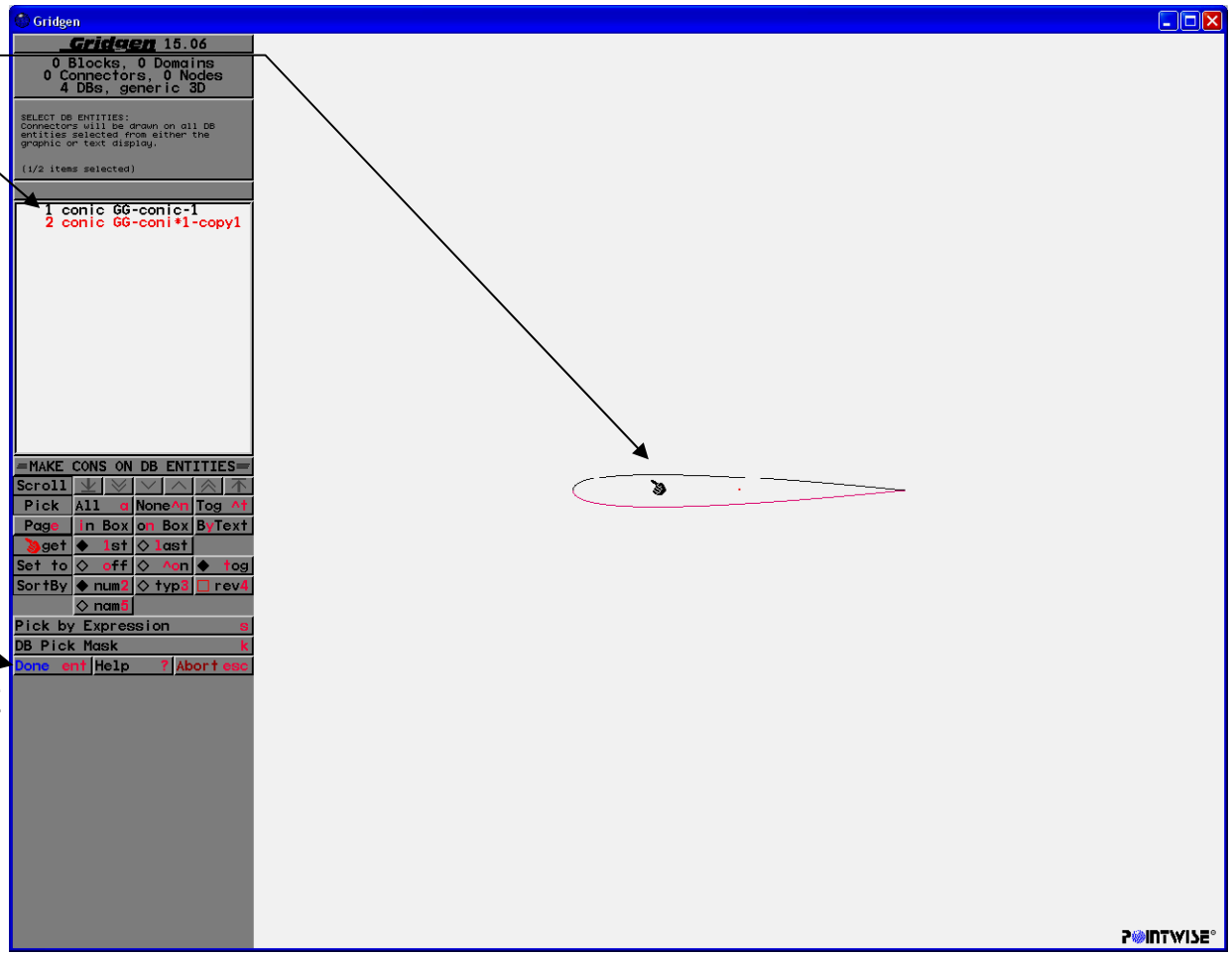
Since there are no connectors yet, only the **Create** command is available in the **CONNECTOR COMMANDS** menu. Select this to open the **Create Connector** submenu

There are three ways to build a connector: individual curve segments can be created and linked together, two endpoints can be chosen to form a connector between them, or a connector can be laid down upon a database entity. This last option will be picked here.

Create Connectors (Continued)

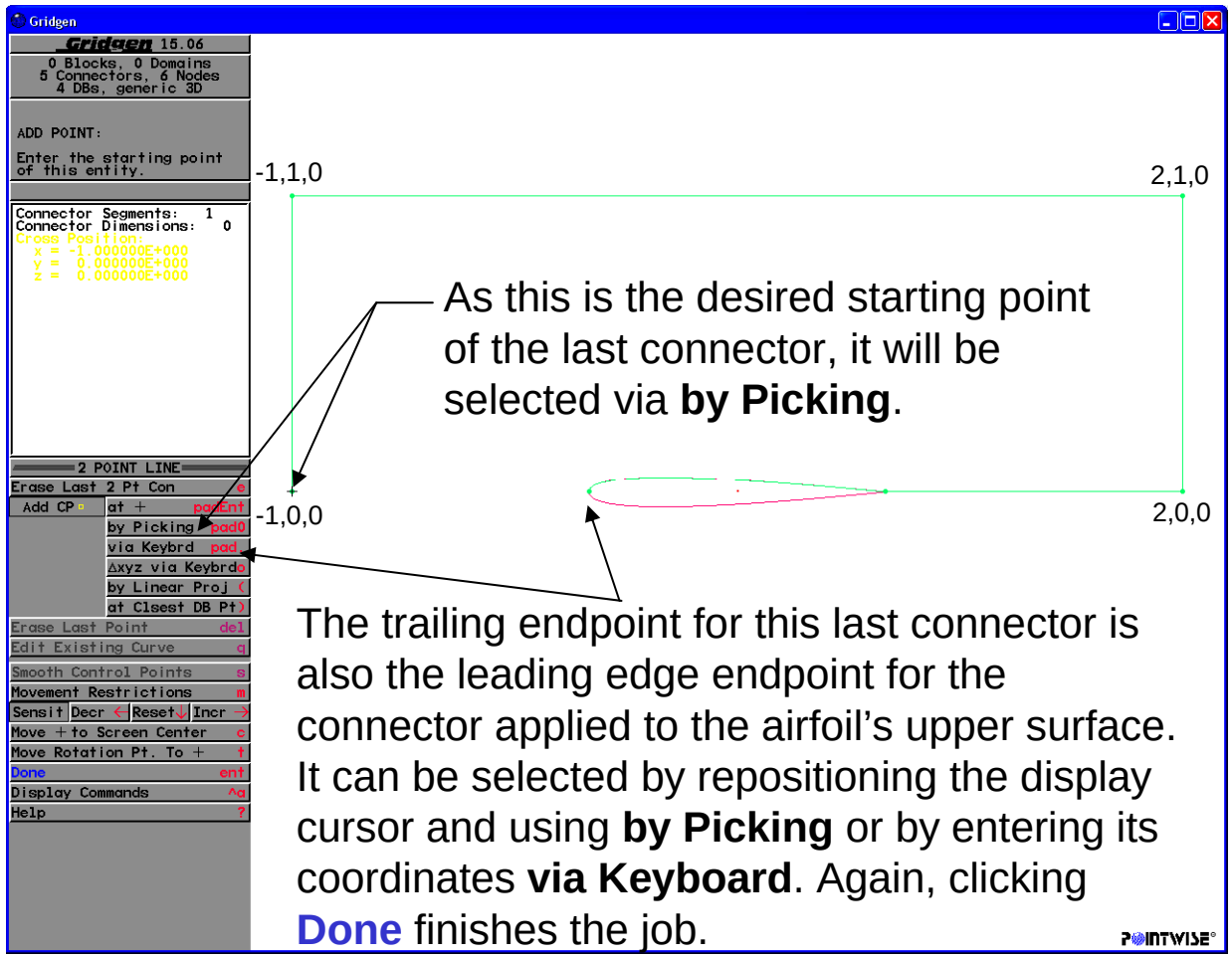
The blackboard window shows the available database entities. The mouse can be used to pick entities from this list or directly from the display window. Once selected, the table entry and the connector in the display change from pink to black in color.

Clicking on **Done** completes the process. Because the default dimension has been set, the new connector appears in pale green (if undimensioned, it would have been bright green).



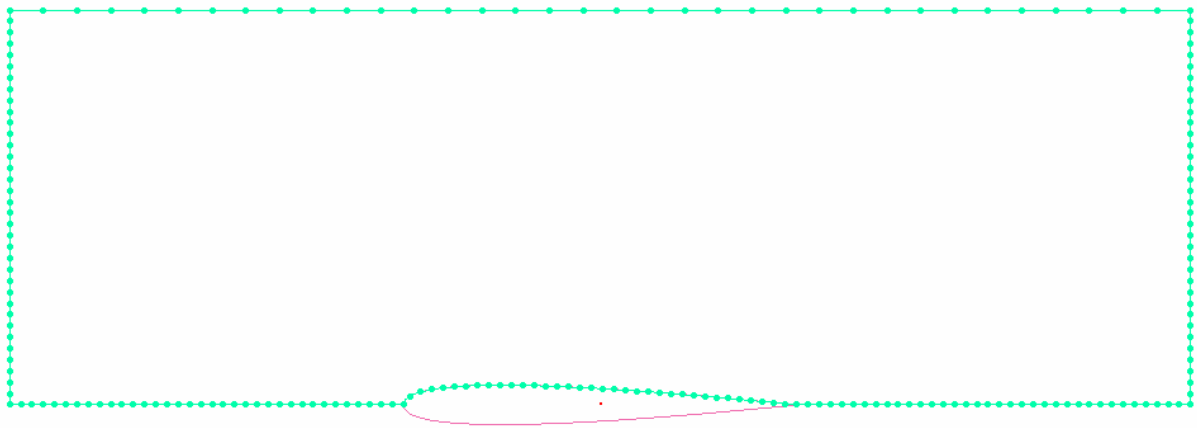
Create Connectors (Continued)

The remaining connectors needed to form the edges of the mesh will be created with the **2 Point Connectors** option. Each CP can be selected from the display via the mouse or entered via the keyboard. For each new connector, Gridgen initially positions the display cursor at the last endpoint of the previously defined connector. Here, all but the last connector have been defined, and the display cursor is positioned at that connector's trailing endpoint.



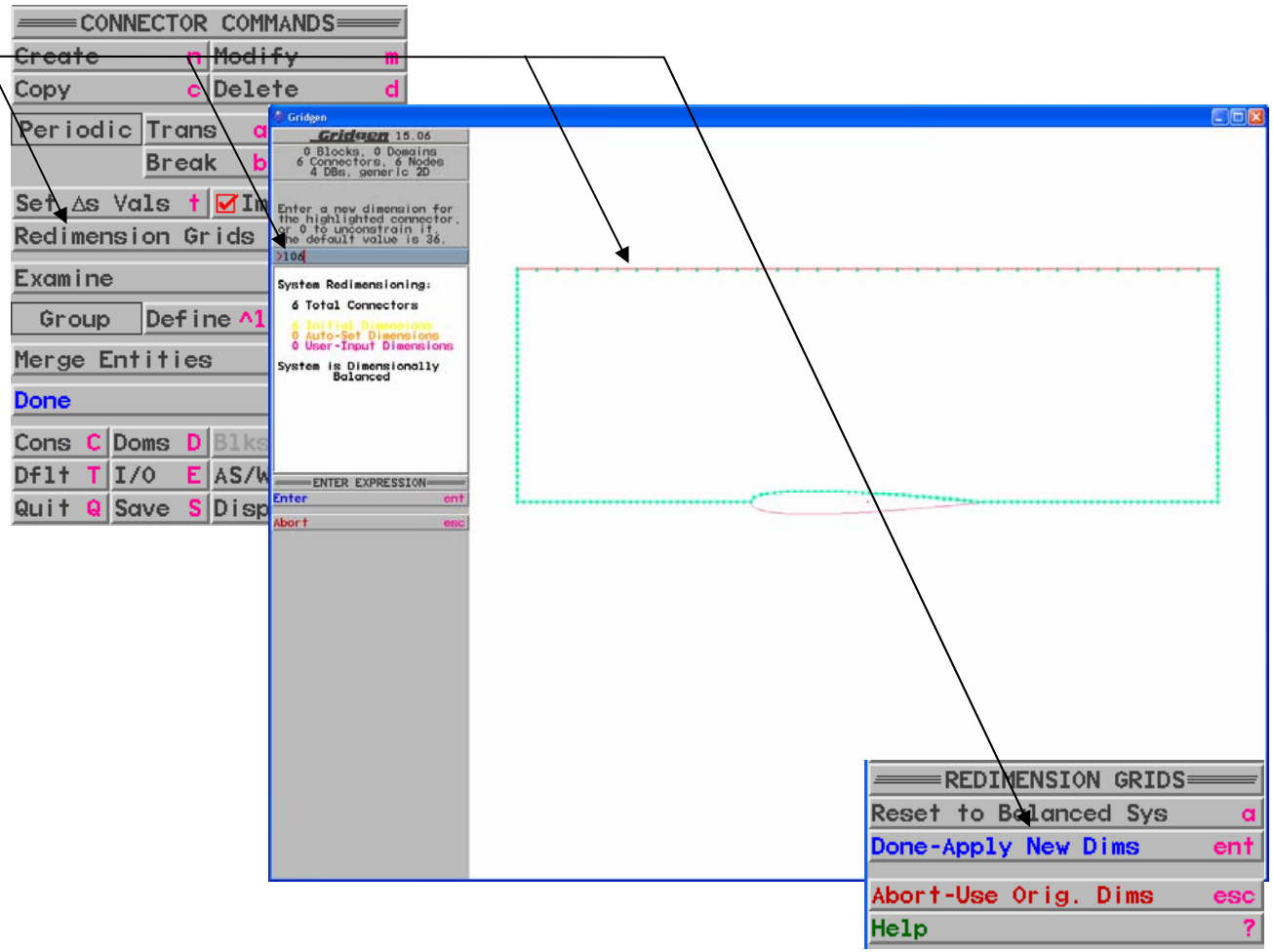
Create Connectors (Continued)

With the last connector completed, click on **Done Creating Conns** in the **CREATE CONNECTOR** menu (or just hit the corresponding hot key, **esc**). Next, to see the grid point distribution, toggle **Con GPs** in the display menu (or hit the hot key combination **^a c**). As shown, it's clear that there are far more grid points along the bottom edge than the top (since there are three connectors below to the one above, but each with the same default dimension). This has to be corrected before the domain can be built.



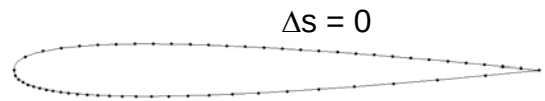
(Re)Dimension Connectors

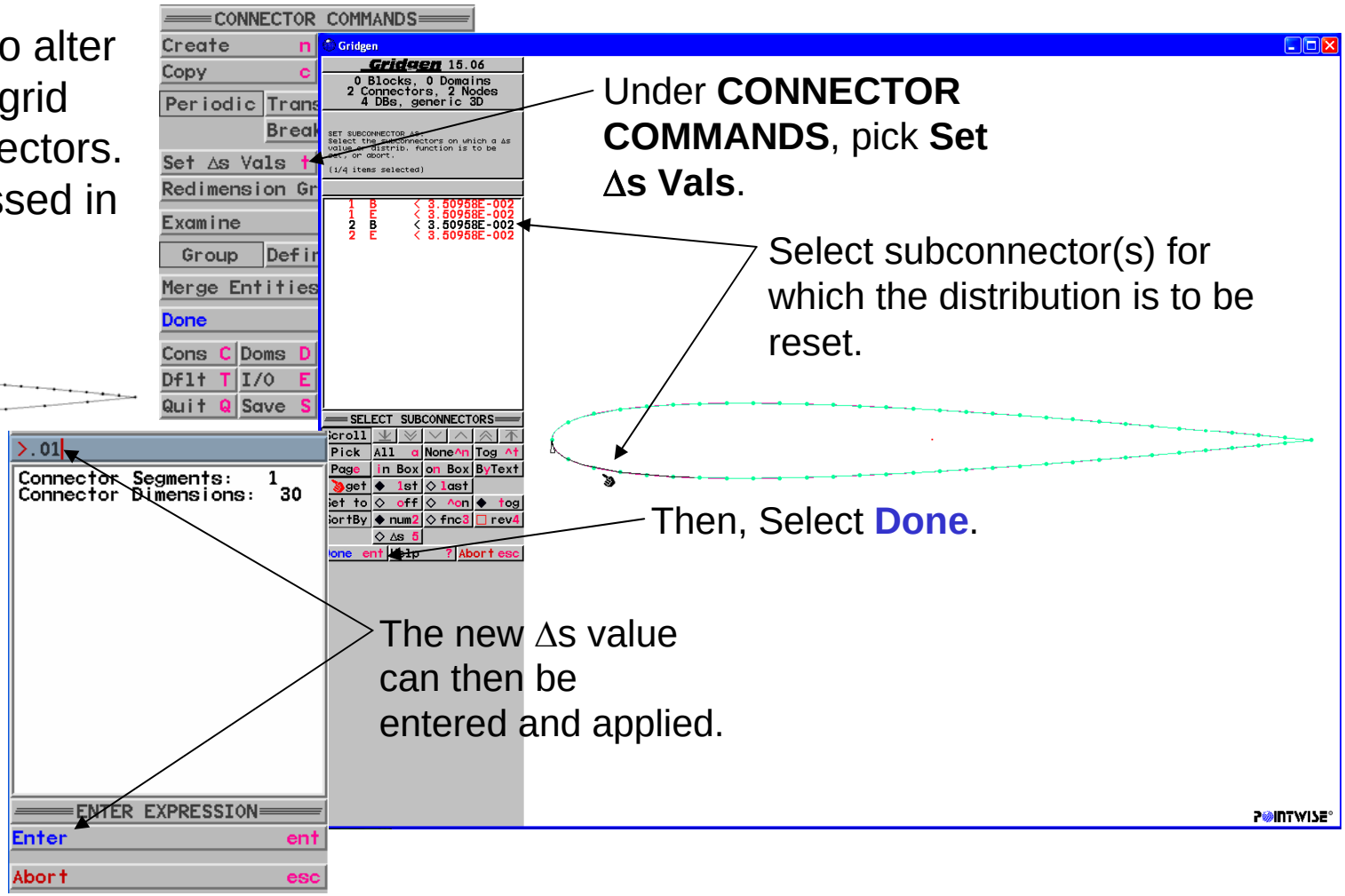
In the **CONNECTOR COMMANDS** menu, select **Redimension Grids**. This will allow the top connector to be picked in the display window via the mouse, and its new dimension to be entered via the keyboard. In this case, 106 is that new dimension (while each of the three bottom connectors has 36 grid points, the center one shares one of its endpoints with each of its neighbors reducing the total for the edge by 2). Once **Enter** has been picked, **Done-Apply New Dims** is clicked to finish.



(Re)Distribute Connectors

It's also possible to alter the distribution of grid points along connectors. This will be discussed in more detail later.





Under **CONNECTOR COMMANDS**, pick **Set Δs Vals**.

Select subconnector(s) for which the distribution is to be reset.

Then, Select **Done**.

The new Δs value can then be entered and applied.

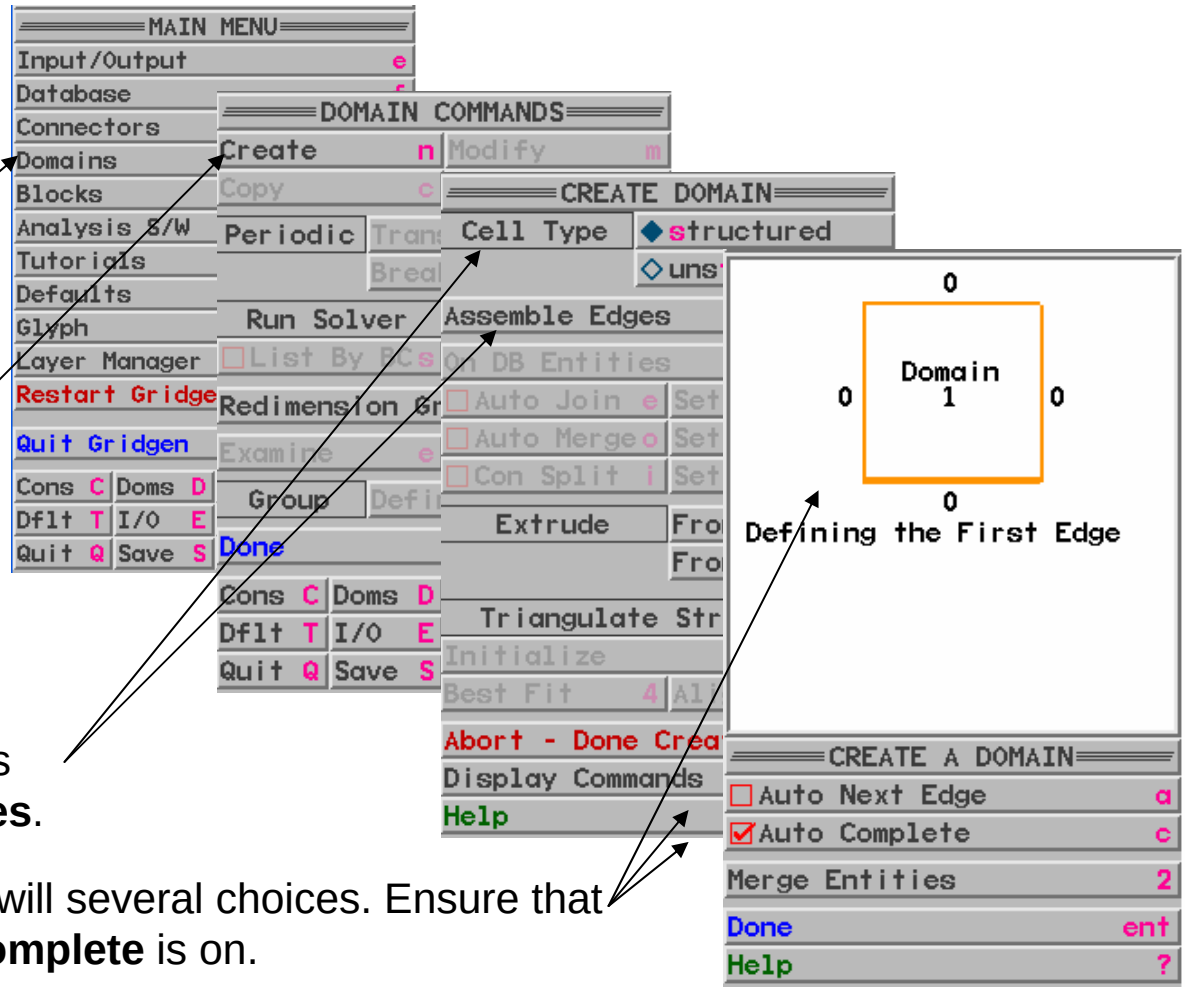
Assemble Domains

All is now ready to assemble the connectors into the four edges that will bound the domain of the mesh.

In the **MAIN MENU**, pick **Domains** to switch to that menu.

Next, select the **Create** command to begin building the new domain.

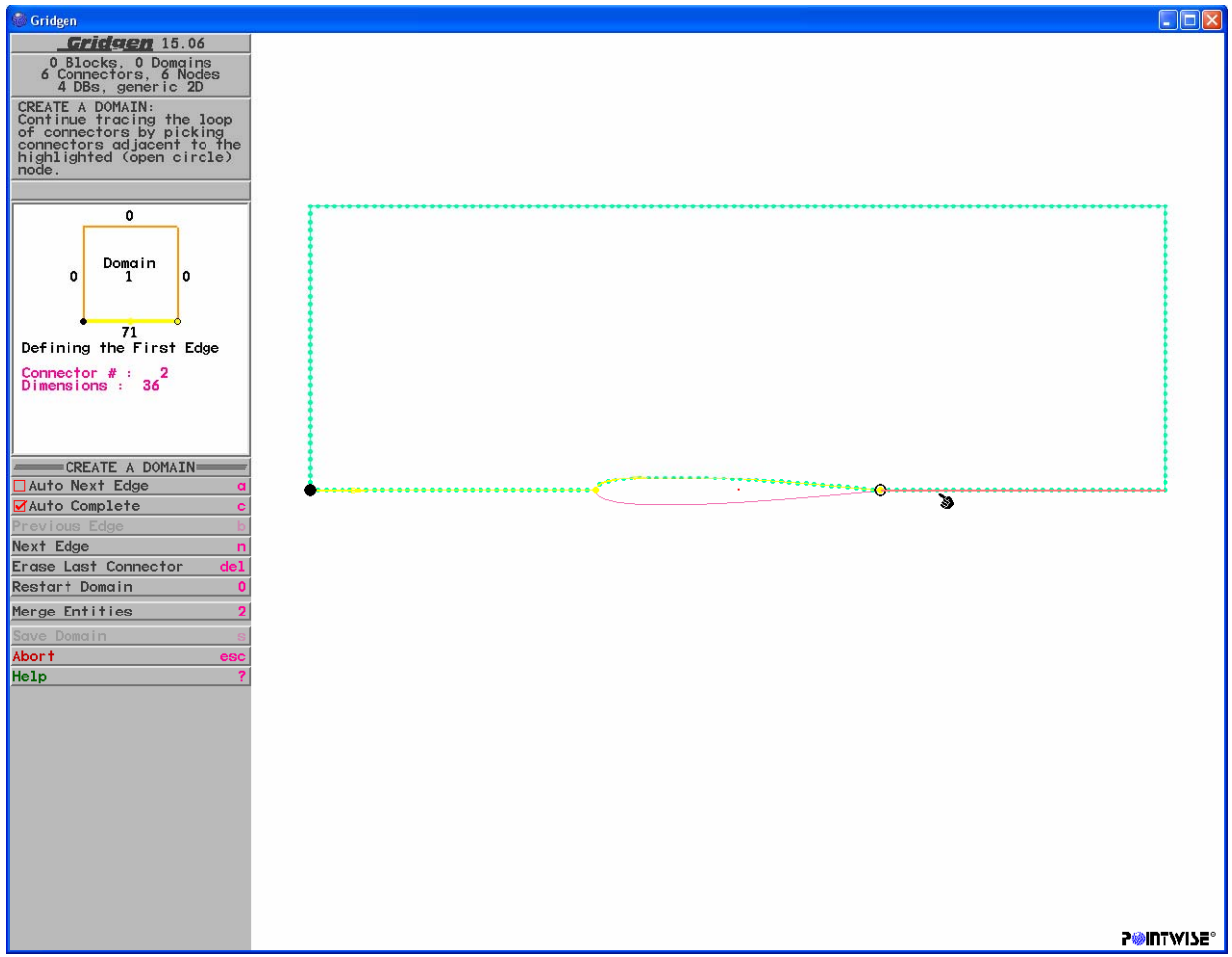
Be sure that **structured** cell type is chosen, then pick **Assemble Edges**.



A domain schematic will appear, as will several choices. Ensure that **Auto Next Edge** is off, and **Auto Complete** is on.

Assemble Domains (Continued)

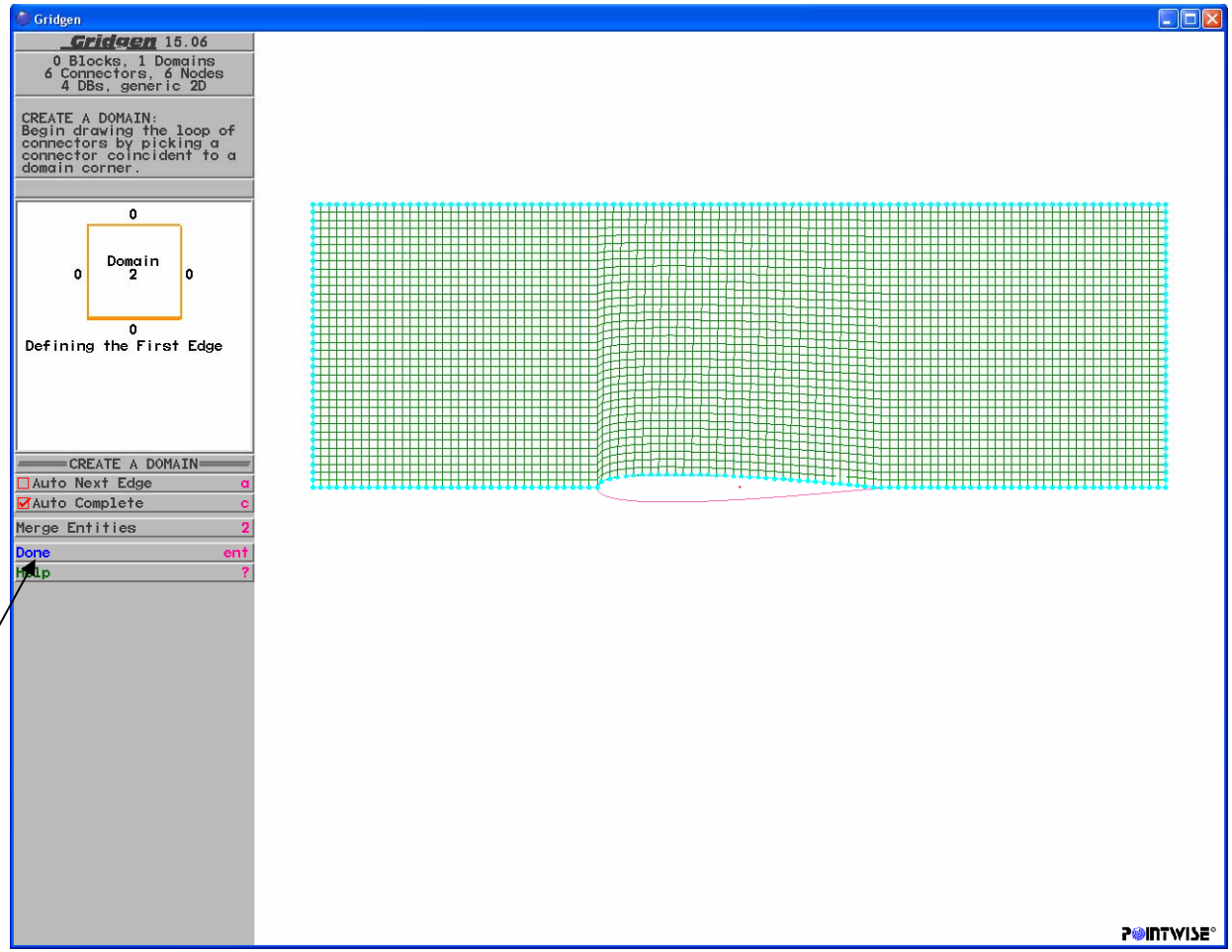
The first edge will consist of the three bottom connectors. The mouse is used to select these in turn from left to right – the picture here shows the scene after the first two have been picked with the mouse cursor poised to select the third. Once this connector has been added, **Next Edge** is selected to tell Gridgen the edge is complete. The mouse is then used to select the connector on the right side of the domain to be. When **Next Edge** is again selected after that, the domain will auto complete itself.



Assemble Domains (Continued)

This is the domain that results. Note that the mesh in the vicinity of the leading edge of the airfoil is a bit rough looking. This can be corrected with a little smoothing done with Gridgen's structured solver.

To get back to the **DOMAIN COMMANDS** menu to do this, select **Done**.

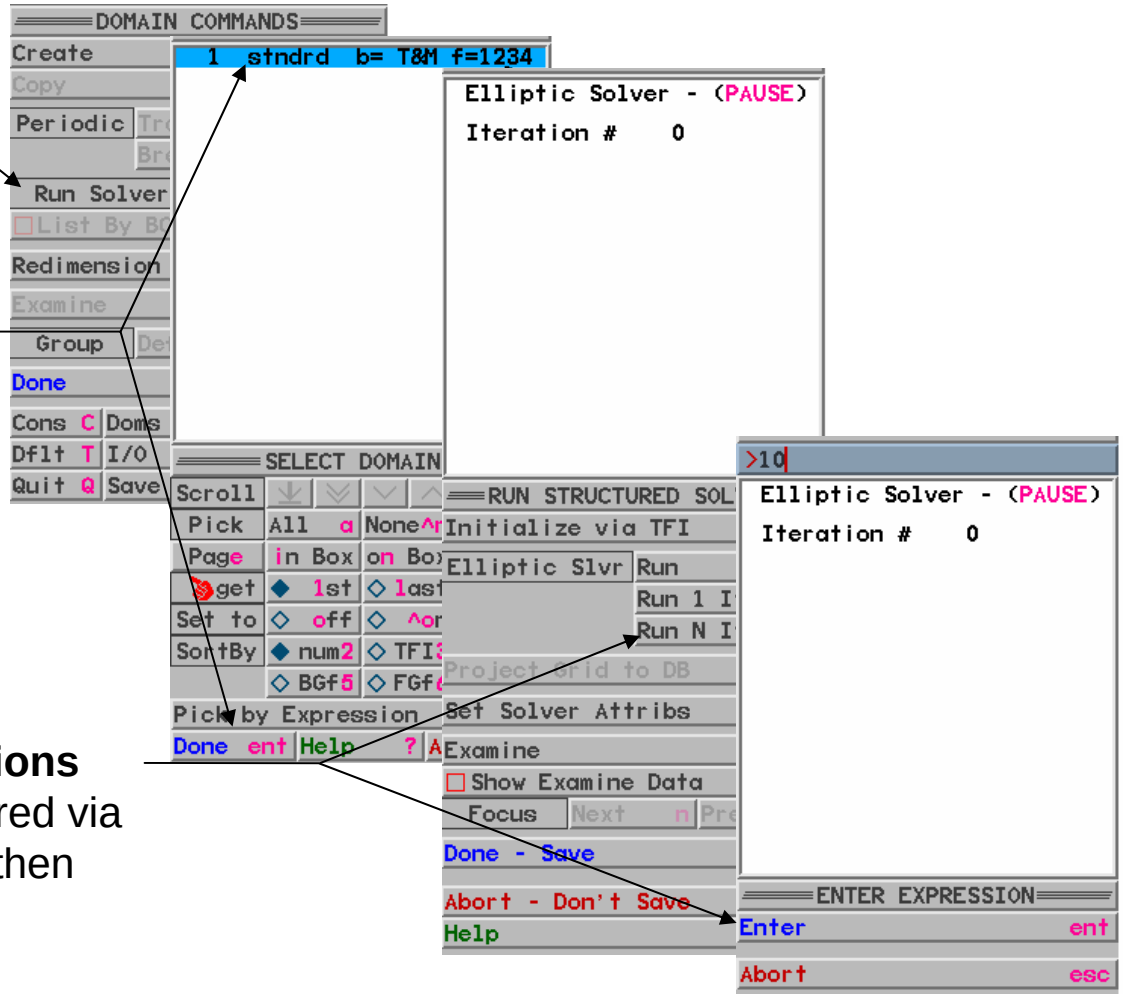


The Structured Solver

To run the solver, first choose **Run Solver Structured** from the **DOMAIN COMMANDS** menu.

The domain is next picked – in this case by highlighting the entry in the browser window. Once **Done** is selected, the parameters for the elliptic solver can be selected.

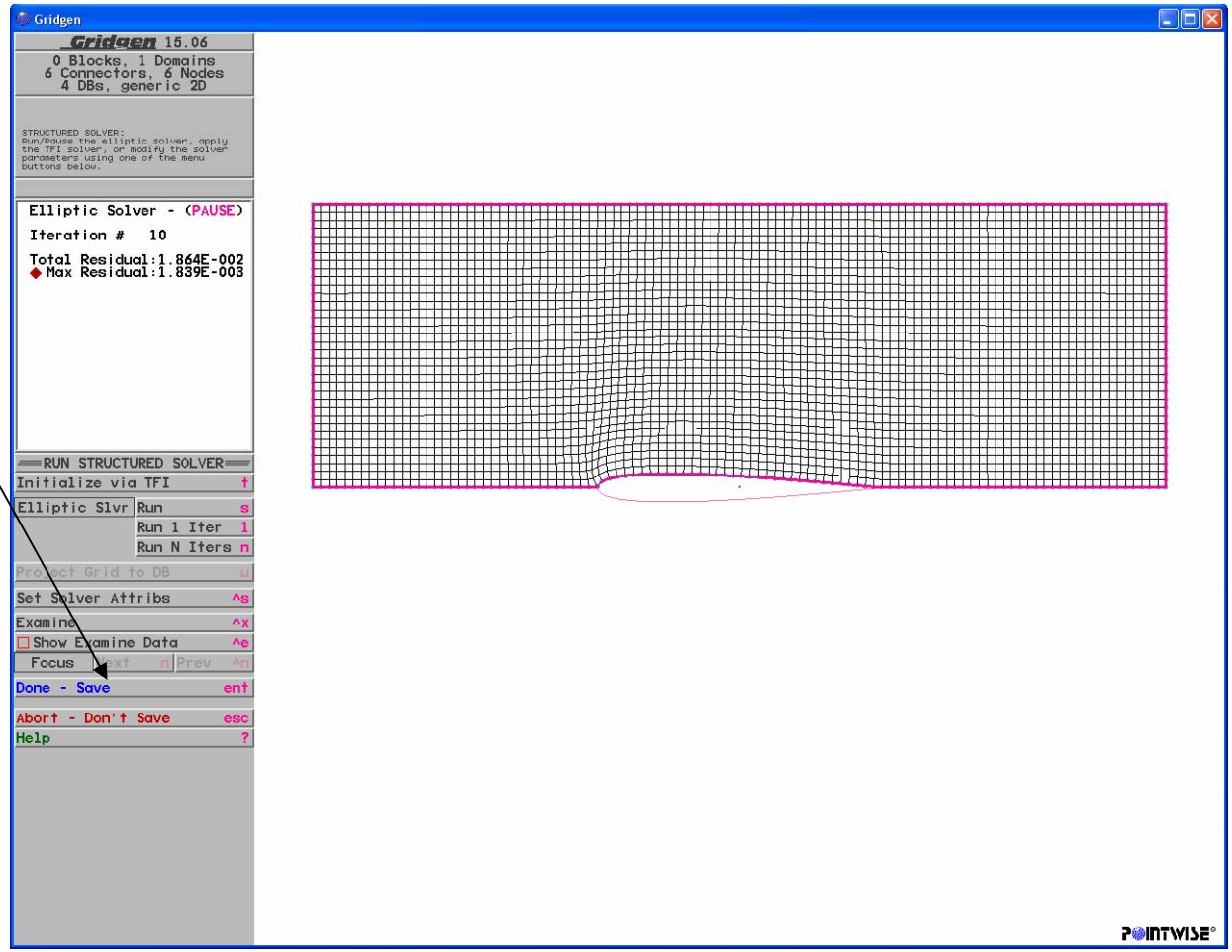
In this instance, the **Run N Iterations** button is selected, and 10 is entered via the keyboard. Clicking **Enter** will then start the solver.



The Structured Solver (Continued)

The mesh lines will shift as the solver runs, ending with the smoothed mesh as shown here.

Done – Save is selected to save the changes.



Assemble Blocks

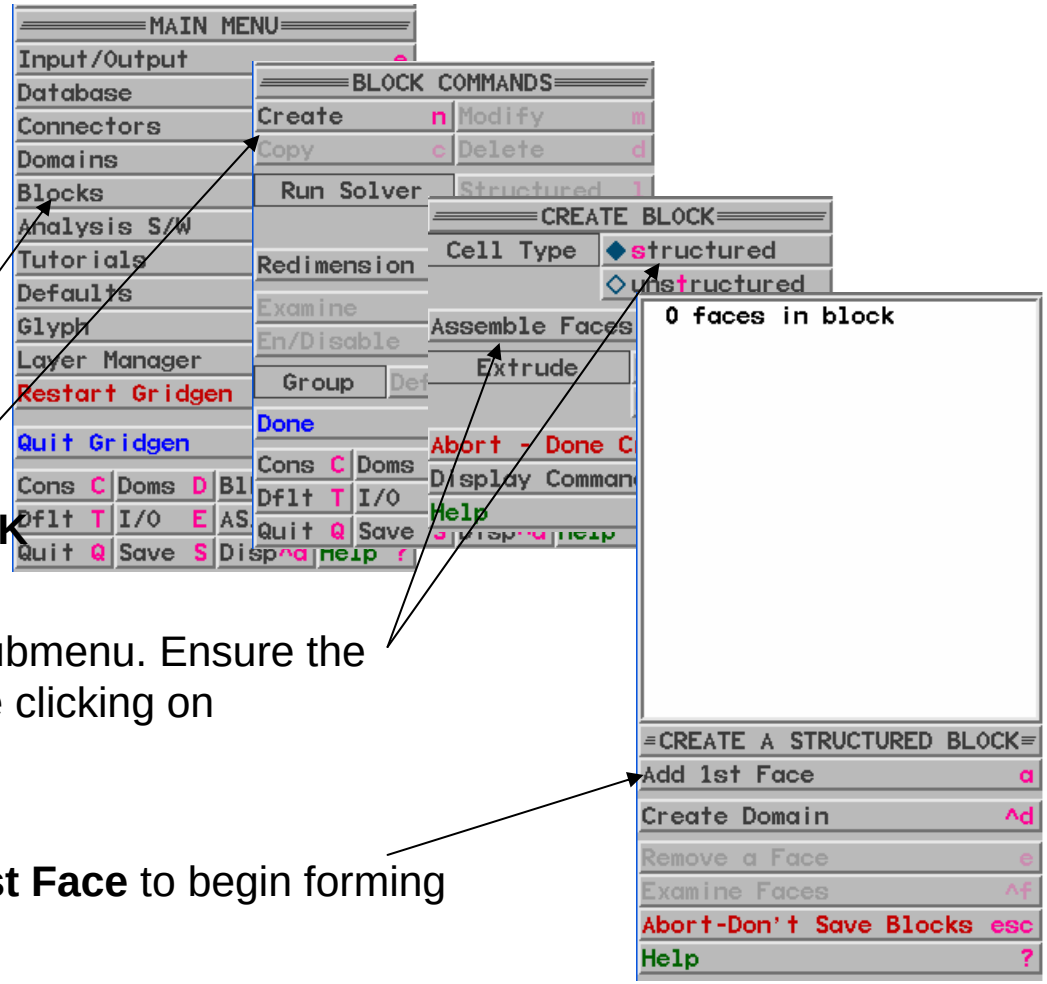
The grid block is next assembled from the domain, in much the same way as a domain is assembled from its connector edges.

From the **MAIN MENU**, **Blocks** is selected.

Create is then selected from the **BLOCK COMMANDS** menu.

This opens the **CREATE BLOCK** submenu. Ensure the structured cell type is toggled before clicking on **Assemble Faces**.

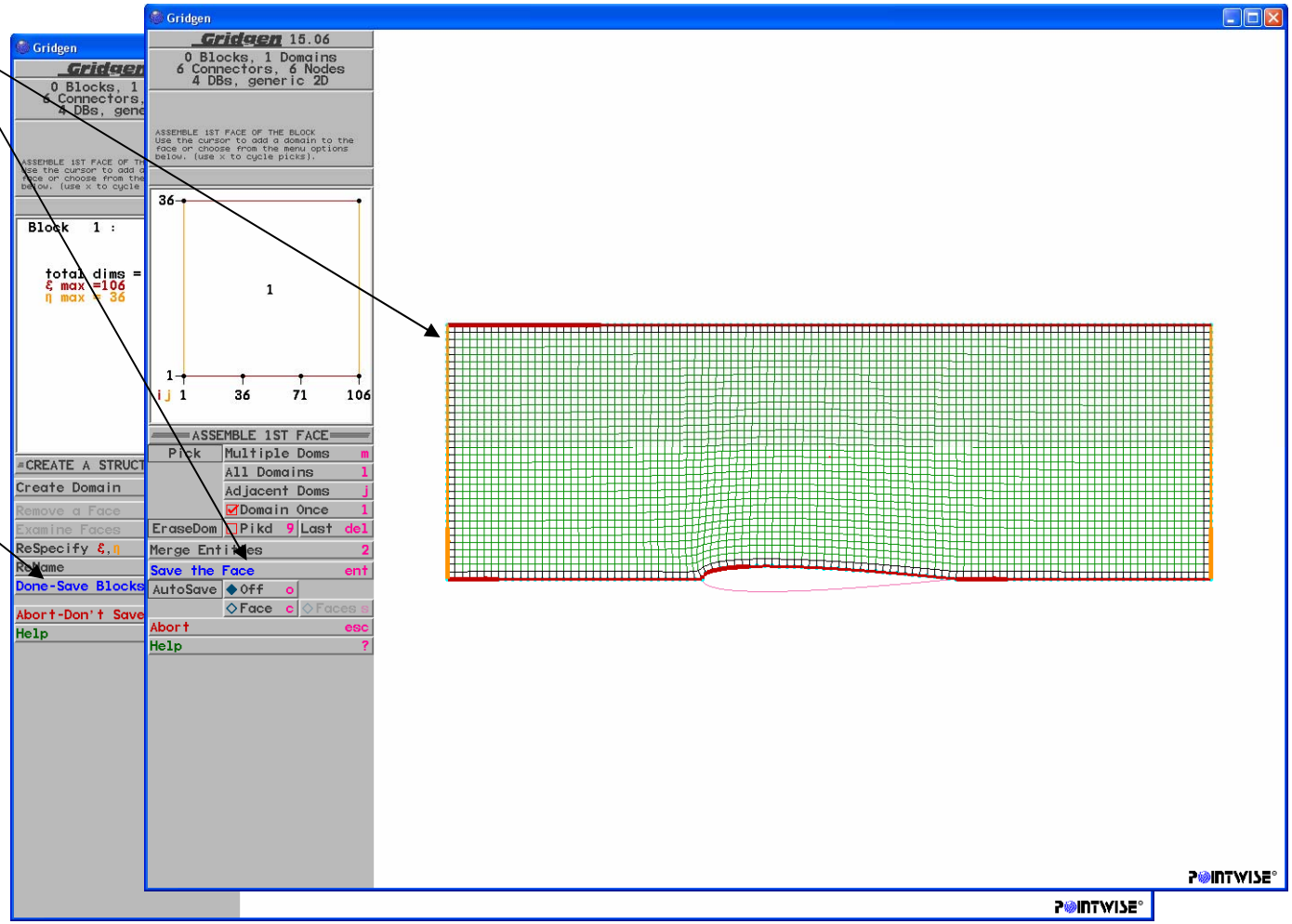
The next step is to pick **Add 1st Face** to begin forming the block.



Assemble Blocks (Continued)

The face is selected by mouse clicking on its image in the display window. Since this is a 2D block, once the **Save the Face** button is pressed, the block is complete.

Press **Done – Save Blocks** to finish this step.

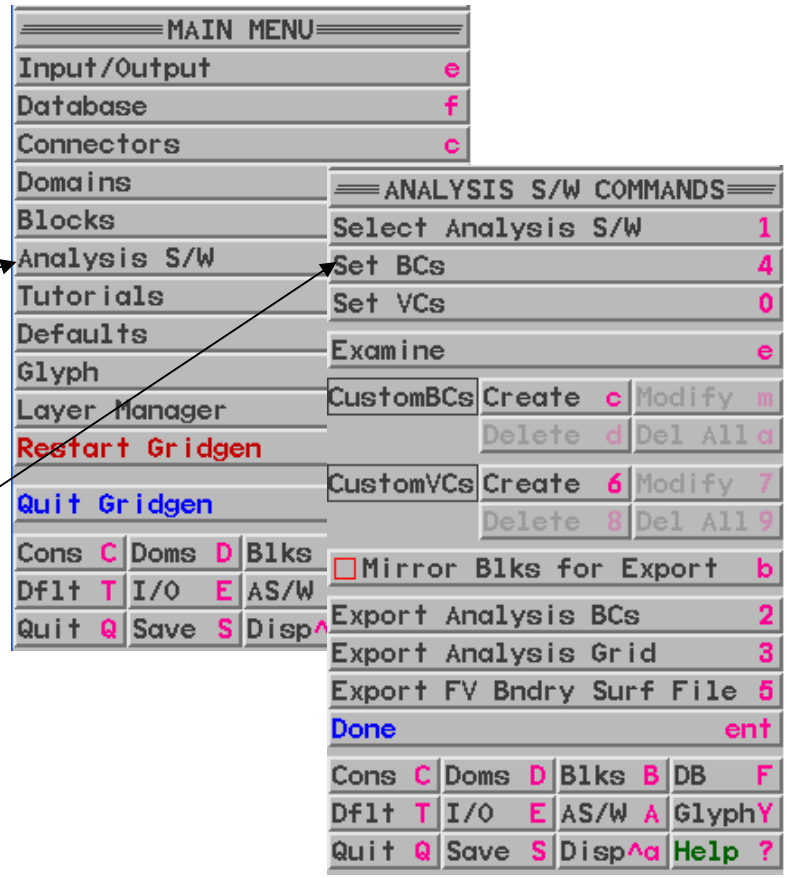


Set Analysis Boundaries

The final step in generating the mesh is setting up the boundaries for the CFD software.

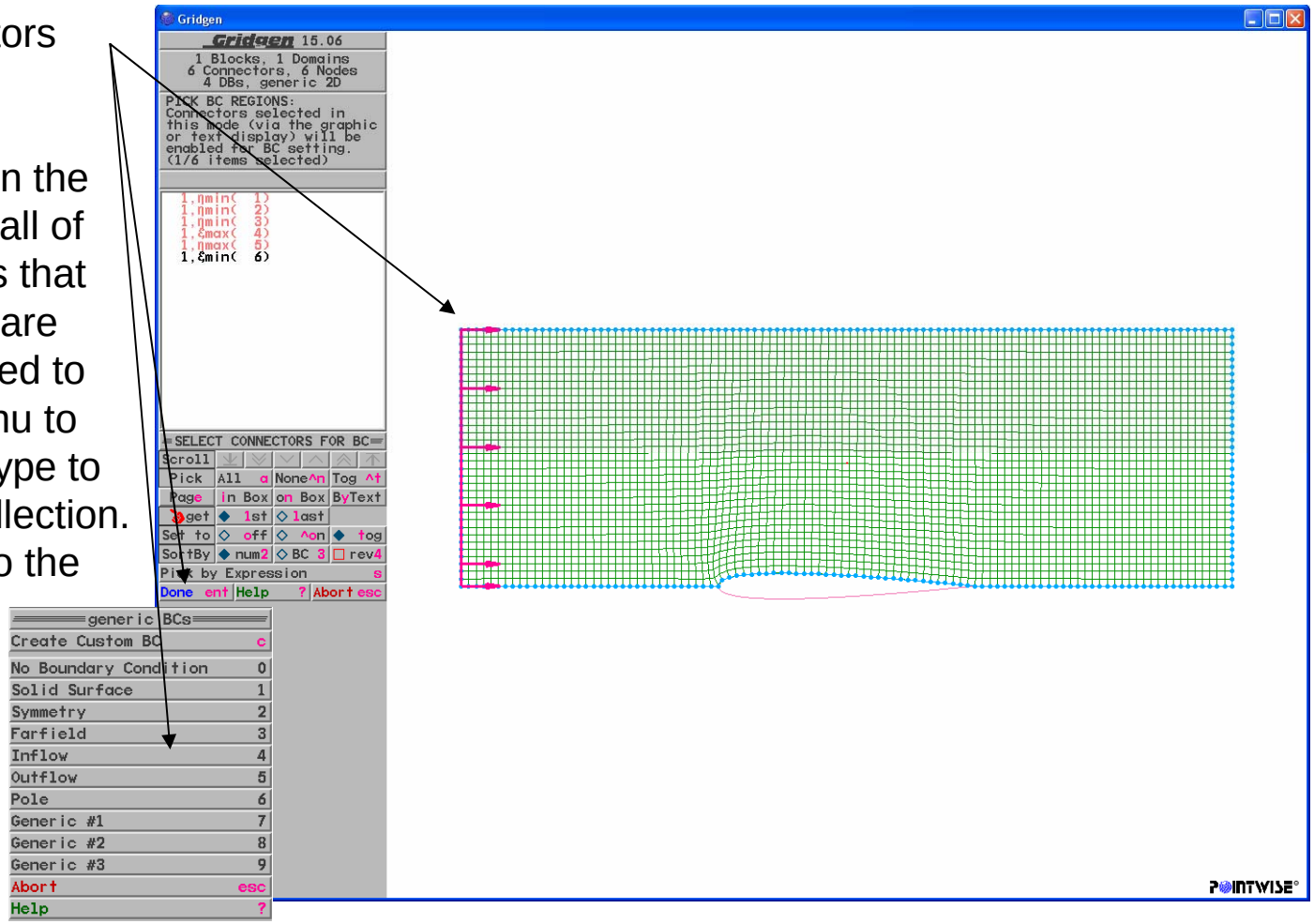
From the **MAIN MENU**, the **Analysis S/W** submenu is selected.

Set BCs is then picked.



Set Analysis Boundaries (Continued)

The individual connectors that will make up each boundary can then be highlighted by mouse in the display window. Once all of the desired connectors that will make a boundary are selected, **Done** is picked to bring up a second menu to specify the boundary type to be assigned to that collection. In this case, the inlet to the flowfield is being selected. The other boundaries are specified in the same manner.



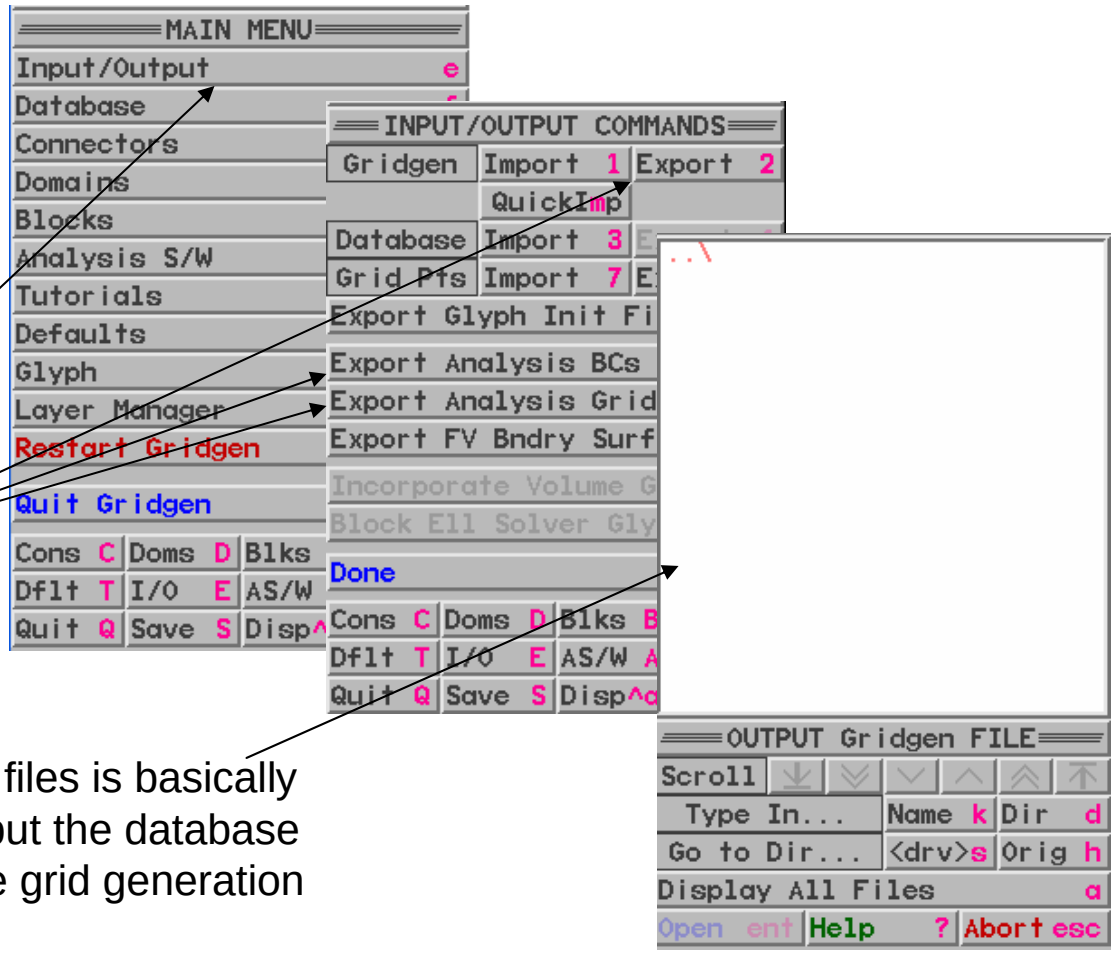
Export and Save Files

The mesh and boundary conditions are now ready to be exported for use with the CFD software.

From the **MAIN MENU**, **Input/Output** is selected.

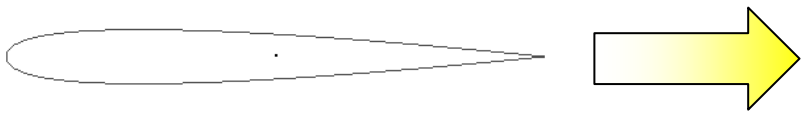
From here, the individual files can be exported.

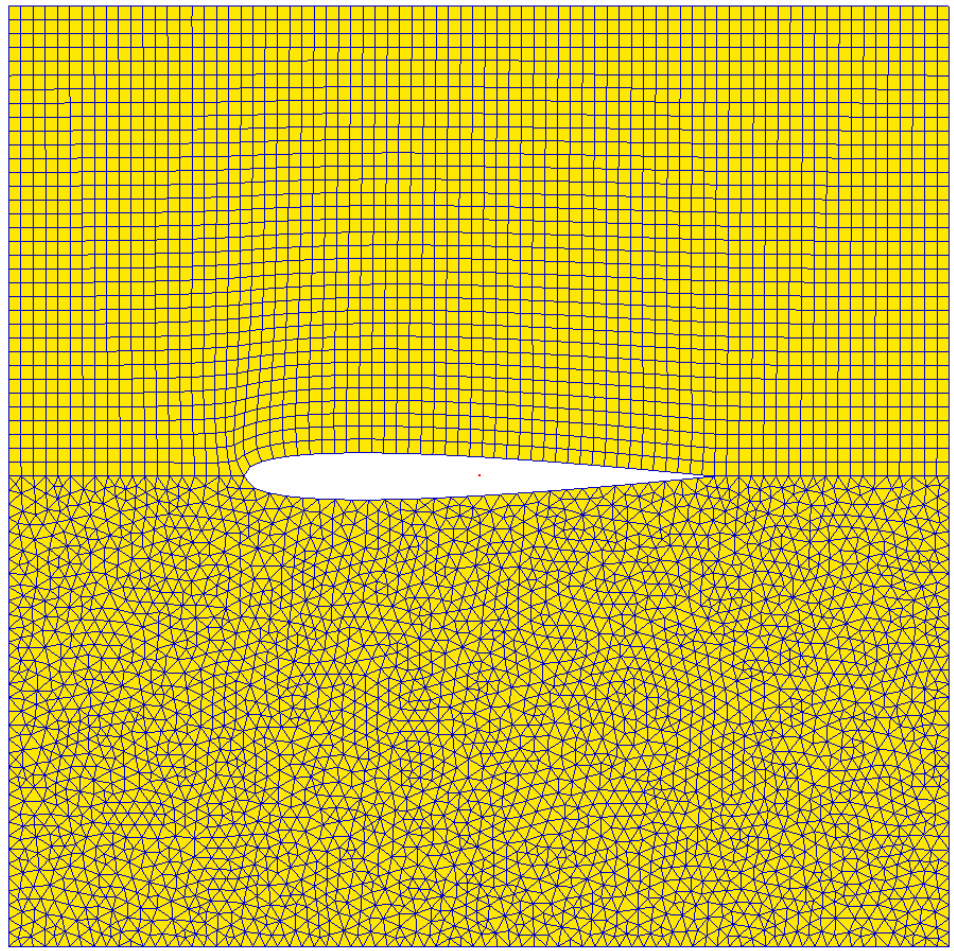
The window for writing the files is basically identical to that used to input the database files at the beginning of the grid generation process.



Building 2D Unstructured Grids

It's now time to build the unstructured block around the lower half of the airfoil.





The first few steps (selecting the analysis software, setting defaults – in this case the connector dimension, and importing the database geometry) are the same as before. To save some time, it will be assumed that these have already been performed.

The “Bottom-Up”Gridgen Process

Select Analysis Software and Mesh Dimensionality

⇩

Import and Manipulate Database Geometry

⇩

Create Segments and Connectors

⇩

Dimension Connectors and Distribute Grid Points

⇩

Assemble Domains

⇩

Assemble Blocks

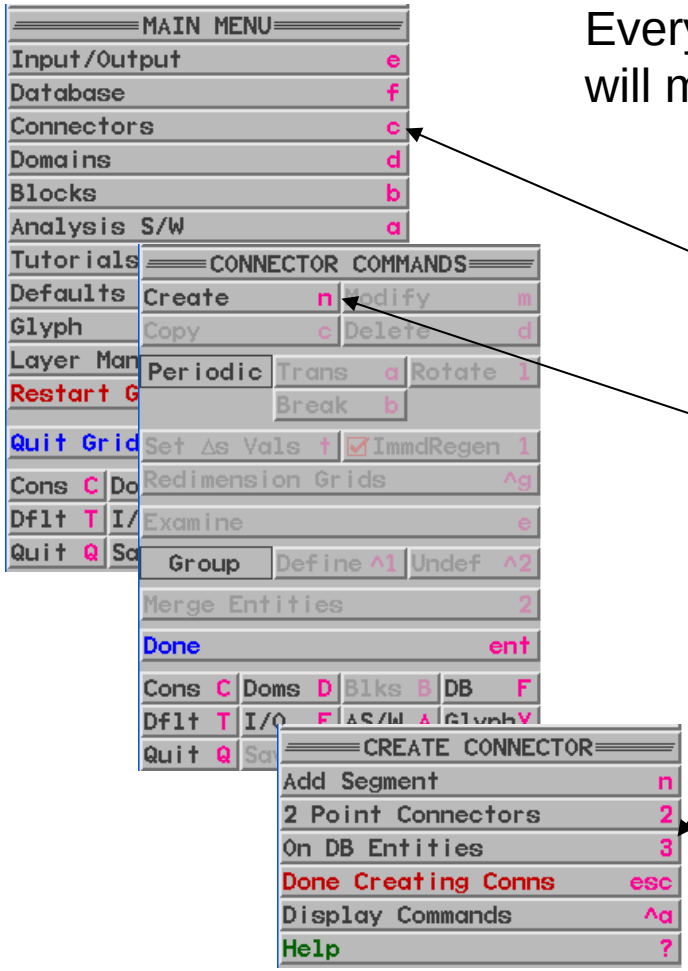
⇩

Setup CFD Analysis Boundaries

⇩

Export and Save Files

Create Connectors



Everything is thus set to start building the connectors that will make up the grid's edges.

In the **MAIN MENU**, **Connectors** is chosen to bring up the corresponding commands menu.

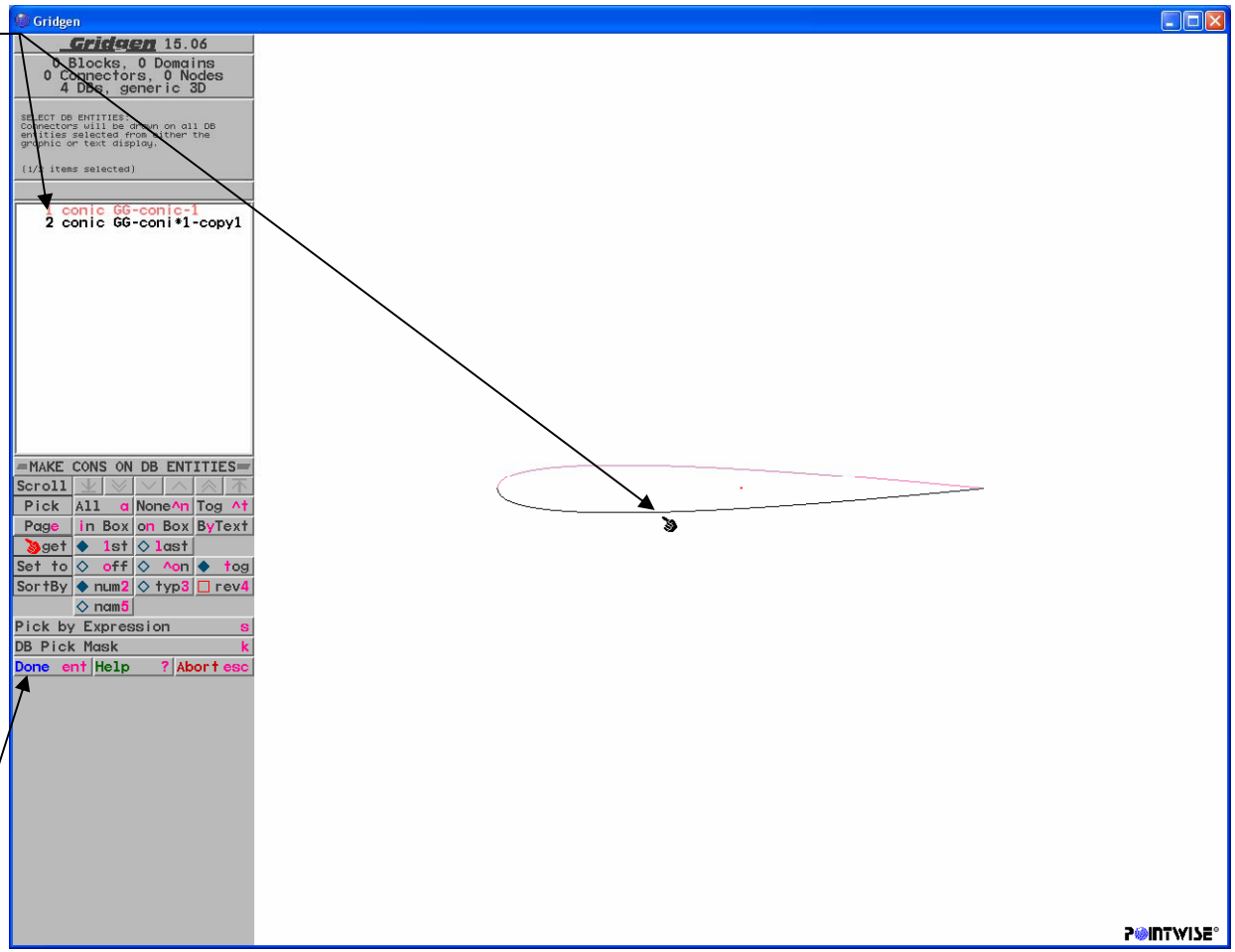
Select **Create** to open the **Create Connector** submenu

As the first connector will be laid down on the airfoil database geometry, pick **On DB Entities**.

Create Connectors (Continued)

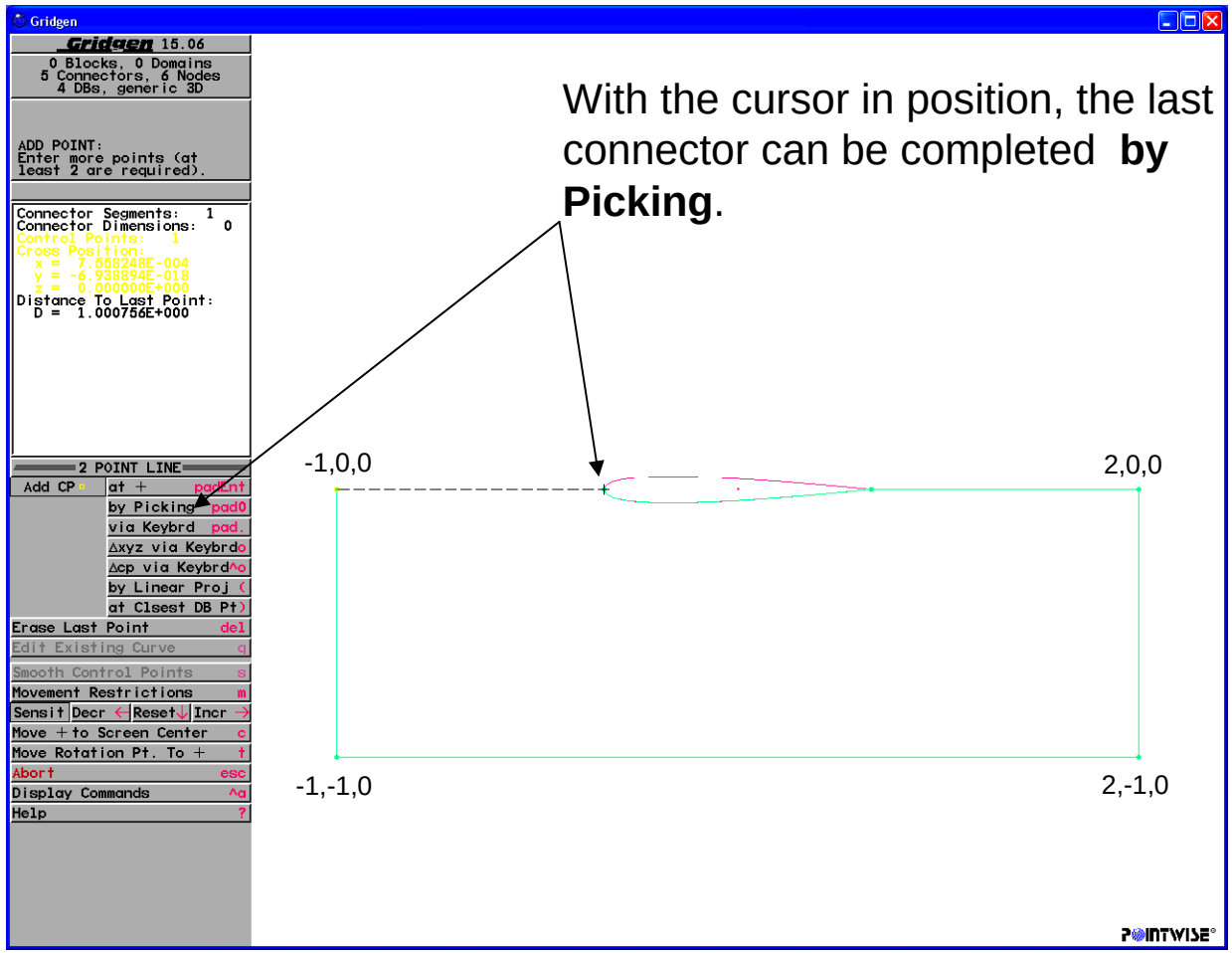
As seen previously, the airfoil is composed of two database entities, one for the upper surface and one for the lower. The mouse can be used to pick the curve for the lower surface from the list in the blackboard window or directly from the display window. Note that placing the mouse cursor over either the entry in the table or the entity in the display highlights both to aid the selection process.

Clicking on **Done** completes the process.



Create Connectors (Continued)

As before, the **2 Point Connectors** option is used to create the remaining connectors. Since Gridgen initially positions the display cursor at the last endpoint of the previously defined connector, it's possible to quickly box in the airfoil from below. Here, the cursor is positioned back at the leading edge of the airfoil, ready to pick the final endpoint of the last connector needed to complete the circuit.



Create Connectors (Continued)

With the last connector completed, click on **Done Creating Conns** in the **CREATE CONNECTOR** menu (or just hit the hot key, **esc**). Unlike a structured domain, the unstructured one being built inside these connectors will not require balanced numbers of grid points on opposite edges (indeed there will be only one edge that surrounds the entire domain). However, to promote a uniform cell density, it will still be desirable to redimension that long connector at the bottom to have roughly as many grid points as the three connectors on the other side have combined. Since the steps to do this are the same as previously shown, just the final result is displayed here.

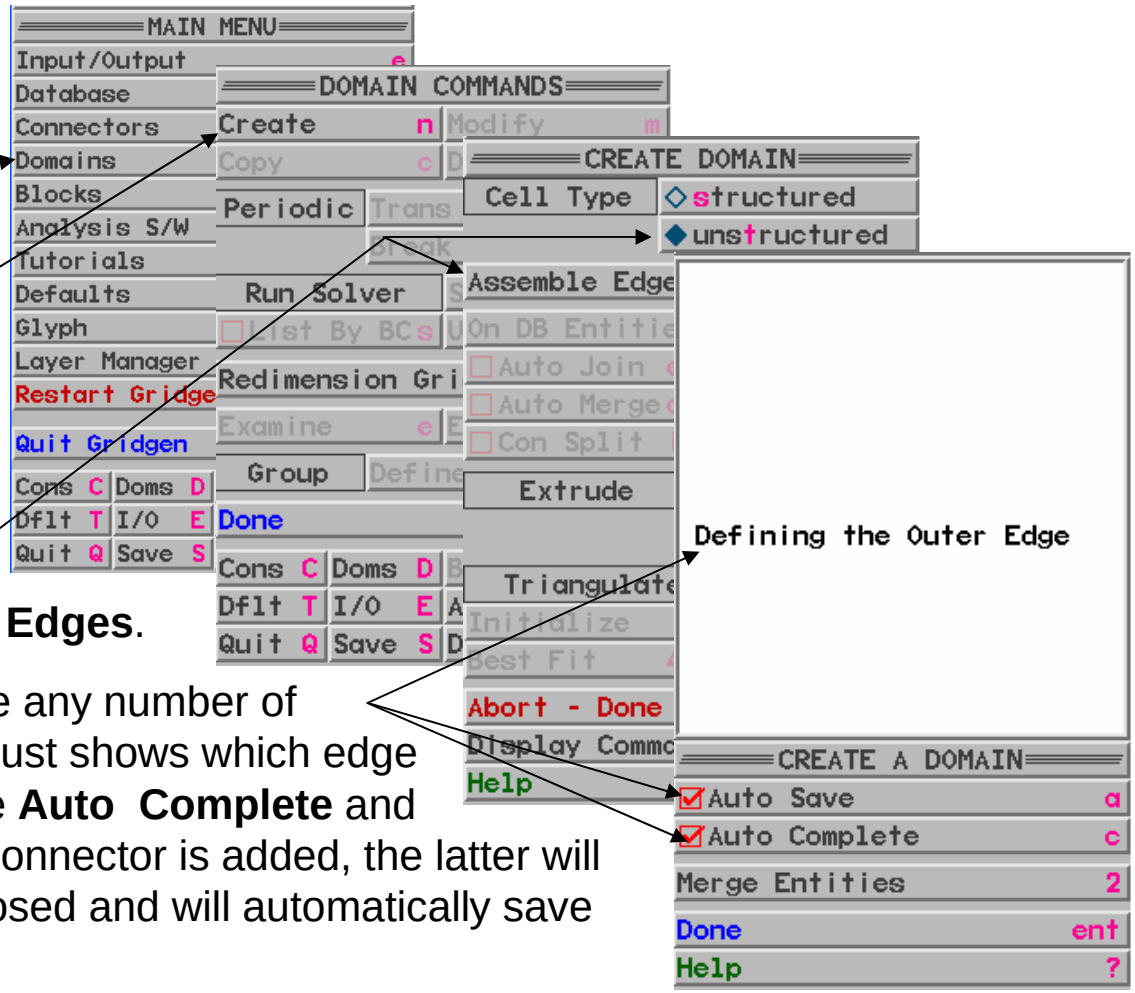
Assemble Domains

Everything is now ready to assemble the connectors that will bound the mesh domain.

In the **MAIN MENU**, pick **Domains** to switch to that menu.

Next, select the **Create** command to begin building the new domain.

Be sure that **unstructured** cell type is chosen, then pick **Assemble Edges**.

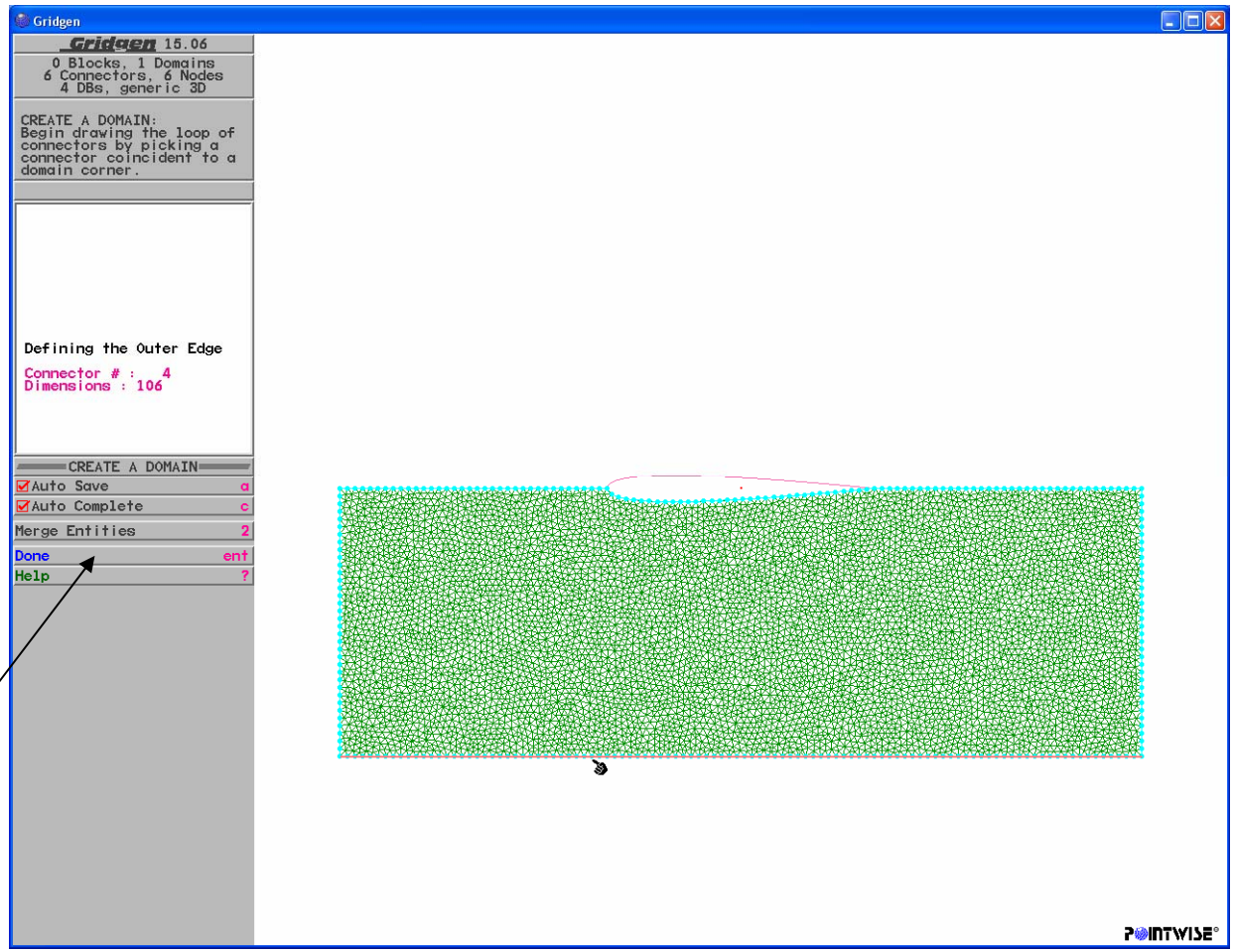


As an unstructured domain can have any number of edges, the blackboard window now just shows which edge is currently being formed. Make sure **Auto Complete** and **Auto Save** are turned on. As each connector is added, the latter will check whether the loop has been closed and will automatically save and initialize the domain when it is.

Assemble Domains (Continued)

The next step is to select the first connector (the bottom one is being picked here) to be added to the edge. Since there is only one possible loop, **Auto Complete** steps in to close it, where upon **Auto Save** saves and initializes it. This is the domain that results.

To get back to the **DOMAIN COMMANDS** menu, select **Done**. From there, **Done** can be selected again to return to the **MAIN MENU**.



Assemble Blocks

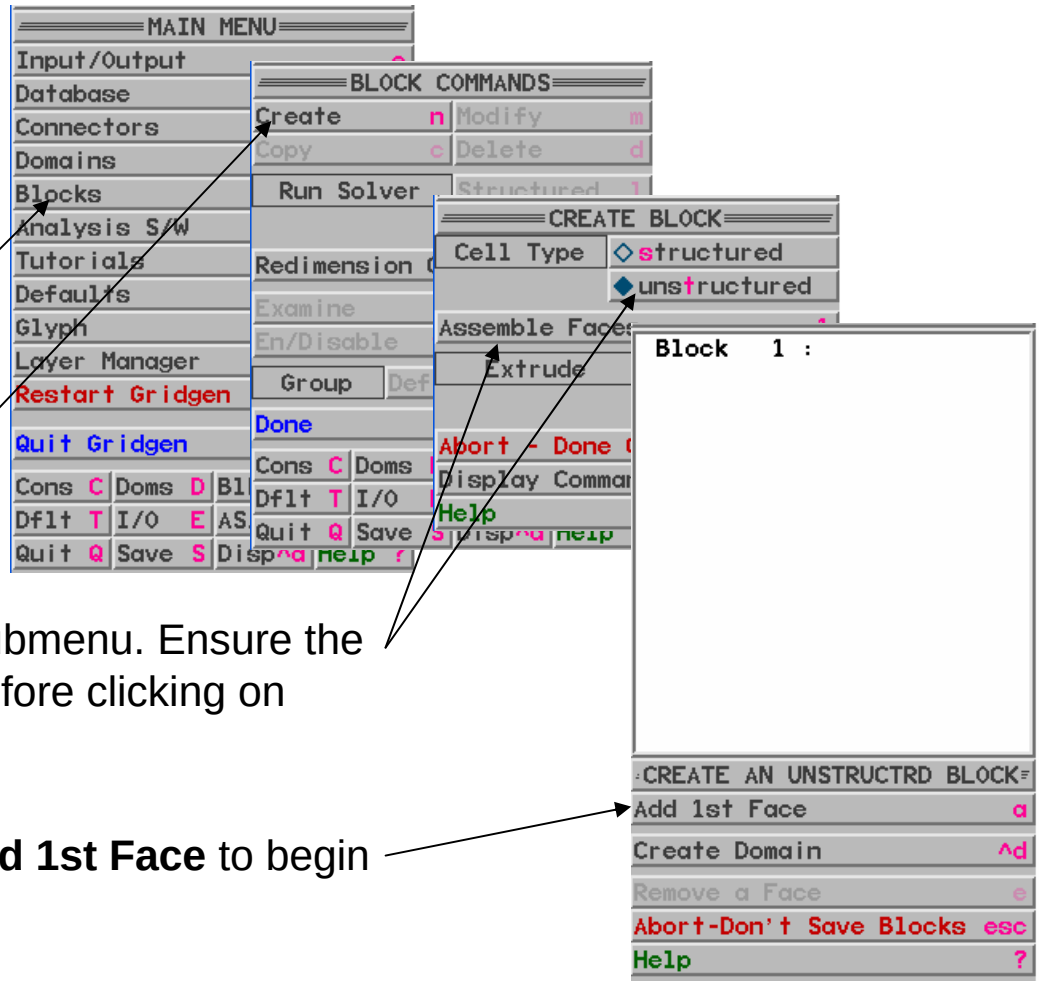
The grid block is next assembled from the domain, following the same basic procedure as for structured blocks.

