Walk-in Slide: AU 2014 Social Media Feed

1. Click on the link below, this will open your web browser

http://aucache.autodesk.com/social/visualization.html

2. Use “Extended Display” to project the website on screen if you plan to work on your computer. Use “Duplicate” to display same image on screen and computer.

Dynamo Hero: Using Revit Scripting Tools to Optimize Real-World Projects

Michael Hudson & Andrea Vaninni

Associate Architectural Scripting Specialist

Twitter: @MikeHudsonArchi

Class summary

This class will present how real-life architectural projects have used the Dynamo visual programming extension to optimize complex design problems within Revit software. The class will give an introduction to optimization and rationalization algorithms, and the attendees will use the Dynamo extension and Python programming language scripts to create façades that adapt to varying design requirements. Attendees will also learn how to generalize the workflow so to apply it to other design problems concerning rationalization and optimization

Class summary

This class will present how real-life architectural projects have used the Dynamo visual programming extension to optimize complex design problems within Revit software. The class will give an introduction to optimization and rationalization algorithms, and the attendees will use the Dynamo extension and Python programming language scripts to create façades that adapt to varying design requirements. Attendees will also learn how to generalize the workflow so to apply it to other design problems concerning rationalization and optimization.

Key learning objectives

At the end of this class, you will be able to:

- Understand new ways to approach complex design problems
- Understand the principles of the Dynamo extensions' visual programming interface
- Learn how to manage the basics of Python scripting
- Learn how to set up a simple optimization algorithm for any specific problem using the Dynamo extension plus Revit software

About your speakers

Michael Hudson

BArch DipArch MArch ARB RIBA

- British Architect
- University Lecturer
- BIM software since 2002
- Autodesk Revit 2009

Andrea Vannini

MArch MSc ARB

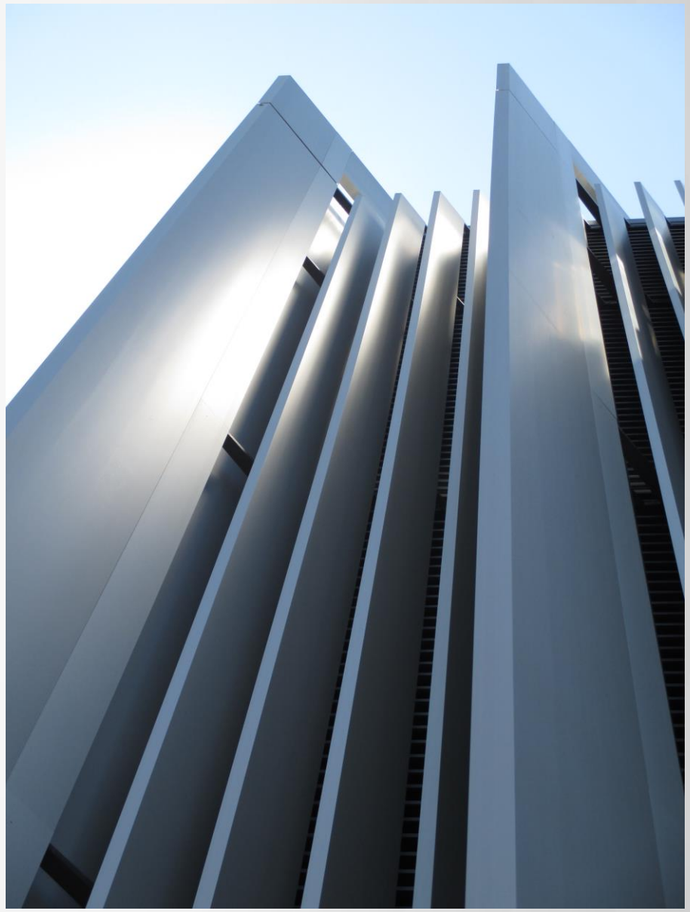
- Italian Architect
- Computational design expert
- Autodesk Revit 2011

Flanagan Lawrence

- Design Architect
- 90 Employees
- RIBA Awarded
- WAF 2012/14
- BD 2014

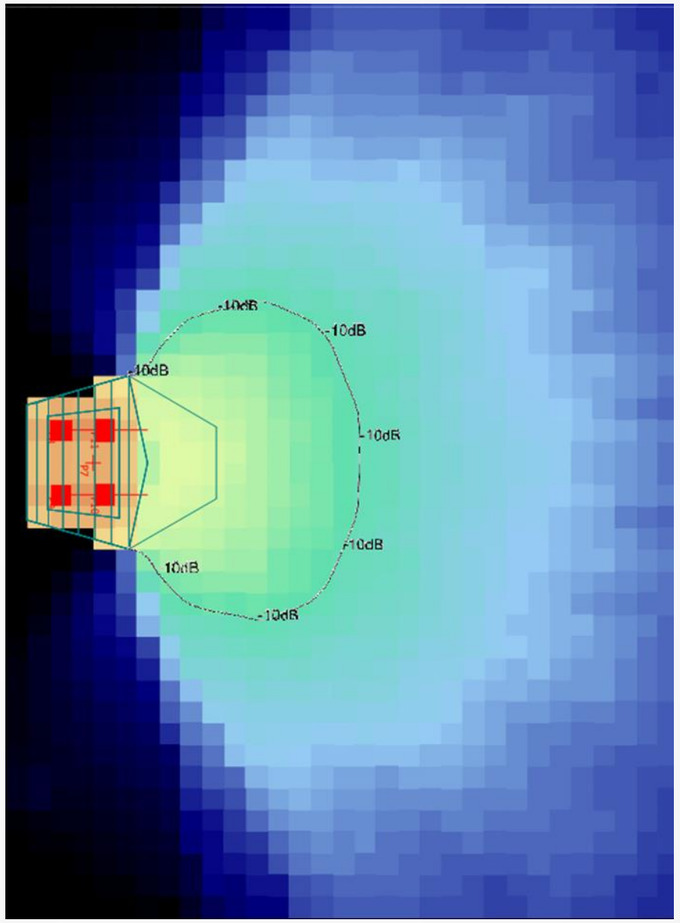
Flanagan Lawrence

- Design Architect
- 90 Employees
- RIBA Awarded
- WAF 2012/14
- BD 2014
- Autodesk Revit
- BIM IPD (Level 2)



Flanagan Lawrence

- Design Architect
- 90 Employees
- RIBA Awarded
- WAF 2012/14
- BD 2014
- Autodesk Revit
- BIM IPD (Level 2)
- Acoustic design
- Digital fabrication



Flanagan Lawrence

- Design Architect
- 90 Employees
- RIBA Awarded
- WAF 2012/14
- BD 2014
- Autodesk Revit
- BIM IPD (Level 2)
- Acoustic design
- Digital fabrication
- Heritage building
- Point clouds

Introduction

The 2-faces of Architecture

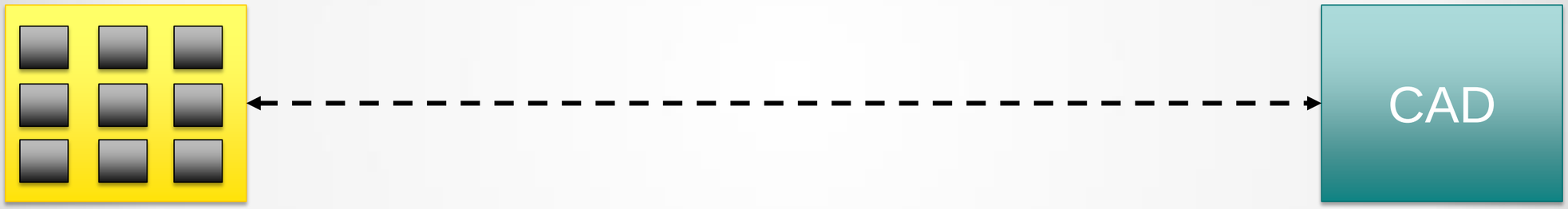
Sculptural

Everyone else

The 2-toolsets within Architecture

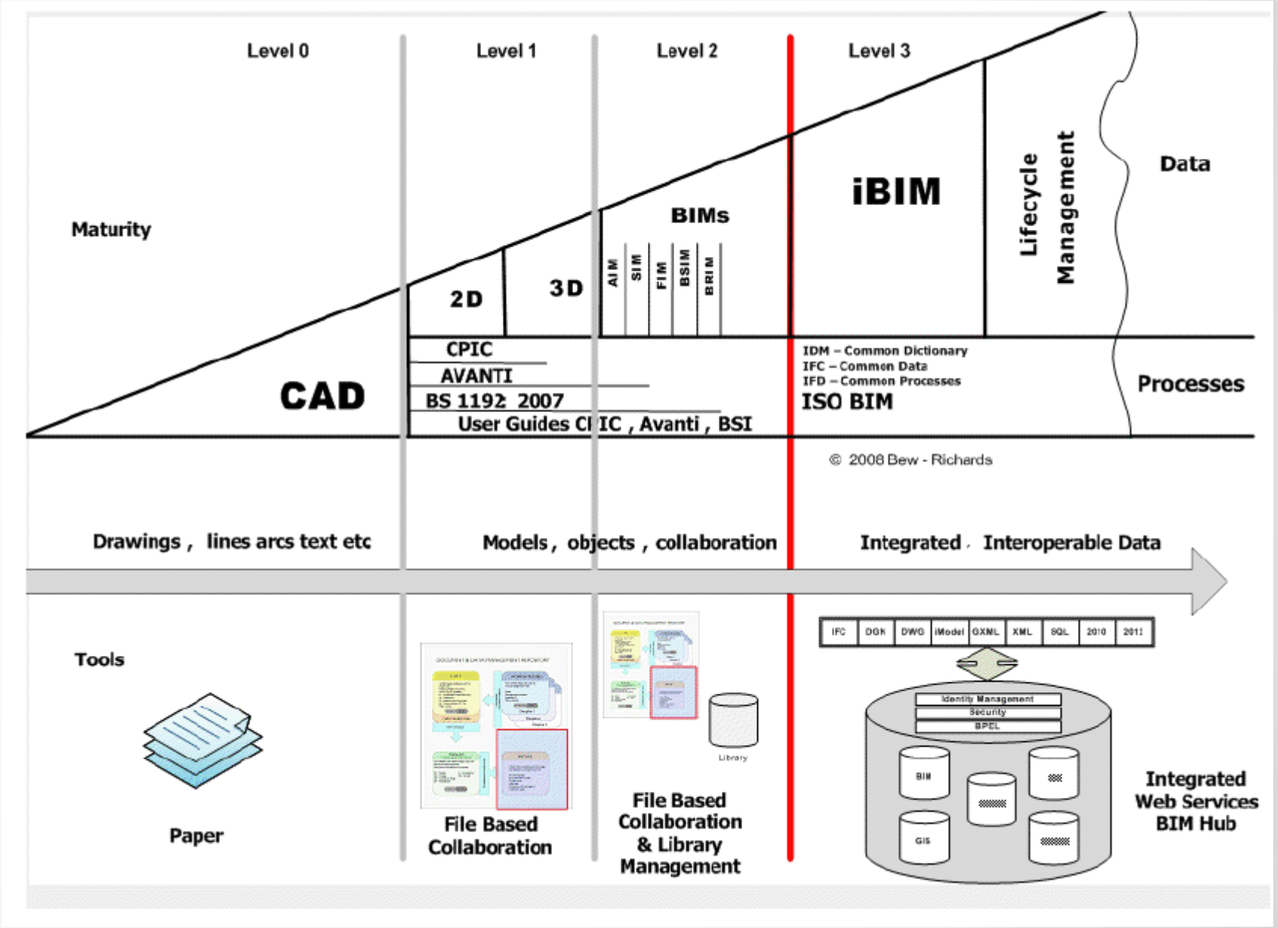
Highly complex

Digital Drafting

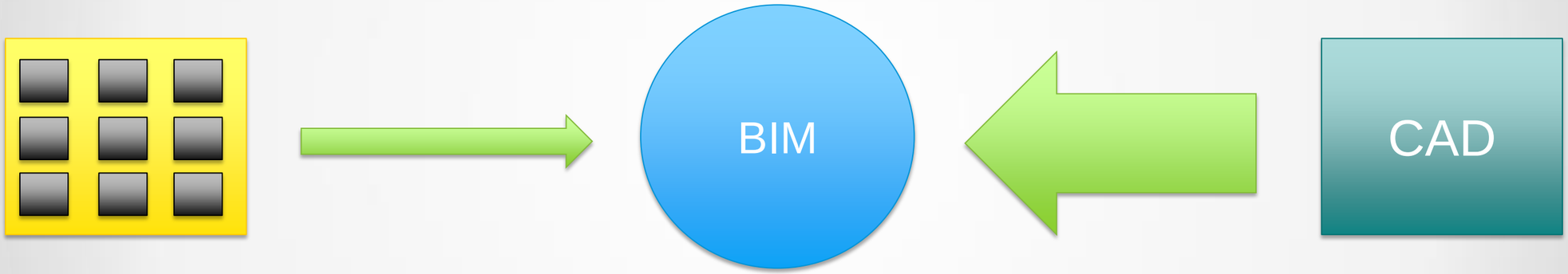


BIM

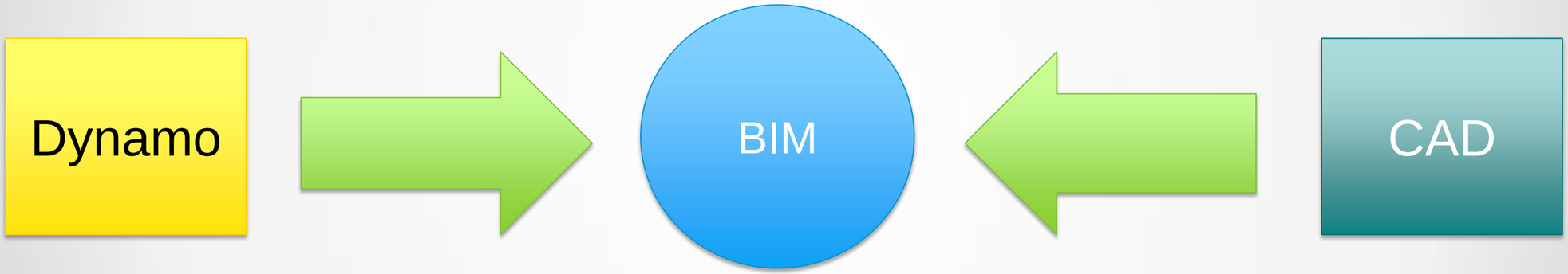
- Global recession 2008
- UK Government 15-20 % more efficient
- Level 2 BIM IPD by 2016



BIM



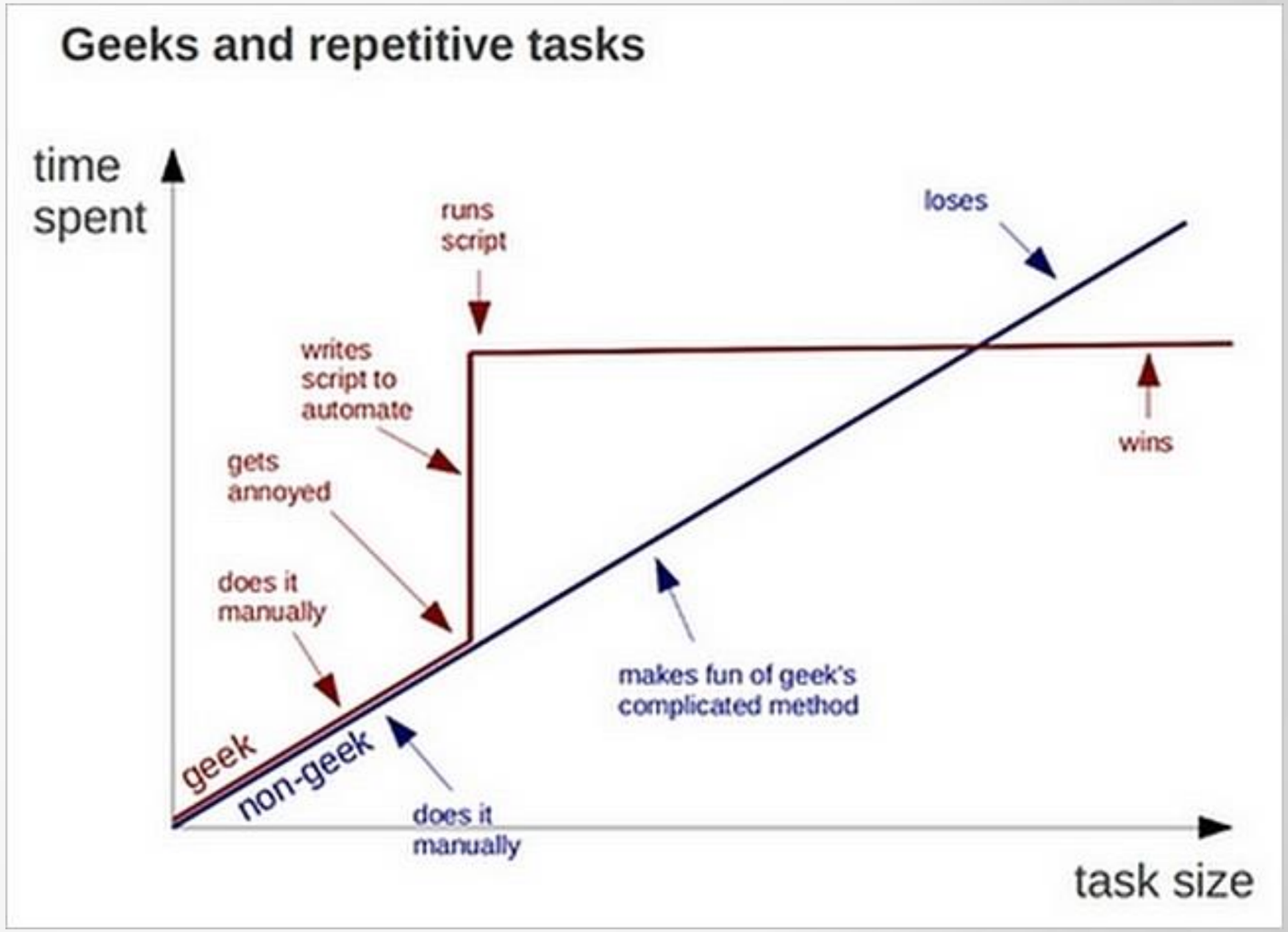
DynamoBIM



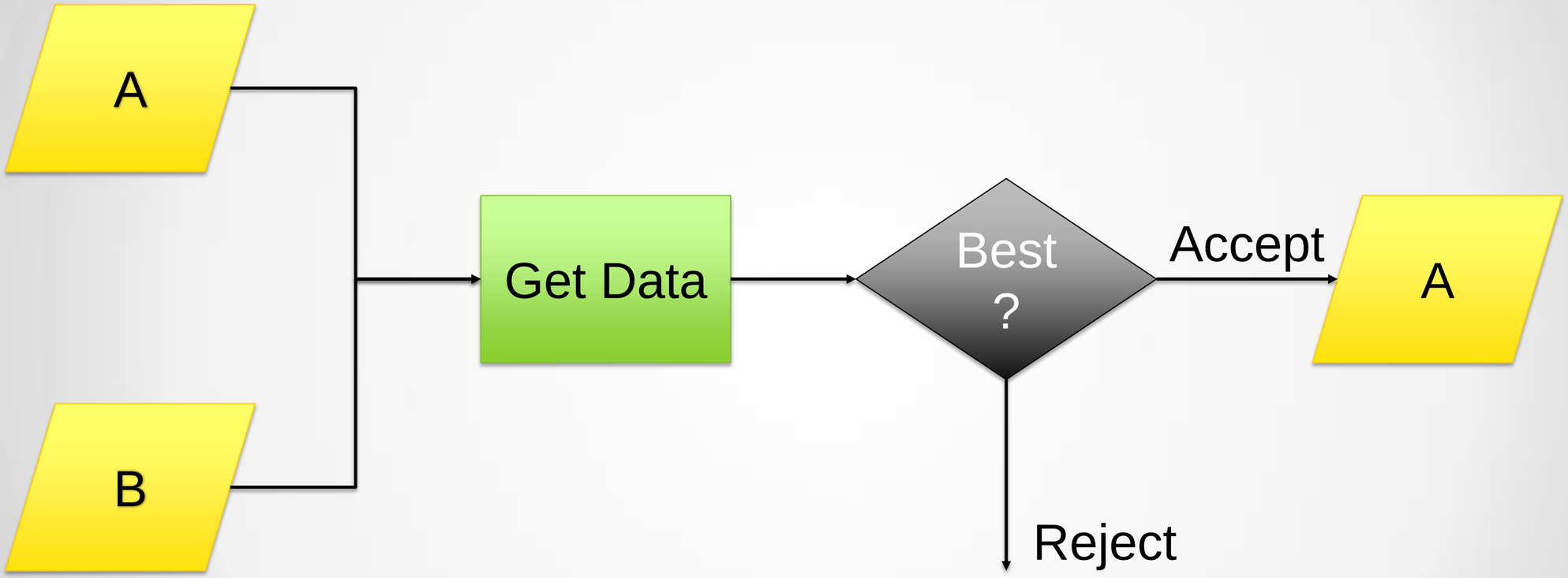
Programing

Why do we need coding skills?

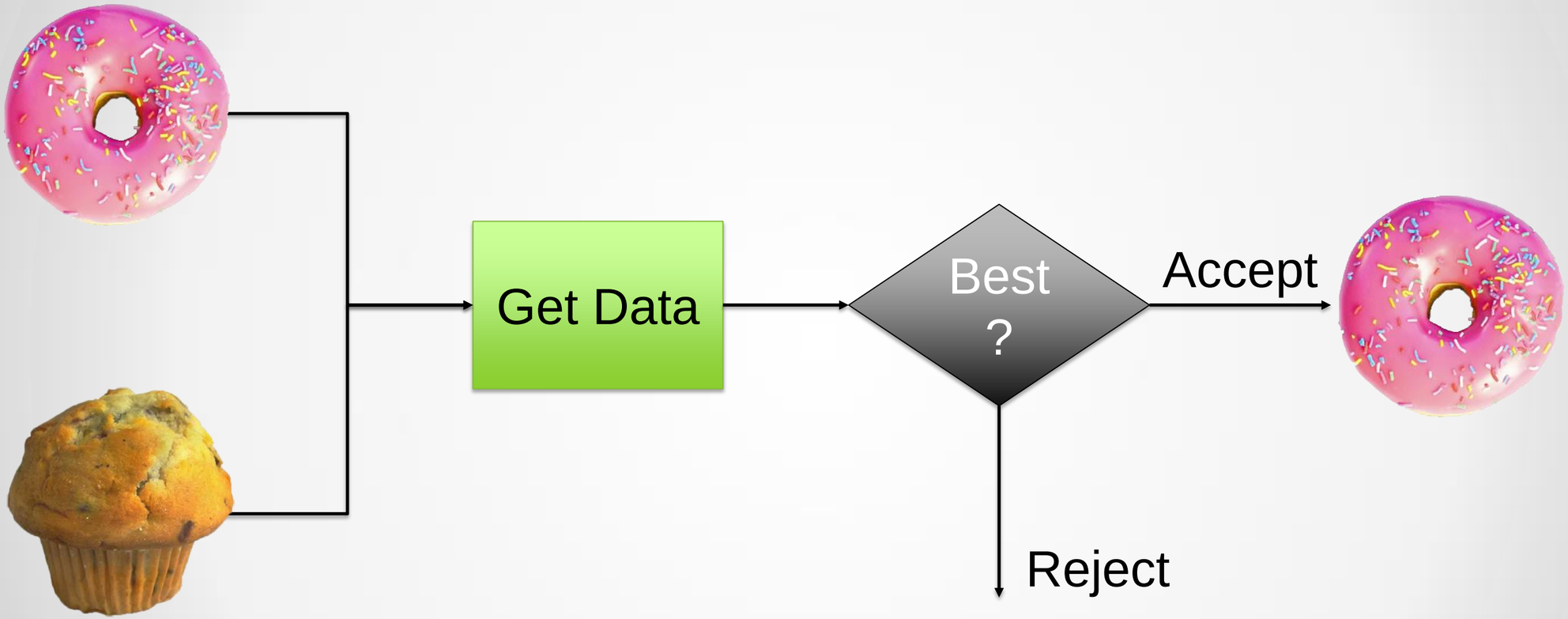
- Lots of repetitive tasks
- Managing data
- Optimization
- Developing the question



