Lesson 5:
Creating a Basic Assembly

Upon Successful completion of this lesson, you will be able to:

- Generatively insert component instances into assemblies.
- Create Orientation Assets to manage SolidWorks Mates.
- Use an Orientation Asset to write an Orientation rule.
- Use Assembly Assets to manage assemblies with pre-existing components.

Master Assembly vs. Generative Design

Traditional automation tools, such as using Design Tables in SolidWorks, rely on a Master Assembly. The Master Assembly contains every component you might ever need, and rules are used to suppress and/or delete components.

Genus Designer is Generative. It creates its own model of the assembly, and uses that model to generate a SolidWorks assembly. This means it is possible to start with an empty SolidWorks assembly, and insert and orient all the components by rule.

Class vs. Instance

An assembly is a collection of components. Each component is created from a template. The template is copied from the Solution directory into the Working directory, and assigned a unique name. The parameters are set, and it is inserted and mated into the assembly.

We refer to the template as a Class, and each component is an Instance of that class.

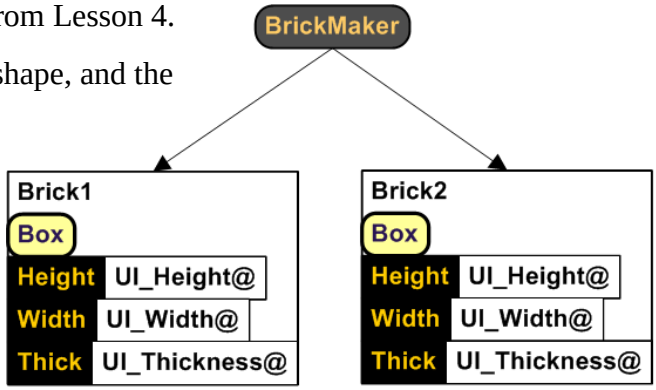
The Class definition for a Part captures the concept of what it means to be that part. The topology, the parameters that drive it, and the rules that calculate the parameter values are all part of the Class.

A component Instance is a specific usage of a Class. All the parameter values have been calculated and set.

To create an Instance of a Class, you must assign the Instance a unique name and indicate what Class is to be used as a template for the Instance. You then have the opportunity to override any attribute values that were defined in the Class. Finally, you must relate this Instance to an already existing Instance (or to the Assembly itself) in order to position it within the Assembly.

Adding Components

Adding additional component instances to an assembly is a simple process. Simply attach another Item shape to the Assembly class. Remember that each Item shape needs to have a unique name inside the Assembly.



1. Reopen the `BrickMaker` Project from Lesson 4.
2. Find the `BrickMaker` **Assembly** shape, and the `Brick` **Item**. Use **Ctrl+Drag-and-Drop** to place a copy of the original `Brick` **Item** next to the first.
3. Use the **Link** shape to add a link from the **Assembly** shape to the new **Item**.
4. Rename the original **Item** `Brick1`, and the new **Item** `Brick2`.
5. Perform a **Quick Test**, and ensure that both bricks are created without errors.
6. **Save** the Project and **Deploy** the Solution.
7. **Run** the Solution in SolidWorks.
8. After a moment, the new assembly will appear. From the graphics window, it may appear as though there is only one block, but notice in the Feature Manager tree that there are in fact two blocks, and they are both positioned at the assembly origin and fixed.

Positioning Components

Once the components exist, it is necessary position them within the assembly. To do this, we must define what SolidWorks mates are to be used, and which components are to be mated to each other.

Default Position

The default position of any component is to have its Origin aligned with its parent's origin. Also, components placed without additional positioning information are automatically set to *fixed* in the SolidWorks assembly.

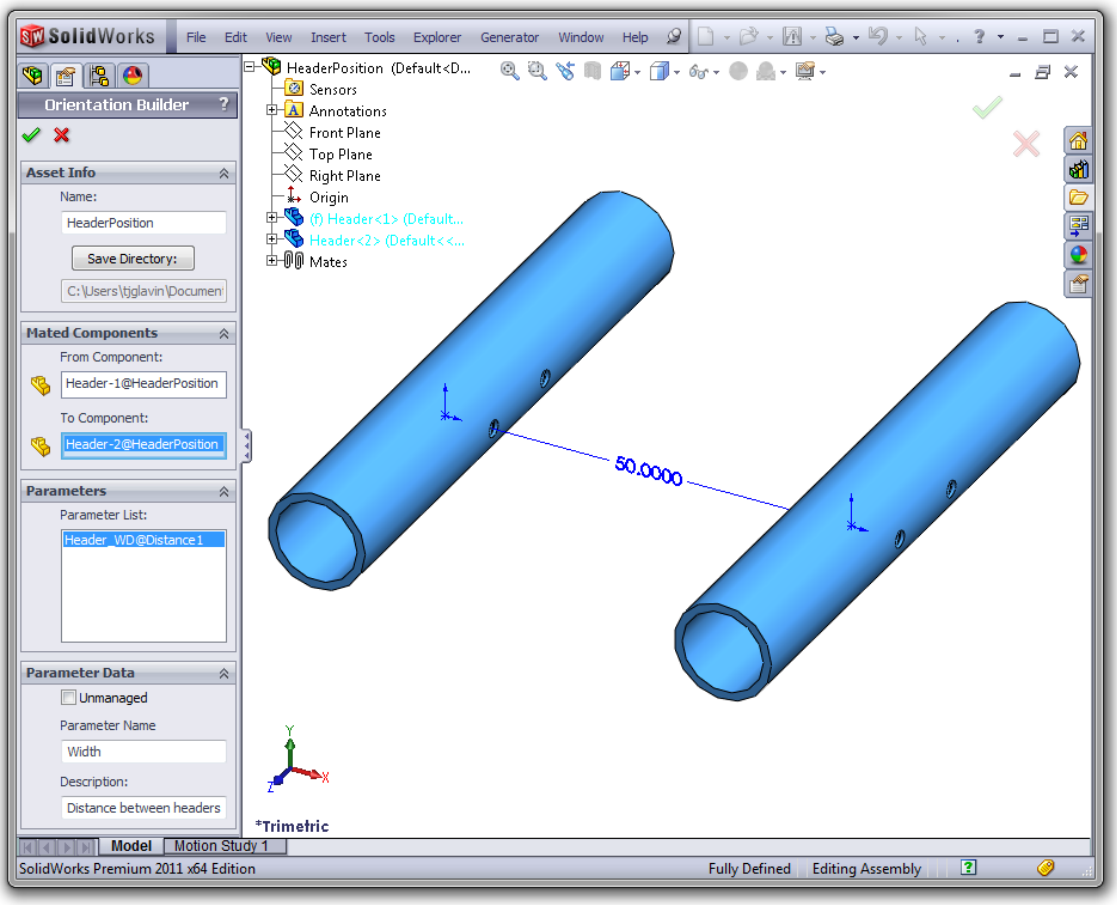
Orientation Assets

SolidWorks mates are brought into Genus Designer by means of Orientation Assets. An Orientation Asset is created from an assembly, which contains an example of how the component is to be mated.