From the **MAIN MENU**, **Blocks** is selected.

Create is then selected from the **BLOCK COMMANDS** menu.

This opens the **CREATE BLOCK** submenu. Ensure the **unstructured** cell type is toggled before clicking on **Assemble Faces**.

The next step is to pick **Add 1st Face** to begin forming the block.

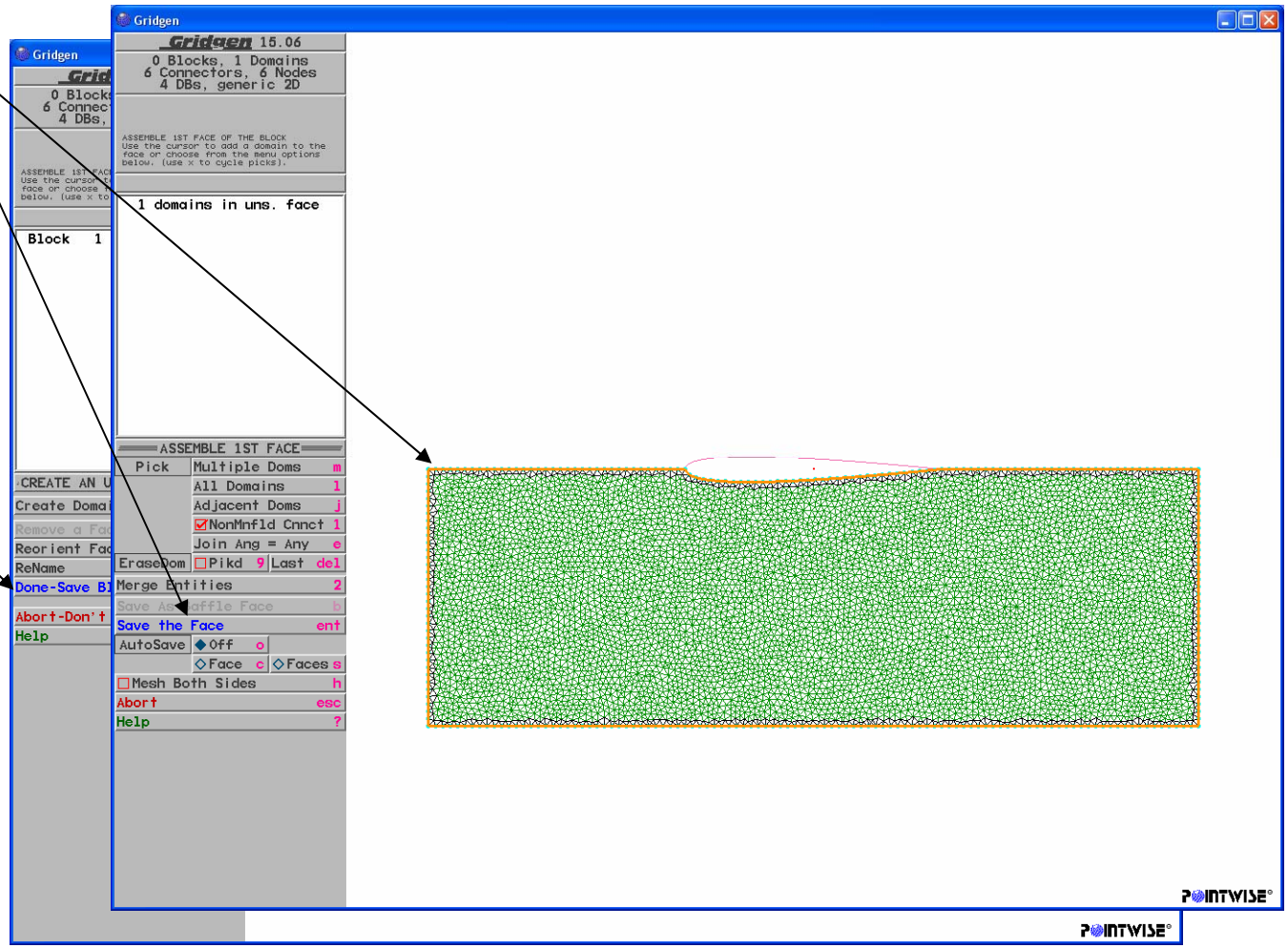


Assemble Blocks (Continued)

The face is selected by mouse clicking on its image in the display window. Since this is a 2D block, once the **Save the Face** button is pressed, the block is complete.

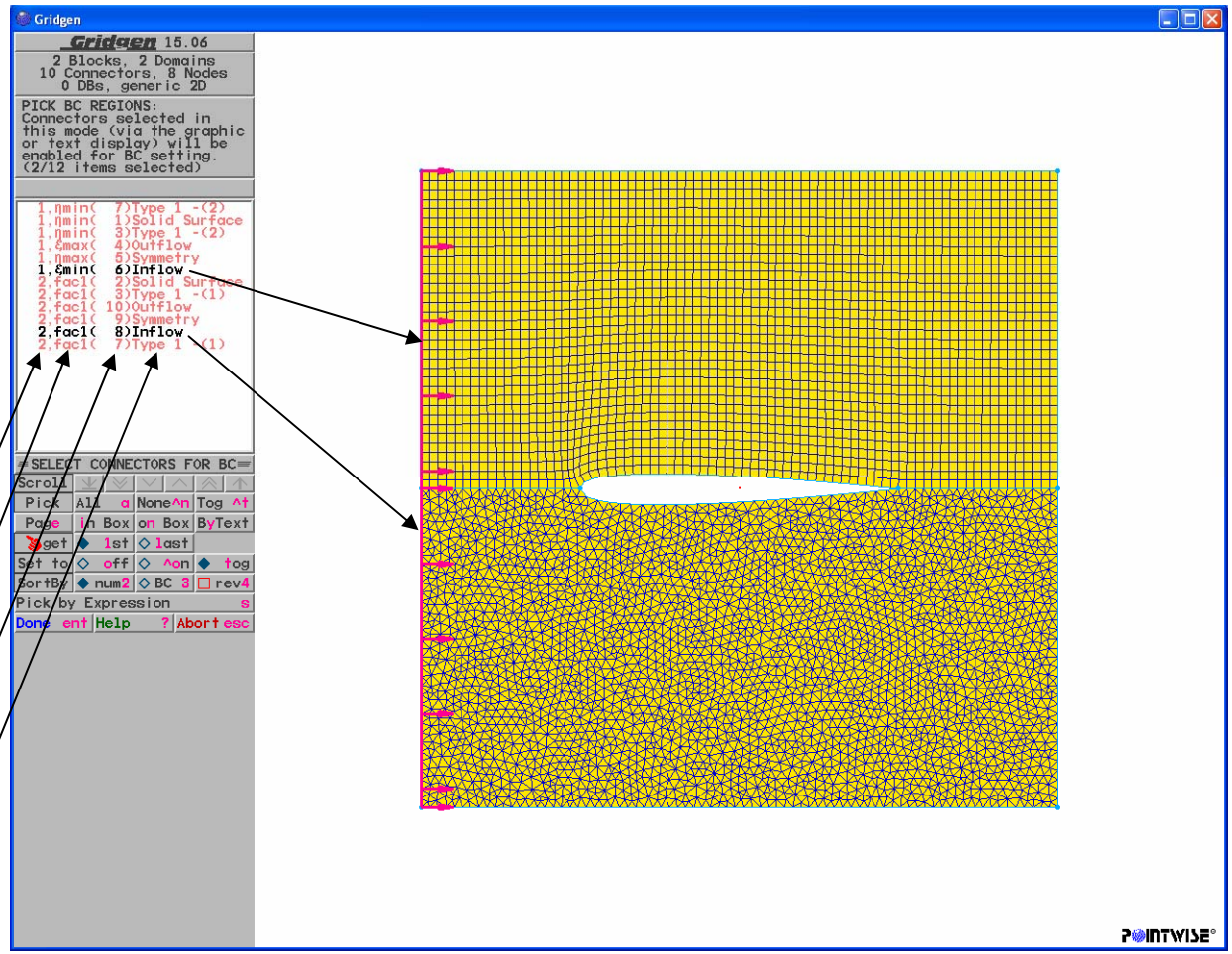
Press **Done – Save Blocks** to finish this step.

From here, setting the boundary conditions and exporting the files for the CFD solver follow the same path as shown for the 2D structured grid, so will not be repeated.



Set Analysis Boundaries

Setting boundary conditions for an unstructured mesh is the same as doing so for a structured one, but there are some differences in how the boundary faces are displayed in the browser.



Block Number

Coordinate Direction (Structured)
Face Number (Unstructured)

Connector Number

Boundary Condition
(Type 1... are block interface connections)

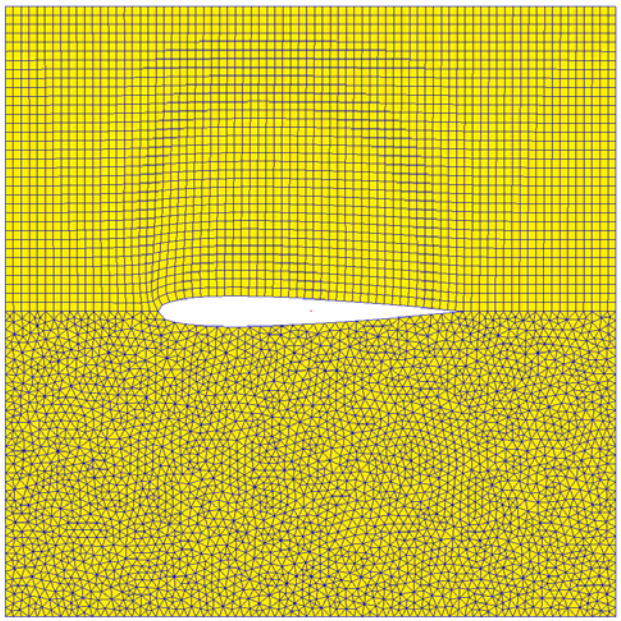
Gridgen**Training**

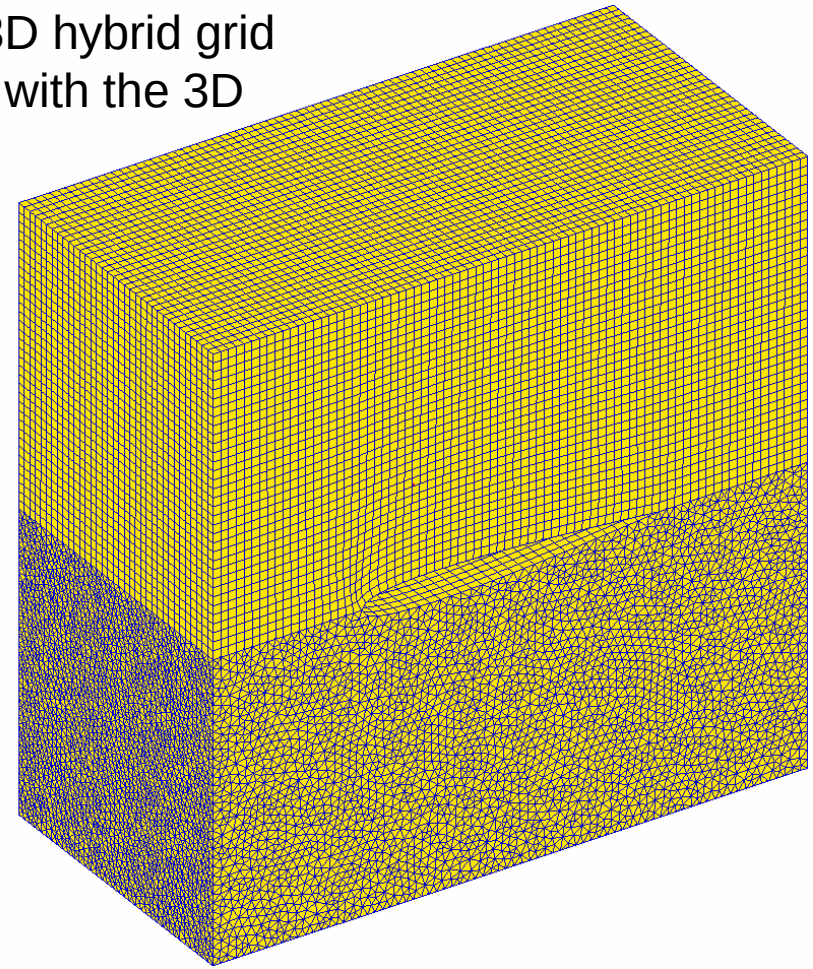
- In this session:
  - Gridgen Terminology.
  - GUI Components.
  - Gridgen Help.
  - Building 2D Structured Grids.
  - Building 2D Unstructured Grids.
  - Tutorial: “2D Bump: Basic Skills.”

- In this session:
  - Building 3D Structured Grids.
  - (Re)Dimensioning and (Re)Distributing Structured Grids.
  - Tutorial: “Swept Ramp: Basic Multi Block Structured Grid.”

Building 3D Structured Grids

Beginning with the 2D airfoil mesh below, the 3D hybrid grid shown at the right will be constructed, starting with the 3D structured block at the top.





The “Bottom-Up”Gridgen Process

Select Analysis Software and Mesh Dimensionality

⇩

Import and Manipulate Database Geometry

⇩

Create Segments and Connectors

⇩

Dimension Connectors and Distribute Grid Points

⇩

Assemble Domains

⇩

Assemble Blocks

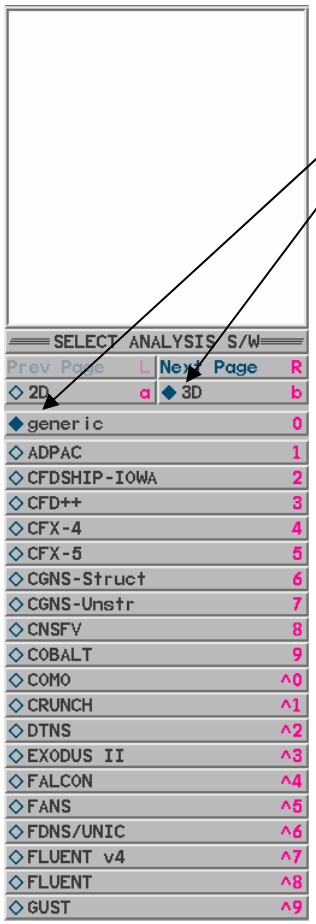
⇩

Setup CFD Analysis Boundaries

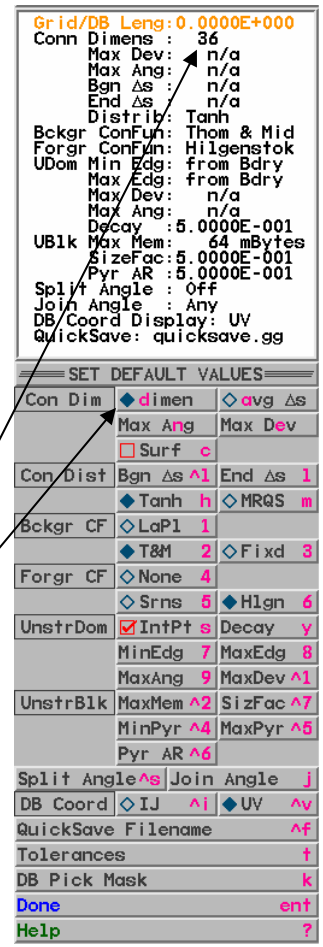
⇩

Export and Save Files

Analysis S/W and Default Settings



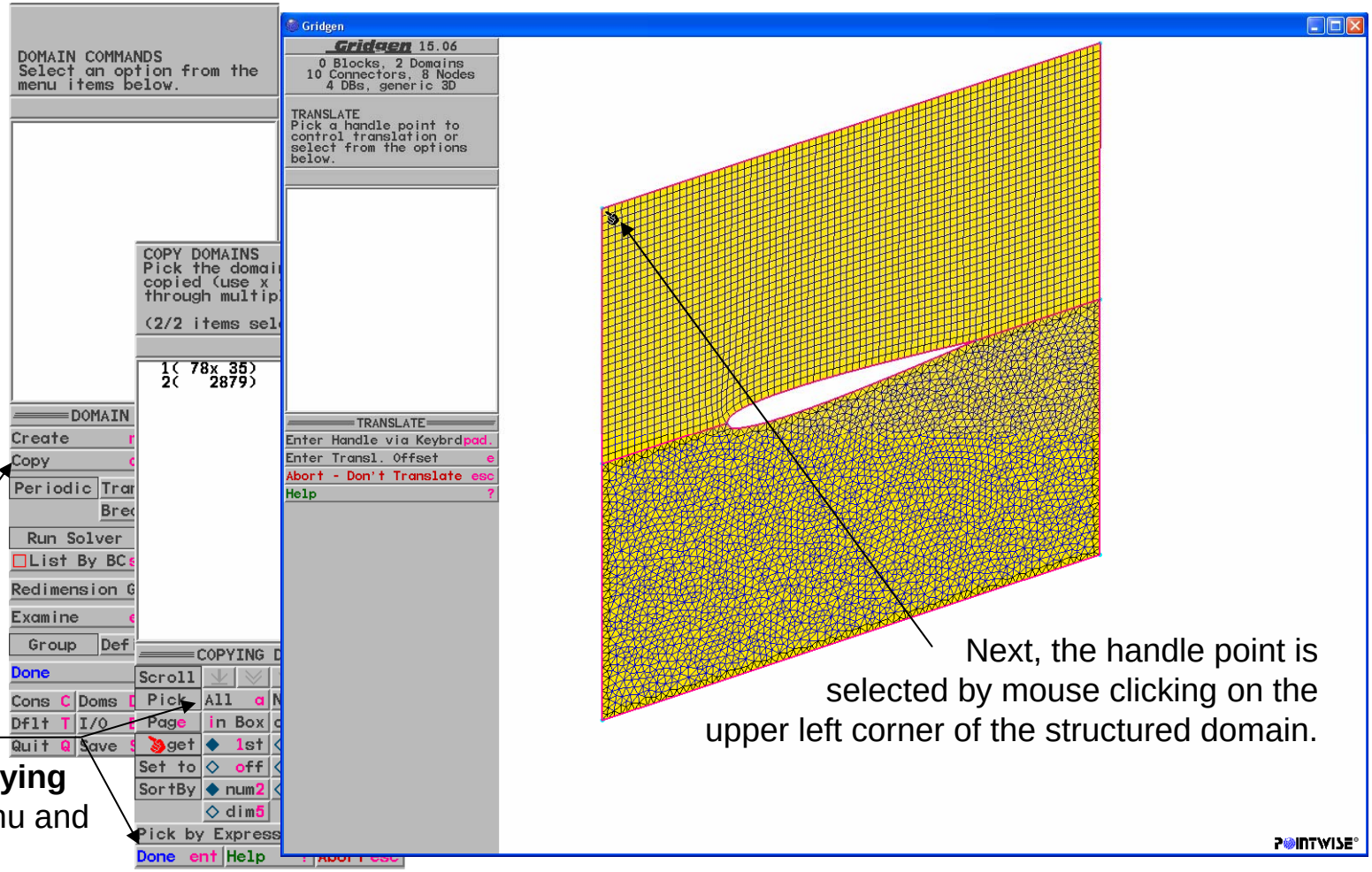
Before proceeding, toggle the **3D** and **generic** solver buttons in **SELECT ANALYSIS S/W**...



and be sure that the default connector dimension (**Conn Dimens**) is set to 36 in **SET DEFAULT VALUES**.

Copying Domains

Before the block can be formed, the remaining domains that make up its faces must be created. First, the existing domains will be copied.

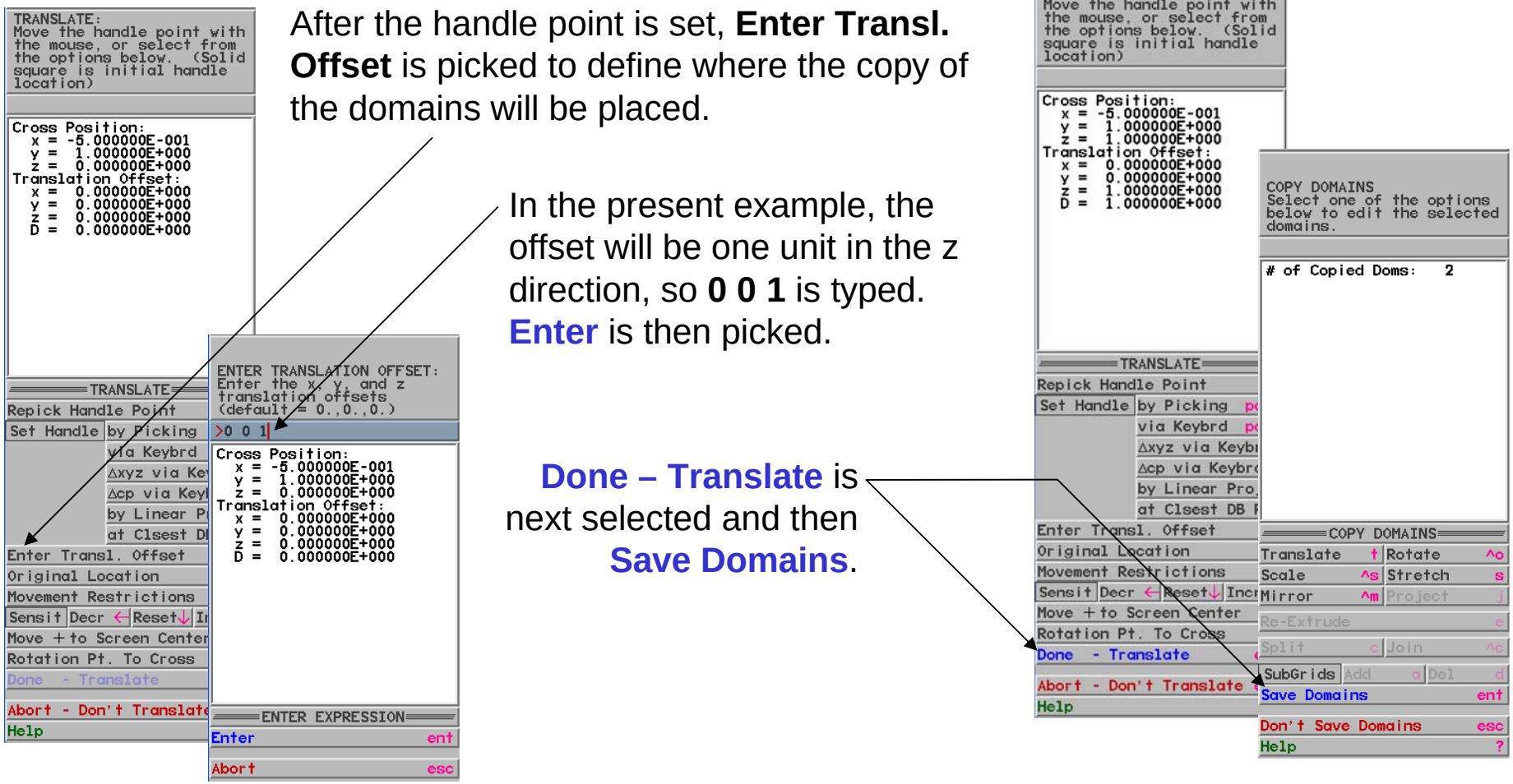


Copy is picked in the **DOMAIN COMMANDS** menu.

Pick All is then selected in the **Copying DOMAINS** sub-menu and then **Done**.

Next, the handle point is selected by mouse clicking on the upper left corner of the structured domain.

Copying Domains (Continued)



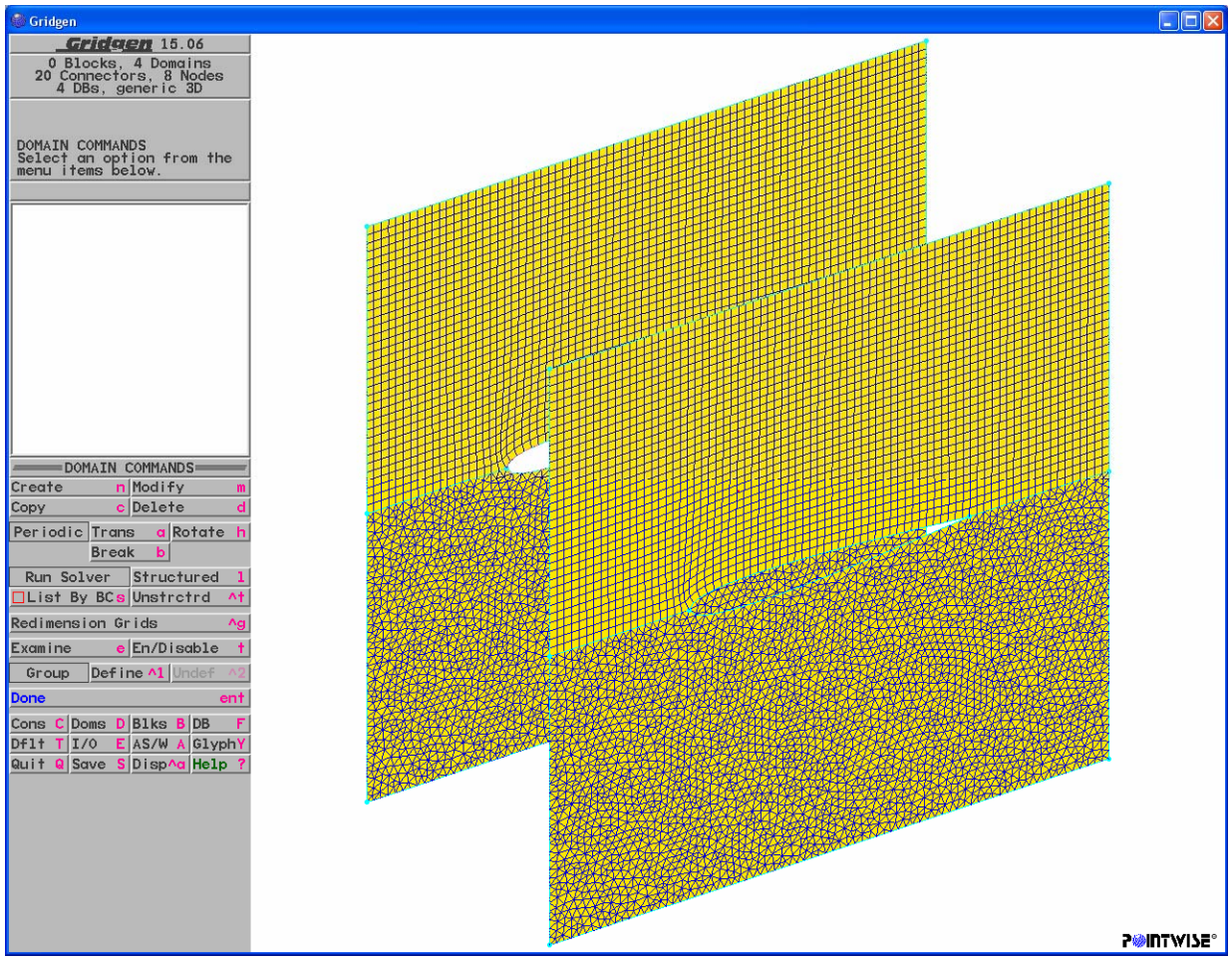
After the handle point is set, **Enter Transl. Offset** is picked to define where the copy of the domains will be placed.

In the present example, the offset will be one unit in the z direction, so **0 0 1** is typed. **Enter** is then picked.

Done – Translate is next selected and then **Save Domains**.

Copying Domains (Continued)

The result will be the two identical, but offset domains, seen here. To return to the **MAIN MENU**, **Done** is selected.

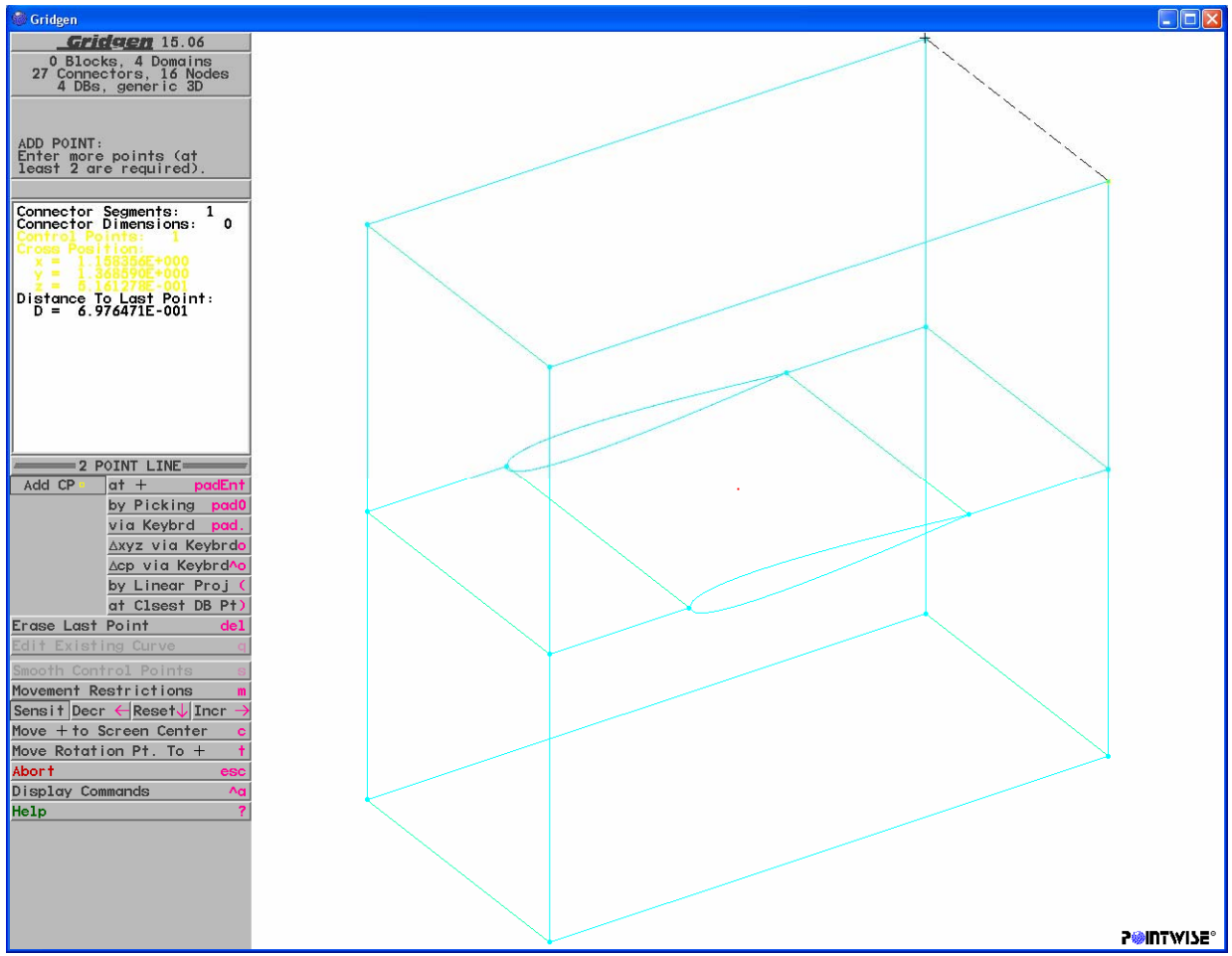


Create Connectors

The next step is to create a series (8 total) of two point connectors between matching nodes on the edges of the domains – to save time later, the connectors for the unstructured block are also defined now. As this process has already been demonstrated, it will not shown in detail again, but here are a couple caveats:

Be sure that the default connector dimension is still set to 36.

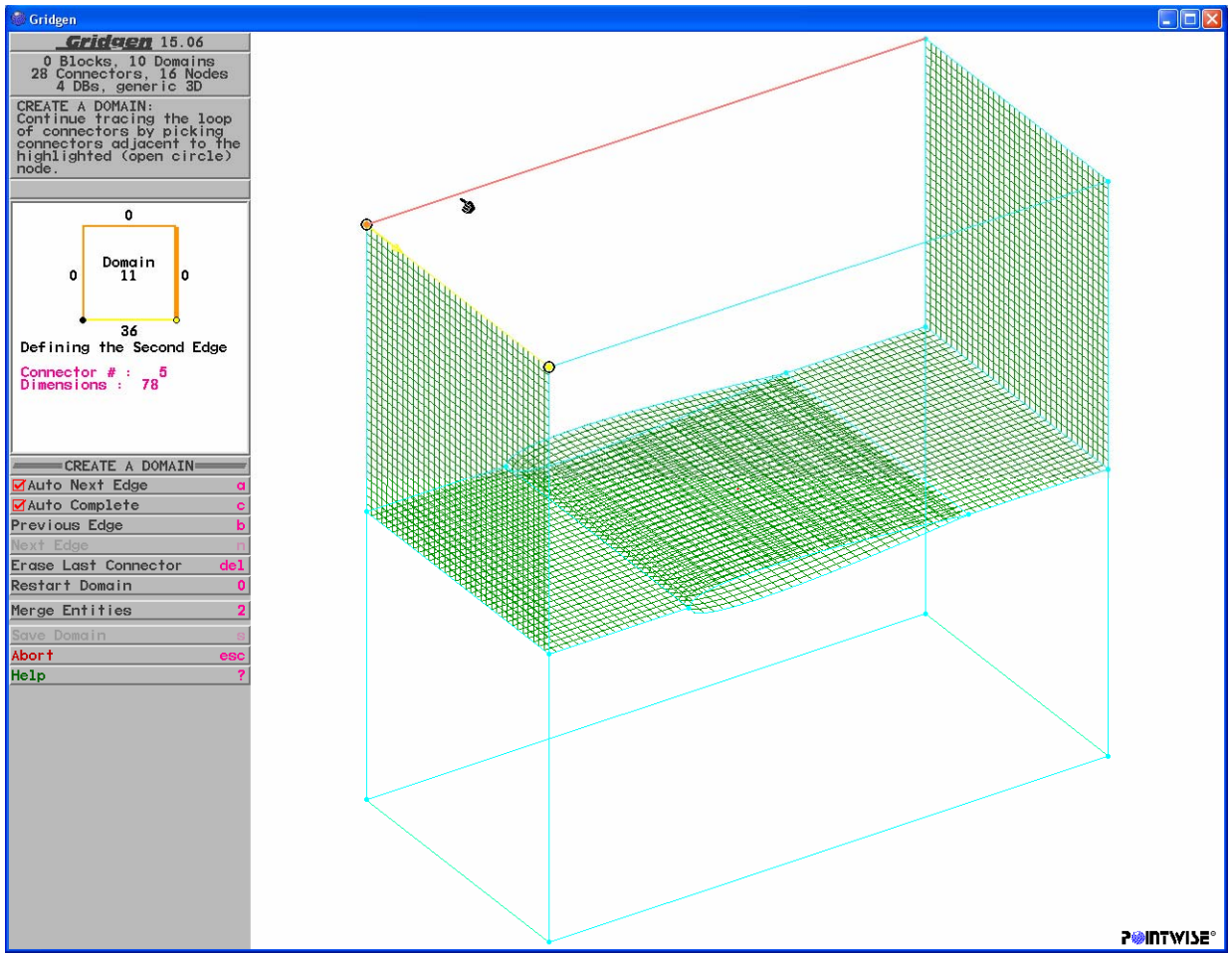
Switching off the rendering of the domains will aid in locating all of the nodes to be connected.



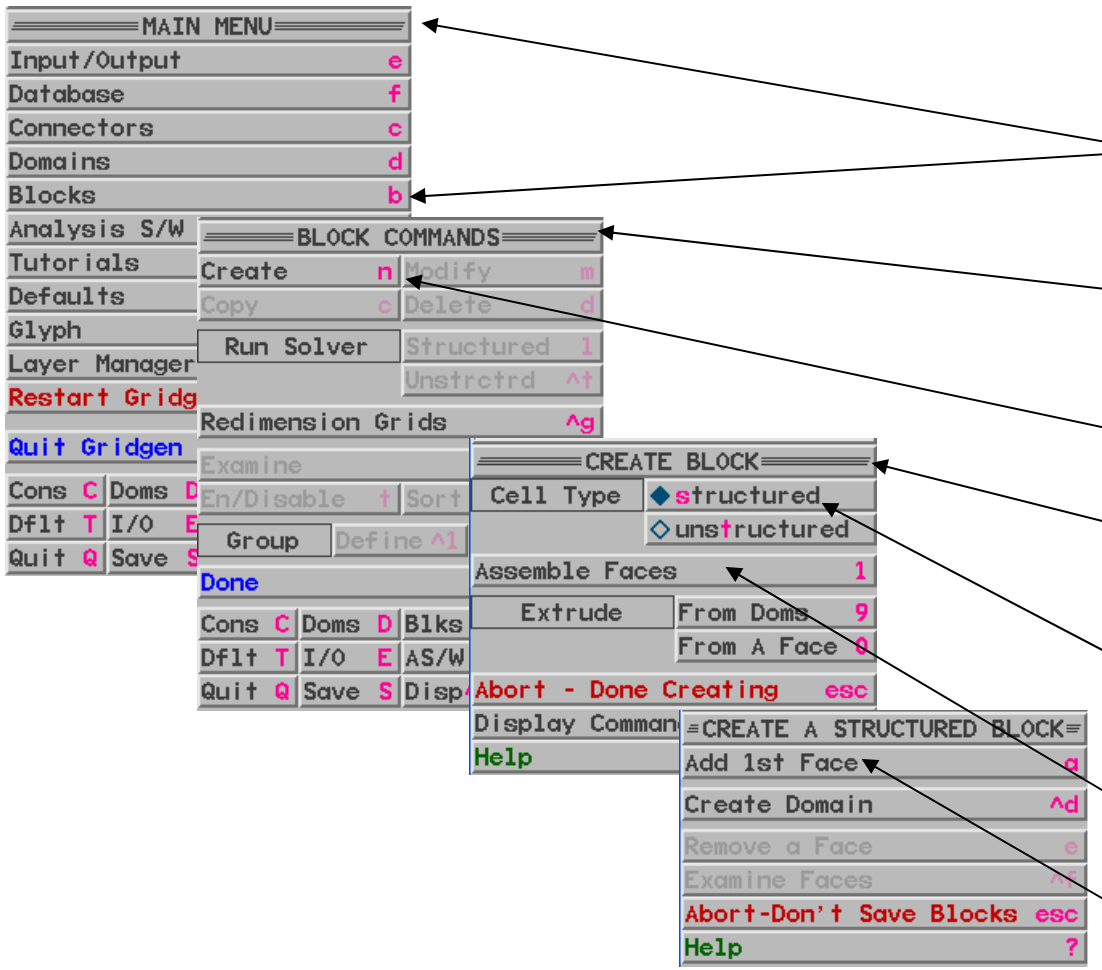
Assemble Domains

Once all of the connectors are defined, the remaining six domains needed for the faces of the structured block can be assembled: left, right, bottom (three total), and top - the latter is just being formed in the picture to the right. As an additional time saving measure, another structured domain is defined for the lower surface of the airfoil for later use.

Since this process has already been demonstrated, it will not be shown in detail again. Just remember using **Auto Next Edge** and **Auto Complete** will save time.



Assemble Blocks



Finally, it's time to assemble the block.

In the **MAIN MENU**, **Blocks** is selected.

This opens the **BLOCK COMMANDS** submenu.

Here **Create** is selected.

This opens the **CREATE BLOCK** submenu.

Be sure that **Cell Type** is set to **structured**

Then select **Assemble Faces**.

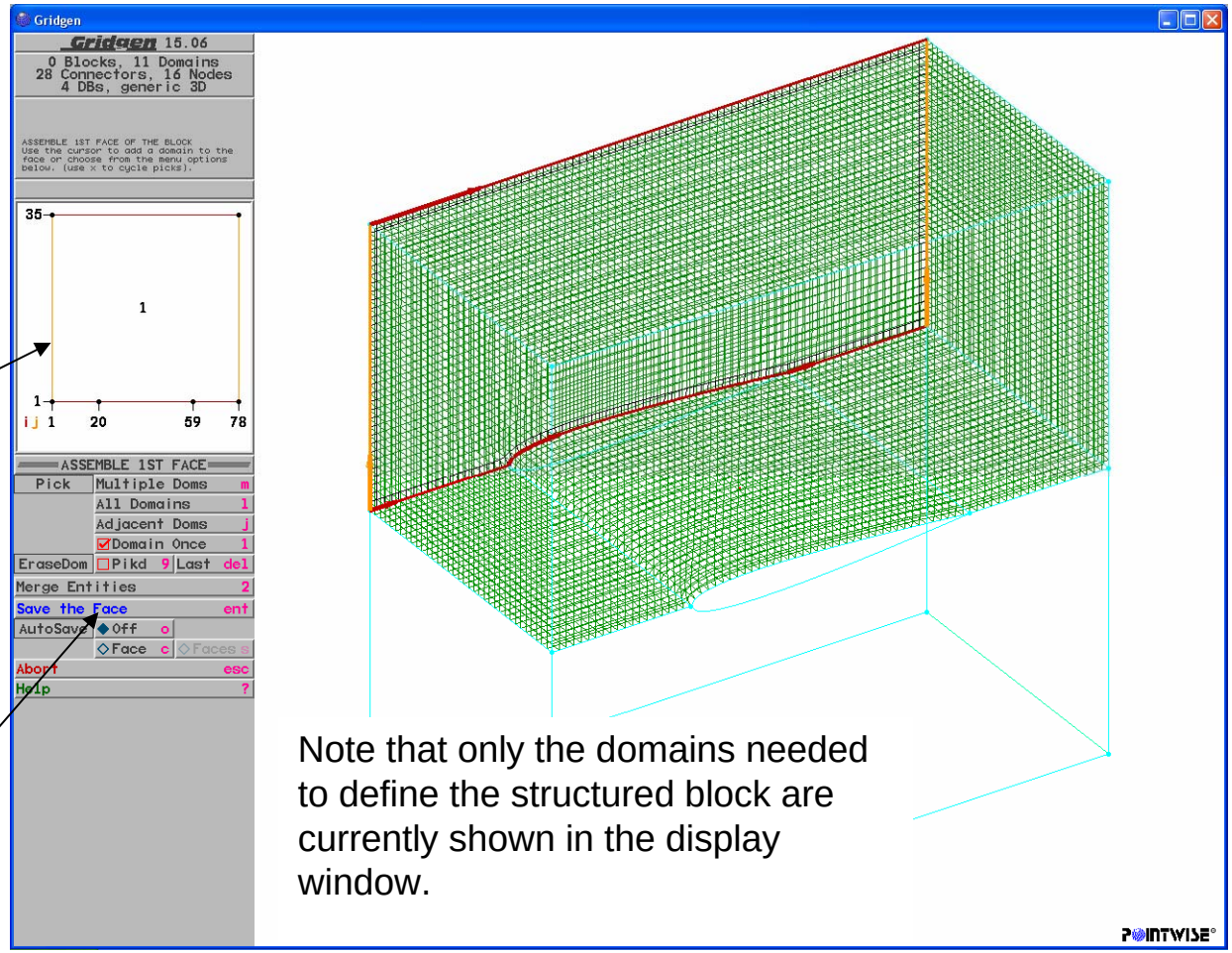
Next, select **Add 1st Face**.

Assemble Blocks (Continued)

The **ASSEMBLE 1st FACE** menu will open and the mouse can be used to pick the first domain to be added. In this case, the domain that makes up the back of the block has been selected.

Note the color coordinated face diagram in the blackboard window. The number in the center identifies the highlighted domain. The numbers along the periphery show the dimensions of its connectors.

As this is the only domain for this face, **Save the Face** is clicked to complete it.



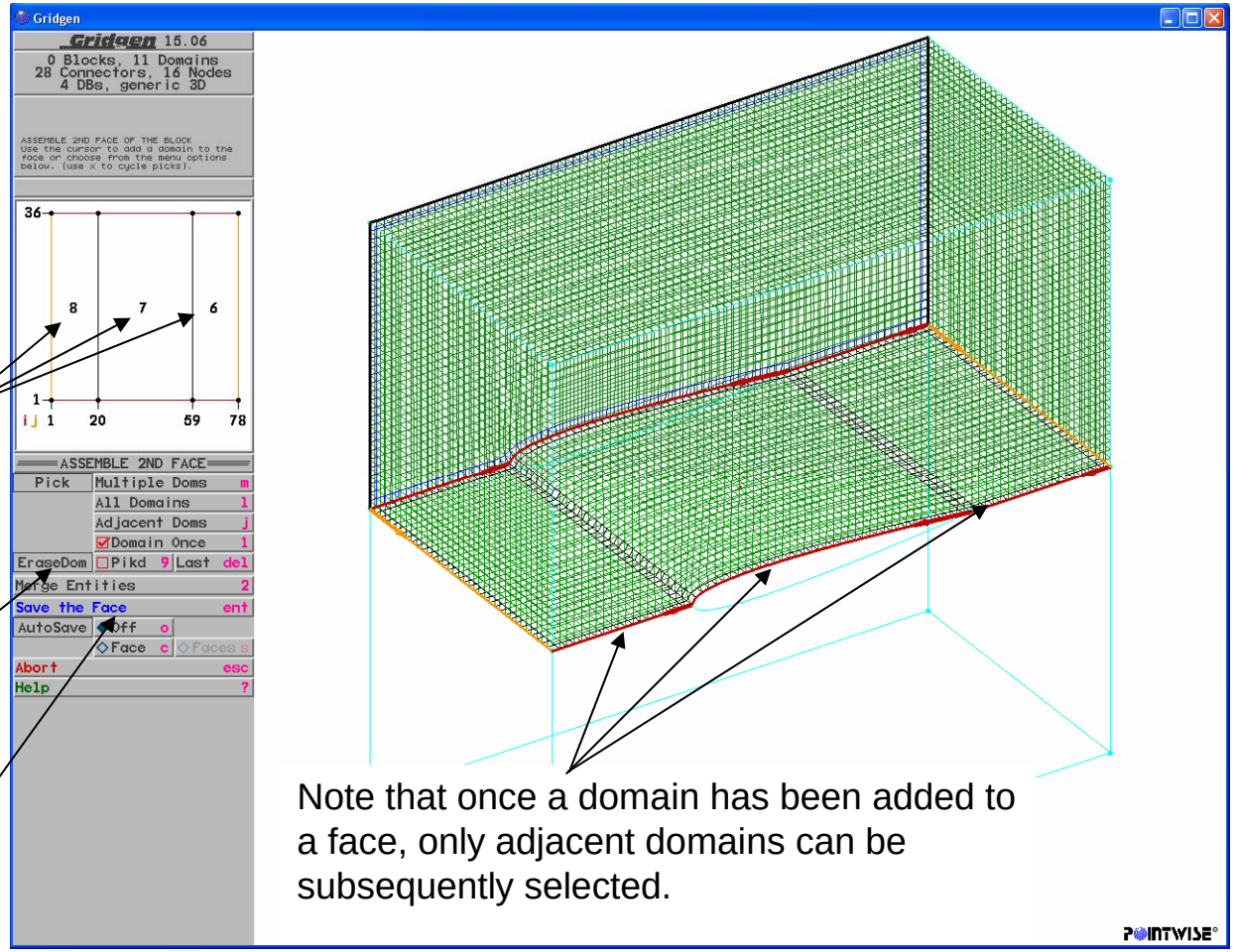
Assemble Blocks (Continued)

The **ASSEMBLE 2nd FACE** menu will next open and the mouse can be used to pick the domains to be added. In this case, the three domains that makes up the bottom of the block have been selected.

Note the face diagram in the blackboard window shows the particulars for all three selected domains.

If the wrong domain is picked, **EraseDom** ☐**Pikd** (to mouse click on the culprit) or **Last** (last one added) can be used.

Save the Face is clicked to move on to the next face.

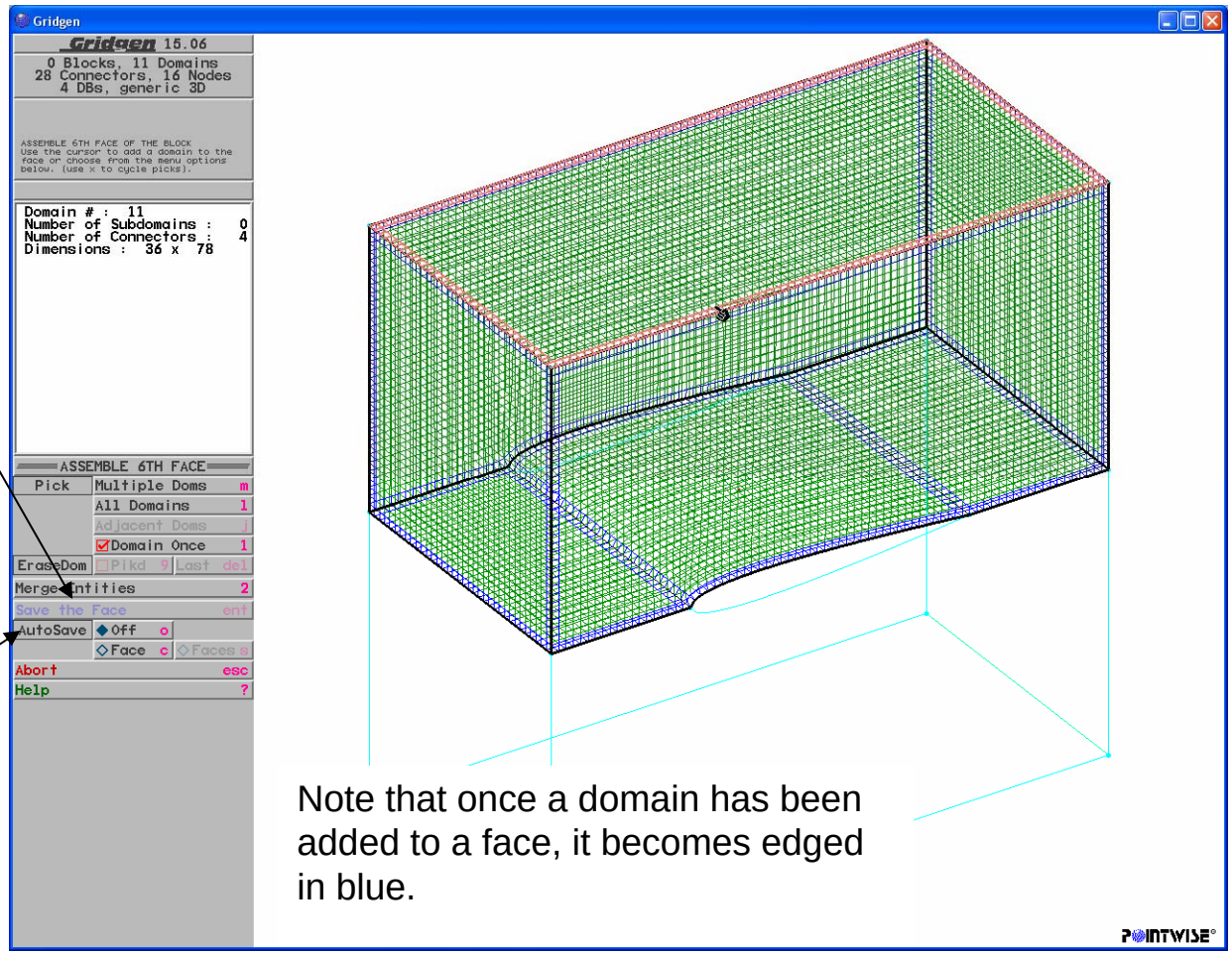


Assemble Blocks (Continued)

The remaining single domain faces are added in a similar fashion to the first. Here, the domain for sixth and last face (the top of the block) is about to be selected.

Once the domain is selected, **Save the Face** can be clicked to finish assembling the faces.

AutoSave Face can speed creation of single domain faces, but make sure it's off when building multi-domain ones.



Note that once a domain has been added to a face, it becomes edged in blue.

Assemble Blocks (Continued)

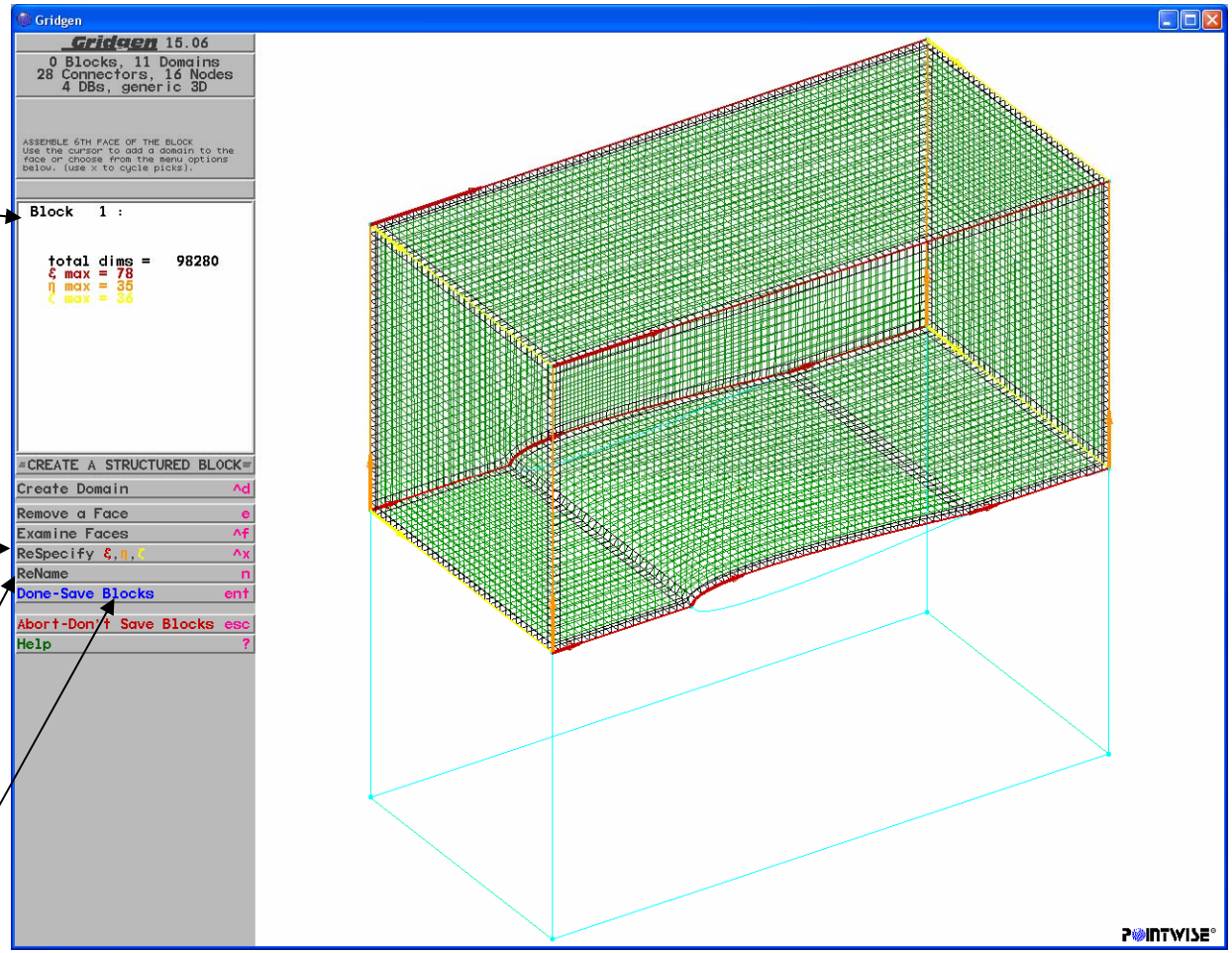
After the six faces are defined, the structured block can be completed.

The blackboard window provides a color coordinated list of the computational coordinates and the corresponding grid dimensions of proposed block.

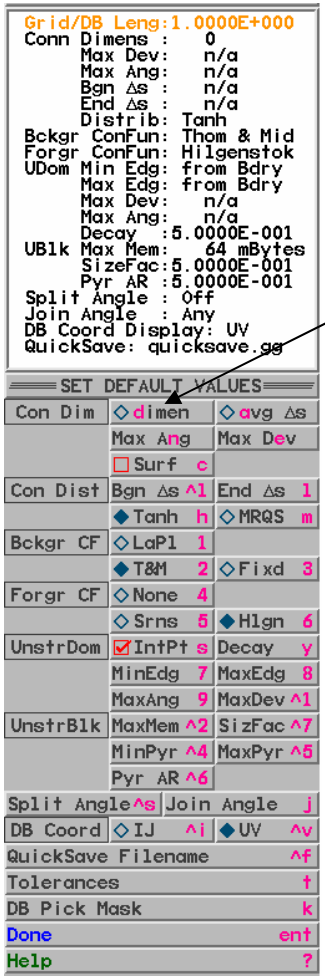
If necessary, the computational coordinates can be changed by selecting **ReSpecify** ξ, η, ζ.

If desired, **ReName** can be used to give the block a name.

Finally, **Done – Save Blocks** will finish the block's creation.



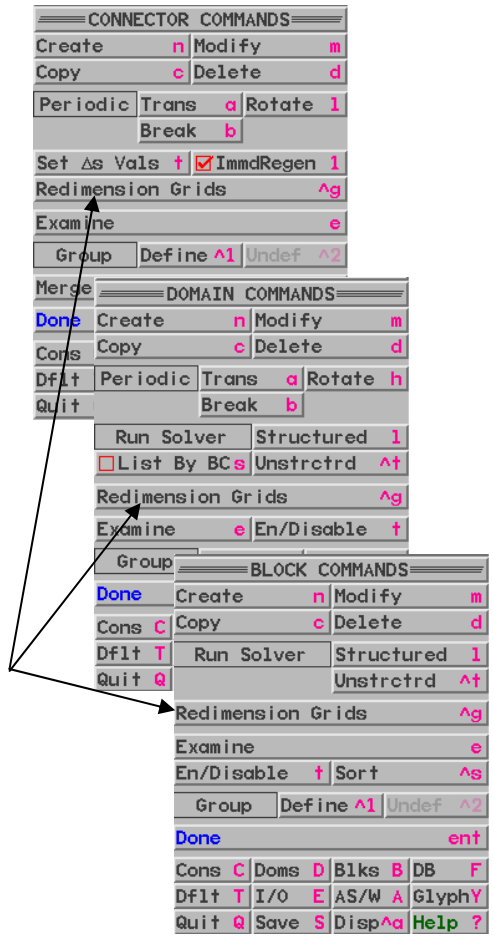
(Re)Dimension Grids



Dimensioning and redimensioning connectors has already been touched upon briefly.

In the **SET DEFAULT VALUES** menu, **Con Dim dimen** can be used to establish a default dimension for all subsequently created connectors.

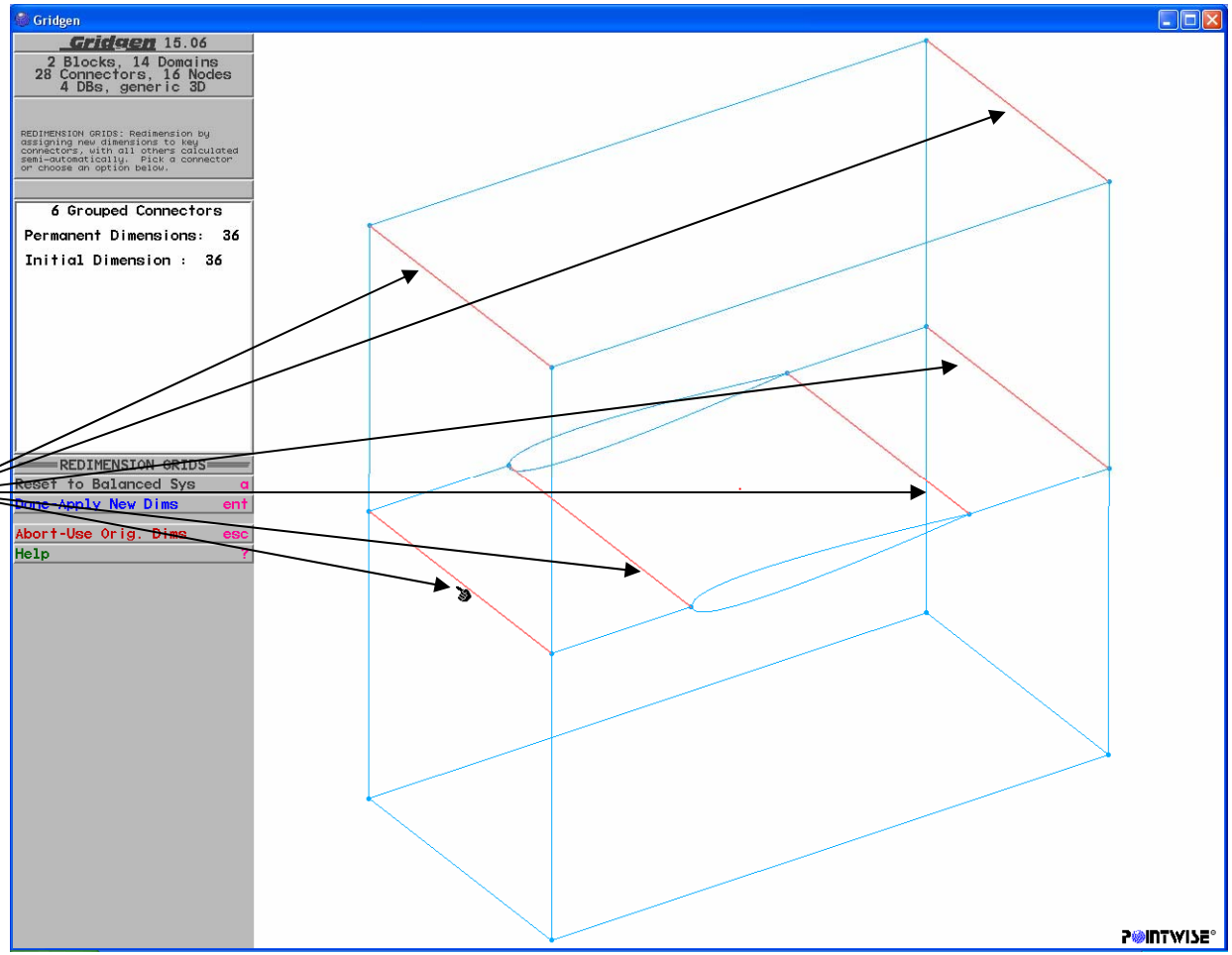
In addition, a previously defined connector can be redimensioned with the **Redimension Grids** command found in the **CONNECTOR**, **DOMAIN**, and **BLOCK COMMANDS** menus.



(Re)Dimension Grids (Continued)

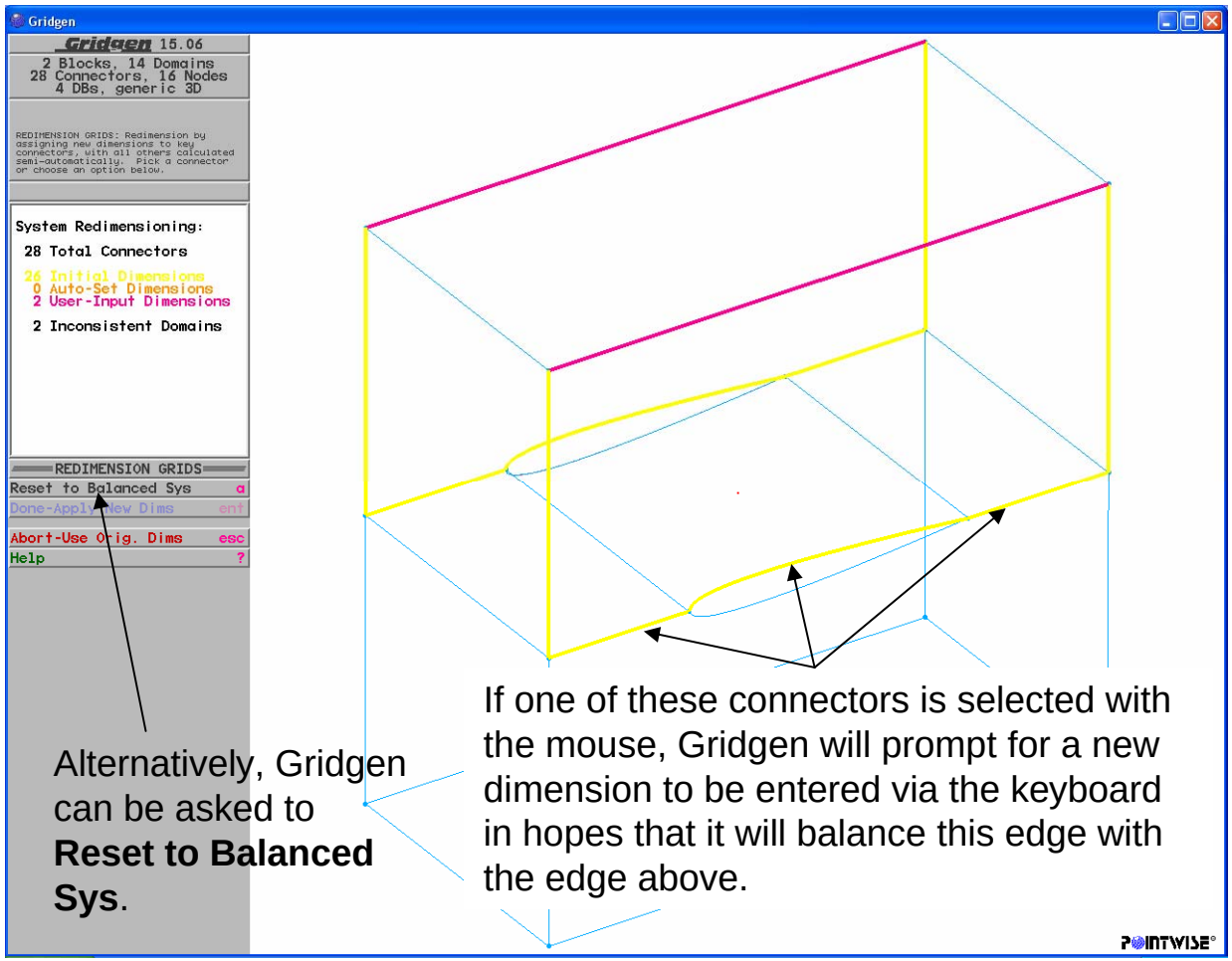
As structured grids require computationally rectangular domains, the **Redimension Grids** command automatically propagates new dimensions to all affected domains and blocks.

When one connector in a block or domain is highlighted for redimensioning, all other connectors that will have to be redimensioned by Gridgen to maintain balance also are highlighted.



(Re)Dimension Grids (Continued)

If a redimensioning leads to an inconsistency in the dimensions of the system of connectors, Gridgen will highlight the problem and require its resolution before continuing. In this case, the dimensions of the two connectors highlighted in pink were changed, leading to an imbalance with the corresponding edges, each made up of three connectors, below. Either the old dimensions can be restored or one (or more) of the connectors below will also have to be redimensioned.



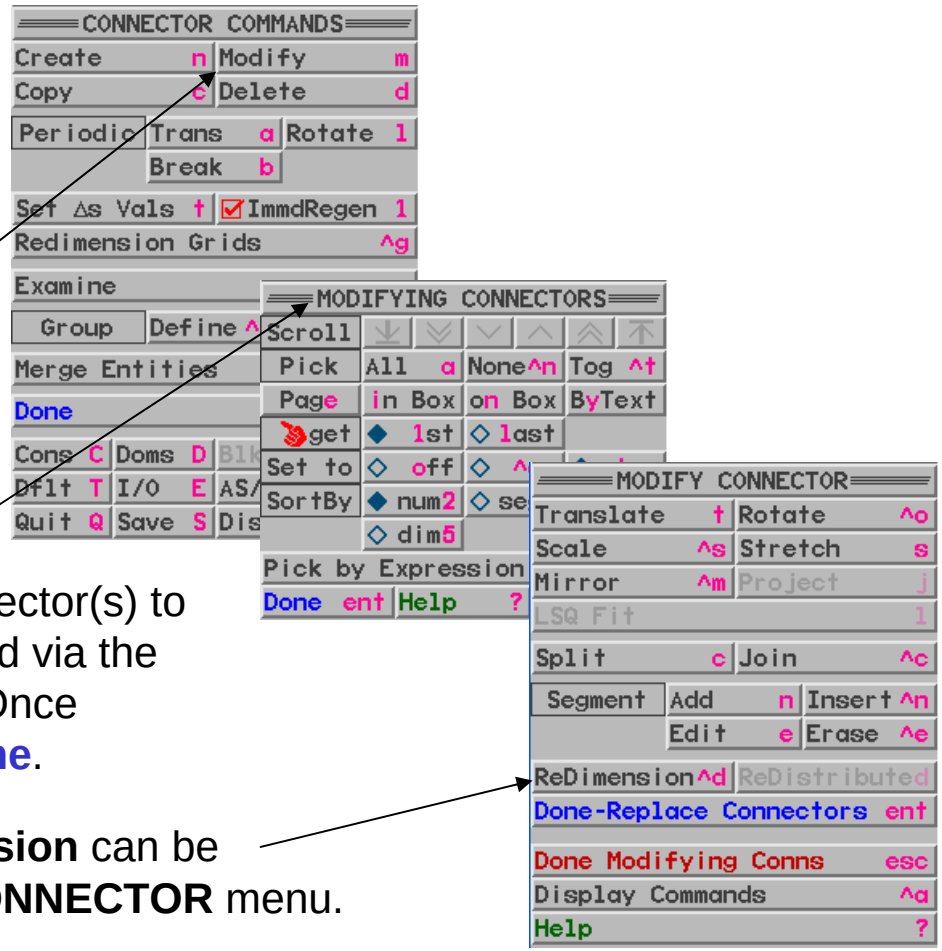
(Re)Dimension Grids (Continued)

It's also possible to redimension a connector via the **Modify** command in the **CONNECTOR COMMANDS** menu. This option is not available for connectors that are already part of a structured domain.

Modify is selected in the the **CONNECTOR COMMANDS** menu.

This opens the **MODIFYING CONNECTORS** menu. The connector(s) to be redimensioned can be selected via the menu options or mouse picked. Once selection is completed select **Done**.

Once selected, **ReDimension** can be picked in the **MODIFY CONNECTOR** menu.

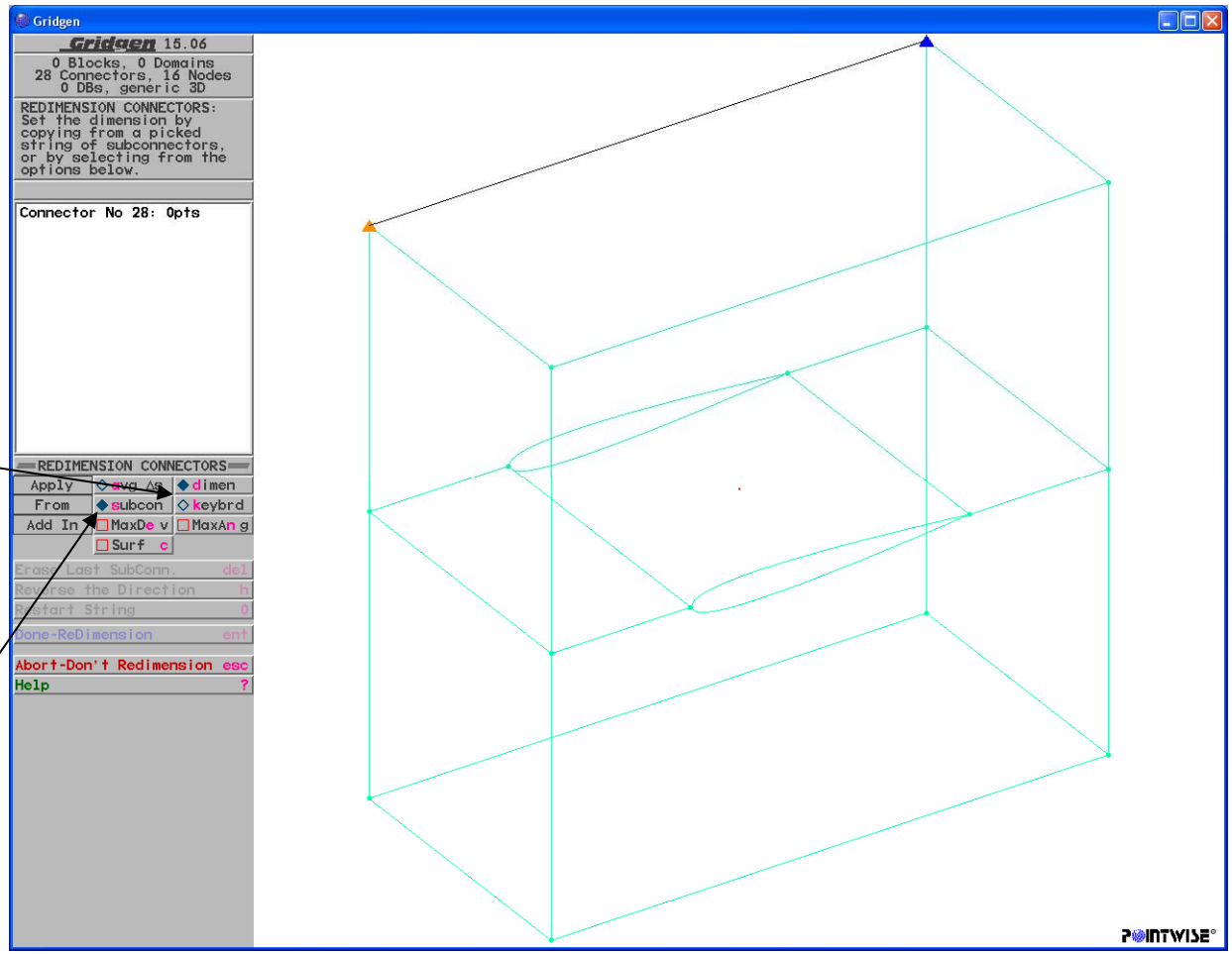


(Re)Dimension Grids (Continued)

After **ReDimension** is selected, the chosen connector(s) become highlighted in black with orange and blue triangles at the beginning and end, respectively.

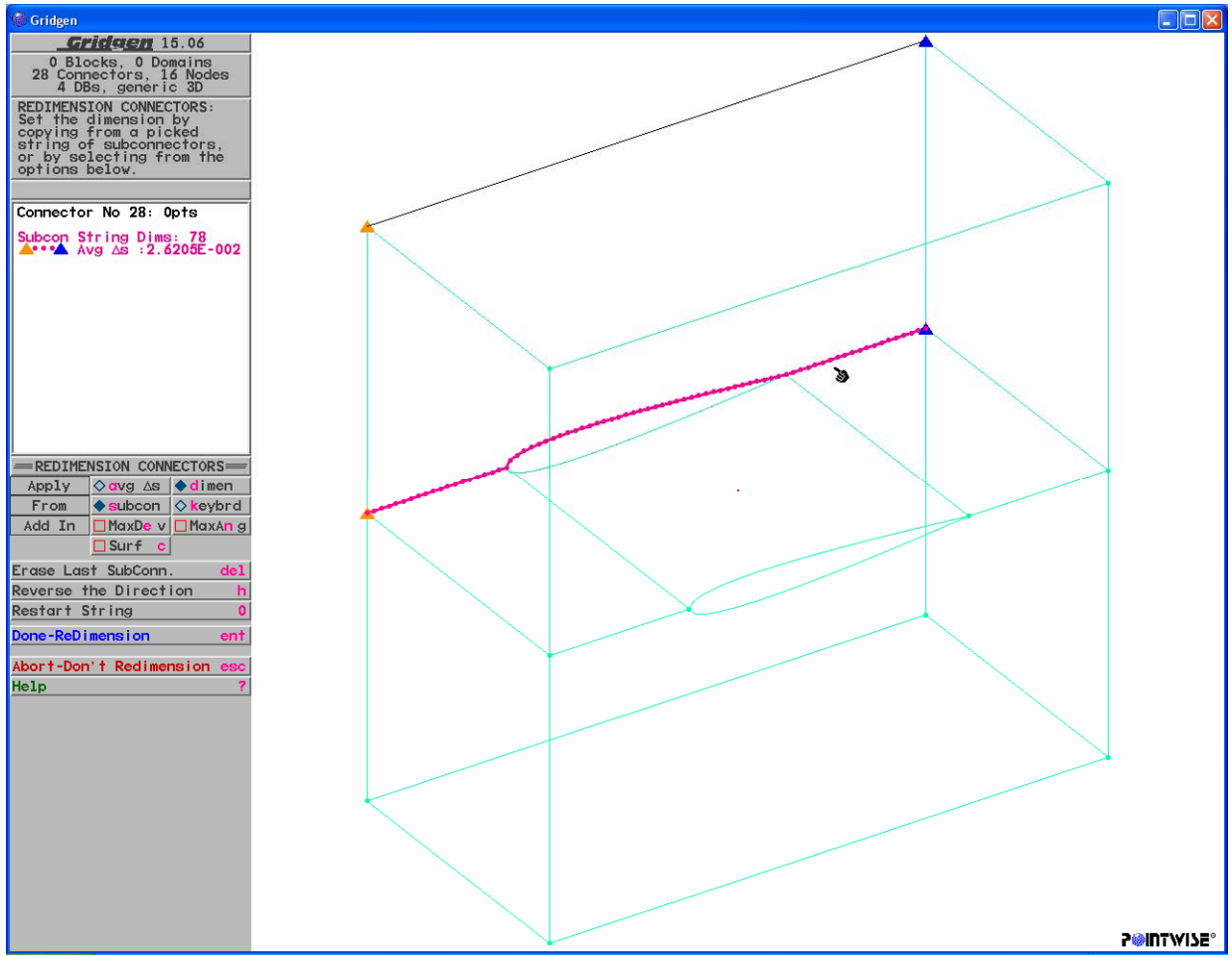
The connector(s) can then be redimensioned by selecting **keybrd** and typing in a dimension when prompted.

Alternatively, **subcon** can be selected. This allows the dimension to be copied from other connector(s) chosen via the mouse.

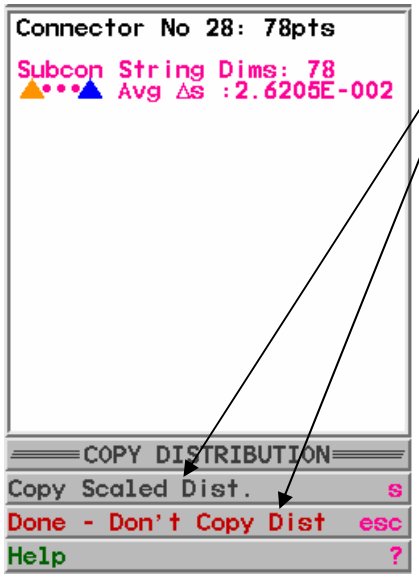


(Re)Dimension Grids (Continued)

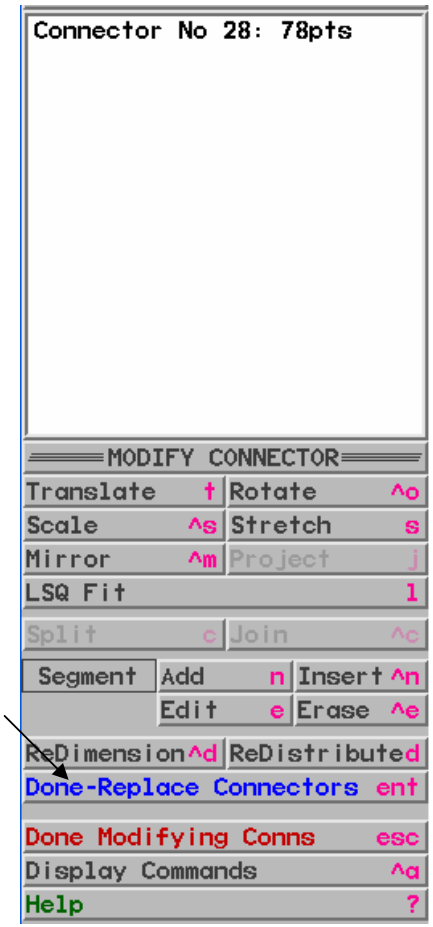
Here, **subcon** has been chosen and the three connectors highlighted in pink have been selected as the source of the dimension for the top connector. Note that the dimension and distribution (evenly spaced in this case) of the source connectors are shown and that their ends are marked with the same orange and blue triangles as the target. The latter mark the direction the copy will be made from the source to the target. **Done – Redimension** is picked to proceed with the redimensioning.



(Re)Dimension Grids (Continued)



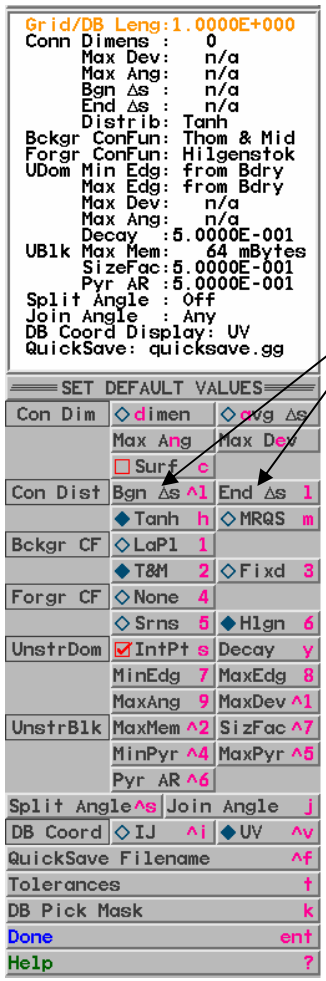
Next, the question of whether the distribution of the grid point spacing of the source connector(s) is also to be copied onto the target(s) must be answered.



Once that decision is made, the **MODIFY CONNECTOR** menu reappears, and the modified connector(s) can be saved by clicking **Done – Replace Connectors**.