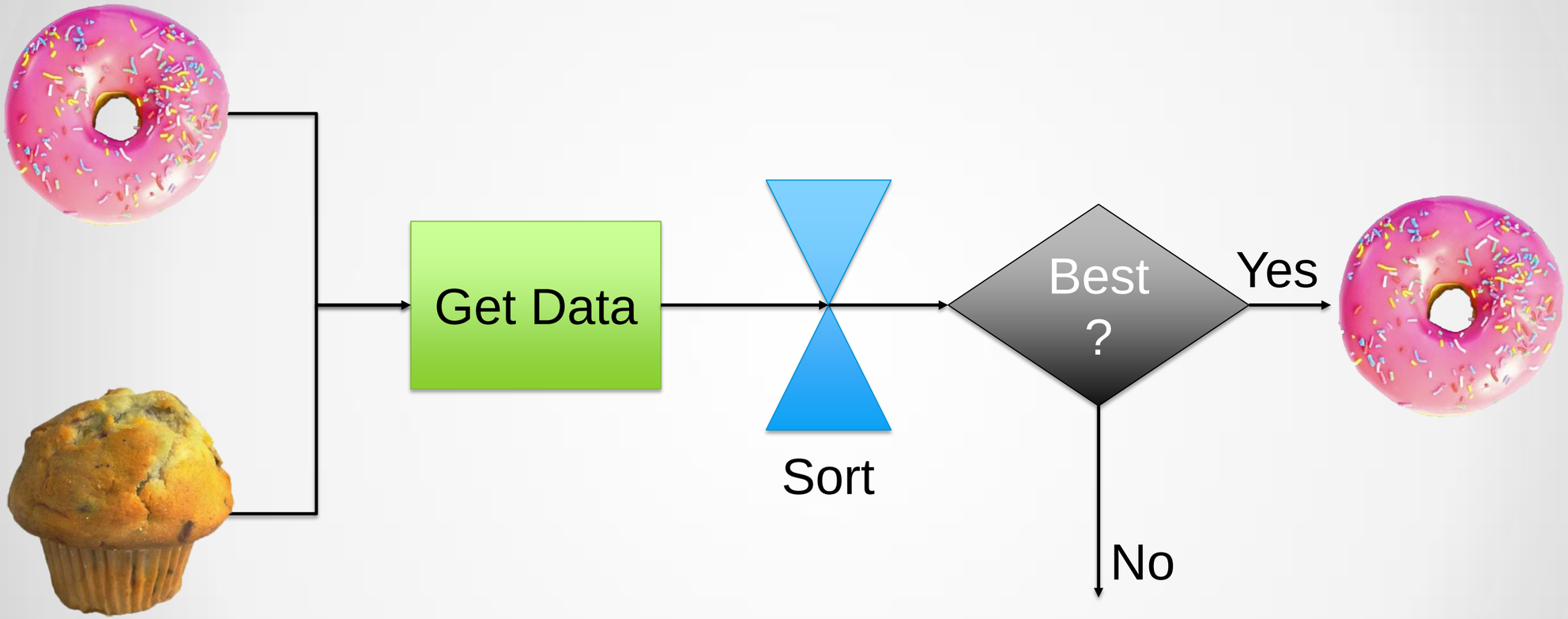
A simple flow chart



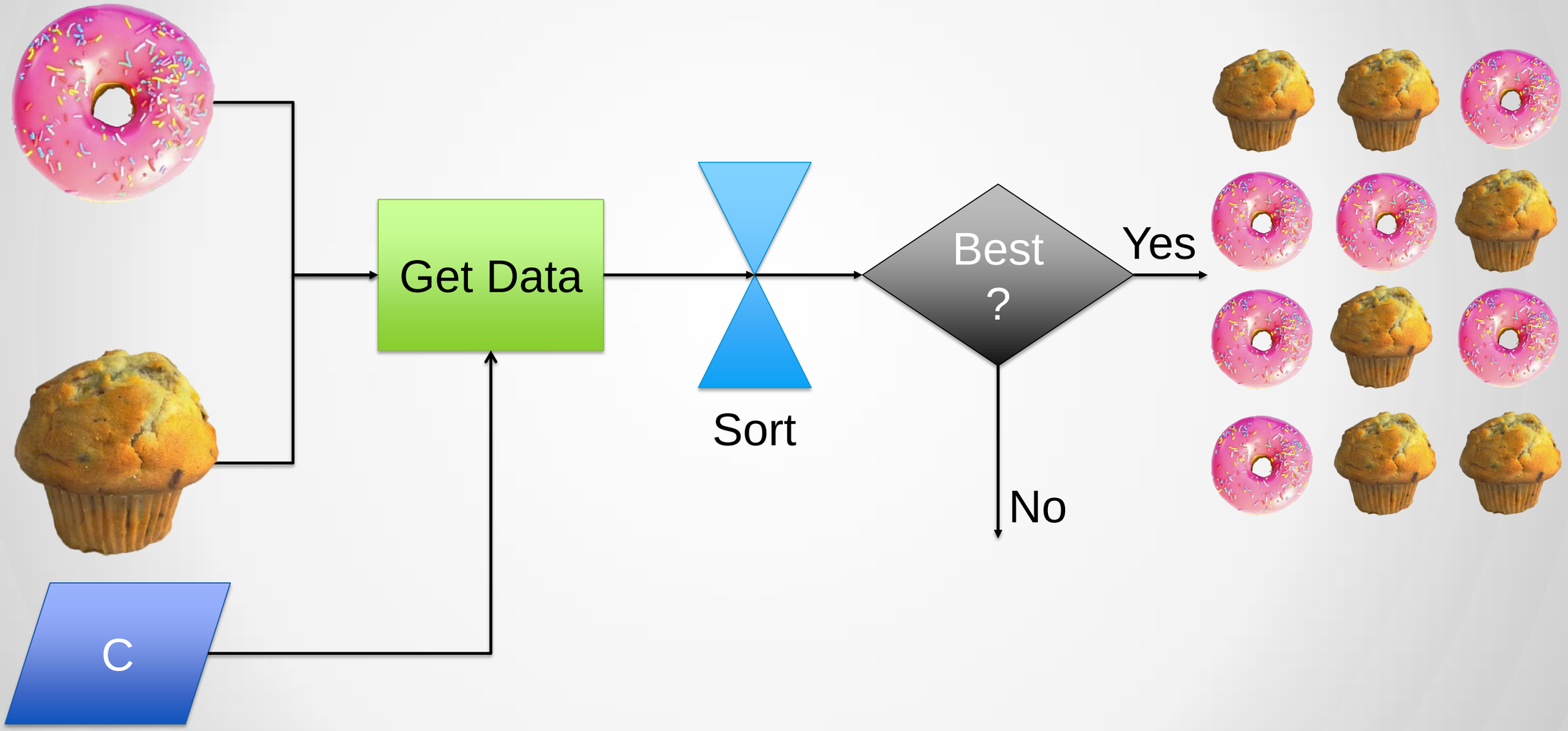
A simple cake chart



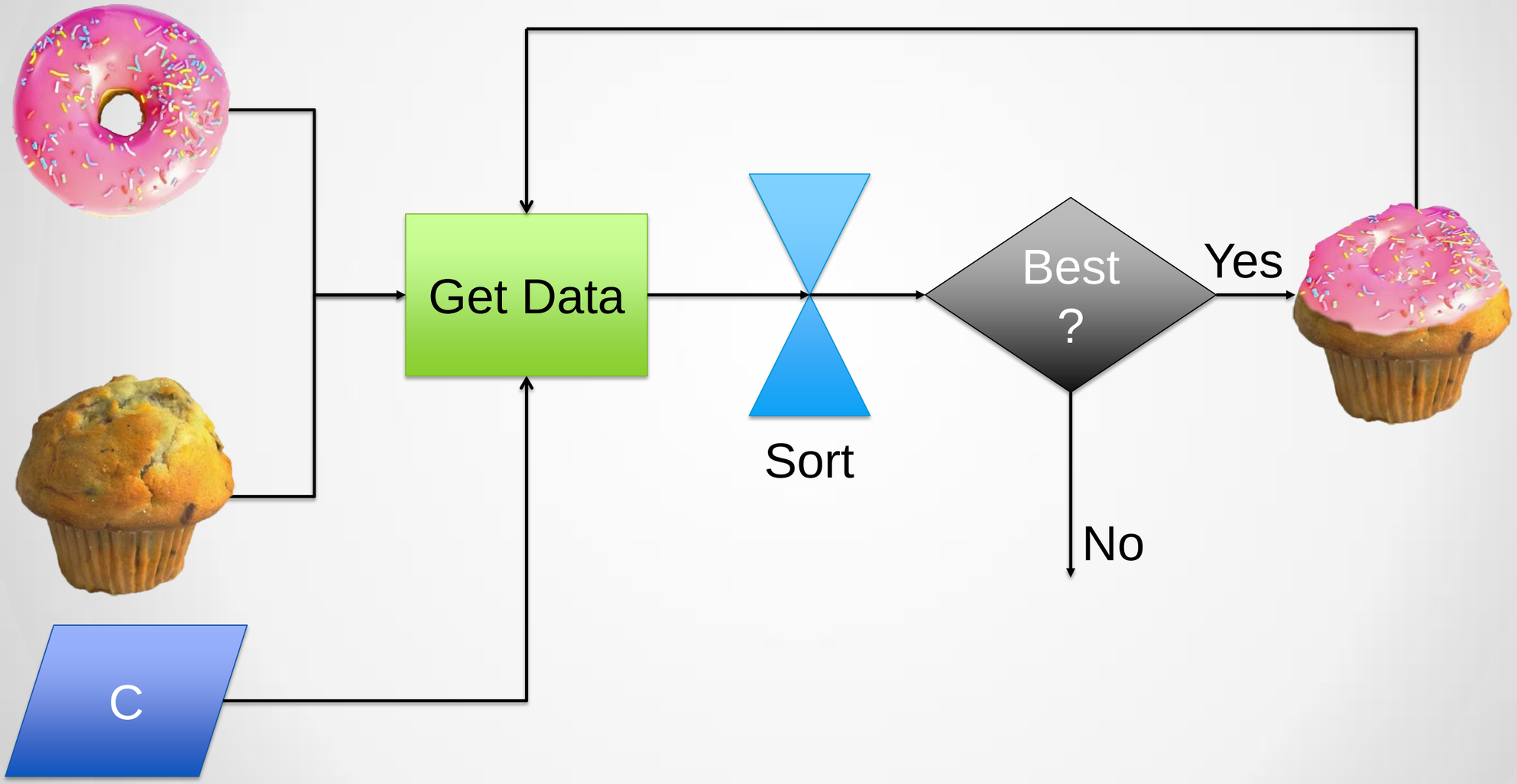
Making an informed decision



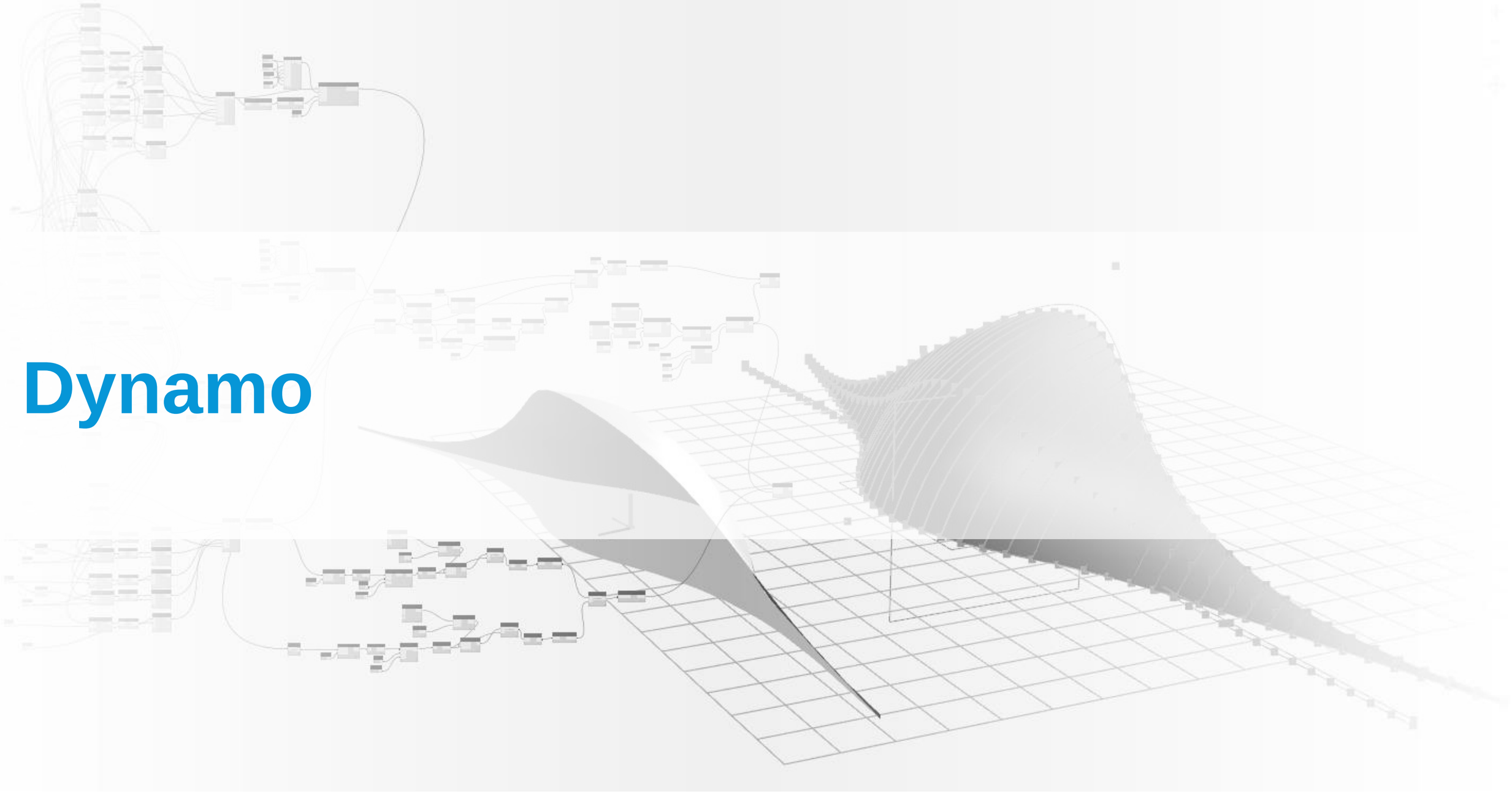
Variable secondary information



Feedback loop

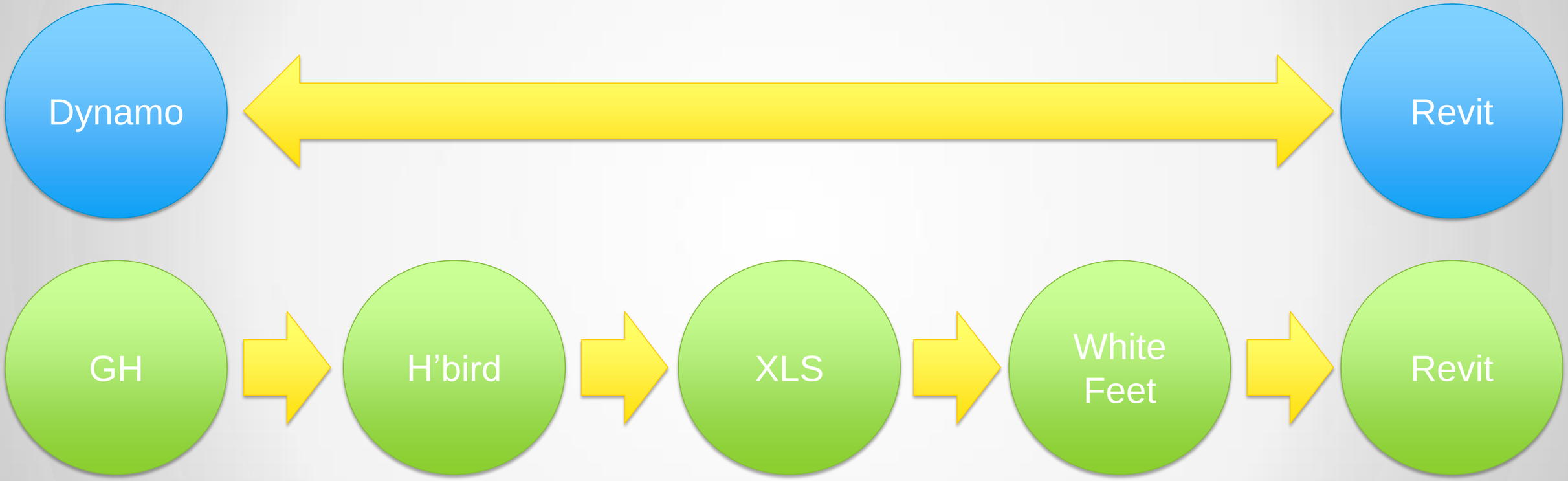


Dynamo

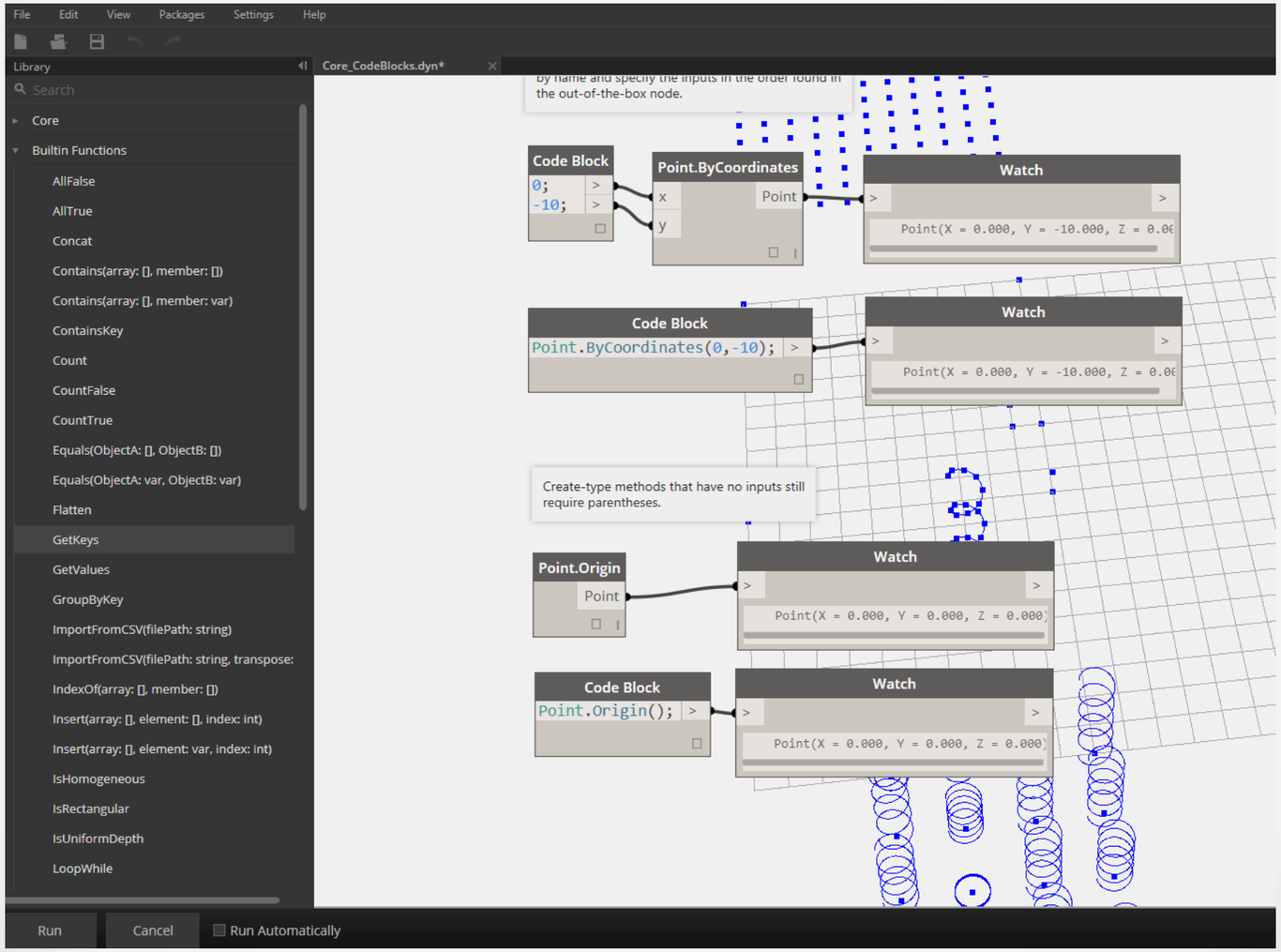


Why did Flanagan Lawrence choose Dynamo?

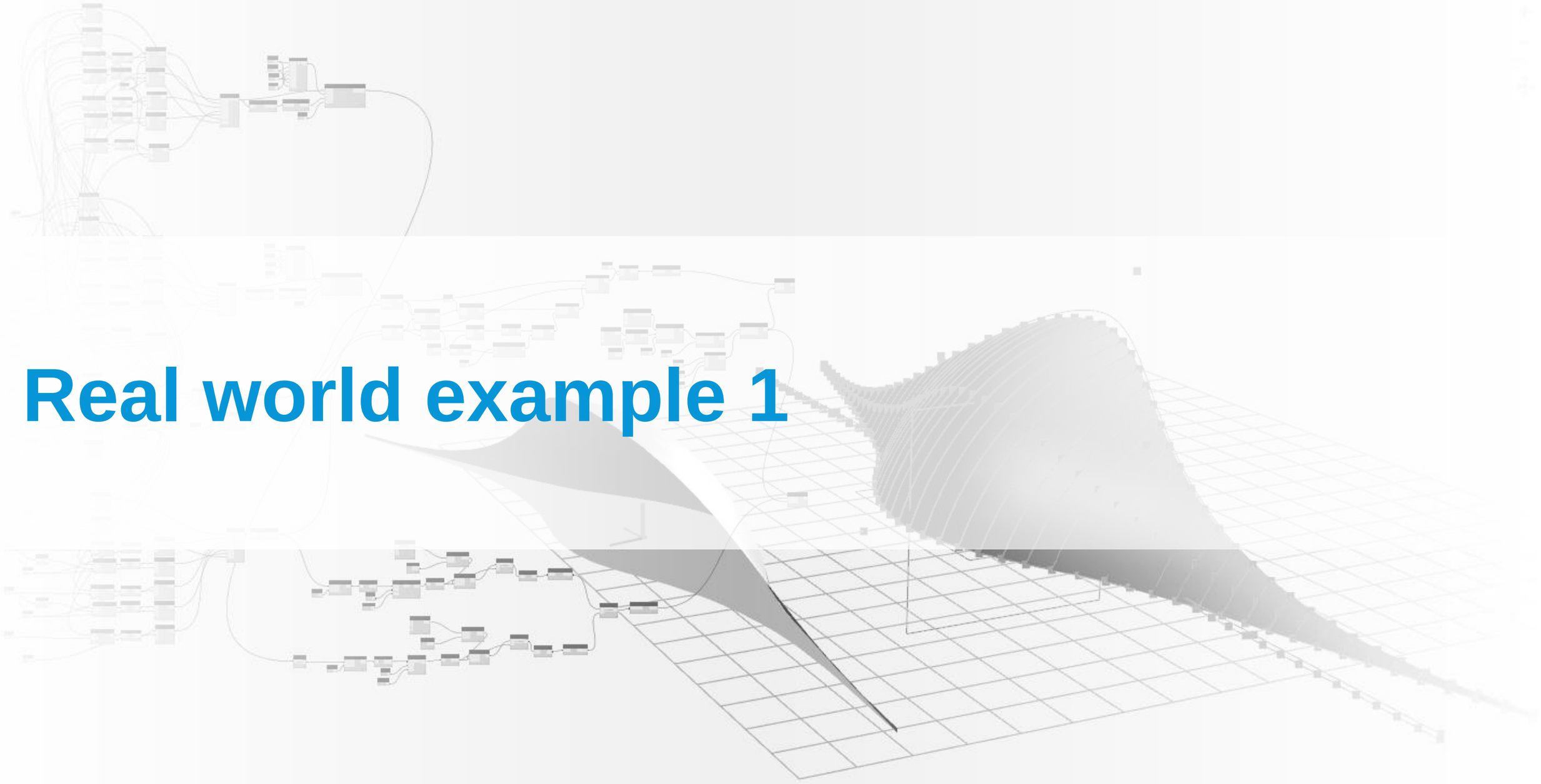
Better than GH



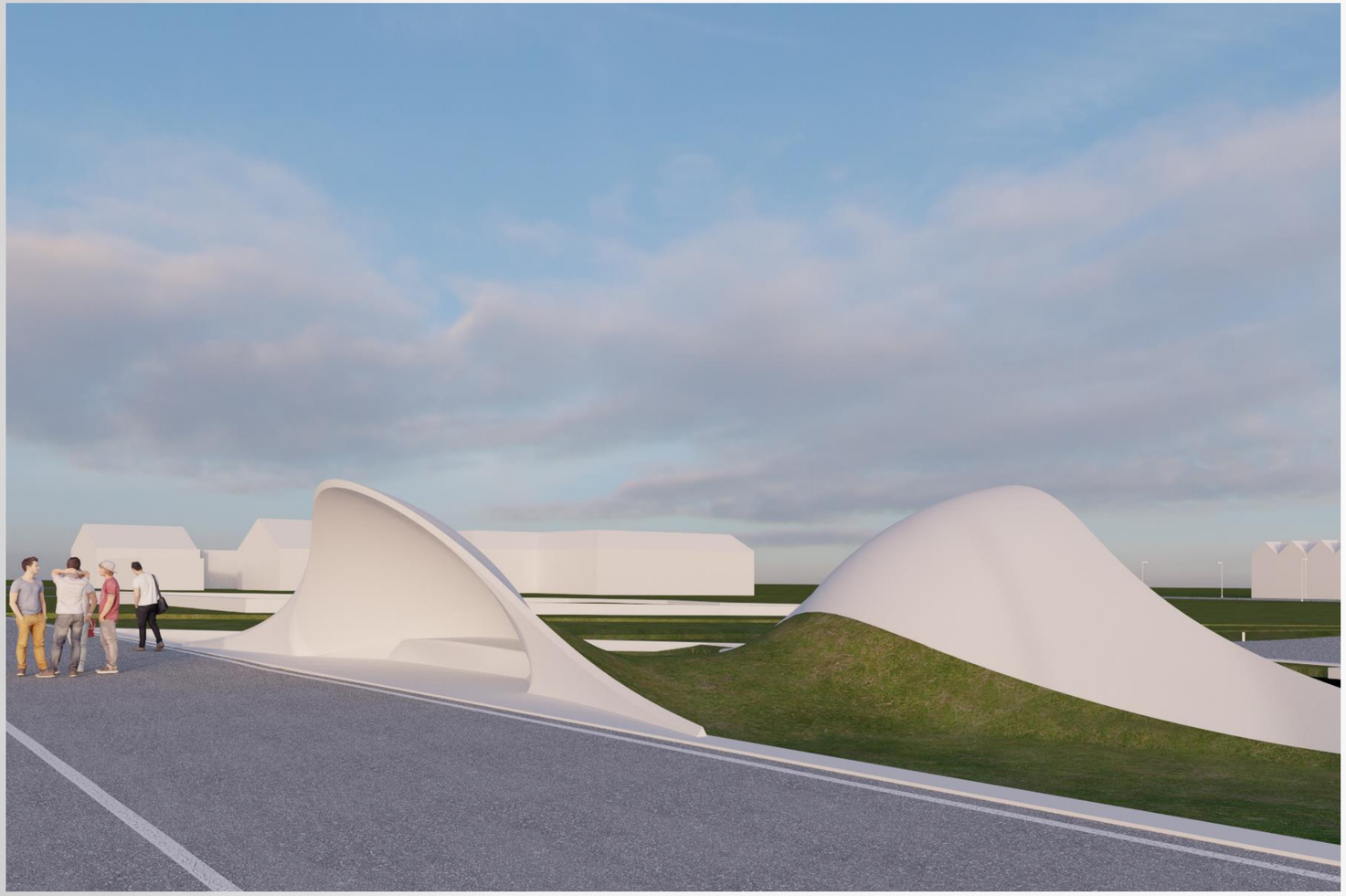
UI and Code blocks

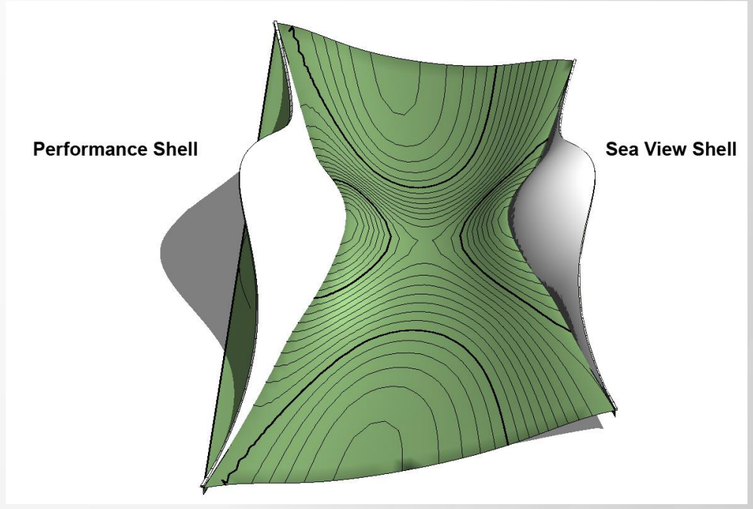


Real world example 1



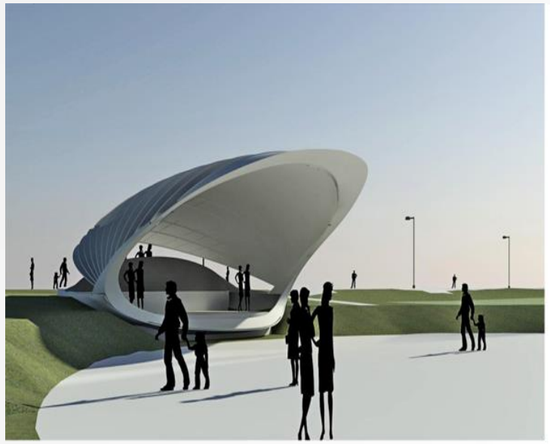
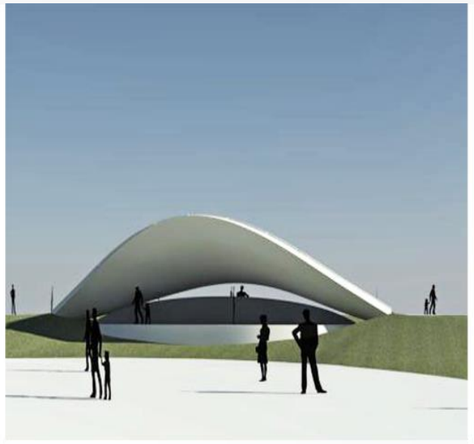
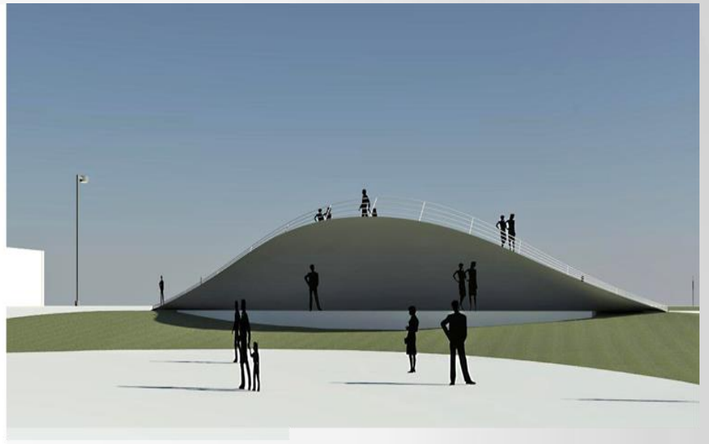
Acoustic Shells, Littlehampton

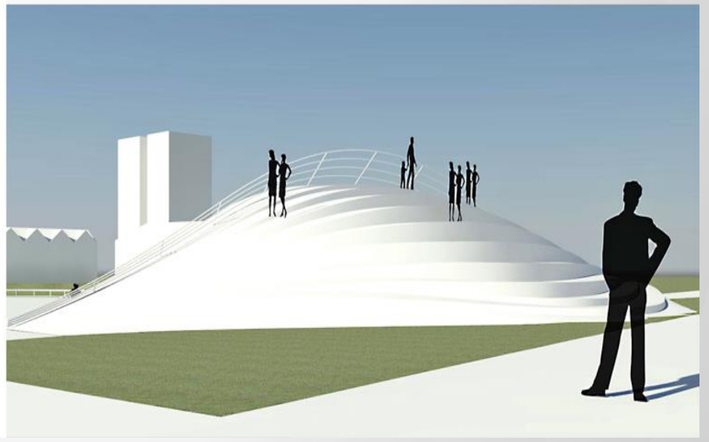


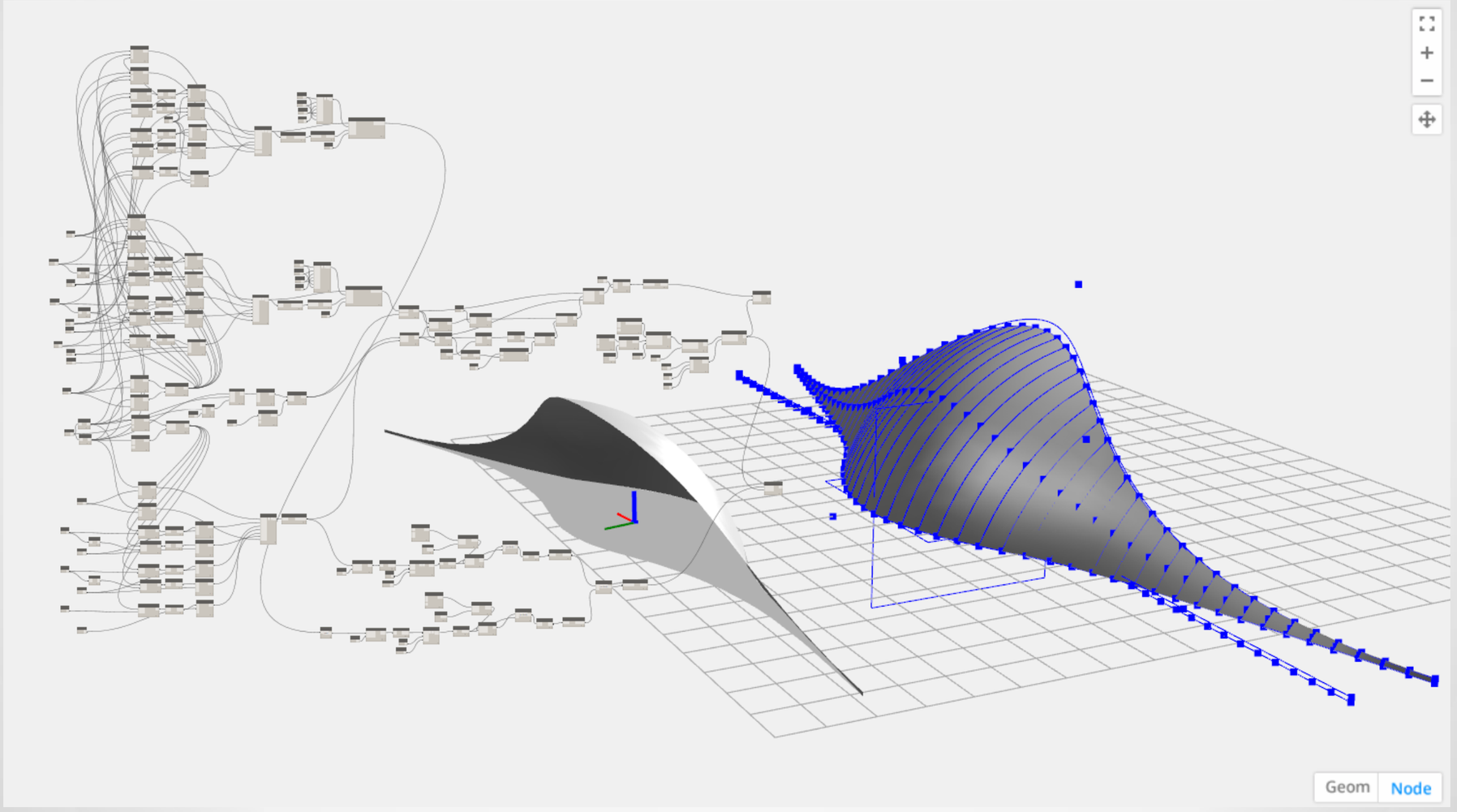


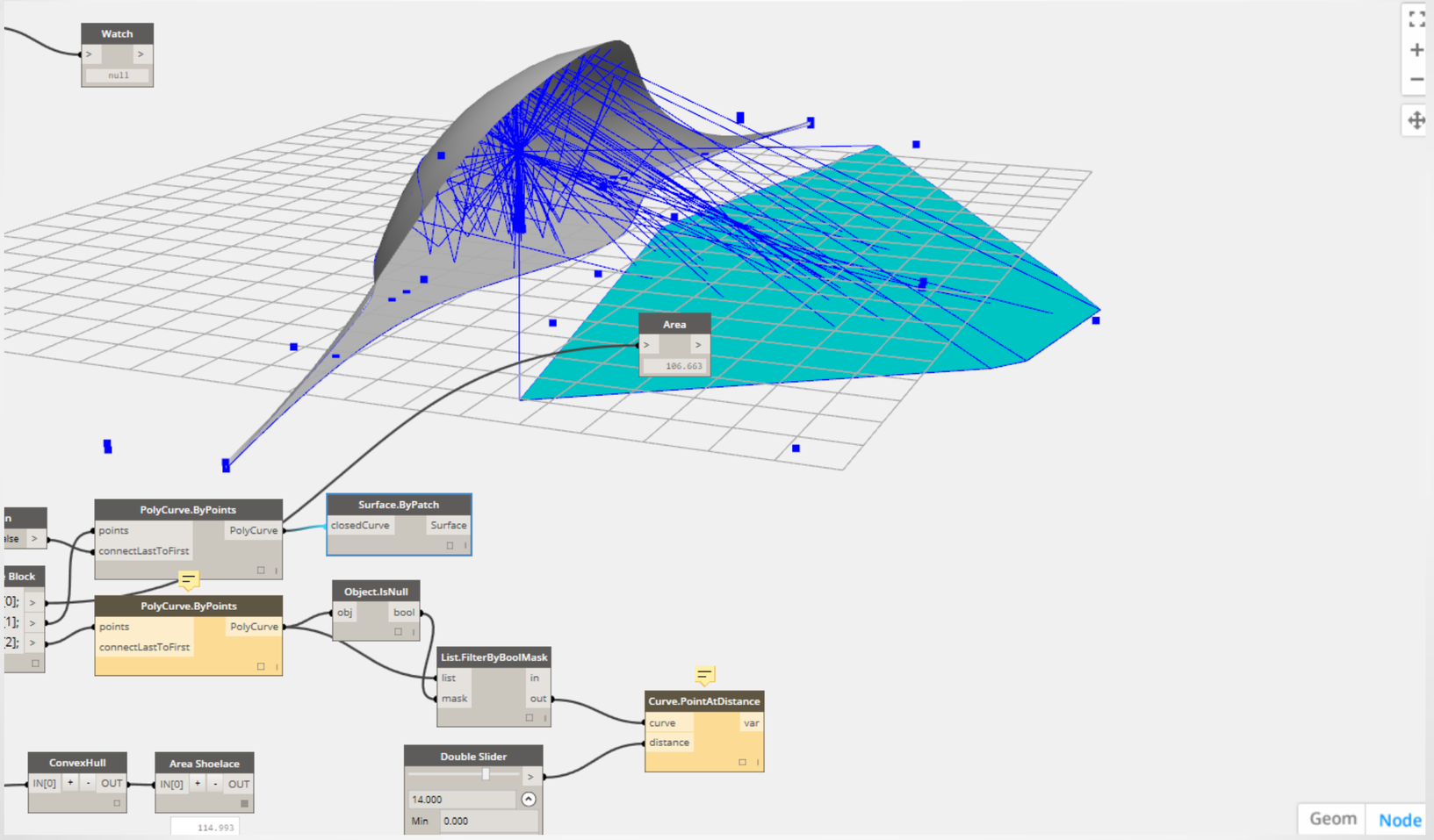
Conceptual development

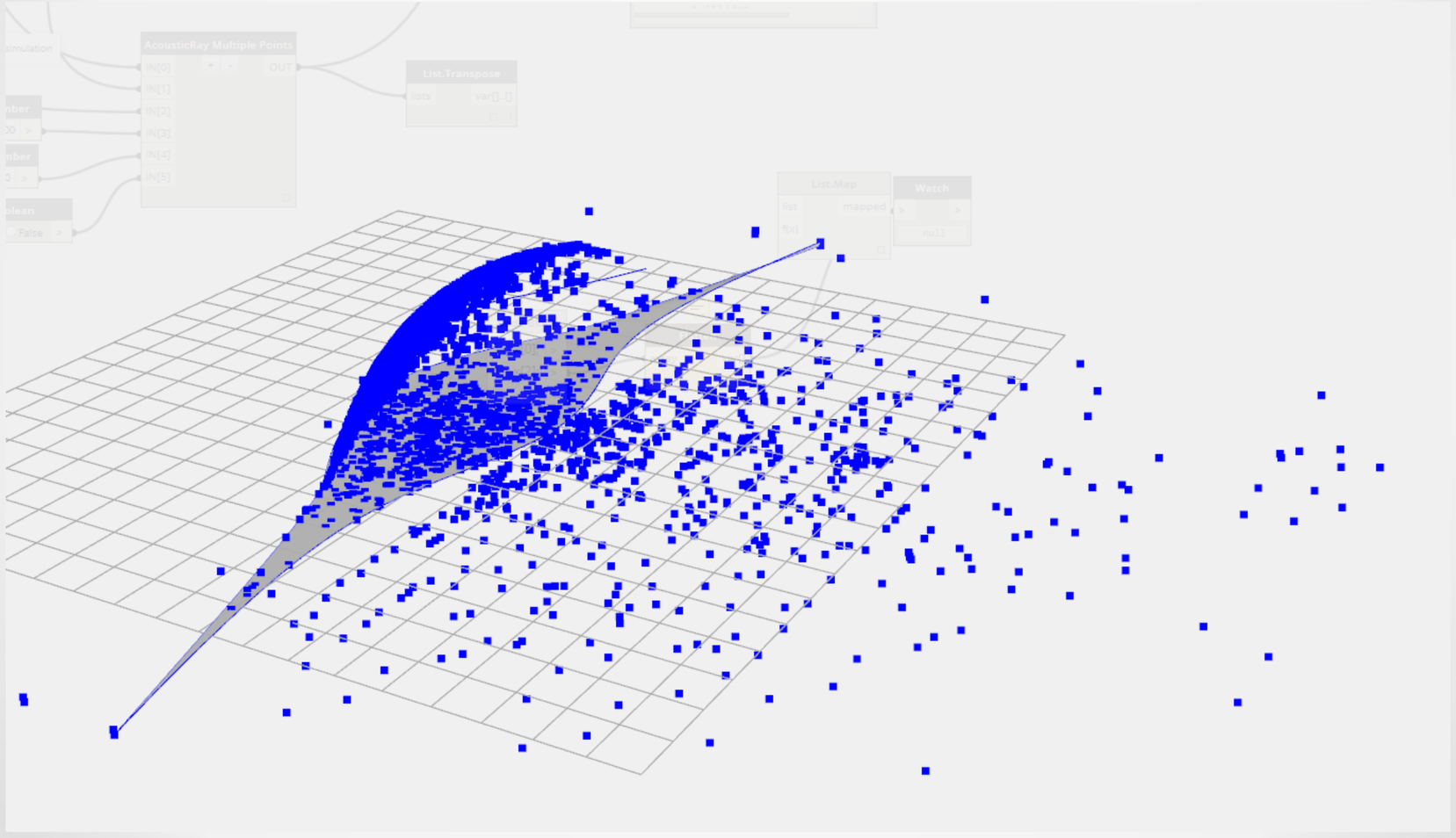
- Design Options
- Simplistic geometry
- Quick

