Civil 3D and Dynamo

Dynamic Culvert Design and Analysis

Andrew Milford

Sr. Implementation Consultant

AUTODESK. UNIVERSITY

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

Austroads

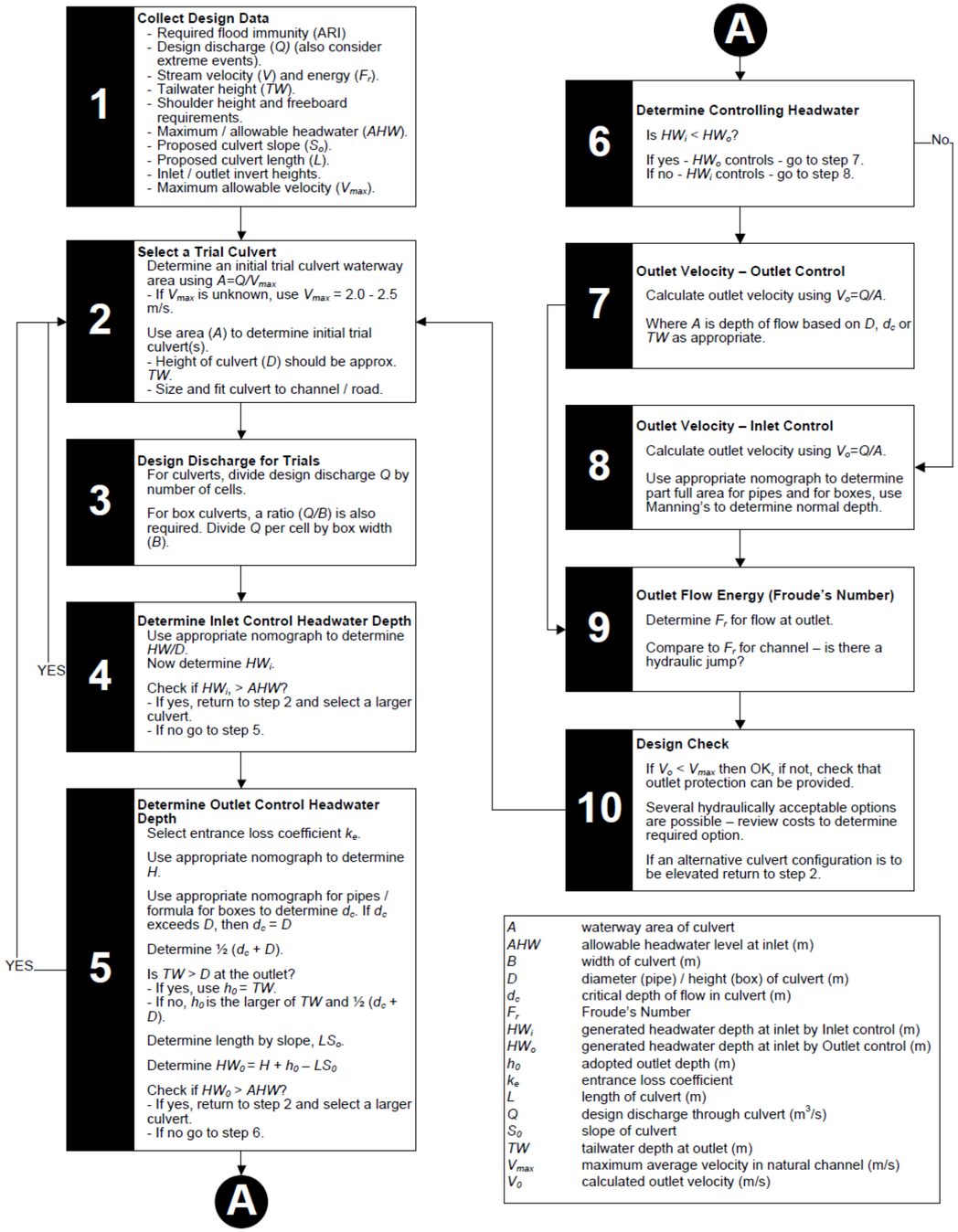
U.S. Department of Transportation
Federal Highway Administration

Culvert Design Guidelines

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

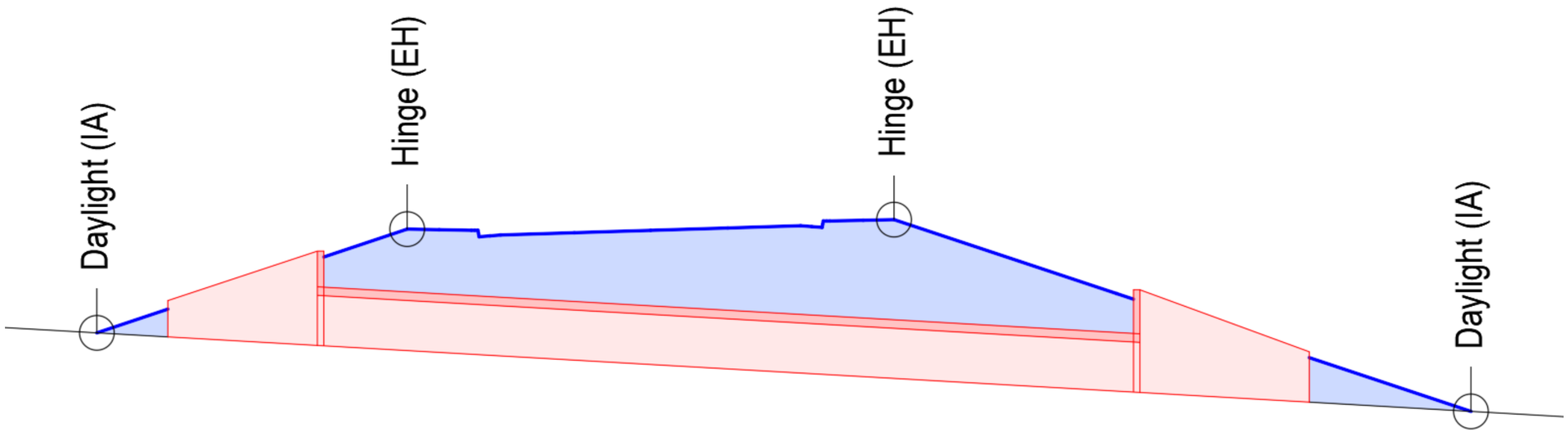
Austroads

U.S. Department of Transportation
Federal Highway Administration



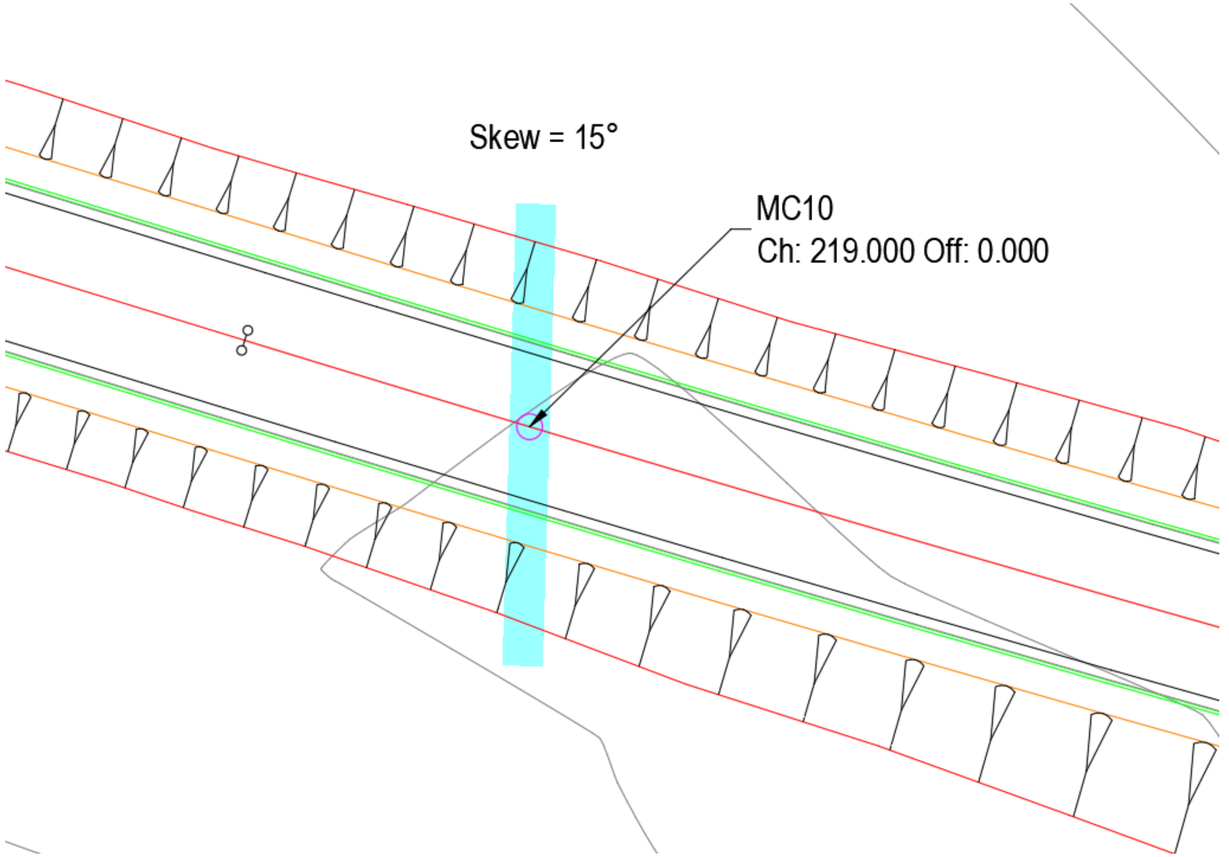
Civil 3D | Project Setup

- Corridor Codes
- Surface

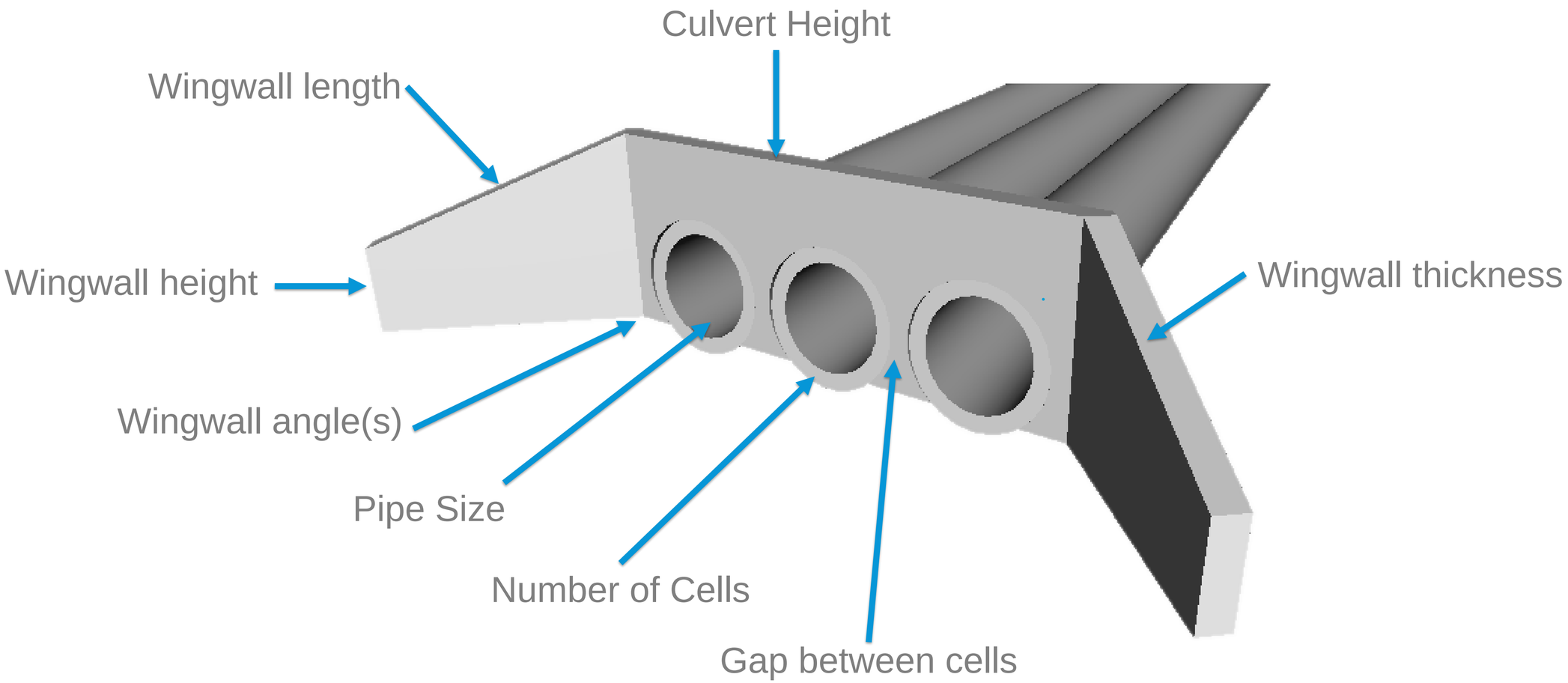


Dynamo | Culvert Setout Parameters

- Alignment String
- Station
- Skew



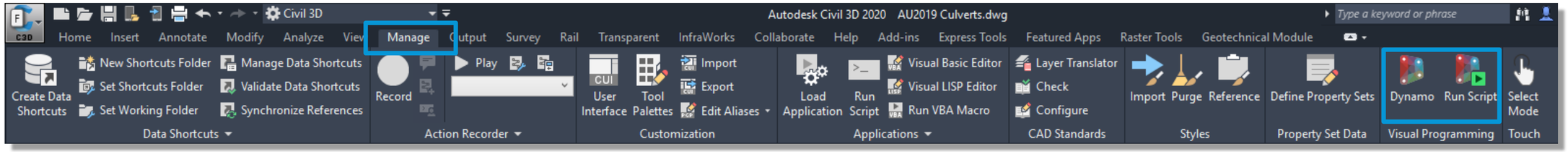
Dynamo | Culvert and Headwall Parameters



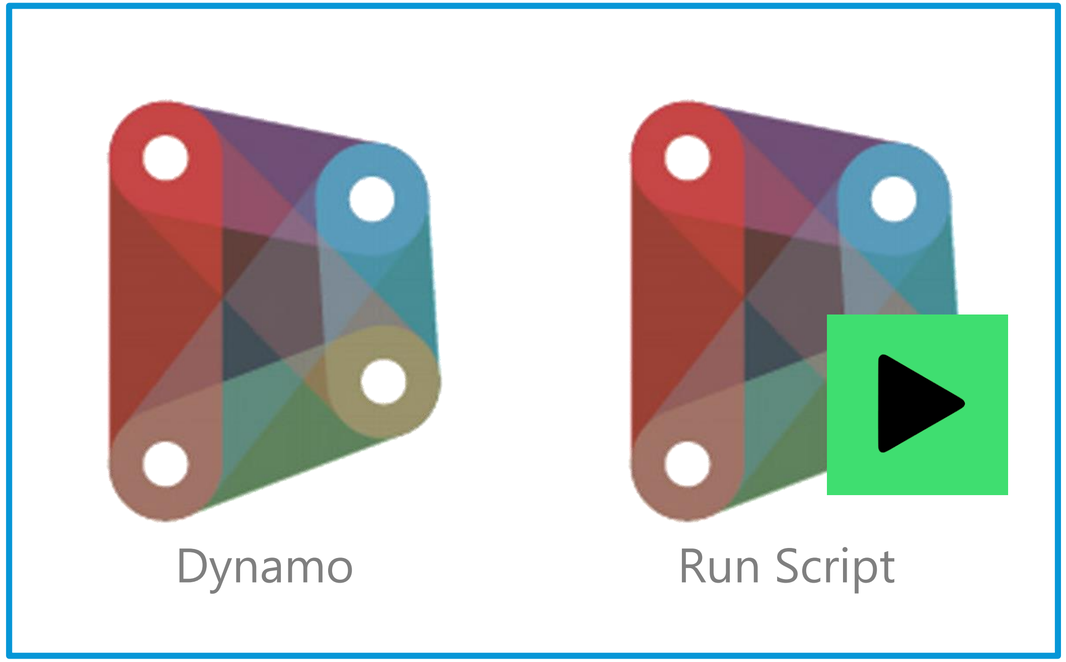
What is Dynamo?