(Re)Distribute Grids

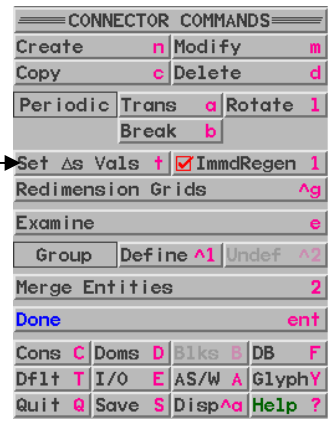
Gridgen Reliable CFD Meshing



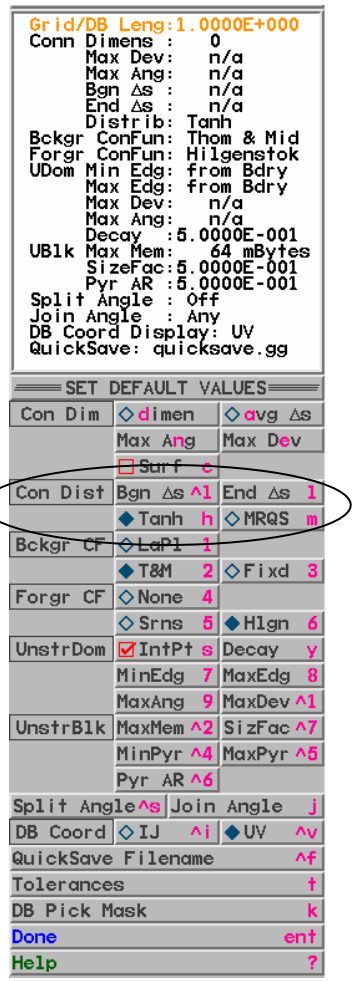
Distributing and redistributing grid points has also already been touched upon briefly.

In the **SET DEFAULT VALUES** menu, **Con Dist** can be used to establish a default distributions at the beginning (**Bgn Δs**) and end (**End Δs**) of all subsequently created connectors.

In addition, grid points along a previously defined connector can be redistributed with the **Set Δs Vals** command found in the **CONNECTOR COMMANDS** menu.



(Re)Distribute Grids (Continued)



There are several options for setting the distribution defaults in the **SET DEFAULT VALUES** menu:

Con Dist

Bgn Δs spacing constraint, i.e., distance between first two grid points, at beginning of connector: -1 ensures spacing continuity with adjacent (sub)connector (this cannot be set as a default); 0 removes constraint – if both ends unconstrained, points will be evenly spaced; >0 constrains spacing to entered value – must be less than 40% of connector length.

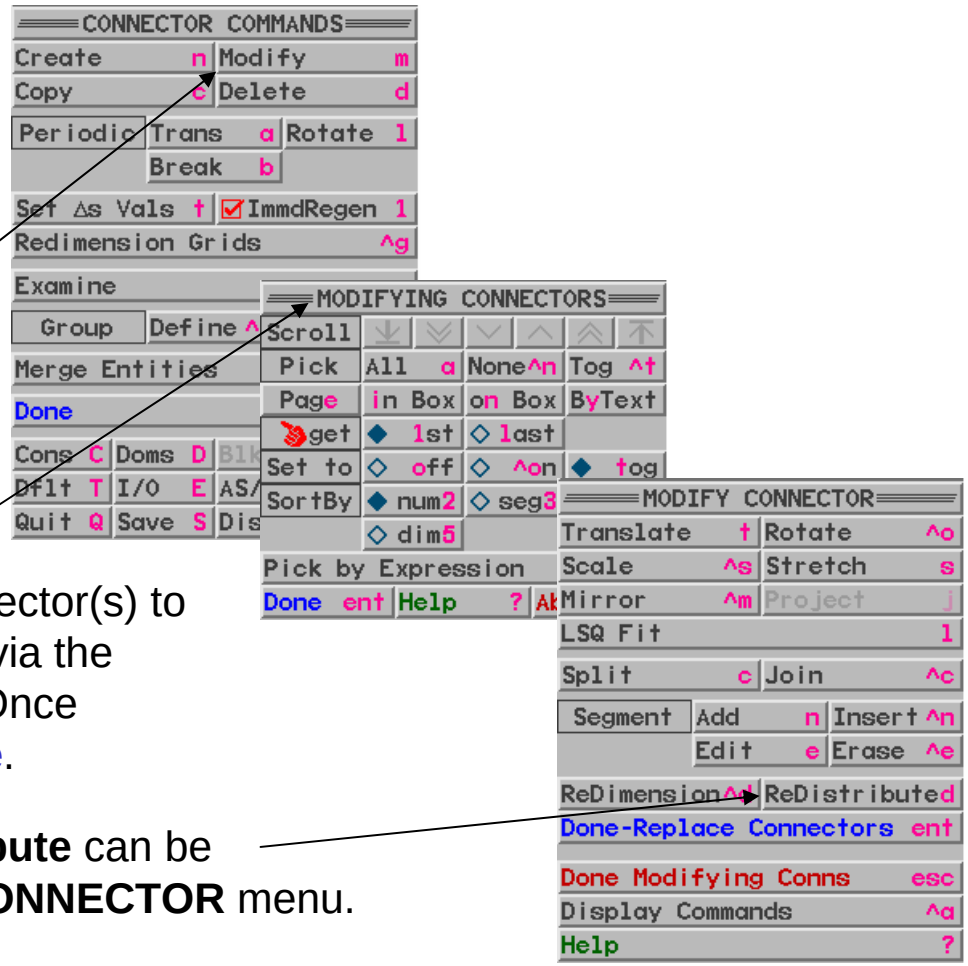
End Δs spacing constraint at end of connector.

Tanh Gridgen's default distribution function is the hyperbolic tangent. It can be applied if both ends of the connector are constrained or if only one end is.

MRQS The Monotonic Rational Quadratic Spline function can be applied to several subconnectors simultaneously. It forces a grid point to be fixed to a break point and automatically provides spacing continuity and variation continuity across the breakpoint. Gridgen will link two MRQS subconnectors if their spline variables and types are the same and if the spacing constraints across their common break point are equal.

(Re)Distribute Grids (Continued)

It's also possible to redistribute grid points via the **Modify** command in the **CONNECTOR COMMANDS** menu.



Modify is selected in the the **CONNECTOR COMMANDS** menu.

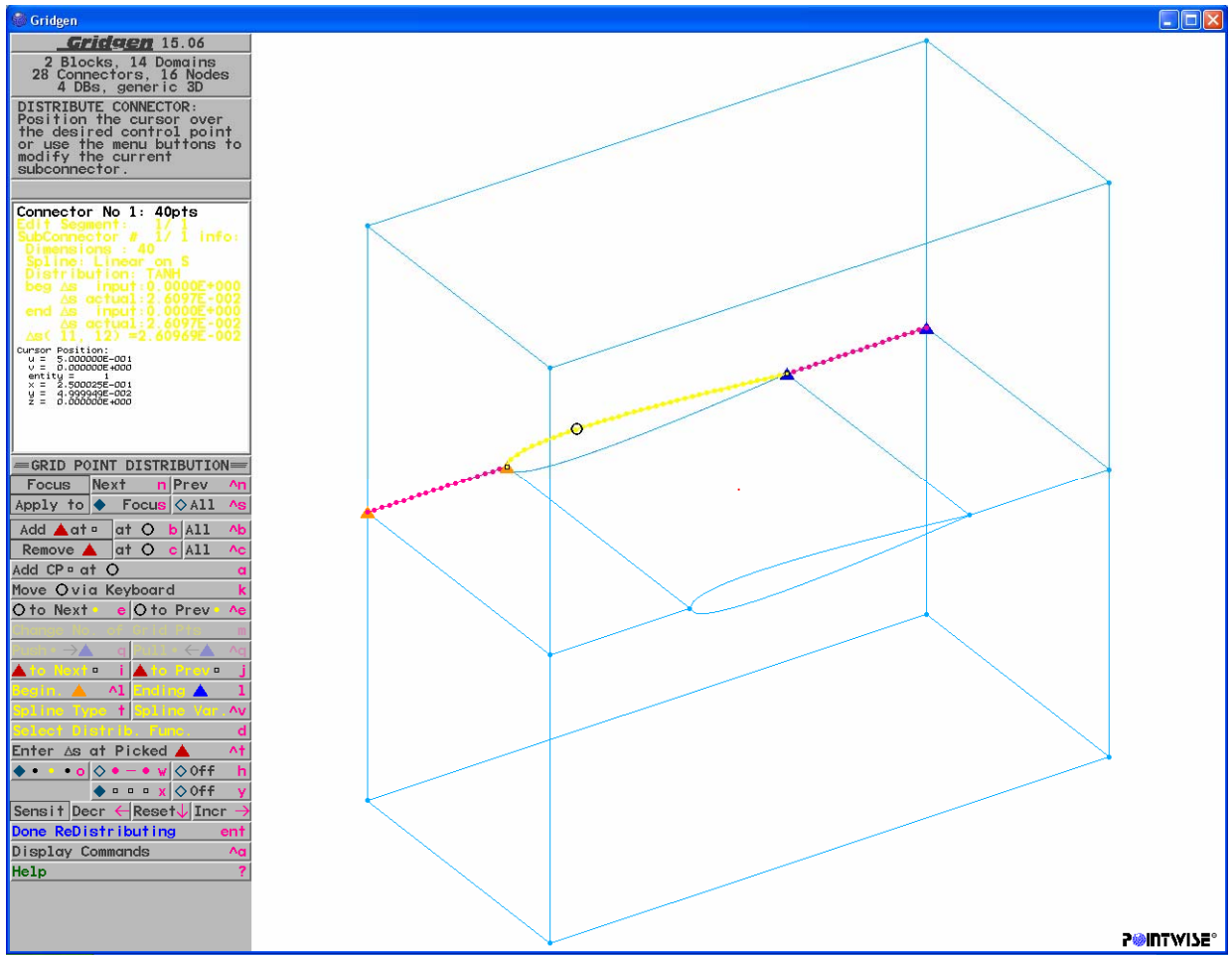
This opens the **MODIFYING CONNECTORS** menu. The connector(s) to be redistributed can be selected via the menu options or mouse picked. Once selection is completed click **Done**.

Once selected, **ReDistribute** can be picked in the **MODIFY CONNECTOR** menu.

(Re)Distribute Grids (Continued)

After **ReDistribute** is selected, the chosen connector(s) become highlighted with orange and blue triangles at their beginnings and ends, respectively. A cursor circle also appears along the connector(s).

A number of operations are now possible based on the commands in the **GRID POINT DISTRIBUTION** menu...



(Re)Distribute Grids (Continued)

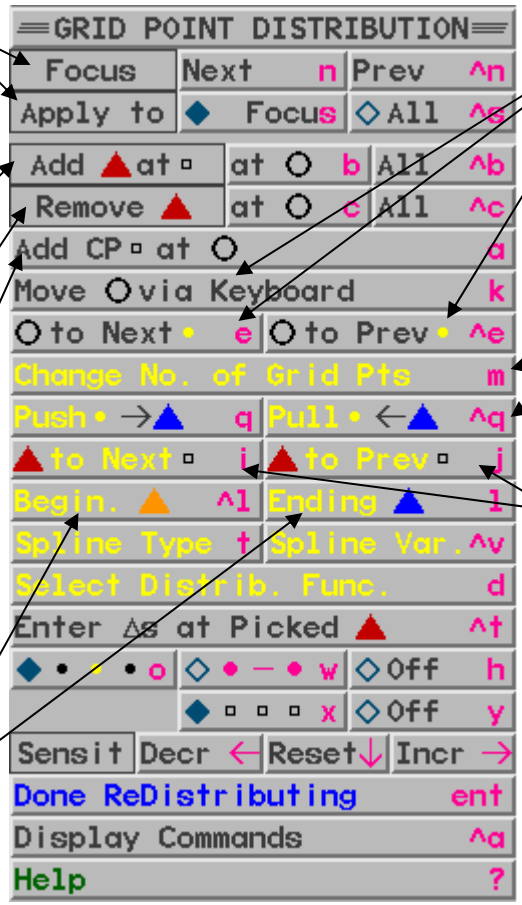
The **Focus** commands choose which connector is to be modified or if all are.

Add ▲at ▫ divides the connector at the control point **at** ○ (the cursor position) or at **All** control points along the connector.

Remove ▲ similarly reassembles the connector.

Add CP ▫ at ○ adds a control point at the cursor position.

Begin. ▲ and **Ending ▲** set the Δs values at the ends of the subconnector.



The cursor circle can be moved via the mouse, by entering coordinates, or by toggling from grid point to grid point.

The division of grid points between subconnectors can be changed by typing (top) or pushing/pulling from one to another, but the connector's total is fixed.

▲to Next ▫ and **▲to Prev ▫** shifts break point to one control point either towards end or beginning of connector.

(Re)Distribute Grids (Continued)

The **DISTRIBUTION FUNCTION** determines how points are distributed under the input spacing constraints:

Equal Spacing means just that.

TANH (hyperbolic tangent) is the Gridgen's default.

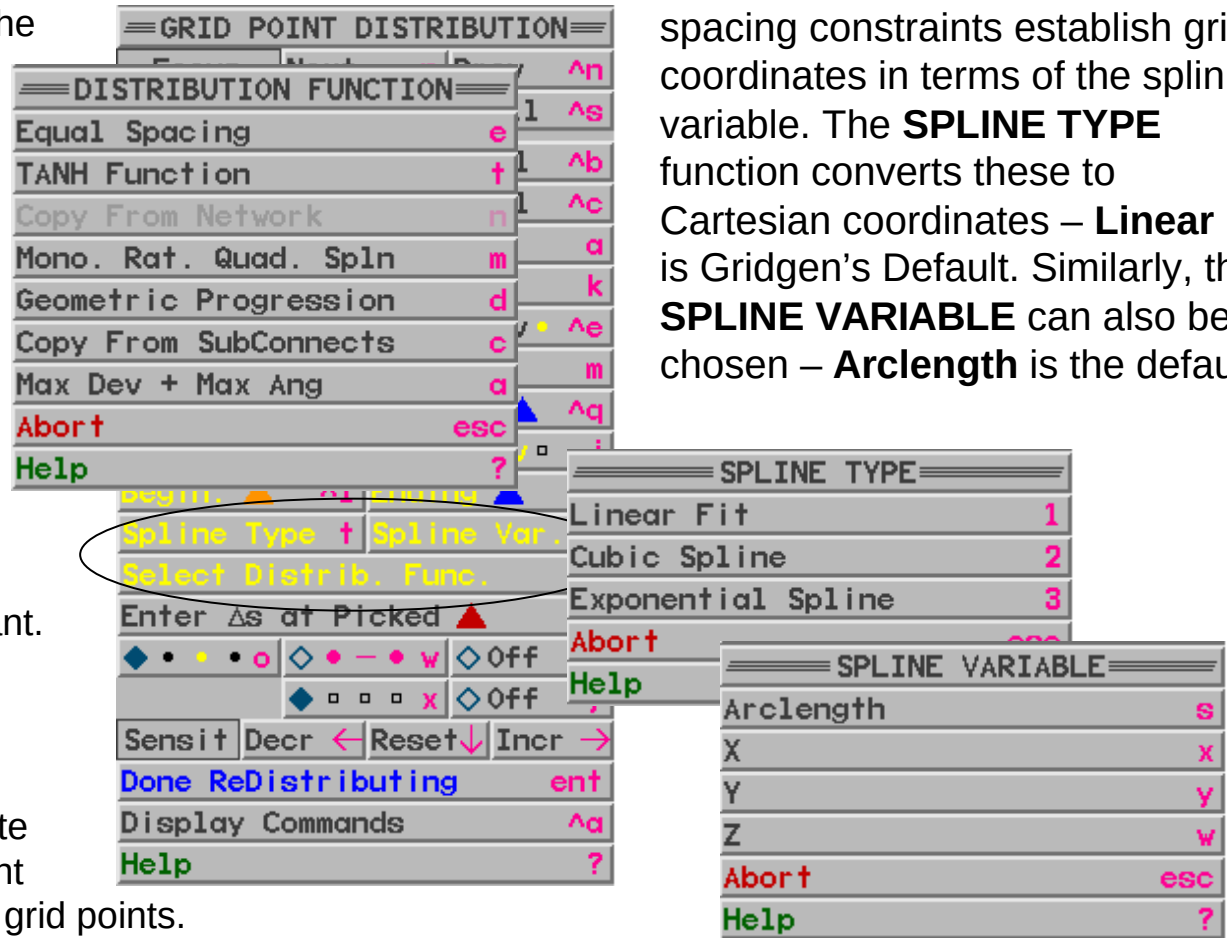
Copy From Network requires a database entity to distribute to (with various criteria).

MRQS, as noted before, works across multiple subconnectors.

Geometric Progression gives a one-sided distribution where ratio between adjacent points is constant.

Copy from SubConnects copies distribution from picked subconnector(s).

Max Dev + Max Ang limits discrete connector deviations from segment shape and turning angle between grid points.



The distribution function and spacing constraints establish grid coordinates in terms of the splining variable. The **SPLINE TYPE** function converts these to Cartesian coordinates – **Linear Fit** is Gridgen's Default. Similarly, the **SPLINE VARIABLE** can also be chosen – **Arclength** is the default.

(Re)Distribute Grids (Continued)

Wrapping up the **GRID POINT DISTRIBUTION** menu...

Enter Δs at Picked ▲ applies specified spacing constraint on both sides of break point where cursor is currently located.

These toggles set different options for rendering grid points (as shown) or turning off their display.

Control points can be rendered as shown or not rendered at all.

These control mouse sensitivity with respect to cursor circle movement.

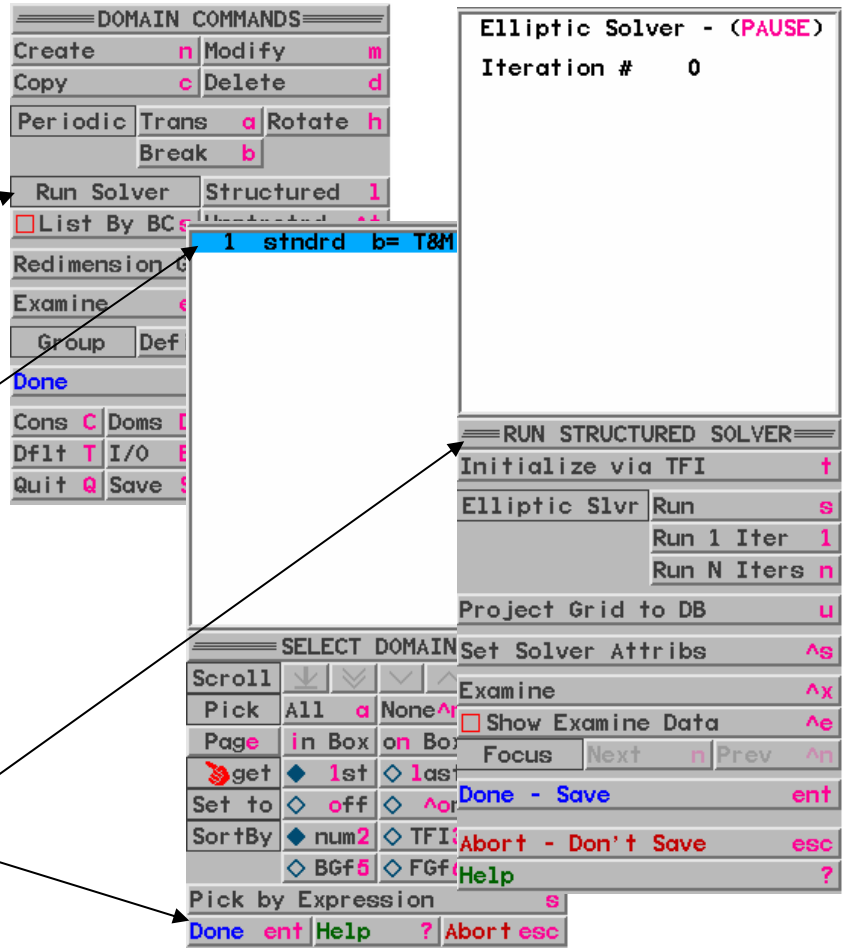
The Structured Solver

Use of the structured solver has also been touched upon briefly in the previous session. For domains, it's mainly used to change the interior distribution of points to smooth mesh or improve orthogonality.

To run the solver, first choose **Run Solver Structured** from the **DOMAIN COMMANDS** menu.

The domain is next picked – in this case by highlighting the entry in the browser window. Alternatively the mouse can be used for selecting the domain.

Once **Done** is selected, the **RUN STRUCTURED SOLVER** menu appears.



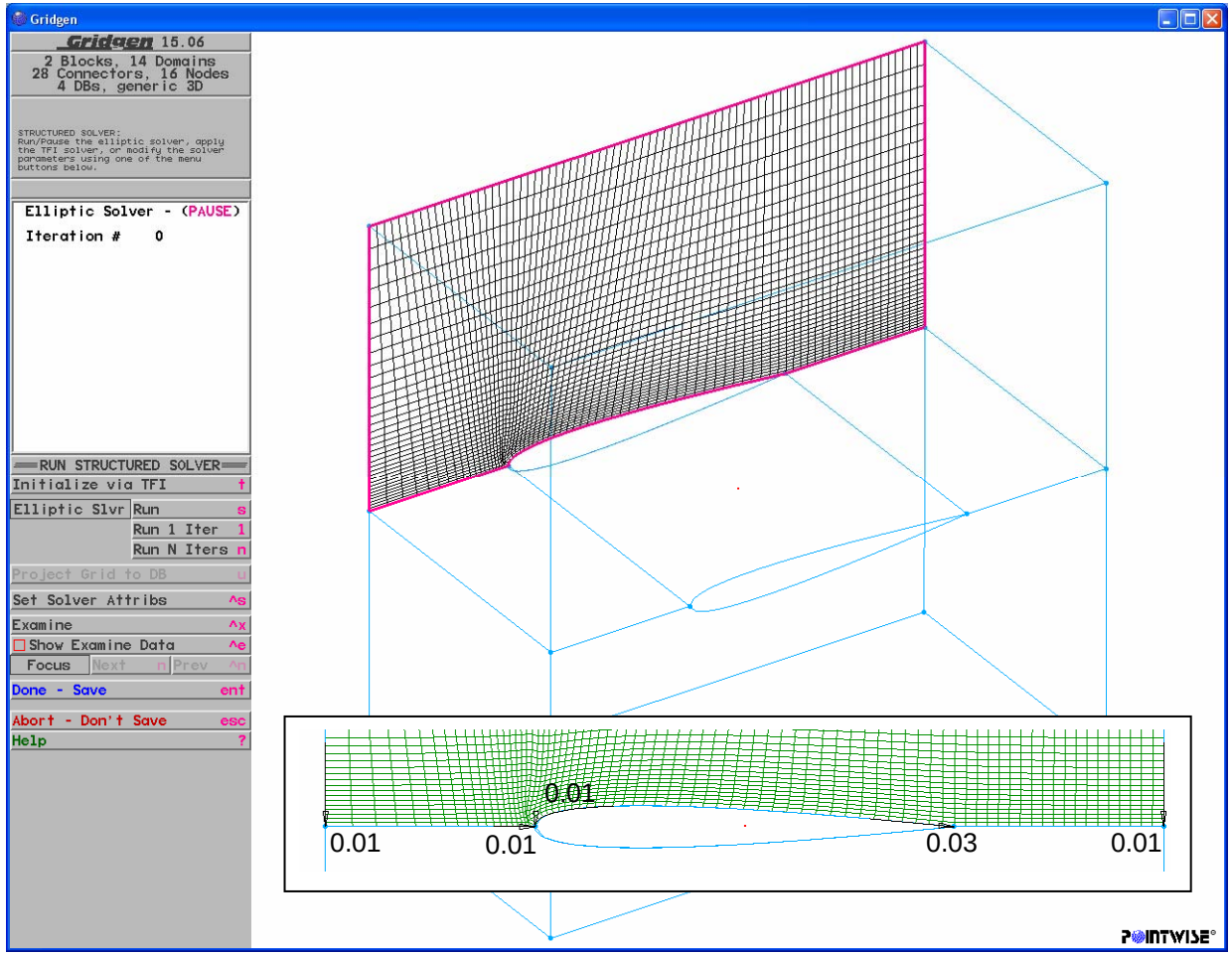
The Structured Solver (Continued)

When to Run the Structured Solver

- When strict orthogonality is needed adjacent to a wall type boundary.
- When cell quality needs improvement.
- When cell problems adjacent to the boundary of a domain or block are generally due to connector or domain problems, respectively, and when those problems cannot be fixed directly.

The Structured Solver (Continued)

As an example, the airfoil mesh has been modified with various spacing constraints (see the inset diagram) to increase the mesh resolution near the leading edge and in the boundary layer region. Unfortunately, an undesirable consequence is the crowding of the cells near the leading edge. The structured solver could be utilized to correct this.



The Structured Solver (Continued)

Here's a look at the options in the **RUN STRUCTURED SOLVER** menu:

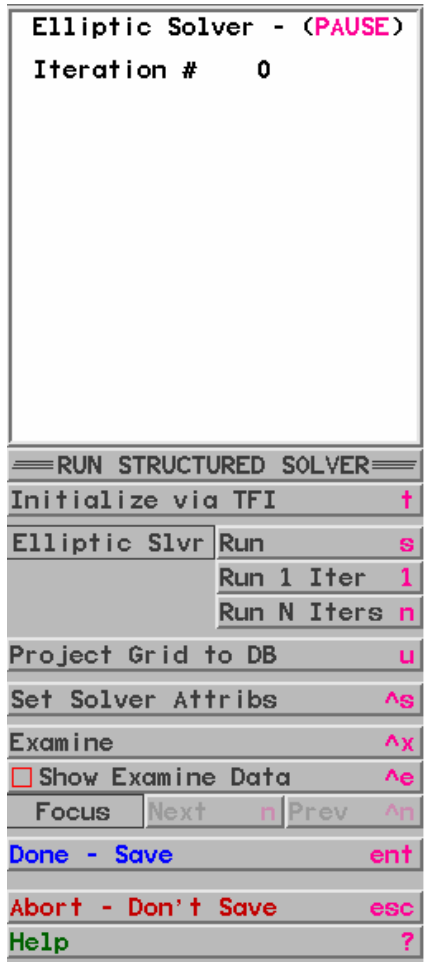
Initialize via TFI (transfinite interpolation) "resets" the domain or block grid in question with the last TFI method used. TFI is an algebraic approach to establish the interior grid point locations. It's what Gridgen uses when a domain or block is first created. There are several mathematical methods available as will be discussed later.

Elliptic Solver runs the elliptic PDE solver to smooth grid either indefinitely (**Run**) until paused, one iteration (**Run 1 Iter**), or N iterations (**Run N Iters**).

Project Grid to DB projects grids with database constrained shapes to their database.

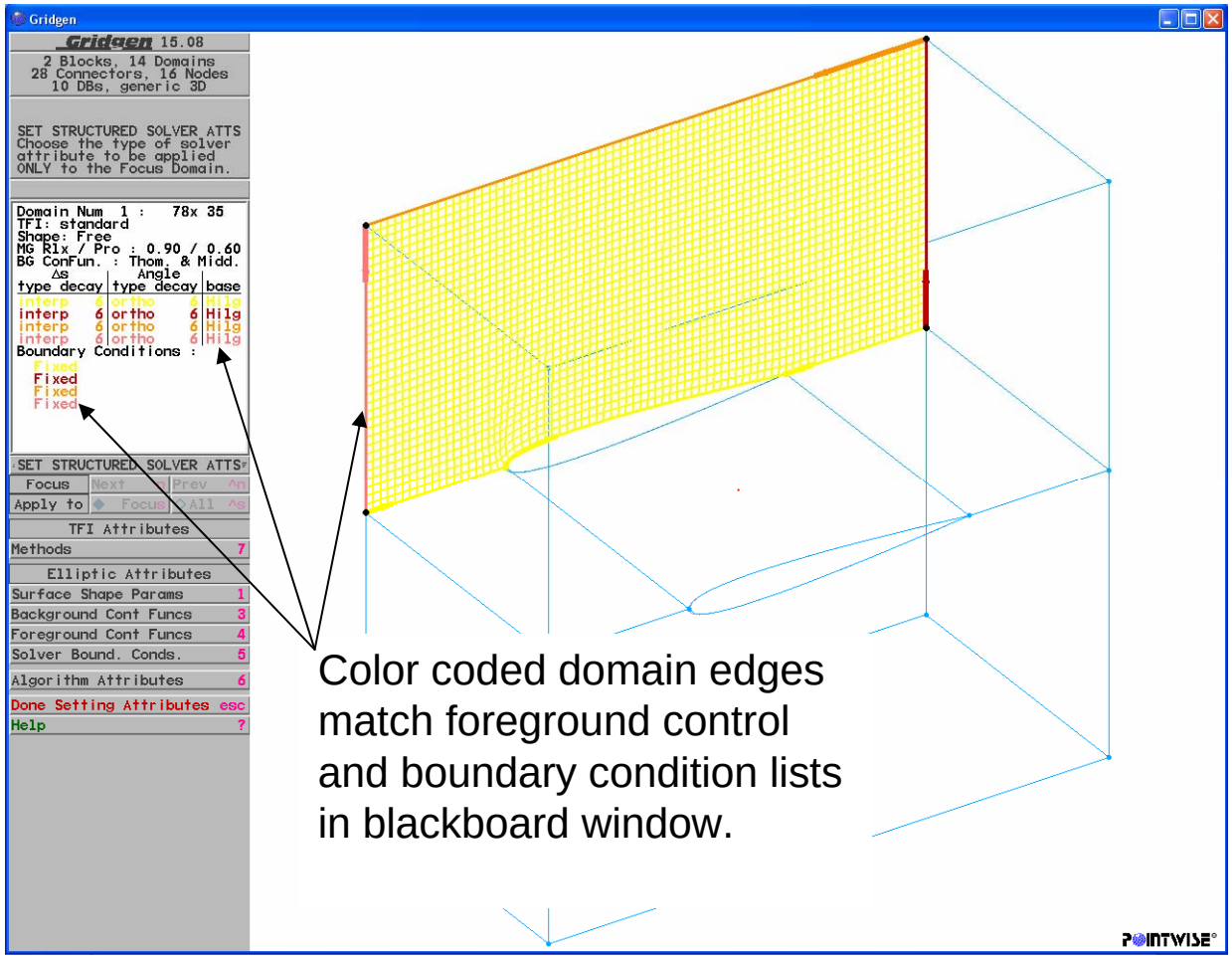
Set Solver Attribs opens a menu for setting solver attributes.

Examine opens a menu of grid quality inspection tools, while **Show Examine Data** will display grid quality information when solver is paused.

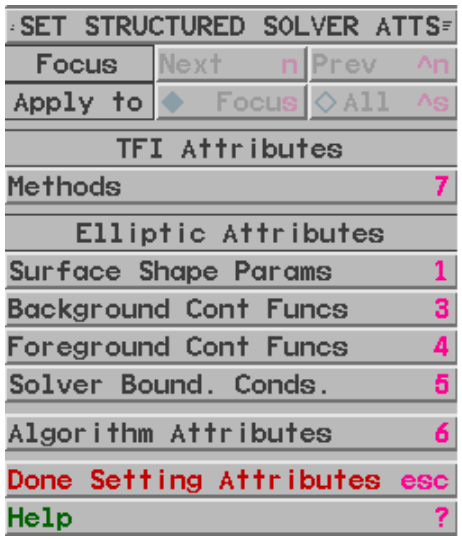


The Structured Solver (Continued)

Selecting **Set Solver Attribs** in the **RUN STRUCTURED SOLVER** opens this menu. Each button here opens yet another menu to control various TFI and elliptic solver attributes.



The Structured Solver (Continued)



TFI Attributes

Methods specifies the mathematical method used to initialize the grid point distribution.

Elliptic Attributes

Surface Shape Params define how the shape of the selected surface grids is computed.

Background Cont Funcs influences the distribution of grid points in the grid's interior.

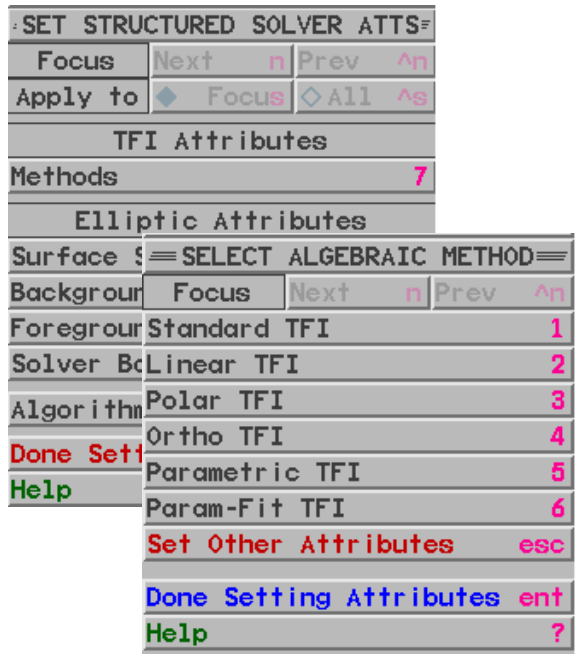
Foreground Cont Funcs influences the distribution of grid points near the grid's boundaries.

Solver Bound. Conds. controls grid point movement on grid boundaries during elliptic refinement.

Algorithm Attributes controls the methodology and convergence rate of the iterative PDE solution.

The Structured Solver (Continued)

Selecting **TFI Attributes Methods** brings up the **SELECT ALGEBRAIC METHOD** menu with the following choices:



Standard TFI – (third choice for domain initialization) arc-length based interpolants provide optimum algebraic domain initialization.

Linear TFI – original transfinite interpolation method developed for Gridgen, generally inferior to standard TFI.

Polar TFI – initialization performed in polar coordinate system according to a user defined axis.

Ortho TFI – provides minimal additional orthogonality at domain edges.

Parametric TFI – (first choice for domain initialization) a standard transfinite interpolation is calculated in the parametric coordinates of an underlying database surface when all bounding connectors are associated to that surface. Method ensures domain has database surface shape.

Param-Fit TFI – (second choice for domain initialization) extension of above to include special cases where some bounding connectors, while physically in the same space as the underlying database surface, are actually associated to different database entities.

The Structured Solver (Continued)

Selecting **Elliptic Attributes Surface Shape Params** brings up the **SELECT SHAPE TYPE METHOD** menu with the following choices:

Free – (default for standard TFI domains) solver runs with no constraint on domain shape.

Fixed – original domain shape is temporarily parameterized so original shape can be maintained.

DB – (default for Parametric/Param-Fit TFI domains) shape is constrained to underlying database surface(s). Picking this option reveals:

Proj Typ

Lin-Def – an average of the domain normals at the four corners is used for the projection vector for the linear projection method.

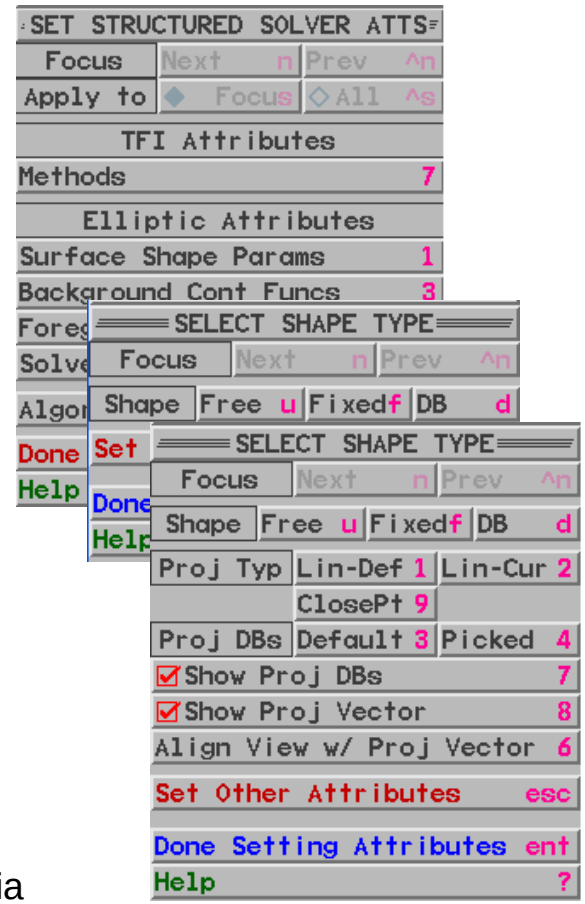
Lin-Cur – the current view orientation is set and saved as the projection vector for the linear projection method.

ClosePt – the closest point projection method is used.

Proj DBs

Default – a default set of database surfaces is chosen based on domain/connector associativities.

Picked – user is allowed to interactively choose the database surfaces via picking in the display window or browser.



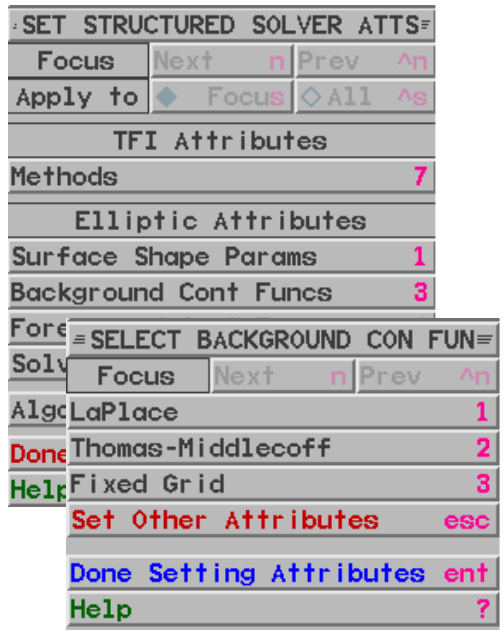
The Structured Solver (Continued)

Selecting **Elliptic Attributes Background Cont Funcs** brings up the **SELECT BACKGROUND CON FUN** menu with the following choices:

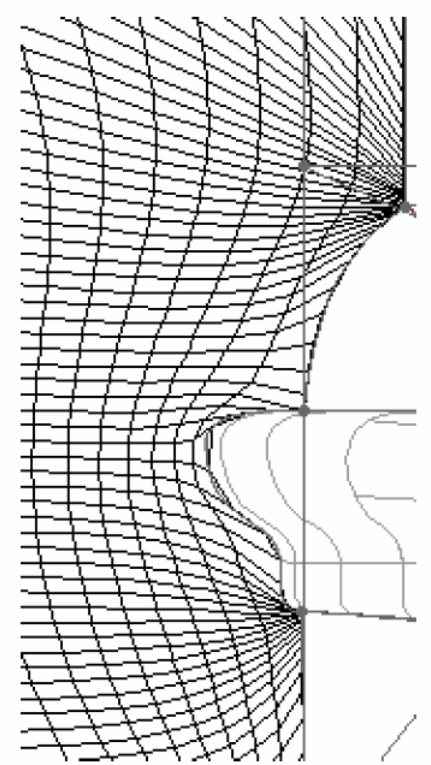
LaPlace – maximize smoothness in the grid interior at the cost of grid point clustering and orthogonality.

Thomas-Middlecoff – (default) produces smooth grid while maintaining boundary point clustering into the interior of the domain.

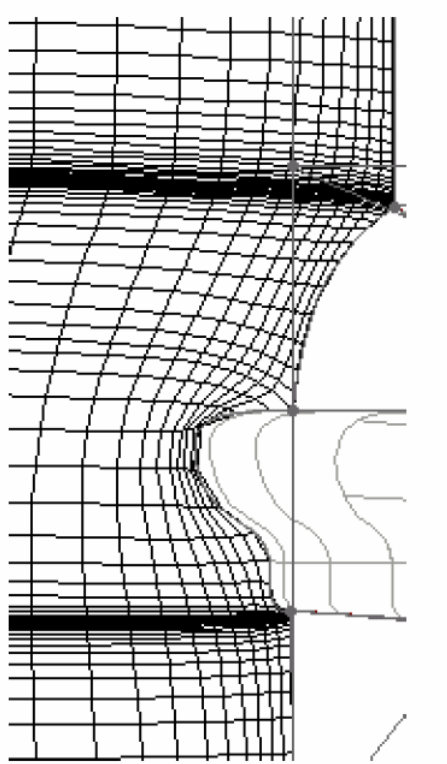
Fixed Grid – This method looks for and corrects local slope discontinuities in the grid.



The Structured Solver (Continued)



LaPlace



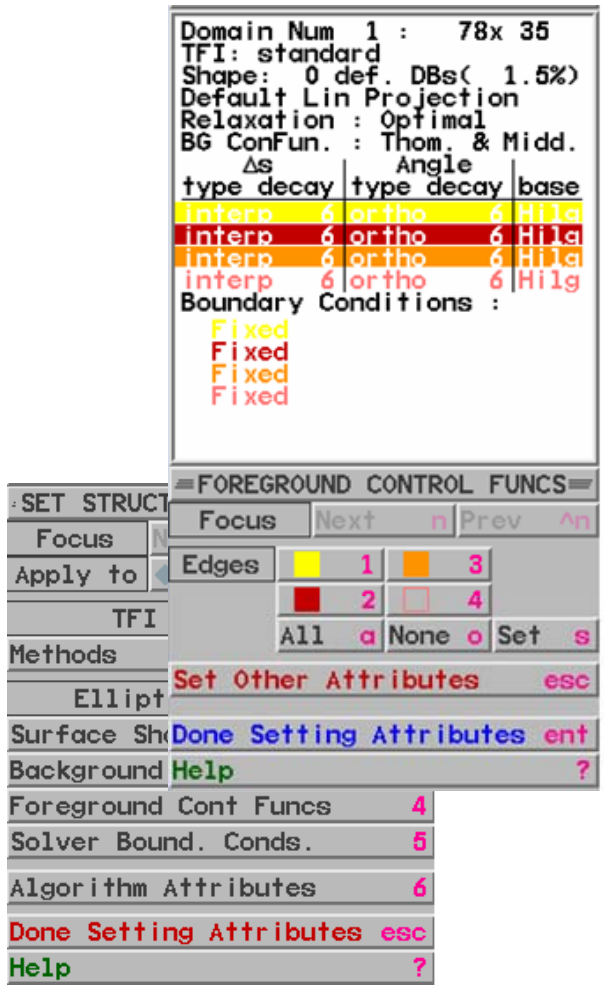
Thomas-Middlecoff

The Structured Solver (Continued)

Selecting **Elliptic Attributes Foreground Cont Funcs** brings up the **FOREGROUND CONTROL FUNCS** menu:

Edges – choose the edge(s) that require change to their foreground control settings or boundary conditions by toggling the corresponding color coded squares (when selected, the box outlines are filled). **All** or **None** can be selected as appropriate to save time.

Set – will reveal the **SET ON EDGES** menu.



The Structured Solver (Continued)

SET ON EDGES options:

Form refers to nature of control over near boundary grid points:

Off – do not use foreground controls.

On – (default) use foreground controls.

Hilgens – (default) method of von Lavante-Hilgenstock-White, providing greatest control over spacing and orthogonality.

Sorensn – method by Steger-Sorensen, providing smoother grids at some cost to edge orthogonality and spacing.

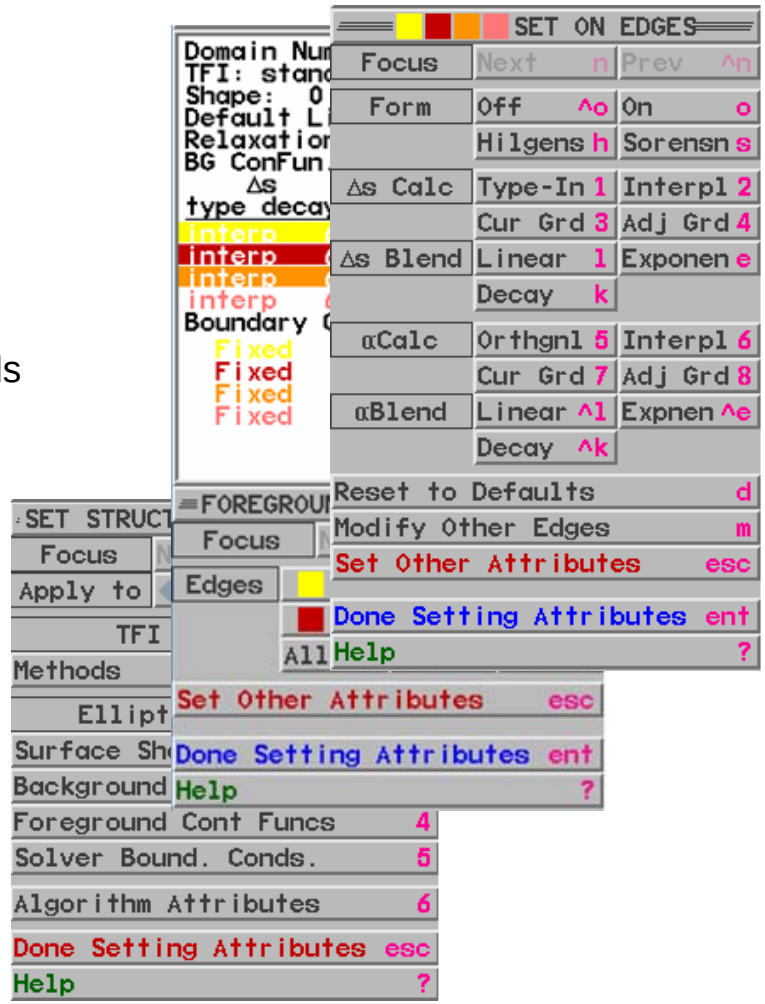
Δs Calc refers to distance between boundary edge/face grid points and the first layer of grid points inward:

Type-In – enter a spacing value via the keyboard.

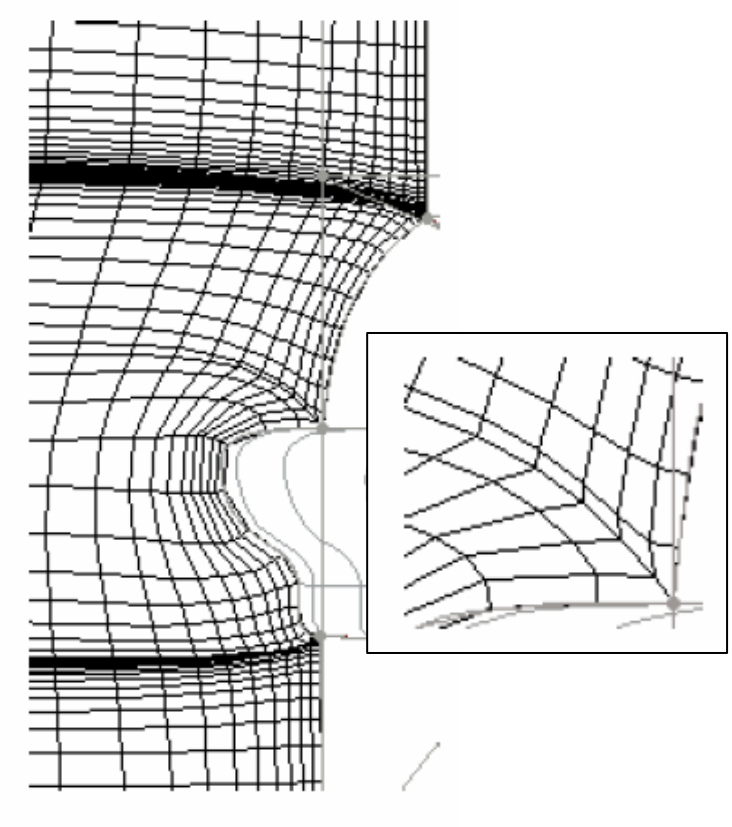
Interpl – (default) linearly interpolate the spacings from the ends of the edge along the length of the edge.

Cur Grd – store and maintain the original spacings along the length of the edge.

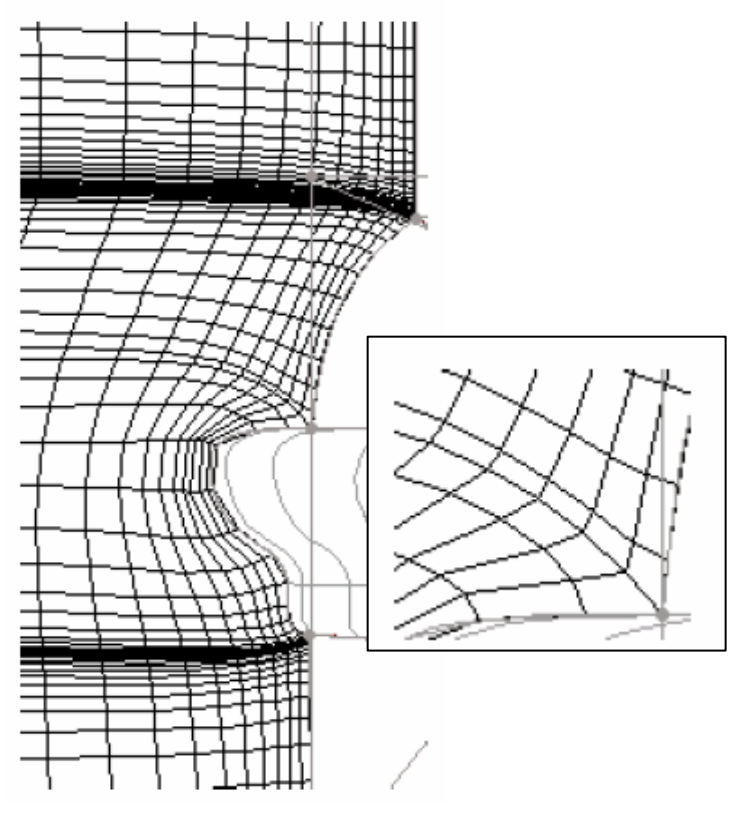
Adj Grd – use the spacings from an adjacent domain along the length of the edge. The adjacent domain must be in the solver simultaneously, reverting to **Cur Grd** if adjacent domain not available.



The Structured Solver (Continued)



Hilgens



Sorensn

The Structured Solver (Continued)

SET ON EDGES options (continued):

Δs Blend controls blending of spacing constraints into grid's interior:

Linear – blending is a linear function of grid point index from the edge.

Exponen – (default) blending is a decaying exponential function.

Decay – number of grid lines into the interior at which foreground control decays to 10% of its maximum edge contribution.

αCalc refers to angle that transverse grid lines make with boundary:

Orthgnl – (default) 90 degree grid line-edge intersections will be enforced as reasonable.

Interpl – linear interpolates spacings from edge ends.

Cur Grd – store and maintain original spacings.

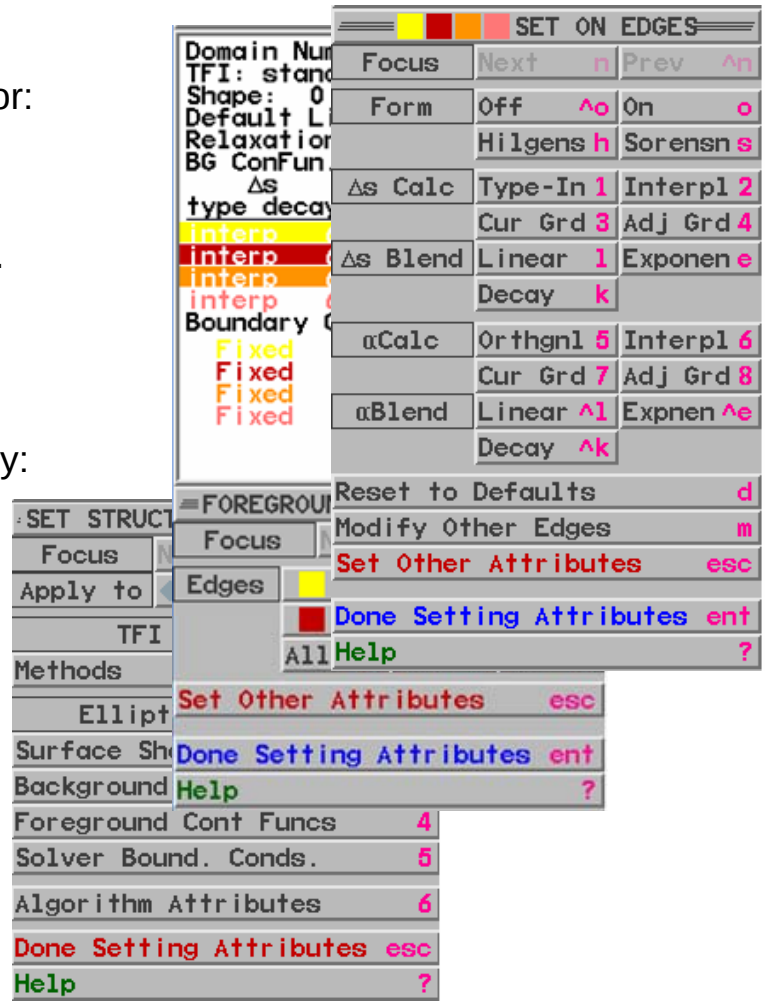
Adj Grd – uses spacing from adjacent domain.

αBlend controls blending of angle constraints into interior:

Linear – see above.

Exponen – see above.

Decay – see above.

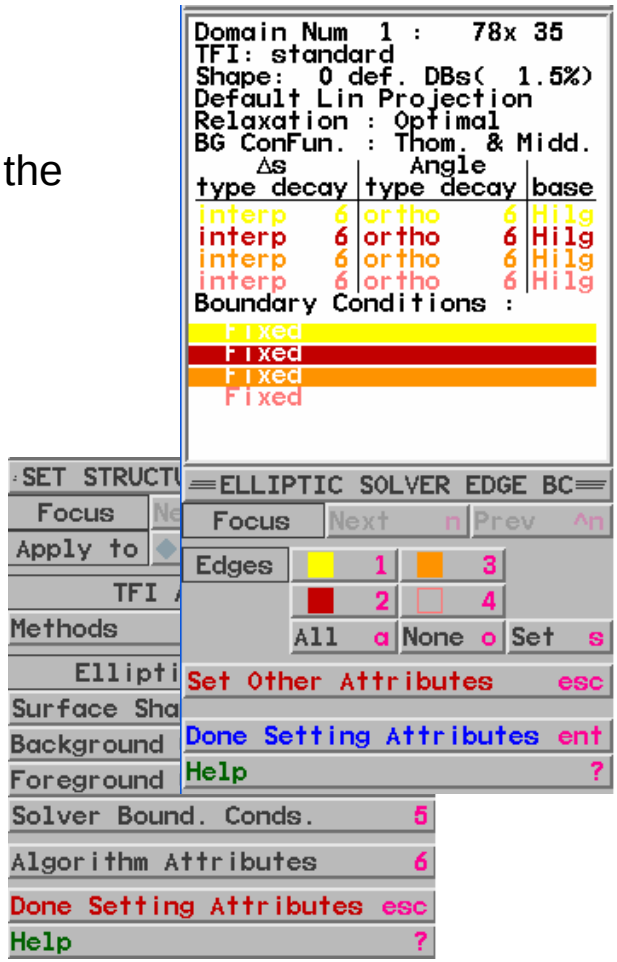


The Structured Solver (Continued)

Selecting **Solver Bound. Conds.** brings up the **ELLIPTIC SOLVER EDGE BC** menu:

Edges – choose the edge(s) for which you wish to change the control settings for their grid point distributions by toggling the corresponding color coded squares (when selected, the box outlines are filled). **All** or **None** can be selected as appropriate to save time.

Set – will reveal the **SELECT FOR EDGE** menu.



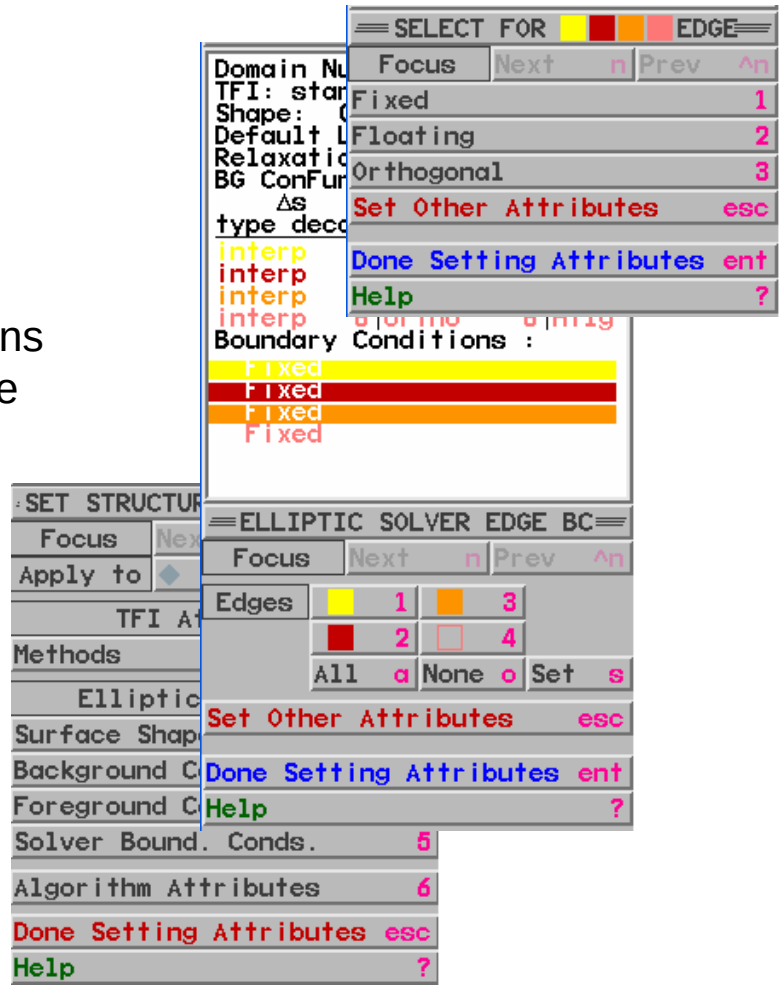
The Structured Solver (Continued)

The **SELECT FOR EDGE** menu provides the following options:

Fixed – (default) edge grid points will not be allowed to move.

Floating – common connectors between adjacent domains will change shape and distribution as if an interior grid line when the adjacent domains are all run in the solver.

Orthogonal – grid point distribution along the edge will be allowed to change in order to achieve the best possible orthogonality.



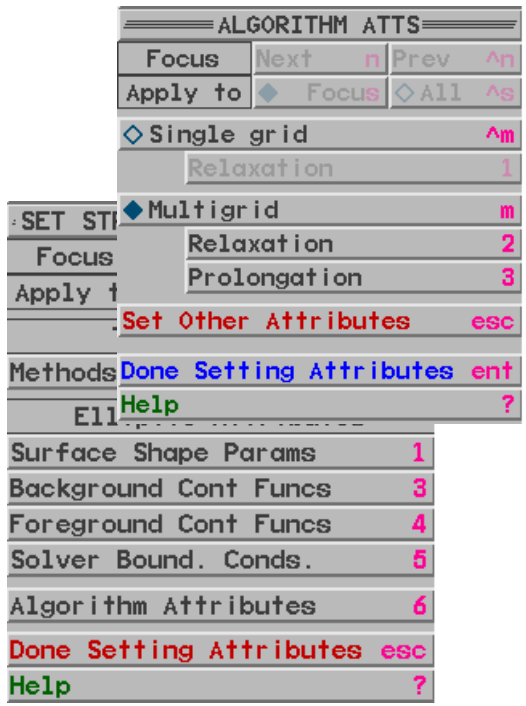
The Structured Solver (Continued)

Selecting **Algorithm Attributes** brings up the **Algorithm Atts** menu with the following choices:

Single grid – solves the elliptic PDEs on the selected entity based on the focus grid's relaxation factor, defined by selecting the **Relaxation** tab below (grayed out when **Multigrid** is toggled).

Multigrid – (default) solves the elliptic PDEs on the selected entity via a multigrid approach, i.e., based on the actual mesh and two coarser intermediates that Gridgen creates. This technique achieves convergence in fewer steps than the single grid method. Two controls are provided:

- **Relaxation** – controls the over-relaxation of the solution. The default is 0.9. Lower values increase robustness but will require more iterations.
- **Prolongation** – controls the transfer of grid point changes from the coarse grid solutions back to the initial finer grid. The default is 0.6. Lower values will again increase robustness at the price of more iterations.



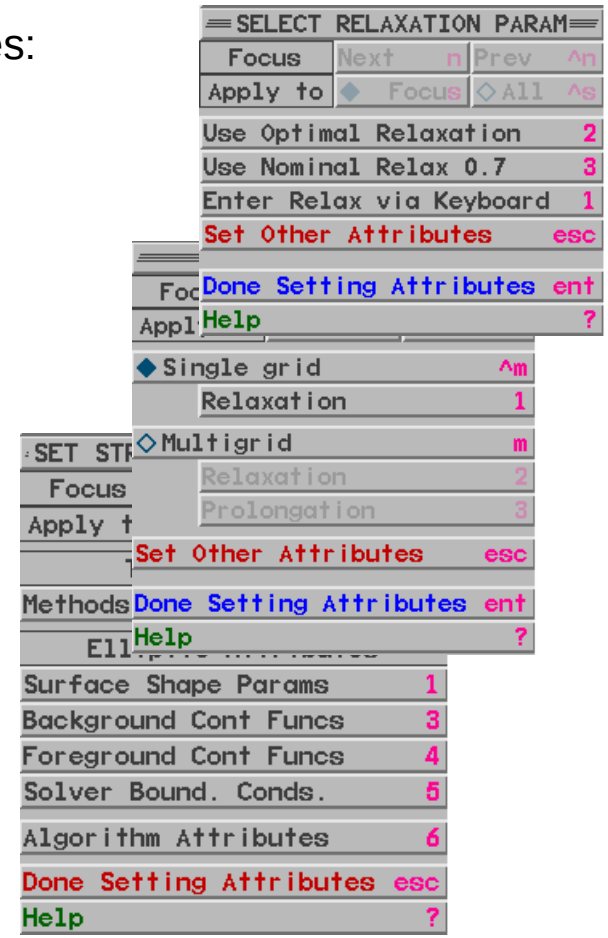
The Structured Solver (Continued)

Toggling **Single Grid** and selecting **Relaxation** brings up the **SELECT RELAXATION PARAM** menu with the following choices:

Use Optimal Relaxation – (default) the highest possible relaxation value is used at each grid point to accelerate the elliptic solution.

Use Nominal Relax 0.7 – a conservative mean value is used at all grid points, sometimes necessary for smooth convergence when using **Shape DB**.

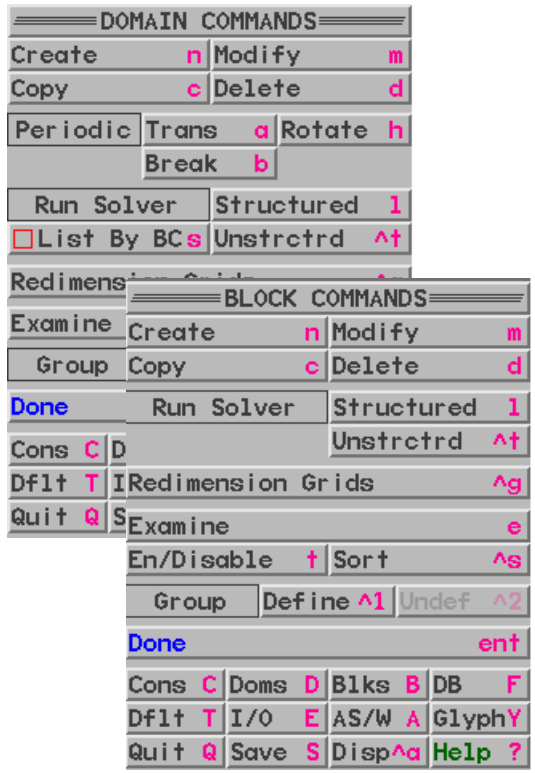
Enter Relax via Keyboard – enter a new value, between 0 and 2, directly when prompted via the keyboard.



The Structured Solver (Continued)

2D to 3D Differences

- Six faces instead of four edges.
- **No Surface Shape Params** or **Project Grid to DB** necessary.
- **No Adj Grd** foreground control.
- Only **Fixed** boundary condition available.



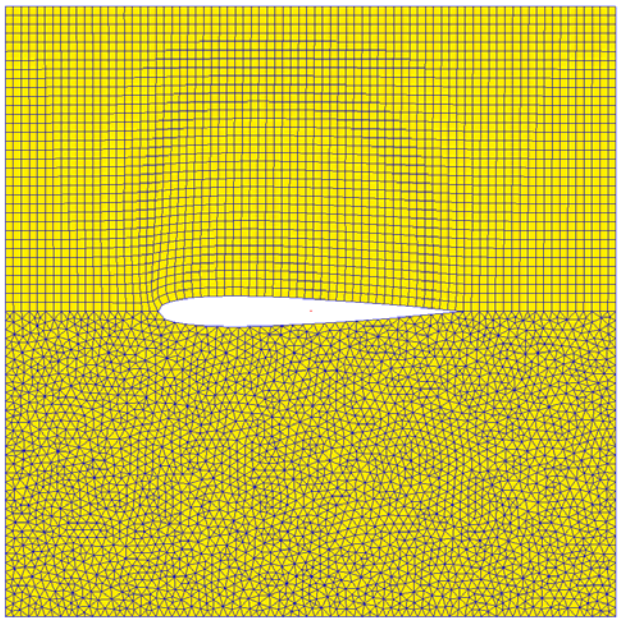
- In this session:
  - Building 3D Structured Grids.
  - (Re)Dimensioning and (Re)Distributing Structured Grids.
  - Tutorial: “Swept Ramp: Basic Multi Block Structured Grid.”

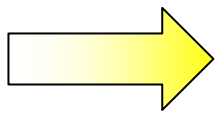
Gridgen**Training**

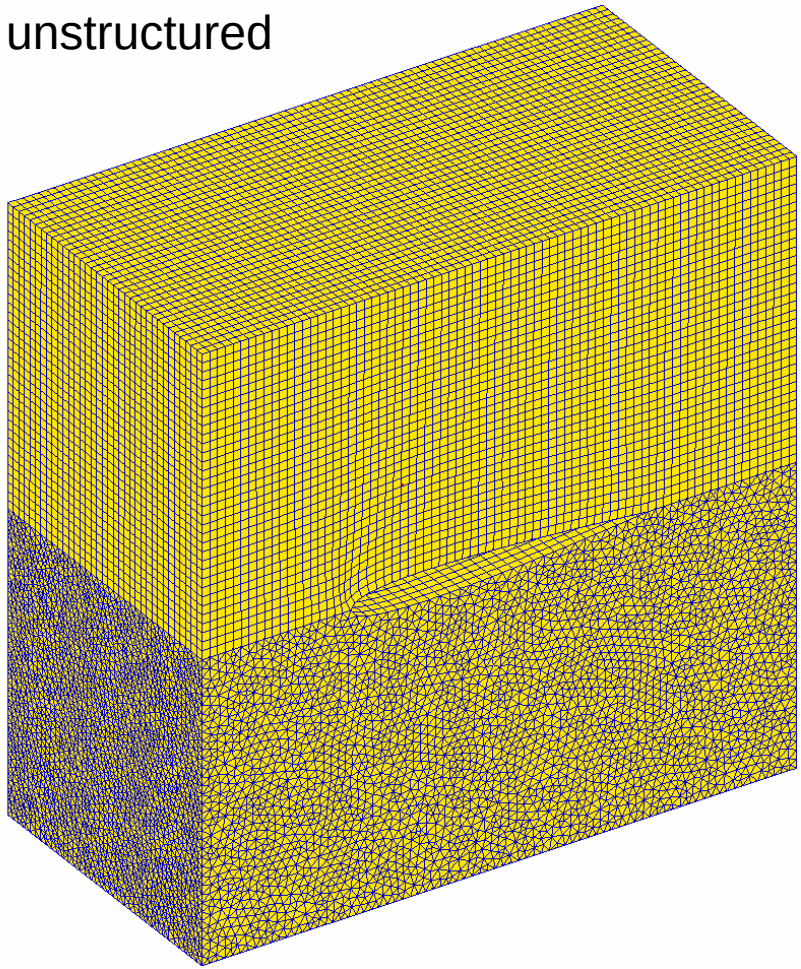
- In this session:
  - Building 3D Unstructured Grids.
  - (Re)Dimensioning and (Re)Distributing Unstructured Grids.
  - Examining and Correcting Grids.
  - Tutorial: “Pierced Elbow: Basic Unstructured Grid.”

Building 3D Unstructured Grids

Turning again to the hybrid airfoil mesh, the 3D unstructured block will now be constructed.







The “Bottom-Up”Gridgen Process

Select Analysis Software and Mesh Dimensionality

⇩

Import and Manipulate Database Geometry

⇩

Create Segments and Connectors

⇩

Dimension Connectors and Distribute Grid Points

⇩

Assemble Domains

⇩

Assemble Blocks

⇩

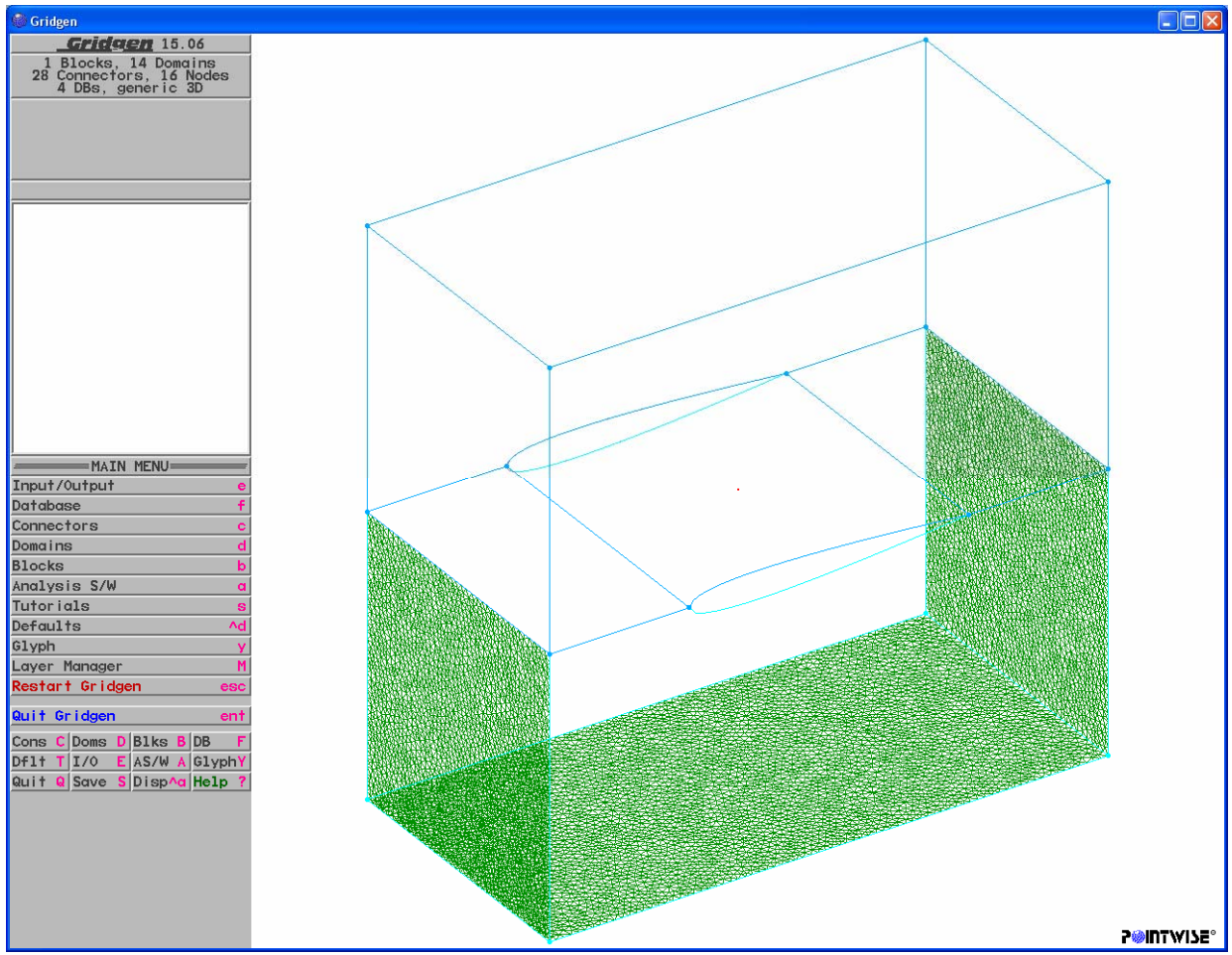
Setup CFD Analysis Boundaries

⇩

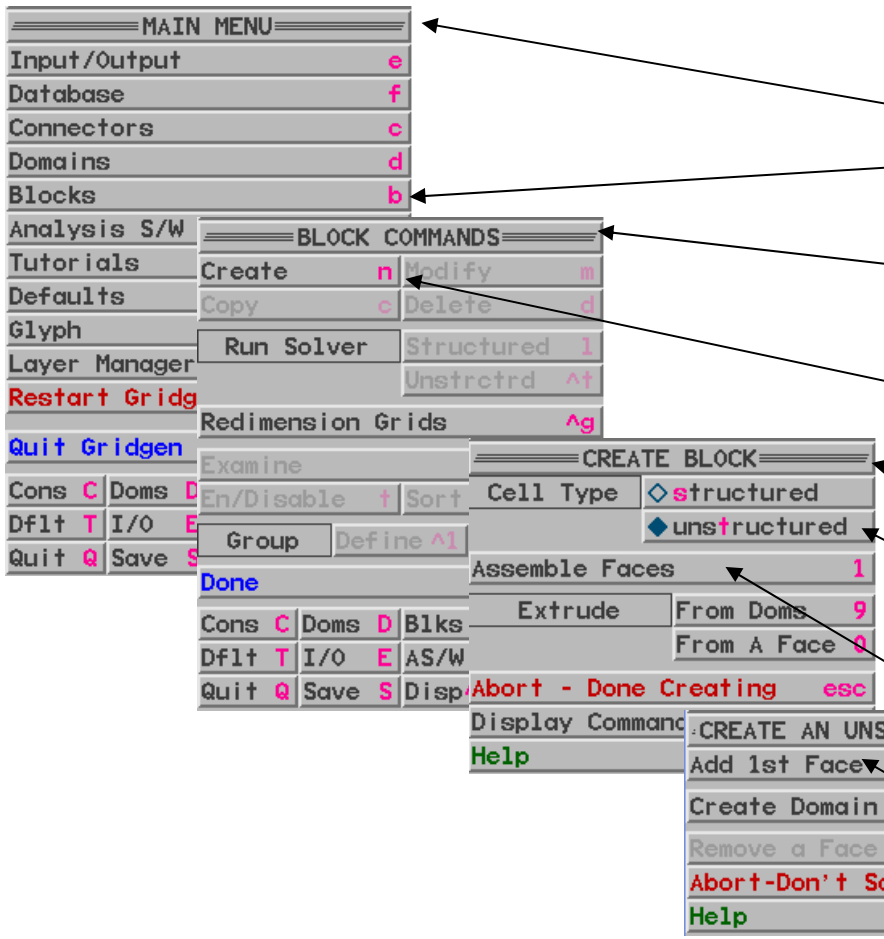
Export and Save Files

Assemble Domains

Most of the faces needed for the unstructured block beneath the airfoil have already been created. Only the bottom and two end domains have yet to be formed. Since this process has already been demonstrated, it will not be shown in detail again. Just remember using **Auto Save** and **Auto Complete** will save time. Here is the scene after these three domains have been formed.



Assemble Blocks



Finally, it's time to assemble the block.

In the **MAIN MENU**, **Blocks** is selected.

This opens the **BLOCK COMMANDS** submenu.

Here **Create** is selected.

This opens the **CREATE BLOCK** submenu.

Be sure that **Cell Type** is set to **unstructured**

Then select **Assemble Faces**.

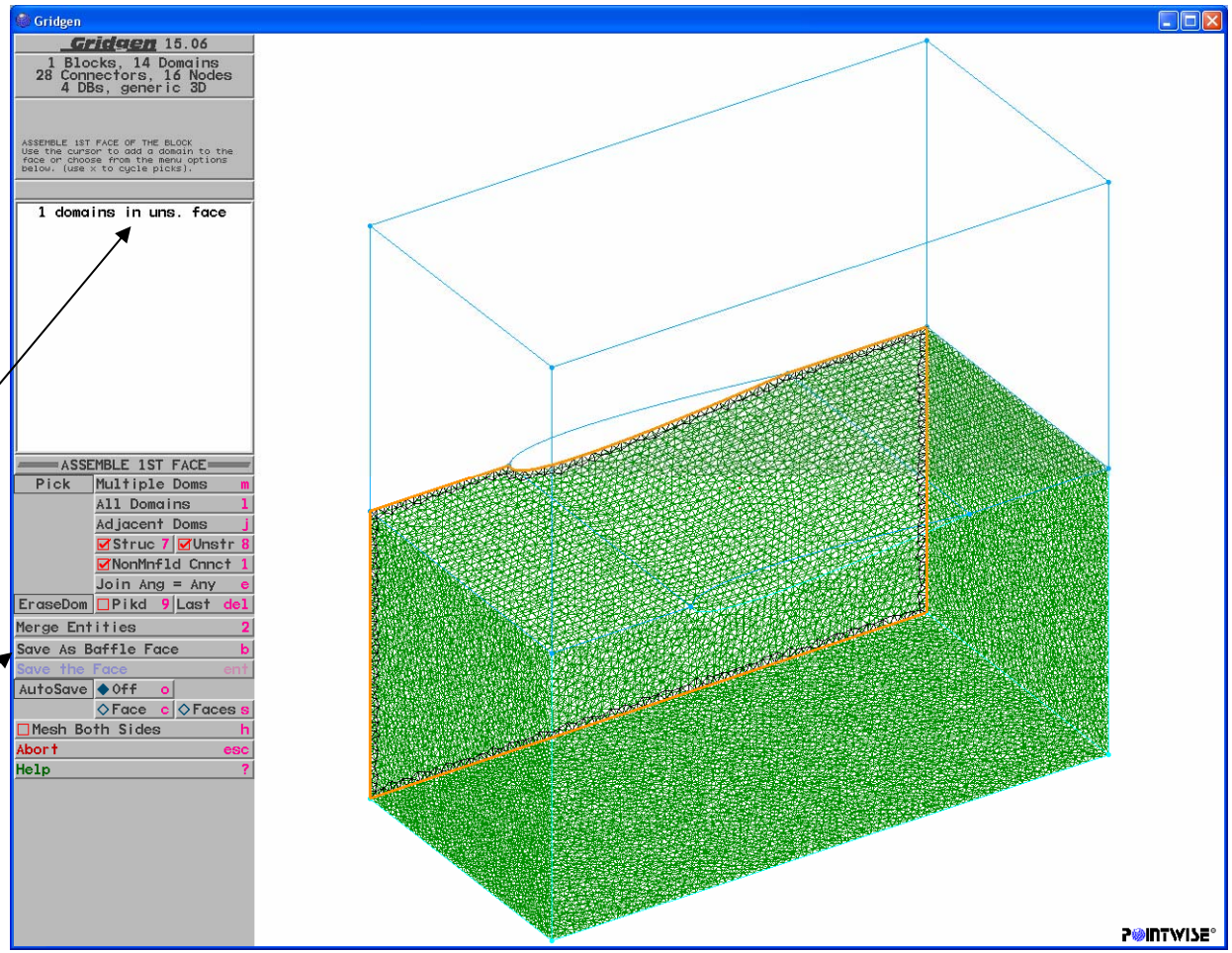
Next, select **Add 1st Face**.

Assemble Blocks (Continued)

The **ASSEMBLE 1st FACE** menu will open and the mouse can be used to pick the first domain to be added. In this case, the domain that makes up the back of the block has been selected.

Note the blackboard window simply states that one domain has been added to the unstructured face. Additional domains will be added until the face completely encloses the block.

The exception would be an unstructured baffle where the face can be open.

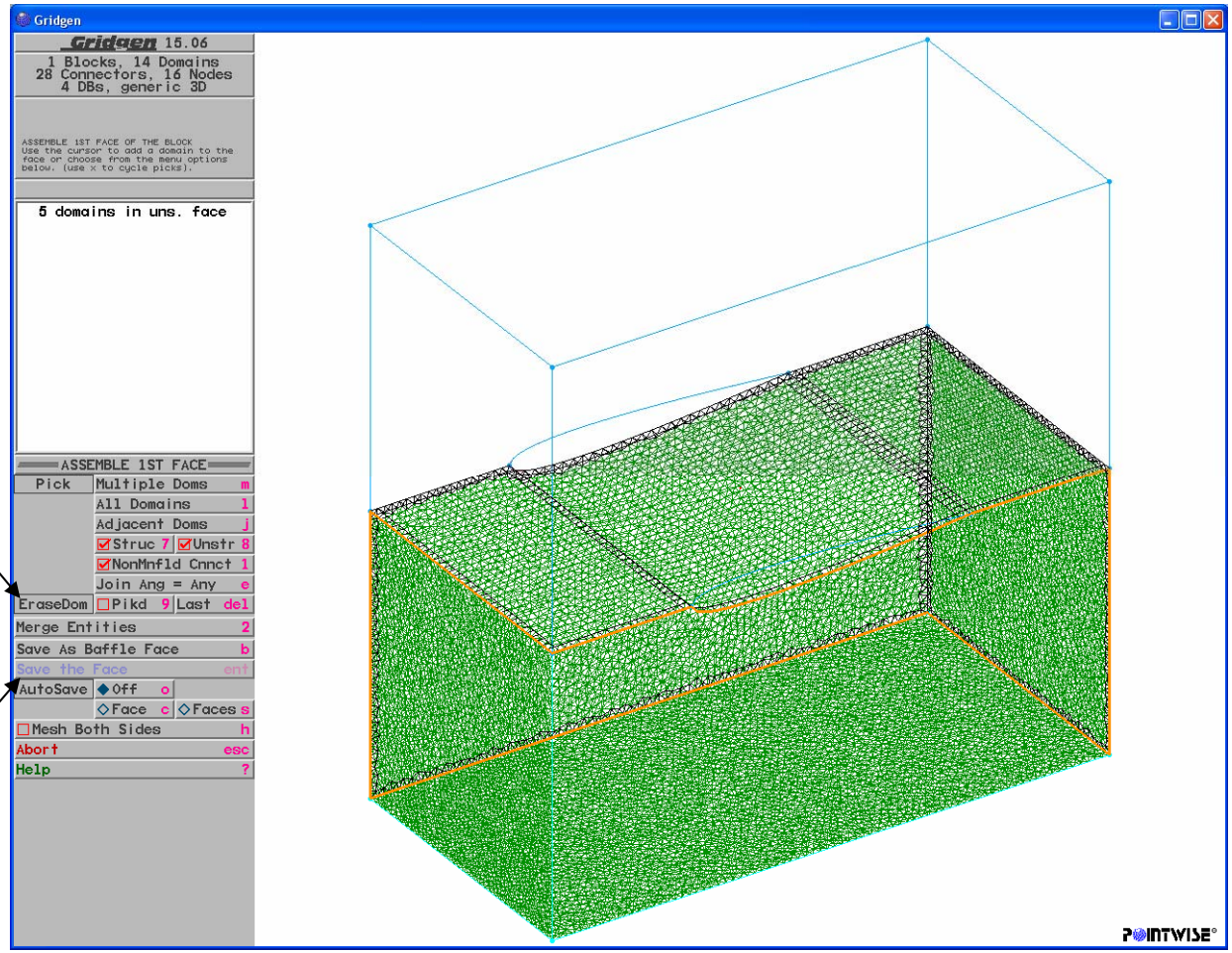


Assemble Blocks (Continued)

The process continues as additional domains are added to the face. Note that a bold yellow outline marks the edge of those domains that are now part of the face – it will disappear when the face is closed.