Because of the way that SolidWorks handles entity identification internally, mates between model geometry such as faces and edges cannot be reliably reproduced. As parameter values change from component instance to component instance, geometric entities can change names or even be deleted entirely from the model.

So that mates are reproducible from instance to instance, we add Reference Geometry to the SolidWorks component, and use that Reference Geometry to create the mates.

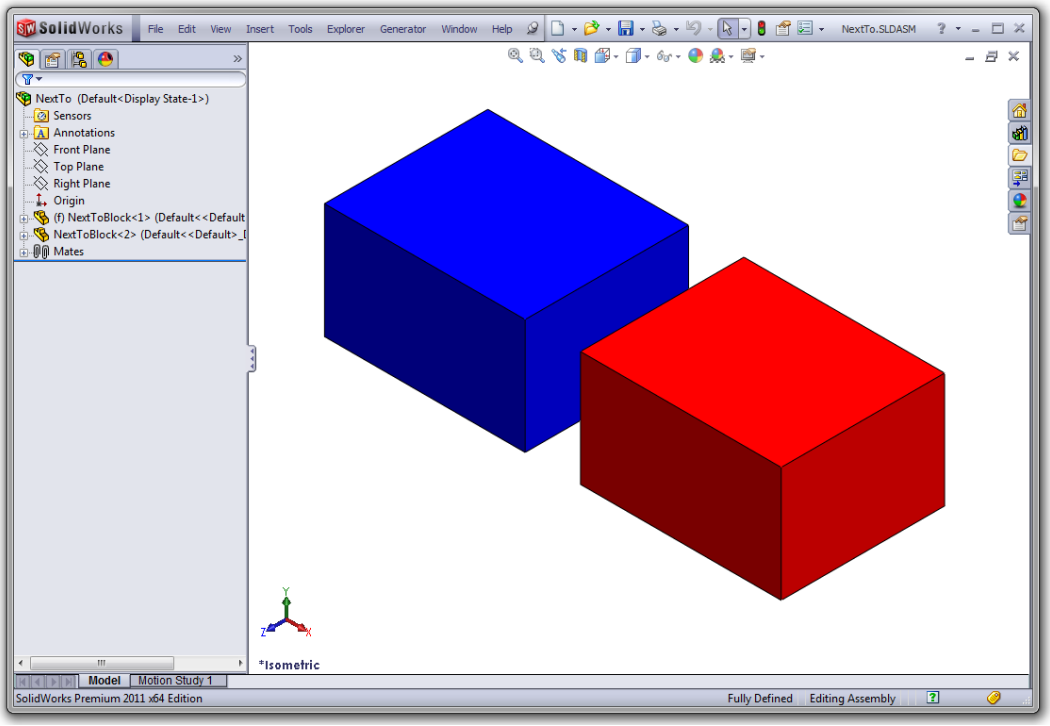
The process for building an Orientation Asset is very similar to the one used for building Part Assets.



With the assembly open, choose Create Asset, then Capture Orientation from the Generator menu in SolidWorks. The default name of the orientation asset is the assembly name. Select the component that will remain fixed as the "From Component", and the component to be oriented as the "To Component". Mates which require a parameter value, such as a distance or angle mate, will have their parameters displayed in the Parameter List. As with Part Assets, all

parameters must either be marked Unmanaged, or have a Name and Description assigned to them so that they can be managed by Genus Designer.

9. Open the `Right Of` assembly file from the ../`Build/` directory in the `Training\Sample Mates\` folder.
10. Review the mates between the two blocks. The blue block is fixed, and two Coincident mates and a Distance mate are used to position the red block.



11. Select **Create Asset**, then **Capture Orientation**, from the **Generator** menu in SolidWorks.
12. The default name for the **Asset** is: `Right Of`.
13. Select the blue block as the **From Component**.
14. Select the red block as the **To Component**.
15. Select the one available **Parameter**, which is the dimension from the Distance mate.
16. Name the **Parameter** `DistanceToRight`.
17. Enter `Distance between Right planes` as the description.
18. Click the green check to finish creating the asset.
19. **Close** the SolidWorks Assembly.

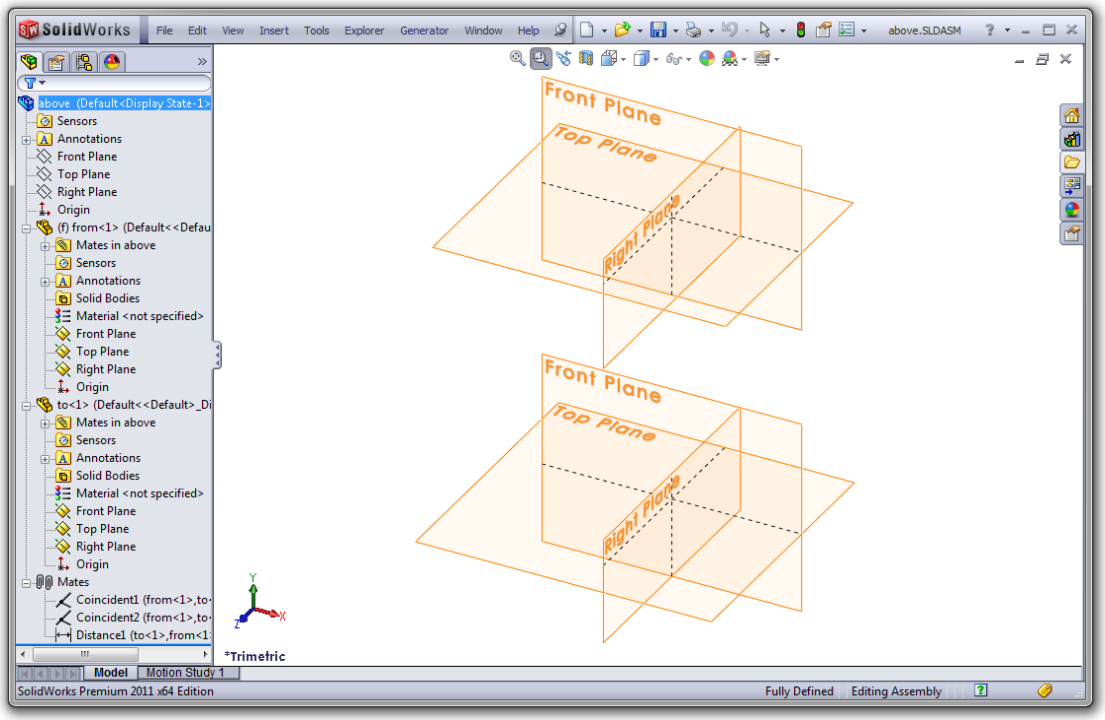
One of the advantages of using Reference geometry for mates is that it is possible to create general orientation assets. Any components containing the necessary Reference geometry can use the Asset. This also means that it is not necessary to use the actual parts being mated in your Project to create the Orientation asset.

