How to Train Your Model with Project Refinery

Matt Jezyk

Tesla (formerly Autodesk)

VIDEO RECORDING IS PERMITTED

DIGITAL BUILT ENVIRONMENT INSTITUTE

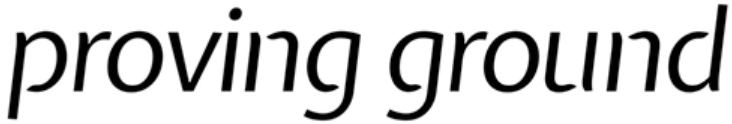
BUILDING CONTENT SUMMIT by DBEI

BiLT EUR BY DBEI

Matt Jezyk
Senior Staff Software Engineer, Tesla
Former Senior Engineering Manager, AEC Generative Design, Autodesk

DIGITAL BUILT WEEK EUROPE
EDINBURGH
10 – 12 OCT 2019

A SPECIAL THANKS TO:



Session Description

Creating parametric options is easier than ever, but creating 'good' options is still quite hard. Creating good design options starts by asking the right questions, and then by embedding your custom design logic into code. We will teach you how to code 'Evaluators' and train your model using Project Refinery.

DBEI DIGITAL BUILT ENVIRONMENT INSTITUTE

Learning Objectives

At the end of this session, participants will be able to:

1. Frame a design problem in terms of goals and constraints
2. Learn how to write psuedocode to help define the logic before writing final code
3. Develop new custom code in multiple programming languages (Dynamo, DesignScript, Python, and C#)
4. Test the custom logic in a larger generative workflow

BUILDING CONTENT SUMMIT by DBEI

BiLT EUR BY DBEI

HOW TO USE GENERATIVE DESIGN TOOLS TO ADVANCE ARCHITECTURAL DESIGN PROCESSES.

Overview

- WHY?
- HOW: How to define goals and metrics
- HOW: Evaluator Examples
  - Hill Climbing – What's an Objective Function?
  - Massing Study – Real-World Goals Made Into Evaluators
  - Solar Analysis – Using Simulation Results
  - Desk Layout – More Evaluators
- Questions

Example files are all Open Source!

- The examples for this class:
  - https://github.com/tatlin/BiLT_NA_2019_How_To_Train_Your_Model
- Refinery Primer:
  - https://refineryprimer.dynamobim.org/
  - https://github.com/DynamoDS/RefineryPrimer
- Refinery Toolkit Dynamo Package and Examples Files:
  - https://github.com/DynamoDS/RefineryToolkits
- Thank You To Our Partners:
  - DesignTech/ENSTOA, ProvingGround, and Autodesk Research

About

Welcome to the Refinery Primer, which aims to introduce AEC practitioners of all experience levels to an exciting new approach to design using generative design workflows. This primer will help you:

- **Understand** what generative design is by defining the base concepts and terminology you need to know
- **Explore** how these techniques can be used to solve practical challenges commonly found in AEC design projects
- **Learn** how to use Autodesk's newest generative design tools such as Project Refinery, through practical workflows

Refinery Toolkits

A collection of packages to accelerate generative design workflows in Dynamo & Refinery.

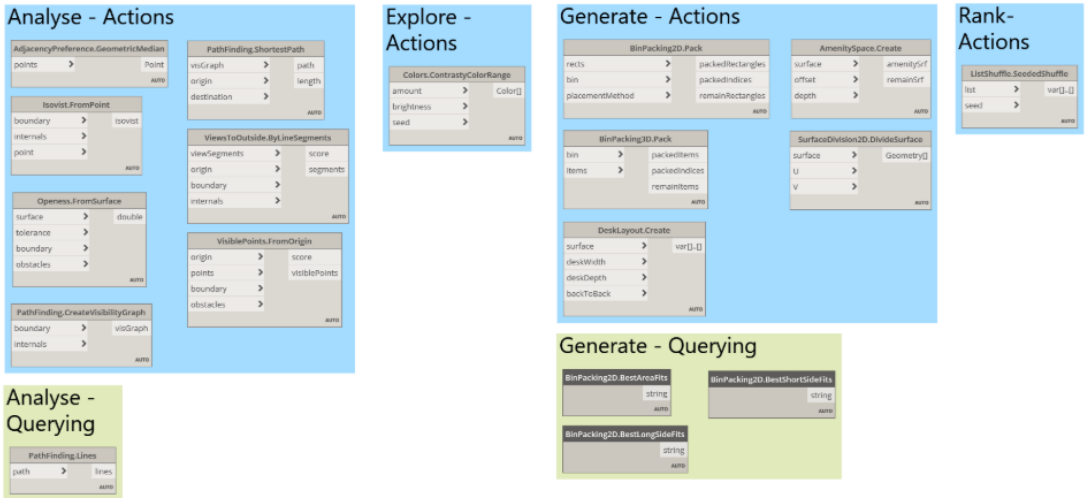
The toolkits

There are currently 2 packages included in the toolkit, each focusing on enabling specific types of workflows:

- SpacePlanning Toolkit
- Massing Toolkit

Space Planning Toolkit

The toolkit offers a range of nodes that help with general space-planning workflows in Dynamo and Revit.



Why?

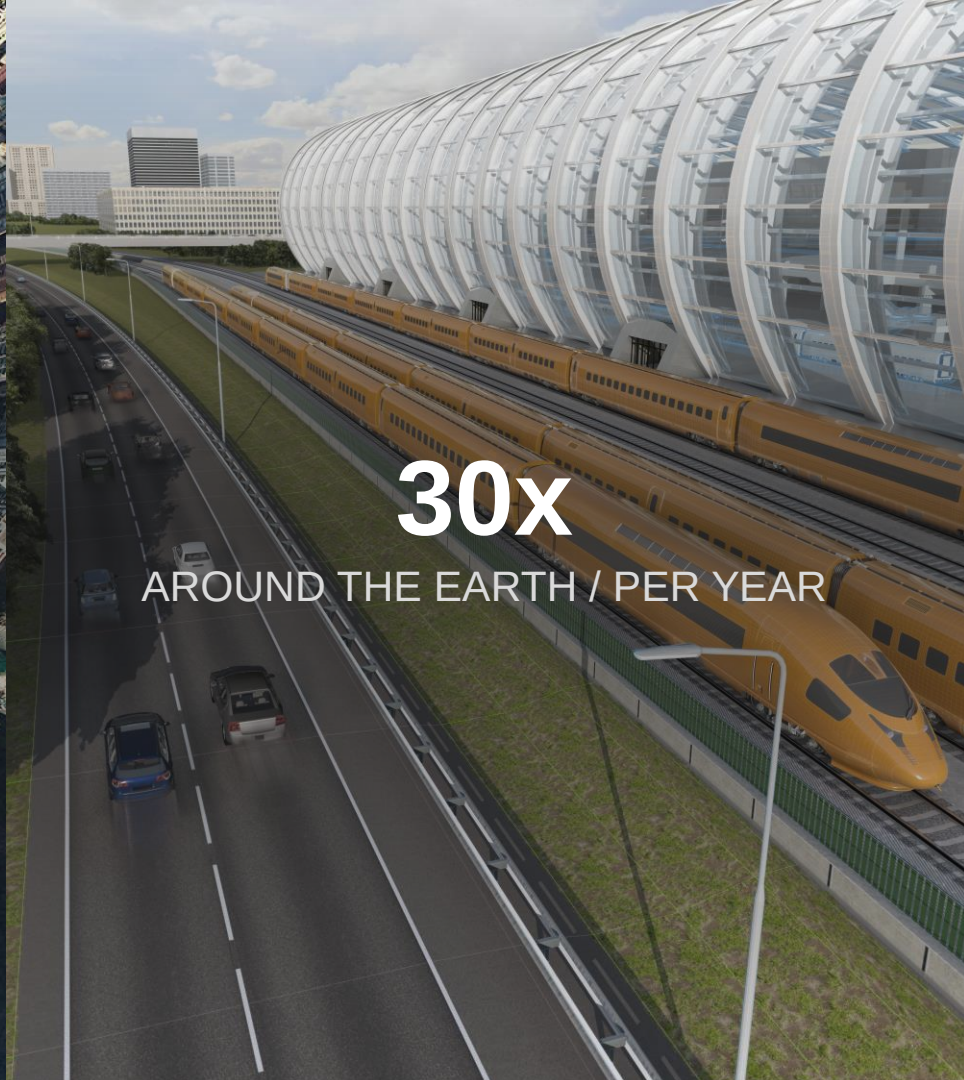
DBEI DIGITAL BUILT ENVIRONMENT INSTITUTE

BUILDING CONTENT SUMMIT by DBEI

BiLT EUR BY DBEI


10 BILLION
PEOPLE BY 2050


13,000
BUILDINGS A DAY

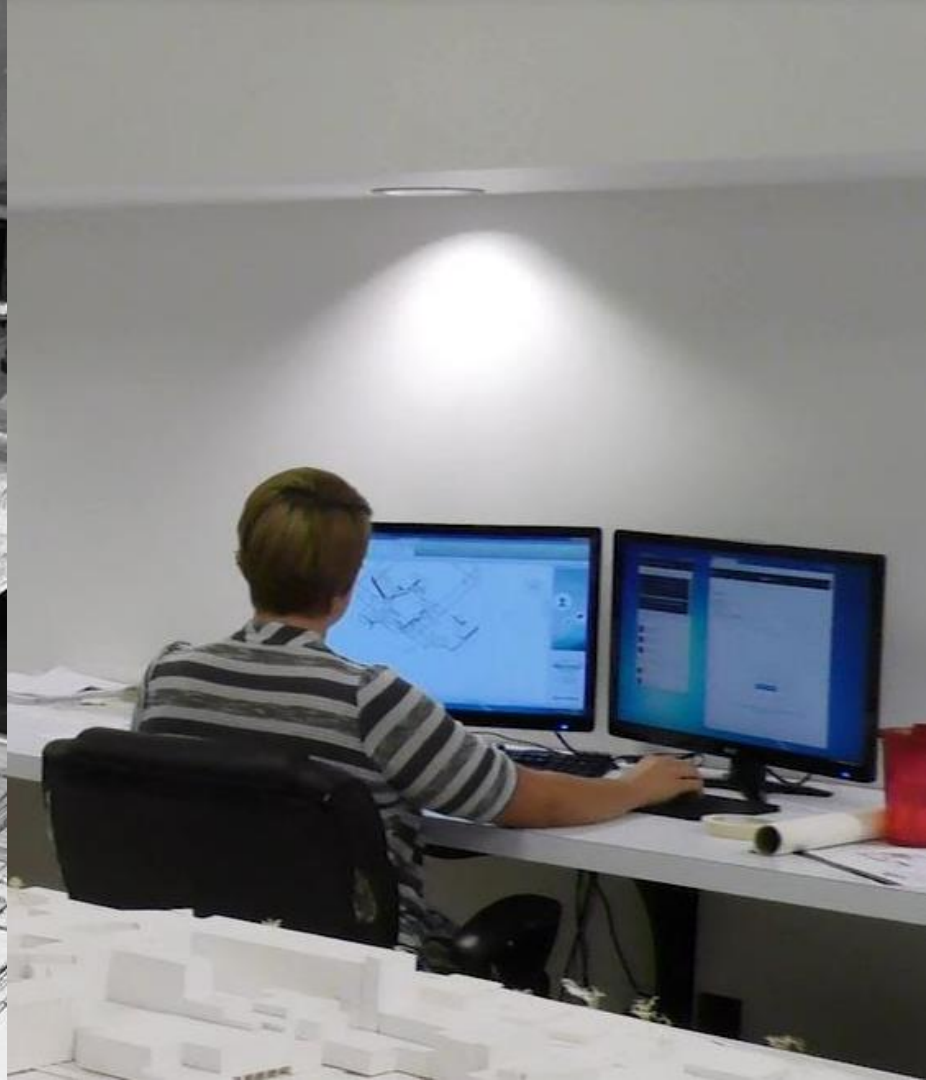

30x
AROUND THE EARTH / PER YEAR

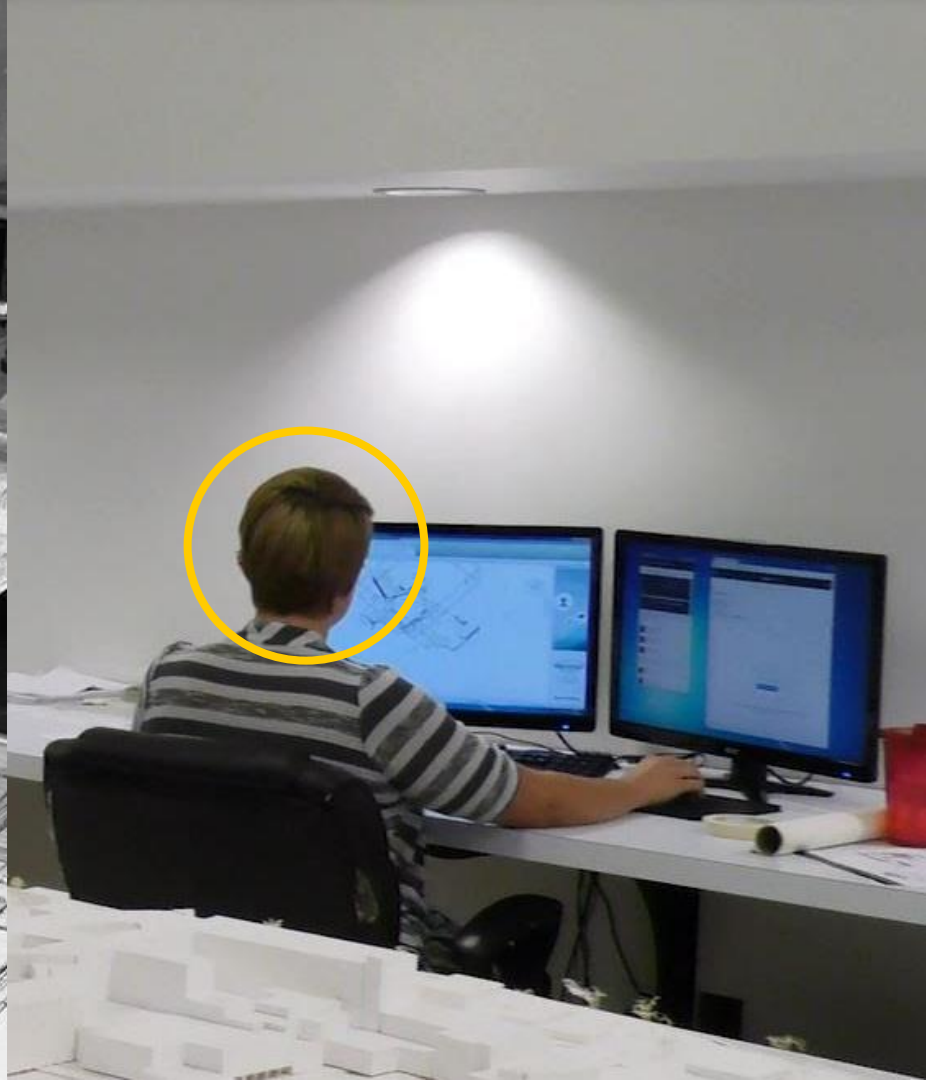
Problem:
more work
at a higher quality
with less resources

DIGITAL BUILT ENVIRONMENT INSTITUTE

BUILDING CONTENT SUMMIT by DBEI

BILT EUR BY DBEI





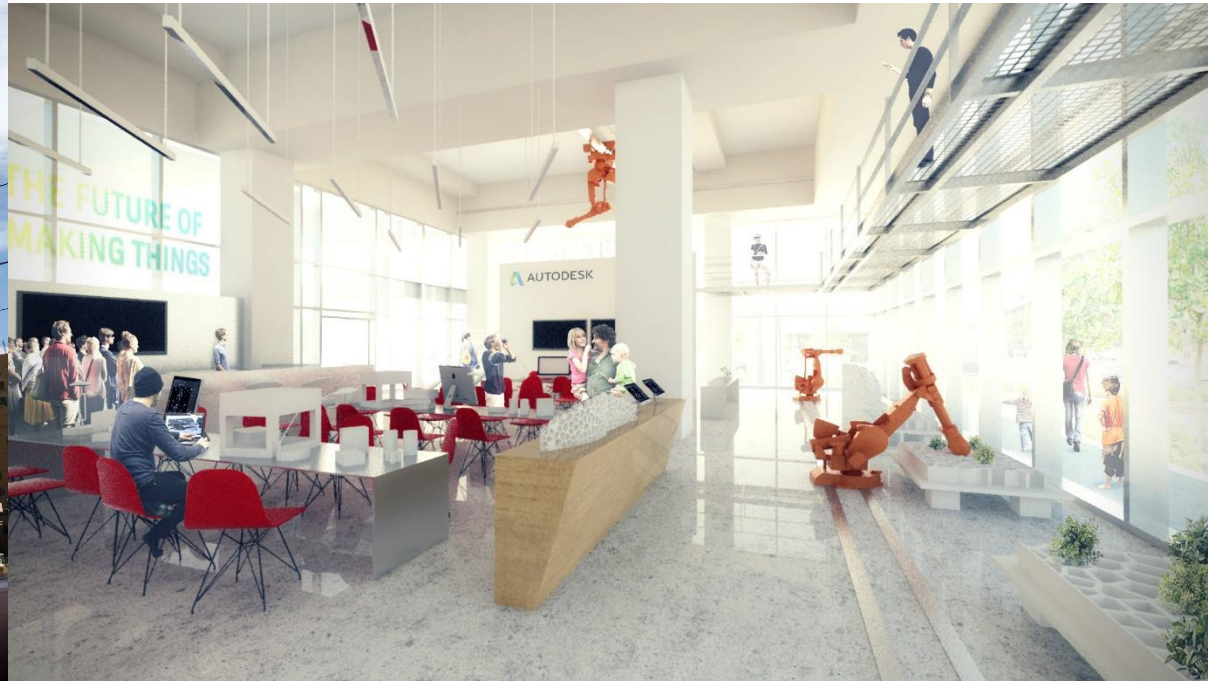

Professional
Intelligence
Algorithmic
Opportunity

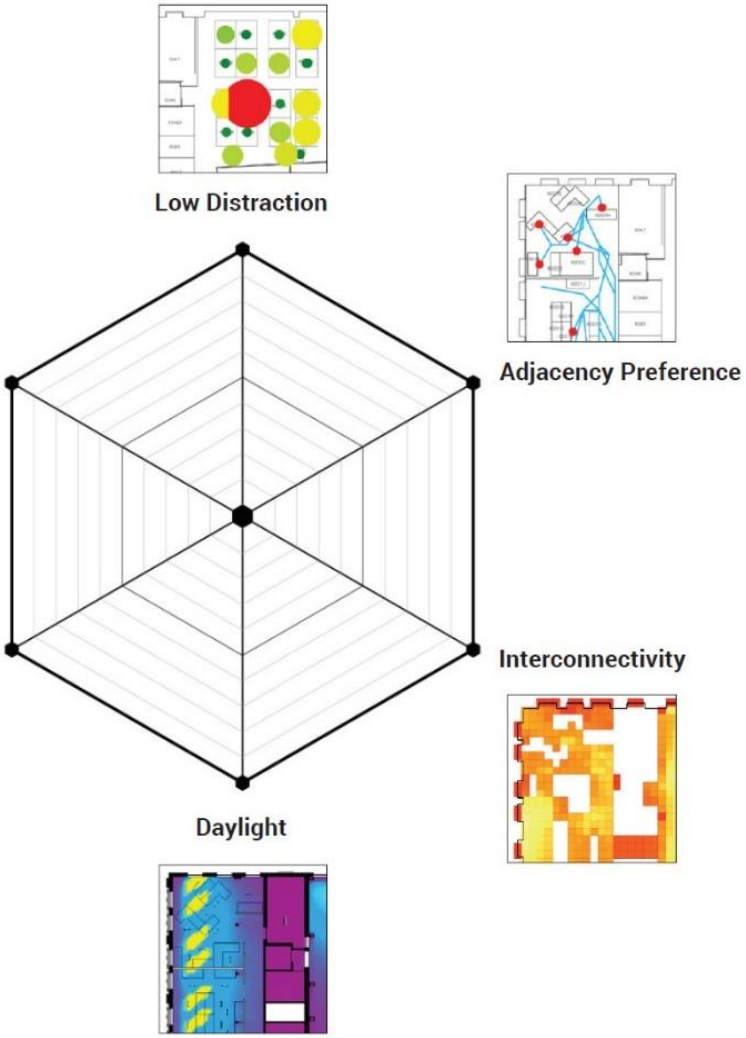
MaRS

THE LIVING

AUTODESK.

RESEARCH





DBEI DIGITAL BUILT ENVIRONMENT INSTITUTE

LOW VISUAL DISTRACTION

Measurement of negative visual activity from workspaces.