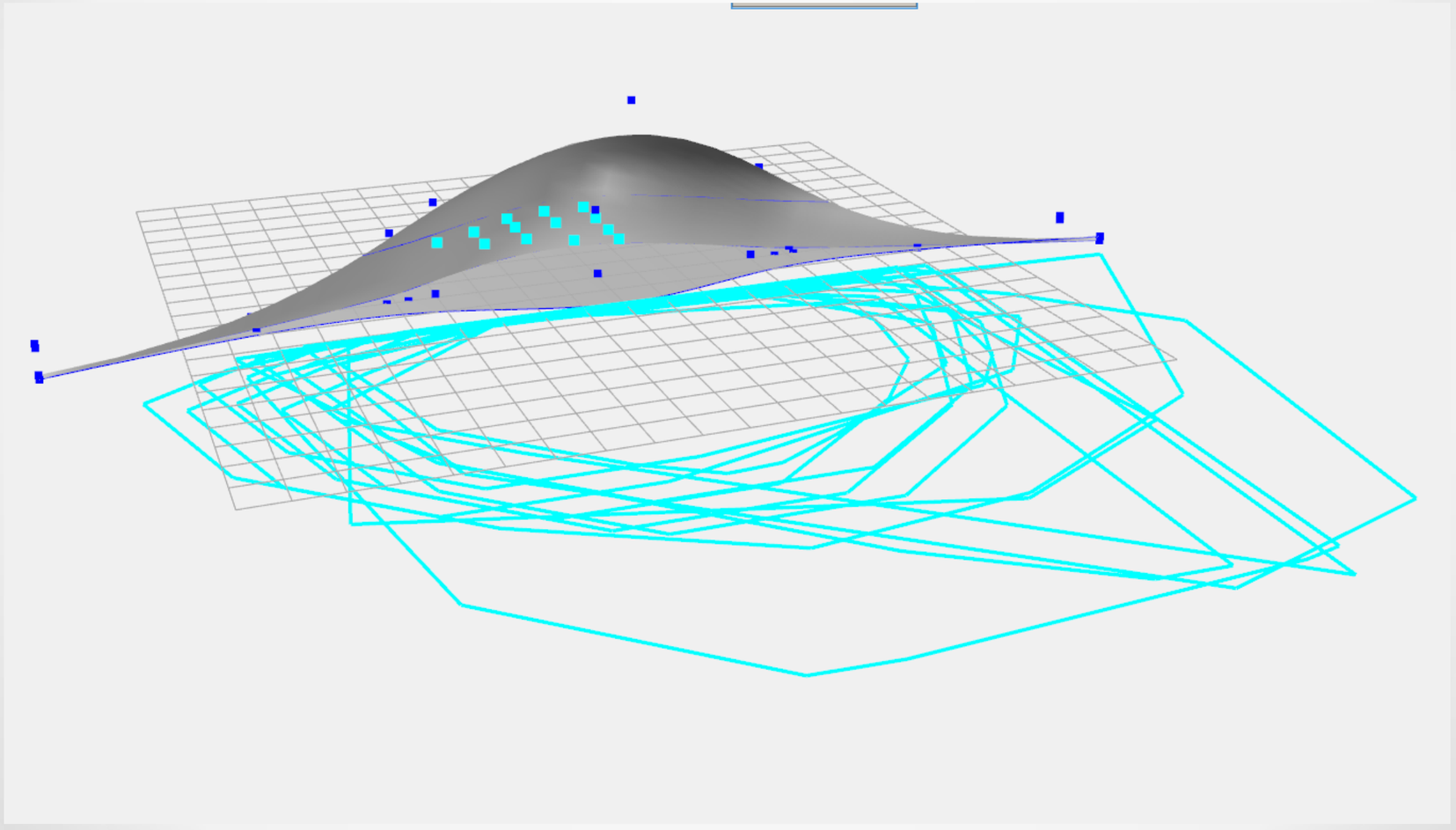




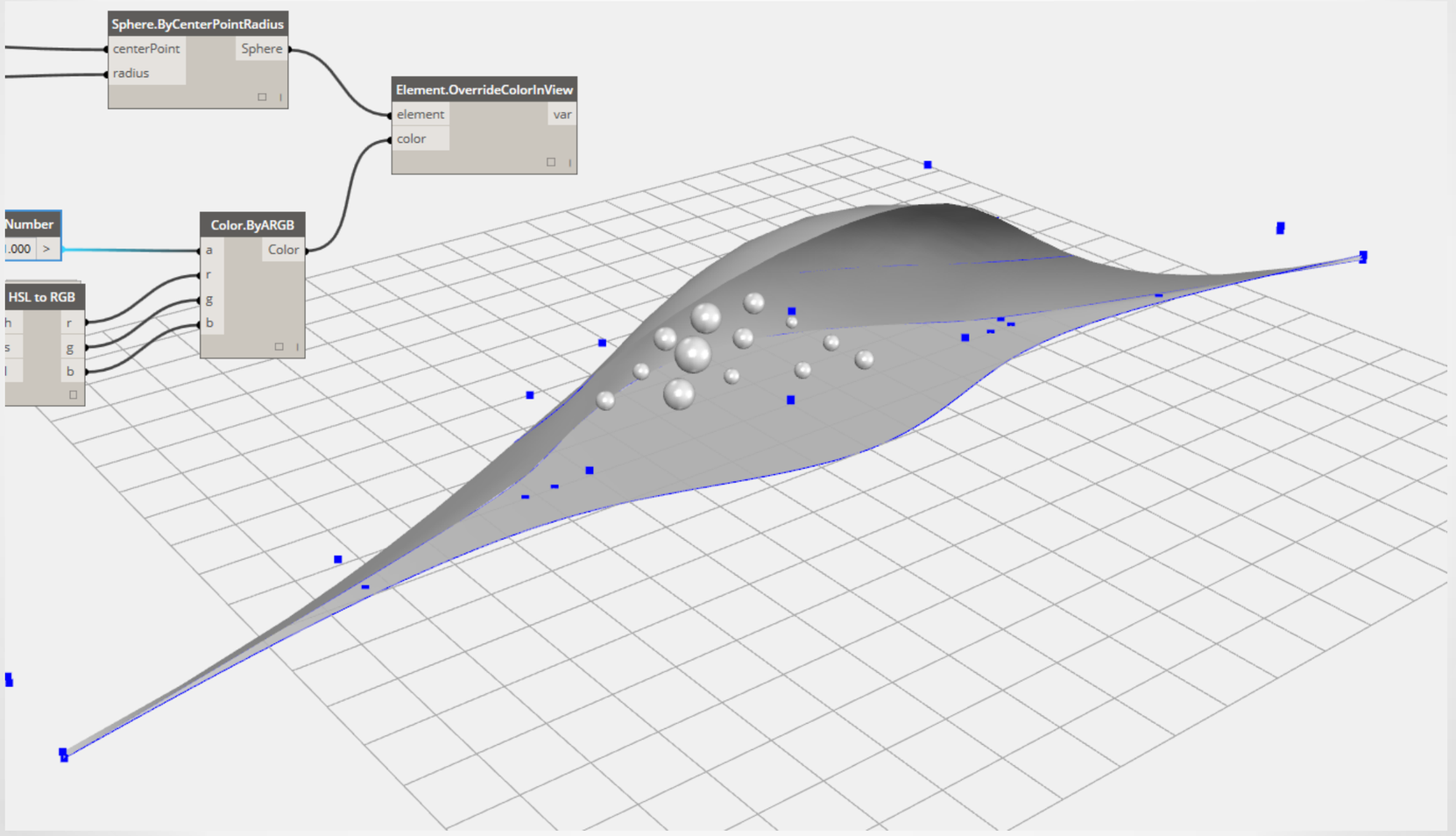




Geom
Node

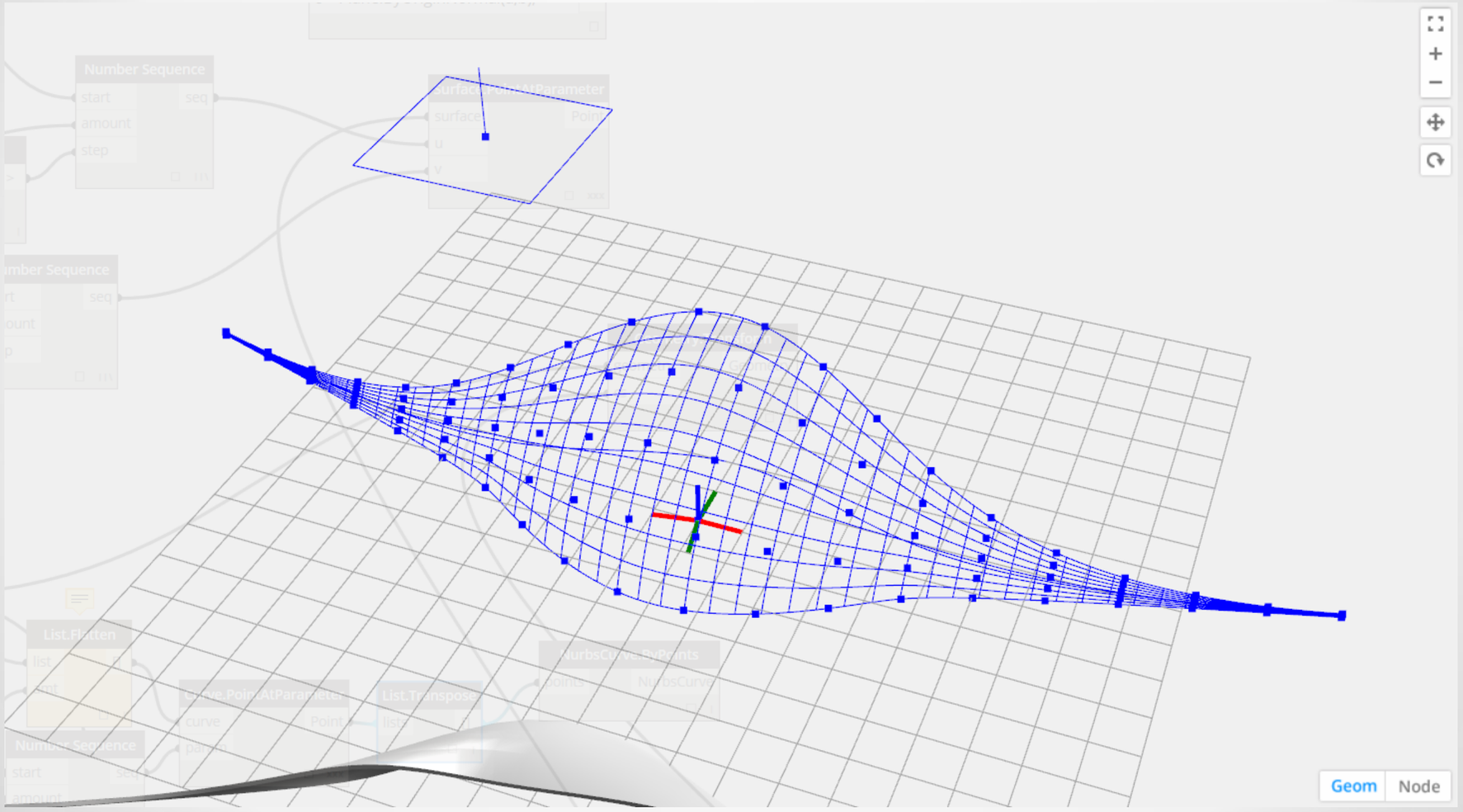

Watch
null
Area
106.663
PolyCurve.ByPoints
points
connectLastToFirst
PolyCurve
Surface.ByPatch
closedCurve
Surface
Block
PolyCurve.ByPoints
points
connectLastToFirst
PolyCurve
Object.IsNull
obj
bool
List.FilterByBoolMask
list
mask
in
out
Curve.PointAtDistance
curve
distance
var
ConvexHull
IN[0]
OUT
Area Shoelace
IN[0]
OUT
114.993
Double Slider
14.000
Min
0.000
Geom
Node

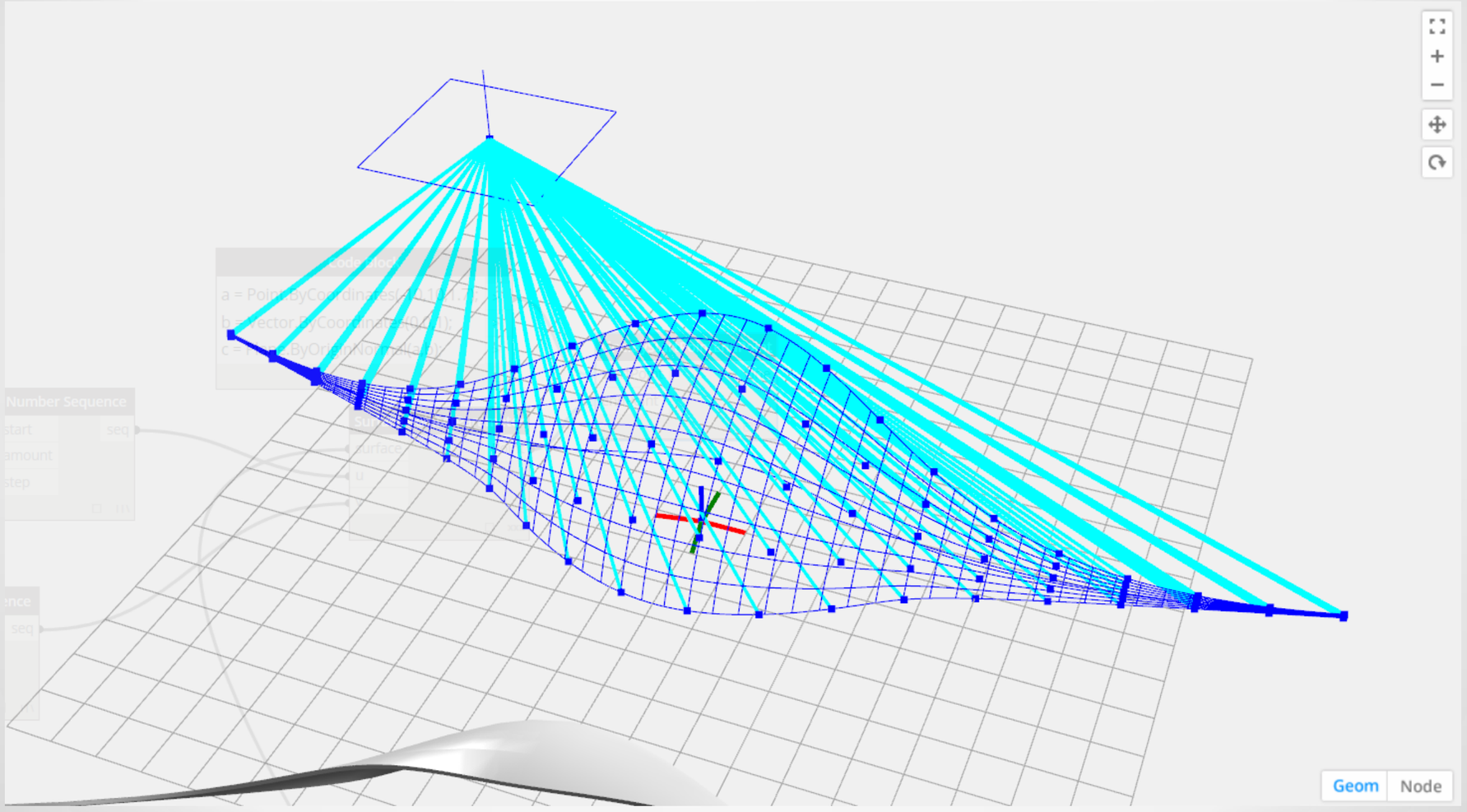

AcousticRay Multiple Points
IN[0]
IN[1]
IN[2]
IN[3]
IN[4]
IN[5]
OUT
List.Transpose
lists
var[]..[]
List.Map
list
mapped
f(x)
Watch
False

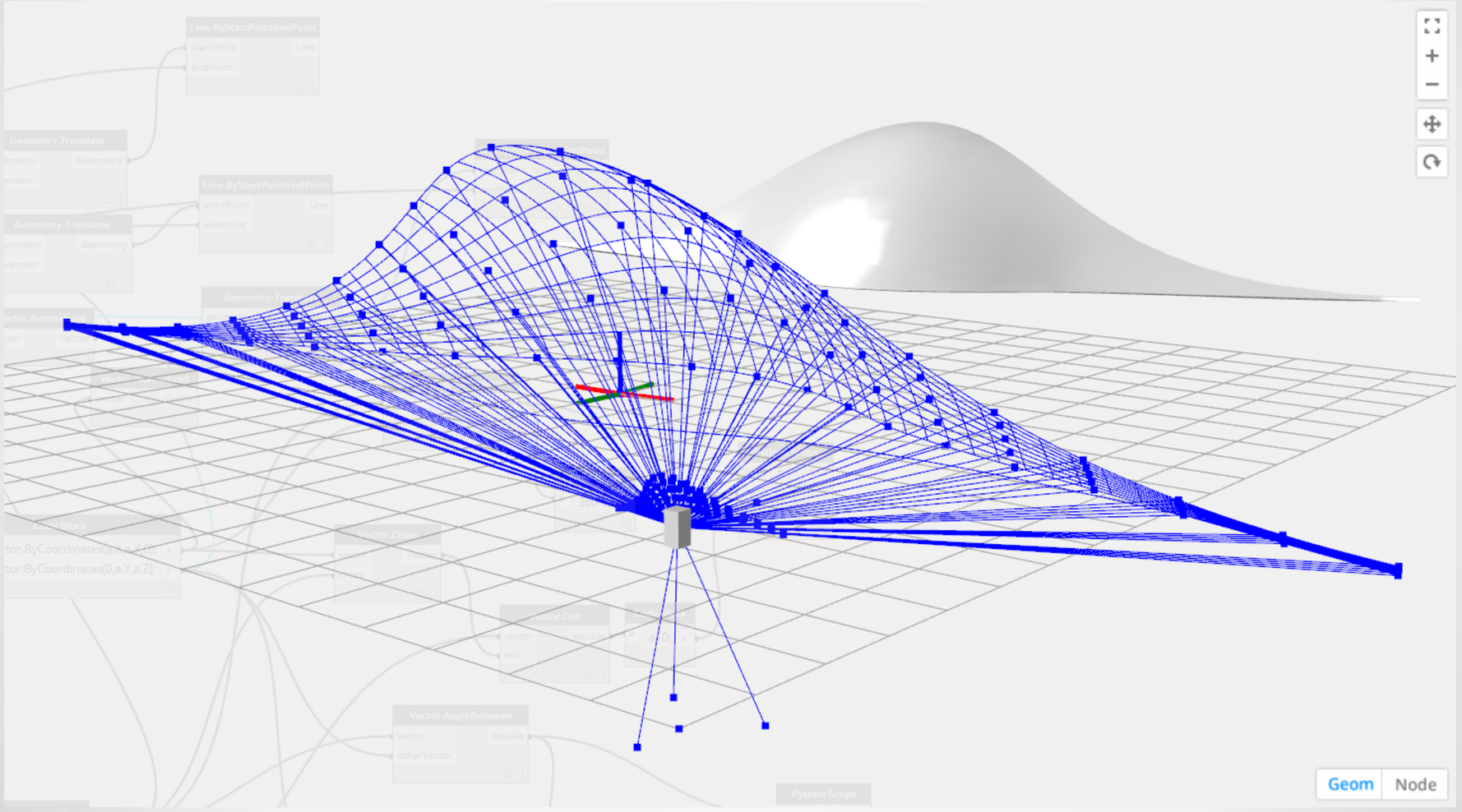


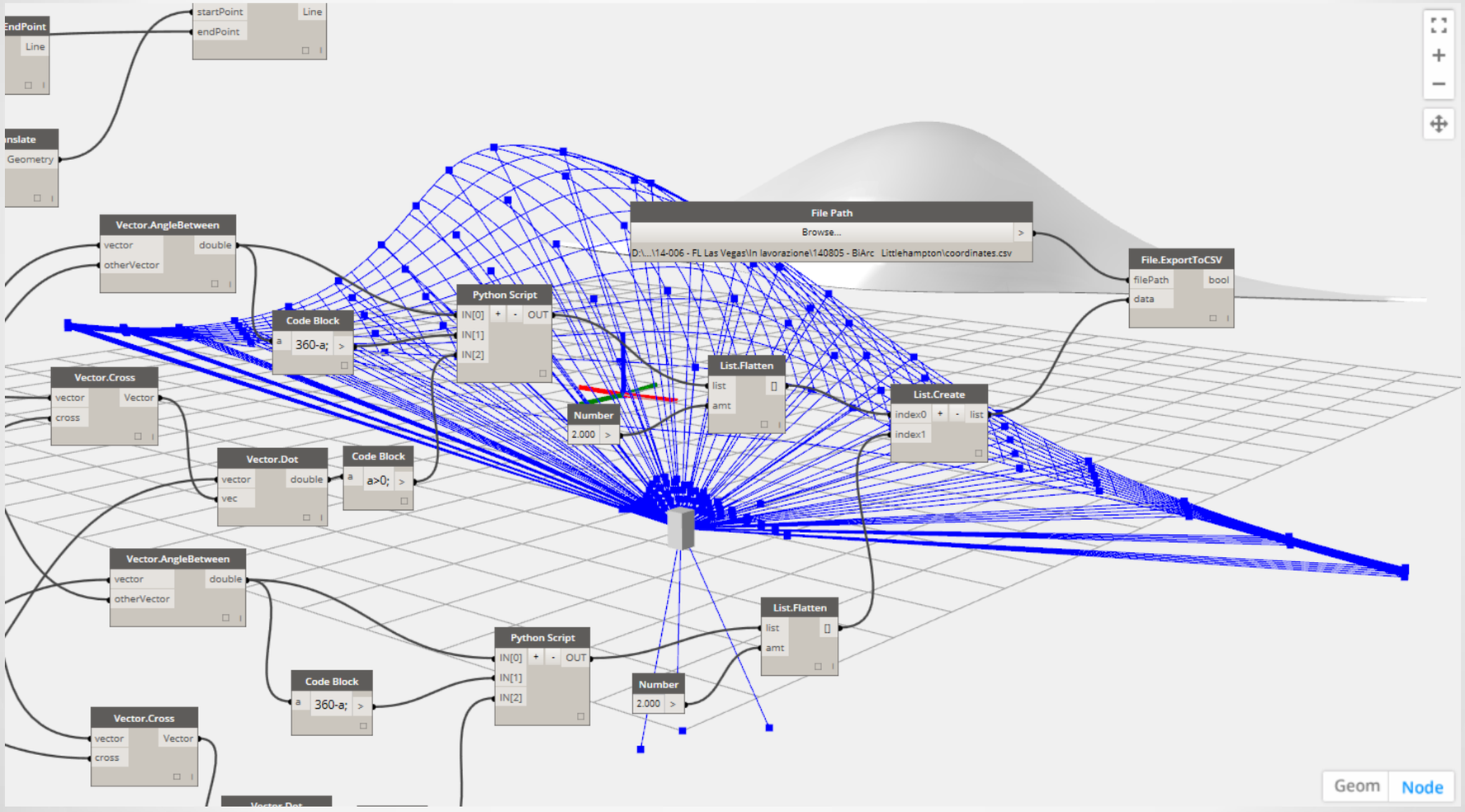

Sphere.ByCenterPointRadius
centerPoint
radius
Sphere
Element.OverrideColorInView
element
color
var
Number
.000
Color.ByARGB
a
r
g
b
Color
HSL to RGB
h
s
l
r
g
b

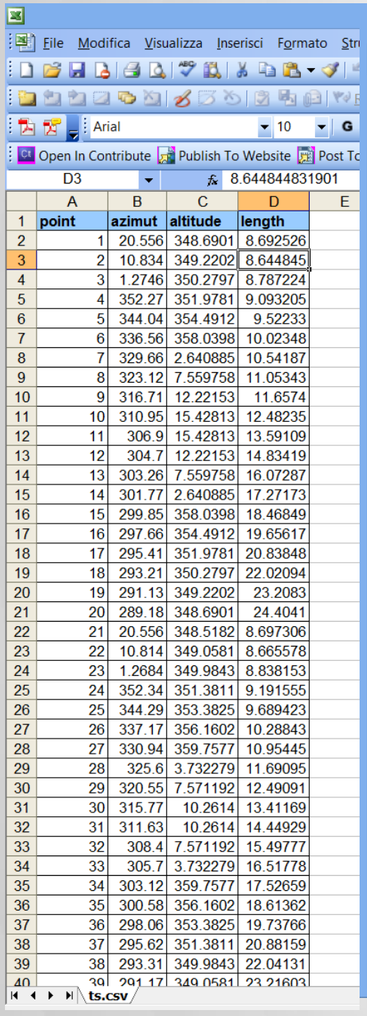
Fabrication


Number Sequence
start
amount
step
seq
Geom
Node


Number Sequence
start
seq
amount
step
Geom
Node

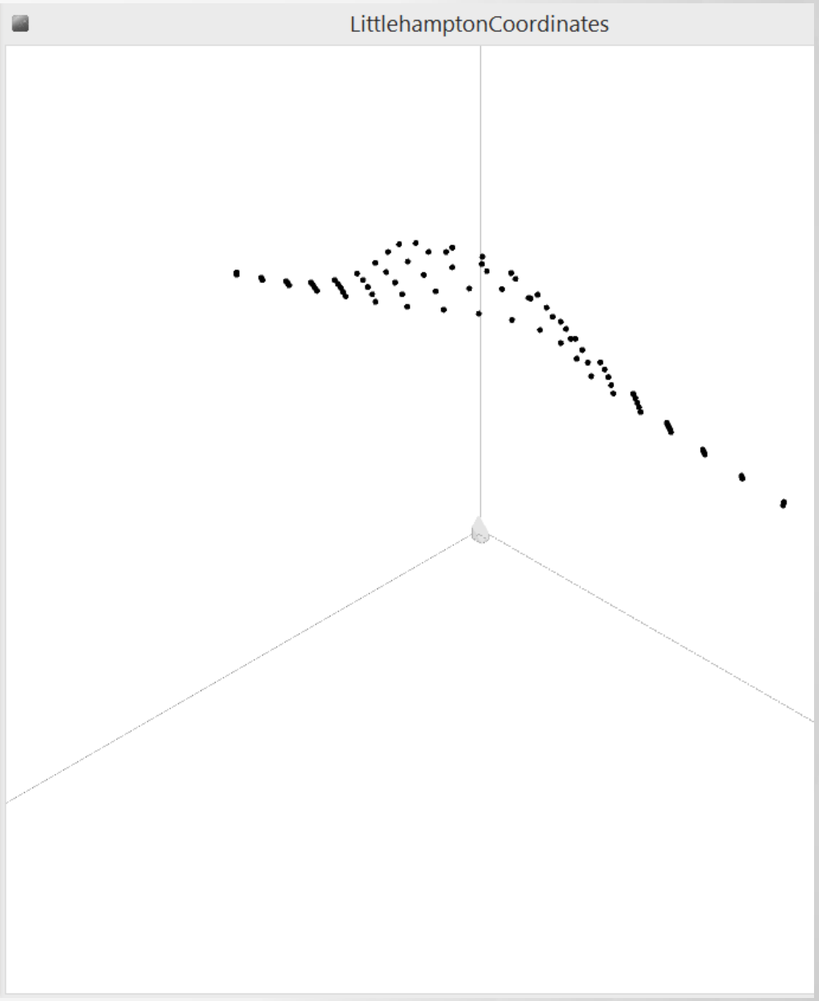

Line.ByStartPointEndPoint
startPoint
endPoint
Line
Geometry.Translate
Geometry
Vector.AngleBetween
vector
otherVector
double
Python Script
Geom
Node


startPoint
Line
endPoint
Line
Geometry
Vector.AngleBetween
vector
otherVector
double
File Path
Browse...
D:\...\14-006 - FL Las Vegas\In lavorazione\140805 - BiArc Littlehampton\coordinates.csv
File.ExportToCSV
filePath
bool
data
Python Script
IN[0]
IN[1]
IN[2]
OUT
Code Block
360-a;
Vector.Cross
vector
Vector
cross
List.Flatten
list
amt
List.Create
index0
index1
Number
2.000
Vector.Dot
vec
Code Block
a>0;
Vector.AngleBetween
Python Script
List.Flatten
Code Block
360-a;
Number
2.000
Vector.Cross
Geom
Node

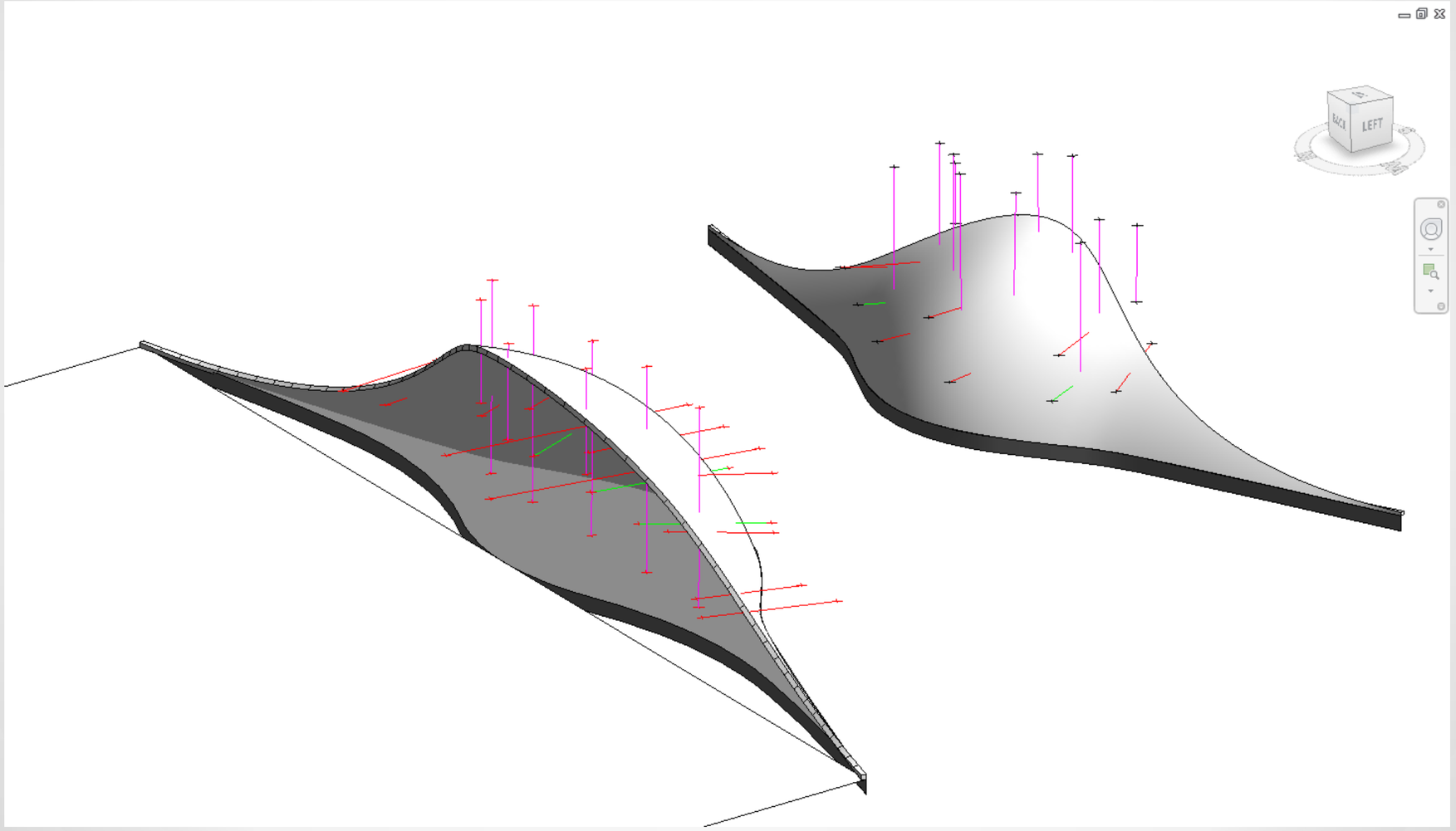


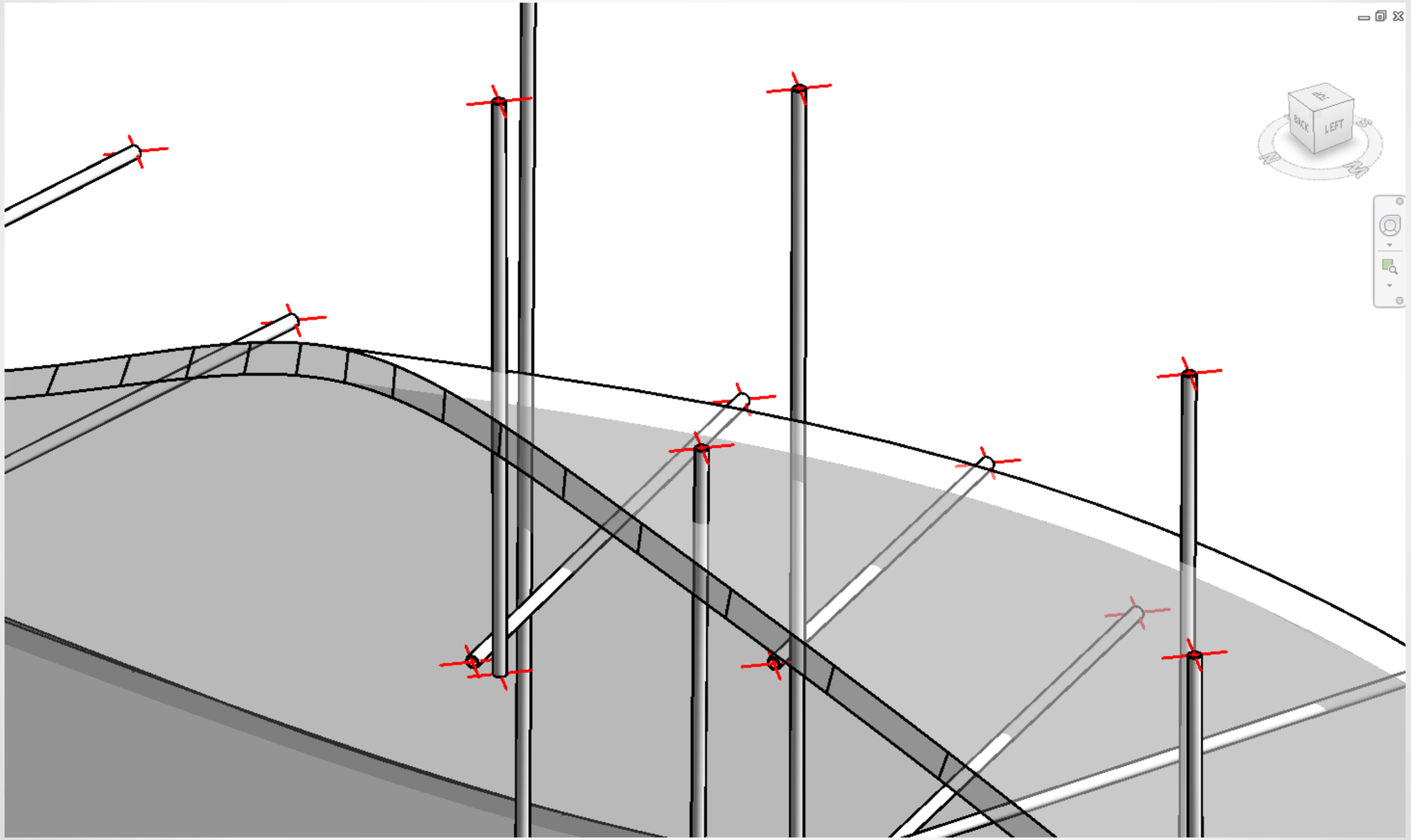
	A	B	C	D	E
1	point	azimut	altitude	length	
2	1	20.556	348.6901	8.692526	
3	2	10.834	349.2202	8.644845	
4	3	1.2746	350.2797	8.787224	
5	4	352.27	351.9781	9.093205	
6	5	344.04	354.4912	9.52233	
7	6	336.56	358.0398	10.02348	
8	7	329.66	2.640885	10.54187	
9	8	323.12	7.559758	11.05343	
10	9	316.71	12.22153	11.6574	
11	10	310.95	15.42813	12.48235	
12	11	306.9	15.42813	13.59109	
13	12	304.7	12.22153	14.83419	
14	13	303.26	7.559758	16.07287	
15	14	301.77	2.640885	17.27173	
16	15	299.85	358.0398	18.46849	
17	16	297.66	354.4912	19.65617	
18	17	295.41	351.9781	20.83848	
19	18	293.21	350.2797	22.02094	
20	19	291.13	349.2202	23.2083	
21	20	289.18	348.6901	24.4041	
22	21	20.556	348.5182	8.697306	
23	22	10.814	349.0581	8.665578	
24	23	1.2684	349.9843	8.838153	
25	24	352.34	351.3811	9.191555	
26	25	344.29	353.3825	9.689423	
27	26	337.17	356.1602	10.28843	
28	27	330.94	359.7577	10.95445	
29	28	325.6	3.732279	11.69095	
30	29	320.55	7.571192	12.49091	
31	30	315.77	10.2614	13.41169	
32	31	311.63	10.2614	14.44929	
33	32	308.4	7.571192	15.49777	
34	33	305.7	3.732279	16.51778	
35	34	303.12	359.7577	17.52659	
36	35	300.58	356.1602	18.61362	
37	36	298.06	353.3825	19.73766	
38	37	295.62	351.3811	20.88159	
39	38	293.31	349.9843	22.04131	
40	39	291.17	349.0581	23.21603	

