

Generative Components 101

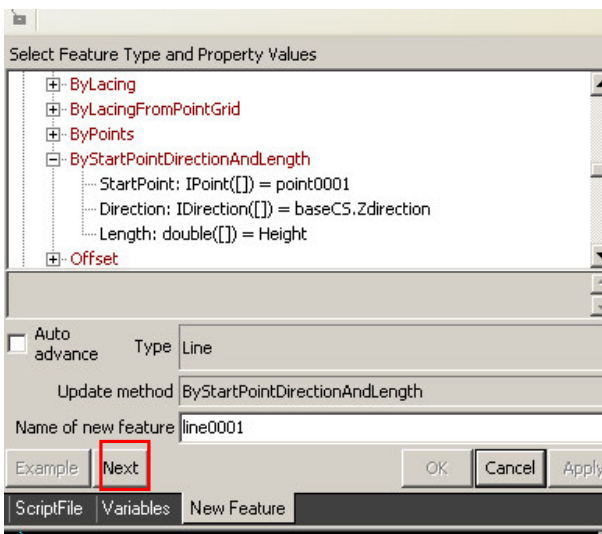
Lesson 8: Flexible Massing Models, Flexible Building Envelopes and Grabbing Metrics

Sounds difficult!

Starts easy! Please note that you may need the latest version of Excel on your machine to be able to save the metrics into a file.

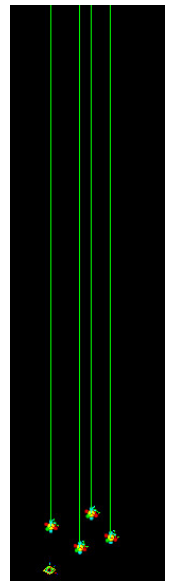
1. Place the first four points of your building.
2. Now we are going to draw the height lines of the building:

Line>ByStartPointDirectionAndLength



Use the baseCS.Zdirection as the Direction of our lines and our GraphVariable Height as the length of our lines.

Use the Next button and change the point to the next one, keeping the other factors the same.

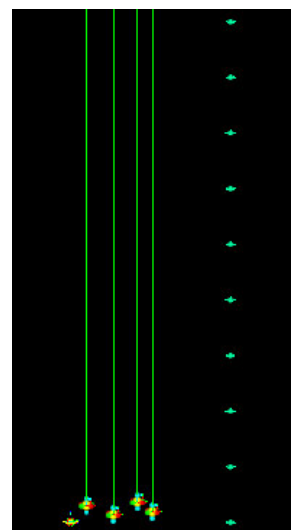
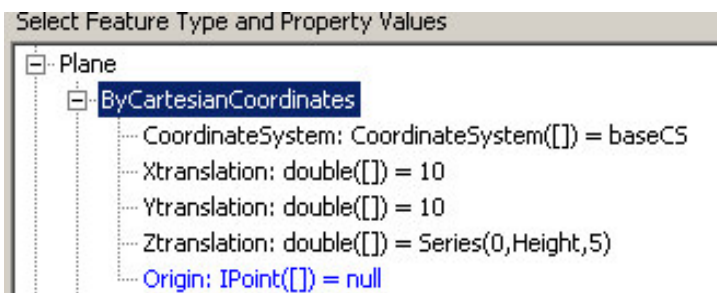


3. Next we want to create some Planes – one for each of our floors.

Plane>ByCartesianCoordinates

The X and Y Translations define the position of the planes from the BaseCS – I've said that I want my planes to be at the coordinate 10,10 from my BaseCS.

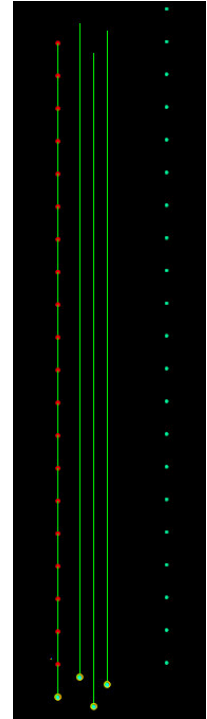
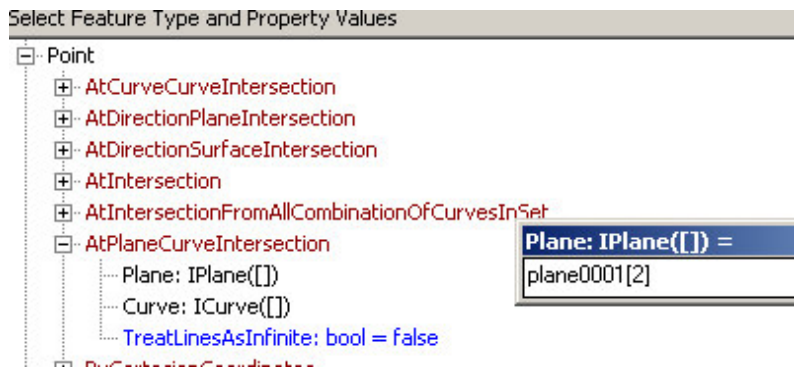
We'd like a Plane on each floor of our building so I have used a Series to define the position of each plane in the Z direction (start, end, increment) or (start at 0, end at GraphVariable Height, in increments of 5 meters.)