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

How to load Dynamo in Civil 3D

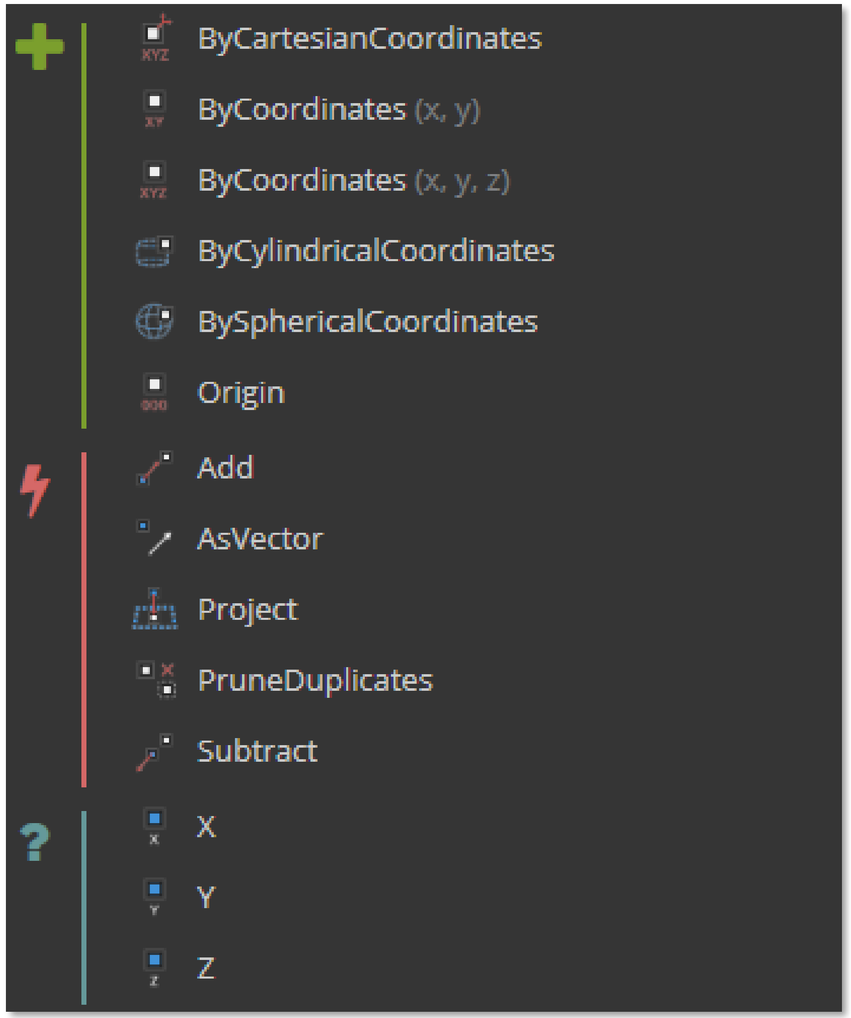


- Manage Tab
  - Visual Programming
    - Dynamo
    - Run Script

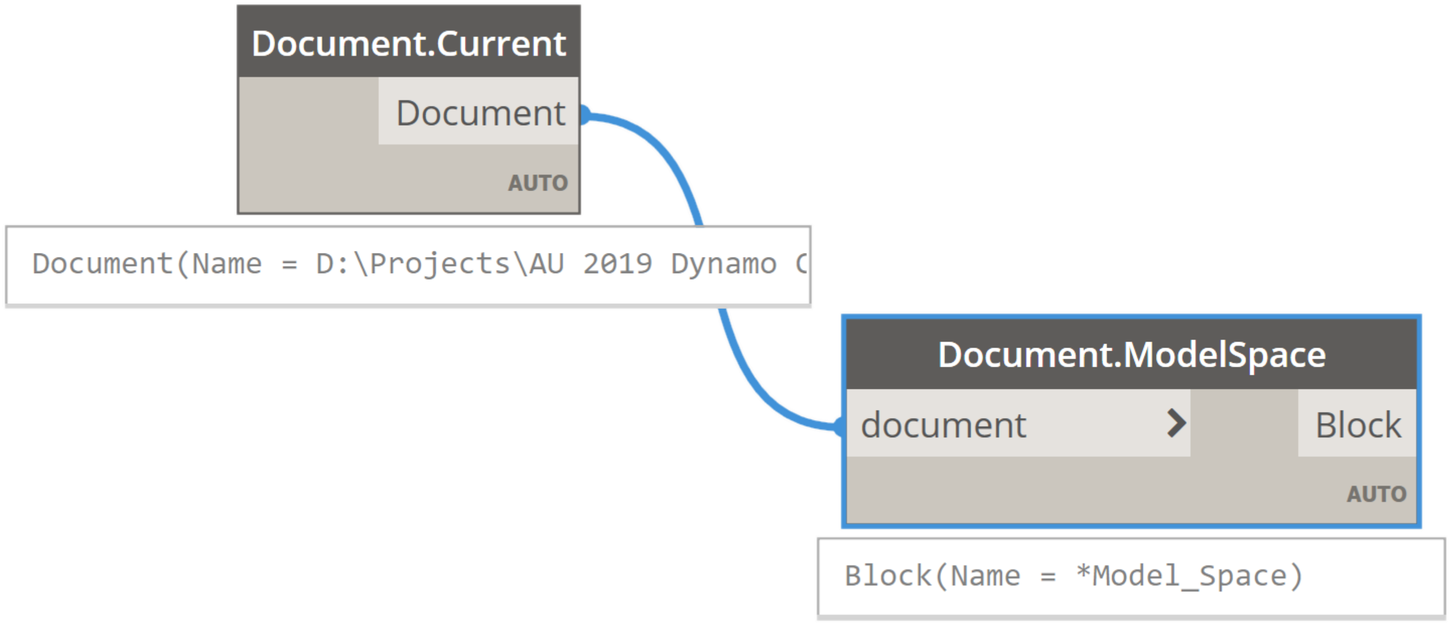


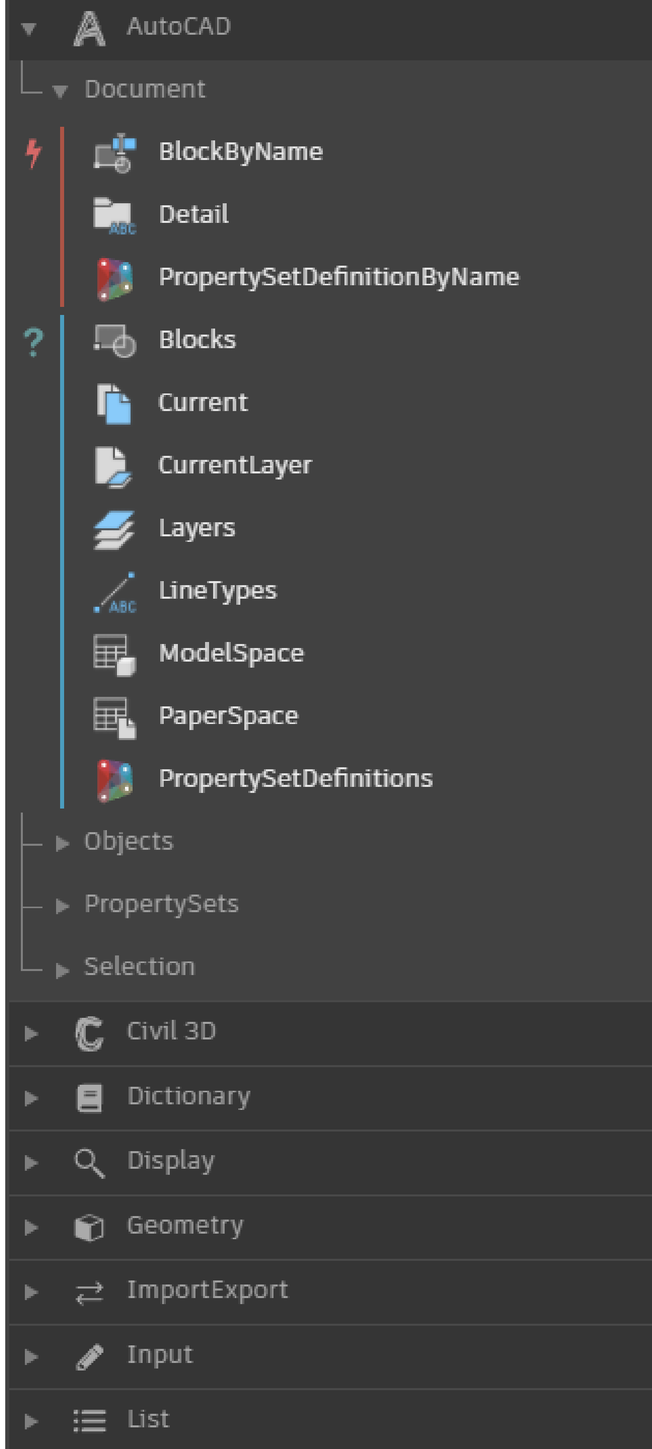
Nodes | General

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

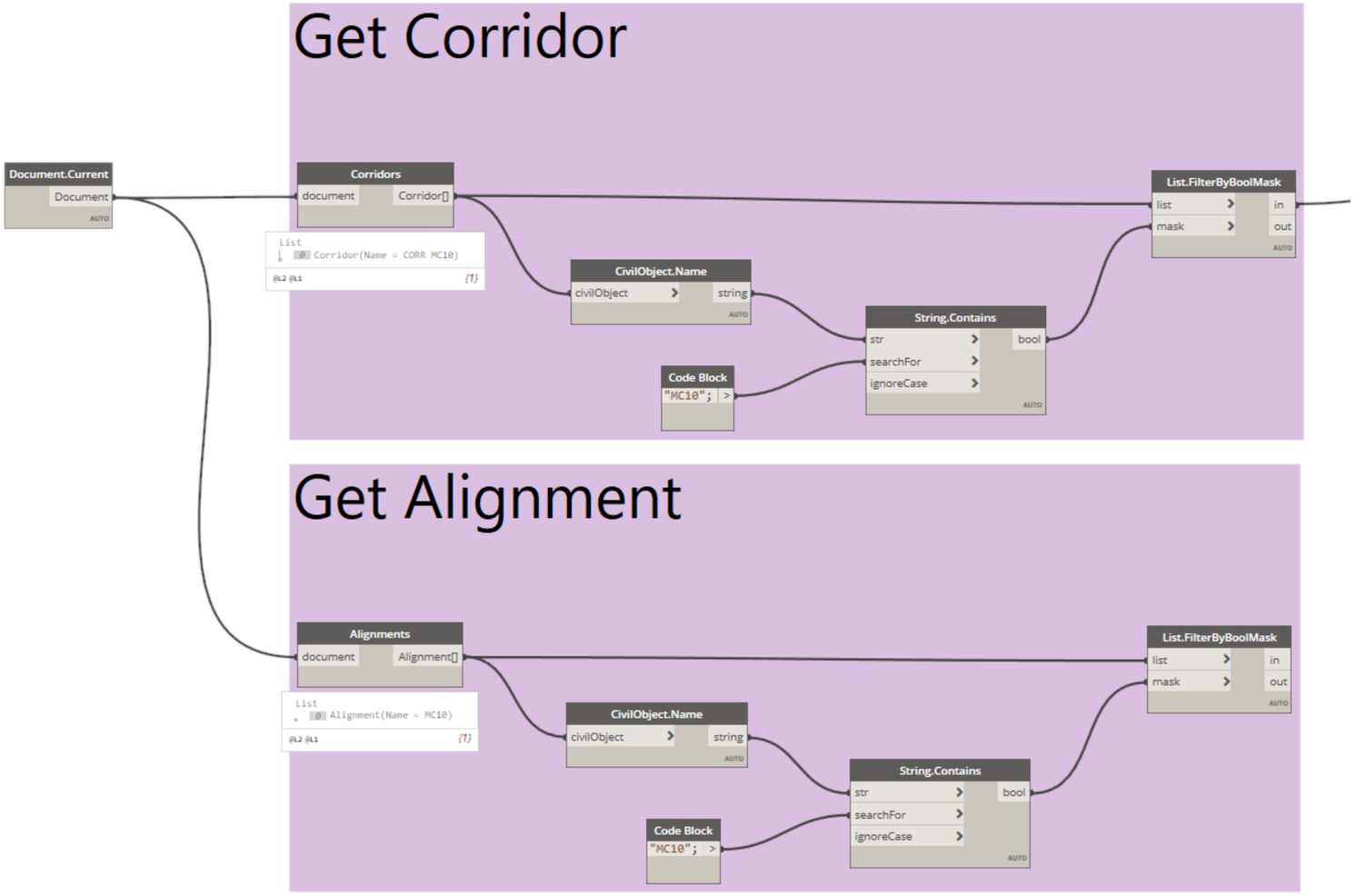


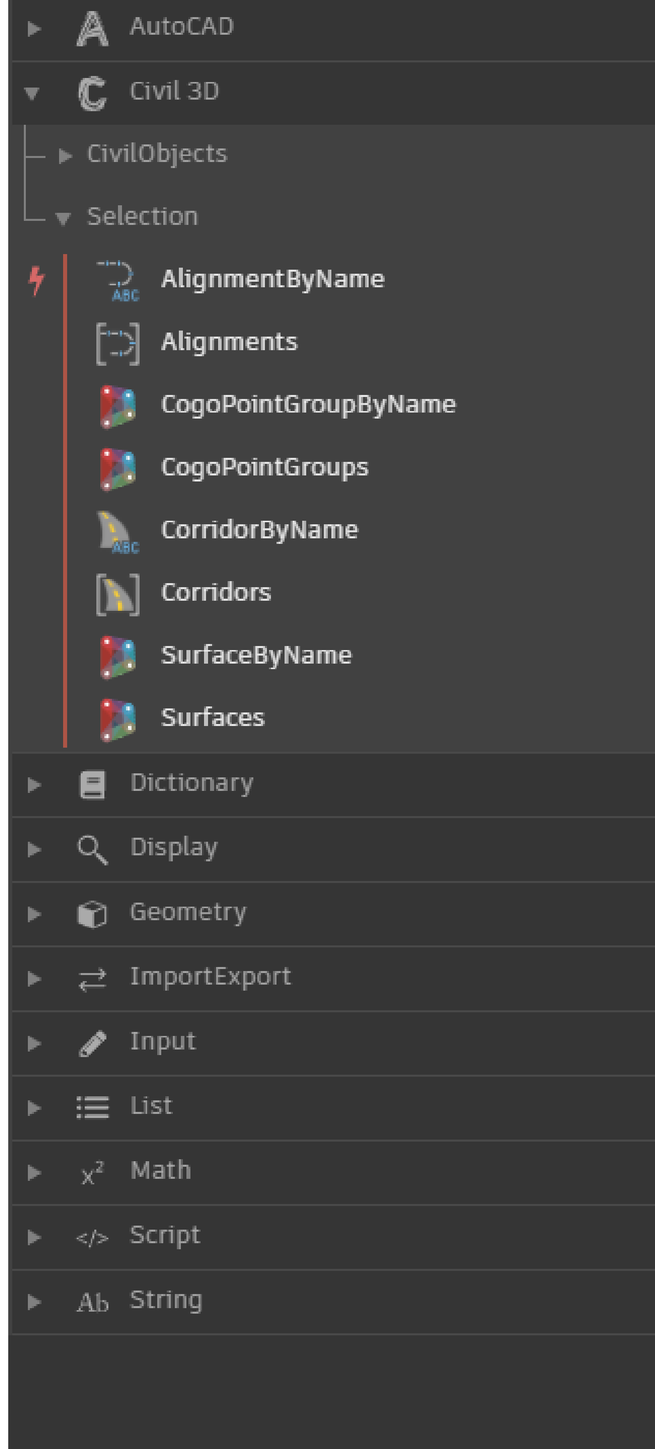
Nodes | AutoCAD



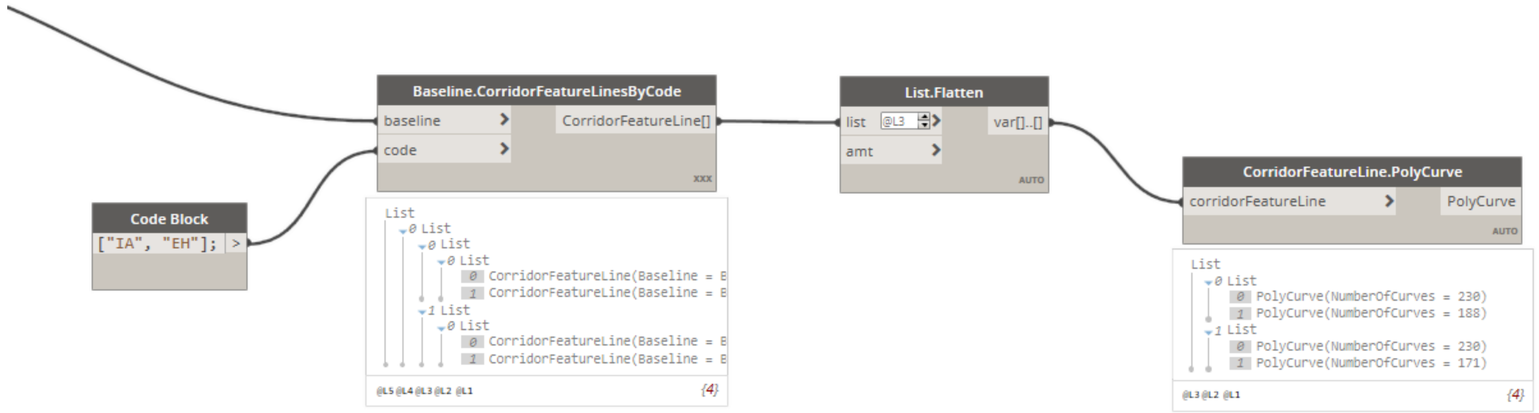


Nodes | Civil 3D Selection