SCORING EQUATION

$$\text{DISTRACTION} = \frac{\sum [\text{ Normalized Count of Visible Coworkers per Workspace }]}{\text{Number of Workspaces}} \times 10.0$$

INPUTS

+ 2D Space Model (Geometry, including Obstructions)
+ Workspace locations and orientation

COMPUTATION METHODS

+ Field of view (200° horizontal)
+ Upper distraction limit: 15 coworkers visible
+ Isovist polygon generation per workspace
+ Point inclusion in isovist

OUTPUTS

+ Individual desk scoring (from 0 to 10)
+ Per floor aggregated scores (from 0 to 10)
+ Global distraction score (from 0 to 10)

SCORE RANGE

+ 0.0 SCORE (WORST CASE) : All employees at or above upper distraction limit
+ 10.0 SCORE (BEST CASE) : All employees have zero visible coworkers

6TH FLOOR

5TH FLOOR

4TH FLOOR

DBEI DIGITAL BUILT ENVIRONMENT INSTITUTE

ADJACENCY PREFERENCE

Measurement of travel distance to preferred neighbors and amenities.

SCORING EQUATION

$$\text{ADJACENCY} = \frac{\sum [\text{Shortest Path Length} \times (1 + \Delta \text{Floors} \times \text{Vertical Multiplier})]}{\text{Number of Shortest Paths}} \times 10.0$$

INPUTS

+ Visibility graph (curve-based graph of possible travel)
+ Individual neighbor adjacency preferences (JSON-formatted survey data)
+ Individual amenity adjacency preferences (JSON-formatted survey data)

COMPUTATION METHODS

+ Geometric shortest path algorithm (curve-based)
+ Horizontal travel distance limit: 100 ft
+ Vertical travel distance limit: 11.5ft (210 King floor-to-floor height)

OUTPUTS

+ Individual adjacency scores (from 0 to 10)
+ Per floor aggregated adjacency scores (from 0 to 10)
+ Global adjacency score (from 0 to 10)

SCORE RANGE

+ 0.0 SCORE (WORST CASE) : All individuals have highest cost travel (max horiz. and vert. distance)
+ 10.0 SCORE (BEST CASE) : All individuals have lowest cost travel (adjacent and same floor)

6TH FLOOR

5TH FLOOR

4TH FLOOR

DBEI DIGITAL BUILT ENVIRONMENT INSTITUTE

CIRCULATION

Measurement of congestion. Determined by cross-referencing simulated movement paths with computed traversability data for given space.

SCORING EQUATION

$$\text{CIRCULATION} = \frac{\sum(\text{Traversed Grid Values})}{\text{Traversed Grid Count} \times \text{Max Value}} \times 10.0$$

INPUTS

+ Adjacency Preference Shortest Paths (Geometry - from Adjacency Metric)
+ 2D Space Model (geometry, including solid obstructions)

COMPUTATION METHODS

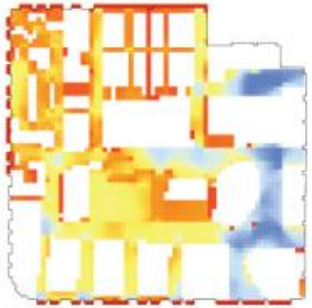
+ Generate analysis grid of relative traversability (range of possible moves from given point)
+ Identify and sum values of intersected grid tiles for every shortest path

OUTPUTS

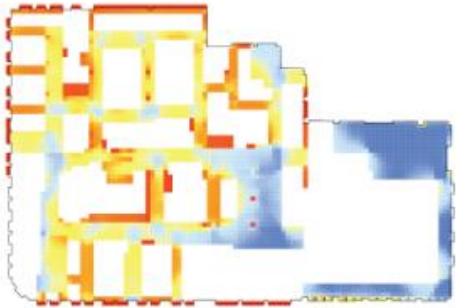
+ Per floor circulation scores (from 0 to 10)
+ Global circulation scoring (from 0 to 10)

SCORE RANGE

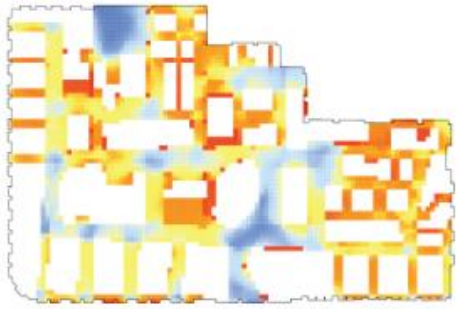
+ 0.0 SCORE (WORST CASE) : All movement through high congestion areas
+ 10.0 SCORE (BEST CASE) : All movement through congestion-free areas



6TH FLOOR

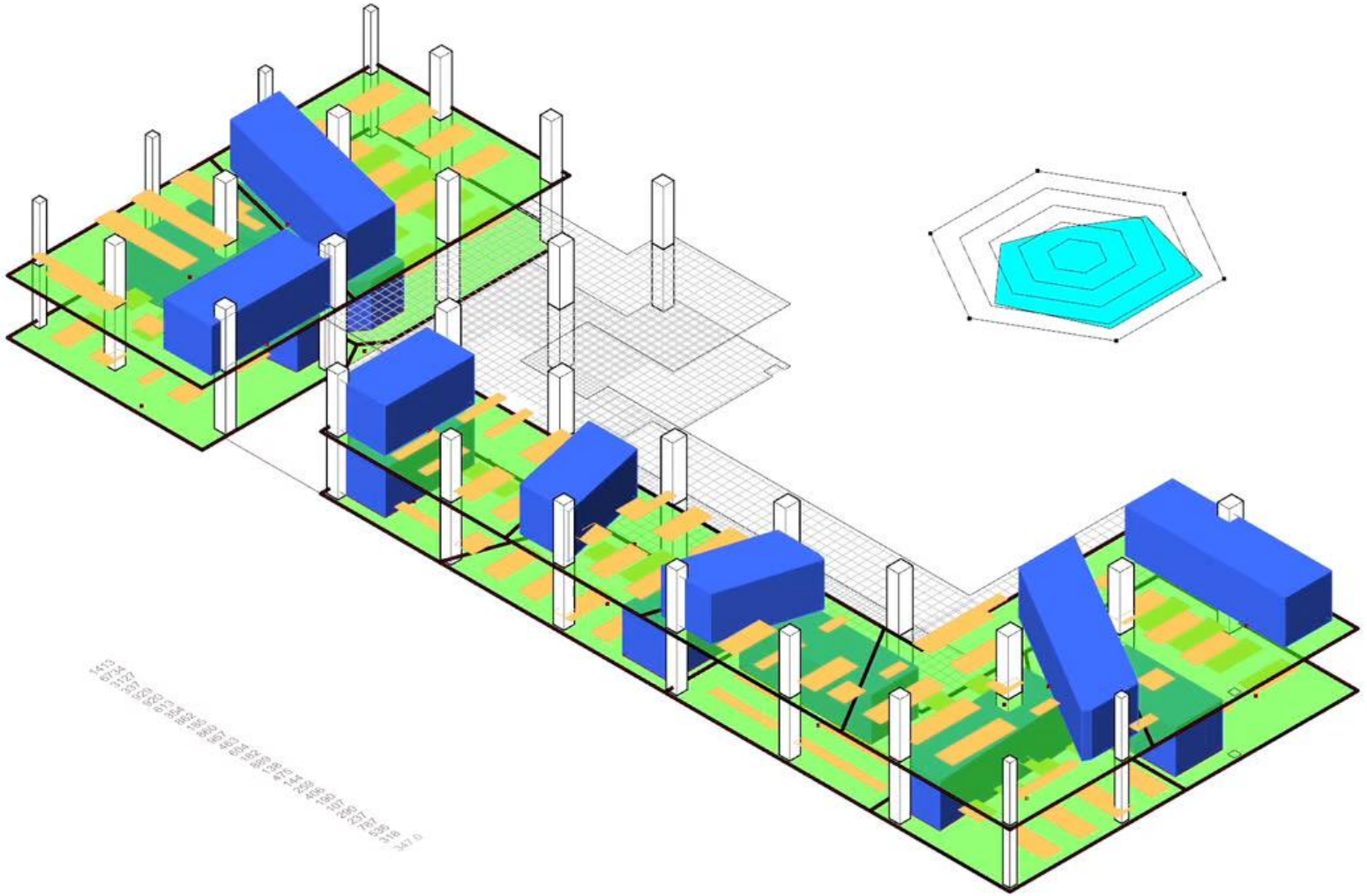


5TH FLOOR

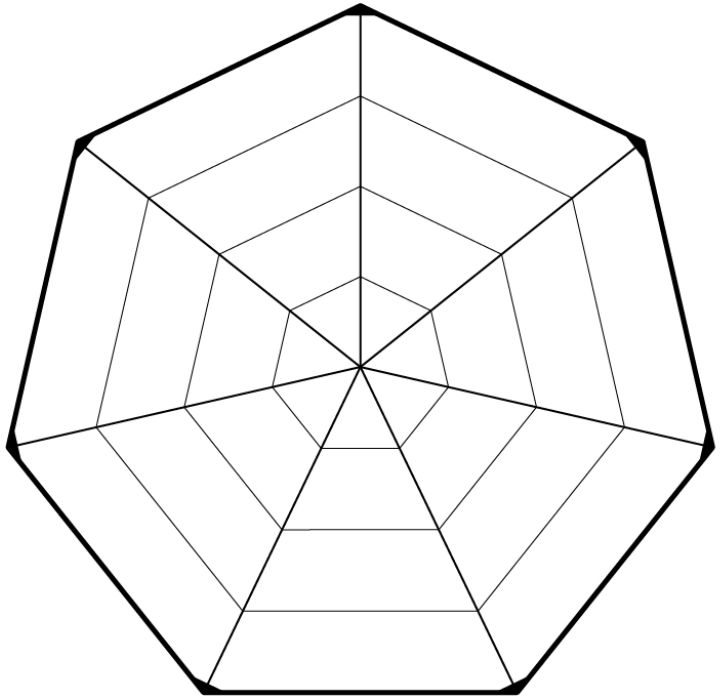


4TH FLOOR

DBEI DIGITAL BUILT ENVIRONMENT INSTITUTE



Generative design is the process of **defining high-level goals and constraints** and then using the **power of computation** to automatically explore a wide design space and identify the best design options.



“We want to be able to learn from more designs than it is physically possible to generate or evaluate ‘by hand’.”

– Danil Nagy, The Living

DIGITAL BUILT ENVIRONMENT INSTITUTE

BUILDING CONTENT SUMMIT by DBEI

BILT EUR BY DBEI

How!?

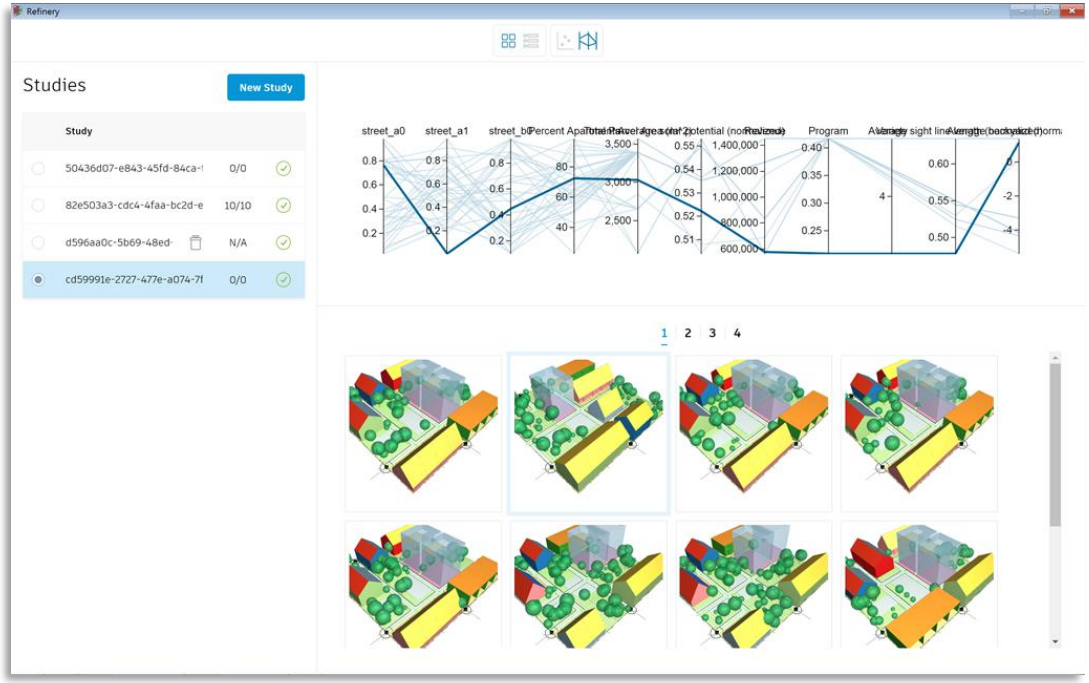
DIGITAL BUILT ENVIRONMENT INSTITUTE

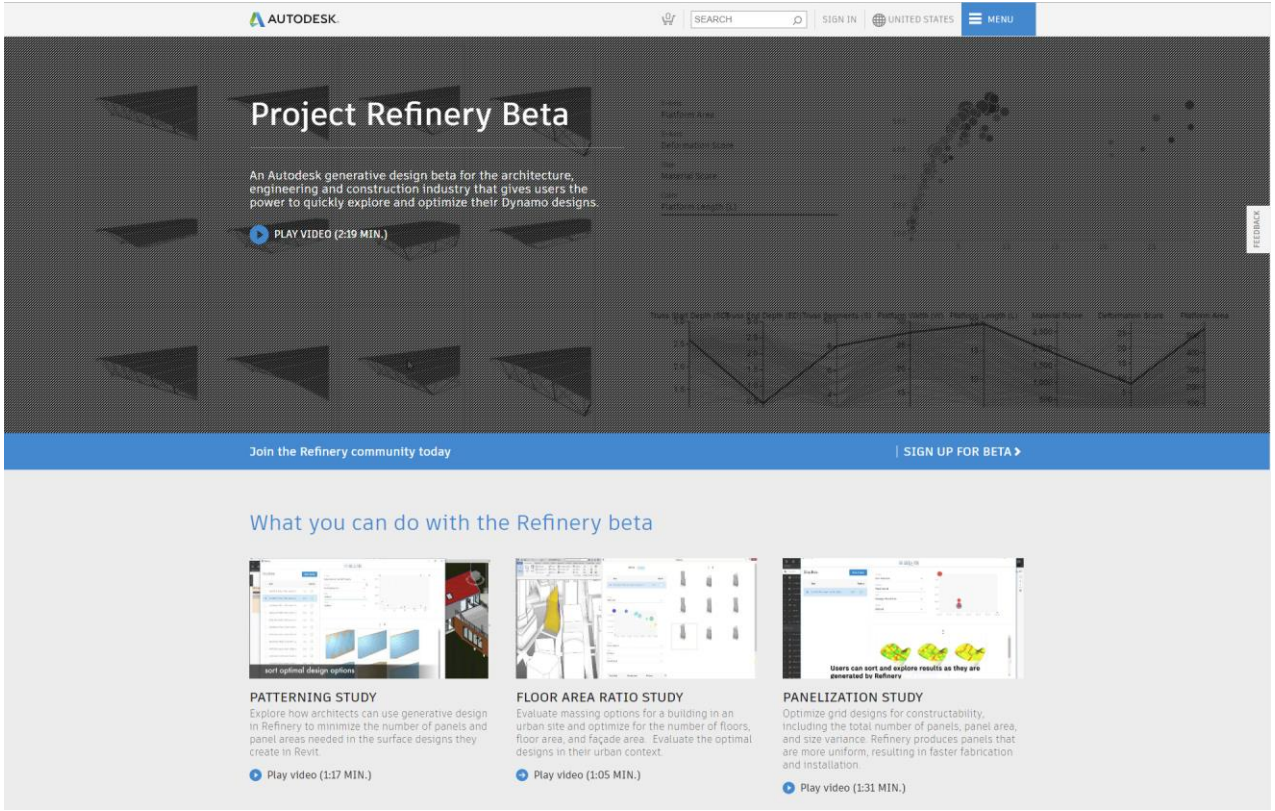
BUILDING CONTENT SUMMIT by DBEI

BILT EUR BY DBEI

What is Project Refinery?

- Automatically run Dynamo created logic to create options
- Works on Dynamo for Revit or Dynamo Sandbox
- Created for one customer VanWijnen 2/2018
- Public Beta 11/2018





autodesk.com/solutions/refinery-beta

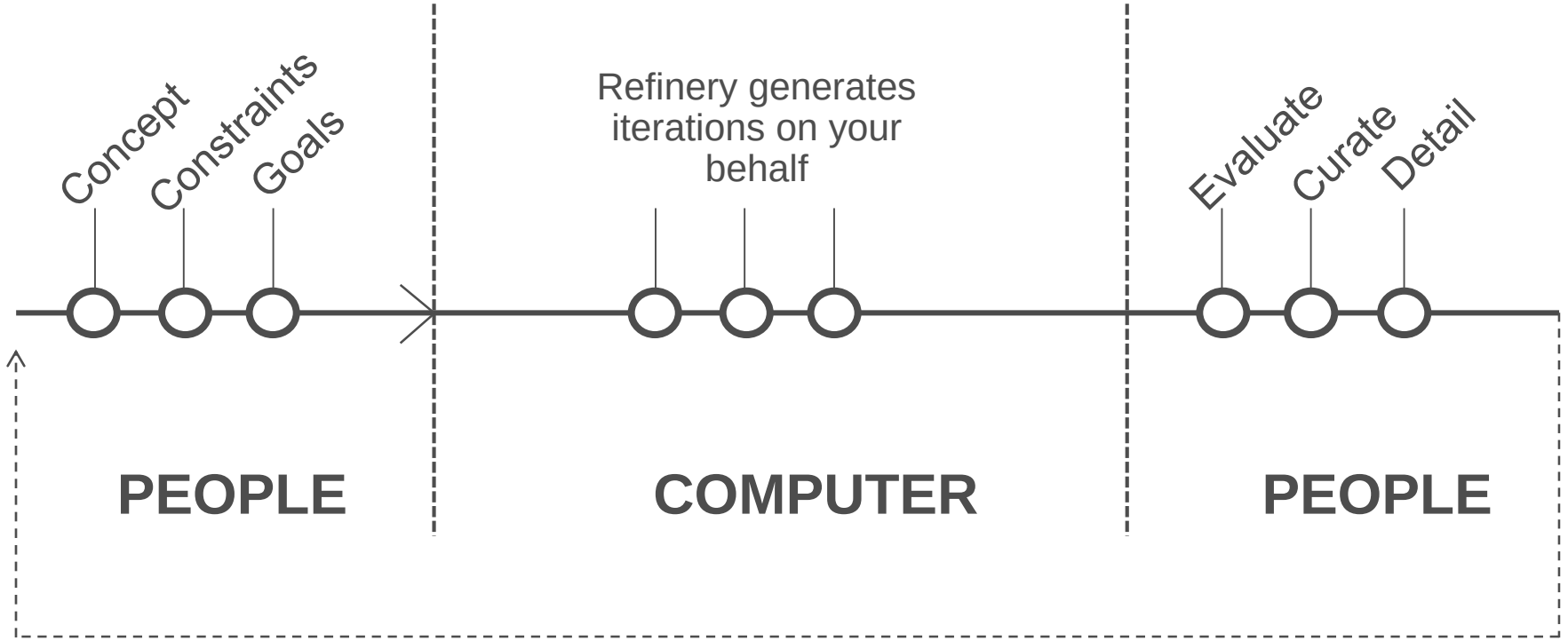
Generative Design Preparation

- How to get started

1. Understand what you want to achieve
2. Define your problem
3. Decide on ways to measure success
4. Think about how to review the results

Traditional Design Process

Generative Design Process



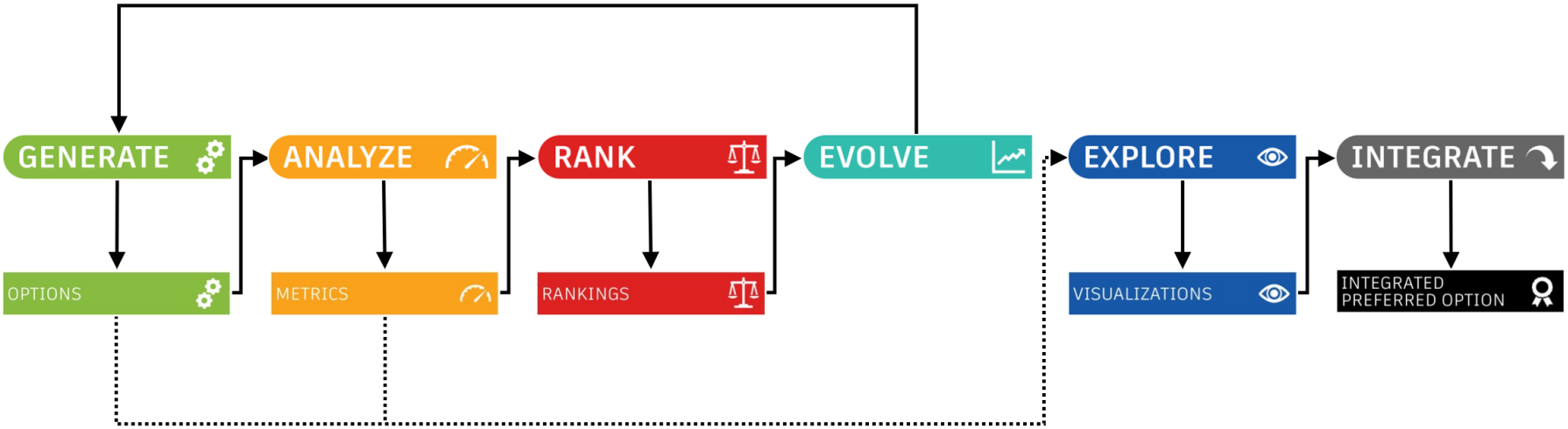
“The goal of generative design is not to automate the design process, or to replace human designers with artificial ones”

