

bimorph

digital engineering

WHAT WE DO



COMPUTATIONAL BIM



APP AND NODE DEVELOPMENT



STRATEGY



BIM COORDINATION



INFORMATION MANAGEMENT



WORKSHOPS + SUPPORT

SUPPORT



AUTODESK
REVIT

Dynamo



GENERATIVE COMPONENTS

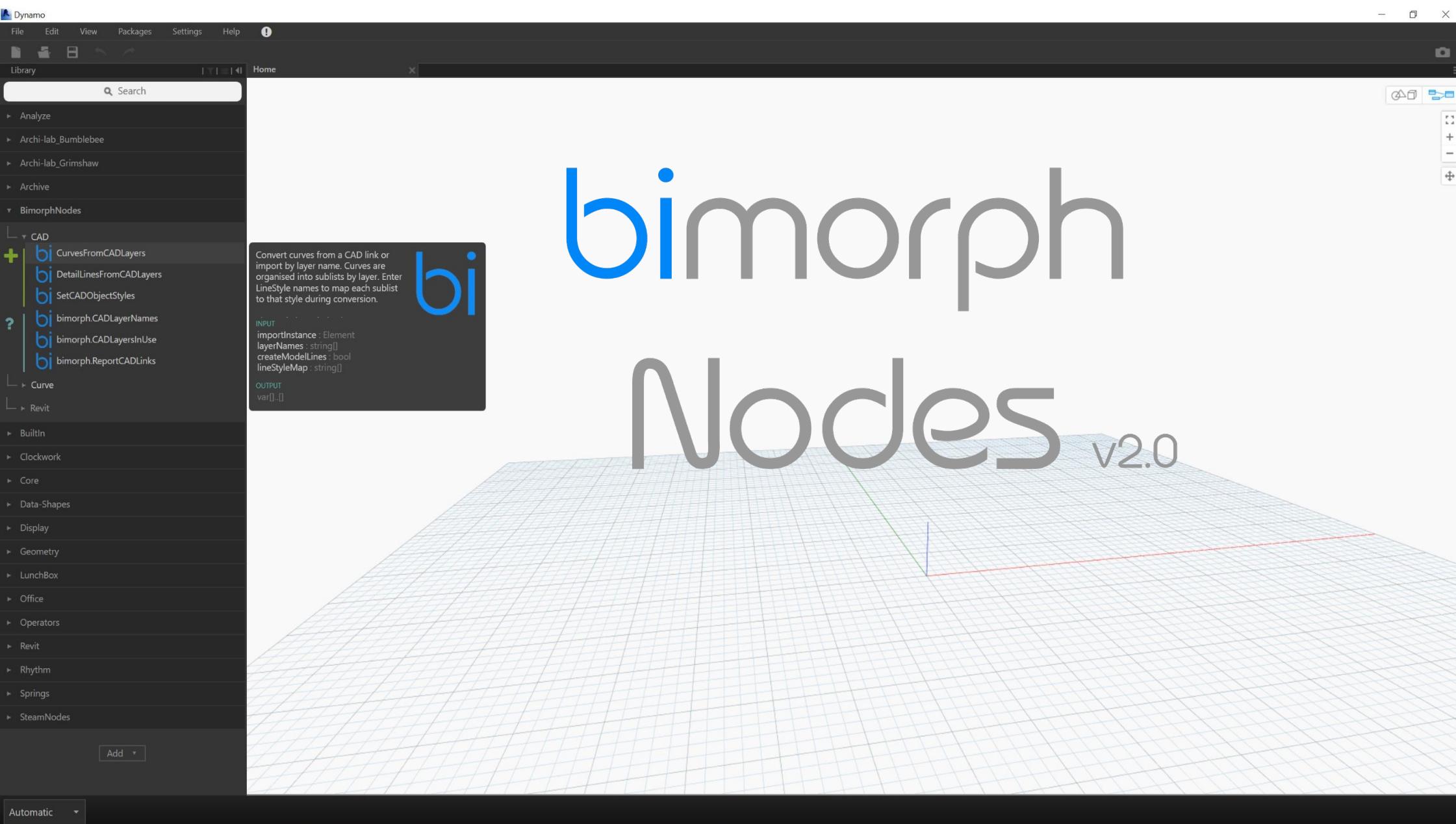


python™



Microsoft®
Visual C#

THE UK'S FIRST COMPUTATIONAL BIM CONSULTANCY

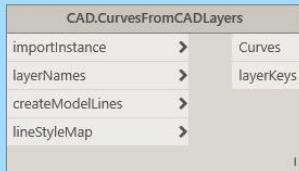


BimorphNodes v2.0

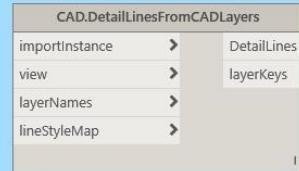
Bimorph Nodes are a collection of powerful utility nodes that extend Dynamo for Revit.

Visit bimorph.co.uk/bimorph-nodes to find out more, see user guides and download example graphs

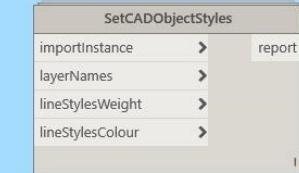
CAD



Visit bimorph.co.uk/bimorph-nodes to find out more, see user guides and download example graphs



Visit bimorph.co.uk/bimorph-nodes to find out more, see user guides and download example graphs



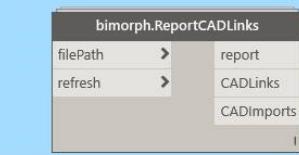
Set weight and colour of CAD link or import layers Object Styles. Enter the layers to set as strings. The layer names are case-sensitive. The line weight and colour lists are mapped to the layerNames list using longest lacing



Get the full list of layer names from a CAD Link or Import



Get the list of layer names from a CAD link or import that are in use and contain curves



Generates a report of all CAD links and imports in the document with key metrics. Linked and Imported CAD links are output into lists to aid with CAD link management.

Curve

Curve.RemoveDuplicateCurves

```
curves > cleanedCurves
retainByLineStyles > duplicateCurves
deleteRevitCurves > report
```

Removes duplicates from a list of Curves. Dynamo Curves, Revit Model or Detail Lines are all supported. Revit Curves can be deleted from the document. Input LineStyle names to retain curves of that style.

LineStyle list are iterated until a match is found. If no matches are found, first-in first-out rules apply. To simplify model element selection in Revit, heterogeneous lists (any element type) can be input as the list is filtered before processing.

Sheets

Sheets.DuplicateSheets

```
sheets > report
run > Sheets
duplicateWithViews > Views
duplicateOption >
suffix >
prefix >
```

Duplicate selected Sheets with options to control if placed Views are also duplicated and the method of duplication.

The report output provides a list of any Sheets and/or Views that fail to duplicate or View names that were cleaned of illegal characters

bimorph.RenumberRenameSheets

```
sheets > report
newNumbers >
newNames >
run >
```

Renumber and/or rename Revit sheets. To renumber or rename only, leave the other input empty. Sheets and newNumbers list's must be equal in length to run.

