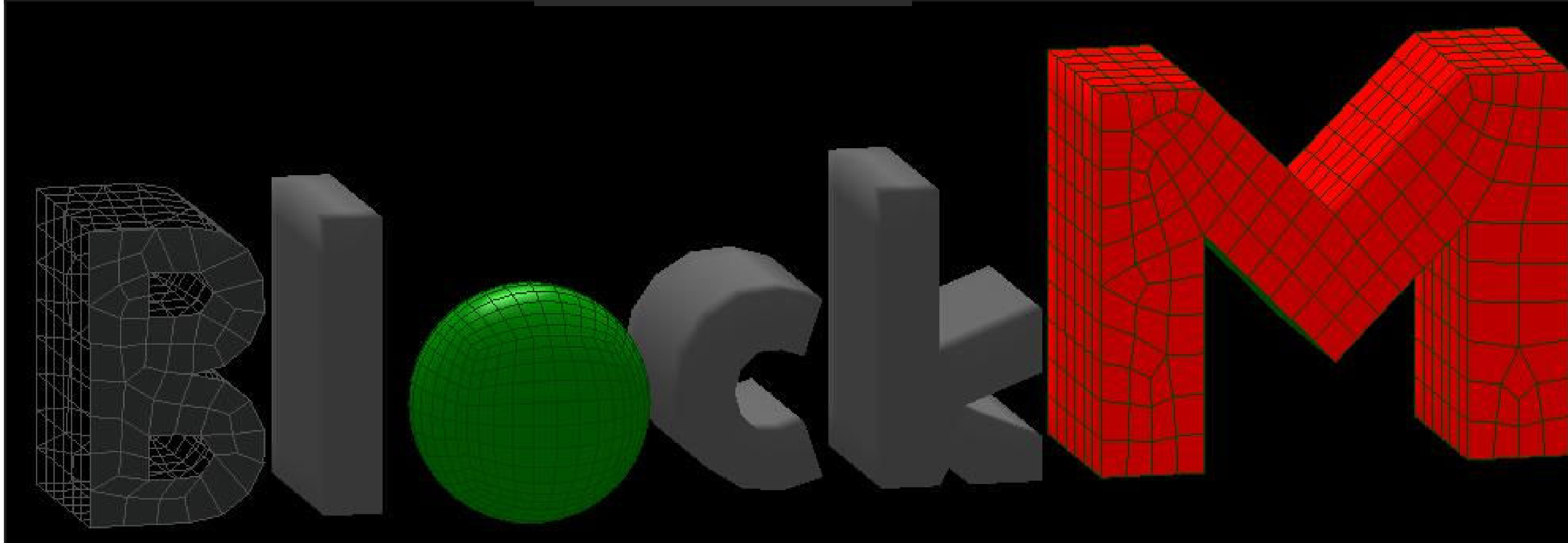
LSTC
Livermore Software Technology Corporation



Multiple Tutorial

BlockM: Table of Content

- Overview ……………………………………………7
- What is Index? ……………………………………..8
- Fundamental of Multiple
  - Index, Indices, Partition, and Region …………………9
  - Function of Index ……………………………………..13
- Initialization: Block Mesh
  - Relationship between node, element, and index ……...14

BlockM: Table of Content

- Concept of Projection
  - Projection ……………………………………………..20
- Windows
  - Main Window ……………………………………...24
  - Computation Window ……………………………...25
    - View menu ……………………………………….26
    - Misc menu ………………………………………..32
    - Selections …………………………………………34

BlockM: Table of Content

- Start Using BlockM

  - Features Overview 38
  - Create 39
  - Blank/Del 52
  - Move Pts. 57
  - Rotate Pts. 66

BlockM: Table of Content

- Start Using BlockM

  - Project ……………………………………………...76
  - Distribute ……………………………………….....101
  - Parameter ……………………………………......114
  - Read/Write ……………………………………….117

BlockM: Multiple

- Overview
  - Although there are lots of mathematics which go into a mesh generator, meshing in general is an art form. This is especially true for blockM. With blockM, a user starts out with a block of elements. Then he would delete, move, and project nodes and elements in the block to form a desired shape. This is analogous to a sculptor starting out with a block of clay, then needling, cutting, and shaping it into a beautiful work of art.
  - BlockM is a powerful mesh generator for hexagonal elements. The basic principals are derived from LS-INGRID. Like LS-INGRID, this is still a purely "manual" mesh generating method. However, the underlying core and user's interface are not same. The user's interface is designed to provide a highly user friendly environment, at the same time allowing the user to generate complex hex meshes efficiently.

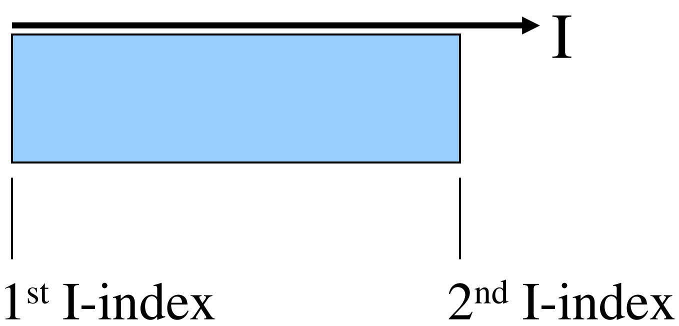
    For users of LS-INGRID and even TRUEGRID, BlockM has virtually no learning curve. For some novice users, however, blockM might be a challenge. The challenge arises from 2 relatively abstract principles known as "index spacing" and "projection". With some practices, any user can become highly proficient at generating mesh with the BlockM.
  - Once a user mastered BlockM, there is virtually nothing that he cannot not mesh.

BlockM: Multiple

- What is Index?
  - Index or indices are divisions or cuts in the "**block**" in the I, J, and K directions. These are synonymous with the X, Y, and Z directions respectively in the Euclidean or Cartesian coordinates. In blockM, the I, J, and K will always refer to directions in index space and they are not be confused with the Euclidean space.

BlockM: Fundamental of Multiple

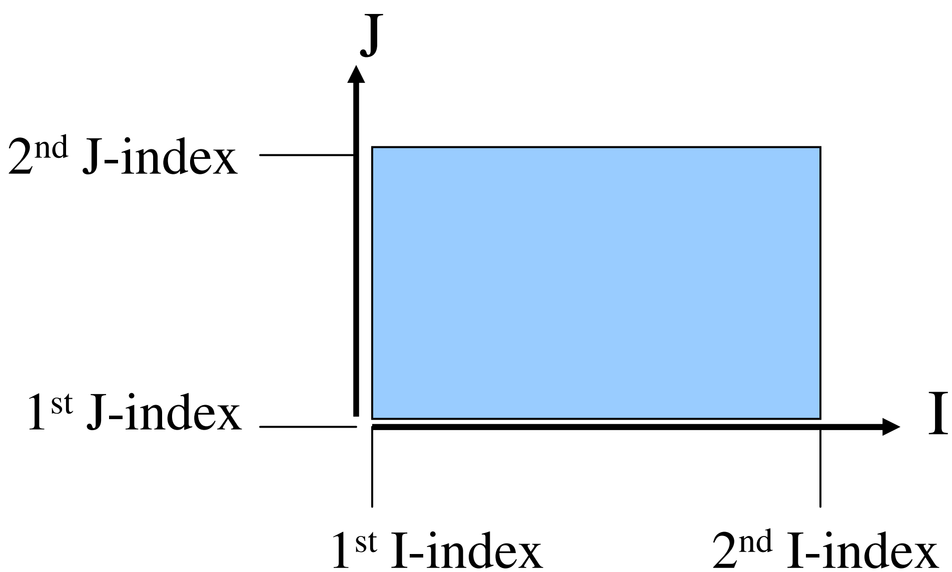
- Index, Partition, and Region
  - Example (1.0): 1D block in Index Space



  - From the example above, the edges are considered index points. It also shows that the minimum number of indices for any block in each of the index direction are 2.
  - The space between the index points is called partition or region.
  - Partition refers to a single unit of space between 2 indices.
  - Region refers to one or more partitions.

BlockM: Fundamental of Multiple

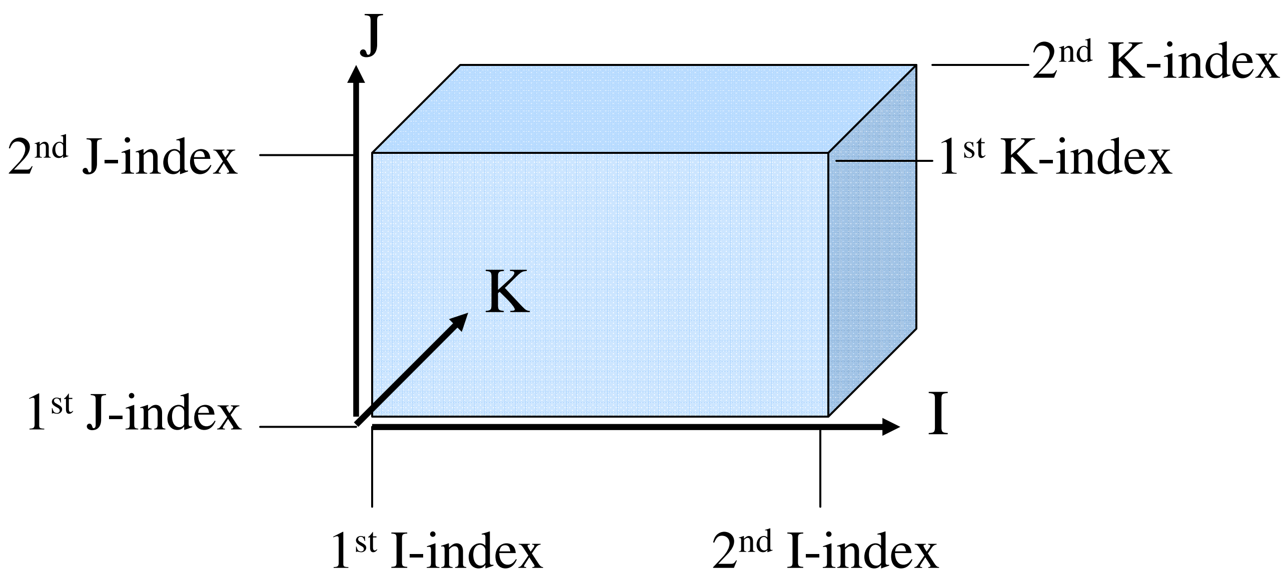
- Index, Partition, and Region (continued)
  - Example (2.0): 2D block in Index Space



  - Note the order of I and J indices. The first indices always correspond to the lowest positions of X and Y in the Euclidean space. Likewise, the second indices correspond to the higher positions in the Euclidean space.

BlockM: Fundamental of Multiple

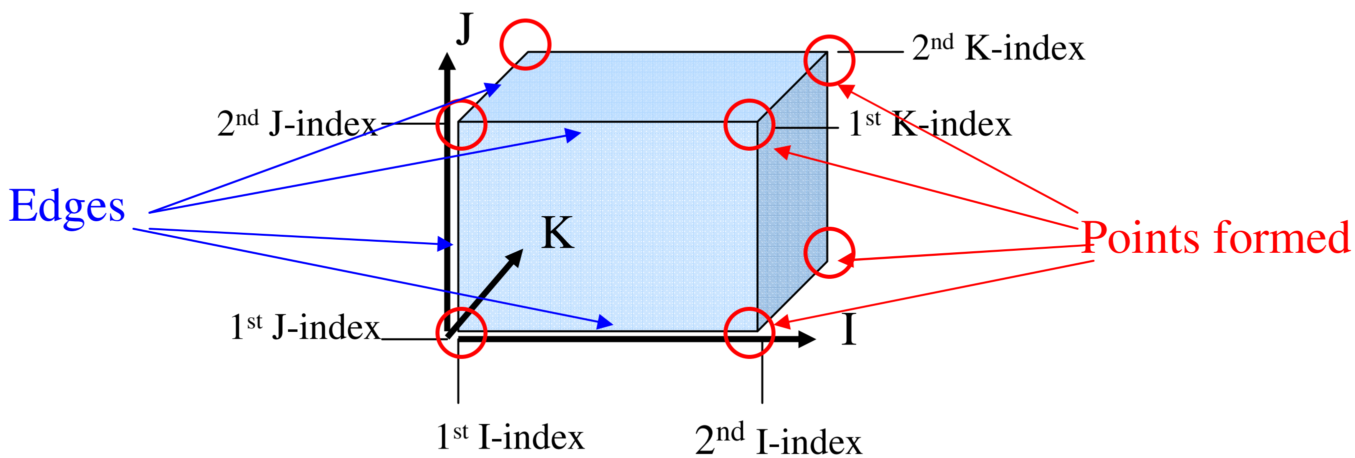
- Index, Partition, and Region (continued)
  - Example (3.0): 3D block in Index Space



  - Note: there is only 1 partition in all 3 index directions.

BlockM: Fundamental of Multiple

- Index, Partition, and Region (continued)
  - The intersection of 2 orthogonal indices form edges or lines.
  - The intersection of 3 orthogonal indices form a point or vertex.
  - A **face** is comprised 4 connecting edges.

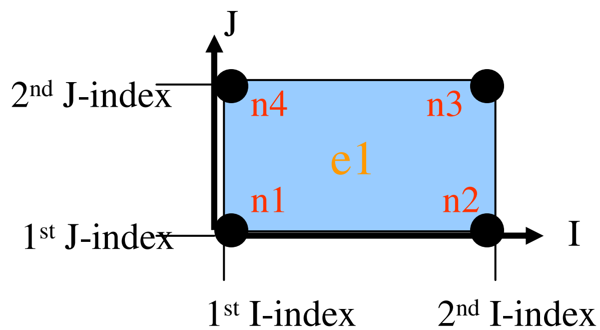


BlockM: Fundamental of Multiple

- Function of Index
  - To achieve a desired shape, the user will have to either move or project nodes and elements to surfaces. In BlockM and LS-INGRID alike, the users do not have direct control of the nodes and elements in the block. The user will have to select the appropriate points, edges, and indices to move or project. This concept will become clearer with examples and further discussions later on.
  - The bottom line:
    - Indices, points, edges, faces, and regions are used for controlling the shape of the mesh.

BlockM: Initialization

- Block Mesh
  - Relationship between node, element, and index
    - Nodes and elements are the basis of any finite element mesh. Now that the concept of index is touched upon, the user would want to know how this is related to nodes and elements.
    - Example (4.0): Let's look at the 2D block example again.

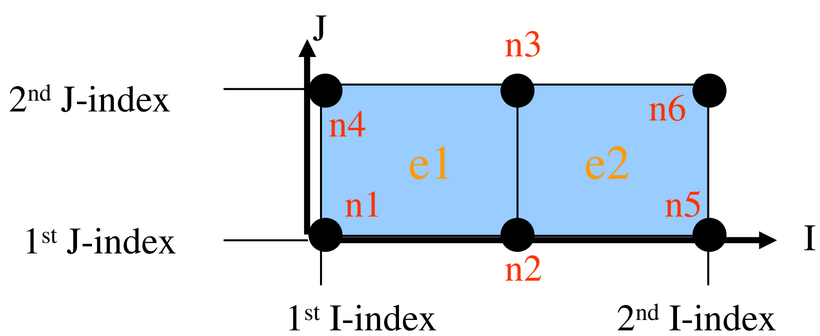


Node #	X-coord.	Y-coord.
1	0	0
2	1	0
3	1	1
4	0	1

  - For a single shell or a 2D element (e1), it is comprised of nodes 1, 2, 3, and 4, represented by n1, n2, n3 and n4.
  - To generate this element, note where the indices are: I-1 at x=0, I-2 at x=1, J-1 at y=0, and J-2 at y=1. This is just an example, the initial position of the indices could be anywhere. They could be the at the same position; let's say I-1 and I-2 could both be at x=0 or J-1 and J-2 at y=1,2,3… etc. However, it is good practice to have the initial index position to be as close the desire shape as practically possible.
  - As seen above, the indices form regions, which in turn contains the element or group of elements.

BlockM: Initialization

- Relationship between node, element, and index (continued)
  - Example (4.1): Let's expand the previous example and add one more element in the I-direction.

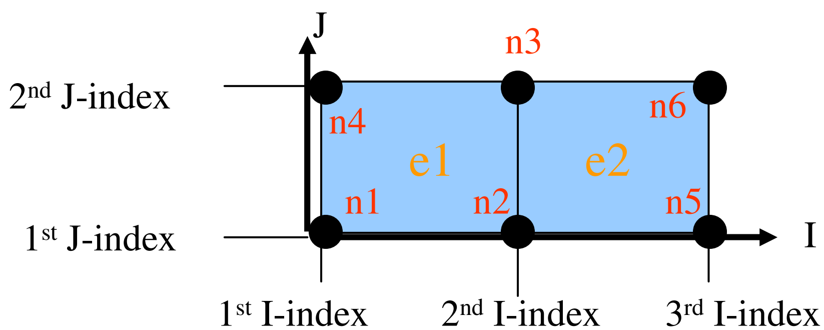


Node #	X-coord.	Y-coord.
1	0	0
2	1	0
3	1	1
4	0	1
5	2	0
6	2	1

- Here the 2nd I-index is moved to x=2. This is for visual demonstration purpose. To show that as more elements are added to a partition, they are automatically divided evenly. The 2nd I-index does not necessarily have to be at x=2, but could be at x=1 instead. In which case, n2 and n3 would be at x=.5.
- In this case, x-length of e1 is equaled to e2, which is equaled to 1.
- If we have 4 elements instead of 2, x-length of e1 is equaled to those of e2, e3, and e4, which all equaled to .5.
- It is important to note that the partitions bounded by the I and J indices contain 2 elements

BlockM: Initialization

- Relationship between node, element, and index (continued)
  - Example (4.2): Let's continue with previous example and add 1 more partition in the I-direction.

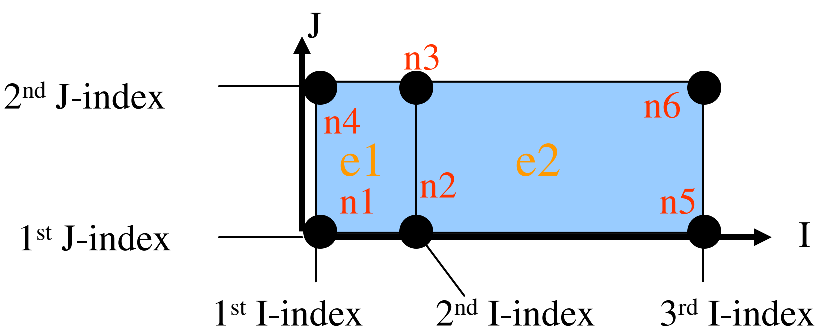


Node #	X-coord.	Y-coord.
1	0	0
2	1	0
3	1	1
4	0	1
5	2	0
6	2	1

- When adding another partition, notice the indices lie right at the boundaries of the elements. Also, notice there are 3 indices in the I-direction and only 2 indices in the J-direction. The order of the indices in the I-direction is for left to right and the bottom to top in the J-direction.
- In this example, the 2nd I-index is set to x=1. Thus, it is no different from the previous example in regards to node and elements. It could well be set anywhere between 0 and 2.
- Because there are 2 elements in I-direction and 1 in the J-direction, it becomes apparent that the number of partition cannot be greater than the number of element for any direction. Also, the number indices is equaled to the number of partition plus 1.

BlockM: Initialization

- Relationship between node, element, and index (continued)
  - Example (4.3): Let's continue with previous example. This time we set the 2nd I-index to x=.5

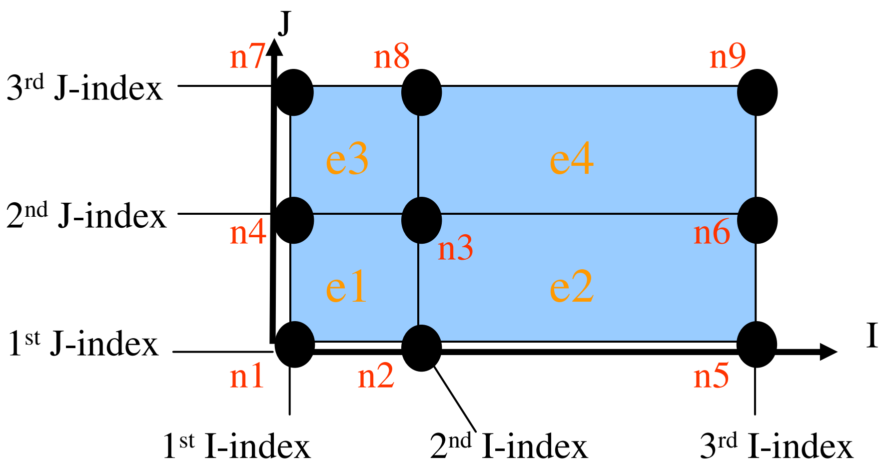


Node #	X-coord.	Y-coord.
1	0	0
2	.5	0
3	.5	1
4	0	1
5	2	0
6	2	1

- This example demonstrates how indices are used to control the size and position of nodes and elements.
- Because the 2nd I-index is set at x=.5, element # 1 is 1/3 the size of element # 2.
- Notice nodes 2 and 3, they follow where ever the 2nd I-index goes. Likewise, nodes 5 and 6 will follow the 3rd I-index.

BlockM: Initialization

- Relationship between node, element, and index (continued)
  - Example (4.4): Let’s add 1 more partition in the J-direction this time.

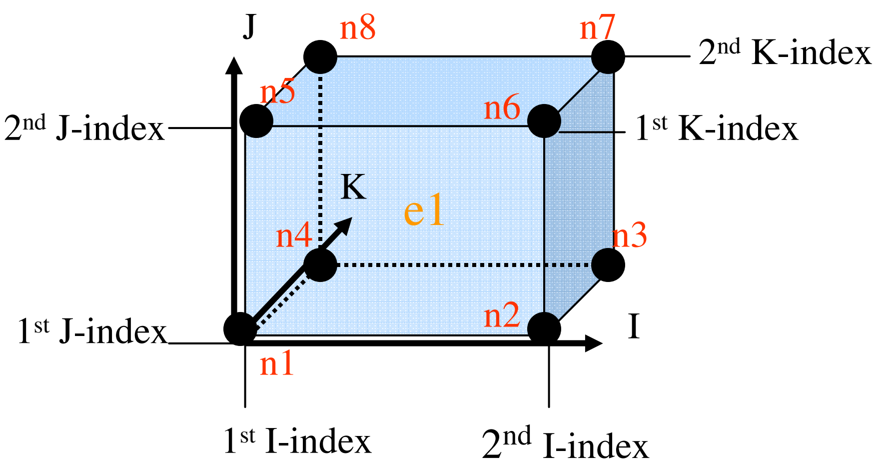


Node #	X-coord.	Y-coord.
1	0	0
2	.5	0
3	.5	.5
4	0	.5
5	2	0
6	2	.5
7	0	1
8	.5	1
9	2	1

- As stated before, the number of partition cannot be more than the number of element in that direction. Since the J-direction has only 1 element, we need to add another element in order to insert a partition.
- Here, we set the 2nd J-index at y=.5, and the 3rd J-index takes the place of the 2nd J-index in the previous example.

BlockM: Initialization

- Relationship between node, element, and index (continued)
  - Example (5.0): Let's look at a 3D example.

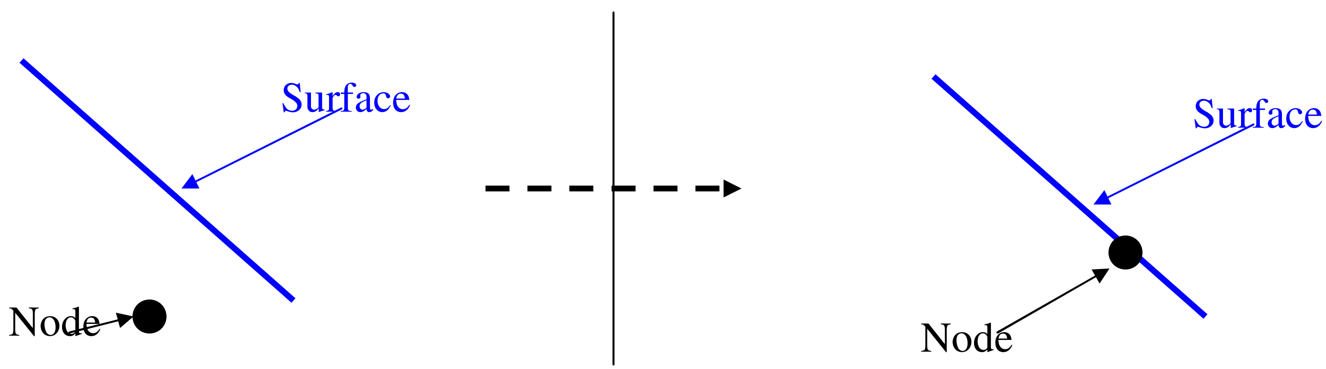


Node #	X-coord.	Y-coord.	Z-coord.
1	0	0	0
2	1	0	0
3	1	0	1
4	0	0	1
5	0	1	0
6	1	1	0
7	1	1	1
8	0	1	1

- Going from 2-D to 3-D is quite trivial, all we have to do is add an element and a partition in the K-direction.
- Again, it is important to note the order of the indices for each principal direction. The I-direction start from left to right. This means the I-1 index is at a lower x-position than I-2 index. The J-direction goes from bottom to top because the J-1 index is at a lower y-position than the J-2 index. Similarly, because the K-1 index is at a lower z-position than the K-2 index, the direction for the K is from the page and going into the page.

BlockM: Concept of Projection

- Projection
  - The concept of projection in blockM could be view similarly to a video projector projecting images onto a screen or wall.
  - The basic idea here is to place the selected nodes onto a surface or set of surfaces much like how light rays are projected onto a screen. Unlike light, the nodes will not bounce off the surface.
  - Another way to view projection is surfaces form the mold of a desired shape and the block is soft clay that sets to the mold.
  - Simple Example (6.0):
    - We want to place a node onto the blue surface by projection

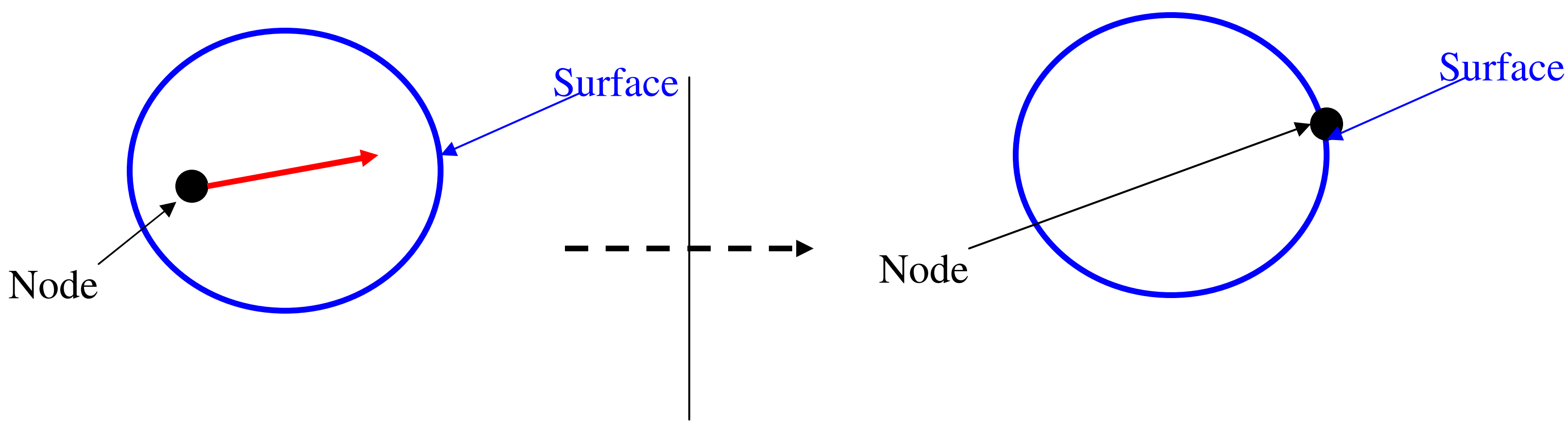


Before projection

After projection

BlockM: Concept of Projection

- Projection (Continued)
  - By default, the projected node takes the shortest path to the surface. However, there is an option for the user to specify a vector for the projection.
  - Simple Example (7.0):
    - Projection with a vector (red)

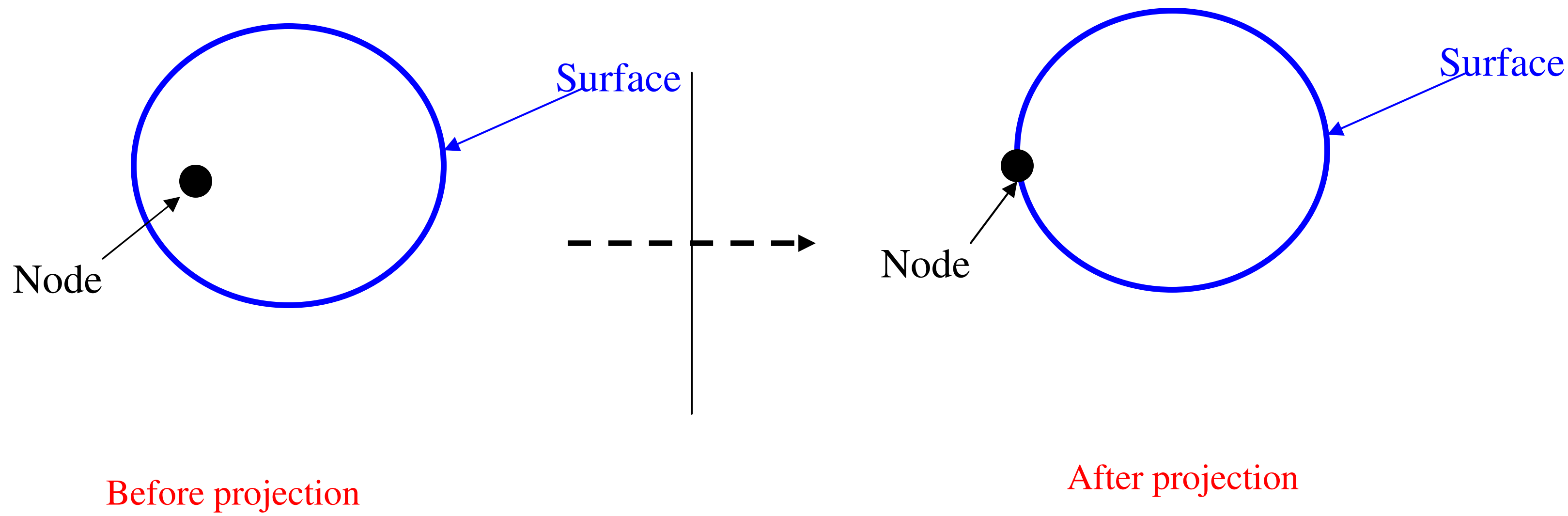


Before projection

After projection

BlockM: Concept of Projection

- Projection (Continued)
  - Simple Example (7.1):
    - Same example as before, but without vector
    - As we could see, the node is moved to the closest position on the surface.
  - Details of how the projection algorithm will not be discussed here. However, more examples of projection will be provided to further demonstrate how it works.