Simply by creating empty part files and mating their default planes, or by adding specifically named reference geometry to a part file with no other geometry, it is possible to create Orientation Assets that can be used for any components that contain the same reference geometry.

When creating Reference planes in different parts, care must be taken not only to name the Planes the same name, but to make sure the same surface is facing outward. When the visibility of planes is turned on in SolidWorks, the Front and the Back of the plane are different colors. Depending upon the construction technique, two planes can appear to be in the same place, but are really "flipped" by 180 degrees.

---

20. Open the assembly file named: `Above`, from the `\Training\Sample Mates\` folder in the `..\Build\` directory.
21. Notice the parts are created without any model geometry.
22. Rotate the graphics, and make sure you understand the difference between the front and the back of a Plane.
23. Review the component mates. The `From` part is fixed, and the `To` part is positioned above the first using two Coincident mates and a Distance mate.
24. Select **Create Asset**, then **Capture Orientation**, from the **Generator** menu in SolidWorks.
25. The default name for the **Asset** is: `Above`.
26. Select the 'from' component in the feature manager tree as the **From Component**.
27. Select the 'to' component in the feature manager tree as the **To Component**.
28. Select the one available **Parameter**, which is the dimension from the Distance mate.
29. Name the **Parameter** `DistanceAbove`.
30. Enter `Distance between Top planes` as the description.
31. Click the green check to finish creating the asset.
32. **Close** the SolidWorks Assembly.

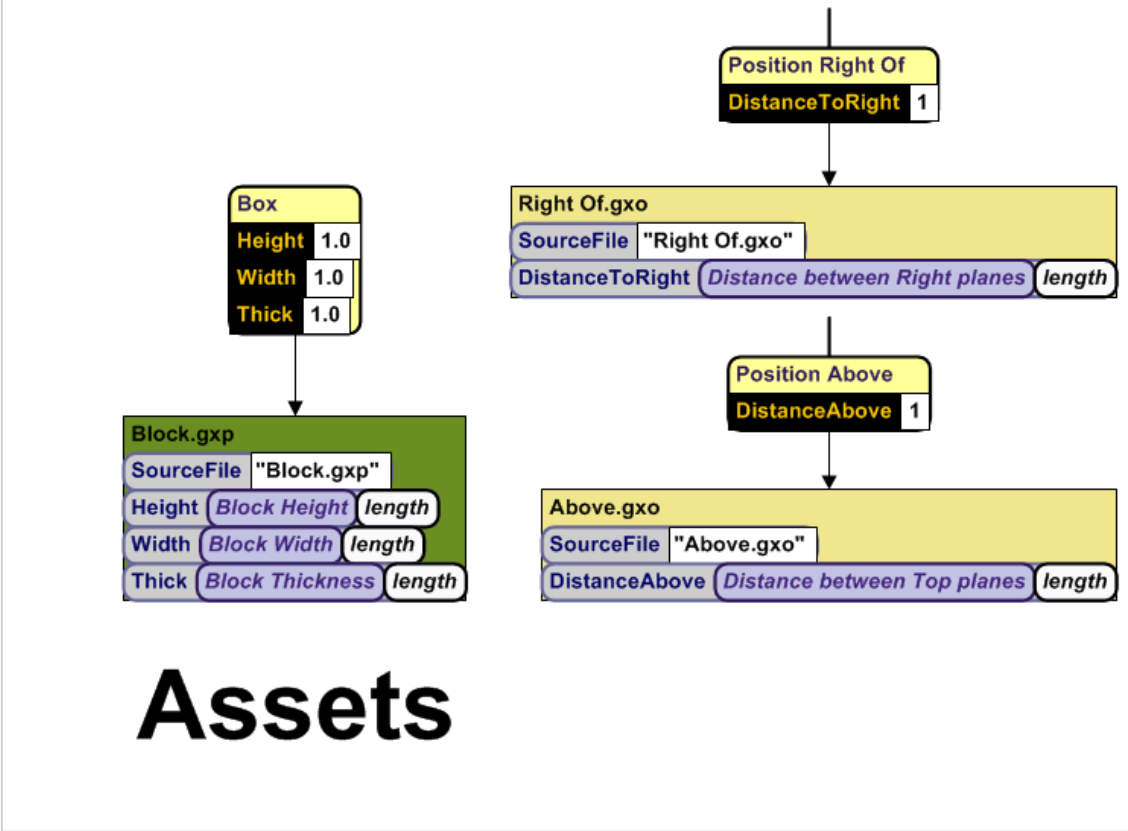


Like all assets, once they are created, we will need to import them into our Project before we can make use of them.

33. Return to the `BrickMaker` Project, and select **File**, **Import Asset…** from the menu. Import the `Right Of` and `Above` assets by selecting both at once.
34. Before dragging the assets into the project, add a new Page to the project, name it `Assets`, and copy the title **Note** onto the page.
35. Use box select to select the `Block` **Part Asset** and the `Box` **Part** it is attached to, then use **Drag-and-drop** to move the shapes to the new `Assets` page. While dragging, hover the mouse over the tab to "turn" the page.
36. From the **Assets** flyout, **Drag-and-drop** the two **Orientation Assets** onto the `Assets` page. If the **Orientation Assets** are not visible in the list, refresh the display.
37. Just as with a **Part Asset**, an **Orientation Asset** must be attached to a class before it can be used. For each asset, drop a **Position** shape from the assembly stencil onto the page, and link it to the asset. Name the one

attached to the `Above` **Orientation Asset**: `Position Above`, and name the one attached to the `Right Of` **Orientation Asset**: `Position Right Of`.

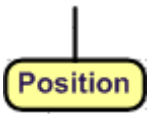
38. Again, just as with parts, add an attribute to the **Position** shape for each asset parameter, and use it to provide a default value for the parameter.



Orientation Shapes

Writing an Orientation rule in Genus Designer requires the use of two new shapes.

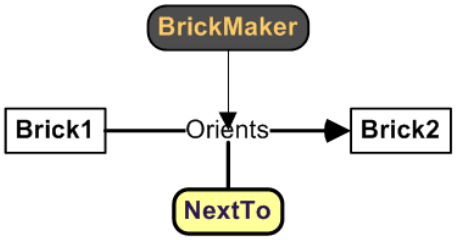
The first is a special link called an Orients Arrow. Like a simple Link, this is directional relationship. It is placed between two Item Shapes, with the From component, as defined by the Orientation Asset, at the tail of the arrow, and the To Component at the head of the arrow.