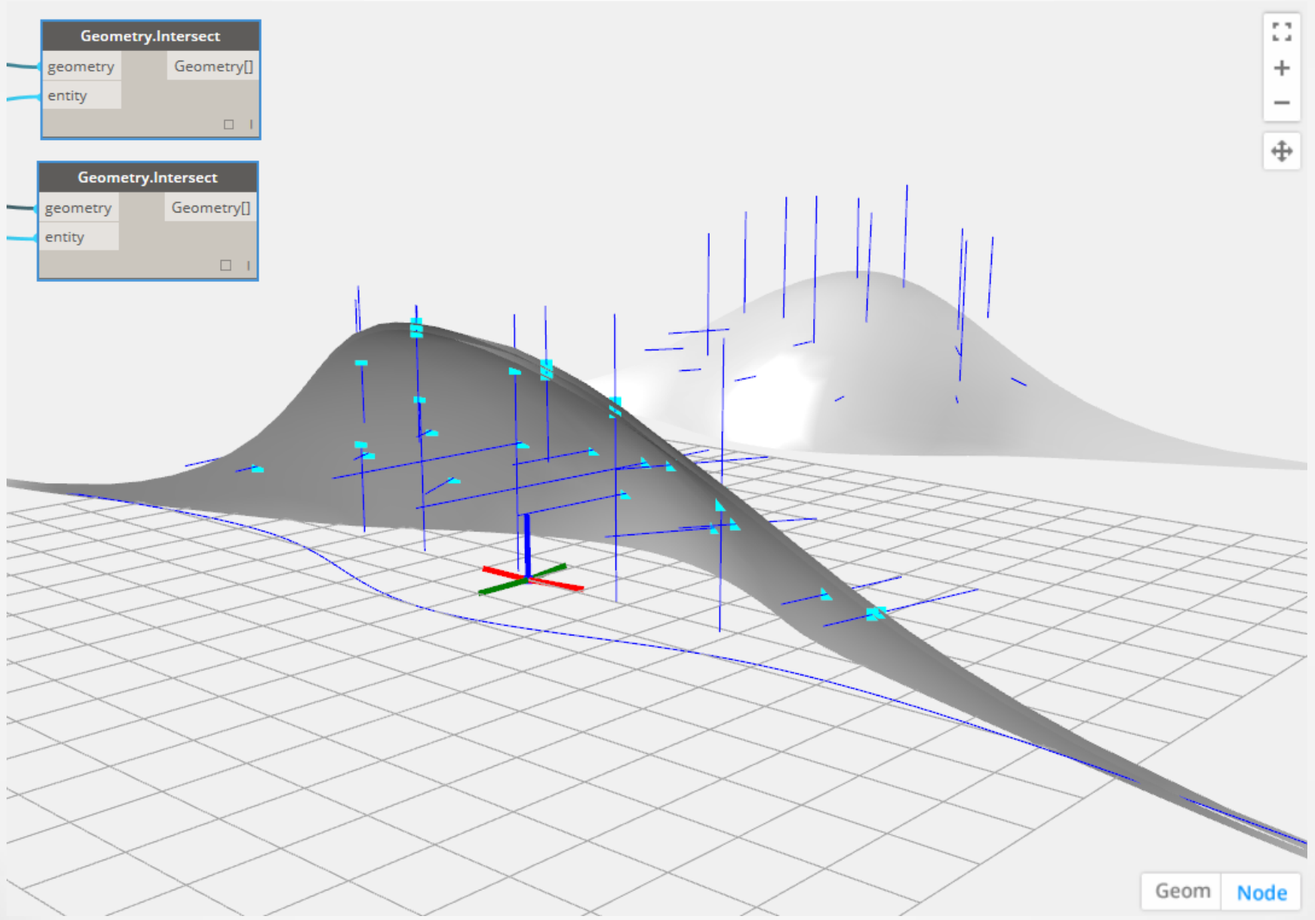
```
/*
void drawPts()
{
  for(int i = 0; i < pts.size(); i++)
  {
    PVector p = pts.get(i);
    point(p.x, p.y, p.z);
  }
}

void drawLine(PVector p)
{
  line(0,0,0,p.x, p.y, p.z);
}*/

void addpti(float az, float alt, float lengt)
{
  pushMatrix();
  rotateX(alt*PI/180);
  rotateZ(az*PI/180);
  point(0,0-lengt,0);
  popMatrix();
}
```

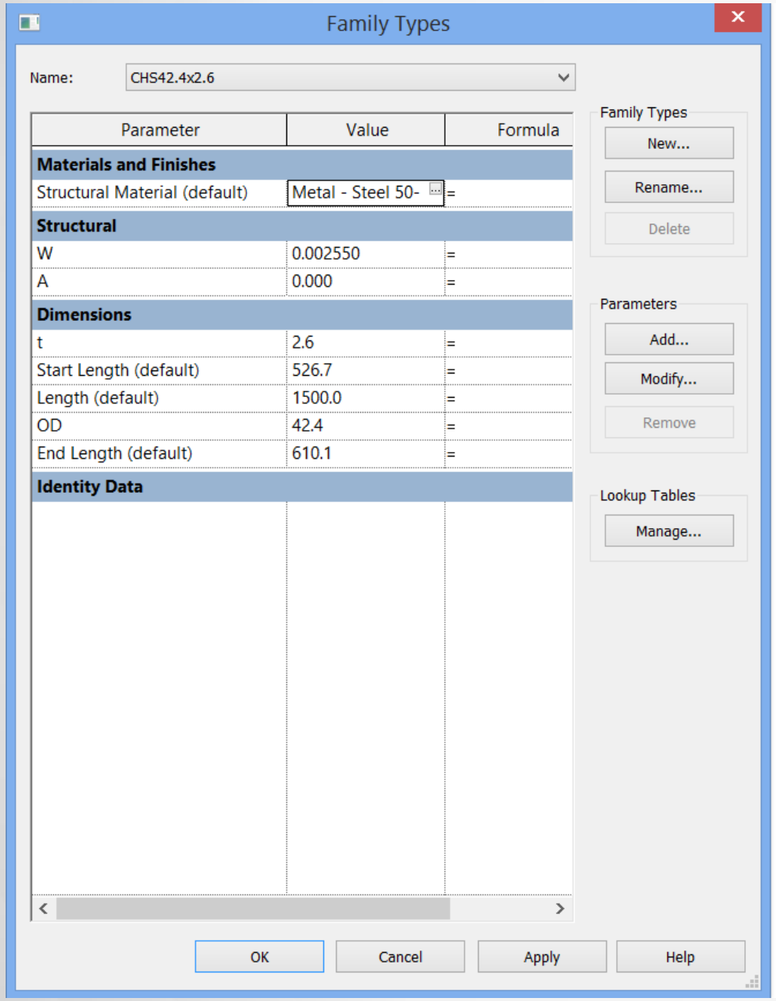


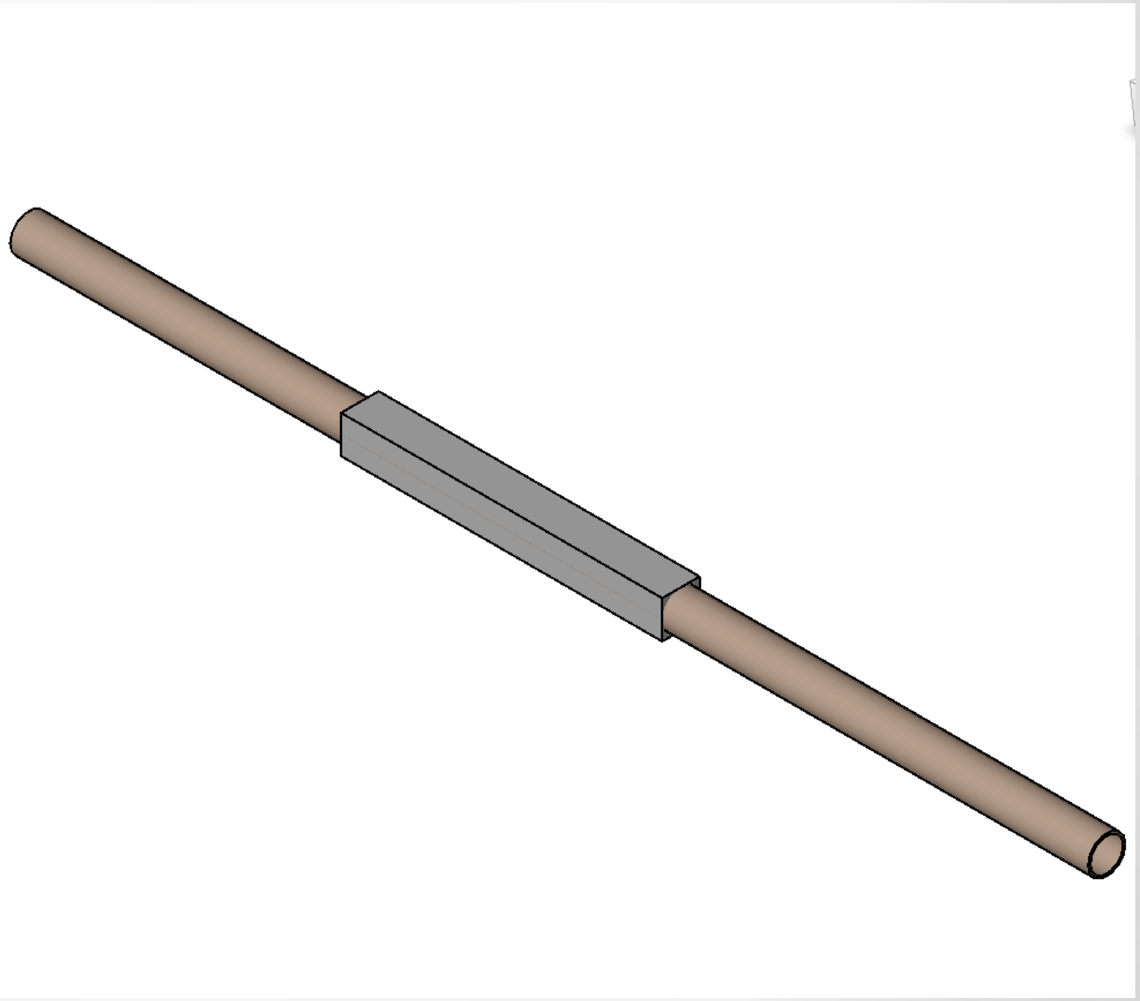
In Reality….

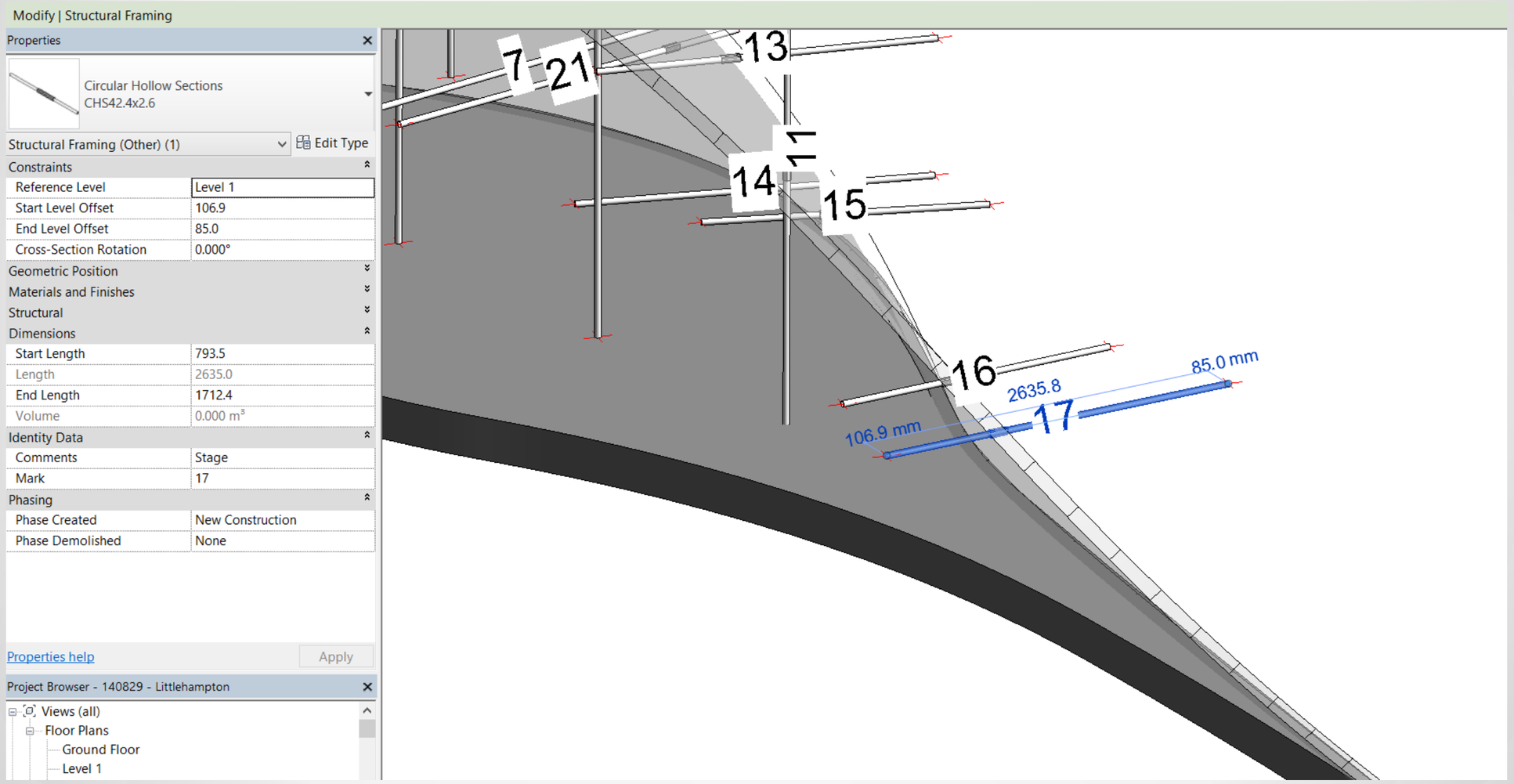

BACK
LEFT

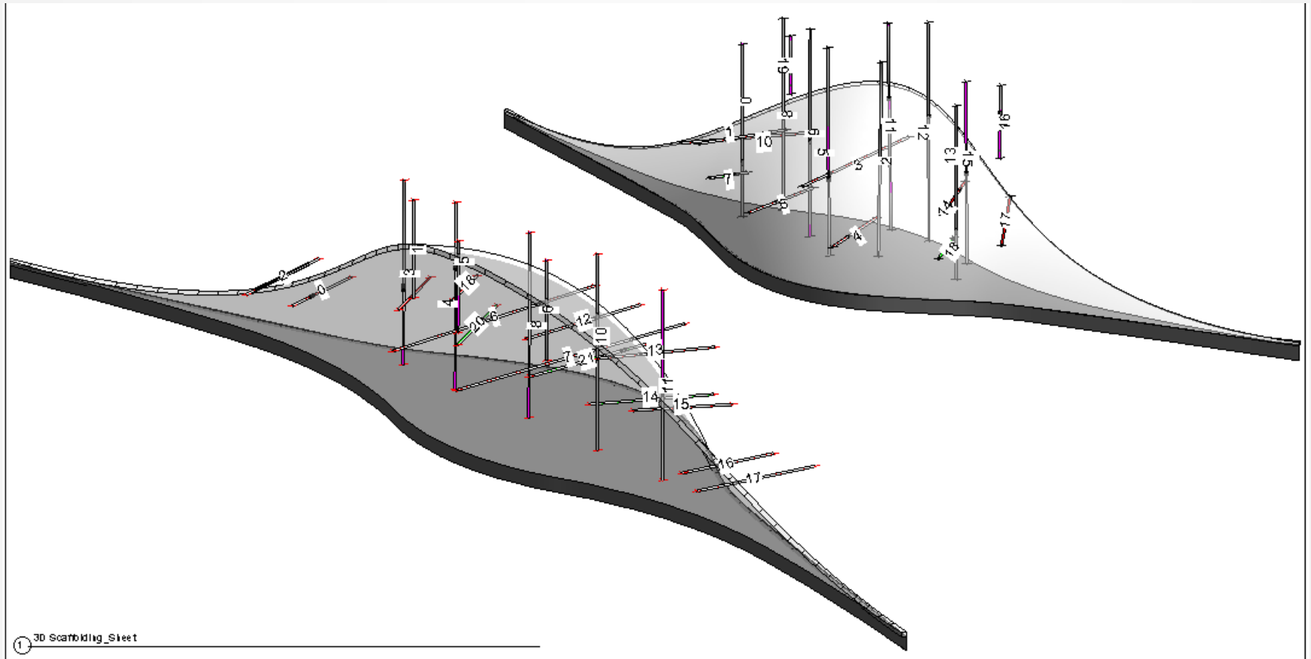

BACK
LEFT


Geometry.Intersect
geometry
entity
Geometry[]
Geometry.Intersect
geometry
entity
Geometry[]
Geom
Node


Family Types
Name:
CHS42.4x2.6
Parameter
Value
Formula
Materials and Finishes
Structural Material (default)
Metal - Steel 50-
=
Structural
W
0.002550
=
A
0.000
=
Dimensions
t
2.6
=
Start Length (default)
526.7
=
Length (default)
1500.0
=
OD
42.4
=
End Length (default)
610.1
=
Identity Data
Family Types
New...
Rename...
Delete
Parameters
Add...
Modify...
Remove
Lookup Tables
Manage...
OK
Cancel
Apply
Help




Modify | Structural Framing
Properties
Circular Hollow Sections
CHS42.4x2.6
Structural Framing (Other) (1)
Edit Type
Constraints
Reference Level
Level 1
Start Level Offset
106.9
End Level Offset
85.0
Cross-Section Rotation
0.000°
Geometric Position
Materials and Finishes
Structural
Dimensions
Start Length
793.5
Length
2635.0
End Length
1712.4
Volume
0.000 m³
Identity Data
Comments
Stage
Mark
17
Phasing
Phase Created
New Construction
Phase Demolished
None
Properties help
Apply
Project Browser - 140829 - Littlehampton
Views (all)
Floor Plans
Ground Floor
Level 1
7
21
13
11
14
15
16
17
2635.8
85.0 mm
106.9 mm



1 3D Scaffolding_Sheet

Structural Framing Schedule

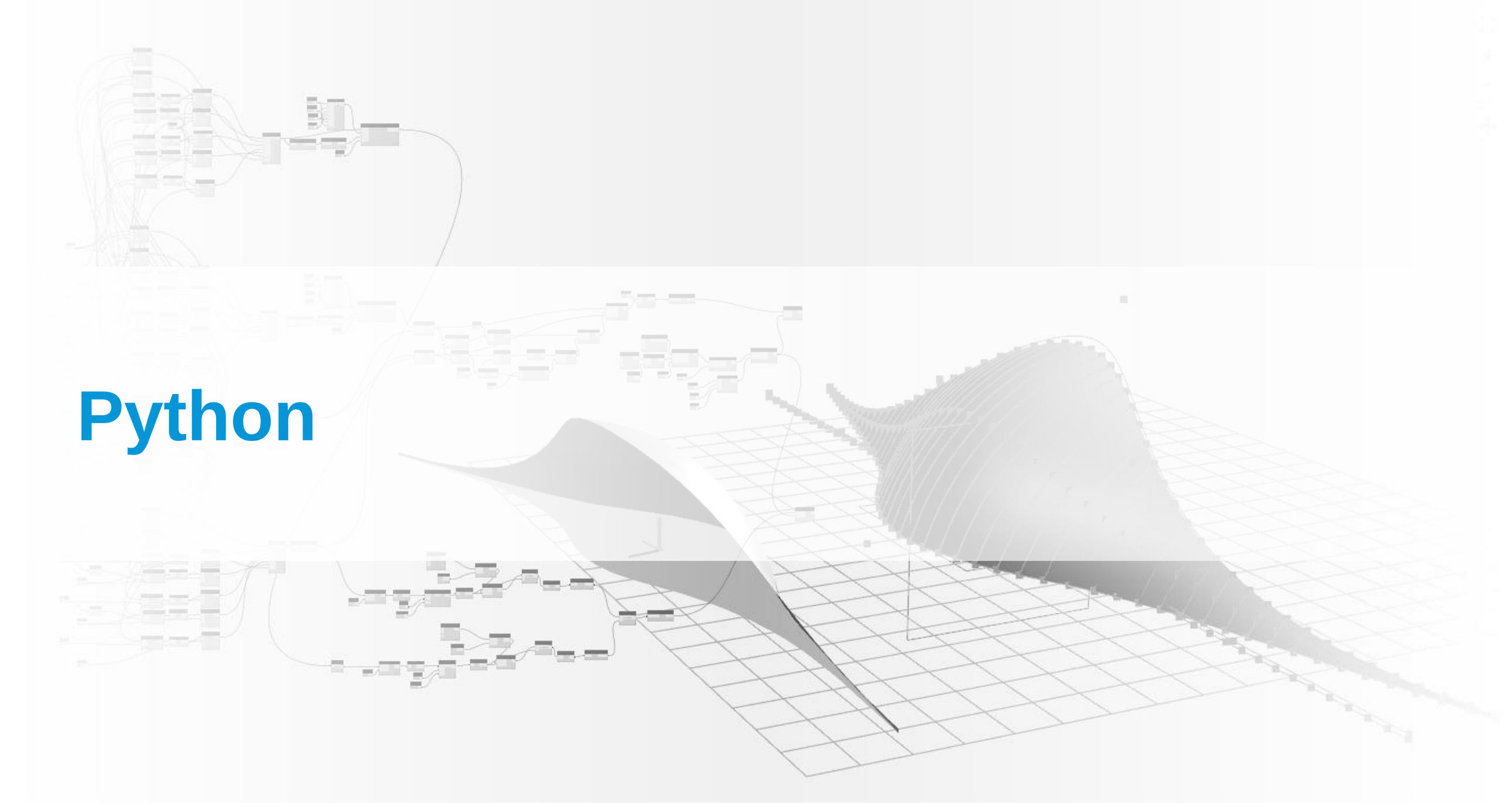
Mark	End Length	Start Length	Comments
0	1202	533	Stage
1	966	966	Stage
2	300	300	Stage
3	1504	2133	Stage
4	2461	1261	Stage
5	1536	269	Stage
6	1294	3700	Stage
7	1203	4231	Stage
8	3031	667	Stage
9	1174	773	Stage
10	2695	1215	Stage
11	1741	2029	Stage
12	1009	1673	Stage
13	1348	847	Stage

Structural Framing Schedule

Mark	End Length	Start Length	Comments
14	647	1671	Stage
15	1018	791	Stage
16	1172	758	Stage
17	1712	794	Stage
18	612	673	Stage
19	1080	631	Stage
0	1067	2373	Skelter
1	1406	305	Skelter
20	755	1327	Stage
21	493	2201	Stage
2	1925	1991	Skelter
3	760	2416	Skelter
4	600	1098	Skelter
5	2668	1530	Skelter

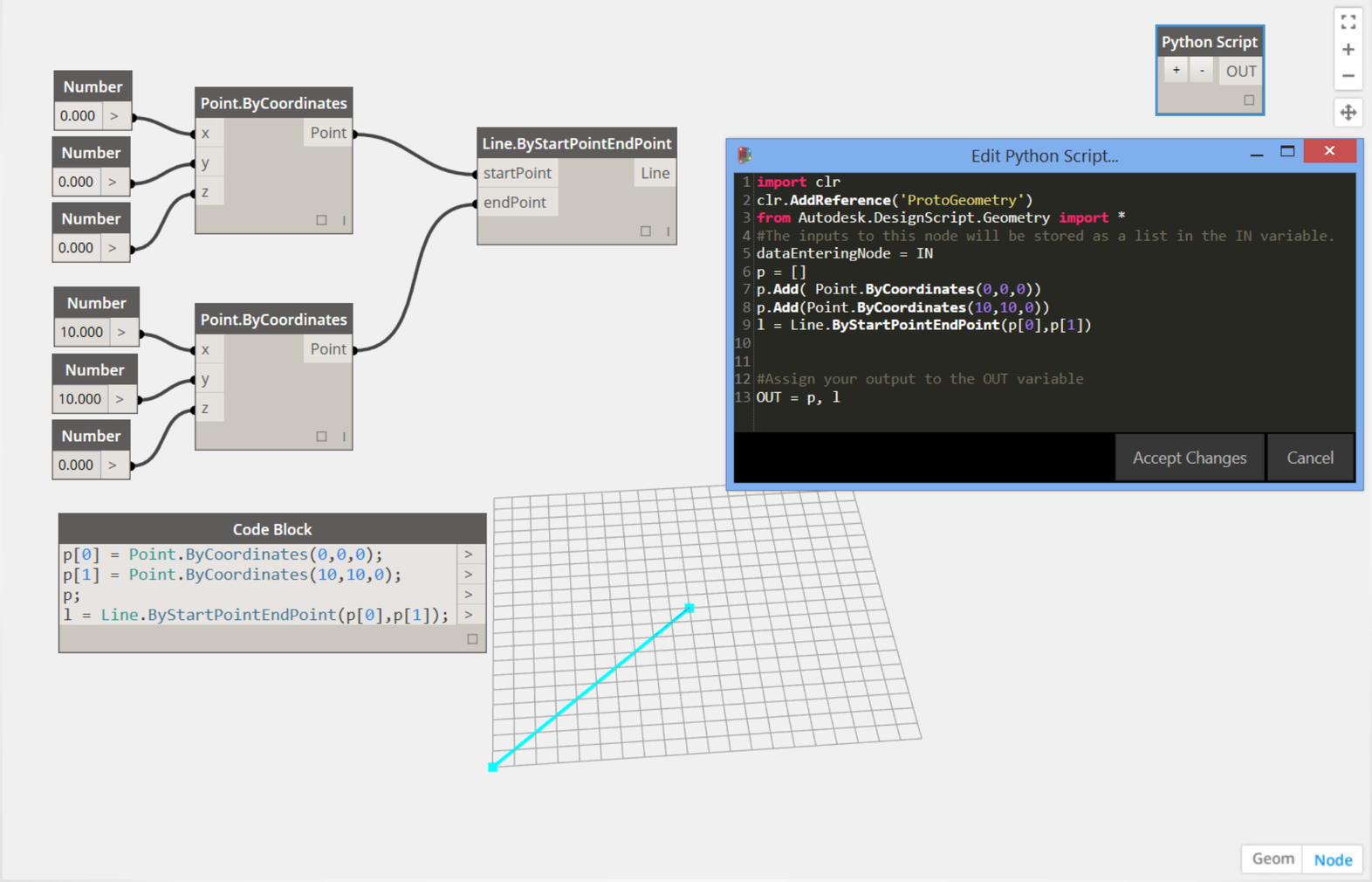
Structural Framing Schedule

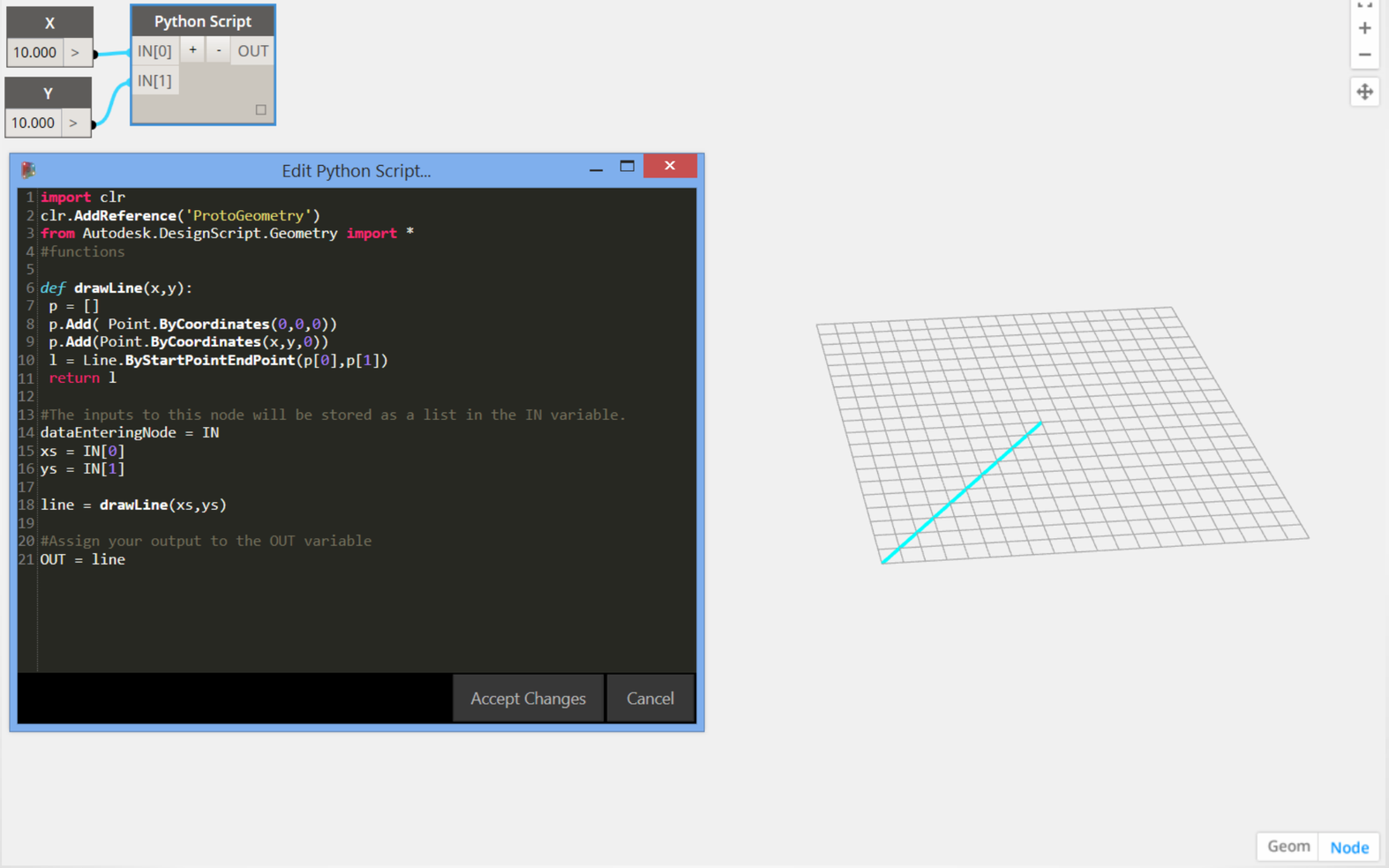
Mark	End Length	Start Length	Comments
6	728	970	Skelter
7	492	195	Skelter
8	1877	1977	Skelter
9	1918	2267	Skelter
10	846	1604	Skelter
11	2663	1509	Skelter
12	1897	2525	Skelter
13	2522	931	Skelter
14	1319	367	Skelter
15	1844	1820	Skelter
16	610	527	Skelter
17	1511	984	Skelter
18	876	162	Skelter
19	610	527	Skelter