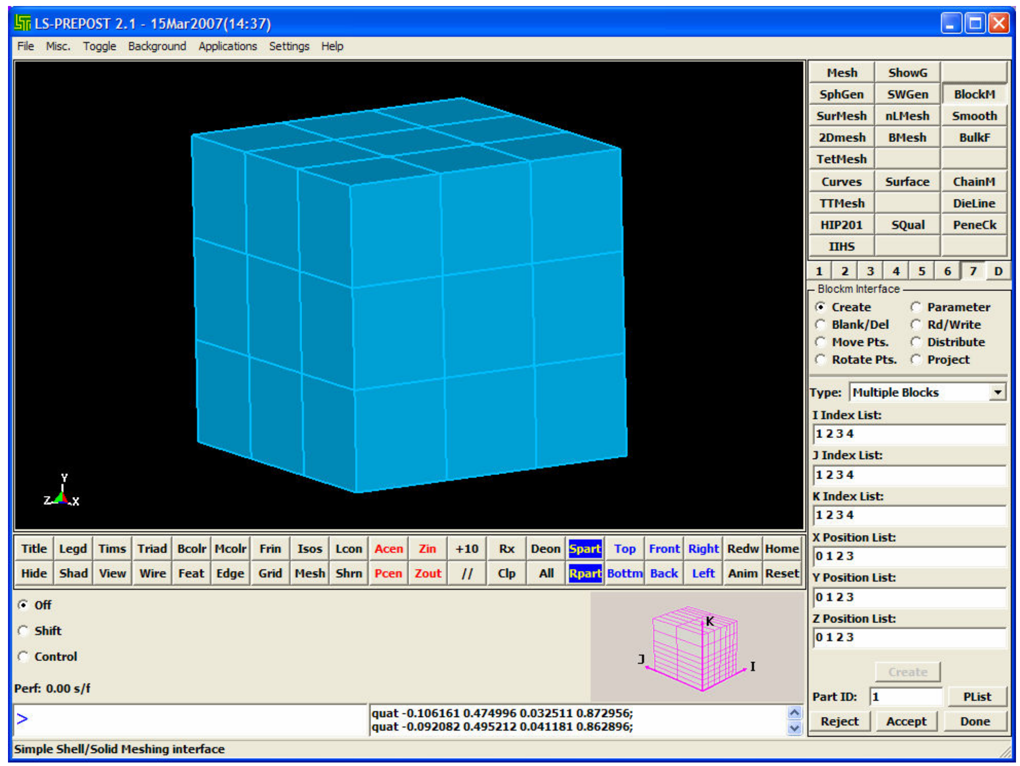


BlockM: Windows

- In BlockM, there are 2 “major” features we need to be familiar with before any meshing could begin.
- Main or Physical Window
  - The Main windows is what we are most familiar with, it’s displays all the nodes, elements, curves, and surfaces.

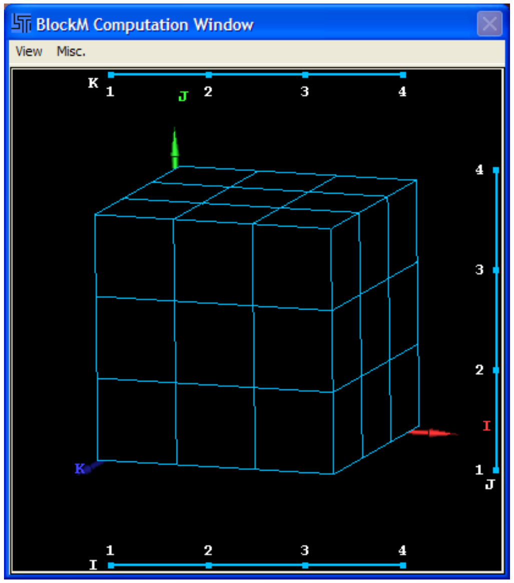
BlockM: Windows

- Main Window
  - Everything is displayed in Euclidean space.
  - This serves to display detailed information about the mesh.
    - Nodes
    - Elements
    - Surfaces
    - Curves
    - Other parts
  - It also allow for user to
    - Zoom
    - Rotate
    - Pan
    - Selection



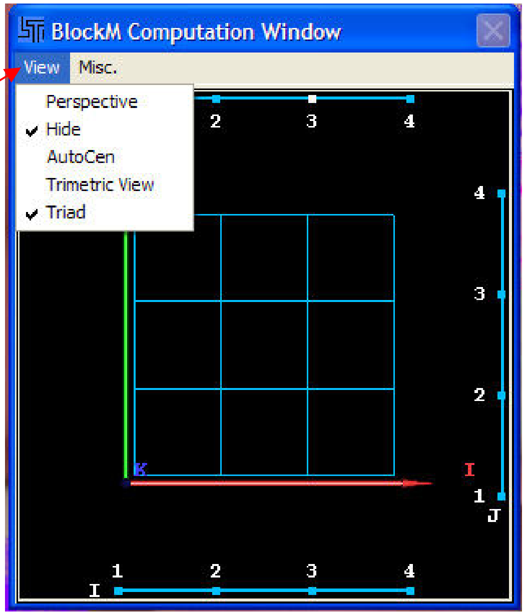
BlockM: Windows

- Computation Window
  - This window pops up as soon as the “create” button is clicked in the main window.
  - The main purpose of the computation window is to provide user-friendly environment for selecting nodes and elements.
  - As mentioned earlier, to select nodes and elements, the user needs to do this via the indices. Hence, the Computation window displays everything in index space.
  - All partitions are displayed as perfect squares or cubes.
  - Translation, rotation, and projection of nodes and elements do not affect the block in this window.
  - Only deletions of partitions or regions will be affected.



BlockM: Windows

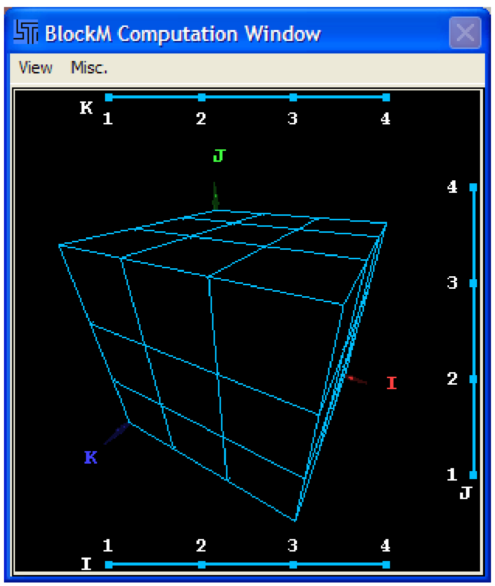
- Computation Window (continued)
  - Let's look at the different aspects within the computation window.
    - View menu – contains different display modes for the block.
      - Perspective
      - Hide*
      - AutoCen
      - Trimetric View
      - Triad*



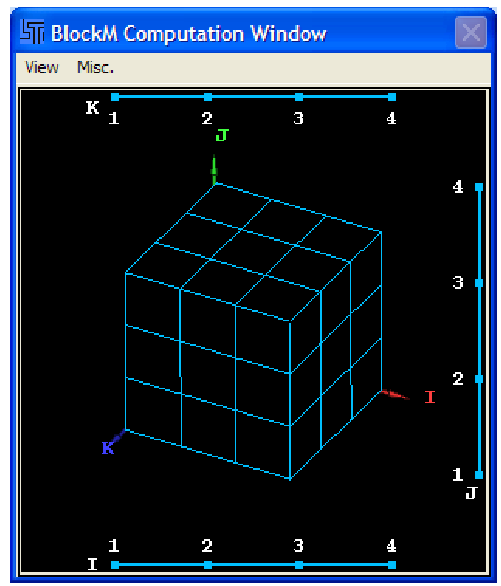
*Default display setting

BlockM: Windows

- Computation Window: View menu (continued)
  - Perspective
    - When activated, this provides the block with a perception of dept.



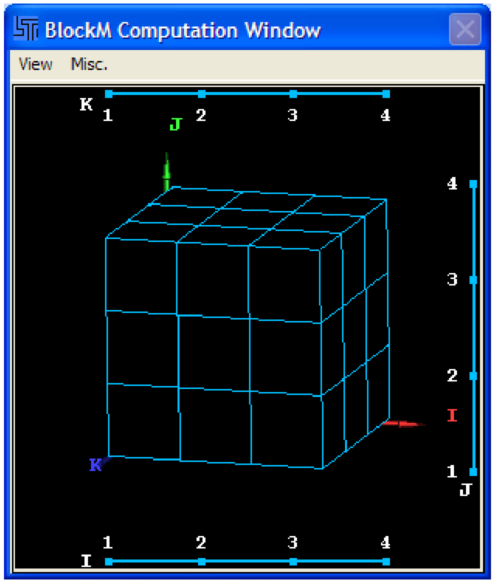
Example of **Perspective** display **on**.



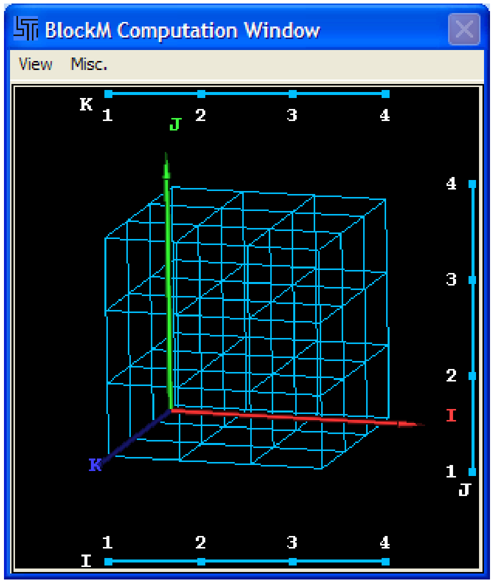
Example of **Perspective** display **off**.

BlockM: Windows

- Computation Window: View menu (continued)
  - Hide
    - When activated, this shows only the outer edges of the block. This is set on by default.



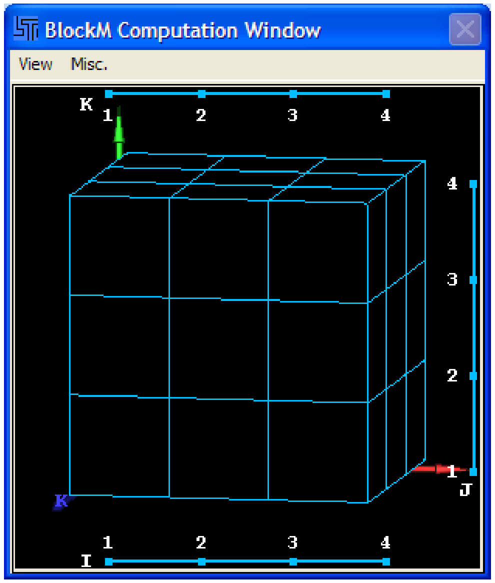
Example of **Hide** display **on**.



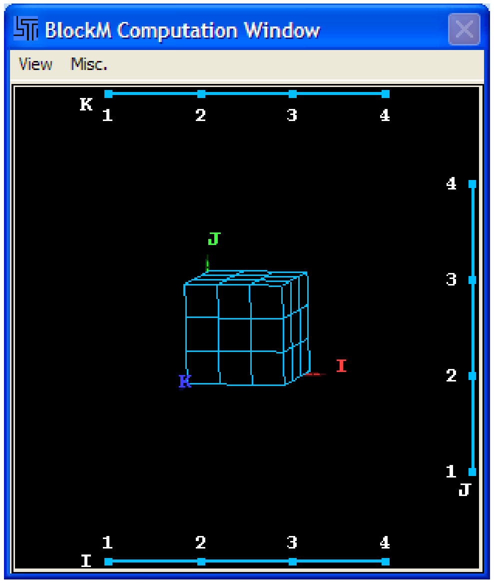
Example of **Hide** display **off**.

BlockM: Windows

- Computation Window: View menu (continued)
  - AutoCen
    - This works much like the Acen feature in the Main window; it automatically centers the block within the window. This is does not "lock" the block in view. Hence, the user will need to activated every time it is needed.



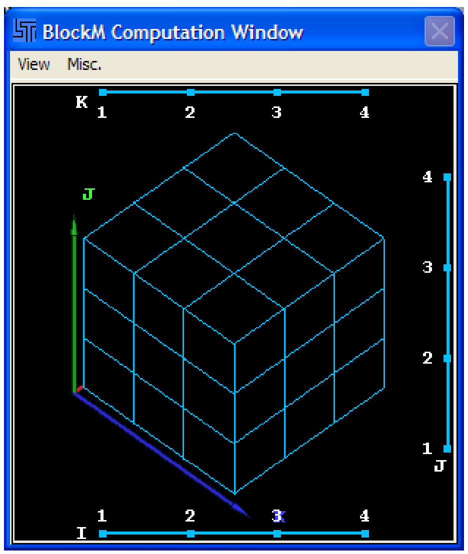
Example of **AutoCen** display **on**.



Example of **AutoCen** display **off**.

BlockM: Windows

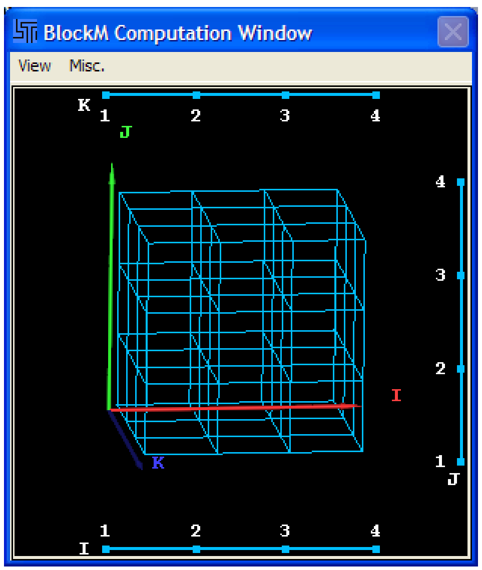
- Computation Window: View menu (continued)
  - Trimetric View
    - This set block in the computation window to trimetric view, such as shown below. Much like the AutoCen feature, this does not hold the view.



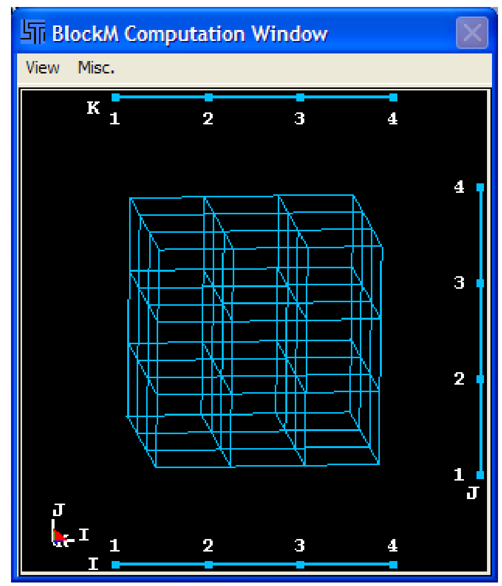
Example of **Trimetric View** display **on**.

BlockM: Windows

- Computation Window: View menu (continued)
  - Triad
    - This is a toggle to turn on and off the display of the triad. It is turned on by default.



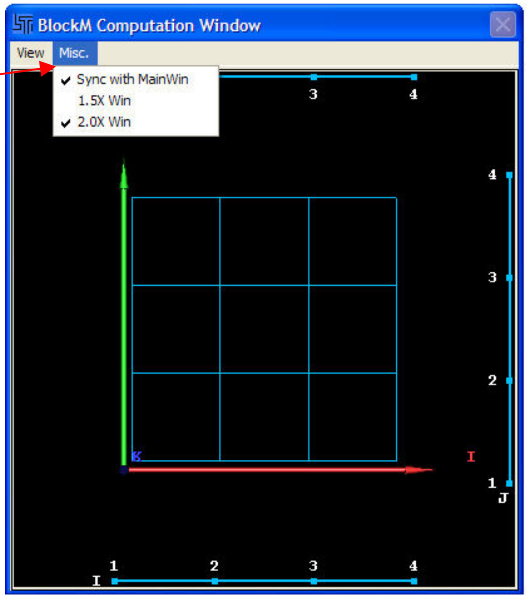
Example of **Triad** display **on**.



Example of **AutoCen** display **off**.

BlockM: Windows

- Computation Window
  - Misc menu – contains additional user's preference settings for the computation window
    - Sync with MainWin*
      - This toggle setting maintains the same orientation for block in the main and computation windows.
      - For example, if the block is rotated by 30 degree about the J-axis, the block in the main window will rotate by the same amount about the Y-axis. Vise versa, if the block in the main window is rotated by 30 degree, the block in the computation window will rotate by the same amount.
      - The Sync is only applicable for rotation. It does not apply to pan and zoom.
    - 1.5x Win
      - Increases the computation window by 1.5 times the default size.
    - 2.0x Win
      - Increases the computation window by 2 times the default size.

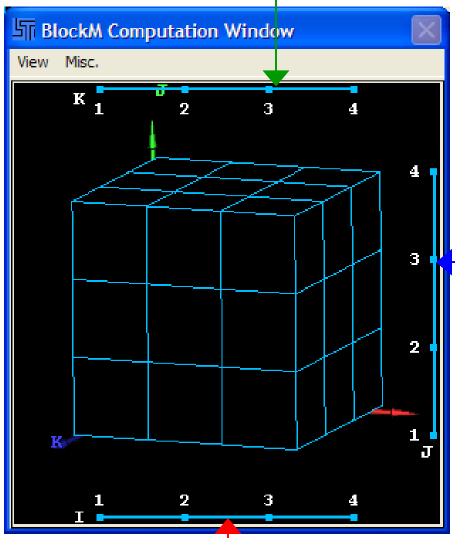


*Default display setting

BlockM: Windows

- Computation Window: selection
  - The most important part of the computation window is the selection of points, edges, faces, and regions.
  - There are 2 ways to do selection.
    1. Using the mouse to pick a point on the block or drag over the region of interest.
    2. Using the mouse to select the "index slider bars" at the top, right, and both of the computation window.
  - It is important to notice there are points on the slider bars and each point has a number associated with it. These points are index number for each direction.
  - To the left, the block has 3 partitions and 4 indices for each direction. Hence, we see 4 points for each slider bar.
  - To clear selection: as long as the Computation window is active, press the "d" key. This might be change to "ESC" key.

Slider bar for K-index

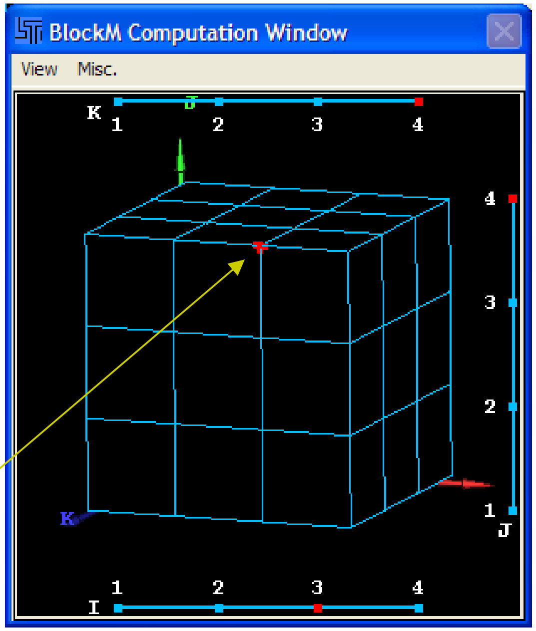


Slider bar for J-index

Slider bar for I-index

BlockM: Windows

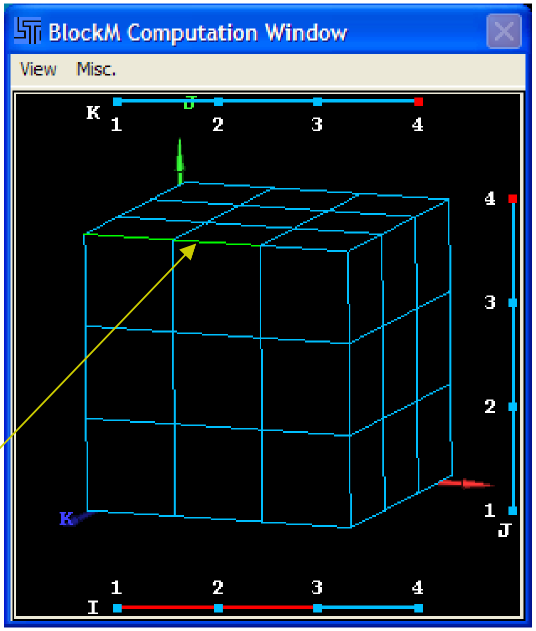
- Computation Window: selections (continued)
  - **Point selection**
    - Example (8.0)
      - Lets select a **point** at
        - I-index: 3
        - J-index: 4
        - K-index: 4
      - We could simply click on the point on the block or click at 3 locations on the slider bars.
      - Notice how the selected point is highlighted in **red**.
      - Also, the corresponding indices also turn red on the slider bars.



Selected point

BlockM: Windows

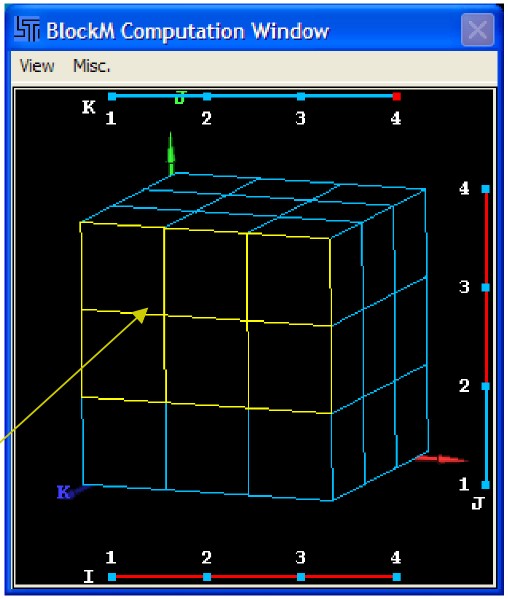
- Computation Window: selection (continued)
  - **Edge selection**
    - Example (8.1)
      - Lets select an **edge** at
        - I-index: 1 3
        - J-index: 4
        - K-index: 4
      - We simply drag-select along the edge.
      - Or we could click and slide along the I-index slider bar from 1 to 3 and click at point 4 on both the J and K index slider bar.
      - Notice the selected edge is highlighted in **green**.
      - The corresponding partitions turn red in the I index slider bar.



Selected edge

BlockM: Windows

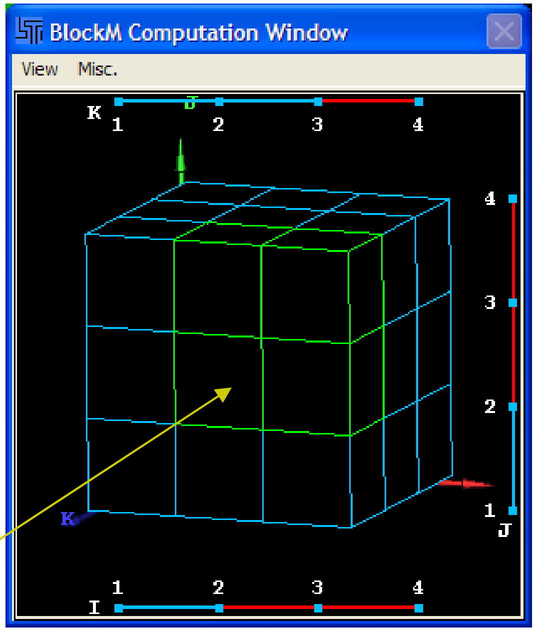
- Computation Window: selection (continued)
  - Face selection
    - Example (8.2)
      - Lets select a face at
        - I-index: 1 4
        - J-index: 2 4
        - K-index: 4
      - Again, we could drag-select over the area of interest.
      - Or we could click and slide along the I-index slider bar from 1 to 4, slide along the J bar from 2 to 4, and click on point 4 on the K bar.
      - Notice the selected face is highlighted in yellow.



Selected face

BlockM: Windows

- Computation Window: selection (continued)
  - **Region selection**
    - Example (8.3)
      - Lets select an **region** at
        - I-index: 2 4
        - J-index: 2 4
        - K-index: 3 4
      - Again, we could drag-select over the area of interest.
      - Or we could click and slide along the I-index slider bar from 2 to 4, slide along the J bar from 2 to 4, and slide along the K bar from 3 to 4.
      - Also, notice the selected region is highlighted in **green**.



Selected region

BlockM: Start Using BlockM

- Features Overview
  - To the right is the BlockM interface and it is the first thing we see when we click on the BlockM button. The Create radio button is selected by default.
  - Current tools and features
    - Create
    - Blank/Del
    - Move Pts.
    - Rotate Pts.
    - Parameter
    - Rd/Write
    - Distribute
    - Project

BlockM: Start Using BlockM

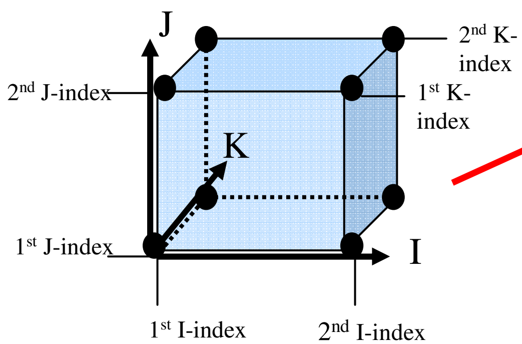
- Create
  - In the earlier sections, we discussed the concept of index and partition. We also discussed how they are related to nodes and elements. It is here that we apply these concepts to generate the block. BlockM could create both SOLID and SHELL elements as separate parts or mixed together in one part.
  - As we see to the right, there are 6 fields. We need to fill in the fields with appropriate values.
  - The top 3 fields:
    - I Index List
    - J Index List
    - K Index List
    - are where we define the number of elements and partitions in the I, J, and K directions. Again, these correspond to X, Y, and Z directions in the Euclidean space, respectively.
  - The bottom 3 fields:
    - X Position List
    - Y Position List
    - Z Position List
    - are where we specify the positions of the indices define in the top 3 fields.

BlockM: Start Using BlockM

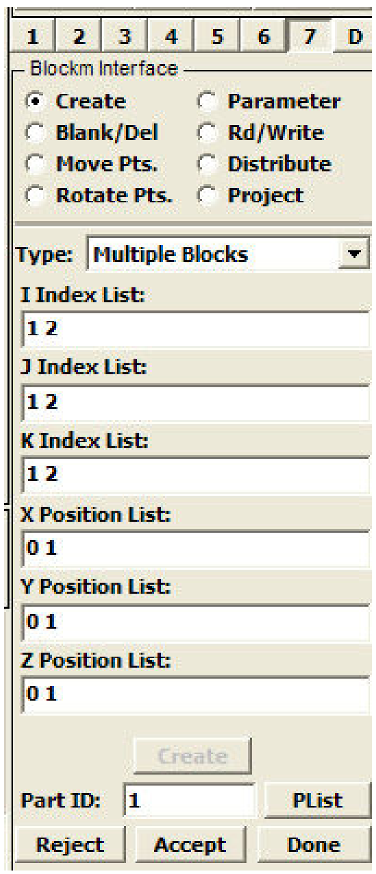
- Create (continued)
  - Create button
    - Once all the field are fill in appropriately, click on this button to bring up the Computation window.
  - Part ID field
    - This sets the part number for the block.
  - PList button
    - We could enter a number in the Part ID field or click on this button and a select a part from a pop-up list.
  - Reject button
    - Once a block is create and for whatever reason we want to start over, just click on this button.
  - Accept button
    - Similarly to the Reject button above, we click this button if everything is acceptable.
  - Done
    - To exit out of BlockM.

BlockM: Start Using BlockM

- Create (continued)
  - Syntax for the fields
    - We could use spaces or commas between values.
  - Examples (9.0): SOLID element
    - Let's create a very simple 3D block, the same one given in earlier section.
    - To the right is how we enter the indices and their positions.
    - Note, the first index in any direction must begin with 1 or higher, but less than the 2 index.

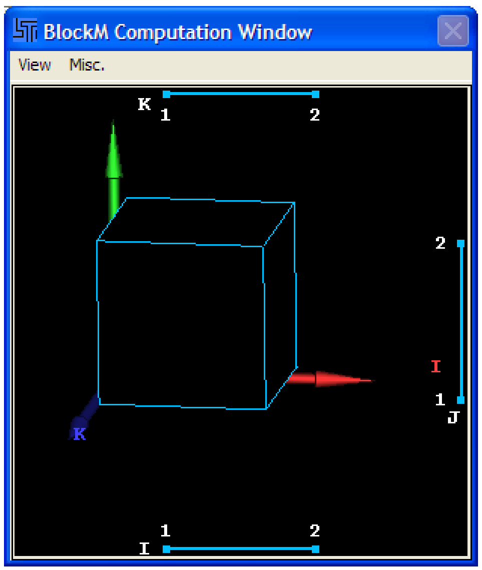


Conceptual Block

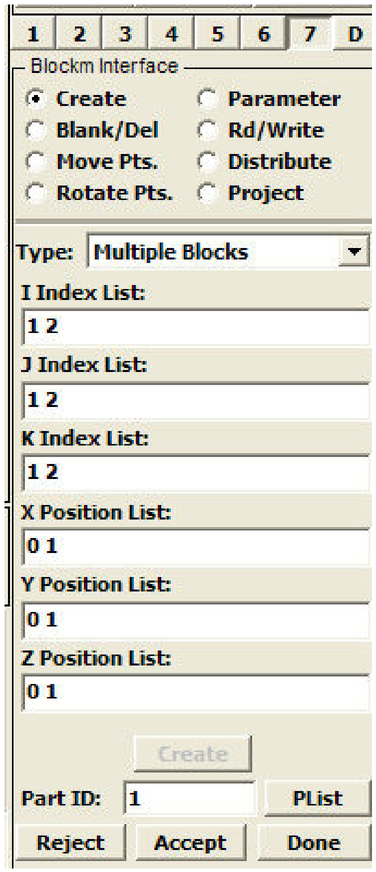


BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.0)
    - Below is what the computation window show when we click on the create button for the specified block.

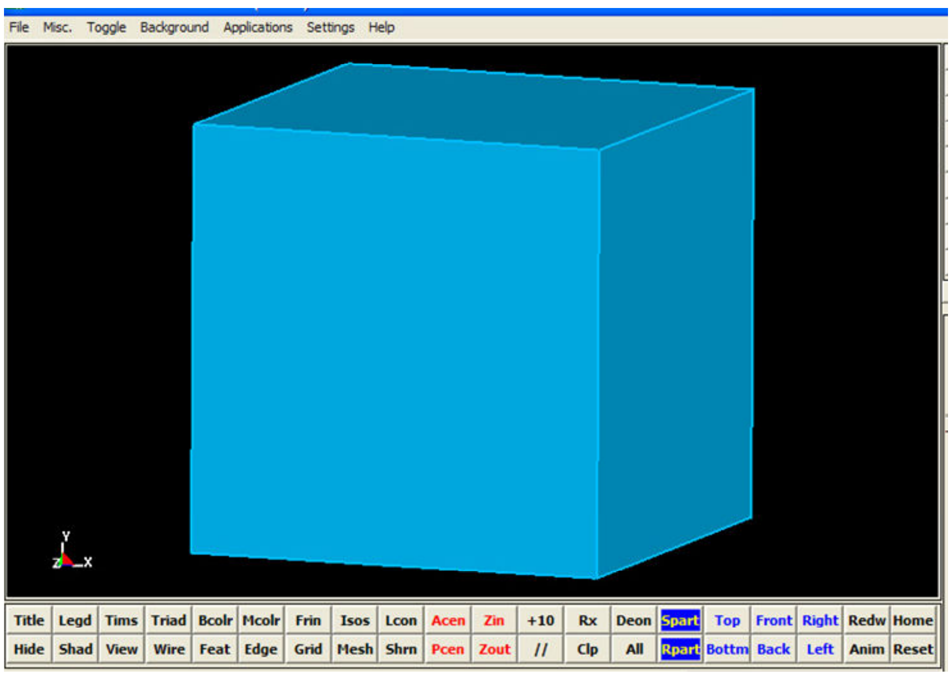


Block in Index space

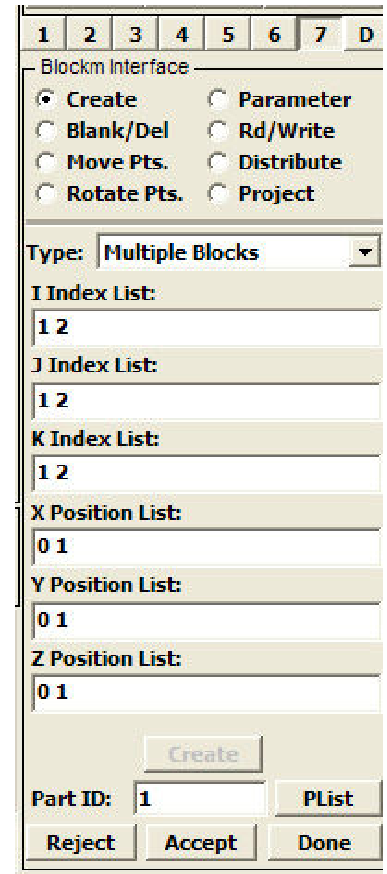


BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.0)
    - The same block is shown in the main window in Euclidean space. This block has only 1 element.