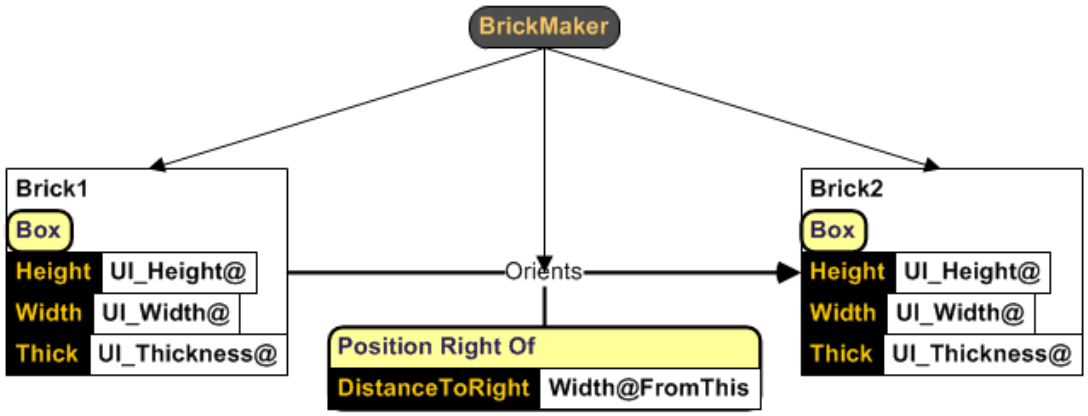
The second is a Position shape, identical to the one used to define the Position class. The Position Shape must be attached to the Orients Arrow. The entire group of shapes can be read as: This <To Component> is positioned with respect

to <From Component> according to the rules in <Position class>.

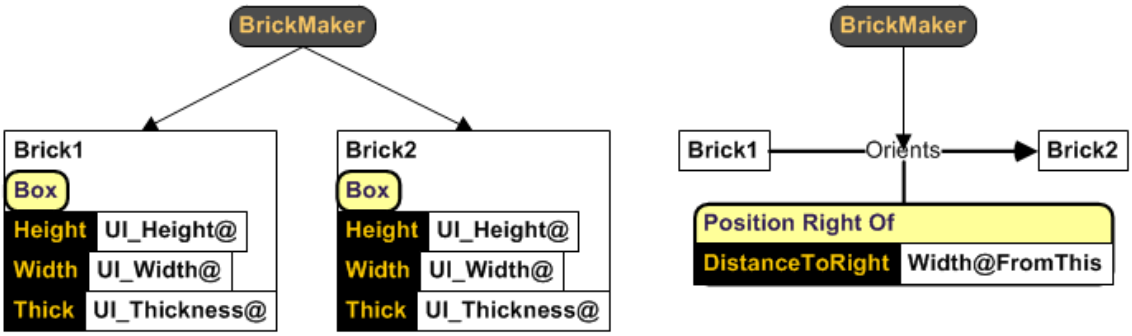
These shapes together are collectively referred to as an Orientation Rule.
An Orientation Rule will search out and attempt to position every Item in the entire project with the given Item name. Since it is not necessary to have globally unique Item names throughout the project, only within individual assemblies, it is necessary to limit the scope of an Orientation. To do this, drop an Assembly shape above the Orients Arrow, and add a Link from the Assembly shape to the Orients Arrow. The Assembly shape must have the same name as the assembly in which this Orientation is to be created.



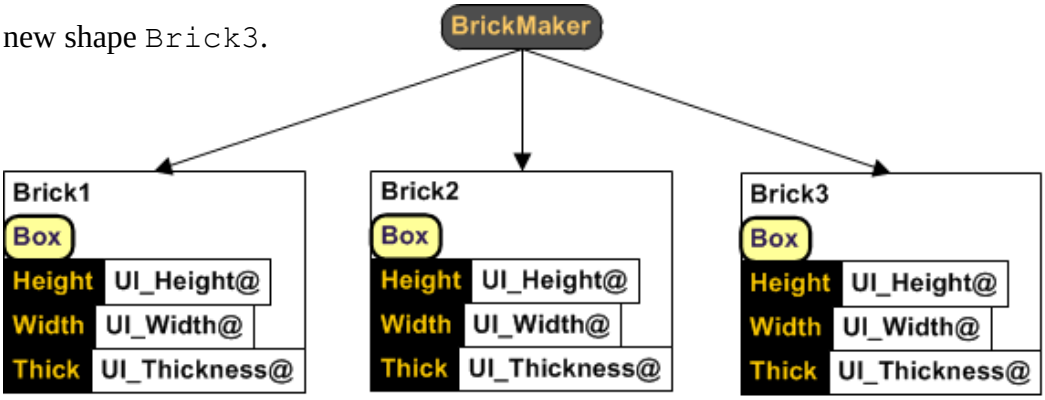
Identically Named Shapes



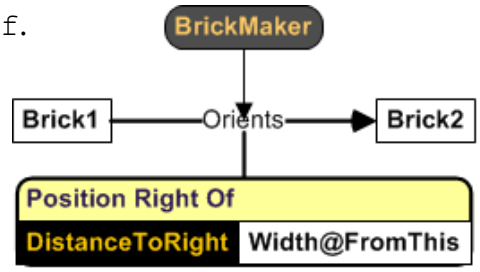
The Knowledge Editor supports partial definitions. This means that any two shapes of the same type, with the same name, are treated as if they were a single shape, even if they are on different pages. We can take advantage of this to keep our project organized. For example, although it is permitted, it is not required that all the Items in an assembly and all the Orientation rules under the scope of the assembly be attached to the same Assembly shape.



39. Go back to the `Main Assembly` page.
40. Make another copy of the `Brick1` **Item**, and link it to the `BrickMaker`. Name the new shape `Brick3`.

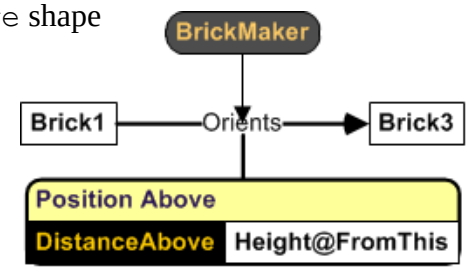


41. Create four **Item** shapes on the `Assets` page. Name two of them `Brick1`, and one each of `Brick2` and `Brick3`.
42. Create an Orients arrow between one Brick1 Item and the Brick2 Item.
43. Create an **Orients** arrow between the `Brick1` **Item**, and the `Brick3` **Item**.
44. Add a **Position** shape to the sheet, and attach it to the first **Orients** arrow.
45. Name the **Position** shape `Position Right Of`.
46. To override the default distance, add an **Attribute/Formula** to the **Position** shape, name it `DistanceToRight`, and set its value to `Width@FromThis`.