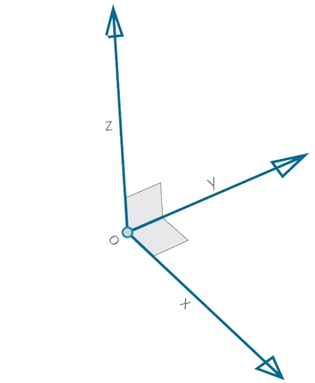


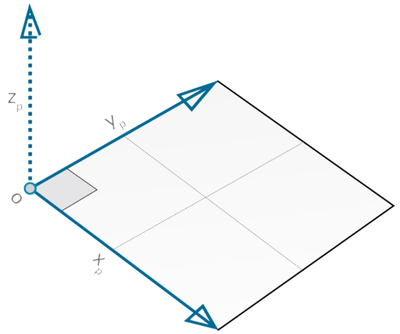
Nodes | Civil 3D CivilObjects

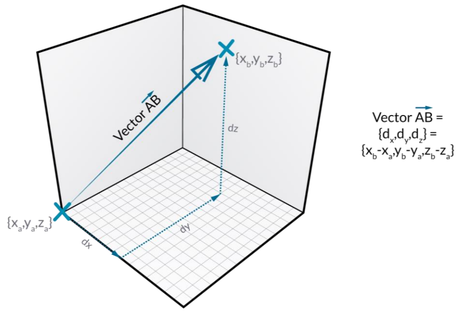


Nodes | Abstract Geometry

- Geometry
  - Abstract
    - Coordinate Systems
    - Plane
    - Vector

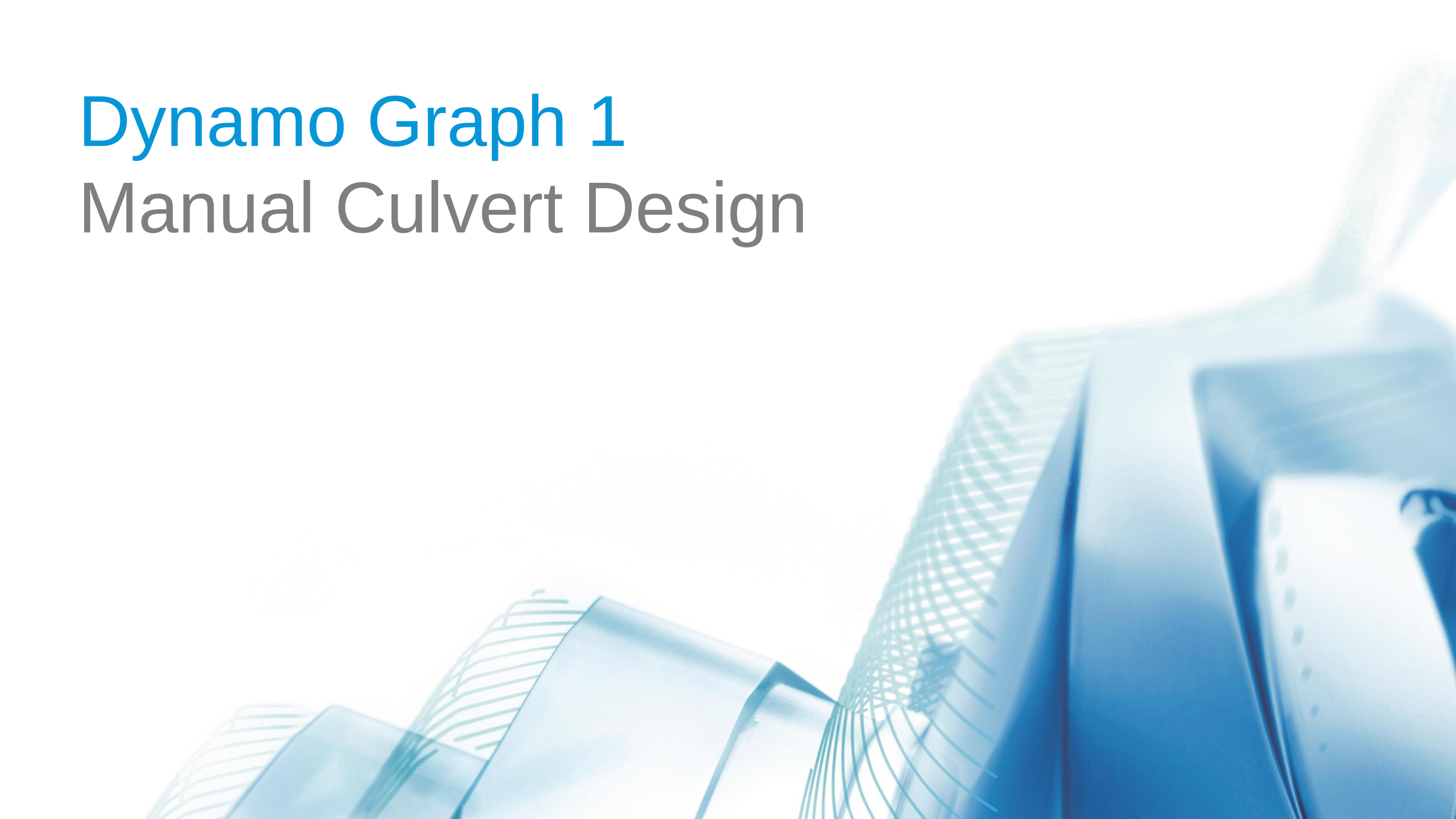




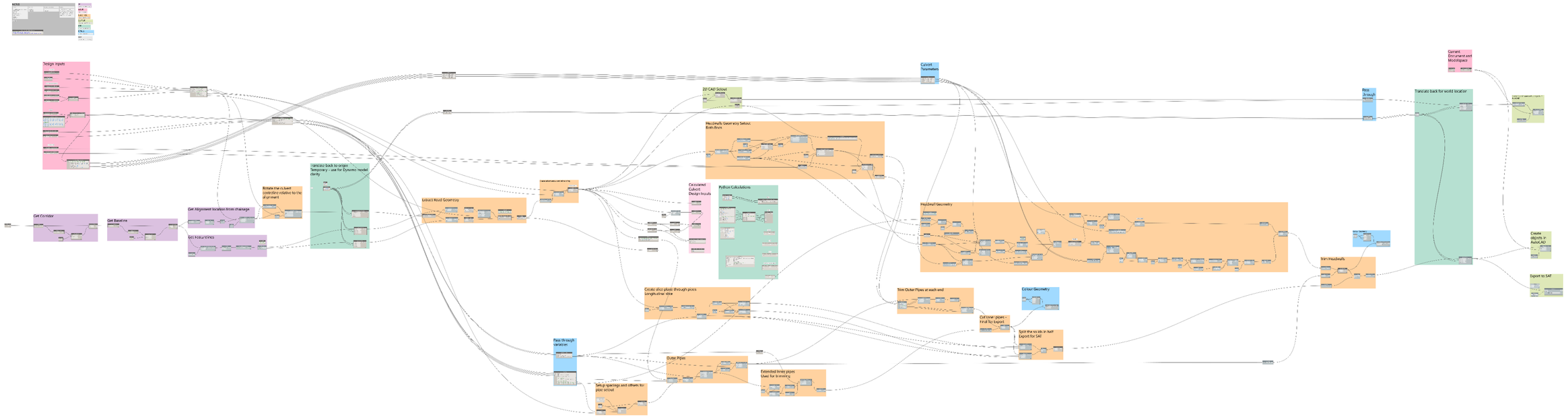


Vector $\overrightarrow{AB}$ =
$\{d_x,d_y,d_z\}$ =
$\{x_b-x_a,y_b-y_a,z_b-z_a\}$

Dynamo Graph 1
Manual Culvert Design

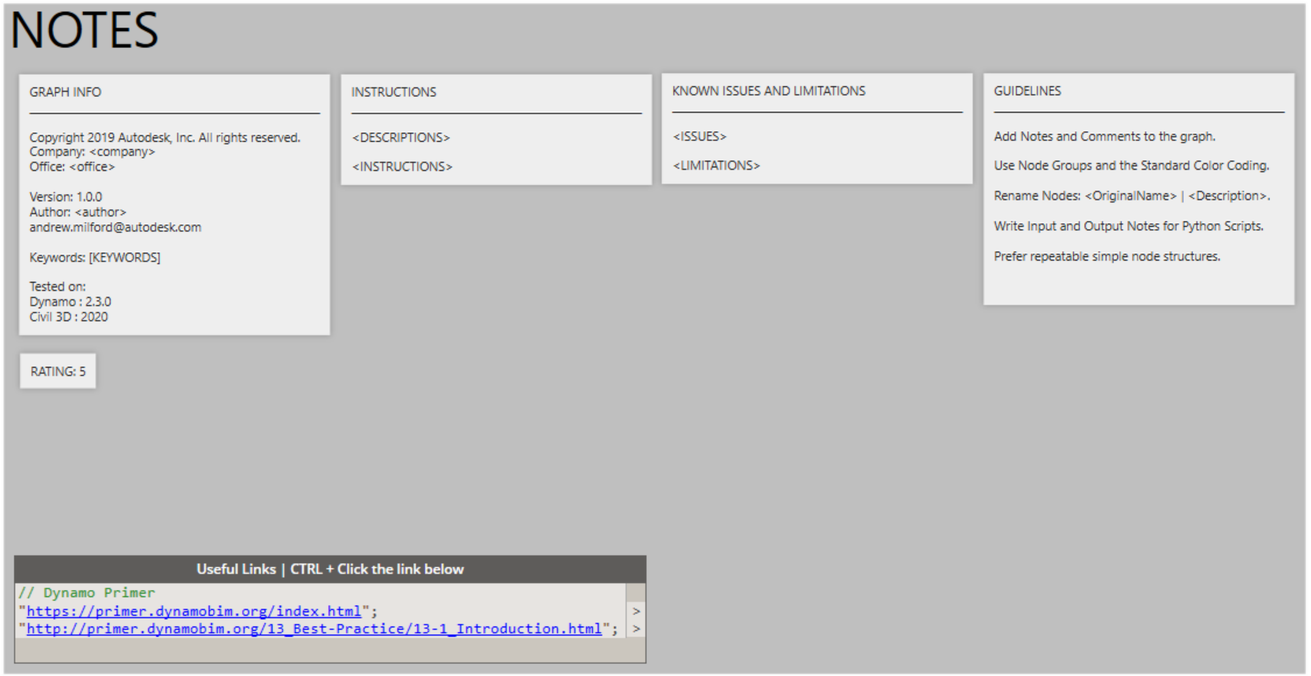