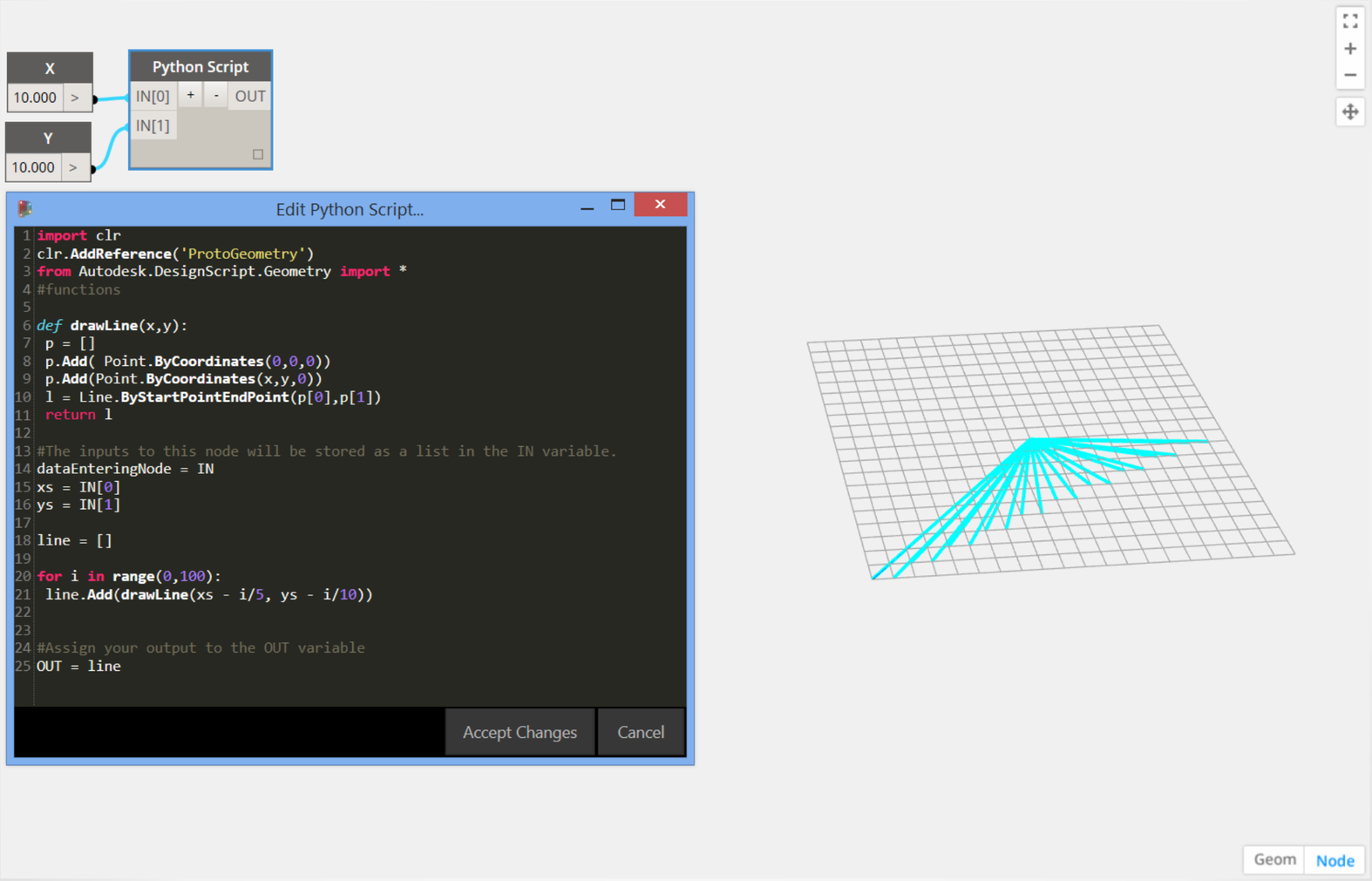


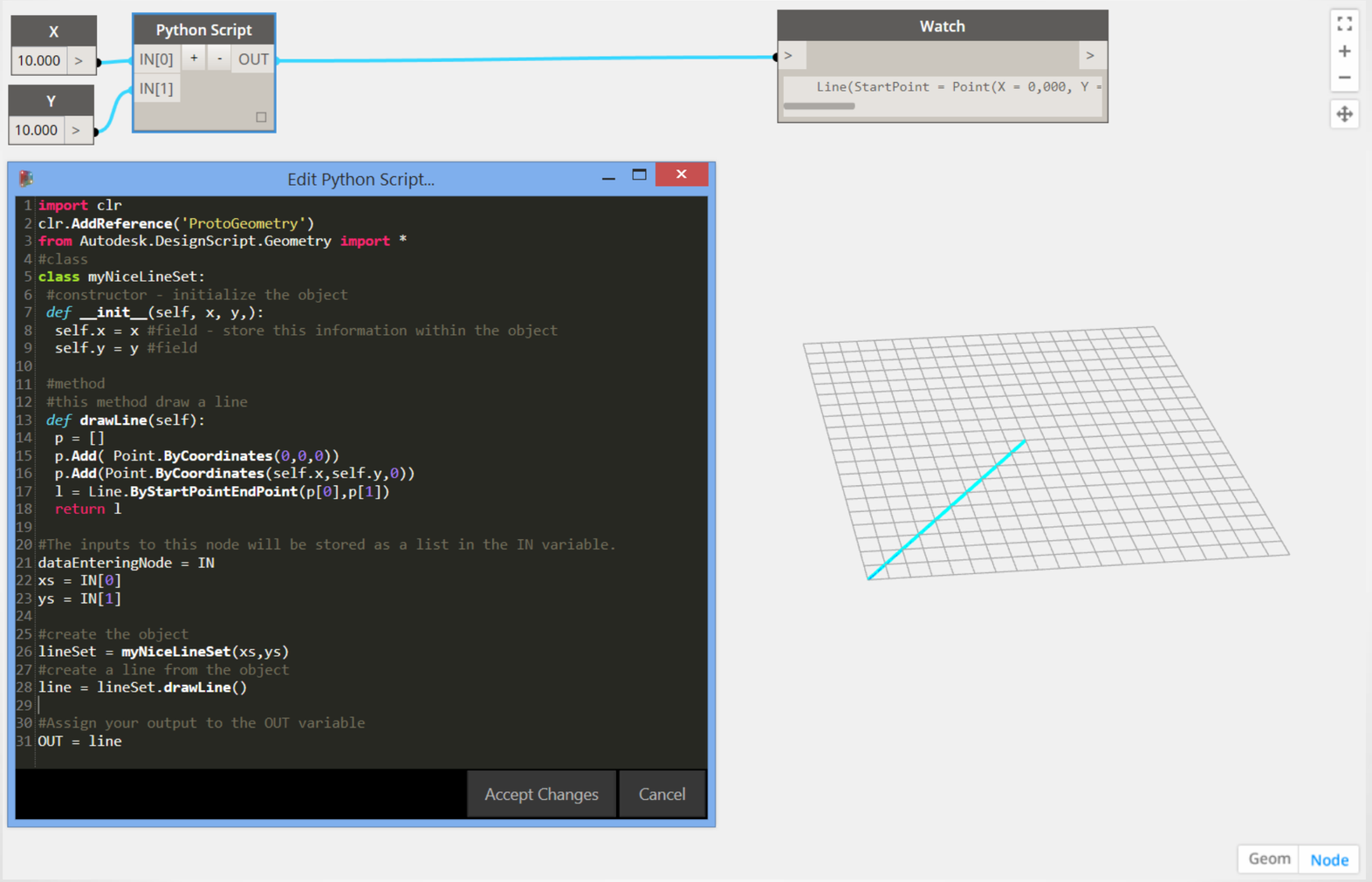
Python

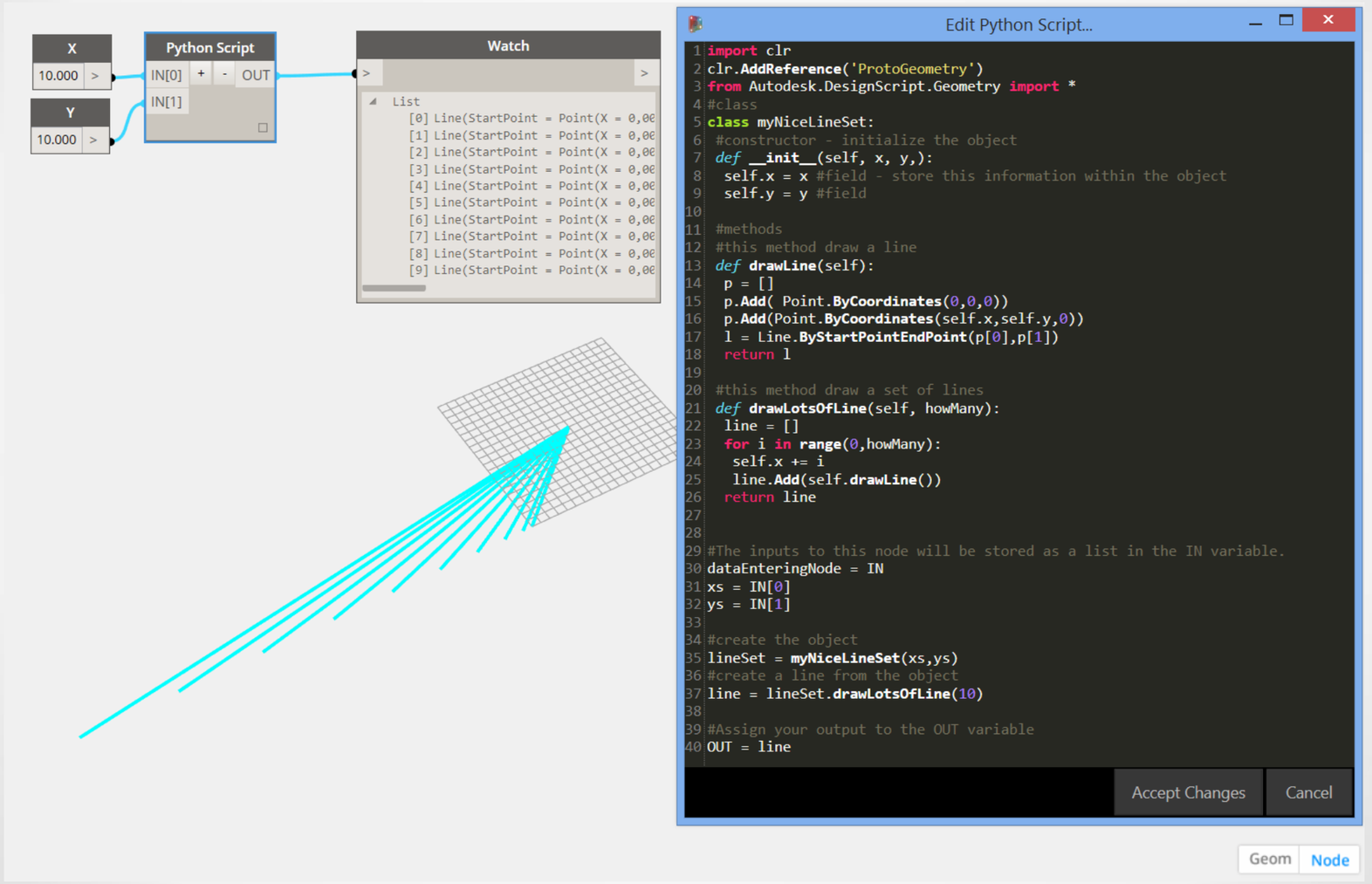
Comparison


Number
0.000
Number
0.000
Number
0.000
Point.ByCoordinates
x
y
z
Point
Line.ByStartPointEndPoint
startPoint
endPoint
Line
Number
10.000
Number
10.000
Number
0.000
Point.ByCoordinates
x
y
z
Point
Python Script
+
-
OUT
Edit Python Script...
1 import clr
2 clr.AddReference('ProtoGeometry')
3 from Autodesk.DesignScript.Geometry import *
4 #The inputs to this node will be stored as a list in the IN variable.
5 dataEnteringNode = IN
6 p = []
7 p.Add(Point.ByCoordinates(0,0,0))
8 p.Add(Point.ByCoordinates(10,10,0))
9 l = Line.ByStartPointEndPoint(p[0],p[1])
10
11
12 #Assign your output to the OUT variable
13 OUT = p, l
Accept Changes
Cancel
Code Block
p[0] = Point.ByCoordinates(0,0,0);
p[1] = Point.ByCoordinates(10,10,0);
p;
l = Line.ByStartPointEndPoint(p[0],p[1]);
Geom
Node


X
10.000
>
Y
10.000
>
Python Script
IN[0]
+
-
OUT
IN[1]
Edit Python Script...
1 import clr
2 clr.AddReference('ProtoGeometry')
3 from Autodesk.DesignScript.Geometry import *
4 #functions
5
6 def drawLine(x,y):
7 p = []
8 p.Add(Point.ByCoordinates(0,0,0))
9 p.Add(Point.ByCoordinates(x,y,0))
10 l = Line.ByStartPointEndPoint(p[0],p[1])
11 return l
12
13 #The inputs to this node will be stored as a list in the IN variable.
14 dataEnteringNode = IN
15 xs = IN[0]
16 ys = IN[1]
17
18 line = drawLine(xs,ys)
19
20 #Assign your output to the OUT variable
21 OUT = line
Accept Changes
Cancel
Geom
Node


X
10.000
>
Y
10.000
>
Python Script
IN[0]
+
-
OUT
IN[1]
Edit Python Script...
1 import clr
2 clr.AddReference('ProtoGeometry')
3 from Autodesk.DesignScript.Geometry import *
4 #functions
5
6 def drawLine(x,y):
7 p = []
8 p.Add(Point.ByCoordinates(0,0,0))
9 p.Add(Point.ByCoordinates(x,y,0))
10 l = Line.ByStartPointEndPoint(p[0],p[1])
11 return l
12
13 #The inputs to this node will be stored as a list in the IN variable.
14 dataEnteringNode = IN
15 xs = IN[0]
16 ys = IN[1]
17
18 line = []
19
20 for i in range(0,100):
21 line.Add(drawLine(xs - i/5, ys - i/10))
22
23
24 #Assign your output to the OUT variable
25 OUT = line
Accept Changes
Cancel
Geom
Node


X
10.000
Y
10.000
Python Script
IN[0]
IN[1]
OUT
Watch
Line(StartPoint = Point(X = 0,000, Y =
Edit Python Script...
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#class
class myNiceLineSet:
#constructor - initialize the object
def __init__(self, x, y,):
self.x = x #field - store this information within the object
self.y = y #field
#method
#this method draw a line
def drawLine(self):
p = []
p.Add(Point.ByCoordinates(0,0,0))
p.Add(Point.ByCoordinates(self.x,self.y,0))
l = Line.ByStartPointEndPoint(p[0],p[1])
return l
#The inputs to this node will be stored as a list in the IN variable.
dataEnteringNode = IN
xs = IN[0]
ys = IN[1]
#create the object
lineSet = myNiceLineSet(xs,ys)
#create a line from the object
line = lineSet.drawLine()
#Assign your output to the OUT variable
OUT = line
Accept Changes
Cancel
Geom
Node

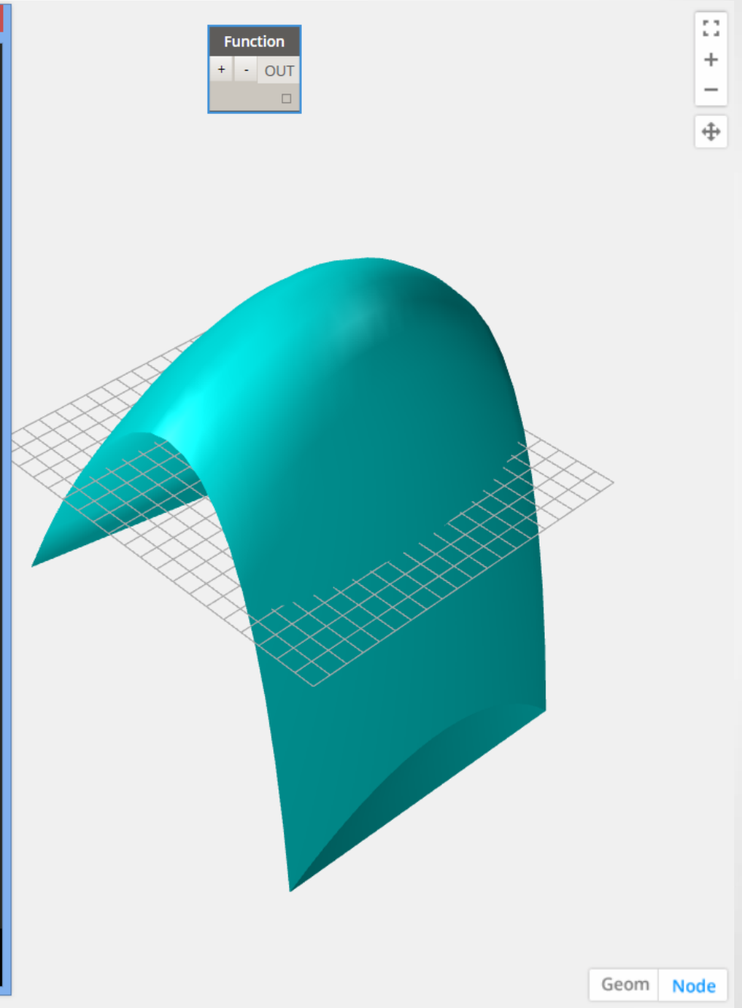

X
10.000
Y
10.000
Python Script
IN[0]
IN[1]
OUT
Watch
List
[0] Line(StartPoint = Point(X = 0,00
[1] Line(StartPoint = Point(X = 0,00
[2] Line(StartPoint = Point(X = 0,00
[3] Line(StartPoint = Point(X = 0,00
[4] Line(StartPoint = Point(X = 0,00
[5] Line(StartPoint = Point(X = 0,00
[6] Line(StartPoint = Point(X = 0,00
[7] Line(StartPoint = Point(X = 0,00
[8] Line(StartPoint = Point(X = 0,00
[9] Line(StartPoint = Point(X = 0,00
Edit Python Script...
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#class
class myNiceLineSet:
#constructor - initialize the object
def __init__(self, x, y,):
self.x = x #field - store this information within the object
self.y = y #field
#methods
#this method draw a line
def drawLine(self):
p = []
p.Add(Point.ByCoordinates(0,0,0))
p.Add(Point.ByCoordinates(self.x,self.y,0))
l = Line.ByStartPointEndPoint(p[0],p[1])
return l
#this method draw a set of lines
def drawLotsOfLine(self, howMany):
line = []
for i in range(0,howMany):
self.x += i
line.Add(self.drawLine())
return line
#The inputs to this node will be stored as a list in the IN variable.
dataEnteringNode = IN
xs = IN[0]
ys = IN[1]
#create the object
lineSet = myNiceLineSet(xs,ys)
#create a line from the object
line = lineSet.drawLotsOfLine(10)
#Assign your output to the OUT variable
OUT = line
Accept Changes
Cancel
Geom
Node

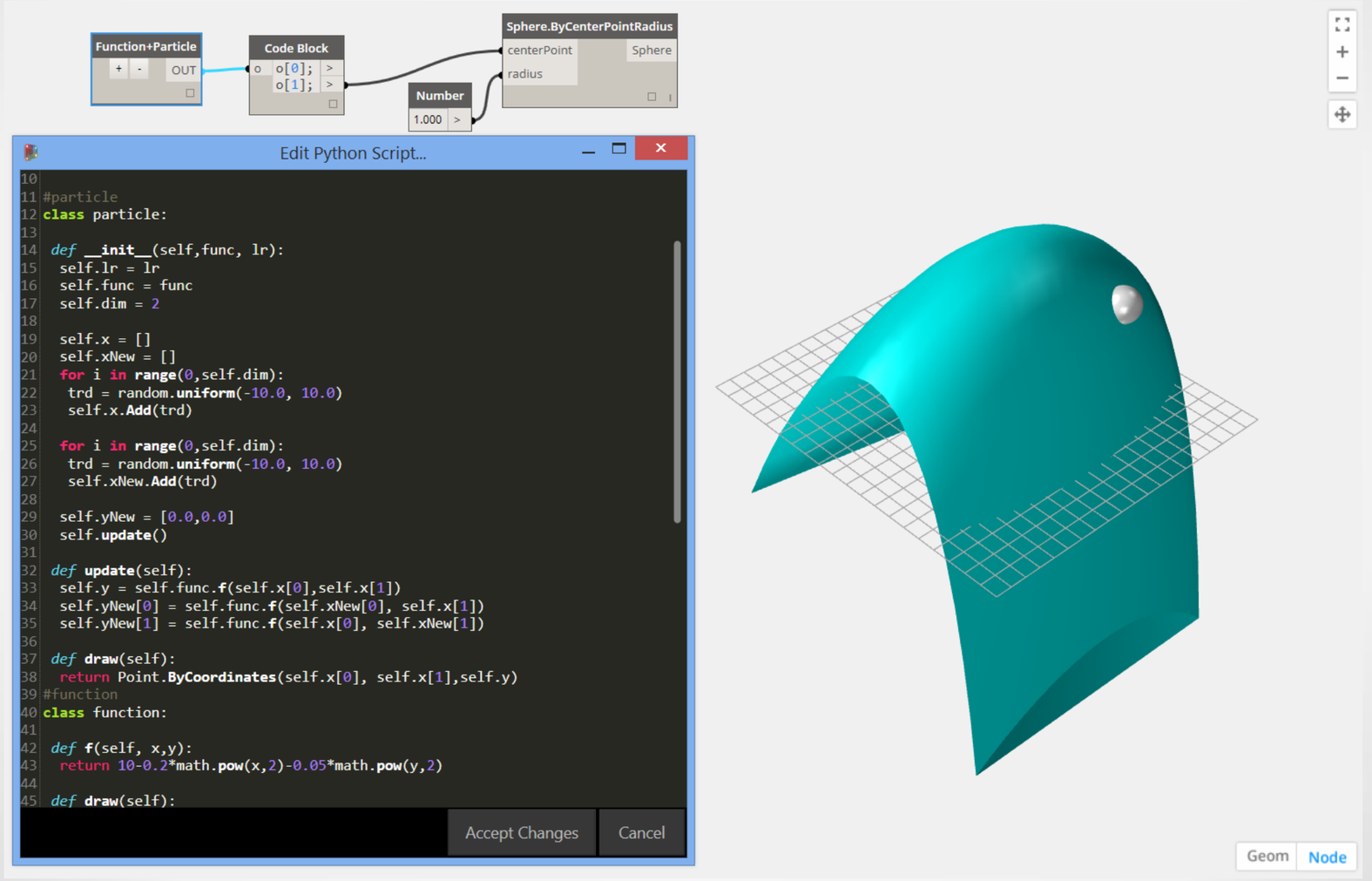
Gradient descent/climbing

Edit Python Script...

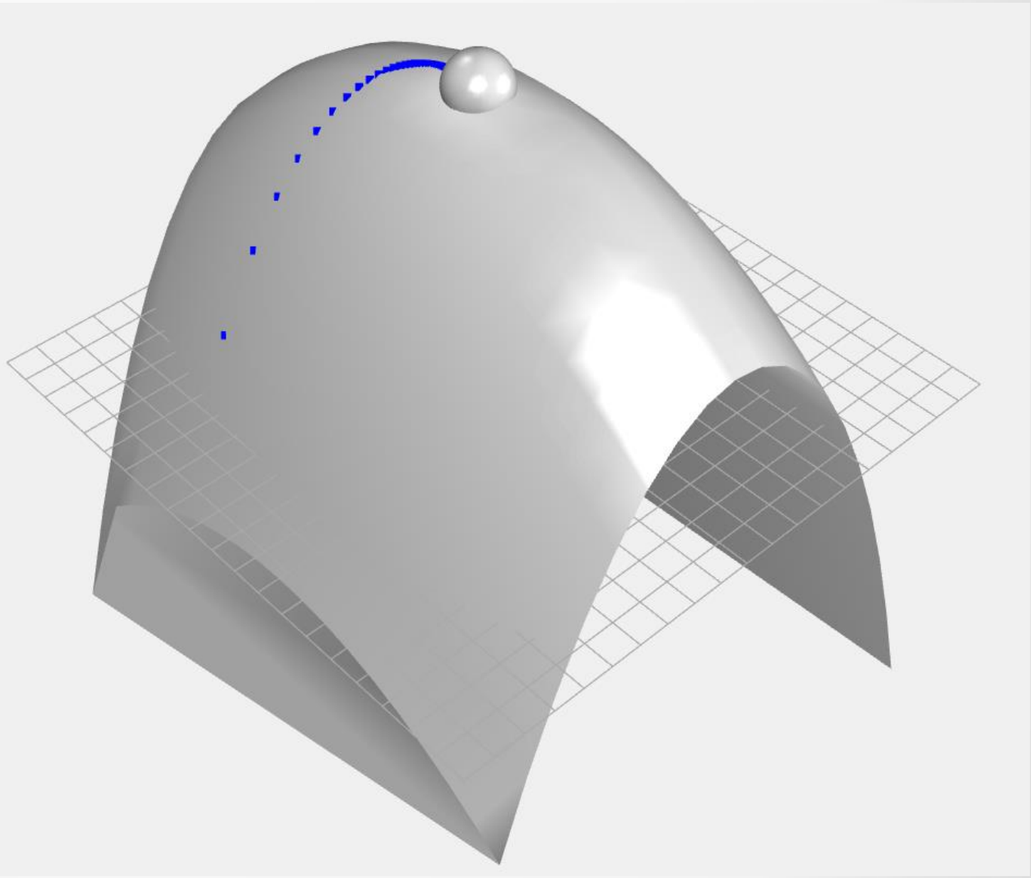
```python

#function
class function:

 def f(self, x,y):
  return 10-0.2*math.pow(x,2)-0.05*math.pow(y,2)

 def draw(self):
  pts = []
  vi = []
  sc = 1.0
  step = 10.0/sc
  st = int(step)
  #points
  for i in range (-st,st):
   for j in range (-st,st):
    xx = i * sc
    yy = j * sc
    zz = self.f(xx,yy)
    pti = Point.ByCoordinates(xx,yy,zz)
    pts.Add(pti)

  #vertex
  n = 2 * st-1
  for i in range (0,n):
   for j in range (0,n):
    a =j+n*i
    b = j+n*(i+1)
    c = (j+1)+n*(i+1)
    d = (j+1)+n*i

    ig = IndexGroup.ByIndices(a,b,c,d)
    vi.Add(ig)

  m = Mesh.ByPointsFaceIndices(pts,vi)
  return m
#The inputs to this node will be stored as a list in the IN variable.
dataEnteringNode = IN

f = function()

#Assign your output to the OUT variable
OUT = f.draw()
```

Accept Changes | Cancel

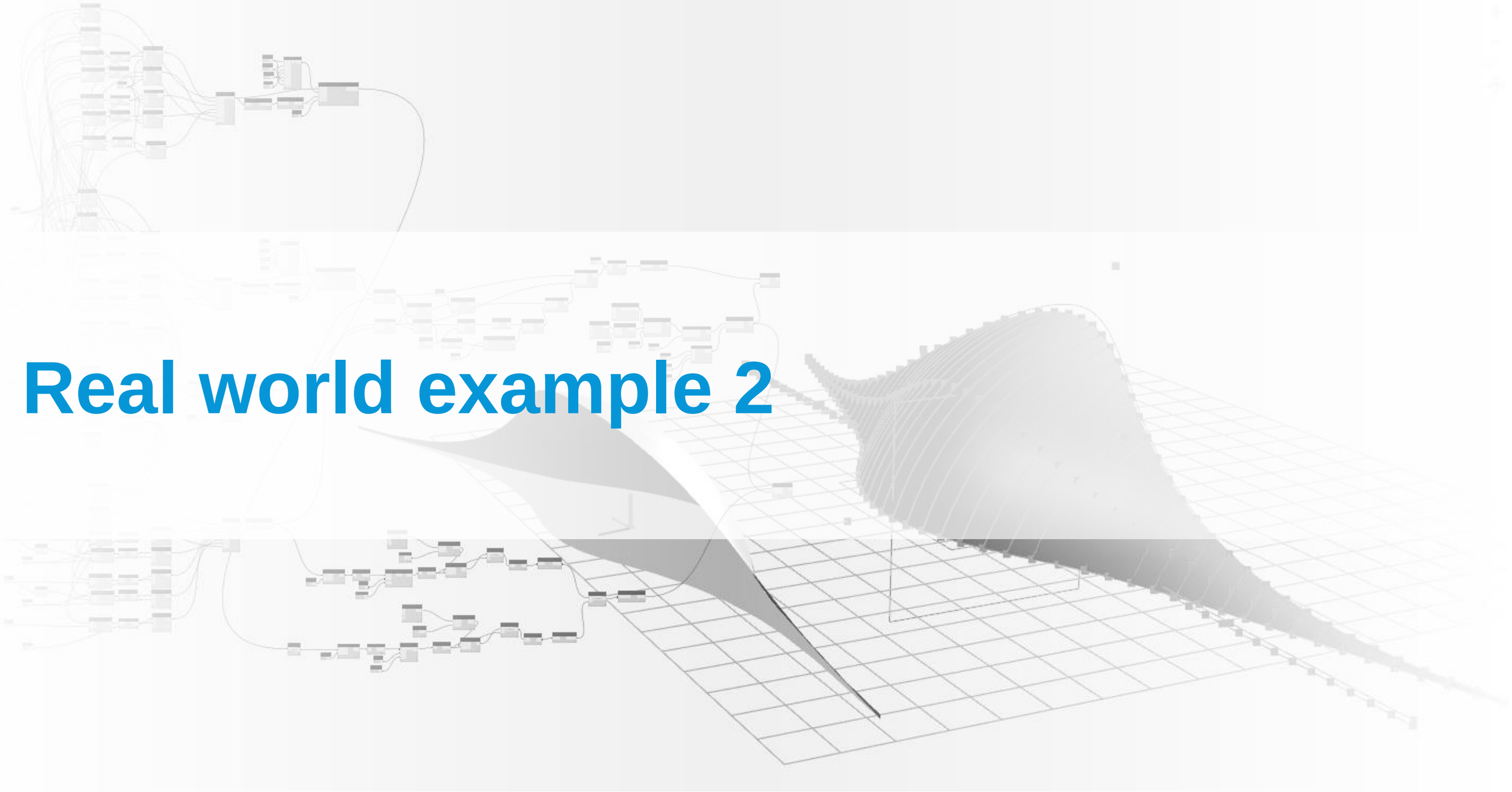



Function+Particle
+
-
OUT
Code Block
o
o[0];
o[1];
Number
1.000
Sphere.ByCenterPointRadius
centerPoint
radius
Sphere
Edit Python Script...
#particle
class particle:
def __init__(self,func, lr):
self.lr = lr
self.func = func
self.dim = 2
self.x = []
self.xNew = []
for i in range(0,self.dim):
trd = random.uniform(-10.0, 10.0)
self.x.Add(trd)
for i in range(0,self.dim):
trd = random.uniform(-10.0, 10.0)
self.xNew.Add(trd)
self.yNew = [0.0,0.0]
self.update()
def update(self):
self.y = self.func.f(self.x[0],self.x[1])
self.yNew[0] = self.func.f(self.xNew[0], self.x[1])
self.yNew[1] = self.func.f(self.x[0], self.xNew[1])
def draw(self):
return Point.ByCoordinates(self.x[0], self.x[1],self.y)
#function
class function:
def f(self, x,y):
return 10-0.2*math.pow(x,2)-0.05*math.pow(y,2)
def draw(self):
Accept Changes
Cancel
Geom
Node

```
#particle
class particle:

 def __init__(self,func, lr):
  self.lr = lr
  self.func = func
  self.dim = 2

  self.x = []
  self.xNew = []
  for i in range(0,self.dim):
   trd = random.uniform(-10.0, 10.0)
   self.x.Add(trd)

  for i in range(0,self.dim):
   trd = random.uniform(-10.0, 10.0)
   self.xNew.Add(trd)

  self.yNew = [0.0,0.0]
  self.update()

 def update(self):
  self.y = self.func.f(self.x[0],self.x[1])
  self.yNew[0] = self.func.f(self.xNew[0], self.x[1])
  self.yNew[1] = self.func.f(self.x[0], self.xNew[1])

 def draw(self):
  return Point.ByCoordinates(self.x[0], self.x[1],self.y)

 def optimize(self):
  for i in range(0,self.dim):
   delta =self.xNew[i] - self.x[i]
   if (math.fabs(delta) > 0.01):
    der = (self.yNew[i]-self.y)/(delta)
    self.x[i] = self.xNew[i]
    self.xNew[i] = self.x[i] + self.lr * der
    self.update()
```