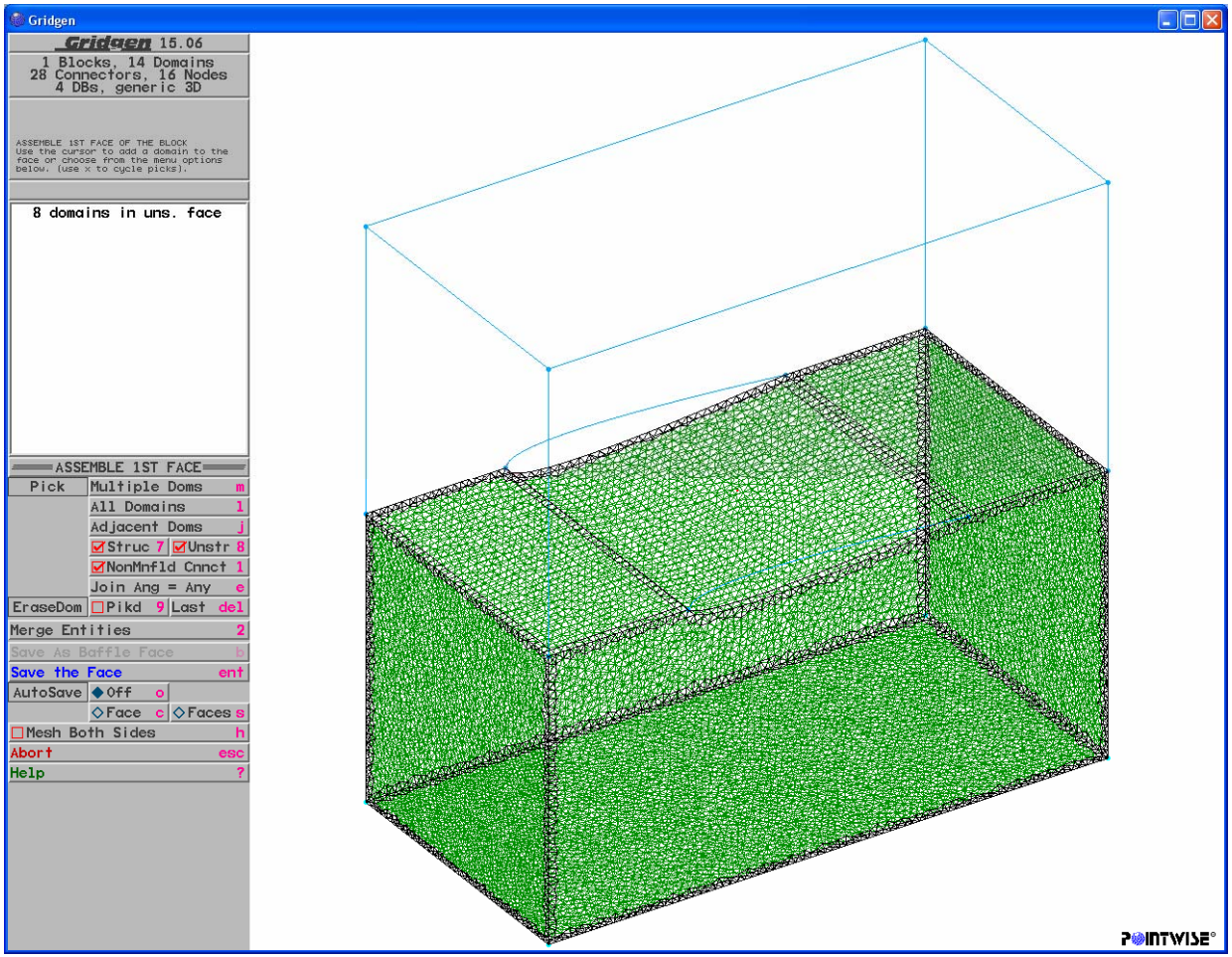
If a mistake is made and an unwanted domain is added, it can be erased with **EraseDom** just as in the case of structured block assembly.

Note that **Save the Face** is not available until the face is closed.



Assemble Blocks (Continued)

Once the face is closed, the bold yellow outline disappears and the **Save the Face** command can be selected.



Assemble Blocks (Continued)

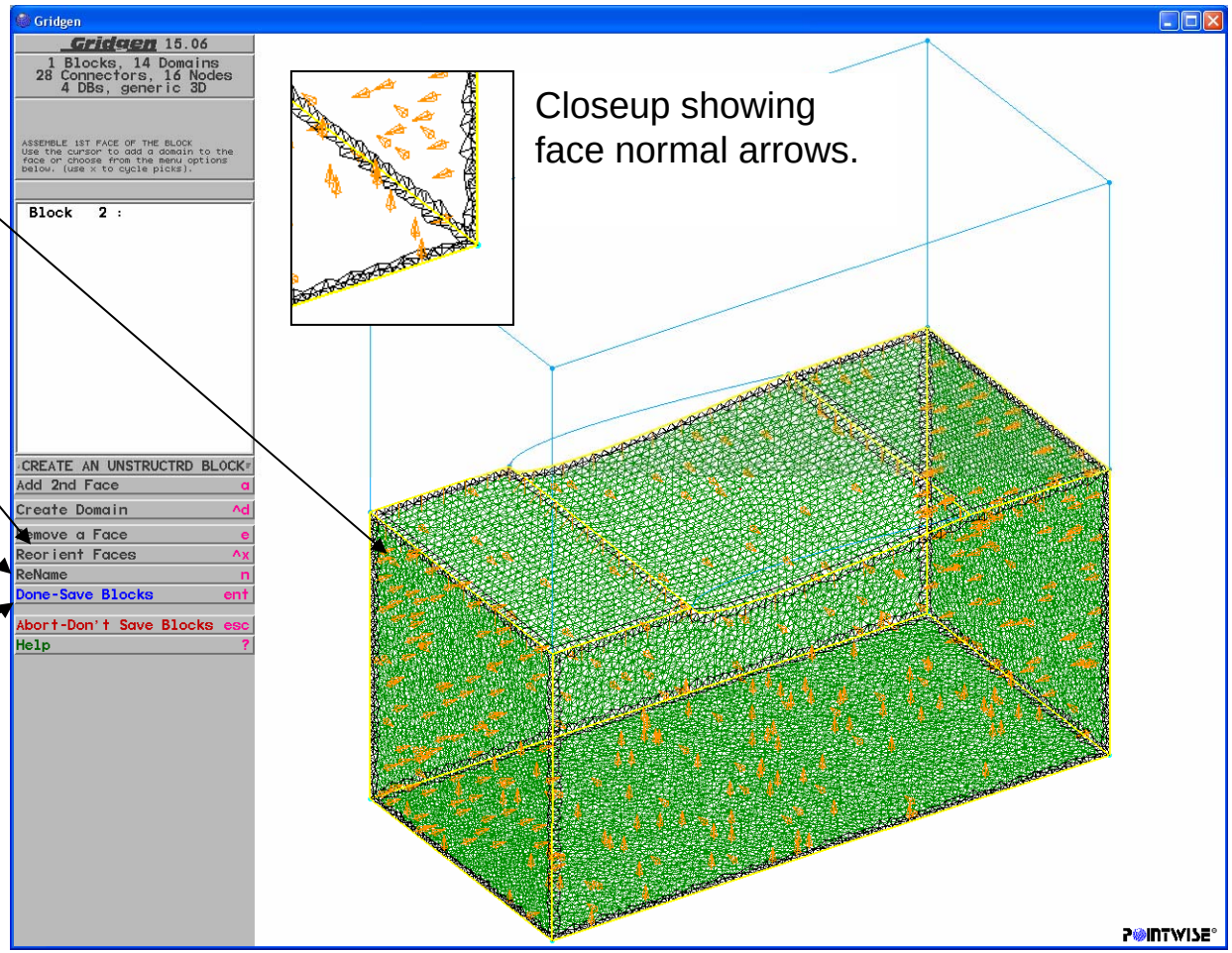
Once the face is saved, the menu changes once more.

Yellow arrows show the face normals.

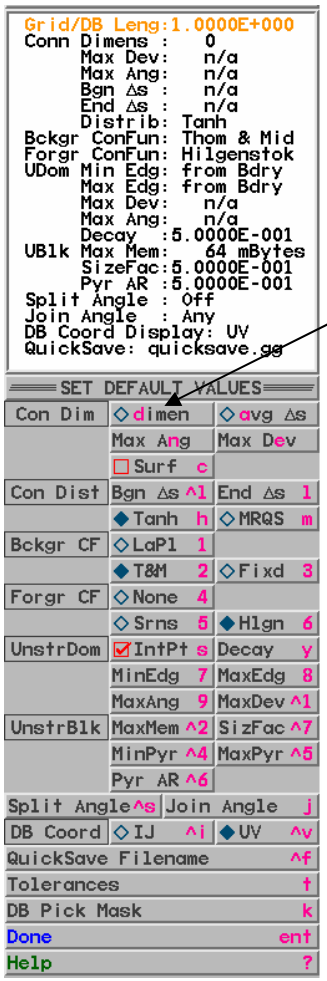
If need be, they can be reversed with the **Reorient Faces** command.

The new block can be given a name with the **ReName** command.

Finally, the block is saved with the **Done – Save Blocks** command. However, the block's interior is not defined until the unstructured solver is run – more on this later.



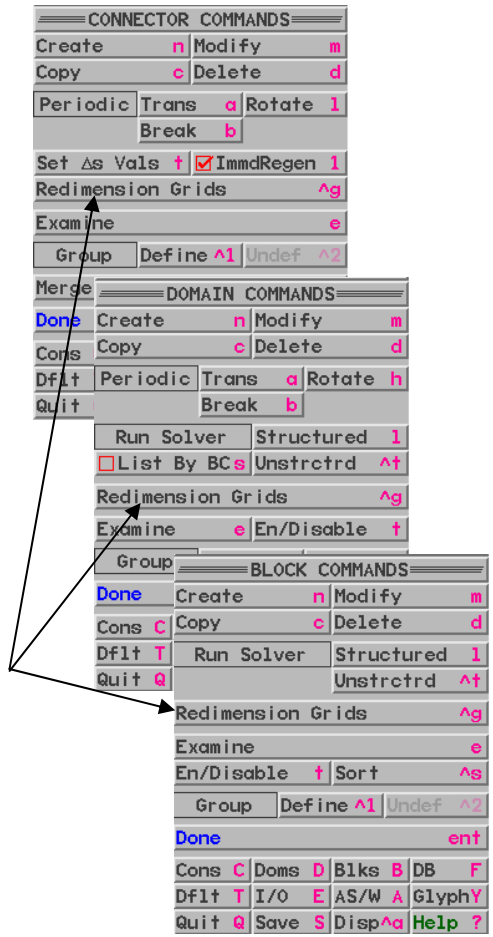
(Re)Dimension Grids



Dimensioning and redimensioning connectors has already been touched upon in discussing (re)dimensioning structured grids.

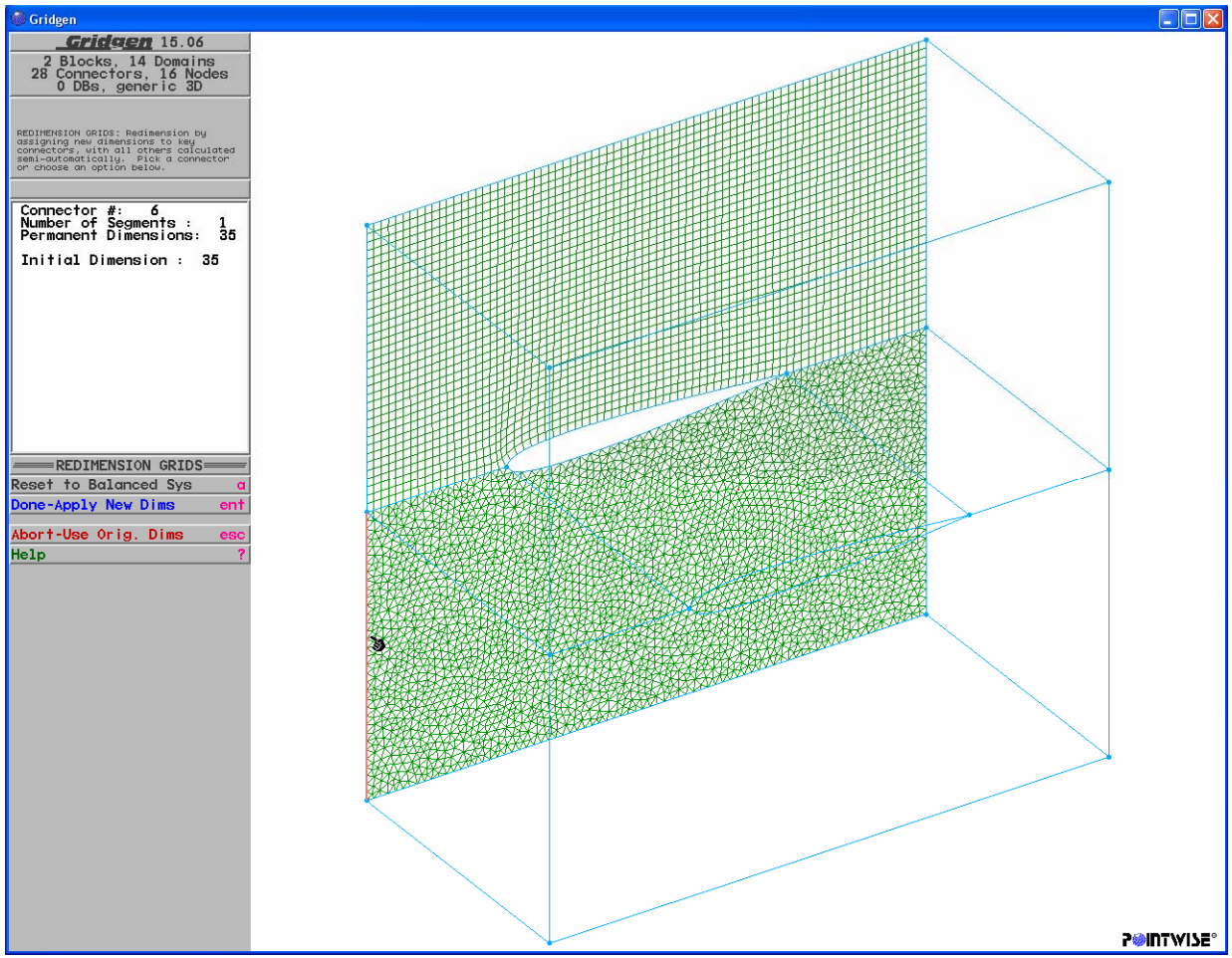
In the **SET DEFAULT VALUES** menu, **Con Dim dimen** can be used to establish a default dimension for all subsequently created connectors.

In addition, a previously defined connector can be redimensioned with the **Redimension Grids** command found in the **CONNECTOR**, **DOMAIN**, and **BLOCK COMMANDS** menus.



(Re)Dimension Grids (Continued)

Since unstructured grids have no topological restrictions, **Redimension Grids** can be used for quick dimensioning of individual connectors whether they are part of a domain or not.



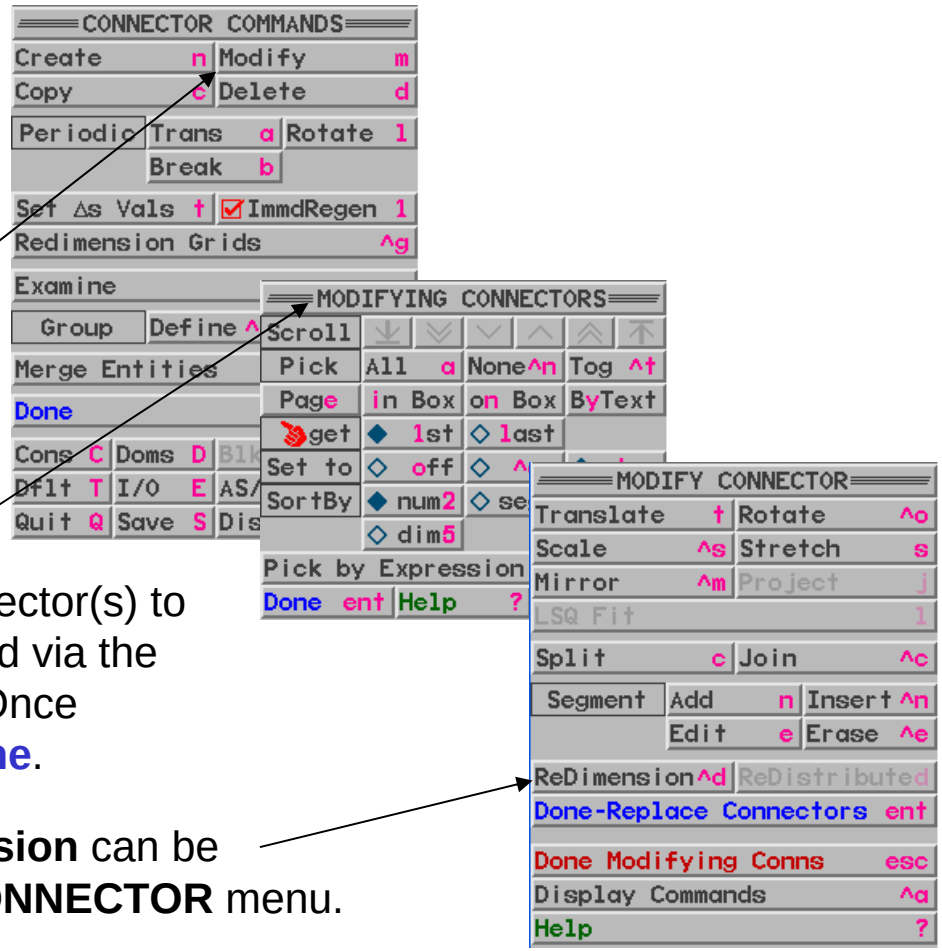
(Re)Dimension Grids (Continued)

As discussed in conjunction with structured meshes, it's also possible to redimension a connector via the **Modify** command in the **CONNECTOR COMMANDS** menu.

Modify is selected in the the **CONNECTOR COMMANDS** menu.

This opens the **MODIFYING CONNECTORS** menu. The connector(s) to be redimensioned can be selected via the menu options or mouse picked. Once selection is completed select **Done**.

Once selected, **ReDimension** can be picked in the **MODIFY CONNECTOR** menu.

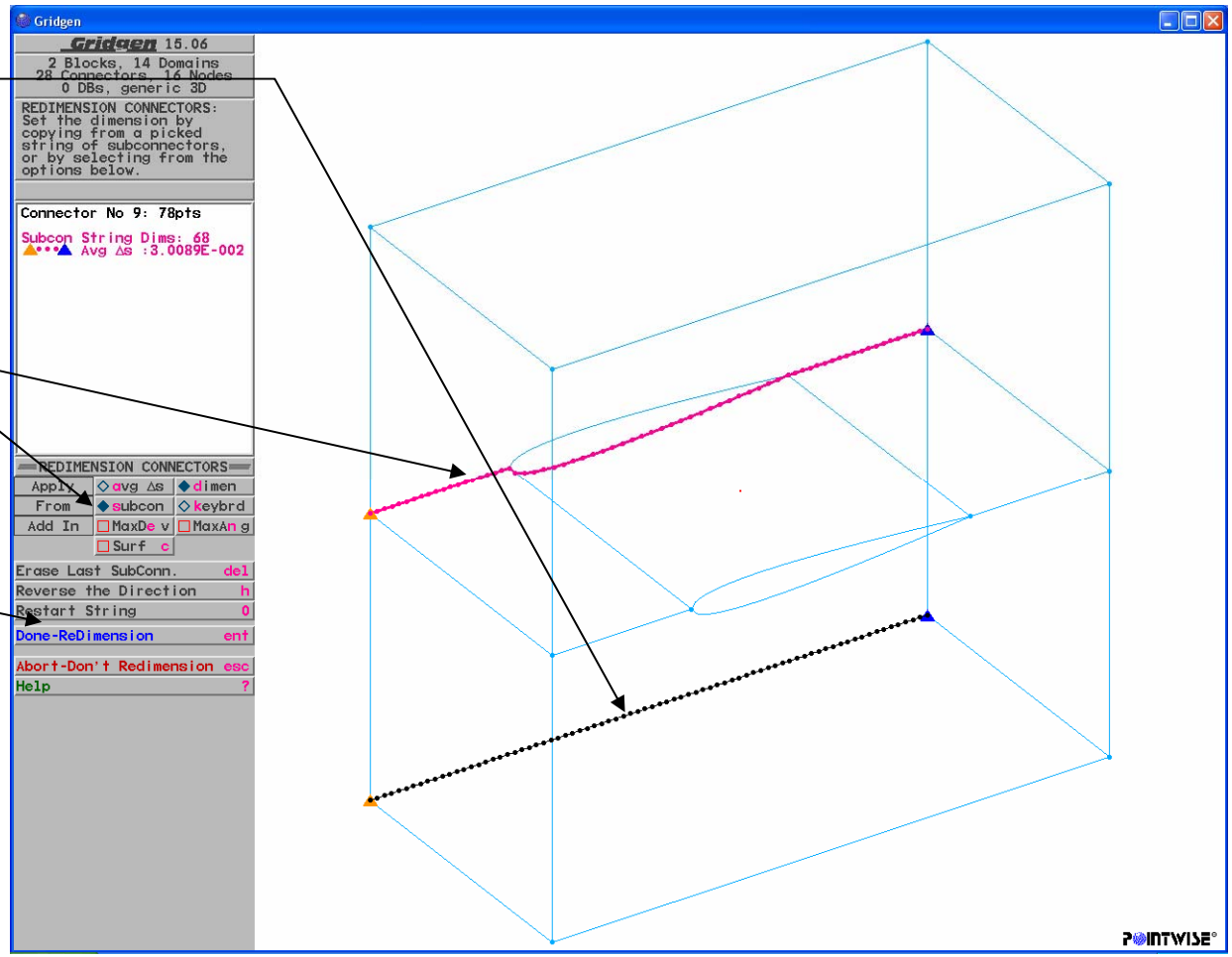


(Re)Dimension Grids (Continued)

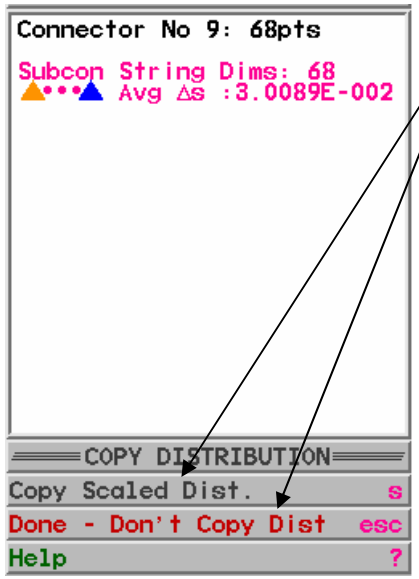
After **ReDimension** is selected, the chosen connector(s) become highlighted in black with orange and blue triangles at the beginning and end.

If **subcon** is selected, the dimension to be copied from other connector(s) chosen via the mouse.

Selecting **Done-ReDimension** will move to the next step of deciding whether to also copy the distribution of grid points.



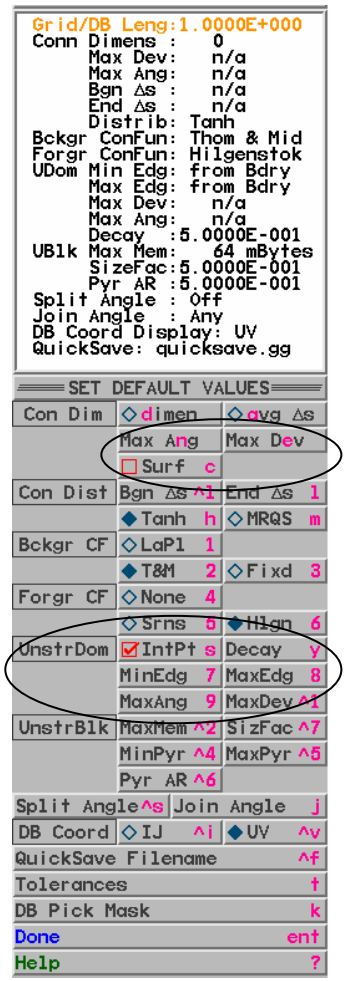
(Re)Dimension Grids (Continued)



Next, the question of whether the distribution of the grid point spacing of the source connector(s) is also to be copied onto the target(s) must be answered. As unstructured meshes are normally built with equal grid point spacing, this is usually a moot point here.

Once that decision is made, the **MODIFY CONNECTOR** menu reappears and the modified connector(s) can be saved by clicking **Done – Replace Connectors**.

(Re)Distribute Grids



Grid points along unstructured connectors are typically equally spaced or have points inserted based on curvature or deviation:

Max Ang adds additional grid points where turning angle for consecutive points is greater than specified.

Max Dev adds additional grid points where discrete grid shape deviates from the true connector or underlying database shape more than the maximum deviation specified.

☐**Surf** considers curvature in all directions of underlying database surface for **Max Ang** and **Max Dev** if toggled. Only applies if either or both are set.

If it is desired, (re)distribution can be handled in the same way as for structured meshes, but more useful in controlling unstructured meshes are:

IntPt (Interior Points) controls whether grid points are inserted into unstructured domains when they are created.

Decay Factor controls how far boundary cell size affects interior cell size, ranges from 0 (no influence) to 1 (0.5 default).

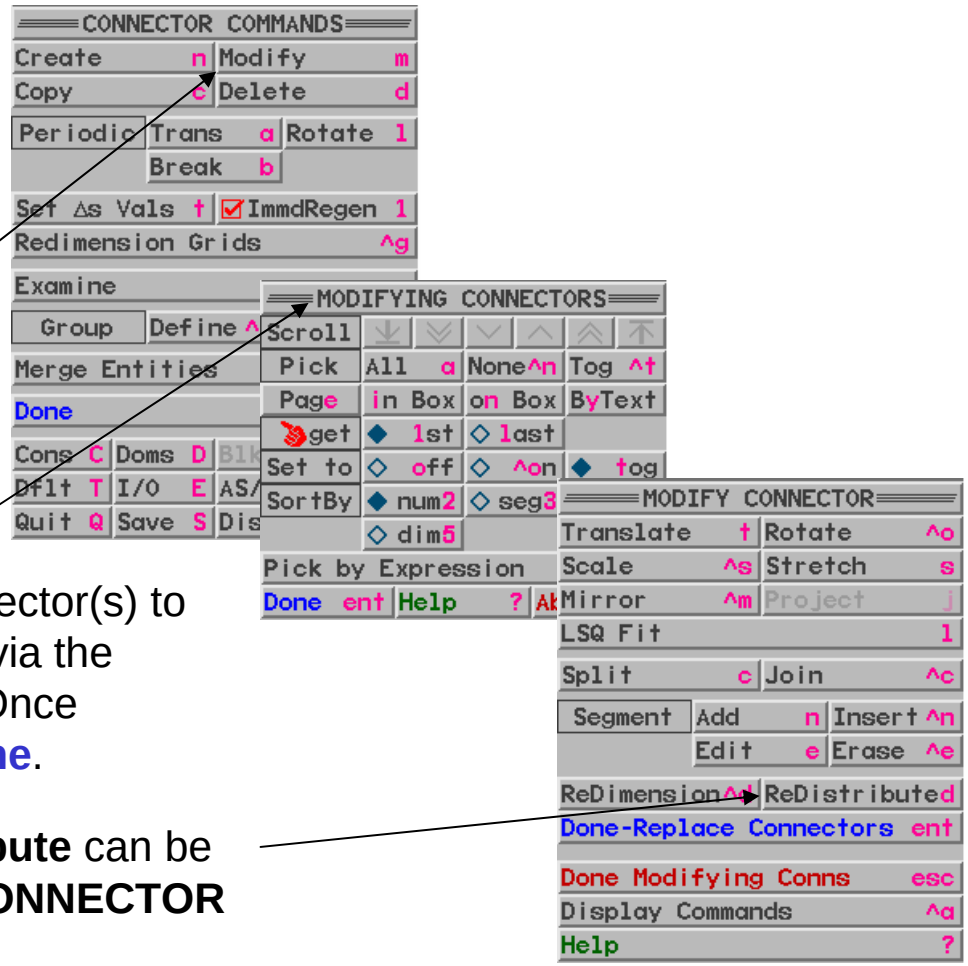
MinEdg and **MaxEdg** specify the minimum and maximum edge lengths for cells, respectively.

MaxAng specifies the maximum dihedral angle allowed between two triangles sharing an edge.

MaxDev specifies the maximum distance allowed between a domain's analytical shape (based on its database) and its discrete shape (the triangular facets).

(Re)Distribute Grids (Continued)

As with structured grids, it's also possible to redistribute grid points via the **Modify** command in the **CONNECTOR COMMANDS** menu.



Modify is selected in the the **CONNECTOR COMMANDS** menu.

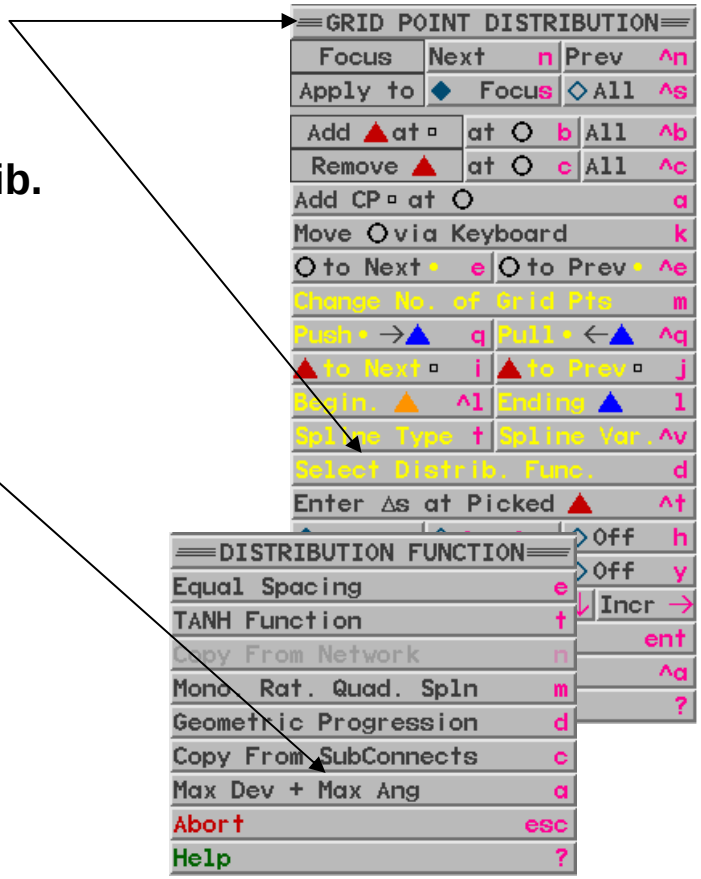
This opens the **MODIFYING CONNECTORS** menu. The connector(s) to be redistributed can be selected via the menu options or mouse picked. Once selection is completed select **Done**.

Once selected, **ReDistribute** can be picked in the **MODIFY CONNECTOR** menu.

(Re)Distribute Grids (Continued)

The connectors to be modified can now be selected by mouse clicking in the display window and various operations can be selected from the **GRID POINT DISTRIBUTION** menu. Amongst these is **Select Distrib. Func.** which opens the **DISTRIBUTION FUNCTION MENU**.

Amongst the options here is **Max Dev + Max Ang**, which can move grid points about to satisfy the input deviation values without changing the overall connector dimension.



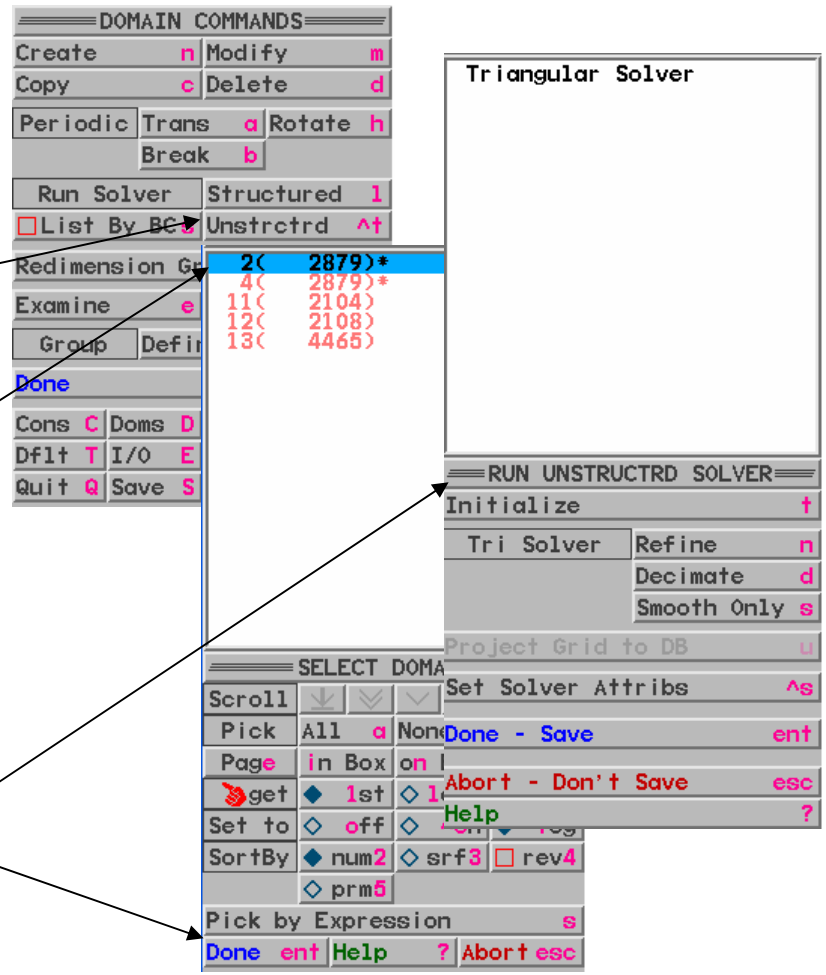
The Unstructured Solver

For domains, the unstructured solver is primarily used to change the interior distribution of points, typically for smoothing or more accurately resolving geometric shape.

To run the solver, first choose **Run Solver Unstrctrd** from the **DOMAIN COMMANDS** menu.

The domain is next picked – in this case by highlighting the entry in the browser window. Alternatively the mouse can be used for selecting the domain.

Once **Done** is selected, the **RUN UNSTRUCTRD SOLVER** menu appears.



The Unstructured Solver (Continued)

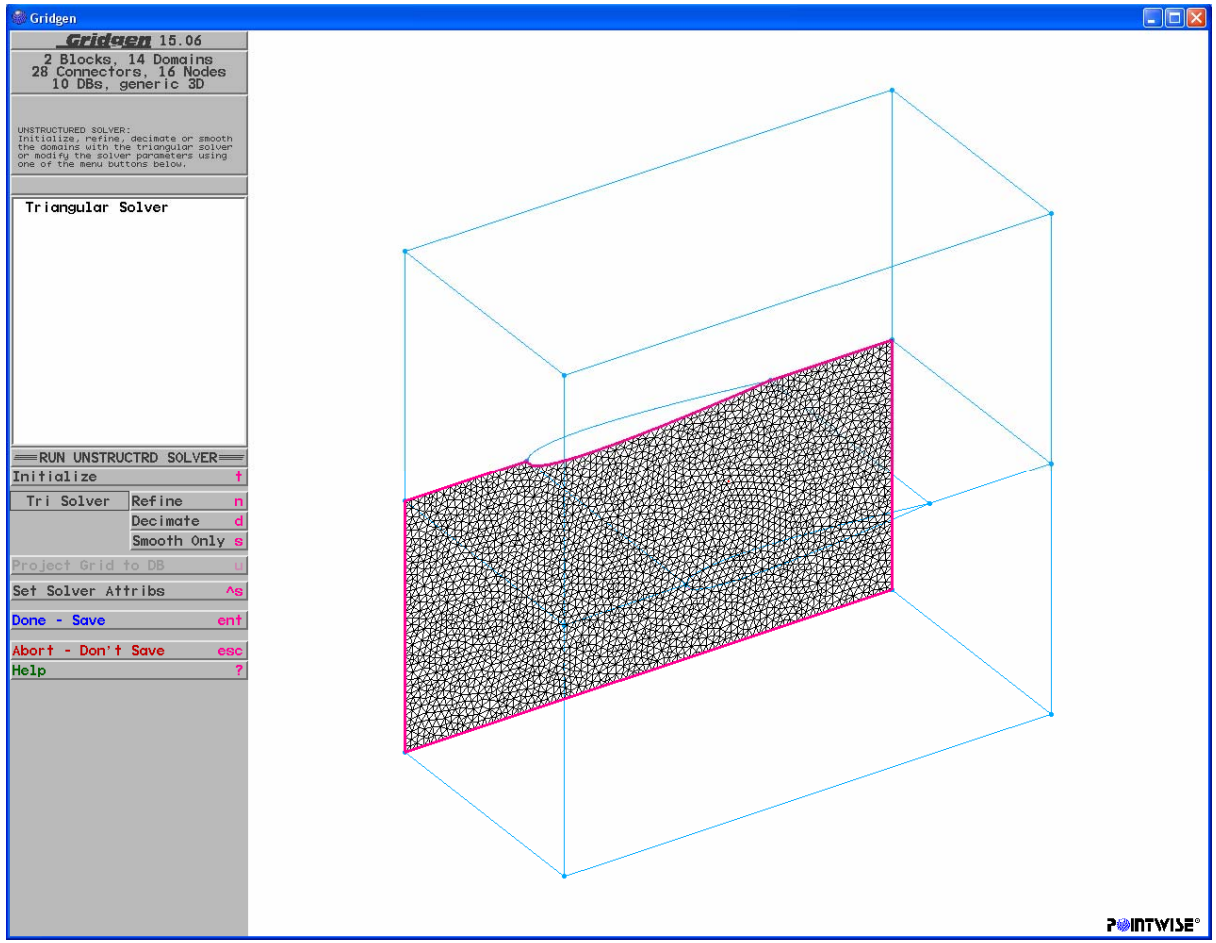
RUN UNSTRUCTRD SOLVER menu options include:

Initialize deletes existing interior cells and repopulates based on current defaults and attributes.

Tri Solver inserts (**Refine**), removes (**Decimate**), or smoothes (**Smooth Only**) points based on attributes. The latter involves LaPlacian smoothing and diagonal swapping.

Project Grid to DB constrains grid to underlining database (grayed out when not present).

Set Solver Attribs opens menu for setting attributes.



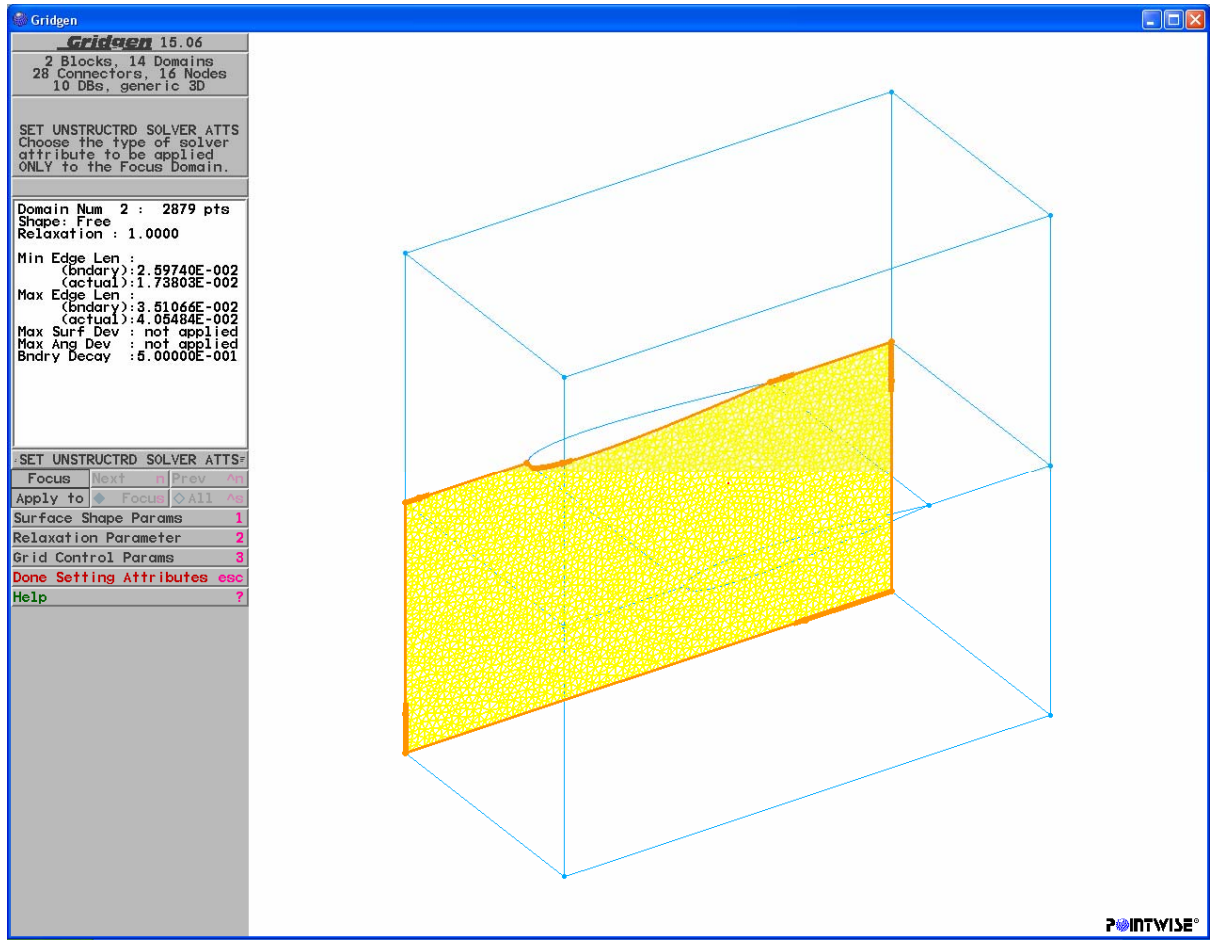
The Unstructured Solver (Continued)

Having selected **Set Solver Attribs**, the **SET UNSTRUCTRD SOLVER ATTS** menus opens and the image of the selected grid changes. The options here are:

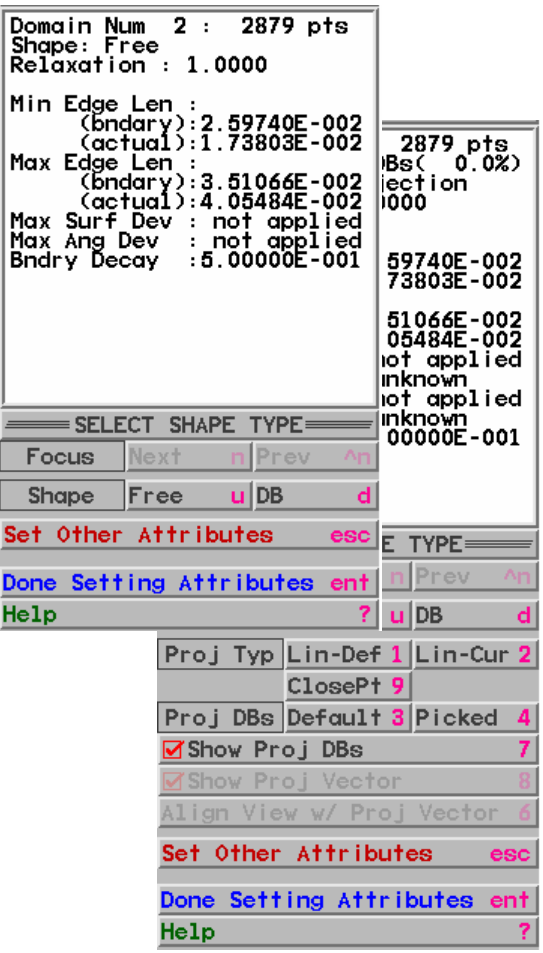
Surface Shape Params opens the **SELECT SHAPE TYPE** menu to choose how the domain surface shape will be determined.

Relaxation Parameter opens the **SELECT RELAXATION PARAM** menu to change the unstructured solver smoothing.

Grid Control Params opens the **SELECT GRID CONTROL PARAM** menu to select and change various grid attributes.



The Unstructured Solver (Continued)



The **SELECT SHAPE TYPE** menu provides these options:

Free – solver runs with no constraint on domain shape.

DB – shape is constrained to underlying database surface(s).

Selecting the latter opens the following options:

Proj Typ

Lin-Def – an average of the domain normals at the four corners is used for the projection vector for the linear projection method.

Lin-Cur – the current view orientation is set and saved as the projection vector for the linear projection method.

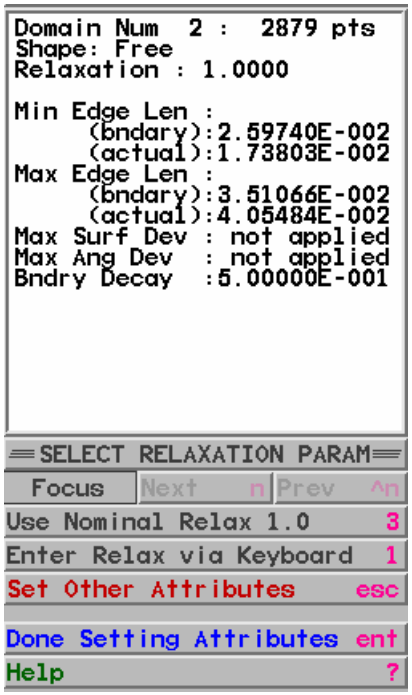
ClosePt – the closest point projection method is used.

Proj DBs

Default – a default set of database surfaces is chosen based on domain/connector associativities.

Picked – the database surfaces are picked via mouse or the browser window.

The Unstructured Solver (Continued)



The **SELECT RELAXATION PARAM** menu has these options:

Use Nominal Relax 1.0 – (default) uses the mean value for determining the number of smoothing iterations.

Enter Relax via Keyboard – replaces default with a new value (between 0 and 2) entered via the keyboard.

The **SELECT GRID CONTROL PARAM** menu offers these options for controlling **Triangle** cell geometry:

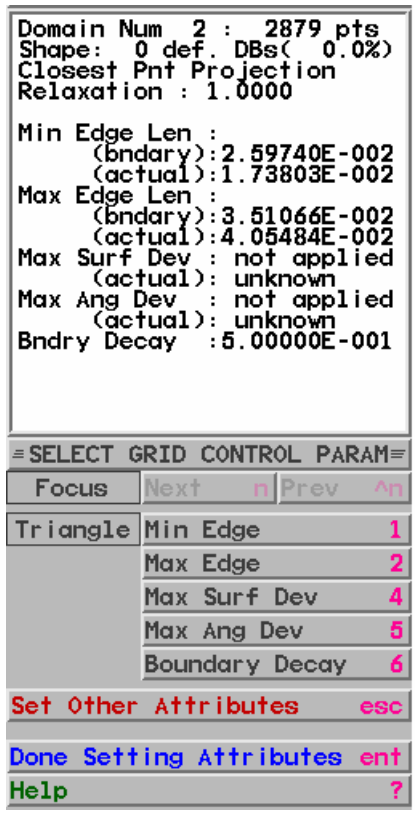
Min Edge – minimum triangle edge length.

Max Edge – maximum triangle edge length.

Max Surf Dev – maximum allowed distance of triangle centroid from underlying database surface, inserting grid points as needed to remain below maximum.

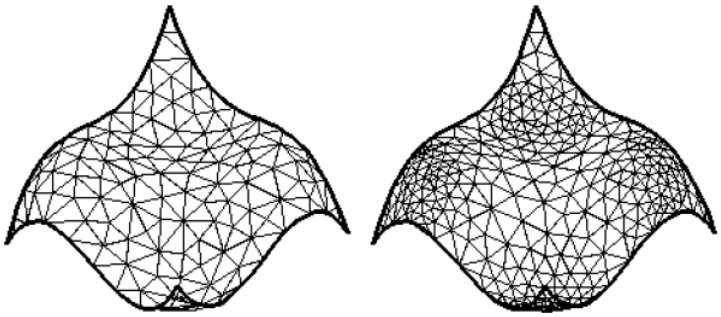
Max Ang Dev – maximum allowed turning angle between adjacent triangles, inserting grid points as needed.

Boundary Decay – distance into domain interior that boundary grid point spacing affects interior triangle edge length.



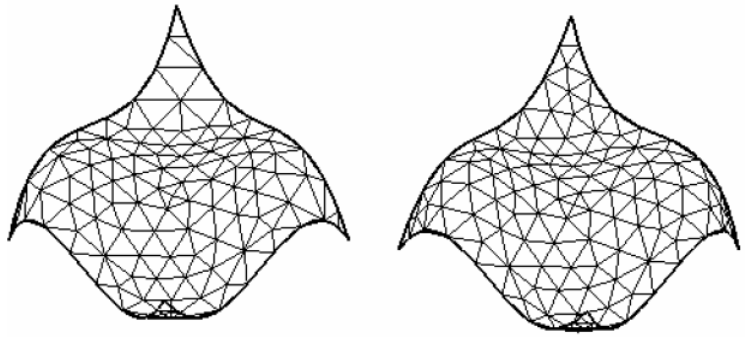
The Unstructured Solver (Continued)

Max Surf Dev



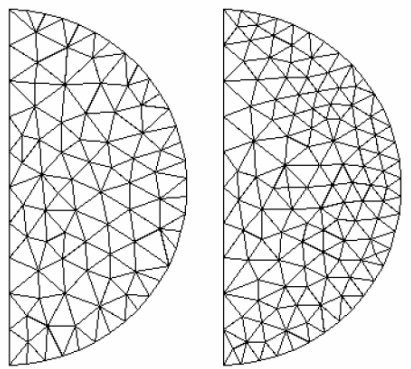
WITHOUT WITH

Max Ang Dev



WITHOUT WITH

Boundary Decay



0 (OFF) 1 (MAX)

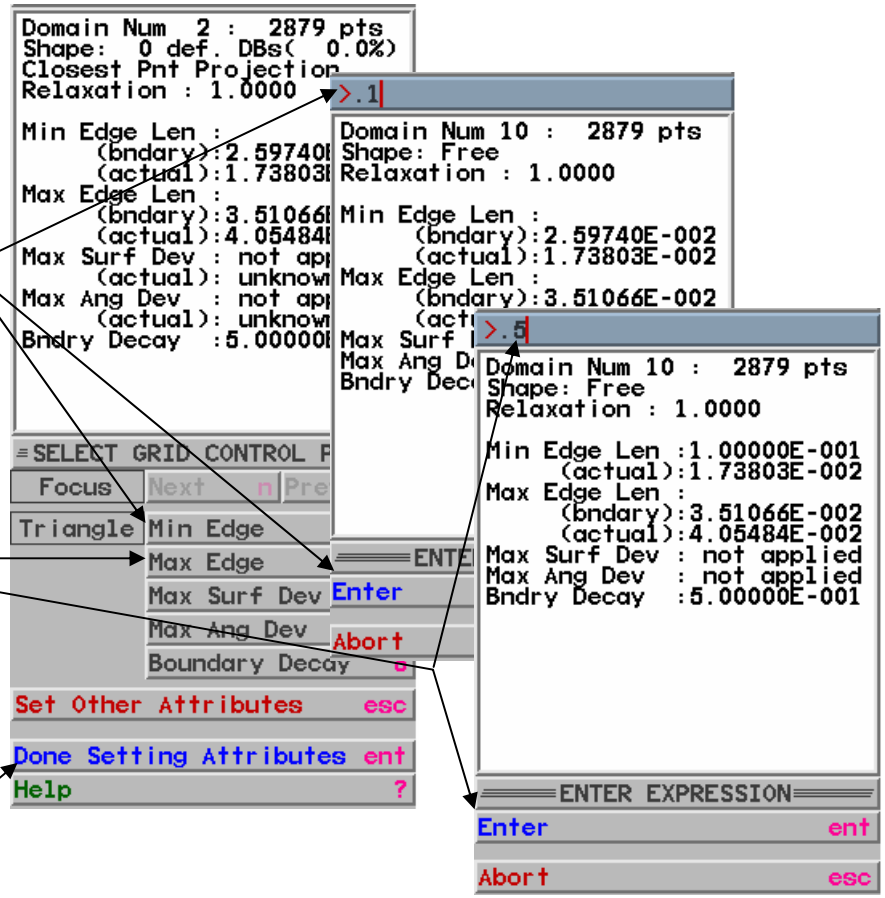
The Unstructured Solver (Continued)

Returning to the airfoil mesh, we'll increase the triangle min and max edge control parameters so we can coarsen the grid with the **Decimate** command.

Selecting **Min Edge** in the **SELECT GRID CONTROL PARAM** menu prompts for a new value. Type 0.1 via the keyboard and then click **Enter**.

Selecting **Max Edge** in the **SELECT GRID CONTROL PARAM** menu prompts for a new value. Type 0.5 via the keyboard and then click **Enter**.

Pick **Done Setting Attributes** to return to the **RUN UNSTRUCTRD SOLVER** screen.

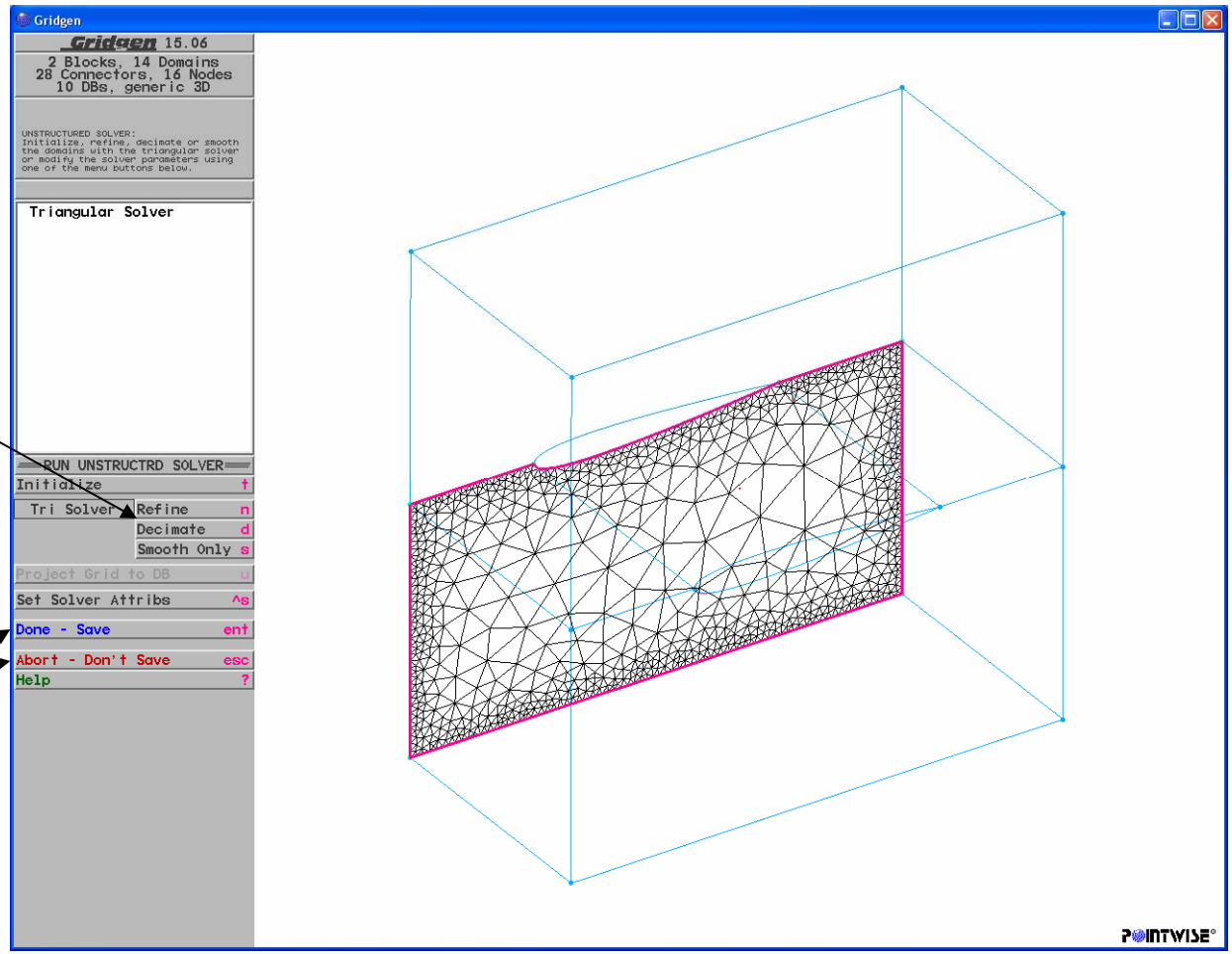


The Unstructured Solver (Continued)

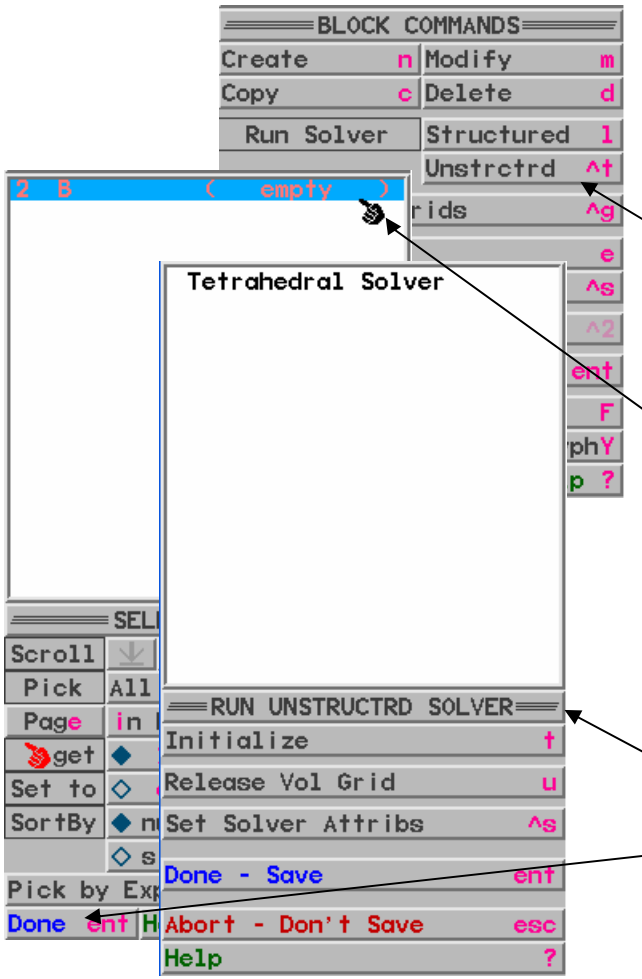
With these new parameters defined, the grid density on the back plane of the unstructured mesh can now be reduced.

Selecting **Decimate** (or typing the d hot key) will run the unstructured solver with these new parameters.

Selecting **Done – Save** will save the modified mesh, while **Abort – Don't Save** will return the mesh to its previous state.



The Unstructured Solver (Continued)



For blocks, the unstructured solver is primarily used to initialize their interiors, i.e., fill them with tetrahedral cells, after their exterior shells have been defined.

To run the solver, first choose **Run Solver Unstrctrd** from the **BLOCK COMMANDS** menu.

The block is next picked – in this case by highlighting the entry in the browser window. Alternatively the mouse can be used for selecting it. As there are no cells defined yet, the block is described as **empty**.

Once **Done** is selected, the **RUN UNSTRUCTRD SOLVER** menu appears.

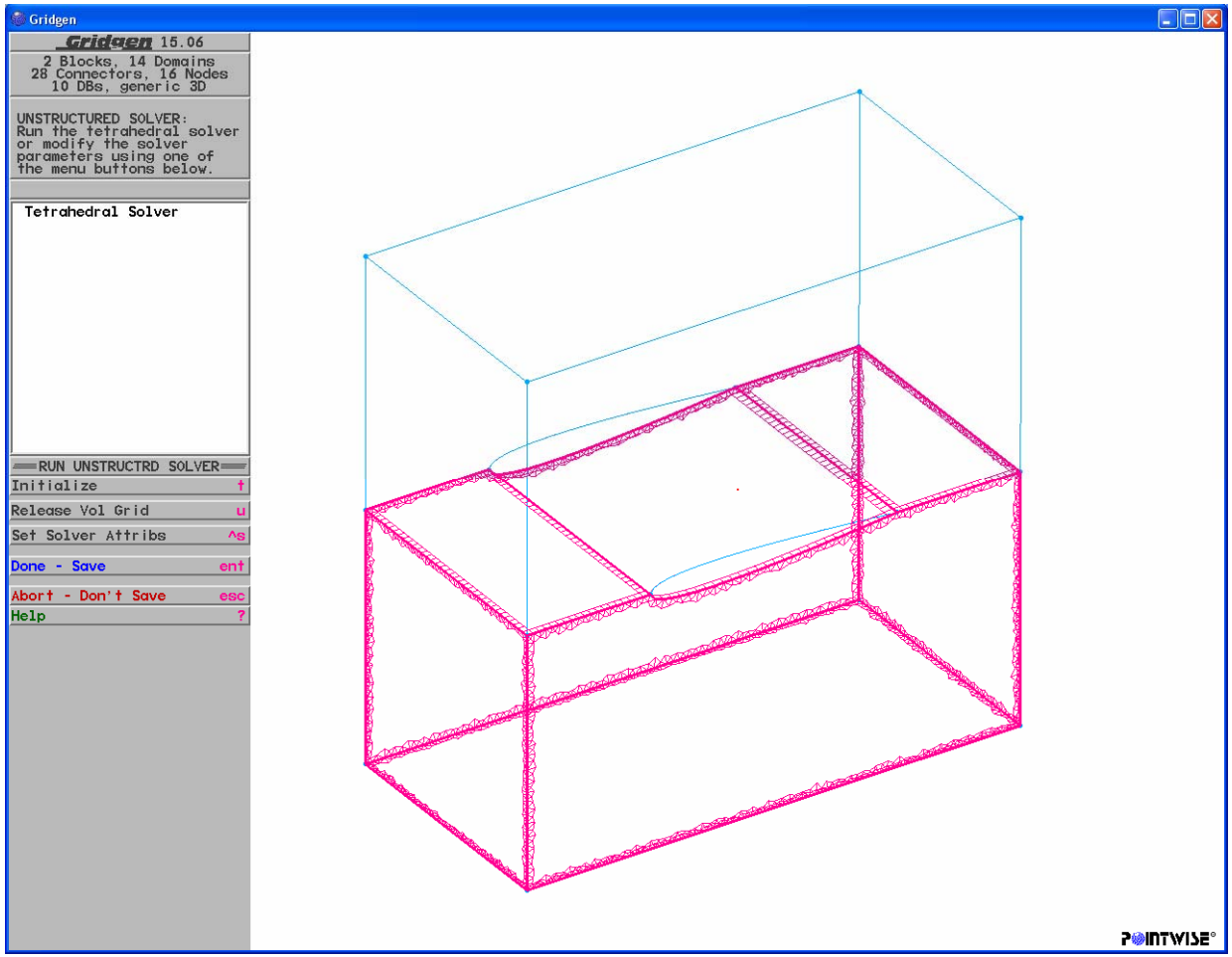
The Unstructured Solver (Continued)

RUN UNSTRUCTRD SOLVER menu options include:

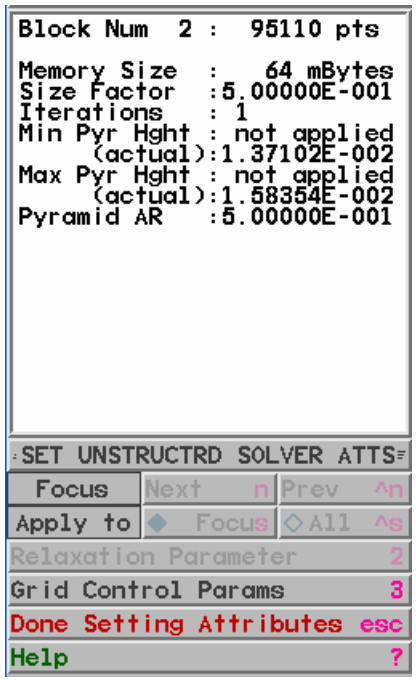
Initialize passes the block information to the tetrahedral solver to fill the block with tet cells. Once selected, the message window will report **UNSTRUCTURED SOLVER: Working...** until the process is completed. During the calculations, the cursor is replaced with an hourglass

Release Vol Grid deletes any tet cells previously created for the block from memory.

Set Solver Attribs opens another menu to set various attributes for the solver.



The Unstructured Solver (Continued)



Having selected **Set Solver Attribs**, the **SET UNSTRUCTRD SOLVER ATTS** menu opens. **Grid Control Params** can be selected to reveal the **SELECT GRID CONTROL PARAM** submenu, with these options for the **Tetra**(hedral) cells:

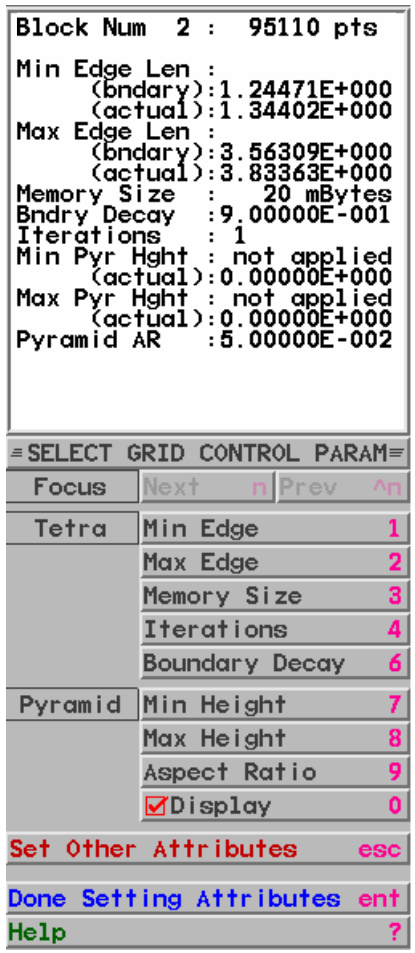
Min Edge – minimum triangle edge length.

Max Edge – maximum triangle edge length.

Memory Size – maximum amount of memory to be allocated to the tetrahedral solver. Default is 64 MB. Rough estimate is 1 MB per 15,000 tetrahedra.

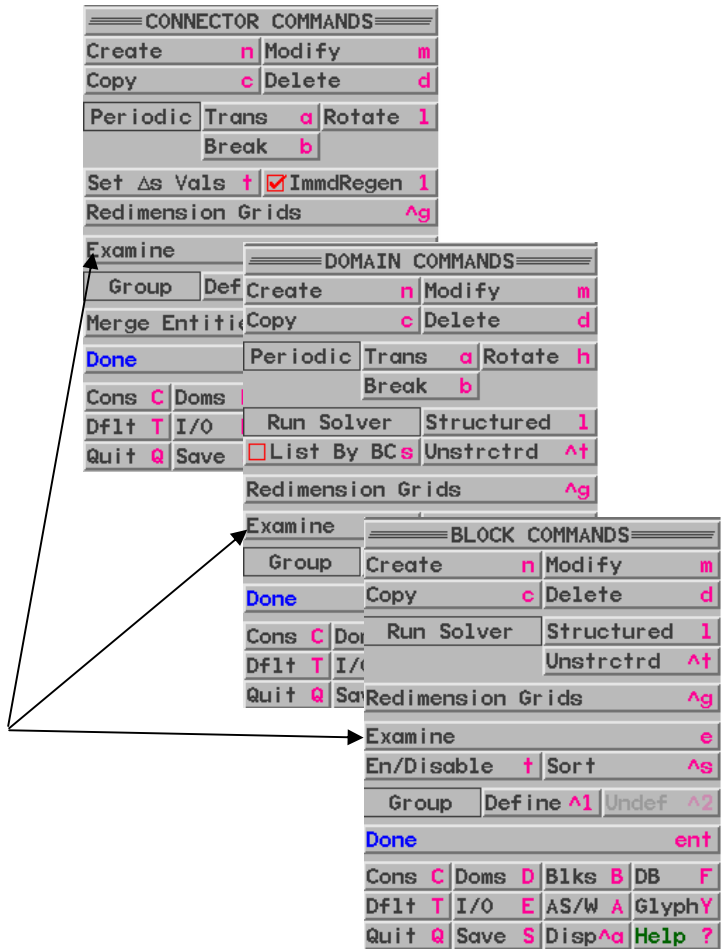
Iterations – controls number of runs through tetrahedral solver. Additional iterations might improve grid quality in inadequate regions of a bad mesh. Default is 1, and max is 10.

Boundary Decay – distance into interior that boundary grid point spacing affects interior triangle edge length.



Examining and Correcting Grids

Gridgen includes extensive tools for examining various mesh characteristics and isolating resulting mesh problems. Specific tools available for studying connectors, domains, and blocks can be accessed via the **Examine** command in the corresponding **COMMANDS** menus.



Clicking on **Examine** (or typing **e**) opens the **Examine** menu.

Examine Connectors

Once **Examine** has been picked in the **CONNECTOR COMMANDS** menu, the individual connectors to be considered can be selected from the browser window or via the mouse in the display window. When the selection is complete, **Done** is picked, opening the **Examine** window.

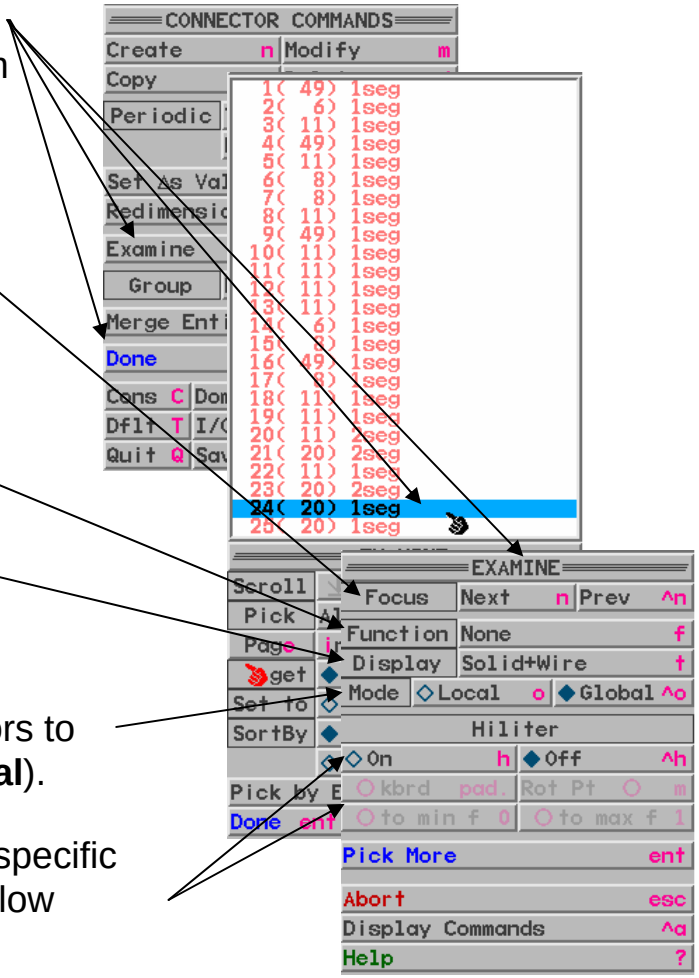
Focus allows the diagnostic function to be reported for specific connectors or grid points, depending on the **Mode** and **Hiliter** status, by toggling **Next** or **Prev**.

Function selects the specific diagnostic function to be applied.

Display opens another menu for selecting display options for diagnostics information.

Mode allows one of the chosen connectors to be examined (**Local**) or all of them(**Global**).

Hiliter On will show diagnostic information about a specific grid point in the blackboard window. The buttons below provide options for picking this grid point.



Examine Connectors (Continued)

Clicking on **None** in the **EXAMINE** menu opens the **DIAGNOSTIC** submenu where the diagnostic function to be applied can be chosen.

Usage will show what entities (domains, etc.) utilize the selected connector(s).

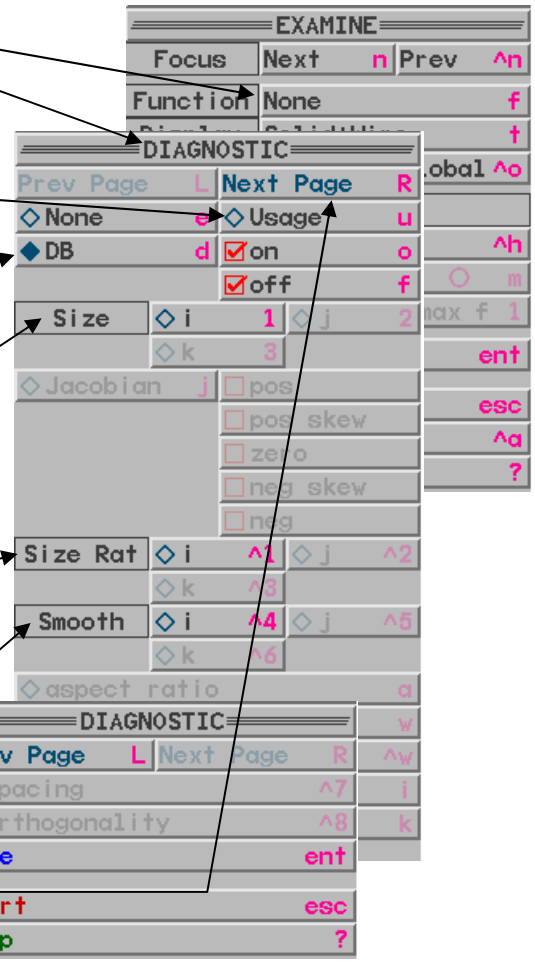
DB will show whether the connectors' grid points are **on** and/or **off** (depending on the toggle settings) the database.

Size will show the lengths between grid points along the connector(s).

Size Rat (Size Ratio) will show the ratio of these lengths amongst adjacent points.

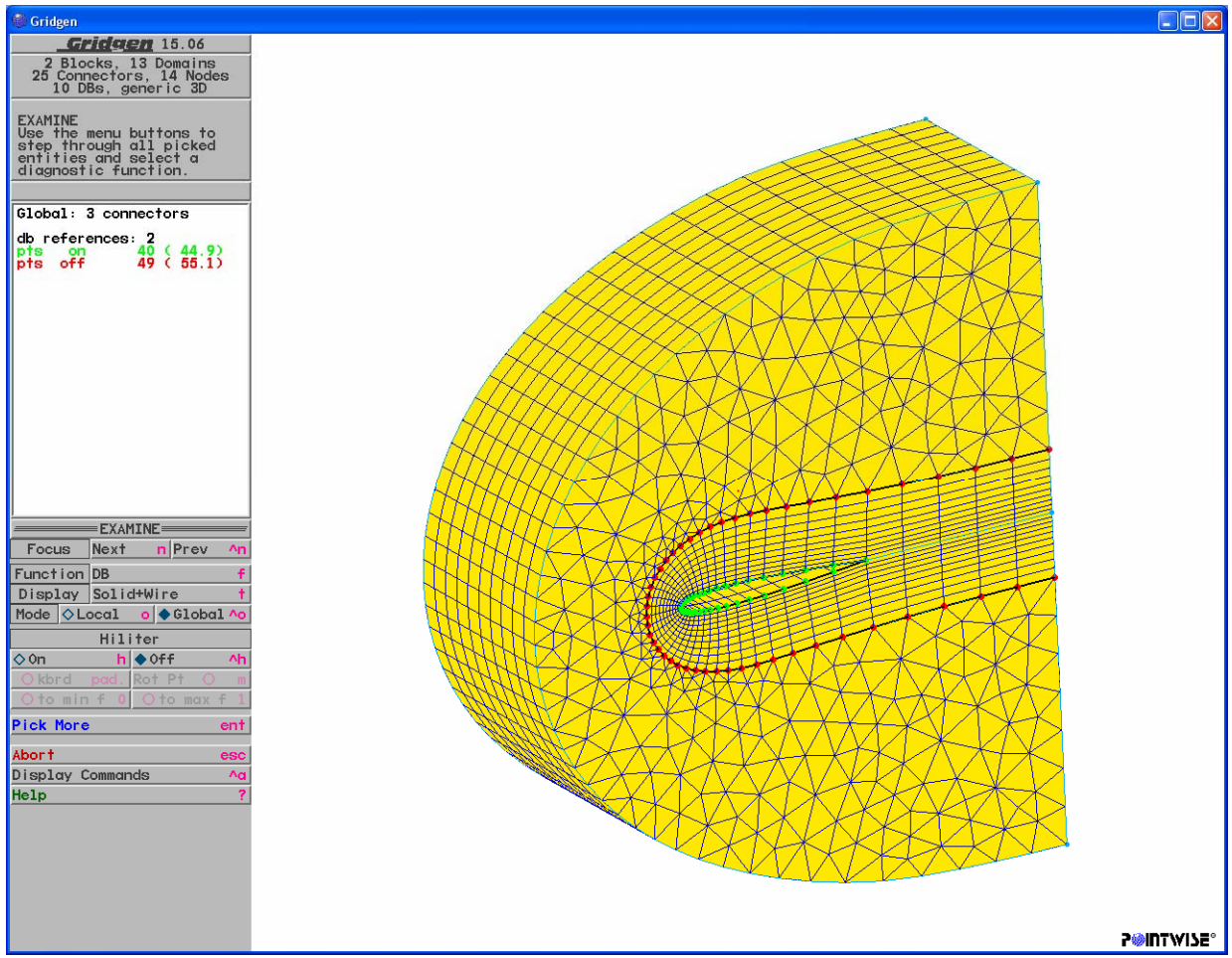
Smooth (Smoothness) will show the normalized turning angle of the connector at each grid point.

Once the function has been selected, **Next Page** must be clicked to reveal the **Done** button to return.



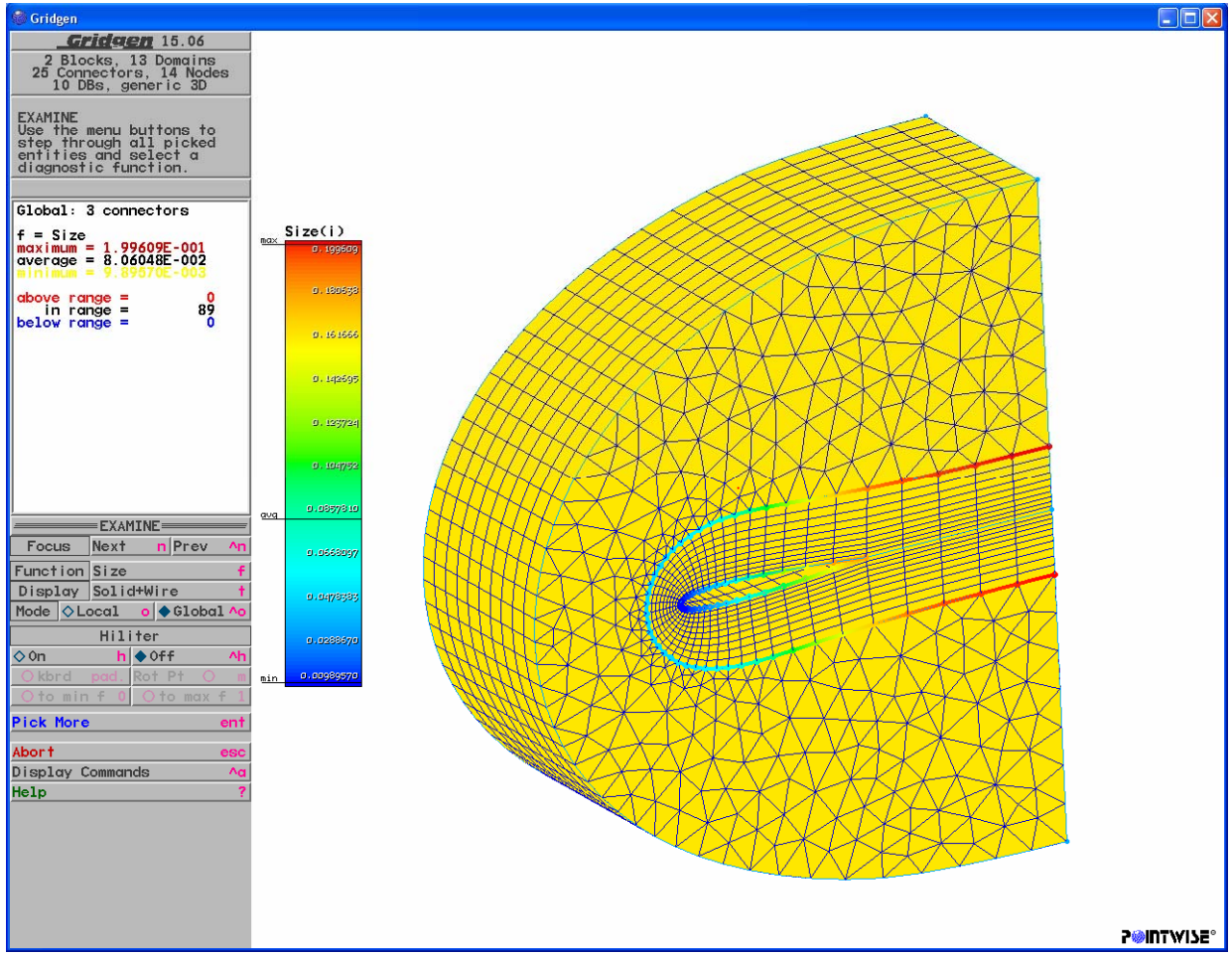
Examine Connectors (Continued)

Here, the **DB Function** has been chosen (note the **DB** identifier next to **Function**), with both **on** and **off** toggled. Three connectors were chosen, the top and bottom of the airfoil and the edge between the structured and unstructured blocks. As the airfoil is referenced to the wing's database, the grid points appear as green, i.e., they're on the database. As the other connector is not on the database, it's grid points are red, i.e., they're off the database. Note the summary of this information in the blackboard window.