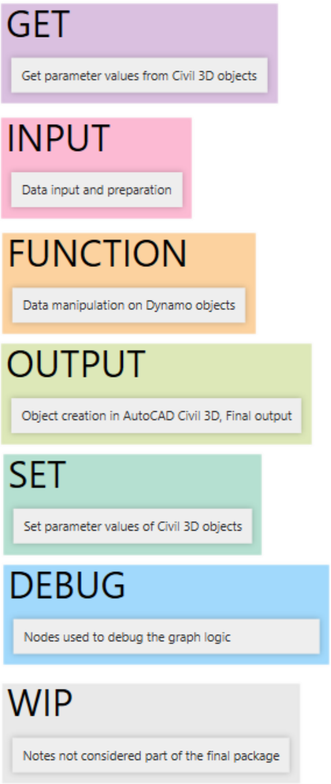
Dynamo Graph 1 | Manual Culvert Design



Graph Setup | Template

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



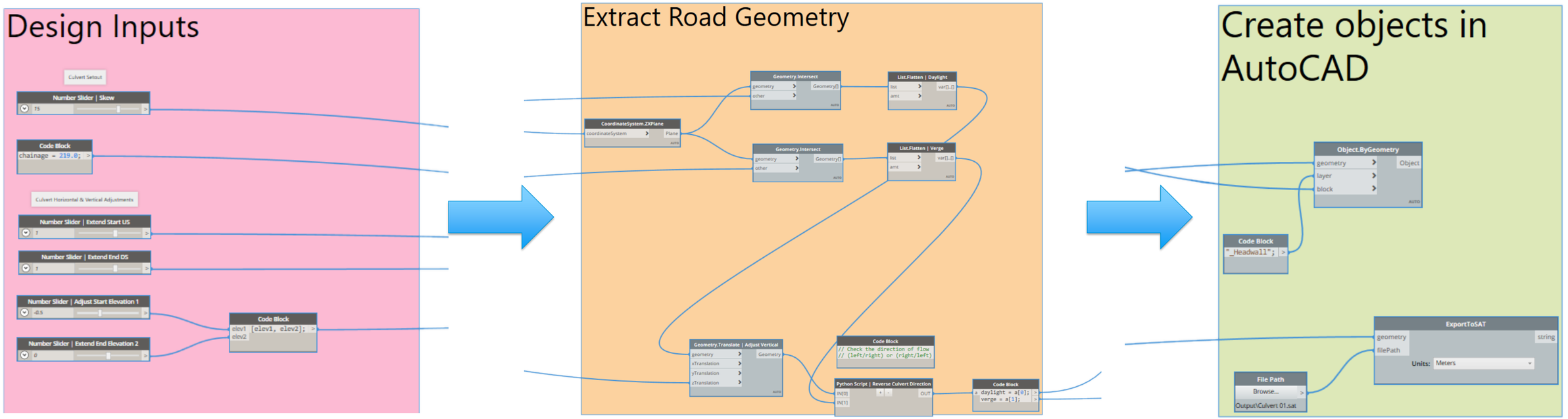


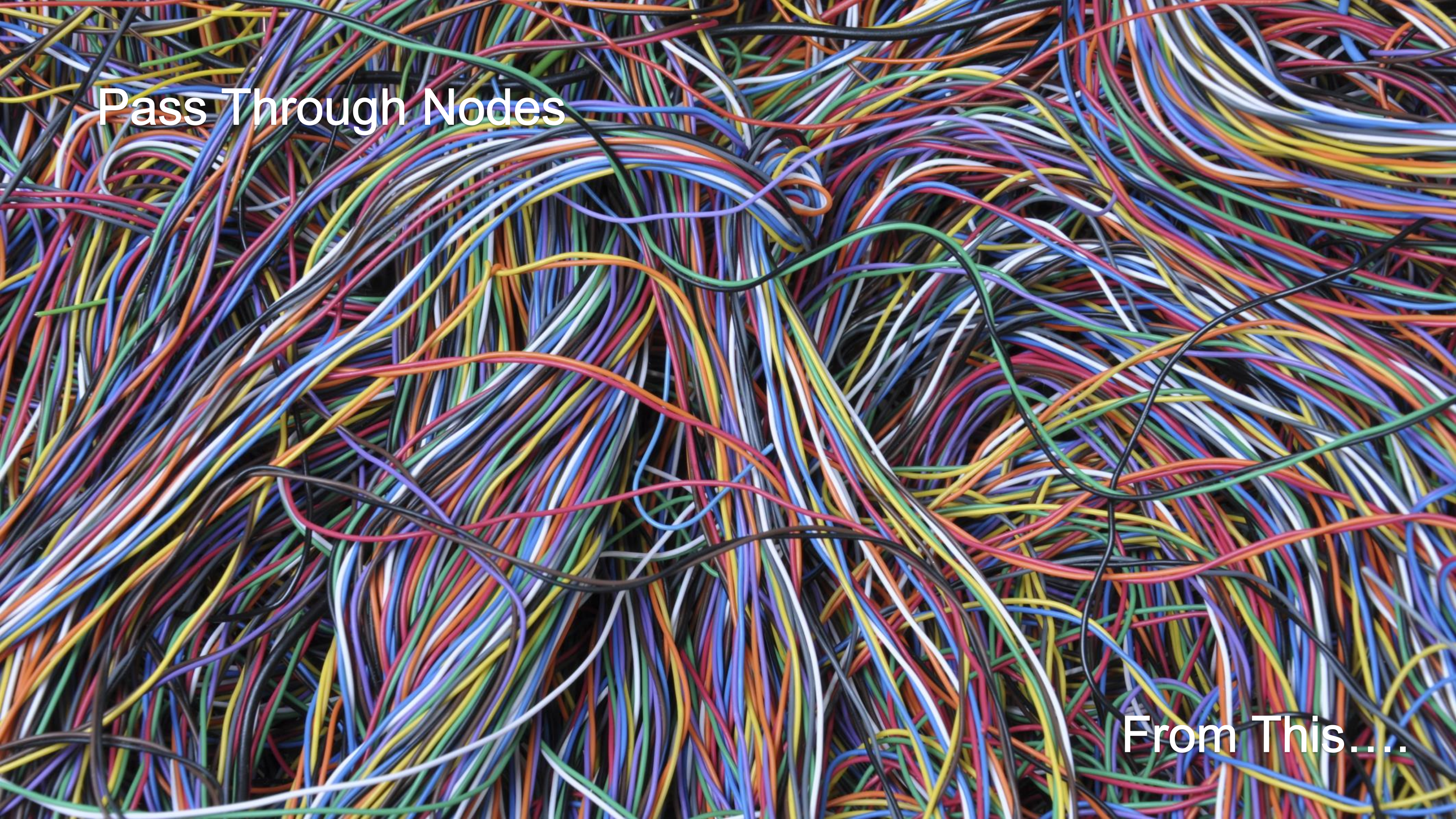
Graph Layout

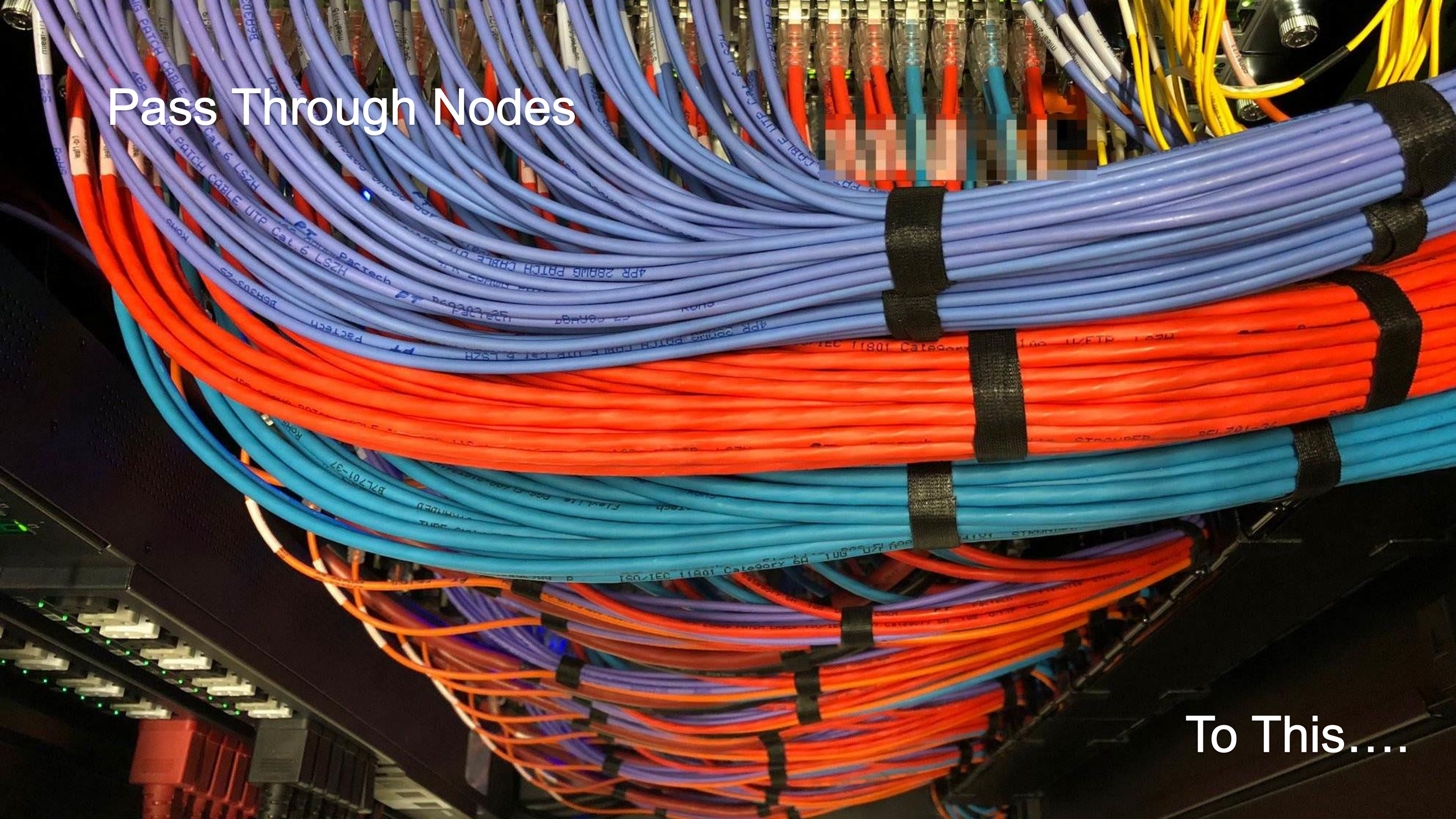
Inputs

Functions

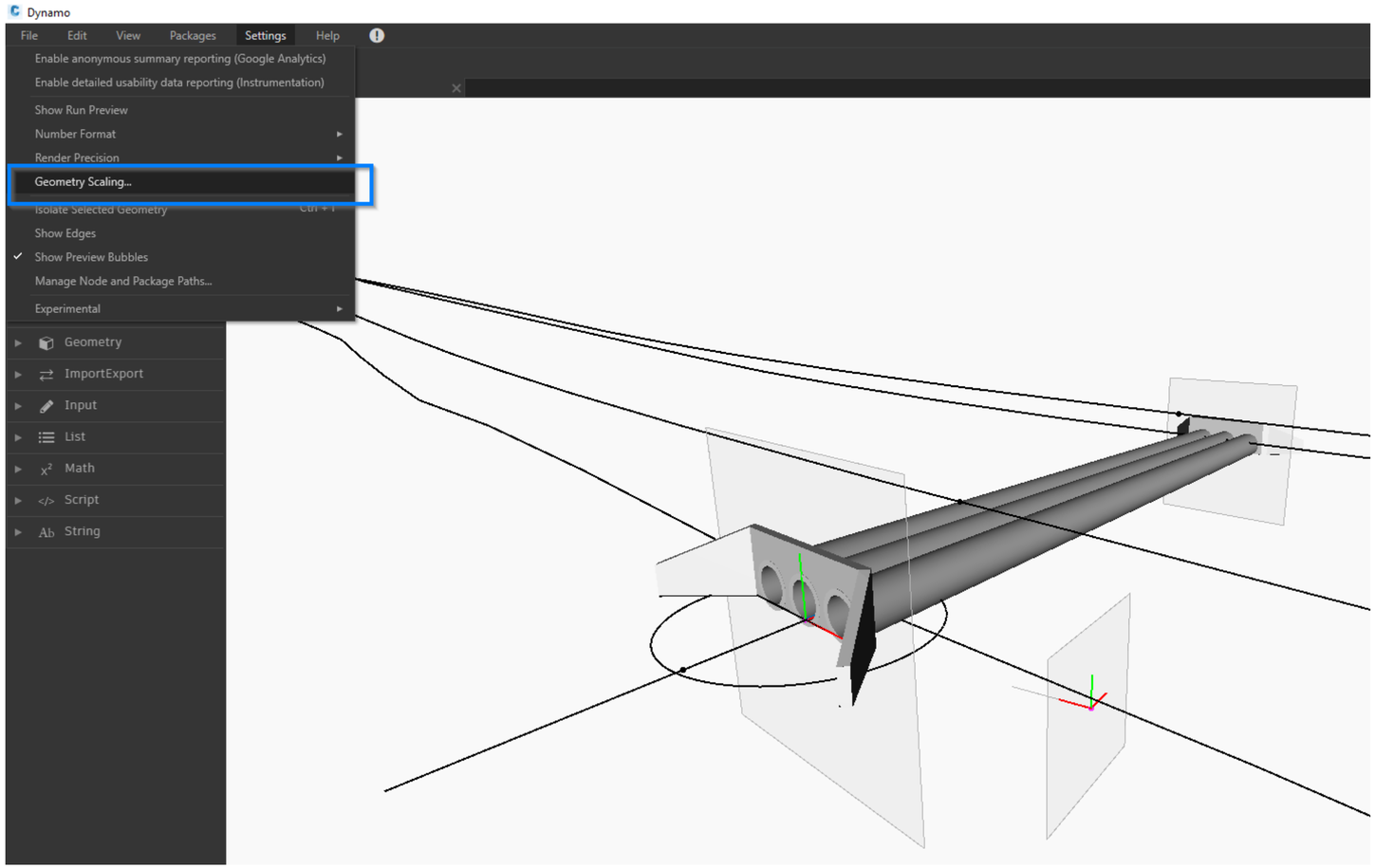
Outputs

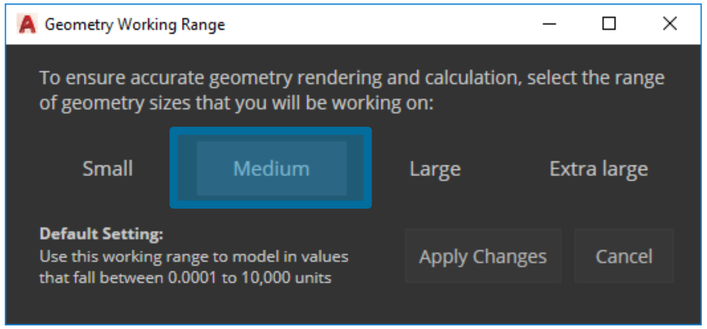


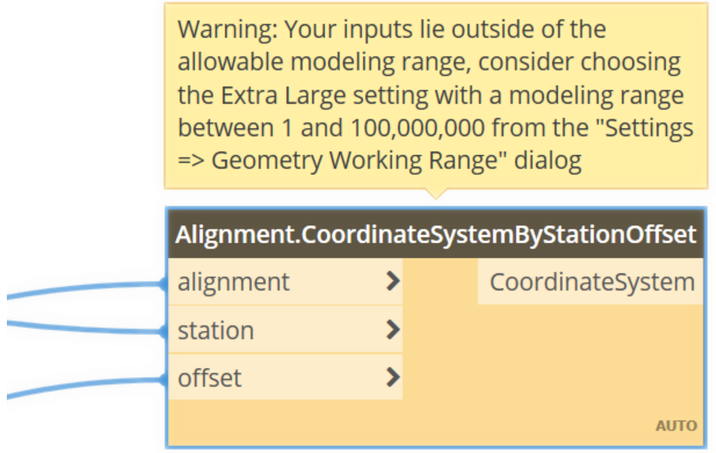

Pass Through Nodes
From This….


Pass Through Nodes
To This….