Note: "FromThis" is another special, reserved name that refers specifically to the From component in an Orientation rule. In this case, we want to access the width of the fixed block in order to determine how far away the second block should be.

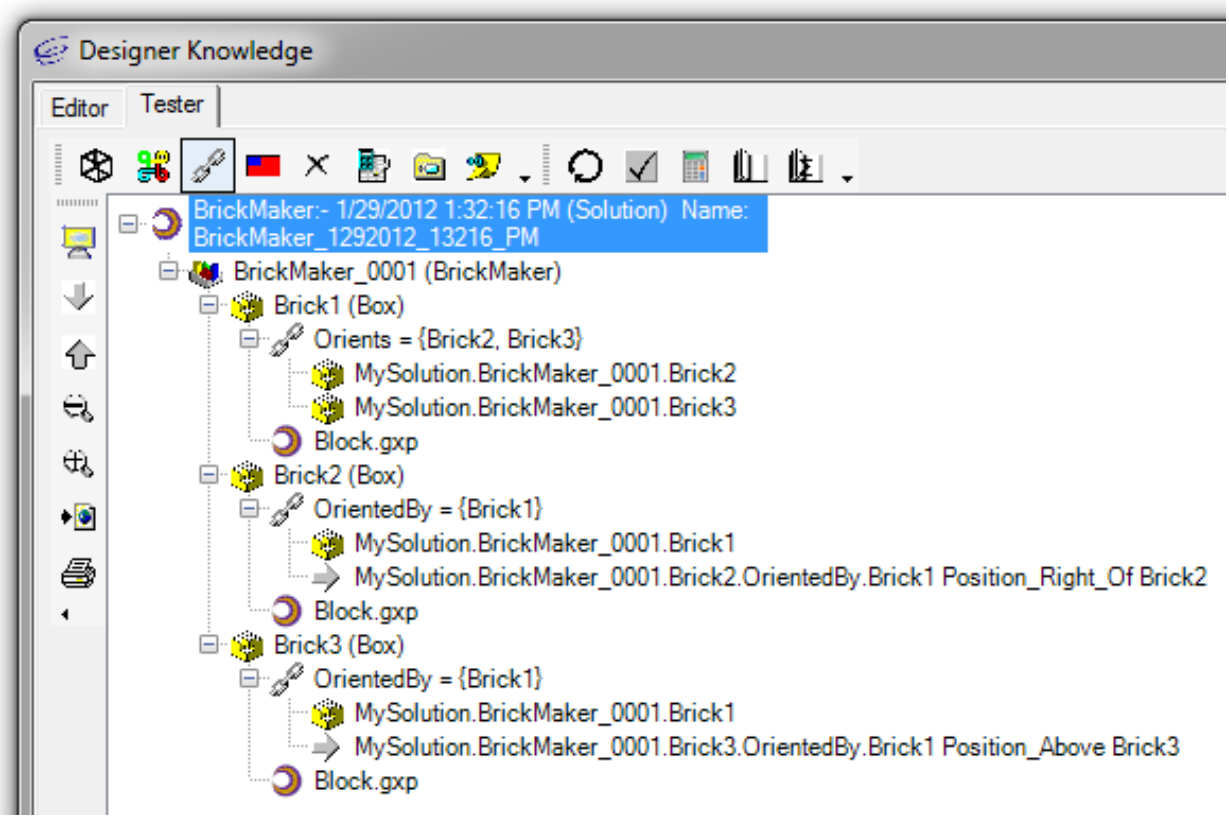
47. Repeat the procedure to create a `Position Above` shape that references `Height@FromThis` to orient `Brick3` with respect to `Brick1`.
48. Define the scope for each Orientation rule by adding an **Assembly** shape named `BrickMaker` above each **Orients** arrow, and link it to the **Orients** arrow.



Debugging Orientations Using QuickTest

When running QuickTest, Orientation rules are hidden by default. Use the toolbar icon Show Hide Links to make Orientation rules visible. Be sure to refresh the display after changing this setting.

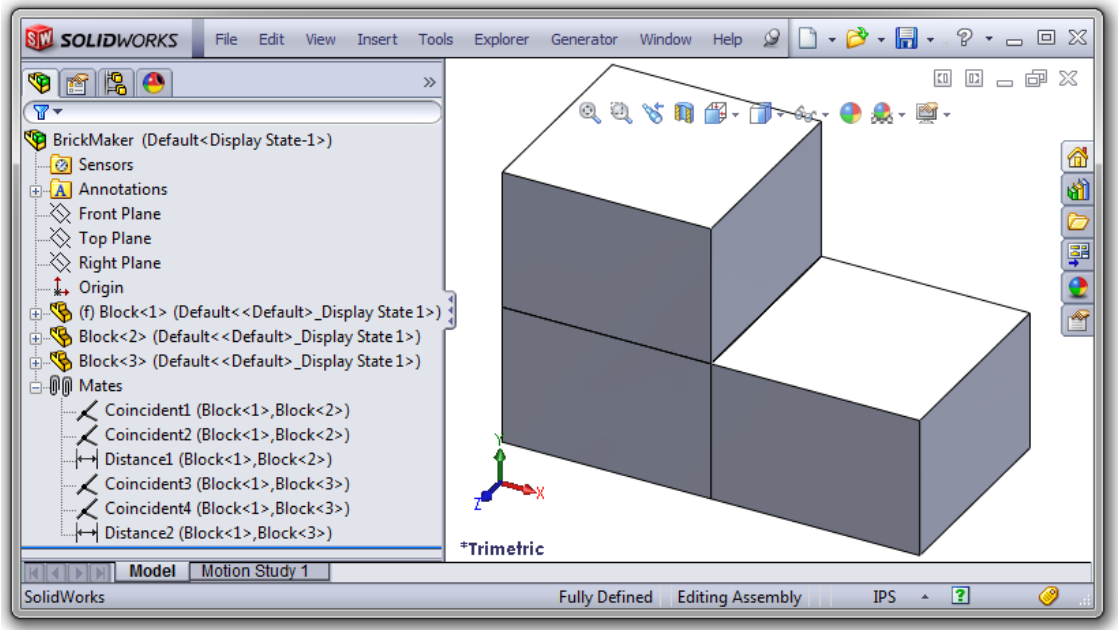
A properly implemented Orientation rule appears as follows in the QuickTest:



Every From Component has a line that says Orients, and contains a list of all the components that it Orients. Every To Component has a line that says OrientedBy. In this list is the From component, and the Position instance itself. Inspecting the Position instance will show any attributes used by the Orientation rule, and the Orientation asset.

If any of these things are missing, the right mates will not be inserted in SolidWorks. Go back and verify that your links are attached correctly, and try a Check Semantics / Learn.

