

# Civil 3D and Dynamo

## Dynamic Culvert Design and Analysis

Andrew Milford

Sr. Implementation Consultant





# About the speaker

## Andrew Milford

- Based in Sydney, Australia
- Over 25 years design experience in the Civil Infrastructure industry
- AutoCAD Civil 3D Certified Professional
- Develop processes and automation tools through scripts, AutoLISP, .NET API (C# and VB) and Python



# Learning Objectives

## LEARNING OBJECTIVE 1

Leverage the Civil 3D Corridor model to create 3D drainage culverts in Dynamo

## LEARNING OBJECTIVE 2

Use Dynamo to create complex 3D Drainage culvert models from user-defined parameters

## LEARNING OBJECTIVE 3

Explore Dynamo's Python node to undertake culvert analysis and sizing

## LEARNING OBJECTIVE 4

Create Civil 3D geometry from within the Dynamo workspace for documentation

# Civil 3D and Dynamo





# Culvert Analysis



U.S. Department of Transportation  
**Federal Highway Administration**



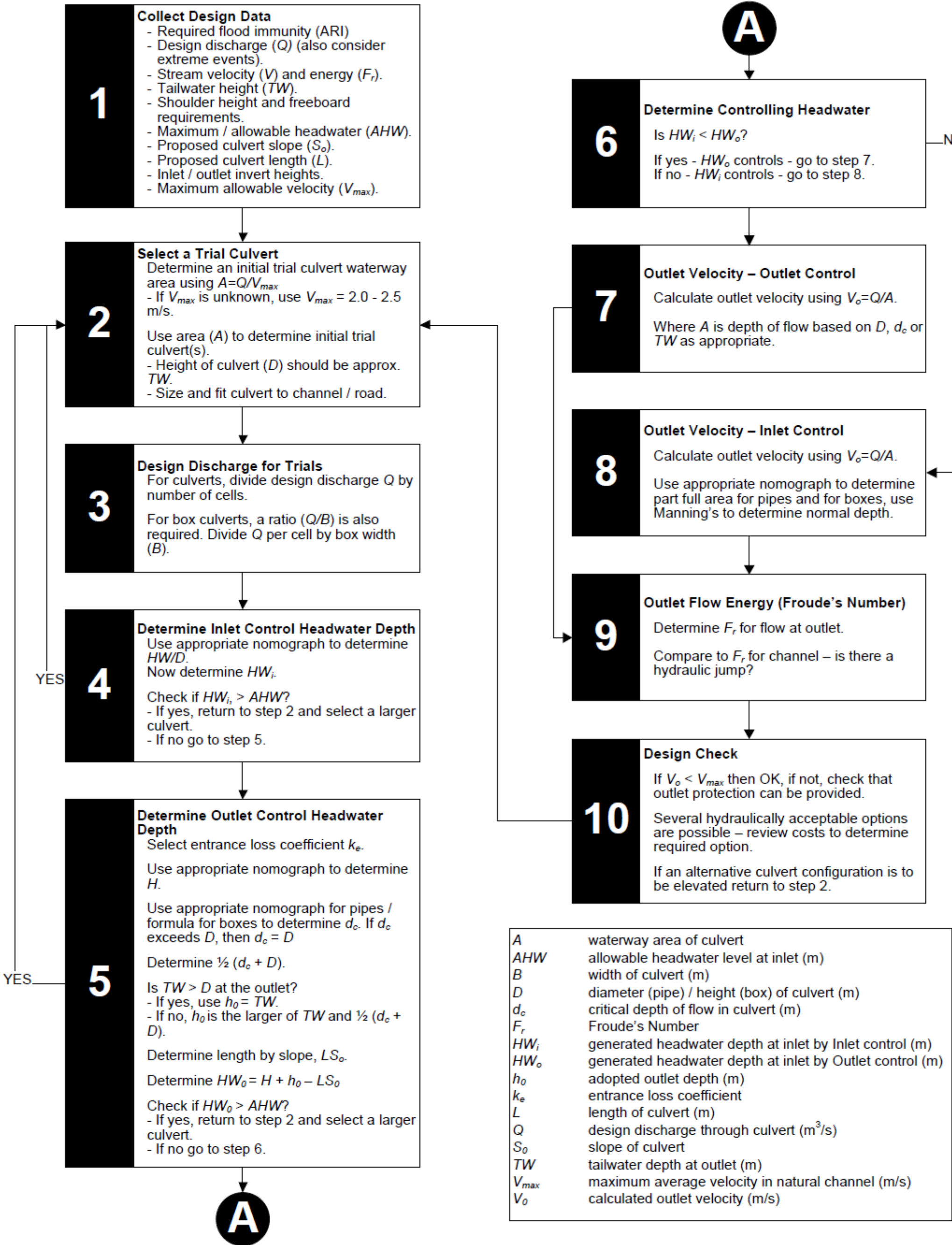


# Culvert Design Guidelines

- Hydraulic Design of Highway Culverts, Third Edition
  - US DOT, Federal Highway Administration
    - HY-8
- Austroroads Guide to Road Design Part 5B
  - Drainage – Open Channels, Culverts and Floodways

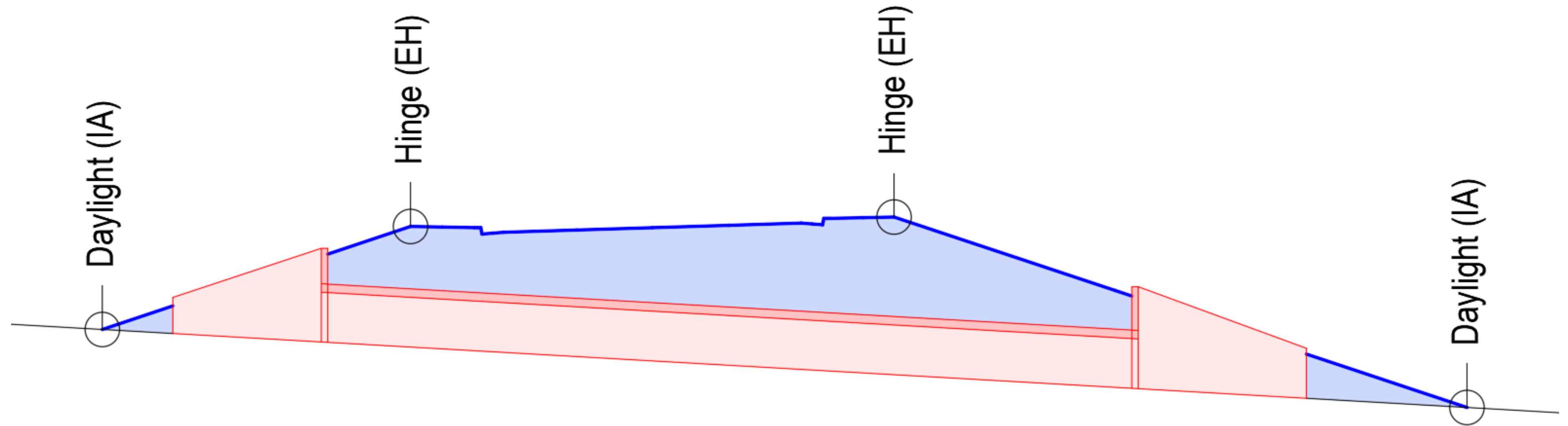


U.S. Department of Transportation  
Federal Highway Administration



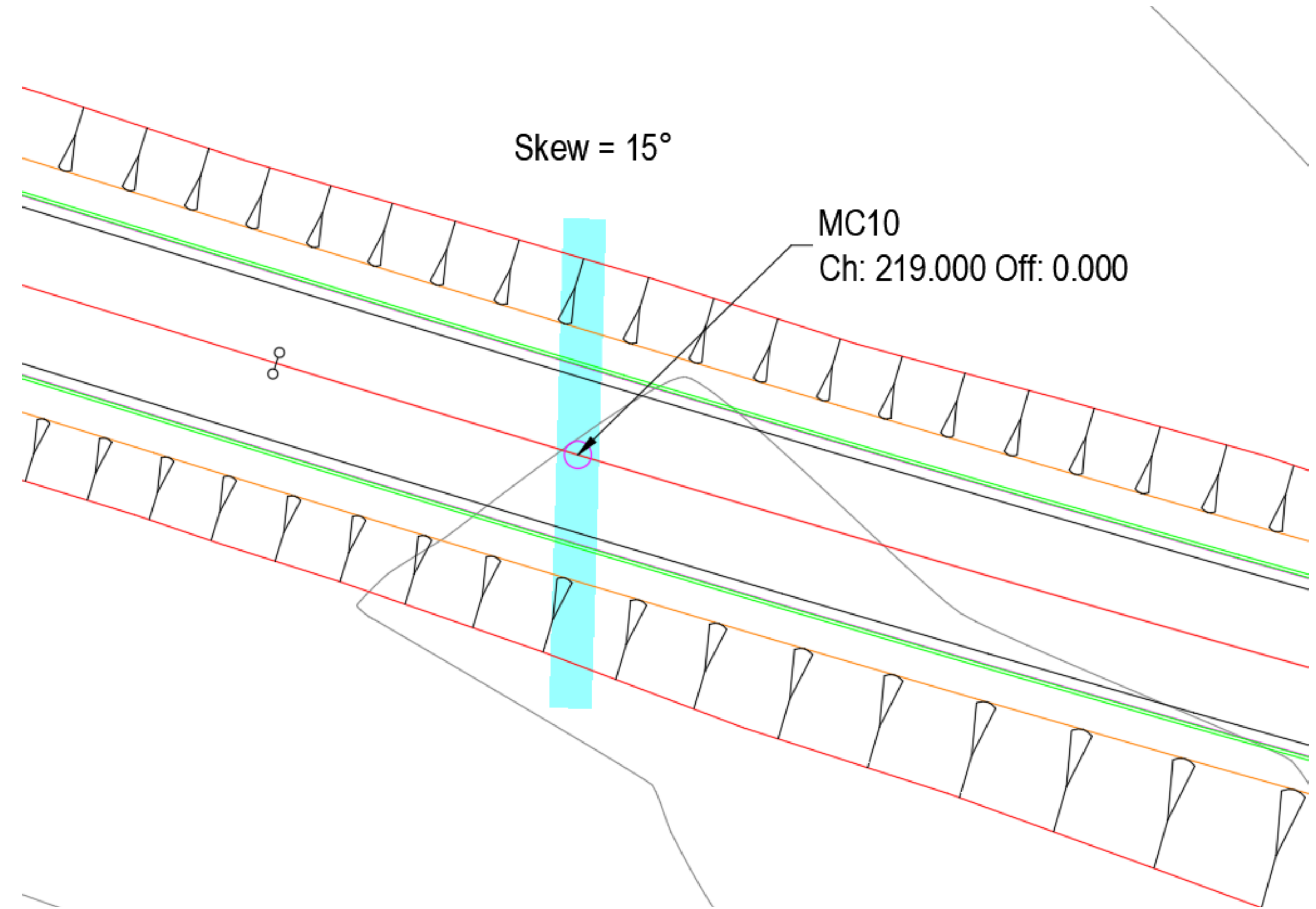
# Civil 3D | Project Setup

- Corridor Codes
- Surface



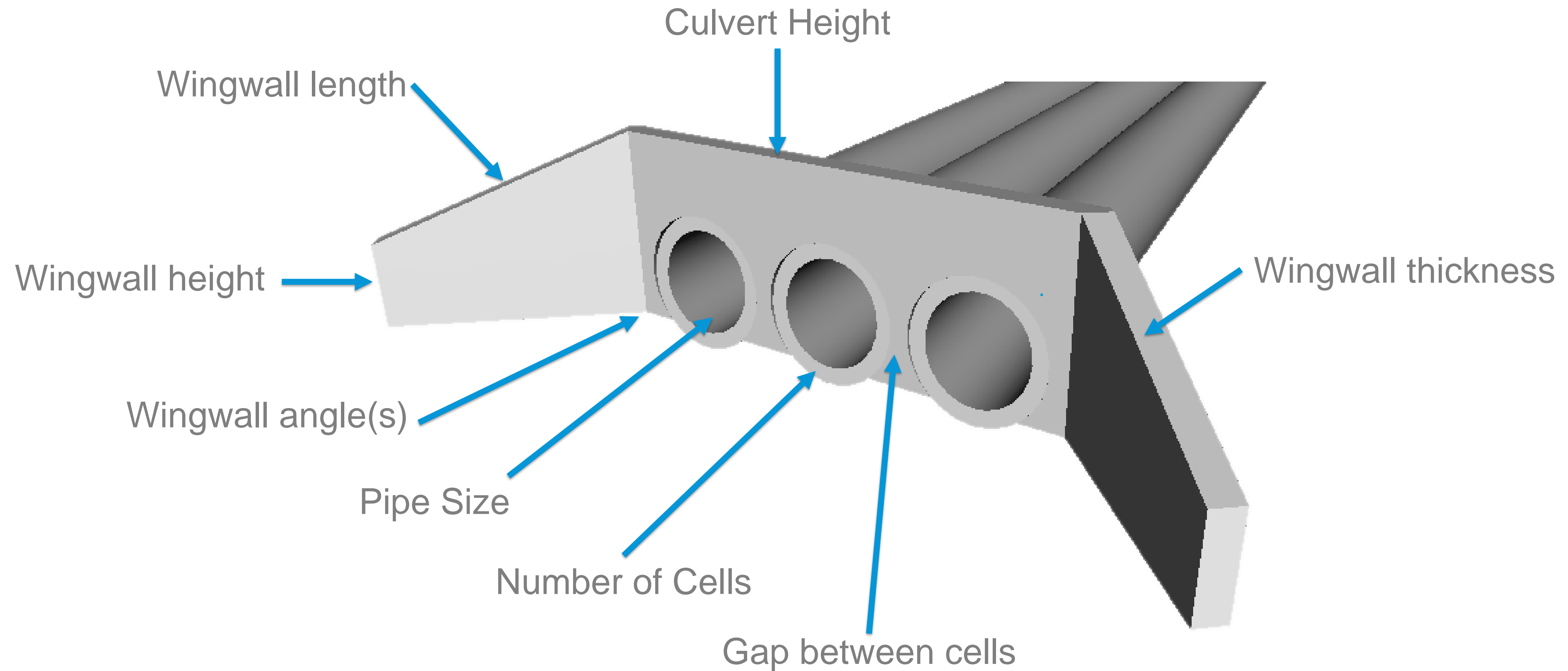
# Dynamo | Culvert Setout Parameters

- Alignment String
- Station
- Skew





# Dynamo | Culvert and Headwall Parameters



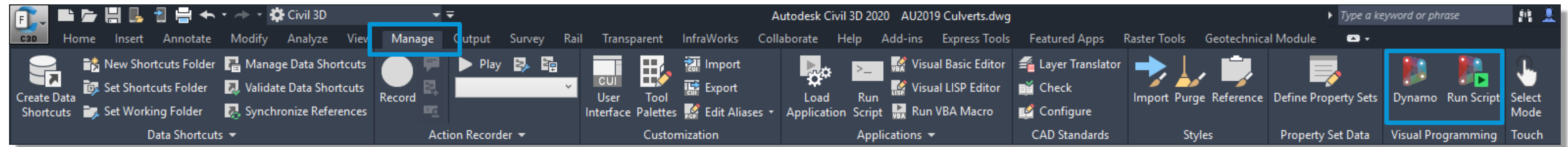
# What is Dynamo?