– Danil Nagy, The Living

DIGITAL BUILT ENVIRONMENT INSTITUTE

Generative Design Method

Process diagram



Example 1

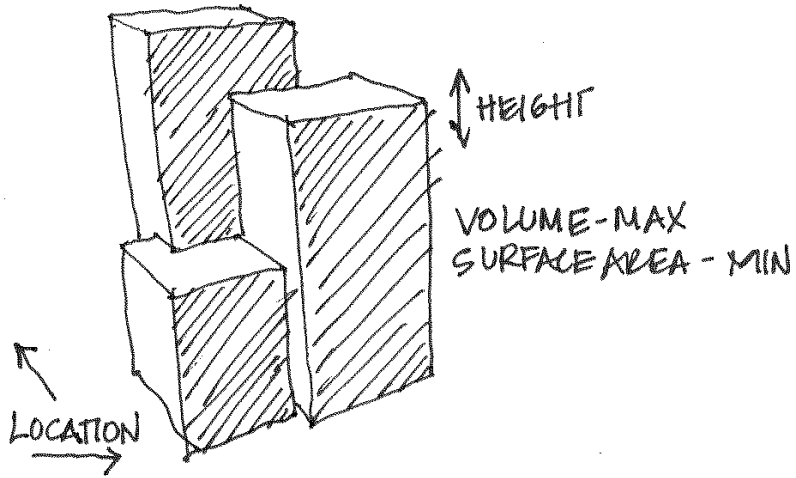
3 Box Building

Let’s start simple to understand concepts

Goals

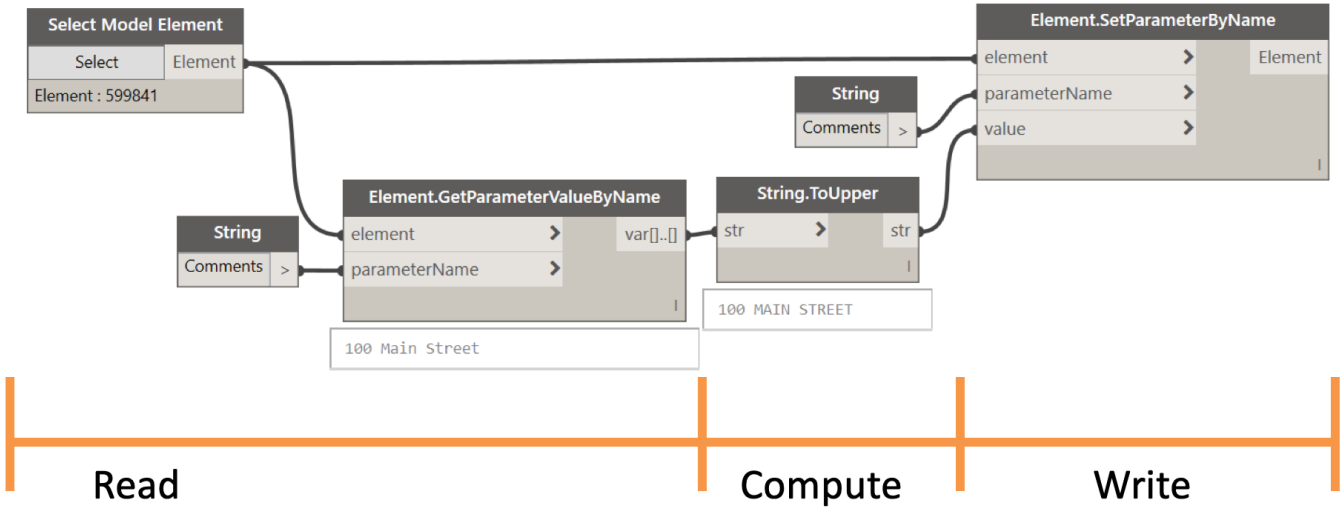
What are we solving for?

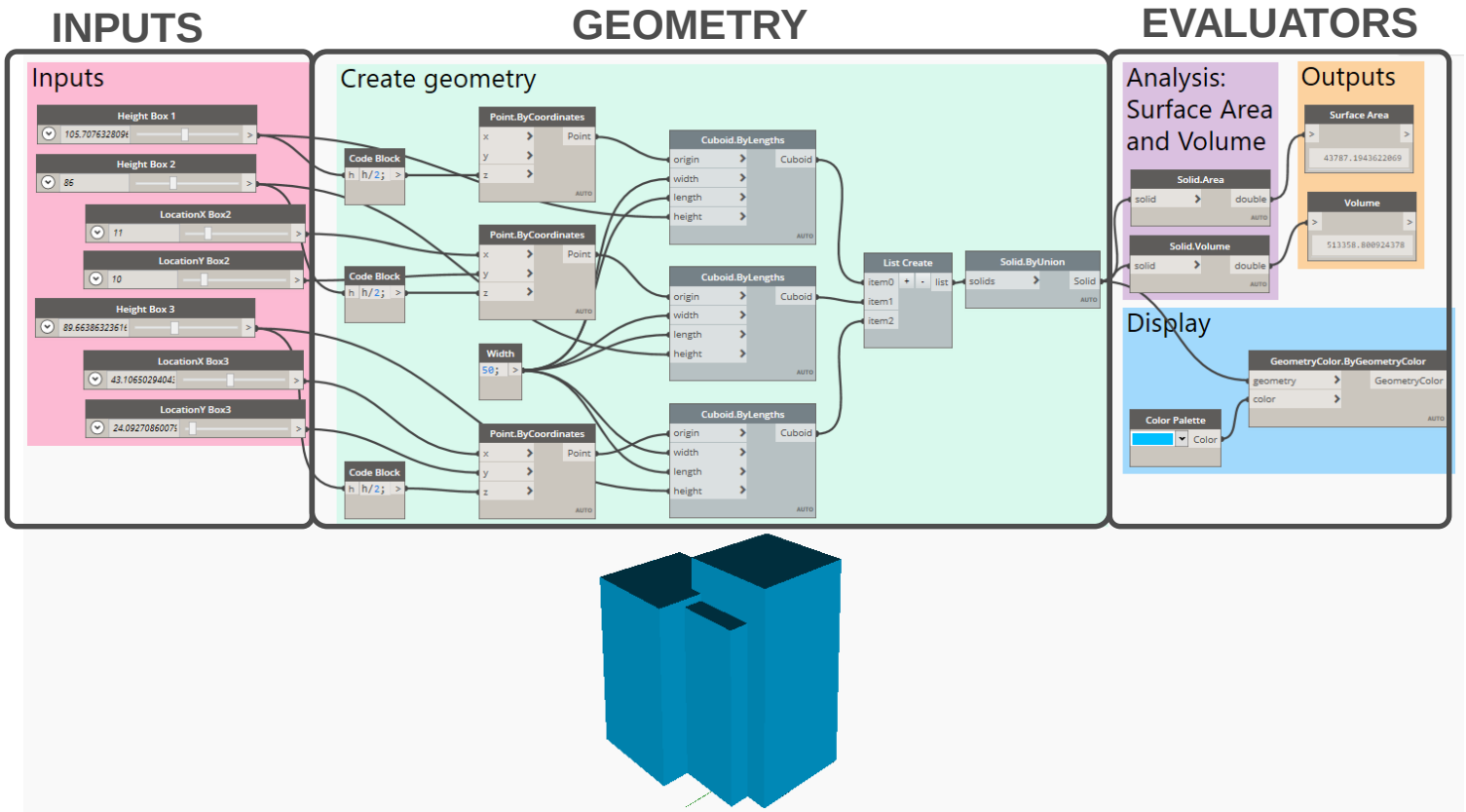
- Study a building mass to look at trade offs between volume and surface area
- Variable Inputs:
  1. Locations of 2 of the boxes
  2. Heights of all 3 boxes
- Goals:
  1. Maximum volume (rentable area)
  2. Minimum surface area (expensive façade)

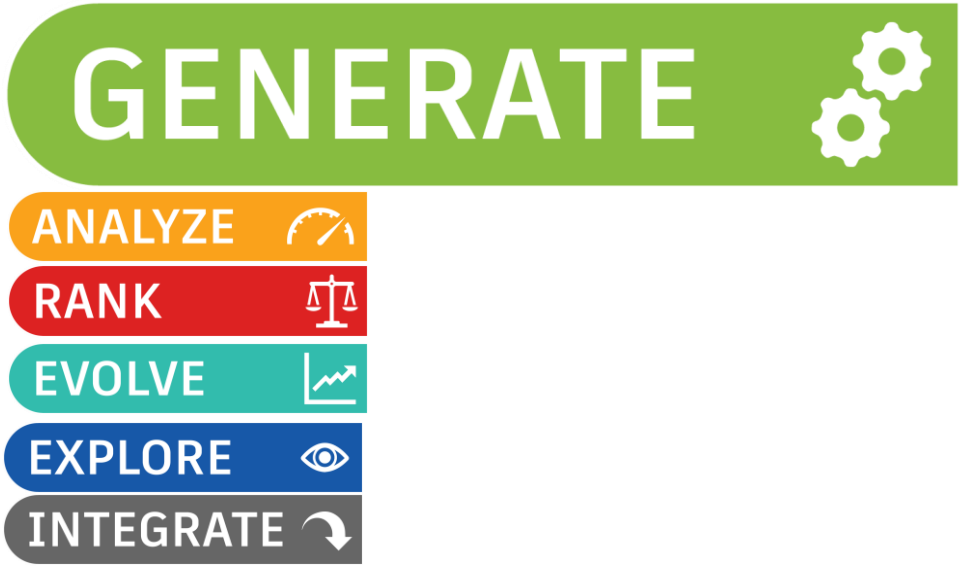


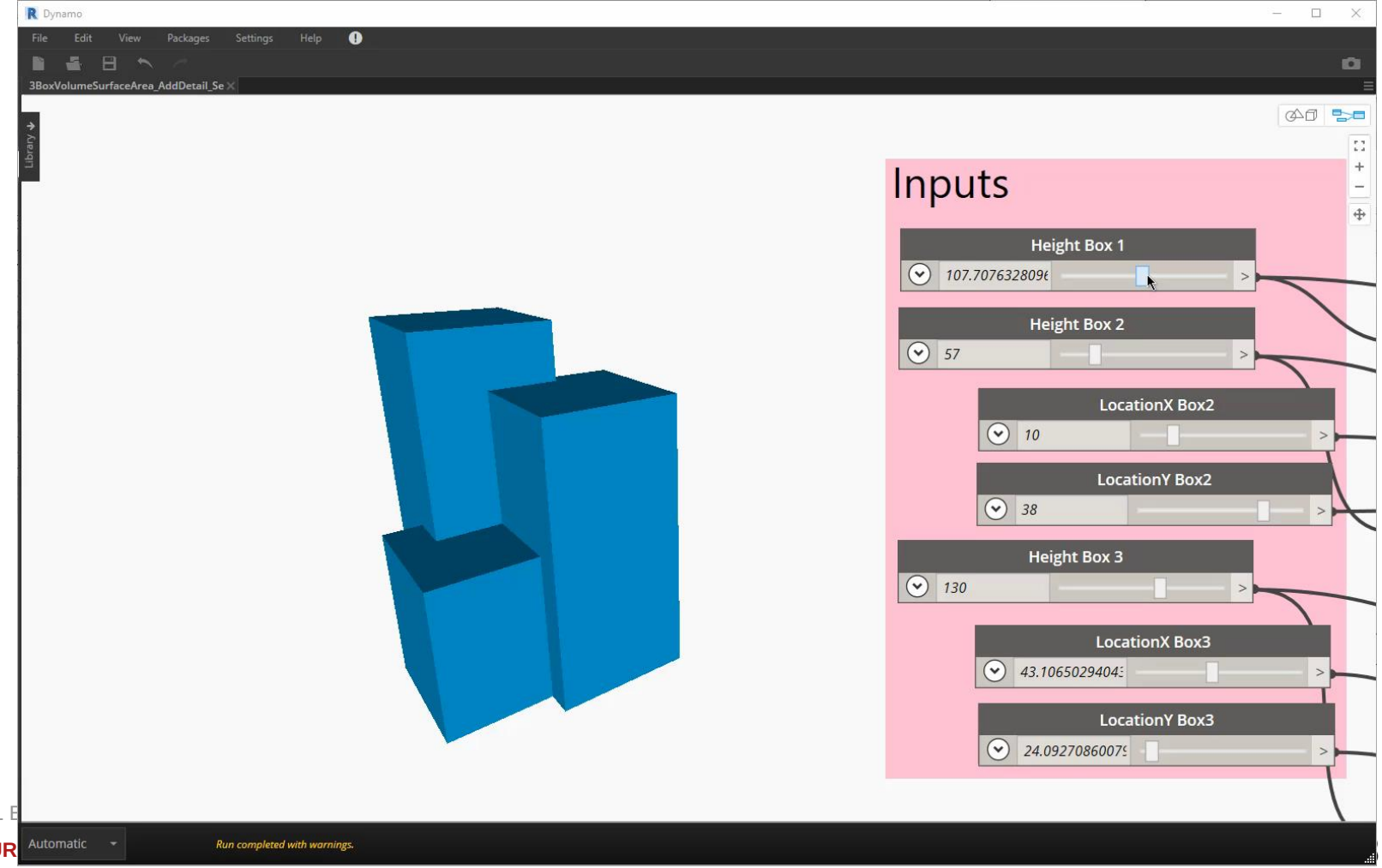
Quick Dynamo Primer

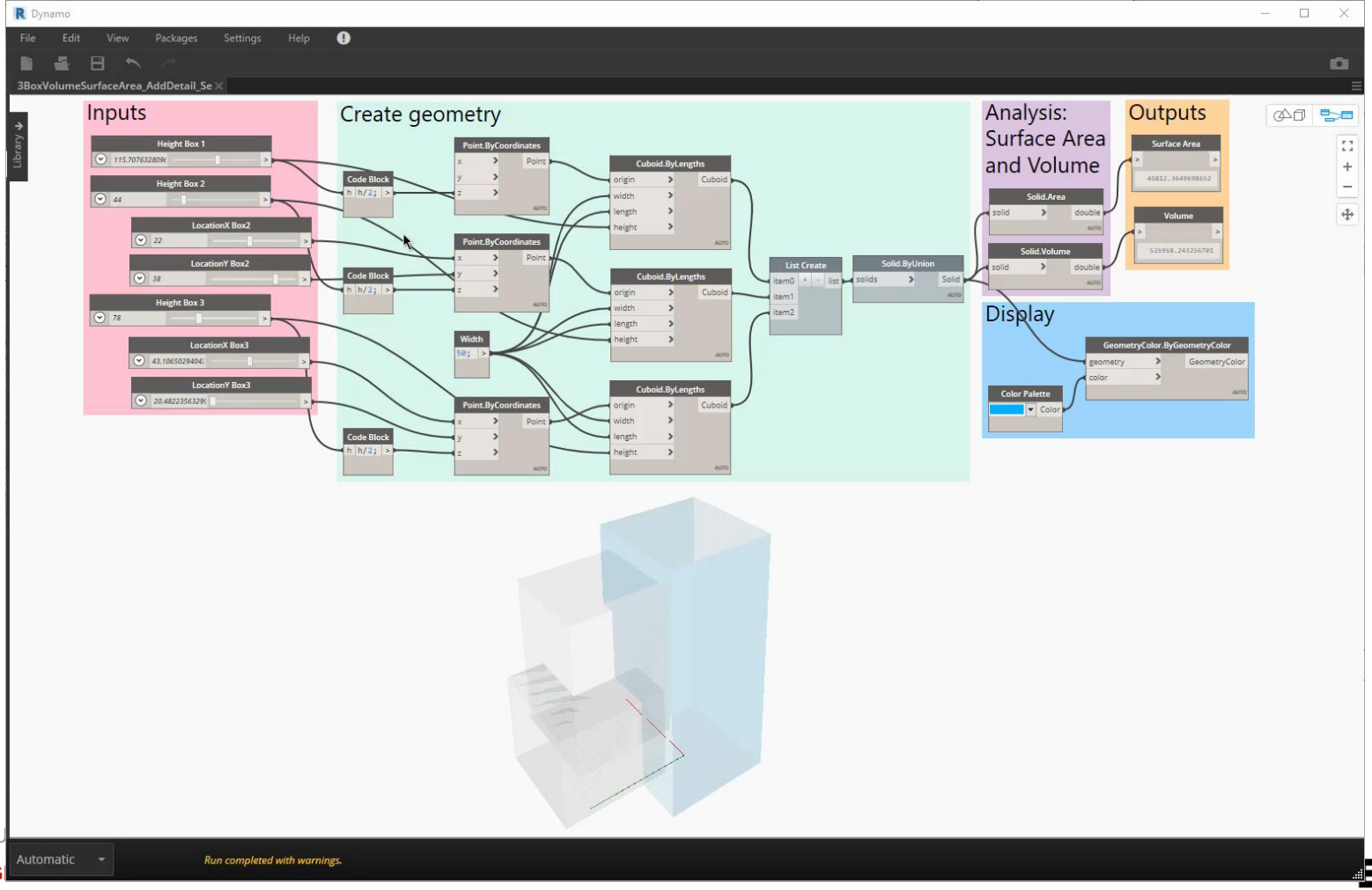
Visual programming for architects and engineers




INPUTS
GEOMETRY
EVALUATORS
Inputs
Height Box 1
Height Box 2
LocationX Box2
LocationY Box2
Height Box 3
LocationX Box3
LocationY Box3
Create geometry
Code Block
Point.ByCoordinates
Cuboid.ByLengths
Width
List Create
Solid.ByUnion
Analysis: Surface Area and Volume
Solid.Area
Solid.Volume
Outputs
Surface Area
Volume
Display
GeometryColor.ByGeometryColor
Color Palette


GENERATE
ANALYZE
RANK
EVOLVE
EXPLORE
INTEGRATE

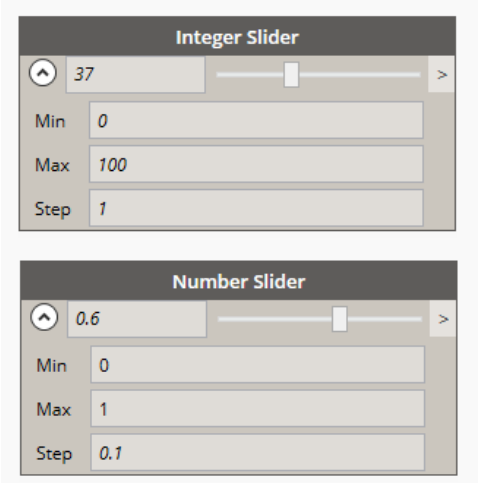

Dynamo
File
Edit
View
Packages
Settings
Help
3BoxVolumeSurfaceArea_AddDetail_Se
Library
Inputs
Height Box 1
107.707632809
Height Box 2
57
LocationX Box2
10
LocationY Box2
38
Height Box 3
130
LocationX Box3
43.1065029404
LocationY Box3
24.0927086007
Automatic
Run completed with warnings.


Dynamo
File
Edit
View
Packages
Settings
Help
3BoxVolumeSurfaceArea_AddDetail_Se
Inputs
Height Box 1
Height Box 2
LocationX Box2
LocationY Box2
Height Box 3
LocationX Box3
LocationY Box3
Create geometry
Code Block
Point.ByCoordinates
Cuboid.ByLengths
Width
List Create
Solid.ByUnion
Analysis: Surface Area and Volume
Solid.Area
Solid.Volume
Outputs
Surface Area
Volume
Display
GeometryColor.ByGeometryColor
Color Palette
Automatic
Run completed with warnings.