49. Run a **Quick Test** of the Project, **Show Links**, and make sure all values evaluate correctly.
50. **Save and Deploy the Project**, then Open it in the SolidWorks environment to see the resulting parts and review the mates that were created.



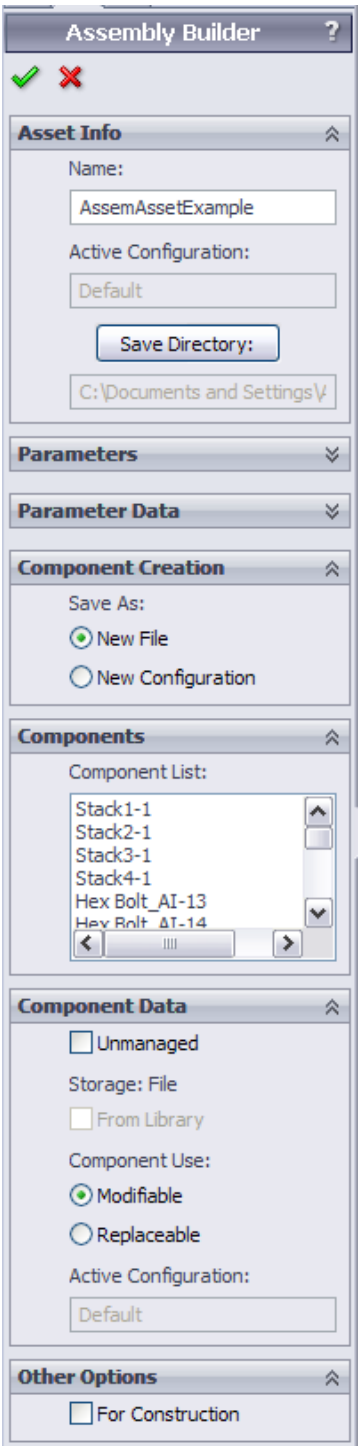
Assemblies with Pre-existing Components

Up to this point we have been constructing assemblies 100% generatively. We start with an empty assembly, and insert components by rule.

In some cases the components we want to include in the assembly do not need to be generated, because they always exist. All we really want to do is change their dimensions. Assemblies with pre-existing components, and their mates, are managed by an Assembly asset.

Working with a hybrid model consisting of Assembly assets and generatively added parts provides all the runtime performance benefits of a Master Assembly, while managing the complex variations within Designer instead of SolidWorks.

Creating an Assembly Asset



Creating an Assembly Asset is very similar to creating Part or Orientation Assets. With the assembly file we want to capture open we will select Create Asset, then Capture Assembly from the Generator menu. This will open the Assembly Builder in the Property manager.

We will begin by setting the name and save location of the Asset. In the Parameter List we will see all of the parameters that are available to be managed by Genus Designer. Primarily these will be dimensions associated with Mates or Assembly Features. We can use the Parameter Data area to enter Names and Descriptions for each Parameter to be managed.

There are two new sections for Components and Component Data. Each of the components present in the assembly will be listed in the Components area, and we can determine how Genus Designer will utilize each component when using this asset, by selecting Unmanaged, Modifiable, or Replaceable in the Component Data area.

Unmanaged: When Unmanaged is selected, Genus Designer is going to use this exact component instance when building this assembly without changing any aspect of the component.

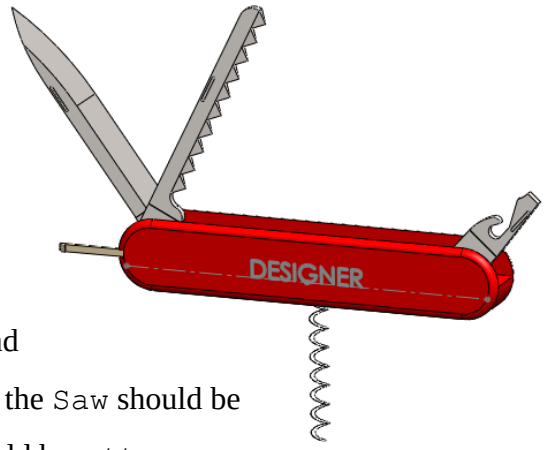
Modifiable: When Modifiable is selected, Genus Designer will be responsible for managing the parameters in this component when we are using the Assembly Asset. The component can be modified as needed (change dimensions, quantities for patterns, etc.) in the same way as we do for individual parts.

Replaceable: When Replaceable is selected, Genus Designer enables us to not only modify the component, but to substitute a different component altogether. For instance, when designing an airplane, setting a propeller to replaceable allows me change dimensions on the propeller, or swap out the propeller for a jet engine (provided the jet engine has the appropriate mate references).

In the following example, we are going to make use of these different selections to drive the design of a Swiss Army Knife. The toothpick, knife, and corkscrew are going to remain present and constant. The saw's teeth are going to be changed while the can opener is going to be replaced with a flash drive.

---

51. Open the `SwissArmyKnife` assembly from the `..\Build\` directory in the `Training\Swiss Army Knife\` folder. From the **Generator** menu select **Create Asset**, then **Capture Assembly**.
52. Leave the default name and directory for the **Asset**.
53. The **Parameter** list should be blank, since there or no parameters available on the assembly to control.
54. The `Casing`, `BigKnife`, `Toothpick`, and `Corkscrew` should be set to **Unmanaged**, the `Saw` should be set to **Modifiable**, and the `CanOpener` should be set to **Replaceable**.

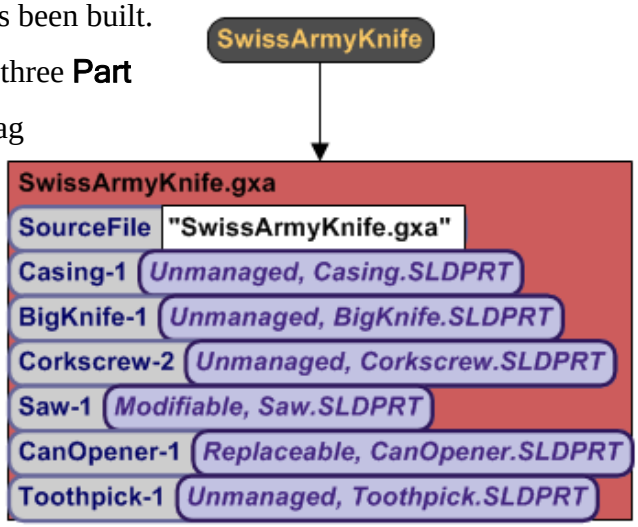


55. Select the Green Check to close the Assembly Builder and create the Asset.
56. Open the `Saw` part from the same directory. From the **Generator** menu select **Create Asset**, then **Capture Part**. Leave the default name for the **Asset**. Find the parameter `ToothWidth@Sketch7` and set the name to `ToothWidth`, and the description to `Width of a single tooth`. Find the parameter `ToothCount@LPattern1` and set the name to `ToothCount`, and description to `Number of Teeth`.
57. Open the `FlashDrive` part. From the **Generator** menu select **Create Asset**, then **Capture Part**. Accept the default name for the **Asset**. Leave all parameters as **Unmanaged**.
58. Open the `CanOpener` part. From the **Generator** menu, select **Create Asset**, then **Capture Part**. Accept the default name for the **Asset**. Leave all parameters as **Unmanaged**.