- Open-source software platform
- Application for creating visual scripts
- Geometry creation
- Workflow automation

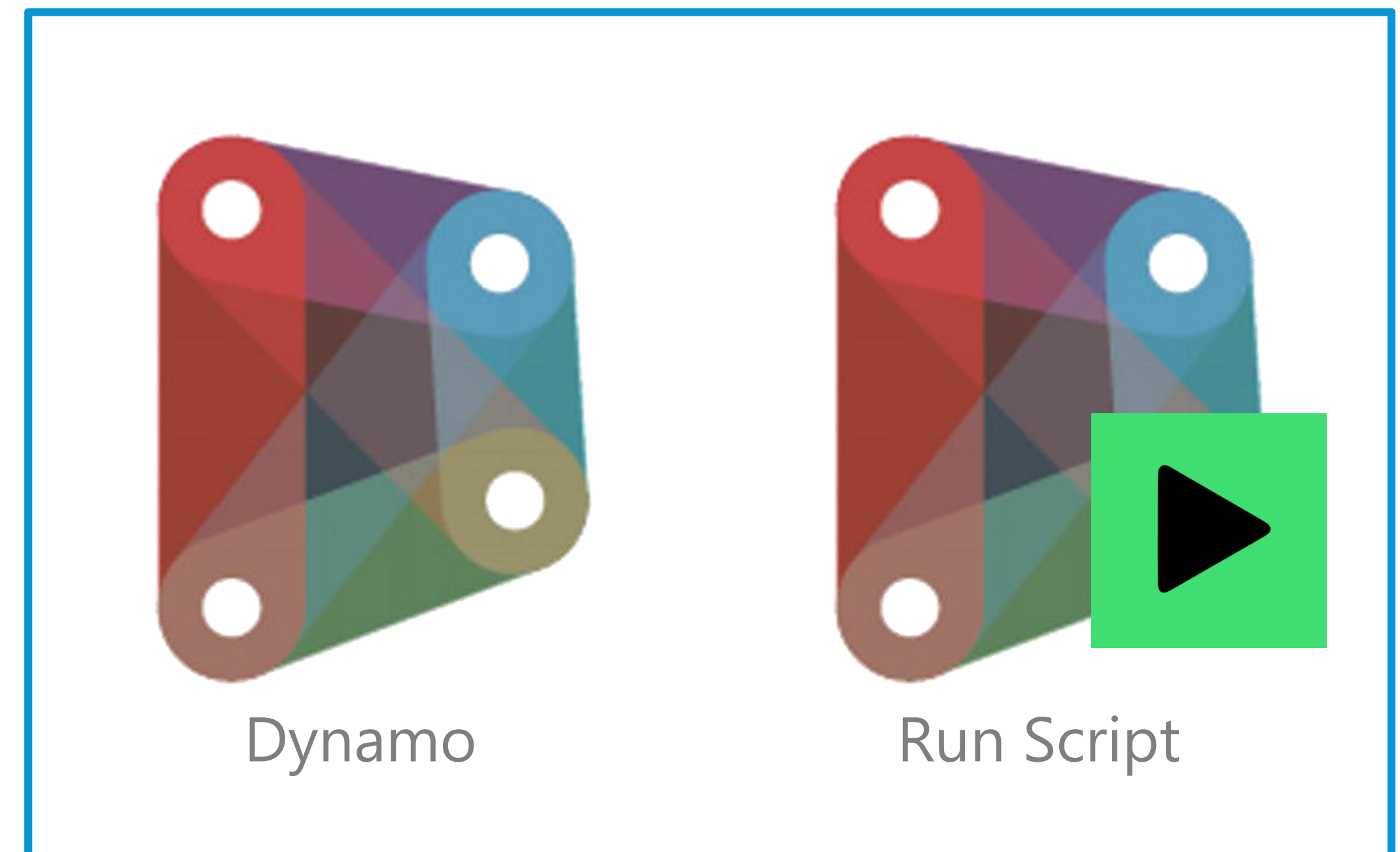




# How to load Dynamo in Civil 3D

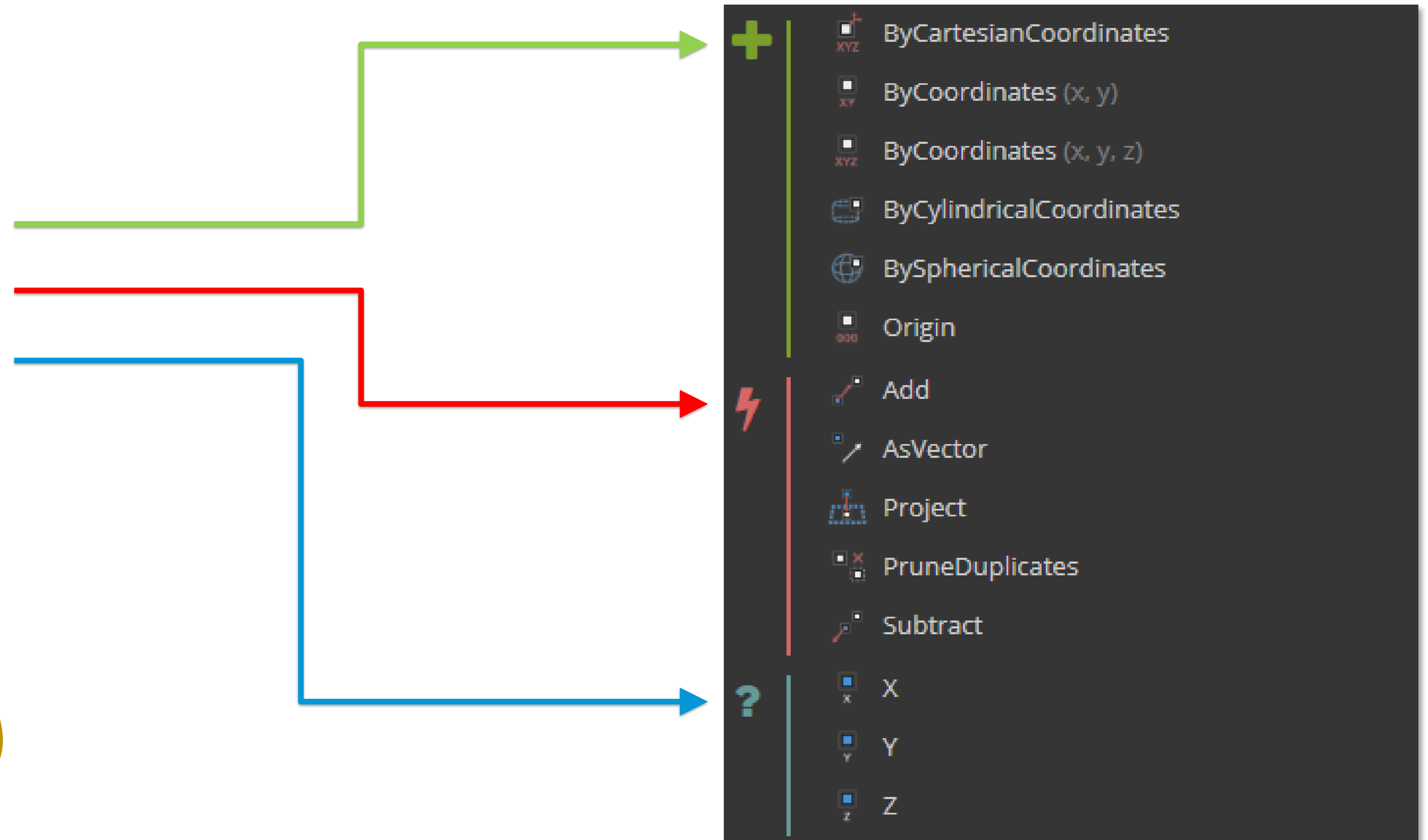


- Manage Tab
  - Visual Programming
    - Dynamo
    - Run Script



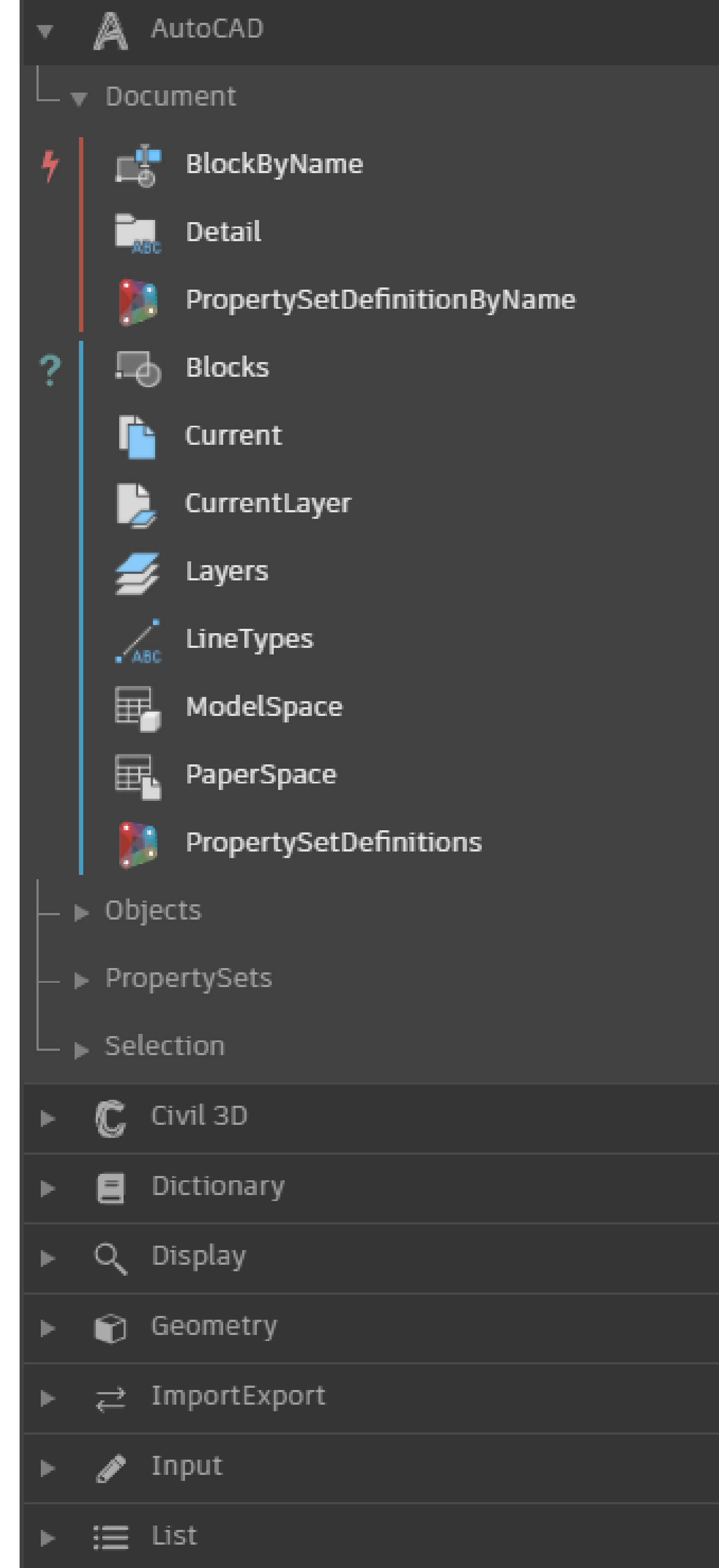
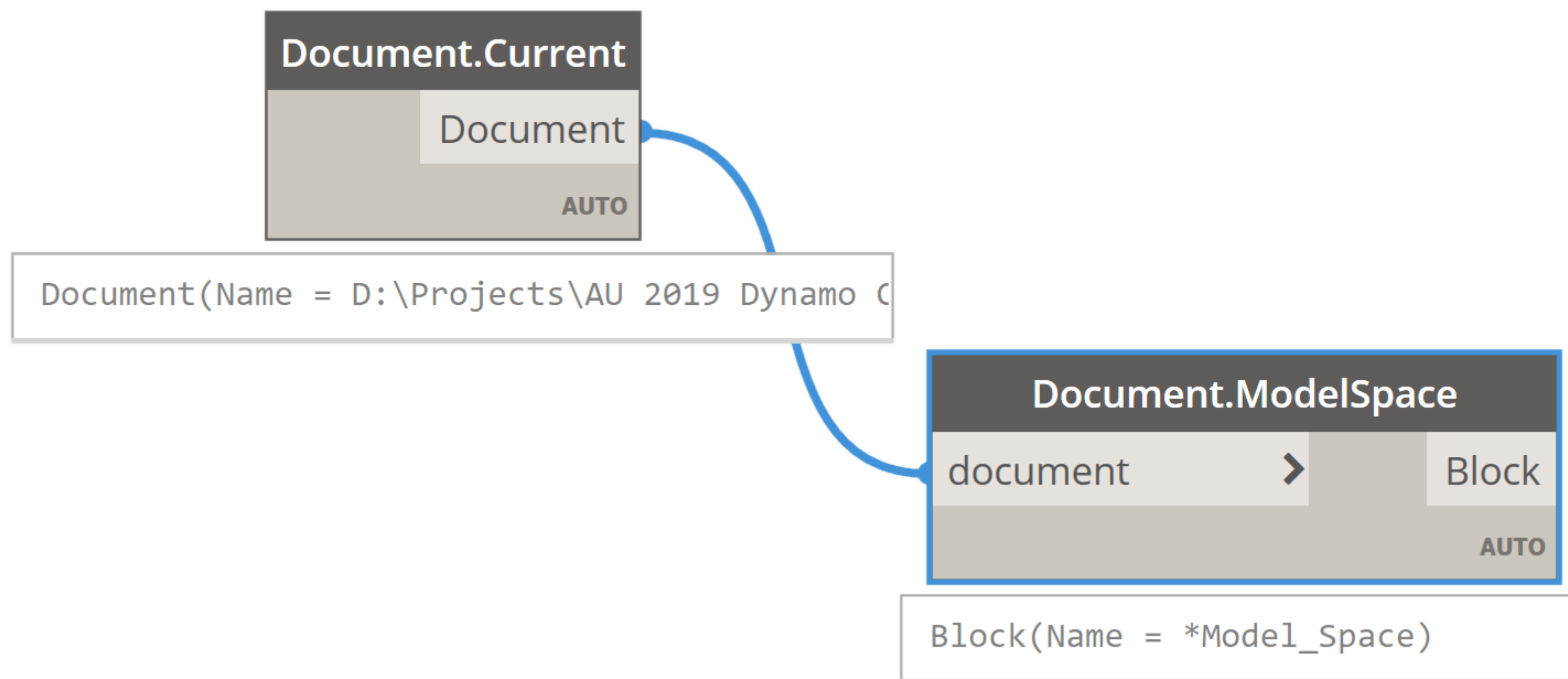
# Nodes | General

- 3 Node Types:
  - Create (Constructor)
  - Action (Method)
  - Query (Property)
- Scripting integration
  - Design Script
    - Code Blocks
  - Python



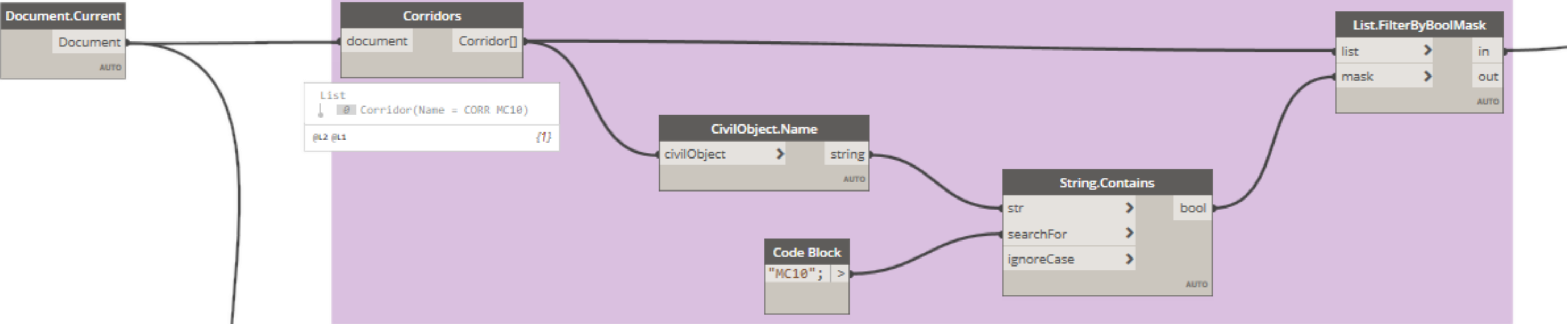


# Nodes | AutoCAD

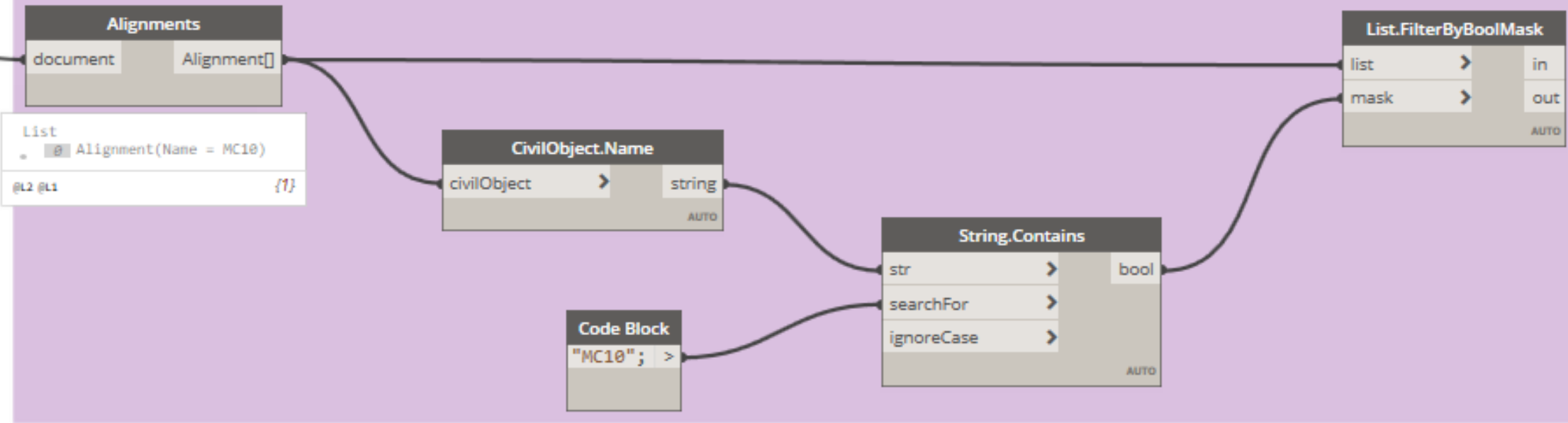


# Nodes | Civil 3D Selection

## Get Corridor



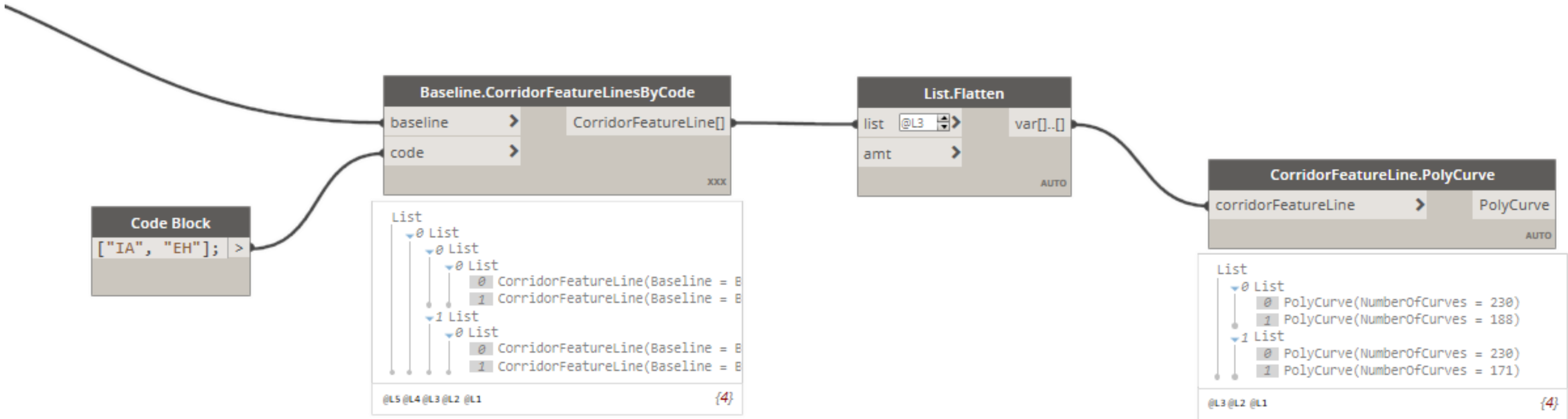
## Get Alignment



- ▶ A AutoCAD
- ▼ C Civil 3D
  - ▶ CivilObjects
  - ▼ Selection
    - ⚡ AlignmentByName
    - [ABC] Alignments
    - 📍 CogoPointGroupName
    - 📍 CogoPointGroups
    - 📍 CorridorByName
    - 📍 Corridors
    - 📍 SurfaceByName
    - 📍 Surfaces
- ▶ 📖 Dictionary
- ▶ 🔍 Display
- ▶ 📦 Geometry
- ▶ ↔ ImportExport
- ▶ ✎ Input
- ▶ ☰ List
- ▶  $x^2$  Math
- ▶ `</>` Script
- ▶ Ab String