Examine Connectors (Continued)

Here, the **Size Function** has been chosen (note the **Size** identifier next to **Function** in the menu). The three connectors are now color coded by the lengths between grid points. The color scale on the left of the display window shows the corresponding lengths in model units. Again, note the summary information in the blackboard window.



Examine Domains

As before, clicking on **None** in the **EXAMINE** menu opens the **DIAGNOSTIC** submenu where the diagnostic function to be applied can be chosen.

Usage will show what entities utilize or are utilized by the selected domain(s).

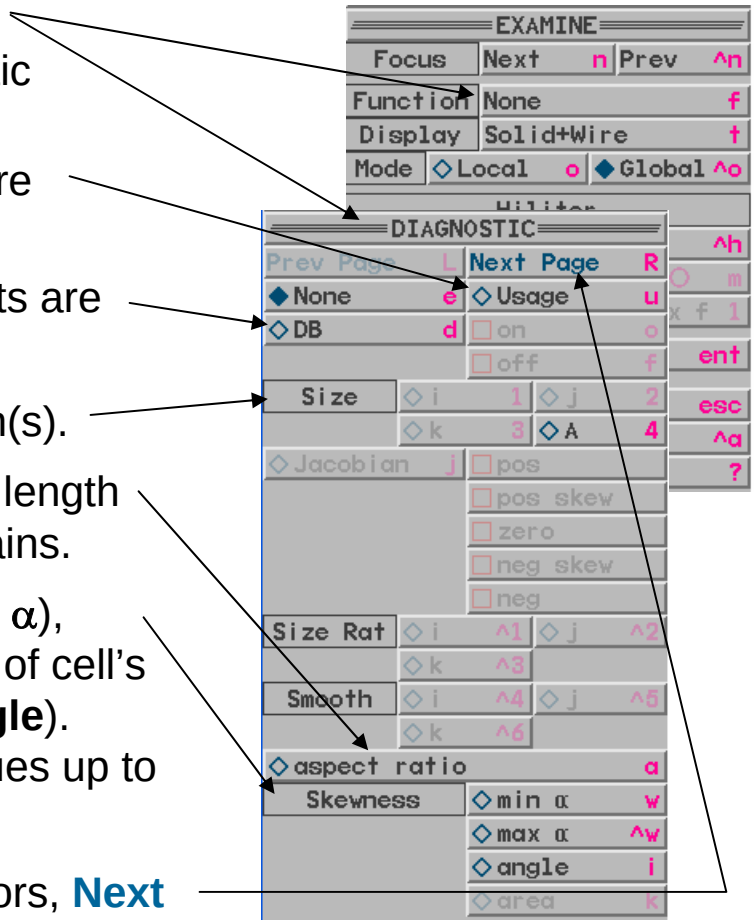
DB will show whether the grid points are **on** and/or **off** the database.

Size will show the cell areas in the selected domain(s).

Aspect Ratio will show ratio of average length to average width of the cells in the domains.

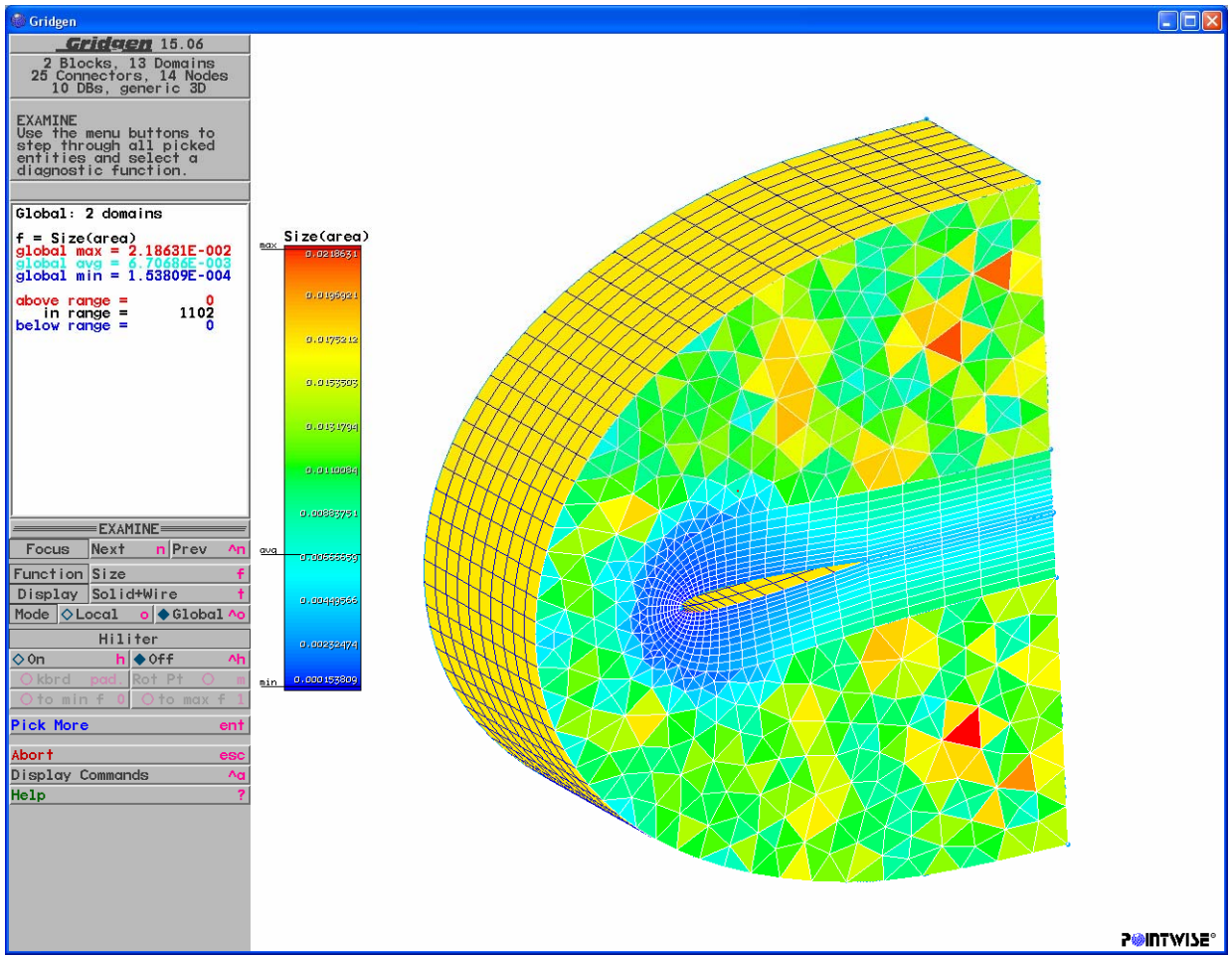
Skewness will show cell's minimum included angle (**min** α), maximum included angle (**max** α), or the maximum ratio of cell's included angle to the angle of an equilateral triangle (**angle**). **angle** values less than 0.8 are considered good, but values up to 0.9 may be acceptable for some solvers.

As in the case of the **DIAGNOSTIC** menu for connectors, **Next Page** must be clicked to reveal the **Done** button to return.



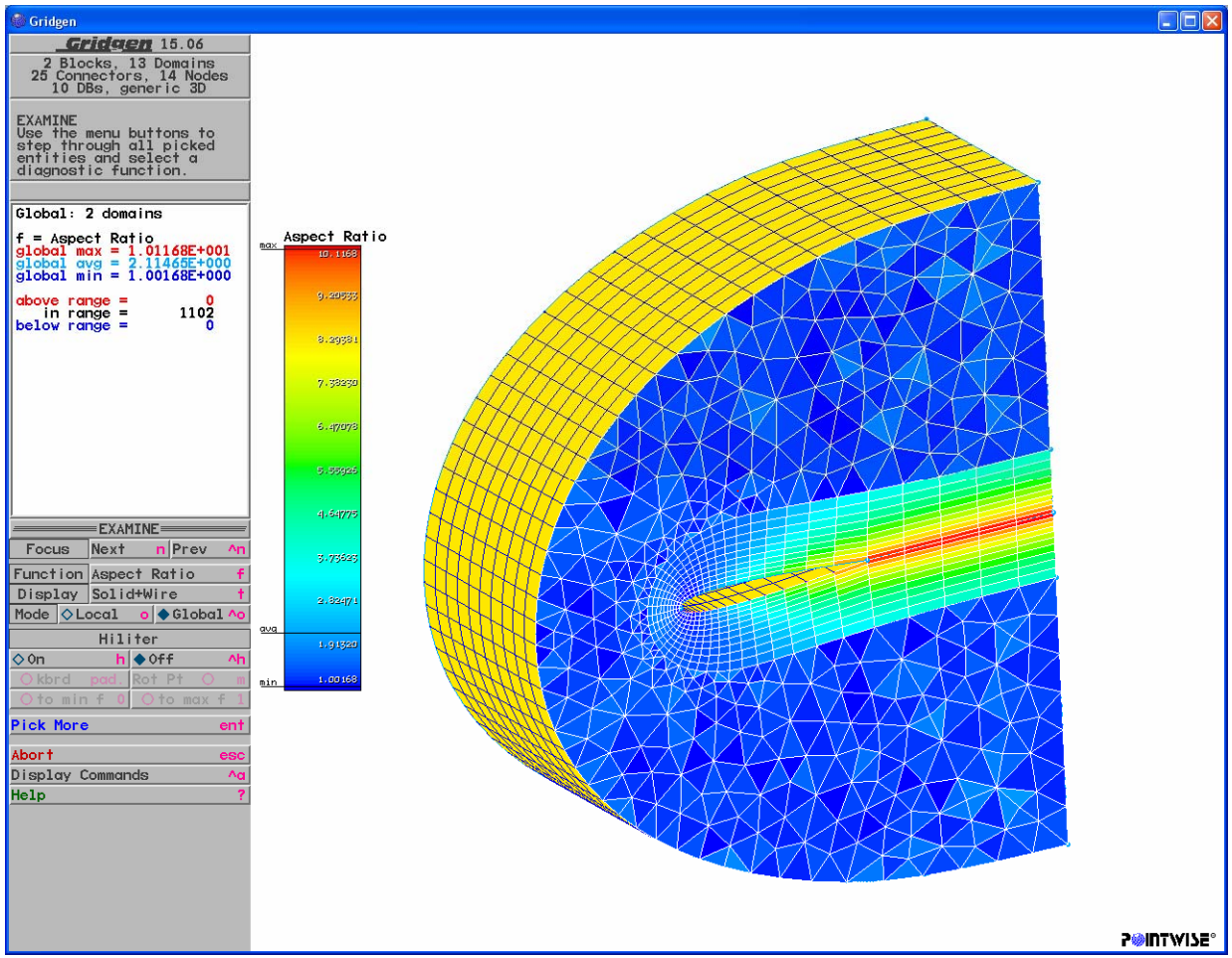
Examine Domains (Continued)

Here, the **Size Function** has been chosen (note the **Size** identifier next to **Function** in the menu). The two domains that make up the front surface of the mesh have been chosen, and their cell faces are now color coded by area. The color scale on the left of the display window shows the corresponding areas in model units. Again, note the summary information in the blackboard window.



Examine Domains (Continued)

Next, the **Aspect Ratio** has been chosen (note the identifier next to **Function** in the menu). Some cells appear to have aspect ratios greater than 10, a problem for some solvers. To find these, some changes to the display may be useful. This can be done by clicking the **Solid+Wire** button next to **Display**.



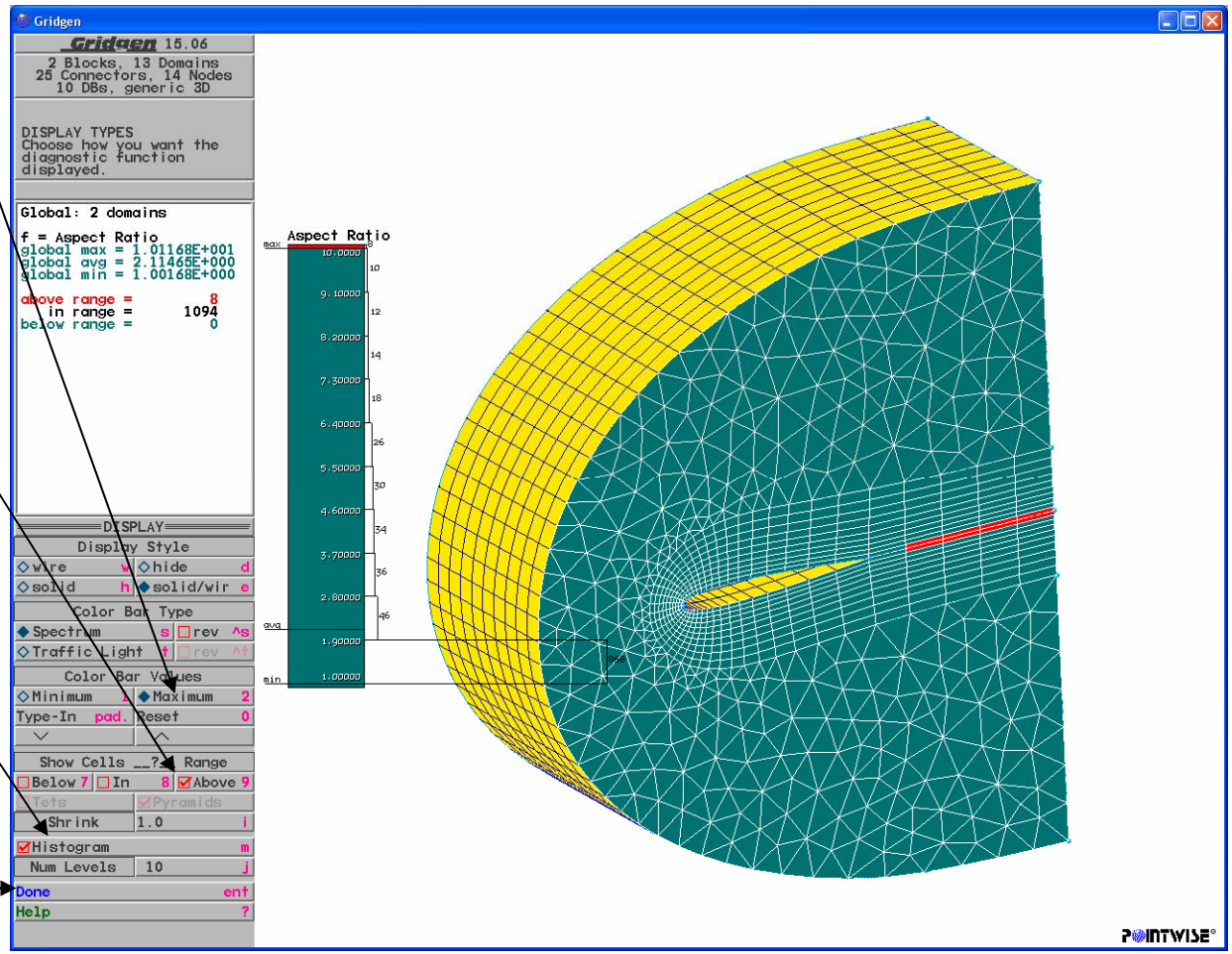
Examine Domains (Continued)

First, **Maximum** is selected under **Color Bar Values**, and then **Type-In** is picked so that 10 can be entered via the keyboard.

Next, **Above** is toggled under **Show Cells _ _ ? _ _ Range** so that only those cells with aspect ratios above 10 will be highlighted.

Then, **Histogram** is toggled to provide more information on numbers of cells vs. their aspect ratios.

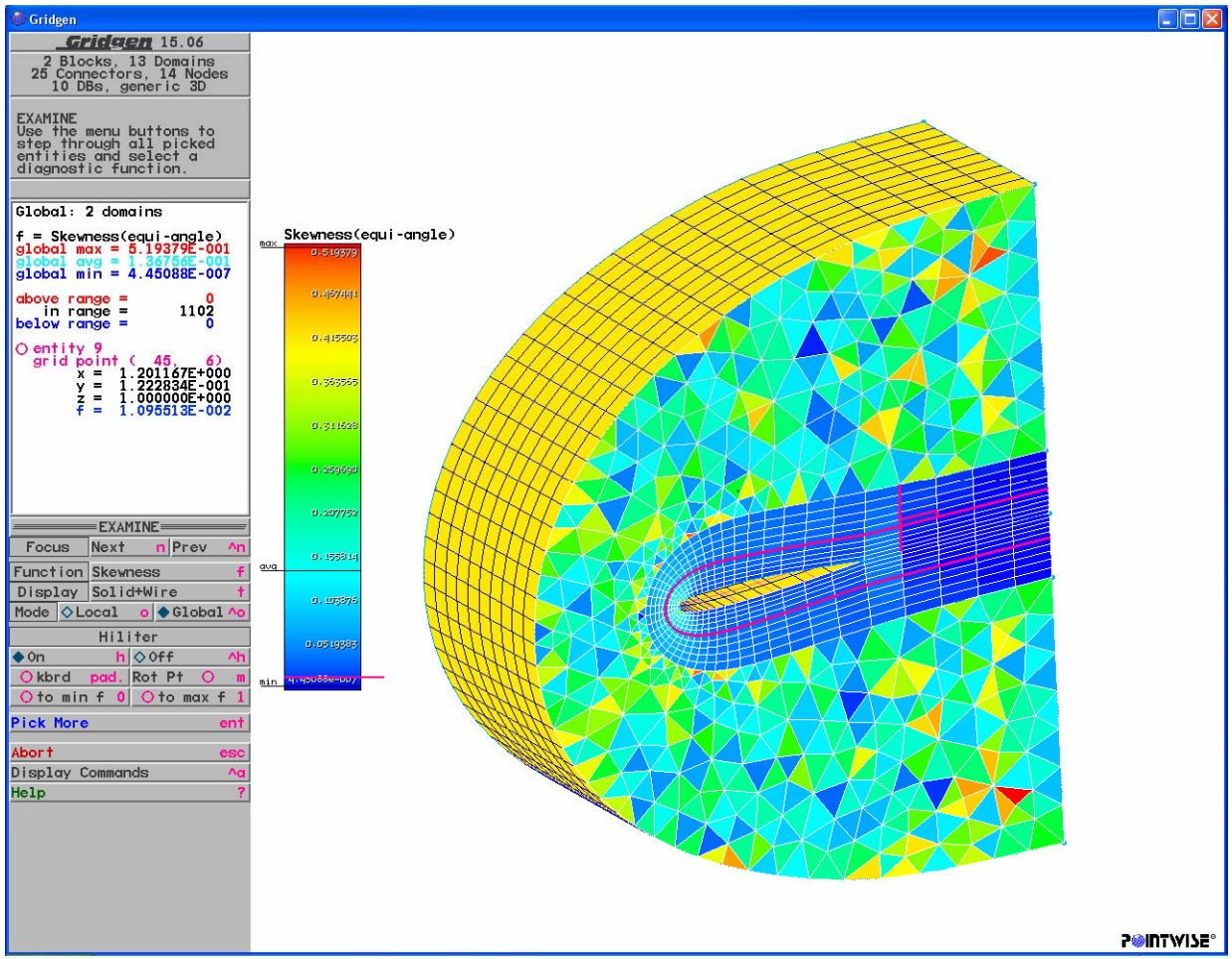
Clicking **Done** returns to the previous menu.



Examine Domains (Continued)

Finally, **Skewness** is chosen, with the **angle** option. As the plot shows, all the cells in these domains have very low skewness (a good thing).

In this case, the **Hiliter** has also been turned on and a set of cell indices (45, 6) entered via the **kbrd**. As a result, Gridgen highlights the cell's location in the display window and shows its skewness value in the blackboard window.



Examine Blocks

As before, clicking on **None** in the **EXAMINE** menu opens the **DIAGNOSTIC** submenu where the diagnostic function to be applied can be chosen.

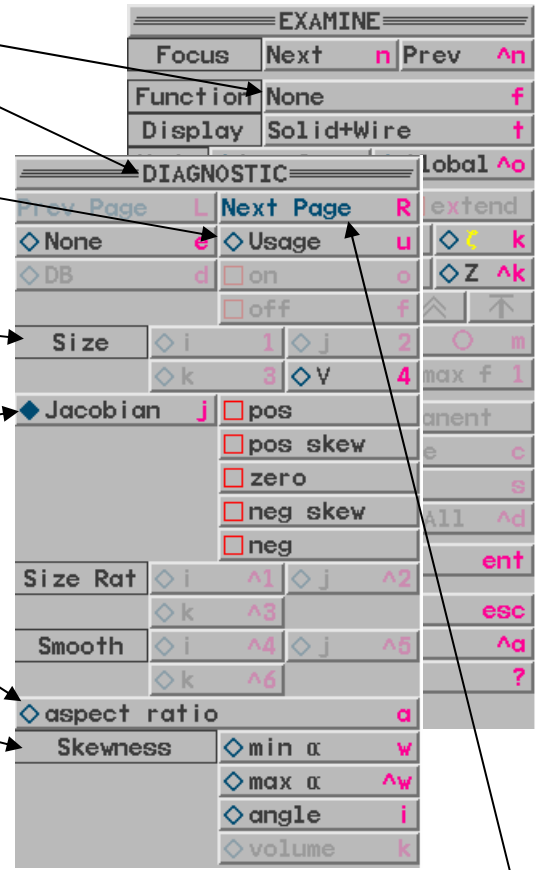
Usage will show what entities utilize or are utilized by the selected block(s).

Size will show the cell volumes in the selected block(s).

Jacobian will show cell's **pos**, **pos skew**, **zero**, **neg skew**, **neg** Jacobian, as toggled.

Aspect Ratio will show ratio of average length to average width of the cells in the block(s).

Skewness will show cell's minimum included angle (**min** α), maximum included angle (**max** α), or the maximum ratio of cell's included angle to equilateral triangle angle (**angle**).



As in the case of the **DIAGNOSTIC** menus for connectors and domains, **Next Page** must be clicked to reveal the **Done** button to return.

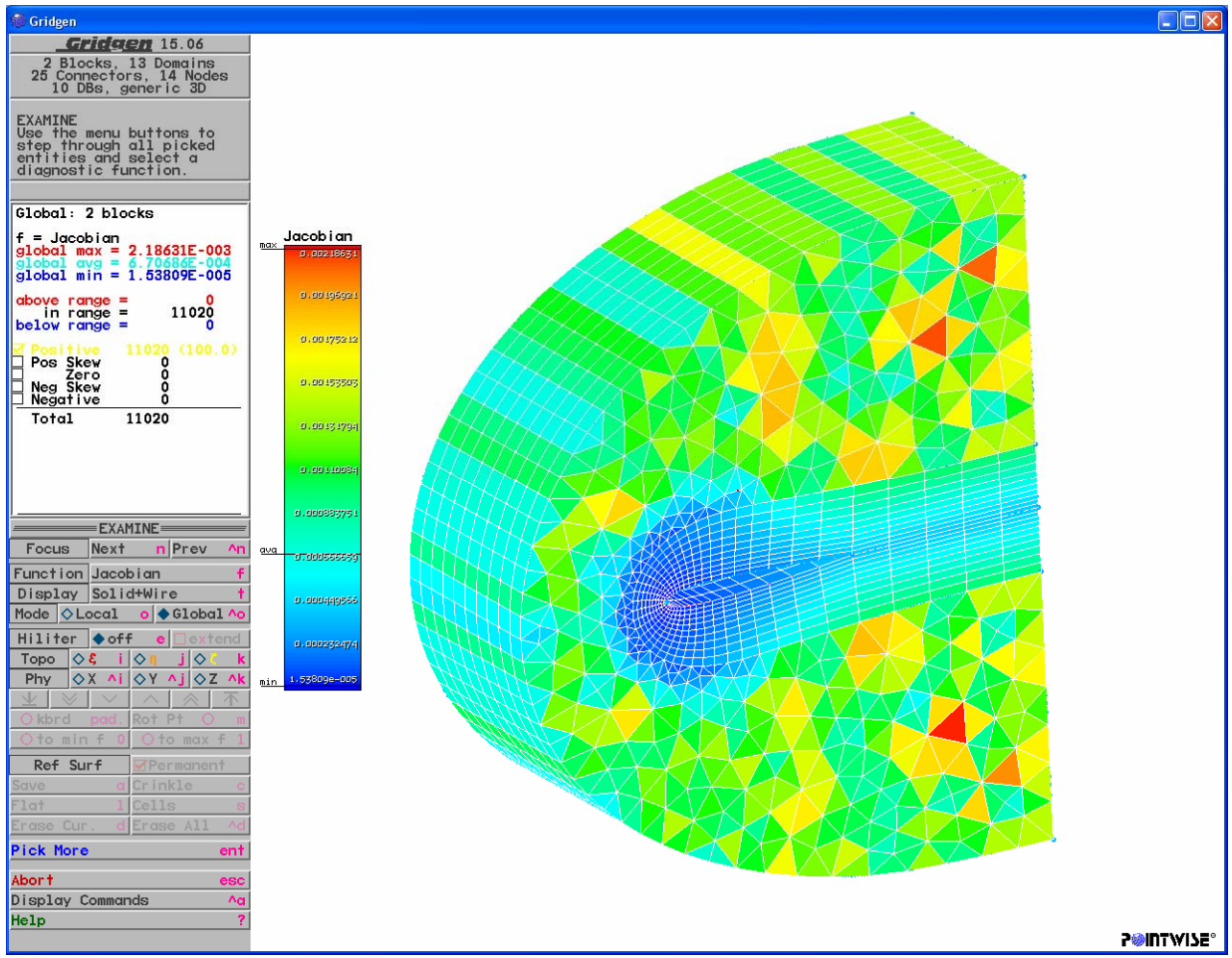
Examine Blocks (Continued)

Cell Jacobian Classifications

Jacobian Classification	Menu Button	Corner Jacobian Avg. Value	Corner Jacobian Signs
Positive	**pos**	Positive	All Positive
Positive Skew	**pos skew**	Positive	Mixed
Zero	**zero**	Zero	N/A
Negative Skew	**neg skew**	Negative	Mixed
Negative	**neg**	Negative	All Negative

Examine Blocks (Continued)

In this case, the positive (**Pos**) **Jacobian** for both blocks is being plotted. The blackboard window in this case shows the number of cells in all five Jacobian classifications (all positive in this instance).

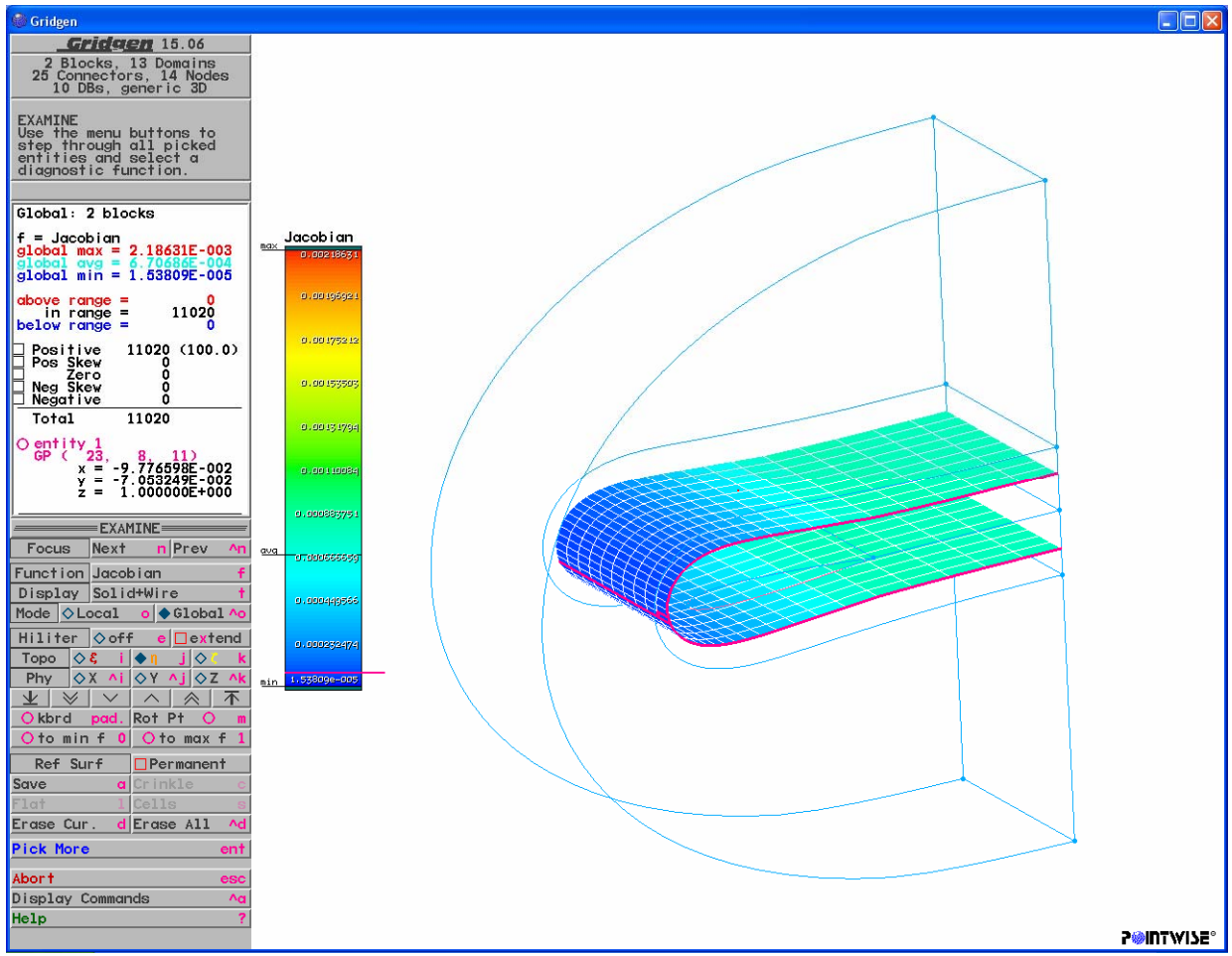


Examine Blocks (Continued)

The **Hiliter** can be used to examine cells on various surfaces within the block(s). There are two hiliters available, topological and physical.

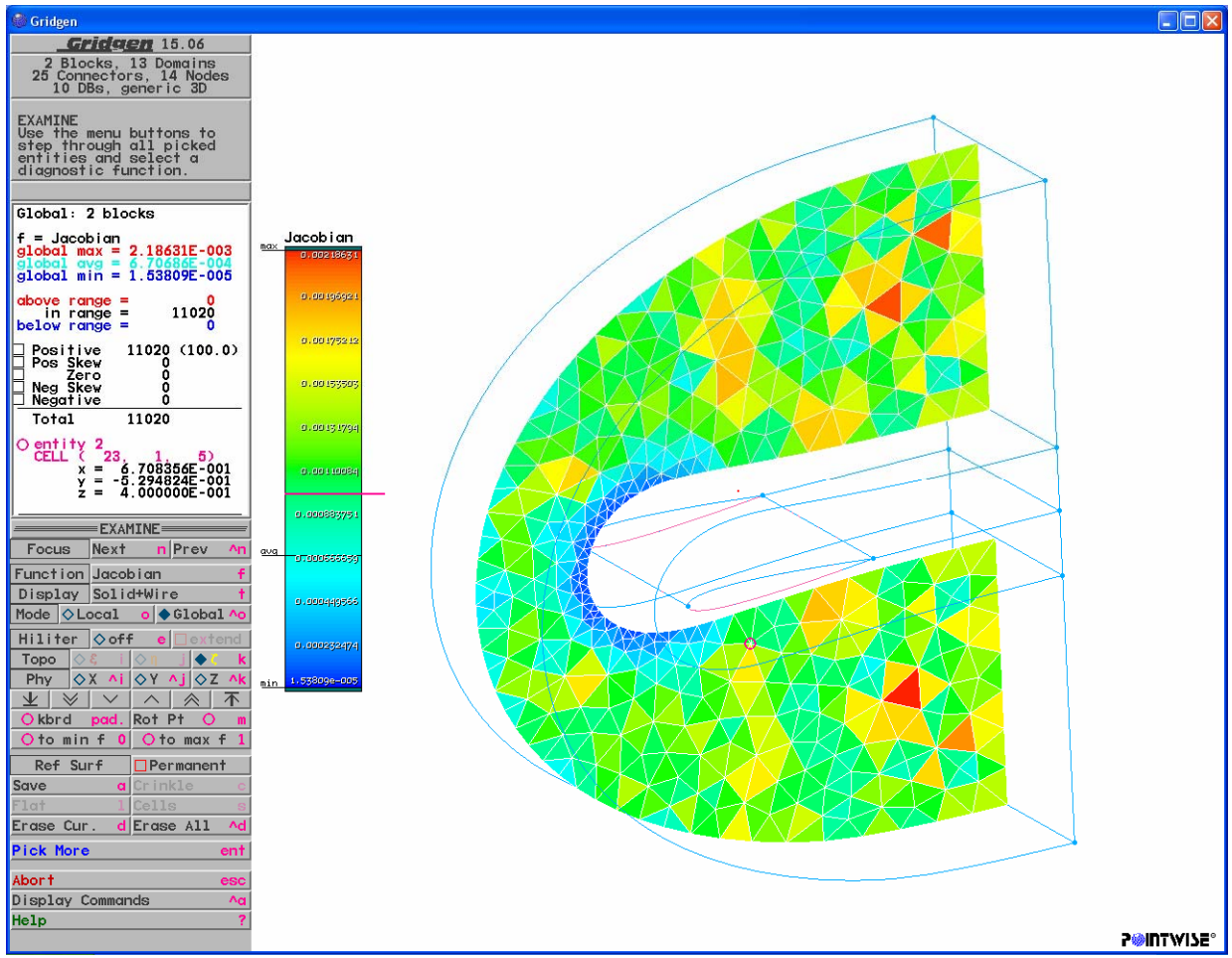
Topo (topological) displays constant computational coordinate planes (ξ, η, ζ) for structured grids blocks. Here a surface of constant η is shown.

The plane can be moved in the chosen coordinate direction by using the arrow keys on the keyboard or the row of arrow buttons (↧, etc.) in the menu.



Examine Blocks (Continued)

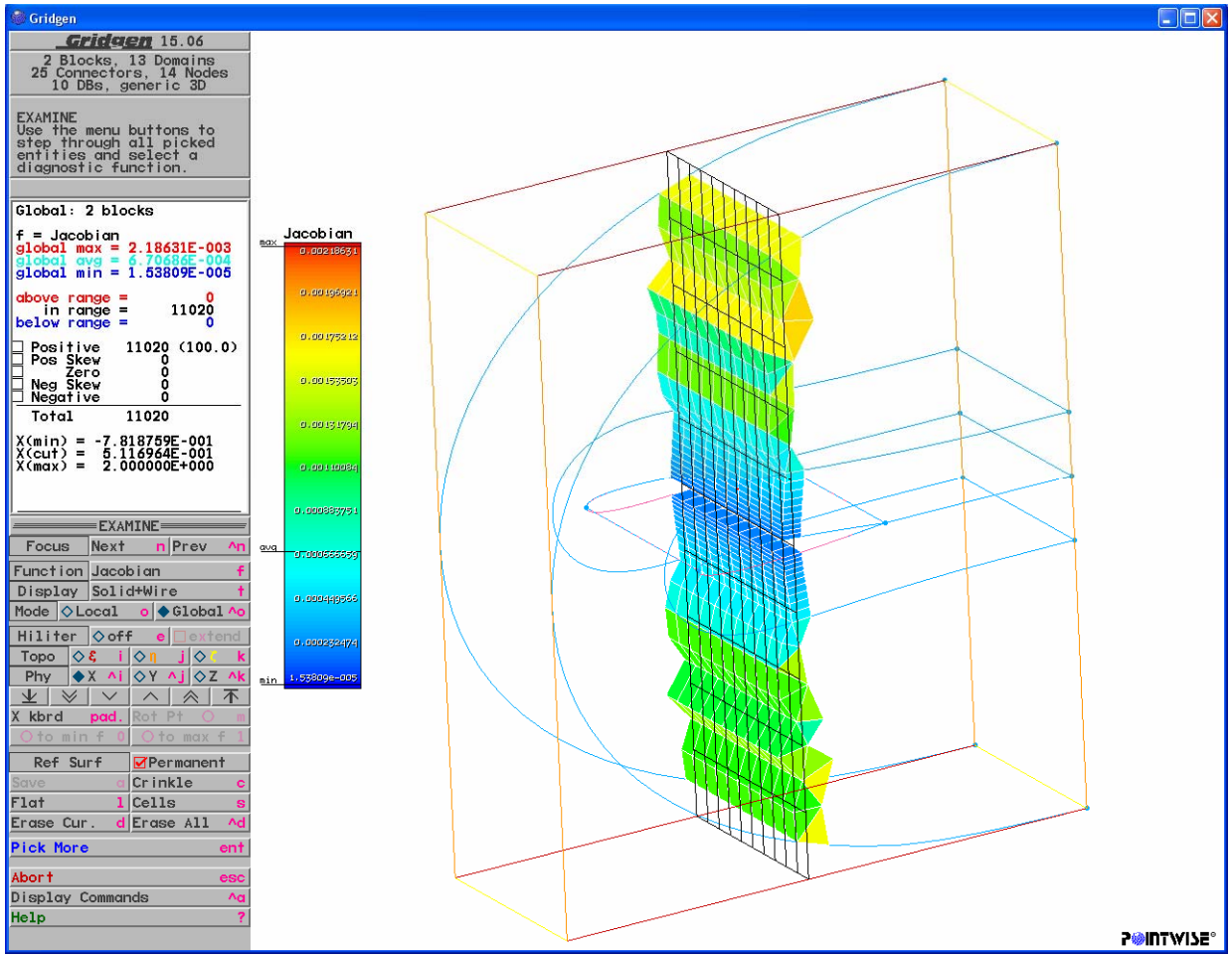
In the case of a prism block, only planes in the extrusion direction can be displayed in **Topo** mode, as shown here.



Examine Blocks (Continued)

Phy (physical) displays constant Cartesian coordinate cuts (**X, Y, Z**). The cutting plane (**Flat**) at that location or the layer of cells (**Cells**) intersecting that surface can be shown. Here the cells cut through a surface of constant **X** are shown.

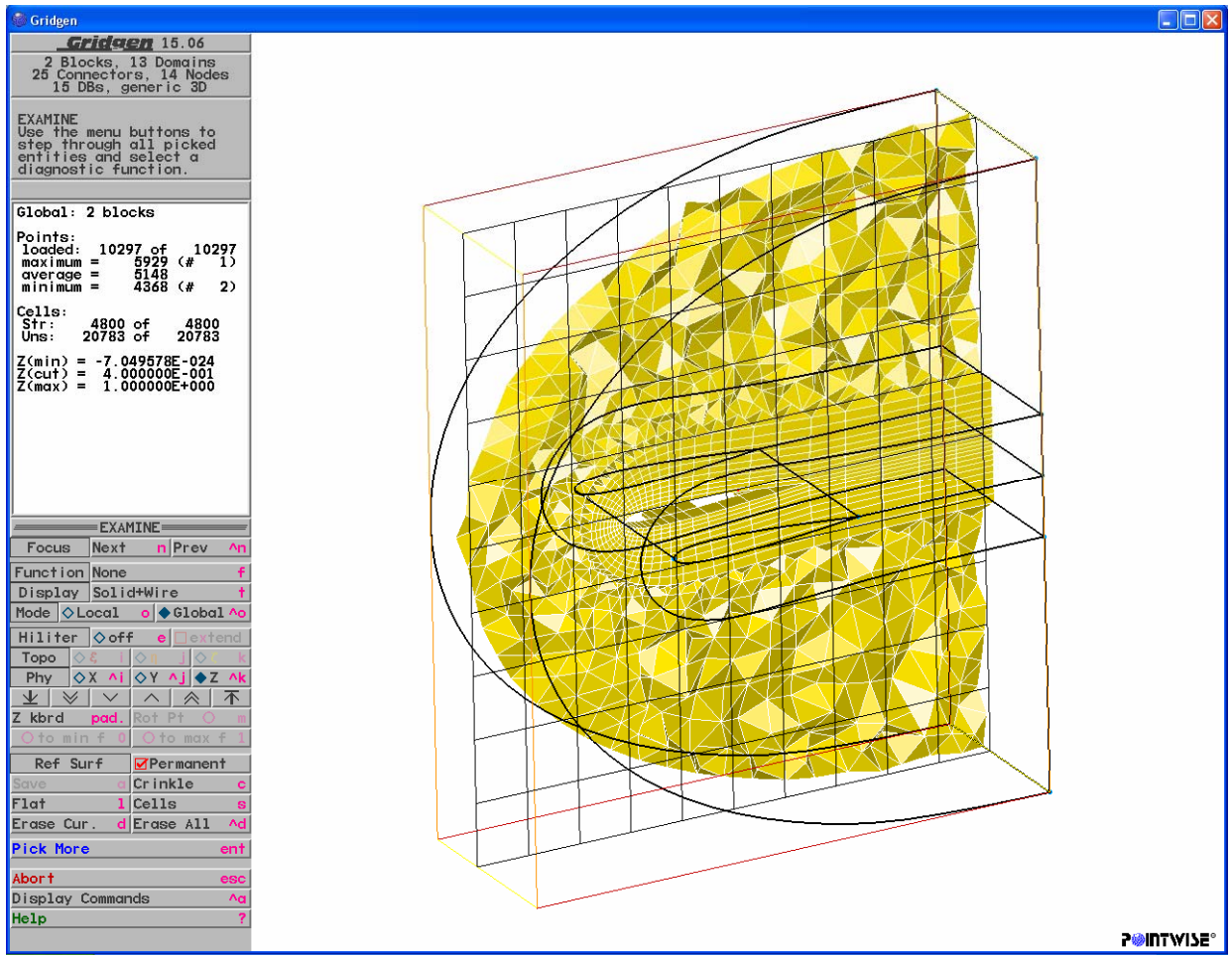
If ☐ **Permanent** is toggled, existing slices shown in display window are retained when a new one is created. If not, the previous plots are removed when a new one is made.



Examine Blocks (Continued)

For tet meshes, **Crinkle** is available. Here, for every touched tetrahedral cell that has only three of its four faces cut by the plane, the fourth uncut face becomes part of the crinkle surface. The resulting three-dimensional surface is not flat, but resembles a crinkled sheet of foil – hence the name.

This example also shows that if no function (**None**) is selected, only the mesh itself is rendered.



Solving Grid Problems

- The vast majority of cell issues encountered in Gridgen result from either (or both):
  - Connector point distribution or shape problems.
  - Domain point distribution or shape problems.
- Check to see if:
  - A change to a connector distribution requires a domain update.
  - A connector and/or domain projection has resulted in poor shape or distribution.
  - Adjacent domains' interiors intersect or cross.
  - Opposing distributions cause excessive pull or twist in the grid interior.

Gridgen**Training**

- In this session:
  - Building 3D Unstructured Grids.
  - (Re)Dimensioning and (Re)Distributing Unstructured Grids.
  - Examining and Correcting Grids.
  - Tutorial: “Pierced Elbow: Basic Unstructured Grid.”

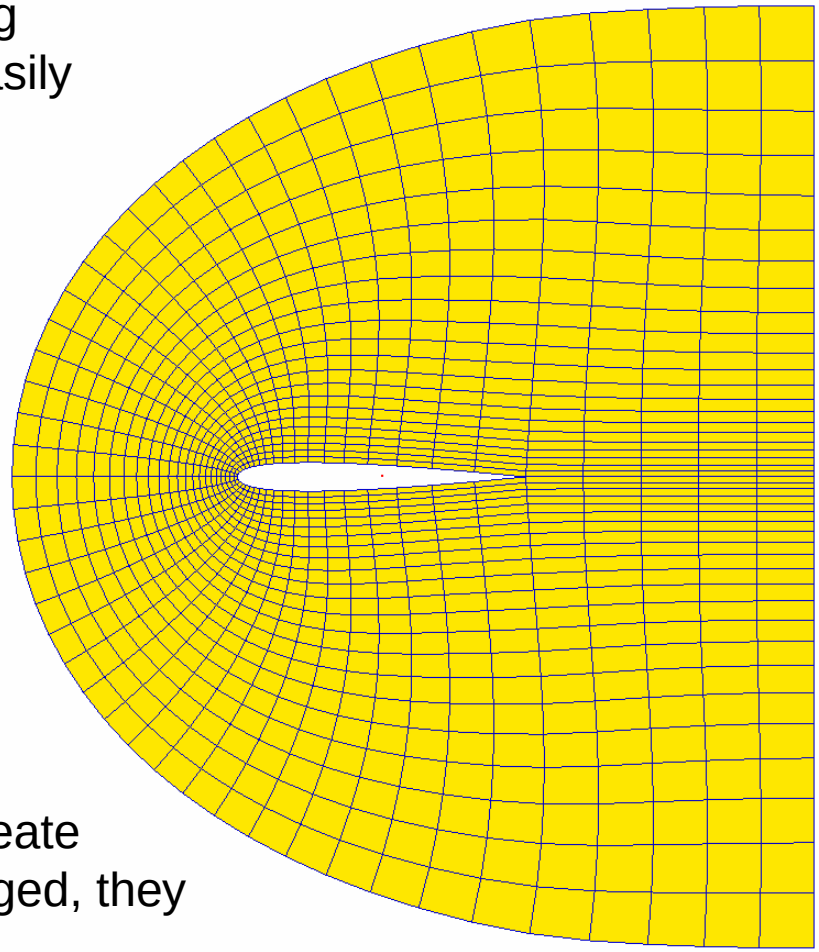
Gridgen**Training**

- In this session:
  - Building 2D Structured Extruded Grids.
  - Building 3D Structured Extruded Grids.
  - Tutorial: “Swept Ramp: Advanced Multi Block Structured Grid.”
  - Building 3D Unstructured Extruded Grids.
  - Creating Hybrid Grids.
  - Tutorial: “Droplet in a Cylinder: Hybrid Grid.”

Building Structured Extruded Grids

Extrusion is another means of quickly building meshes with Gridgen. 2D domains can be easily extruded from connectors or edges, while 3D blocks can be readily extruded from domains or faces. In either case, there are a number of different marching schemes available:

- Hyperbolic.
- Normal.
- Rotational.
- Translational.
- Path.



Let's first look at applying these to build structured extruded grids. Since the steps to select analysis software, import geometry, create connectors, and dimension them are unchanged, they will be skipped.

The “Bottom-Up”Gridgen Process

Select Analysis Software and Mesh Dimensionality

⇩

Import and Manipulate Database Geometry

⇩

Create Segments and Connectors

⇩

Dimension Connectors and Distribute Grid Points

⇩

Extrude Domains

⇩

Assemble Blocks

⇩

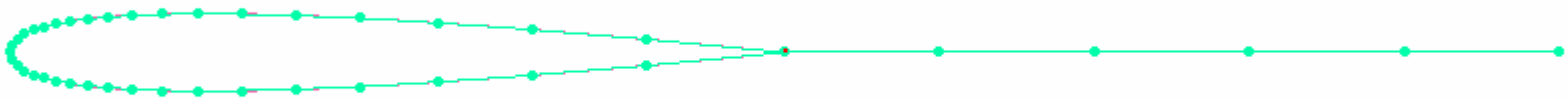
Setup CFD Analysis Boundaries

⇩

Export and Save Files

Creating Demo Connectors

For the classroom extrusion demos, it will be convenient to define the system of three connectors shown below.



1. Import the airfoil database file (reext.dba).
2. Create connectors on the upper and lower airfoil database entities.
3. Dimension them to 20 and set Δs to 0.01 on both at the airfoil leading edge.
4. Create a two point connector from 1,0,0 (the trailing edge) to 2,0,0.
5. Dimension this connector to 6.
6. Save the file for later use.

Extrude Domains

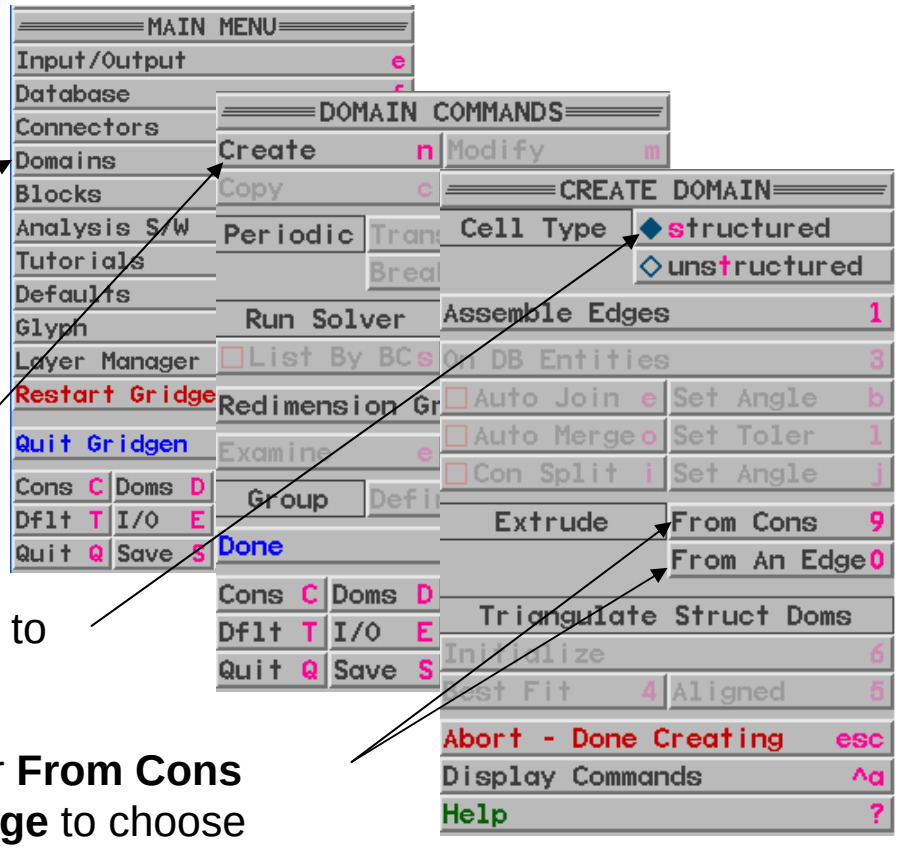
With one or more dimensioned connectors in place, domains can be extruded from either the individual connectors or an edge formed by combining the connectors.

From the **MAIN MENU**, select **Domains** to open the **DOMAIN COMMANDS** submenu.

From here, select **Create** to open **CREATE DOMAINS**.

Be sure that the **Cell Type** is set to **structured**.

Next, it's time to select either **From Cons** (connectors) or **From An Edge** to choose which will be the basis of the extrusion.

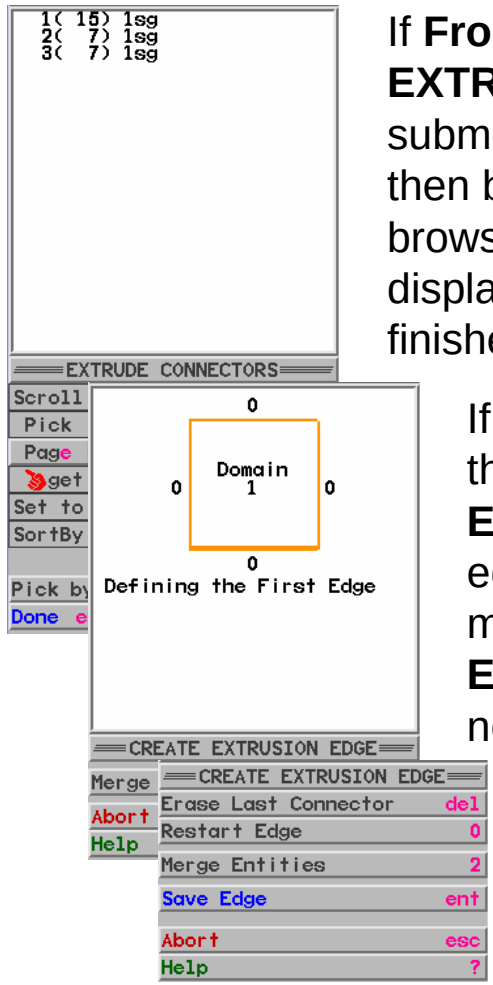


Cons vs. Edges

From Cons – A separate extrusion is performed on each individual connector selected, thus allowing each to have it's own extrusion attributes. A floating boundary condition is applied at shared nodes so adjacent extrusions blend smoothly. Each connector extrudes to a separate domain.

From An Edge – The selected connectors form a single edge from which a single new domain is extruded.

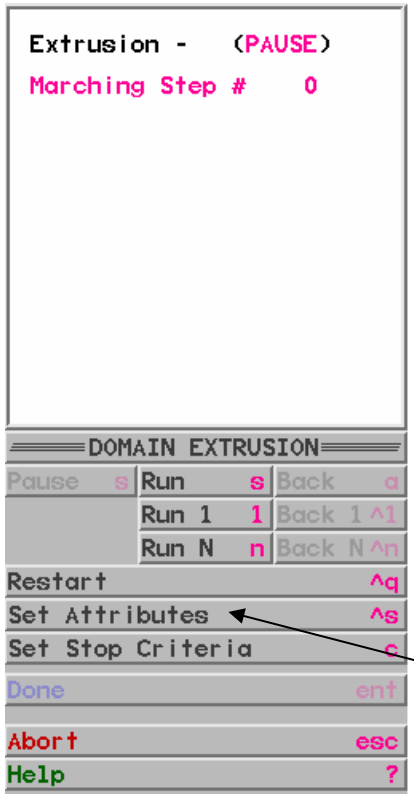
Generally, **From An Edge** is preferred for mesh quality and extrusion algorithm numerical stability reasons.



If **From Cons** was selected, the **EXTRUDE CONNECTORS** submenu appears. Connectors can then be picked by mouse from the browser window or directly from the display window. When picking is finished, **Done** is clicked.

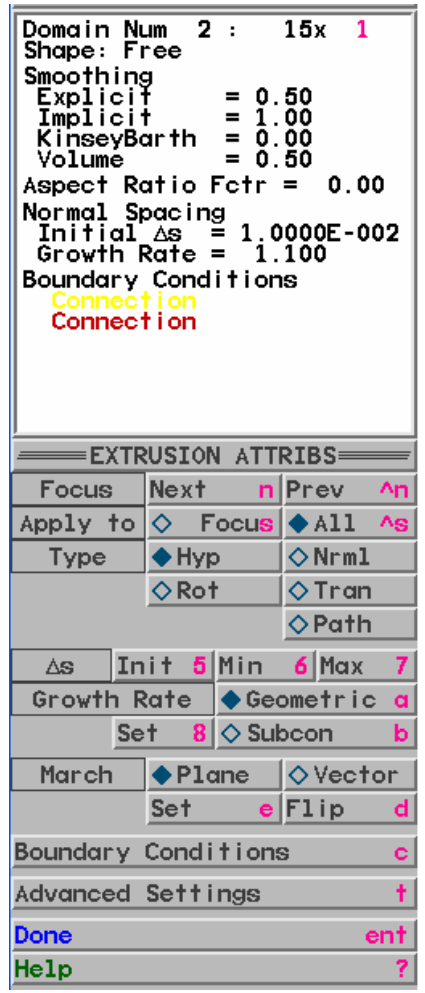
If **From An Edge** was selected, the **CREATE EXTRUSION EDGE** submenu appears. The edge is formed in the same manner as with the **Assemble Edges** command. Once connectors begin to be added, the menu changes to reveal more commands. **Save Edge** is clicked when the edge is complete.

Extrusion Attributes

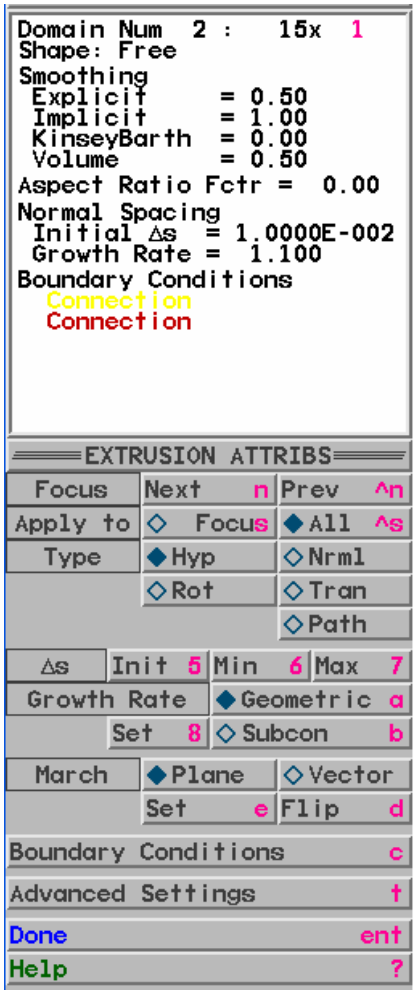


When the connector(s) or edge for the extrusion is defined, the **DOMAIN EXTRUSION** menu appears, but before marching through the extrusion, there are several attributes for the process that need to be set or, at least, reviewed.

Selecting **Set Attributes** brings up the **EXTRUSION ATTRIBS** submenu, with the blackboard window now showing the current values for most of the attributes.



Extrusion Types



The first attribute that needs to be selected is the **Type** as this will determine the basic character of the domain to be extruded and also bring up the appropriate command buttons to define the extrusion:

Hyperbolic (Hyp) – Based on the solution of a set of PDEs prescribing a orthogonal trajectory for the grid lines from the initial grid entity.

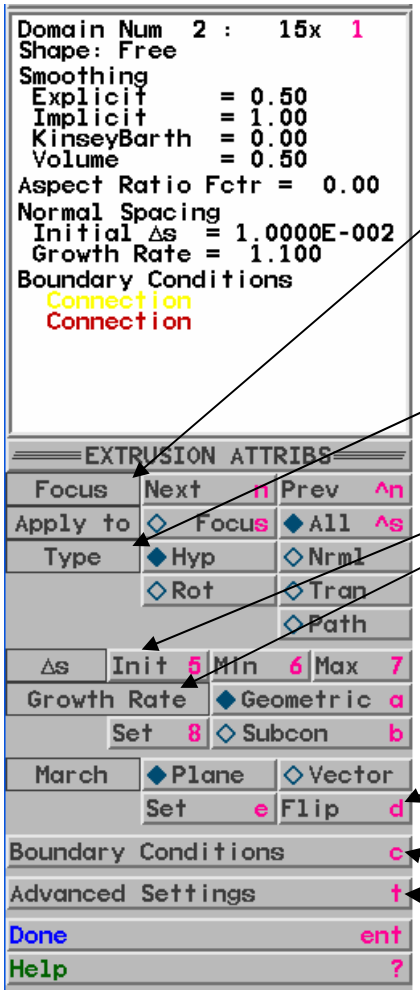
Normal (Nrml) – An algebraic method that mimics the hyperbolic technique. The hyperbolic method is preferred.

Rotational (Rot) – An algebraic method that sweeps grid in a circular path about a prescribed axis.

Translational (Tran) – An algebraic method that sweeps grid in a linear path in a prescribed direction.

Path – An algebraic method that sweeps grid along a connector defined path.

Hyp and **Nrml** Extrusion Attributes

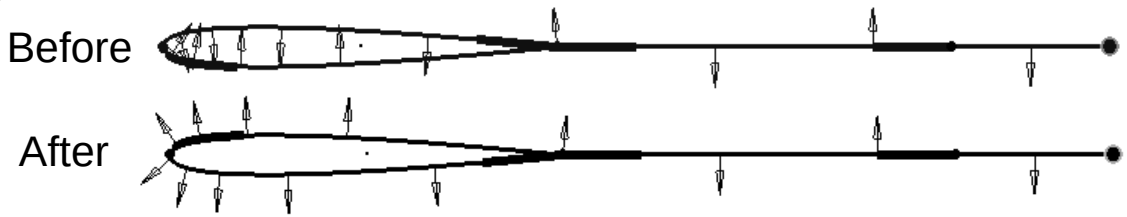


If extruding from a set of connectors, **Focus** allows the attributes to be set for individual connectors. This feature is not available if extruding from an edge.

Hyp or **Nrml** are the **Type** of extrusion set.

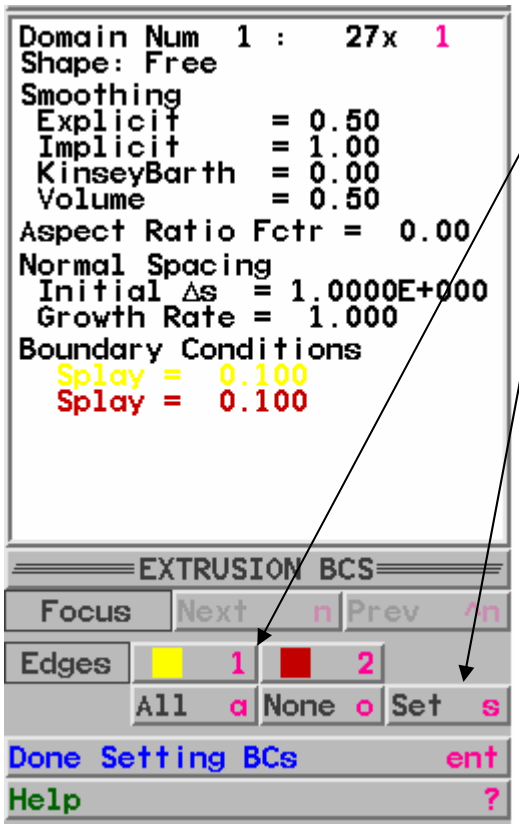
The initial spacing, **Δs Init**, for the extruded cells can be set, as can the characteristics of the **Growth Rate**.

If necessary, **flip** can be used to change the marching direction.



Boundary Conditions and **Advanced Settings** opens more menus for setting additional controls.

Hyp and **Nrml** Boundary Conditions



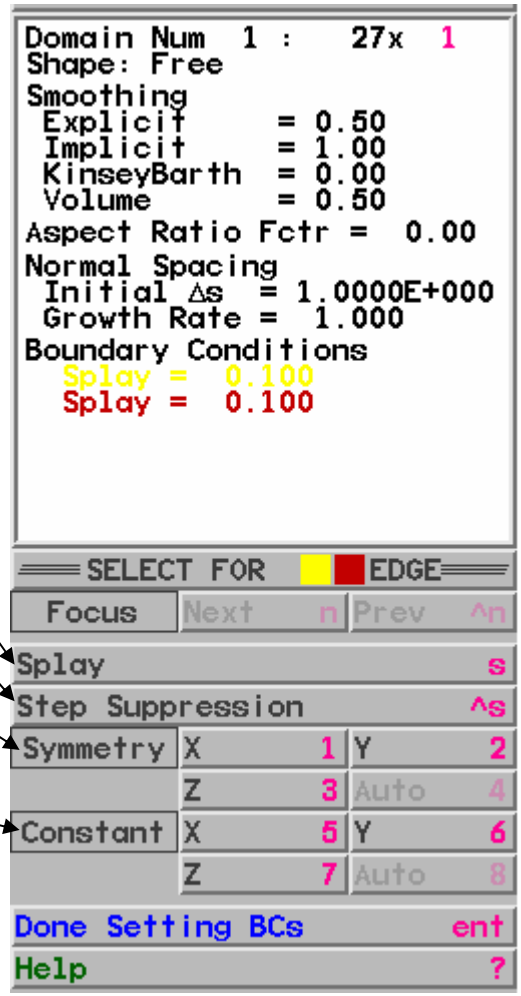
Once the desired boundaries have been selected (by picking the appropriate color coded boxes), the boundary conditions can be **Set**.

Splay sets the degree of marching front expansion.

Step Suppression lags or accelerates the marching front at nodes.

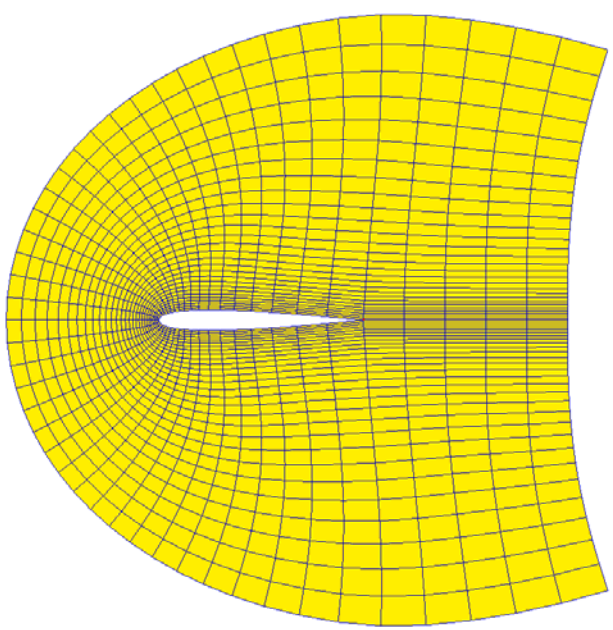
Symmetry X, Y, Z provides true symmetry boundary condition involving ghost.

Constant X, Y, Z forces constant value boundary condition. Gridlines may intersect symmetry plane non-normal.

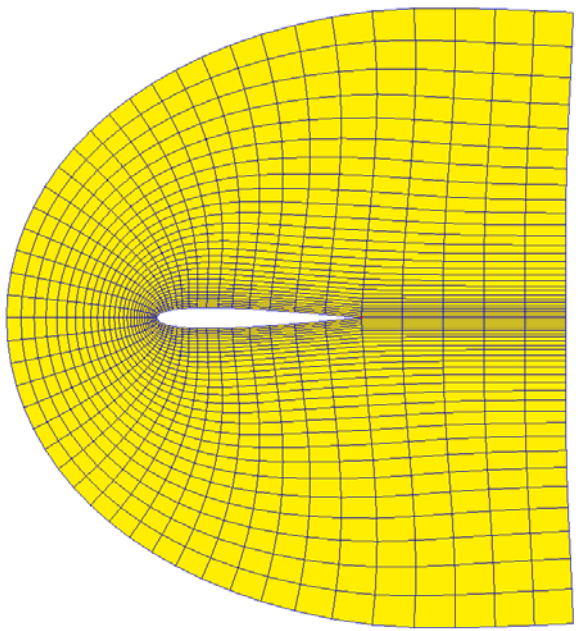


Hyp and **Nrml** BC Effects

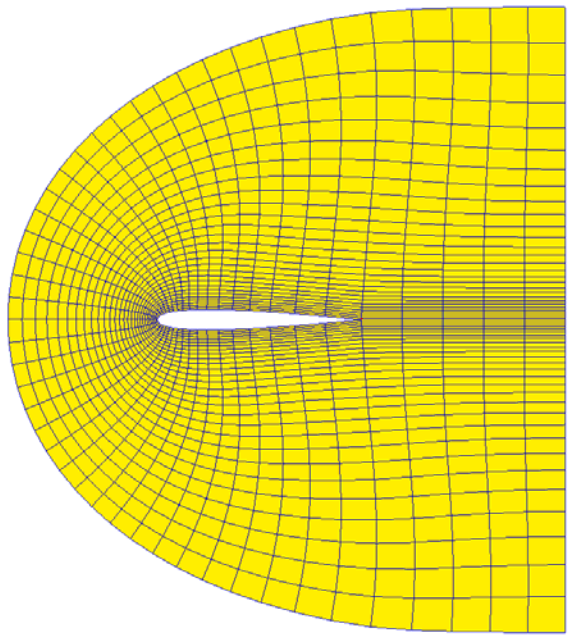
The effect of varying **Splay** or holding a **Constant X** at the downstream edge of the advancing grid front is shown below. Note that even zero **Splay** still leads to a slight downstream drift in the boundary as marching progresses.



Splay = 0.1 (Default)



Splay = 0.0

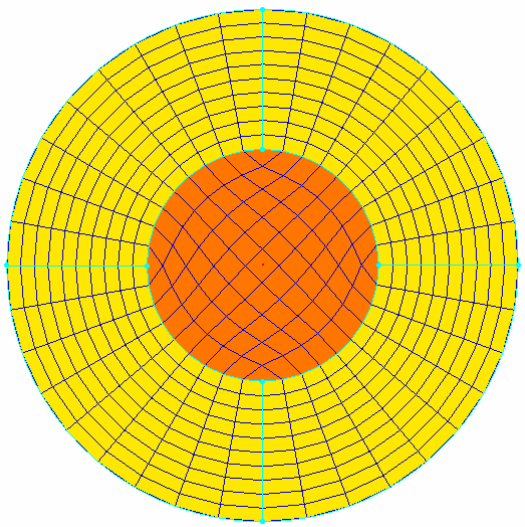


Constant X

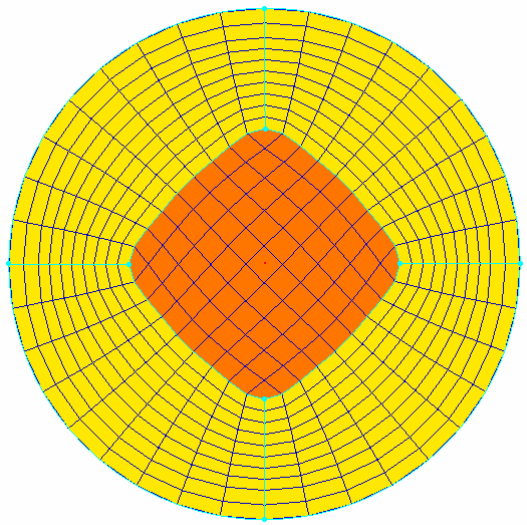
Hyp and **Nrml** BC Effects (Continued)

The effect of **Step Suppression** for **Nrml** extrusions is illustrated below. The outer portion of the grids is generated first by extruding inward from the four connectors on the outer edge - on the left with the default **Step**

Suppression of 0, on the right it set to –0.5. The negative coefficient retards the extrusion front at the boundaries of the growing domains, producing a more square middle region. This leads to a better quality H grid in the center.

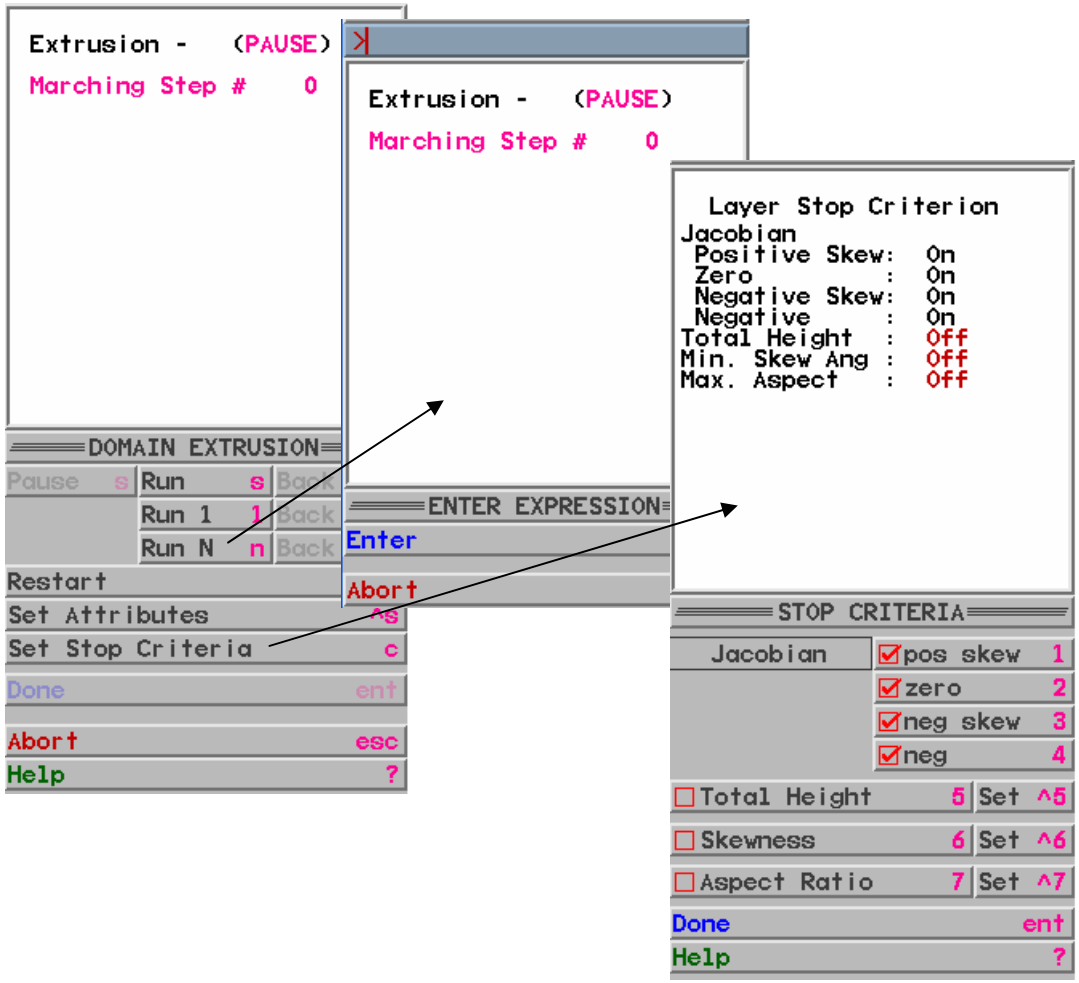


Step Suppression = 0 (Default)



Step Suppression = -0.5

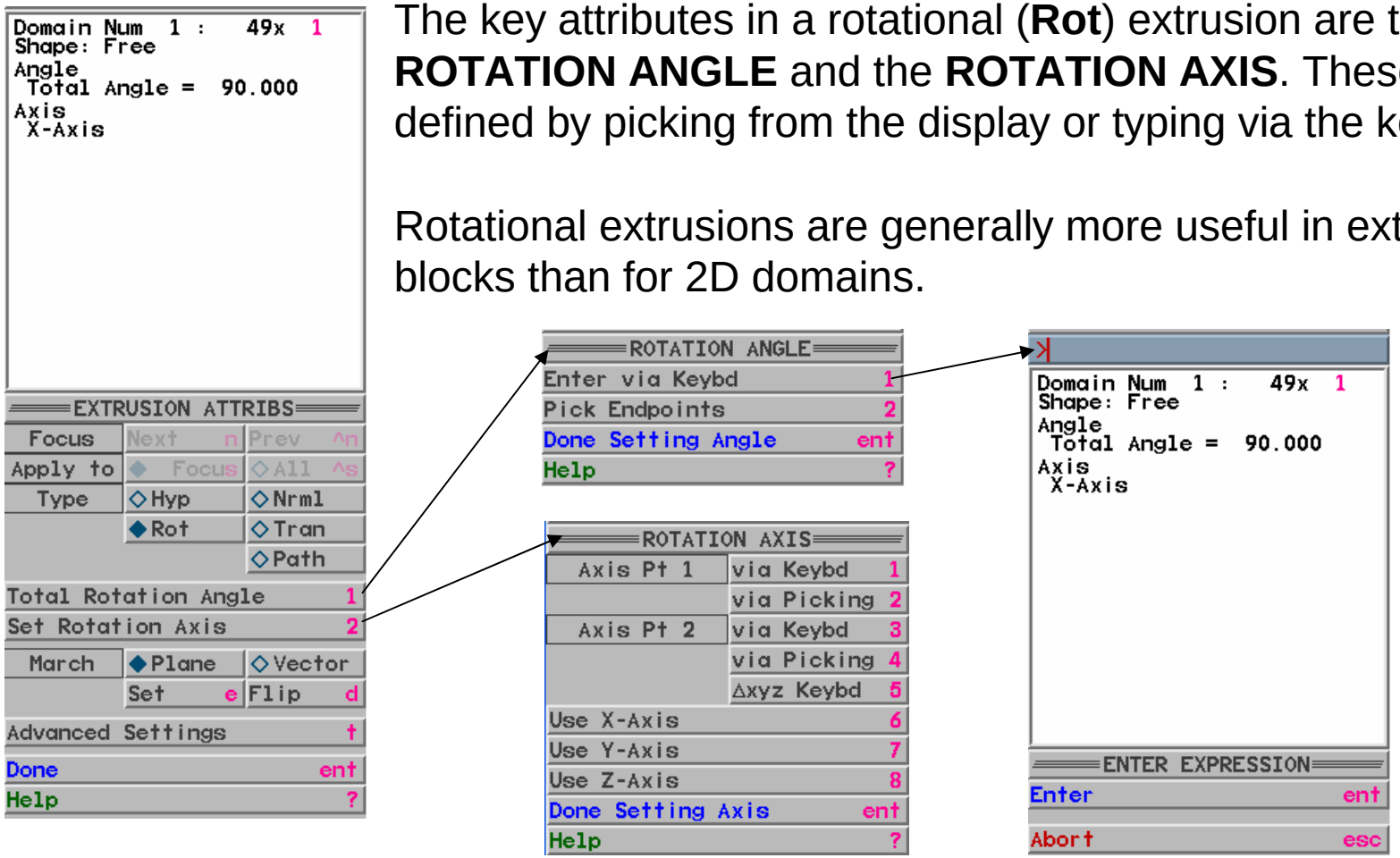
Hyp and **Nrml** Extrusion



After selecting **Done** in the **EXTRUSION ATTRIBS** menu, the marching process can begin back in the DOMAIN EXTRUSION menu. Extrusion can be advanced one layer at a time (**Run 1**), for multiple layers (**Run N**), or until a variety of criteria are met (**Set Stop Criteria**).

If the latter two are selected, the windows change to offer appropriate options to complete the selection process.

Rot Extrusion Attributes



The key attributes in a rotational (**Rot**) extrusion are the **ROTATION ANGLE** and the **ROTATION AXIS**. These can be defined by picking from the display or typing via the keyboard.

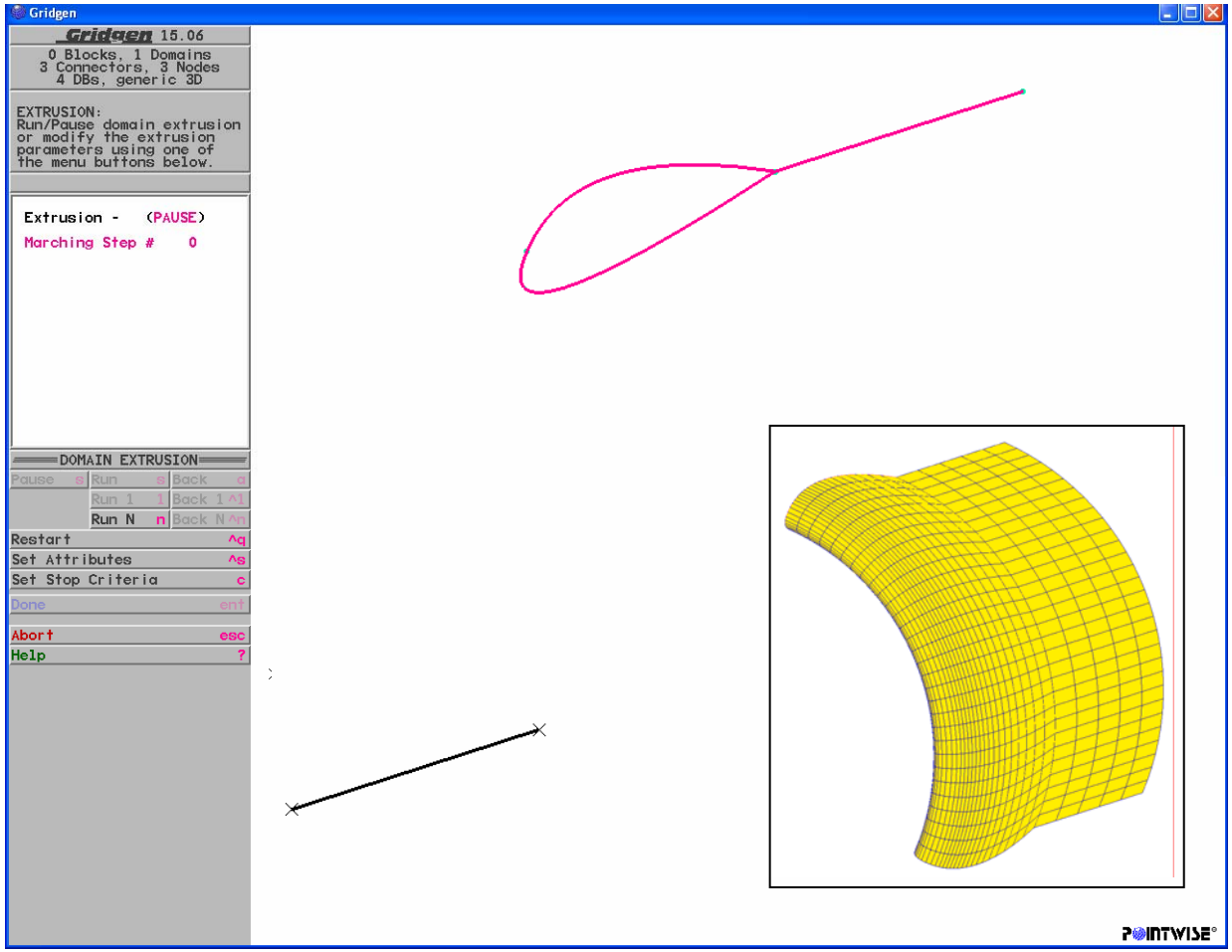
Rotational extrusions are generally more useful in extruding 3D blocks than for 2D domains.

Keyboard entry for axis points is similar.