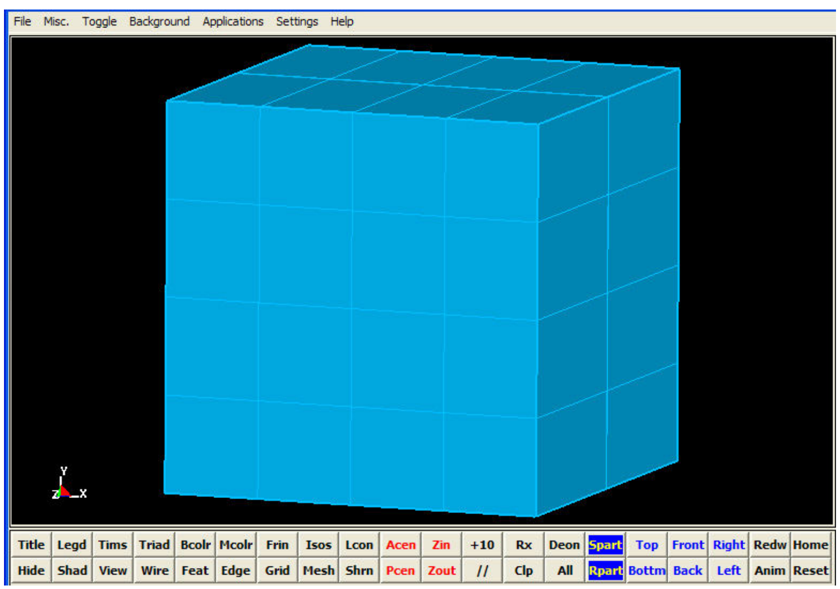


Block in Euclidean space

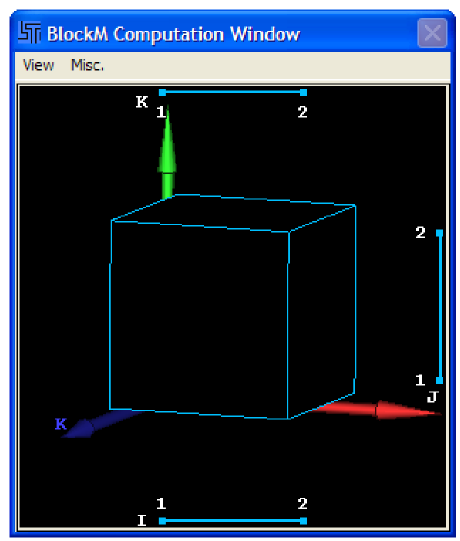


BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.0)
    - Let's add more elements to the previous example.
    - Notice the block in the computation window. It looks the same as the previous example. However, the block in the Main window appears slightly different. It has more elements than the previous example.



Main window



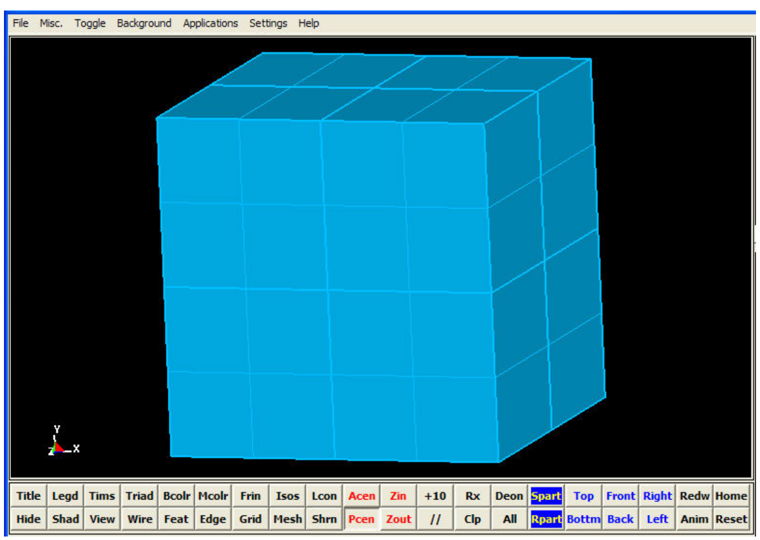
Computation window

BlockM: Start Using BlockM

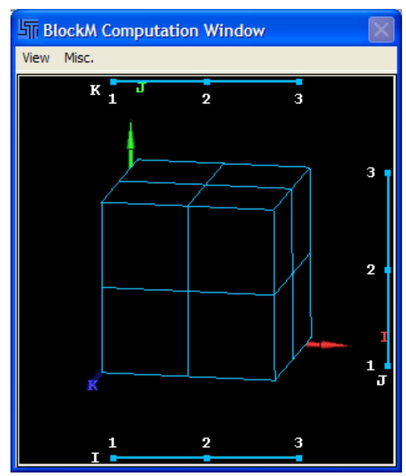
- Create (continued)
  - Examples (9.0)
    - Notice the block in the main window has 4 elements in the X and Y, and 2 in the Z direction.
    - Let's see how this works. In the fields to the right, we see the only differences between this set of inputs and that of the previous examples are the top 3 fields.
      - I-index: 1 5 ----------> 4 elements
      - J-index: 1 5 ----------> 4 elements
      - K-index:1 3 ----------> 2 elements
    - There is a simple relationship between the number of elements and indices for any partition.
      - N element = higher index # - lower index #
      - Higher index # = lower index # + N element
    - It is not necessary that the 1^{st} index has to be 1. It just has to be less than the 2^{nd} and 3^{rd} index.
    - There are countless number of ways to specify the same block. We could initialize the same block in another way.
      - I-index: 4 8
      - J-index: 10 14
      - K-index: 5 7

BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.1)
    - Let's add 1 partition to each of the index direction for the block in example 8.0. We set the 2nd index right in the middle. Each partition has 2 elements, except for the ones in the K-direction.



Main window



Computation window

BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.1)
    - Let's see how the relationship stated above works.
      - I-index: **1** + (2 el.)= **3** + (2 el.) = **5**
      - J-index: **1** + (2 el.)= **3** + (2 el.) = **5**
      - K-index: **1** + (1 el.) = **2** + (1 el.) = **3**
    - The relationship works out quite nicely.
    - Notice the bottom 3 fields:
      - X-pos : 0 .5 1
      - Y-pos : 0 .5 1
      - Z-pos : 0 .5 1
    - Because we have 3 indices in each of the index fields, we need to have 3 values for each of the position field as well.

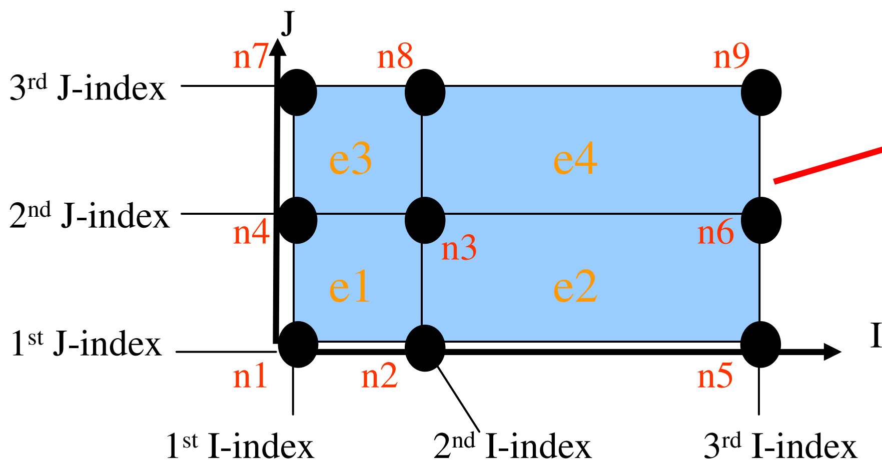
BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.1)
    - The table below illustrates the order on the indices and their corresponding spatial initial positions.

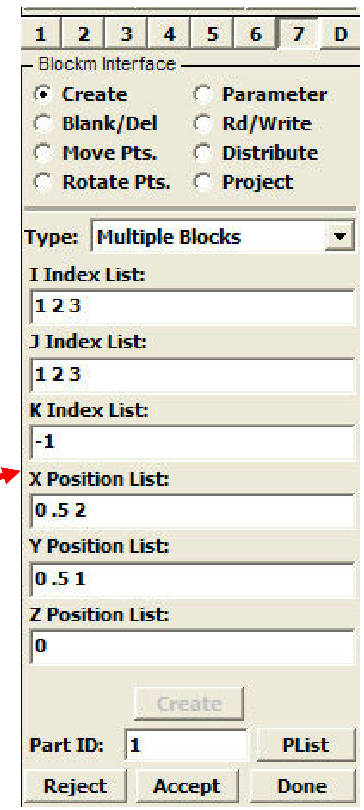
	1^{st} Index	2^{nd} Index	3^{rd} Index
I-index	1	3	5
X-position	0	.5	1
J-index	1	3	5
Y-position	0	.5	1
K-index	1	2	3
Z-position	0	.5	1

BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.2): SHELL element
    - Let's create a very simple SHELL or 2D block based on example 4.4 above.
    - A shell block is created in the same manner as that for the solid, except for thickness direction in index space is specified with a minus sign (-).

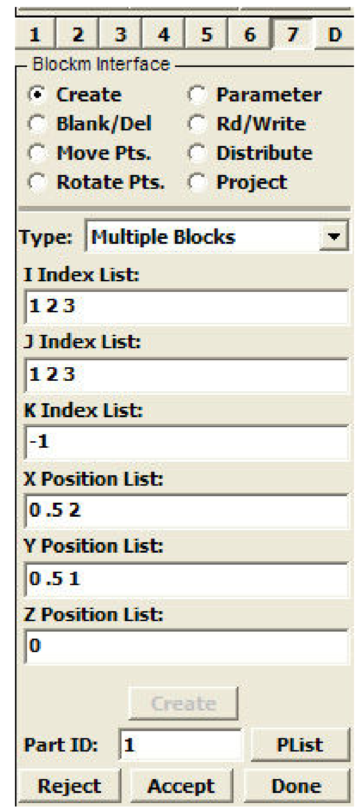


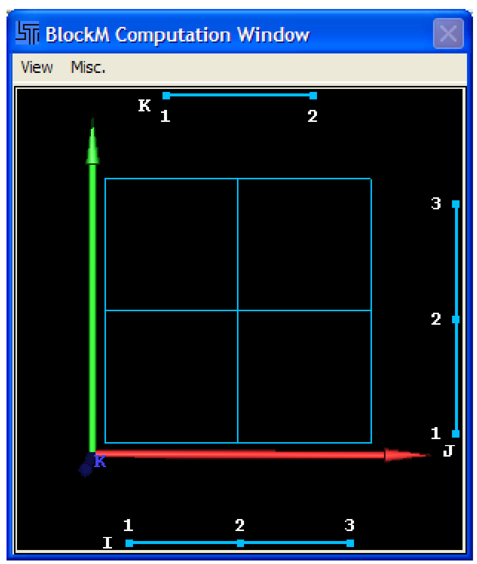
Conceptual Block



BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.2): SHELL element
    - Notice the -1 in the K Index List field.
    - Below is the shell block shown in Computation window.

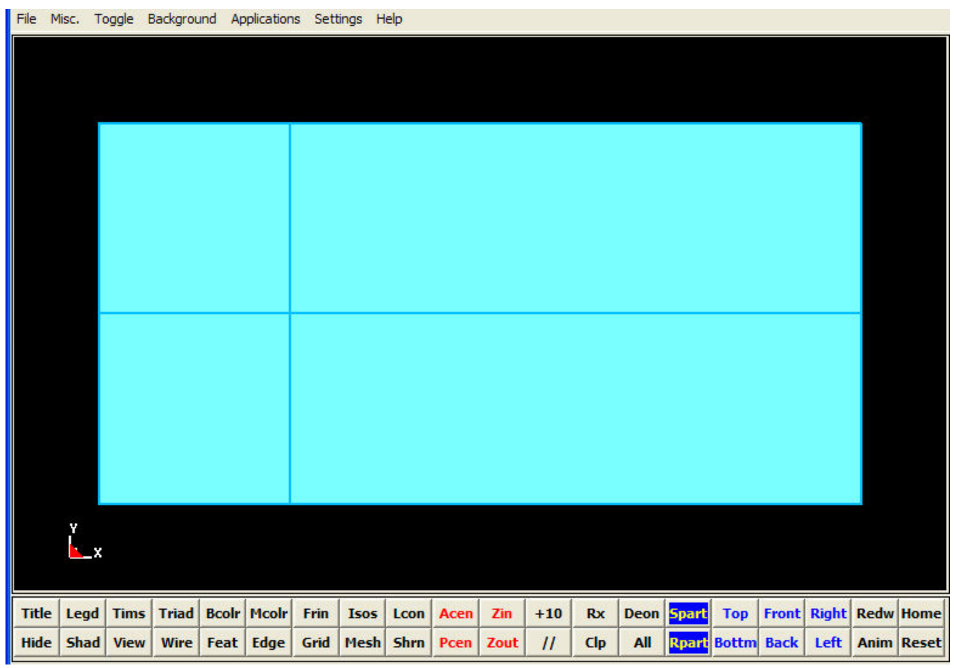


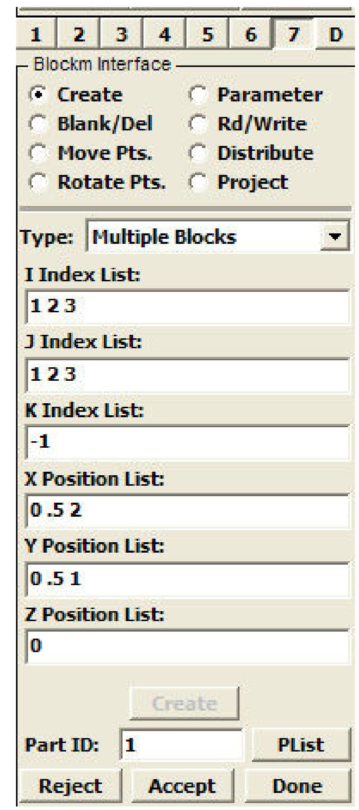


Block in Index space

BlockM: Start Using BlockM

- Create (continued)
  - Examples (9.2): SHELL element
    - Below is the shell block shown in Main window.

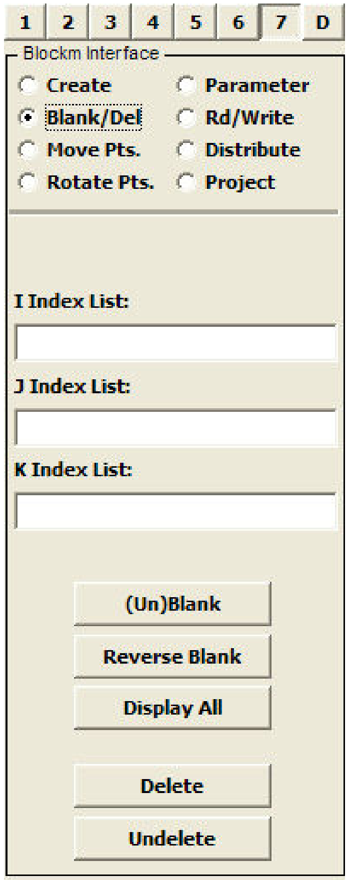




Block in Euclidean space

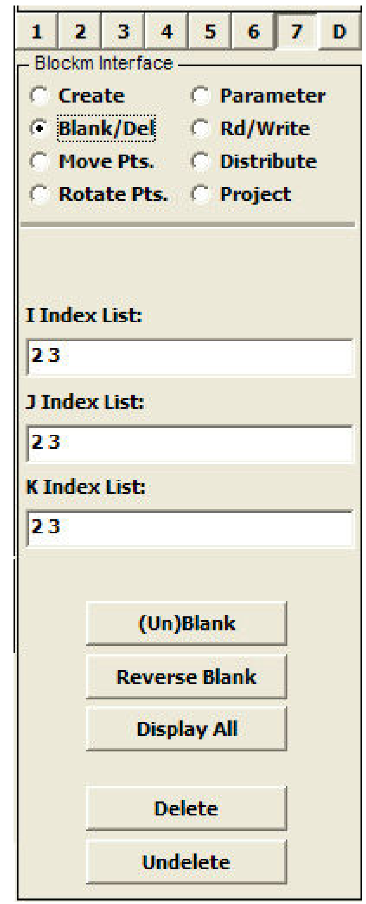
BlockM: Start Using BlockM

- Blank/Del
  - This feature allows for both visually disabling (blanking) and for deleting elements and nodes.
  - To blank or delete, we need to first select a region. Blank and delete are not applicable to point, edge, and face.
  - There are 2 ways to select a region. We could either specify the region with indices in the index list fields, as seen on the right, or select from the computation window.
  - If selection is made from computation window, the index list fields will be automatically filled in accordingly.
  - Once a region is selected, click on the blank or delete button to the right.



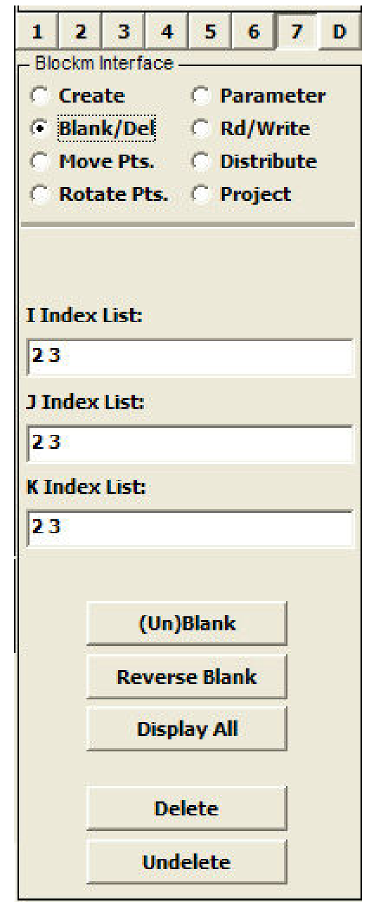
BlockM: Start Using BlockM

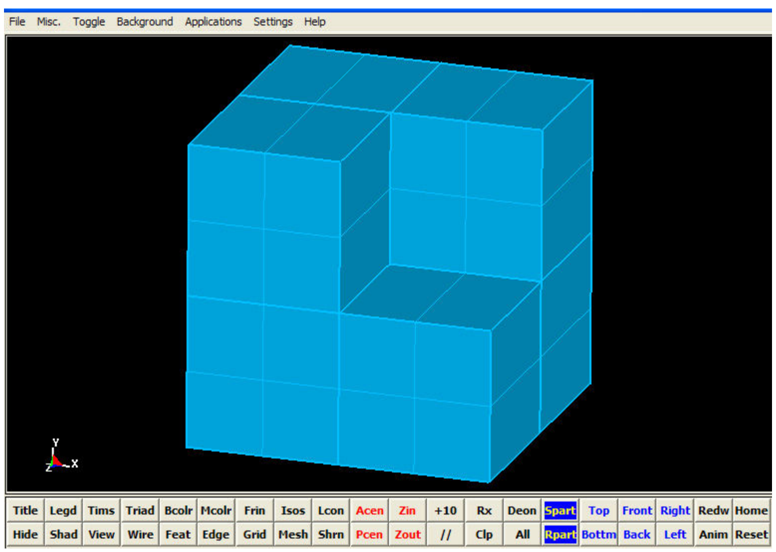
- Blank/Del (continued)
  - (Un)Blank button
    - Blanking is for disabling the display of nodes and elements. Click this button again will unblank and enable the display of nodes and elements again.
    - This is useful for visually checking interior elements.
    - Example (10.0)
      - Let's use the same block as that in example 8. We want to blank out the region:
      - I-index: 2 3
      - J-index: 2 3
      - K-index: 2 3



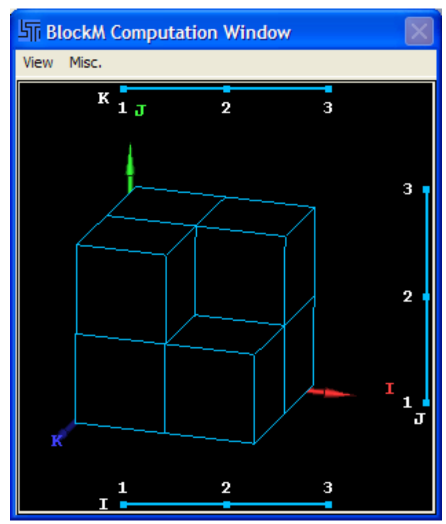
BlockM: Start Using BlockM

- Blank/Del (continued)
  - (Un)Blank button
    - Example (10.0)
      - Notice the blank applies to both the main and computation window.





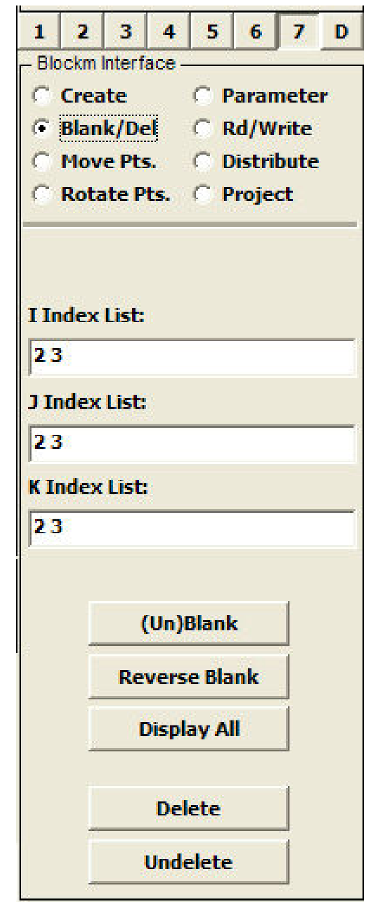
Main window

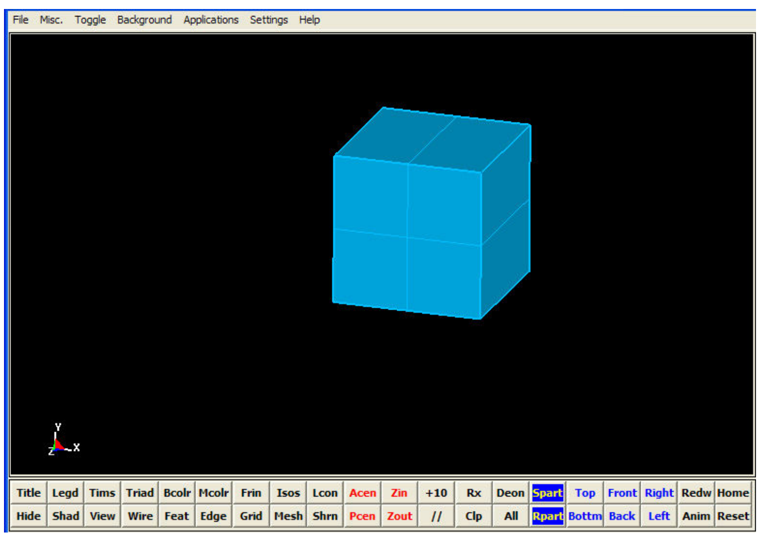


Computation window

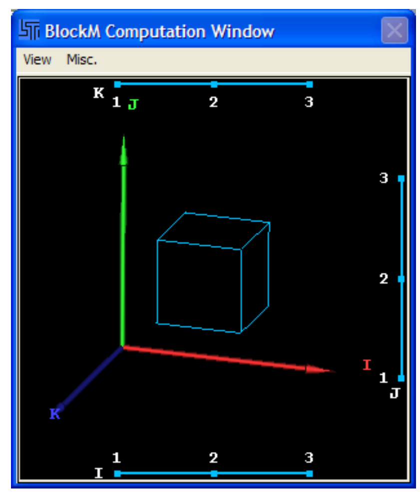
BlockM: Start Using BlockM

- Blank/Del (continued)
  - Reverse Blank button
    - This blanks the region that is currently display and unblank the visually disabled region.
    - Example (10.1)
      - Note the fields to the right do not update to reflect the corresponding displayed region.





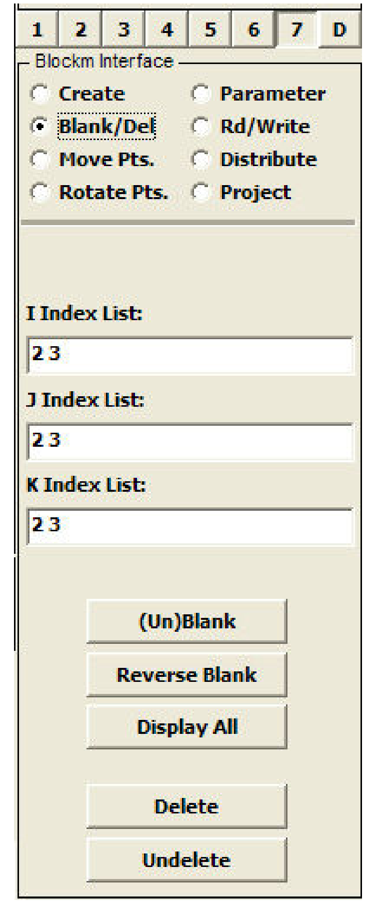
Main window



Computation window

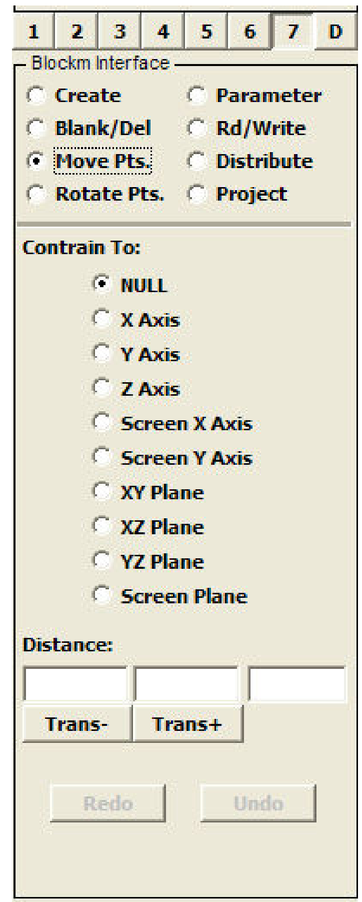
BlockM: Start Using BlockM

- Blank/Del (continued)
  - Display All button
    - This enables the display of all blank regions.
  - Delete button
    - This feature is similar to the blank. Visually, the blank and delete features appear to be the same. However, the delete actually remove nodes and elements from the block once the "accept" button is clicked .
    - Unlike the blank feature, unblanking the missing region will not bring it back.
  - Undelete button
    - This feature brings back the last deleted region.



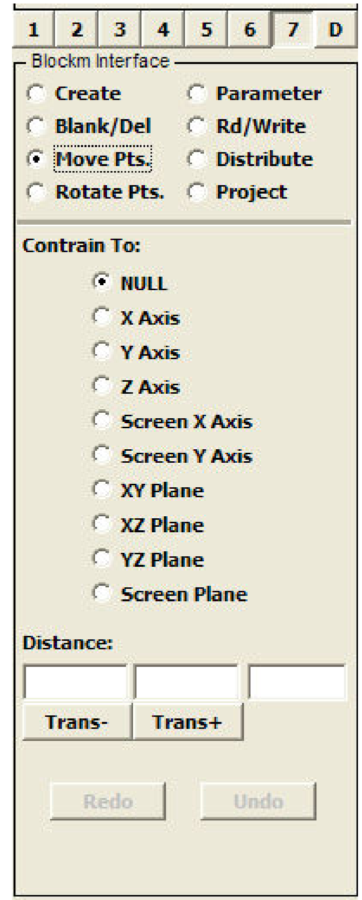
BlockM: Start Using BlockM

- Move Pts.
  - This feature lets the user move points, edges, faces, and regions.
  - There are a number of options to choose from depending on the intend of the translations.
  - To move:
    1. Select the point, edge, face or region using the computation window as explained previously.
    2. Select one of the constrained options to the right.
    3. Enter the distance to translate. This does not apply to any screen constrained translation.
    4. Click "Trans-" or "Trans+" button. A less precise method is to use the mouse to drag in the Main window.



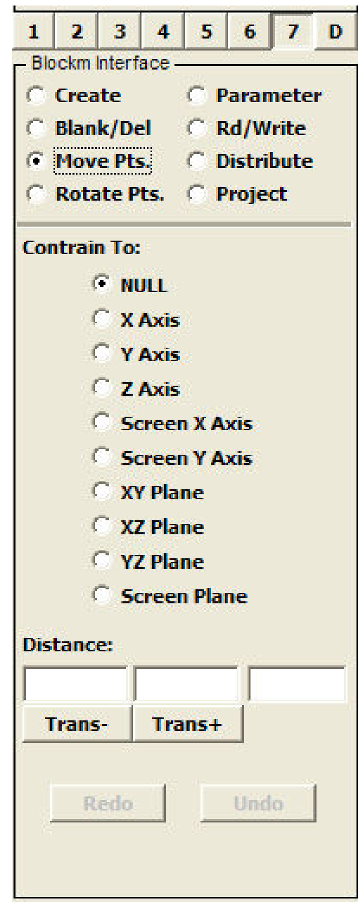
BlockM: Start Using BlockM

- Move Pts. (continued)
  - Constrain To:
    - NULL*
      - This is a default setting. It means no constraint. We could move the nodes and elements in any direction. Hence, there are 3 fields for the 3 global directions (X, Y, and Z).
    - X-Axis
      - Nodes and elements could only be translated along the X-axis.
    - Y-Axis
      - Likewise, nodes and elements could only be translated along the Y-axis.
    - Z-Axis
      - Similarly, nodes and elements could only be translated along the Z-axis.
    - Screen X Axis
      - Nodes and element could only be translated along the screen's X-axis or horizontal direction. This not the same as the global X-axis.



BlockM: Start Using BlockM

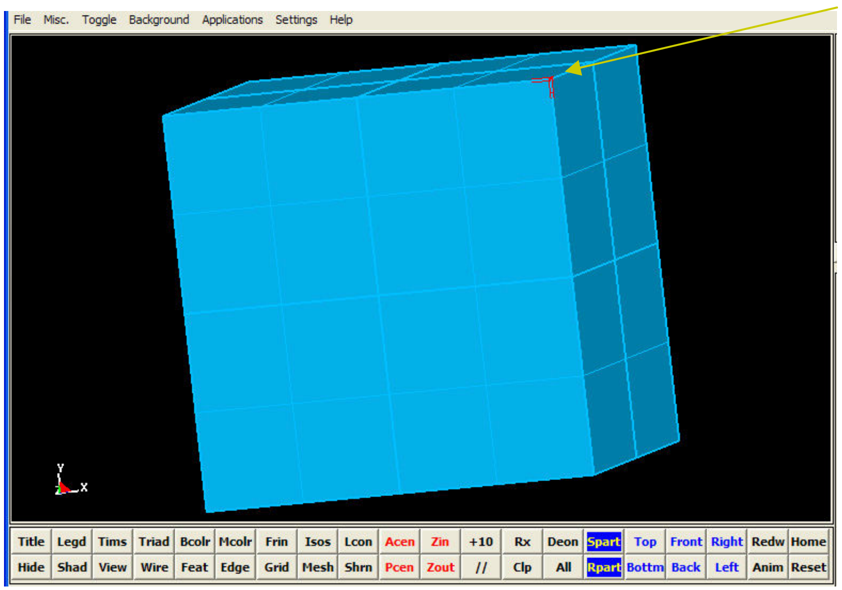
- Move Pts. (continued)
  - Constrain To:
    - Screen Y Axis
      - Nodes and element could only be translated along the screen's Y-axis or vertical direction. This not the same as the global Y-axis.
    - XY Plane
      - Nodes and element could only be translated on the global XY plane.
    - XZ Plane
      - Nodes and element could only be translated on the global XZ plane.
    - YZ Plane
      - Nodes and elements could only be translated on the global YZ plane.
    - Screen Plane
      - Nodes and elements could only be translated along both the screen's X and Y axis.