# Nodes | Civil 3D CivilObjects



► CogoPoints

▼ Corridor

⚡  BaselineByName

 Codes

 GetSolids

 Rebuild

 SurfaceByName

?  Baselines

 Surfaces

► AppliedSubassembly

► Baseline

► BaselineRegion

▼ CorridorFeatureLine

⚡  CoordinateSystemByStation

 OffsetElevationByStation

?  Code

 EndStation

 Points

 PolyCurve

 Side

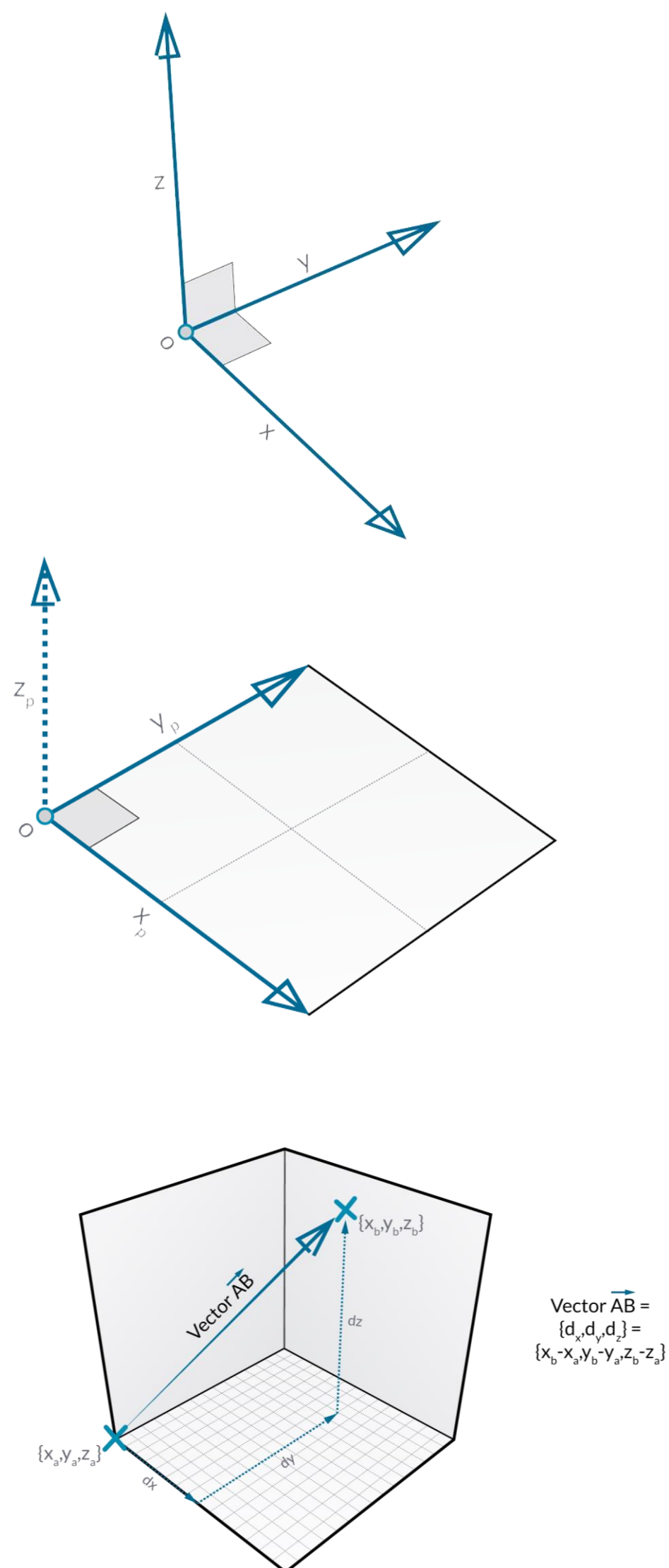
 StartStation

► SubassemblyParameter

► Surface

# Nodes | Abstract Geometry

- Geometry
  - Abstract
    - Coordinate Systems
    - Plane
    - Vector



► 🔍 Display

▼ 📦 Geometry

└─ ▼ Abstract

└─ ► BoundingBox

└─ ► CoordinateSystem

└─ ► Edge

└─ ► Face

└─ ► Plane

└─ ► Topology

└─ ► Vector

└─ ► Vertex

└─ ► Curves

└─ ► Meshes

└─ ► Modifiers

└─ ► Points

└─ ► Solids

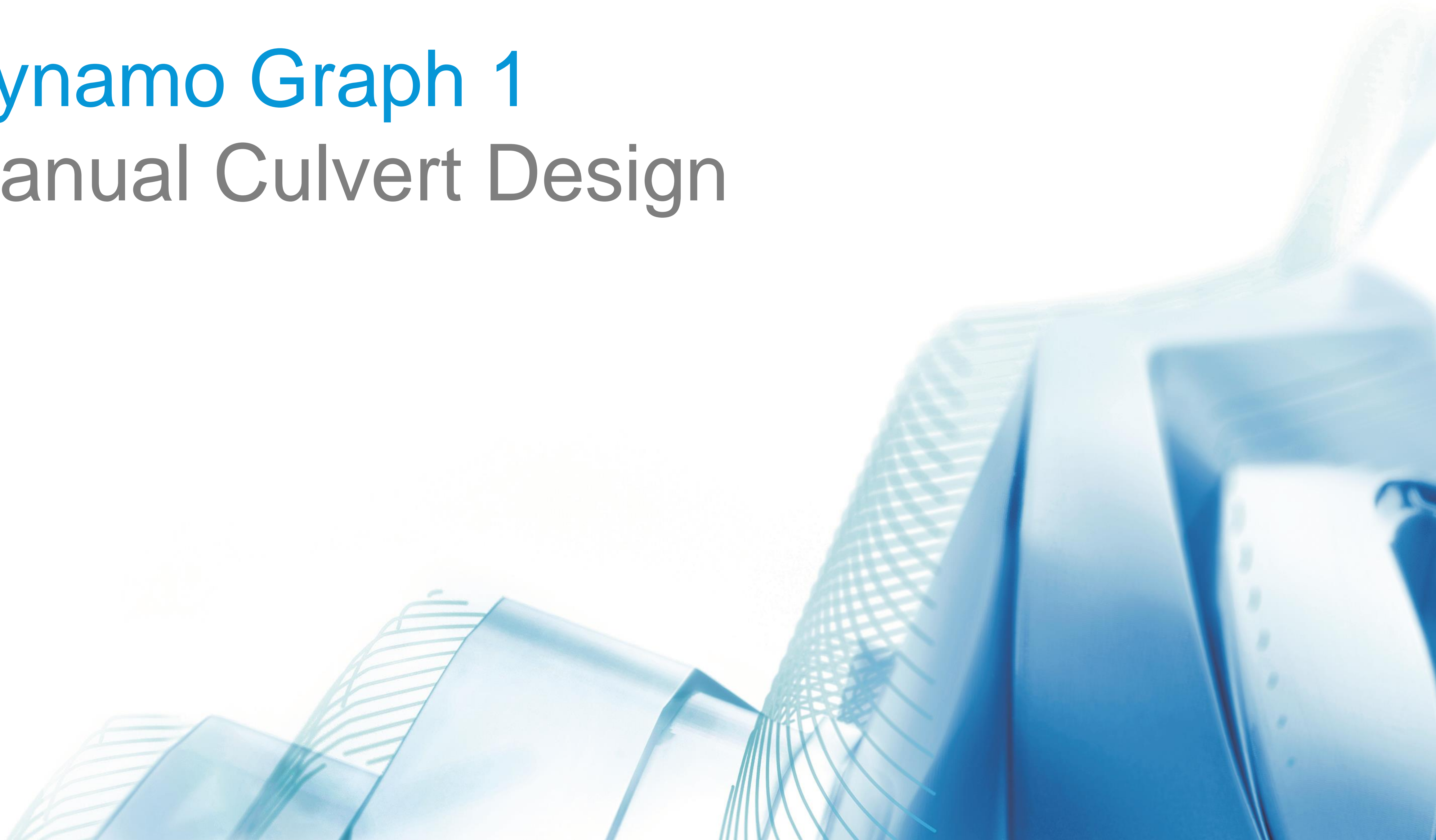
└─ ► Surfaces

└─ ► Tessellation

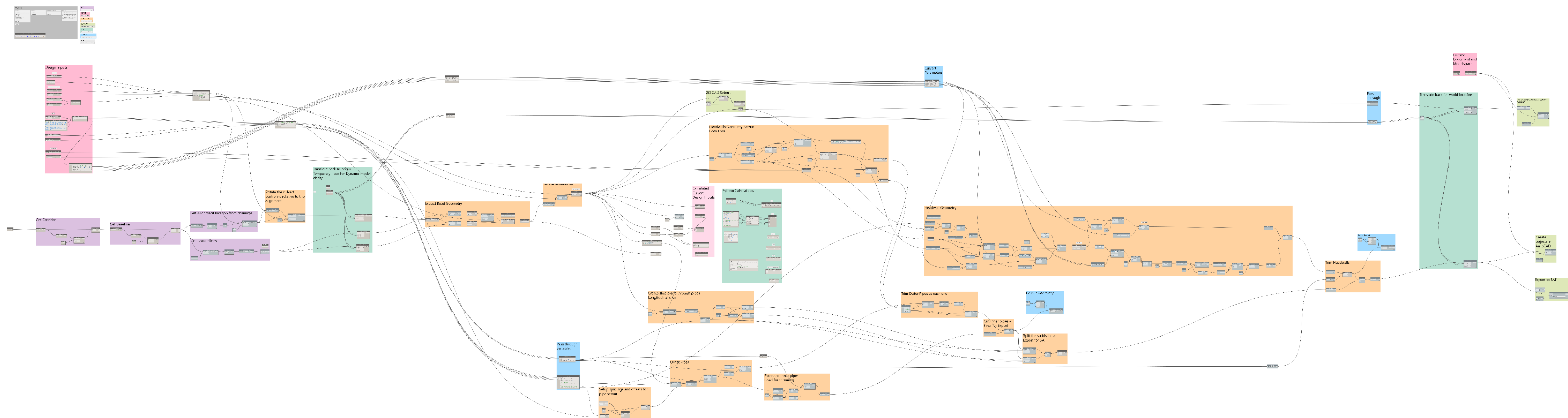


# Dynamo Graph 1

## Manual Culvert Design



# Dynamo Graph 1 | Manual Culvert Design



# Graph Setup | Template

- Graph Information
  - Version
  - Author
- Instructions
- Known Issues
- External Packages

## NOTES

<b>GRAPH INFO</b>  Copyright 2019 Autodesk, Inc. All rights reserved. Company: <company> Office: <office>  Version: 1.0.0 Author: <author> andrew.milford@autodesk.com  Keywords: [KEYWORDS]  Tested on: Dynamo : 2.3.0 Civil 3D : 2020	<b>INSTRUCTIONS</b>  <DESCRIPTIONS>  <INSTRUCTIONS>	<b>KNOWN ISSUES AND LIMITATIONS</b>  <ISSUES>  <LIMITATIONS>	<b>GUIDELINES</b>  Add Notes and Comments to the graph. Use Node Groups and the Standard Color Coding. Rename Nodes: <OriginalName>   <Description>. Write Input and Output Notes for Python Scripts. Prefer repeatable simple node structures.
---	---	--	---

RATING: 5

**Useful Links | CTRL + Click the link below**  

```
// Dynamo Primer  
"https://primer.dynamobim.org/index.html";  
"http://primer.dynamobim.org/13_Best-Practice/13-1_Introduction.html";
```

## GET

Get parameter values from Civil 3D objects

## INPUT

Data input and preparation

## FUNCTION

Data manipulation on Dynamo objects

## OUTPUT

Object creation in AutoCAD Civil 3D, Final output

## SET

Set parameter values of Civil 3D objects

## DEBUG

Nodes used to debug the graph logic

## WIP

Notes not considered part of the final package



# Graph Layout

Inputs

Functions

Outputs

Design Inputs

Culvert Setout

Number Slider | Skew

75

Code Block

chainage = 219.0;

Culvert Horizontal & Vertical Adjustments

Number Slider | Extend Start US

1

Number Slider | Extend End OS

1

Number Slider | Adjust Start Elevation 1

-0.5

Code Block

elev1 [elev1, elev2];

Number Slider | Extend End Elevation 2

0

Extract Road Geometry

CoordinateSystem.ZXPlane

coordinateSystem

Plane

Geometry.Intersect

geometry

other

Geometry.Intersect

geometry

other

Geometry.Translate | Adjust Vertical

geometry

xTranslation

yTranslation

zTranslation

Code Block

// Check the direction of flow  
// (left/right) or (right/left)

Python Script | Reverse Culvert Direction

IN[0]

IN[1]

OUT

List.Flatten | Daylight

list

var[0..1]

List.Flatten | Verge

list

var[0..1]

Code Block

a daylight = a[0];  
verge = a[1];

Create objects in AutoCAD

Object.ByGeometry

geometry

layer

block

Code Block

"\_Headwall";

ExportToSAT

geometry

filePath

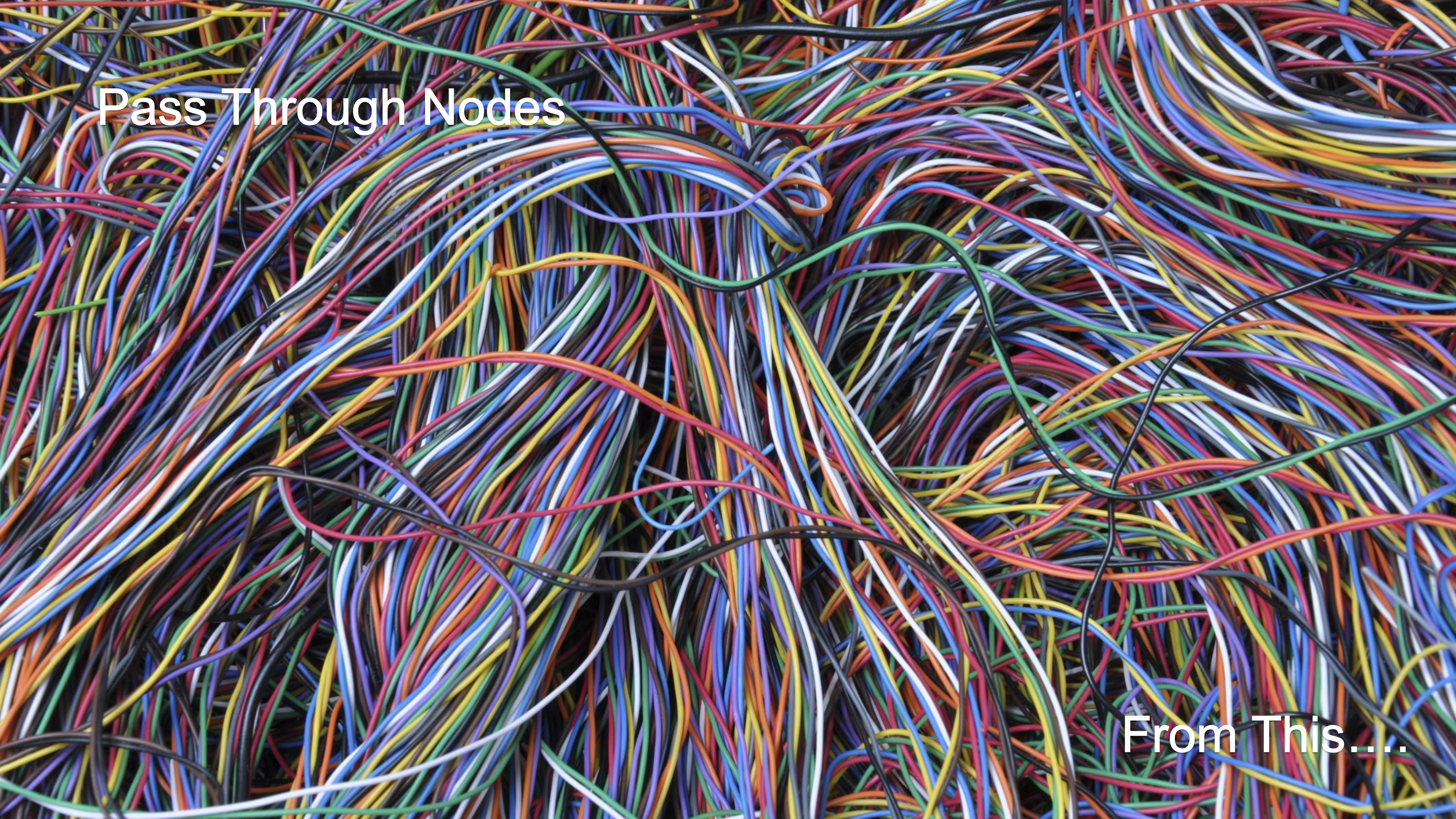
Units: Meters

File Path

Browse...