Our GraphVariable Height is now controlling the height of the lines and the number of planes.

4. Now we want to create a parametric relationship between the lines and the planes. In other words we would like a point located at the intersection of our lines and the Planes.

Points>PlaneCurveIntersection

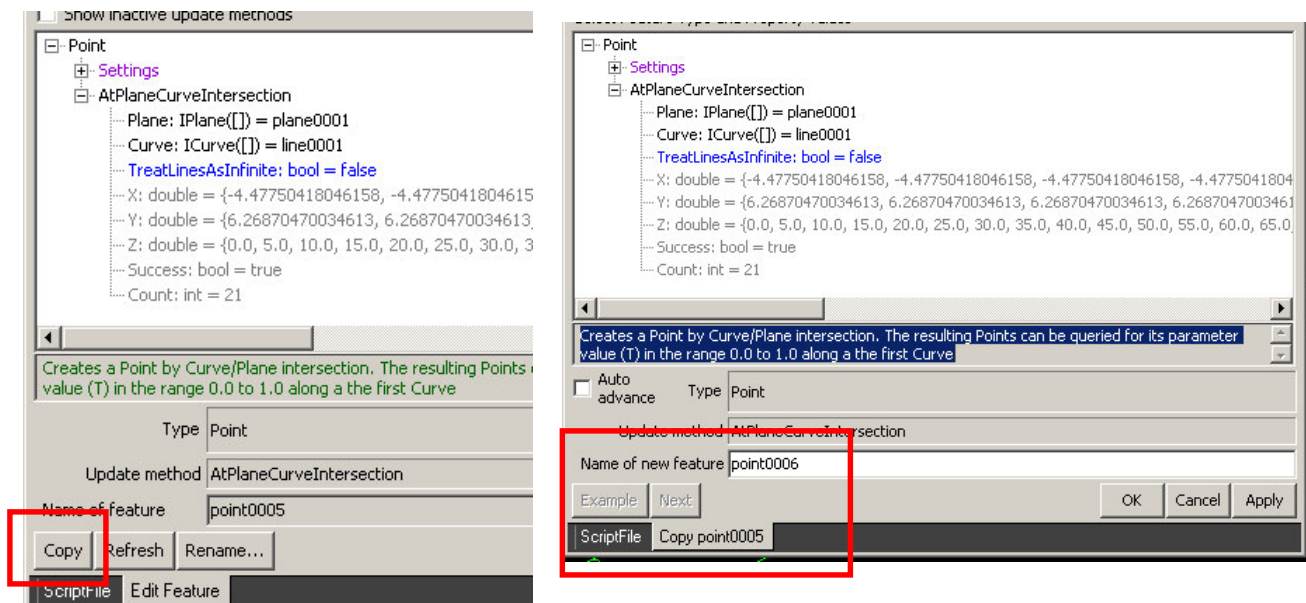


Choose any of the planes but take off the incidence number in brackets. This will mean that we are dealing with all the planes and not just a specific one. Choose the first line for the Curve.

Note: I had to save and reopen my script at this stage – if your points (or planes) don't look to be the same units as the lines then closing and reopening the script seems to sort it out!

5. Now we need to do the same for the other lines. To make it easy lets use the Edit feature tool.

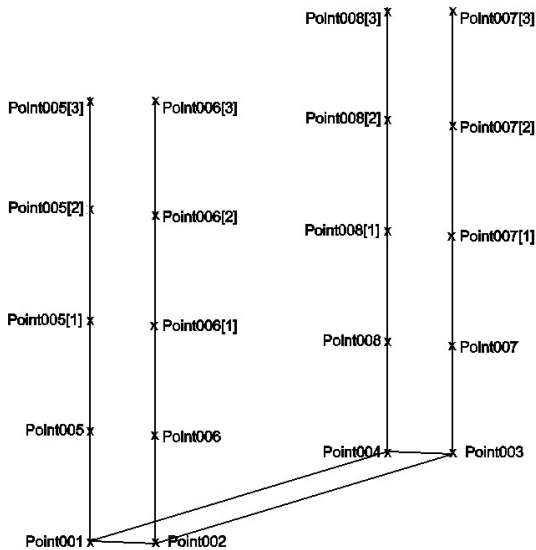
Click on the Edit tool and choose one of the red points. Choose the Copy button at the bottom of the dialog. This will change the feature name to the next incremental number. All you have to do now is change the Line001 to line002 and Apply. Again, copy for lines 3 and 4.