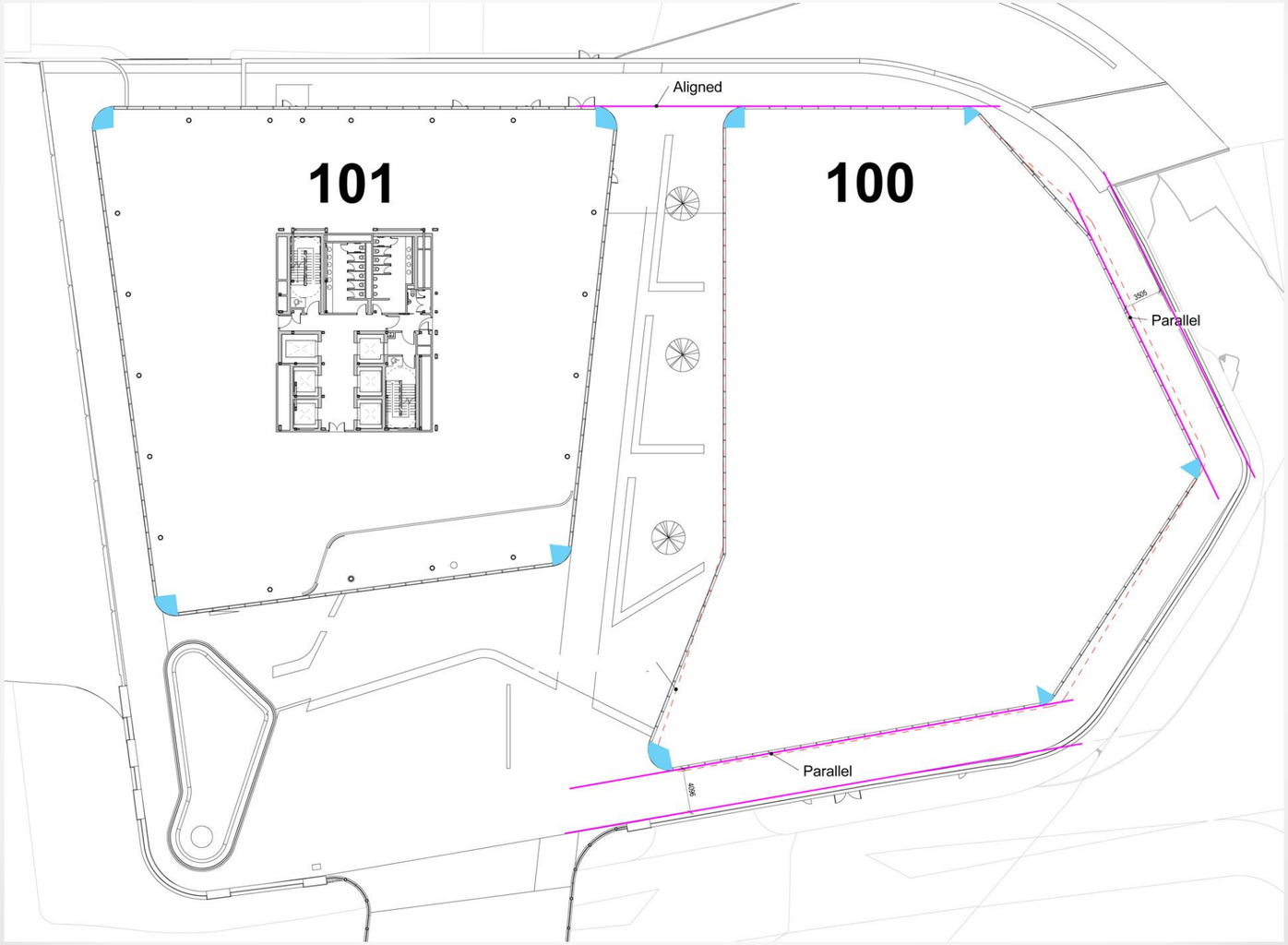


Real world example 2



Façade optimization

Site Constraints



GG-Curtain Panel - Flat - Mullion

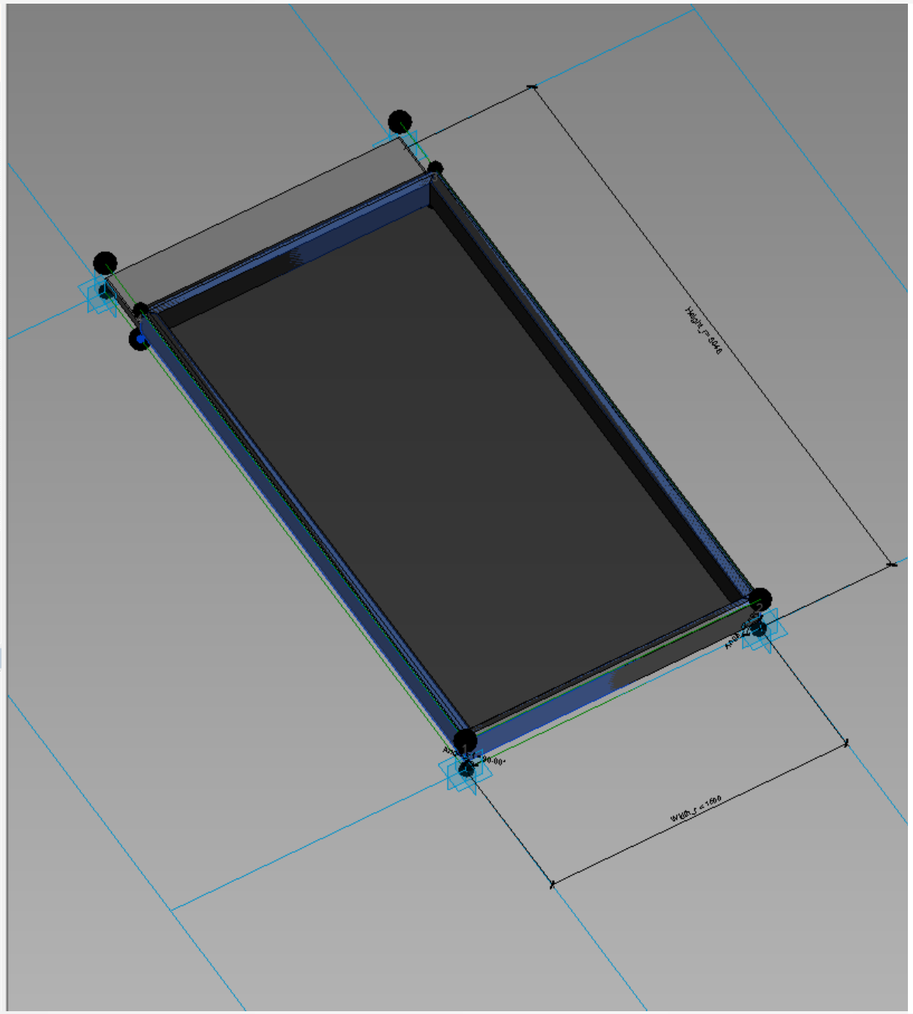
Generic Models (1) | Edit Type

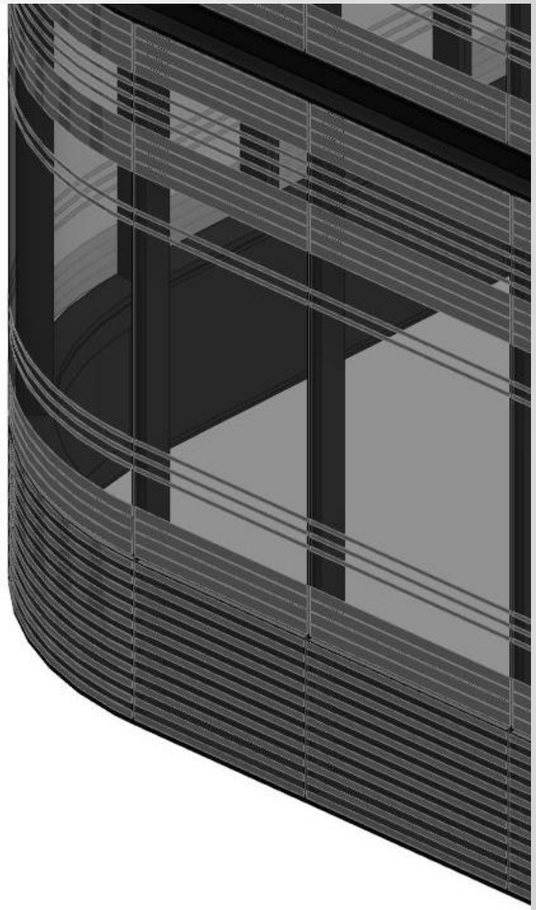
Constraints	
Level	Level 1
Elevation	0.0
Moves With Nearby Elements	
Graphics	
Visibility/Graphics Overrides	Edit...
Visible	
Materials and Finishes	
Material	GG-Curtain Mullion
Dimensions	
Mullion Thickness	50.0
Mullion Offset	52.0
Mullion Height	201.0
Volume	0.062 m³
Identity Data	
Comments	
Mark	
Adaptive Component	
Flip	
Other	
Label	<None>

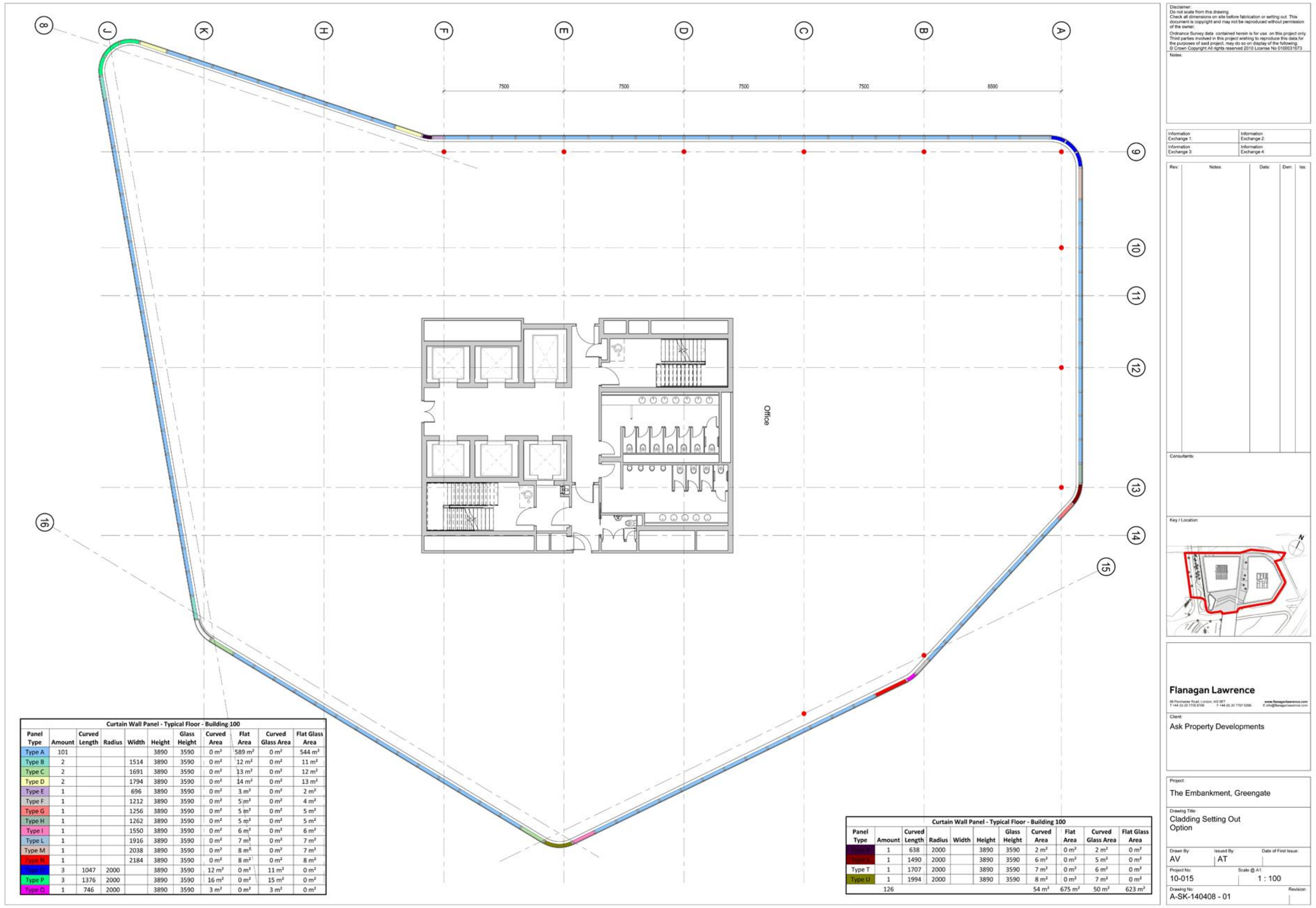
Properties help | Apply

Project Browser - GG-Curtain Panel - Pattern Based - Fritted Flat_52mm.rfa

- Views (all)
  - Floor Plans
    - Level 1
  - 3D Views
- Sheets (all)
- Families







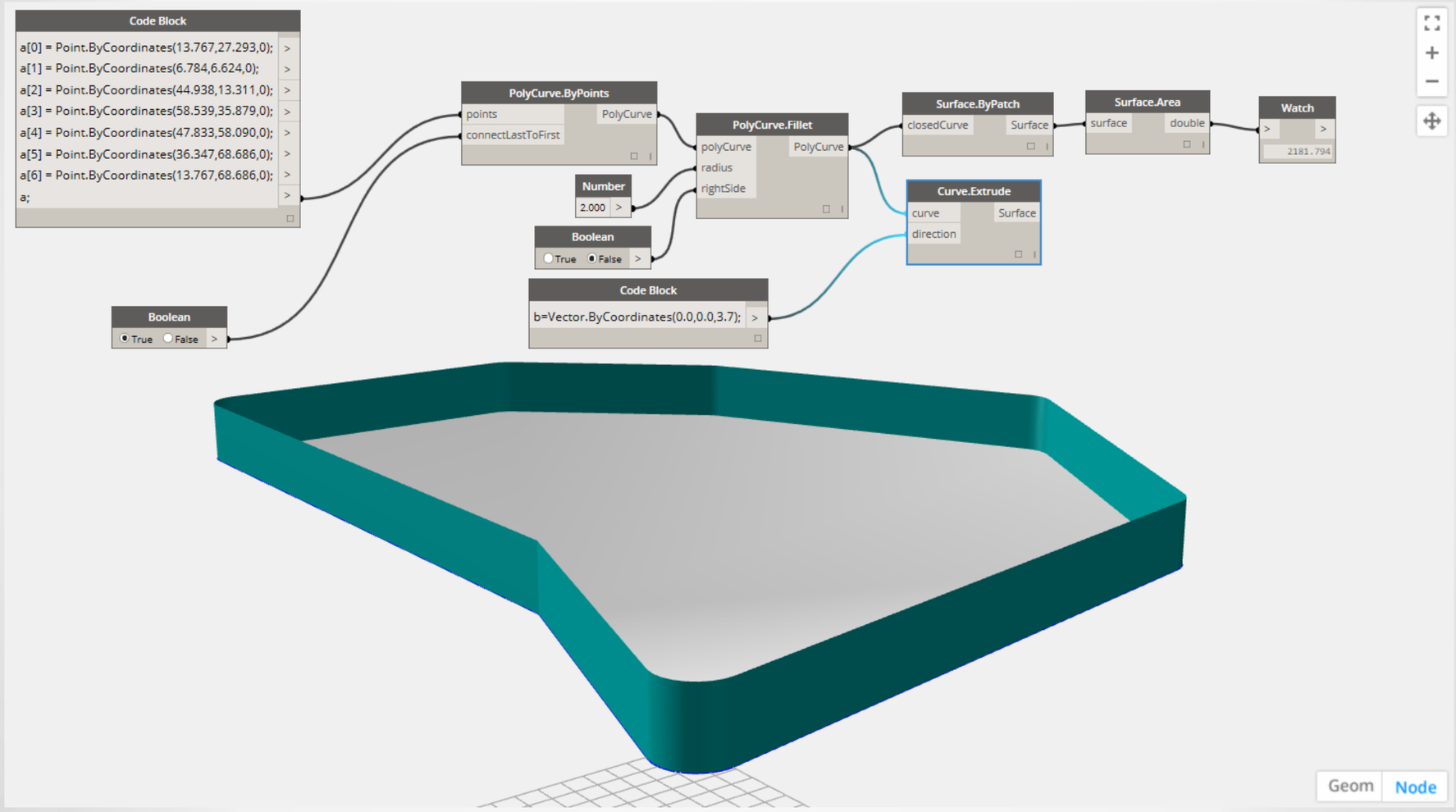
Curtain Wall Panel - Typical Floor - Building 100

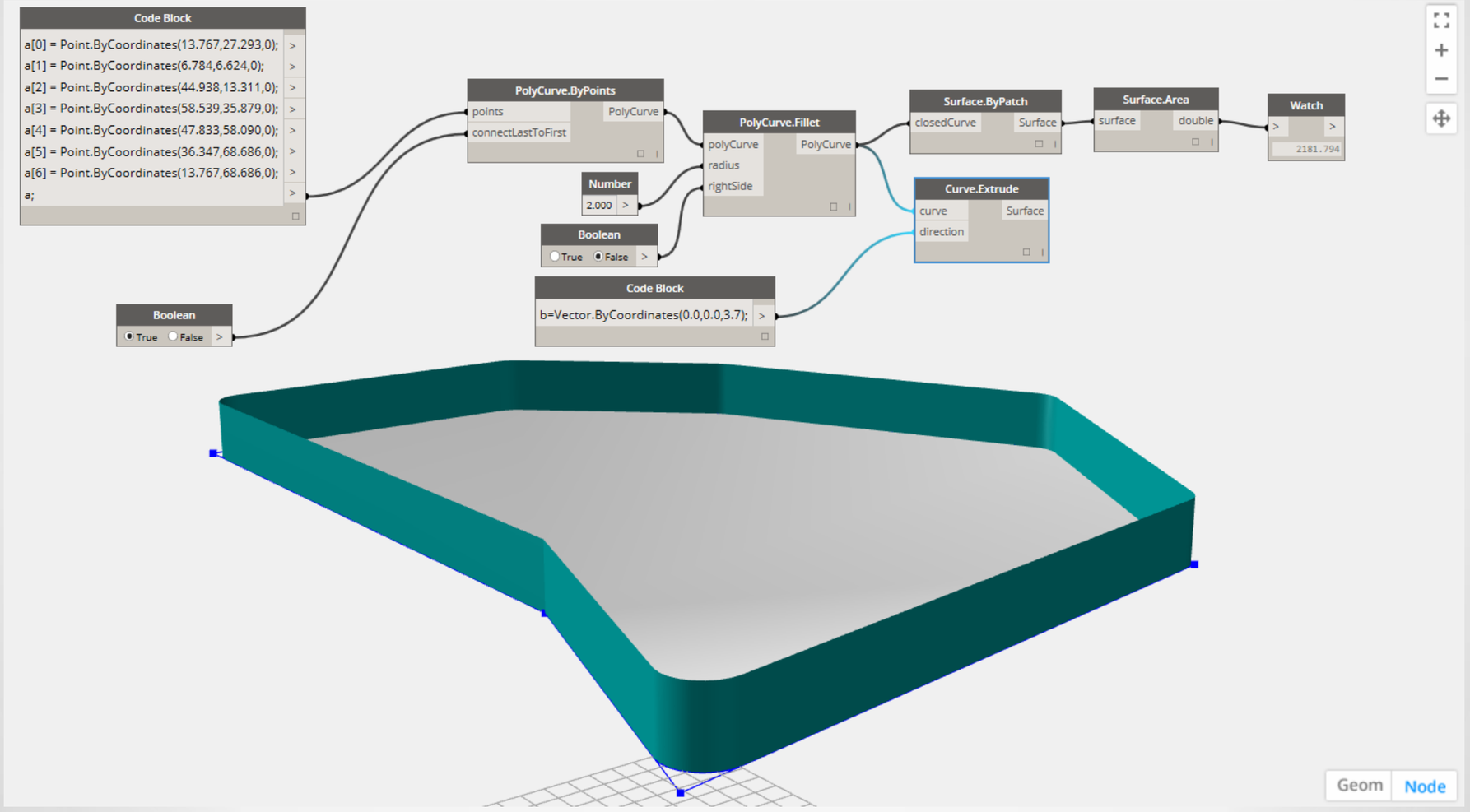
Panel Type	Amount	Curved Length	Radius	Width	Height	Glass Height	Curved Area	Flat Area	Curved Glass Area	Flat Glass Area
Type A	101				3890	3590	0 m²	589 m²	0 m²	544 m²
Type B	2			1514	3890	3590	0 m²	12 m²	0 m²	11 m²
Type C	2			1691	3890	3590	0 m²	13 m²	0 m²	12 m²
Type D	2			1794	3890	3590	0 m²	14 m²	0 m²	13 m²
Type E	1			696	3890	3590	0 m²	3 m²	0 m²	2 m²
Type F	1			1212	3890	3590	0 m²	5 m²	0 m²	4 m²
Type G	1			1256	3890	3590	0 m²	5 m²	0 m²	5 m²
Type H	1			1262	3890	3590	0 m²	5 m²	0 m²	5 m²
Type I	1			1550	3890	3590	0 m²	6 m²	0 m²	6 m²
Type L	1			1916	3890	3590	0 m²	7 m²	0 m²	7 m²
Type M	1			2038	3890	3590	0 m²	8 m²	0 m²	7 m²
Type N	1			2184	3890	3590	0 m²	8 m²	0 m²	8 m²
Type O	3	1047	2000		3890	3590	12 m²	0 m²	11 m²	0 m²
Type P	3	1376	2000		3890	3590	16 m²	0 m²	15 m²	0 m²
Type Q	1	746	2000		3890	3590	3 m²	0 m²	3 m²	0 m²

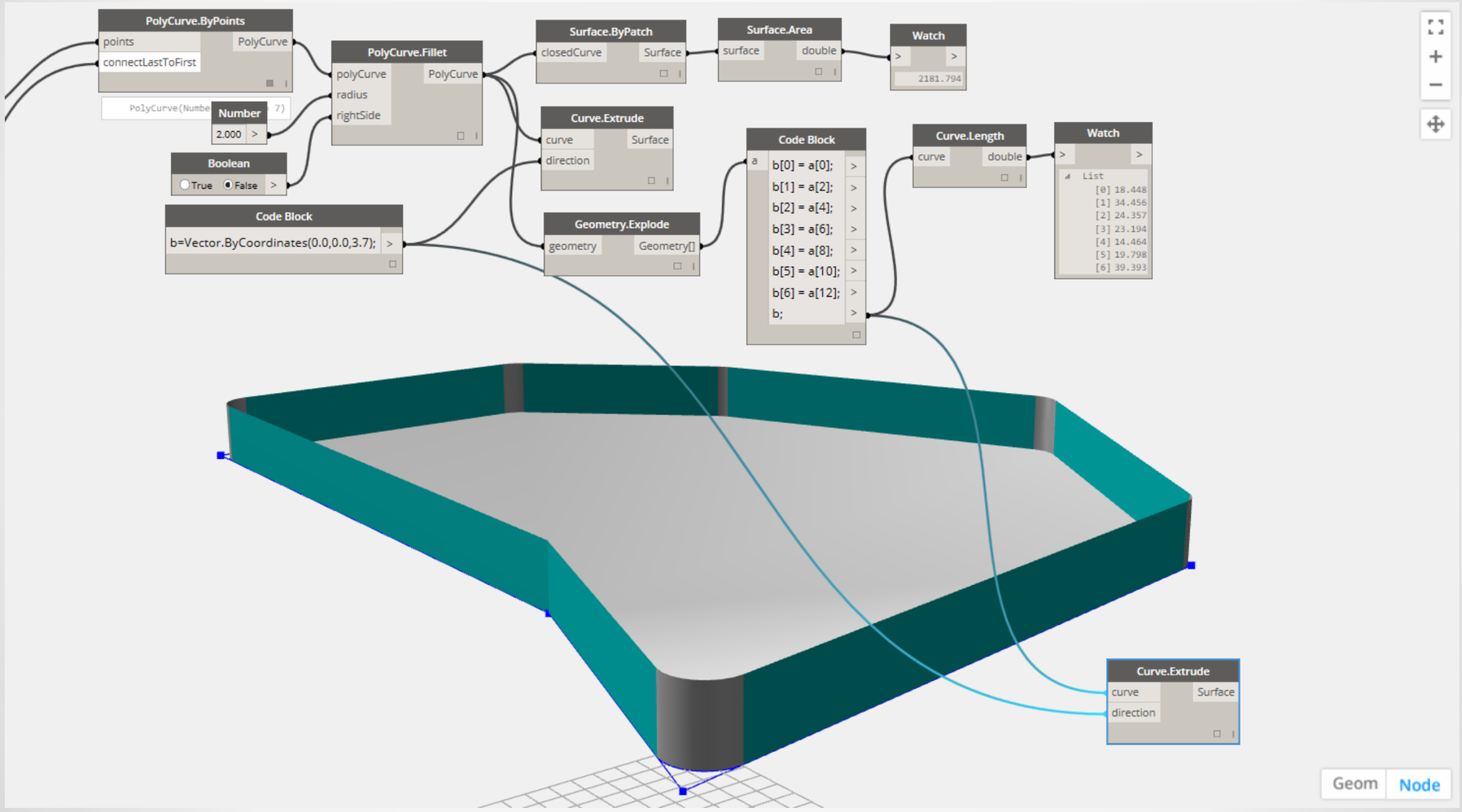
Curtain Wall Panel - Typical Floor - Building 100

Panel Type	Amount	Curved Length	Radius	Width	Height	Glass Height	Curved Area	Flat Area	Curved Glass Area	Flat Glass Area
Type R	1	638	2000		3890	3590	2 m²	0 m²	2 m²	0 m²
Type S	1	1490	2000		3890	3590	6 m²	0 m²	5 m²	0 m²
Type T	1	1707	2000		3890	3590	7 m²	0 m²	6 m²	0 m²
Type U	1	1994	2000		3890	3590	8 m²	0 m²	7 m²	0 m²
	126						54 m²	675 m²	50 m²	623 m²

Curtain Wall Panel - Typical Floor - Building 100										
Panel Type	Amount	Curved Length	Radius	Width	Height	Glass Height	Curved Area	Flat Area	Curved Glass Area	Flat Glass Area
Type A	101				3890	3590	0 m²	589 m²	0 m²	544 m²
Type B	2			1514	3890	3590	0 m²	12 m²	0 m²	11 m²
Type C	2			1691	3890	3590	0 m²	13 m²	0 m²	12 m²
Type D	2			1794	3890	3590	0 m²	14 m²	0 m²	13 m²
Type E	1			696	3890	3590	0 m²	3 m²	0 m²	2 m²
Type F	1			1212	3890	3590	0 m²	5 m²	0 m²	4 m²
Type G	1			1256	3890	3590	0 m²	5 m²	0 m²	5 m²
Type H	1			1262	3890	3590	0 m²	5 m²	0 m²	5 m²
Type I	1			1550	3890	3590	0 m²	6 m²	0 m²	6 m²
Type L	1			1916	3890	3590	0 m²	7 m²	0 m²	7 m²
Type M	1			2038	3890	3590	0 m²	8 m²	0 m²	7 m²


Code Block
a[0] = Point.ByCoordinates(13.767,27.293,0);
a[1] = Point.ByCoordinates(6.784,6.624,0);
a[2] = Point.ByCoordinates(44.938,13.311,0);
a[3] = Point.ByCoordinates(58.539,35.879,0);
a[4] = Point.ByCoordinates(47.833,58.090,0);
a[5] = Point.ByCoordinates(36.347,68.686,0);
a[6] = Point.ByCoordinates(13.767,68.686,0);
a;
PolyCurve.ByPoints
points
connectLastToFirst
PolyCurve
PolyCurve.Fillet
polyCurve
radius
rightSide
PolyCurve
Surface.ByPatch
closedCurve
Surface
Surface.Area
surface
double
Watch
2181.794
Number
2.000
Boolean
True
False
Boolean
True
False
Code Block
b=Vector.ByCoordinates(0.0,0.0,3.7);
Curve.Extrude
curve
direction
Surface
Geom
Node

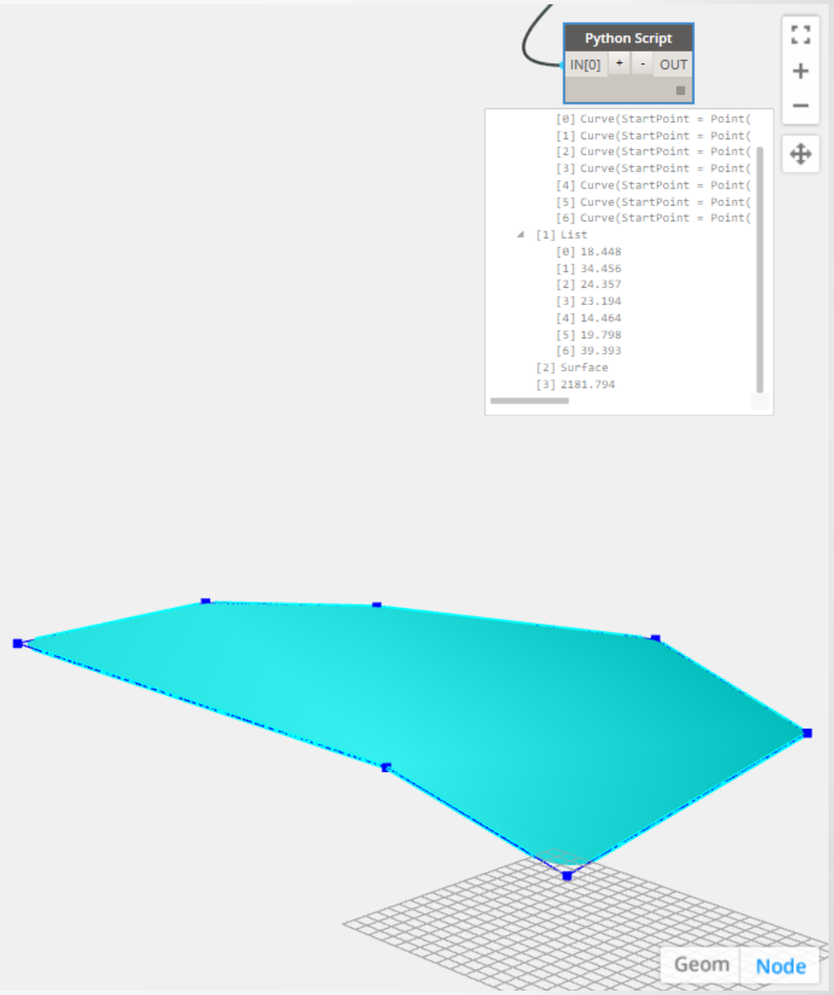

Code Block
a[0] = Point.ByCoordinates(13.767,27.293,0);
a[1] = Point.ByCoordinates(6.784,6.624,0);
a[2] = Point.ByCoordinates(44.938,13.311,0);
a[3] = Point.ByCoordinates(58.539,35.879,0);
a[4] = Point.ByCoordinates(47.833,58.090,0);
a[5] = Point.ByCoordinates(36.347,68.686,0);
a[6] = Point.ByCoordinates(13.767,68.686,0);
a;
PolyCurve.ByPoints
points
connectLastToFirst
PolyCurve
PolyCurve.Fillet
polyCurve
radius
rightSide
PolyCurve
Number
2.000
Boolean
True
False
Boolean
True
False
Code Block
b=Vector.ByCoordinates(0.0,0.0,3.7);
Curve.Extrude
curve
direction
Surface
Surface.ByPatch
closedCurve
Surface
Surface.Area
surface
double
Watch
2181.794
Geom
Node

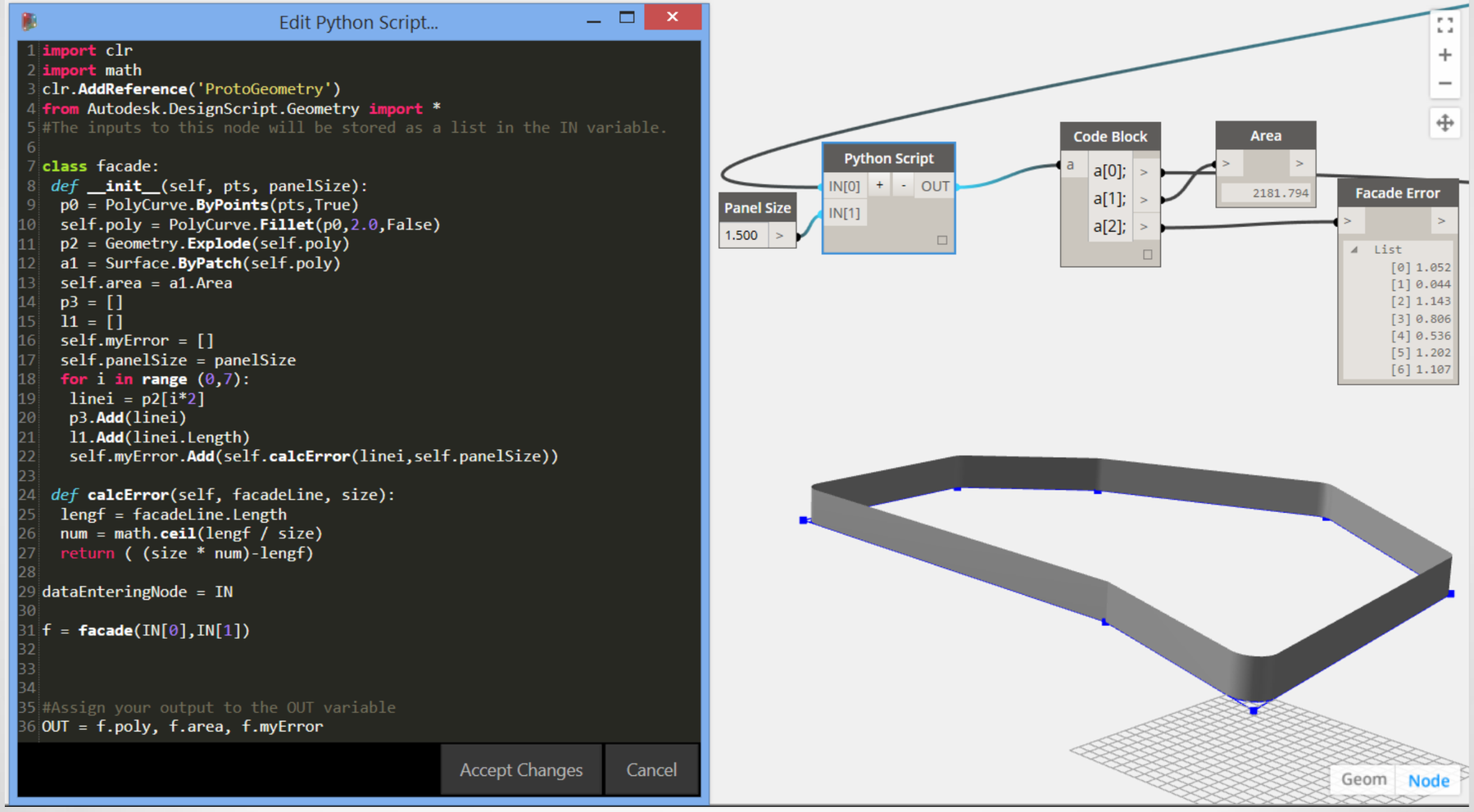

PolyCurve.ByPoints
points
connectLastToFirst
PolyCurve
Number
2.000
Boolean
True
False
PolyCurve.Fillet
polyCurve
radius
rightSide
PolyCurve
Surface.ByPatch
closedCurve
Surface
Surface.Area
surface
double
Watch
2181.794
Curve.Extrude
curve
direction
Surface
Code Block
b=Vector.ByCoordinates(0.0,0.0,3.7);
Geometry.Explode
geometry
Geometry[]
Code Block
a
b[0] = a[0];
b[1] = a[2];
b[2] = a[4];
b[3] = a[6];
b[4] = a[8];
b[5] = a[10];
b[6] = a[12];
b;
Curve.Length
curve
double
Watch
List
[0] 18.448
[1] 34.456
[2] 24.357
[3] 23.194
[4] 14.464
[5] 19.798
[6] 39.393
Curve.Extrude
curve
direction
Surface
Geom
Node

Edit Python Script...

```python
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#The inputs to this node will be stored as a list in the IN variable.
dataEnteringNode = IN

a = IN[0]

p0 = PolyCurve.ByPoints(a,True)
p1 = PolyCurve.Fillet(p0,2.0,False)
p2 = Geometry.Explode(p1)
a1 = Surface.ByPatch(p1)
a2 = a1.Area
p3 = []
l1 = []
for i in range (0,7):
 p3.Add(p2[i*2])
 l1.Add(p2[i*2].Length)

#Assign your output to the OUT variable
OUT = p3, l1, a1, a2
```