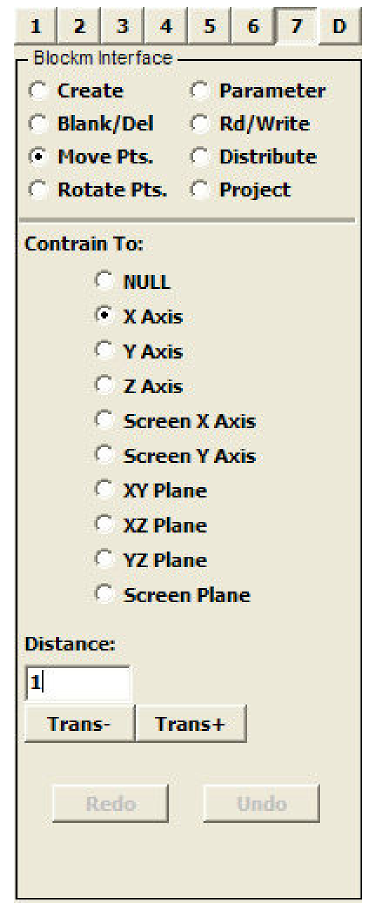
BlockM: Start Using BlockM

- Move Pts. (continued)
  - Examples (11.0): Constrain to X-Axis
    - Using the same block as in example 10.0, let’s move a point along the X-axis a distance of 1.0.

Point to move



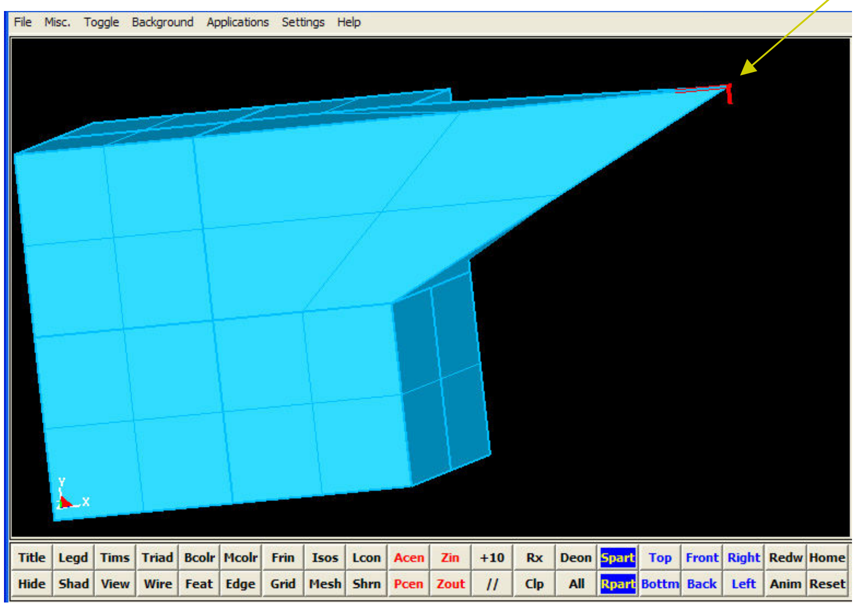
Before



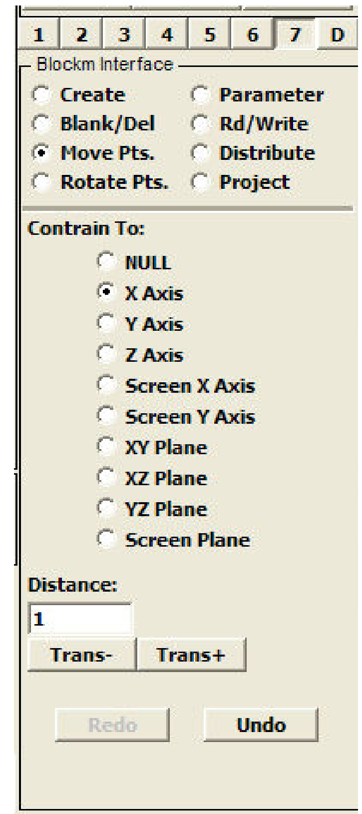
BlockM: Start Using BlockM

- Move Pts. (continued)
  - Examples (11.0): Constrain to X-Axis
    - Notice the **Undo** button is active.

Point moved



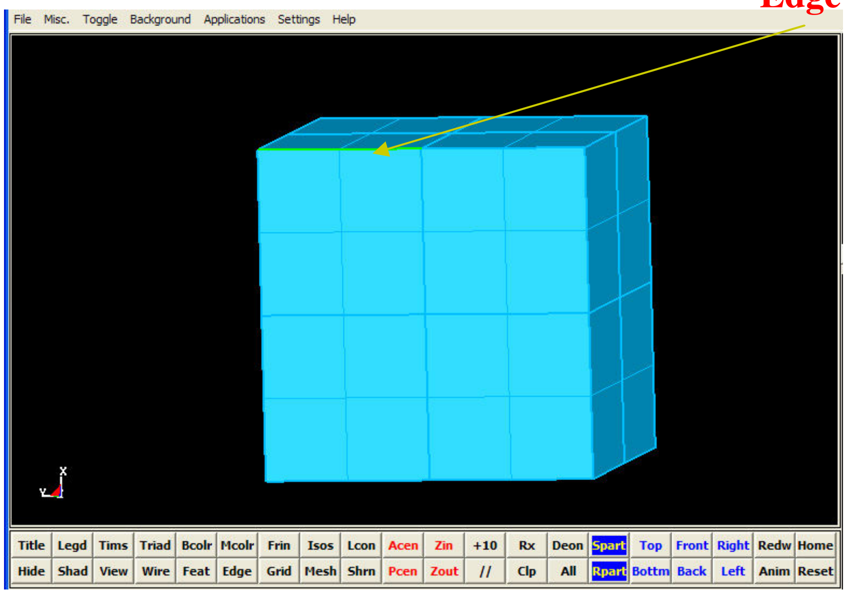
After

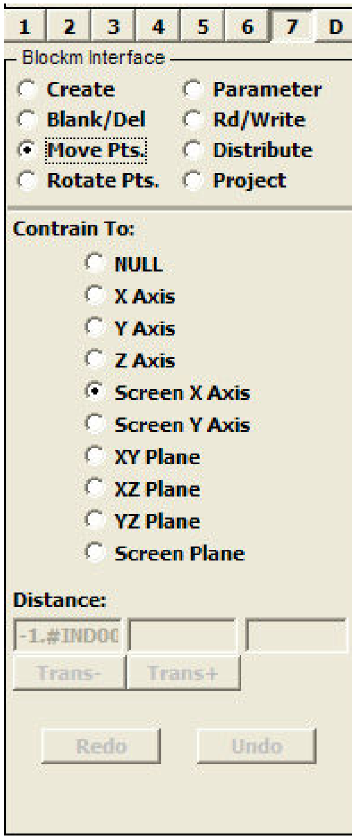


BlockM: Start Using BlockM

- Move Pts. (continued)
  - Examples (11.1): Constrain to Screen's X-Axis
    - Using the same block as in example 10.0, let's move an edge along the screen's X-axis by some distance. As mentioned before, we need to drag in the Main window. The block is rotated 90 degree so that the global X is pointing up.

Edge to move

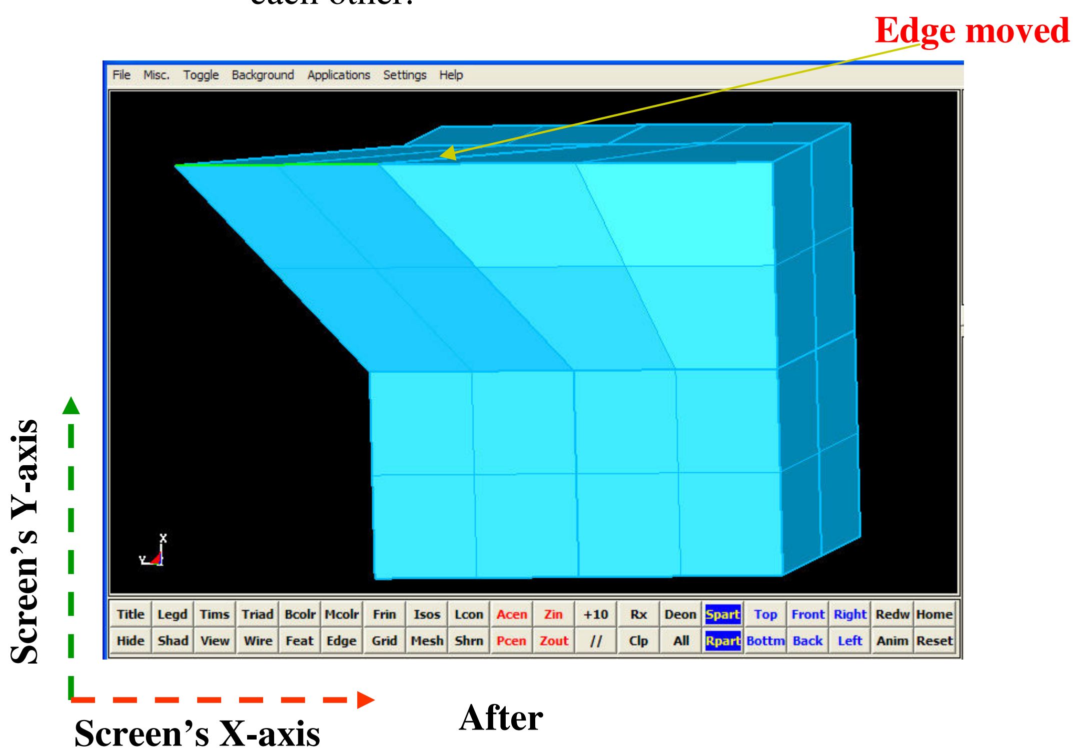




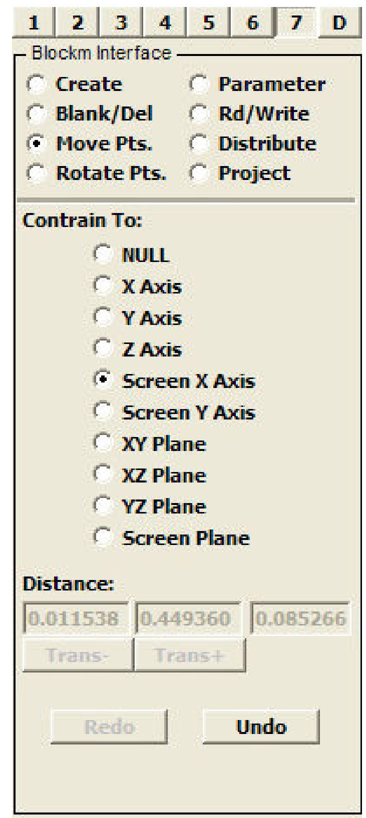
Before

BlockM: Start Using BlockM

- Move Pts. (continued)
  - Examples (11.1): Constrain to Screen's X-Axis
    - Notice that the screen X-axis is not the same as that on the global X-axis. In this example, they are almost orthogonal to each other.



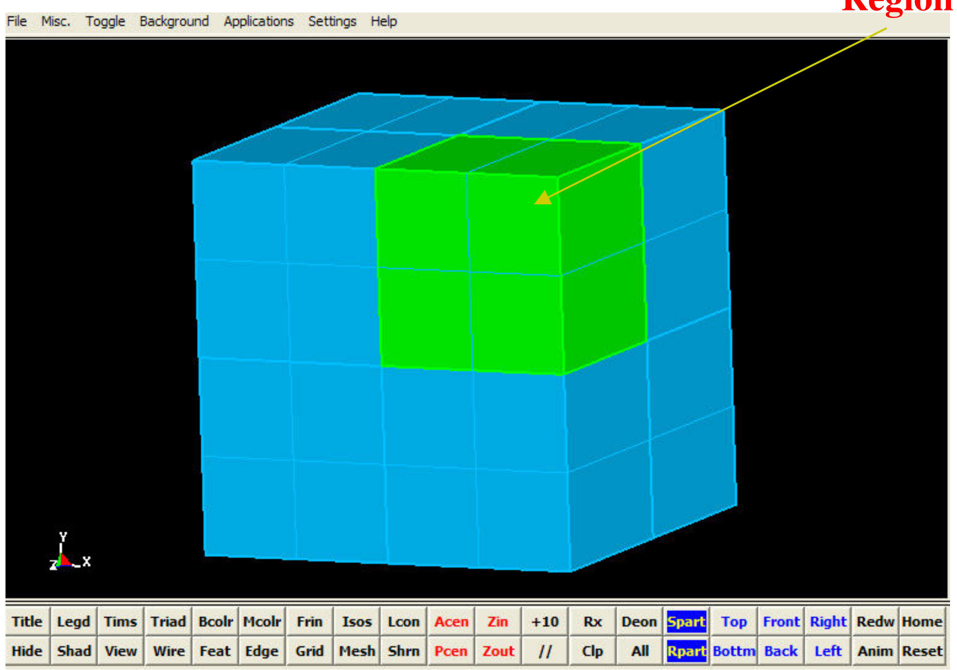
After

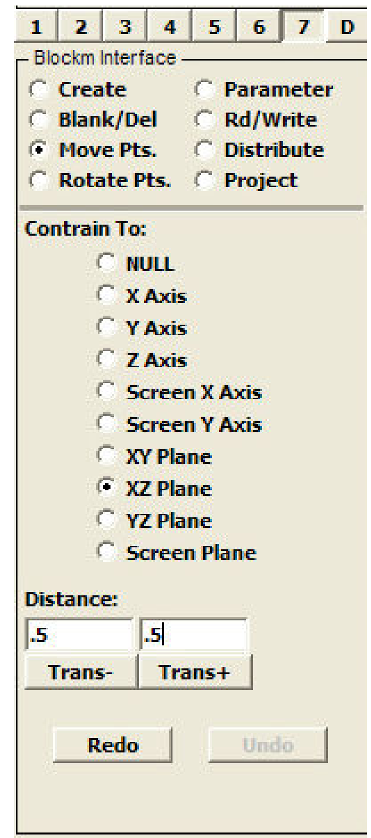


BlockM: Start Using BlockM

- Move Pts. (continued)
  - Examples (11.2): Constrain to XZ Plane
    - Using the same block as in example 10.0, let’s move **region** along the screen’s X-axis and Z-axis by 0.5.

Region to move



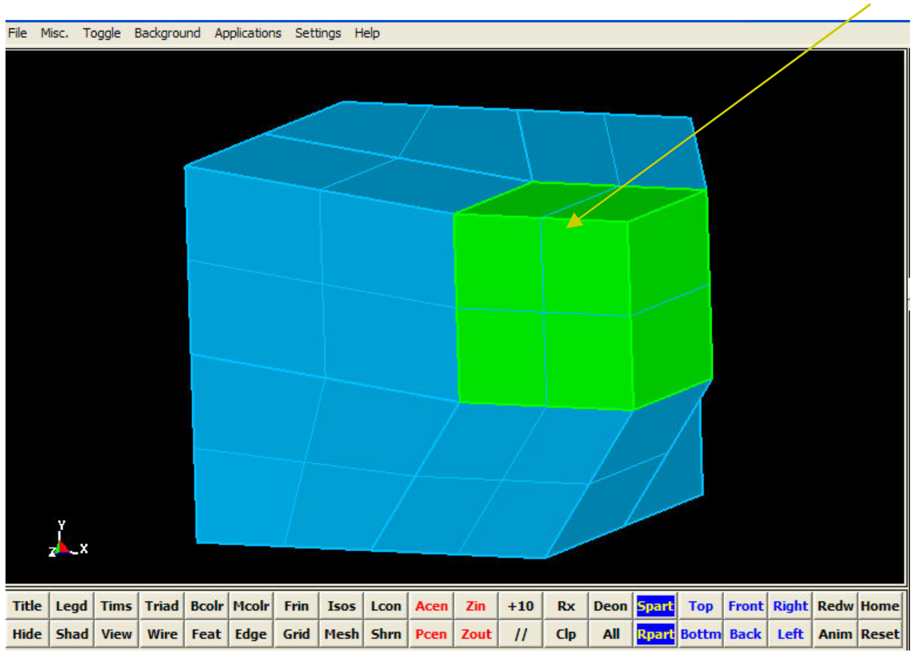


Before

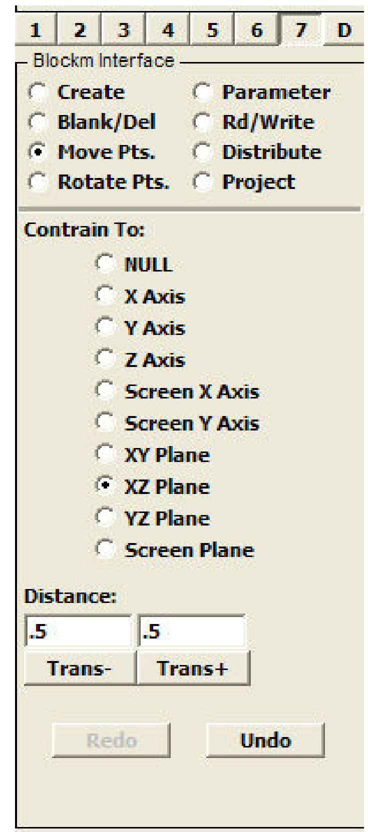
BlockM: Start Using BlockM

- Move Pts. (continued)
  - Examples (11.2): Constrain to XZ Plane

Region moved

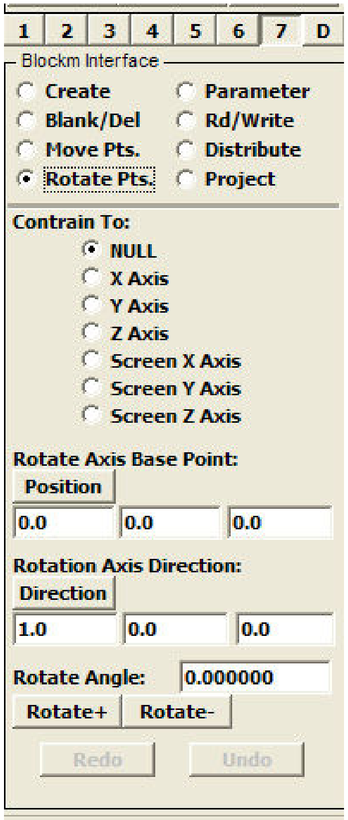


After



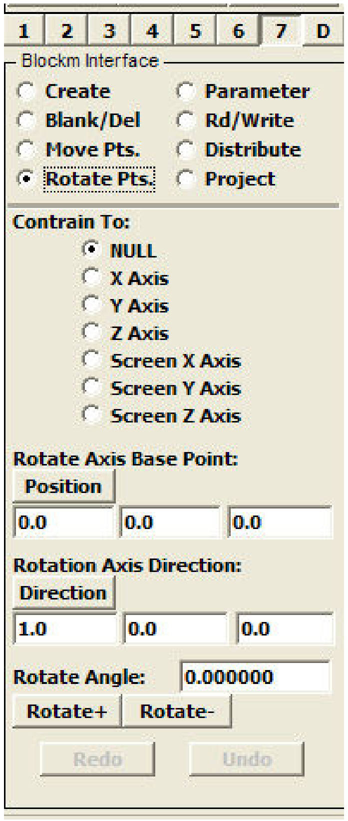
BlockM: Start Using BlockM

- Rotate Pts.
  - This feature lets the user rotate points, edges, faces, and regions about a point.
  - Similar to the Move Pts. feature, there are a number of options to choose from depending on the intend of the rotations.



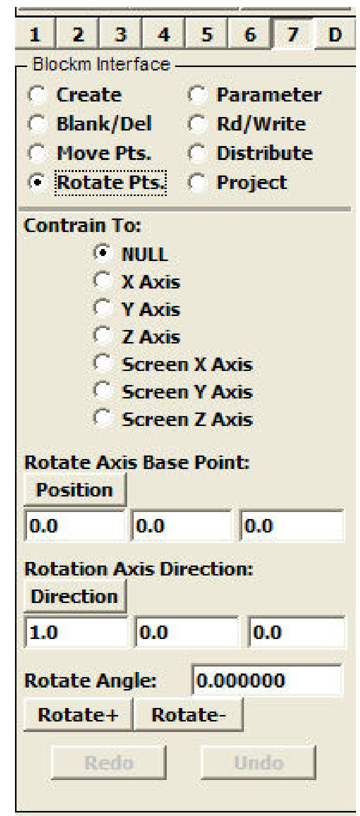
BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - To rotate:
    1. Select the point, edge, face or region using the computation window as explained previously.
    2. Select one of the constrained options to the right.
    3. Enter the "Rotation Axis Base Point". This point could be entered manually by its x, y, and z coordinates in the fields right below the "Position" button. Or the point could be picked from the Main window, by clicking on the Position button.
    4. Enter the "Rotation Axis Direction". This is the vector that points along the axis of rotation. Or the "Direction" button could be used to pick the vector graphically. If a constraint is chosen, there is no need to enter the direction.
    5. Enter the angle of rotation in degree in the "Rotate Angle" field.
    6. Click on either "Rotate+" or "Rotate-" to rotate counter-clockwise or clockwise direction.



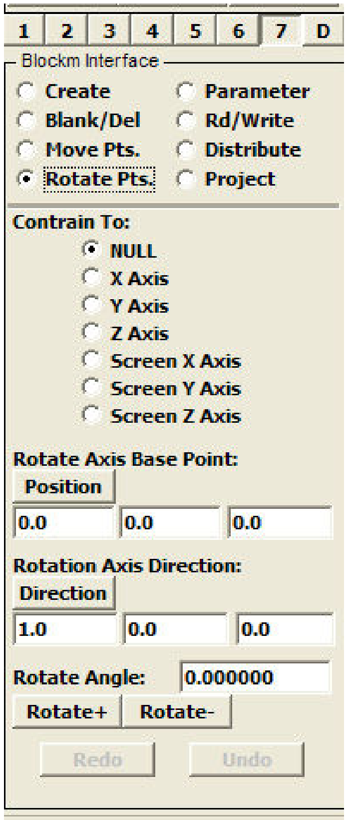
BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Constrain To:
    - NULL*
      - Nodes and elements are free to rotate about any axis.
    - X Axis
      - Nodes and elements could only be rotated about the global X-axis.
    - Y Axis
      - Nodes and elements could only be rotated about the global Y-axis.
    - Z Axis
      - Nodes and elements could only be rotated about the global Z-axis.



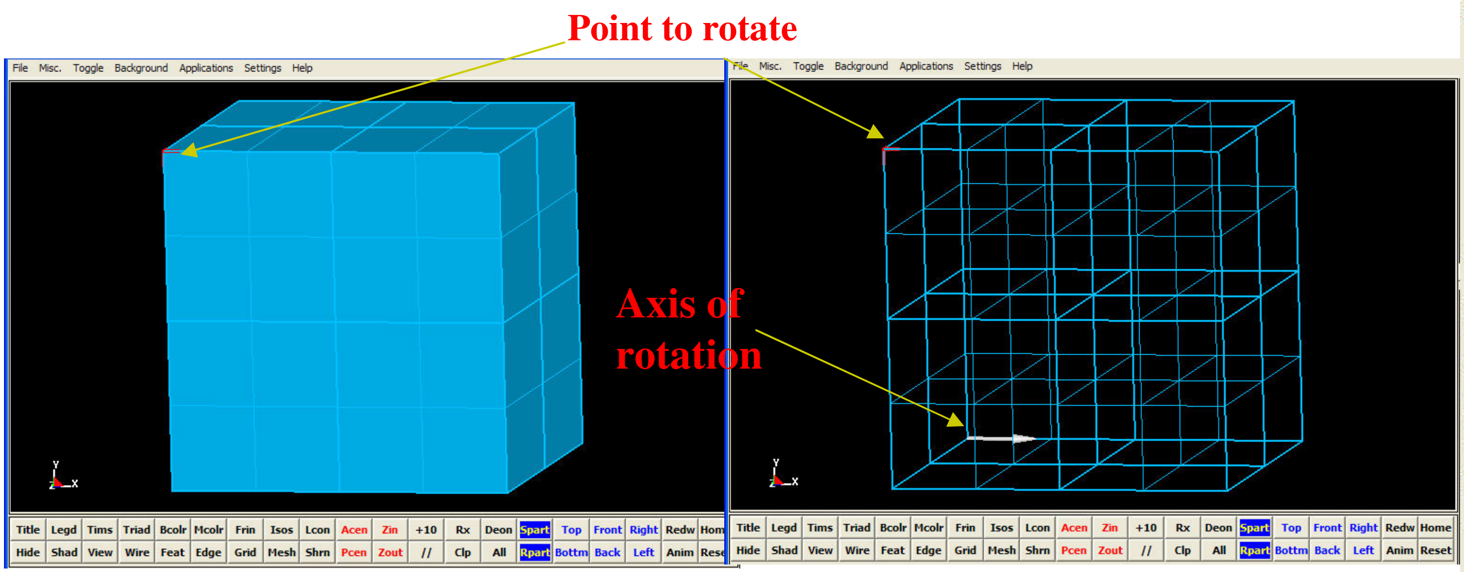
BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Constrain To:
    - Screen X Axis
      - Nodes and elements could only be rotated about the screen's X-axis.
    - Screen Y Axis
      - Nodes and elements could only be rotated about the screen's Y-axis.
    - Screen Z Axis
      - Nodes and elements could only be rotated about the screen's Z-axis. The screen's Z-axis is always coming out of the screen or normal to the screen.



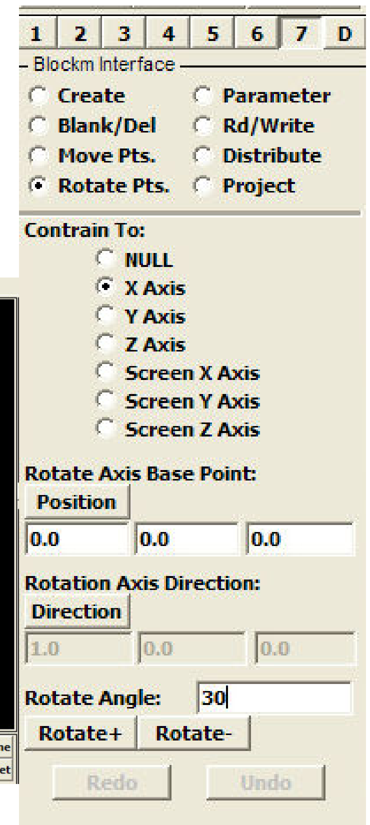
BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Example (12.0): Constrain to X Axis
    - Let’s rotate a point about the X-axis by 30 degree.



Shaded view

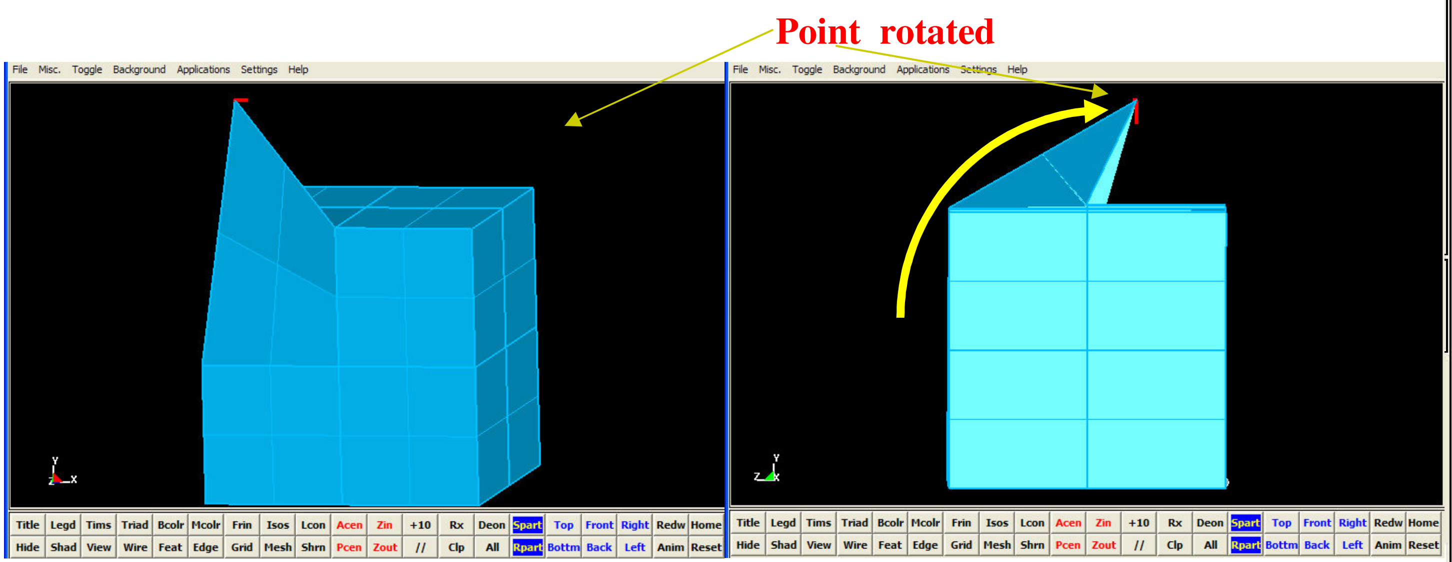
Wire frame view

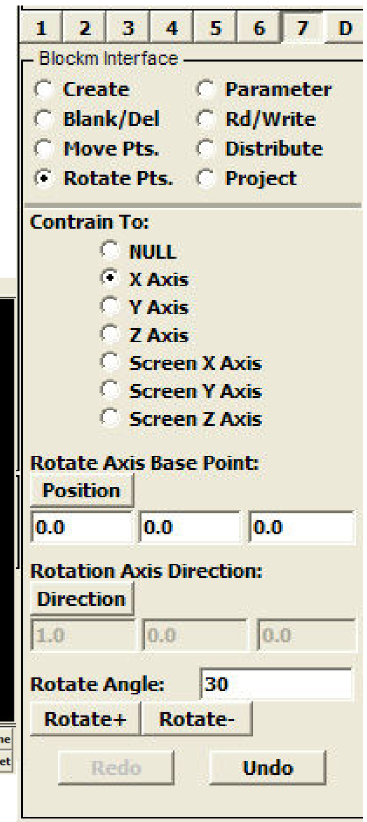


Before

BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Example (12.0): Constrain to X Axis





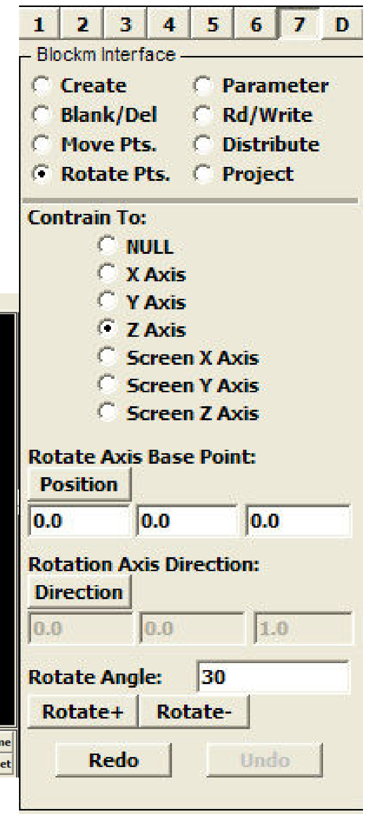
Original orientation

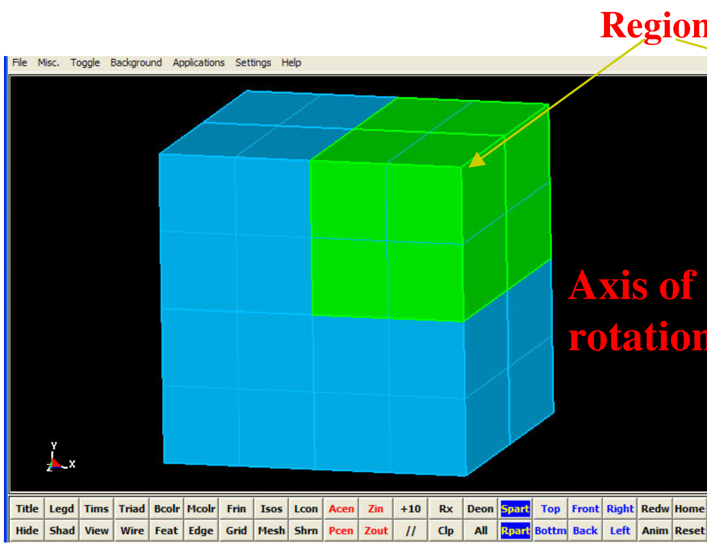
Normal to rotation orientation

After

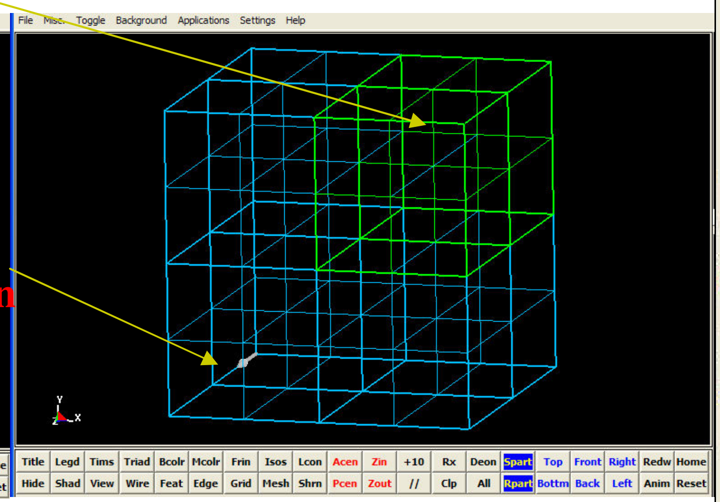
BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Example (12.1): Constrain to Z Axis
    - Let's rotate a region about the Z-axis by 30 degree.