GENERATE

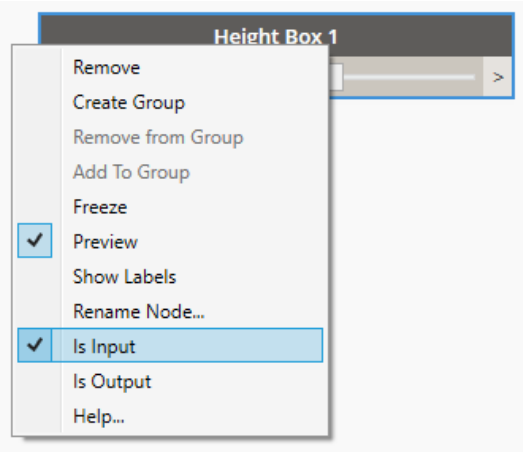
Input types

VARIABLE



FIXED

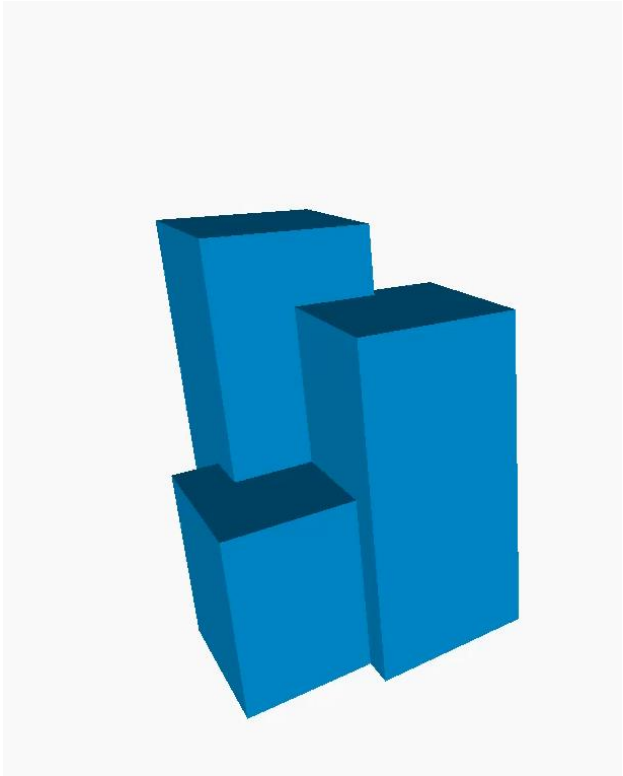
REFINERY



GENERATE

Create a design system: 3 key concepts

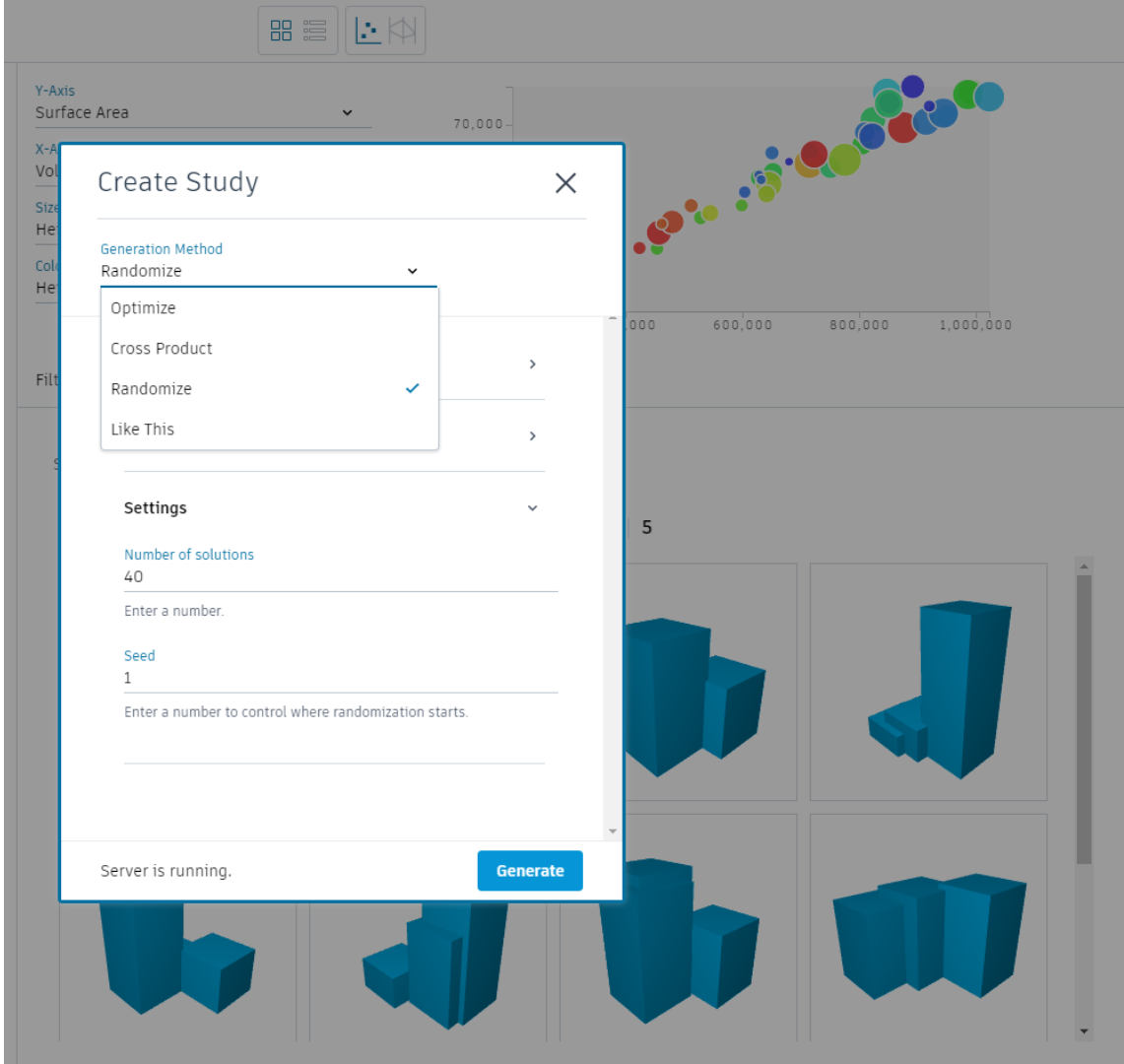
- How you parametrize the model is critical
- Too many inputs will lead to a design space that is too big to explore
- Too few inputs won't yield a big enough design space



GENERATE

Refinery Generation Methods

1. Optimize
2. Cross Product
3. Randomize
4. Like This



GENERATE

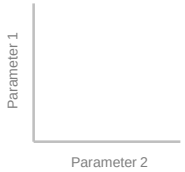
Refinery Methods

4 TYPES

RANDOMIZE

INPUTS

DESIGN SPACE



OUTCOME

GENERATE

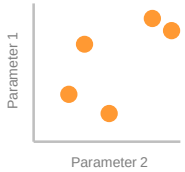
Refinery Methods

4 TYPES

RANDOMIZE

INPUTS

DESIGN SPACE



OUTCOME

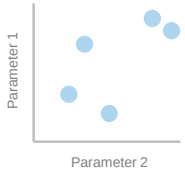
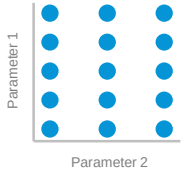
5 DESIGNS

defined by user

GENERATE

Refinery Methods

4 TYPES

	INPUTS	DESIGN SPACE	OUTCOME
RANDOMIZE			**5 DESIGNS** # defined by user
CROSS PRODUCT	5x 3x 		**5 x 3 = 15 DESIGNS** user defines sampling density

GENERATE

Refinery Methods

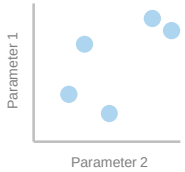
4 TYPES

INPUTS

DESIGN SPACE

OUTCOME

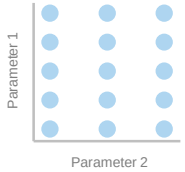
RANDOMIZE



5 DESIGNS
defined by user

CROSS PRODUCT

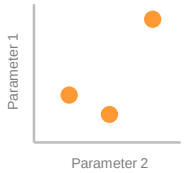
5x
3x



5 x 3 = 15 DESIGNS
user defines sampling density

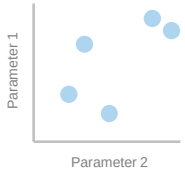
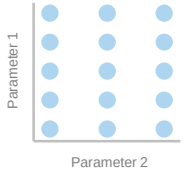
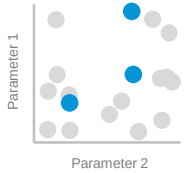
OPTIMIZE

generation 1 population of 3



GENERATE

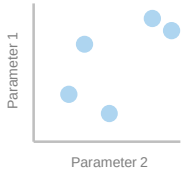
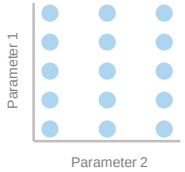
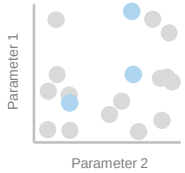
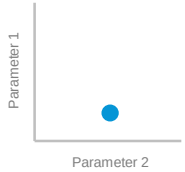
Refinery Methods

4 TYPES	INPUTS	DESIGN SPACE	OUTCOME
RANDOMIZE			**5 DESIGNS** # defined by user
CROSS PRODUCT	5x 3x 		**5 x 3 = 15 DESIGNS** user defines sampling density
OPTIMIZE	generation 1 population of 3 generation 2 population of 3 … **generation 6 population of 3** 		**3 x 6 = 18 DESIGNS** user defines population size and number of generations

GENERATE

Refinery Methods

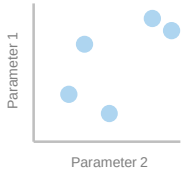
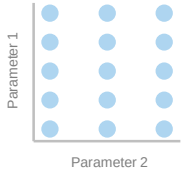
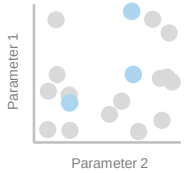
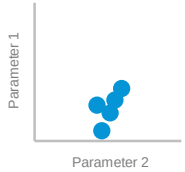
4 TYPES

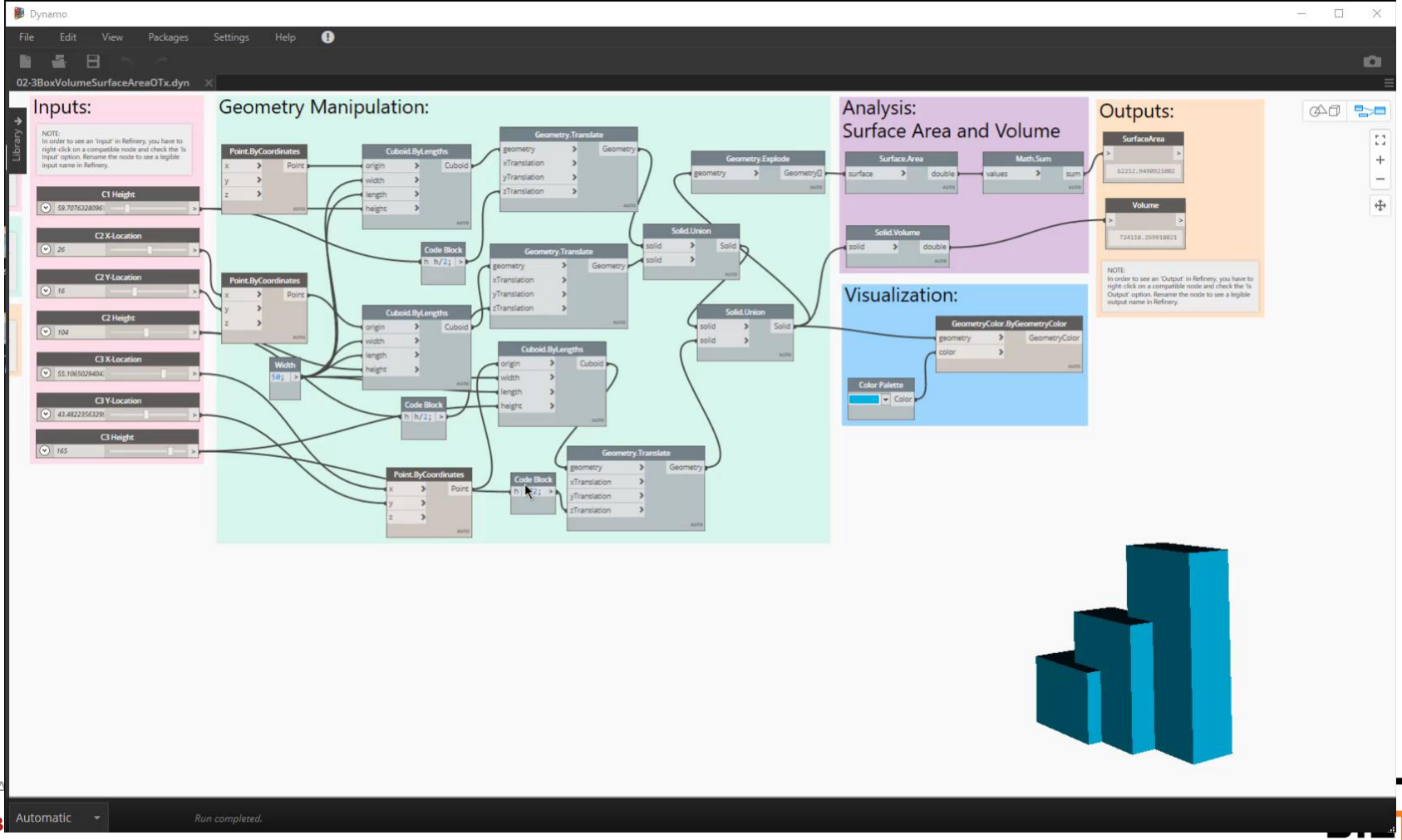
	INPUTS	DESIGN SPACE	OUTCOME
RANDOMIZE			**5 DESIGNS** # defined by user
CROSS PRODUCT	5x 3x 		**5 x 3 = 15 DESIGNS** user defines sampling density
OPTIMIZE generation 1 population of 3 generation 2 population of 3 … generation 6 population of 3			**3 x 6 = 18 DESIGNS** user defines population size and number of generations
LIKE THIS			

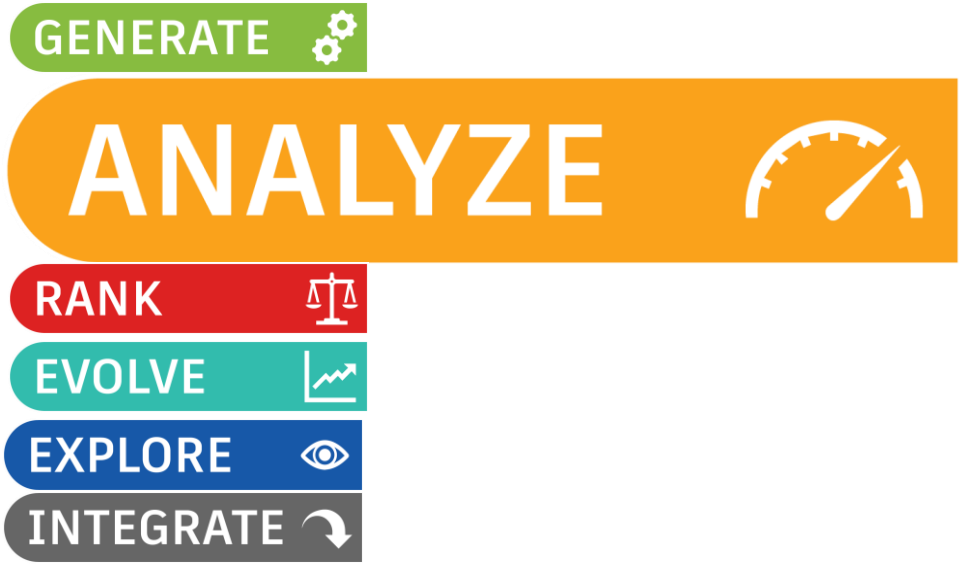
GENERATE

Refinery Methods

4 TYPES

	INPUTS	DESIGN SPACE	OUTCOME
RANDOMIZE			**5 DESIGNS** # defined by user
CROSS PRODUCT	5x 3x 		**5 x 3 = 15 DESIGNS** user defines sampling density
OPTIMIZE generation 1 population of 3 generation 2 population of 3 … generation 6 population of 3			**3 x 6 = 18 DESIGNS** user defines population size and number of generations
LIKE THIS			**5 DESIGNS** # defined by user


Dynamo
File
Edit
View
Packages
Settings
Help
02-3BoxVolumeSurfaceAreaOTx.dyn
Inputs:
Geometry Manipulation:
Analysis:
Surface Area and Volume
Outputs:
Visualization:
C1 Height
C2 X-Location
C2 Y-Location
C2 Height
C3 X-Location
C3 Y-Location
C3 Height
Width
Point.ByCoordinates
Cuboid.ByLengths
Geometry.Translate
Geometry.Explode
Solid.Union
Code Block
Surface.Area
Math.Sum
Solid.Volume
SurfaceArea
Volume
GeometryColor.ByGeometryColor
Color Palette
Automatic
Run completed.



DIGITAL BUILT WEEK EUROPE

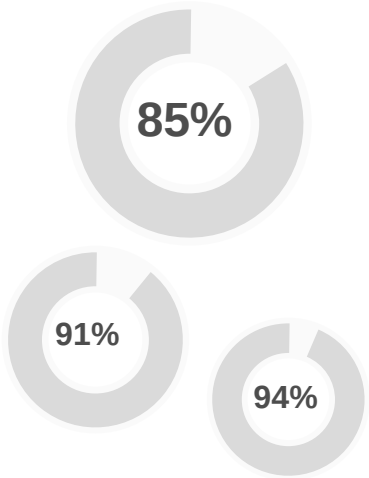
EDINBURGH

10 – 12 OCT 2019

ANALYZE

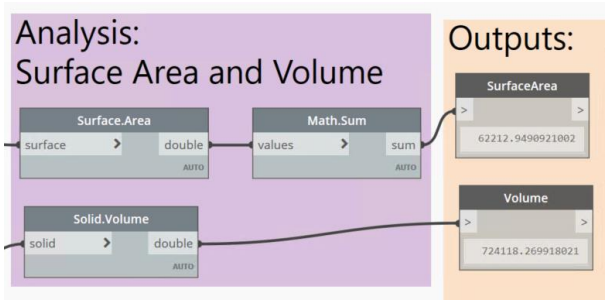
Judging outcomes

- Critical 2nd half of the problem
- Set of measures that determine how the building is performing
- Why?
  - Compare designs objectively (apples to apples)
  - Evaluate designs based on non-intuitive measures
  - Explore more designs than is possible manually