Accept Changes | Cancel




Edit Python Script...
import clr
import math
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#The inputs to this node will be stored as a list in the IN variable.
class facade:
def __init__(self, pts, panelSize):
p0 = PolyCurve.ByPoints(pts,True)
self.poly = PolyCurve.Fillet(p0,2.0,False)
p2 = Geometry.Explode(self.poly)
a1 = Surface.ByPatch(self.poly)
self.area = a1.Area
p3 = []
l1 = []
self.myError = []
self.panelSize = panelSize
for i in range (0,7):
linei = p2[i*2]
p3.Add(linei)
l1.Add(linei.Length)
self.myError.Add(self.calcError(linei,self.panelSize))
def calcError(self, facadeLine, size):
lengf = facadeLine.Length
num = math.ceil(lengf / size)
return ((size * num)-lengf)
dataEnteringNode = IN
f = facade(IN[0],IN[1])
#Assign your output to the OUT variable
OUT = f.poly, f.area, f.myError
Accept Changes
Cancel
Panel Size
1.500
Python Script
IN[0]
+
-
OUT
IN[1]
Code Block
a
a[0];
a[1];
a[2];
Area
2181.794
Facade Error
List
[0] 1.052
[1] 0.044
[2] 1.143
[3] 0.806
[4] 0.536
[5] 1.202
[6] 1.107
Geom
Node

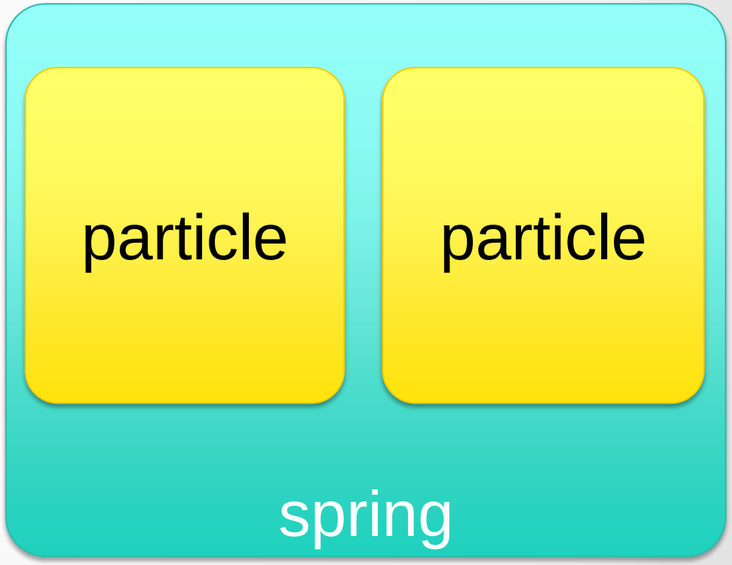
Corners

```
#control point class
#store position and constraints
class particle:
 def __init__(self, position, constraints):
  self.p = position
  self.c = constraints #1 free 0 constrained
  self.r = 0
 def move(self,e):
  pNew = Point.ByCoordinates(self.p.X + (e.X * self.c.X),
  self.p.Y + (e.Y * self.c.Y), 0.0)
  self.p = pNew

#spring class
#store the spring elements which connetcs the nodes
```

particle

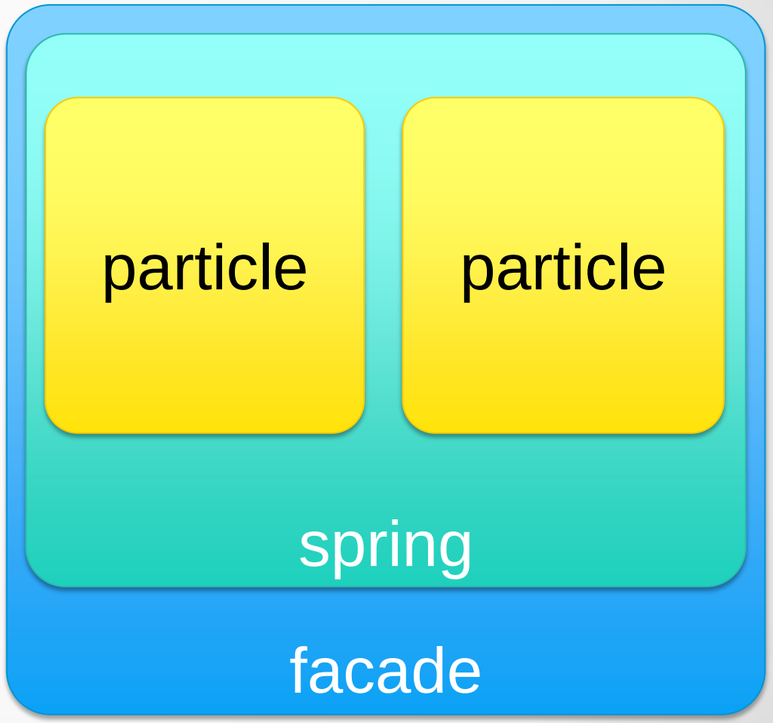
Individual facades

```
class spring:
 def __init__(self, n1, n2):
  self.a = n1
  self.b = n2
  self.vds = Vector.ByTwoPoints(self.b.p, self.a.p)
  self.lengf = 0.0
  self.error = 0.0
  #Subtract(self.a.p.AsVector(), self.b.p.AsVector())

 def update(self, size, lr, iter):
  #length
  self.vds = Vector.ByTwoPoints(self.b.p, self.a.p)
  self.lengf =  self.vds.Length
  self.lengf -= self.a.r
  self.lengf -= self.b.r

  #error
  self.error = self.calcError(self.lengf, size, iter)
  e1 = Vector.ByCoordinates(self.vds.X, self.vds.Y, 0.0)
  e1n = e1.Normalized()
  errorScale = self.error/2.0 * lr
  e1ns = e1n.Scale(-errorScale)
  e2ns = e1n.Scale(errorScale)
  self.a.move(e1ns)
  self.b.move(e2ns)

 def calcError(self, lengf, size, iter):
  num = 0
  if(iter < 5):
   num = int(math.ceil(lengf / size))
  else:
   num = int(round(lengf / size))
  return (lengf - (size * num))

```



Entire facade

```
class facade:
 def __init__(self, panelSize):
  self.cp = []
  self.sp = []
  self.panelSize = panelSize
  self.rad = 2.0
  self.maxError = []

 #add point to the polyline
 def addPt(self, pt,fx,fy):
  fsx = 0
  fsy = 0
  if(fx):
   fsx = 1.0
  else:
   fsx = 0.0
  if(fy):
   fsy = 1.0
  else:
   fsy = 0.0
  constr = Vector.ByCoordinates(fsx,fsy,0.0)
  self.cp.Add(particle(pt,constr))

 def generate(self):
  #generate springs
  for i in range (0,self.cp.Count):
   ni = self.cp[i]
   nj = self.cp[(i+1)%self.cp.Count]
   self.sp.Add(spring(ni,nj))

```

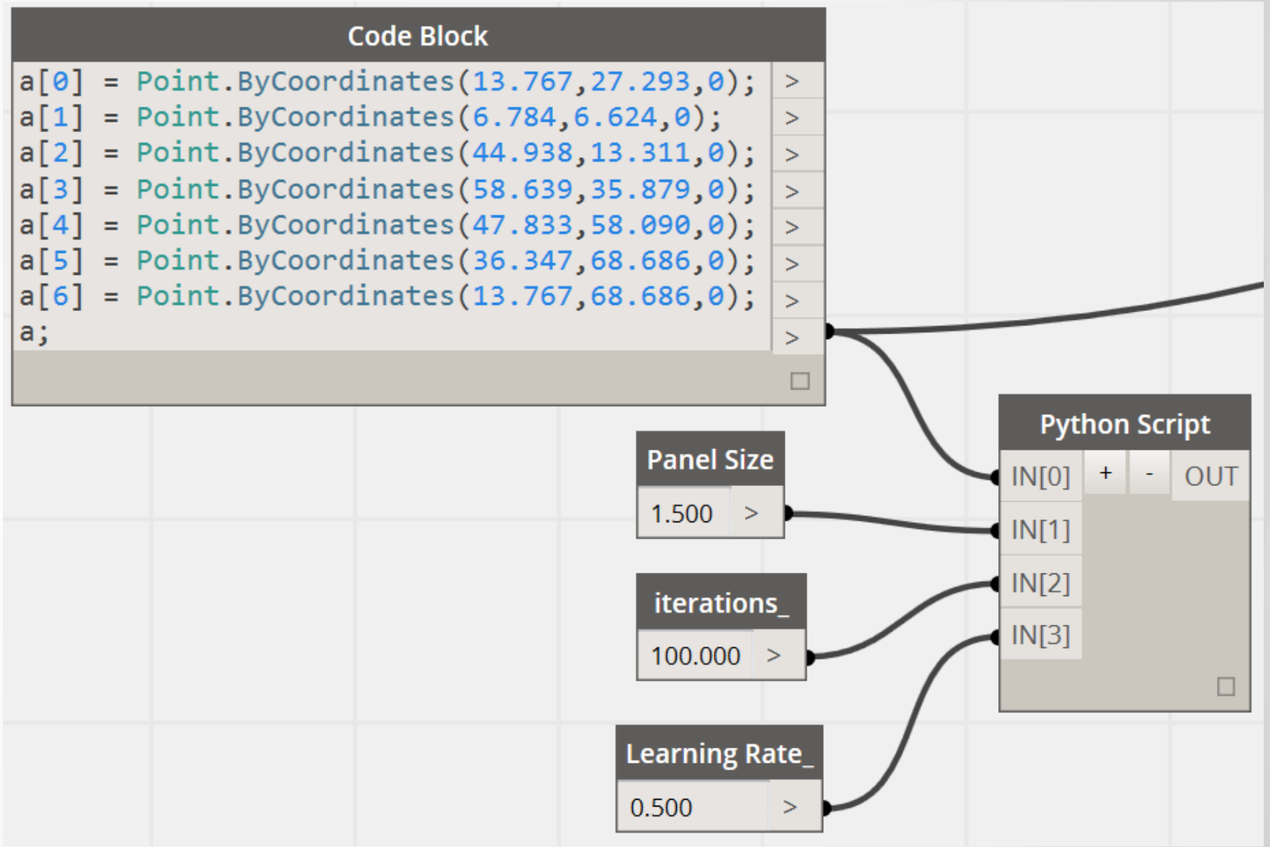


```
def update(self, iter):
 #update the corner reduction
 cps = self.cp.Count
 for i in range (0,cps):
  ip = i-1
  if i == 0:
   ip = cps -1
  iin = (i+1) % cps
  previousL = Vector.ByCoordinates(
  self.cp[i].p.X-self.cp[ip].p.X,
  self.cp[i].p.Y-self.cp[ip].p.Y,
  0.0)
  successiveL = Vector.ByCoordinates(
  self.cp[i].p.X-self.cp[iin].p.X,
  self.cp[i].p.Y-self.cp[iin].p.Y,
  0.0)
  angle = angleBetweenVectors(successiveL, previousL)
  shrink = self.rad  / (math.tan(angle/2.0))
  self.cp[i].r = shrink
 #3D model
 pts = []
 for i in range (0,cps):
  pti = self.cp[i].p
  ptix = Point.ByCoordinates(pti.X, pti.Y, pti.Z)
  pts.Add(ptix)

 p0 = PolyCurve.ByPoints(pts,True)
 self.poly = PolyCurve.Fillet(p0,self.rad ,False)
 a1 = Surface.ByPatch(self.poly)
 self.area = a1.Area
```

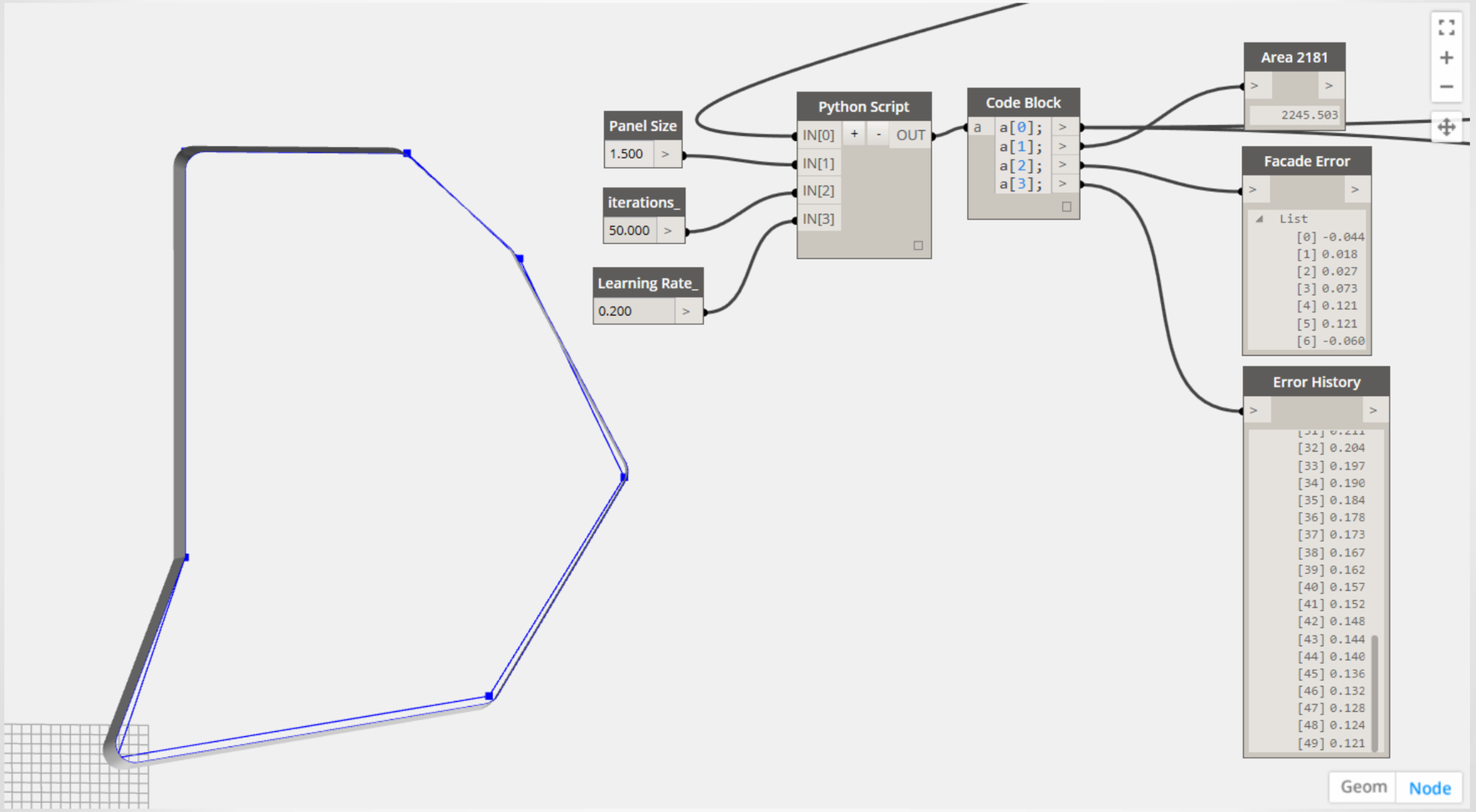
```
118
119 def optimize(self, learningr, iter):
120  #maxIt = 0
121  maxIt = 0.0
122  for i in range (0,self.sp.Count):
123   #update polyline
124   self.update(i)
125
126  for i in range (0,self.sp.Count):
127   #update corners
128   self.sp[i].update(self.panelSize, learningr, iter)
129
130   #maxIt = max(maxIt, math.fabs(self.sp[i].error))
131   maxIt = max(maxIt,math.fabs(self.sp[i].error))
132
133  self.maxError.Add(maxIt)
```

Main script

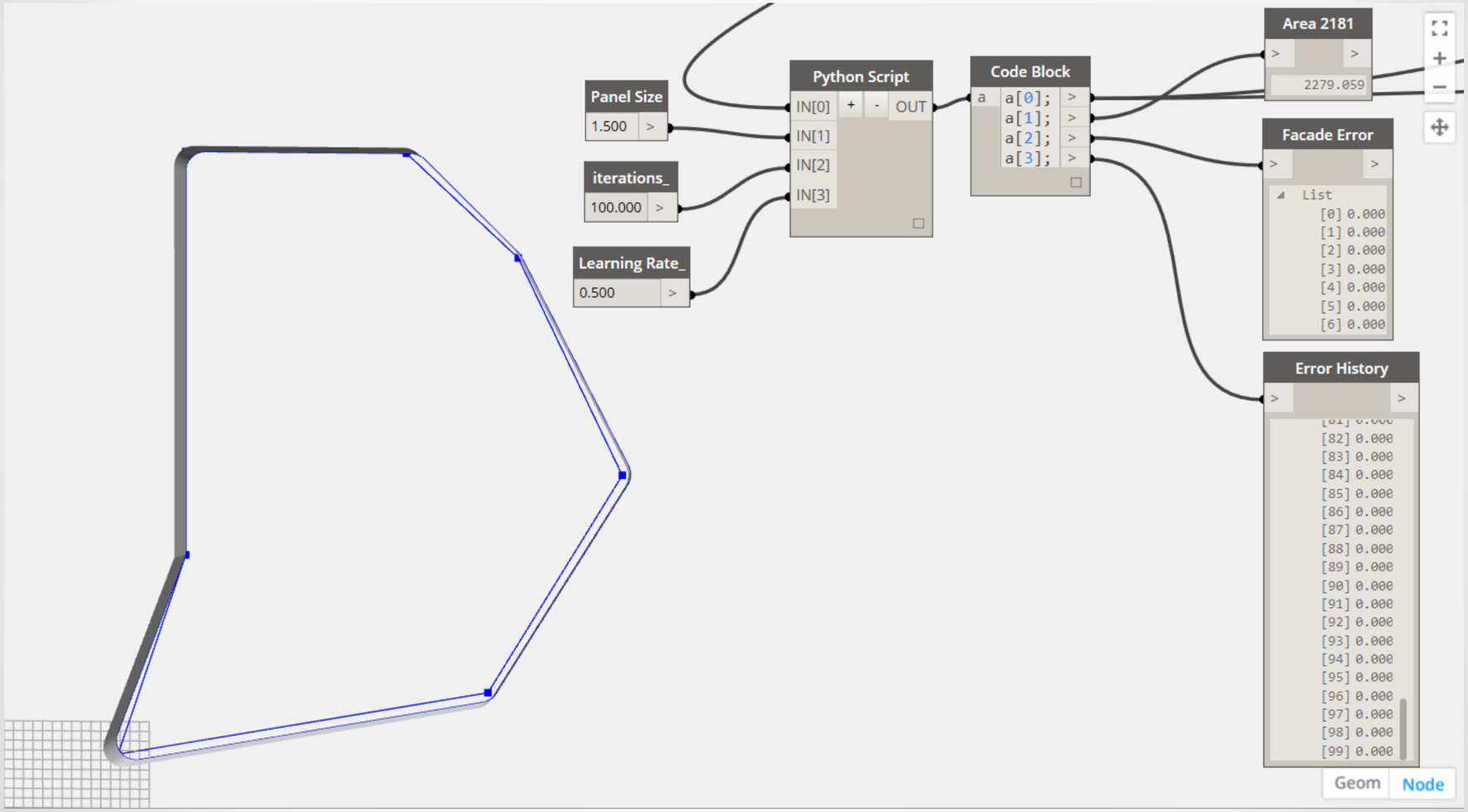
```
dataEnteringNode = IN

points = IN[0]
iterations = IN[2]
learningRate = IN[3]

f = facade(IN[1])
f.addPt(points[0],False,True)
f.addPt(points[1],True,True)
f.addPt(points[2],True,True)
f.addPt(points[3],True,True)
f.addPt(points[4],True,True)
f.addPt(points[5],True,False)
f.addPt(points[6],False,False)
f.generate()
f.optimizeLoop(learningRate, iterations)

#Assign your output to the OUT variable
OUT = f.poly, f.area, f.getError(), f.maxError
```



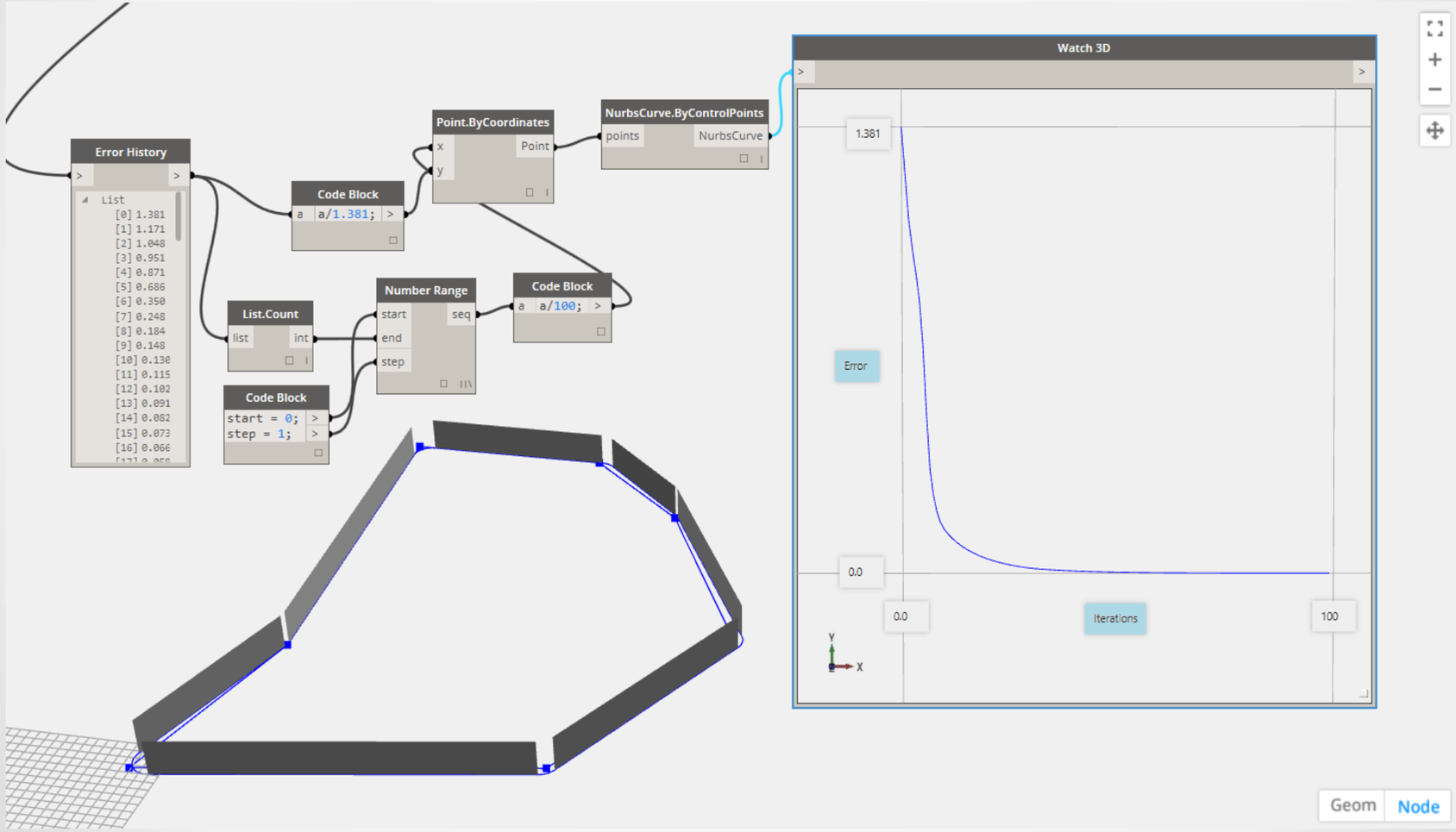
Results 1 – No convergence



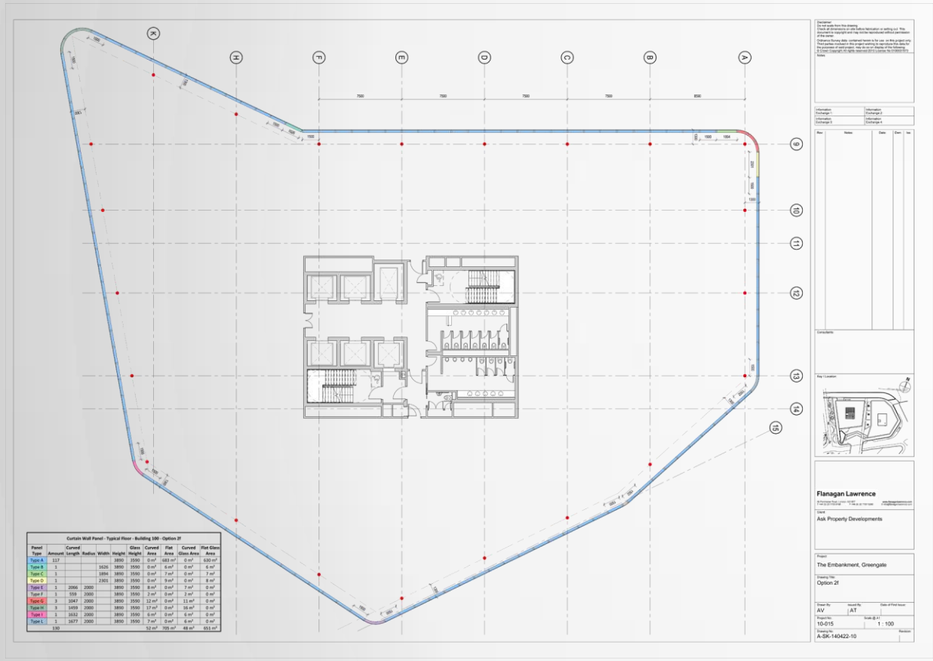
Results 2


Panel Size
1.500
iterations_
100.000
Learning Rate_
0.500
Python Script
IN[0]
IN[1]
IN[2]
IN[3]
OUT
Code Block
a[0];
a[1];
a[2];
a[3];
Area 2181
2279.059
Facade Error
List
[0] 0.000
[1] 0.000
[2] 0.000
[3] 0.000
[4] 0.000
[5] 0.000
[6] 0.000
Error History
[82] 0.000
[83] 0.000
[84] 0.000
[85] 0.000
[86] 0.000
[87] 0.000
[88] 0.000
[89] 0.000
[90] 0.000
[91] 0.000
[92] 0.000
[93] 0.000
[94] 0.000
[95] 0.000
[96] 0.000
[97] 0.000
[98] 0.000
[99] 0.000
Geom
Node

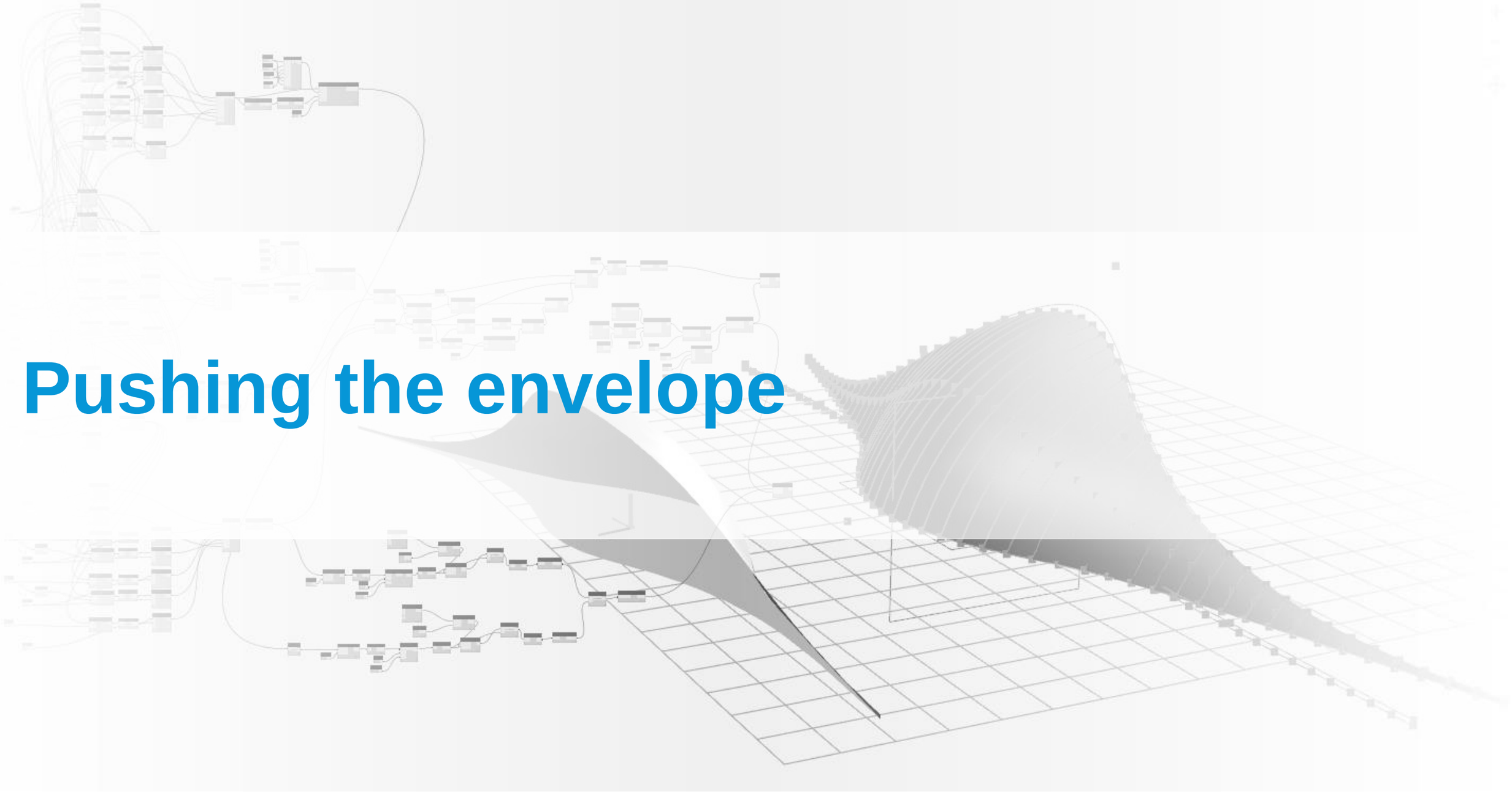
Convergence graphics



Final design

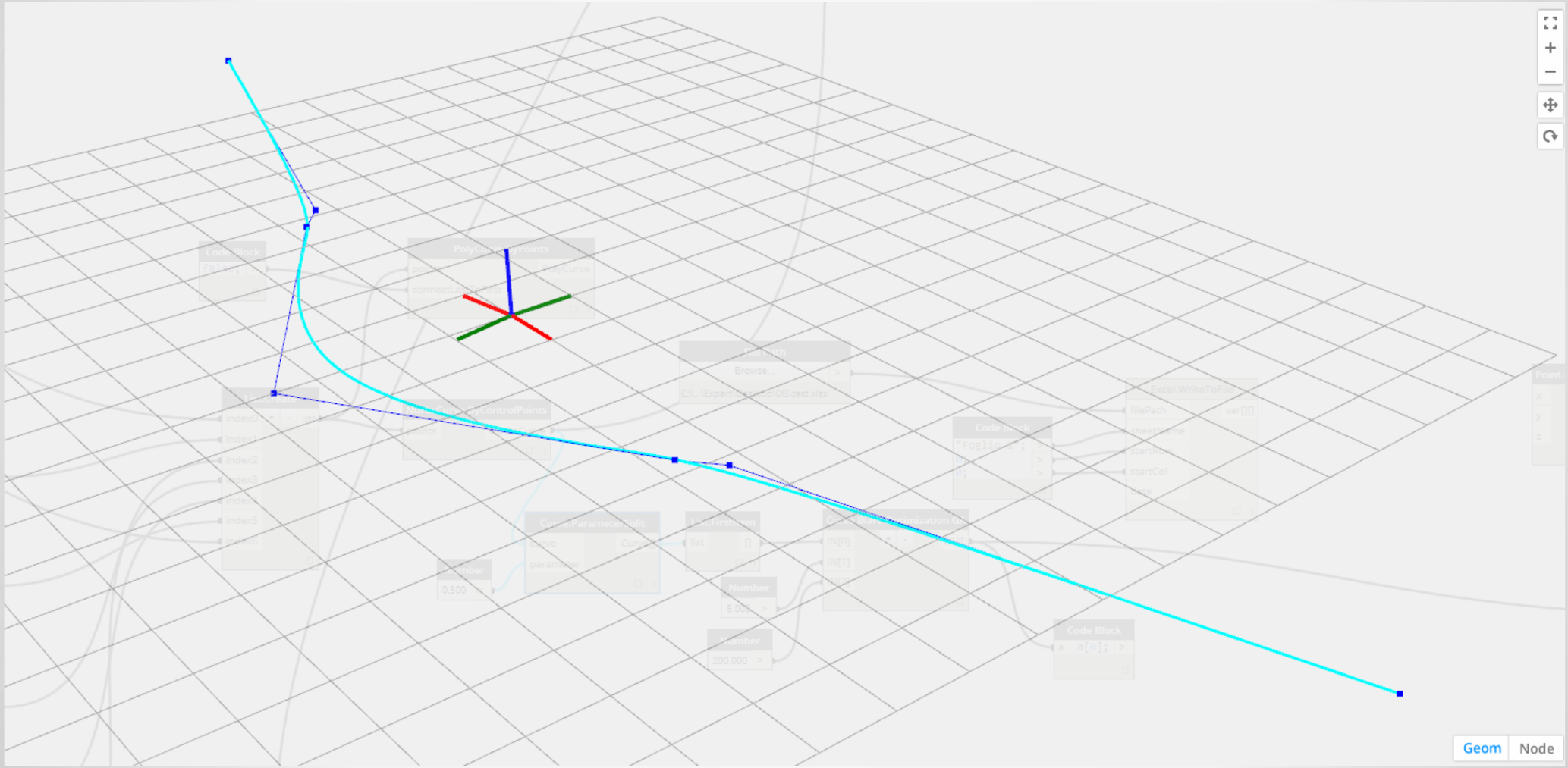


Curtain Wall Panel - Typical Floor - Building 100 - Option 2f										
Panel Type	Amount	Curved Length	Radius	Width	Height	Glass Height	Curved Area	Flat Area	Curved Glass Area	Flat Glass Area
Type A	117				3890	3590	0 m²	683 m²	0 m²	630 m²
Type B	1			1626	3890	3590	0 m²	6 m²	0 m²	6 m²
Type C	1			1894	3890	3590	0 m²	7 m²	0 m²	7 m²
Type D	1			2301	3890	3590	0 m²	9 m²	0 m²	8 m²