Note: Again, some of my points didn't quite end up on my lines! I just saved and reopened the script to clear it!

6. Now we want to create a shape for each of our floors.

If we examine the points on our lines so far. They are arranged like the diagram. We started off with points 1-4, then created points 5-8 and instances of these points rise in the Z direction.



If we drew this in a two dimensional array or matrix it would look like a continuation of this:

| | | | |
|----------|-------------|-------------|-------------|
| Point005 | Point005[1] | Point005[2] | Point005[3] |
| Point006 | Point006[1] | Point006[2] | Point006[3] |
| Point007 | Point007[1] | Point007[2] | Point007[3] |
| Point008 | Point008[1] | Point008[2] | Point008[3] |

Glossary: Array. An array is a group of individual items

A one-dimensional array looks like a simple linear list: {value1, value2, value3}.

A two dimensional array is a Matrix or Grid: {value1, value2, value3},{value4, value5, value6}.

When a feature requires an input of an array it is indicated by []

When a feature requires an input of a single value or an array it is indicated by ([])

These two approaches can also be combined []([]) meaning that the property expects an array of points, but optionally can be given an array of an array of points!!

We want to create a shape from the vertices point005 to point006 to point007 to point008 to point005 using **Shape>Vertices**.

Our matrix however is read from left to right. This means that even though we place in the correct collection of points, it wants to create our shape up the z axis through Point005, Point005[1], Point005[2], Point005[3] instead. As a result we don't get the shape we are hoping for!

Instead we must use the Function called Transpose.

Glossary: Transpose. Ask the Microsoft thesaurus what Transpose means and it comes up with swap/reverse/switch/invert/reorder/rearrange/change. In simple terms it swaps the rows and columns of a two dimensional array. For example array {{A,B},{C,D}} becomes {{A,C},{B,D}}.

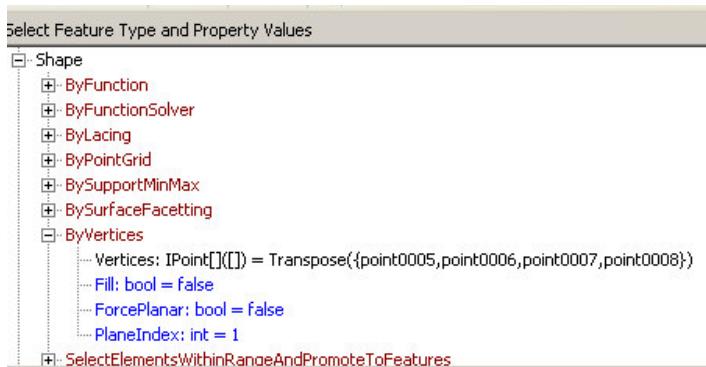
Note: A one-dimensional array looks like a simple linear list: {value 1, value 2, value 3}.

A two-dimensional array is a Matrix or Grid: {value 1, value 2, value 3},{value 4, value 5, value 6}.

Using Transpose will change our Matrix so that the rows are columns, and the columns are rows:

| | | | |
|-------------|-------------|-------------|-------------|
| Point005 | Point006 | Point007 | Point008 |
| Point005[1] | Point006[1] | Point007[1] | Point008[1] |
| Point005[2] | Point006[2] | Point007[2] | Point008[2] |
| Point005[3] | Point006[3] | Point007[3] | Point008[3] |

Now when we read our matrix from left to right we are getting the right points that match our collection.



When we write the collection we take of the [1] instance number to indicate that we want shapes on all our points, not just the defined ones.

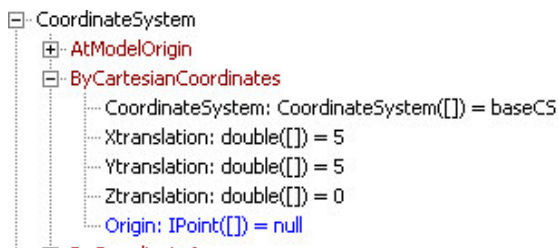
Remember the () Round Brackets are used to define **functions** like Transpose. The Curly brackets are defining the **collection of points**.