The node uses an algorithm that handles common Revit sheet renumber warnings for more consistent results.

bimorph.SheetsFromSchedule

```
sheetListScheduleView > Sheets
```

Collects and returns all Revit sheets from a Sheet List Schedule

Ensure the Sheet Number field is included in the Schedule

Schedules

bimorph.GetScheduleData

```
scheduleView > scheduleData
removeHeadings >
```

Gets all the table data from a given schedule as strings

bimorph.GetScheduleDataColumns

```
scheduleView > scheduleDataColumn
columnIndex >
removeHeadings >
```

Get the specified column data from a given schedule as strings

Column indexes start at 0

bimorph.GetScheduleDataRow

```
scheduleView > scheduleDataRow
rowIndex >
```

Get the specified row data from a given schedule as strings

Schedules have a blank row after the headings. Row indexes start at 0

LineStyles

bimorph.CreateNewLineStyles

```
lineStylesName > report
lineStylesWeight > LineStyles
lineStylesColour >
run >
```

Create single or multiple new Line Styles in a Revit document from name, line weight and RGB values

bimorph.GetLineStylesAttributes

```
refresh > lineStylesName
lineStylesWeight >
lineStylesColour >
GraphicStyle >
```

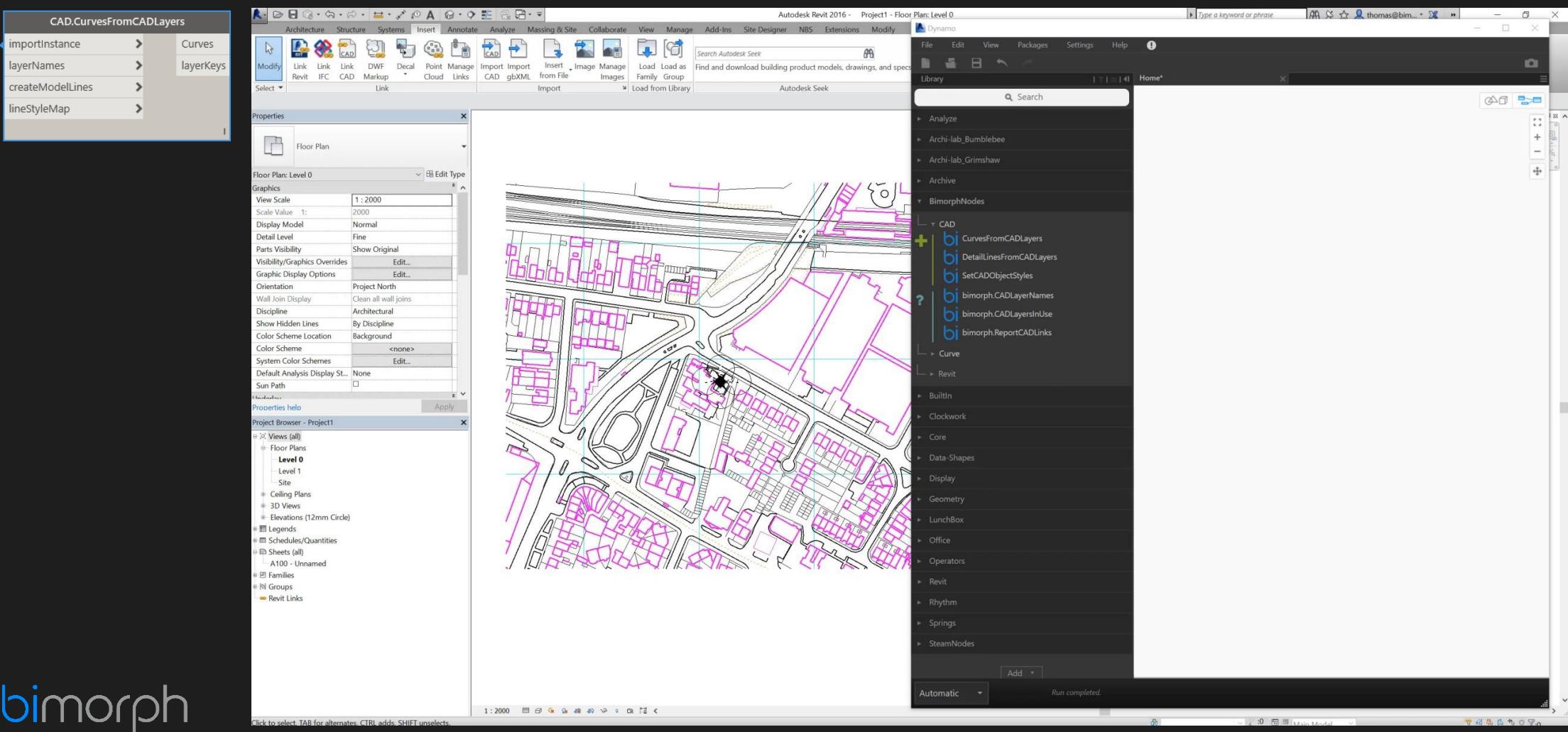
Gets all the Revit Line Styles in the document and reports their Line Style Name, Weight and Colour as strings and integers. The Revit GraphicsStyle (Category) element for each Line Style is also output.

BIMORPH NODES V2.0

Bimorph Nodes are a collection of powerful utility nodes
that extend Dynamo for Revit

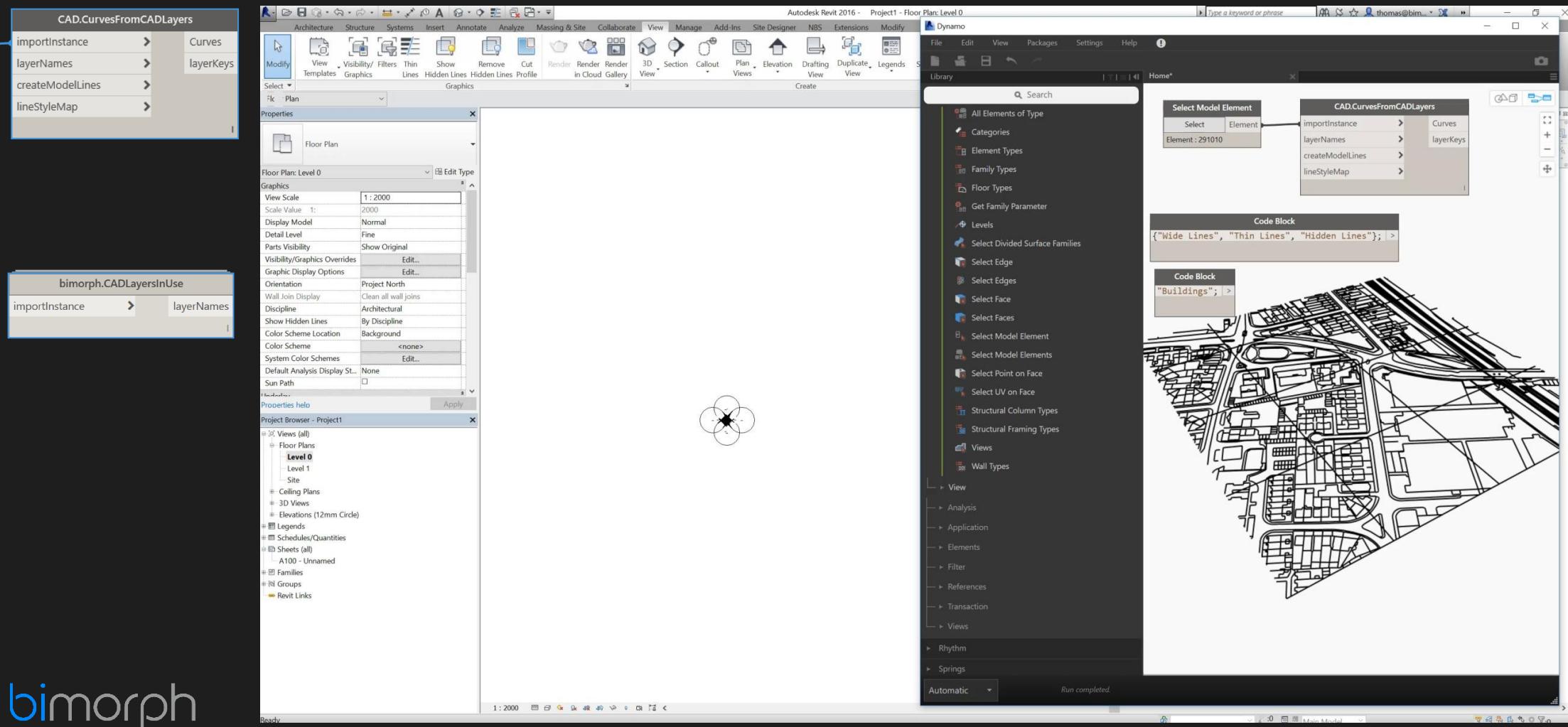
1. 15 nodes designed for Revit interop workflows
2. V2.0 Released 26 February 2017
3. First to enable filtering of curves from CAD Links or Imports via layer name
4. First to provide duplicate curve clean-up