Shaded view



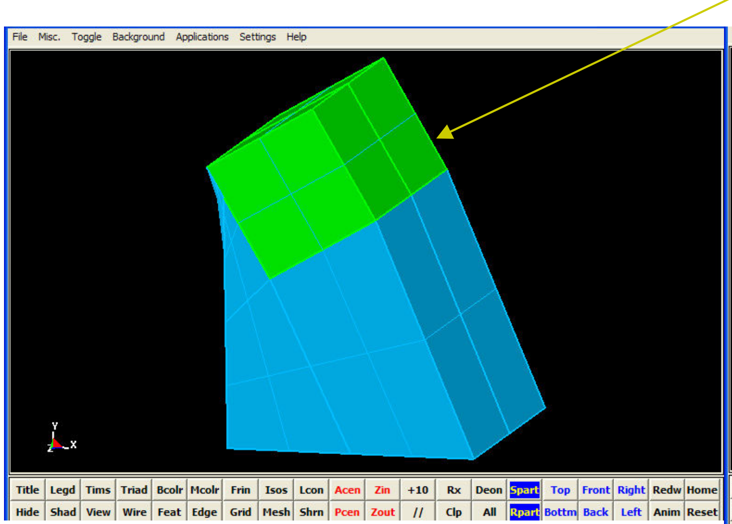
Wire frame view

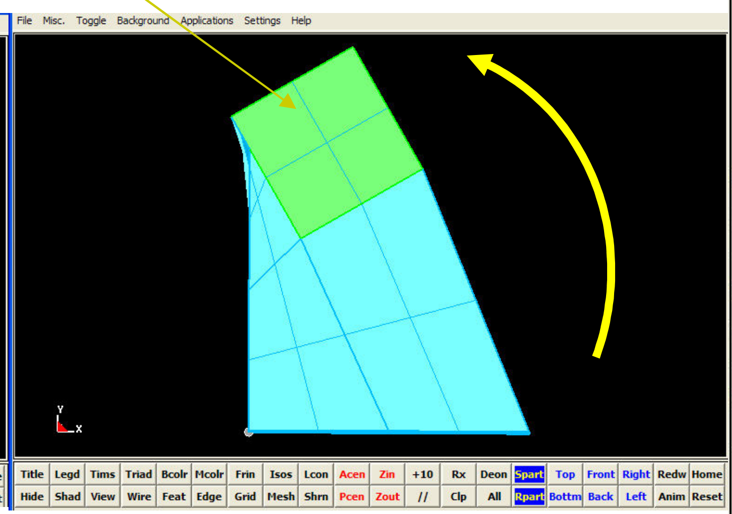
Before

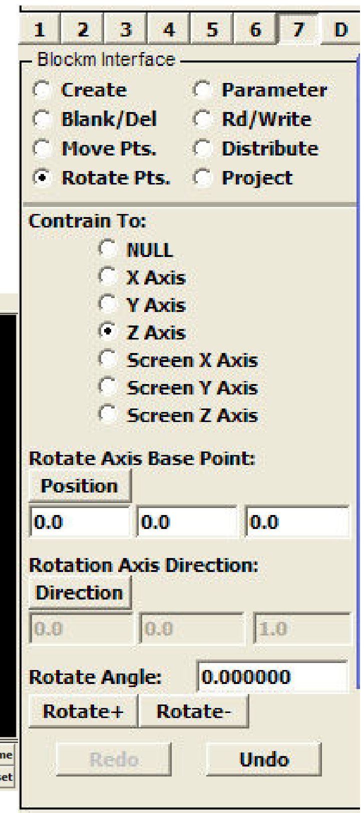
BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Example (12.1): Constrain to Z Axis

Region rotated







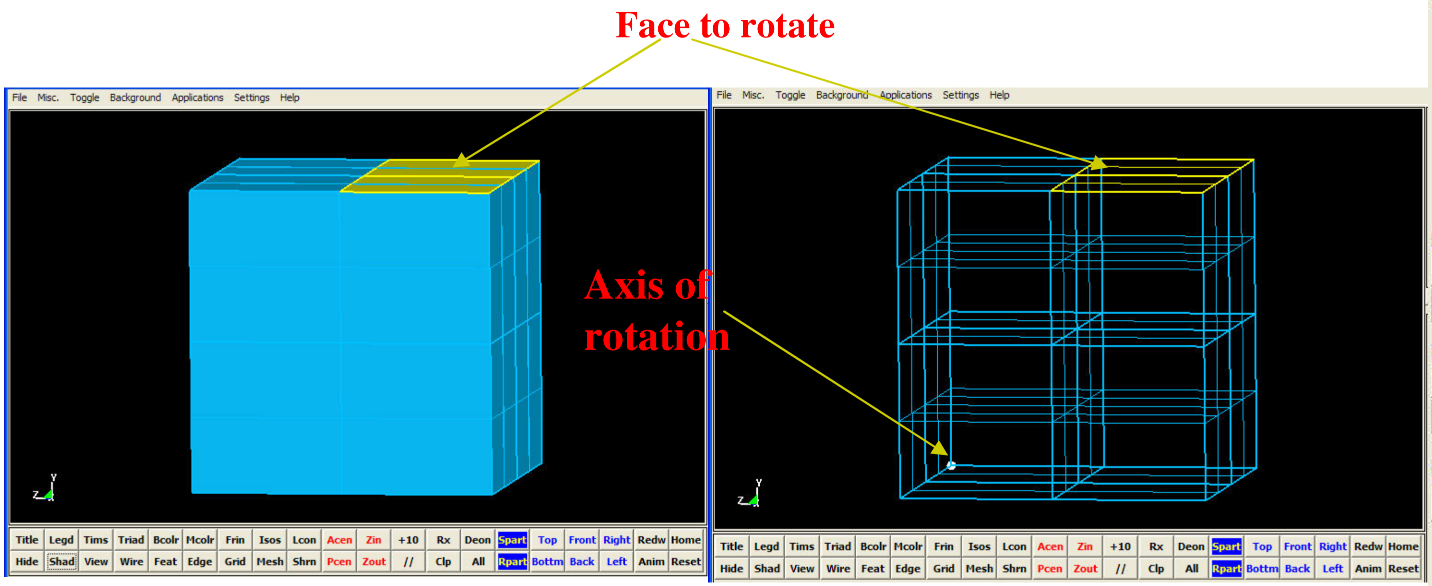
Original orientation

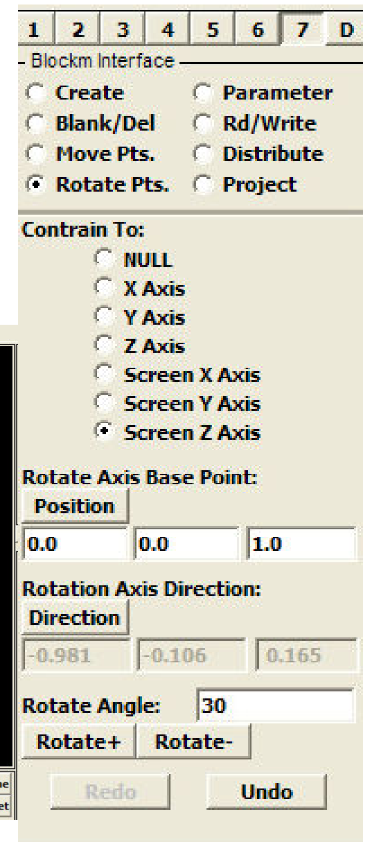
Normal to rotation orientation

After

BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Example (12.2): Constrain to Screen Z Axis
    - Let's rotate a face about the screen's Z-axis by 30 degree. The model is rotated 90 degree about the global Y direction. The rotation base point is set at z=1.0.





Shaded view

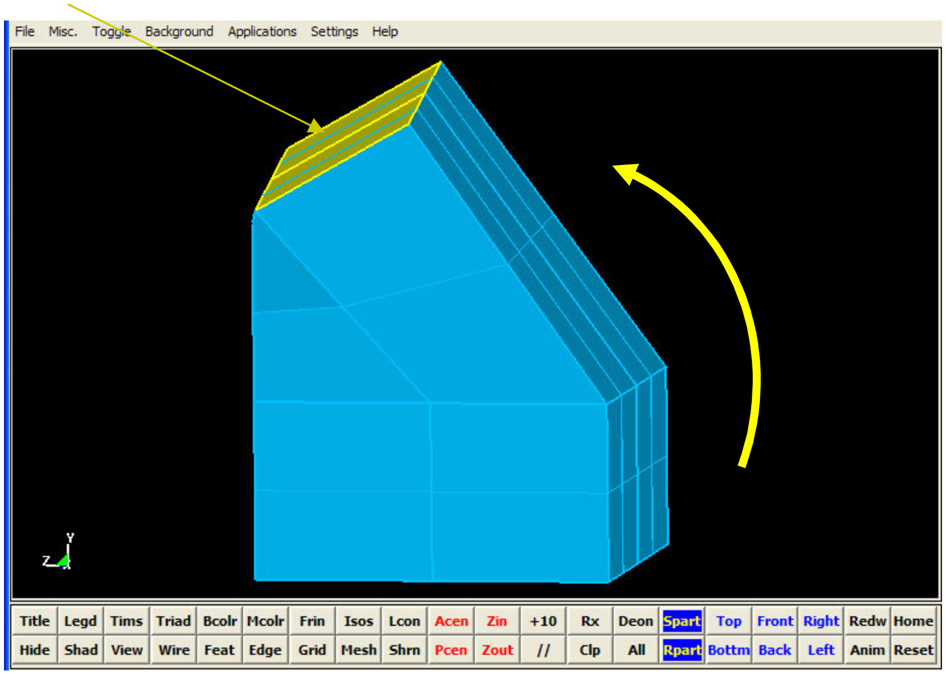
Wire frame view

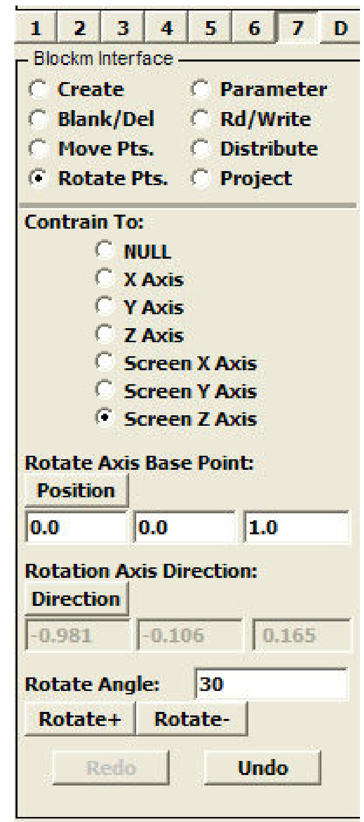
Before

BlockM: Start Using BlockM

- Rotate Pts. (continued)
  - Example (12.2): Constrain to Z Axis

Face rotated

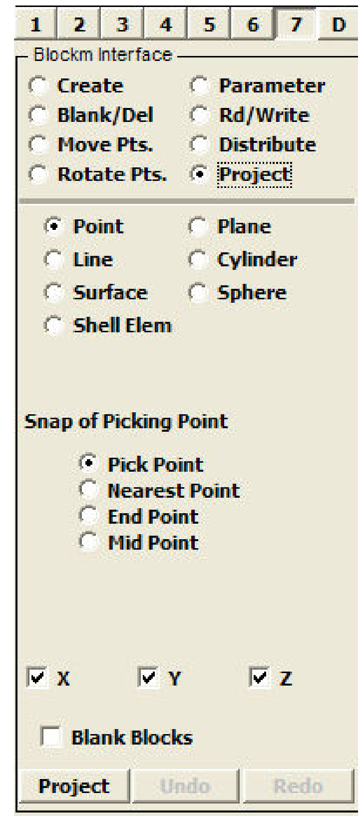




After

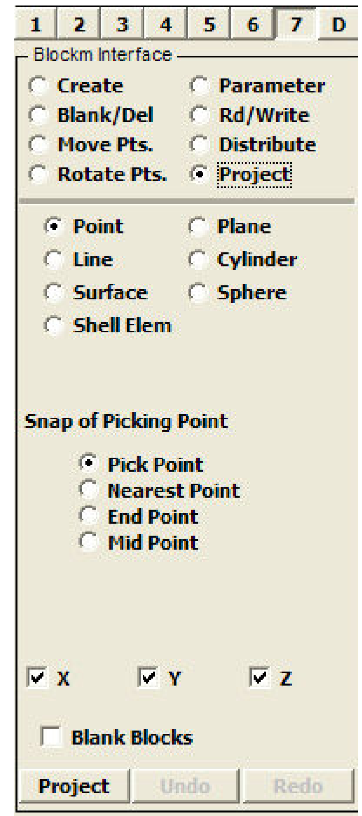
BlockM: Start Using BlockM

- Project
  - Early on, the concept of projection was introduced. It is here that this concept is implemented.
  - There are 7 different types of "target". More targets will be added in the future.
    - Point
    - Line
    - Surface
    - Shell Elem
    - Plane
    - Cylinder
    - Sphere
  - Individual components of the projection could be specified with X, Y, and Z radio buttons at the bottom of the menu.
  - Point, Line, Surface, and Shell Elem all requires external source. These must be generated outside of BlockM.
  - Plane, Cylinder, and Sphere are generated within BlockM.



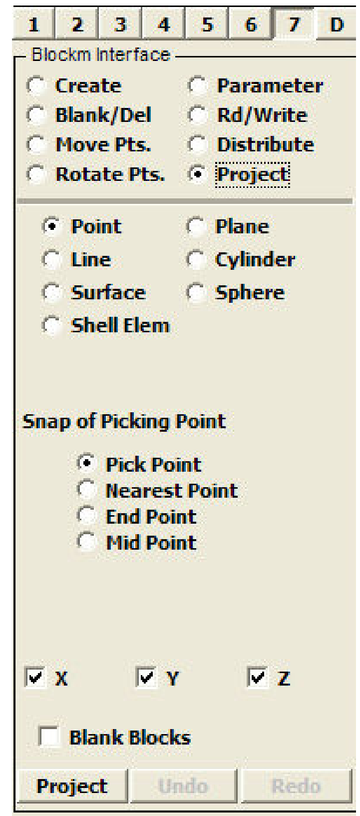
BlockM: Start Using BlockM

- Project (continued)
  - Point
    1. In order to project to a point, we need to generate a point. To do this, we need an “external” surface or curve such as igest surface and curve.
    2. To select a target point, simply click on a desired location on a surface or curve in the **Main** window.
       - It is good practice to project a point to a point, rather than an edge or face to a point.
    3. Select a “source” point from the Computation window by one the processes described above.
    4. Click on the Project button.



BlockM: Start Using BlockM

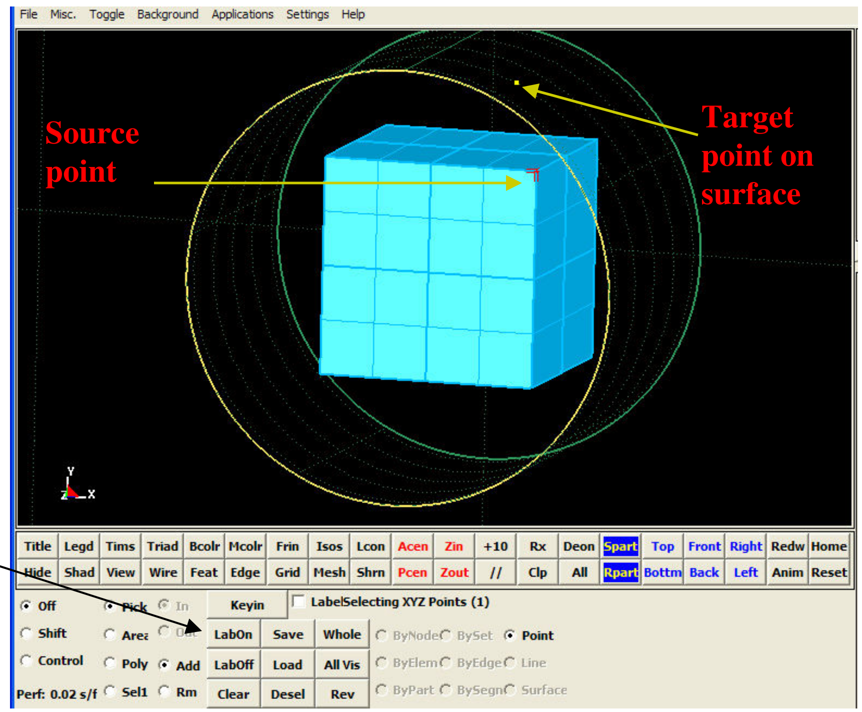
- Project (continued)
  - Point
    - There are 4 snapping options for point selection. This simplifies the point selection process, and it is more applicable to picking a point on a curve or line.
      - Pick Point – This is the default setting. The point selected is right where the cursor is positioned.
      - Nearest Point – This option allows the selection of point nearest to the cursor.
      - End Point – This options allows picking of the nearest end point on a curve or line.
      - Mid Point – Similar to the previous options, this allows for the selection of a mid-point of a line.

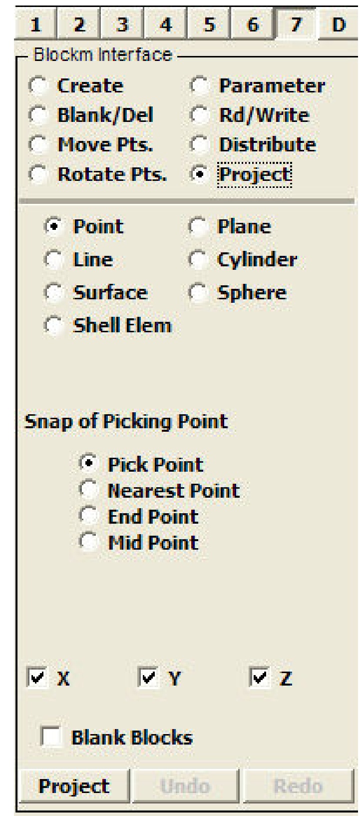


BlockM: Start Using BlockM

- Project (continued)
  - Point
    - Example (13.0) : Projecting point to a point on a surface

Point selection panel – contains options for picking and unpicking points .

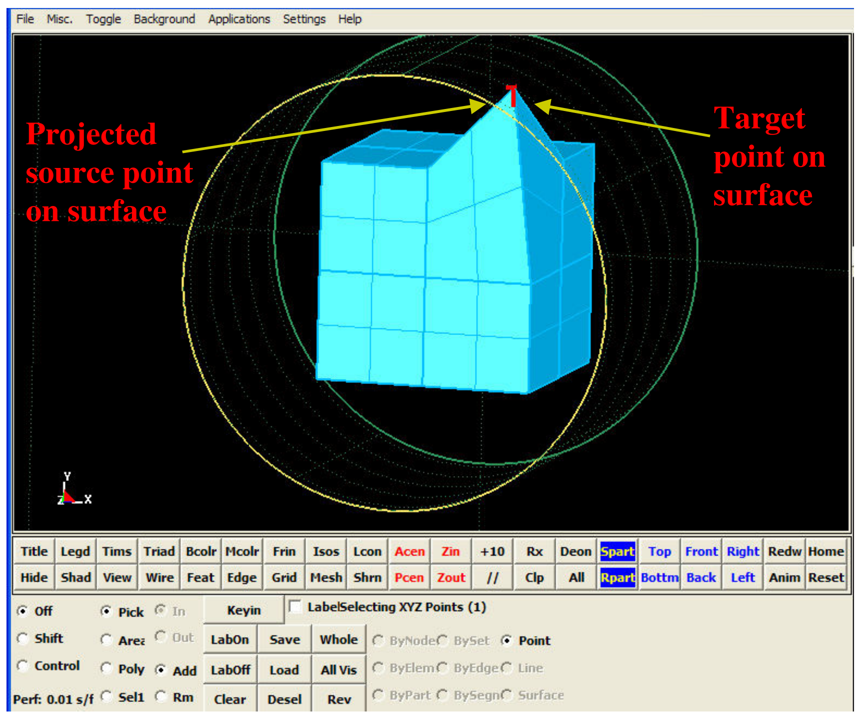


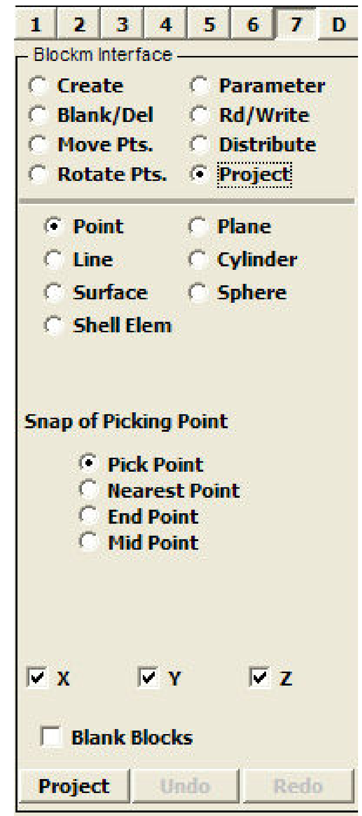


Before

BlockM: Start Using BlockM

- Project (continued)
  - Point
    - Example (13.0) : Projecting point to a point on a surface

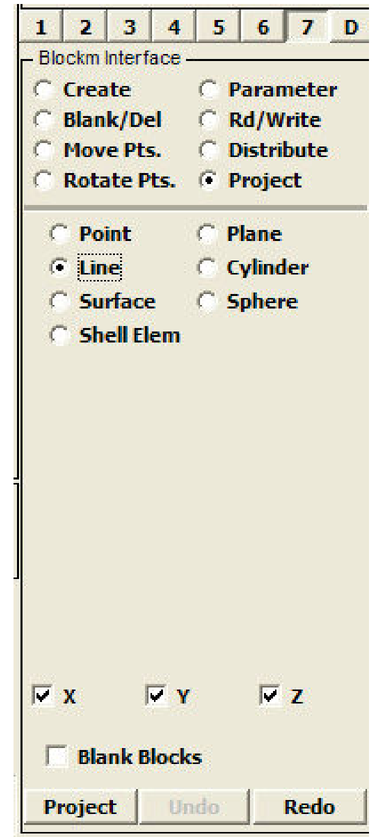




After

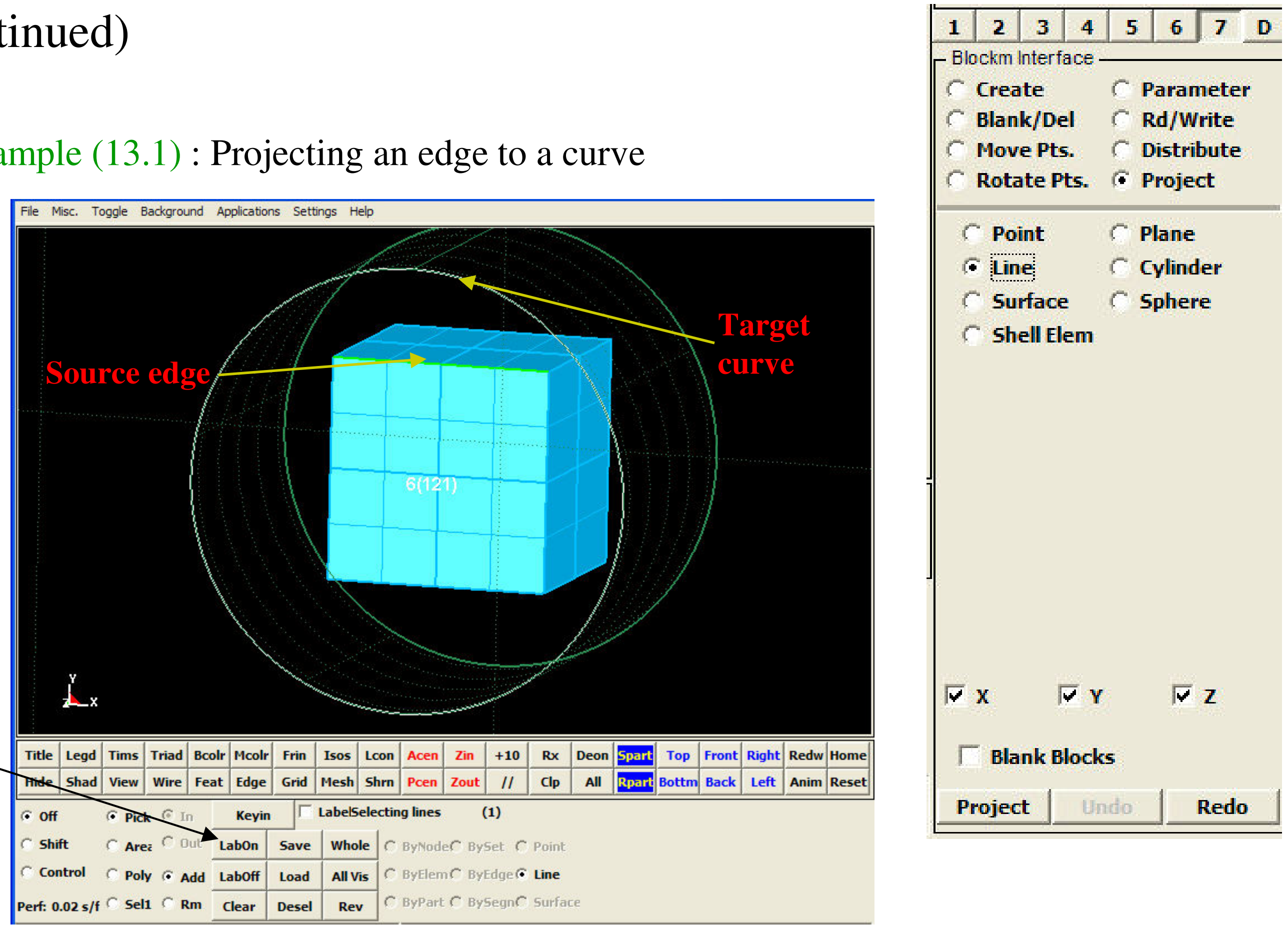
BlockM: Start Using BlockM

- Project (continued)
  - Line
    1. Much like that for the Point, we need an external curve or line such as igest curve. A curve could be generated in the **Curves** interface on pages 2 and 7.
    2. To select a target curve, simply click on a desired curve in the **Main** window.
       - It is good practice to project a point or an edge to a line or curve, rather than a face or a region to a line.
    3. Select a source edge from the Computation window by one the processes described above.
    4. Click on the Project button.



BlockM: Start Using BlockM

- Project (continued)
  - Line
    - Example (13.1) : Projecting an edge to a curve

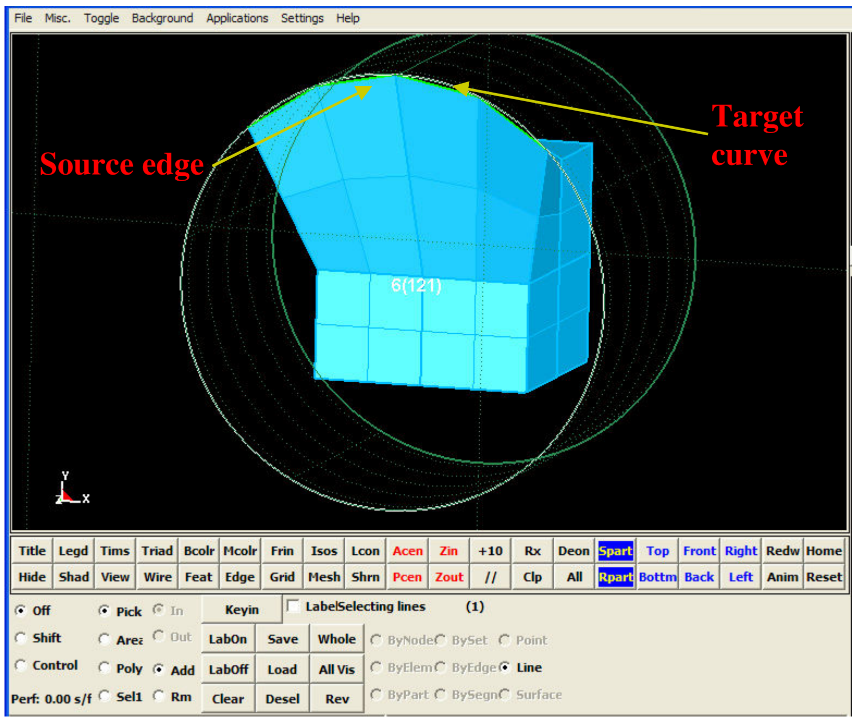


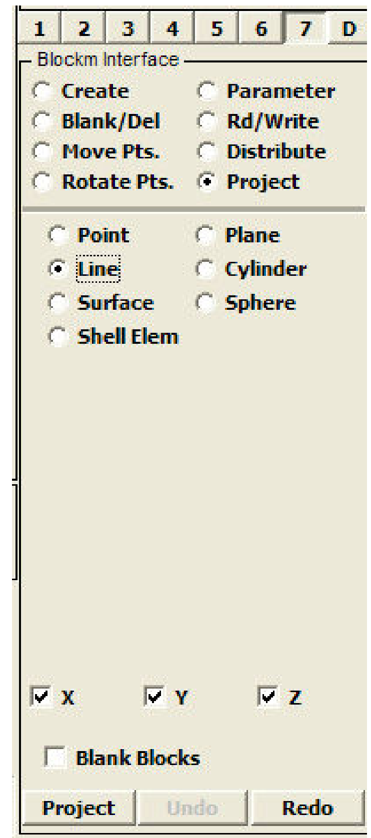
Line selection panel – contains options for picking and unpicking lines and curves .