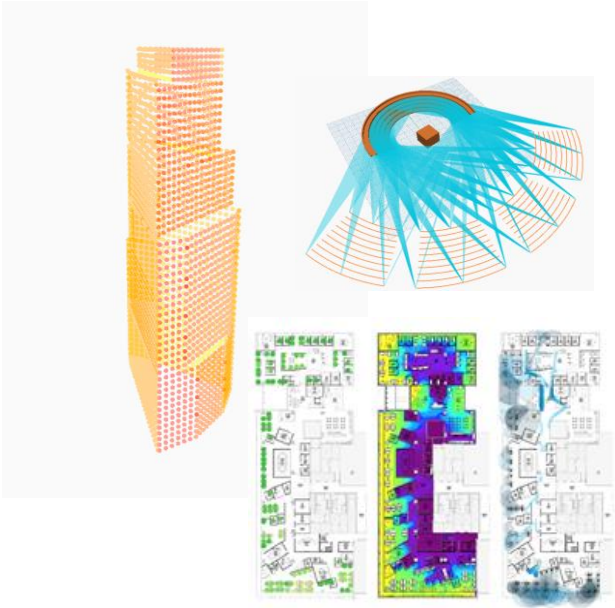


ANALYZE

Types of measurement



SIMPLE
length x width=area



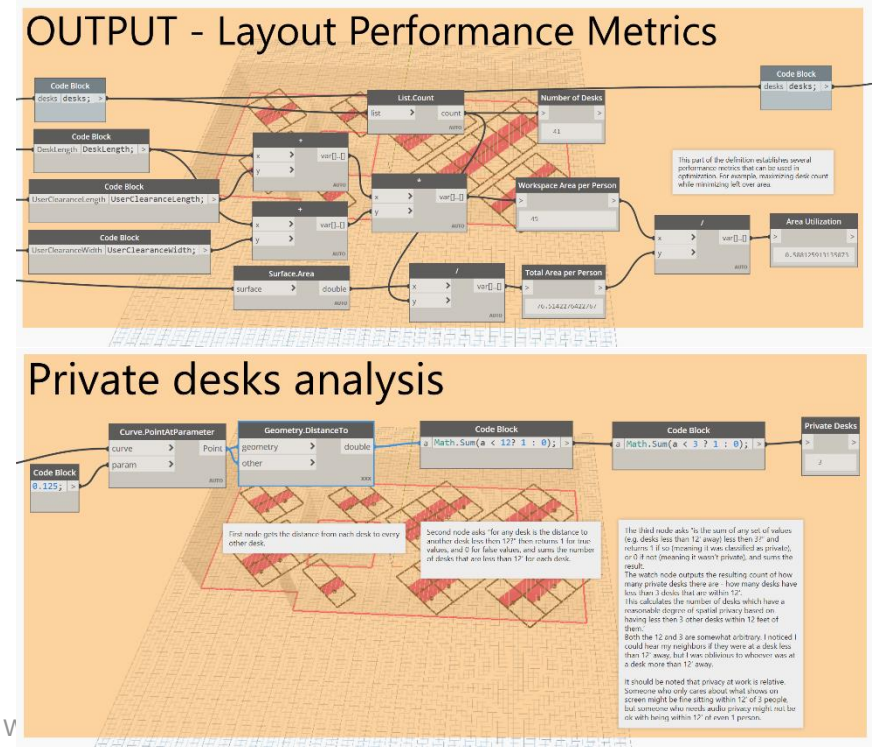
COMPLEX
Simulation

BEAUTY
PERSONAL
PREFERENCE
AESTHETIC
SENSIBILITY

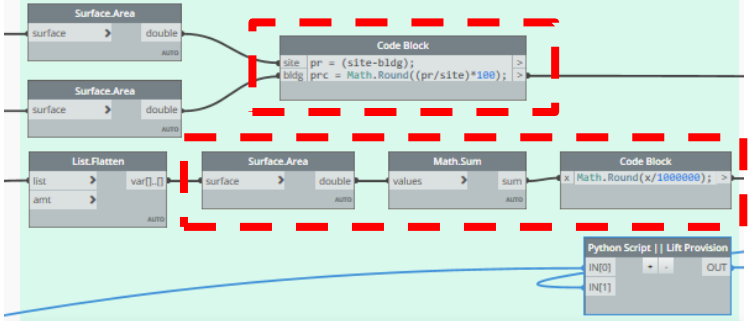
UNMEASURABLE

ANALYZE

Types of measurement



Math:
Calculate public realm area, total building area and lift provision area



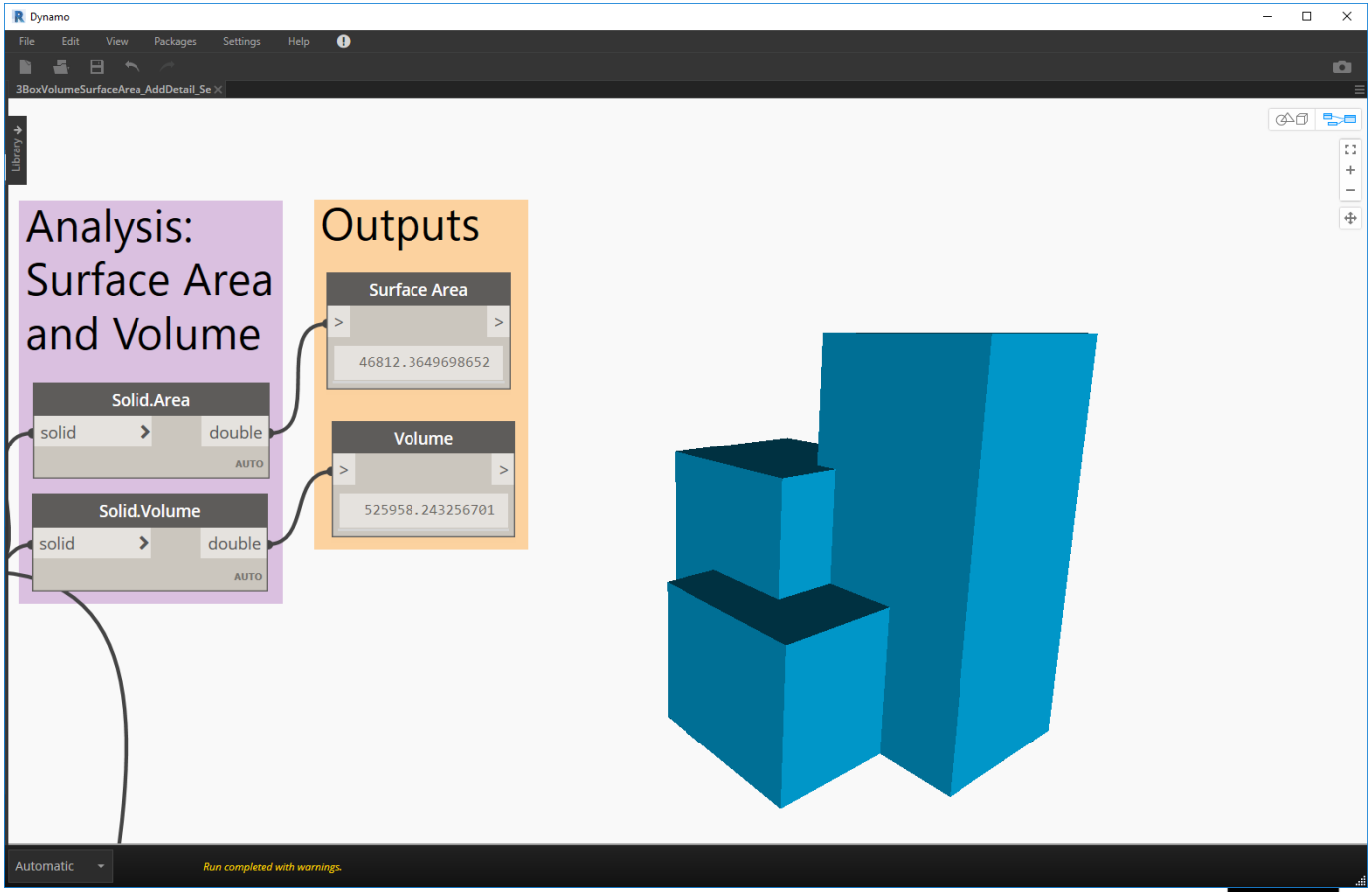
Python Script || Lift Provision

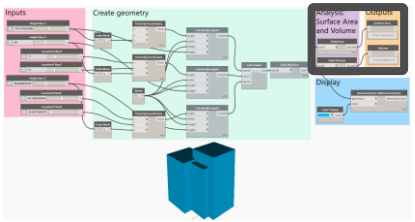
```
__author__ = 'Antonio Rodriguez'
__version__ = '1.0.0'

import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
import math

nFloors = IN[0]
buildingTotalArea = IN[1]

liftProvision = (buildingTotalArea*(-0.18))/(nFloors-47.5)

OUT = abs(liftProvision)
```

Run | Save Changes | Revert

ANALYZE

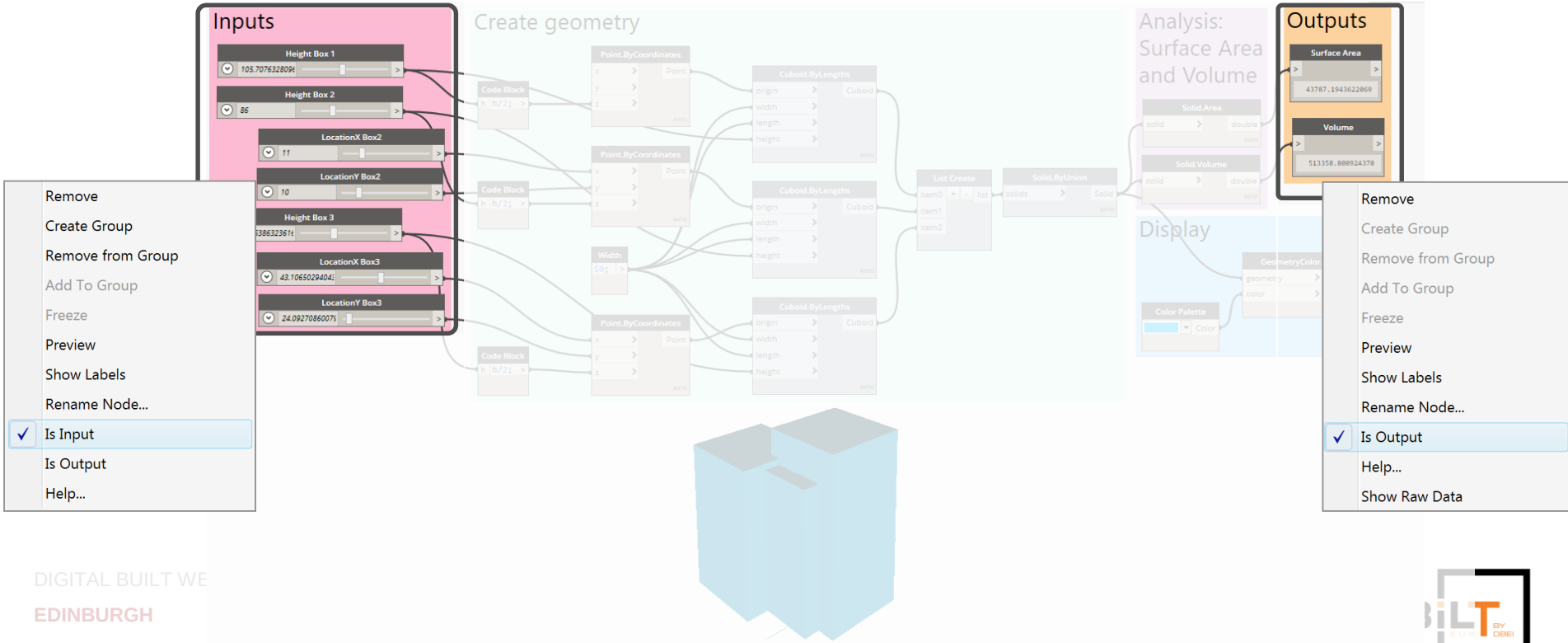
Measuring goals





ANALYZE

Designating Refinery inputs and outputs



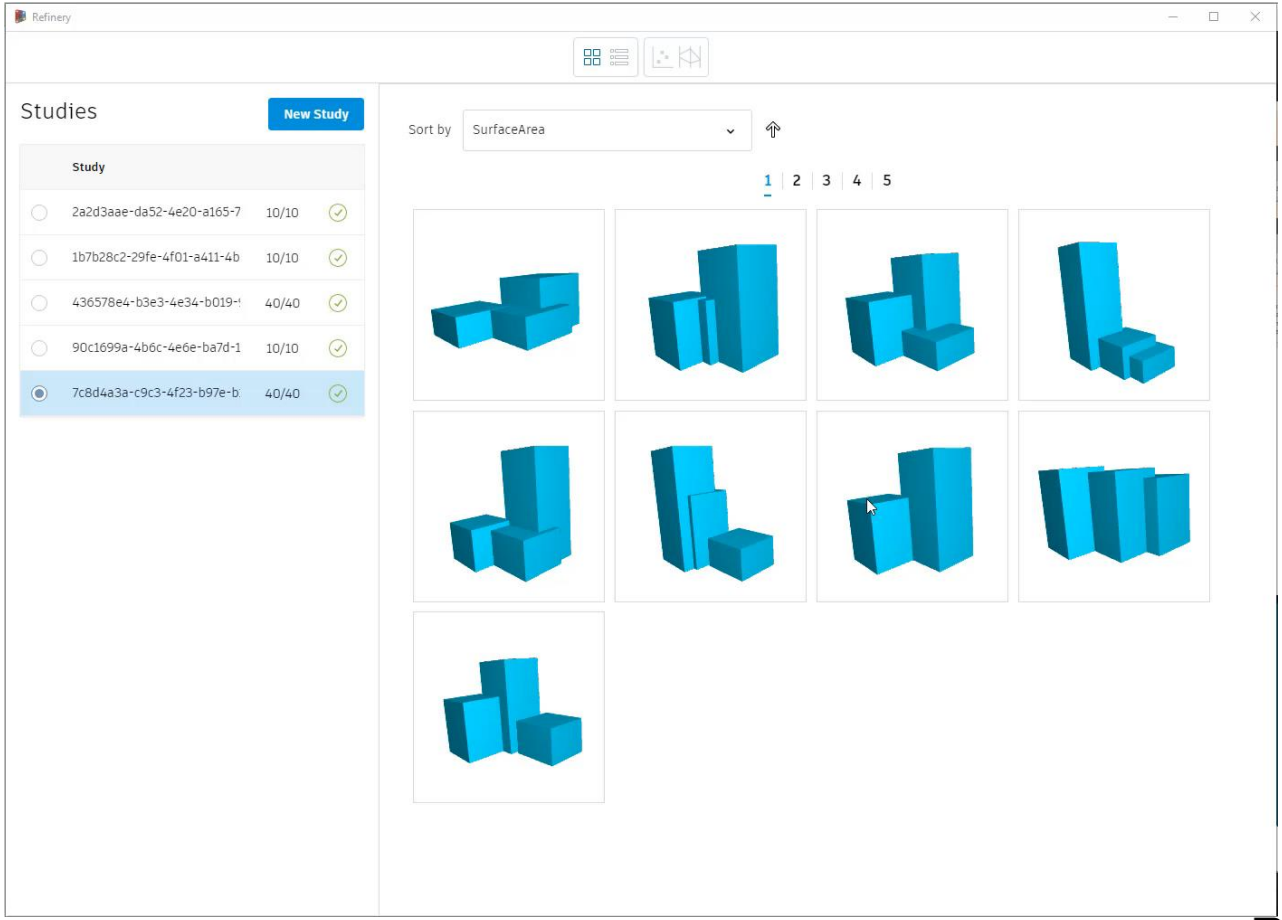
DIGITAL BUILT WEEK EUROPE

EDINBURGH

10 – 12 OCT 2019

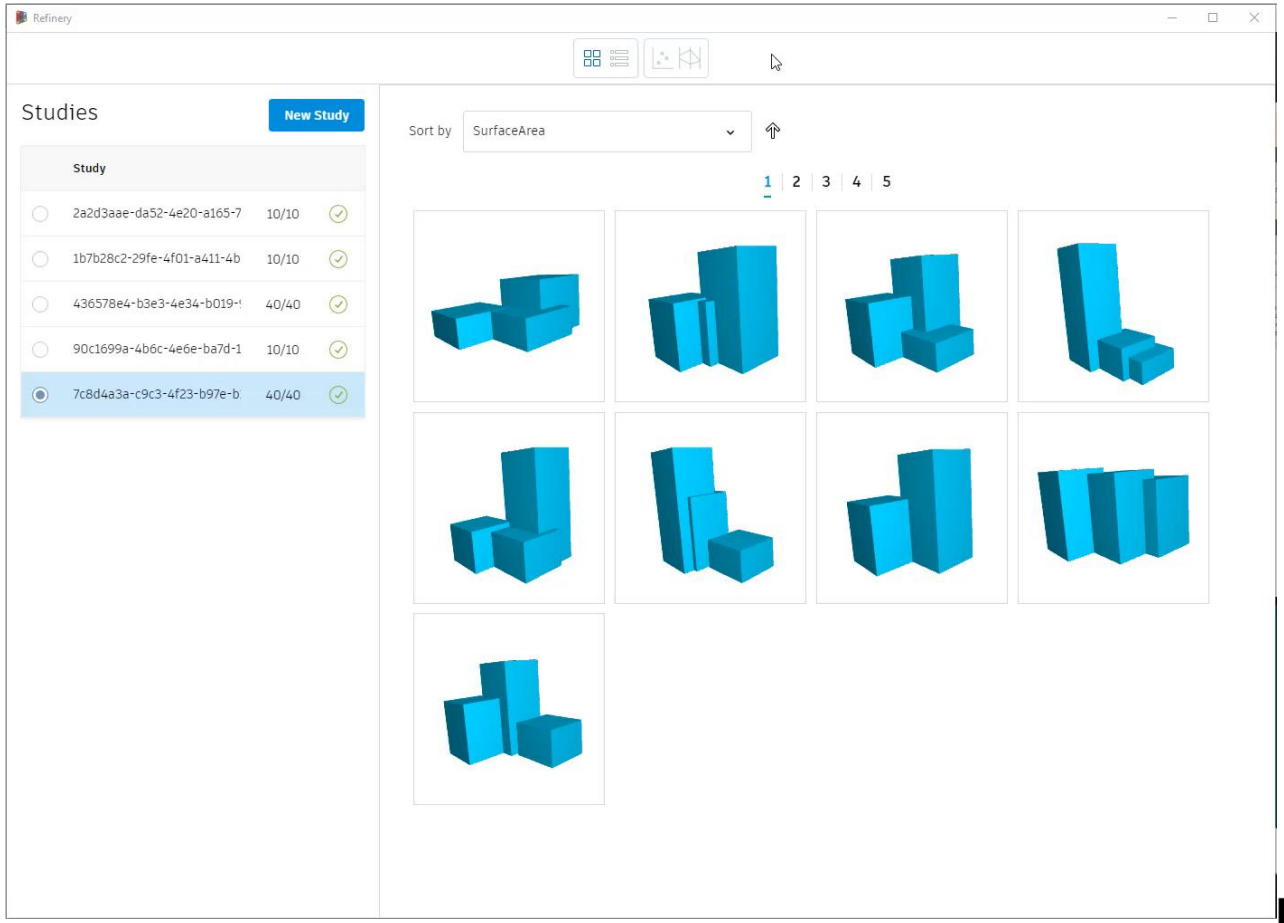
RANK

Refinery sort



RANK

Manual "optimization"