NODE: CurvesFromCADLayers



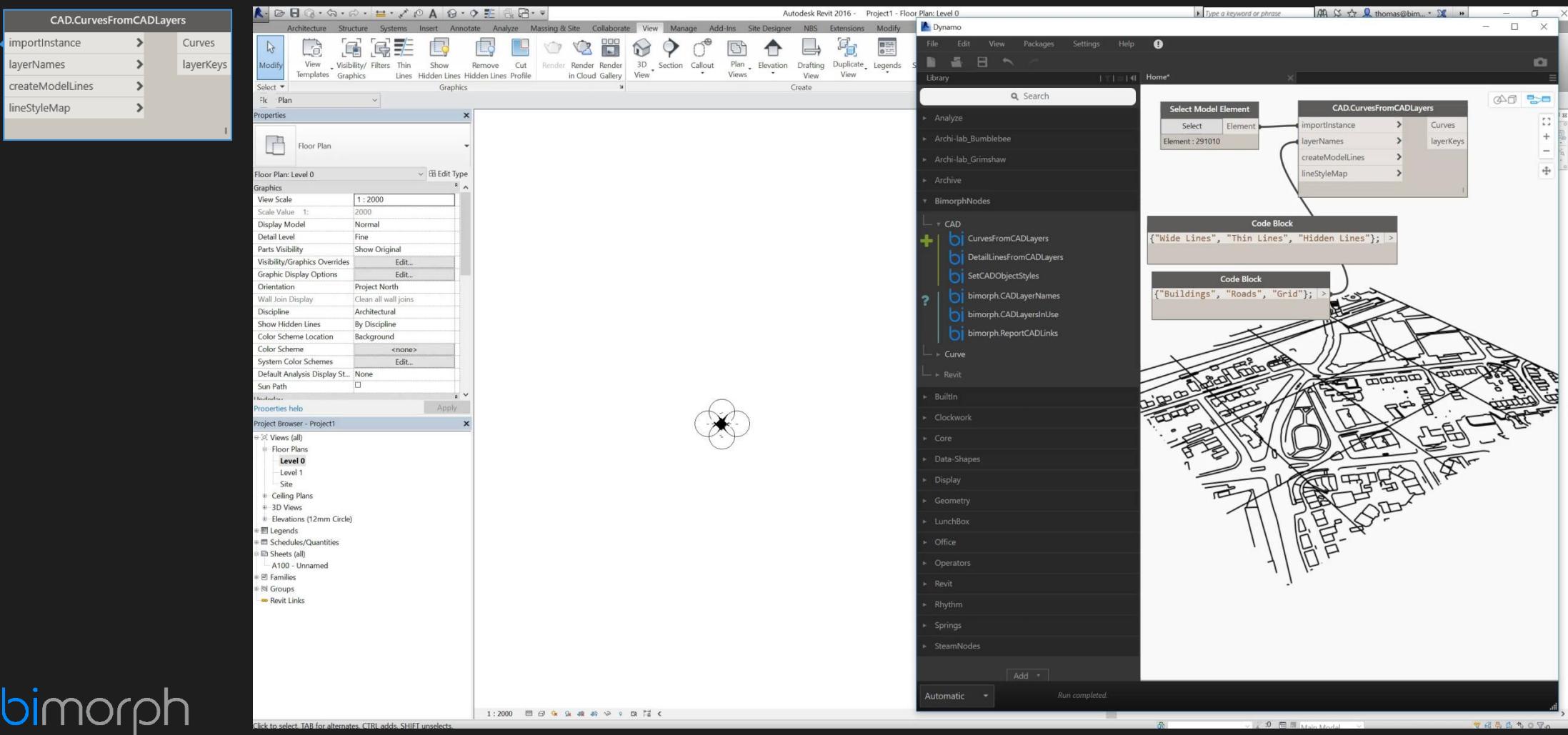
bimorph

FILTER CURVES BY LAYER NAME

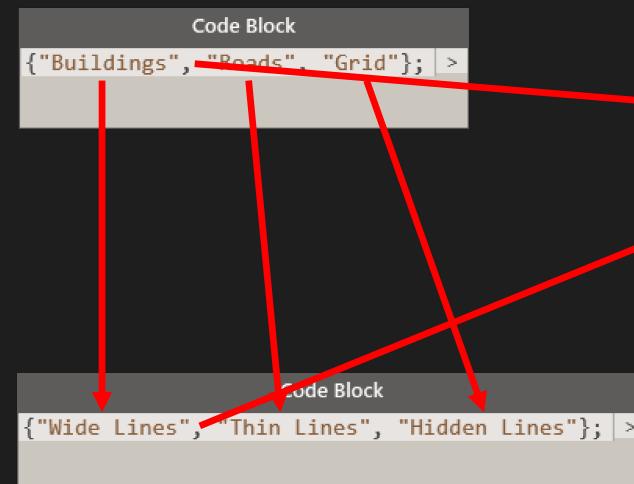
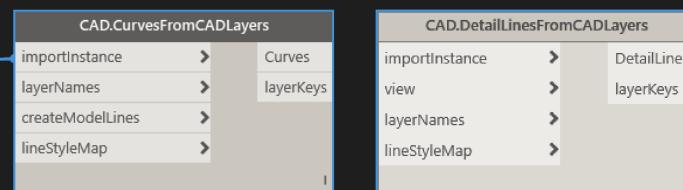