Output\Culvert 01.sat





Pass Through Nodes

From This....



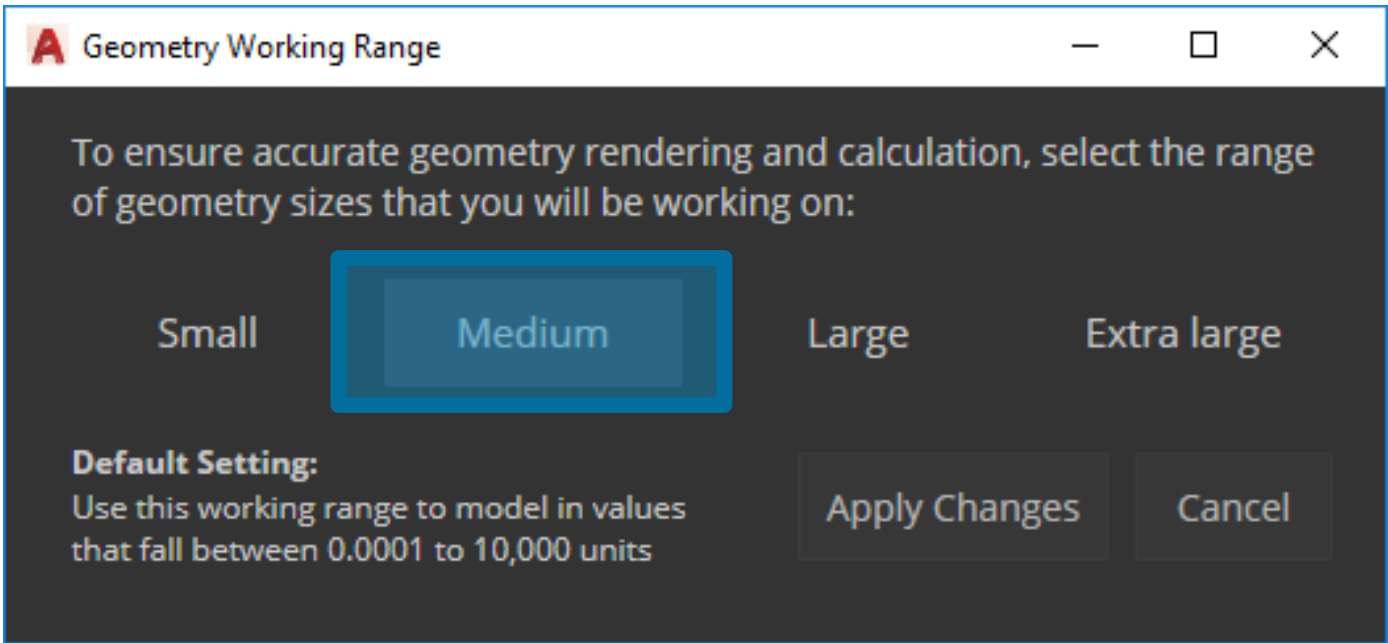
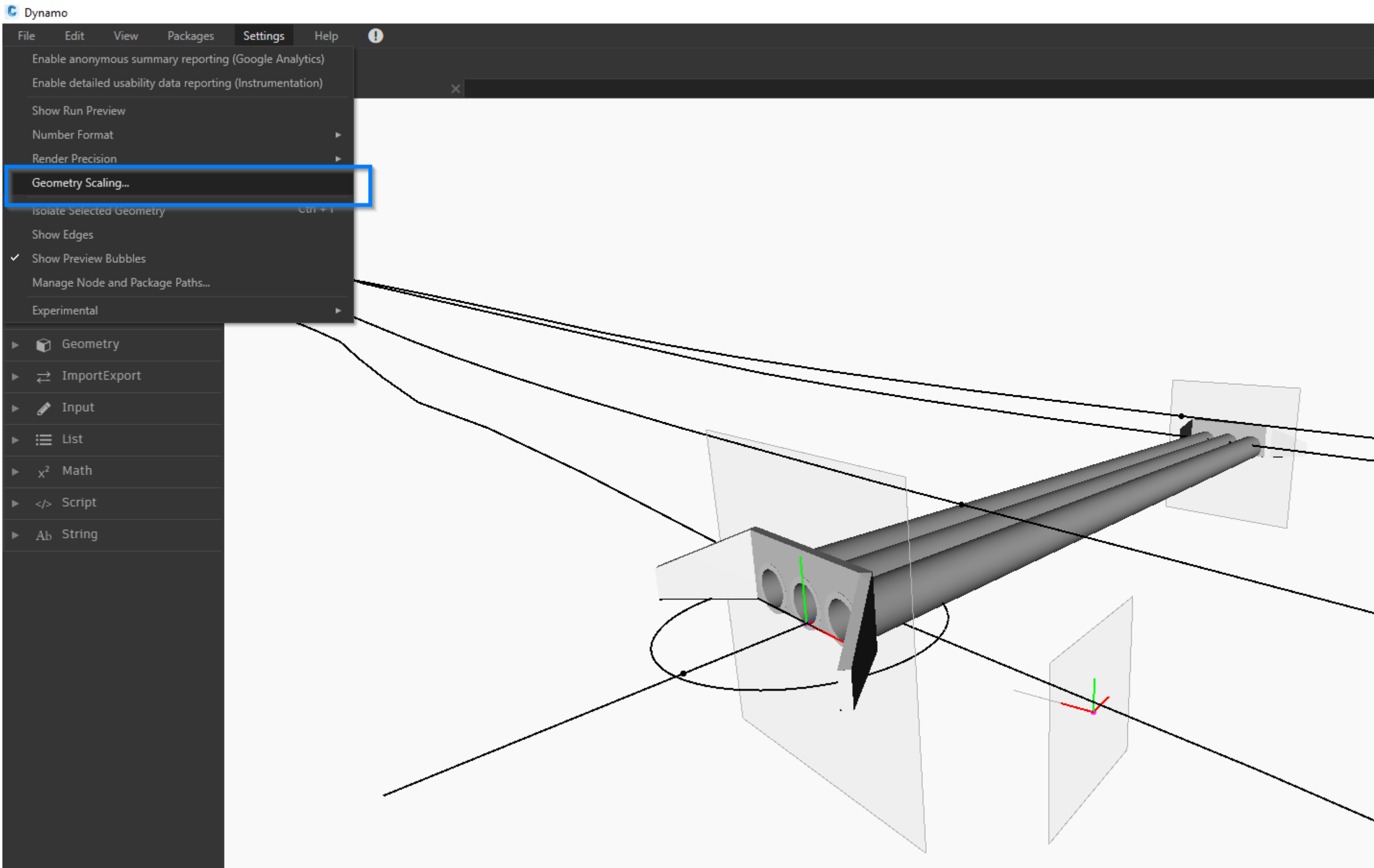


Pass Through Nodes

To This....



# Dynamo Warnings | Geometry Scaling



Warning: Your inputs lie outside of the allowable modeling range, consider choosing the Extra Large setting with a modeling range between 1 and 100,000,000 from the "Settings => Geometry Working Range" dialog

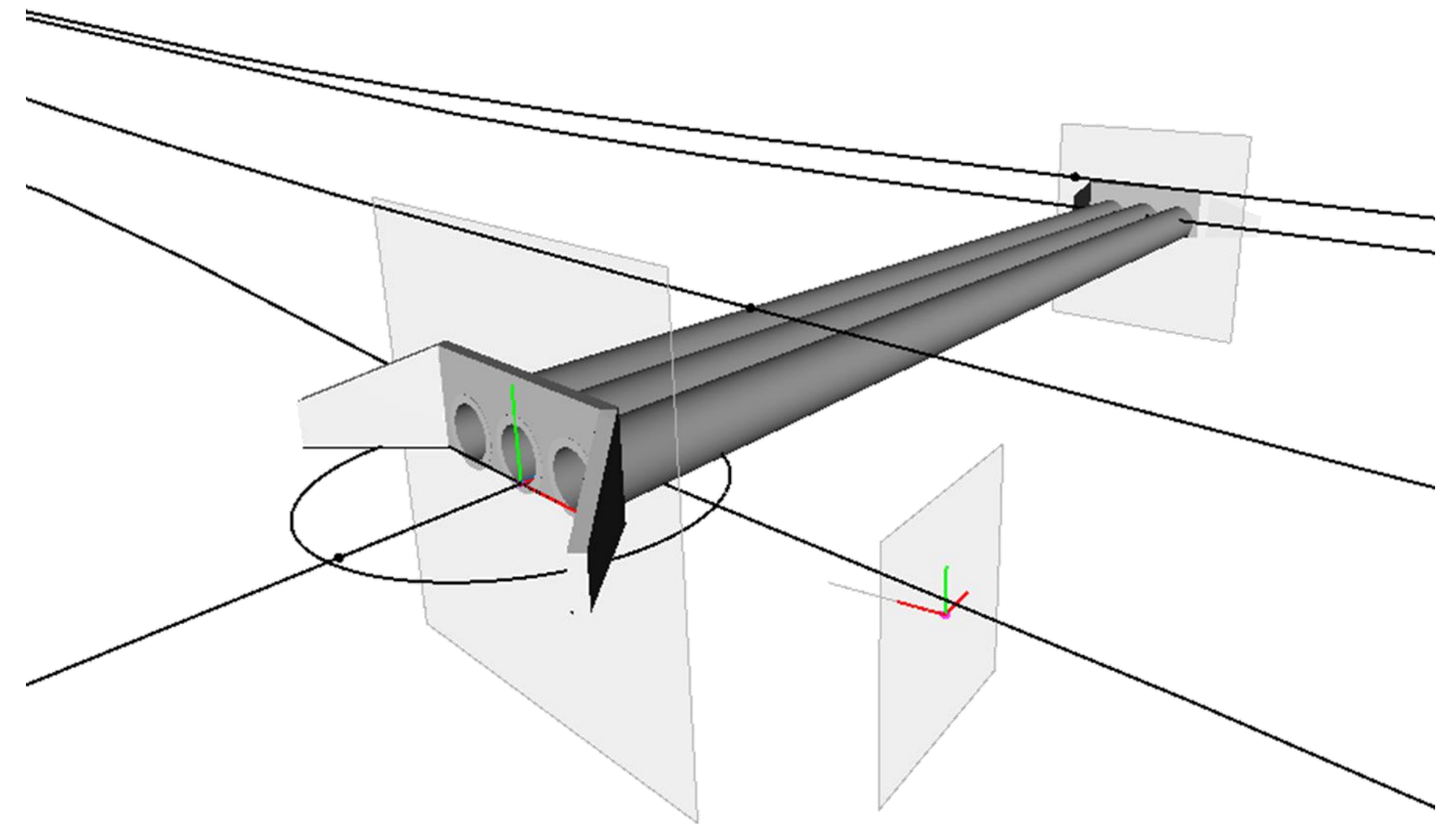
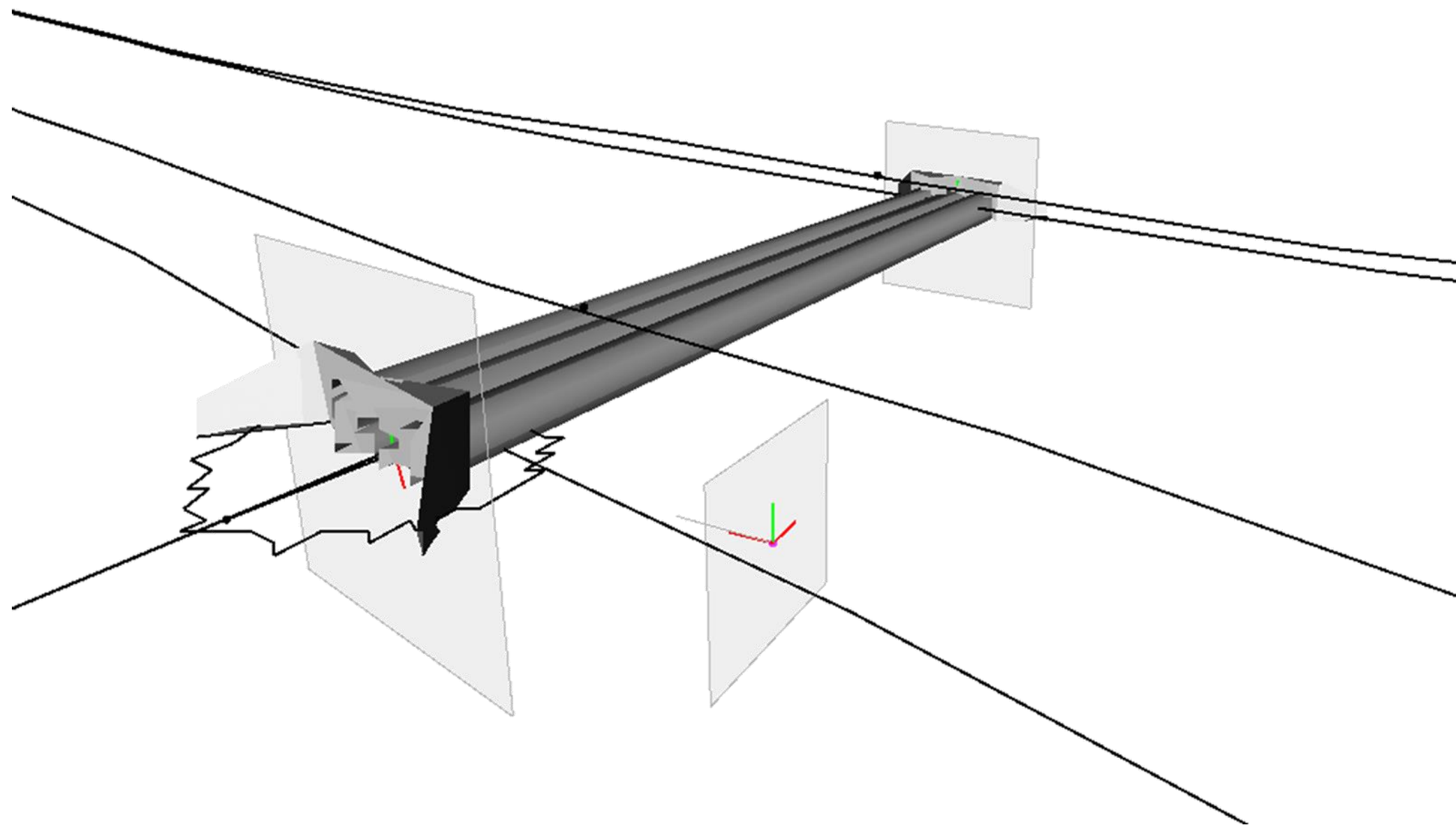
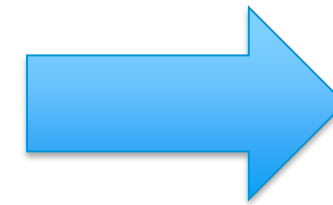
Alignment.CoordinateSystemByStationOffset		
alignment	>	CoordinateSystem
station	>	
offset	>	
AUTO		

Ignore this warning

# Dynamo | Transform to Origin

From this

To this



Don't forget to translate back to the world!