DIGITAL BUILT WEEK EUROPE

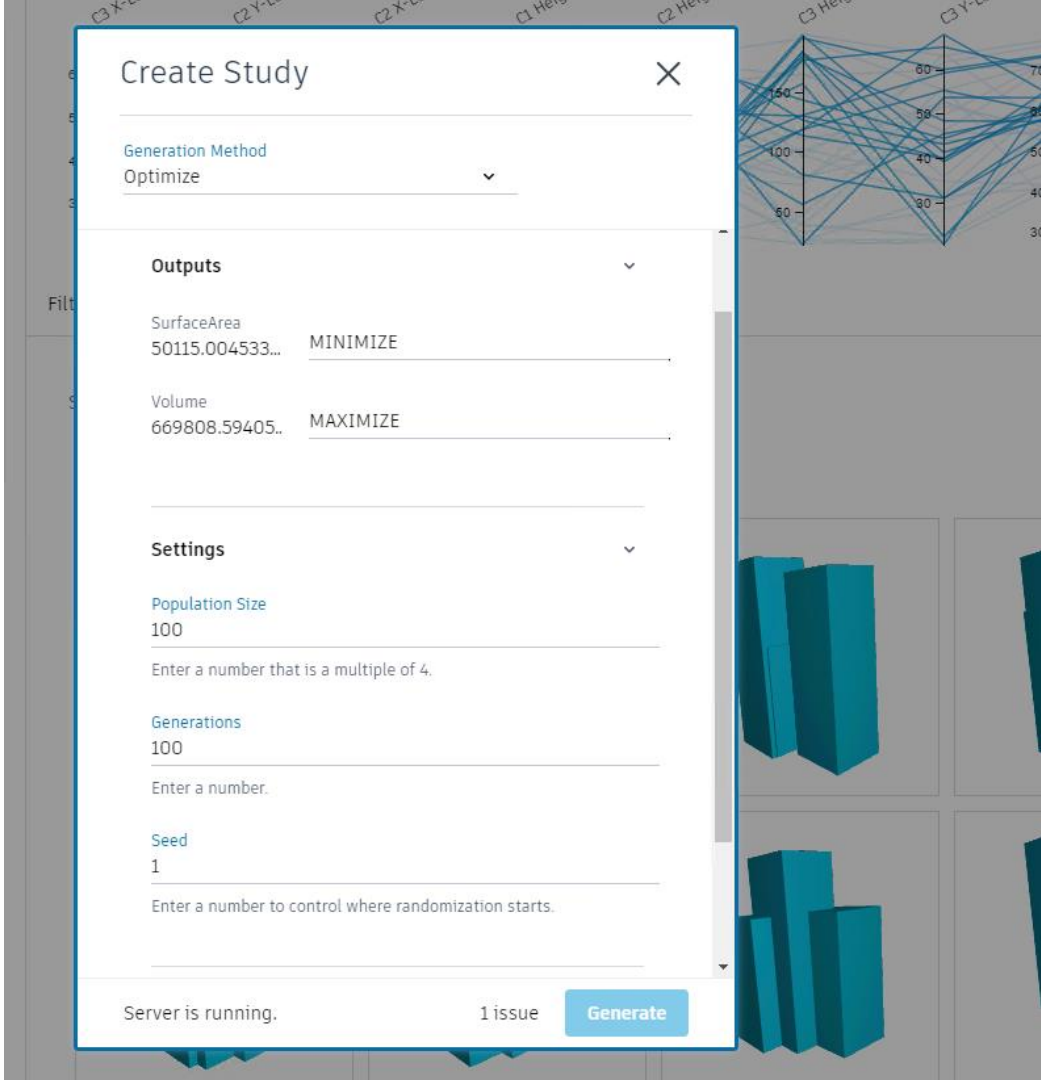
EDINBURGH

10 – 12 OCT 2019

EVOLVE

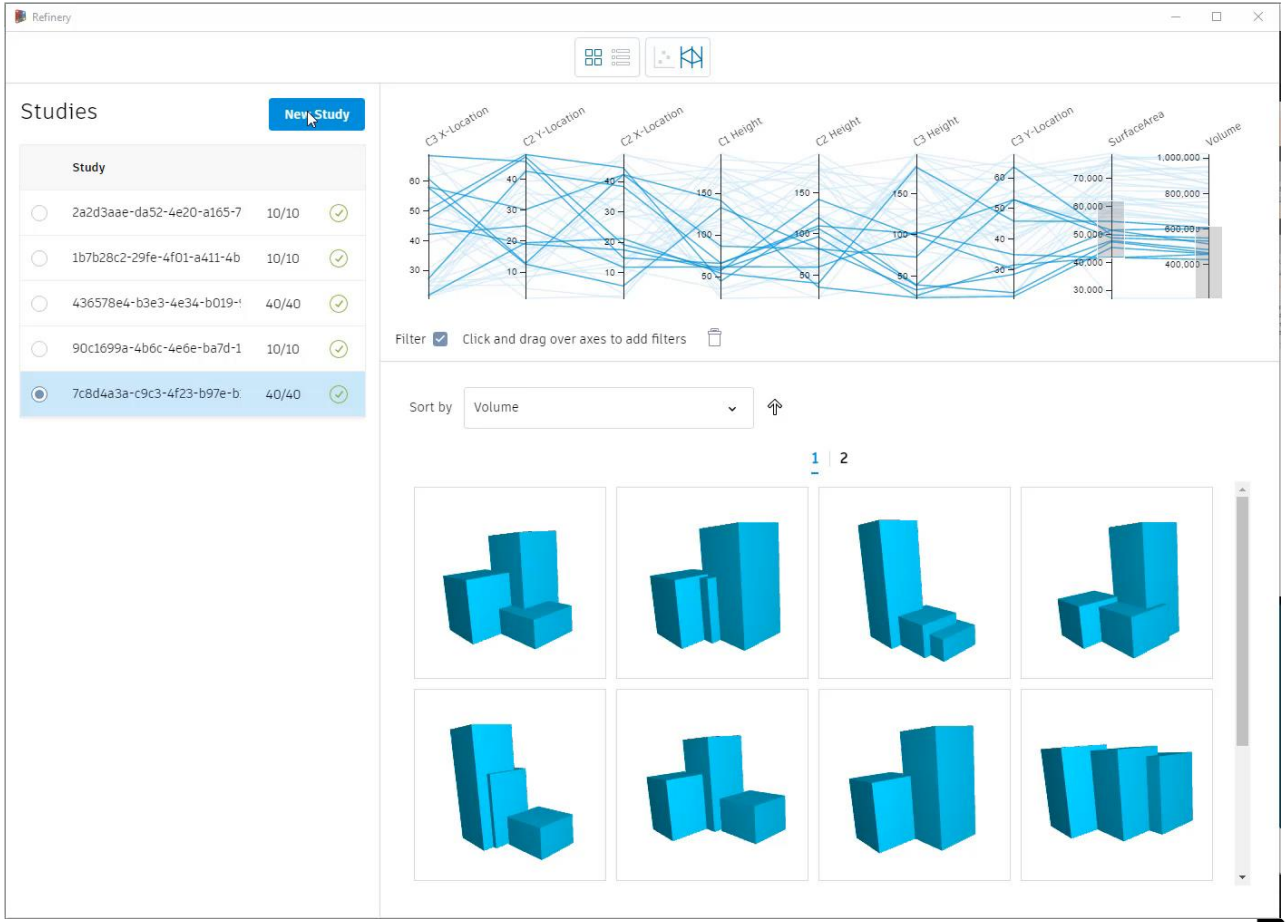
Parameters that affect the algorithm

1. Output goals
2. Population size
3. Generation size
4. Seed



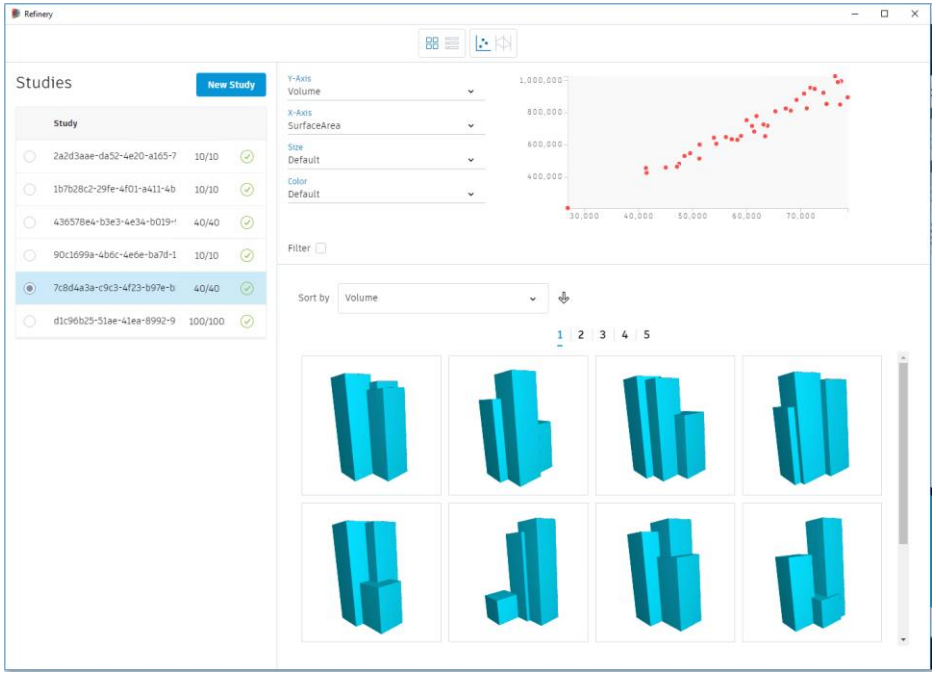
EVOLVE

Refinery optimization

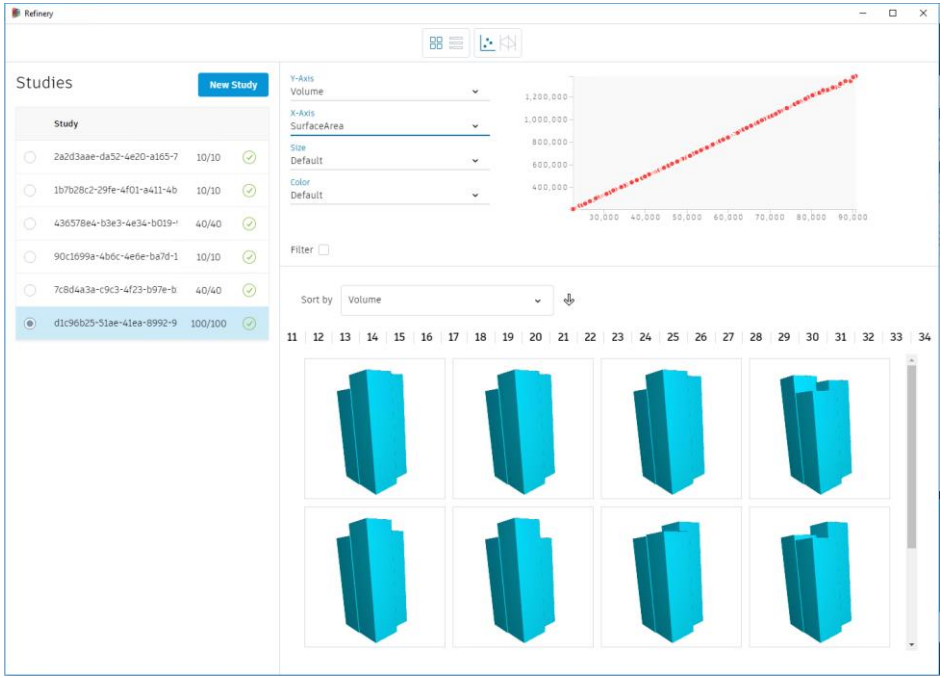


EVOLVE

Refinery randomization vs. optimization



40 random runs – 10 sec



100x100 optimization – 4 min

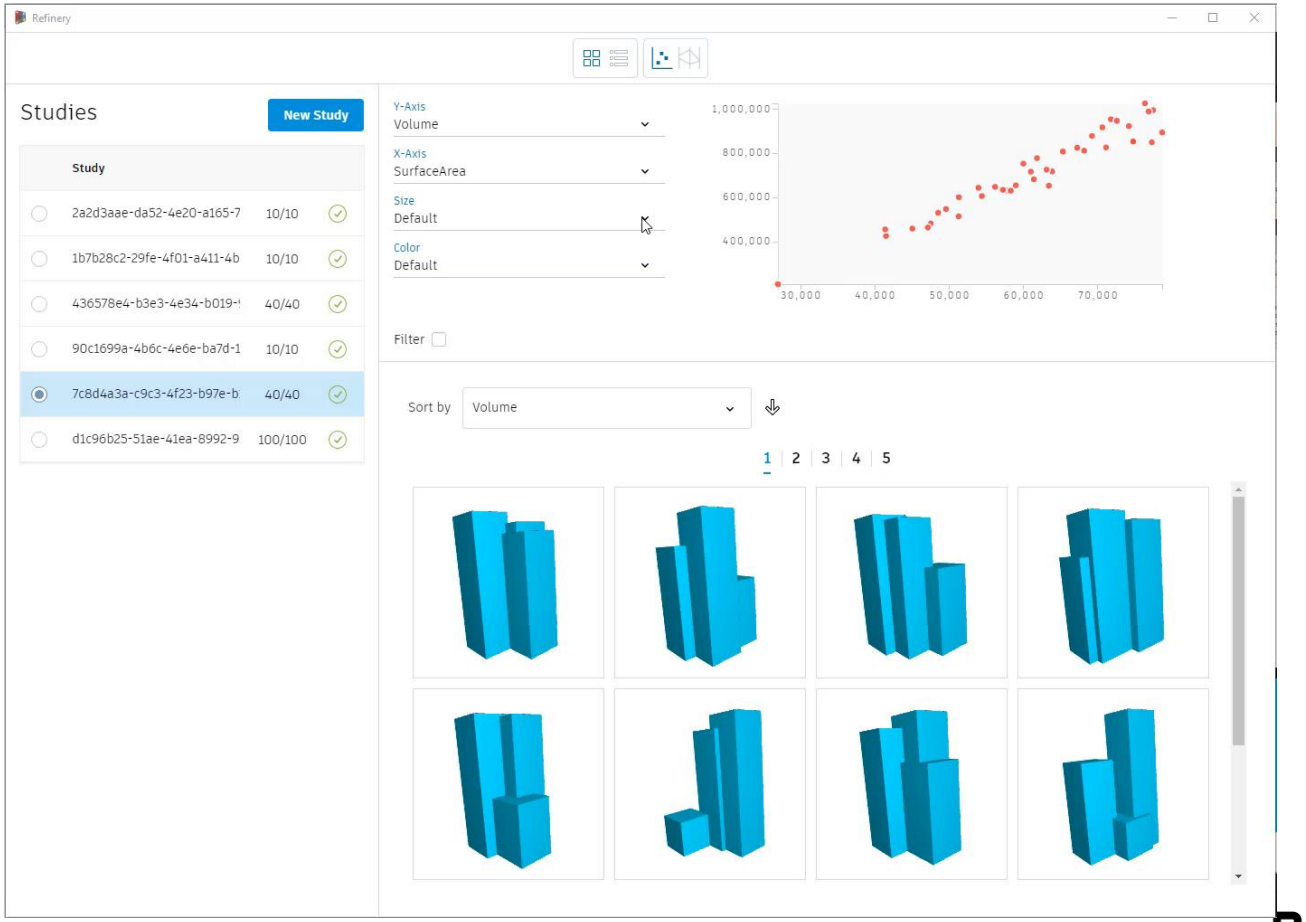
DIGITAL BUILT WEEK EUROPE

EDINBURGH

10 – 12 OCT 2019

EXPLORE

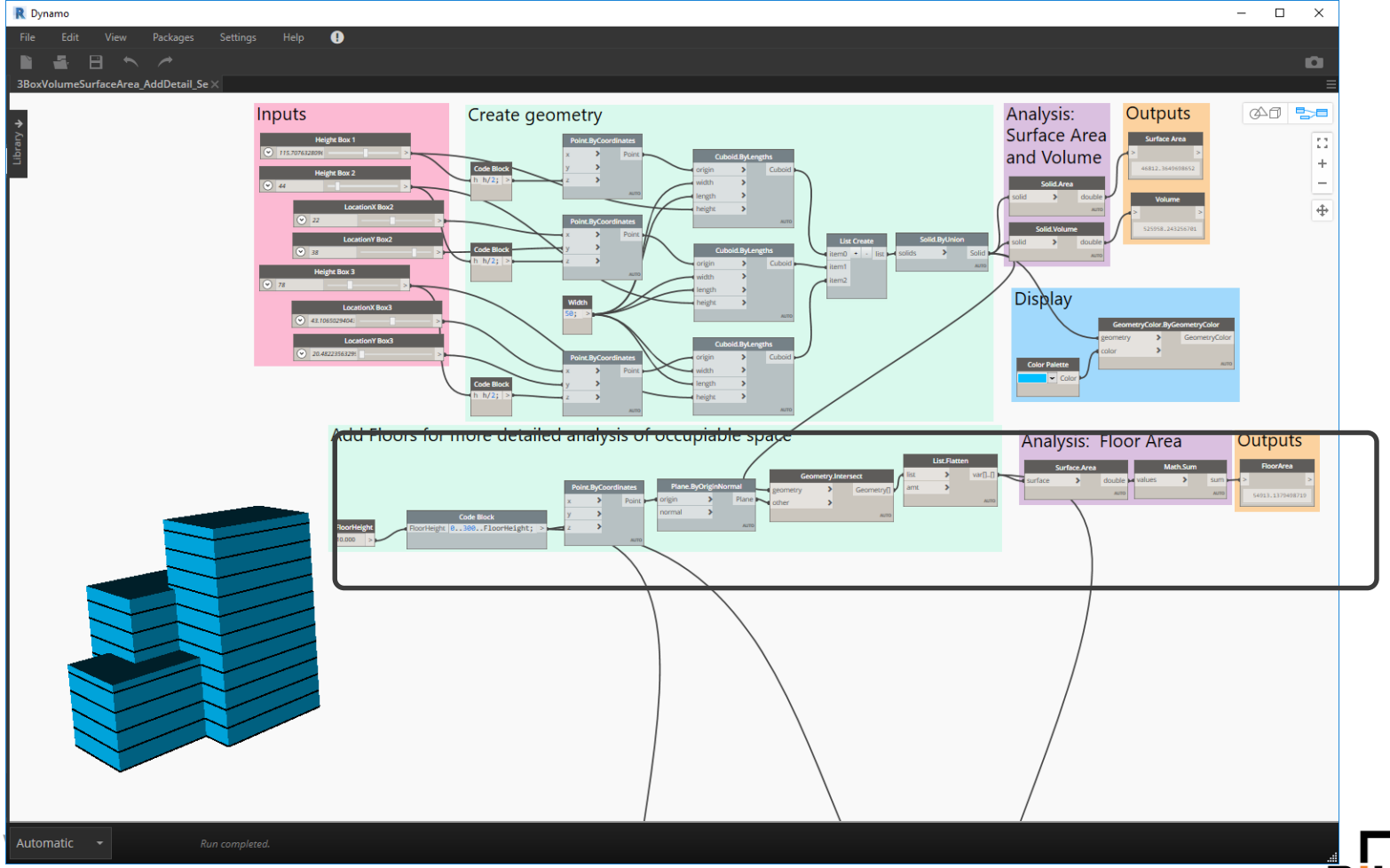
Refinery Explore UI



DIGITAL BUILT WEEK EUROPE

EDINBURGH

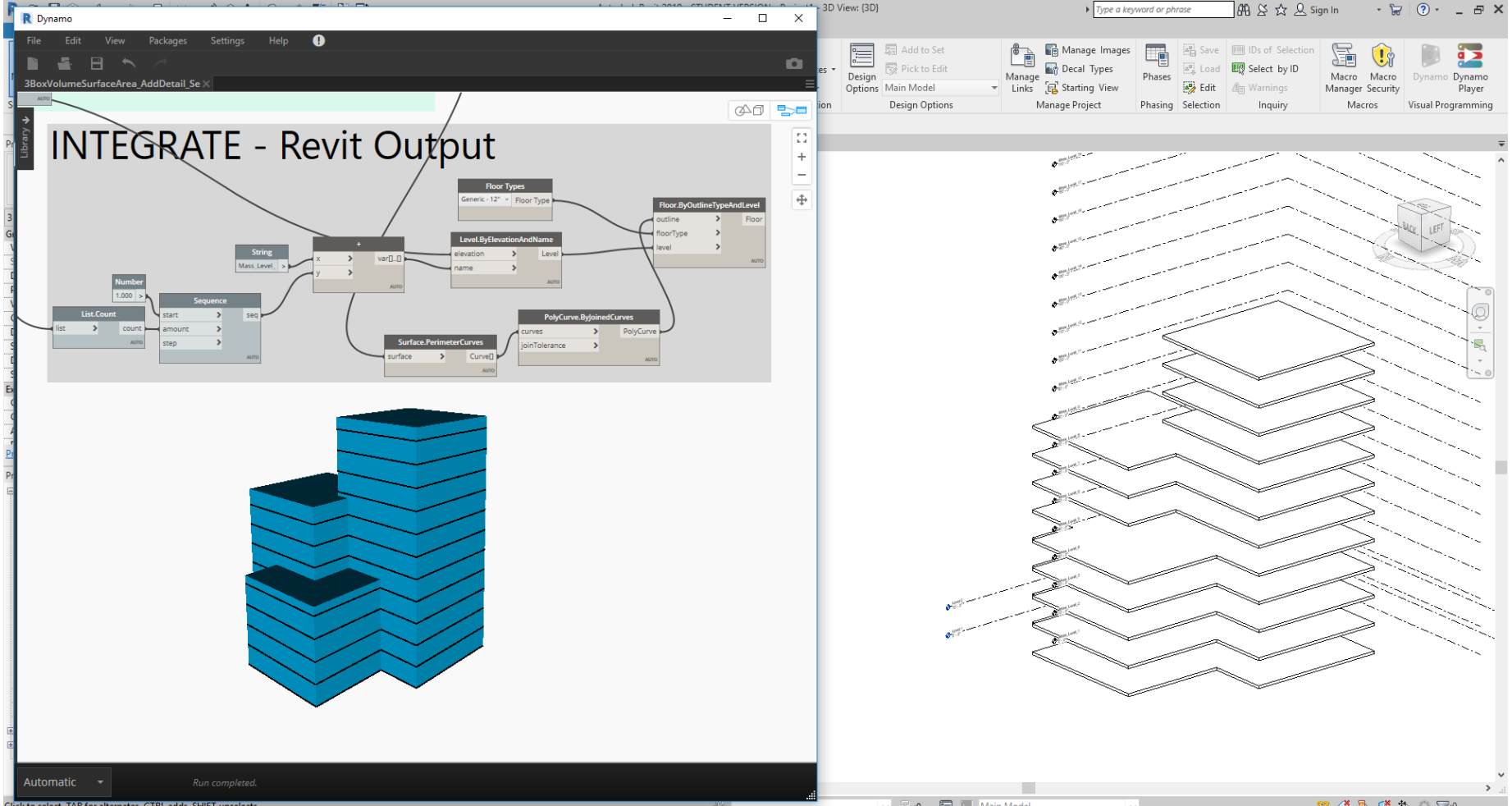
10 – 12 OCT 2019



DIGITAL BUILT

EDINBURGH

10 – 12 OCT 2019


Dynamo
File Edit View Packages Settings Help
3BoxVolumeSurfaceArea_AddDetail_Se
INTEGRATE - Revit Output
Floor Types
Floor.ByOutlineTypeAndLevel
Level.ByElevationAndName
String
Number
Sequence
List.Count
Surface.PerimeterCurves
PolyCurve.ByJoinedCurves
Automatic
Run completed.
Click to select, TAB for alternates, CTRL adds, SHIFT unselects.
Design Options
Manage Links
Manage Images
Decal Types
Starting View
Phases
Macro Manager
Macro Security
Dynamo Player
Main Model

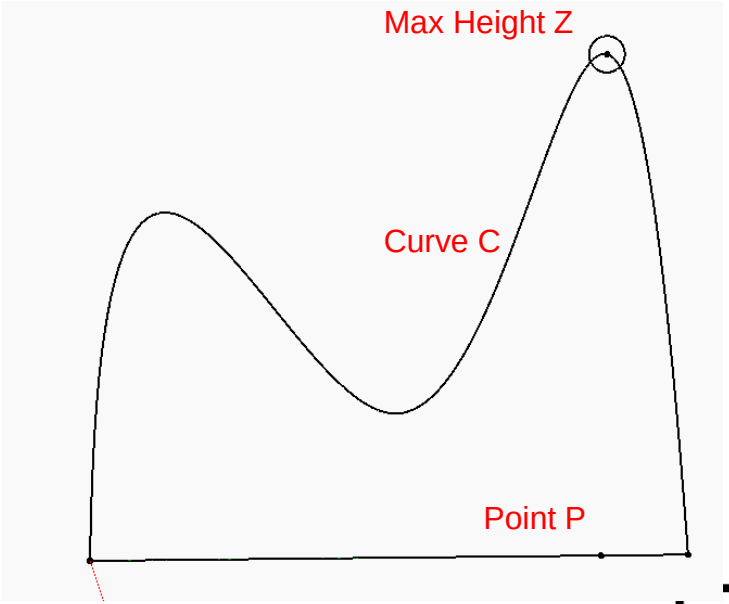
Example 2 (or 0)

2D Hill Climbing
Let’s go back a step and focus on goal-setting and how to make evaluators

Goals

What are we solving for?