Dynamo Warnings | Geometry Scaling



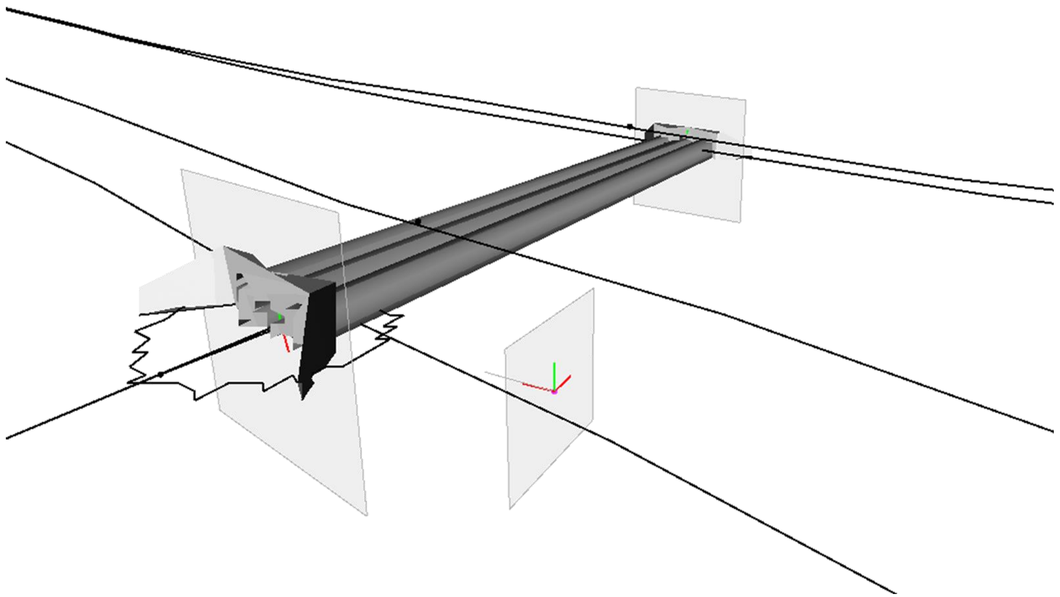




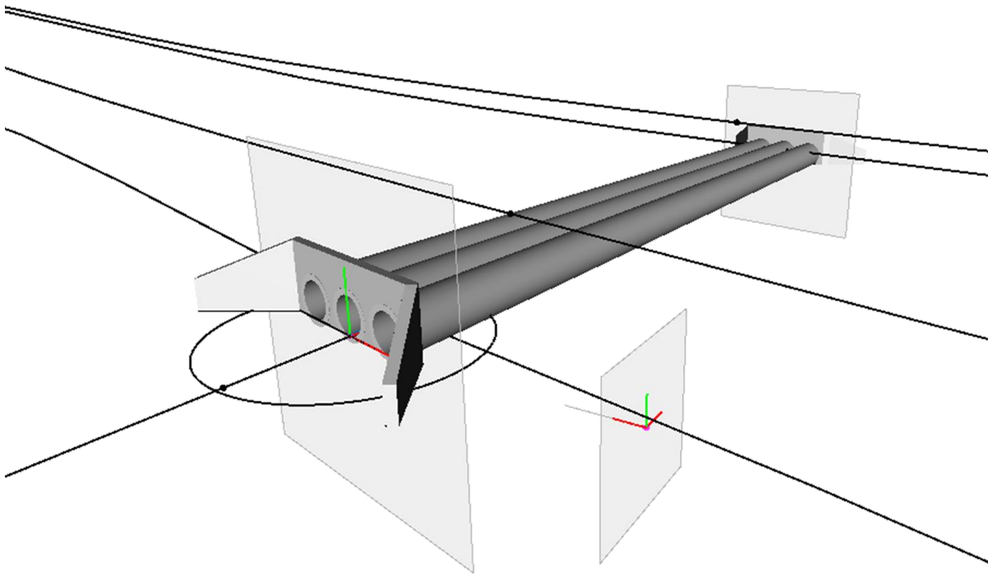
Ignore this warning

Dynamo | Transform to Origin

From this



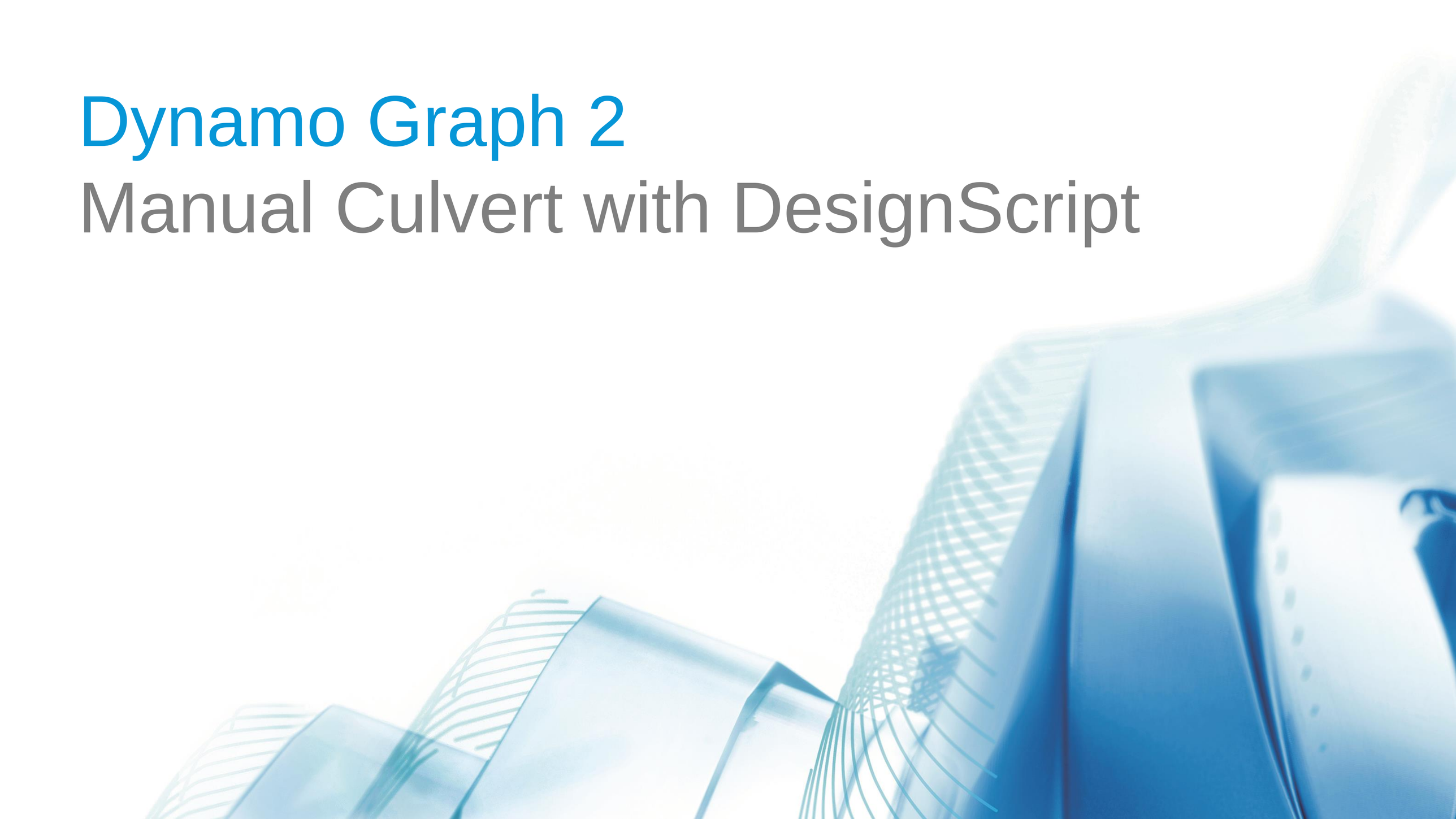
To this



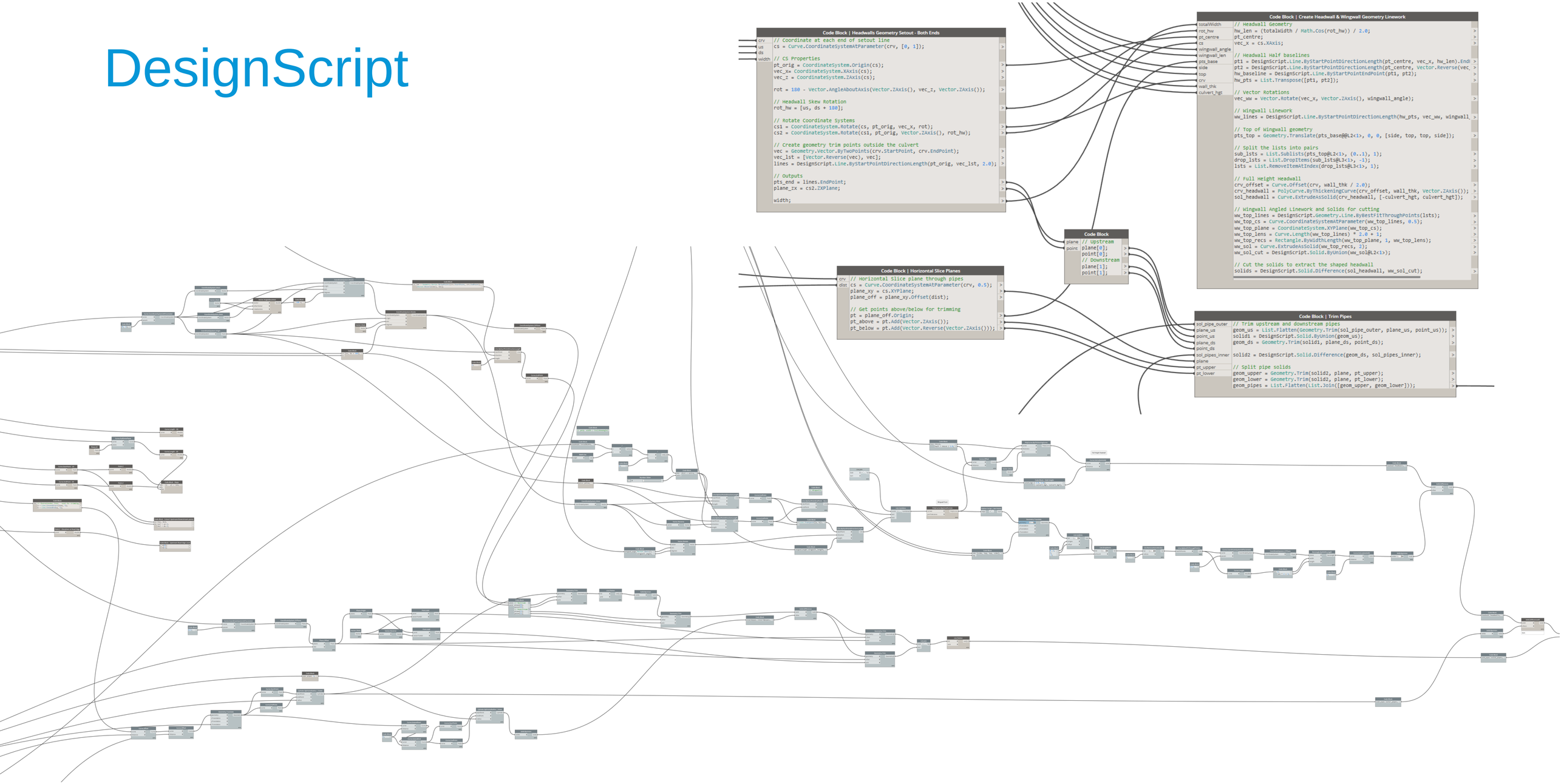
Don’t forget to translate back to the world!

Dynamo Graph 2

Manual Culvert with DesignScript

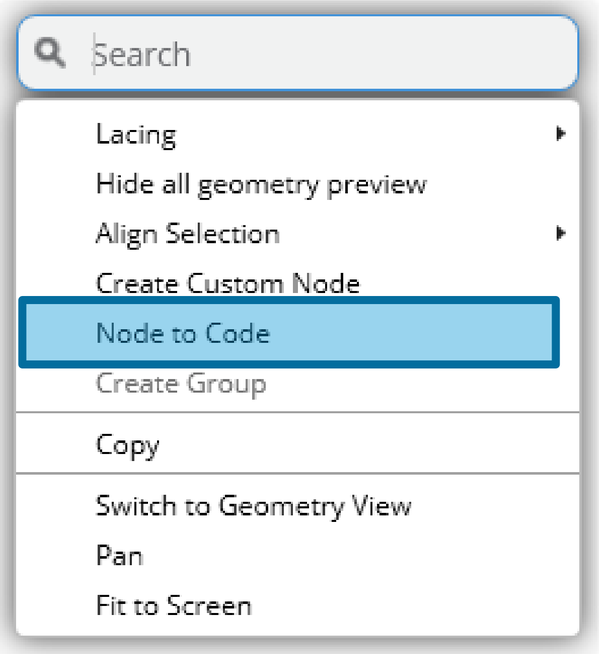


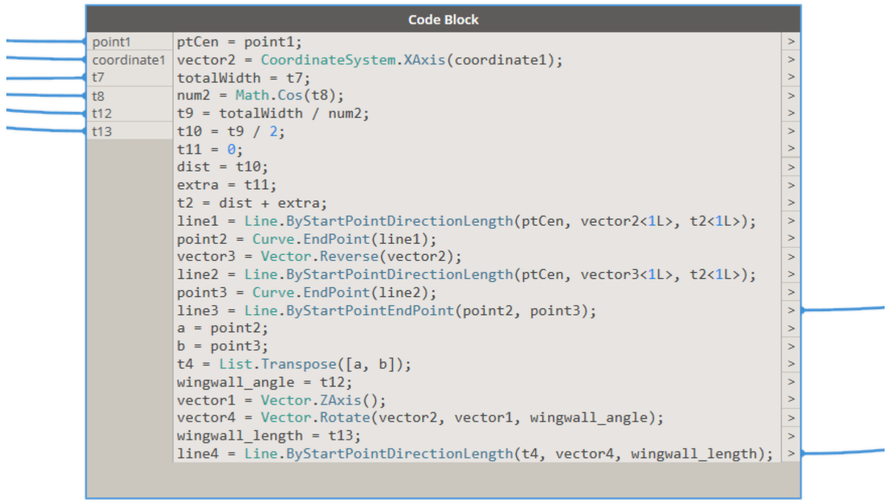
DesignScript

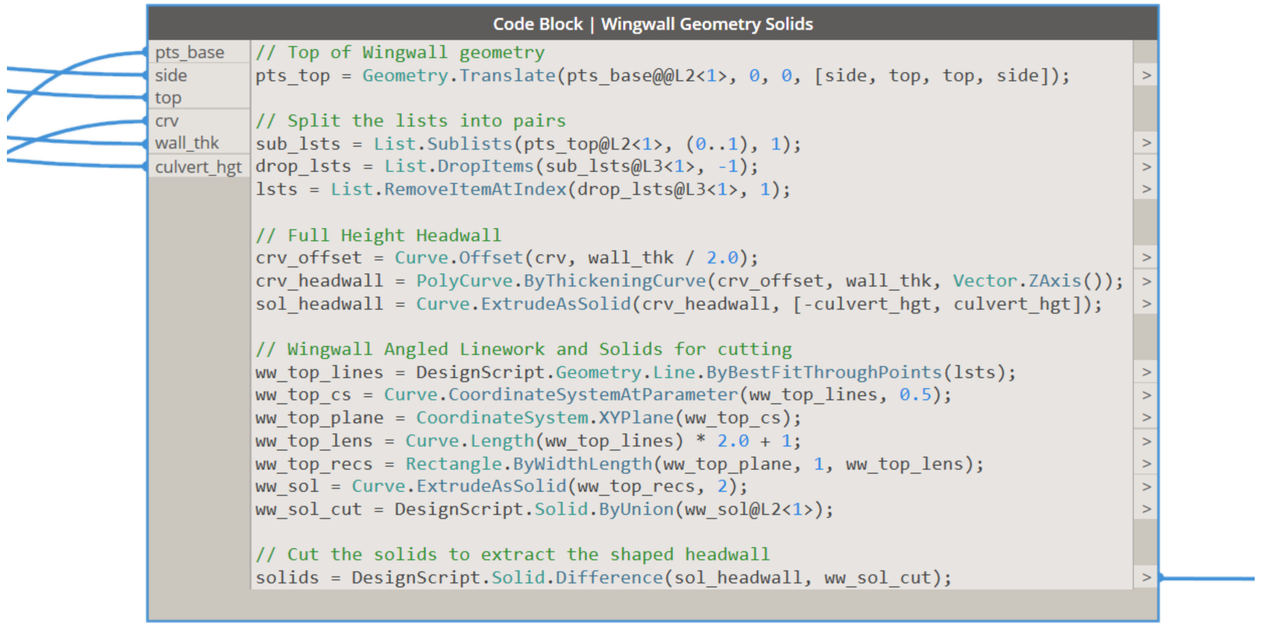


Node To Code

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

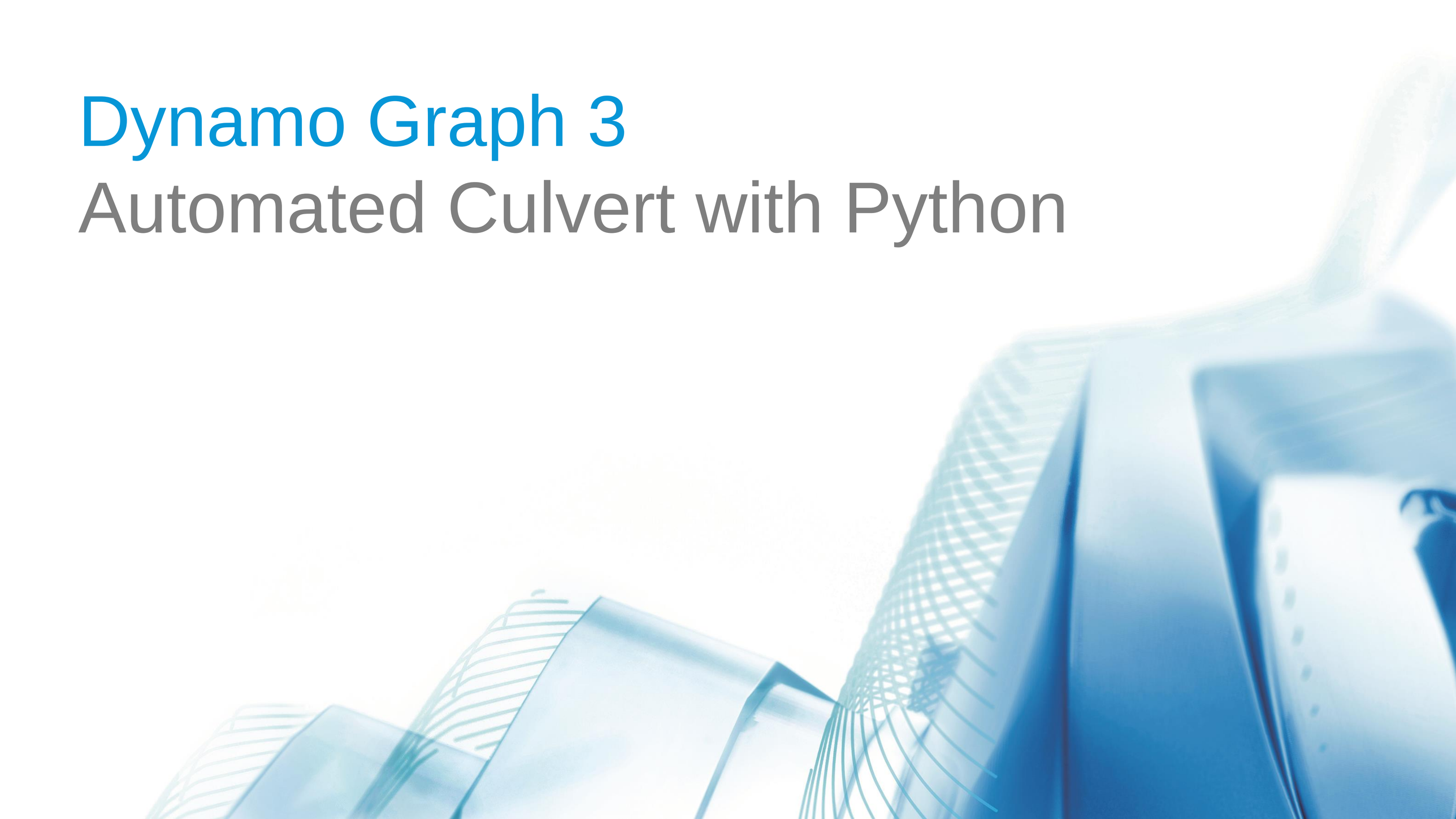






Nodes vs Code Blocks

Dynamo Graph 3
Automated Culvert with Python



Python

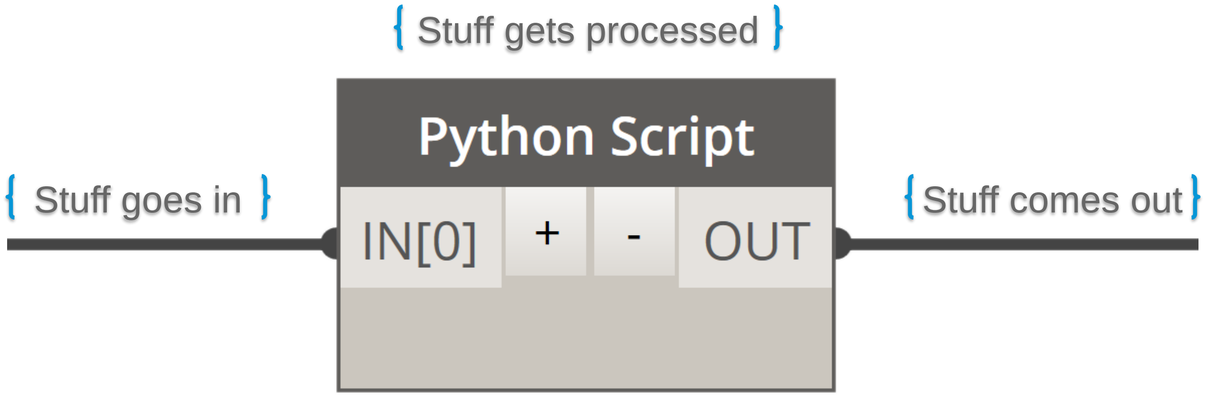
- Legible Scripting Language
- Object-Oriented
- Open Source
- Monty Python