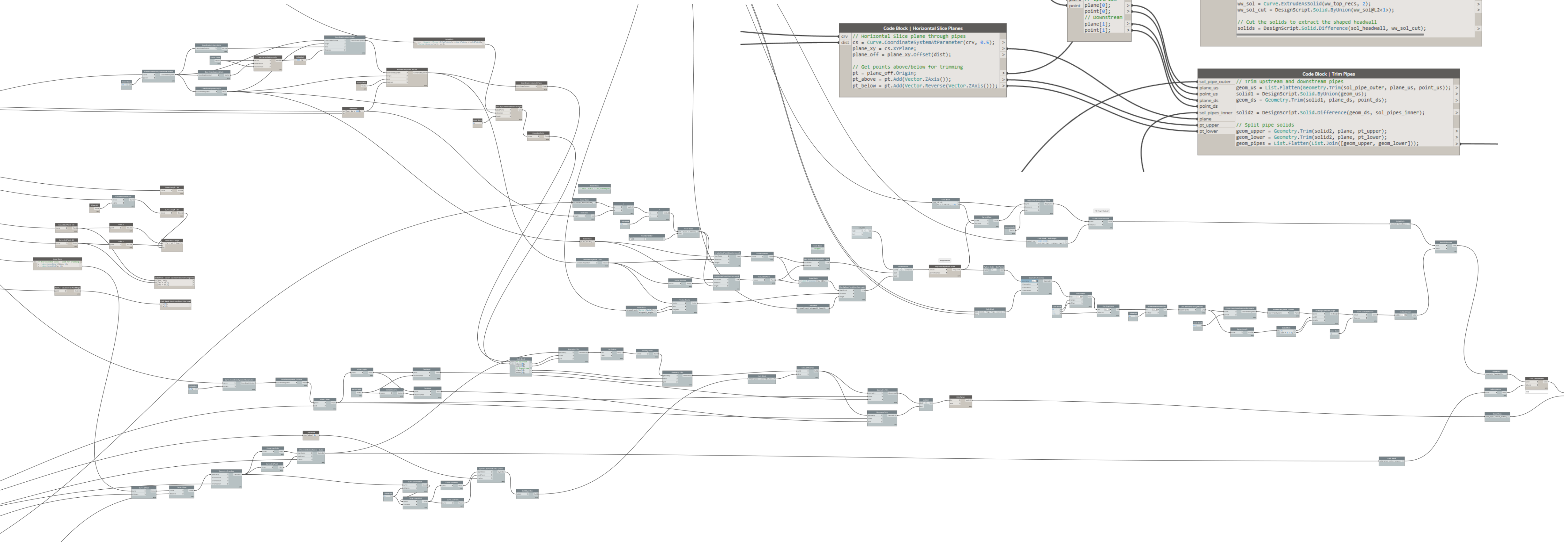
# Dynamo Graph 2

## Manual Culvert with DesignScript





# DesignScript



```
Code Block | Headwalls Geometry Setout - Both Ends
crv // Coordinate at each end of setout line
us  //
ds  //
width // CS Properties
pt_orig = CoordinatesSystem.Origin(cs);
vec_x = CoordinatesSystem.XAxis(cs);
vec_z = CoordinatesSystem.ZAxis(cs);

rot = 180 - Vector.AngleAboutAxis(Vector.ZAxis(), vec_z, Vector.ZAxis());

// Headwall Skew Rotation
rot_hw = [us, ds + 180];

// Rotate Coordinate Systems
cs1 = CoordinatesSystem.Rotate(cs, pt_orig, vec_x, rot);
cs2 = CoordinatesSystem.Rotate(cs1, pt_orig, Vector.ZAxis(), rot_hw);

// Create geometry trim points outside the culvert
vec = Geometry.Vector.ByTwoPoints(crv.StartPoint, crv.EndPoint);
vec_lst = [Vector.Reverse(vec), vec];
lines = DesignScript.Line.ByStartPointDirectionLength(pt_orig, vec_lst, 2.0);

// Outputs
pts_end = lines.EndPoint;
plane_xz = cs2.ZXPlane;
width;
```

```
Code Block | Horizontal Slice Planes
crv // Horizontal Slice plane through pipes
dist cs = Curve.CoordinatesSystemAtParameter(crv, 0.5);
plane_xy = cs.XYPlane;
plane_off = plane_xy.Offset(dist);

// Get points above/below for trimming
pt = plane_off.Origin;
pt_above = pt.Add(Vector.ZAxis());
pt_below = pt.Add(Vector.Reverse(Vector.ZAxis()));
```

```
Code Block
plane // upstream
point plane[0];
point point[0];
// Downstream
plane[1];
point[1];
```

```
Code Block | Create Headwall & Wingwall Geometry Linework
totalWidth // Headwall Geometry
rot_hw hw_len = (totalWidth / Math.Cos(rot_hw)) / 2.0;
pt_centre pt_centre;
cs vec_x = cs.XAxis;

wingwall_angle // Headwall Half baselines
pts_base pt1 = DesignScript.Line.ByStartPointDirectionLength(pt_centre, vec_x, hw_len).EndPoint;
side pt2 = DesignScript.Line.ByStartPointDirectionLength(pt_centre, Vector.Reverse(vec_x), hw_len).EndPoint;
top hw_baseline = DesignScript.Line.ByStartPointEndPoint(pt1, pt2);
crv hw_pts = List.Transpose([pt1, pt2]);
wall_thk // Vector Rotations
culvert_hgt vec_wv = Vector.Rotate(vec_x, Vector.ZAxis(), wingwall_angle);

// Wingwall Linework
ww_lines = DesignScript.Line.ByStartPointDirectionLength(hw_pts, vec_wv, wingwall_angle);

// Top of Wingwall geometry
pts_top = Geometry.Translate(pts_base@L2<1>, 0, 0, [side, top, top, side]);

// Split the lists into pairs
sub_lists = List.Sublists(pts_top@L2<1>, (0..1), 1);
drop_lists = List.DropItems(sub_lists@L3<1>, 1);
lists = List.RemoveItemAtIndex(drop_lists@L3<1>, 1);

// Full Height Headwall
crv_headwall = Curve.Offset(crv, wall_thk / 2.0);
crv_headwall = PolyCurve.ByThickeningCurve(crv.Offset, wall_thk, Vector.ZAxis());
sol_headwall = Curve.ExtrudeAsSolid(crv_headwall, [-culvert_hgt, culvert_hgt]);

// Wingwall Angled Linework and Solids for cutting
ww_top_lines = DesignScript.Geometry.Line.ByBestFitThroughPoints(lists);
ww_top_cs = Curve.CoordinatesSystemAtParameter(ww_top_lines, 0.5);
ww_top_plane = CoordinatesSystem.XYPlane(ww_top_cs);
ww_top_recs = Rectangle.ByWidthLength(ww_top_plane, 1, ww_top_lines);
ww_sol = Curve.ExtrudeAsSolid(ww_top_recs, 2);
ww_sol_cut = DesignScript.Solid.ByUnion(ww_sol@L2<1>);

// Cut the solids to extract the shaped headwall
solids = DesignScript.Solid.Difference(sol_headwall, ww_sol_cut);
```

```
Code Block | Trim Pipes
sol_pipe_outer // Trim upstream and downstream pipes
plane_us geom_us = List.Flatten(Geometry.Trim(sol_pipe_outer, plane_us, point_us));
point_us solid1 = DesignScript.Solid.ByUnion(geom_us);
plane_ds geom_ds = Geometry.Trim(solid1, plane_ds, point_ds);
point_ds solid2 = DesignScript.Solid.Difference(geom_ds, sol_pipes_inner);
sol_pipes_inner // Split pipe solids
plane geom_upper = Geometry.Trim(solid2, plane, pt_upper);
pt_upper geom_lower = Geometry.Trim(solid2, plane, pt_lower);
pt_lower geom_pipes = List.Flatten(List.Join([geom_upper, geom_lower]));
```

# Node To Code

- Code difficult to read
- Begin in DesignScript
- Comment your work

Search

Lacing ▶  
Hide all geometry preview  
Align Selection ▶  
Create Custom Node  
**Node to Code**  
Create Group  
Copy  
Switch to Geometry View  
Pan  
Fit to Screen

```
Code Block
point1 ptCen = point1;
coordinate1 vector2 = CoordinateSystem.XAxis(coordinate1);
t7 totalWidth = t7;
t8 num2 = Math.Cos(t8);
t12 t9 = totalWidth / num2;
t13 t10 = t9 / 2;
t11 = 0;
dist = t10;
extra = t11;
t2 = dist + extra;
line1 = Line.ByStartPointDirectionLength(ptCen, vector2<1L>, t2<1L>);
point2 = Curve.EndPoint(line1);
vector3 = Vector.Reverse(vector2);
line2 = Line.ByStartPointDirectionLength(ptCen, vector3<1L>, t2<1L>);
point3 = Curve.EndPoint(line2);
line3 = Line.ByStartPointEndPoint(point2, point3);
a = point2;
b = point3;
t4 = List.Transpose([a, b]);
wingwall_angle = t12;
vector1 = Vector.ZAxis();
vector4 = Vector.Rotate(vector2, vector1, wingwall_angle);
wingwall_length = t13;
line4 = Line.ByStartPointDirectionLength(t4, vector4, wingwall_length);
```

```
Code Block | Wingwall Geometry Solids
pts_base // Top of Wingwall geometry
side pts_top = Geometry.Translate(pts_base@@L2<1>, 0, 0, [side, top, top, side]);
top
crv // Split the lists into pairs
wall_thk sub_lsts = List.Sublists(pts_top@L2<1>, (0..1), 1);
culvert_hgt drop_lsts = List.DropItems(sub_lsts@L3<1>, -1);
lst = List.RemoveItemAtIndex(drop_lsts@L3<1>, 1);

// Full Height Headwall
crv_offset = Curve.Offset(crv, wall_thk / 2.0);
crv_headwall = PolyCurve.ByThickeningCurve(crv_offset, wall_thk, Vector.ZAxis());
sol_headwall = Curve.ExtrudeAsSolid(crv_headwall, [-culvert_hgt, culvert_hgt]);

// Wingwall Angled Linework and Solids for cutting
ww_top_lines = DesignScript.Geometry.Line.ByBestFitThroughPoints(lst);
ww_top_cs = Curve.CoordinateSystemAtParameter(ww_top_lines, 0.5);
ww_top_plane = CoordinateSystem.XYPlane(ww_top_cs);
ww_top_lens = Curve.Length(ww_top_lines) * 2.0 + 1;
ww_top_recs = Rectangle.ByWidthLength(ww_top_plane, 1, ww_top_lens);
ww_sol = Curve.ExtrudeAsSolid(ww_top_recs, 2);
ww_sol_cut = DesignScript.Solid.ByUnion(ww_sol@L2<1>);

// Cut the solids to extract the shaped headwall
solids = DesignScript.Solid.Difference(sol_headwall, ww_sol_cut);
```



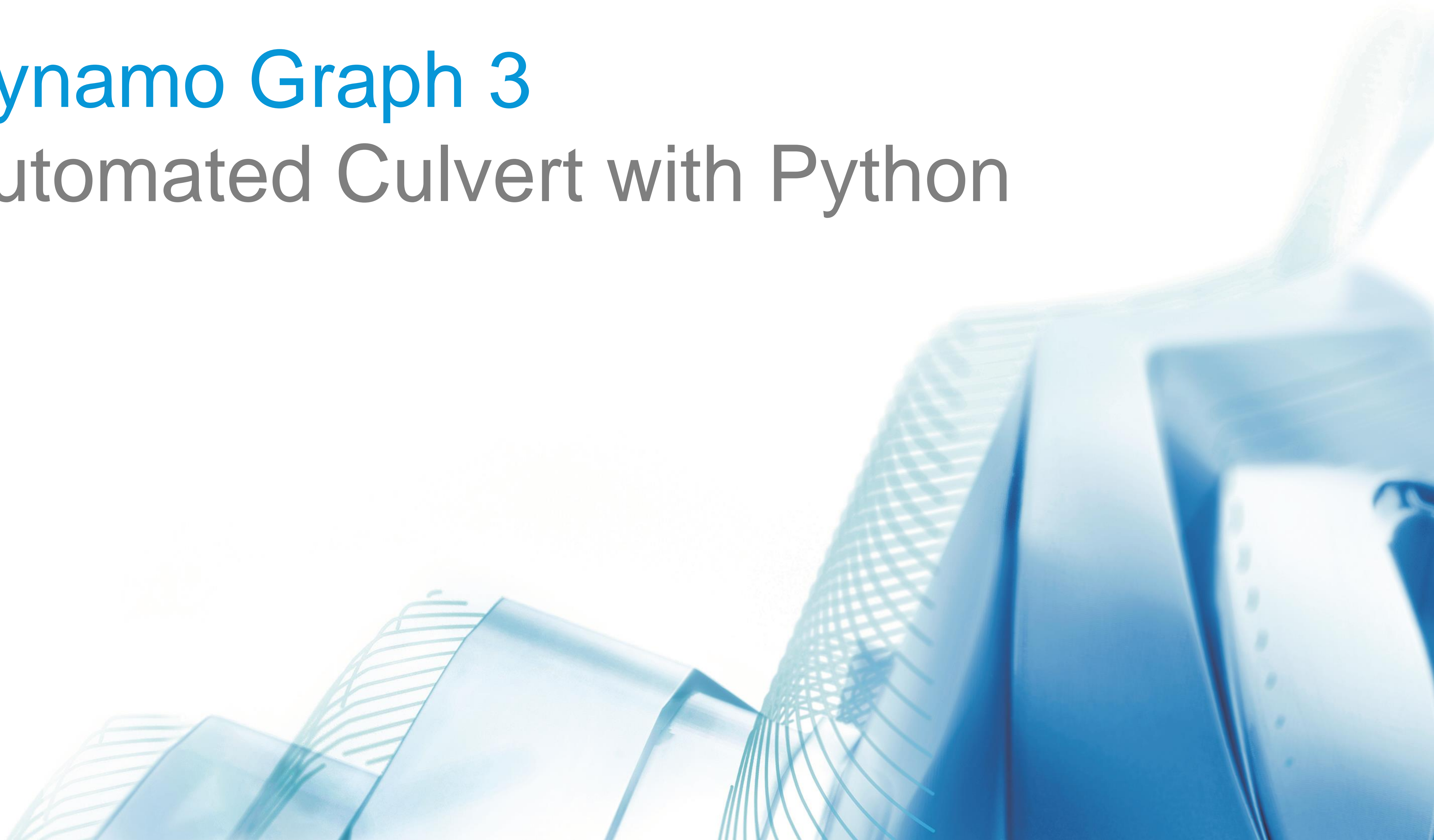


Nodes vs Code Blocks



# Dynamo Graph 3

## Automated Culvert with Python



# Python

- Legible Scripting Language
- Object-Oriented
- Open Source
- Monty Python







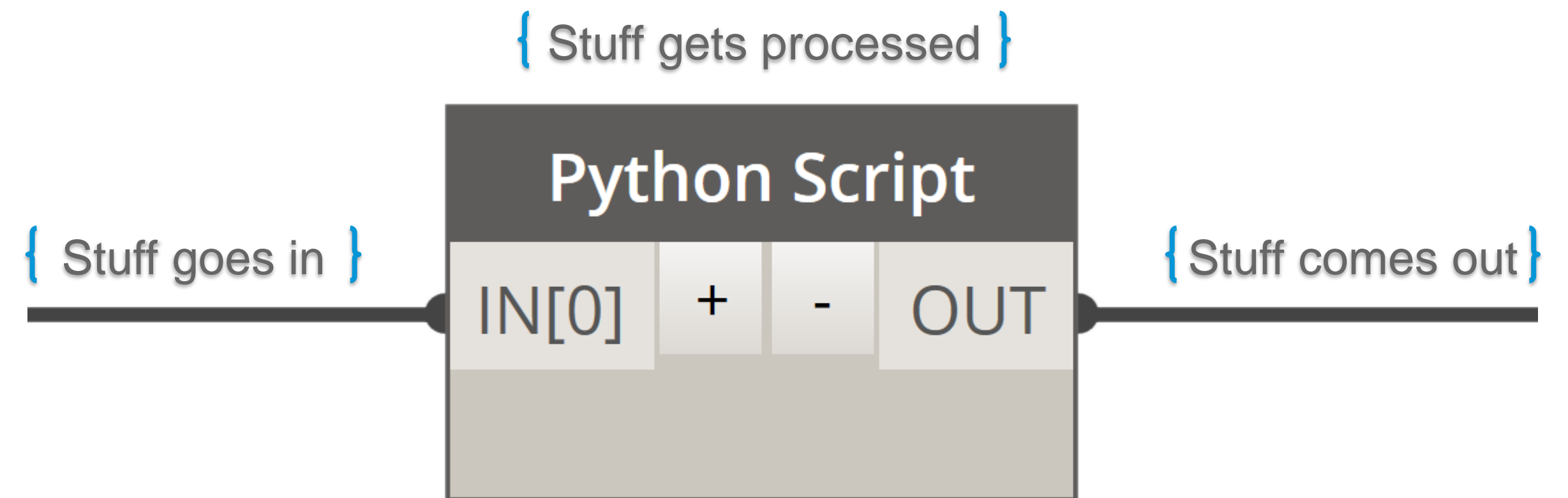
Microsoft®  
**.NET**

## Iron Python

C# implementation of Python language

Microsoft .Net Framework

Common Language Runtime (CLR)

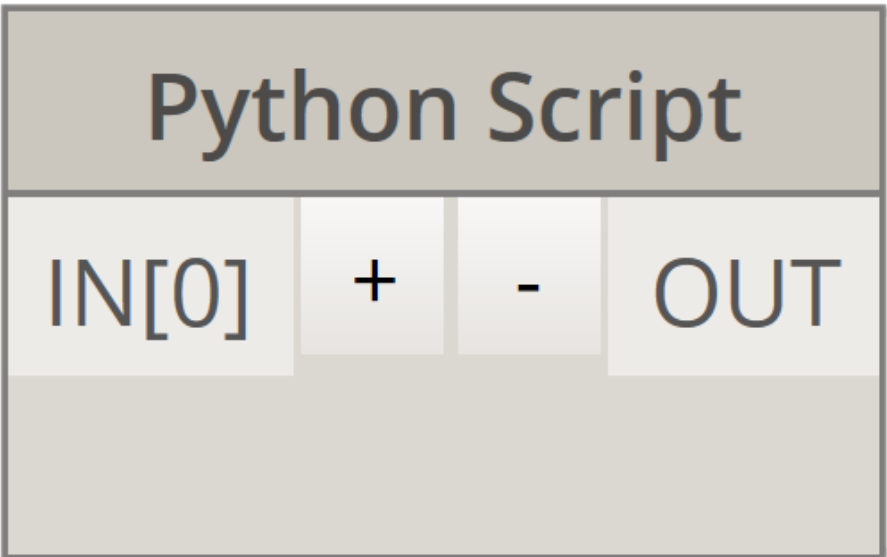


## Python Node

Simplified Integrated Developer Environment (IDE)