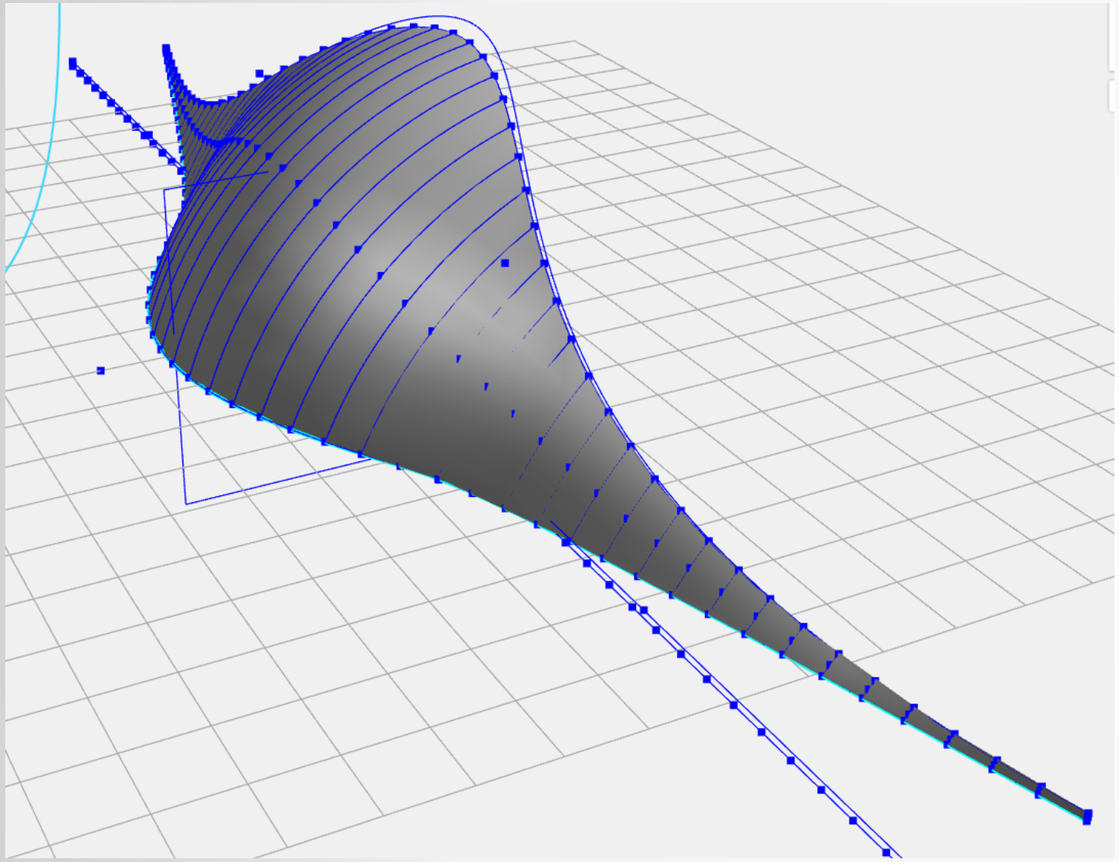


Pushing the envelope

NURBS are not so clever



Genetic algorithm – rationalize NURBS in arcs



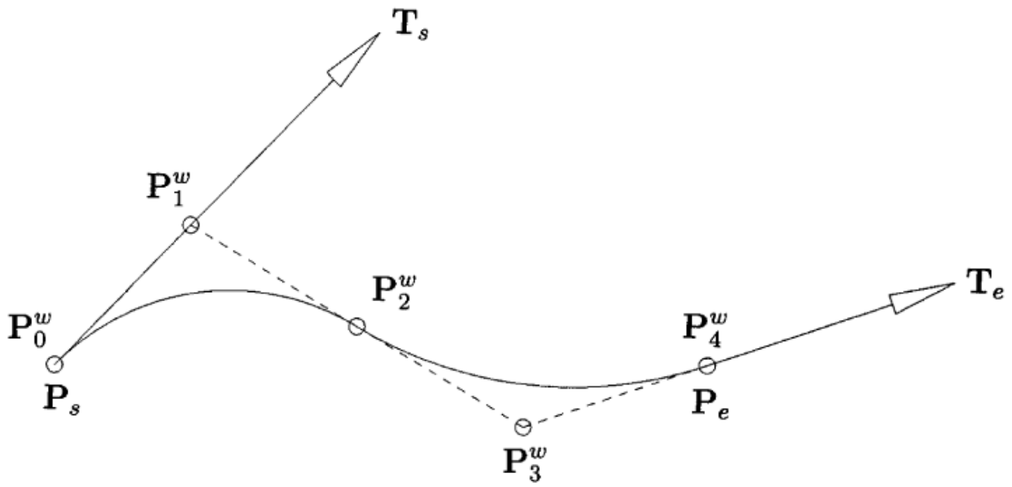
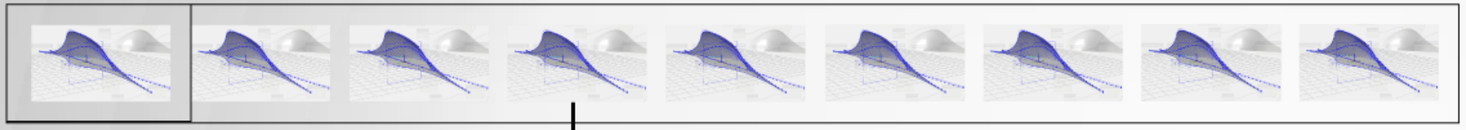


Fig. 1. Biarc formulation.

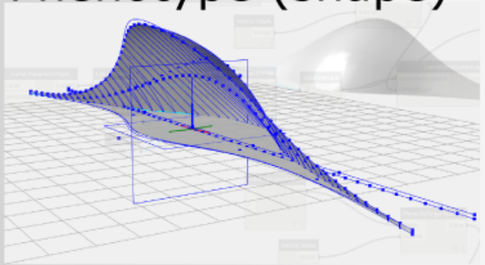
Genetic algorithm

Population



Individual

Phenotype (shape)

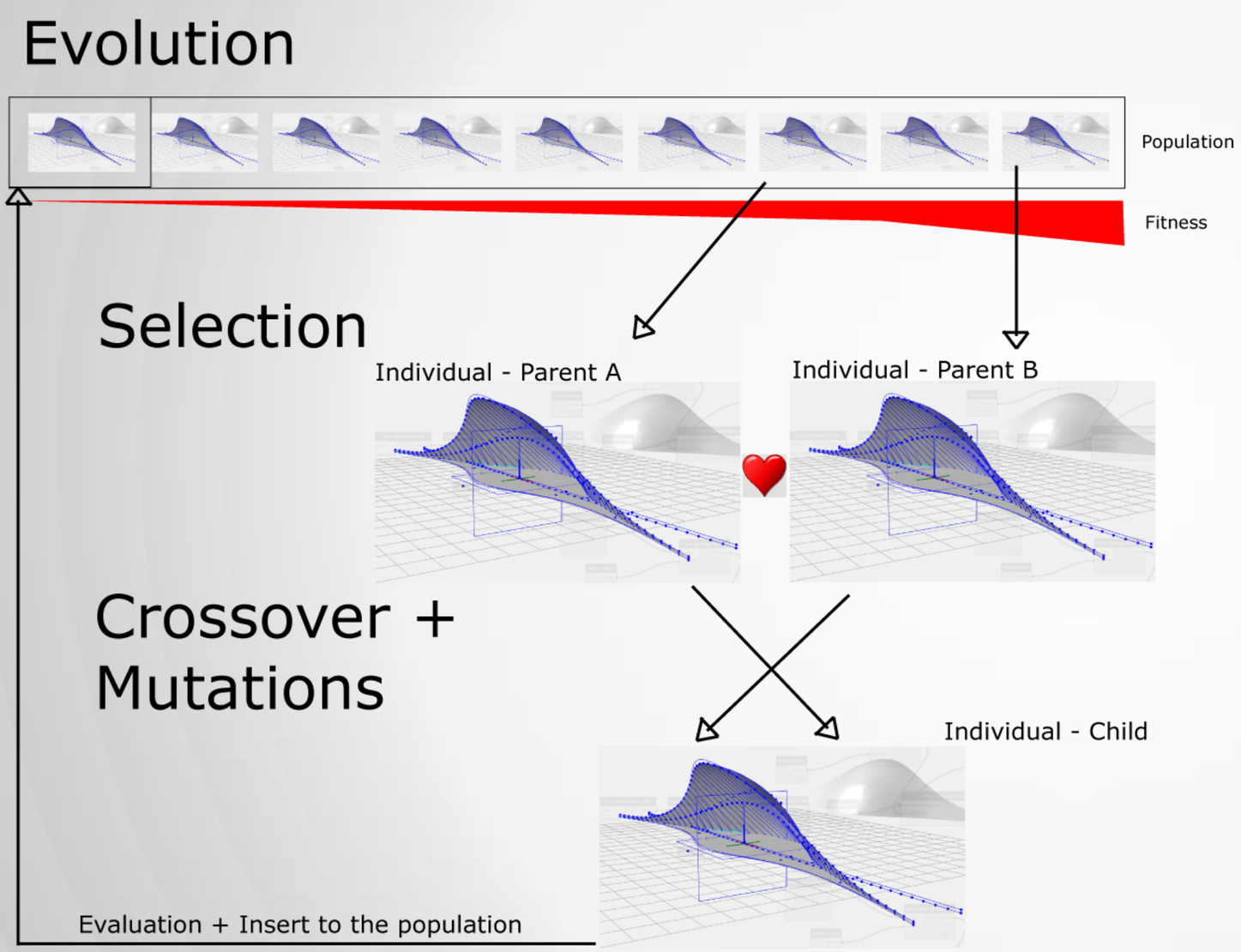


Genotype (parameters)

Fitness (performance)

```
139 class Population:
140  def __init__(self, curve, number):
141   popSize = number*10
142   self.pop = []
143   for i in range(0,popSize):
144    ind = Individual(curve, number)
145    ind.evaluate()
146    self.pop.Add(ind)
147   self.pop.sort(key=lambda idiv: idiv.fitness)
148   self.minFitness = []
149   self.maxFitness = []
150 
151  def fitness(self):
152   fit = []
153   for i in range(0,self.pop.Count):
154    fit.Add(self.pop[i].fitness)
155   return fit
156  def select(self):
157   which = int(math.floor((self.pop.Count-1e-6)*(1.0-math.pow(random.uniform(0,1),2) )))
158 
159   return self.pop[which]
160 
161  def evolve(self):
162   a = self.select()
163   b = self.select()
164   x = a.breed(b)
165   x.evaluate()
166   self.pop[self.pop.Count-1] = x
167   self.pop.sort(key=lambda idiv: idiv.fitness)
168   self.minFitness.Add(self.pop[0].fitness)
169   self.maxFitness.Add(self.pop[self.pop.Count-2].fitness)
170 
171  def getBest(self):
172   return self.pop[0].p.arcs
173 
124 class Individual:
125  def __init__(self, curve, number):
126   self.g = Genotype(number)
127   self.p = Phenotype(curve, self.g)
128   self.fitness = 0.0
129  def evaluate(self):
130   self.fitness = self.p.evaluate()
131 
132  def breed(self, b):
133   c = Individual(self.p.c, self.g.n)
134   c.g = self.g.crossover(b.g)
135   c.g.mutate()
136   c.p = Phenotype(c.p.c, c.g)
137   return c
```

Genetic algorithm




```
#Inputs
INcurve = IN[0]
INnumber = IN[1]
INiter = IN[2]

#script
#Generate the population of solution
popul = Population(INcurve, INnumber)
#evolve the population for the number of iterations
for i in range(0,INiter):
 popul.evolve()

OUT = popul.getBest(), popul.minFitness, popul.maxFitness
```

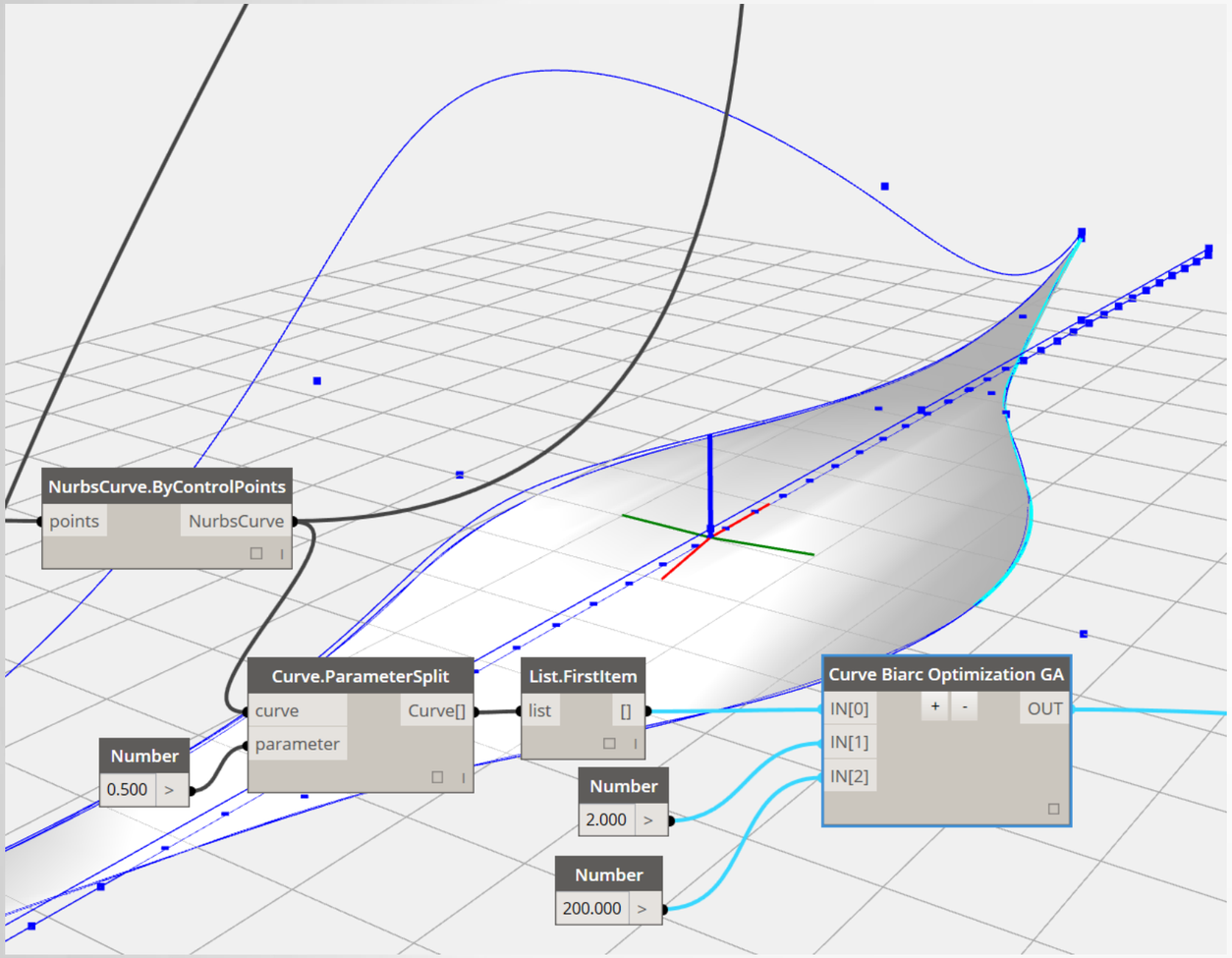
```
def crossover(self, b):
 c = Genotype(self.n)
 for i in range(0,self.g1.Count):
  if (random.uniform(0,1) < 0.5): #crossover 50% probability
   c.g1[i] = self.g1[i] #gene 1 can switch the values
   if i > 0:
    c.g2[i] = self.g2[i]
  else :
   c.g1[i] = b.g1[i]
   if i > 0:
    c.g2[i] = b.g2[i]
  c.updateG2()
 return c

def mutate(self):
 for i in range(0,self.g1.Count):
  if (random.uniform(0,1) < 0.1): #mutation 10% probability
   x = random.uniform(0.2, 0.9)
   self.g1[i] += 0.01*math.exp(x*7)
   #gene 2
  if i > 0:
   if (random.uniform(0,1) < 0.1): #mutation 10% probability
    y = self.g2[i]
    delta = 1.0/self.n/10
    y += random.uniform(-delta,delta)
    val = self.g2[i] + y
    if val > 0.0 and val < 1.0:
     self.g2[i] = val
 self.updateG2()
```

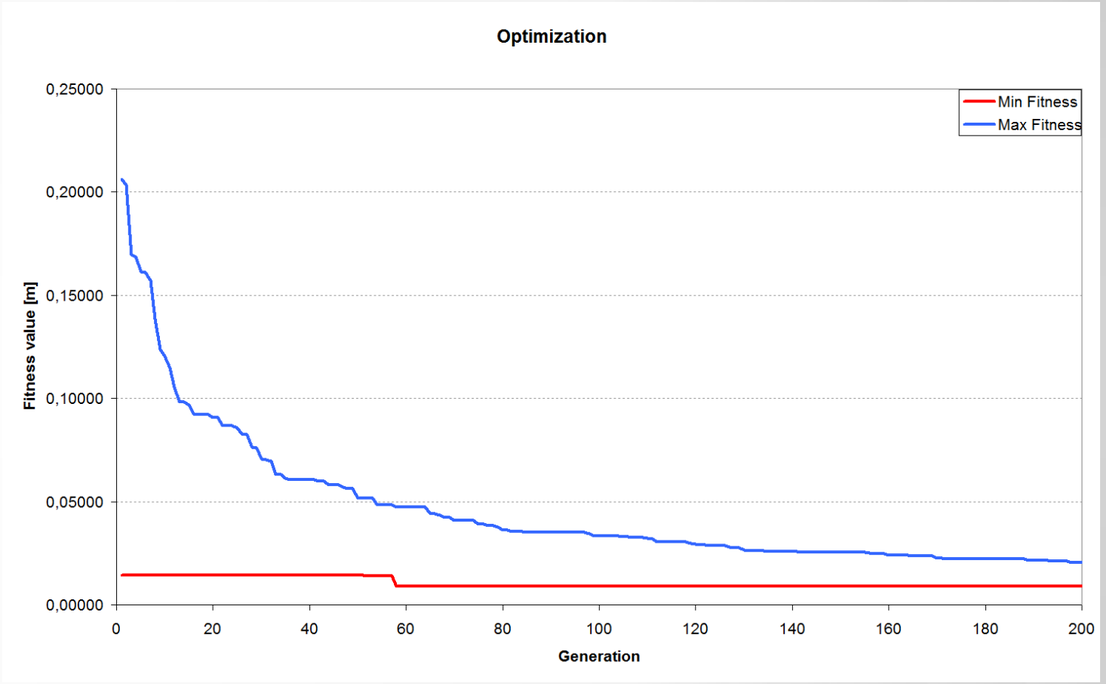
Python Code

```
#Genotype
class Genotype:
 def __init__(self, n):
  self.n = n #number of biarcs
  self.g1 = [] #genotype r value
  self.g2 = [] #genotype t value
  delta = 1.0 / self.n
  self.g2.Add(0.0)
  self.var = 1.0/self.n/10.0
  for i in range(0,self.n):
   x = random.uniform(0.2, 0.9)
   self.g1.Add(0.1*math.exp(x*7))
   if i > 0:
    tr = random.uniform(0.0+self.var,1.0-self.var)
    self.g2.Add(tr)
  self.g2.Add(1.0)
  self.updateG2()
  #self.g2.sort()

 def updateG2(self):
  self.g2.sort()
  for i in range(1,self.n-1):
   delta = math.fabs(self.g2[i] - self.g2[i-1])
   tol = self.var / 10.0
   if delta < tol:
    self.g2[i] = self.g2[i] + tol

 def crossover(self, b):
  c = Genotype(self.n)
  for i in range(0,self.g1.Count):
   if (random.uniform(0,1) < 0.5): #crossover 50% probability
    c.g1[i] = self.g1[i] #gene 1 can switch the values
    if i > 0:
     c.g2[i] = self.g2[i]
   else :
    c.g1[i] = b.g1[i]
    if i > 0:
     c.g2[i] = b.g2[i]
  c.updateG2()
  return c
```

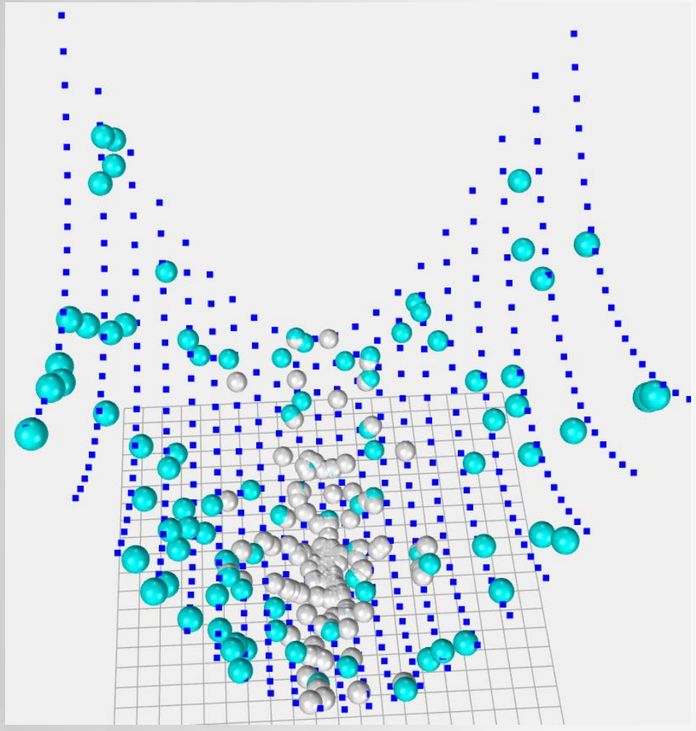
```
#Phenotype
class Phenotype:
 def __init__(self, curve, g):
  self.c = curve #curve to optimize
  self.genes = g
  self.arcs = self.generateBiarcs(self.c, self.genes.g2, self.genes.g1)

 def generateBiarcs(self, curve, t, r):
  #generate the biarcs
  bi = []
  for i in range(0,self.genes.n):
   p1 = curve.PointAtParameter(t[i])
   t1 = curve.TangentAtParameter(t[i])
   p4 = curve.PointAtParameter(t[i+1])
   t4 = curve.TangentAtParameter(t[i+1])
   b1 = BiArc(p1,t1,p4,t4,r[i])
   bi.Add(b1)
  #generate the arc list
  arcs = []
  for i in range(0,self.genes.n):
   arci = []
   arci = bi[i].getArcs()
   for j in range(len(arci)):
    arcs.Add(arci[j])
  return arcs

 # calculate the fitness
 def evaluate(self):
  polyArc = PolyCurve.ByJoinedCurves(self.arcs)
  fitness = self.deviation(self.c,polyArc)
  return fitness

 #maximum distance between 2 curves
 def deviation(self, c1, c2):
  num = self.genes.n*10
  step = 1.0 / (num)
  maxError = 0.0

  for i in range(0,num+1):
   si = step * i
   p1 = c1.PointAtParameter(si)
   p2 = c2.ClosestPointTo(p1)
   li = p1.DistanceTo(p2)
   maxError = max(maxError, li)
  return maxError
```

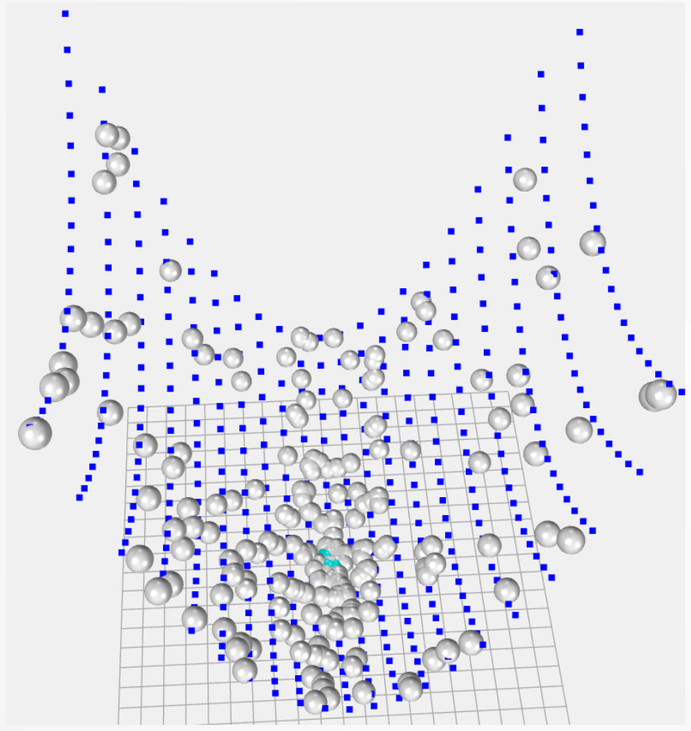
Results

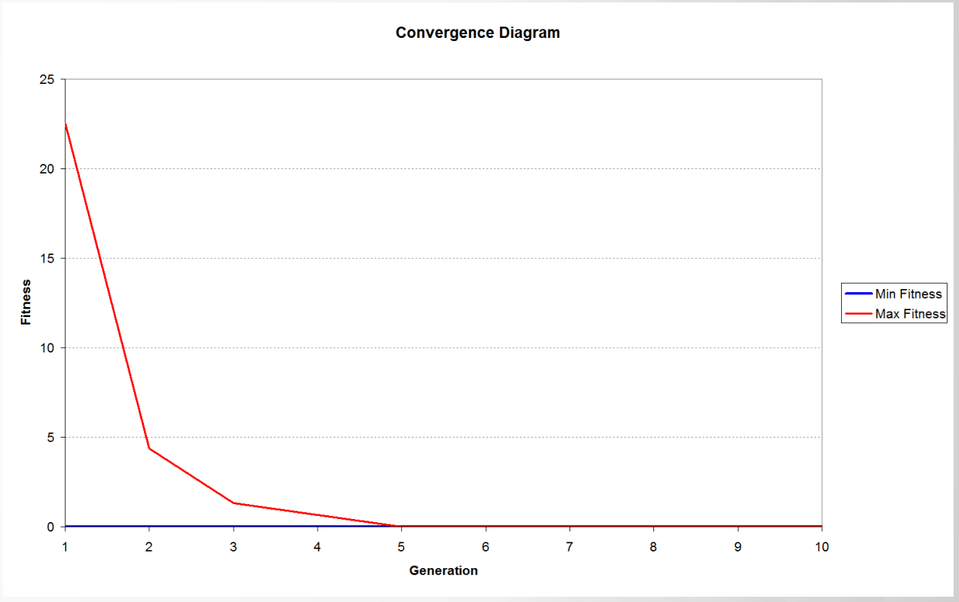

NurbsCurve.ByControlPoints
points
NurbsCurve
Curve.ParameterSplit
curve
parameter
Curve[]
List.FirstItem
list
[]
Number
0.500
Number
2.000
Number
200.000
Curve Biarc Optimization GA
IN[0]
IN[1]
IN[2]
+
-
OUT


Optimization
Min Fitness
Max Fitness
Fitness value [m]
0,25000
0,20000
0,15000
0,10000
0,05000
0,00000
0
20
40
60
80
100
120
140
160
180
200
Generation

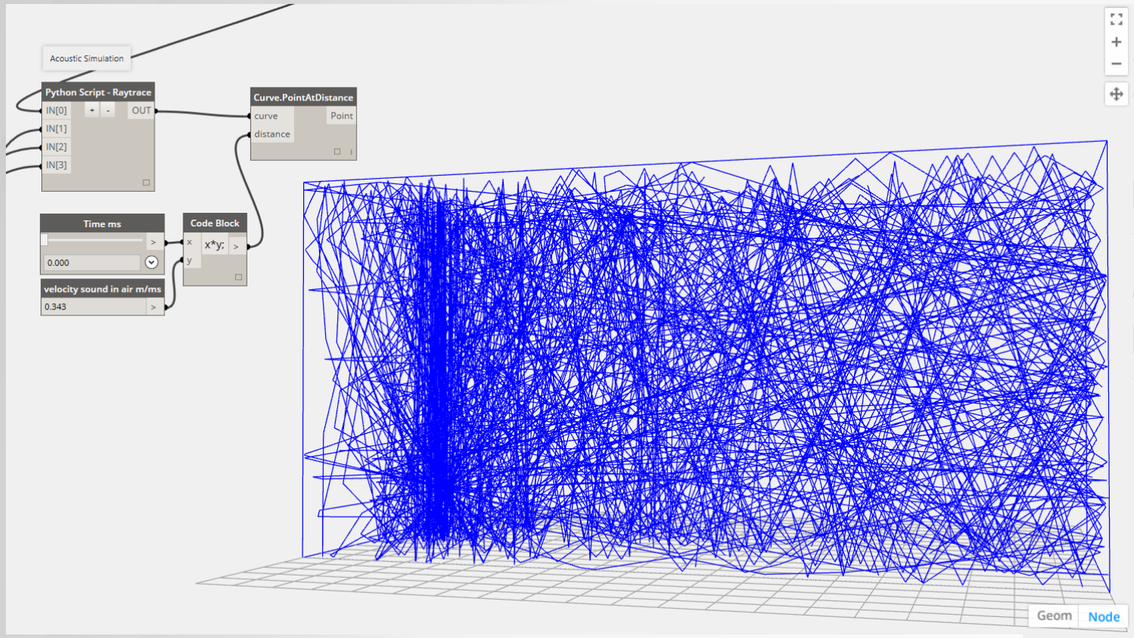
Optimo by Mohammad Rahmani Asl

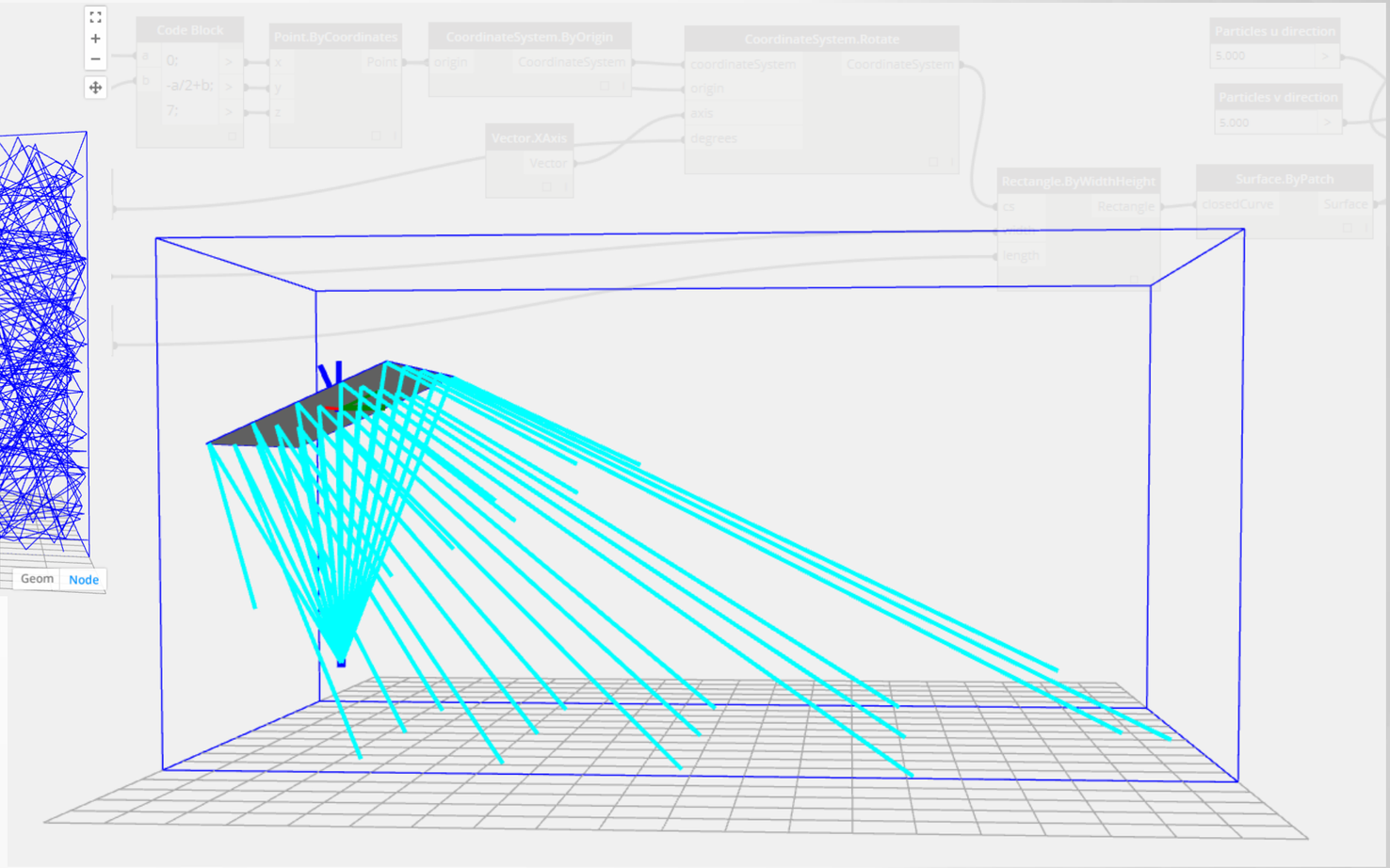






Simulation - Acoustics





http://vimeo.com/113553984

Conclusion

Conclusion

- Flanagan Lawrence used Dynamo and Pythonscript to create award-winning architecture
- Developing the question is just as important as finding the answer
- Dynamo excellent introduction to other scripting languages
- Genuine BIM software with Revit
- Pythonscript is very stable and opens up many possibilities for design optimisation

Session Feedback

- Via the Survey Stations, email or mobile device
- AU 2014 passes given out each day!
- Best to do it right after the session
- Instructors see results in real-time

***Students, educators,* and *schools* now have**

FREE access to Autodesk design software & apps.

Download at www.autodesk.com/education

Earn your professional Autodesk Certification at AU

Visit the AU Certification Lab