bimorph

MAPPING LINE STYLES

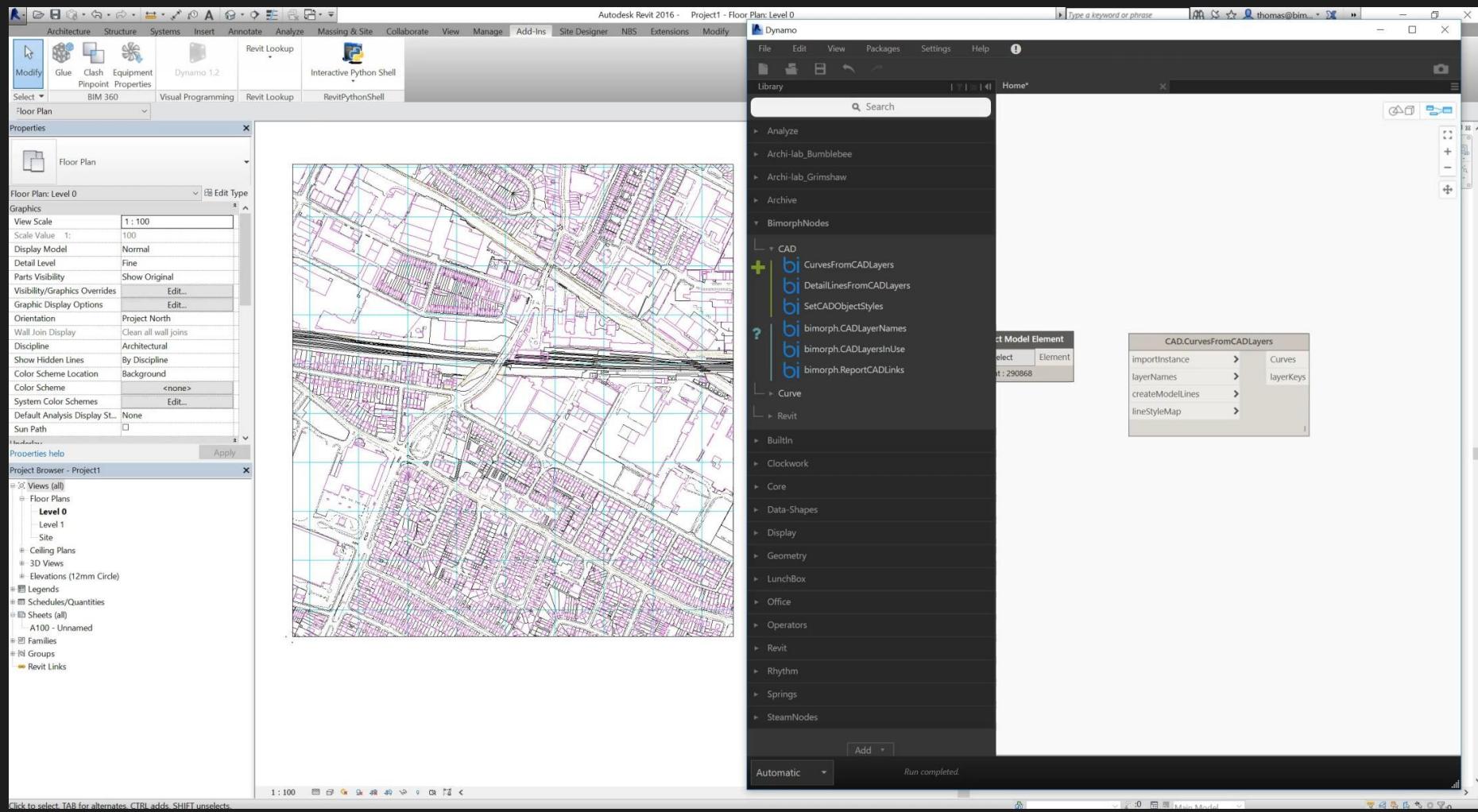


MAPPING LINE STYLES



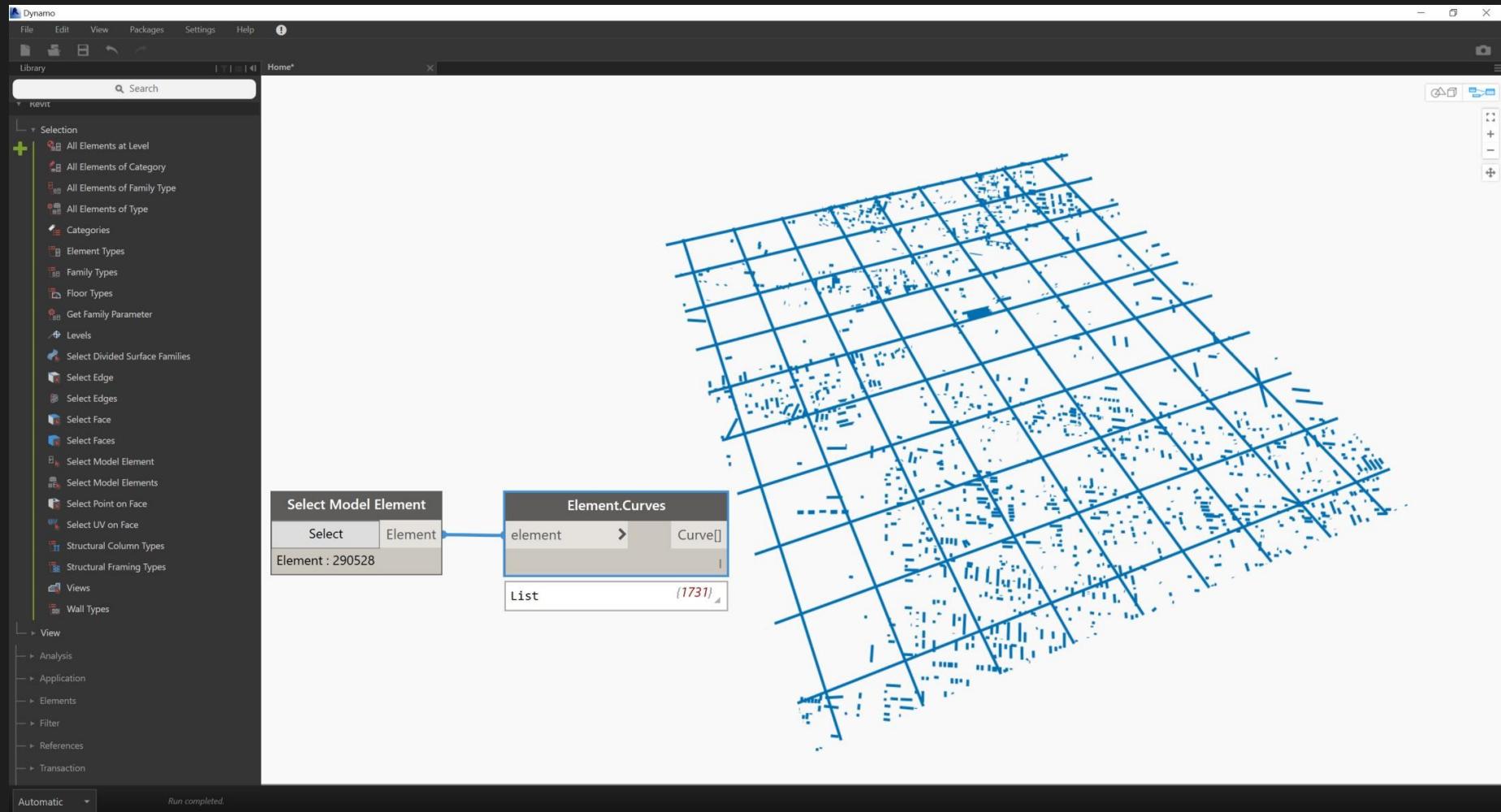
bimorph

CIRCUMVENT 10K ELEMENT LIMIT



bimorph