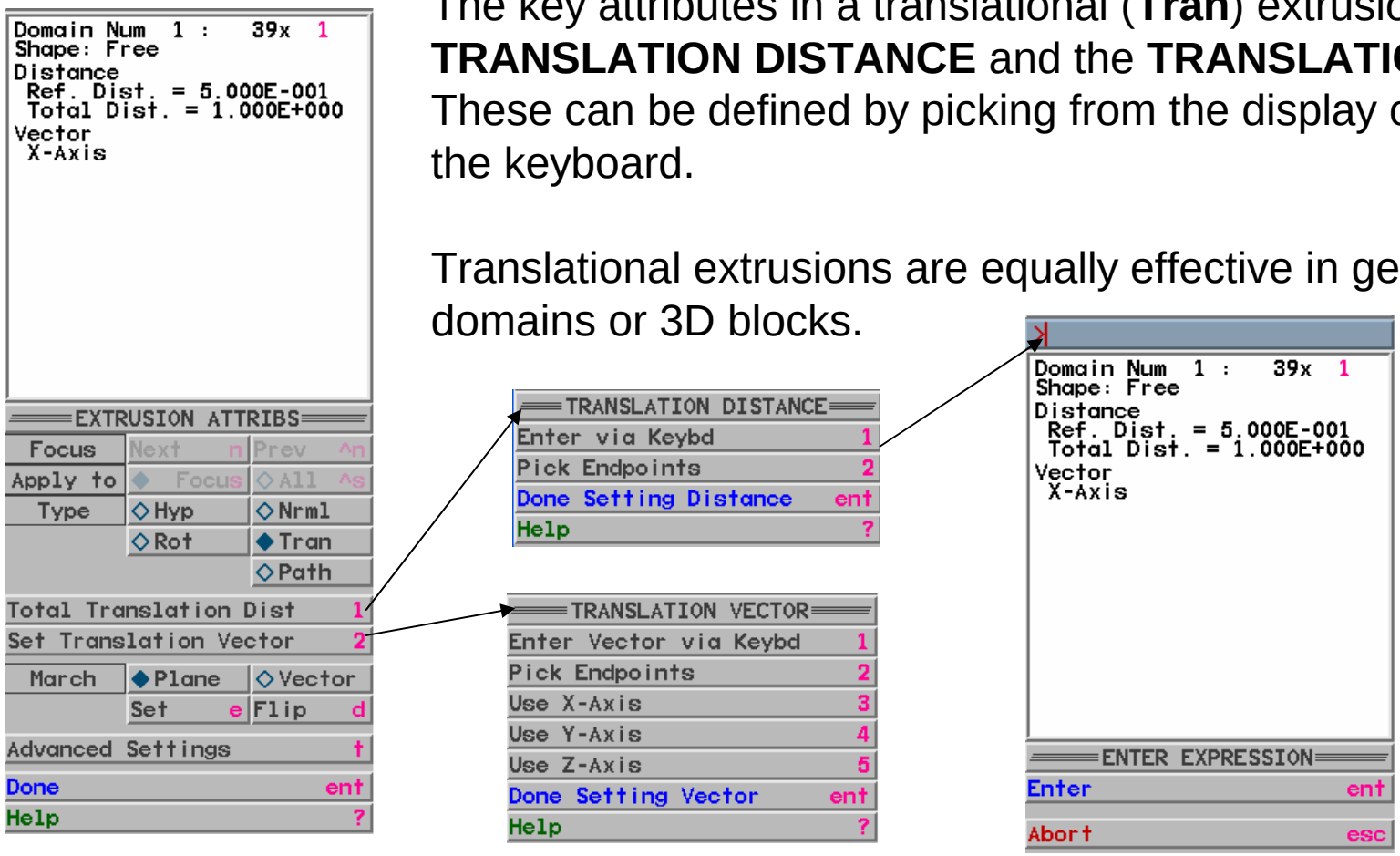
Rot Extrusion

With the key attributes set, all that remains is to define the number of steps that will be taken to march through the specified rotation angle. Once **Run N** is clicked, this can be entered via the keyboard.

Note that the rotation axis defined for the extrusion is shown in the display window. In this case, the axis was defined as stretching from 0,-0.5,0 to 1,-0.5,0. The inset image shows the extruded grid.



Tran Extrusion Attributes



The key attributes in a translational (**Tran**) extrusion are the **TRANSLATION DISTANCE** and the **TRANSLATION VECTOR**. These can be defined by picking from the display or typing via the keyboard.

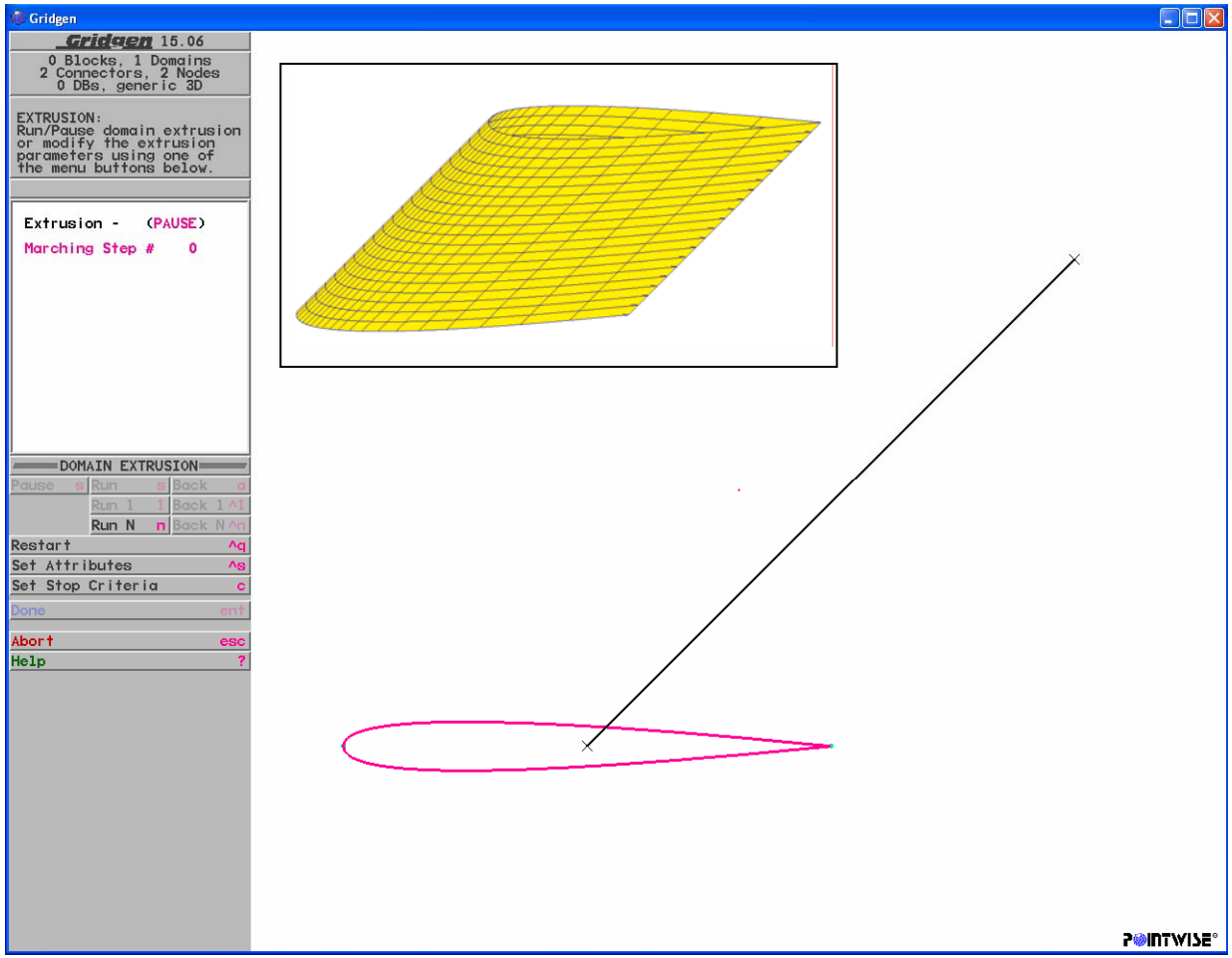
Translational extrusions are equally effective in generating 2D domains or 3D blocks.

Keyboard entry for translation vector is similar.

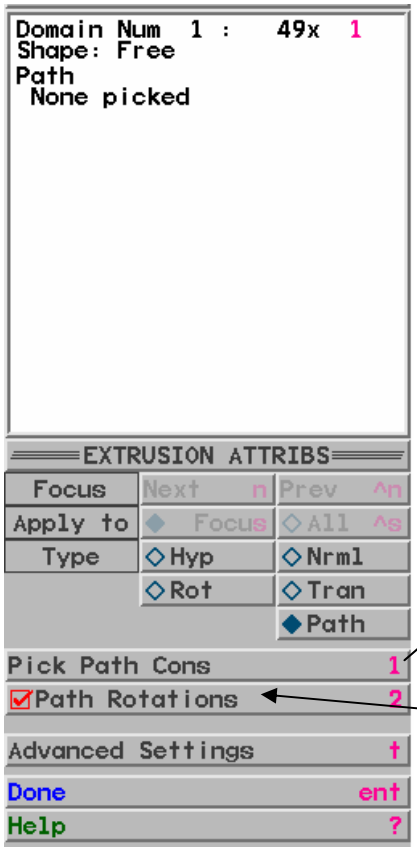
Tran Extrusion

With the key attributes set, all that remains is to define the number of steps that will be taken to march the specified distance. Once **Run N** is clicked, this can be entered via the keyboard.

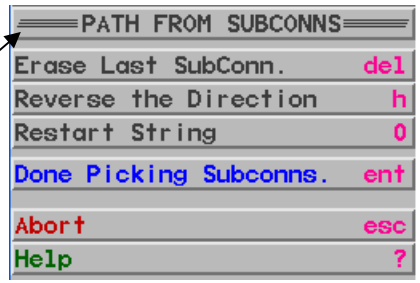
Note that the translation vector defined for the extrusion is shown in the display window. In this case, the vector was defined as 1,1,1. The inset image shows the extruded grid.



Path Extrusion Attributes



Unlike rotational and translational extrusions, the direction and, to a large extent, the distance that a path extrusion will encompass are defined not by any set attributes, but rather the dimensioned connector(s) that make up the path, since successive layers are spaced according to the node distribution along these connector(s).



Path Rotations controls whether successive extrusion layers will maintain the same relative angle to the path – toggled off gives a stacked grid.

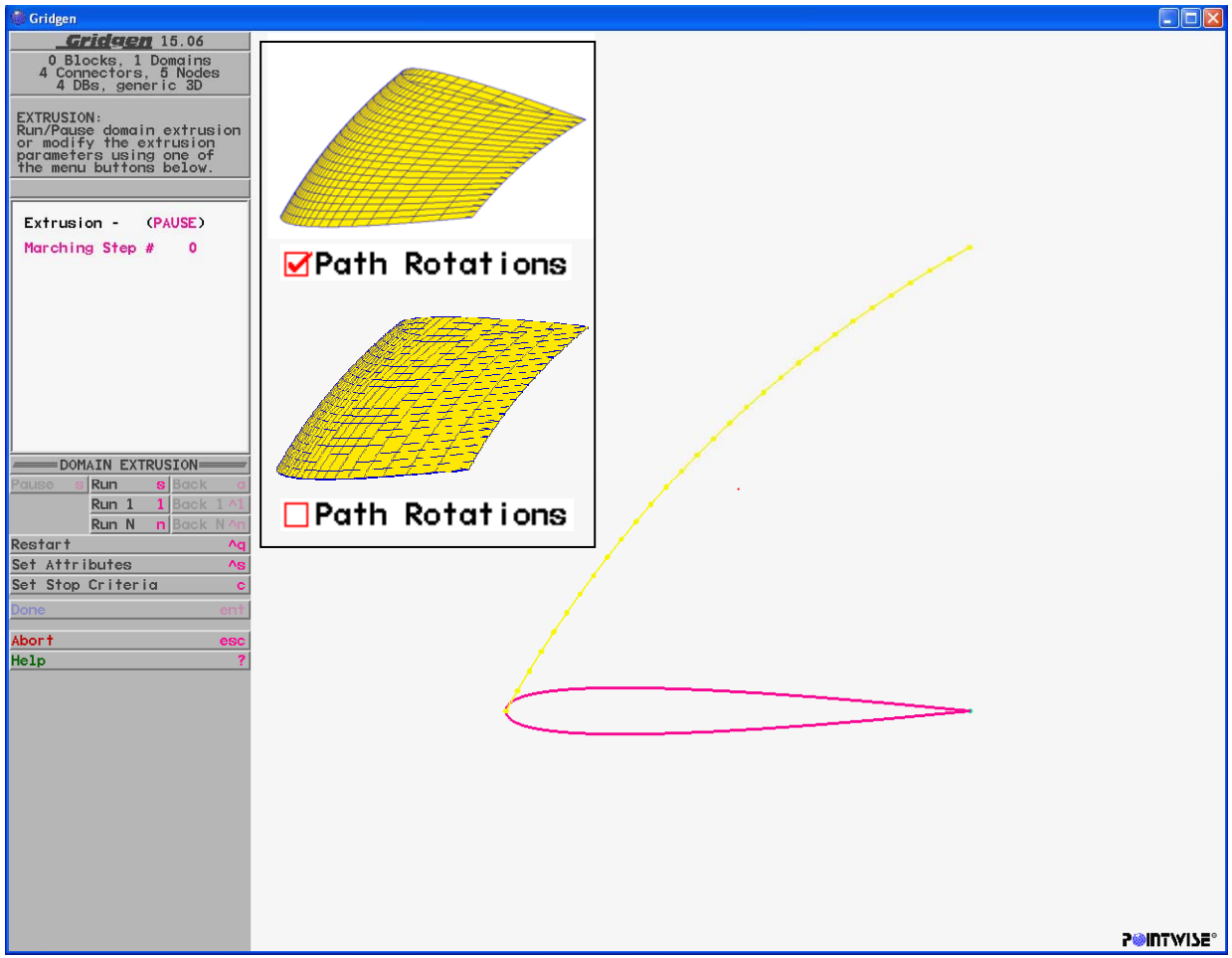
Once **Pick Path Cons** is selected, the menu changes and the connector(s) that will make up the path can be picked via the mouse from the display window. Various options for erasing subconnectors from the path, reversing the direction of the path, etc. are also available. **Done Picking Subconns.** ends path selection.

Path Extrusion

With the path selected, all that remains is to define the number of nodes to be marched down the path connector(s). Once **Run N** is clicked, this can be entered via the keyboard.

Note that the path connector(s) are shown as are their nodes in the display window.

The inset images show the resulting extrusions, with and without the **Path Rotations** toggle selected.

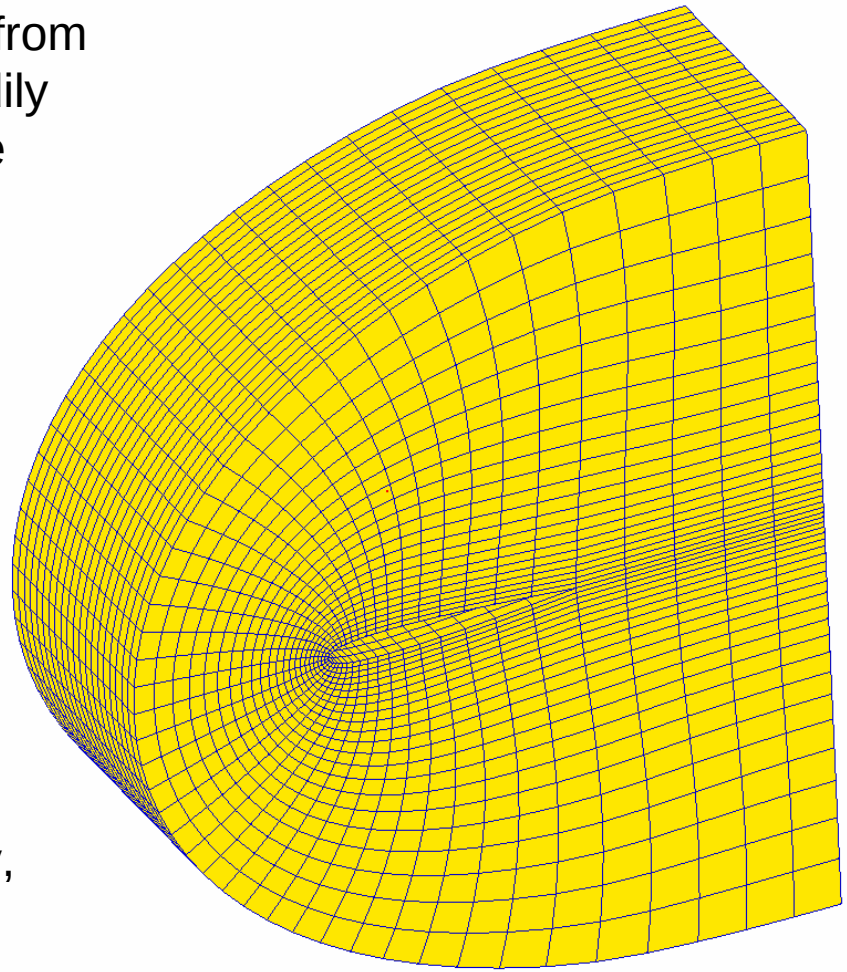


Building 3D Structured Extruded Grids

Just as 2D domains can be easily extruded from connectors or edges, 3D blocks can be readily extruded from domains or faces. Indeed, the techniques are basically the same, and the same marching schemes are available:

- Hyperbolic.
- Normal.
- Rotational.
- Translational.
- Path.

Building on our 2D extrusion experience, let's continue onto building 3D structured blocks. As before, we'll skip the initial steps to select analysis software, import geometry, create connectors, and assemble domains.



The “Bottom-Up”Gridgen Process

Select Analysis Software and Mesh Dimensionality

⇩

Import and Manipulate Database Geometry

⇩

Create Segments and Connectors

⇩

Dimension Connectors and Distribute Grid Points

⇩

Create Domains

⇩

Extrude Blocks

⇩

Setup CFD Analysis Boundaries

⇩

Export and Save Files

Extrude Blocks

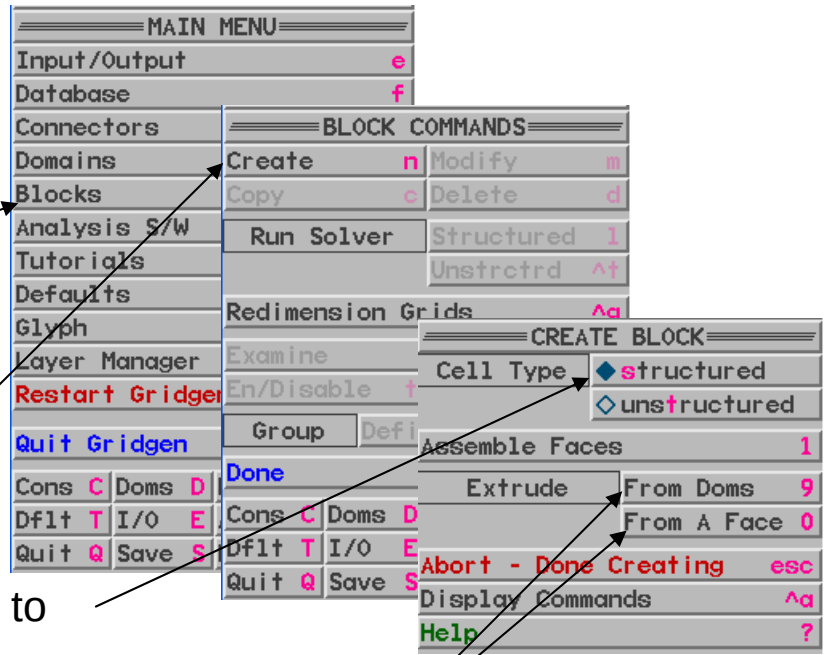
With one or more domains in place, blocks can be extruded from either the individual domains or a face formed by combining them.

From the **MAIN MENU**, select **Blocks** to open the **BLOCK COMMANDS** submenu.

From here, select **Create** to open **CREATE BLOCK**.

Be sure that the **Cell Type** is set to **structured**.

Next, it's time to select either **From Doms** (domains) or **From A Face** to choose which will be the basis of the extrusion.

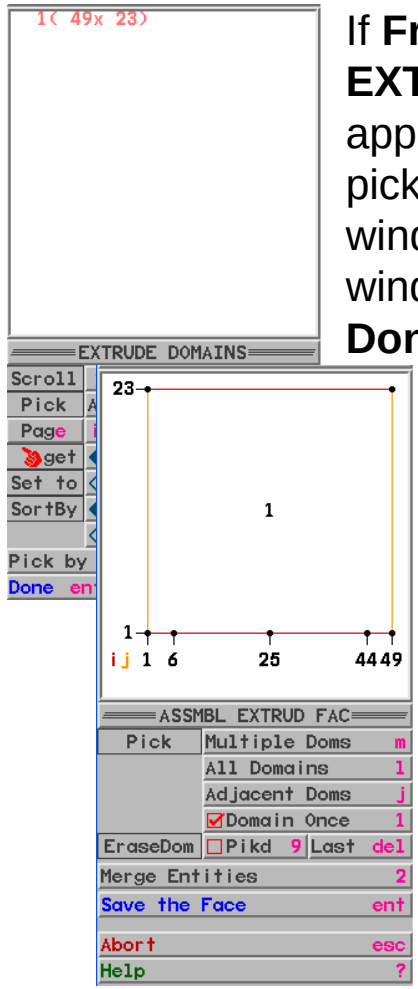


Doms vs. Faces

From Doms – A separate extrusion is performed on each individual domain selected, thus allowing each to have it's own extrusion attributes. A floating boundary condition is applied at shared connectors so adjacent extrusions blend smoothly. Each domain extrudes to a separate block.

From A Face – The selected domains form a single face from which a single new block is extruded.

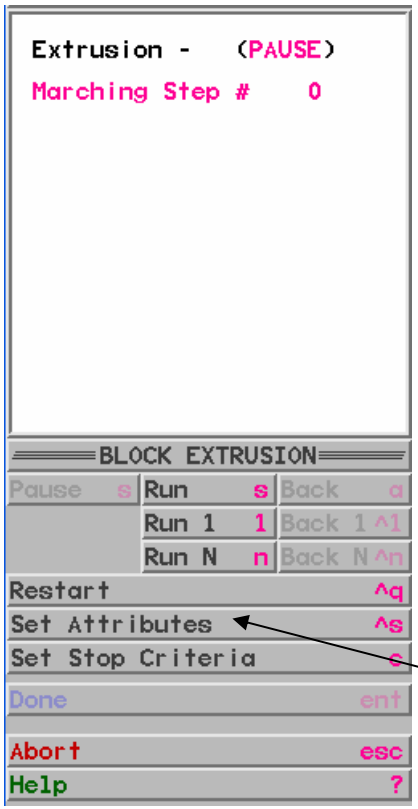
As might be expected from the case of domain extrusion, **From A Face** is preferred for mesh quality and extrusion algorithm numerical stability reasons.



If **From Doms** was selected, the **EXTRUDE DOMAINS** submenu appears. Domains can then be picked by mouse from the browser window or directly from the display window. When picking is finished, **Done** is clicked.

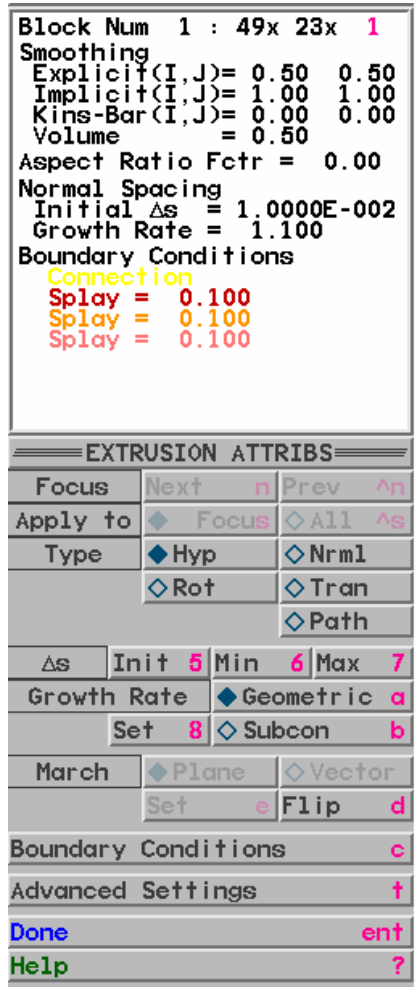
If **From An Face** was selected, the **ASSMBL EXTRUD FAC** submenu appears. The face is formed in the same manner as with the **Assemble Faces** command. Once complete, **Save the Face** is selected to finish the process.

Extrusion Attributes

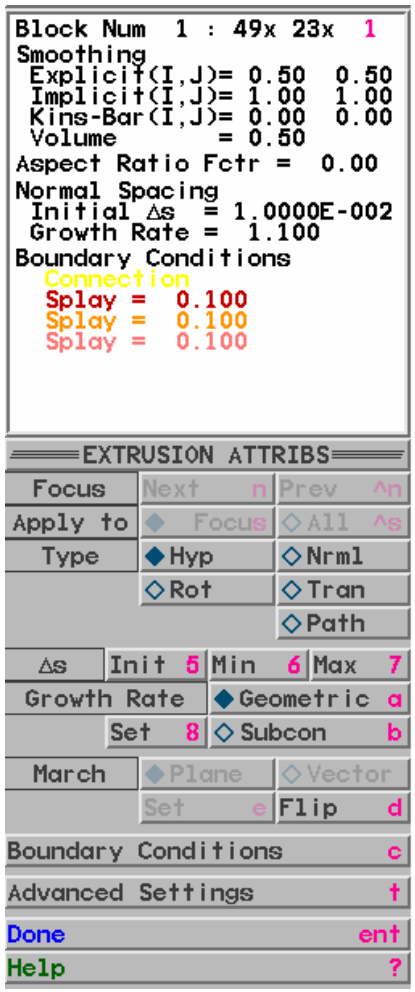


When the domain(s) or face for the extrusion is defined, the **BLOCK EXTRUSION** menu appears, but before marching through the extrusion, there are several attributes for the process that need to be set or, at least, reviewed.

Selecting **Set Attributes** brings up the **EXTRUSION ATTRIBS** submenu, with the blackboard window now showing the current values for most of the attributes.



Extrusion Types



As with domain extrusion, the first attribute that needs to be selected is the **Type** as this will determine the basic character of the block to be extruded and bring up the appropriate options defining the extrusion:

Hyperbolic (Hyp) – Based on the solution of a set of PDEs prescribing a orthogonal trajectory for the grid lines from the initial grid entity.

Normal (Nrml) – An algebraic method that mimics the hyperbolic technique. The hyperbolic method is preferred.

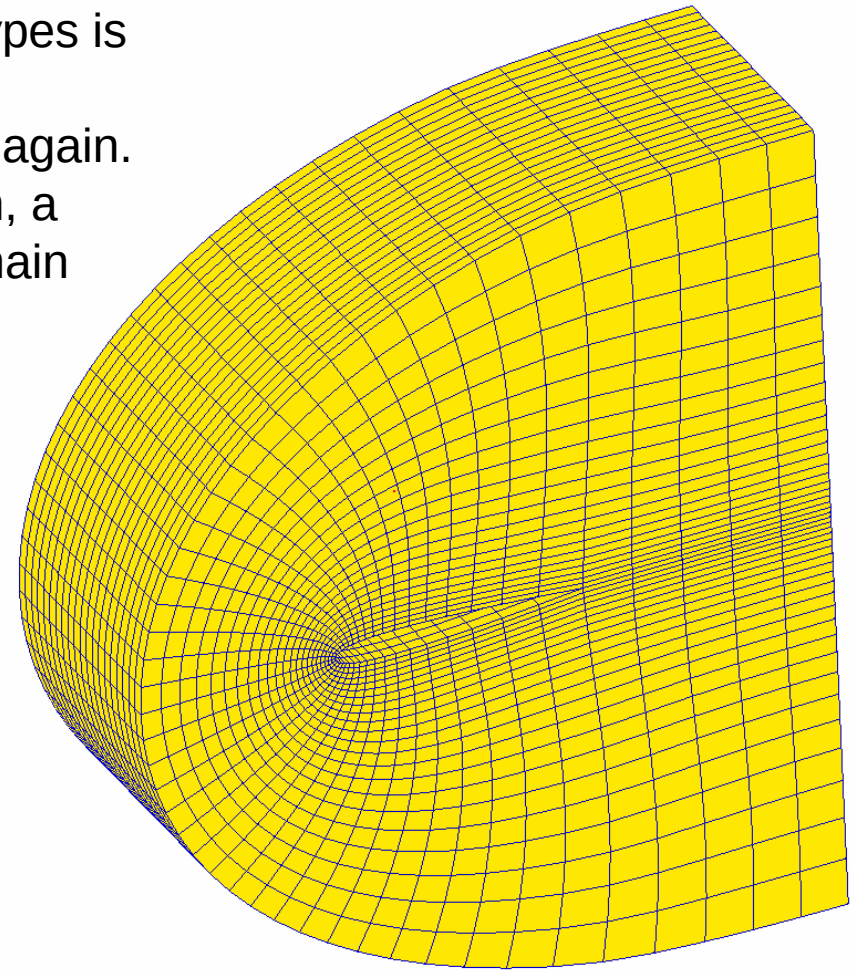
Rotational (Rot) – An algebraic method that sweeps grid in a circular path about a prescribed axis.

Translational (Tran) – An algebraic method that sweeps grid in a linear path in a prescribed direction.

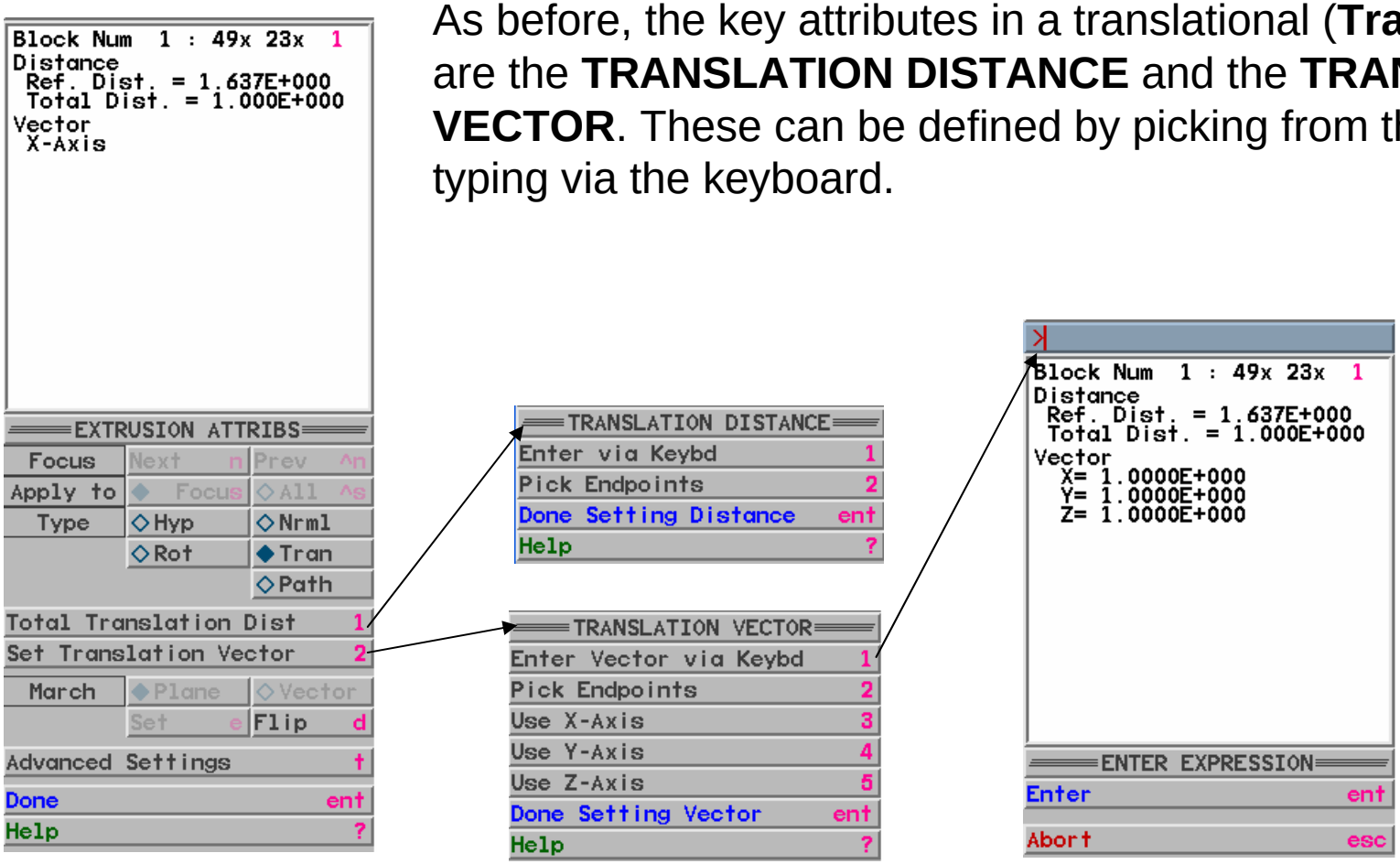
Path – An algebraic method that sweeps grid along a connector defined path.

Tran Extrusion Example

Since the setup for the various extrusion types is the same for 2D and 3D cases, the corresponding menus will not be reviewed again. However, to show the 3D process in action, a translational extrusion of the 2D airfoil domain will be performed.



Tran Extrusion Attributes



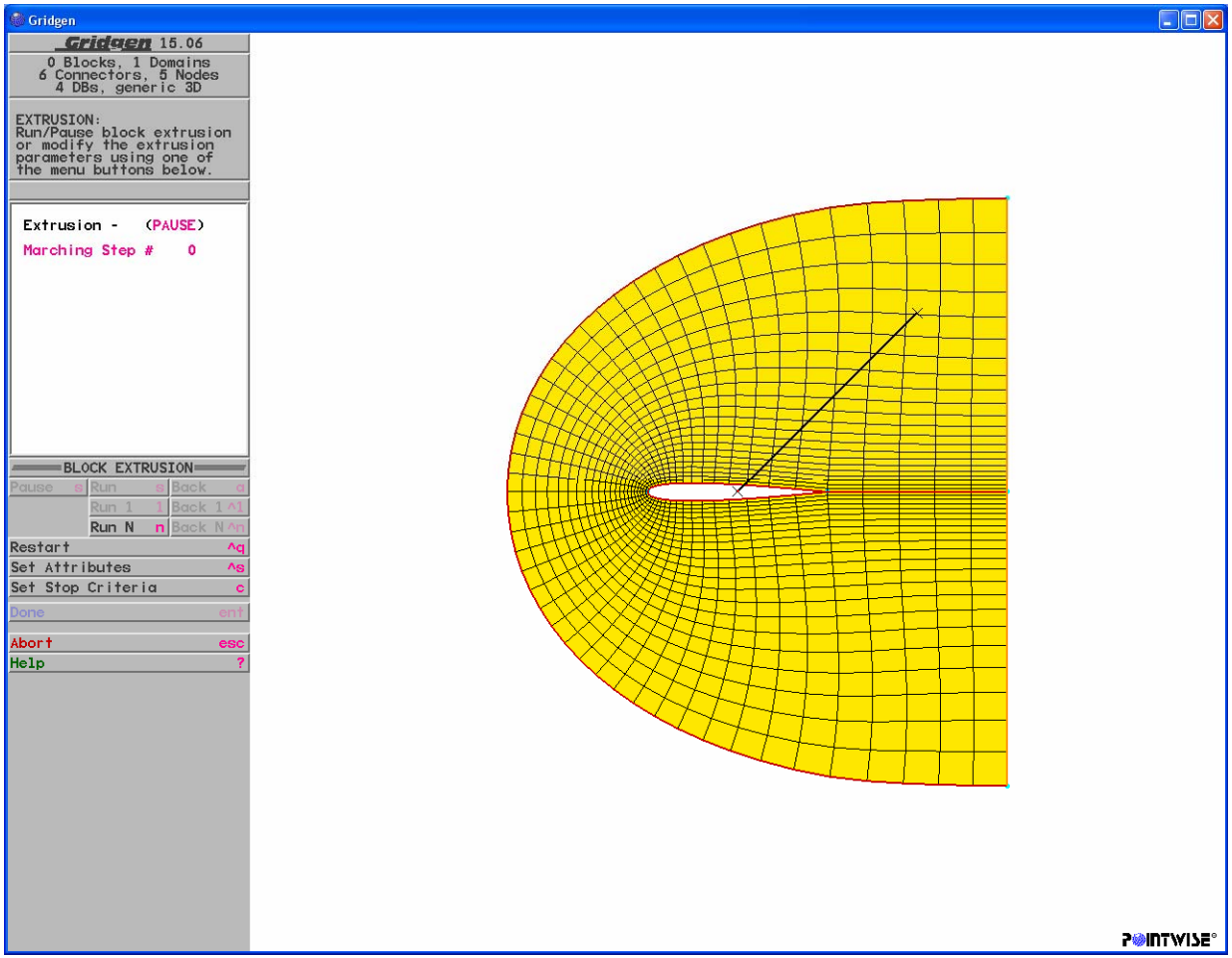
As before, the key attributes in a translational (**Tran**) extrusion are the **TRANSLATION DISTANCE** and the **TRANSLATION VECTOR**. These can be defined by picking from the display or typing via the keyboard.

In the present example, the vector is 1, 1, 1 and the distance is 1.

Tran Extrusion

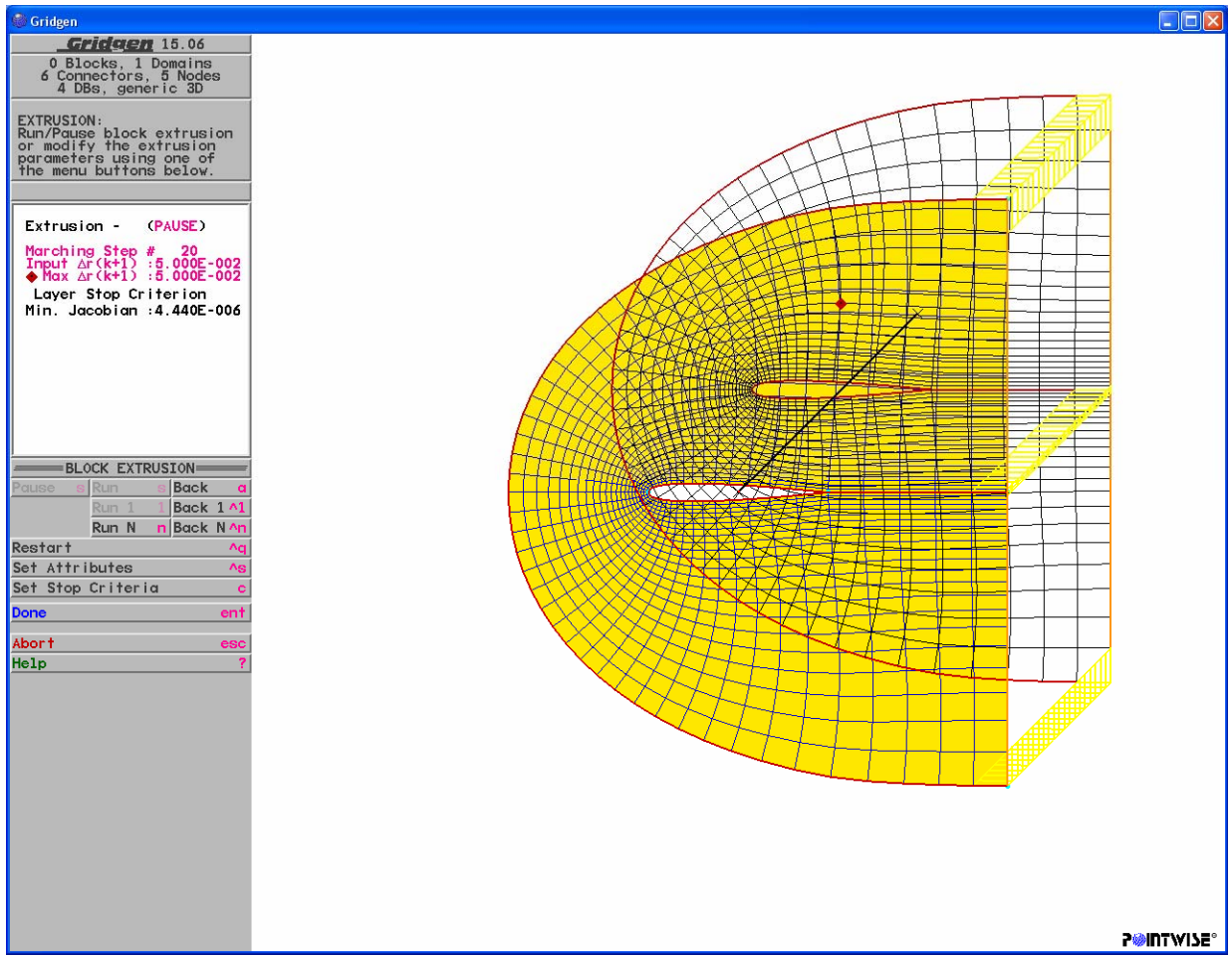
With the key attributes set, all that remains is to define the number of steps that will taken to march the specified distance. Once **Run N** is clicked, this can be entered via the keyboard. In this case, 20 will be input.

Note that the translation vector defined for the extrusion is shown in the display window.



Tran Extrusion (Continued)

After the steps are entered and the **Enter** button is clicked, this is the result. If this is not the desired outcome, the extrusion can be reversed to the beginning (**Back**), by one step (**Back 1**), or by N step (**Back N**). If the new block is satisfactory, **Done** can be selected.

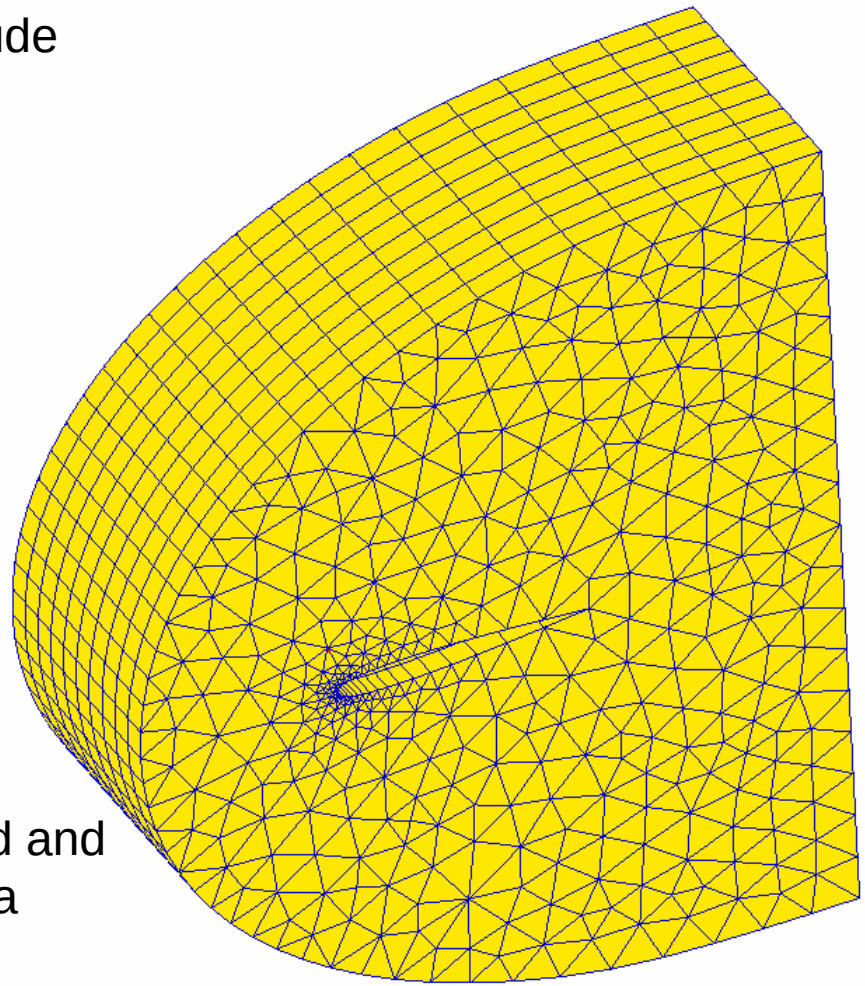


Building 3D Unstructured Extruded Grids

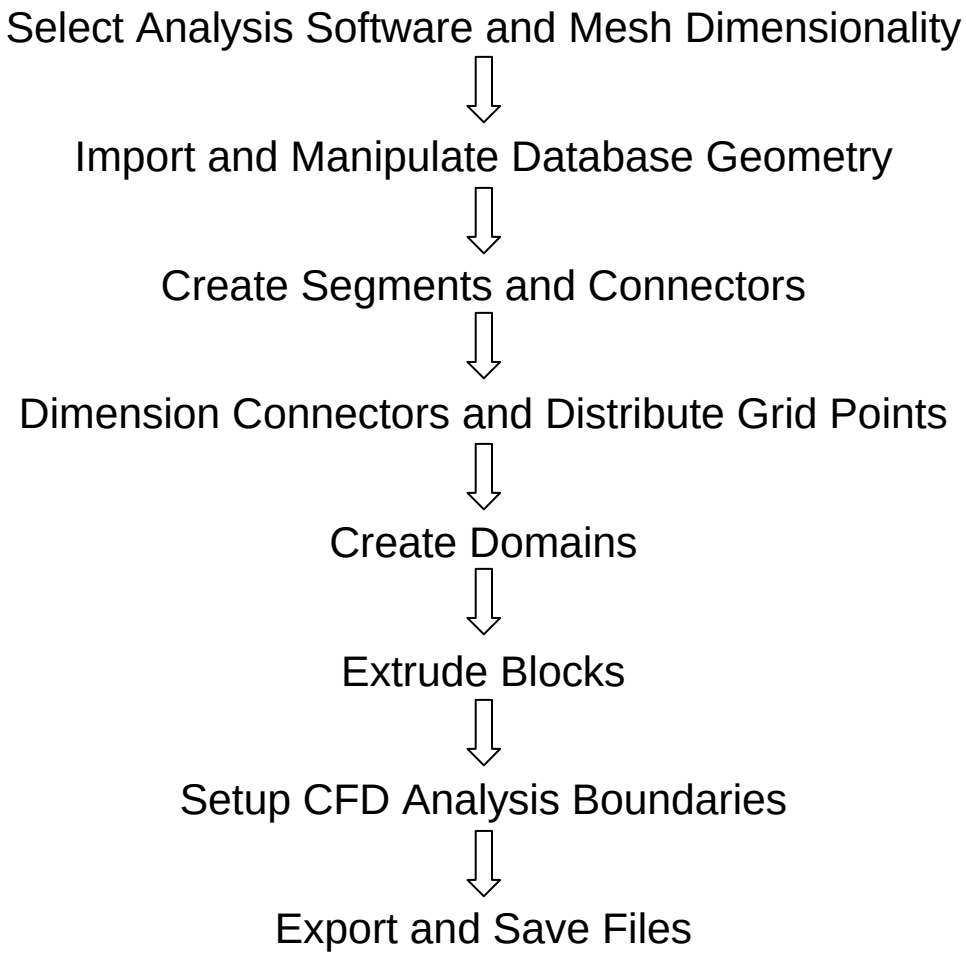
As might be imagined, it's not possible to extrude a 2D unstructured domain from a connector or edge, which leaves extruding 3D prism blocks from unstructured domains and faces. Many aspects of this process are identical to those already discussed with respect to structured blocks, so won't be repeated here, but there are some differences. For example, only four marching schemes are available:

- Normal.
- Rotational.
- Translational.
- Path.

As we'll see, it's also possible to mix structured and unstructured domains as the starting point for a combined or combo extrusion.



The “Bottom-Up”Gridgen Process



Extrude Blocks

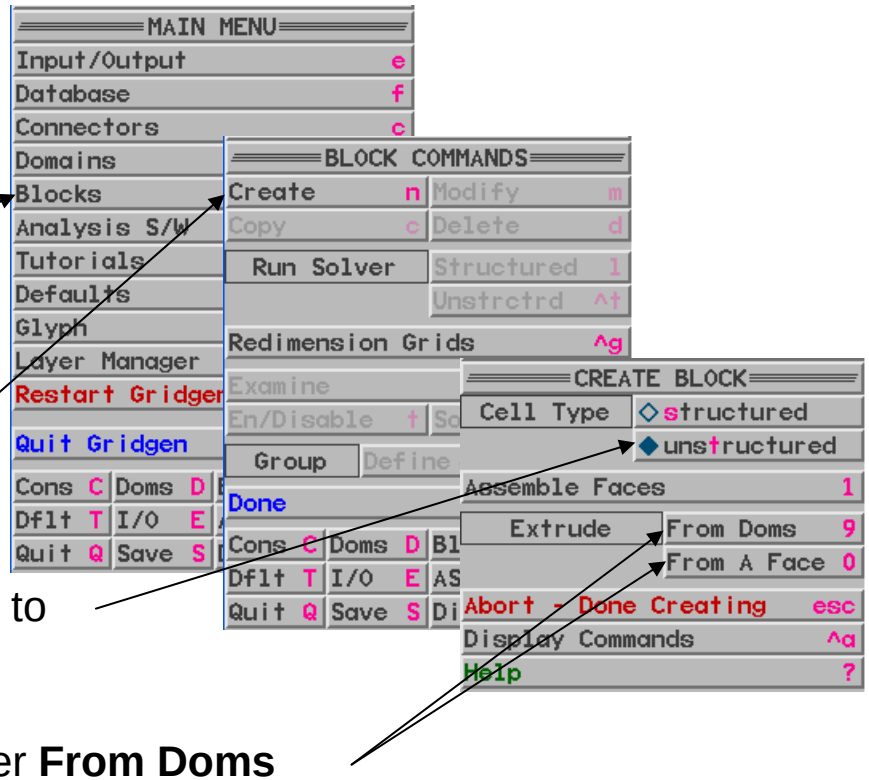
With one or more domains in place, blocks can be extruded from either the individual domains or a face formed by combining them.

From the **MAIN MENU**, select **Blocks** to open the **BLOCK COMMANDS** submenu.

From here, select **Create** to open **CREATE BLOCK**.

Be sure that the **Cell Type** is set to **unstructured**.

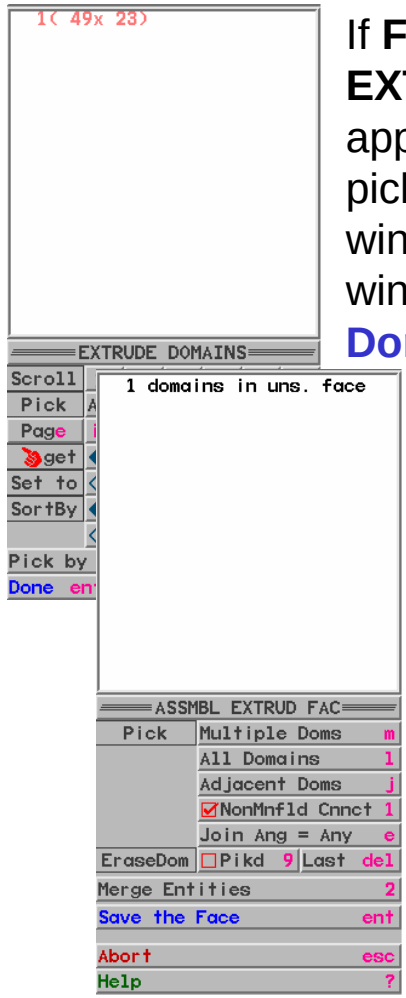
Next, it's time to select either **From Doms** (domains) or **From A Face** to choose which will be the basis of the extrusion.



Doms vs. Faces

From Doms – For unstructured domains that do not share any edges, separate extrusions are performed on each individual domain selected, thus allowing each to have it's own extrusion type. However, connected unstructured domains will auto-matically be combined into a single face. For domains that do remain separate, separate blocks are generated. Of course, this is the only choice for combo extrusions, i.e., from a mix of structured and unstructured domains.

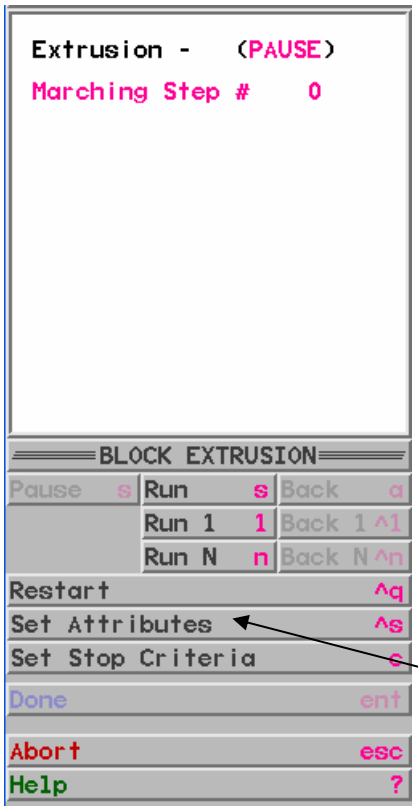
From A Face – The selected domains form a single face from which a single new block is extruded.



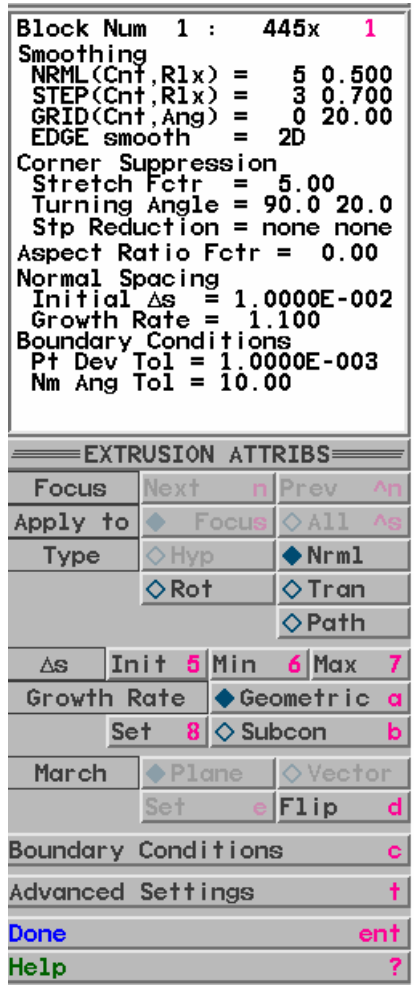
If **From Doms** was selected, the **EXTRUDE DOMAINS** submenu appears. Domains can then be picked by mouse from the browser window or directly from the display window. When picking is finished, **Done** is clicked.

If **From A Face** was selected, the **ASSMBL EXTRUD FAC** submenu appears. The face is formed in the same manner as with the **Assemble Faces** command. Once complete, **Save the Face** is selected to finish the process.

Extrusion Attributes

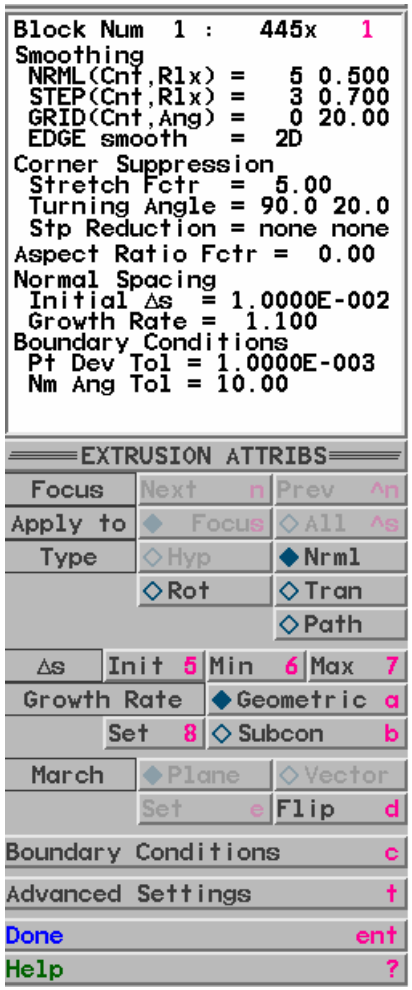


When the domain(s) or face for the extrusion is defined, the **BLOCK EXTRUSION** menu appears, but before marching through the extrusion, there are several attributes for the process that need to be set or, at least, reviewed.



Selecting **Set Attributes** brings up the **EXTRUSION ATTRIBS** submenu, with the blackboard window now showing the current values for most of the attributes.

Extrusion Types



As with structured block extrusions, the first attribute that needs to be selected is the **Type** as this will determine the basic character of the block to be extruded and bring up the appropriate options defining the extrusion:

Hyperbolic (Hyp) – Not available for unstructured block extrusion.

Normal (Nrml) – An algebraic method that mimics the hyperbolic technique.

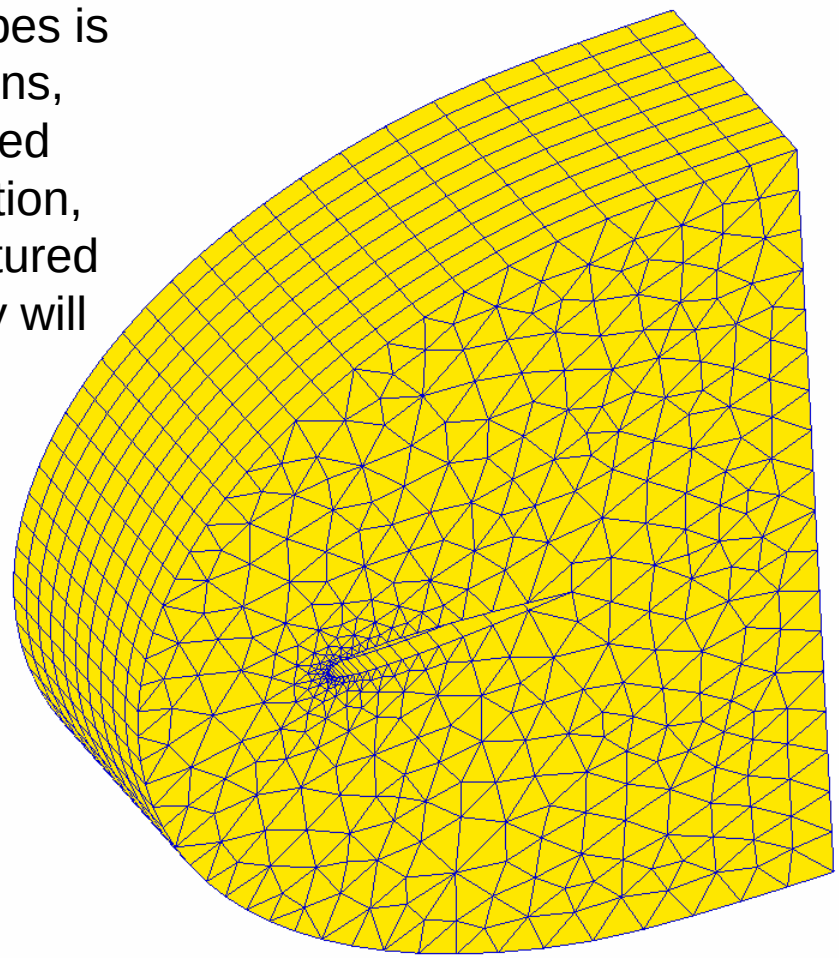
Rotational (Rot) – An algebraic method that sweeps grid in a circular path about a prescribed axis. Note that axis cannot lie on an edge of the initial front as this will result in degenerate prism cells along edge.

Translational (Tran) – An algebraic method that sweeps grid in a linear path in a prescribed direction.

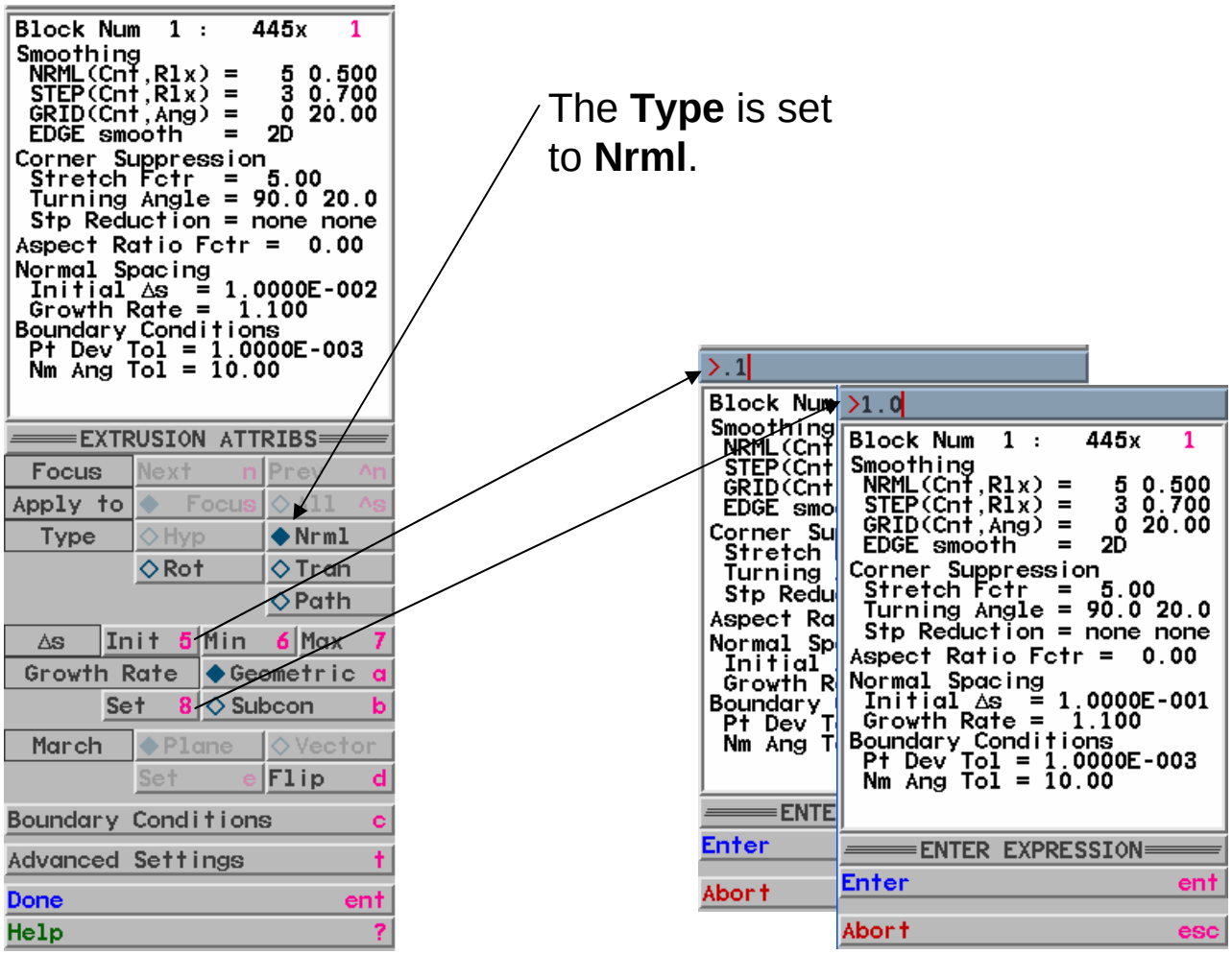
Path – An algebraic method that sweeps grid along a connector defined path.

Nrml Extrusion Example

Since the setup for the various extrusion types is the same as with the 3D structured extrusions, the corresponding menus will not be reviewed again. However, to show the process in action, a normal extrusion starting from an unstructured 2D domain built around the airfoil geometry will be performed.



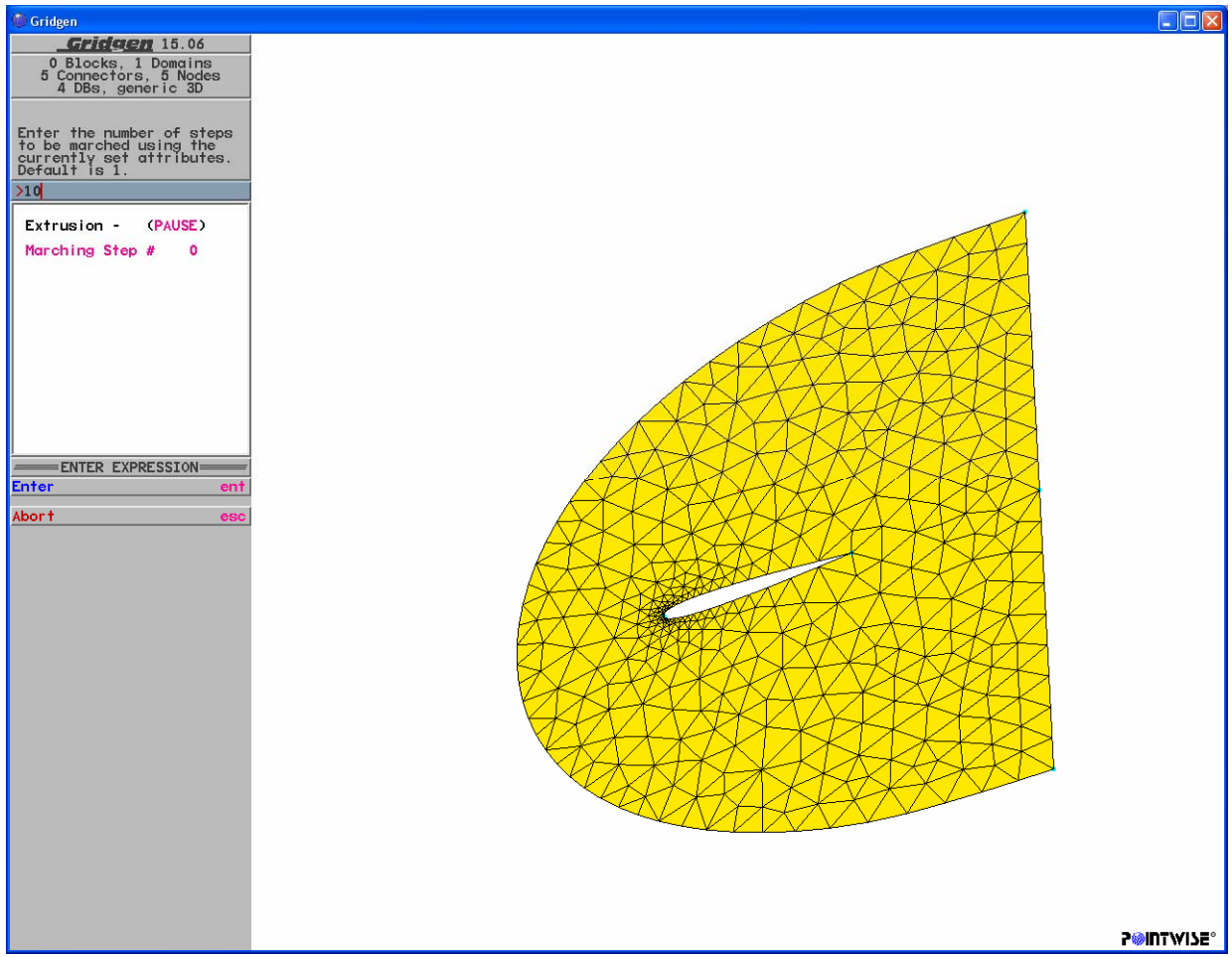
Nrml Extrusion Attributes



In this particular case, we want a uniform marching distance, so the initial **Δs** will be set to the desired step size (0.1) and the **Growth Rate** will be set to 1. Once the values are entered via the keyboard, **Enter** is clicked to return to the **EXTRUSION ATTRIBS** menu. Note that the blackboard window shows the updated values for the changed attributes.

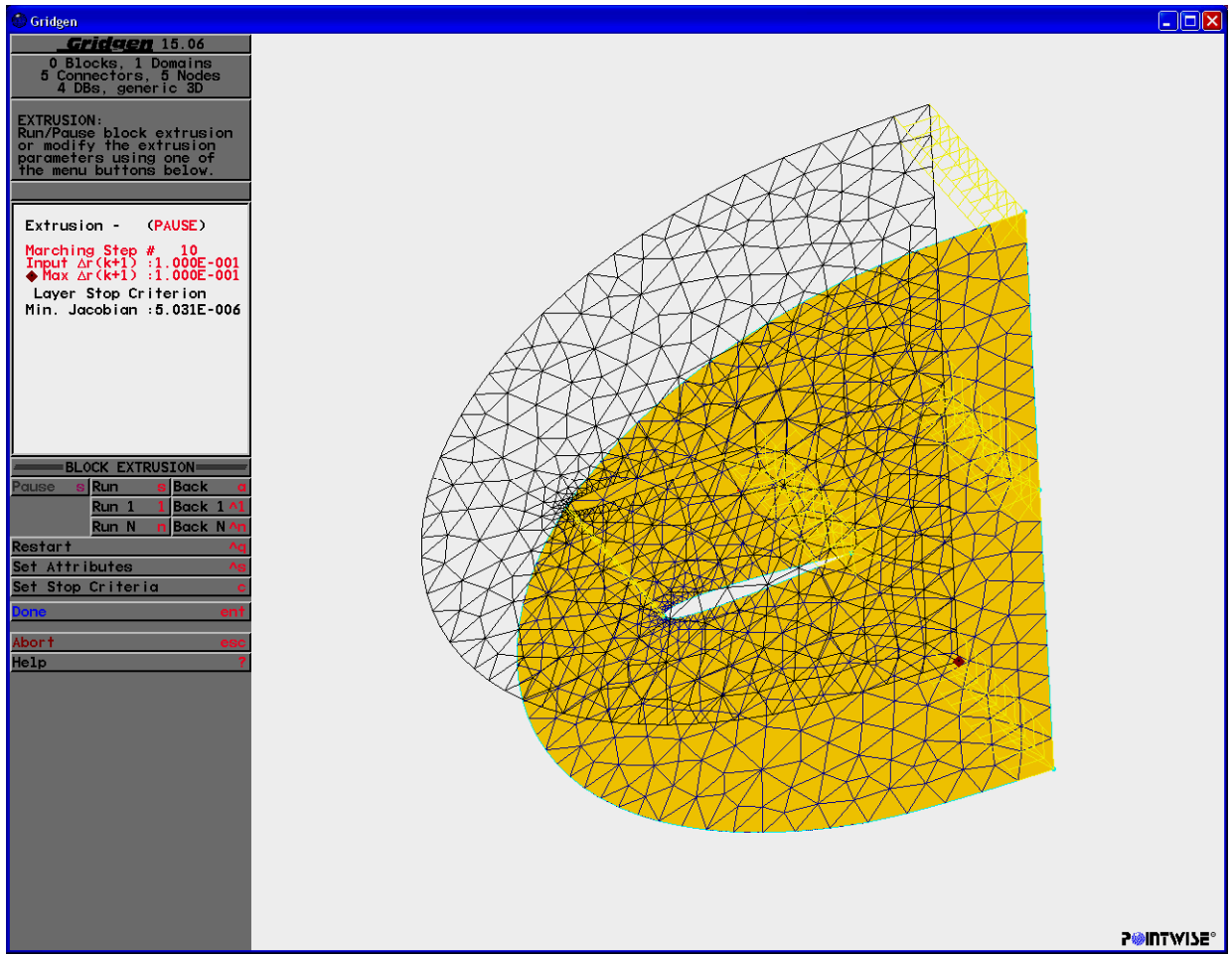
Nrml Extrusion

With the key attributes set, all that remains is to define the number of steps that will taken to march the specified distance. Once **Run N** is clicked, this can be input via the keyboard. In this case, 10 will be entered.



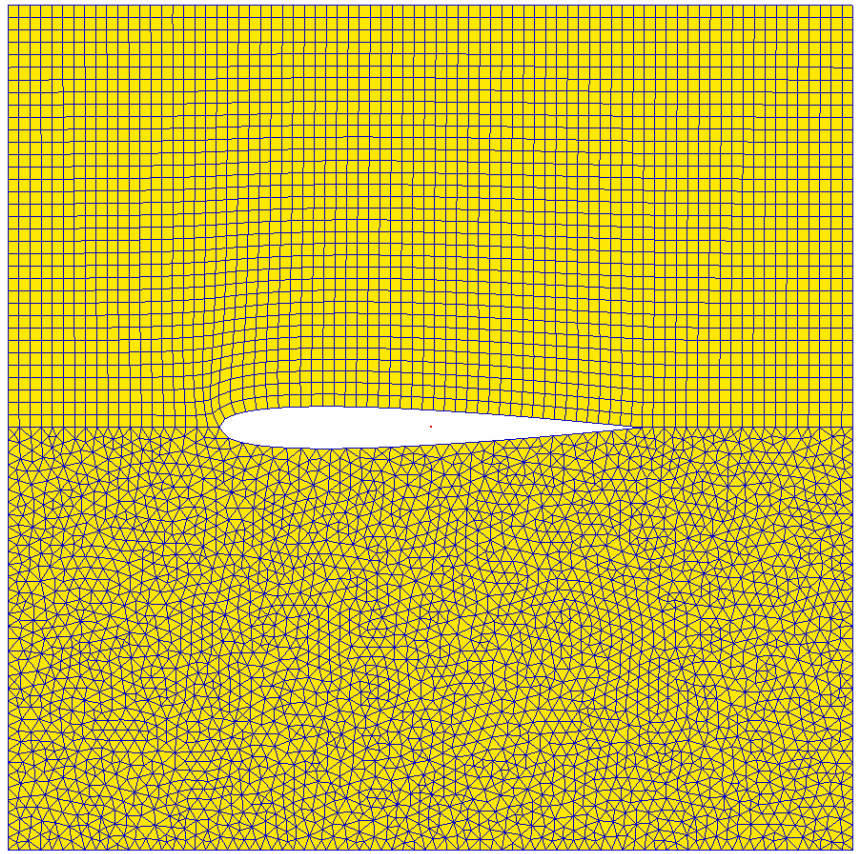
Nrml Extrusion (Continued)

After the steps are entered and the **Enter** button is clicked, this is the result. If this is not the desired outcome, the extrusion can be reversed to the beginning (**Back**), by one step (**Back 1**), or by N step (**Back N**). If the new block is satisfactory, **Done** can be selected.



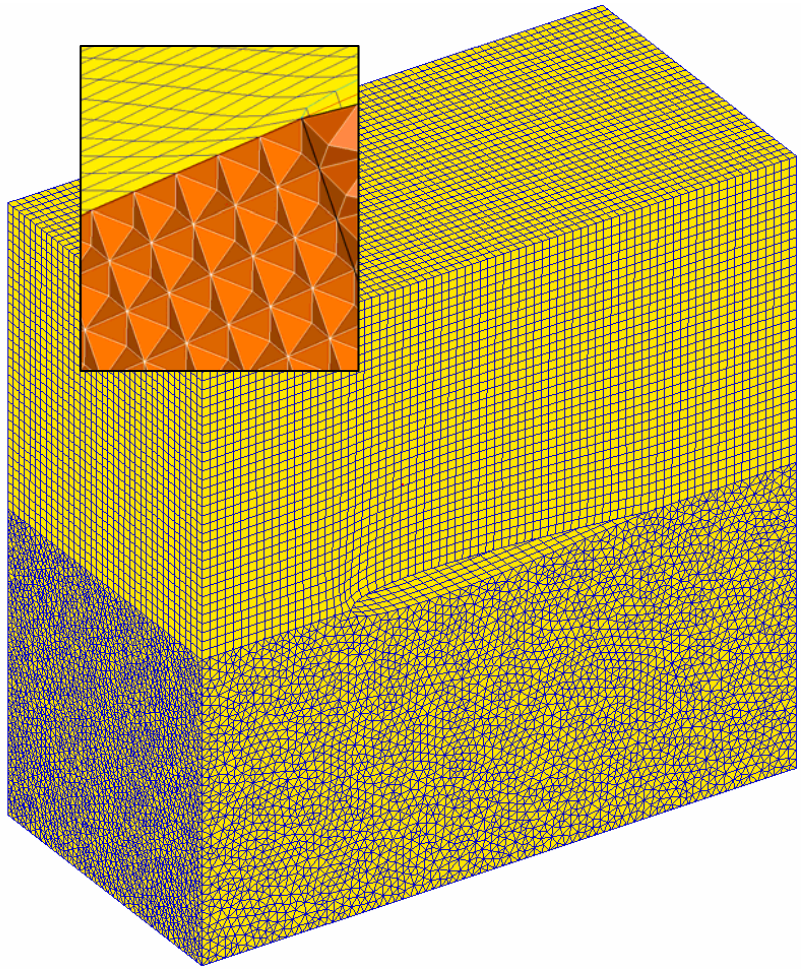
Creating Hybrid Grids

Gridgen supports construction and export of hybrid grids – grids which contain a mixture of cell types. Cell types currently supported by Gridgen include quadrilaterals, triangles, hexahedra, tetrahedra, prisms, and pyramids. With the exception of pyramids, grid construction with all of these cell types has already been discussed, so the tools needed to generate hybrid grids are already known for the most part. Indeed, a 2D example, the airfoil mesh here, has already been considered. Other 2D hybrid grids could be built in a similar fashion.

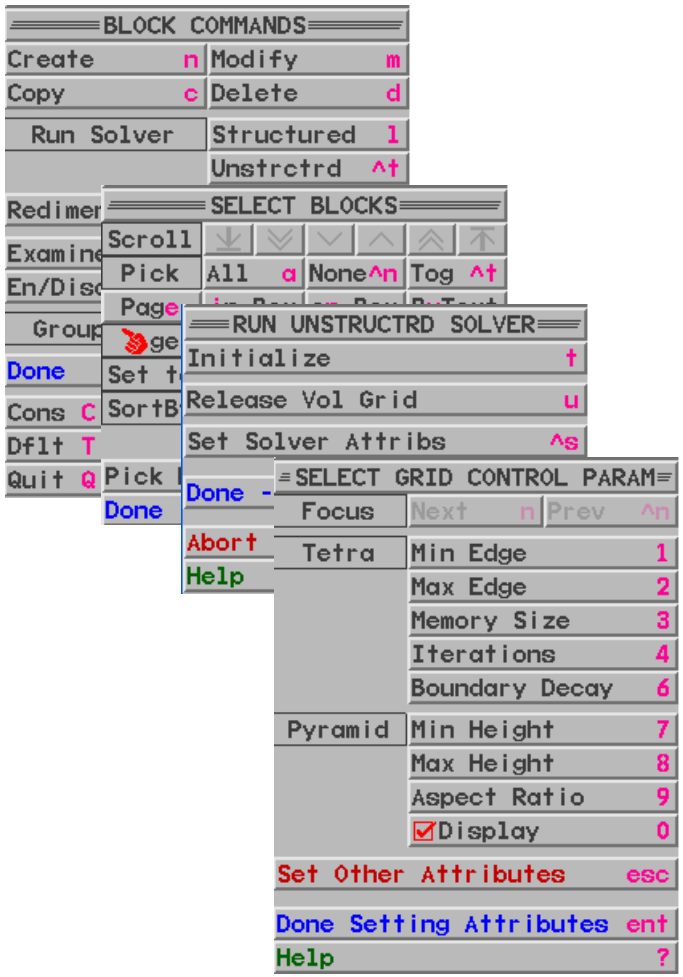


Creating Hybrid Grids (Continued)

Similarly, an example of a 3D hybrid grid has also been built. Interestingly, the unstructured block here contains two types of cells, tetrahedra and pyramids. The pyramids were created automatically by the unstructured solver along the upper surface of the unstructured block to interface the tets below with the structured domains bounding that surface. This is the only case where Gridgen creates blocks with mixed cell types, although a structured block may contain degenerate hexes when the block contains a singularity. The inset shows the pyramids in the first layer of cells in the unstructured block in a view looking up from below – the wing leading edge is on the right.



Pyramids and the Unstructured Solver



The characteristics of the pyramids generated by the unstructured solver can be controlled just like those of the tetrahedra. The controls are located in the **SELECT GRID CONTROL PARAM** submenu and include:

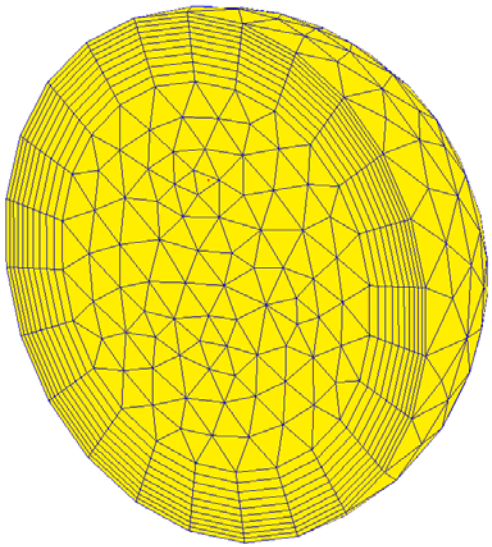
Min Height setting the minimum allowable pyramid height.

Max Height setting the maximum allowable height.

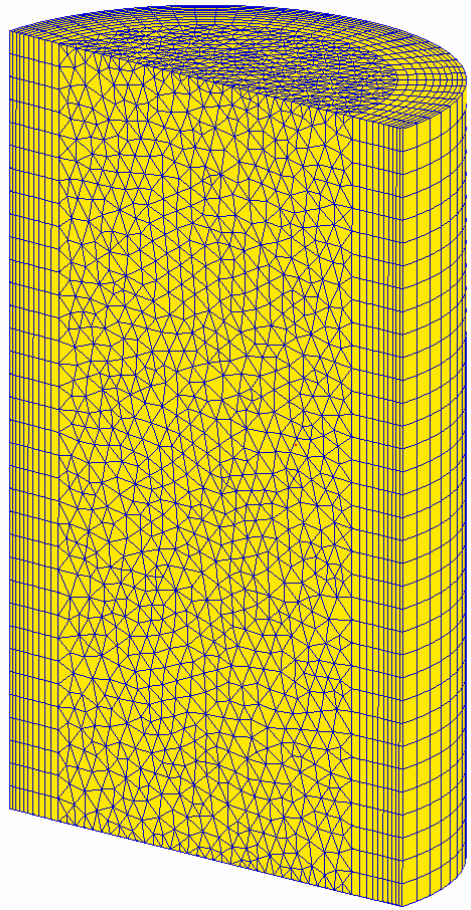
Aspect Ratio specifying the aspect ratio used to scale nominal pyramid height to give the true pyramid height (default is 0.5).

Display toggling whether the pyramids will be displayed while within the solver.