# Dynamo Python Node

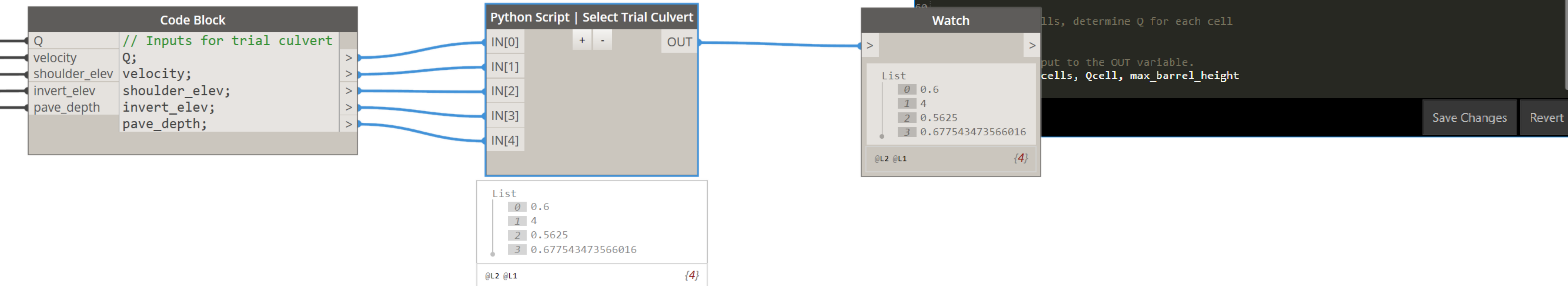


```
Python Script
1 # Load the Python Standard and DesignScript Libraries
2 import sys
3 import clr
4
5 # Add Assemblies for AutoCAD and Civil3D
6 clr.AddReference('AcMgd')
7 clr.AddReference('AcCoreMgd')
8 clr.AddReference('AcDbMgd')
9 clr.AddReference('AecBaseMgd')
10 clr.AddReference('AecPropDataMgd')
11 clr.AddReference('AeccDbMgd')
12
13 # Import references from AutoCAD
14 from Autodesk.AutoCAD.Runtime import *
15 from Autodesk.AutoCAD.ApplicationServices import *
16 from Autodesk.AutoCAD.EditorInput import *
17 from Autodesk.AutoCAD.DatabaseServices import *
18 from Autodesk.AutoCAD.Geometry import *
19
20 # Import references from Civil3D
21 from Autodesk.Civil.ApplicationServices import *
22 from Autodesk.Civil.DatabaseServices import *
23
24 # The inputs to this node will be stored as a list in the IN variables.
25 dataEnteringNode = IN
26
27 adoc = Application.DocumentManager.MdiActiveDocument
28 editor = adoc.Editor
29
30 with adoc.LockDocument():
31     with adoc.Database as db:
32
33         with db.TransactionManager.StartTransaction() as t:
34             # Place your code below
35             #
36             #
37
38             # Commit before end transaction
39             t.Commit()
40             pass
41
42 # Assign your output to the OUT variable.
43 OUT = 0
44
45
46 Run Save Changes Revert
```

- ▶ A AutoCAD
- ▶ C Civil 3D
- ▶ Dictionary
- ▶ Display
- ▶ Geometry
- ▶ ImportExport
- ▶ Input
- ▶ List
- ▶ Math
- ▼ Script
  - ▶ Control Flow
  - ▼ Editor
    - ⚡ [I] Code Block
    - Python Script**
    - Python Script From String
    - ▶ Evaluate
    - ▶ String

# Pipe Sizing with Python

- For loops
- While loops
- If Statements



# Headwater Depth

- While loops
- Goal Seek

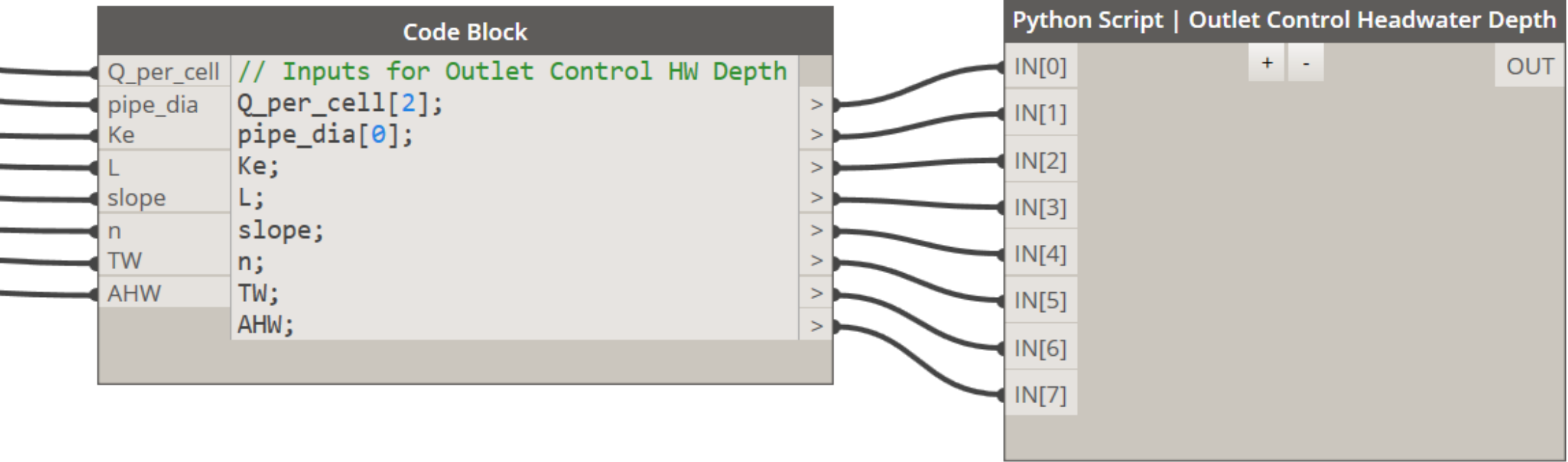


Flowchart Step 5 | Inputs

```
// IN[0] = Q;  
// IN[1] = Pipe Dia;  
// IN[2] = Ke;  
// IN[3] = Length of Pipe;  
// IN[4] = So;  
// IN[5] = n;  
// IN[6] = TW;  
// IN[7] = AHW;
```

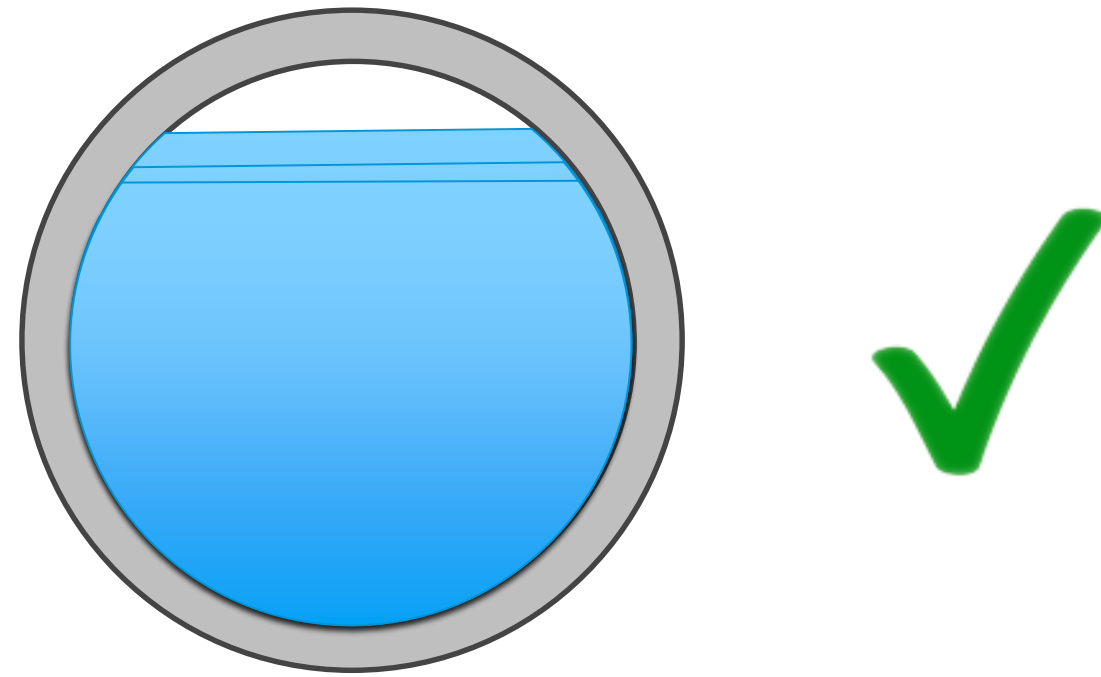
Flowchart Step 5 | Outputs

```
// OUT[0] = Outlet Headwater Depth;  
// OUT[1] = dc critical depth;  
// OUT[2] = (dc + D)/2;
```





# Headwater Depth



Flowchart Step 5 | Inputs

```
// IN[0] = Q;  
// IN[1] = Pipe Dia;  
// IN[2] = Ke;  
// IN[3] = Length of Pipe;  
// IN[4] = So;  
// IN[5] = n;  
// IN[6] = TW;  
// IN[7] = AHW;
```

Flowchart Step 5 | Outputs

```
// OUT[0] = Outlet Headwater Depth;  
// OUT[1] = dc critical depth;  
// OUT[2] = (dc + D)/2;
```

Code Block

Input	Code
Q_per_cell	// Inputs for Outlet Control HW Depth
pipe_dia	Q_per_cell[2];
Ke	pipe_dia[0];
L	Ke;
slope	L;
n	slope;
TW	n;
AHW	TW;
	AHW;

Python Script | Outlet Control Headwater Depth

IN[0]	+	-	OUT
IN[1]			
IN[2]			
IN[3]			
IN[4]			
IN[5]			
IN[6]			
IN[7]			

Python Script | Outlet Control Headwater Depth

```
1# Establish flow depth at outlet  
2# Setup a loop, starting with the half depth of pipe/culvert  
3# Maximum iterations = 15  
4i = 0  
5tolerance = 0.005  
6diff = 100  
7temp = []  
8  
9depth = radius  
10depth_half = depth # initialise half depth to begin  
11  
12while i < 15 or diff < tolerance:  
13    depth_from_centre = abs(radius - depth)  
14    top_surface_width = 2 * math.sqrt(radius ** 2 - abs(radius - depth) ** 2)  
15  
16    if depth > radius:  
17        central_angle = 360 - math.acos(depth_from_centre / radius) * (180 /  
18            math.pi) * 2.0  
19    else:  
20        central_angle = math.acos(depth_from_centre / radius) * (180 /  
21            math.pi) * 2.0  
22  
23    wetted_area = ((radius ** 2) / 2) * (((math.pi / 180) * central_angle) -  
24        \  
25            math.sin(math.radians(central_angle)))  
26    wetted_perimeter = 2.0 * math.pi * radius * (central_angle / 360)  
27    hydraulic_radius = wetted_area / wetted_perimeter  
28  
29    # setup check conditions -  
30    # Si Units = Q / AD ** 0.5  
31    # Si Units = Ap / A * (g * yh / D) ** 0.5  
32    Ap = wetted_area  
33    si1 = Q / (Afull * (D ** 0.5))  
34    si2 = (Ap / Afull) * ((9.8 * (depth / D)) ** 0.5)  
35  
36    diff = abs(si1 - si2)  
37  
38    if diff < tolerance:  
39        break  
40  
41    depth_half = depth_half / 2  
42    if si2 >= si1:  
43        depth = depth - depth_half # Lower the depth  
44    else:  
45        depth = depth + depth_half # Increase the depth  
46  
47    i += 1
```

Run

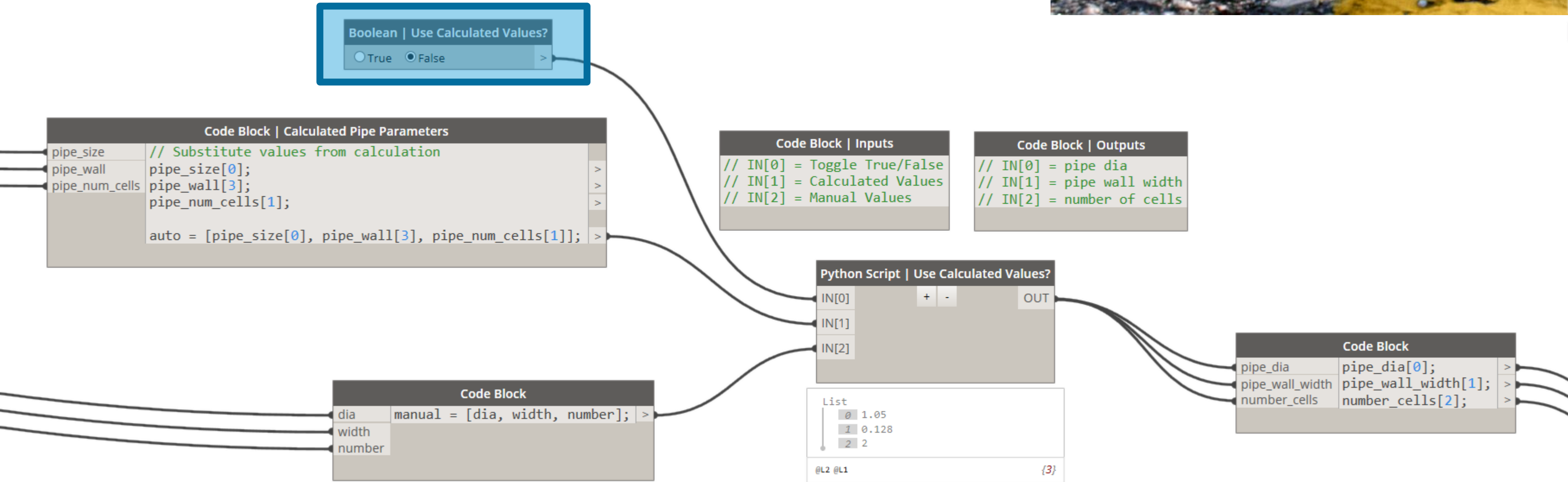
Save Changes

Revert



# Automatic or Manual ?

```
Python Script | Use Calculated Values?  
1 # Load the Python Standard and DesignScript Libraries  
2 import clr  
3  
4 #Inputs  
5 toggle = IN[0]  
6 auto = IN[1]  
7 manual = IN[2]  
8  
9 # Lists  
10 pipe_size = manual[0]  
11 wall_width = manual[1]  
12 number_cells = manual[2]  
13  
14 if toggle:  
15     pipe_size = auto[0]  
16     wall_width = auto[1]  
17     number_cells = auto[2]  
18  
19 # Assign your output to the OUT variable.  
20 OUT = pipe_size, wall_width, number_cells  
21  
22
```





# Export Values to Excel

AutoSave Off

Culvert\_Python\_Export.xlsx - Excel

Andrew Milford

File Home Insert Draw Page Layout Formulas Data Review View Developer Results Connect Help Autodesk Vault Team Tell me Share Comments

Clipboard: Paste, Cut, Copy, Format Painter

Font: Calibri, 11, Bold, Italic, Underline, Text Color, Background Color

Alignment: Wrap Text, Merge & Center

Number: General, Currency, Percentage, Decimals

Styles: Conditional Formatting, Format as Table, Cell Styles

Cells: Insert, Delete, Format

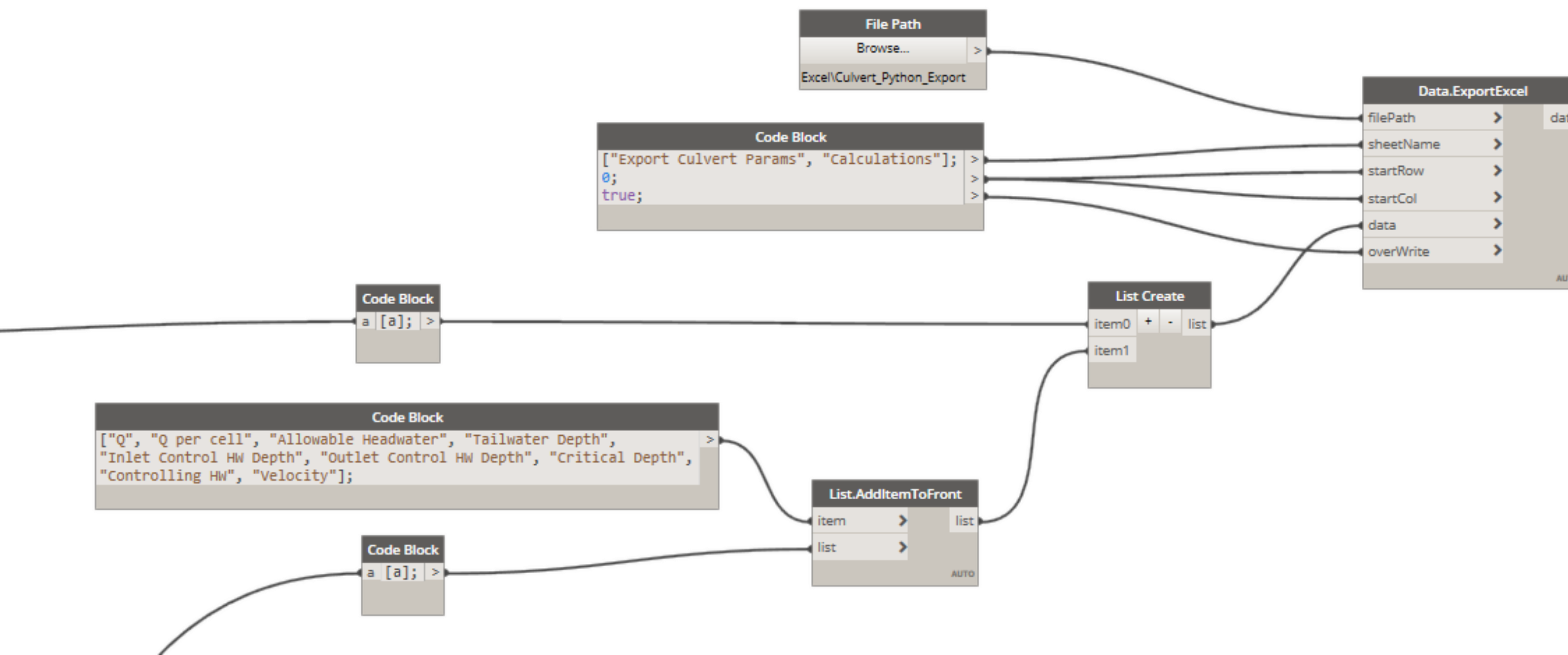
Editing: AutoSum, Fill, Clear, Sort & Filter, Find & Select