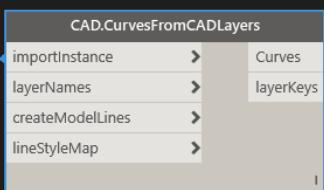
OPTIMISED ALGORITHM



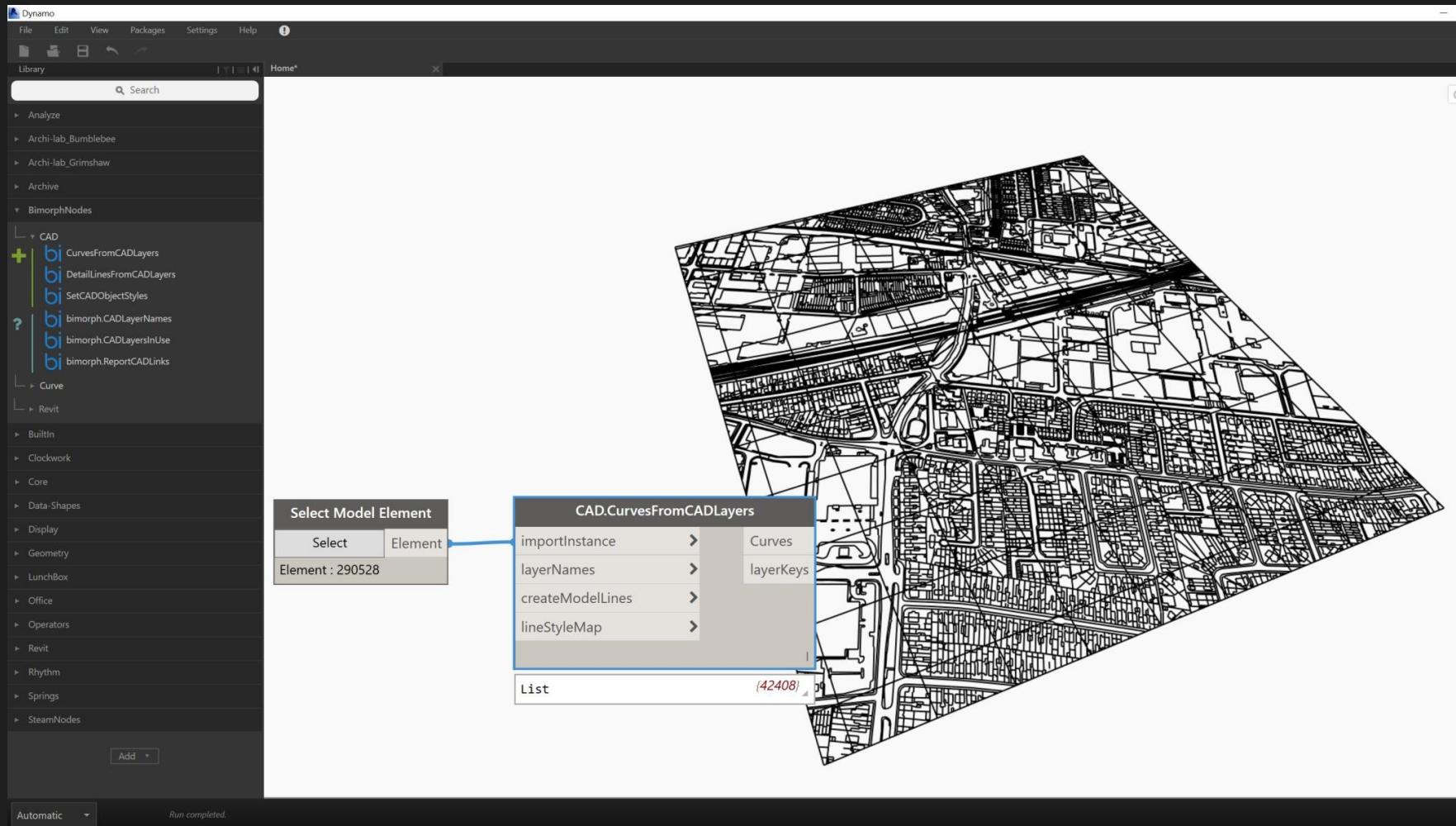
bimorph

Without BimorphNodes,
using OOTB nodes fail to
convert many curve
types from CAD Links or
Imports