Iron Python

C# implementation of Python language

Microsoft .Net Framework

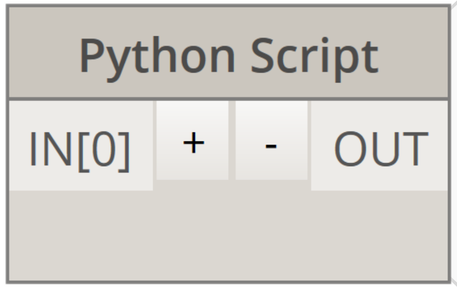
Common Language Runtime (CLR)



Python Node

Simplified Integrated Developer Environment (IDE)

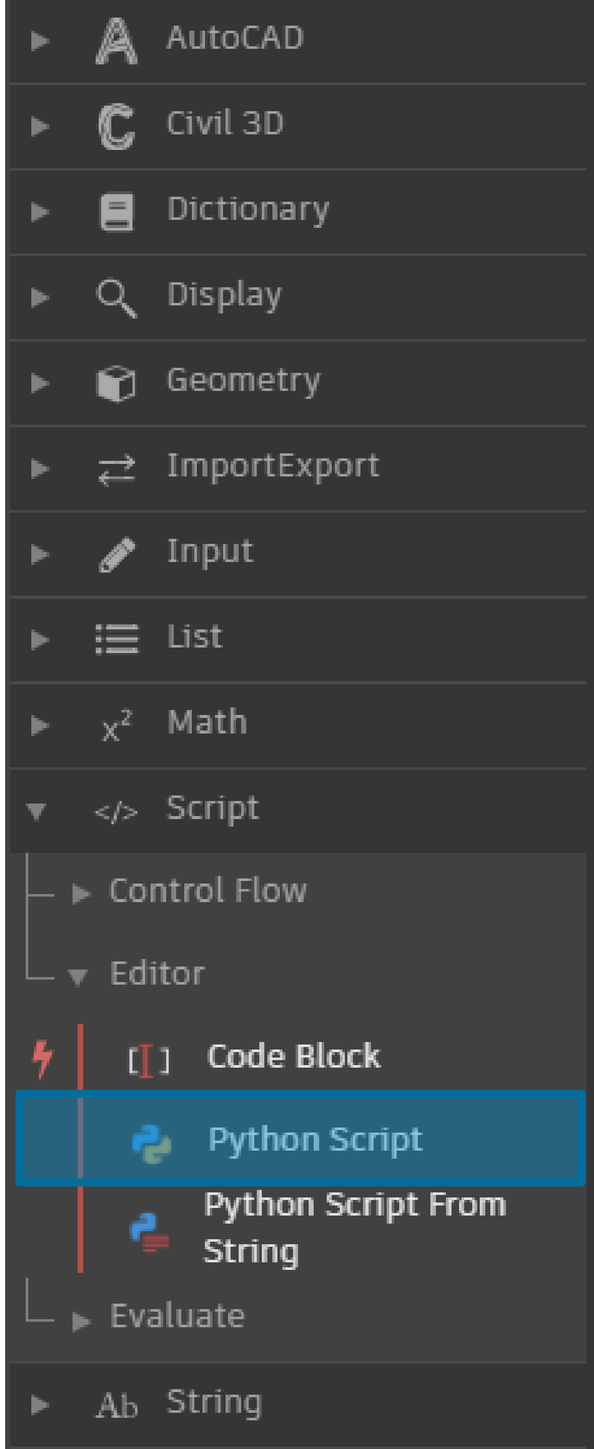
Dynamo Python Node



Python Script

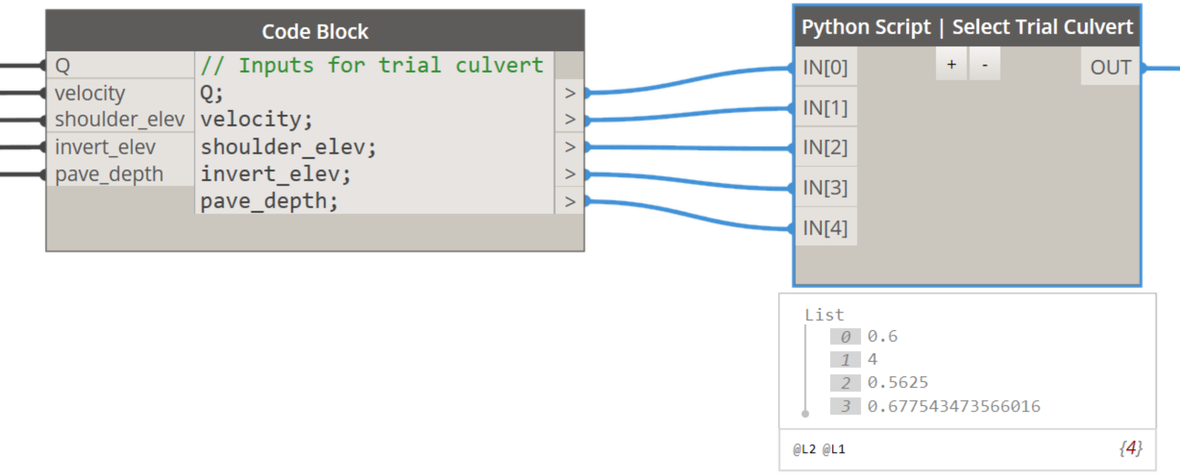
```
# Load the Python Standard and DesignScript Libraries
import sys
import clr

# Add Assemblies for AutoCAD and Civil3D
clr.AddReference('AcMgd')
clr.AddReference('AcCoreMgd')
clr.AddReference('AcDbMgd')
clr.AddReference('AecBaseMgd')
clr.AddReference('AecPropDataMgd')
clr.AddReference('AeccDbMgd')

# Import references from AutoCAD
from Autodesk.AutoCAD.Runtime import *
from Autodesk.AutoCAD.ApplicationServices import *
from Autodesk.AutoCAD.EditorInput import *
from Autodesk.AutoCAD.DatabaseServices import *
from Autodesk.AutoCAD.Geometry import *

# Import references from Civil3D
from Autodesk.Civil.ApplicationServices import *
from Autodesk.Civil.DatabaseServices import *

# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN

adoc = Application.DocumentManager.MdiActiveDocument
editor = adoc.Editor

with adoc.LockDocument():
    with adoc.Database as db:

        with db.TransactionManager.StartTransaction() as t:
            # Place your code below
            #
            #

            # Commit before end transaction
            #t.Commit()
            pass

# Assign your output to the OUT variable.
OUT = 0
```

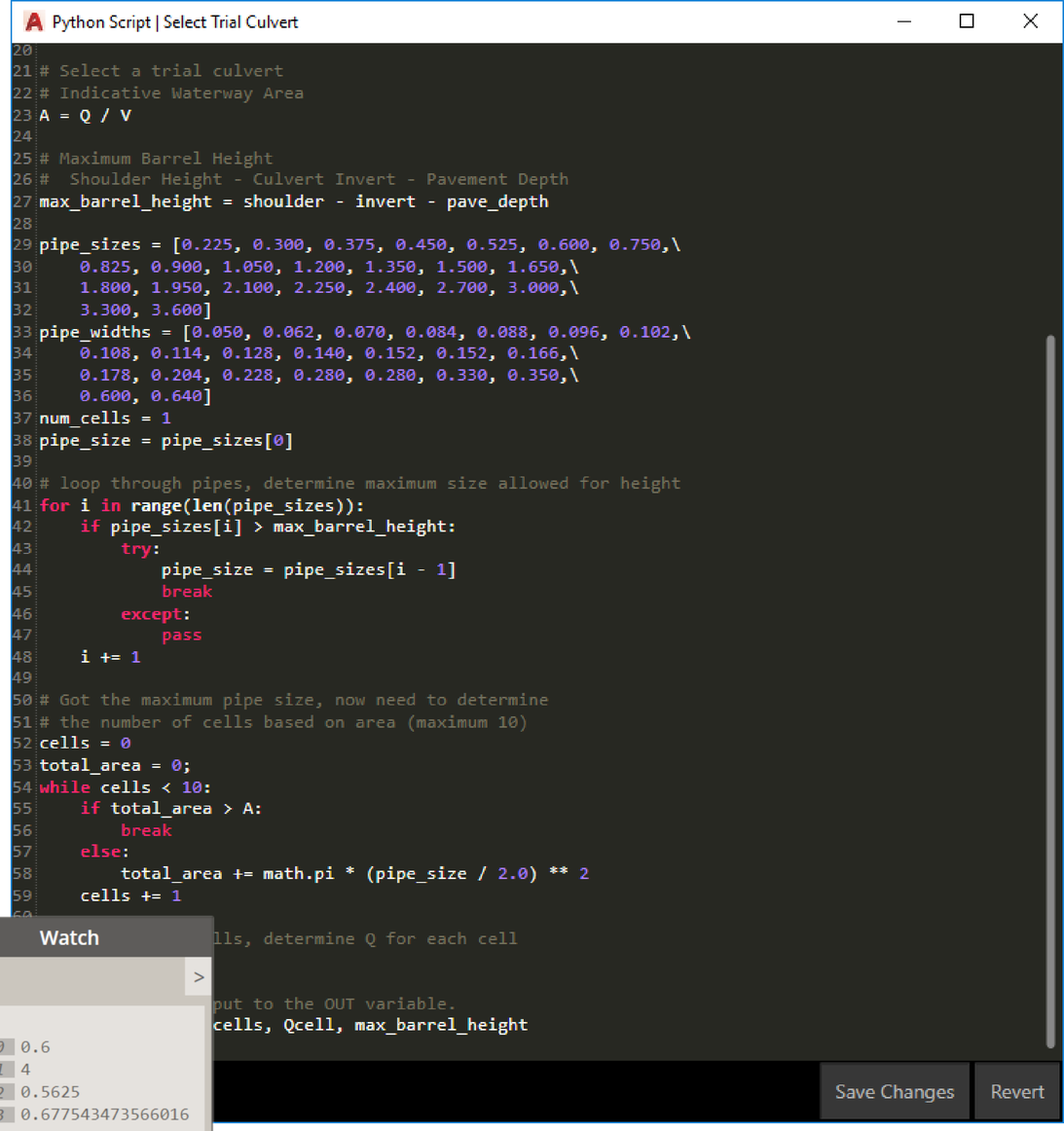
Run | Save Changes | Revert



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;
```

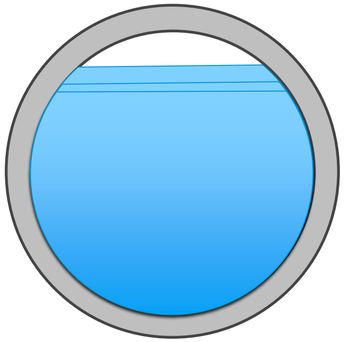
Code Block

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] IN[1] IN[2] IN[3] IN[4] IN[5] IN[6] IN[7] → OUT

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

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] IN[1] IN[2] IN[3] IN[4] IN[5] IN[6] IN[7] → OUT

Python Script | Outlet Control Headwater Depth

```python
# Establish flow depth at outlet
# Setup a loop, starting with the half depth of pipe/culvert
# Maximum iterations = 15
i = 0
tolerance = 0.005
diff = 100
temp = []

depth = radius
depth_half = depth # initialise half depth to begin

while i < 15 or diff < tolerance:
    depth_from_centre = abs(radius - depth)
    top_surface_width = 2 * math.sqrt(radius ** 2 - abs(radius - depth) ** 2)

    if depth > radius:
        central_angle = 360 - math.acos(depth_from_centre / radius) * (180 / math.pi) * 2.0
    else:
        central_angle = math.acos(depth_from_centre / radius) * (180 / math.pi) * 2.0

    wetted_area = ((radius ** 2) / 2) * (((math.pi / 180) * central_angle) - \
                  math.sin(math.radians(central_angle)))
    wetted_perimeter = 2.0 * math.pi * radius * (central_angle / 360)
    hydraulic_radius = wetted_area / wetted_perimeter

    # setup check conditions -
    #  Si Units = Q / AD ** 0.5
    #  Si Units = Ap / A * (g * yh / D) ** 0.5
    Ap = wetted_area
    si1 =  Q / (Afull * (D ** 0.5))
    si2 = (Ap / Afull) * ((9.8 * (depth / D)) ** 0.5)

    diff = abs(si1 - si2)

    if diff < tolerance:
        break

    depth_half = depth_half / 2
    if si2 >= si1:
        depth = depth - depth_half  # Lower the depth
    else:
        depth = depth + depth_half  # Increase the depth