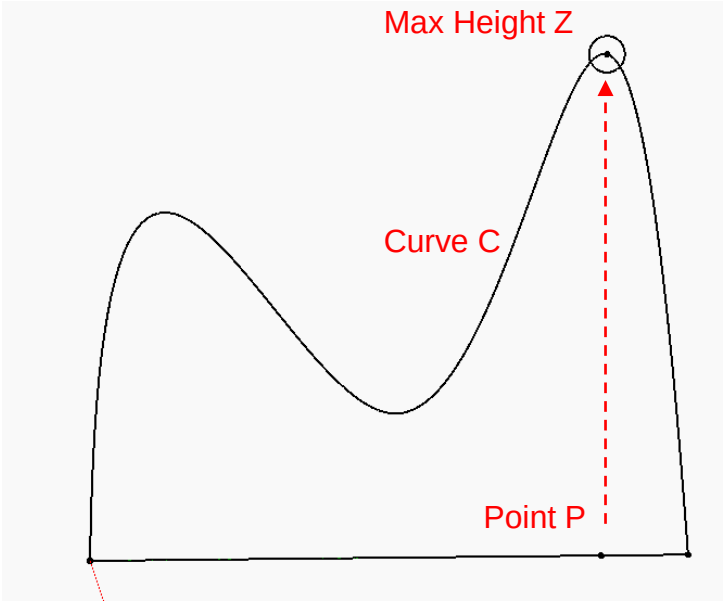
- Find Maximum Height Z for Point P
- Variable Inputs:
  1. Curve C
  2. Point P
- Goals:
  1. Maximum Height Z



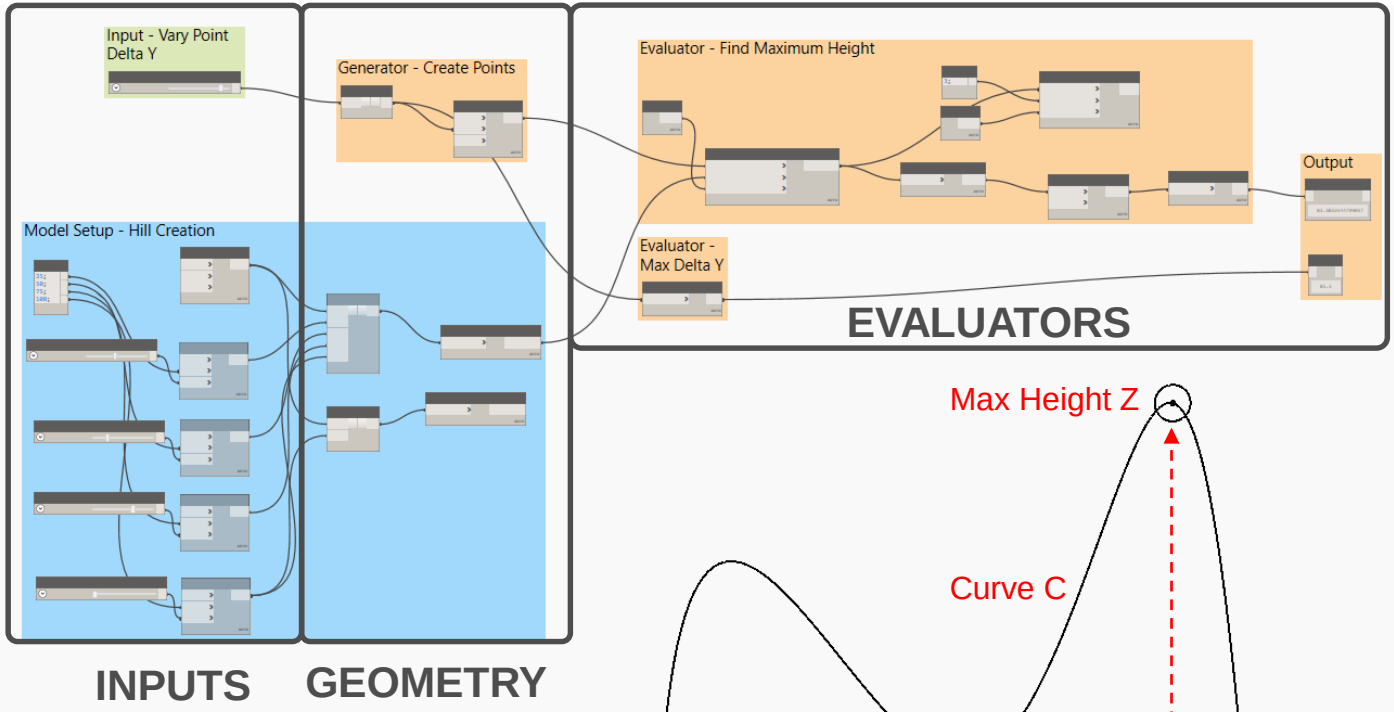
Goals

What are we solving for?

- Find Maximum Height Z for Point P
  - z = f(c,p)
  - maximize z
- Variable Inputs:
  1. Curve C
  2. Point P
- Objective Function:
  - z = f(c,p)
  - Find maximum height
  - f(c,p) =
    - Projected Point P’ = P.Project(C)
    - Height Z = P’.z
    - x = P.x



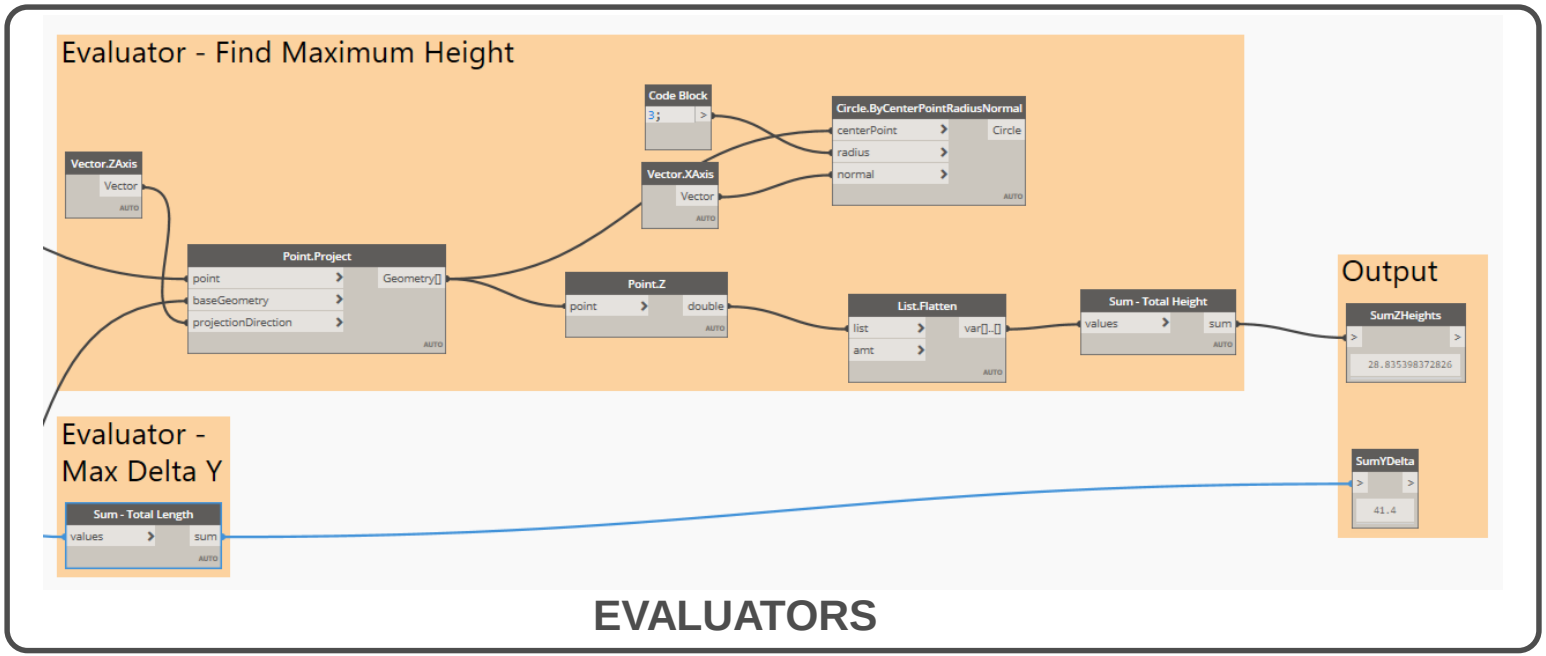
BiLT NA 2019
How To Train Your Model
2d Hill Climbing - 1 Point

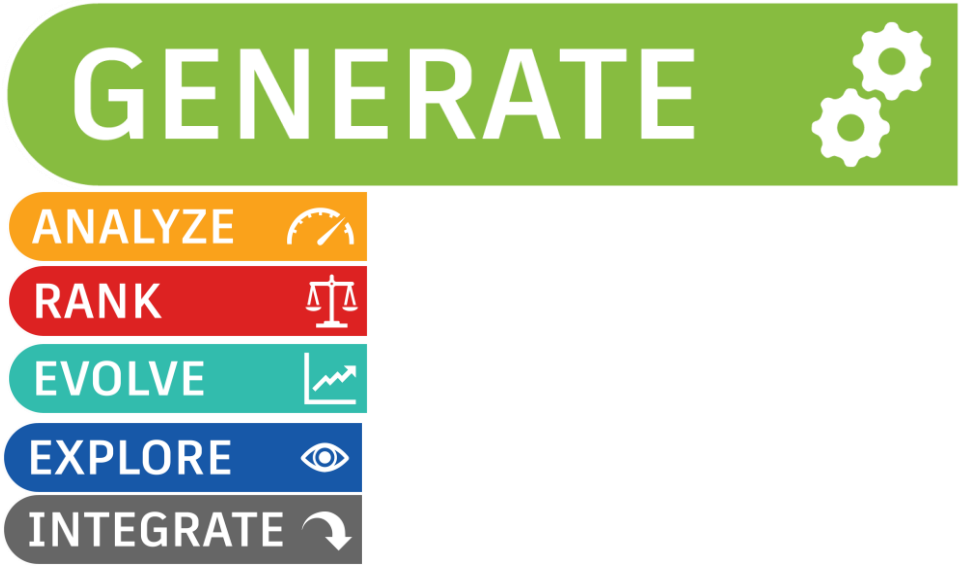


- Projected Point P' = P.Project(C)
- Height Z = P'.z
- x = P.x

ANALYZE

Measuring goals




GENERATE
ANALYZE
RANK
EVOLVE
EXPLORE
INTEGRATE

GENERATE

Refinery Methods

TYPES

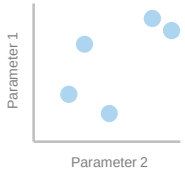
RANDOMIZE

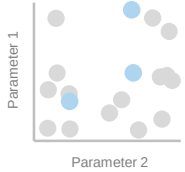
OPTIMIZE

generation 1 population of 3
generation 2 population of 3
...
generation 6 population of 3

INPUTS

DESIGN SPACE





OUTCOME

5 DESIGNS
defined by user

3 x 6 = 18 DESIGNS
user defines population size
and number of generations

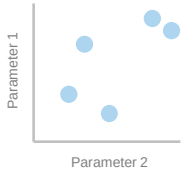
GENERATE

Refinery Methods

TYPES

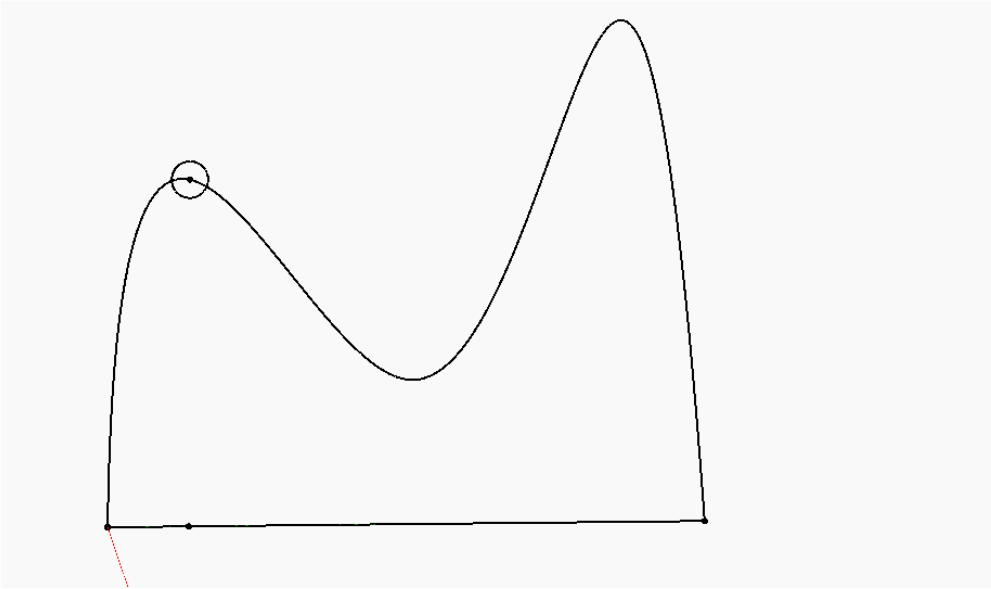
RANDOMIZE

INPUTS



5 DESIGNS

defined by user



Refinery

Studies

New Study

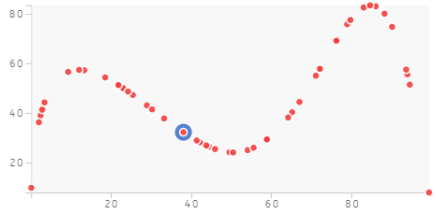
Study		
0b4ea7dd-cb84-4b37-9dd...	60/60	✓

Y-Axis
SumZHeights

X-Axis
pt0 delta y

Size
Default

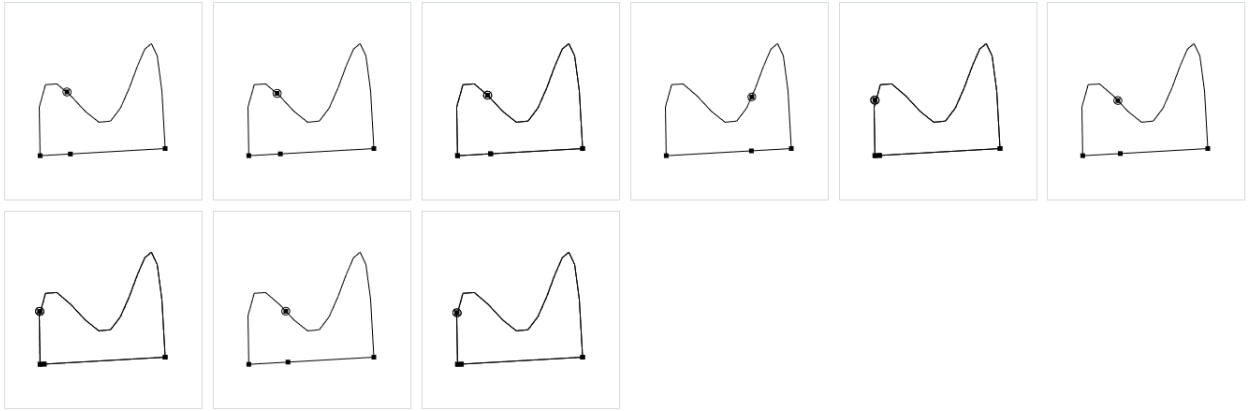
Color
Default



Filter ☐

Sort by SumZHeights

1 | 2 | 3 | 4 | 5 | 6 | 7



Refinery

Studies

New Study

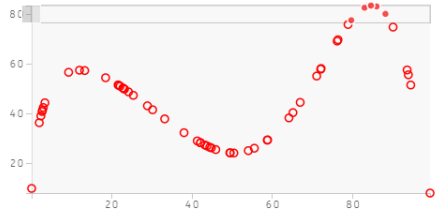
Study		
0b4ea7dd-cb84-4b37-9dd…	60/60	✓

Y-Axis
SumZHeights

X-Axis
pt0 delta y

Size
Default

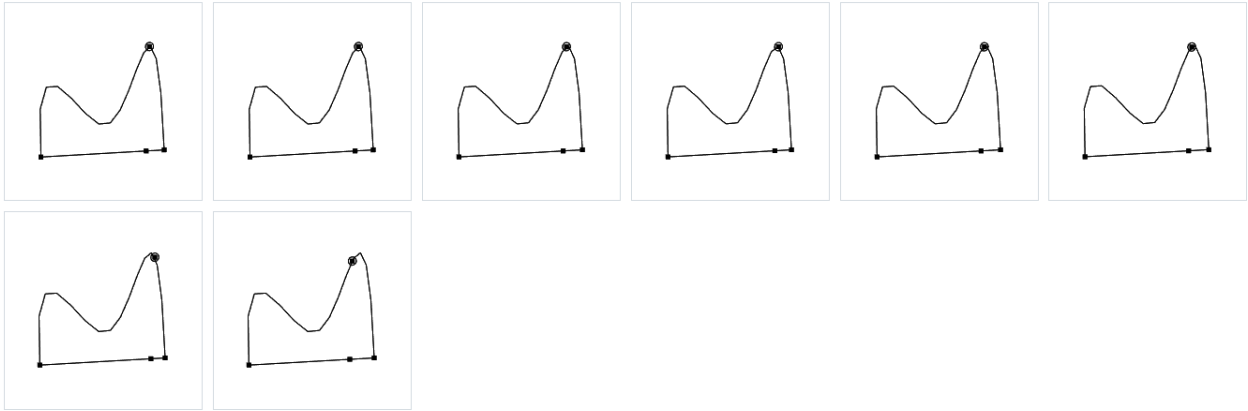
Color
Default



Filter ☑ Click and drag over axes to add filters

Sort by SumZHeights

1



Refinery

Studies

New Study

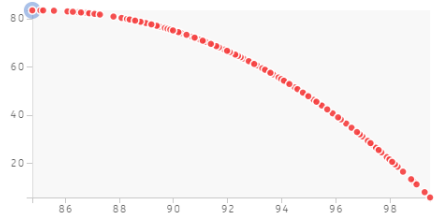
Study		
0b4ea7dd-cb84-4b37-9dd...	60/60	✓
7397463d-75bd-42cf-9a1b-...	40/40	✓

Y-Axis
SumZHeights

X-Axis
SumYDelta

Size
Default

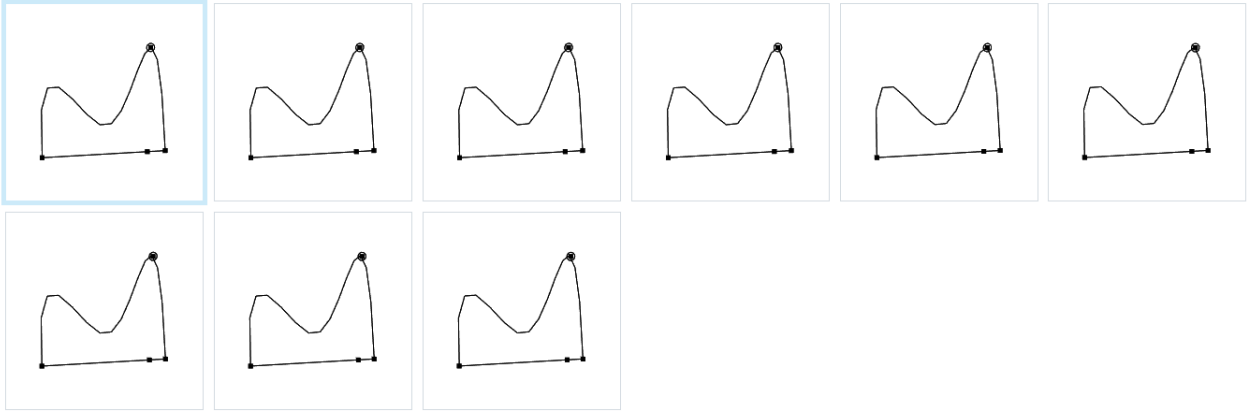
Color
Default



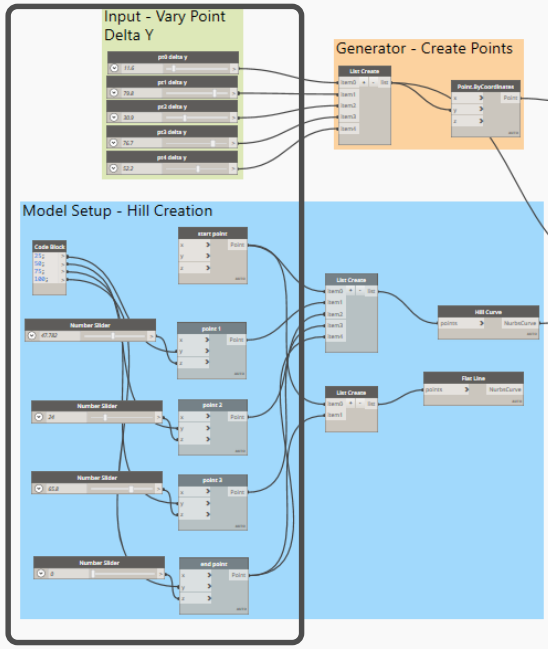
Filter

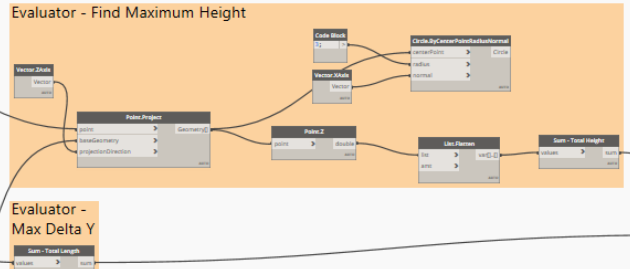
Sort by SumZHeights

1 2 3 4 5 6 7 8 9 10 11 12

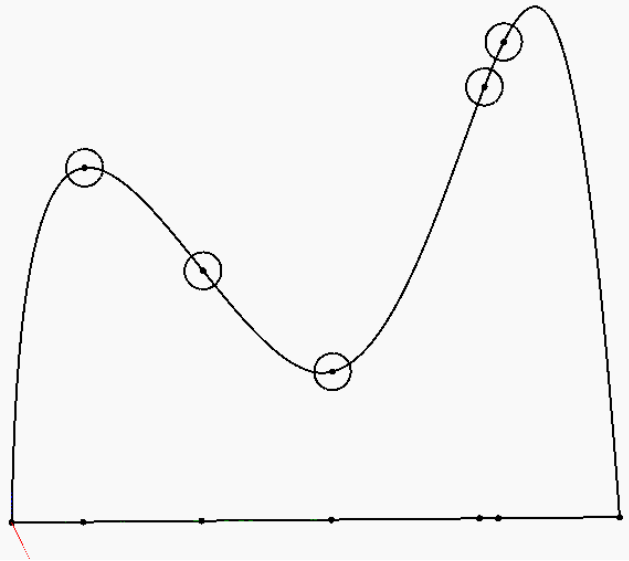


BiLT NA 2019
How To Train Your Model
2d Hill Climbing - 5 Points





MORE INPUTS!