Structured Viscous Layer Hybrid Grids

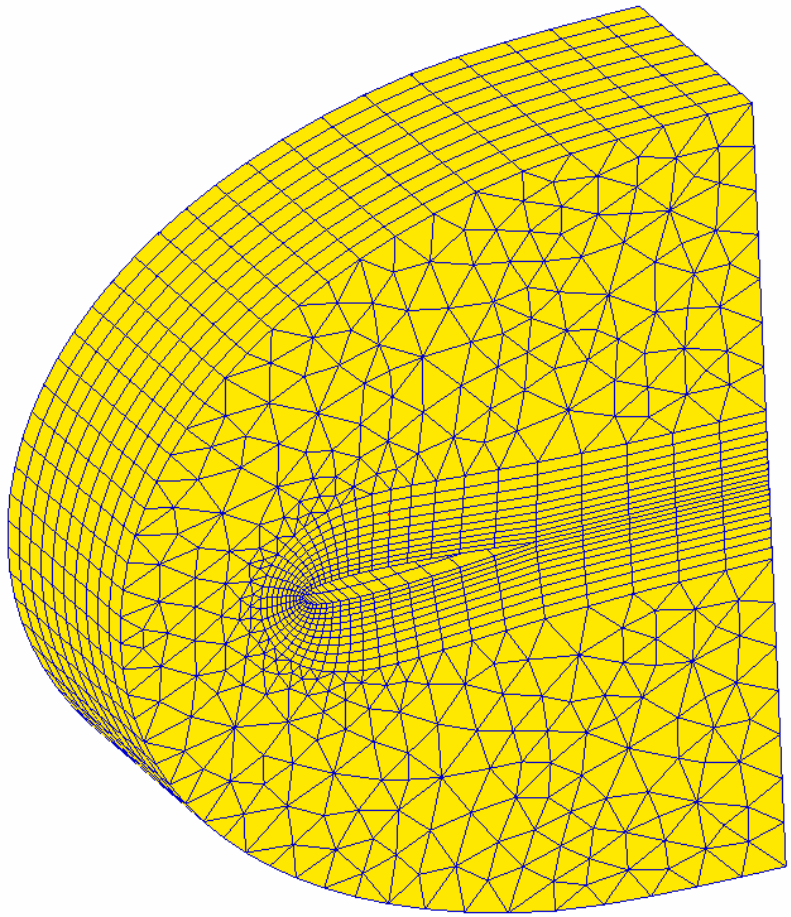


Most hybrid grids are constructed for viscous analyses. Generally they are created with either a prism (left) or hex (right) viscous layer, both of which provide proper grid structure and growth through the thickness of the layer. The remainder of the volume is usually filled with tets.



Combo Extrusion

It’s also possible to combine the extrusion of structured and unstructured domains into a hybrid grid. The extrusion has to be performed **From Doms** (domains), and then the remaining attributes for each domain are individually set. Both the structured and unstructured blocks can then be extruded simultaneously. Here is an example of a translational combo extrusion of the airfoil geometry.

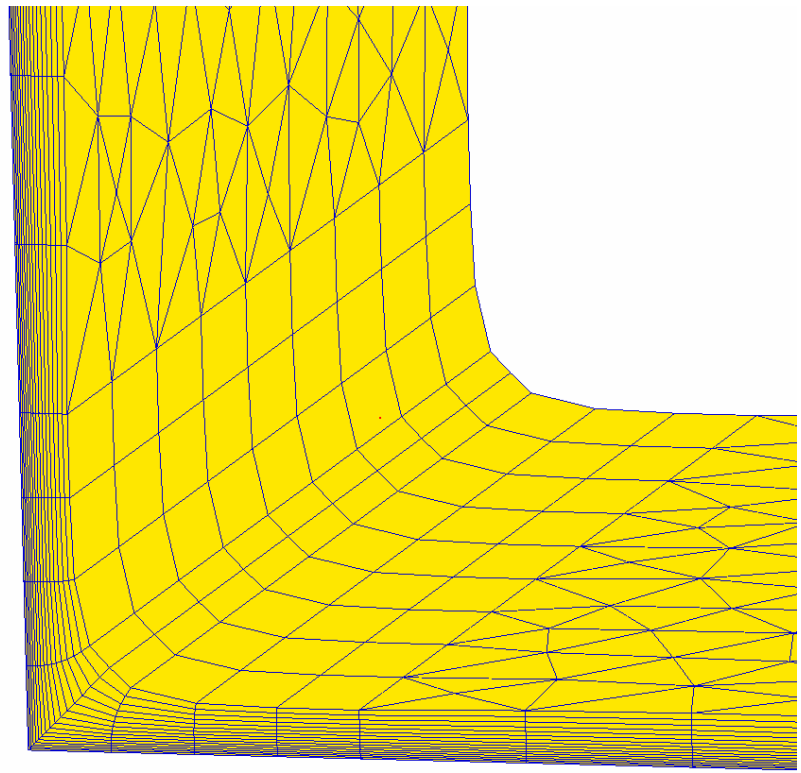


Mixing Layer Types

Sometimes it may be desirable to have mixed layer types. In other words, have prism layers on some or most of the geometry surface and fill in with hex layer blocks. This type of construction is suitable for:

- Cases where certain parts of the geometry cannot be easily extruded.
- Cases where certain parts of the geometry require the resolution or distribution control of a structured block.

In these instances, combo extrusion can often be effectively employed (as at right). If not, the prism blocks are to be extruded first, and the structured blocks then assembled from the exposed prism layer quad faces and additional structured domains as required.



Gridgen**Training**

- In this session:
  - Building 2D Structured Extruded Grids.
  - Building 3D Structured Extruded Grids.
  - Tutorial: “Swept Ramp: Advanced Multi Block Structured Grid.”
  - Building 3D Unstructured Extruded Grids.
  - Creating Hybrid Grids.
  - Tutorial: “Droplet in a Cylinder: Hybrid Grid.”

- In this session:
  - Layer Manager.
  - Tutorial: “Cubes and Spheres: Layer Manager.”
  - Advanced Tools.
  - Tutorial: “B-747 Nacelle: Working with an IGES Database.”

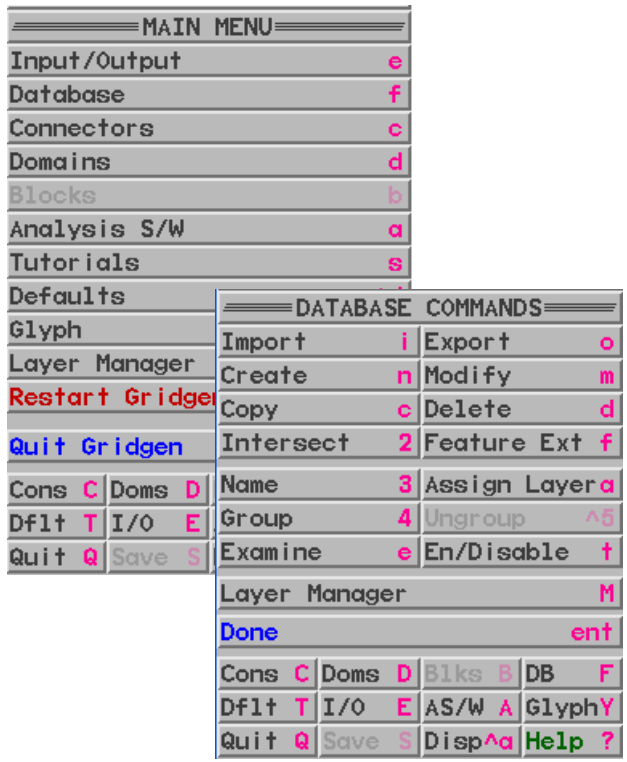
The Layer Manager

Gridgen's layer manager provides flexible control over what portions of a database model are displayed. It's very similar in style to that of common CAD software packages. Indeed, the layer manager preserves layer information from the original CAD models when they are imported into Gridgen

Each database entity can be assigned to only one layer (there are 1024 total layers available). Each layer can be likened to a transparency overlay, the user choosing which layers are visible at any given time in the Display Window.

Once defined, the layer information is saved in the composite database file (.dba) for future editing sessions.

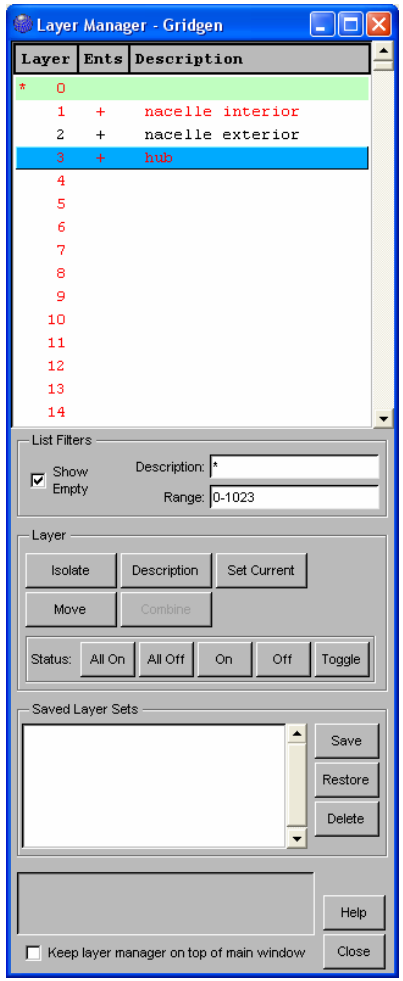
The layer manager is accessed by clicking on the **Layer Manager** button in the **MAIN MENU** or the **DATABASE COMMANDS** menu (or via the **M** hot key).



The Layer Manager Panel

The layer manager panel is a separate desktop window. The **M** hot key can be used to pop it to the top of all other desktop windows at any time, even when the panel has already been opened. The box on the bottom of the window can be toggled to always keep it on top of the main Gridgen window.

The panel has four logical sections: The Layer Browser List, List Filters, Layer Operations, and Saved Layer Sets.



The Layer Browser List

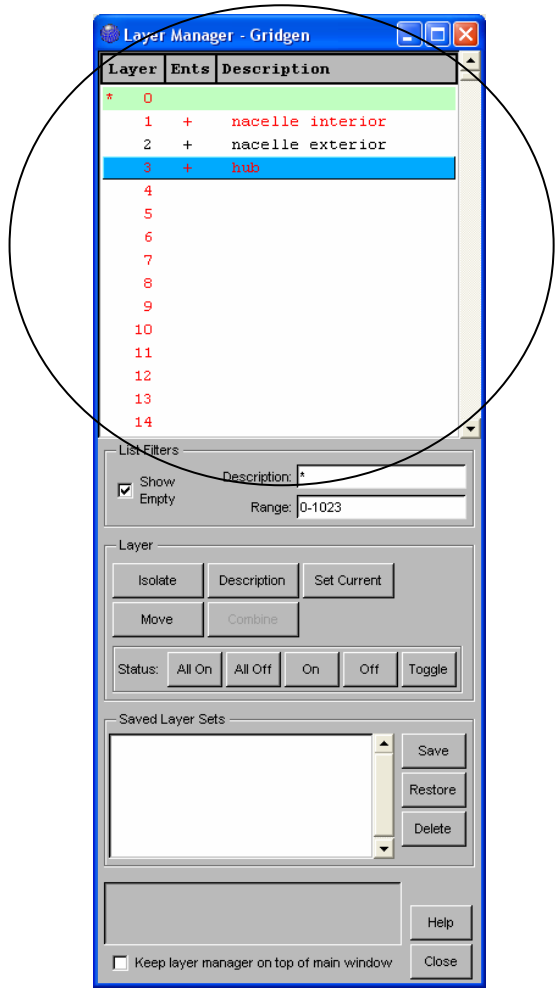
The Layer Browser List is used to select individual or sets of layers to operate on. The list has three columns:

Layer lists the layer numbers (0 – 1023), with * indicating the current working layer where, for example, newly created database entities will be placed.

Ents contains **+** or **–** symbols indicating whether the layer contains enabled or disabled, respectively, database entities (no symbol indicates a layer is empty).

Description contains each layer's user-supplied name.

Text color indicates layer status: layers in red are currently on and their entities rendered in the display (if enabled), while layers in black are currently off and their entities aren't rendered (even if enabled). Layers highlighted in blue are currently selected for layer operations. In addition to the *, the current working layer is also highlighted in green.



List Filters

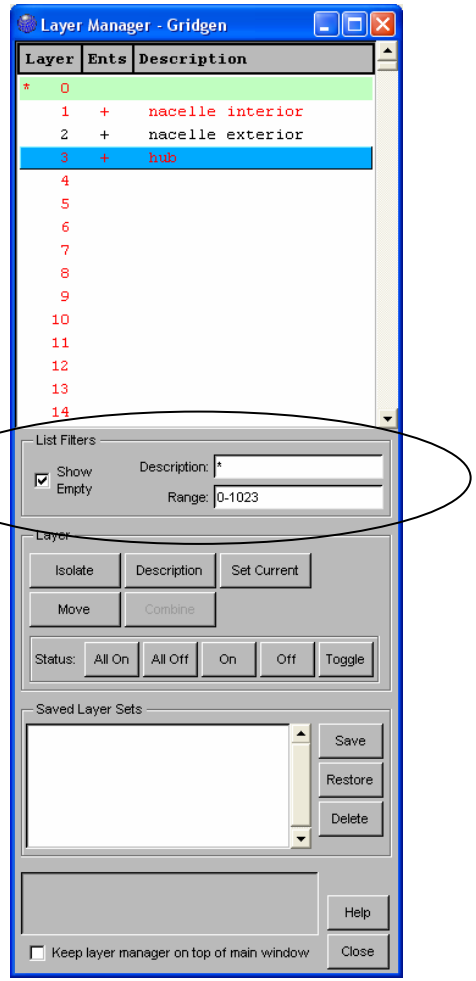
The Layer Filters are used to manage which layers will be shown in the Layer Browser List. They are, in order of precedence:

Show Empty – if toggled (the default), the browser list will show all layers, including those containing no entities. If off, empty layers will not be listed.

Description – Layers whose descriptions do not include the text string entered in the adjacent text entry field (including wildcards, *) are removed from the browser list.

Range – Layers whose layer numbers are not in the list entered in the adjacent text entry field are removed from the browser list. The list can contain no spaces with non-sequential numbers separated by commas and ranges of numbers separated by dashes.

Remember that the current working layer is always shown, regardless of any filtering applied.



Layer Operations

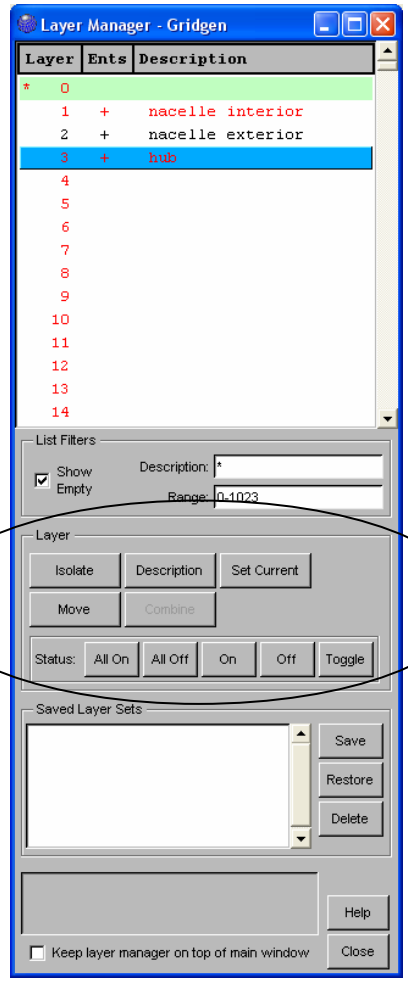
These functions operate on the layer(s) selected in the Layer Browser List:

Isolate sets the selected layer (it's only available if a single layer is picked) to the current working layer and turns off all others.

Description opens a dialog panel where an alpha-numeric description can be typed which will be applied to the layer(s) selected.

Set Current sets the selected layer (only available when a single layer is picked) to the current working layer. All subsequently created database entities will be assigned to this layer (until another is made the current working layer).

Move opens a dialog panel where a target layer number can be entered. The selected layer will be moved to this layer. Options allow moved layers to take their descriptions and on/off status as indicated.



Layer Operations (Continued)

Combine opens a dialog panel where a target layer number can be entered. The entities in the selected layers will be combined into this layer.

The **Status** commands simply change the on/off state of the selected layer(s):

All On turns on display of all layers.

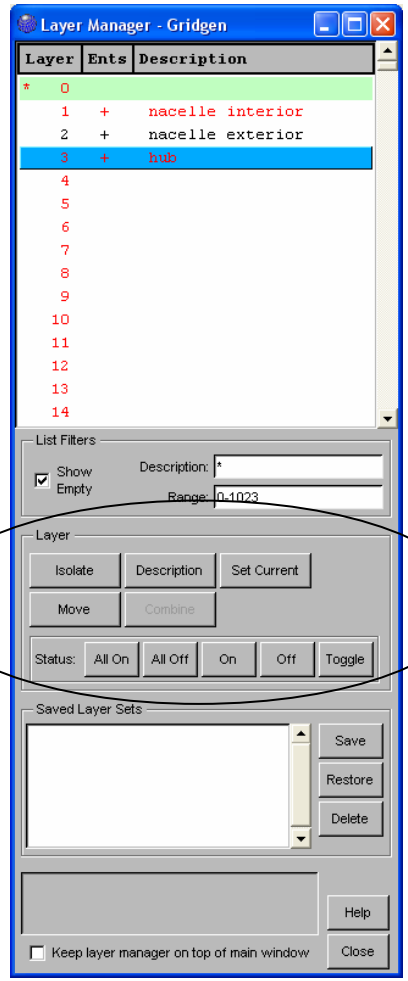
All Off turns off display of all layers except the current working layer.

On turns on the layers currently selected in the browser list.

Off turns off the layers currently selected in the browser list.

Toggle reverses the on/off state of all layers.

Right clicking on individual or multiple layers can access some of these operations.



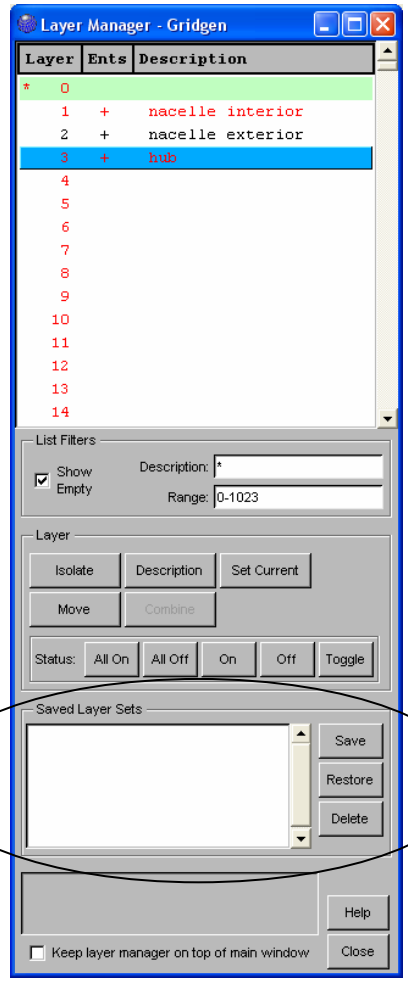
Saved Layer Sets

The Saved Layer Sets tools store and recall on/off states of all layers, as well as the designated current working layer. Saved layer sets do not preserve any filter settings.

Save opens a dialog panel where the name of the new saved layer set is entered.

Restore restores the saved layer set selected from the Saved Layer Sets browser.

Delete removes the selected saved layer set from the Saved Layer Sets Browser.

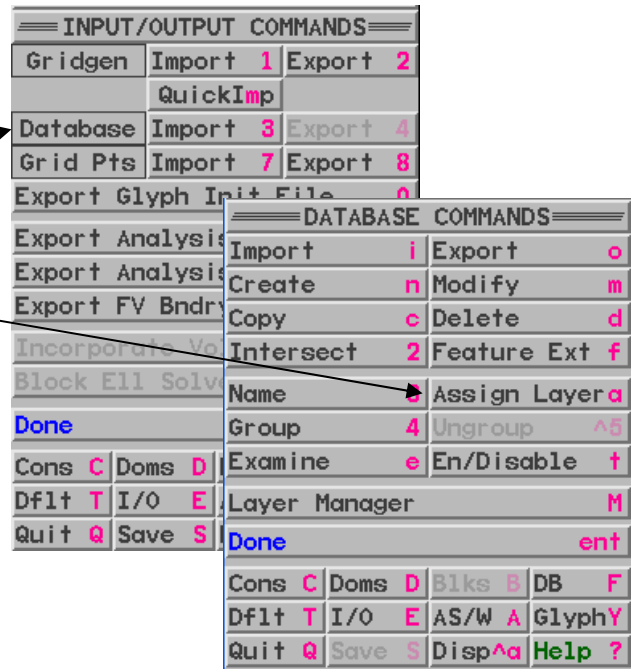


Layer Assignment

In addition to the **Move** and **Combine** commands previously discussed, there are two other ways that database entities can be assigned to layers:

If an IGES or native CAD file is imported via **Database Import** in the **INPUT/OUTPUT COMMANDS** menu, all level information in the file is transferred to Gridgen's layers.

Assign Layer in the **DATABASE COMMANDS** menu can also be used to manually reassign the layers to which entities belong. Once selected, the browser window changes to list all available database entities. This list only shows entities that can be assigned to a layer. Entities in layers whose status is off are not available for reassignment and are not shown. This list is also subject the enable/disable status of each entity in the currently displayed layer set – disabled entities are not available. Once the entities to be reassigned are chosen and **Done** is clicked, the browser window changes to show all 1024 layers by number, a + or – indicating their enabled or disabled status, and finally their layer descriptions. Selection from this list immediately assigns the chosen entities.



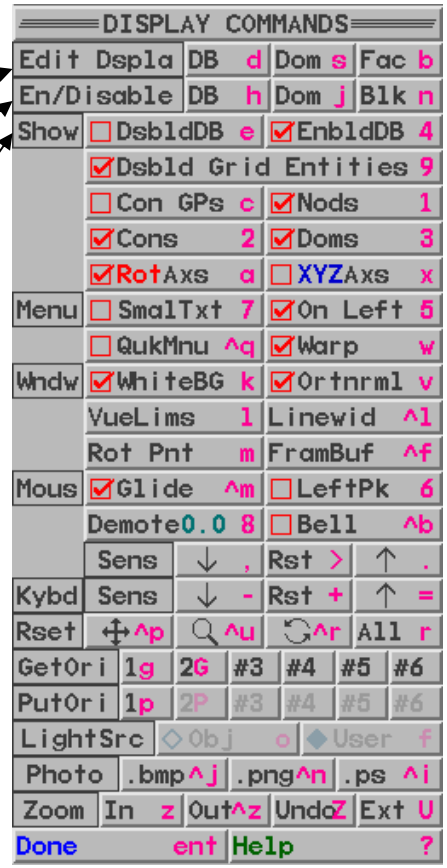
En/Disable

Another way that database information can be controlled in the GUI display is via the appropriate commands in the **DISPLAY COMMANDS** menu.

How individual entities are rendered can be selected via the **Edit Dspla DB** command.

The **En/Disable DB** command allows database entities to be temporarily “turned off,” minimizing how they are rendered and removing them from the pick list.

The **Show DsbldDB** and **EnbldDB** toggles provides further quick controls on how these two groups of entities are rendered.



Of course, similar controls are available for domains and blocks as well.

Advanced Tools

A number of advanced tools exist in Gridgen for manipulating grid and database entities. Most of these can be found in the **DB Modification** and the **Modify Connector**, **Domains**, and **Blocks** sub-menus, as indicated in the table on the right. Some of these tools have already been discussed or applied in the tutorials.

In this section, a more systematic look at these and a few other useful commands not found in the modify menus will be provided.

Modify Commands And The Entities On Which They Act

command	database	connector	str domain	unstr domain	str block	unstr block
translate	✓	✓	✓	✓	✓	✓
scale	✓	✓	✓	✓	✓	✓
stretch	✓	✓	✓	✓	✓	✓
rotate	✓	✓	✓	✓	✓	✓
re-extrude			✓		✓	✓
split	✓	✓	✓		✓	
join	✓	✓	✓	✓	✓	
mirror	✓	✓	✓	✓	✓	✓
project	✓	✓	✓	✓		
smooth	✓					
add		✓				
insert		✓				
erase		✓				
edit	✓	✓				
least squares fit		✓				

Transformations

Transformation commands can be applied to all entity types from the corresponding menu. They include:

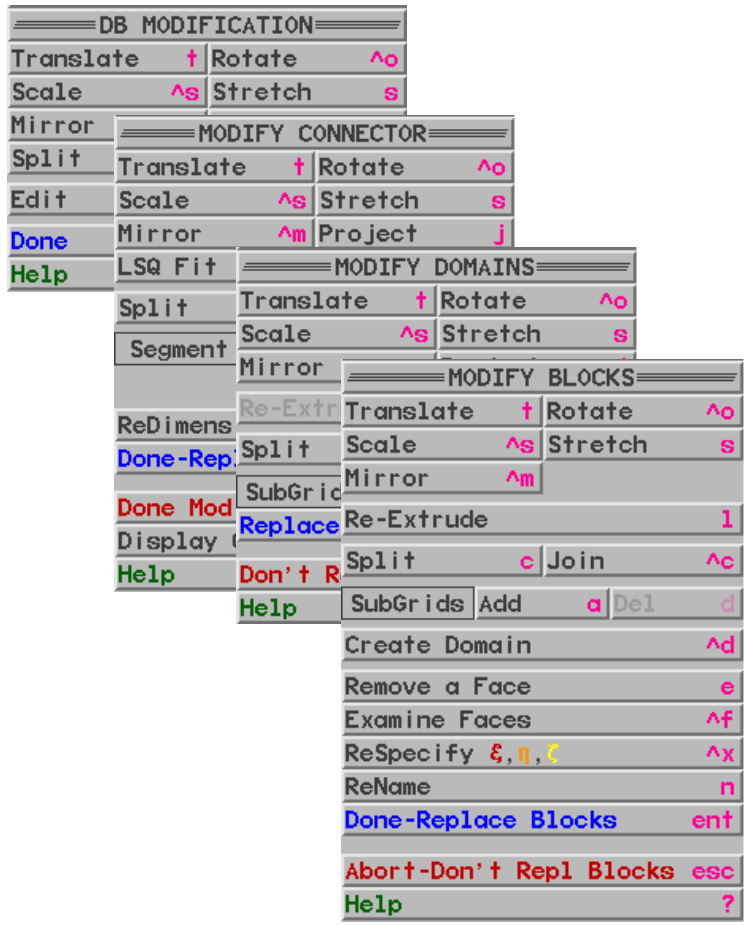
Translate alters one or more selected entities by moving them to a new position in Cartesian space. User picks handle and provides x, y, and z offsets.

Rotate alters one or more selected entities by rotating them about a specified axis. User picks axis and provides rotation angle.

Scale alters one or more selected entities by scaling. User picks anchor (fixed point during scaling) and provides handle or x, y, and z scale factors.

Stretch alters one or more selected entities by scaling along a defined vector. User picks anchor and handle.

Mirror reflects selected entity about a specified plane.



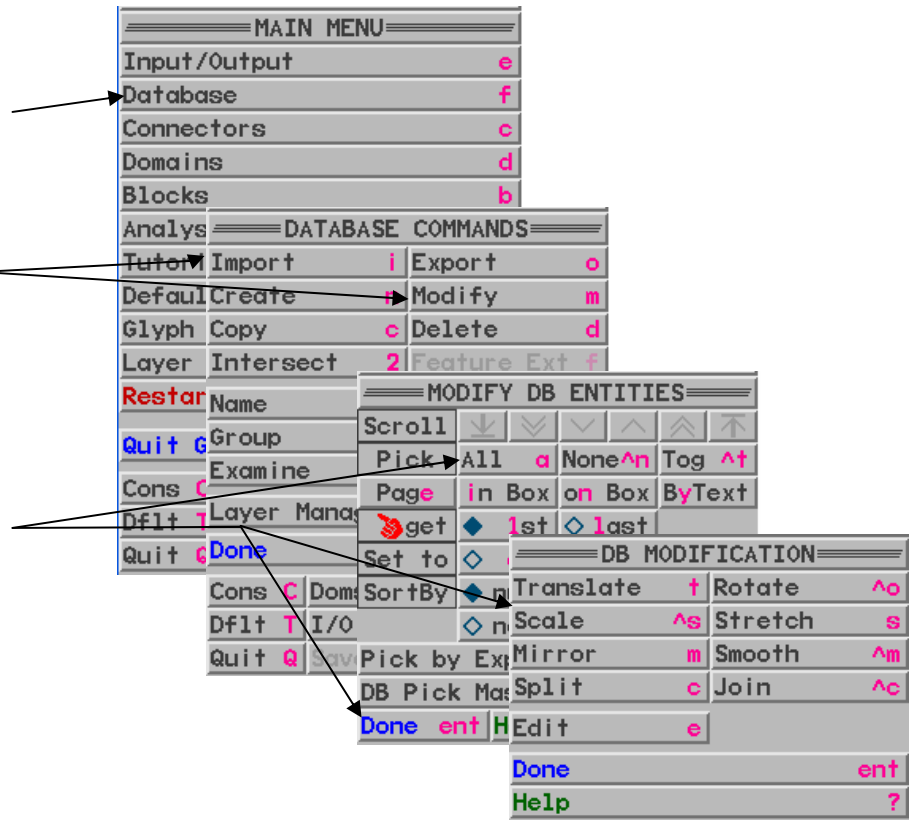
Transformations (Continued)

As an example, the **Scale** command will be used to modify the airfoil database geometry.

From the **MAIN MENU**, **Database** is selected to open the **DATABASE COMMANDS** menu.

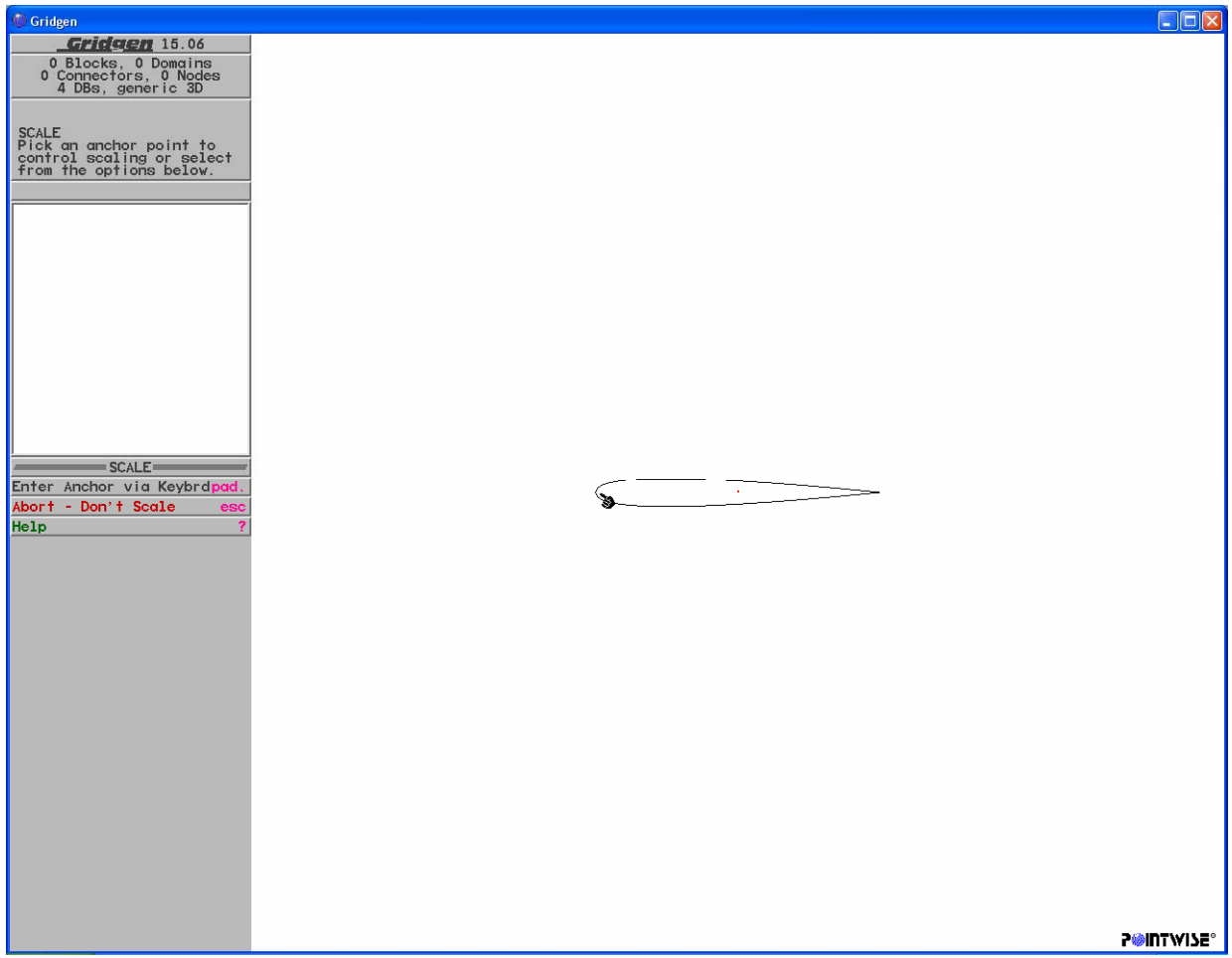
Import is used to read in the reext.dba file from the appropriate directory, and **Modify** is then chosen to open the **MODIFY DB ENTITIES** menu.

Pick All is selected to pick the pair of entities that comprise the airfoil geometry. When **Done** is selected, the **DB MODIFICATION** menu will open, and **Scale** can be picked.



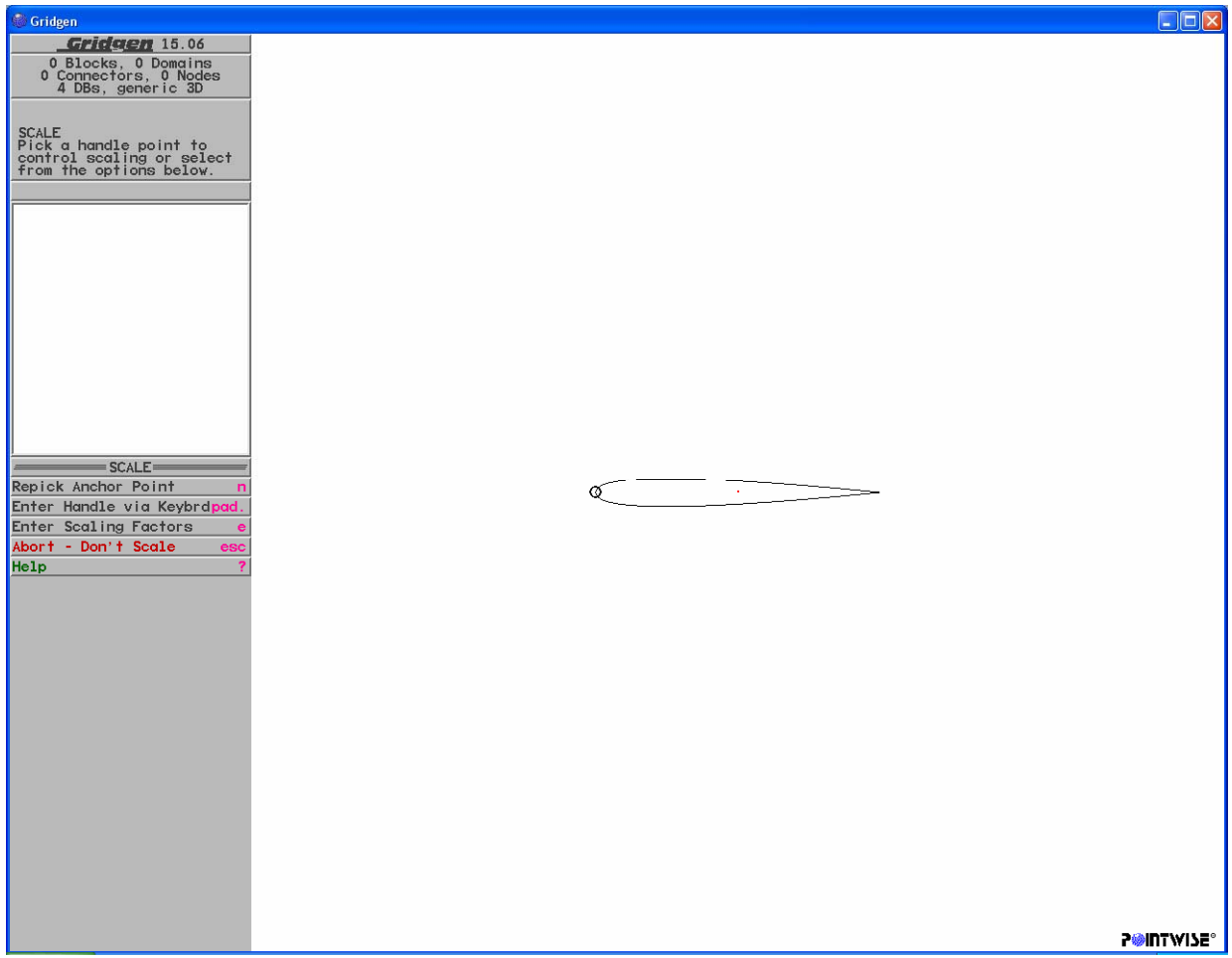
Transformations (Continued)

Once **Scale** is selected, the menu changes to allow the anchor point to be selected via the mouse from the Display or entered via the keyboard. In this example, the leading edge will serve as the anchor, and the mouse cursor has been moved into position.



Transformations (Continued)

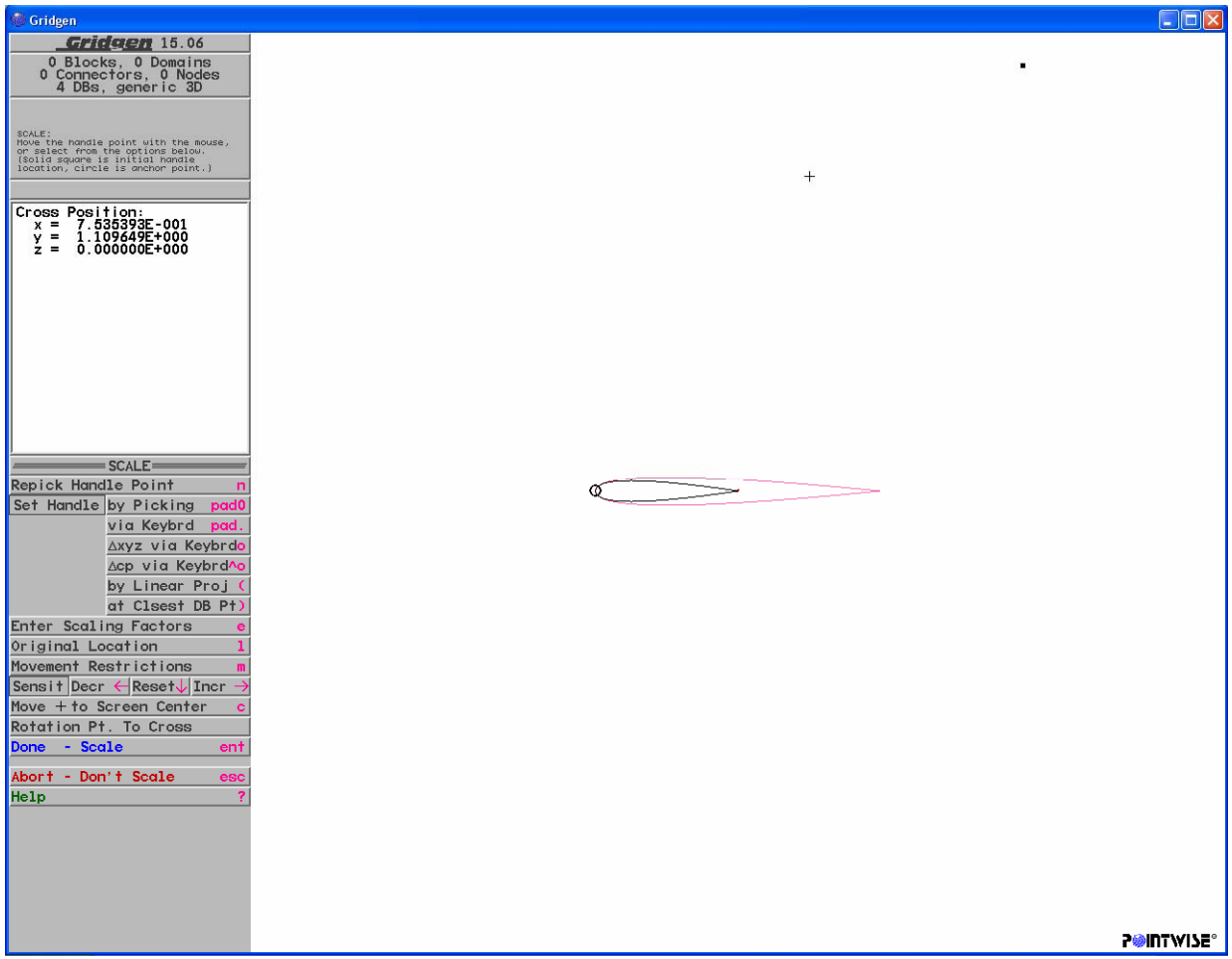
Once the anchor point has been selected, it becomes highlighted with a circle. The scale can be set graphically by picking a handle point, either by mouse clicking in the Display window or by typing via the keyboard. Alternatively, scale factors can be entered via the keyboard. In this example, **Enter Handle via Keybrd** will be selected, and 1.5, 1.5, 0 is entered for the handle point.



Transformations (Continued)

Once the handle point is defined, a black square (the handle point) appears with a cross superimposed in the Display window. By holding down the right mouse button, the cross is moved, scaling the selected database entities, as seen at right. The original database entities are shown in pink, while the scaled entities appear in black. When the desired scaling has been achieved, **Done – Scale** is selected to complete the process.

Stretch is setup similar to **Scale** by defining anchor and handle points.

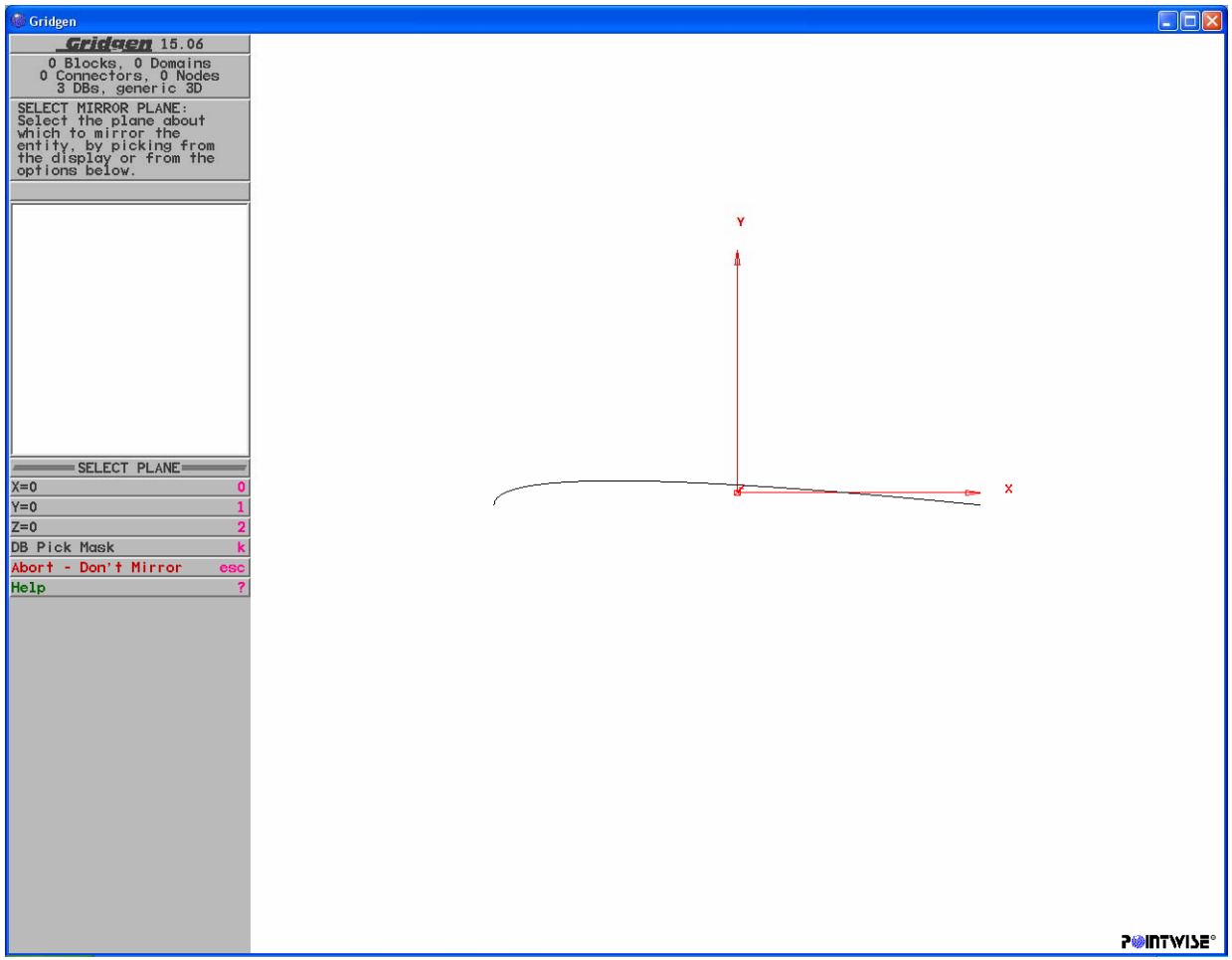


Transformations (Continued)

Mirror is a special version of the scale command that reflects either database or grid entities about a plane (either a coordinate plane or a previously defined plane in the database).

To see how it works, the lower database curve from the airfoil has been deleted so it can be replaced by mirroring the upper surface database curve.

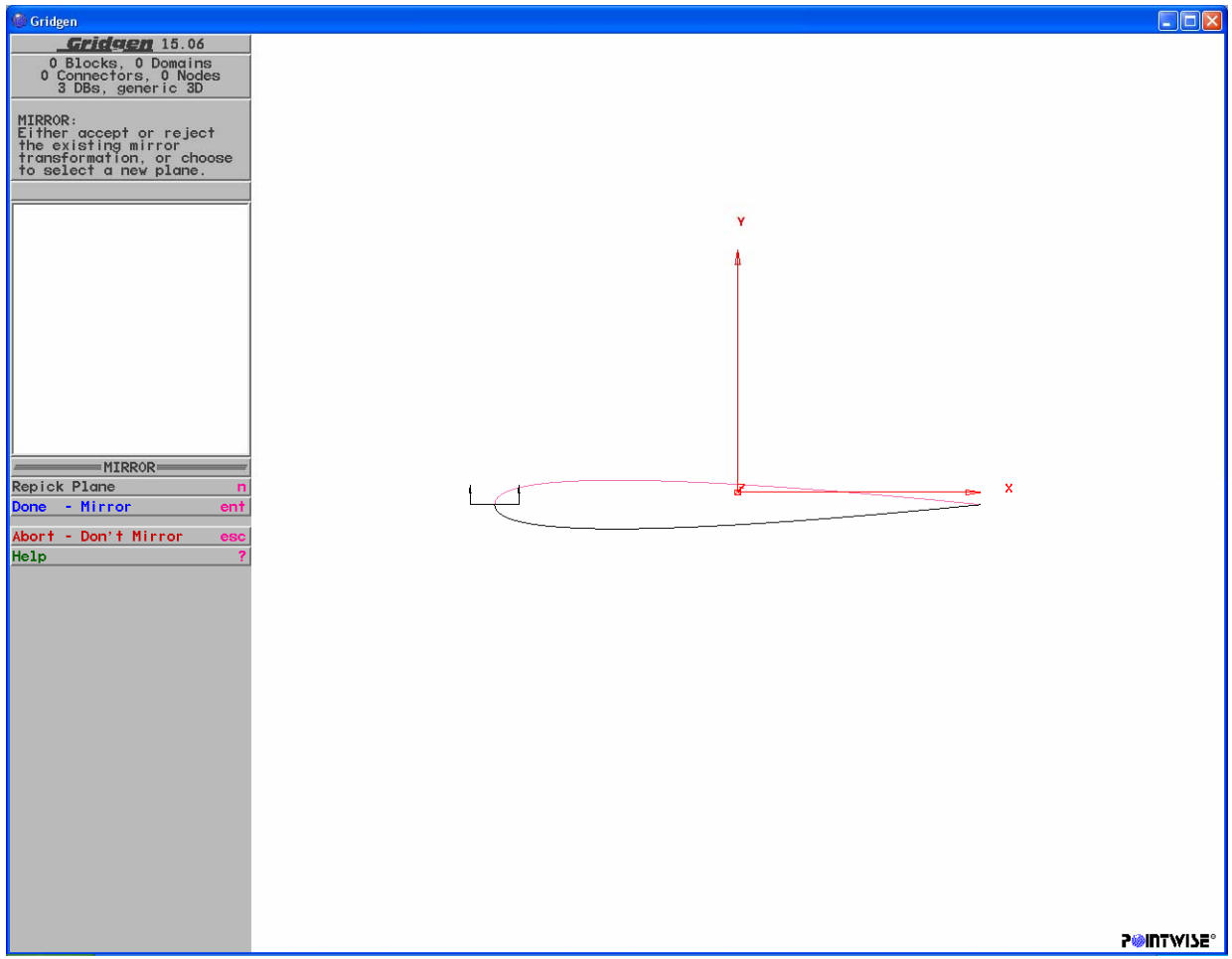
After selecting **Modify** in the **DATABASE COMMANDS** menu, the upper airfoil database entity is selected and **Mirror** selected in the **DB MODIFICATION** menu. This leads to the **SELECT PLANE** menu at right. The **Y=0** plane is selected to perform the reflection.



Transformations (Continued)

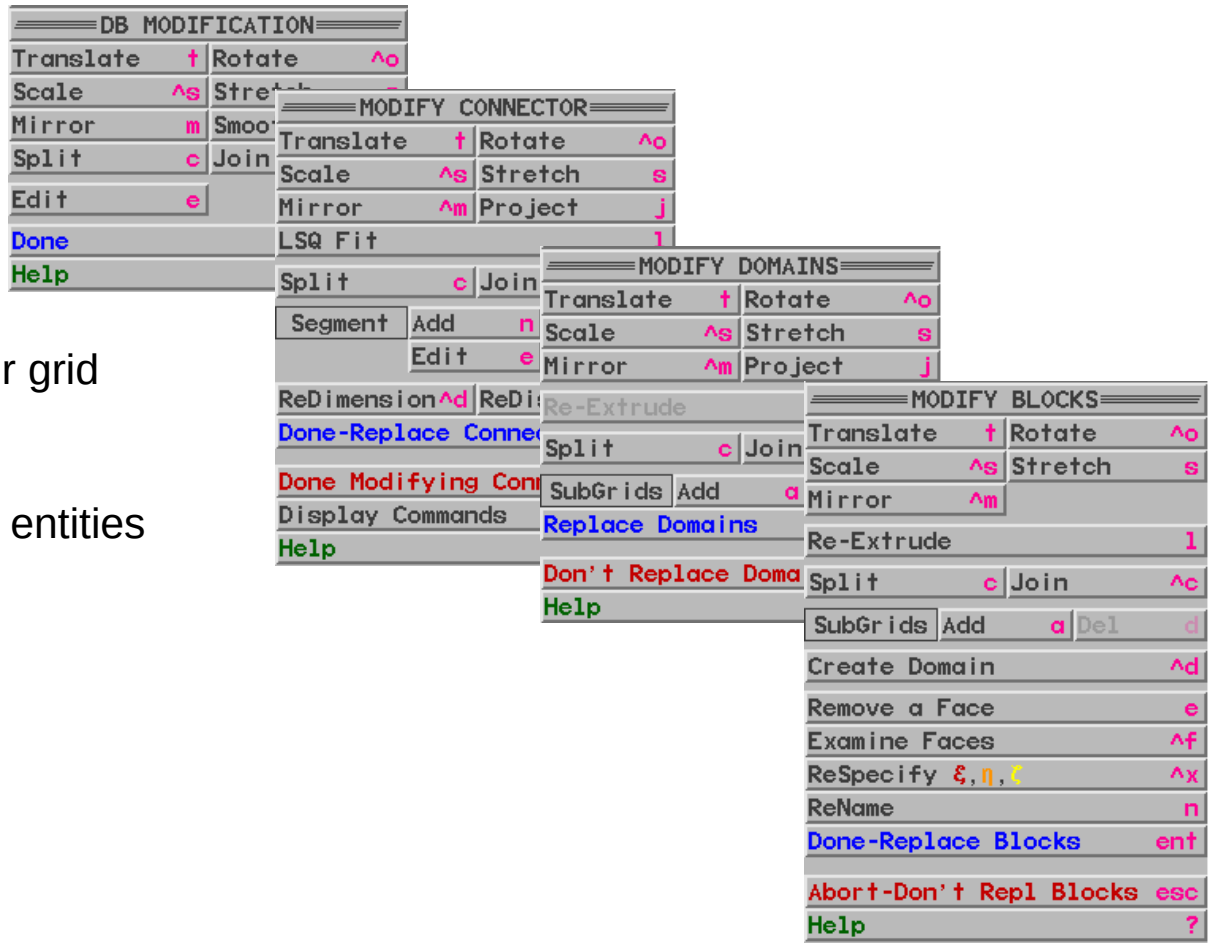
This is the result. If satisfactory, **Done – Mirror** is selected to complete the process. If not, **Abort – Don't Mirror** can be selected to cancel.

Note: The **Mirror** command here in the **Modify** submenu will simply move the existing database entity. To recreate the airfoil, the **Mirror** command in the **Copy** submenu would be used to make a mirror copy of the upper surface database entity, leaving the upper surface entity in place.



Split/Join

As with the transformation commands, **Split** and **Join** are found in the **DB MODIFICATION** and **MODIFY CONNECTOR**, **DOMAINS**, and **BLOCKS** menus.



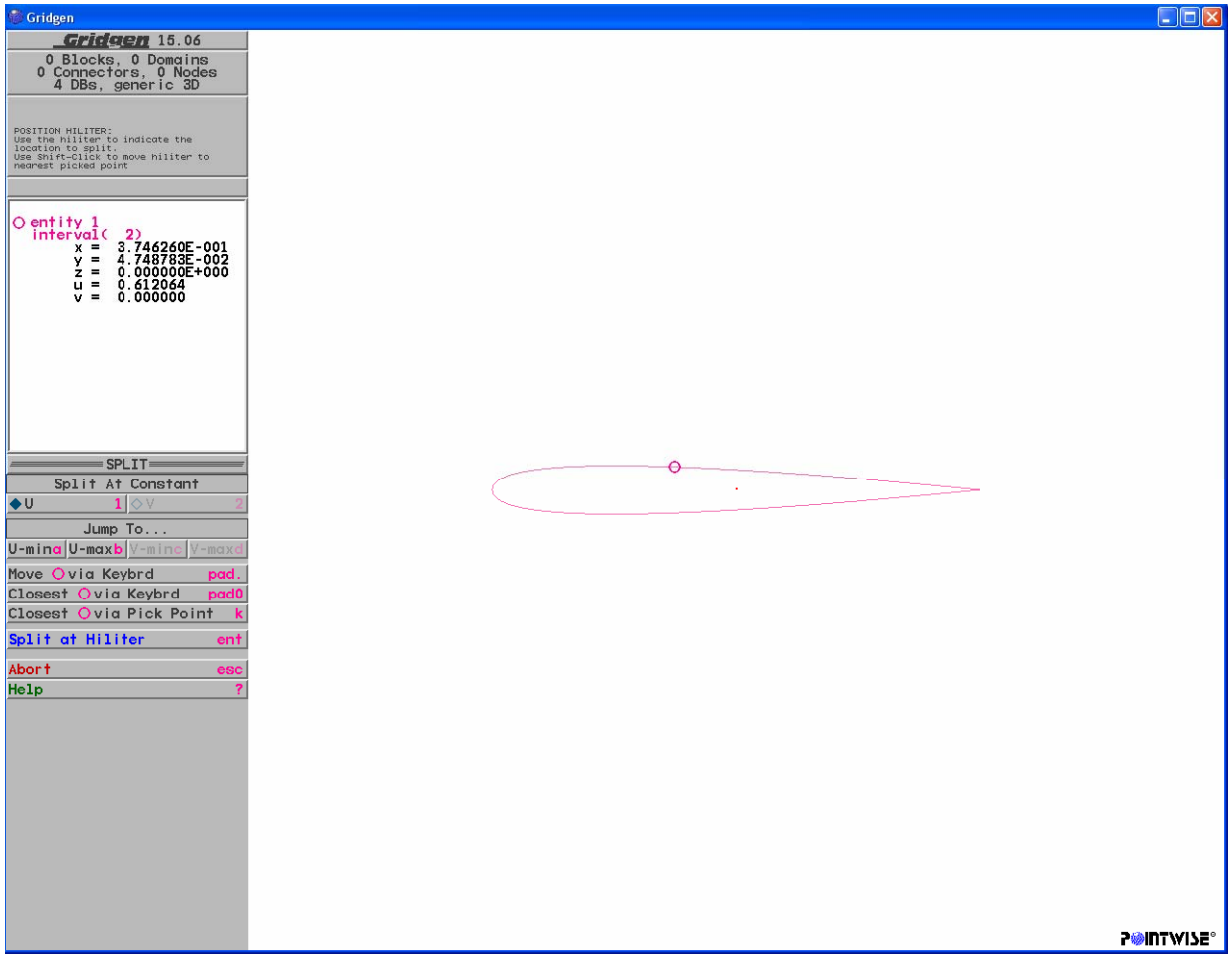
Split is used to divide a database or grid entity into two separate entities.

Join is used to assemble individual entities into a larger entity.

Split/Join (Continued)

As an example, the upper surface of the airfoil database will be split.

After selecting **Modify** in the **DATABASE COMMANDS** menu, the upper airfoil database entity is selected and **Split** selected in the **DB MODIFICATION** menu. This leads to the **SPLIT** menu at right. The Hiliter, ○, is moved along the entity via the mouse while holding down the right mouse button or via the menu commands. Once the desired location is found, **Split at Hiliter** is selected. The menu changes, and the choice is confirmed by clicking on **Split It?**

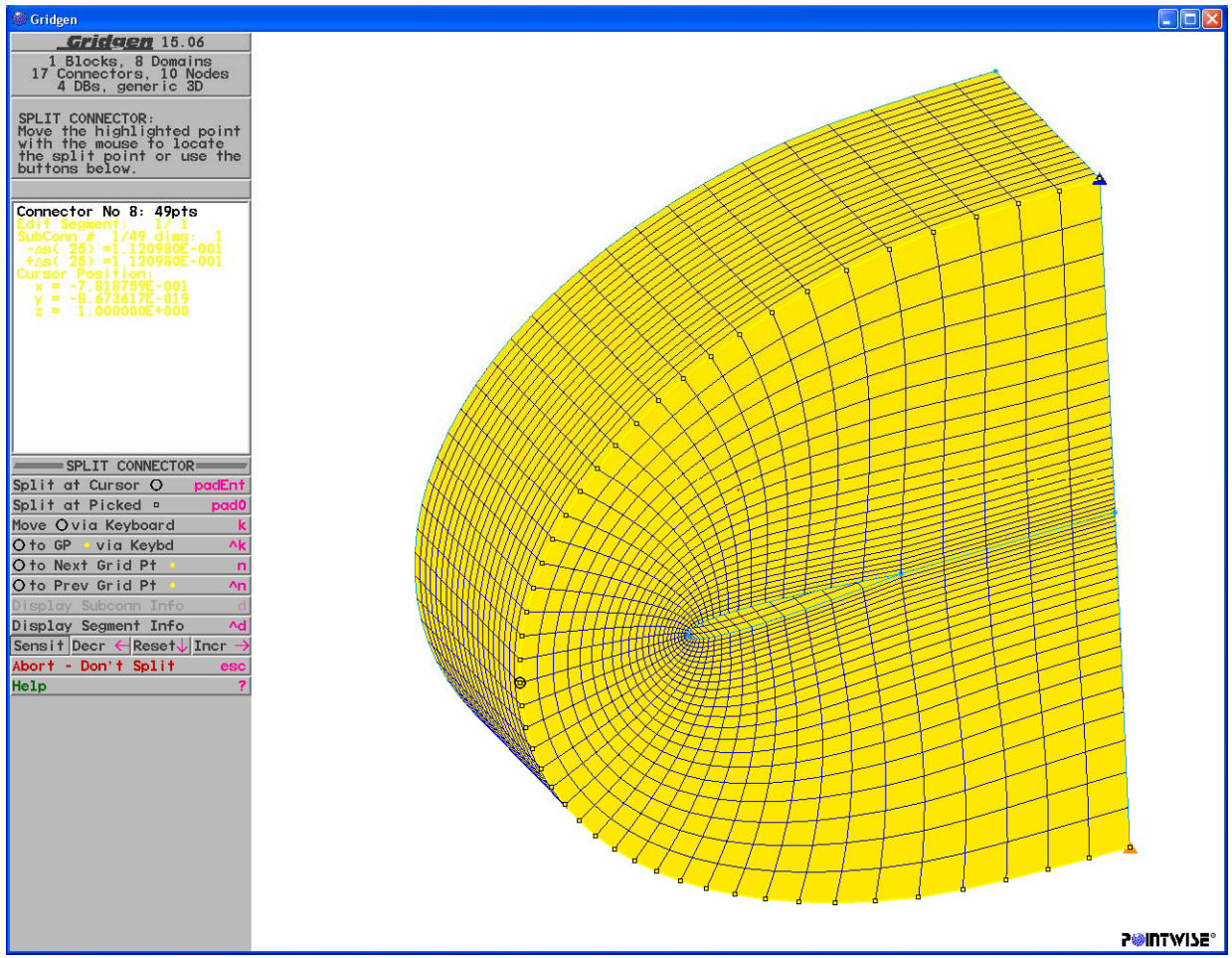


Split/Join (Continued)

In the case of connectors, the **SPLIT CONNECTOR** menu opens after the connector to be split has been selected and **Split** in the **MODIFY CONNECTOR** menu chosen.

Here, the cursor, ○, is moved along the connector via the mouse while holding down the right mouse button or via the menu commands. Once the desired location is found, **Split at Cursor** ○ or **Split at Picked** □ is selected.

The split is immediately made, and the cursor moves to the next connector to begin again. If no more splits are needed, **Abort – Don't Split** should then be selected.

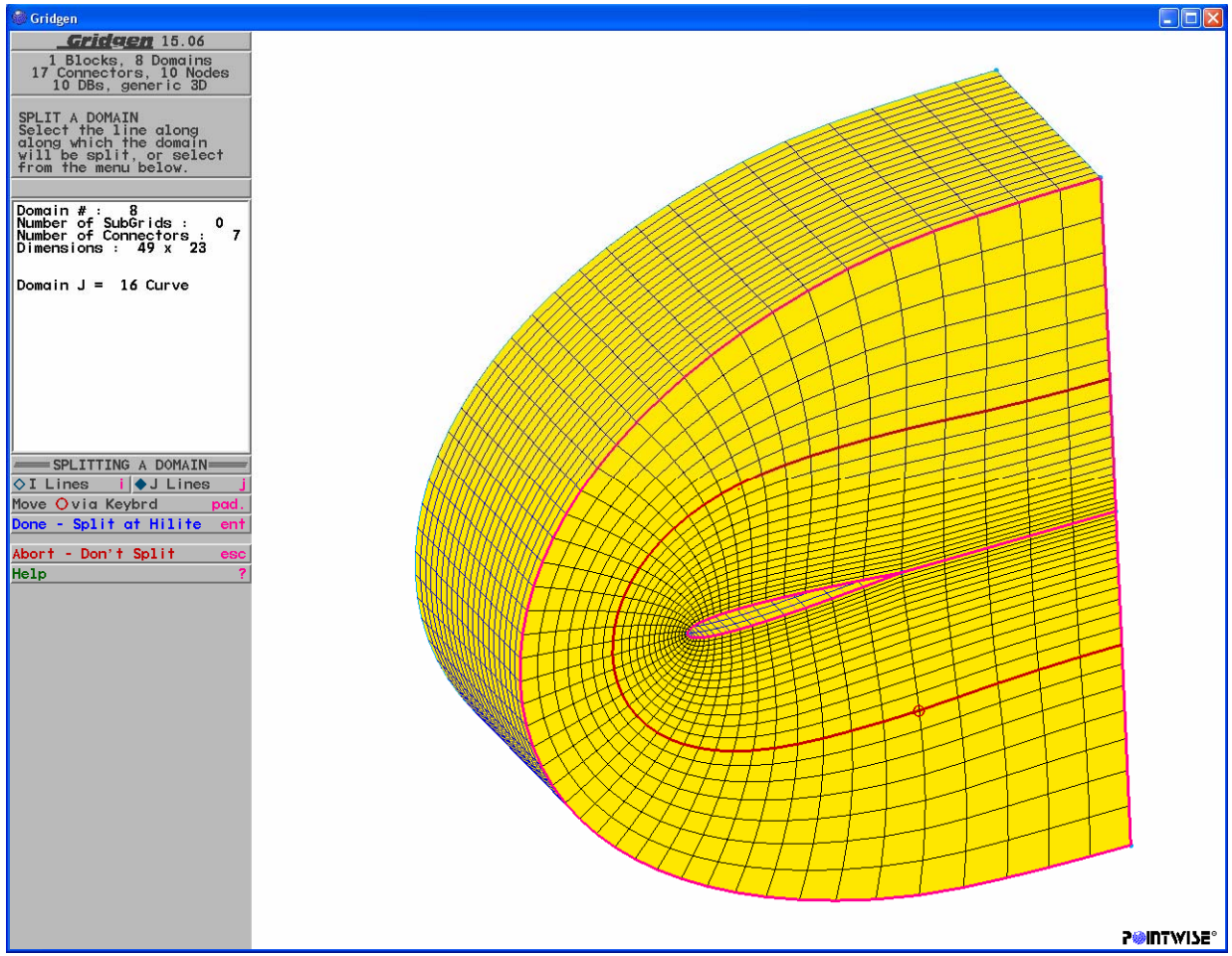


Split/Join (Continued)

For structured domains, the **SPLITTING A DOMAIN** menu opens after the domain has been picked and **Split** in the **MODIFY DOMAINS** menu chosen.

Here, the choice of constant **I Lines** or **J Lines** (as in I and J coordinates) must be made first by toggling the corresponding button. The hilite is then moved across the corresponding coordinate curves via the mouse while holding down the right mouse button or via the menu's keyboard command. Once the desired location is found, **Done - Split at Hilite** is selected.

The split is immediately made, and the GUI returns to the **DOMAIN COMMANDS** menu.

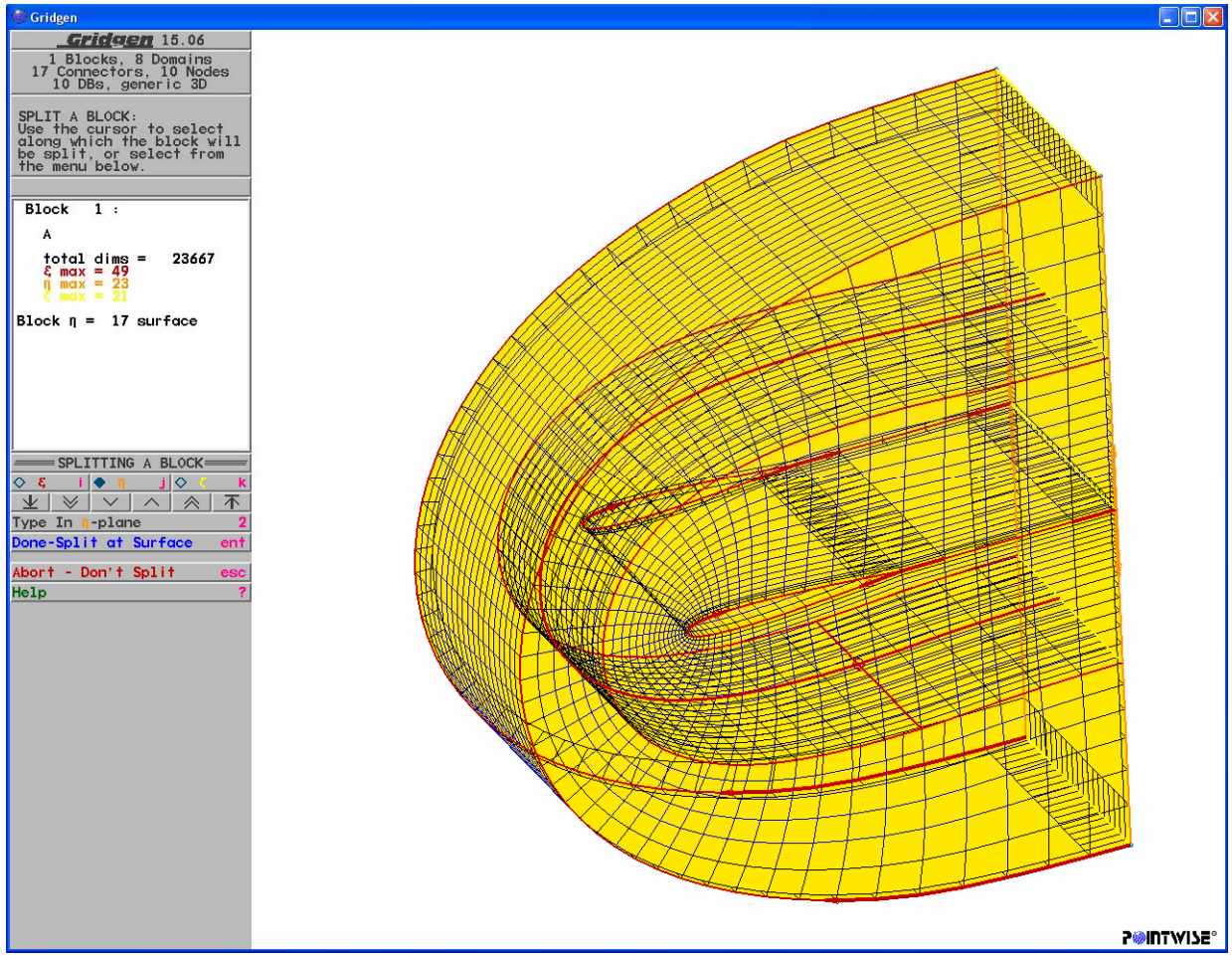


Split/Join (Continued)

For structured blocks, the **SPLITTING A BLOCK** menu opens after the block has been picked and **Split** in the **MODIFY BLOCKS** menu chosen.

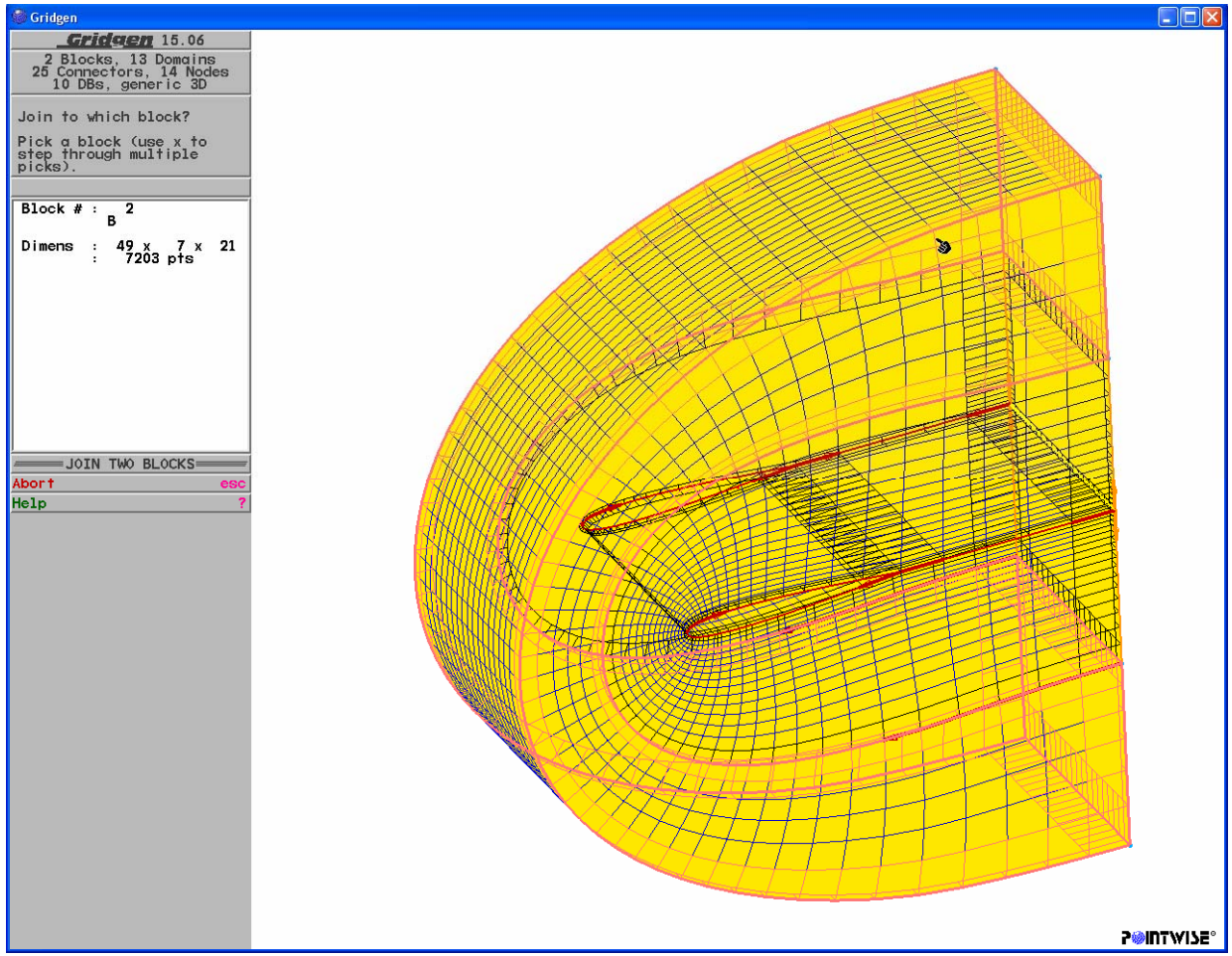
Here, the choice of constant computational coordinate surfaces (ξ, η, ζ) must be made first by toggling the corresponding button. The selection surface is then moved via the mouse while holding down the right mouse button, the arrow buttons, or the menu's keyboard command. Once the desired surface is found, **Done - Split at Surface** is picked.

The split is immediately made, and the GUI returns to the **BLOCK COMMANDS** menu.



Split/Join (Continued)

Joining entities is even easier and follows a similar process for all entities. As an example, the two blocks just split from the airfoil model will be reassembled. The first block to be joined is picked from **MODIFYING BLOCKS** menu. Once selected, the **MODIFY BLOCKS** menu opens, and **Join** is selected. Here the other block is picked via the mouse from the Display window. Once picked, the two blocks are immediately joined, and the GUI returns to the **BLOCK COMMANDS** menu.



Database Creation

For the next example, the two-dimensional airfoil database must be modified into a three-dimensional wing. The initial steps are:

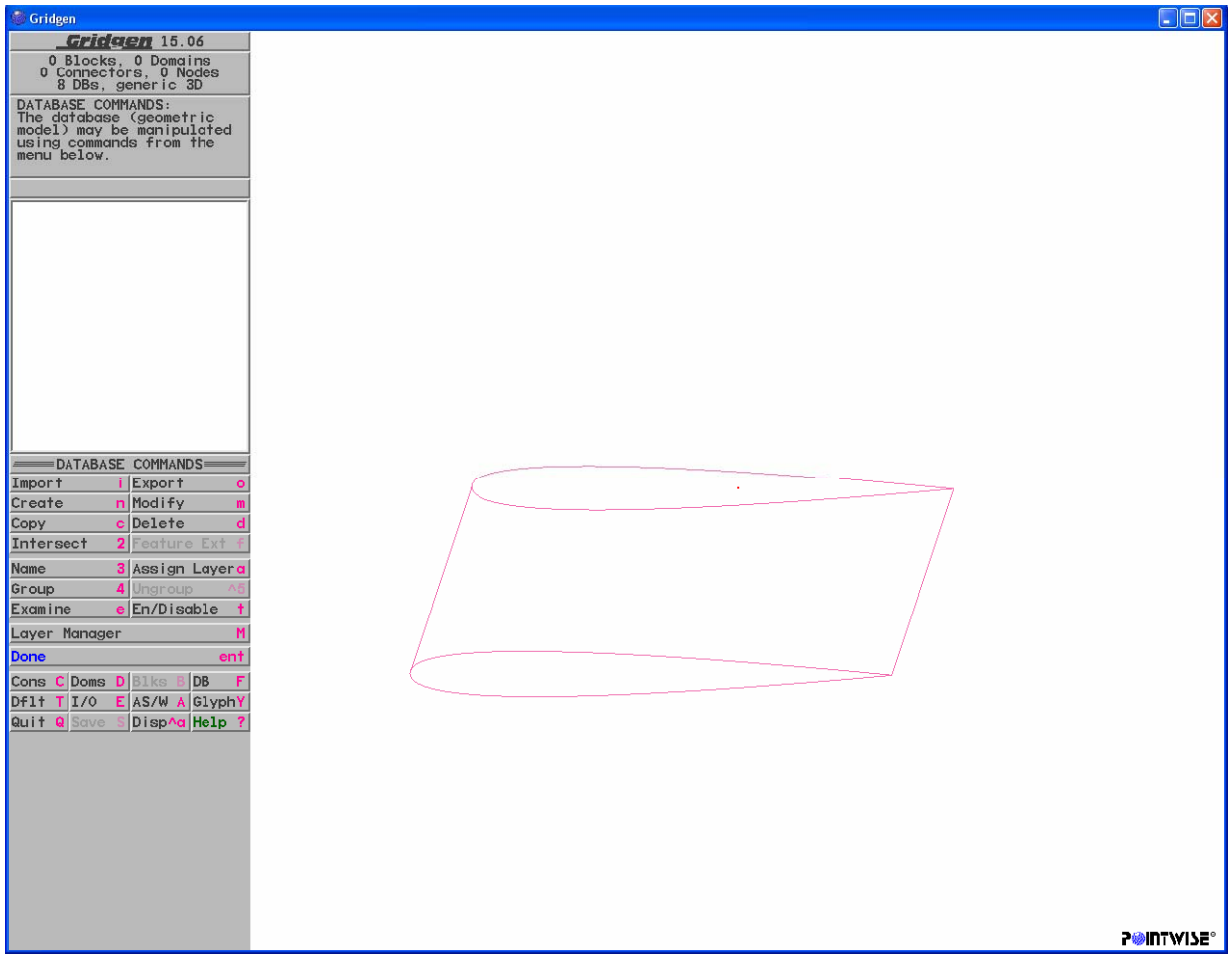
1. The reext.dba database file is imported.

2. The two database entities making up the airfoil are copied.

3. The copies are translated 1 unit in the z-direction.

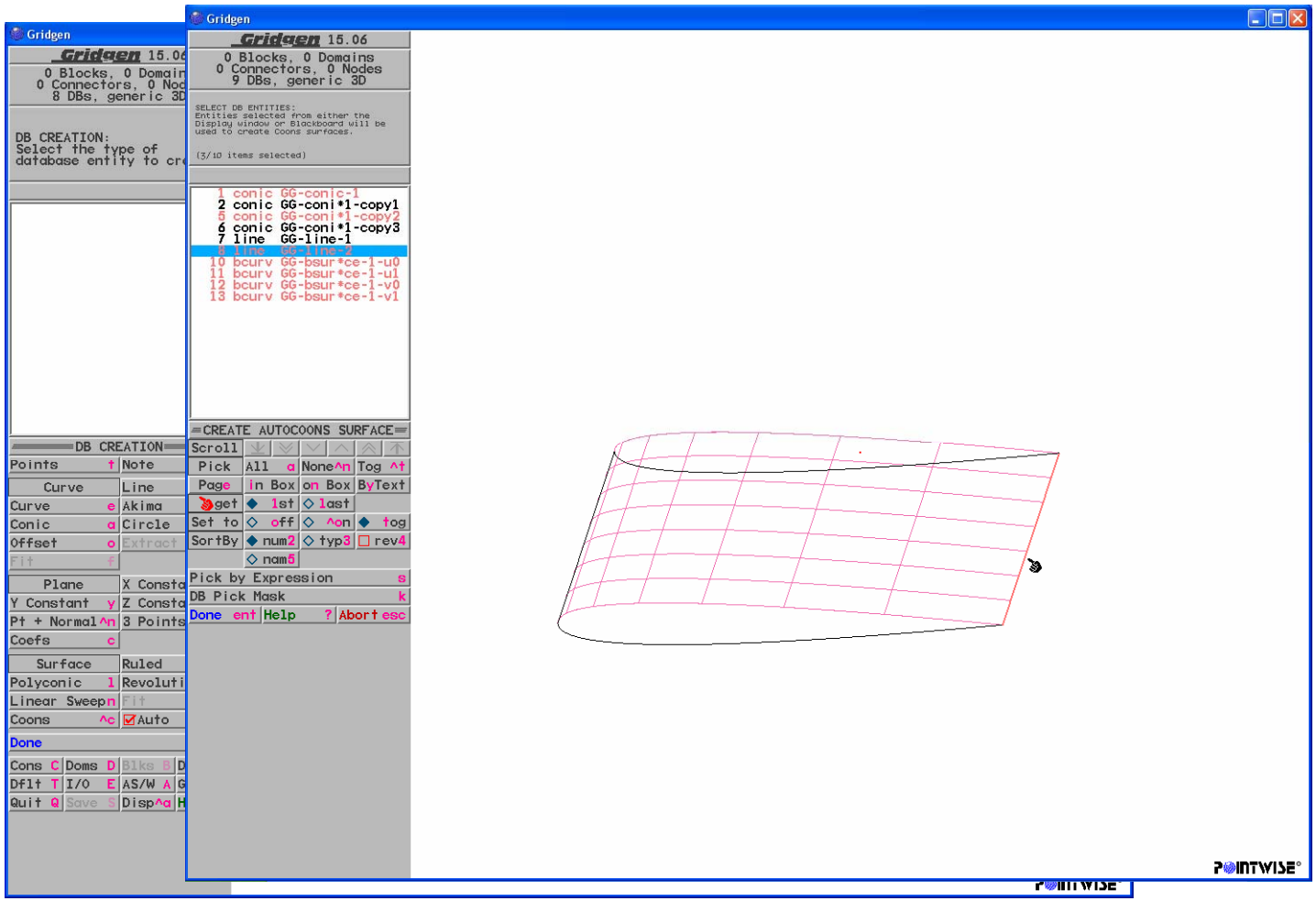
4. Two **Curve Line** entities are created connecting leading and trailing edges of the original airfoil and the copy.