Before

BlockM: Start Using BlockM

- Project (continued)
  - Line
    - Example (13.1) : Projecting an edge to a curve

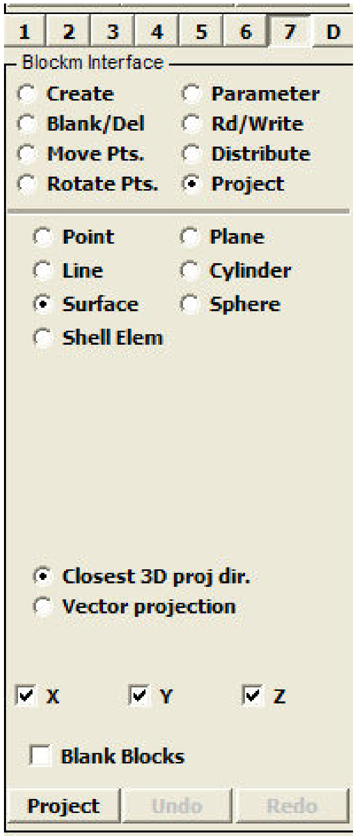




After

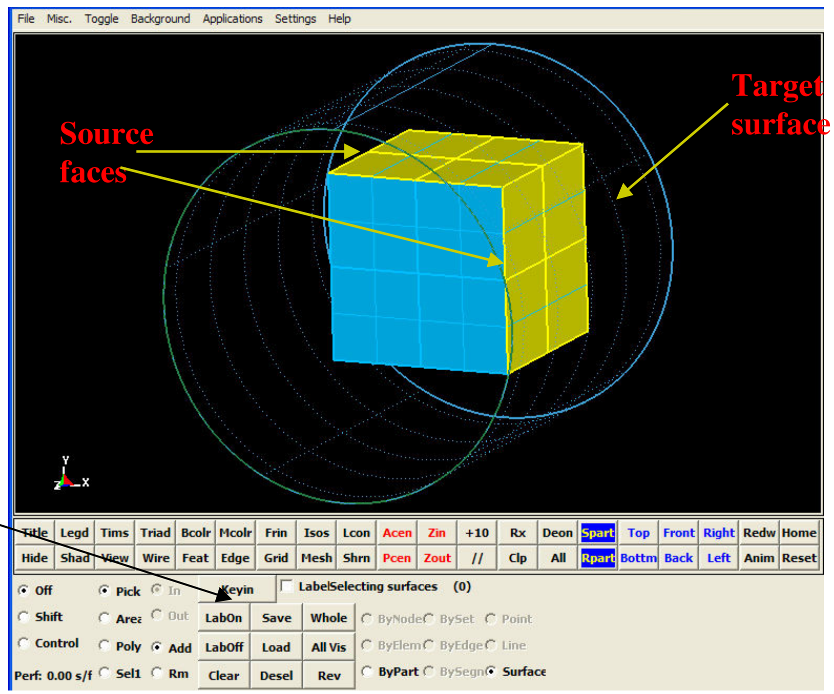
BlockM: Start Using BlockM

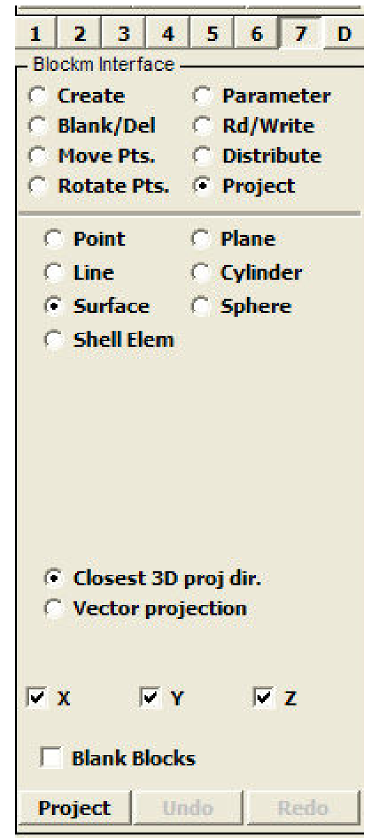
- Project (continued)
  - Surface
    1. Like the previous 2 projection options, we need an external surface such as igest surface. A surface could be generated from curves in the **Surface** interface on page 7.
    2. To select a target surface or surfaces, simply click on a desired ones in the **Main** window.
    - It is good practice to project a point, edge, or face to a surface. It is not a good idea to project a region to a surface.
    3. Select a “source” point, edge or face from the **Computation** window by one the processes described above.
    4. Click on the Project button.



BlockM: Start Using BlockM

- Project (continued)
  - Surface
    - Example (13.2) : Projecting 2 faces to a cylindrical surface.
      - Use Closest 3D proj dir.



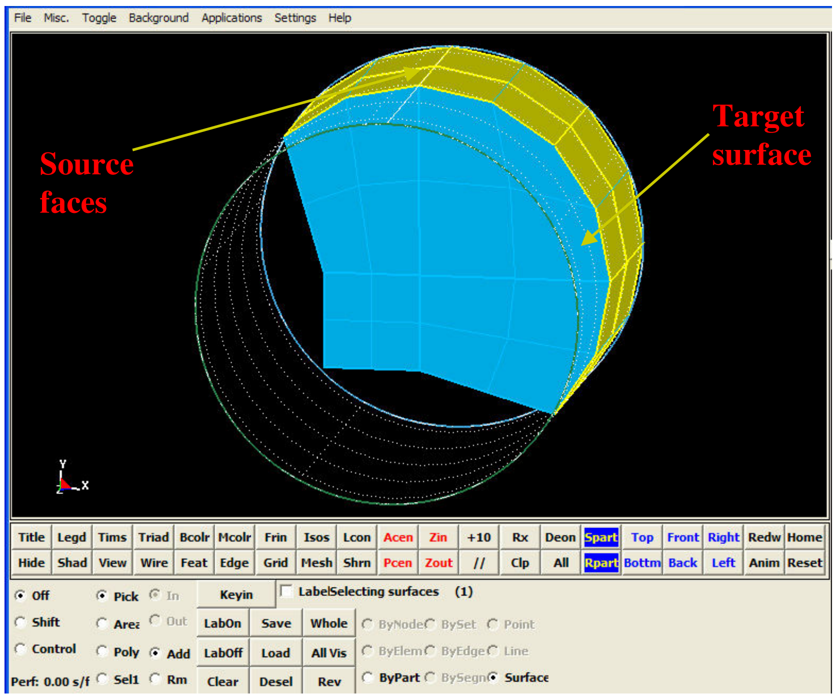


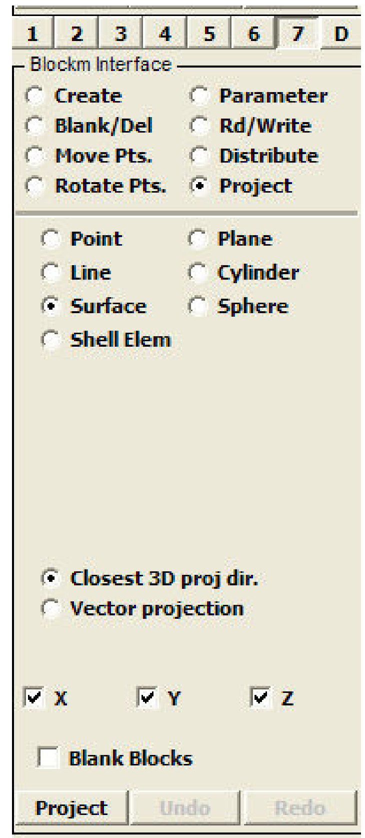
Surface selection panel – contains options for picking and unpicking surface .

Before

BlockM: Start Using BlockM

- Project (continued)
  - Surface
    - Example (13.2) : Projecting faces to a surface

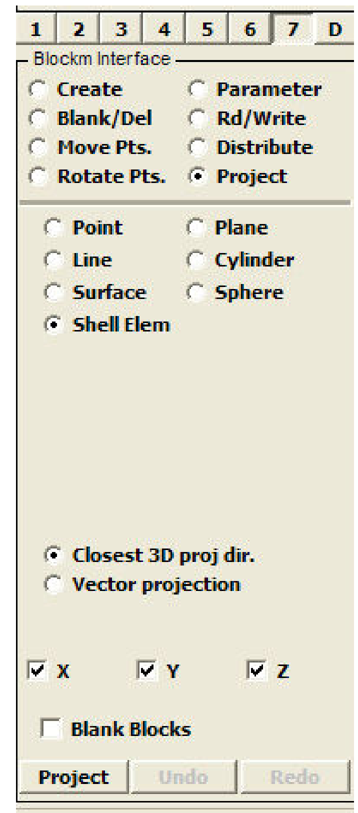




After

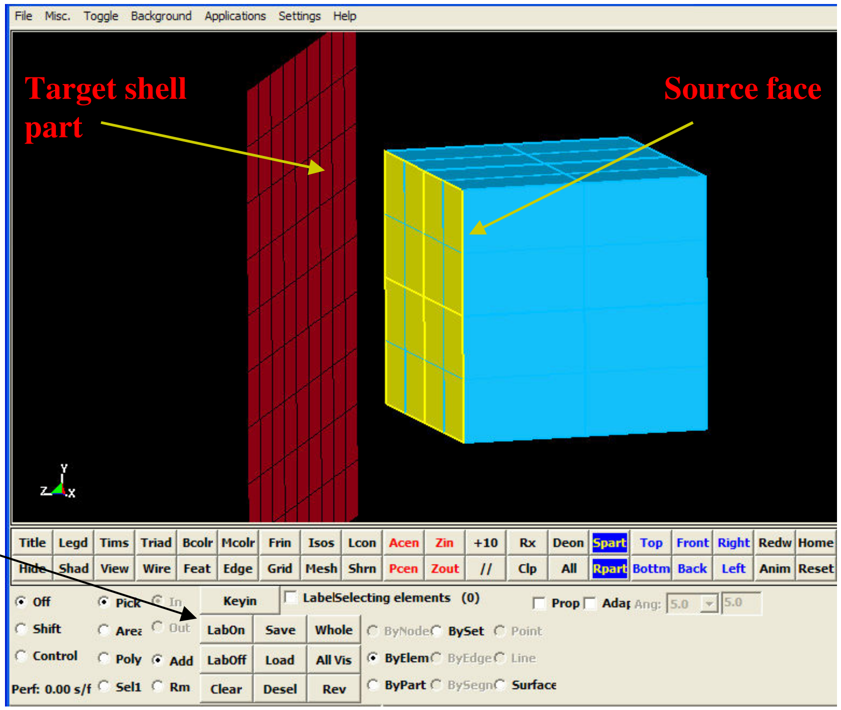
BlockM: Start Using BlockM

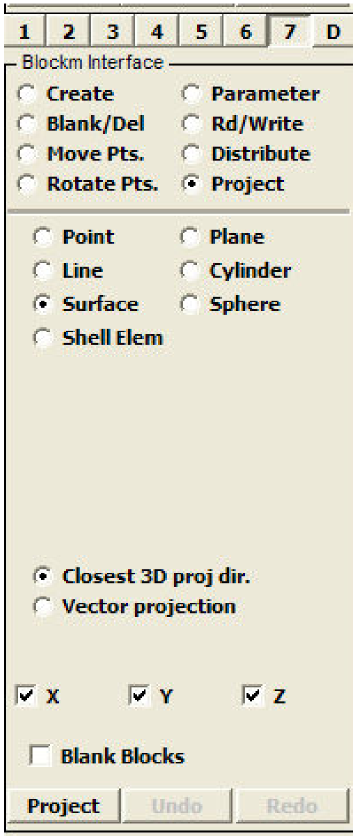
- Project (continued)
  - Shell Elem
    1. This option requires some sort of shell elements to be generated first.
    2. To select target shell elements, simply click on a desired ones in the **Main** window. Shell elements could be selected with either propagation, area, or part option.
       - It is good practice to project a point, edge, or face to shell elements. It is not a good idea to project a region to shell elements, though.
    3. Select a source point, edge or face from the **Computation** window by one the processes described above.
    4. Click on the Project button.



BlockM: Start Using BlockM

- Project (continued)
  - Surface
    - Example (13.3) : Projecting a face to shell elements.
      - Use Closest 3D proj dir



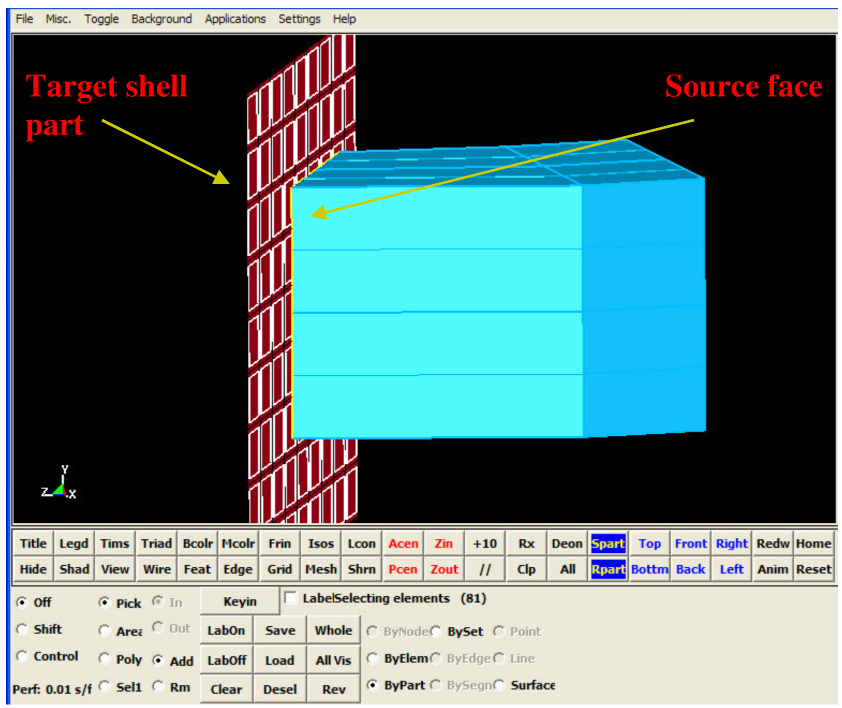


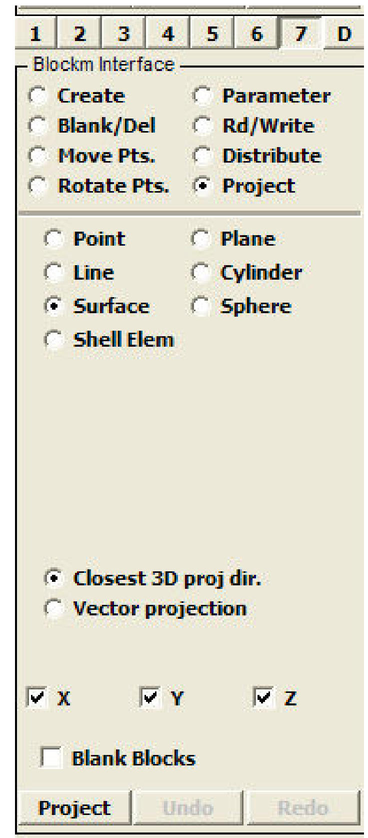
Element selection panel – contains options for picking and unpicking shell elements .

Before

BlockM: Start Using BlockM

- Project (continued)
  - Surface
    - Example (13.3) : Projecting a face to shell elements.
      - Use Closest 3D proj dir





After

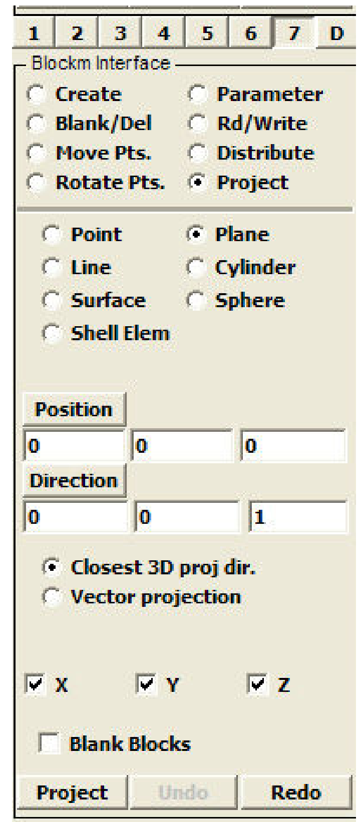
BlockM: Start Using BlockM

- Project (continued)
  - Plane
    - Unlike the previous 4 options, the Plane is generated in BlockM. By default, the plane is at x=0, y=0, z=0 and the normal vector is along the z-axis.

    1. Position the plane by either entering the XYZ coordinates in the fields to the right or click on the Position button to select with a mouse.
    2. Specify the orientation of the plane by either entering the vector components in the direction fields, or click on the Direction button to visually select.

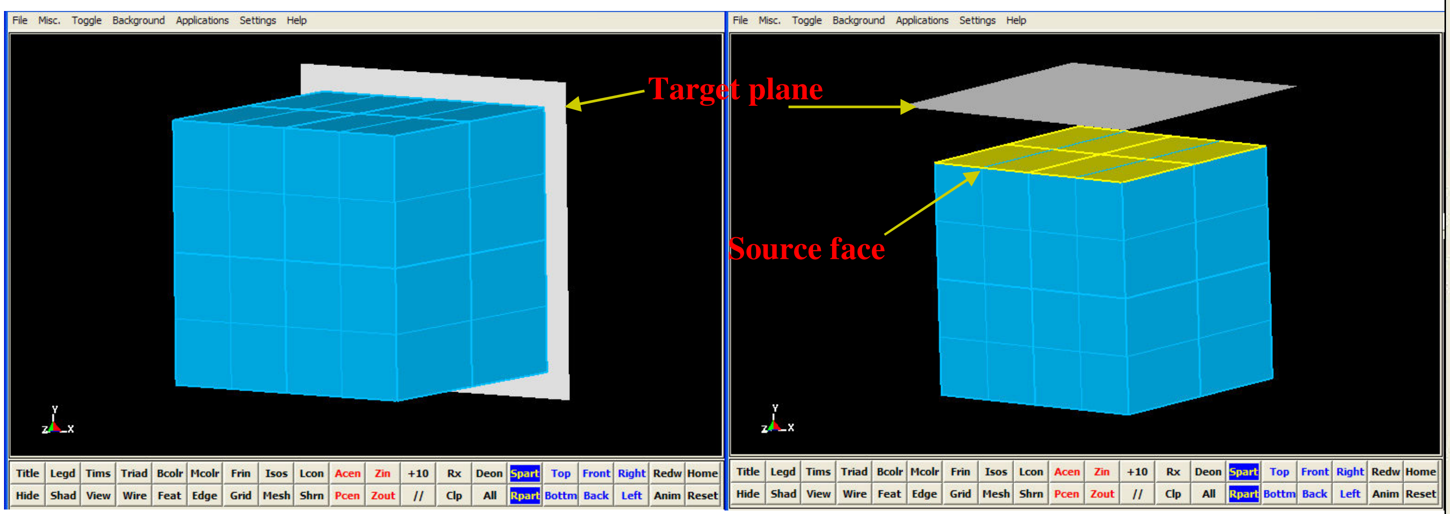
    - Plane is an infinite plane, although visually the plane is “clipped”.

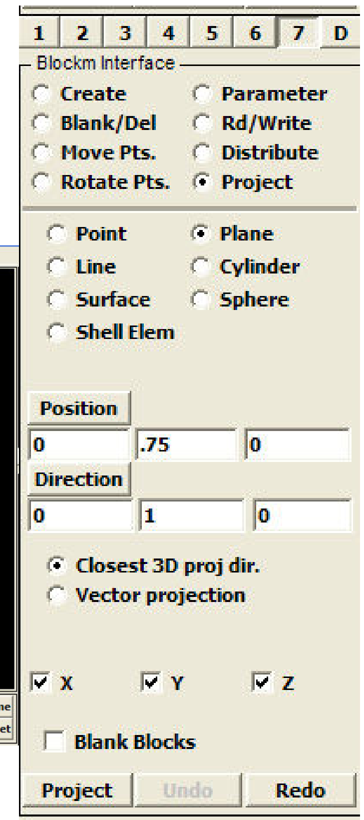
    3. Select a source point, edge or face from the **Computation** window by one the processes described above.
    4. Click on the Project button.



BlockM: Start Using BlockM

- Project (continued)
  - Plane
    - Example (13.4) : Projecting a face to a plane.
      - Use Closest 3D proj dir





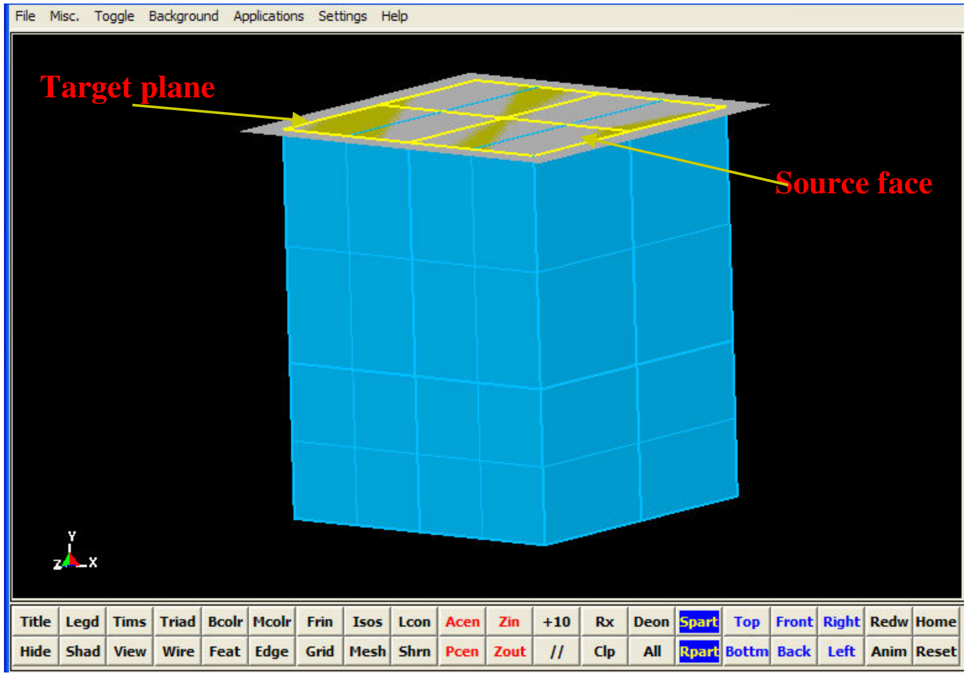
Plane in default position and orientation

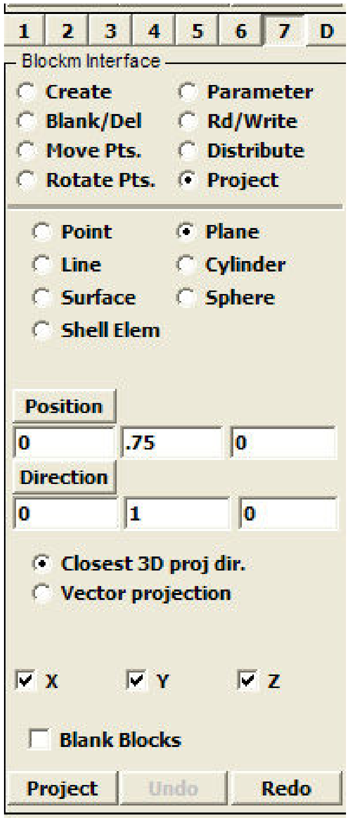
Plane is moved to x=0, y=.75, z=0 and the normal is oriented along the y-direction.

Before

BlockM: Start Using BlockM

- Project (continued)
  - Plane
    - Example (13.4) : Projecting a face to a plane.
      - Use Closest 3D proj dir





After

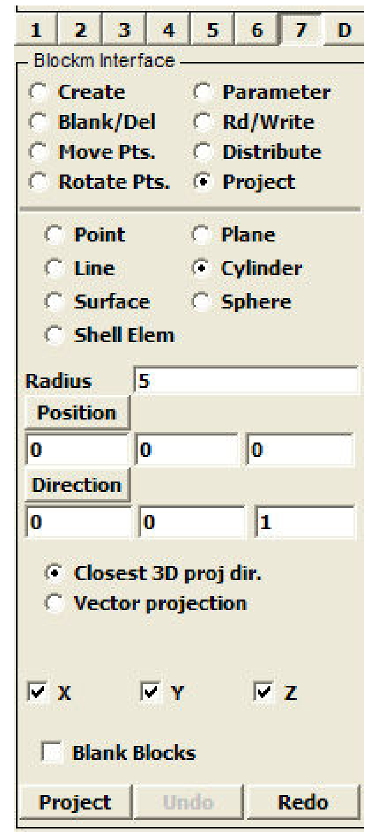
BlockM: Start Using BlockM

- Project (continued)
  - Cylinder
    - Similar to the Plane option above, the cylinder is generated in BlockM. By default, the cylinder is at x=0, y=0, z=0, the length is along the z-axis, and radius=5.

    1. Specify the cylinder's radius in the Radius field.
    2. Position the cylinder by either entering the XYZ coordinates in the fields to the right or click on the Position button to select with a mouse.
    3. Specify the orientation of the cylinder's axial direction by either entering the vector components in the direction fields, or click on the Direction button to visually select.

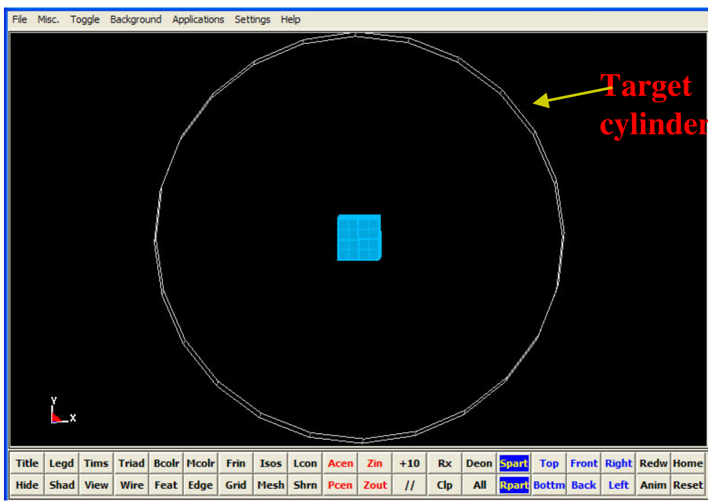
    - A cylinder has infinite length. It is visually clipped.

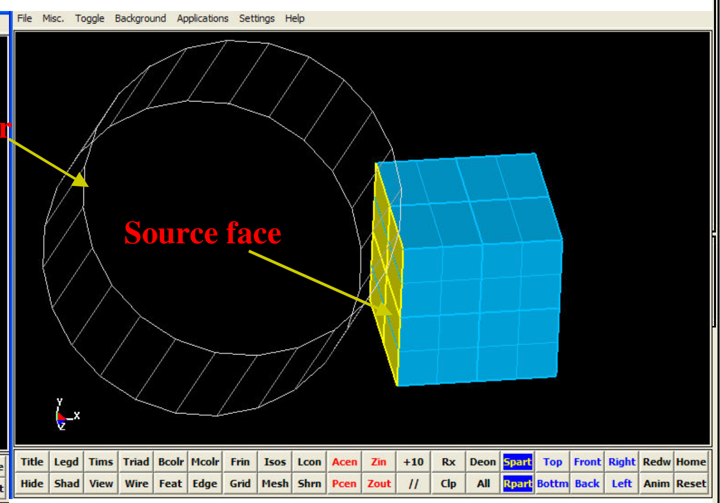
    4. Select a source point, edge or face from the **Computation** window by one the processes described above.
    5. Click on the Project button.

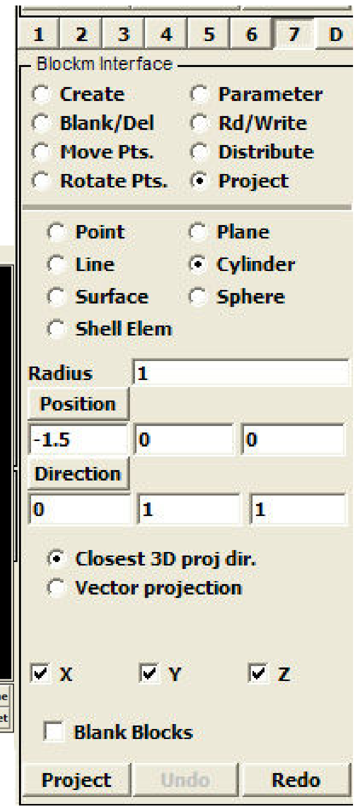


BlockM: Start Using BlockM

- Project (continued)
  - Cylinder
    - Example (13.5) : Projecting a face to a cylindrical surface.
      - Use Closest 3D proj dir







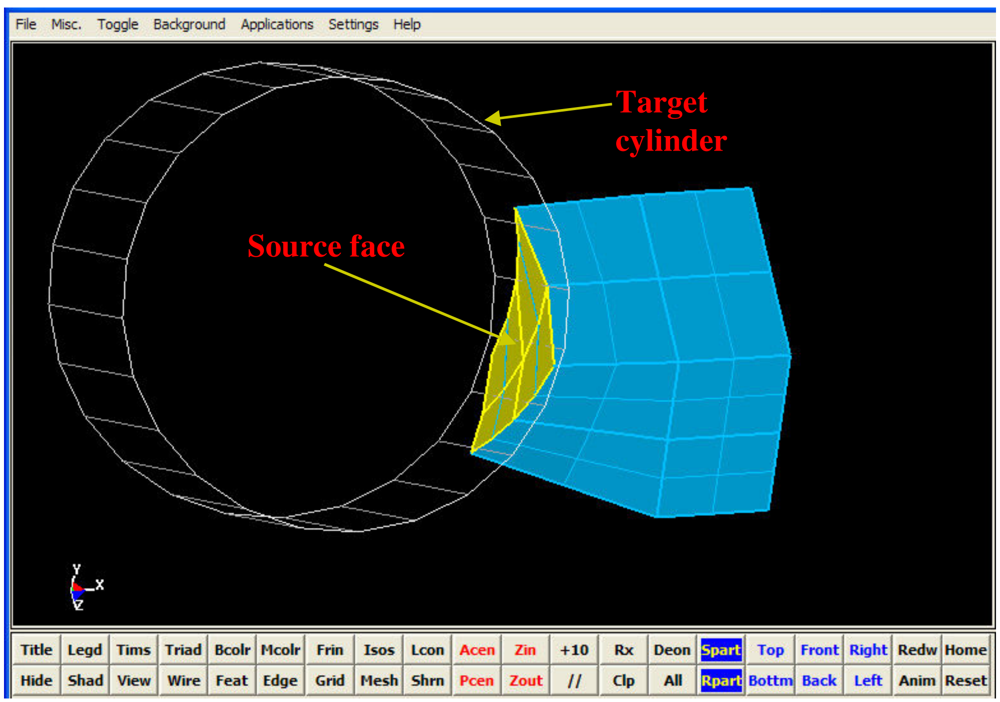
Cylinder with default radius, position, and orientation

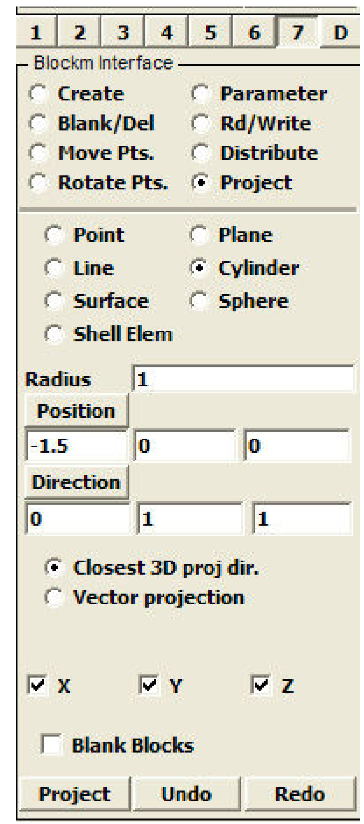
Cylinder is positioned at x=-1.5, y=0, z=0, radius=1.0, and axial direction is 45-degree on the YZ plane.