Formulas: A1, 219

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	219	15	0	0	-0.7	0.1	0.6	0.096	0.1	4	-13.9	-10	315	225	225	315	1.5	1.5	1.5	1.5	0.15	0.5	0.5
2																							
3																							
4																							

Calculations Export Culvert Params Sheet1

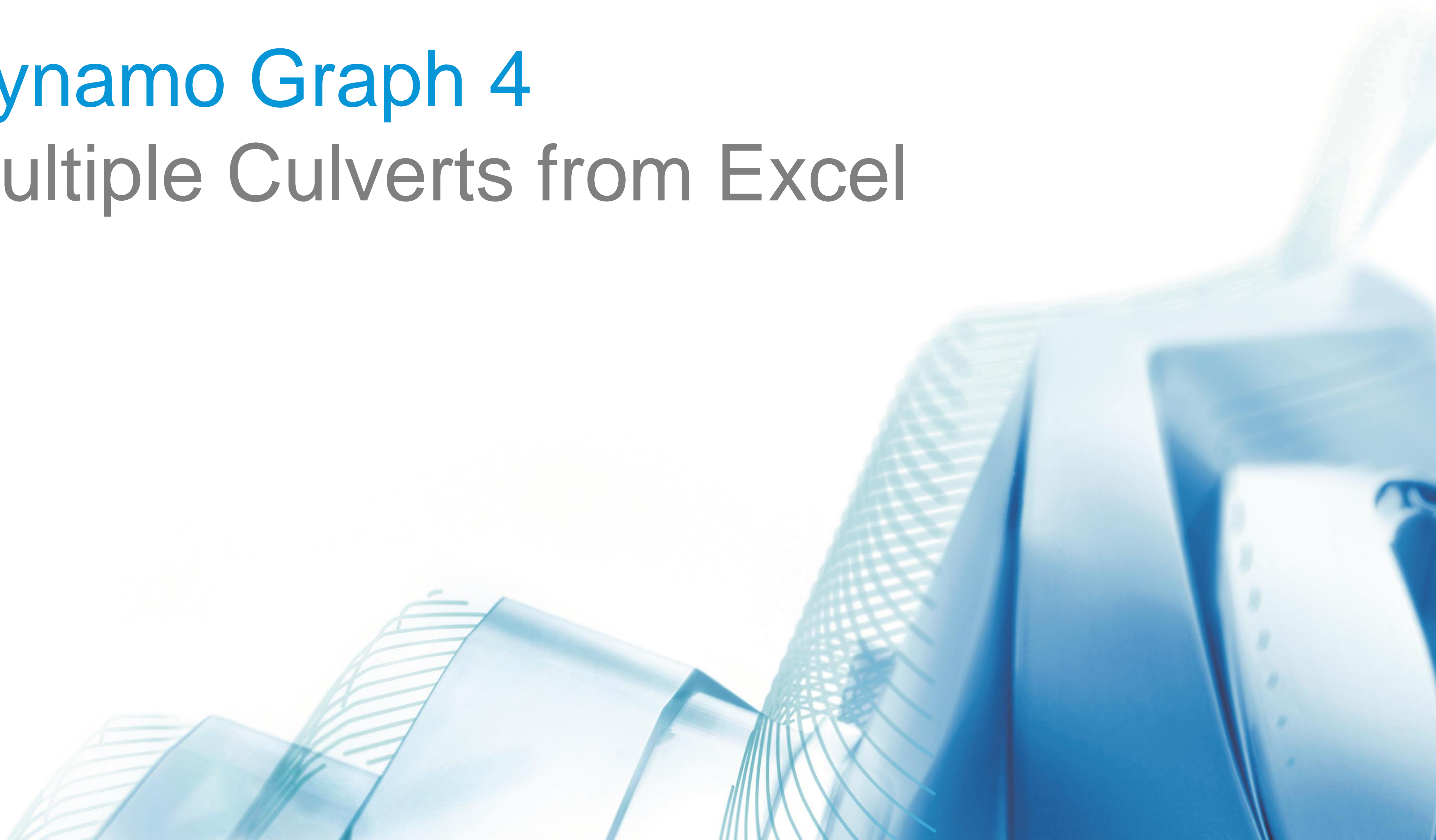
100%





# Dynamo Graph 4

## Multiple Culverts from Excel





# Import Culvert Parameters from Excel

AutoSave Off | Culverts\_Setout.xlsx - Excel | Andrew Milford

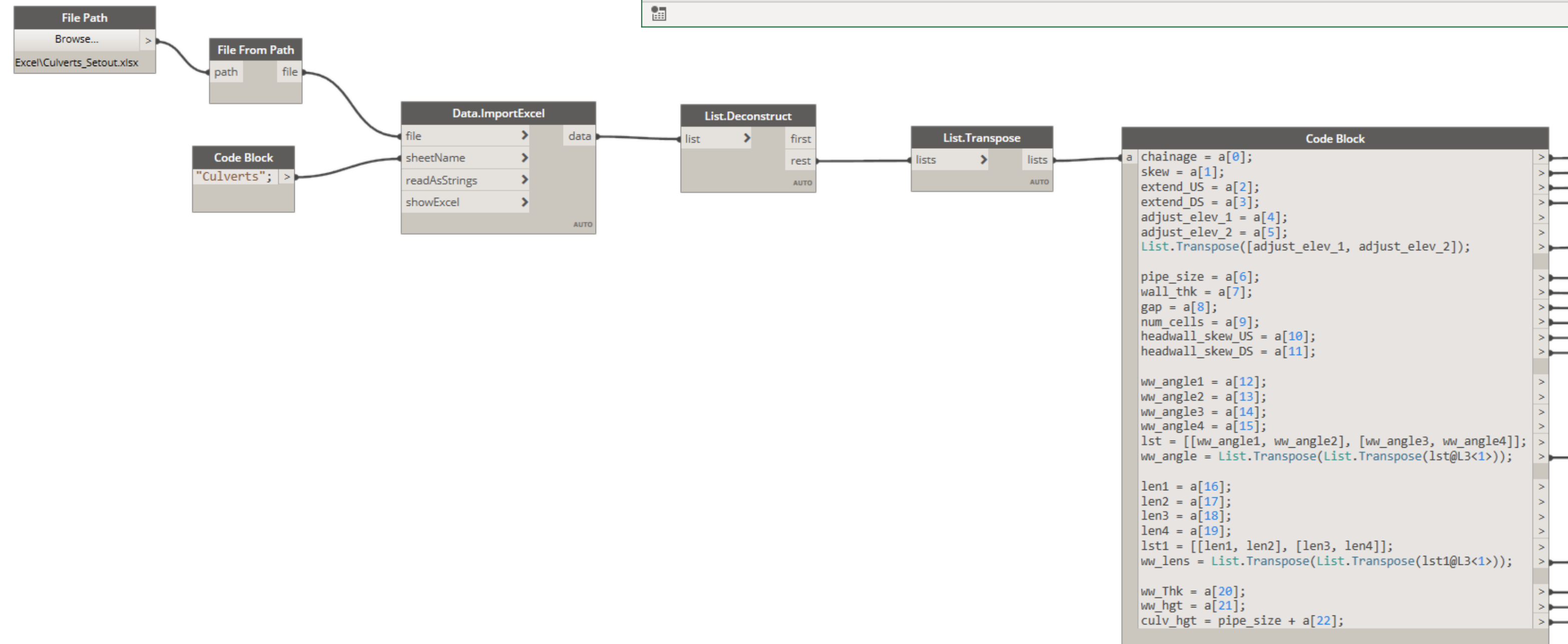
File Home Insert Draw Page Layout Formulas Data Review View Developer Help

Clipboard: Paste, Cut, Copy, Format Painter | Font: Calibri, 11, Bold, Italic, Underline, Color, Background Color | Alignment: Wrap Text, Merge & Center | Number: General, Currency, Percentage, Decimals | Styles: Normal, Good, Bad, Neutral | Cells: Insert, Delete, Format | Editing: AutoSum, Fill, Clear, Sort & Filter, Find & Select

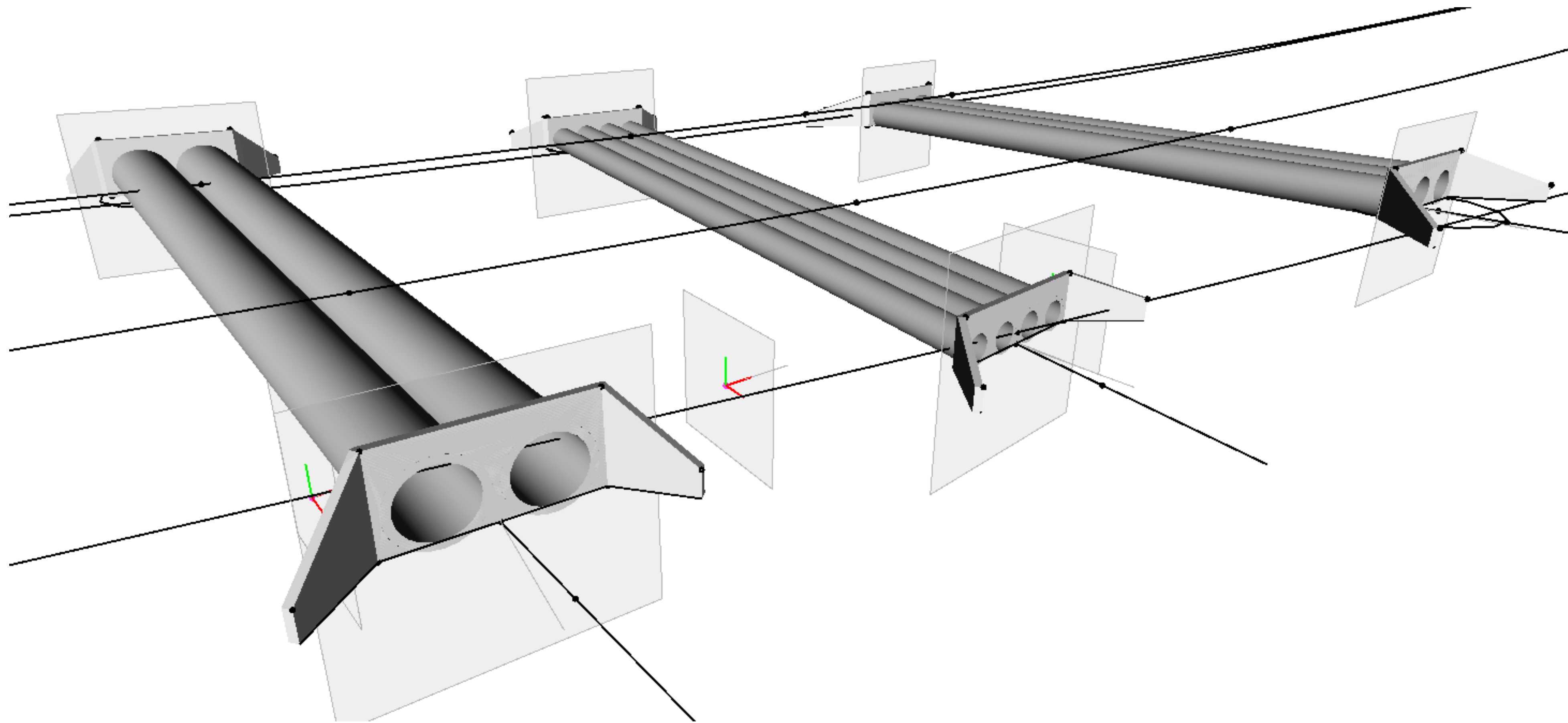
Formula Bar: B1 | Skew

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Skew	Extend US	Extend DS	Adjust Elev 1	Adjust Elev 2	Pipe Size	Wall Thick	Gap	Number	Headwall Skew US	Headwall Skew DS	WW Angle 1	WW Angle 2	WW Angle 3	WW Angle 4	Len1	Len2	Len3	Len4	WW Thk	WW Hgt	Cul Hgt
2	12	1	1	-0.5	0	1.2	0.14	0.1	2	-14	-10	315	225	225	315	1.5	1.5	1.5	1.5	0.15	0.5	0.5
3	5	0	0	-0.25	0.25	0.6	0.096	0.1	4	0	5	312.5	216.5	228.25	319.25	1.4	1.3	1.65	1.75	0.15	0.5	0.5
4	7	-0.5	0	0.5	-0.5	0.9	0.114	0.2	3	-5	3	315	227	229.5	320	2.5	2.5	2.6	2.7	0.15	0.5	0.5
5																						
6																						
7																						