    i += 1
```

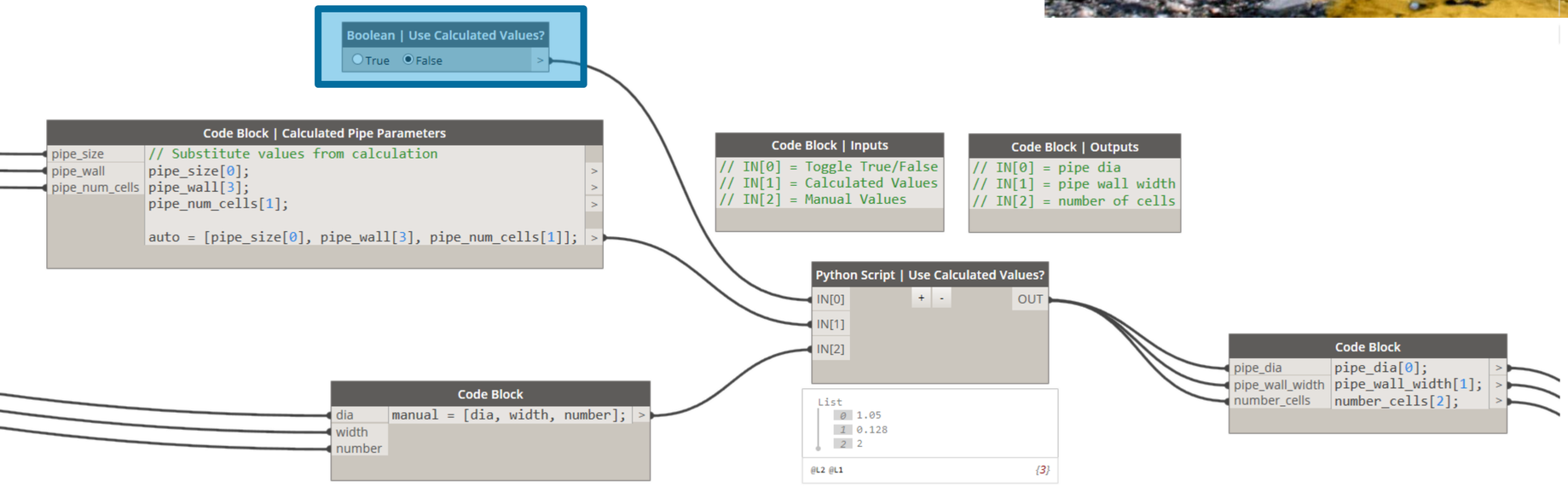
Run | Save Changes | Revert

Automatic or Manual ?

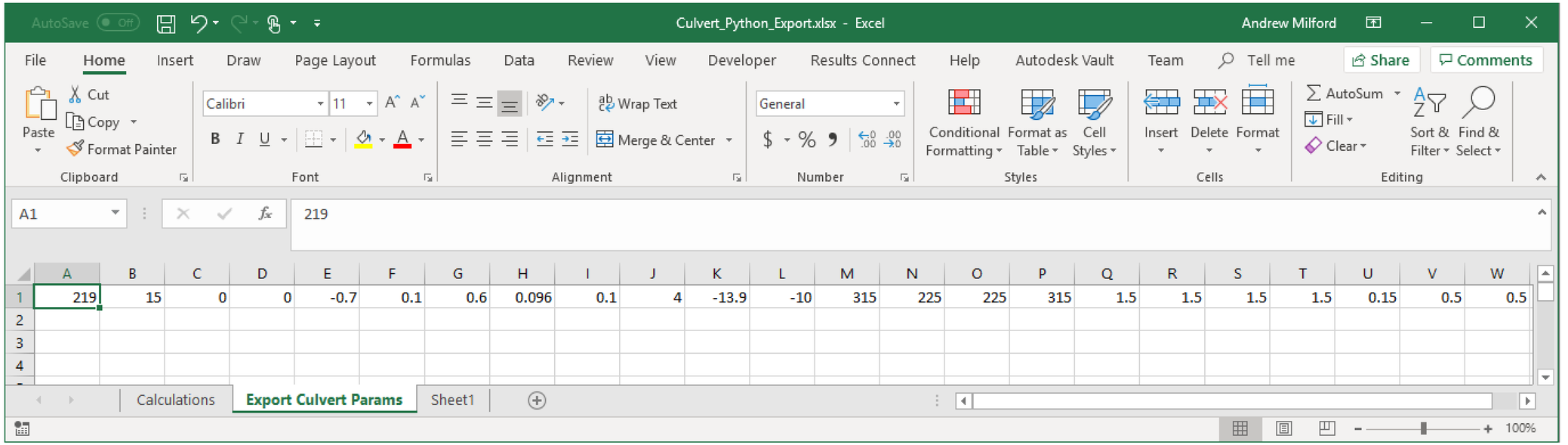
Python Script | Use Calculated Values?

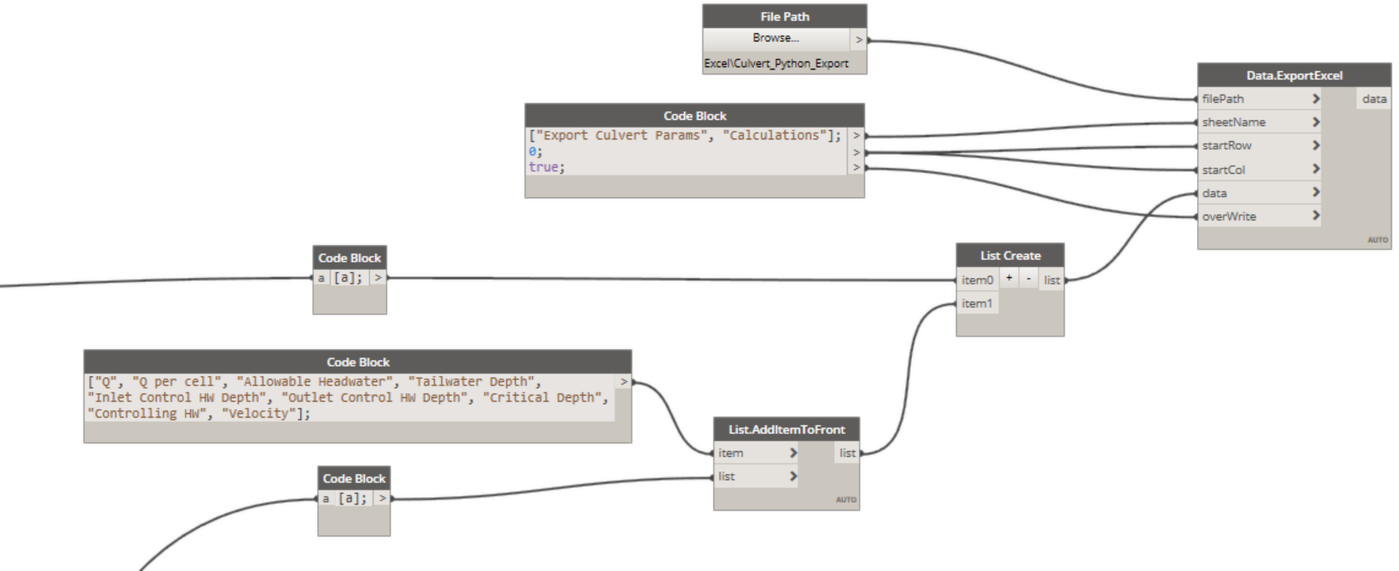
```
# Load the Python Standard and DesignScript Libraries
import clr

#Inputs
toggle = IN[0]
auto = IN[1]
manual = IN[2]

# Lists
pipe_size = manual[0]
wall_width = manual[1]
number_cells = manual[2]

if toggle:
    pipe_size = auto[0]
    wall_width = auto[1]
    number_cells = auto[2]