At this point, we have all the components needed to redesign our pocket knife. The Context for this project has already been created, leaving only the product model to be laid out in the Designer environment.

---

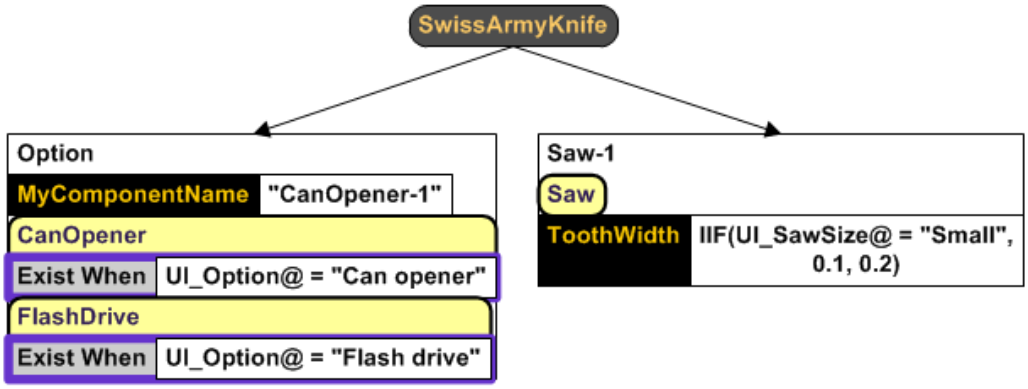
59. Open the Knowledge Editor, and open the existing Project named `Swiss Army Knife`. Review the **Context** that has been built.
60. Import the **Assembly Asset** and all three **Part Assets** that were just created and drag them onto the workspace. Examine the **Assembly Asset**. Note that the names of all the components are listed, as well as what must be managed.



61. Drag an **Assembly** shape onto the page and name it `SwissArmyKnife`. Link the shape to the **Assembly Asset**.
62. Drag three **Part** shapes onto the page and name them `Saw` and `FlashDrive` and `CanOpener`, then link them to their **Part Assets**.
63. In the `Saw` **Part**, add an attribute named `SawLength`, and assign it a value of `1.5` inches. Provide a default value for `ToothWidth` of `0.15` inches. Calculate the `ToothCount` from the formula: `Floor(SawLength/ToothWidth)`.
64. Create a copy of the `SwissArmyKnife` **Assembly** shape. **Drag-and-drop** two **Item** shapes onto the **Assembly**, creating links.
65. Name the **Item** shapes `Option` and `Saw-1`.

Note: To modify or replace a component in an Assembly Asset, the Item shape name must match the component name exactly, or the Item must contain an attribute named MyComponentName, whose value is the name of the Component.

66. Copy the **Part** shapes and drag them into their corresponding **Item** shapes. Put the `CanOpener` and the `FlashDrive` into the `Option` **Item**, and the `Saw` into the `Saw-1` **Item**.
67. Drag an **Attribute/Formula** shape into the `Saw-1` **Item** and name it `ToothWidth` to override the default value in the **Part Asset**. In the formula, enter `IIF(UI_SawSize@ = "Small", 0.1, 0.2).`



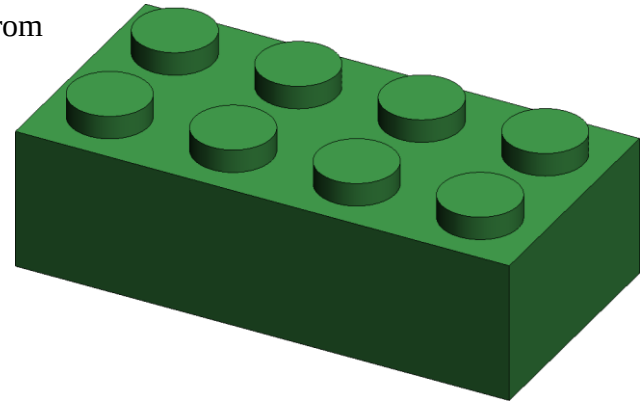
68. Drag another **Attribute/Formula** shape into the `Option` **Item**, and name it `MyComponentName`. Set the formula to `CanOpener-1`.

Note: A single Item cannot have two types at the same time. We must write a rule that states when to use the CanOpener and when to use the FlashDrive.

69. Drag an **ExistWhen** shape into the `CanOpener` **Part** inside the `Option` **Item**. Enter the following formula: `UI_Option@ = "Can Opener".`
70. Drag an **ExistWhen** shape into the `FlashDrive` **Part** inside the `Option` **Item**. Enter the following formula: `UI_Option@ = "Flash drive".`
71. **QuickTest** the Project.
72. Complete the **Deployment Specification**, and **Save and Deploy** the Project.
73. **Run** the solution and observe the results.

Lesson 5: Exercises

Exercise 5.1:



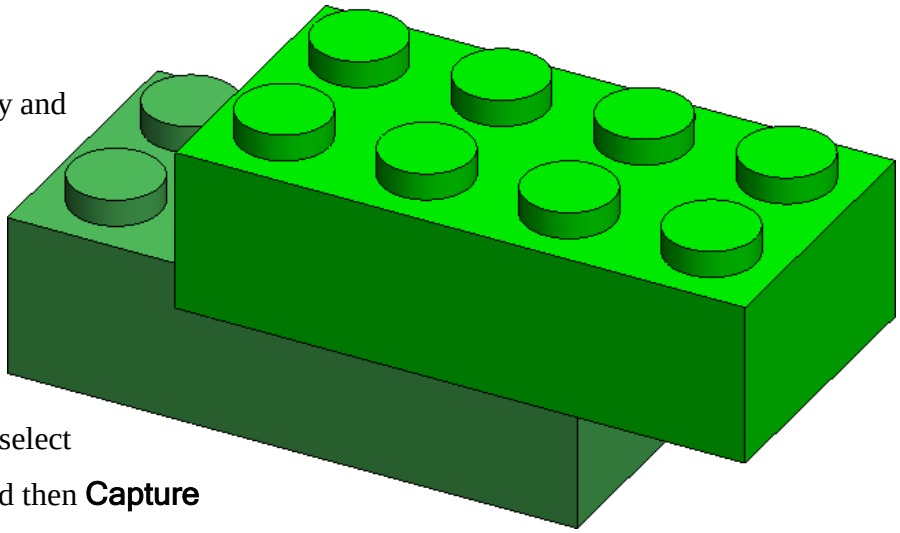
1. Open the `Building Block` Part, from the `Training\BuildingBlock\` folder in the `..\Build\` directory.