Before

BlockM: Start Using BlockM

- Project (continued)
  - Cylinder
    - Example (13.5) : Projecting a face to a cylindrical surface.
      - Use Closest 3D proj dir



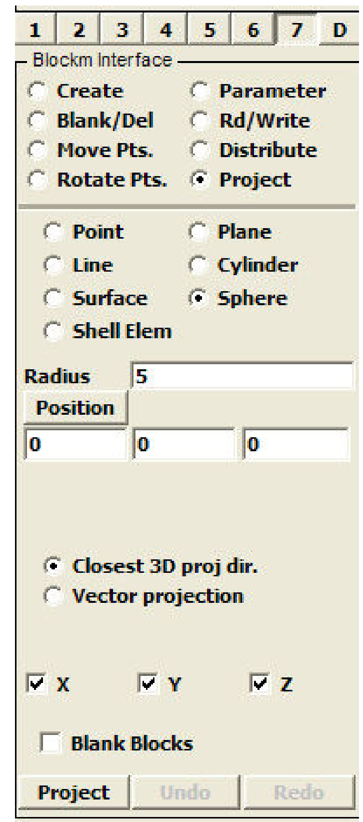


After

BlockM: Start Using BlockM

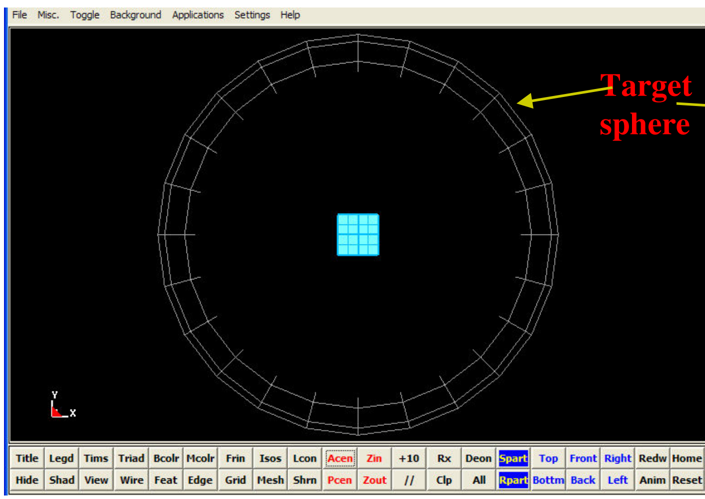
- Project (continued)
  - Sphere
    - Similar to the Plane and Cylinder options above, the sphere is also generated in BlockM. By default, the sphere is positioned at x=0, y=0, z=0, and a radius=5.

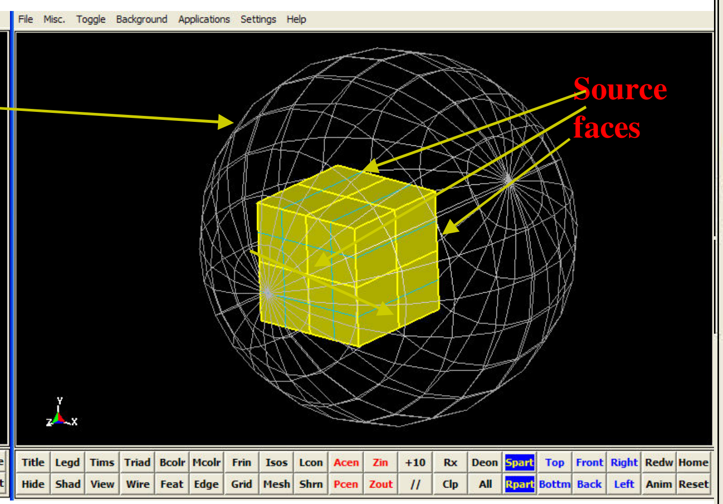
1. Specify the cylinder’s radius in the Radius field.
2. Position the cylinder by either entering the XYZ coordinates in the fields to the right or click on the Position button to select with a mouse.
4. Select a source point, edge or face from the **Computation** window by one the processes described above.
5. Click on the Project button.

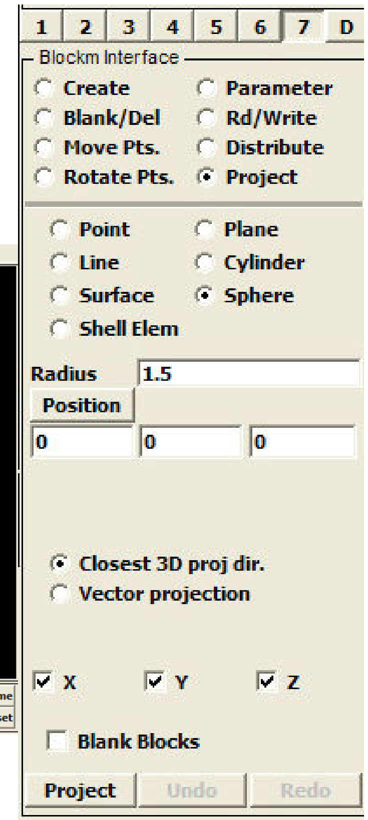


BlockM: Start Using BlockM

- Project (continued)
  - Sphere
    - Example (13.6) : Projecting 3 faces to a spherical surface.
      - Use Closest 3D proj dir







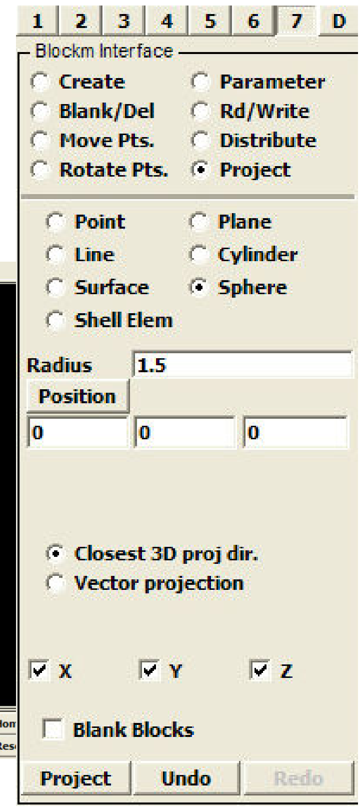
Spherical shell with default radius and position.

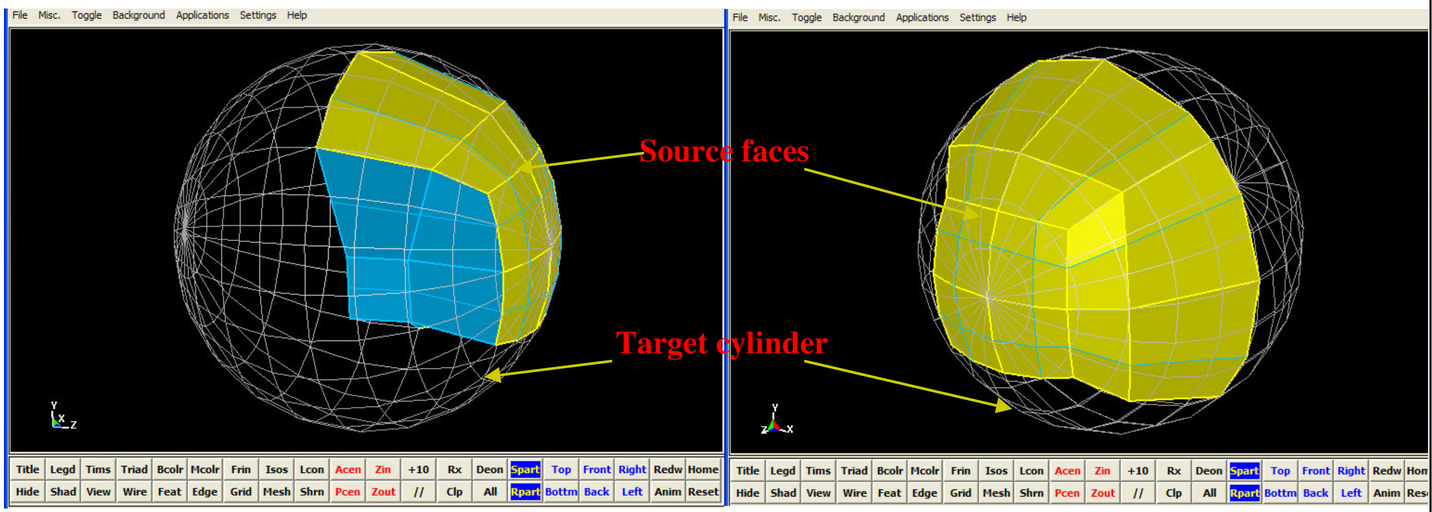
Sphere is positioned at x=0, y=0, z=0, and radius=1.5.

Before

BlockM: Start Using BlockM

- Project (continued)
  - Sphere
    - Example (13.6) : Projecting 3 faces to a spherical surface.
      - Use Closest 3D proj dir



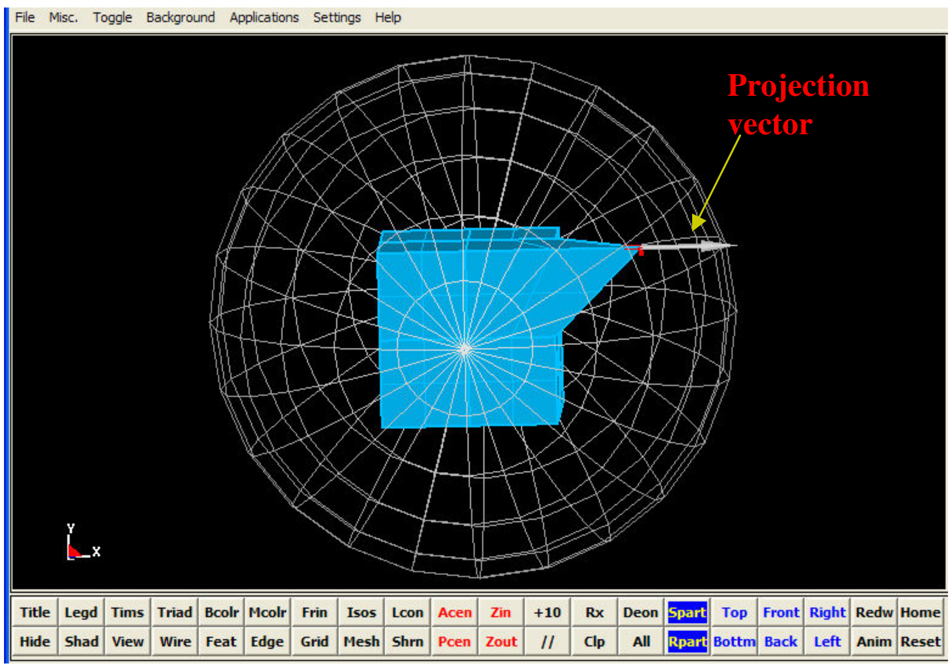


Side view **Original view**

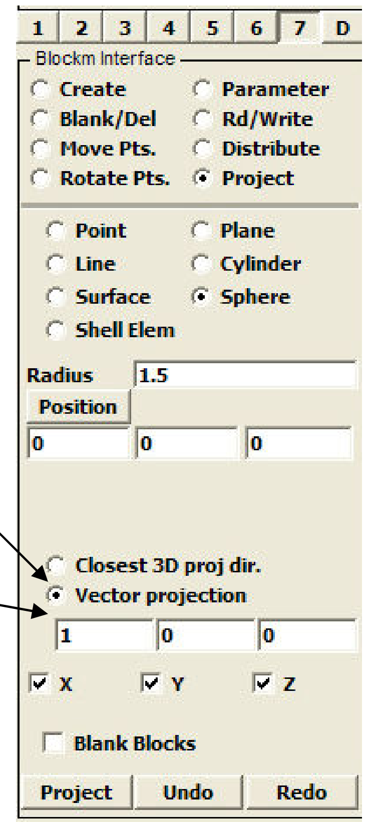
After

BlockM: Start Using BlockM

- Project (continued)
  - Sphere
    - Example (13.7) : Projecting a to a spherical surface.
      - Use Vector projection: X-direction



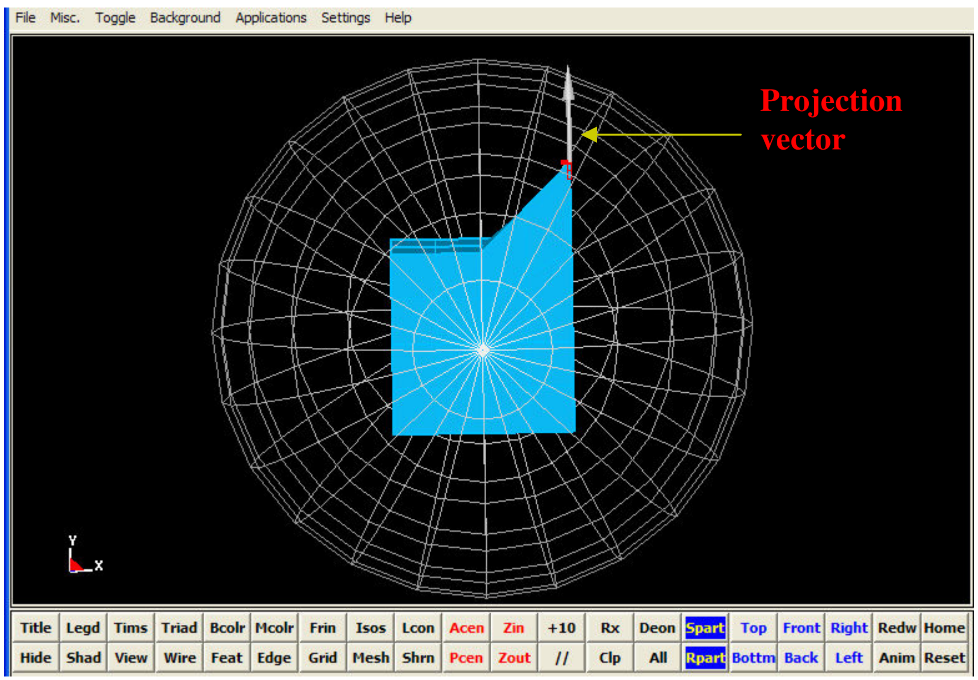
1. Click here
2. Specify projection vector



After

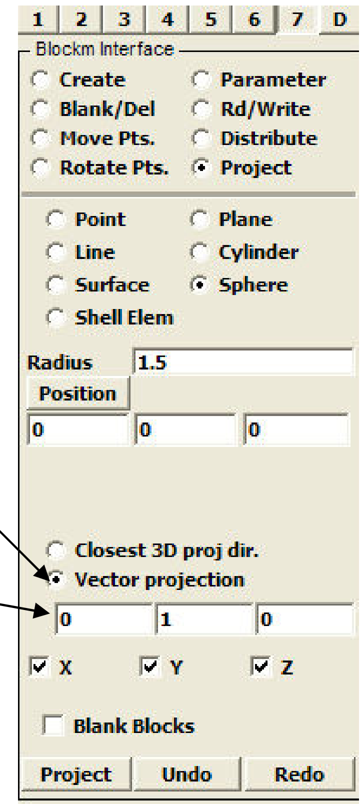
BlockM: Start Using BlockM

- Project (continued)
  - Sphere
    - Example (13.8) : Projecting a to a spherical surface.
      - Use Vector projection: Y-direction



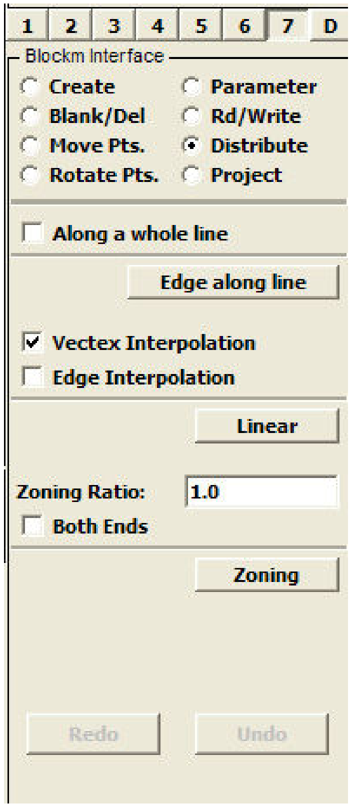
After

1. Click here
2. Specify projection vector



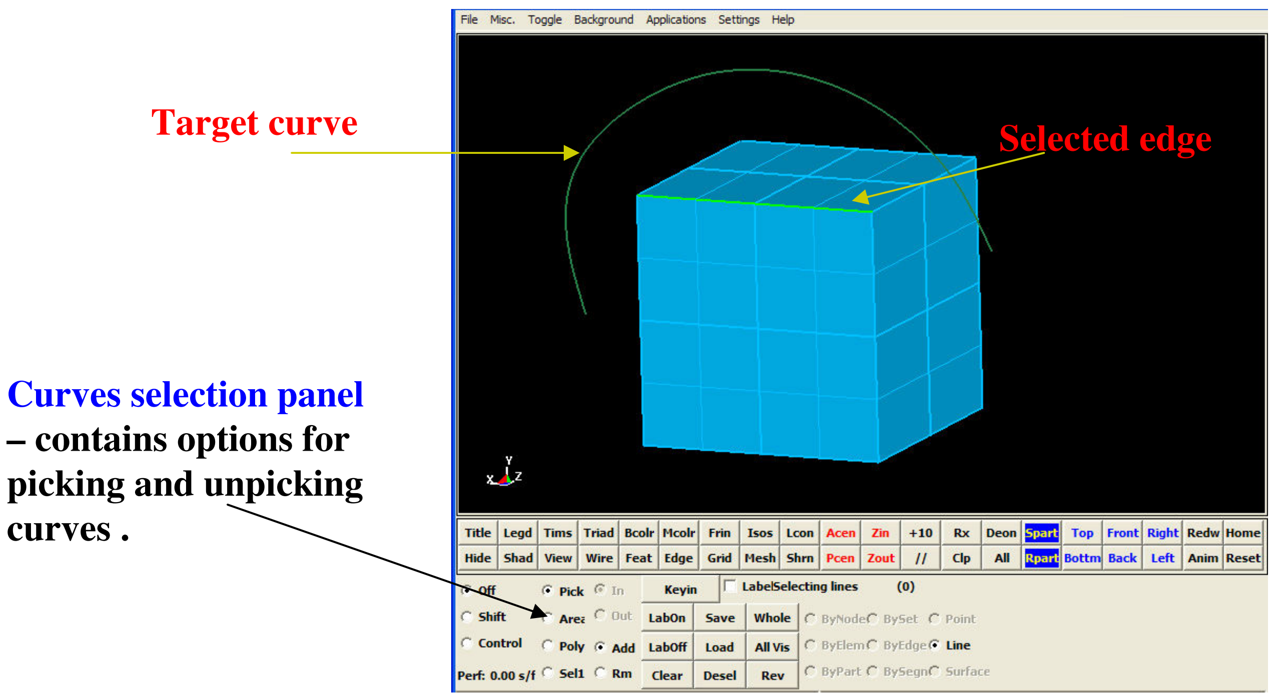
BlockM: Start Using BlockM

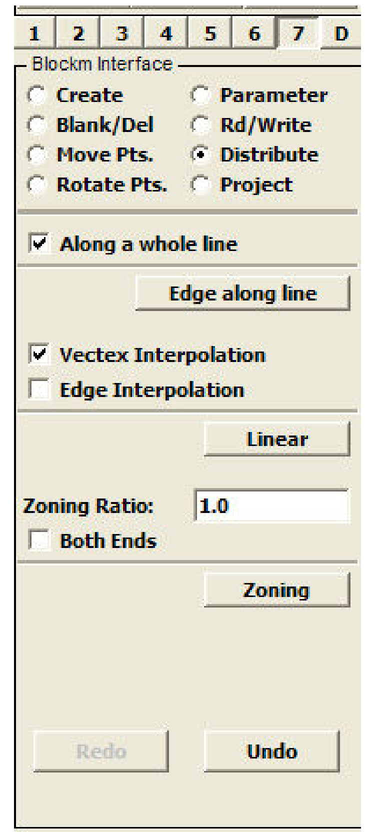
- Distribute
  - This feature allows for added mesh refinement by redistributing nodes with minimum user's inputs. Currently, the distribution feature is limited to edges only. However, nodal distribution for faces and regions will be added shortly.
  - There are 3 options are available:
    - Edge along line
      - This projects an edge to a curve and then distributes the nodes evenly along it.
      1. Select an edge using the Computation window
      2. Click on the "Along a whole line" radio button
      3. Select a curve from the Main window.
      4. To apply, click on the "Edge along line" button.



BlockM: Start Using BlockM

- Distribute (continued)
  - Edge along line
    - Example (14.0) : Distribute nodes along a curve.

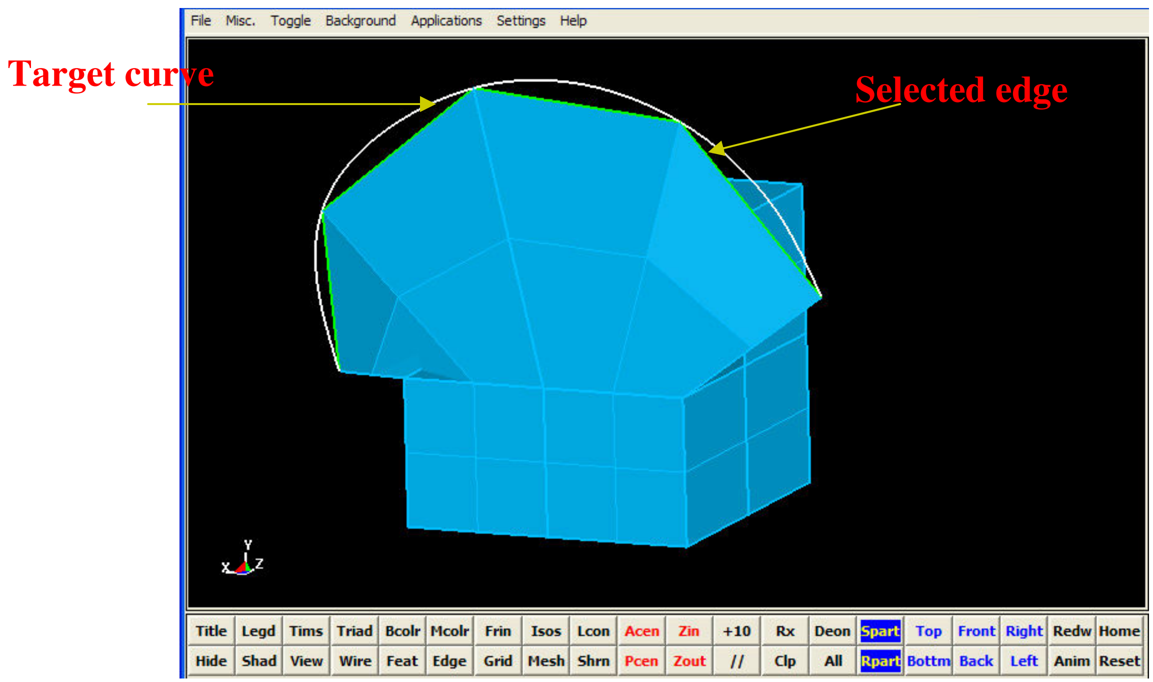


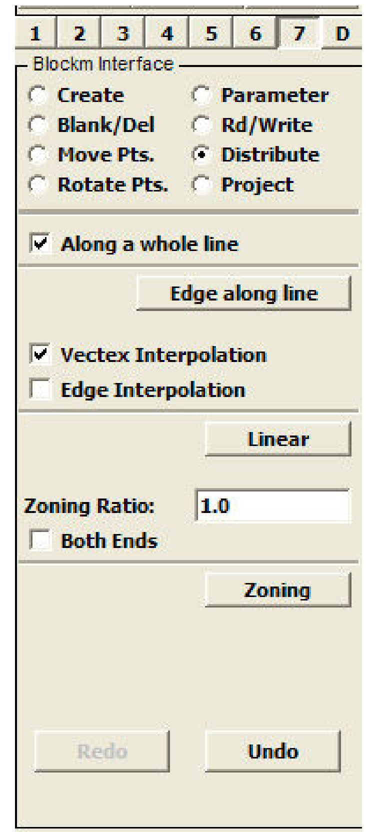


Before

BlockM: Start Using BlockM

- Distribute (continued)
  - Edge along line
    - Example (14.0) :
      - Note how the nodes are evenly distributed a long the entire curve.



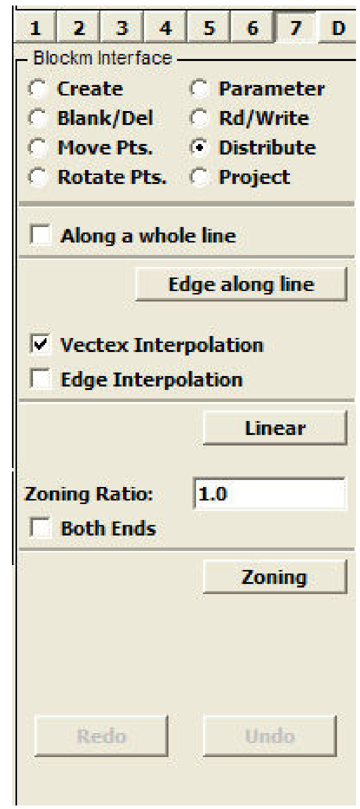


After

BlockM: Start Using BlockM

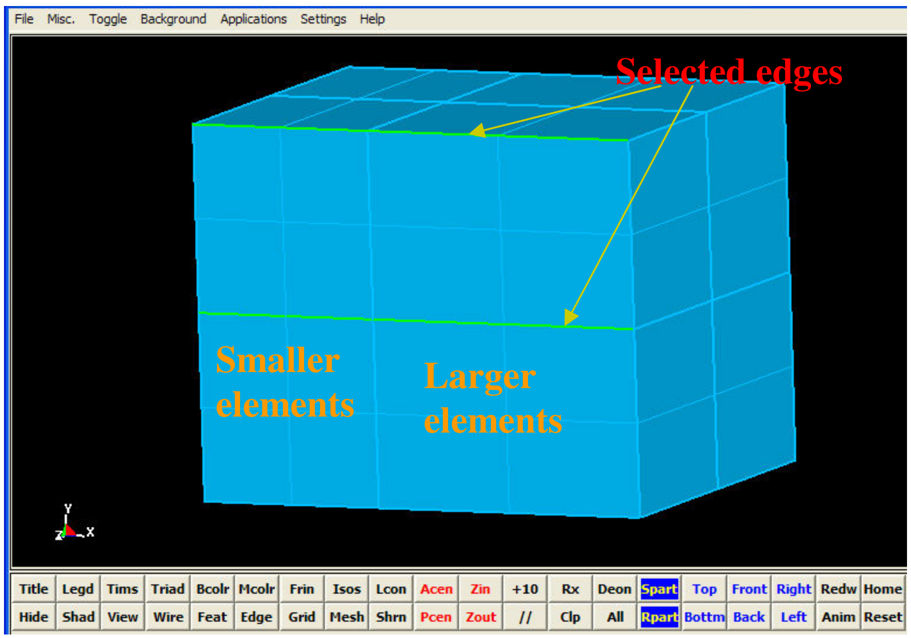
- Distribute (continued)
  - Linear
    - This feature distributes nodes evenly along a selected edge. This option is quite straight forward. Vectex Interpolation is selected by default. Edge Interpolation option could be selected as well.

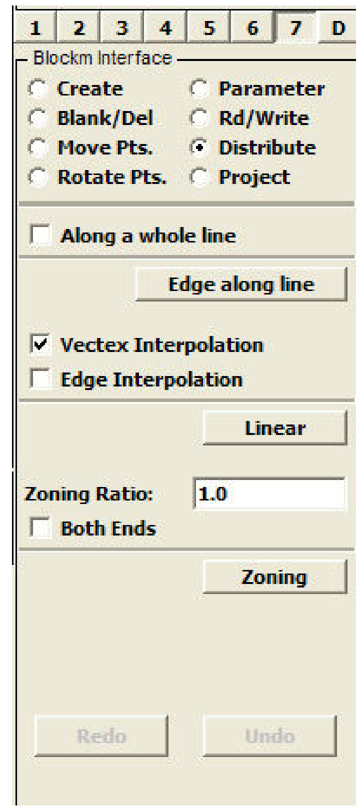
    1. Select the edge or edges to linear distribute.
    2. Click the "Linear" button to apply.



BlockM: Start Using BlockM

- Distribute (continued)
  - Linear
    - Example (14.1) : Distribute nodes on 2 edges.

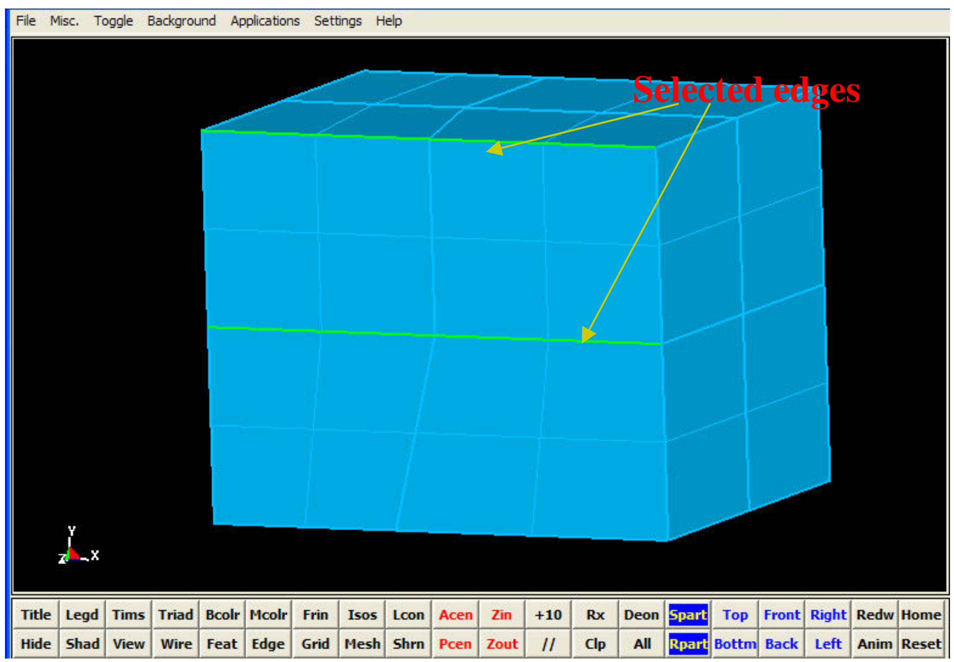


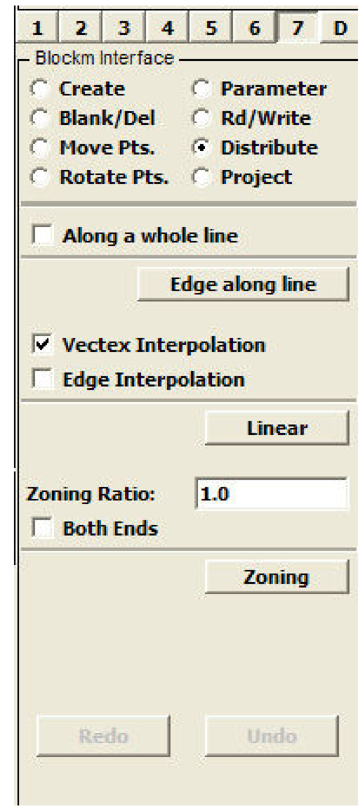


Before

BlockM: Start Using BlockM

- Distribute (continued)
  - Linear
    - Example (14.1) : Distribute nodes on 2 edges.