Create is next picked.



Database Creation (Continued)

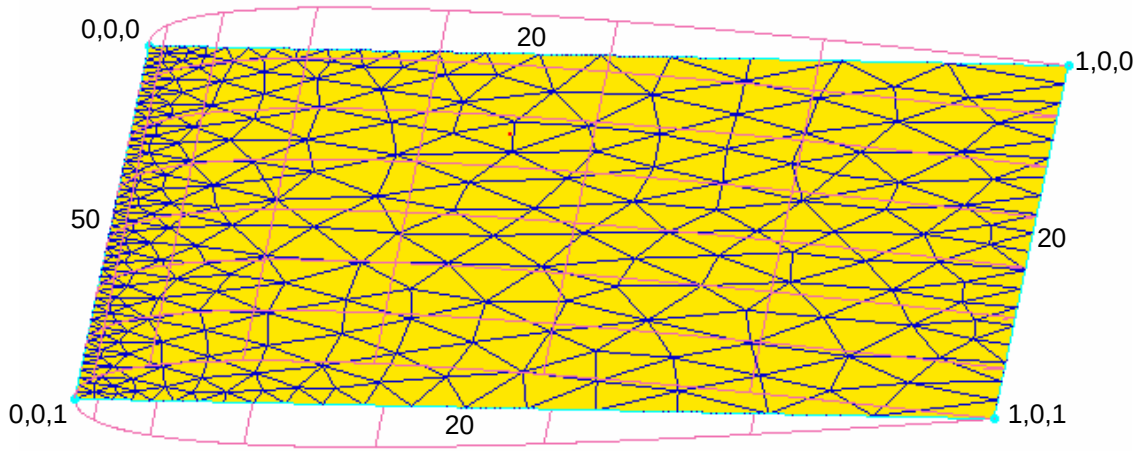
In the **DB CREATION** menu, **Coons** is selected to define a Coon surface forming the upper wing. The four database entities marking the boundaries of the upper surface are then picked via the mouse. Once selected, **Done** is picked and the surface saved. The process is then repeated for the lower wing surface – the last entity, the trailing edge, is just about to be selected.



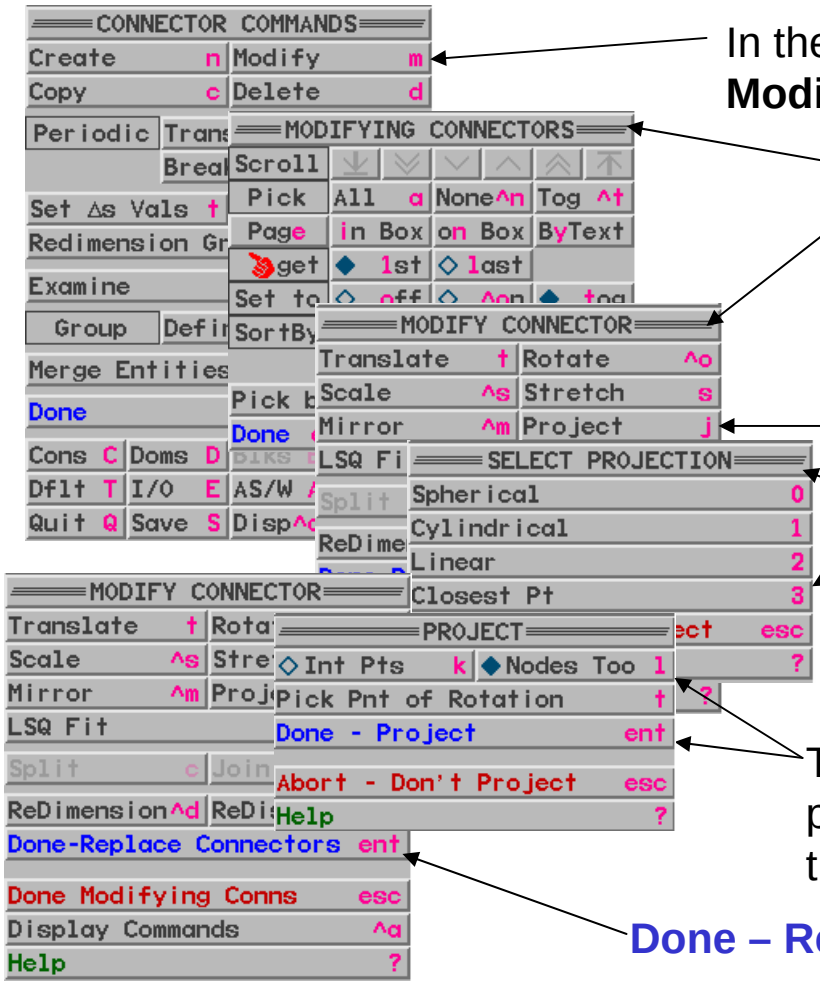
Projection

This unstructured domain will now be projected to the upper surface wing database.

The first step will be to disable the database entities that make up the lower surface of the wing. Once that is done, the four connectors can be projected to the upper wing surface, followed by the rest of the domain.



Projection (Continued)



In the **CONNECTOR COMMANDS** menu, **Modify** is selected.

This opens the **MODIFYING CONNECTORS** menu, where the four connectors can be chosen via the mouse. When **Done** is selected, the **MODIFY CONNECTOR** menu opens.

Here, **Project** is selected.

This opens **SELECT PROJECTION**, where **Closest Pt** (i.e., the connectors' grid points will be moved to the closest points on the database entities) is chosen.

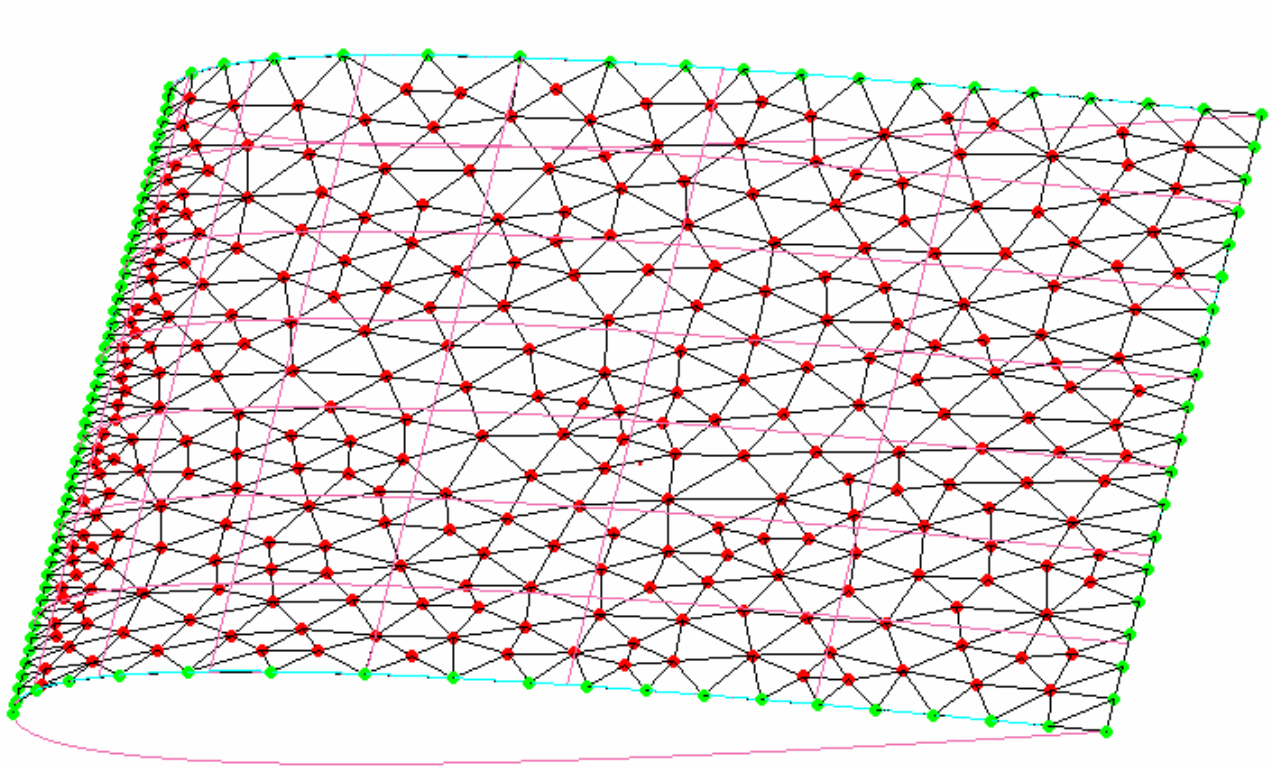
To make sure all grid points on the connectors are projected, **Nodes Too** is selected. **Done – Project** is then clicked.

Done – Replace Connectors finishes the process.

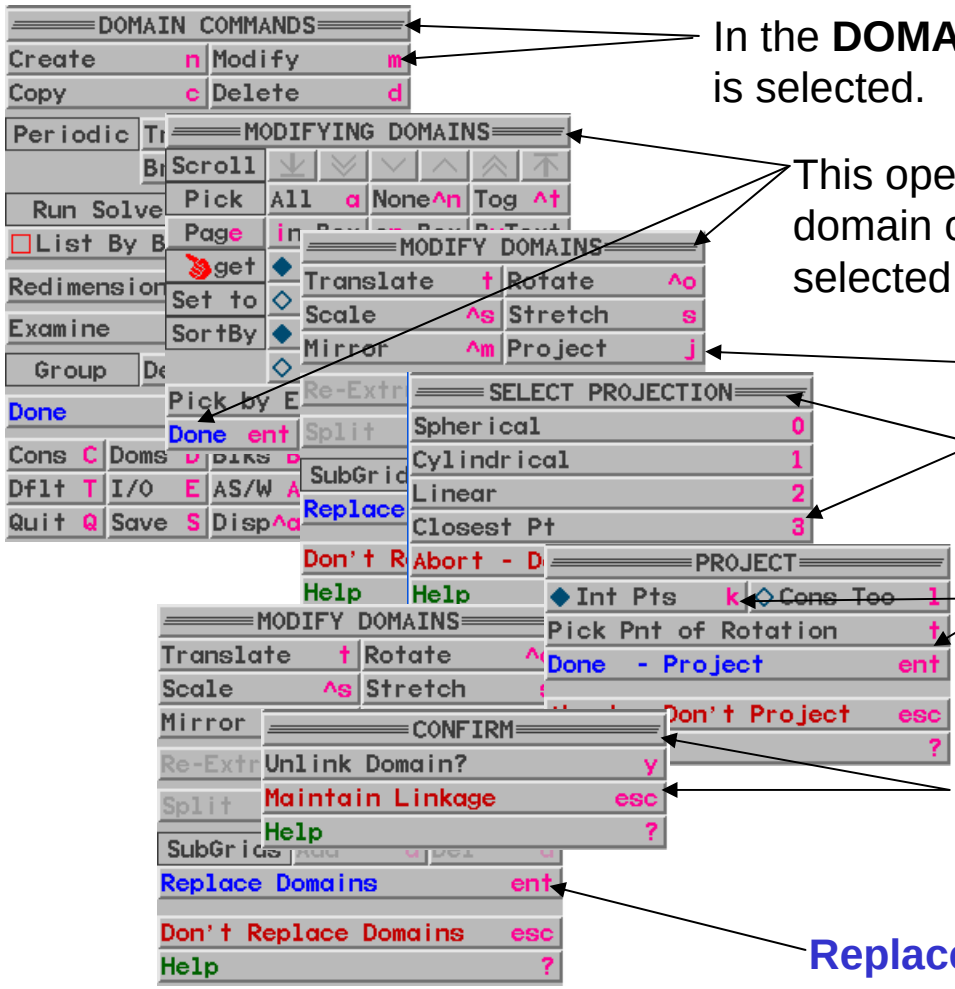
Projection (Continued)

While all of the edges of the domain are now coincident with the database, the interior of the domain is not. This can be seen using the Examine command.

Again, the first step is disabling lower surface database entities.



Projection (Continued)



In the **DOMAIN COMMANDS** menu, **Modify** is selected.

This opens the **MODIFYING DOMAINS** menu, where the domain can be chosen via the mouse. When **Done** is selected, **MODIFY DOMAINS** opens.

Here, **Project** is selected.

This opens **SELECT PROJECTION**, where **Closest Pt** is again chosen.

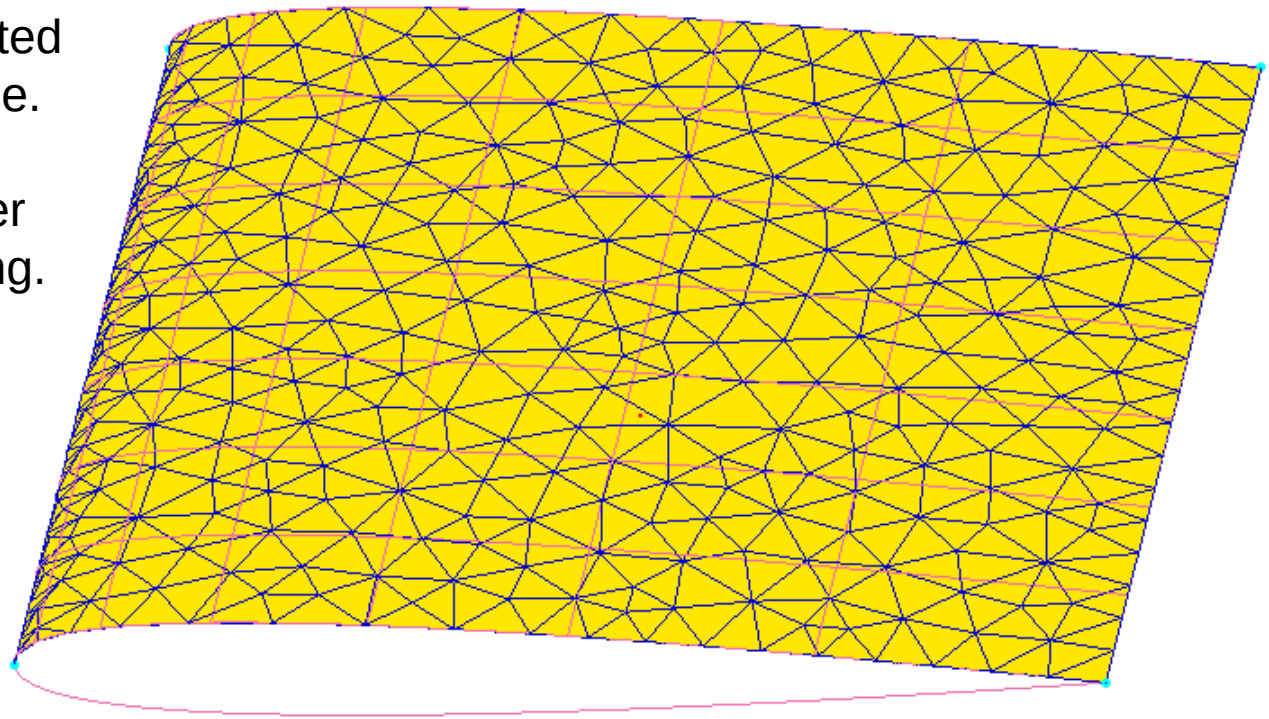
Since the edges are already on the database, **Int Pts** is chosen. **Done – Project** is then clicked.

Maintain Linkage is clicked in the **CONFIRM** menu.

Replace Domains finishes the process.

Projection (Continued)

The domain is now projected to the upper wing database. The process could be repeated to project another domain onto the lower wing.



Miscellaneous

The remaining commands in this table have more limited uses so will only be summarized:

Re-extrude allows previously extruded domains and blocks to be extruded again.

Smooth provides a means of smoothing database curves through fitting.

Add is used to add a segment on the end of an existing connector.

Insert is used to insert a segment into a connector.

Erase will remove segment(s) from a connector.

Edit allows individual control points in a segment to be re-positioned.

Least Squares Fit replaces a dimensioned connector's shape with a least squares fit of its grid points.

Modify Commands And The Entities On Which They Act

command	database	connector	str domain	unstr domain	str block	unstr block
translate	✓	✓	✓	✓	✓	✓
scale	✓	✓	✓	✓	✓	✓
stretch	✓	✓	✓	✓	✓	✓
rotate	✓	✓	✓	✓	✓	✓
re-extrude			✓		✓	✓
split	✓	✓	✓		✓	
join	✓	✓	✓	✓	✓	
mirror	✓	✓	✓	✓	✓	✓
project	✓	✓	✓	✓		
smooth	✓					
add		✓				
insert		✓				
erase		✓				
edit	✓	✓				
least squares fit		✓				

- In this session:
  - Layer Manager.
  - Tutorial: “Cubes and Spheres: Layer Manager.”
  - Advanced Tools.
  - Tutorial: “B-747 Nacelle: Working with an IGES Database.”

- In this session:
  - Glyph.
  - Tutorial: “Converging-Diverging Nozzle: Using Glyph Scripting.”

Glyph

Glyph, Gridgen's scripting language, is based on the Tcl programming language and provides a text-based, procedural interface to Gridgen's features. Glyph scripts can be useful for:

- Establishing preferred display states and default values.
- Starting Gridgen and automatically importing database and Gridgen files for current mesh building projects.
- Encapsulating repetitive tasks.
- Running Gridgen in batch mode.
- Developing specialized meshing applications.

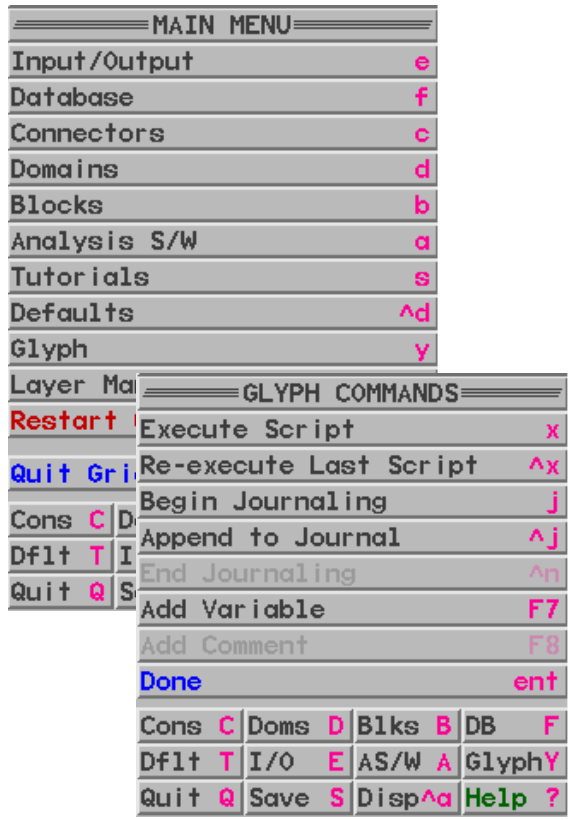
```
# Gridgen (V.15.08 REL 1) Initialization Script
# Written Mon Jan 1 00:00:00 2000
#
# This file will be automatically loaded if named "_gridgenrc"
# and located in the working directory or your home directory.
#

# DEFAULTS
gg::defDbCoordDisp UV
gg::defCFBG TM
gg::defCFFG HILGENSTOCK
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
gg::de
```

```
#######################
# RUN STRUCTURED SOLVER
#######################

gg::domTFISolverRun $dom_A

gg::domEllSolverAtt $dom_A -bg_control LAPLACE
gg::domEllSolverBegin $dom_A
  set results [gg::domEllSolverStep -iterations 100]
  # Print out the residual results neatly
  puts "iter   residual   max resid"
  foreach {i res maxres} $results {
    puts [format "%4d  %-8.4e  %8.4e" $i $res $maxres]
  }
gg::domEllSolverEnd

#########################
# EXPORT PLOT3D GRID FILE
#########################
gg::domExport "ALL" "bump.grd" \
  -style PLOT3D \
  -format ASCII \
  -precision DOUBLE

#################
# SET FLOW SOLVER
#################
```

Executing Glyph Scripts

There are four ways to execute a Glyph script :

- Placing the script file *_gridgenrc* (on Windows platforms) or *.gridgenrc* (on Unix platforms) in the home or root folder, respectively, or setting the environment variable *GRIDGEN_RCFILE* to the path/name of the Glyph script will lead to the script's execution on Gridgen startup.
- Entering the script file's name on the Gridgen command line (will execute after any RC script).
- Running Gridgen in batch mode by entering *gridgen –b my_glyph_script.glf* on the command line (Unix) or at the DOS prompt (Windows).
- Selecting **Execute Script** in the **GLYPH COMMANDS** menu.



Glyph Commands

The following commands are available in the **GLYPH COMMANDS** menu:

Execute Script – select script for execution using the file browser.

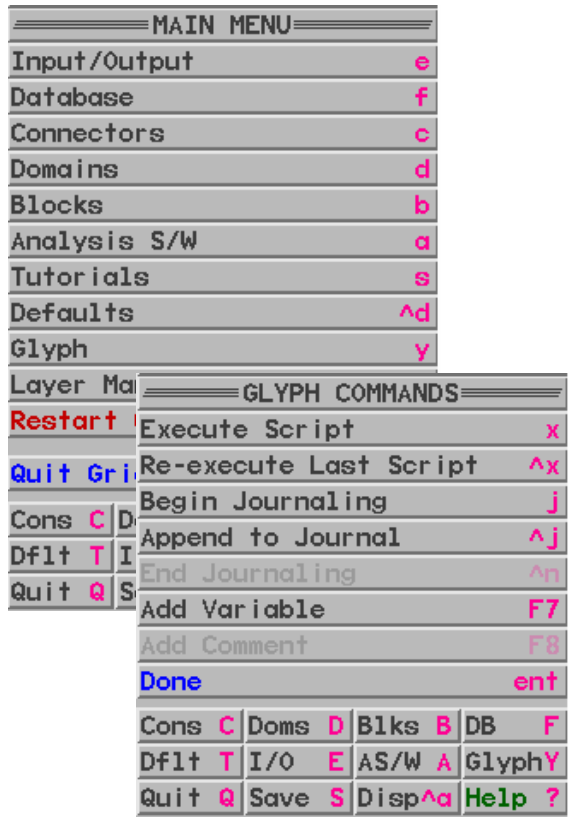
Re-execute Last Script – executes the previously executed script.

Begin Journaling – begin writing all Gridgen actions to a Glyph script file.

End Journaling – cease writing to Glyph script file.

Add Variable – add variable which is recorded in the current journal or available during the current non-journaled session.

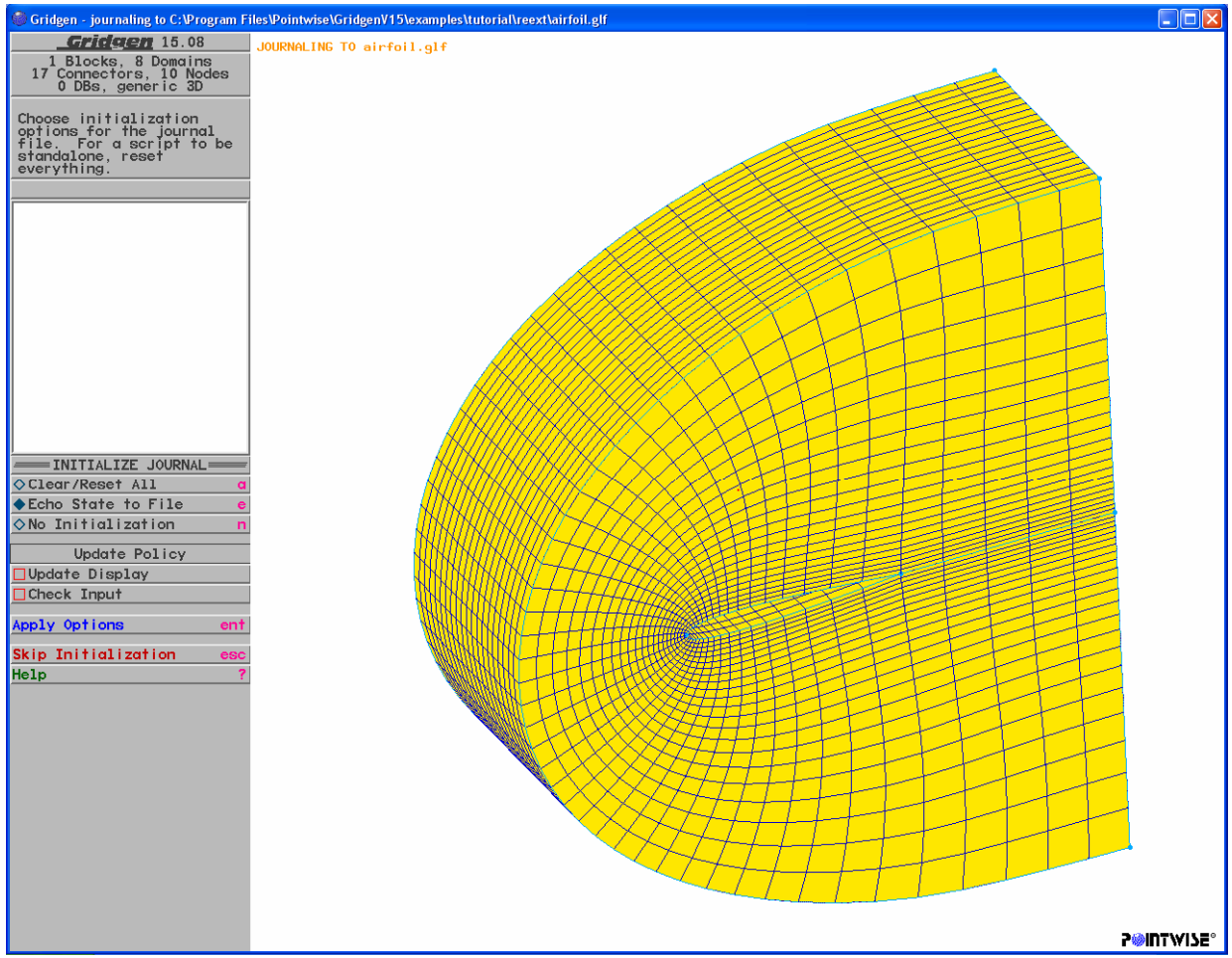
Add Comment – insert comment into script file at the current point in a journal session.



Journaling

Journaling is a Gridgen mode during which GUI commands are written to a file as their Glyph equivalents, thus allowing a meshing session to be captured and later replayed. As there isn't a one-to-one correspondence between GUI and Glyph commands, commands without Glyph equivalents will be grayed out during journaling.

During journal sessions, a notation to that effect appears in the top left of the Display window, along with the name of the journal file being written to.



Begin Journaling

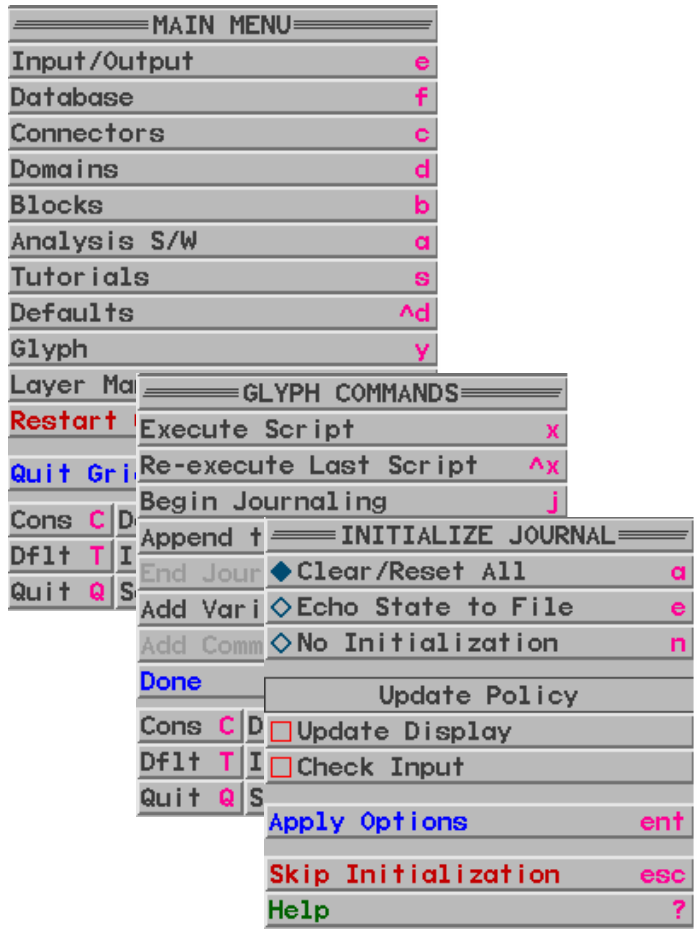
When **Begin Journaling** is selected in the **GLYPH COMMANDS** menu, the **INITIALIZE JOURNAL** menu appears. The first three buttons offer choices for initialization commands to be written at the top of the journaled Glyph script:

Clear/Reset All will clear all entities and reset all tolerances and defaults. This would be chosen to ensure that the replayed script starts completely afresh.

Echo State to File writes current defaults and tolerance settings to the journal file, but does not clear any existing grid or database entities.

No Initialization as indicated no initialization commands will be saved.

Care must be taken with the latter two options that the resulting script file is not dependent on the starting conditions or that the required starting conditions exist when it is replayed.



Begin Journaling (Continued)

The **Update Policy** options define how the journaled script will update the Display window when replayed:

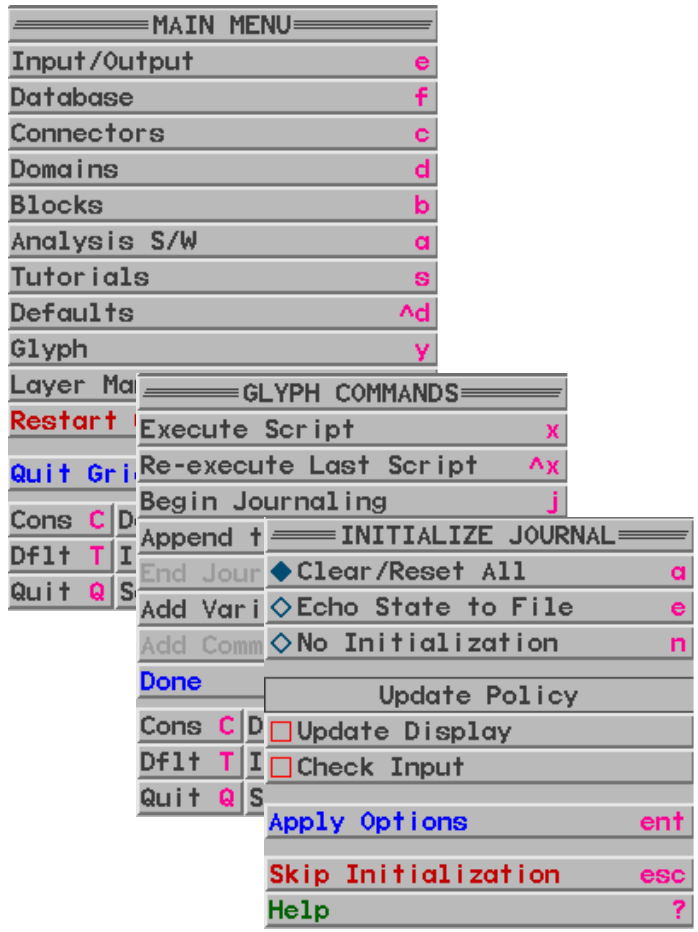
Update Display will lead to the Display being updated when any entities change appearance (including creation).

Check Input will lead to the Display being updated only when an explicit Tcl call is included in the script. The results of the script will otherwise only appear once the script finishes.

Apply Options writes the initialization and update policy commands to the journal file.

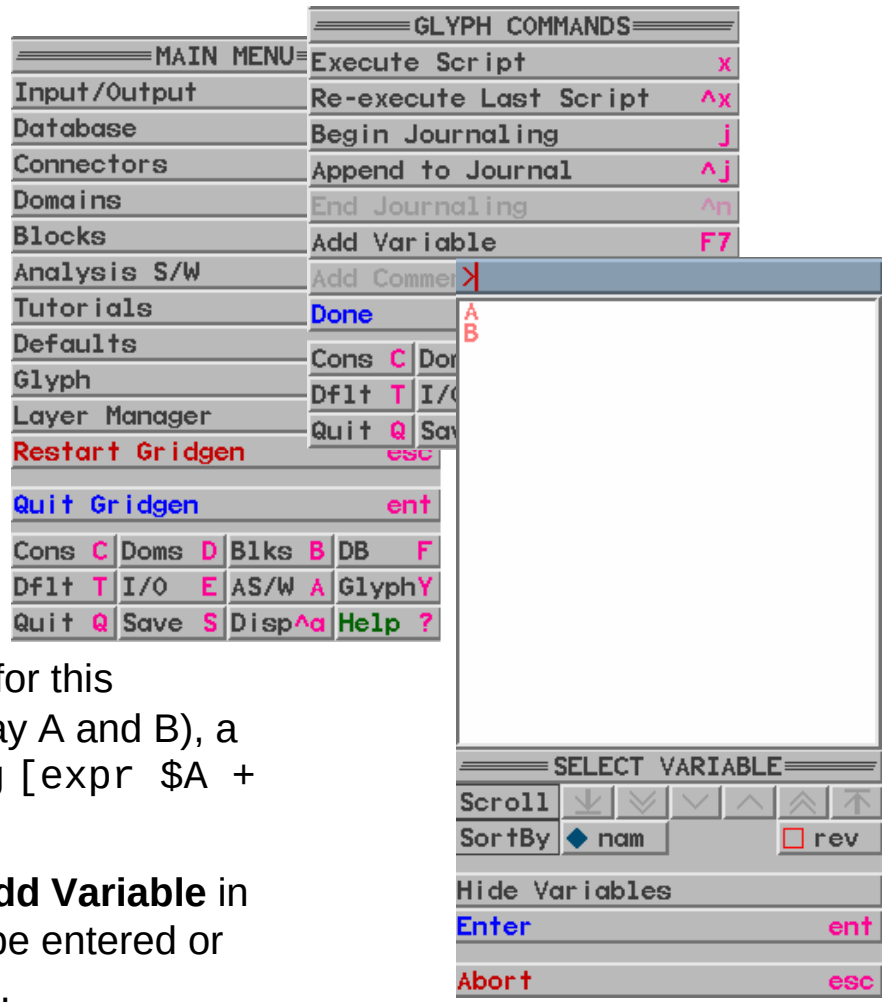
Skip Initialization indicates that no such commands should be written to the journal file.

Once one of these two buttons is selected, journaling will be enabled.



Variables

Variables are name-value pairs that can be defined and used in Gridgen for parameters that will be used often. Used in conjunction with journaling, variables can be used for parameters that need to be easily changeable for remeshing. Such a variable need only be changed in one location to effect changes throughout the mesh. Their values may be scalar or vector, real or integer or string. Once defined, they may be used whenever Gridgen requires input via the Text Input window.



For example, suppose a variable named A is defined and assigned an integer value. When prompted for the number of grid points to dimension a connector, $A could be typed in, and A's value would then be used for this dimension. If two such integer variables were defined (say A and B), a connector could be dimensioned with their sum by typing `[expr $A + $B]` at the prompt.

The process of defining a variable begins by selecting **Add Variable** in the **GLYPH COMMANDS** menu. A new name can then be entered or an existing variable can be selected to reassign its value.

Add Variables

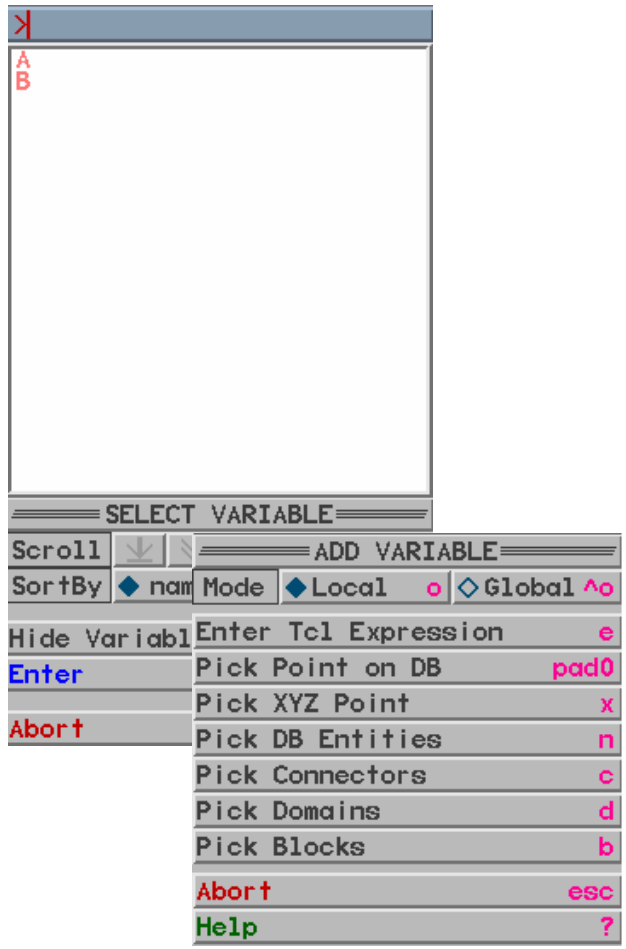
Once a new variable name has been entered or an existing variable chosen, **Enter** is selected and the **ADD VARIABLE** menu appears with the following options:

Mode Local, when journaling, limits the scope of the variable to the journaled script only.

Mode Global, when journaling, makes a variable persist after a script ends.

Enter Tcl Expression defines the variable by a Tcl expression entered in the Text Input window. The expression can simply be a numerical value or a line of Tcl programming commands.

Pick Point on DB assigns to the variable a 3-element list of the parametric u and v coordinates and ID of the point on a database entity picked in the Display window.



Add Variables (Continued)

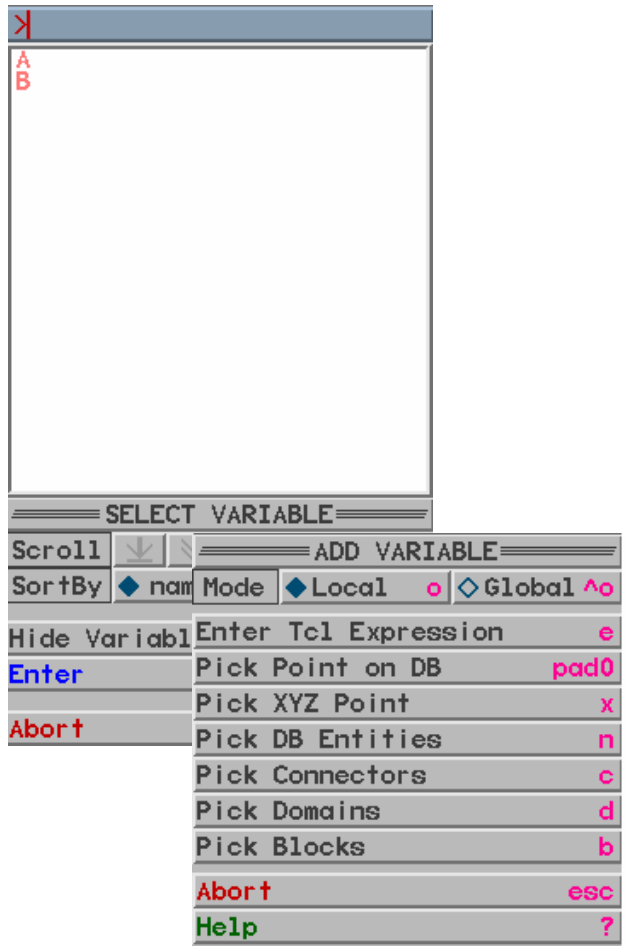
Pick XYZ Point assigns to the variable the 3-element list of x, y, and z coordinates of the point picked in the Display window.

Pick DB Entities assigns to the variable the list of IDs of the database entities picked in the Display window.

Pick Connectors assigns to the variable the list of IDs of the connectors picked in the Display window.

Pick Domains assigns to the variable the list of IDs of the domains picked in the Display window.

Pick Blocks assigns to the variable the list of IDs of the blocks picked in the Display window.



Modes

Glyph is modal. Certain commands are only available after a given mode has been entered. Each command's reference page in the Glyph Reference Manual lists the mode or modes where the command is available. The table on the following slides summarizes all of the modes, which command enters the mode, which command exits the mode, and a brief description of the mode.

Additionally, some commands can be used during text input via Gridgen's Text Input window. Only Glyph commands that perform queries without altering the grid system or requiring a specific mode can be used.

Modes (Continued)

mode	description	begin	end
BLOCK	block creation	`gg::blkBegin`	`gg::blkEnd`
BLOCK_ELLIPTIC	block elliptic solver	`gg::blkEllBegin`	`gg::blkEllEnd`
BLOCK_EXTRUSION	block extrusion solver	`gg::blkExtrusionBegin`	`gg::blkExtrusionEnd`
BLOCK_RE_EXTRUSION	block re-extrusion	`gg::blkReExtrusionBegin`	`gg::blkReExtrusionEnd`
CONNECTOR	connector creation	`gg::conBegin`	`gg::conEnd`
DB_POINT	database point creation	`gg::dbPtsBegin`	`gg::dbPtsEnd`
DB_CURVE	database curve creation	`gg::dbCurveBegin`	`gg::dbCurveEnd`
DB_SURF	database surface creation	`gg::dbSurfBegin`	`gg::dbSurfEnd`
DOMAIN	domain creation	`gg::domBegin`	`gg::domEnd`
DOMAIN_ELLIPTIC	domain elliptic solver	`gg::domEllBegin`	`gg::domEllEnd`

Modes (Continued)

DOMAIN_EXTRUSION	domain extrusion solver	`gg::domExtrusionBegin`	`gg::domExtrusionEnd`
DOMAIN_RE_EXTRUSION	domain re-extrusion	`gg::domReExtrusionBegin`	`gg::domReExtrusionEnd`
EDGE	edge definition	`gg::edgeBegin`	`gg::edgeEnd`
FACE	face definition	`gg::faceBegin`	`gg::faceEnd`
GLOBAL	commands may be called at any time except from a startup script		
MIRROR	block mirroring for analysis software export	`gg::aswMirrorBegin`	`gg::aswMirrorEnd`
RC	startup script		
SEGMENT	segment definition	`gg::segBegin`	`gg::segEnd`
TOP	default mode		
XFORM	transformation	`*CopyBegin`, `*TransformBegin`	`*CopyEnd`, `*TransformEnd`

Tcl Syntax

Commands

A Glyph script is composed of a series of commands. These commands may be native Tcl functions, Glyph-specific functions, or user functions defined elsewhere in the script.

Each line in a Tcl script starts with a command followed by zero or more arguments separated by spaces. Commands terminate by the end-of-the-line or by a semicolon ; enabling multiple commands to be entered on the line:

```
set A 10; set B 10
```

Variables

As noted before, variables are name-value pairs. Values are assigned to variables for substitution in subsequent commands.

Tcl variables do not need to be declared before using them.

Variable names are case sensitive.

As shown above, variables are assigned values with the set command, and their values obtained by prepending $ to their names:

```
set C $A
```

Tcl Syntax (Continued)

Arrays

Arrays are special types of variables used to form more complex data structures. They use string-valued indices to access the multiple values stored under a single variable name. As the index is string-valued, it can be any type of string: a number, a word, a phrase, or any combination. The indices are delimited by parentheses (). As the parentheses are not a grouping mechanism, care should be used in using indices with spaces (or, better yet, avoid using spaces in indices altogether). The syntax for defining an array element is:

```
set array(index) value
```

Once defined as an array, a variable cannot be subsequently used as a regular, non-array variable, i.e., given the above declaration, a subsequent declaration of the form below would result in an error:

```
set array value
```

The index of an array can itself be a variable.

Tcl Syntax (Continued)

Grouping

As noted previously, spaces are used to separate arguments of a command. Multiple words are grouped into a single argument by double quotes “” or braces { }. Double quotes allow variable and command substitution, but braces do not. Braces are typically used when no substitutions are desired or to delay substitution as in a conditional statement or loop.

Braces can also be used to delimit odd variable names, including those containing characters other than letters, numbers, and underscore:

```
set {new var} 5
```

Braces and double quotes include everything between them, including semicolons and new lines. Braces inside double quotes are treated as regular characters. A backslash \ is needed before a double quote or brace to use it as a regular character:

```
set A “Use \” for quote”
```

Tcl Syntax (Continued)

Command Substitution

The value returned by a function can be substituted into a command by enclosing the call in square brackets:

```
set B 81
set root [expr {sqrt ($B)}]
```

Here root is being assigned the value of the square root of B. As Tcl does not support math functions directly, Tcl commands such as expr are required. Note the use of the braces to group the argument for expr.

Functions can be nested. In such instances, the most deeply nested commands are performed first. Commands at the same nesting level are evaluated from left to right.

Comments

A comment is indicated by inserting a pound character # at the beginning of a line. The comment is continued until the end of the line or can be carried onto the next line by using a backslash \. Semicolons will not terminate a comment, but they can be used to terminate a command so that a comment can be added on the same line:

```
set B 81; # Set the value of B to 81
```

Glyph Conventions

Function Naming

All Glyph commands begin with either `gg::` or `ggu::` to avoid possible naming conflicts with standard Tcl commands. Commands starting with `gg` are the main routines that correspond to Gridgen commands, while those beginning with `ggu` are generally supporting routines that have no direct correlation to functionality within Gridgen.

Glyph commands follow the form:

`gg::<noun><verb>` *`?entity_ids? ?arguments?`*

where `<noun>` is the type of object or entity that is the subject of the action and `<verb>` is the action itself. The first argument after the command's name is the list of entities to be operated on, followed possibly by additional flags or arguments. For commands that can operate on more than one entity, `All` can be used to collectively indicate all appropriate entities as the argument.

Here are the <noun> values used in Glyph commands:

`asw`	analysis software settings				
`blk`	blocks	`con`	connectors	`face`	block faces
`db`	database entities	`disp`	display settings	`seg`	connector segments
`def`	default settings	`dom`	domains	`tol`	tolerance settings
		`edge`	domain edges	`xform`	transformations

Glyph Conventions (Continued)

Entities

Each individual entity of the four major types in Gridgen (Connectors, Domains, Blocks, and Database entities) has a unique identifier. Sub-blocks and sub-domains also receive unique identifiers. These identifiers are obtained either as return values when creating the entity, loading the entity, or through direct query. For example, a list of all connector identifiers in a model could be obtained with:

```
set conIDList [gg::conGetALL]
```

Identifiers should never be hardcoded in a script. There is no guarantee that they will remain the same across Gridgen releases.

Glyph Conventions (Continued)

Points

Gridgen points have two forms:

Free points consist of lists of three floating-point values representing X, Y, and Z coordinates, e.g.,

```
set pt [list 1.0 2.0 3.0]
```

Database constrained points consist of two floating-point values (normalized U and V coordinates) and a third value representing the database entity identifier. The UV values are always between 0 and 1, inclusive.

Some Glyph commands can take either form with an appropriate flag, but most commands require XYZ coordinates. Fortunately, `gg::dbUVToXYZ` can convert from UV to XYZ coordinates:

```
set db_pt [list 0.5 1.0 $db_curve]
gg::conSetBreakPt $con [gg::dbUVToXYZ $db_pt]
```

Conversely, `gg::dbSurfClosestPt` will convert XYZ to UV coordinates:

```
set xyz [list 1.0 2.0 3.0]
set db_pt [gg::dbSurfClosestPt ALL $xyz]
```

If the point xyz is on the surface of any enabled database entity, the returned value will be that point in UV form. Otherwise, the command will return the closest point on an enabled database entity. To find the point or closest point projection on a specific database entity, that entity's identifier replaces *ALL*.

Control Structures

If Then Else

Conditional statement of the form:

`if` *`expression`* `?then?` *`body1`* `?else?` *`body2`*

If `expression` is true, then the command listed in *`body1`* is executed, otherwise that in *`body2`* is executed. The `then` and `else` are optional. Multiple conditionals can be chained together with `elseif`. Typically, braces are placed around the command bodies and conditionals to avoid confusion. Here's an example:

```
if {$sp == 0} {
  set spacing 0.5
} elseif {$sp < 0} {
  set spacing 1.0
} else {
  set spacing $sp
}
```

Control Structures (Continued)

Switch

Conditional statement of the form:

```
switch -flags value pattern1 body1 pattern2 body2 ...
```

If `value` matches a `pattern`, the corresponding command `body` is executed. Any number of `pattern-body` pairs can be supplied, but only the command `body` of first matching `pattern` is used. If the last `pattern` is `default`, then the associated `body` is executed if no patterns match. If a `body` is composed of a dash -, then execution falls to the next `body`. Different `flag` values control how `value` is matched:

`exact` matches the value exactly (default).

`glob` uses glob-style pattern matching (* and ?).

`regexp` uses regular expression pattern matching.

- means no flags or end of flags. It appears as - - and is needed when value can start with -.

Here's an example:

```
switch –glob -- $filename {
  *.igs-
  *.igs { set format IGES }
  *.stl { set format STL }
  default { set format DBA }
 }
```

Control Structures (Continued)

While

Conditional statement of the form:

```
while expression body
```

While the `expression` is true, the command listed in `body` is executed. Since expression is repeatedly evaluated, it is important to place it in braces to defer substitution until the proper time. Otherwise, the expression will be evaluated only initially, possibly resulting in an endless loop.

Here's an example:

```
set ds $spacing
while {$ds > 0.5} {
  set ds [expr {0.5 * $ds}]
}
```

Control Structures (Continued)

Foreach

Conditional statement of the form:

```
foreach var values body
```

This command loops over `body` once for each value in the list of `values` with `var` taking on successive values in the list. It is recommended that `list` be used to define the list of `values` (double quotes can be used, but may lead to unexpected results). Here's an example:

```
foreach db [list $db_1 $db_2 $db_3] {
  gg::dbEnable $db
}
```

Control Structures (Continued)

For

Conditional statement of the form:

```
for start test next body
```

The `start`, `next`, and `body` arguments are commands. The `test` argument is an expression. The command begins with `start`. While `test` is true, `body` is executed followed by `next`. This is repeated until `test` is false.

```
set value 0
for {set j 0} {$j < 10} {incr j} {
  set value [expr {$value + $j}]
}
```

The `incr` command used here increments `j` by 1 – an optional second argument for `incr` could be used to specify a different increment.

Break and Continue

Loop execution can be immediately exited with `Break`, while `Continue` skips the remainder of the current loop and begins the next iteration.

Procedures

New functions, or procedures, can be defined within a script using the following:

```
proc name args body
```

The new function is called `name`, replacing any previous function called `name` (caution is advised). The name is case sensitive. The formal parameter list, `args`, is next, and the actions to be executed by the function follow in `body`. If a value is to be returned from the procedure, the `return` command is used. Here's an example:

```
proc xVal { pt } {
  set x [lindex $pt 0]
  return $x
}

set ptA [list 1.0 2.0 3.0]
set x [xVal $ptA]
```

Here, `xVal` is given a list of values representing the XYZ coordinates of a point. It uses the `lindex` function to extract the first element in the series (Tcl starts numbering list elements at 0) and returns the value. In the second section, `ptA` is defined, and *x* is then defined with the results of `xVal`. In this case, it will end up with the value of 1.0.

Procedures (Continued)

Procedures have a separate scope from the rest of a script, meaning that variables defined within a procedure are not shared outside of the procedure unless they are explicitly defined as `global` variables. Conversely, a variable defined outside of a procedure cannot be used within unless it is passed into it or defined as a `global` variable. The result is that variables sometimes need to be shared inside and/or outside of a procedure.

The `global` command can do this:

```
global varname1 varname2
```

defining variables named `varname1`, `varname2`, etc. The `global` command has to appear in each scope where access to the same variable is desired. Hence, for a variable to be shared in both inside and outside a procedure, the corresponding `global` command must also appear inside and outside of that procedure.

Lists

Tcl lists are simply sequences of values and can be established in a variety of ways:

```
set list1 [list a b 1.0]
set list2 “a b 1.0”
set list3 {a b 1.0}
set list4 {a b {1.0 2.0 3.0} {c d}}
```

Note that the last example above has lists inside the list. The first example is preferred as it is somewhat more efficient.

Elements of a list are accessed via the `lindex` command (remembering that Tcl starts element numbering with 0 – hence the third element has the index of 2):

```
set List [list a b c d e]
# Get the value ‘c’
set el [lindex $List 2]
# Get the length of the list
set len [llength $List]
```

The `llength` command gives the length of a list.

Filenames

As file conventions on UNIX and Windows platforms are different, Tcl provides for file naming and handling in a cross-platform manner. The file command has numerous options for operating on file names. Here are a few:

`file dirname`	`name`	Returns the parent directory of the file `name`
`file exists`	`name`	Returns 1 if `name` exists, 0 otherwise
`file extension`	`name`	Returns the extension of file `name`, including the dot
`file join path`	`path...`	Joints pathname components into a new pathname
`file nativename`	`name`	Returns the platform-native version of `name`
`file readable`	`name`	Returns 1 if `name` has read permissions, 0 otherwise
`file rootname`	`name`	Returns all but extension of `name` (including directory)
`file tail`	`name`	Returns the last pathname component of `name`
`file writable`	`name`	Returns 1 if `name` has write permissions, 0 otherwise

Error Handling

Errors occasionally occur, e.g., when invalid values or an improper number of arguments are passed to a function. If uncaught, the error will cause the termination of the script's execution. The `catch` command can be used to trap and handle errors:

```
catch command ?result?
```

The `command` argument is a body of commands to be executed. If the commands return an error, `catch` returns a non-zero value. If the optional `result` argument is present, it names a variable which is set to the error message. If the commands execute successfully, `catch` returns a zero value and, if present, the `result` argument is set to the command's return value. Here's an example:

```
if {[catch {command arg1 ...} result]} {
  puts "error: $result"
  return
} else {
  # no error, continue on with execution, result
  # contains return value
}
```

An error condition is raised in a function with the `error` function:

```
error ?message? ?info? ?code?
```

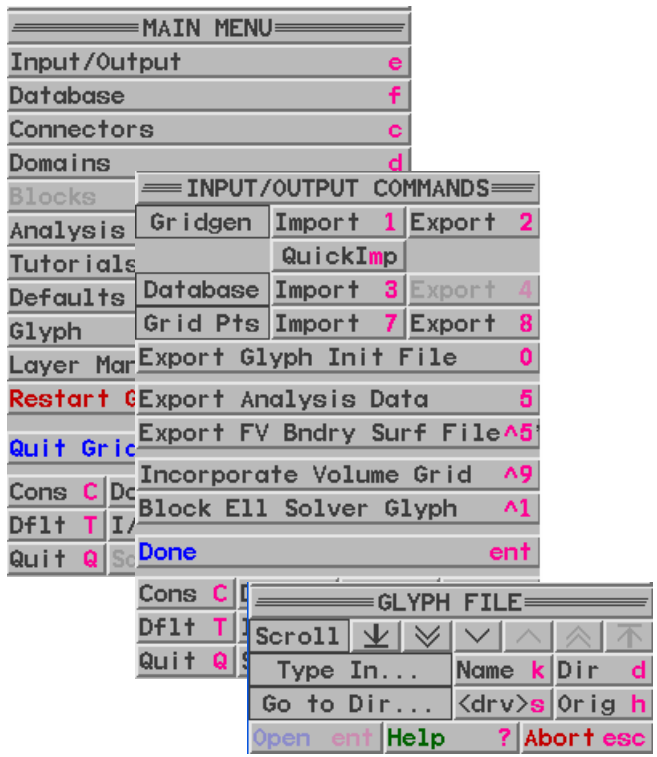
The message argument is a string that would be stored in the result variable of `catch`.

Initializing Gridgen

Placing the script file *_gridgenrc* (on Windows platforms) or *.gridgenrc* (on Unix platforms) in the home or root folder, respectively, or setting the environment variable *GRIDGEN_RCFILE* to the path/name of the Glyph script will lead to the script's execution on Gridgen startup.

A quick way to create such a file is by selecting **Export Glyph Init File** in the **INPUT/OUTPUT COMMANDS** menu. This opens the **GLYPH FILE** submenu where the new file to be created or existing file to overwrite can be selected. The *_gridgenrc* or *.gridgenrc* file created here will contain all of the default and display command settings from the current e Gridgen session.

The file can be subsequently edited as desired.



Initializing Gridgen (Continued)

```
# Gridgen (V.15.08 REL 1) Initialization Script
# Written Mon Jan 1 00:00:00 2000
#
# This file will be automatically loaded if named "_gridgenrc"
# and located in the working directory or your home directory.
#

# DEFAULTS
gg::defDbCoordDisp UV
gg::defCFBG TM
gg::defCFFG HILGENSTOCK
gg::defConDistBeg 0.0
gg::defConDistEnd 0.0
gg::defConDistFunc TANH
gg::defConMaxAng 0.0
gg::defConMaxDev 0.0
gg::defConSurCrv 0
gg::defQuickSave "quicksave.gg"
gg::defUnsIpts 1
gg::defUnsMaxAng 0.0
gg::defUnsMaxDev 0.0
gg::defUnsMaxMem 64
gg::defUnsPyrAR 0.5
gg::defUnsPyrMax 0.0
gg::defUnsPyrMin 0.0
gg::defUnsTetDecay 0.5
gg::defUnsTriBndDecay 0.5
gg::defUnsTetBndDecay 0.5
gg::defUnsTetMax 0.0
gg::defUnsTetEdgeMax 0.0
gg::defUnsTetMin 0.0
gg::defUnsTetEdgeMin 0.0
gg::defUnsTriDecay 0.5
gg::defUnsTriMax 0.0
gg::defUnsTriEdgeMax 0.0
gg::defUnsTriMin 0.0
gg::defUnsTriEdgeMin 0.0
gg::defJoinAng 0.0
gg::defSplitAng 0.0
```

Here is a typical *_gridgenrc* file. Indeed, with a couple of exceptions, all settings are at their normal defaults.

As with most programs, it begins with a header of comment statements, including a time stamp of when Gridgen wrote the file.

Next the default settings are listed. Notice that most, if not all, of the Glyph equivalents can be readily deduced, e.g., `gg::defConDistBeg` is the same as the **Con Dist Bgn Δs** command found in the **SET DEFAULT VALUES** menu.

Should the values need to be changed, this file can be edited directly or a new file simply generated from within Gridgen.

Initializing Gridgen (Continued)


```
gg::defImportVisibility 0
gg::defConDim none

# TOLERANCES
gg::tolCon 0.0001
gg::tolNode 0.0001
gg::tolGP 1e-007
gg::tolModelSize 1000.0

# DISPLAY
gg::dispBGColor WHITE
gg::dispBell 0
gg::dispBodyAxes 0
gg::dispColor 0 "#E63C87"
gg::dispColor 1 "#005F00"
gg::dispColor 2 "#FF4500"
gg::dispColor 3 "#0000AF"
gg::dispColor 4 "#FFD700"
gg::dispColor 5 "#7D0000"
gg::dispColor 6 "#9100FF"
gg::dispConGPS 0
gg::dispDemote 0.0
gg::dispDisabledDB 0
gg::dispDisabledGrid 1
gg::dispDoublePrecision 0
gg::dispEnabledDB 1
gg::dispFrameBuffer RGB
gg::dispGlide 1
gg::dispLeftPick 0
gg::dispLight "0 0 0"
gg::dispLineWidth 1
gg::dispMenuSide LEFT
gg::dispMouseWarp 1
gg::dispQuickMenu 0
gg::dispSmallText 0
gg::dispViewProj PERSPECTIVE
gg::dispViewCons 1
gg::dispViewDoms 1
gg::dispViewNodes 1
gg::dispXYZAxes 0
```

Here is the remainder of the *_gridgenrc* file from the previous slide.

Notice how the default settings wrap up by setting the default connector dimensions, `gg:defConDim`.

Next, some key tolerances are set next, again repeating parameters that can be set via the GUI, in this case in the **SET TOLERANCE VALUES** submenu of the **SET DEFAULT VALUES** menu.

Finally, the display settings can be fixed. In this particular file, two settings have been changed from the defaults, `gg::dispBGColor` (changed from `BLACK` to `WHITE`) and `gg::dispBodyAxes` (changed from `1` to `0`, i.e., on to off).

Project Start-Up

A Glyph script can be used to simplify the start-up process by having the script clear out old information and import the files needed for the project.

Here is an example of a start-up script that does exactly this:

```
# Gridgen Journal File V1 (Gridgen 15.08 REL 1)
# Created Mon Jan 1 00:00:00 2005
#
package require PWI_Glyph 1.6.8

# Delete any existing grids and database entities. Reset AS/W, defaults, and
# tolerances.
gg::tolReset
gg::defReset
gg::aswDeleteBC -glob "*"
gg::aswDeleteVC -glob "*"
gg::aswSet GENERIC -dim 3

# Read database and grid files.
gg::dbImport "C:/Program Files/Pointwise/GridgenV15/My Grids/gridgen.dba" -type DBA
gg::gridImport "C:/Program Files/Pointwise/GridgenV15/My Grids/gridgen.gg"
```

Banner comments indicating Gridgen version and script creation date.

This series of Glyph commands performs the same operations as the **Restart Gridgen** command in the **MAIN MENU** with all options picked.

This section reads in the new composite database, called *gridgen.dba* here, and Gridgen restart, called *gridgen.gg* here, files.

Project Start-Up (Continued)

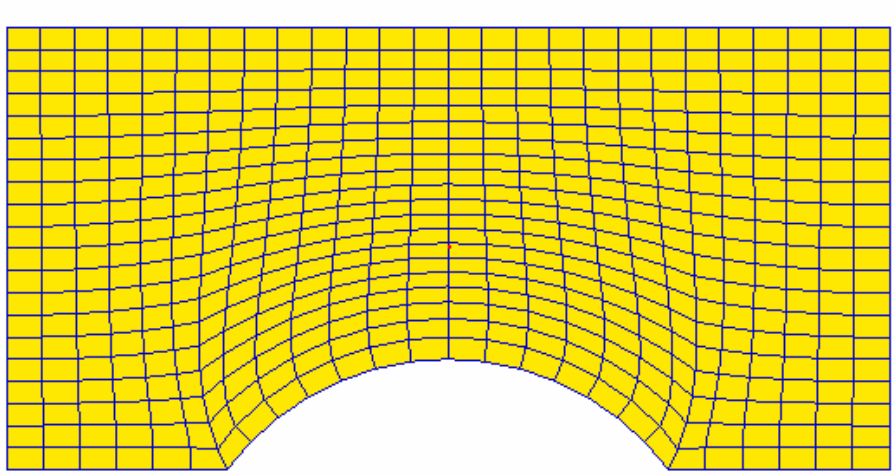
Actual file paths can be troublesome if the script and files are moved to another location. Therefore, it's helpful to make scripts independent of a specific path, as in the rewritten version of the start-up script below:

```
# Gridgen Journal File V1 (Gridgen 15.08 REL 1)
# Created Mon Jan 1 00:00:00 2005
#
package require PWI_Glyph 1.6.8

# Delete any existing grids and database entities. Reset AS/W, defaults, and
# tolerances.
gg::tolReset
gg::defReset
gg::aswDeleteBC -glob "*"
gg::aswDeleteVC -glob "*"
gg::aswSet GENERIC -dim 3

# Read database and grid files.
gg::dbImport [file join [file dirname [info script]] "gridgen.dba"] -type DBA
gg::gridImport [file join [file dirname [info script]] "gridgen.gg"]
```

This section no longer calls out a fixed path to the files to be imported. Instead, `info script` returns the complete path and name of the script file being executed, while `file dirname [info script]` strips away the script file's name and returns just the path. It's then left for `file join` to combine this path with the dba or gg file listed.

Anatomy of a Script

Of course, Glyph scripts can also be used to build entire grids. To illustrate this, a script which builds the familiar mesh shown here will be examined. This script, as well as other grid generation examples, is provided as part of the standard Gridgen installation in the examples folder. In addition, some utility scripts are located in the utils folder. Finally, a repository of even more useful scripts is maintained at:



http://www.pointwise.com/glyph/

The files contained in this “Glyph Exchange,” besides being useful tools, are also useful instructional aids.

Anatomy of a Script (Continued)


```
#
# Copyright 2002 (c) Pointwise, Inc.
# All rights reserved.
#
# This sample Gridgen script is not supported by Pointwise, Inc.
# It is provided freely for demonstration purposes only.
# SEE THE WARRANTY DISCLAIMER AT THE BOTTOM OF THIS FILE.
#

##########################################################################
# Glf "Low Level API" script for the Bump on a Wall tutorial.
# Structured grid version.
##########################################################################

#  G----------------------F
#  |                      |
#  |                      |
#  |                      |
#  |          H           |
#  |         / \          |
#  |        /   \         |
#  B-------C     D--------E
#
#             A

# Clear out any previous grids
gg::memClear

# Reset the default values to avoid surprises
gg::defReset
```

The script starts off with a comment block indicating where it came from and what it does. A cartoon diagram mimicking the diagram accompanying the original tutorial shows the basic geometry and defines the names of the nodes that will be used to define it.

The command `gg::memClear` removes any previous information out of memory to avoid any possible conflicts.

Just to be safe, `gg::defReset` is invoked to restore Gridgen's defaults.

Anatomy of a Script (Continued)

```
set A "  0 -8 0"
set B "-20  0 0"
set C "-10  0 0"
set D " 10  0 0"
set E " 20  0 0"
set F " 20 20 0"
set G "-20 20 0"
set H "  0  5 0"

# set number of grid points on connectors
set numgp_BG 21
set numgp_EF $numgp_BG
set numgp_BC 7
set numgp_CD 15
set numgp_DE 7
set numgp_FG [expr $numgp_BC + $numgp_CD + $numgp_DE - 2]

###################
# CREATE CONNECTORS
###################

### CD
gg::conBegin
  # Circular arc segment
  gg::segBegin -type CIRCULAR_ARC
    gg::segAddControlPt $C
    gg::segAddControlPt $D
    gg::segAddControlPt $H
  gg::segEnd
set con_CD [gg::conEnd]
# dimension
gg::conDim $con_CD $numgp_CD
```

Next, the nodes that will define the grid's connectors are delimited. Notice that each node's coordinates are grouped within double quotes.

The dimensions for the connectors are now assigned. Since the connector EF must have the same dimension as connector BG, numgp_EF is defined in terms of numgp_BF. Similarly, connector FG's dimension is defined in terms of the dimensions of connectors BC, CD, and DE.

The first connector to be defined will be the circular arc segment CD. Notice that the arc is defined from its two endpoints C and D and a third point, H, between them. Once the curve is defined, it's given the name con_CD and then dimensioned.

Anatomy of a Script (Continued)


```
### BC
gg::conBegin
  # Line segment
  gg::segBegin -type 3D_LINE
    gg::segAddControlPt $B
    gg::segAddControlPt $C
  gg::segEnd
set con_BC [gg::conEnd]
# dimension
gg::conDim $con_BC $numgp_BC

### DE
gg::conBegin
  # Line segment
  gg::segBegin -type 3D_LINE
    gg::segAddControlPt $D
    gg::segAddControlPt $E
  gg::segEnd
set con_DE [gg::conEnd]
# dimension
gg::conDim $con_DE $numgp_DE

### BG
gg::conBegin
  # Line segment
  gg::segBegin -type 3D_LINE
    gg::segAddControlPt $B
    gg::segAddControlPt $G
  gg::segEnd
set con_BG [gg::conEnd]
# dimension
gg::conDim $con_BG $numgp_BG
```

The remaining connectors all consist of single line segments, simplifying their definition. Once told that a connector is being defined with `gg::conBegin`, Gridgen is next told that a `3D_LINE` segment is being added via the `gg::segBegin` command. After the control points at each end are defined, the `gg::segEnd` command is issued to complete the segment definition. The completed connector is then named (`gg::conEnd` not only ends creation, but returns the ID) and dimensioned.

Once the connectors are all made, a little housekeeping is done by reseting the view in the display window.

```
### EF
gg::conBegin
  # Line segment
  gg::segBegin -type 3D_LINE
    gg::segAddControlPt $E
    gg::segAddControlPt $F
  gg::segEnd
set con_EF [gg::conEnd]
# dimension
gg::conDim $con_EF $numgp_EF

### FG
gg::conBegin
  # Line segment
  gg::segBegin -type 3D_LINE
    gg::segAddControlPt $F
    gg::segAddControlPt $G
  gg::segEnd
set con_FG [gg::conEnd]
# Dimension
gg::conDim $con_FG $numgp_FG

# Reset view
gg::dispViewReset
```

Anatomy of a Script (Continued)


```
##########################
# CREATE STRUCTURED DOMAIN
##########################

gg::domBegin -type STRUCTURED
  ### edge BCDE (jmin)
  gg::edgeBegin
    gg::edgeAddCon $con_BC
    gg::edgeAddCon $con_CD
    gg::edgeAddCon $con_DE
  set edge_BE [gg::edgeEnd]
  ### edge EF (imax)
  gg::edgeBegin
    gg::edgeAddCon $con_EF
  set edge_EF [gg::edgeEnd]
  ### edge FG (jmax)
  gg::edgeBegin
    gg::edgeAddCon $con_FG
  set edge_FG [gg::edgeEnd]
  ### edge BG (jmin)
  gg::edgeBegin
    gg::edgeAddCon $con_BG
  set edge_BG [gg::edgeEnd]
set dom_A [gg::domEnd]
```

Next, the structured domain is assembled from the connectors. The process starts with the `gg::domBegin` command, with the `-type` option set to `STRUCTURED` (not necessary actually as this is the default). Each edge is then defined starting with the `gg::edgeBegin` command. The bottom edge BCDE is composed of three connectors (BC, CD, and DE), while the remaining three edges have one connector each. Each connector is added with the `gg::edgeAddCon` command. Each edge is also named when the `gg::edgeEnd` command is issued to finish the corresponding edge (the command returning the edge number of the newly created edge). The domain is similarly named when the `gg::domEnd` command is issued (the command returning the ID of the new domain).

Anatomy of a Script (Continued)

```
#########################
# RUN STRUCTURED SOLVER
#########################

gg::domTFISolverRun $dom_A

gg::domEllSolverAtt $dom_A -bg_control LAPLACE
gg::domEllSolverBegin $dom_A
  set results [gg::domEllSolverStep -iterations 100]
  # Print out the residual results neatly
  puts "iter   residual   max resid"
  foreach {i res maxres} $results {
    puts [format "%4d  %-8.4e  %8.4e" $i $res $maxres]
  }
gg::domEllSolverEnd

############################
# EXPORT PLOT3D GRID FILE
############################
gg::domExport "ALL" "bump.grd" \
  -style PLOT3D \
  -format ASCII \
  -precision DOUBLE

##################
# SET FLOW SOLVER
##################

gg::aswSet "FLUENT V5" -dim 2
```

With the domain formed, the structured solver is called to do some smoothing of the mesh (just as in the tutorial). In this script, the structured domain `$dom_A` is first initialized using TFI (strictly speaking not required here as this was done automatically by the last command `gg::domEnd`). Next, the background control function attribute is set to `LAPLACE` with the `gg:domEllSolverAtt` command. The solver is then told to step through 100 iterations (`gg::domEllSolverStep`) with the residuals "neatly" printed to the screen. The solver is then exited.

The domain is then exported as a `PLOT3D` grid file. Note the use of \ for continuation of the `gg::domExport` options.

The solver is next set with the `gg::aswSet` command to `FLUENT V5`, and the mesh is declared to be two-dimensional.

Anatomy of a Script (Continued)


```
##############
# CREATE BLOCK
##############
gg::blkBegin -type STRUCTURED
  gg::faceBegin
    gg::faceAddDom $dom_A
  gg::faceEnd
set blk_A [gg::blkEnd]

############################
# APPLY AND EXPORT FLUENT BCS
############################

gg::aswSetBC "$con_EF" OUTFLOW
gg::aswSetBC "$con_BC $con_CD $con_DE" WALL
gg::aswSetBC "$con_FG" "PRESSURE FAR FIELD"
gg::aswSetBC "$con_BG" "PRESSURE INLET"

gg::aswExport "bump.cas"

####################
# EXPORT GRIDGEN FILE
####################
gg::gridExport "bump-str.gg"

#
# DISCLAIMER:
# TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, POINTWISE DISCLAIMS
# ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED
# TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
# PURPOSE, WITH REGARD TO THIS SCRIPT.  TO THE MAXIMUM EXTENT PERMITTED BY
# APPLICABLE LAW, IN NO EVENT SHALL POINTWISE BE LIABLE TO ANY PARTY FOR
```

Block creation follows, simplified here as this is a two-dimensional mesh and hence there's only one face. As with connector and domain creation, the corresponding block creation mode has to be entered (`gg::blkBegin`) to start and exited (`gg::blkEnd`) to end.

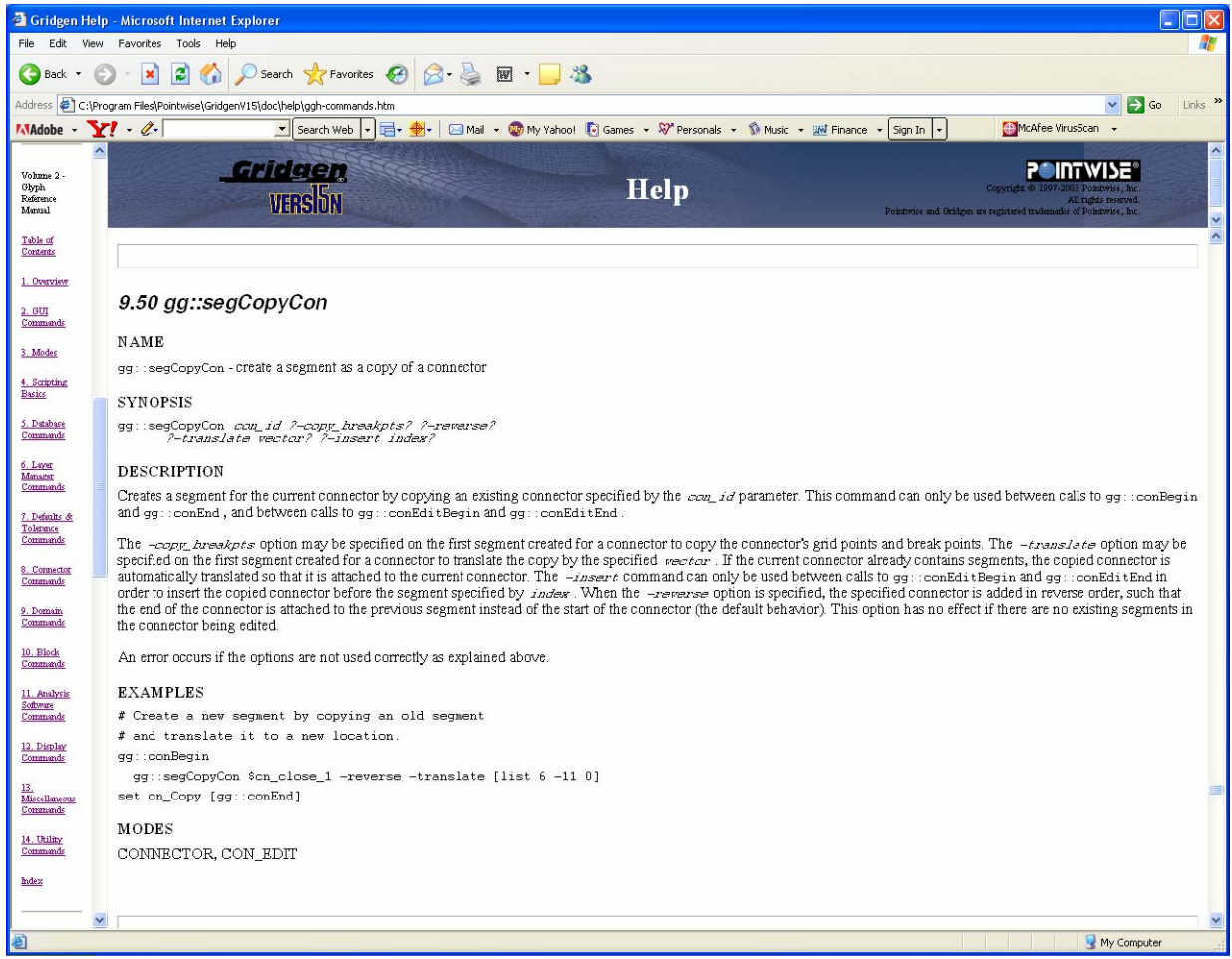
Fluent-specific boundary conditions are next set. Since Gridgen has been previously told that this is a two-dimensional problem, it expects the entities to be connectors.

The final commands, `gg::aswExport` and `gg::gridExport`, write the analysis software and Gridgen restart files, respectively.

Finally, there is a comment block for a "Disclaimer." Unfortunately, space precludes listing it here in its entirety...

Man Pages

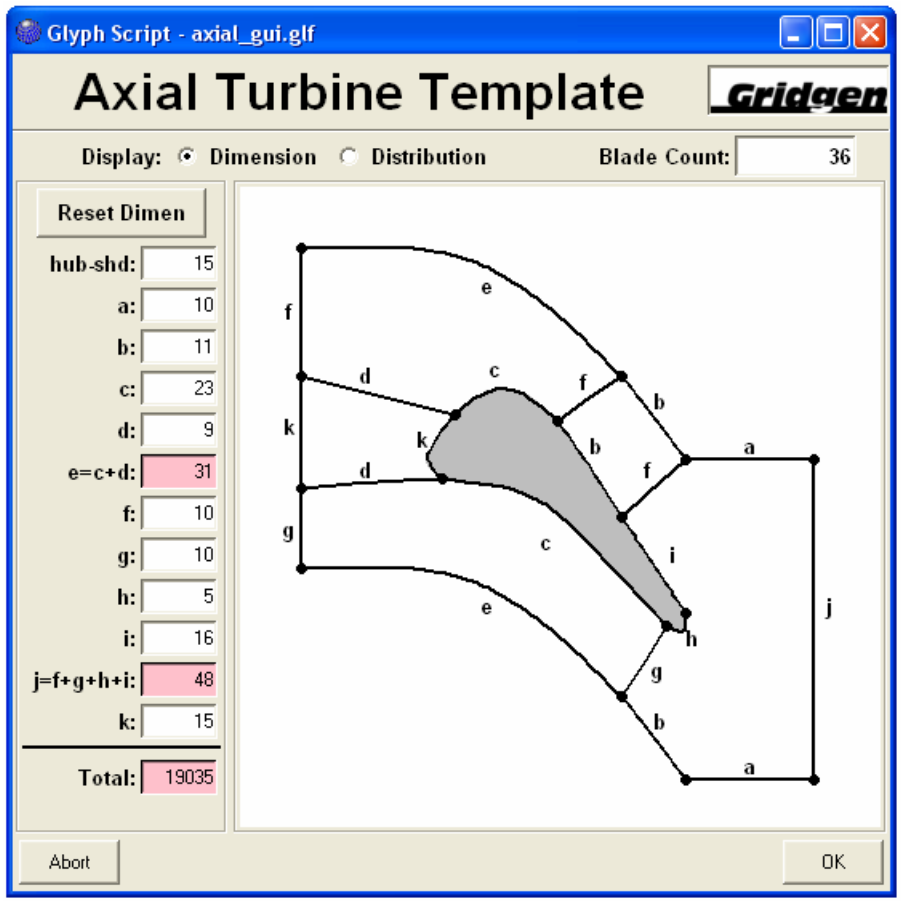
As even this simple example shows, there's a great number of Glyph commands, and each has very specific rules governing its use. A vital resource to learning about and understanding these commands is the Glyph Reference Manual. This is available online through the **Help** button in the GUI (printed copies are also available by order). Contained in this manual are detailed descriptions, instructions, and examples for all Glyph commands.



Tk, Tcl's Graphical Toolkit

Tk is Tcl's graphical toolkit for creating graphical user interfaces, like that shown here. Tk is actually a set of Tcl commands for creating widgets, a window in the interface with a particular appearance and behavior. Examples of widgets include buttons, sliders, text input windows, menus, and the general canvas to place them on.

Delving into the intricacies of Tk is beyond the scope of this course. However, there are a number of examples included with the standard installation of Gridgen and on our website.



Tk, Tcl's Graphical Toolkit (Continued)

Tk was originally developed for X window systems, so Tk widgets are stored in a hierarchy just like X widgets. Thus, there is usually a parent window with several child windows, each of which can have additional children widgets and so on. This hierarchy forms the basis of identifying widgets and arranging them graphically on the screen.

Adding a widget to a hierarchy involves two steps: creation and placement. Each widget type has a specific creation routine, while geometry managers that control a child's size and location relative to their parent govern placement.

Tk programming differs from Tcl in that Tk is event driven while Tcl essentially executes commands serially. A Tk program typically starts with the creation of widgets and definition of routines to be called in response to specific events and then enters an event loop to wait for user action.

There are many references that provide additional details on Tk. These are listed in the Glyph Reference Manual.

Tk, Tcl's Graphical Toolkit (Continued)

Gridgen includes a standalone Tk interpreter, called `ggwish`, for running or testing Tk scripts (note that this interpreter only understands non-Gridgen-specific commands as those commands are not recognized outside Gridgen itself). This interpreter is virtually identical to the standard `wish` interpreter that normally comes with standard Tcl/Tk distributions except for a few bug fixes.

Running a Tk-enabled script in Gridgen differs slightly from running it in `wish` or `ggwish` in that two commands have to be added:

One informs Gridgen that Tk widgets will be used to define an interface other than the default one. This is `gg:tkLoad` which has to be placed before any Tk commands can be issued. This command causes Gridgen to dispose of the default interface and expose the Tk commands to the script.

The other prevents the script from terminating prematurely. When Gridgen normally encounters the end of a script, script execution is terminated and control reverts back to the standard GUI interface. In `wish` (and `ggwish`), an event loop is entered when the end of a script is reached. This facilitates the operation of the Tk program as described on the previous slide. To duplicate this behavior in Gridgen, `gg:tkLoop` has to be called. This puts Gridgen into an event loop until `exit` (exits the script normally), `gg::abort` (exits and reports the script was aborted), or `gg:terminate` (exits script and closes Gridgen) is called.

- In this session:
  - Glyph.
  - Tutorial: “Converging-Diverging Nozzle: Using Glyph Scripting.”