Sheet: Culverts | Culverts (2) | 100%

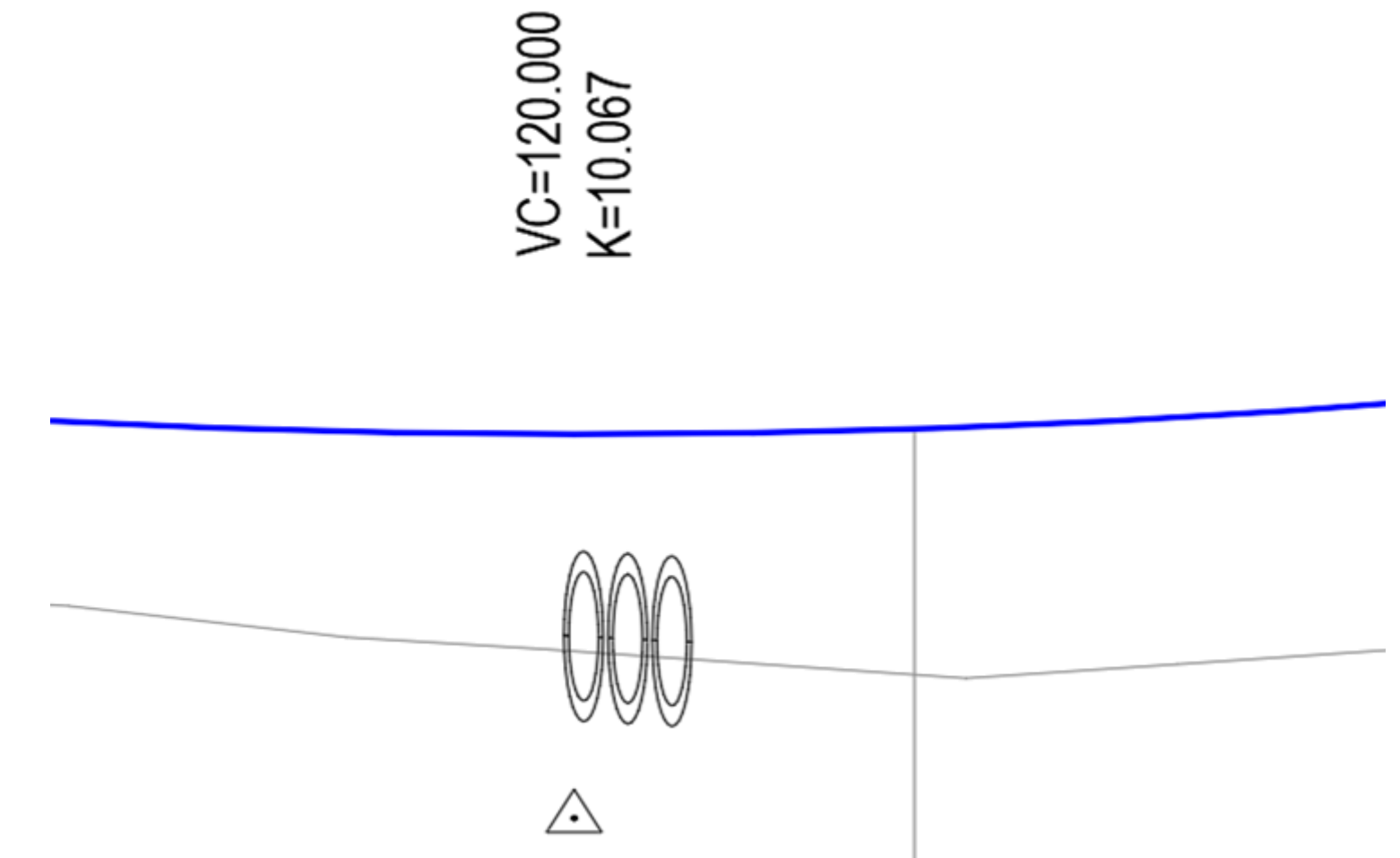
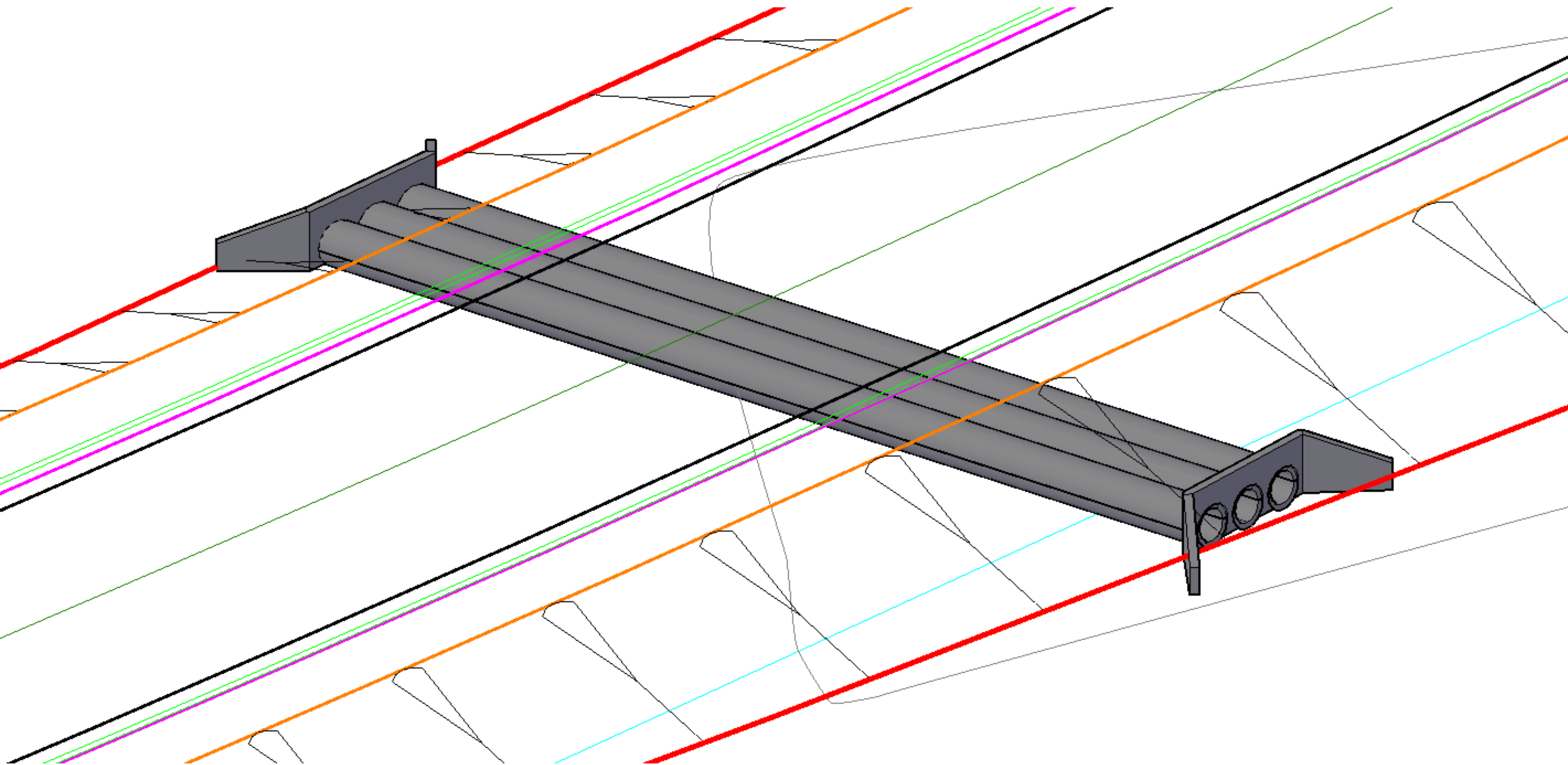
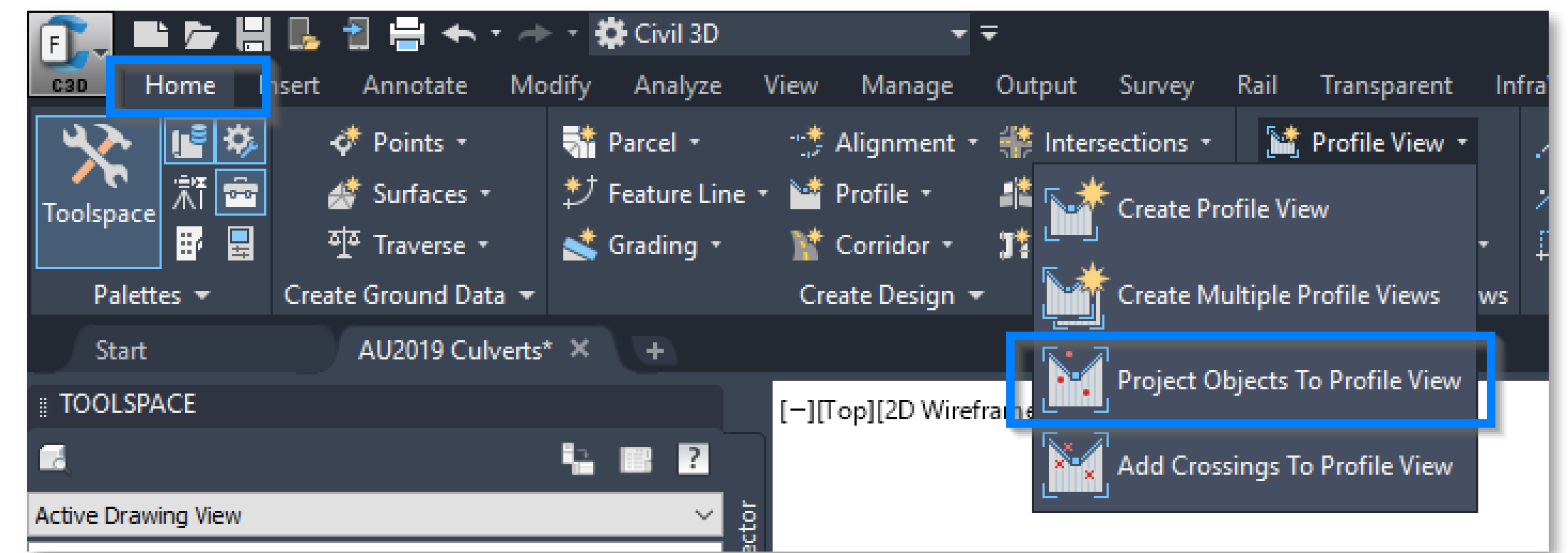






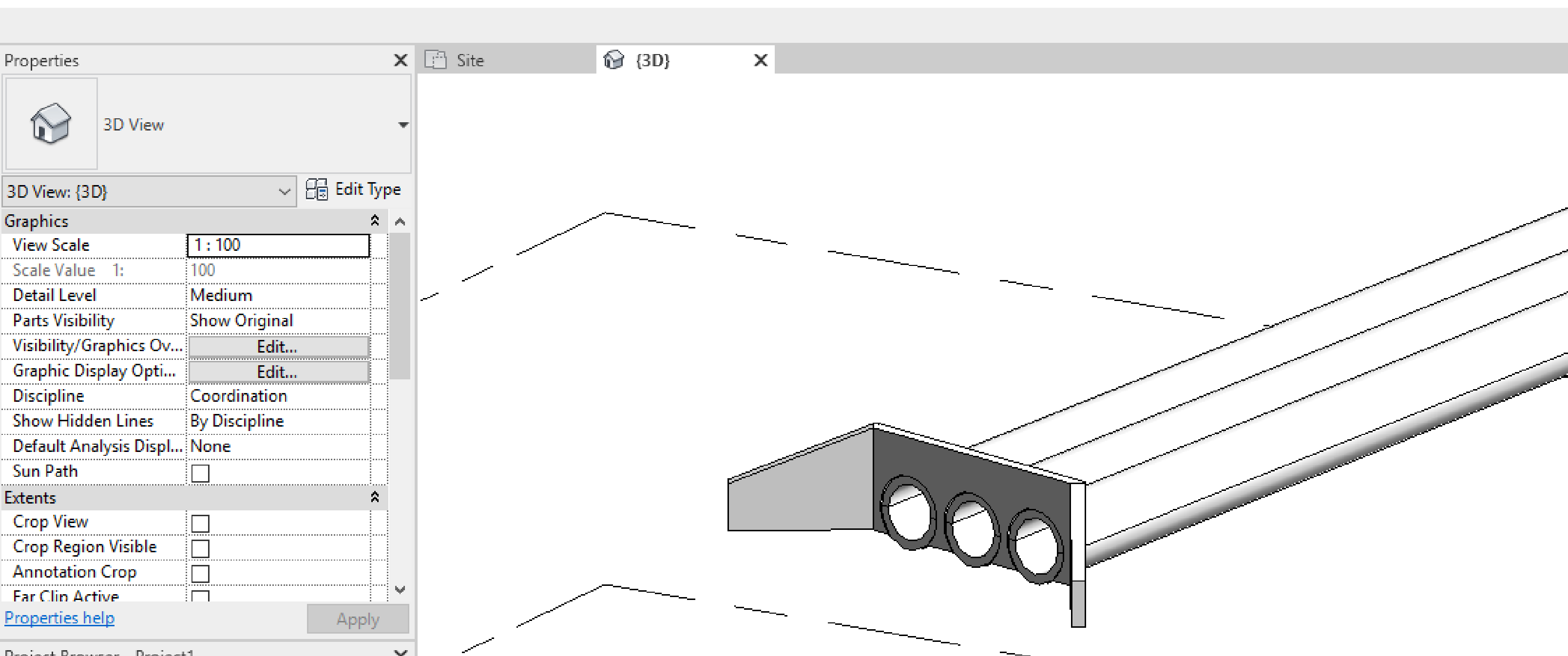
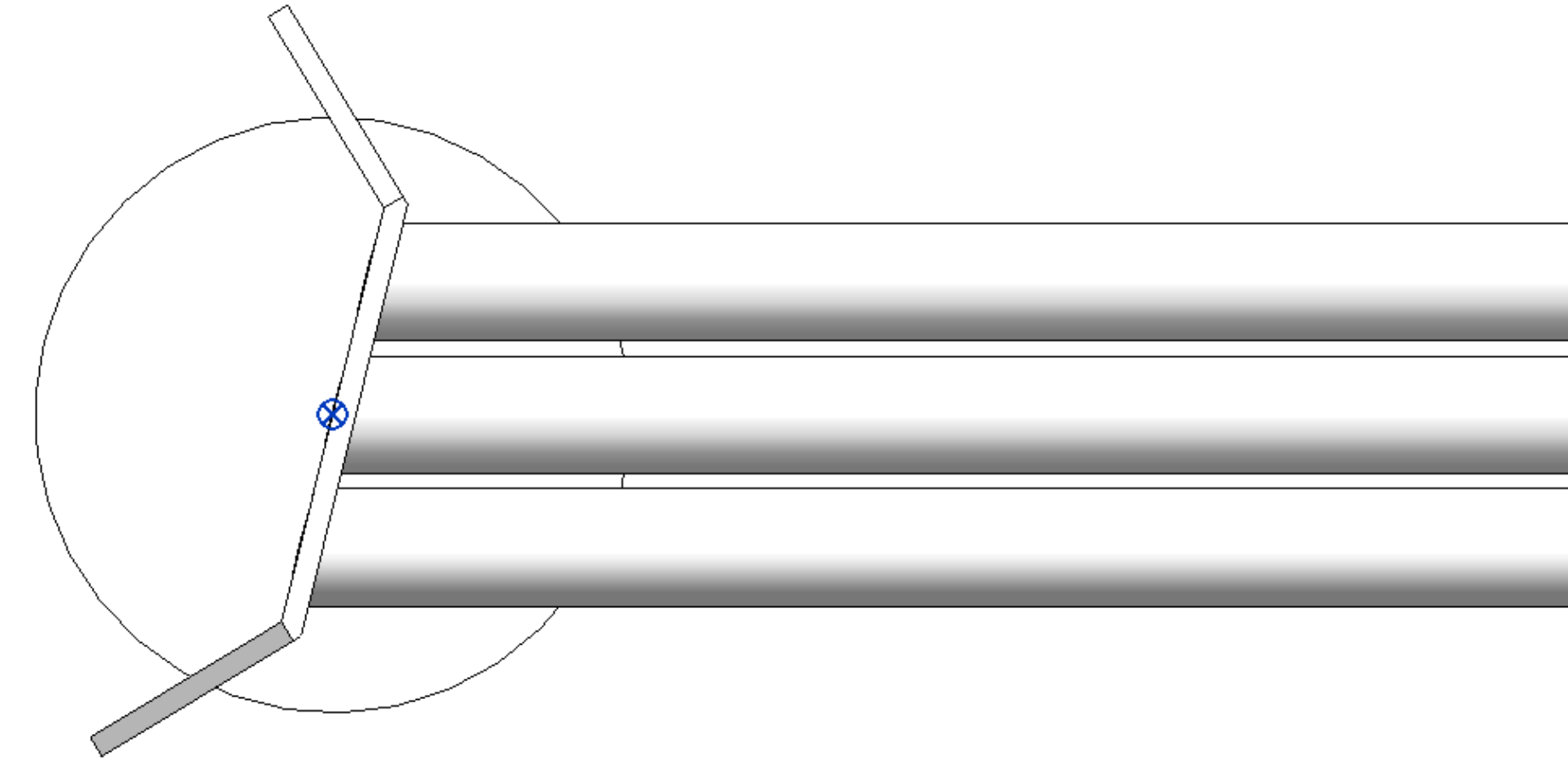
Multiple Culverts





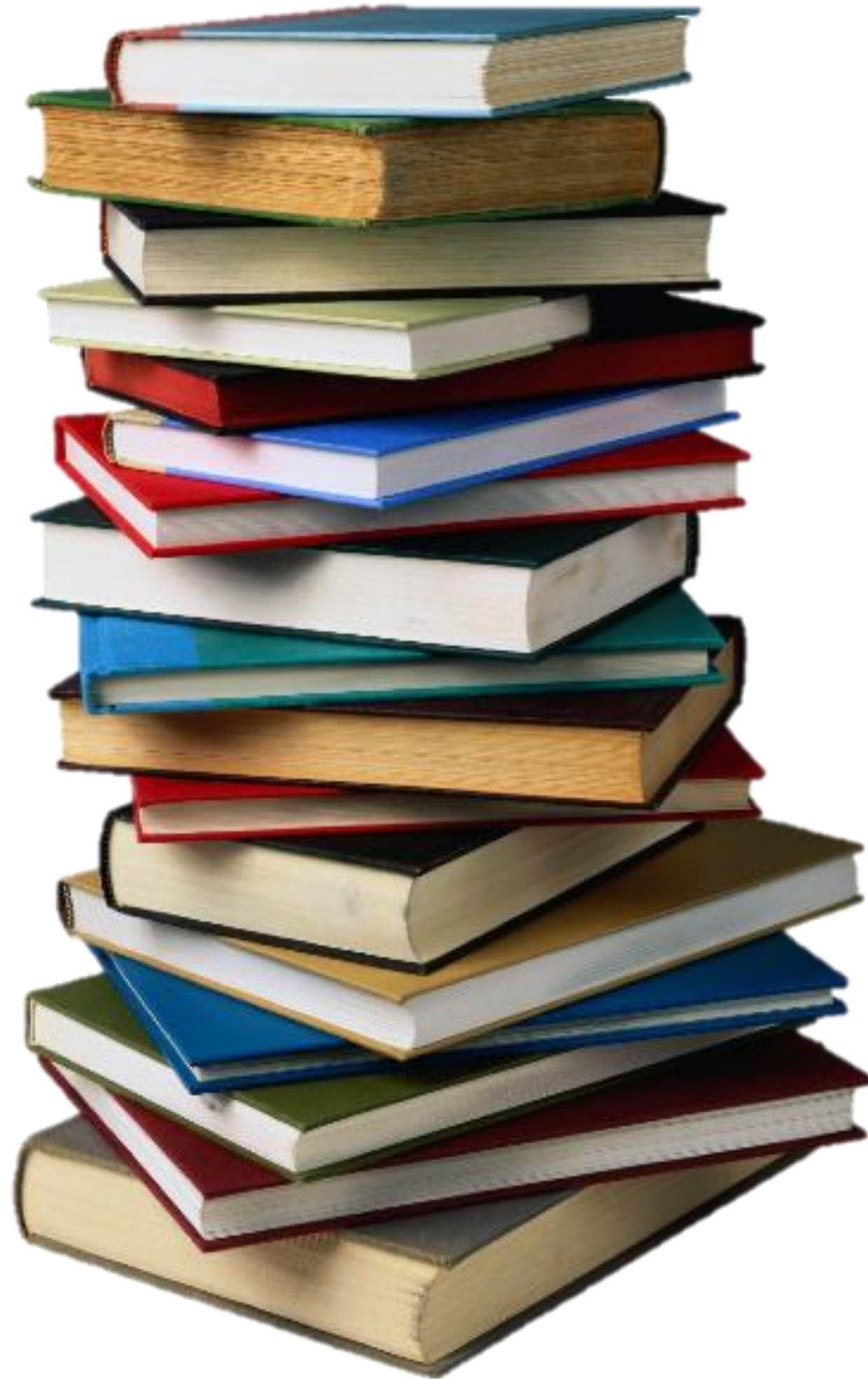
# Culverts in Civil 3D





# Culverts in Revit

# Additional Resources



- [DynamoBIM.org](https://dynamobim.org)
- [DynamoPrimer.com](https://dynamoprimer.com)
- [GitHub/DynamoDS](https://github.com/DynamoDS)
- Blogs, YouTube videos
- AU lessons and handouts
- LinkedIn Learning / CadLearning / Pluralsight etc





# AUTODESK®

## Make anything™

Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2019 Autodesk. All rights reserved.