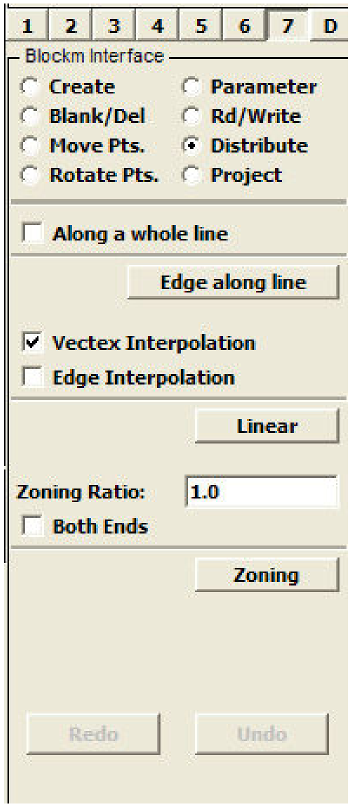


After

BlockM: Start Using BlockM

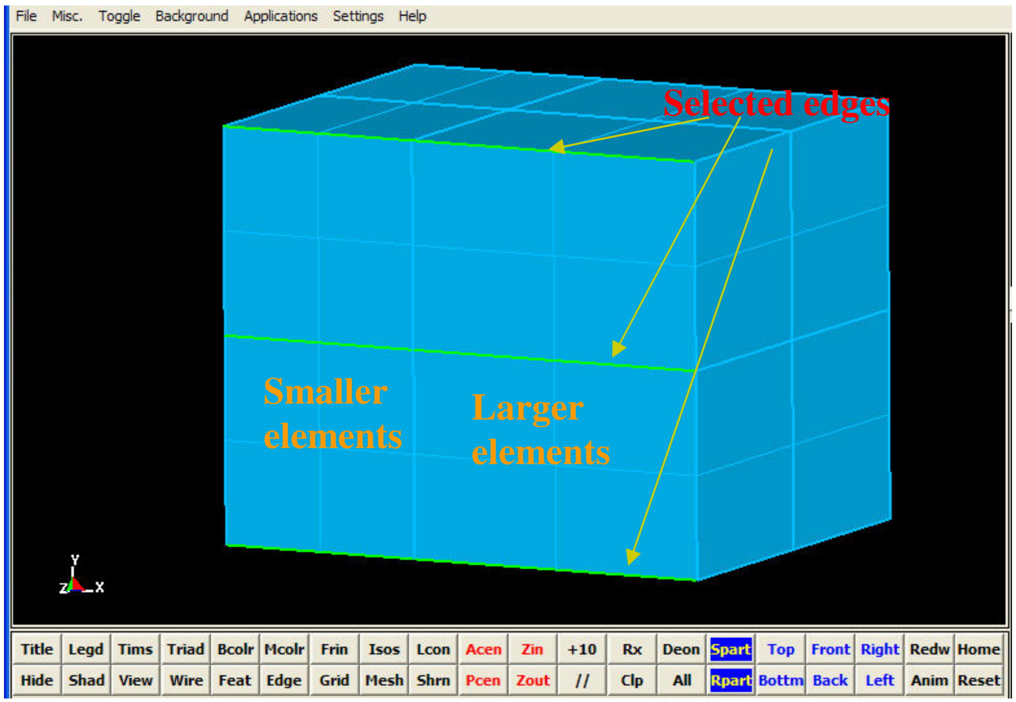
- Distribute (continued)
  - Zoning
    - This option distributes nodes along an edge such that the adjacent segment length divided by the current segment length equals the Zoning Ratio. By default, the Zoning Ration is equaled to 1.0.

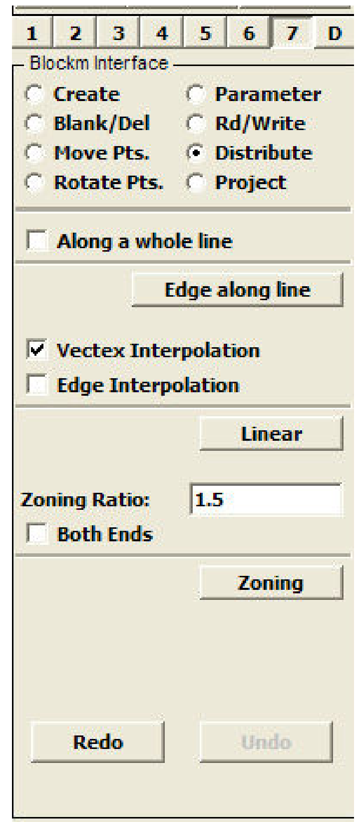
    1. Select the edge or edges to be distributed.
    2. Set the Zoning Ratio. This number cannot be negative.
    3. If both ends of the edge are to be the same than click the Both Ends radio button.
    4. To apply, click on the Zoning button.



BlockM: Start Using BlockM

- Distribute (continued)
  - Zoning
    - Example (14.2) : Distribute nodes on 3 edges with Zoning ration = 1.5.

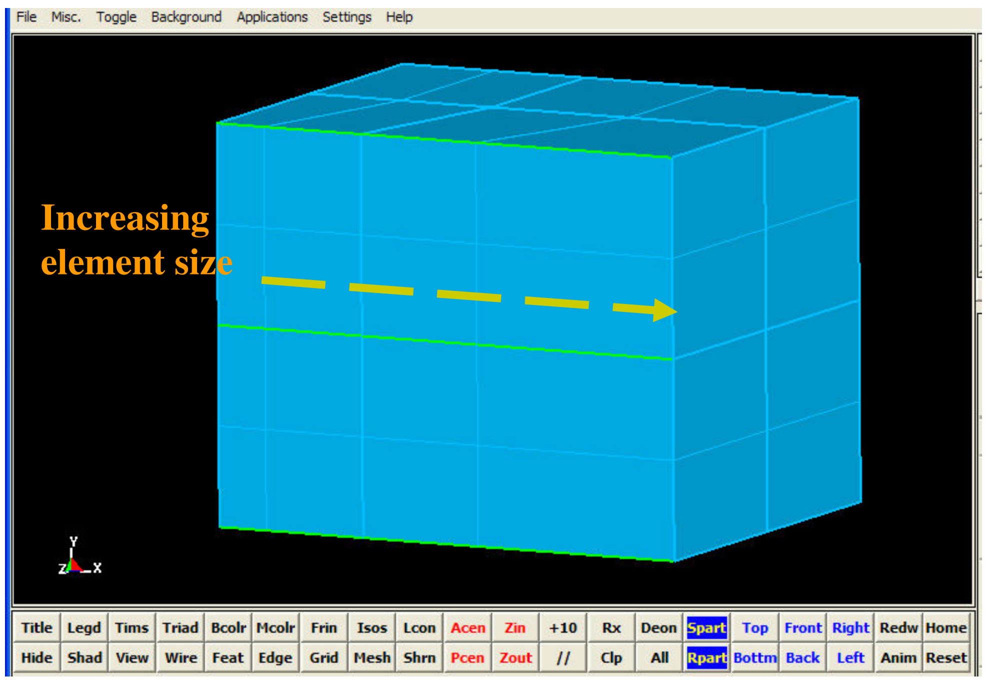


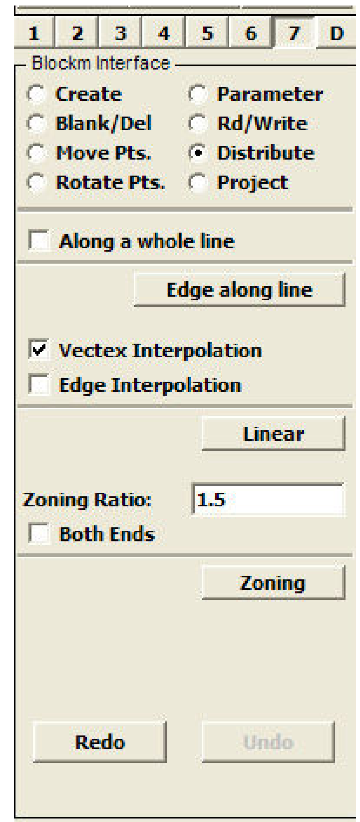


Before

BlockM: Start Using BlockM

- Distribute (continued)
  - Zoning
    - Example (14.2) :
      - Zoning ration = 1.5, note how the element sizes are changing.

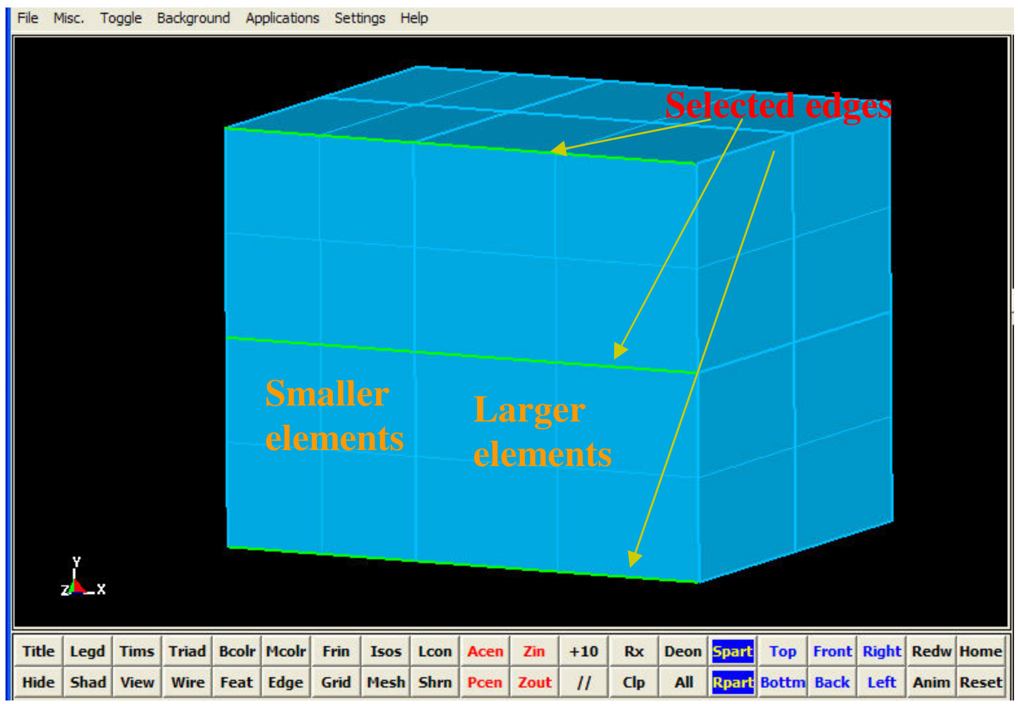




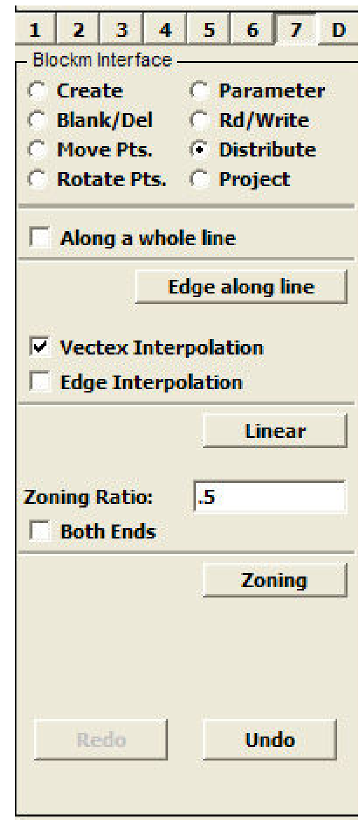
After

BlockM: Start Using BlockM

- Distribute (continued)
  - Zoning
    - Example (14.3) : Distribute nodes on 3 edges with Zoning ration = .5.

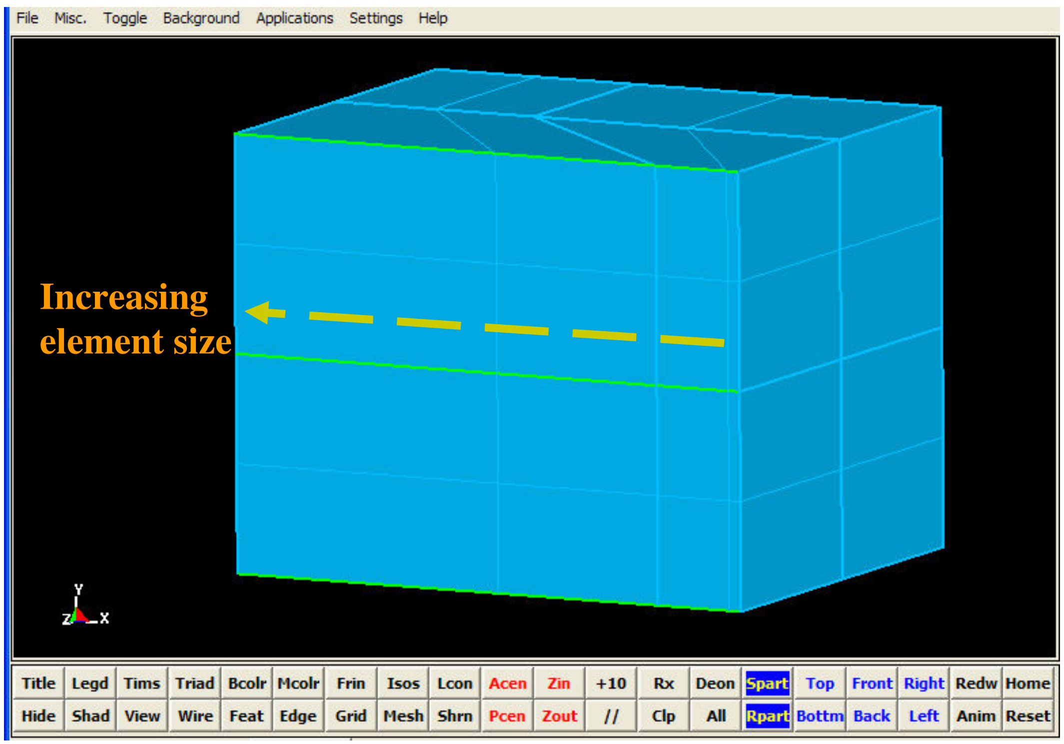


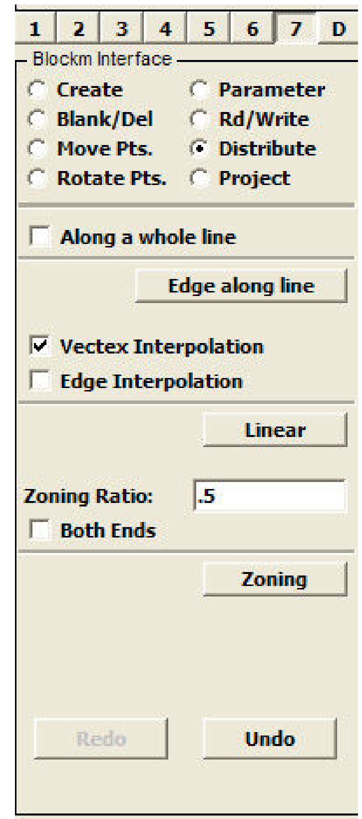
Before



BlockM: Start Using BlockM

- Distribute (continued)
  - Zoning
    - Example (14.3) :
      - Zoning ration = .5, note how the element sizes are reversed.

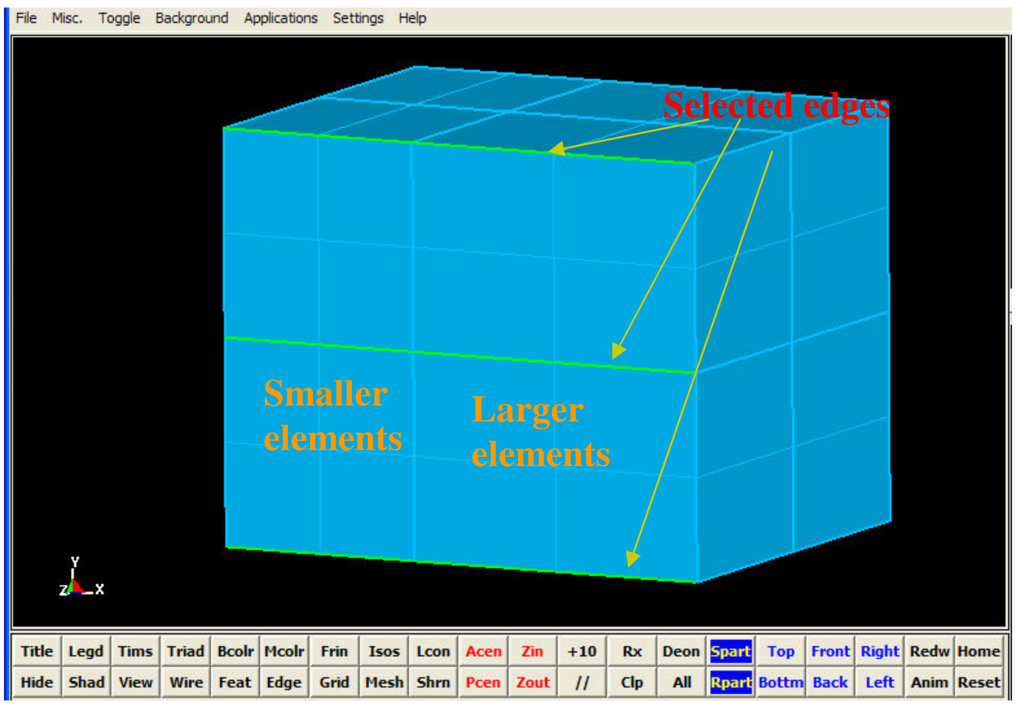




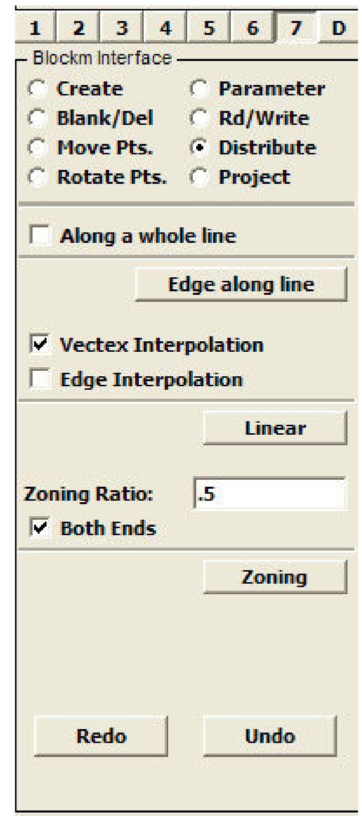
After

BlockM: Start Using BlockM

- Distribute (continued)
  - Zoning
    - Example (14.4) : Distribute nodes on 3 edges with Zoning ration = .5 and Both Ends.

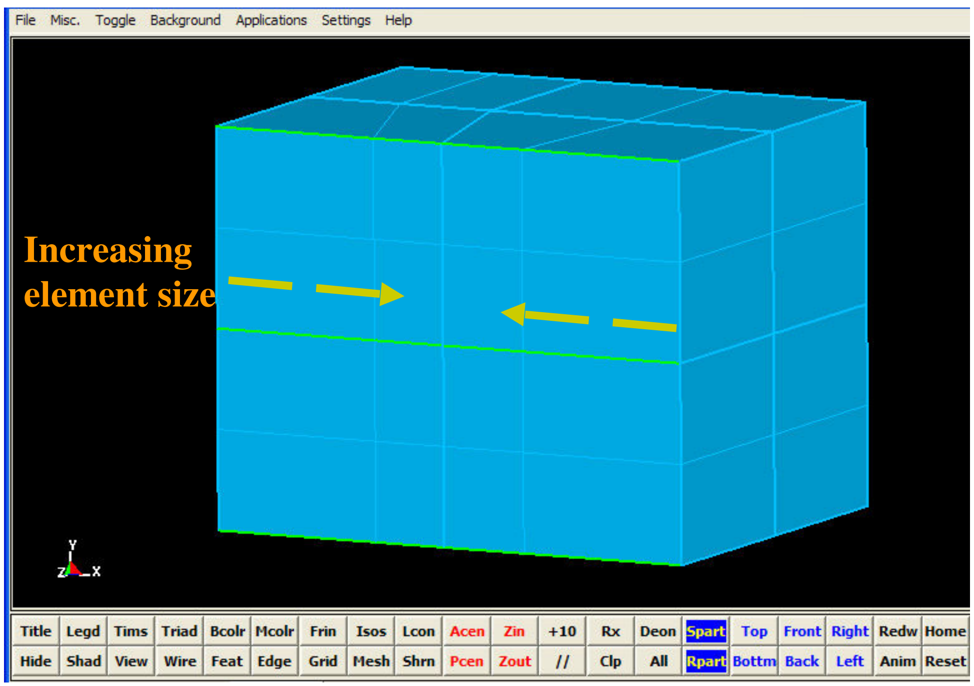


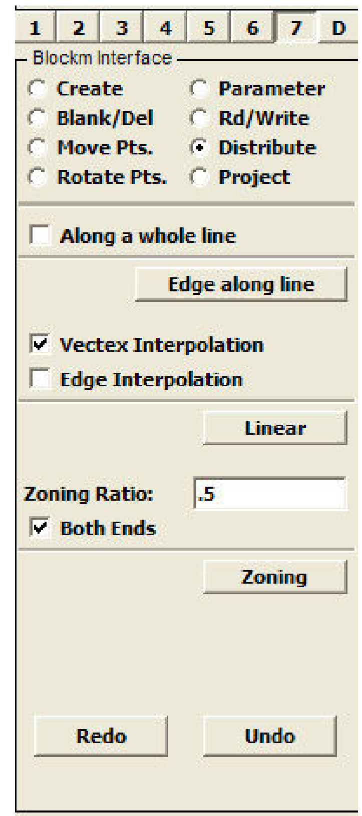
Before



BlockM: Start Using BlockM

- Distribute (continued)
  - Zoning
    - Example (14.4) :
      - Zoning ration = .5 and Both Ends, note how the smaller elements are concentrated at the center.

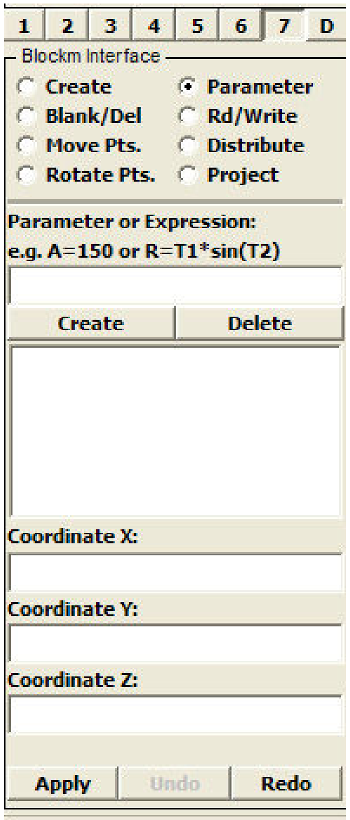




After

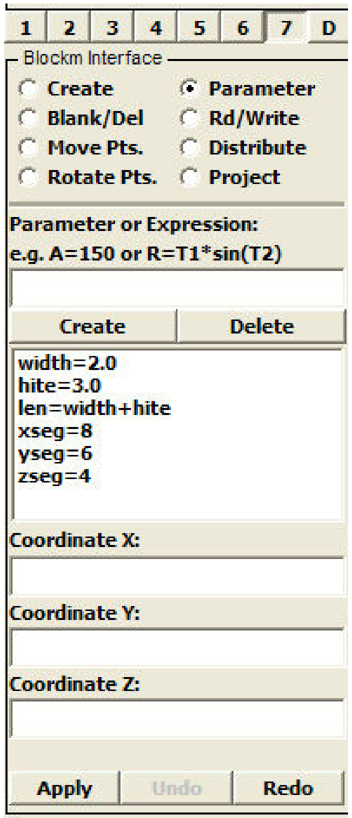
BlockM: Start Using BlockM

- Parameter
  - This feature allows for parameterized mesh generations. What this means is that a mesh could be changed and updated with the change of a parameter's value.
  - Functions and expressions could be added here and applied directly to a point, edge, face, and or region.
  - To implement Parameter:
    1. Enter parameters or expression in the 1st field
    2. Click Create button or press the Enter key. This puts the parameters or expressions in the list below.
    3. To Delete: select an expression or parameter from the list and click the Delete button.
    4. To apply to the block directly: enter the parameters in the corresponding X, Y, and Z Coordinate fields.
    5. Select a point, edge, face and region using the Computation window.
    6. Click the Apply button.



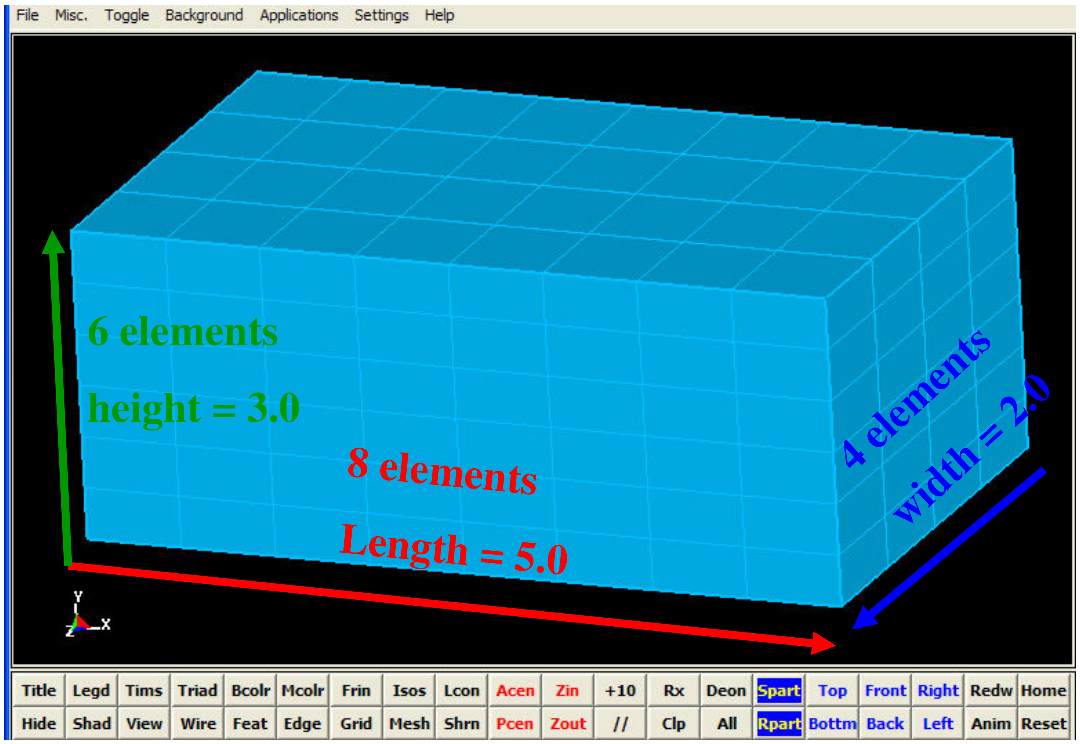
BlockM: Start Using BlockM

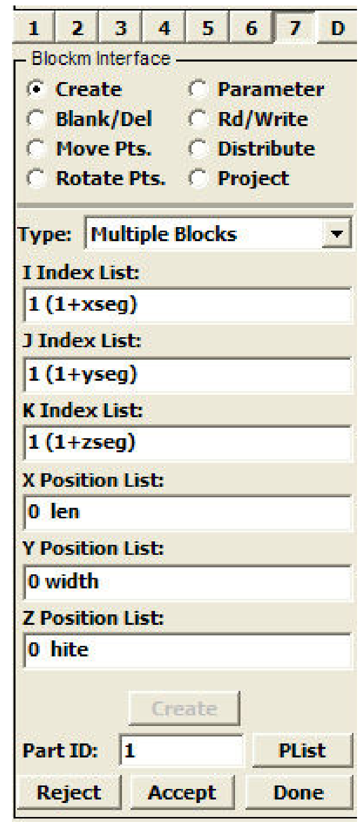
- Parameter (continued)
  - Example (15.0): Let’s create a parametric block.
    - Set the following parameters:
      - width= 2.0
      - hite = 3.0
      - len = width + hite
      - xseg = 8
      - yseg = 6
      - zseg = 4



BlockM: Start Using BlockM

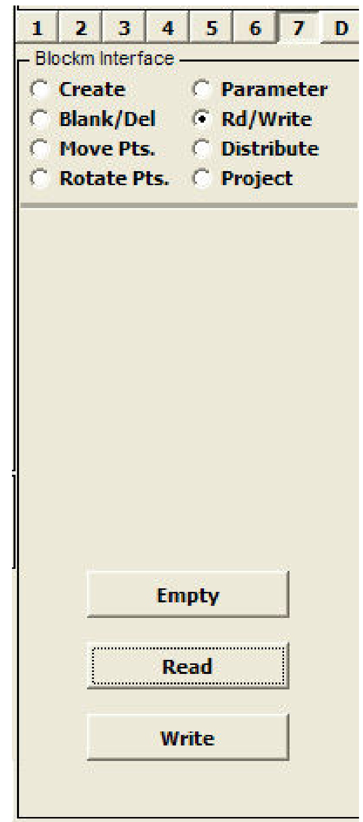
- Parameter (continued)
  - Example (15.0):
    - Below is what a parametric block looks like.





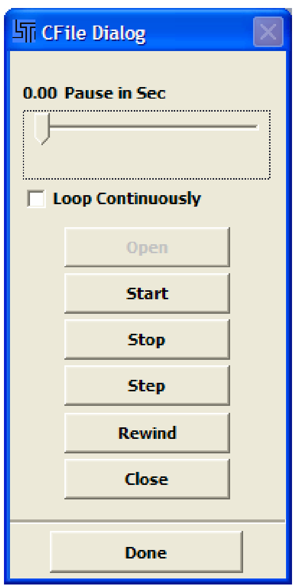
BlockM: Start Using BlockM

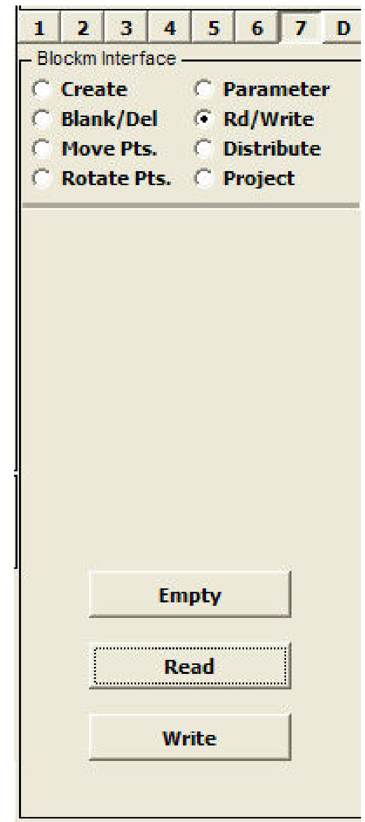
- Rd/Write
  - Within BlockM there is also a way to save the current work to a file. This is convenient to save just features and information related to BlockM. The file written out is in ASCII format. The user could modify the file with a text editor such as Notepad, WordPad under Windows and Vi under Linux/Unix.
  - To save or Write:
    1. Click the Write button.
    2. Enter the file name and path in the pop-up window. By default, the file as the .cfile extension.
    3. Click on the Save button.



BlockM: Start Using BlockM

- Rd/Write (continued)
  - To open an existing BlockM file:
    1. Simply click on the Read button.
    2. Select the file in the pop-up window.
    3. Click the Open button.
    4. A CFile Dialog window pop-up such as the one to the left. Click the Start button for the file to be processed.
    5. Click the Done button at the bottom.



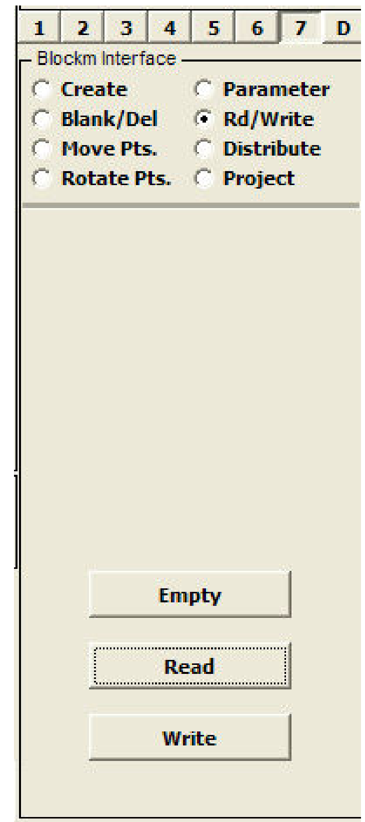


BlockM: Start Using BlockM

- Rd/Write (continued)
  - There is a special button here, Empty.
    - The purpose is to clear all the BlockM commands from the buffer.
    - Use this button with caution because all works could be loss if saves to the same file that is currently loaded.
    - To avoid possibly losing works, it is a good idea to save to a different file.

BlockM: Start Using BlockM

- Rd/Write (continued)
  - Example (16.0): Saved block file from example 15.0
    - As seen below, the BlockM input file is composed of a series of commands. As a user become familiar with BlockM's commands, he could "program" a part without using the BlockM's visual interface. This could be handy because it is a lot faster to generate and modify meshes.




```
bmparam create "width=2.0"
bmparam create "hite=3.0"
bmparam create "len=width+hite"
bmparam create "xseg=8"
bmparam create "yseg=6"
bmparam create "zseg=4"
bmcreate multiple i 1 (1+xseg) j 1 (1+yseg) k 1
(1+zseg) x 0  len y 0 width z 0  hite
```