OPTIMISED ALGORITHM

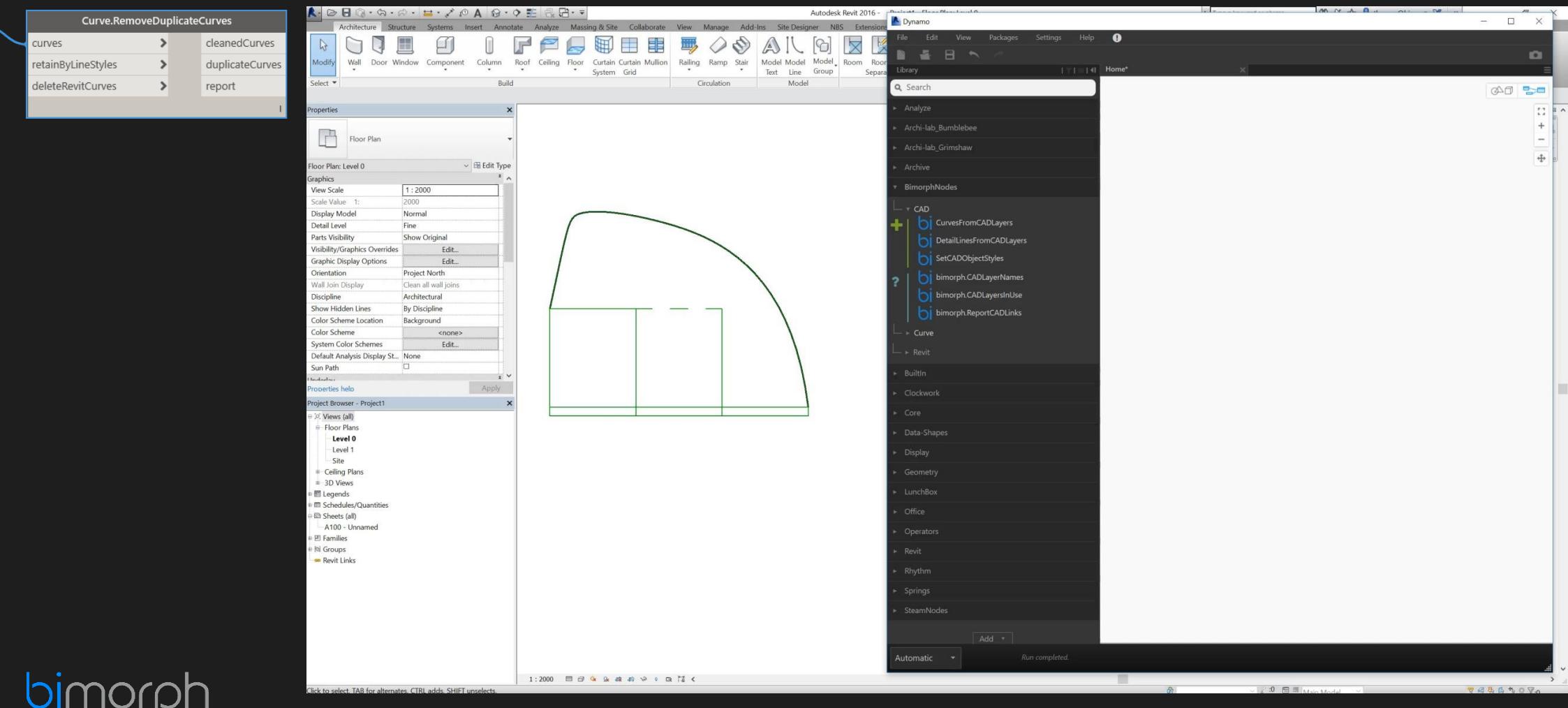


With BimorphNodes, curve conversion is optimised to convert many curve types found in CAD Links or Imports. In this example, **42K** curves were converted vs only **1.7K** using OOTB *Element.Curve* node (shown in the previous slide)



bimorph

NODE: RemoveDuplicateCurves



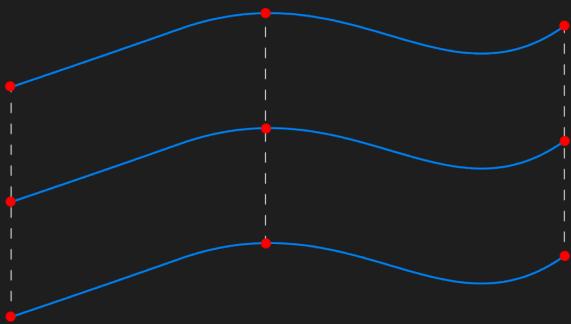
bimorph

COMPUTATIONAL THINKING + APPLIED MATHEMATICS

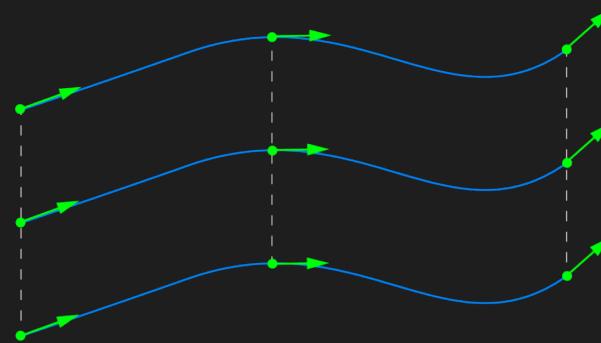
Curve.RemoveDuplicateCurves		
curves	>	cleanedCurves
retainByLineStyles	>	duplicateCurves
deleteRevitCurves	>	report

All 3 tests == `true` equates to a duplicate:

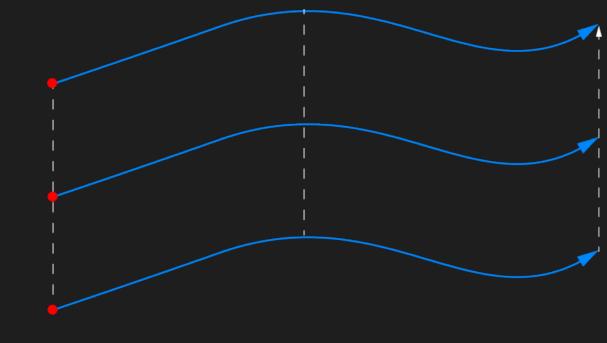
1. Coincident start, mid,
end points?