Note: “Good god”, I hear you scream, “I put in the correct points, why the hell do I need to transpose at all? It doesn’t make sense!” I agree – Bentley, what is going on here?

7. Now that we have our shapes we want to be able to create some text for our area calculations.

Lets create a new Coordinate System for the text.

CoordinateSystem>ByCatesianCoordinates



I defined mine as being (5,5,0) from the BaseCS.

Now we can place our text field:

Text>ByCatesianCoordinates

All pretty straight forward until we get to the field:

TextString: string =

We want our text to say the total area of all the floors. Because we don’t know what this will be we need to use the property Shape001.TotalArea. (NOTE: when you usually place a dot you can pick the property from a list. This time we need to write it in ourselves.) We also need to convert this property into text. To do this we use the function CStr. This means, “convert to text string”

TextString: string = CStr(Shape001.TotalArea)

This will give the total area of all our shapes.

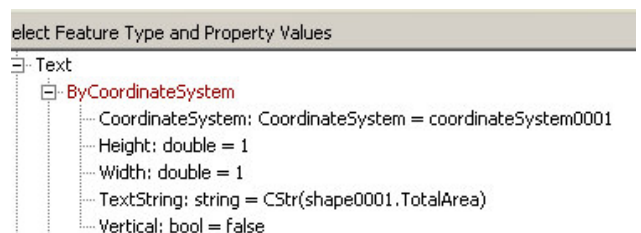
TextString: string = CStr(Shape001[10]TotalArea)

This will give the total area of shape 10 only.

The Bool = false

This makes the text flat on the Xyplane.

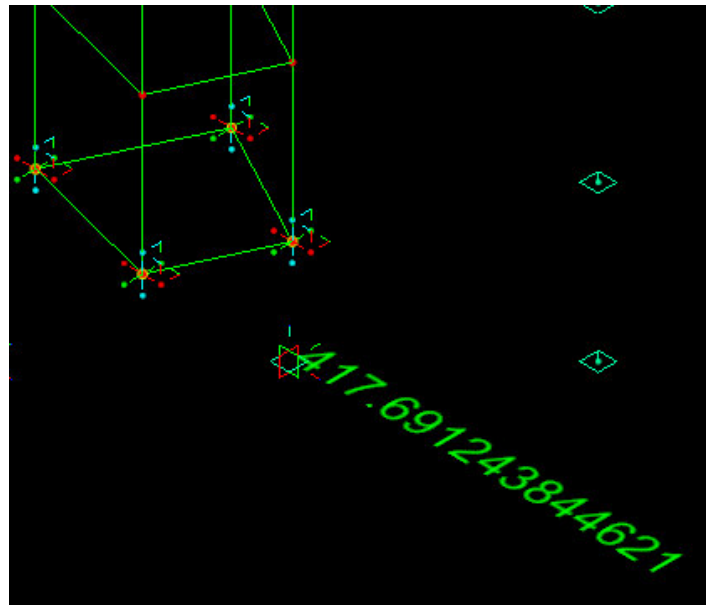
True would make the text vertical on the XZ plane.



8. Now we need to get this area into an Excel file.

Please note that you need office 2003.

Open Excel and create book1.xls (or the name you prefer), save it. In the document decide the area you wish your values to go eg A1:A10 (column A, row 1 to 10). Close Excel.



```
]- ExcelRange
  + ByFunction
  + ByFunctionSolver
  + ReadValue
  - WriteValue
    WorkbookFileName: string = "c:\\book2.xls"
    SheetName: string = "sheet1"
    RangeAddress: string = "A1:A20"
    Value: IValue = shape0001.Area
```

Open the excel file again and your list of areas per floor should be entered!

| | A | B | C | D | E |
|----|----------|---|---|---|---|
| 1 | 19.89006 | | | | |
| 2 | 19.89006 | | | | |
| 3 | 19.89006 | | | | |
| 4 | 19.89006 | | | | |
| 5 | 19.89006 | | | | |
| 6 | 19.89006 | | | | |
| 7 | 19.89006 | | | | |
| 8 | 19.89006 | | | | |
| 9 | 19.89006 | | | | |
| 10 | 19.89006 | | | | |
| 11 | 19.89006 | | | | |
| 12 | 19.89006 | | | | |
| 13 | 19.89006 | | | | |
| 14 | 19.89006 | | | | |
| 15 | 19.89006 | | | | |
| 16 | 19.89006 | | | | |
| 17 | 19.89006 | | | | |
| 18 | 19.89006 | | | | |
| 19 | 19.89006 | | | | |
| 20 | 19.89006 | | | | |
| 21 | | | | | |