2. Create an Assembly from the Part using the `Assem_MM` template.

3. Place one `Building Block` Part at the assembly origin.

4. Add a second `Building Block` Part to the assembly.

5. Build Distance Mates between the two Parts' matching planes (Front to Front, etc.). Use `20mm` as the starting value for the distance mates.

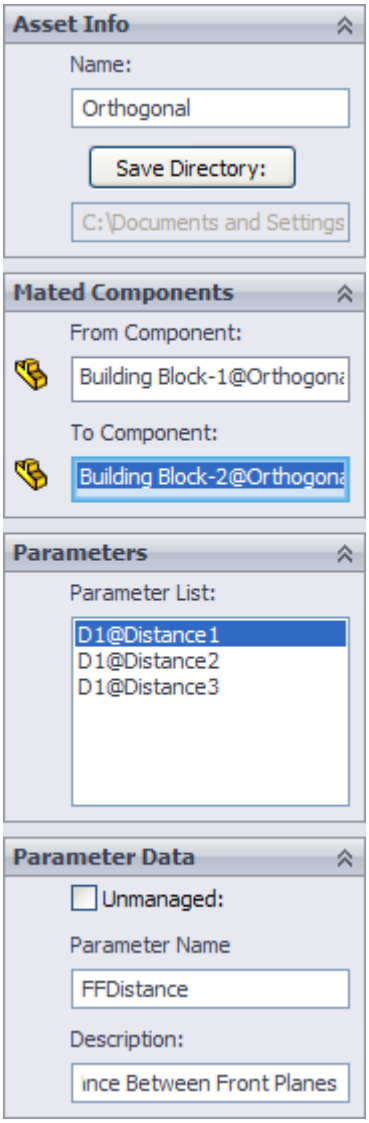


6. Save the Assembly and name it `Orthogonal`.

7. From the **Generator** menu select **Create Asset**, and then **Capture Orientation**.

8. Accept the defaults for the name and save directory.

9. Select the fixed part as the **From Component**, and the second part as the **To Component**.



10. Name the dimension parameter between front planes as `FFdistance`, and use the description `Distance Between Front Planes`.

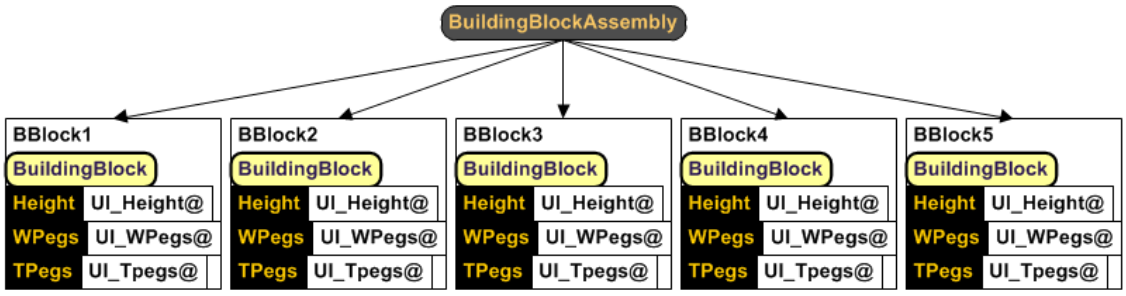
11. Name and describe the remaining distance parameters using the same pattern.

12. Click the green check mark to save the asset and exit the **Orientation Builder**.

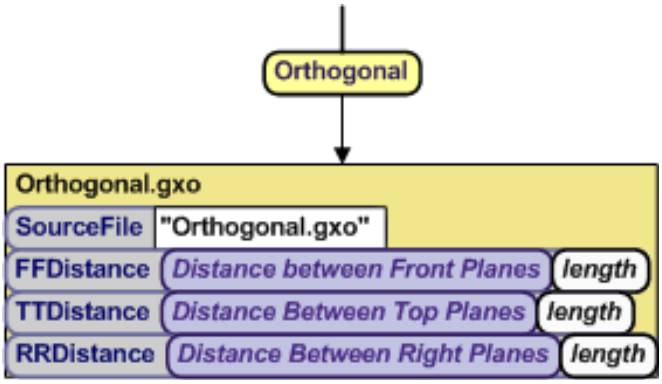
13. Open the `Building Block` Project.

14. Import the `Orthogonal` **Orientation Asset** into the Project. Create additional pages as needed.

15. Create additional `Building Block` **Items** so that there are a total of five, numbered 1 through 5. (`BBlock1`, `BBlock2`, etc.)



16. Add the `Orthogonal` **Orientation Asset**, and attach it to a **Position** shape named `Orthogonal`.



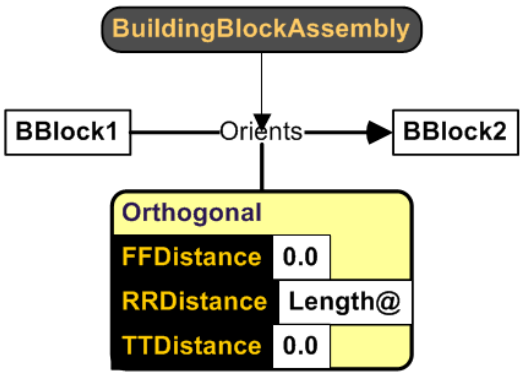
17. Provide default values for the distances.

18. Create new **Item** shapes named `BBlock1` and `BBlock2`.

19. Place an **Orients** arrow between `BBlock1` and `BBlock2`.

20. Attach a **Position** shape to the Orients arrow, named `Orthogonal`.

21. Add **Attribute/Formula** shapes to the new `Orthogonal` **Position** shape, named `FFDistance`, `RRDistance`, and `TTDistance`, to override the default values.



22. Set the value of `FFDistance` to `0.0`, `RRDistance` to `Length@`, and `TTDistance` to `0.0`.

23. Link an **Assembly** shape named `BuildingBlockAssembly` to the **Orients** arrow to define the scope of the orientation rule.

24. Create additional orientation rules based on the following chart:

From	To	FFDistance	RRDistance	TTDistance
BBlock1	BBlock3	0.0	Length@ / 2.0	Height@
BBlock3	BBlock4	0.0	0.0	Height@
BBlock4	BBlock5	Width@ / 2.0	0.0	Height@

25. Change the **Context Variable** for `UI_Lpegs` by adding a **Valid Values** shape with `2; 4; 6; 8; 10; 12` as the values.

26. **Quick Test** the Project to check for errors.

27. **Save, Deploy, and Run** the Project to see the resulting Assembly.