2. Start, mid, end vectors
dot product = 1.0?

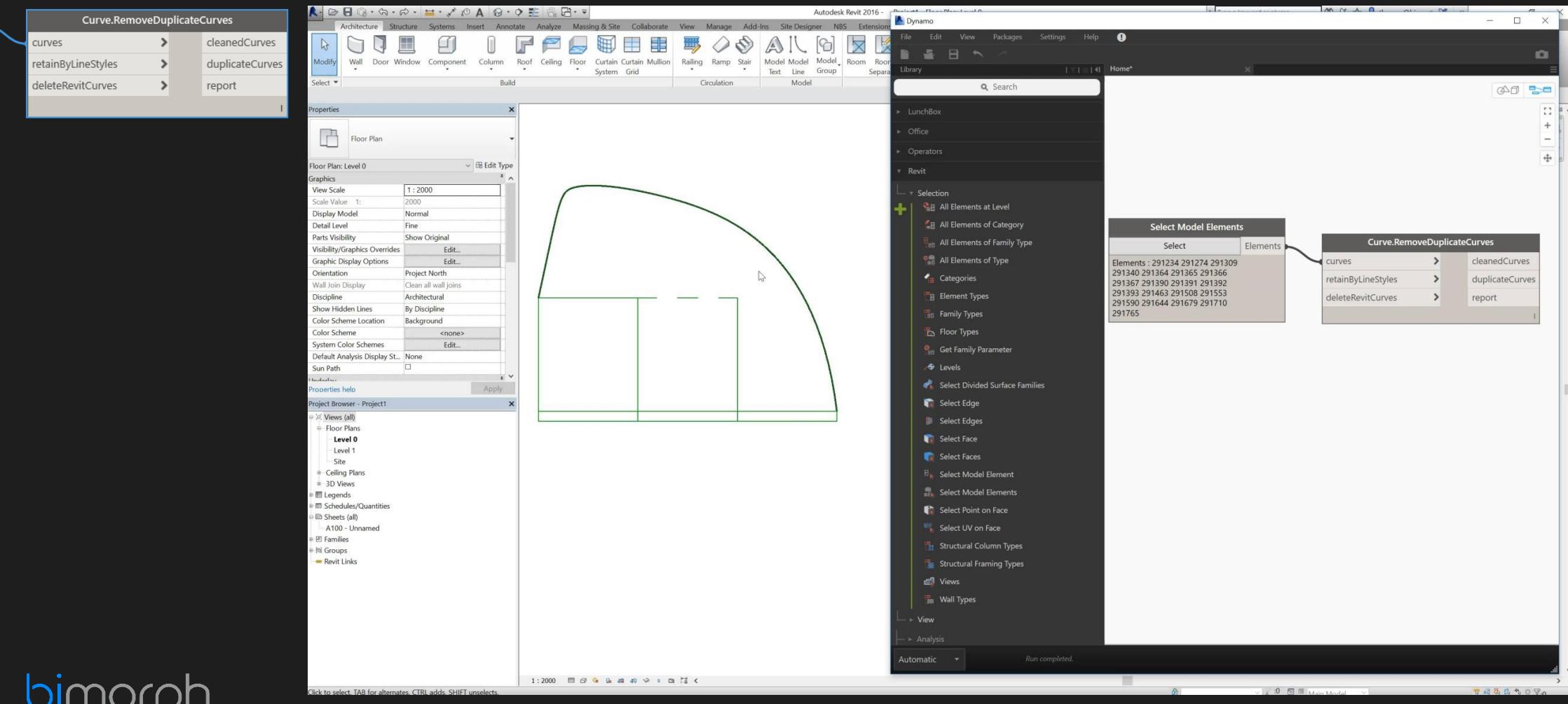


3. Qualify length



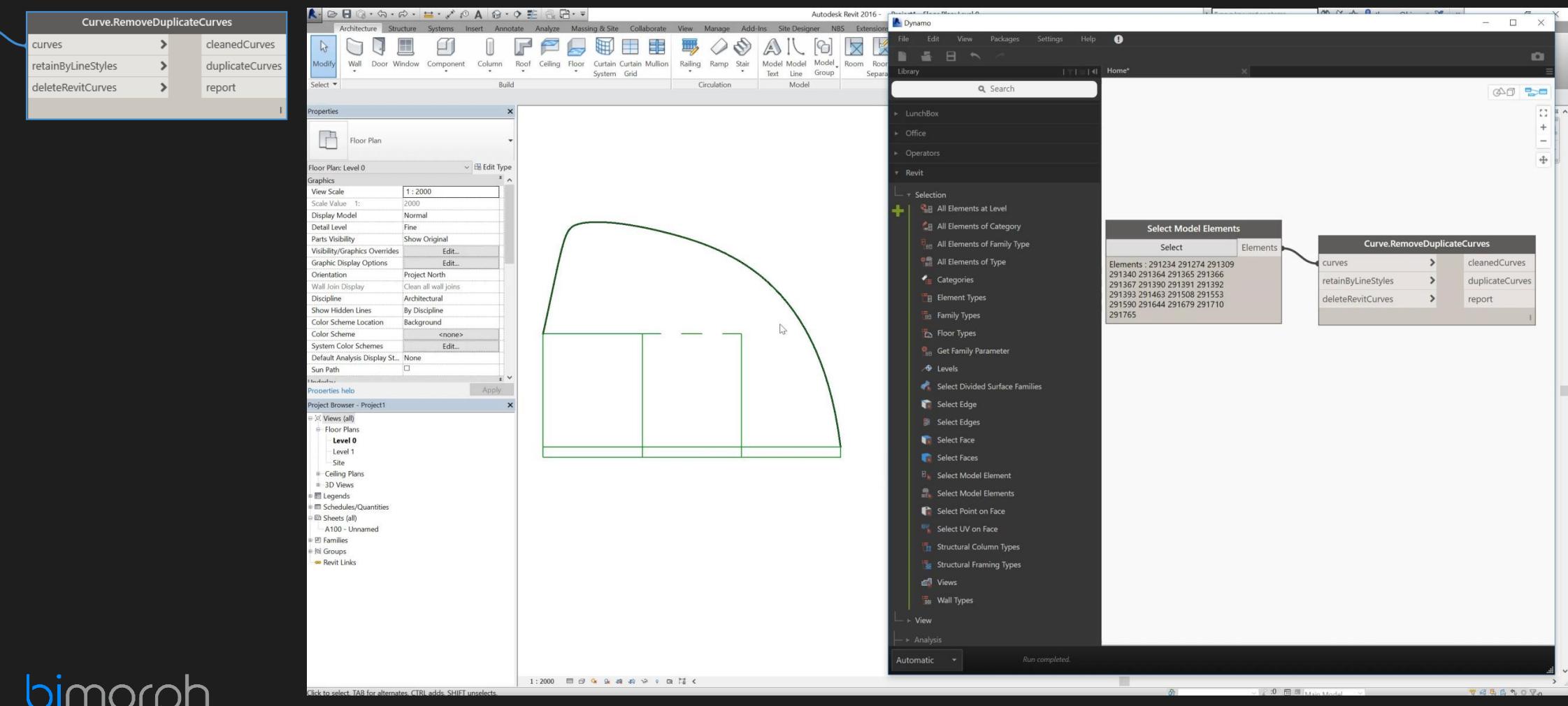
bimorph

DELETE DUPLICATE CURVES

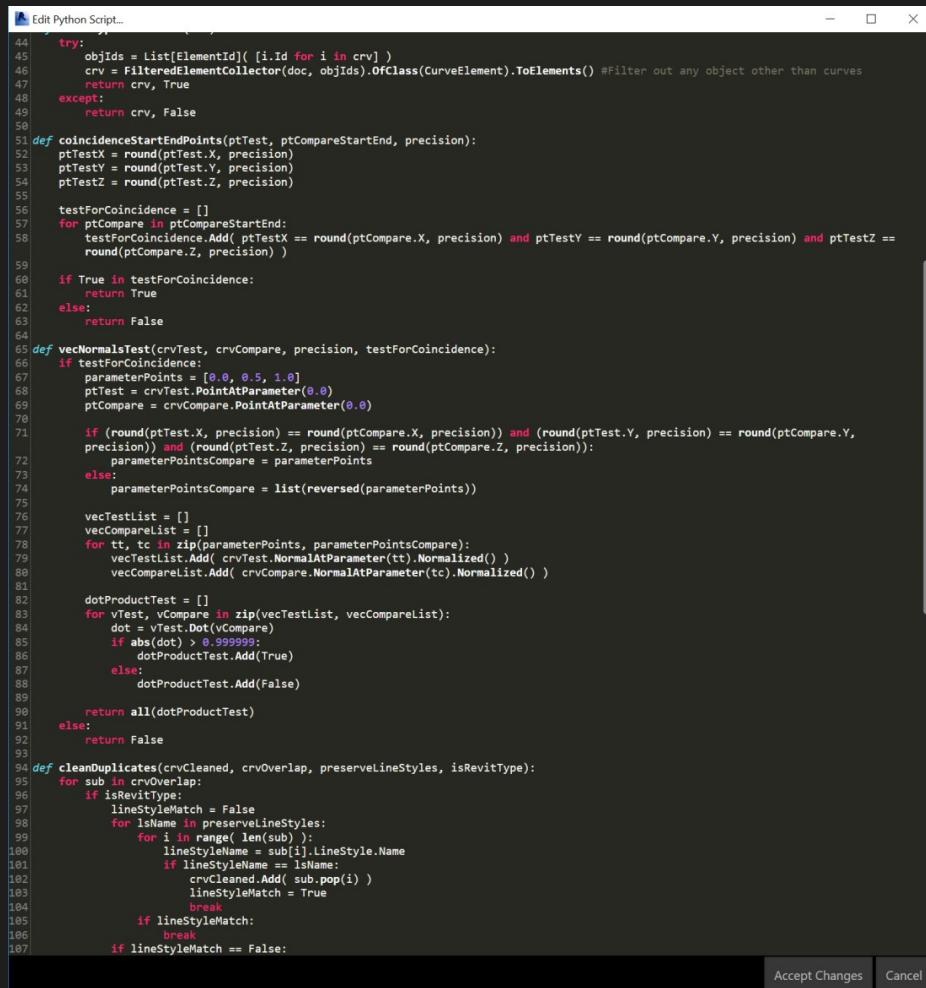


bimorph

RETAIN BY LINE STYLES



SOURCE CODE: RemoveDuplicateCurves



The screenshot shows a dark-themed Python script editor window titled "Edit Python Script...". The code is a Python script named "RemoveDuplicateCurves.py" with line numbers 44 to 107. The script performs several operations:

- It filters a list of elements to find curves.
- It defines a function to check if two points are coincident based on their rounded coordinates.
- It defines a function to test if two vectors are nearly identical by comparing their normalized dot products.
- It defines a function to clean up duplicate curves while preserving line styles.

```
44 try:
45     objIds = List[ElementId]( [i.Id for i in crv] )
46     crv = FilteredElementCollector(doc, objIds).OfClass(CurveElement).ToElements() #Filter out any object other than curves
47     return crv, True
48 except:
49     return crv, False
50
51 def coincidencesStartEndPoints(ptTest, ptCompareStartEnd, precision):
52     ptTestX = round(ptTest.X, precision)
53     ptTestY = round(ptTest.Y, precision)
54     ptTestZ = round(ptTest.Z, precision)
55
56     testForCoincidence = []
57     for ptCompare in ptCompareStartEnd:
58         testForCoincidence.Add( ptTestX == round(ptCompare.X, precision) and ptTestY == round(ptCompare.Y, precision) and ptTestZ == round(ptCompare.Z, precision) )
59
60     if True in testForCoincidence:
61         return True
62     else:
63         return False
64
65 def vecNormalstTest(crvTest, crvCompare, precision, testForCoincidence):
66     if testForCoincidence:
67         parameterPoints = [0.0, 0.5, 1.0]
68         ptTest = crvTest.PointAtParameter(0.0)
69         ptCompare = crvCompare.PointAtParameter(0.0)
70
71         if (round(ptTest.X, precision) == round(ptCompare.X, precision)) and (round(ptTest.Y, precision) == round(ptCompare.Y, precision)) and (round(ptTest.Z, precision) == round(ptCompare.Z, precision)):
72             parameterPointsCompare = parameterPoints
73         else:
74             parameterPointsCompare = list(reversed(parameterPoints))
75
76         vecTestList = []
77         vecCompareList = []
78         for tt, tc in zip(parameterPoints, parameterPointsCompare):
79             vecTestList.Add( crvTest.NormalAtParameter(tt).Normalized() )
80             vecCompareList.Add( crvCompare.NormalAtParameter(tc).Normalized() )
81
82     dotProductTest = []
83     for vTest, vCompare in zip(vecTestList, vecCompareList):
84         dot = vTest.Dot(vCompare)
85         if abs(dot) > 0.999999:
86             dotProductTest.Add(True)
87         else:
88             dotProductTest.Add(False)
89
90     return all(dotProductTest)
91 else:
92     return False
93
94 def cleanDuplicates(crvCleaned, crvOverlap, preserveLineStyles, isRevitType):
95     for sub in crvOverlap:
96         if isRevitType:
97             lineStyleMatch = False
98             for lsName in preserveLineStyles:
99                 for i in range( len(sub) ):
100                     lineStyleName = sub[i].LineStyle.Name
101                     if lineStyleName == lsName:
102                         crvCleaned.Add( sub.pop(i) )
103                         lineStyleMatch = True
104                         break
105                     if lineStyleMatch:
106                         break
107                     if lineStyleMatch == False:
```

170 lines in total

bimorph

VISIT BIMORPH.CO.UK/BIMORPH-NODES



[HOME](#) [WHAT WE DO](#) [WORKSHOPS](#) [BIMORPH NODES](#) [CONTACT US](#)

Search...

BimorphNodes bi

- ▼ CAD
 - CADLayerNames
 - CADLayersInUse
 - CurvesFromCADLayers
 - DetailLinesFromCADLayers
 - ReportCADLinks
 - SetCADOObjectStyles
- ▼ Curve
 - RemoveDuplicateCurves
- ▼ Revit
 - ▼ LineStyles
 - CreateNewLineStyles
 - GetLineStylesAttributes
 - ▼ Schedule
 - GetScheduleData
 - GetScheduleDataColumns
 - GetScheduleDataRows
 - ▼ Sheets
 - DuplicateSheets
 - RenumberRenameSheets
 - SheetsFromSchedule

CurvesFromCADLayers

INPUTS

importInstance : ImportInstance
layerNames : string[] (optional)
createModelLines : bool (optional)
lineStyleNames : string[] (optional)

OUTPUTS

Curves: Curves[][] or ModelLines[][]
layerNameKeys : string[]

DESCRIPTION

Convert curves from a CAD link or import by layer name to Dynamo curves or Revit Model Lines.

Layer names can be input to filter and convert only curves on those layers. Curves are output in sublists based on their layer name and mapped to one LineStyle to prevent unwanted CAD LineStyles entering your project. Alternatively, LineStyle names can be input to map each curve sublist to that style during conversion. To convert all layers, leave the layerName input unconnected. Note that any input layerNames and LineStyleNames are case-sensitive.

The node is optimised to convert curve types that typically fail when using Dynamo Element.Curves node or Revit Explode. The node also circumvents the 10000 element limit set by Revit for greater workflow flexibility.

FEATURES

- ▶ Extract curves from CAD Links or Imports and filter by layer name
- ▶ Revit Model Lines can be created directly
- ▶ Line Style names can be input to map Line Styles to curves from each layer
- ▶ Optimised to convert curves that normally fail using standard Dynamo nodes or Revit Explode
- ▶ Circumvents Revit's 10k element limit from import geometry

USER GUIDE

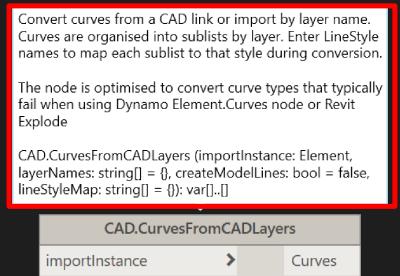
We are busy recording the video user guide; we'll add it to the page soon!



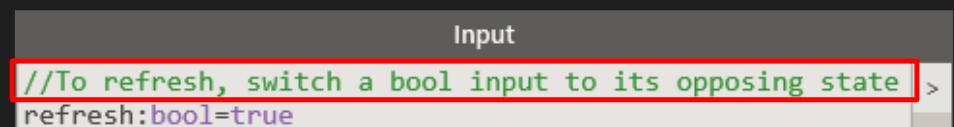
bimorph

CUSTOM NODE PRO TIPS

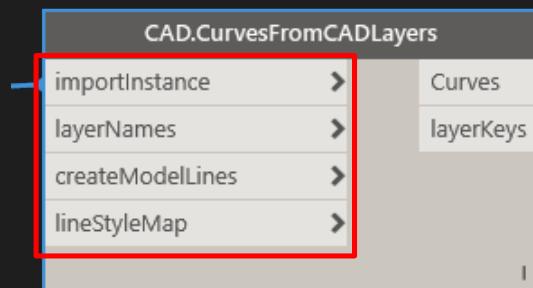
1. Add descriptions to your nodes



3. Add input hints using DesignScript



2. CamelCase to name input/outputs. Be concise. Don't abbreviate



4. Strong-type inputs and declare rank (for arrays)



bimorph

THANKS

bimorph

 info@bimorph.co.uk

 +44 (0) 20 7060 2050

 bimorph.co.uk

 @bimorphBIM

 linkedin.com/company/bimorph-bim

 facebook.com/bimorphBIM