# Assign your output to the OUT variable.
OUT = pipe_size, wall_width, number_cells
```

Run | Save Changes | Revert



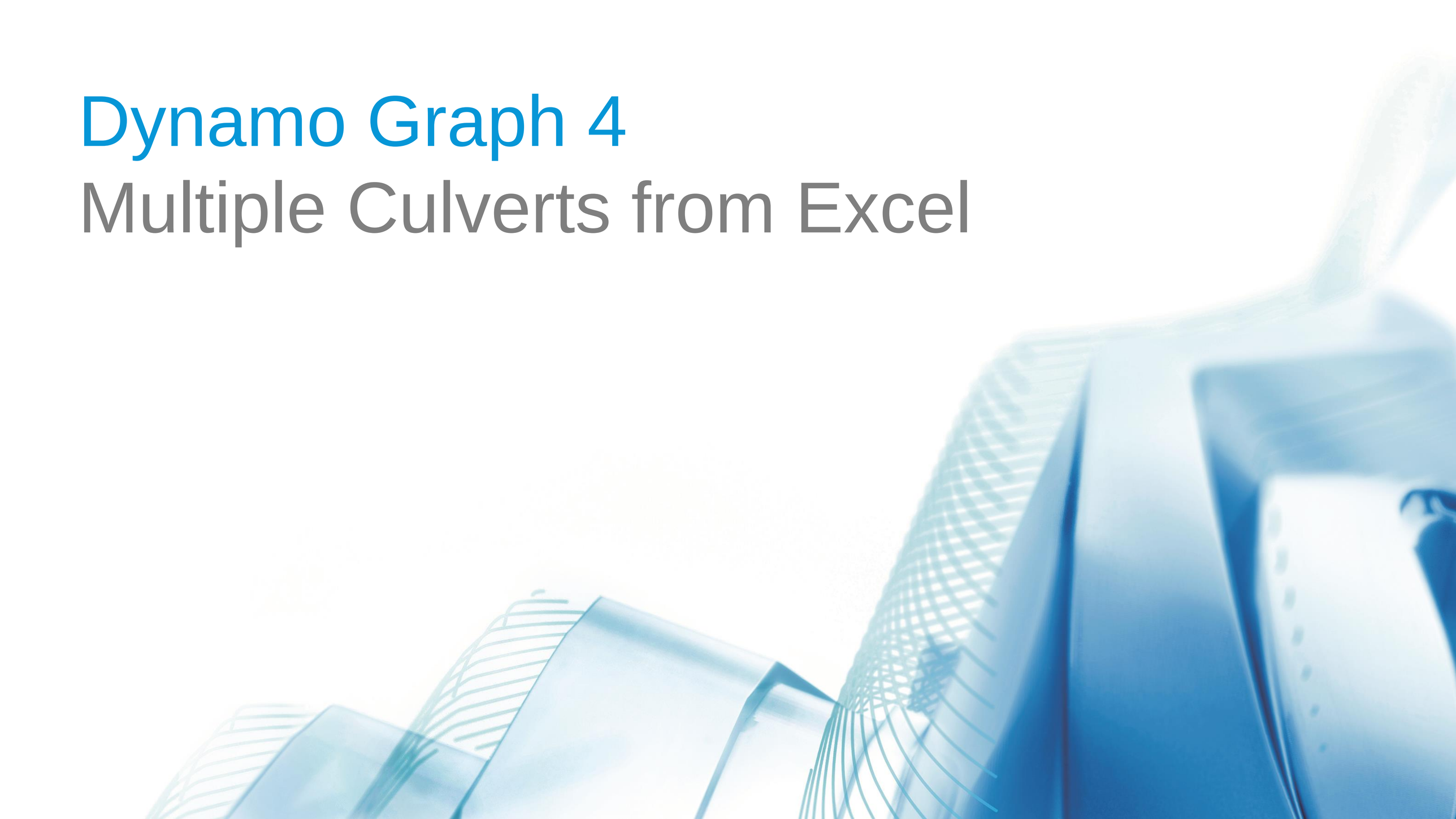
Export Values to Excel



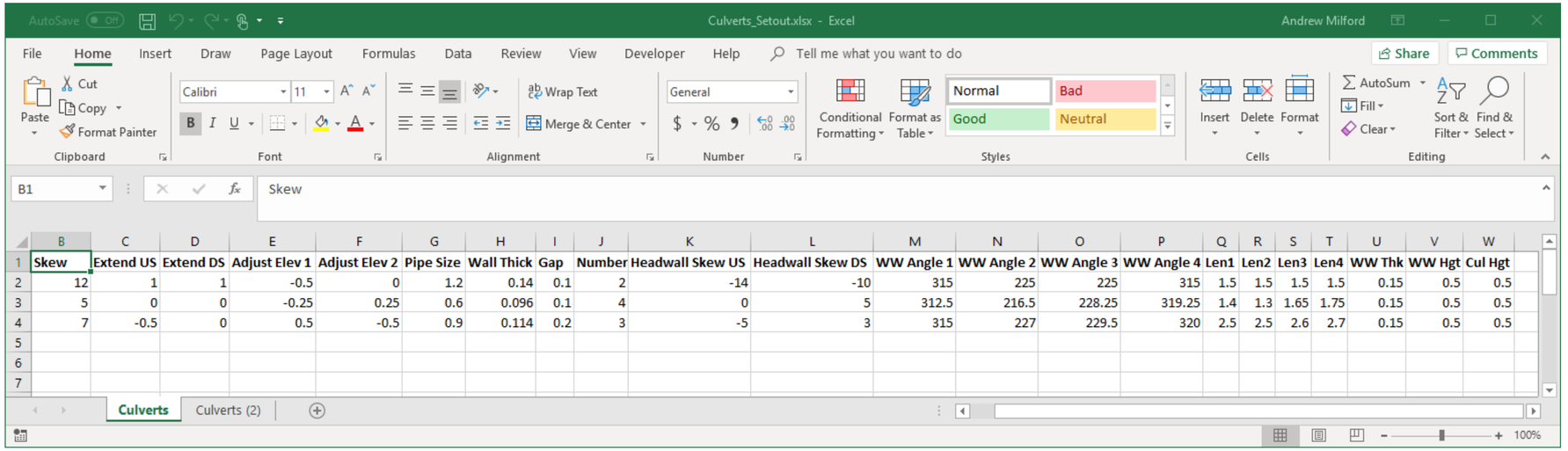


Dynamo Graph 4

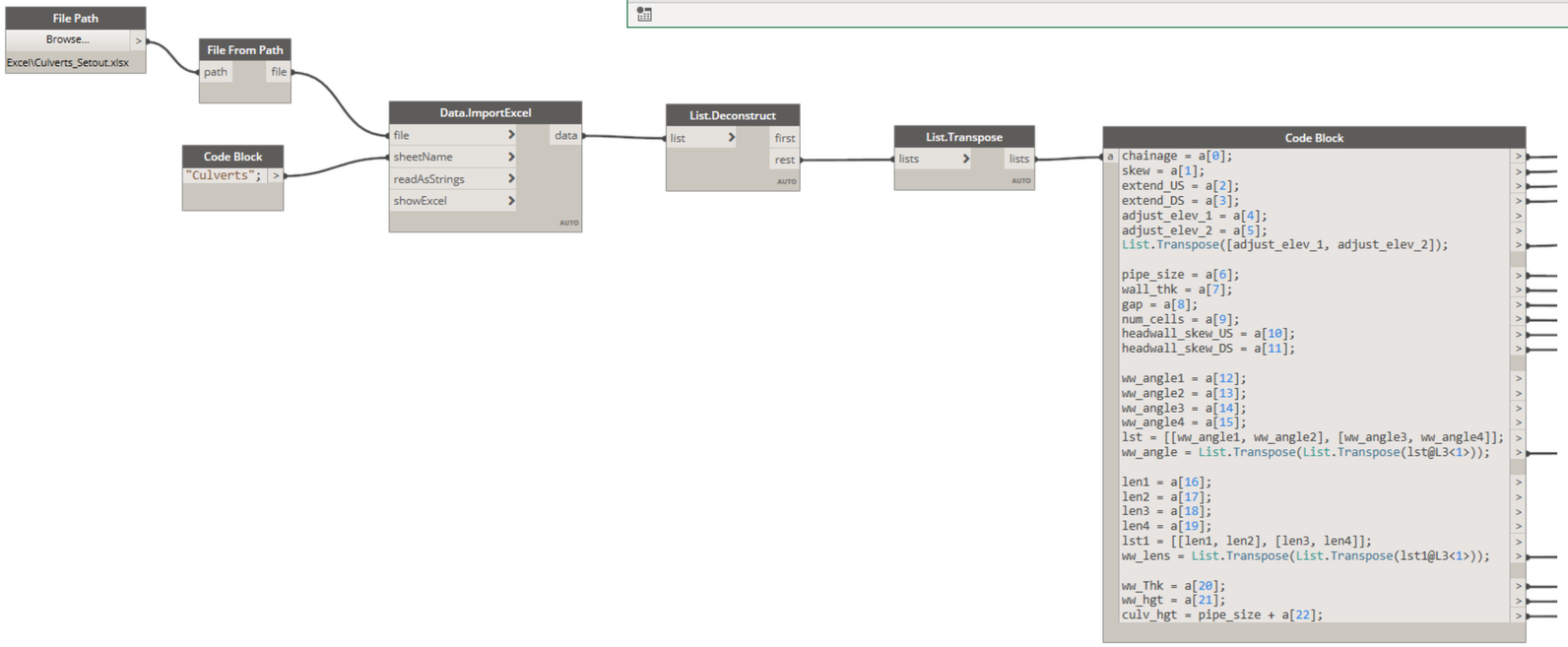
Multiple Culverts from Excel

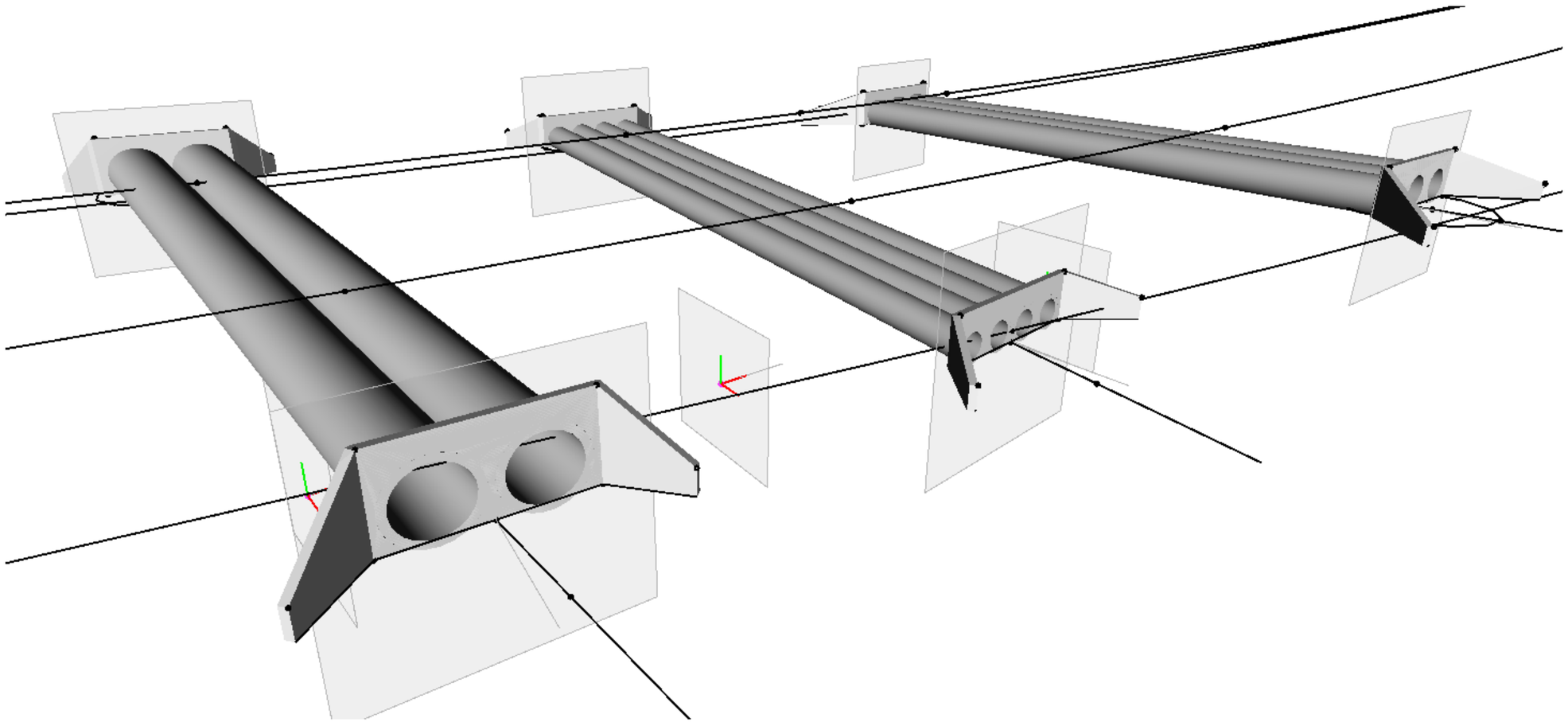


Import Culvert Parameters from Excel

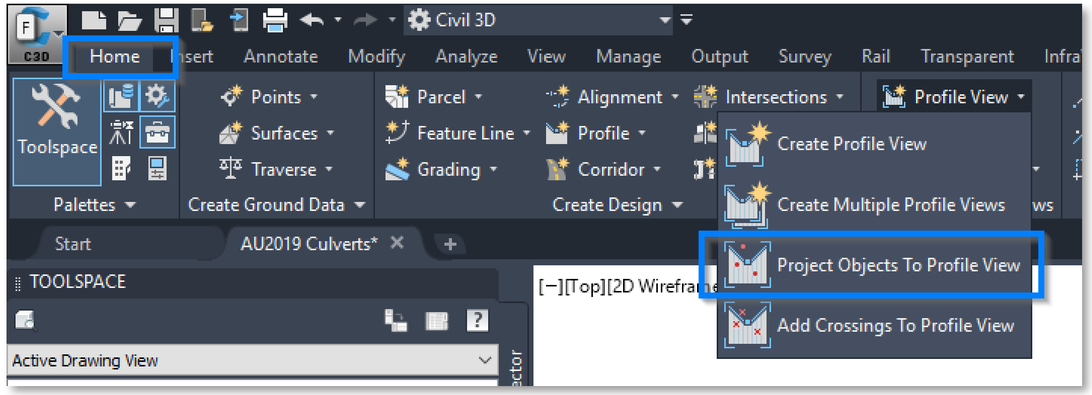


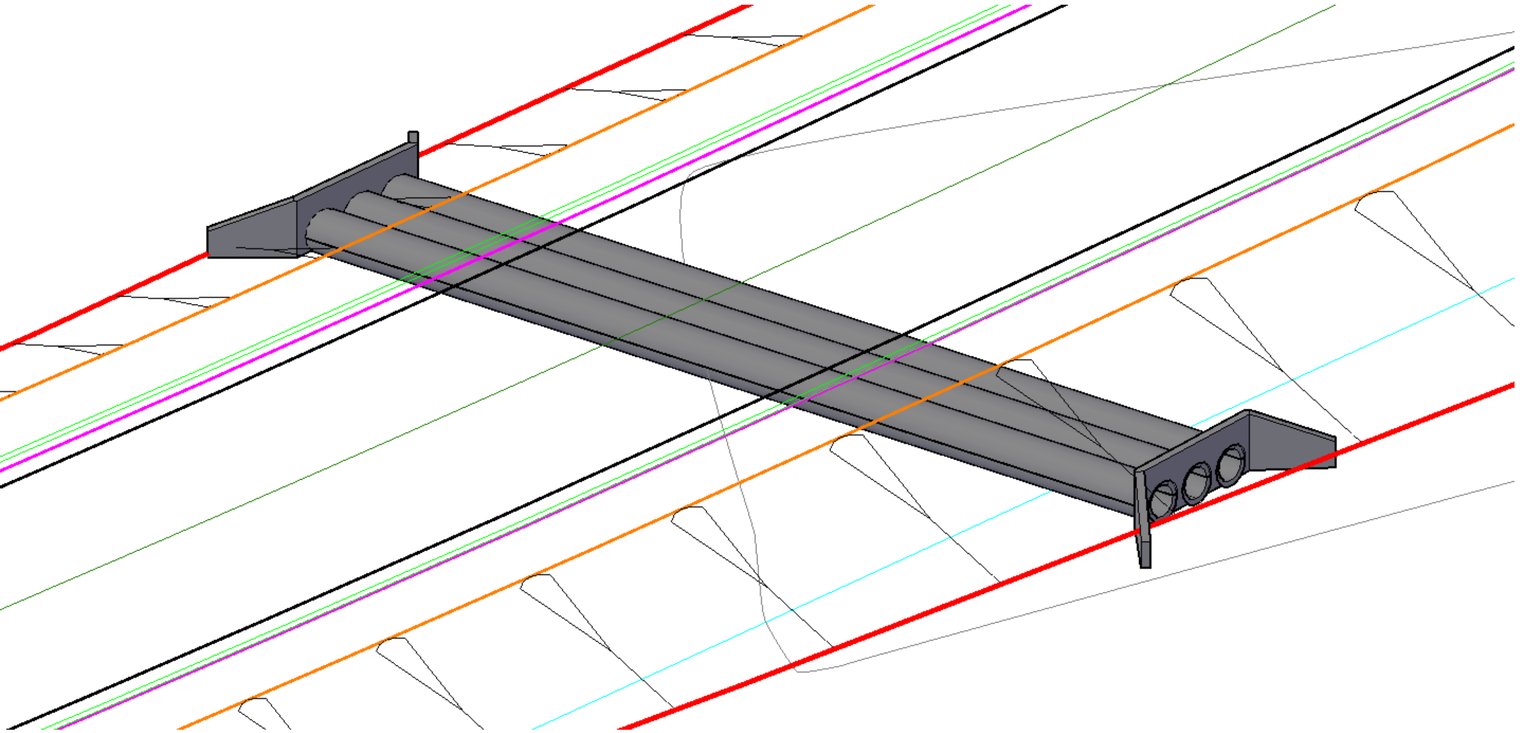
	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Skew	Extend US	Extend DS	Adjust Elev 1	Adjust Elev 2	Pipe Size	Wall Thick	Gap	Number	Headwall Skew US	Headwall Skew DS	WW Angle 1	WW Angle 2	WW Angle 3	WW Angle 4	Len1	Len2	Len3	Len4	WW Thk	WW Hgt	Cul Hgt
2	12	1	1	-0.5	0	1.2	0.14	0.1	2	-14	-10	315	225	225	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																						

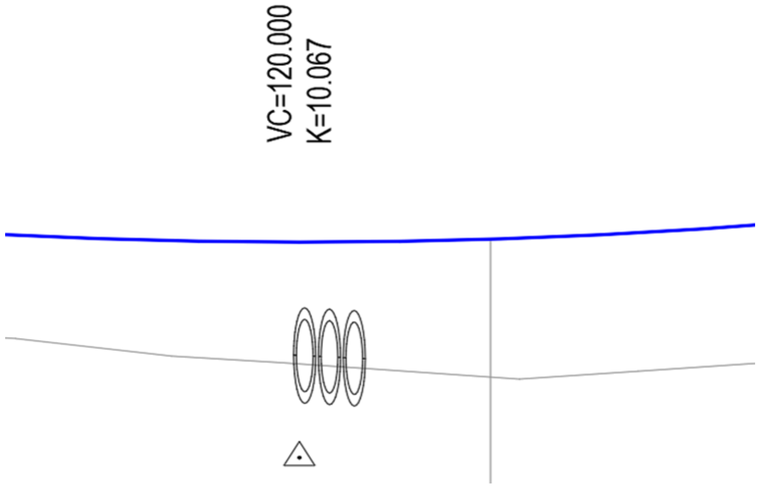




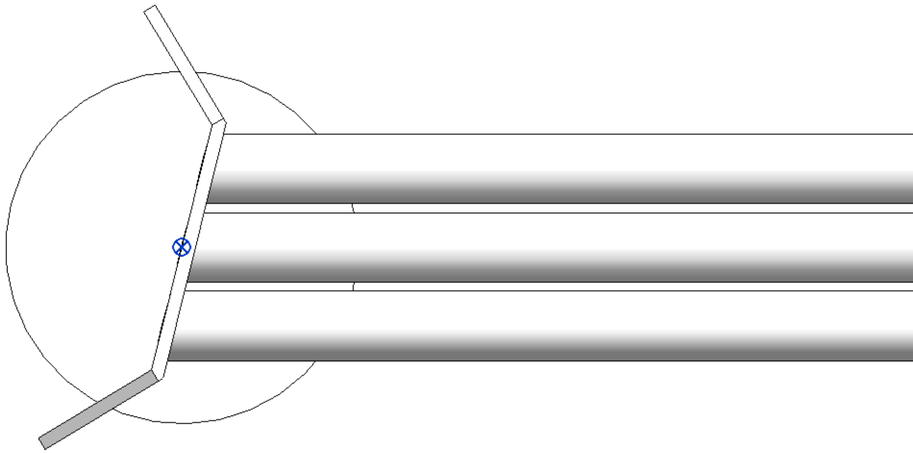
Multiple Culverts

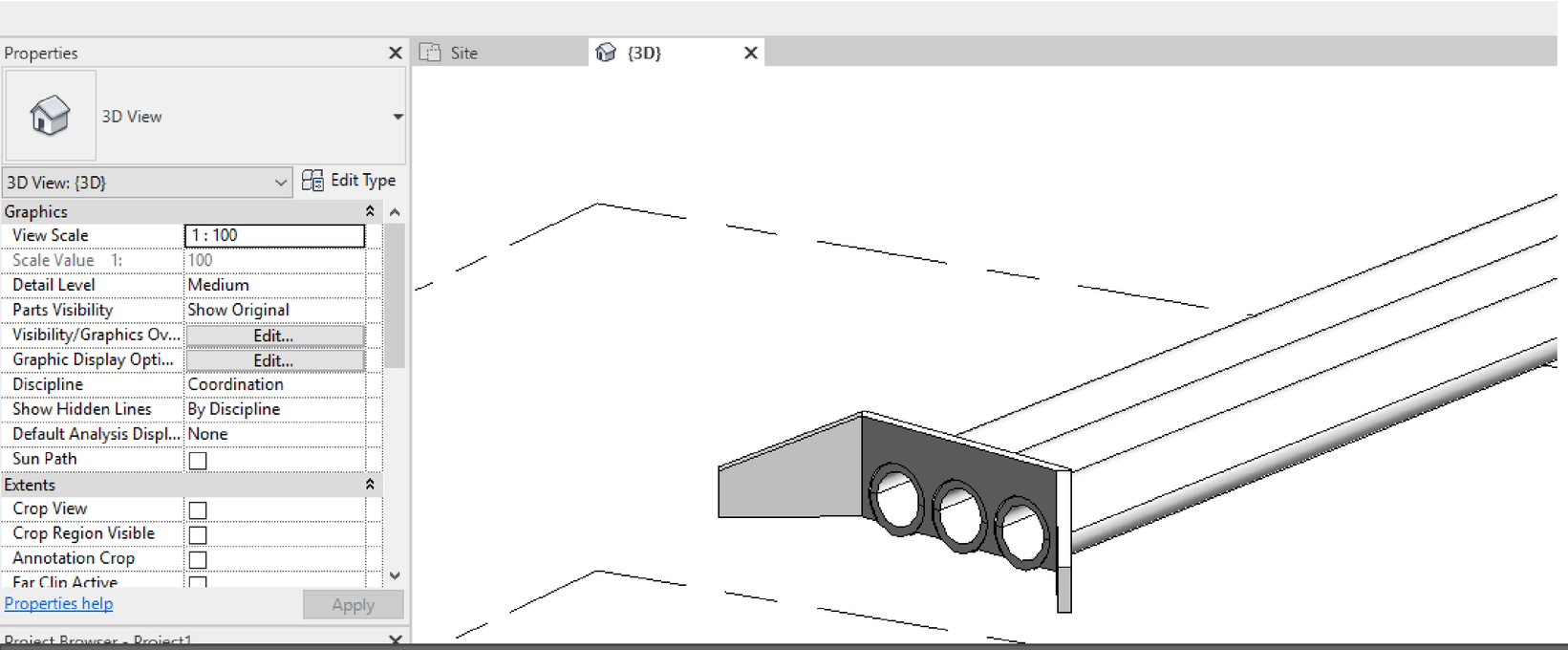






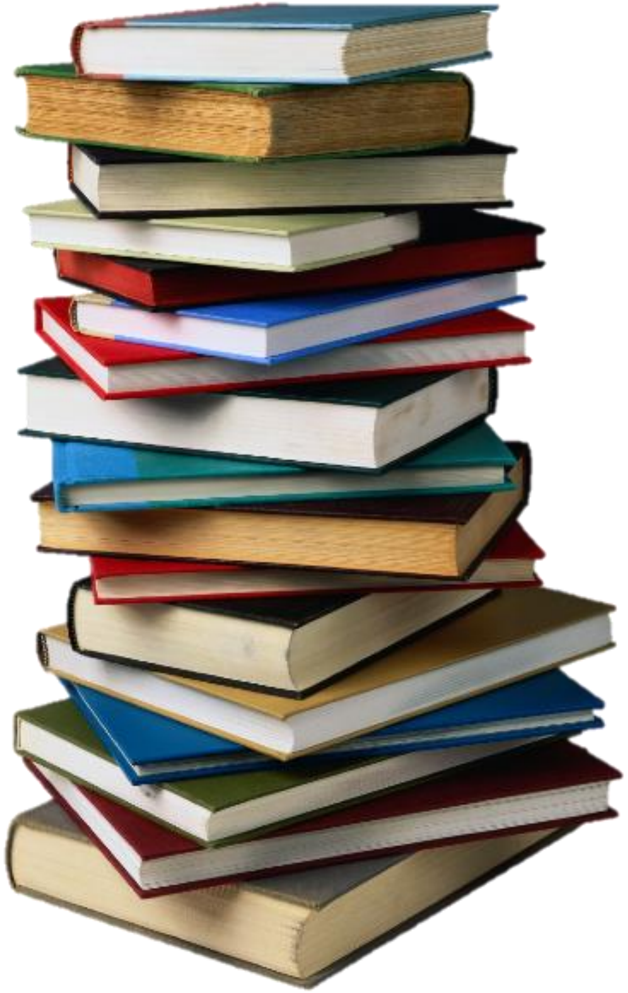
Culverts in Civil 3D





Culverts in Revit

Additional Resources



- DynamoBIM.org
- DynamoPrimer.com
- GitHub/DynamoDS
- Blogs, YouTube videos
- AU lessons and handouts
- LinkedIn Learning / CadLearning / Pluralsight etc

AUTODESK®

Make anything™