DIGITAL BUILT WEEK EUROPE

EDINBURGH

10 – 12 OCT 2019

GENERATE

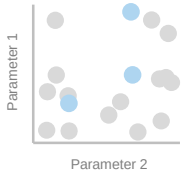
Refinery Methods

TYPES

OPTIMIZE

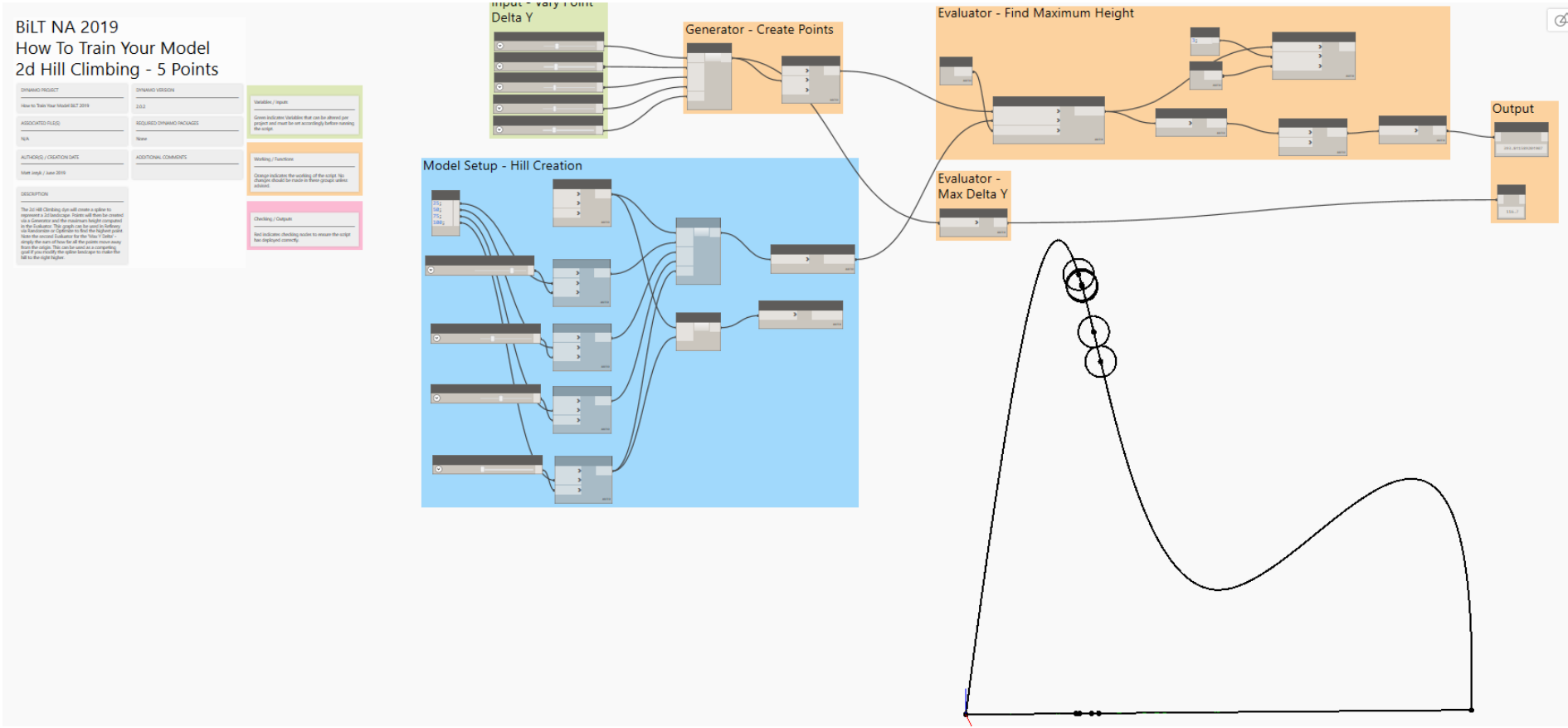
generation 1 population of 3
generation 2 population of 3
...
generation 6 population of 3

INPUTS



3 x 6 = 18 DESIGNS

user defines population size
and number of generations


BiLT NA 2019
How To Train Your Model
2d Hill Climbing - 5 Points
Delta Y
Generator - Create Points
Evaluator - Find Maximum Height
Output
Model Setup - Hill Creation
Evaluator -
Max Delta Y

GENERATE

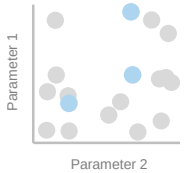
Refinery Methods

TYPES

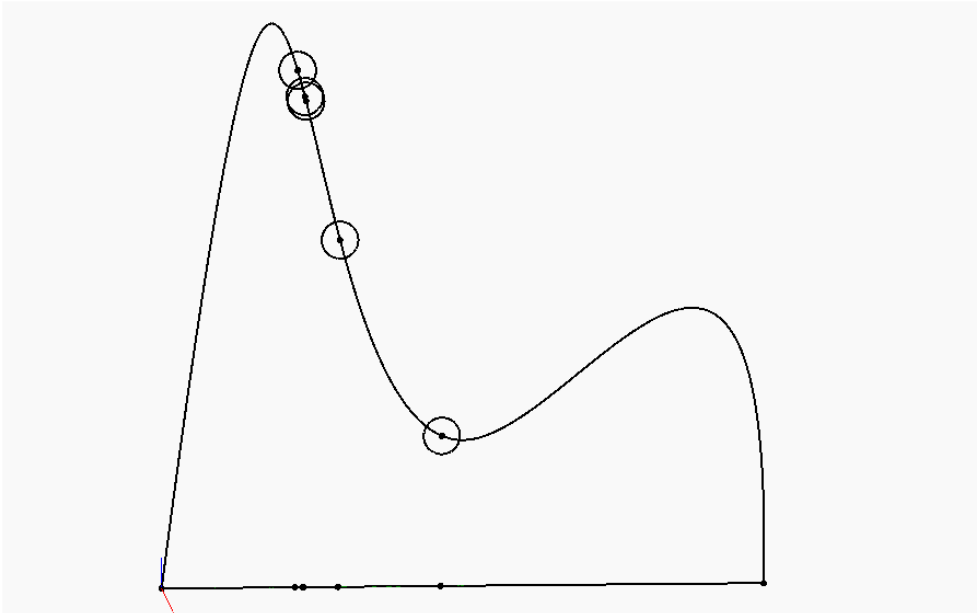
OPTIMIZE

generation 1 population of 3
generation 2 population of 3
...
generation 6 population of 3

INPUTS



3 x 6 = 18 DESIGNS
user defines population size
and number of generations



Refinery

Studies

New Study

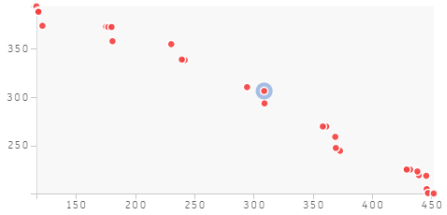
Study		
95ac2551-2c3e-4736-aefb-...	50/50	✓
236e9587-aa2a-43e4-898...	70/70	✓

Y-Axis
SumZHeights

X-Axis
SumYDelta

Size
Default

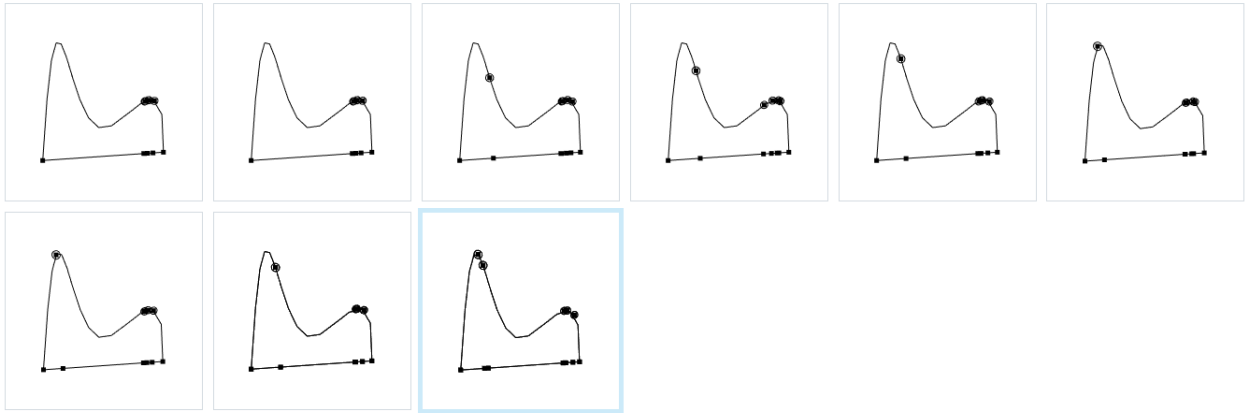
Color
Default



Filter ☐

Sort by SumYDelta

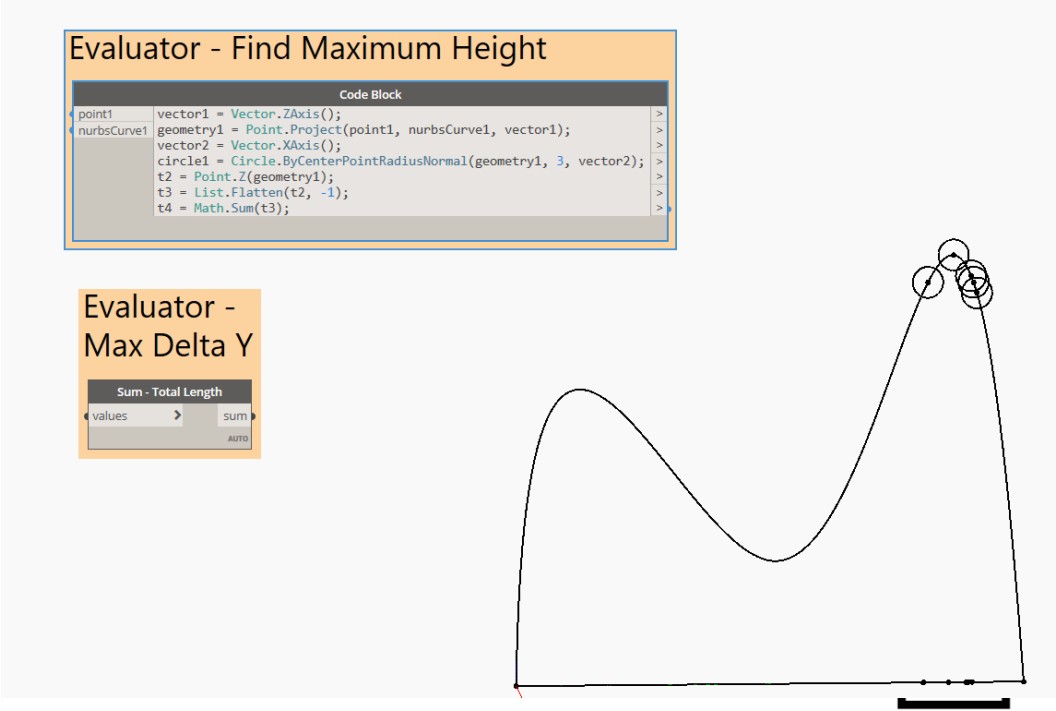
1 | 2 | 3 | 4

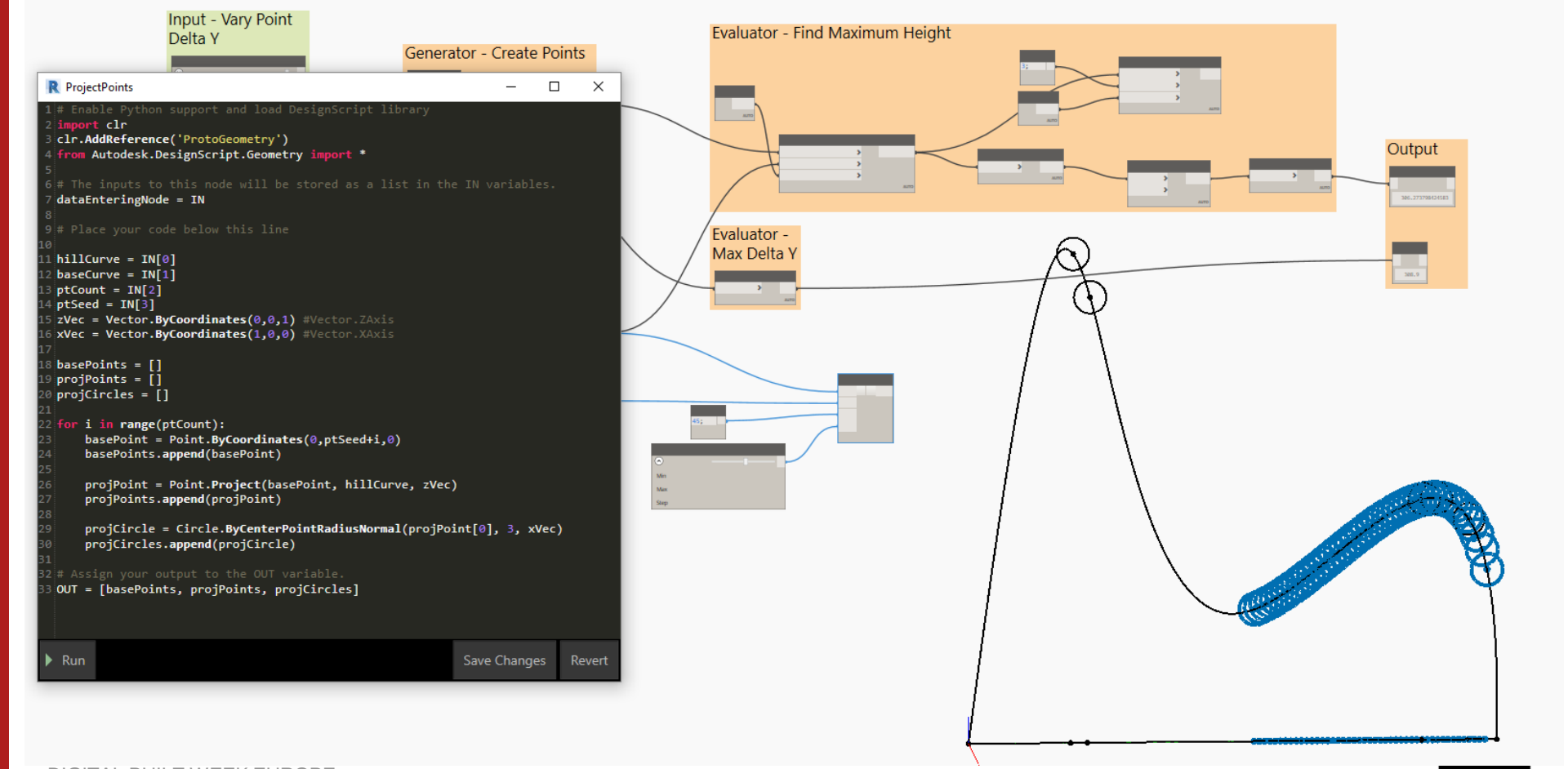


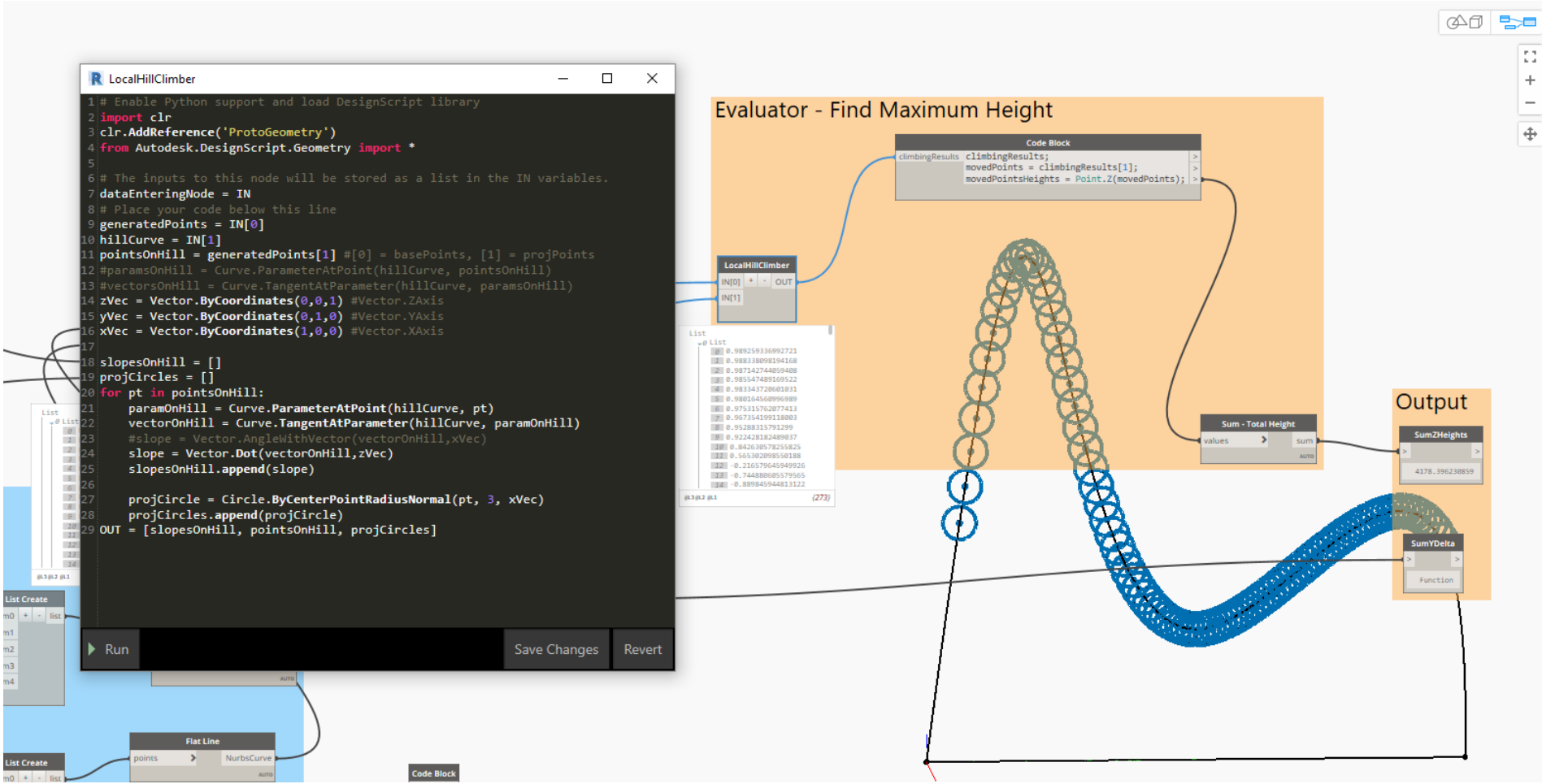
ANALYZE

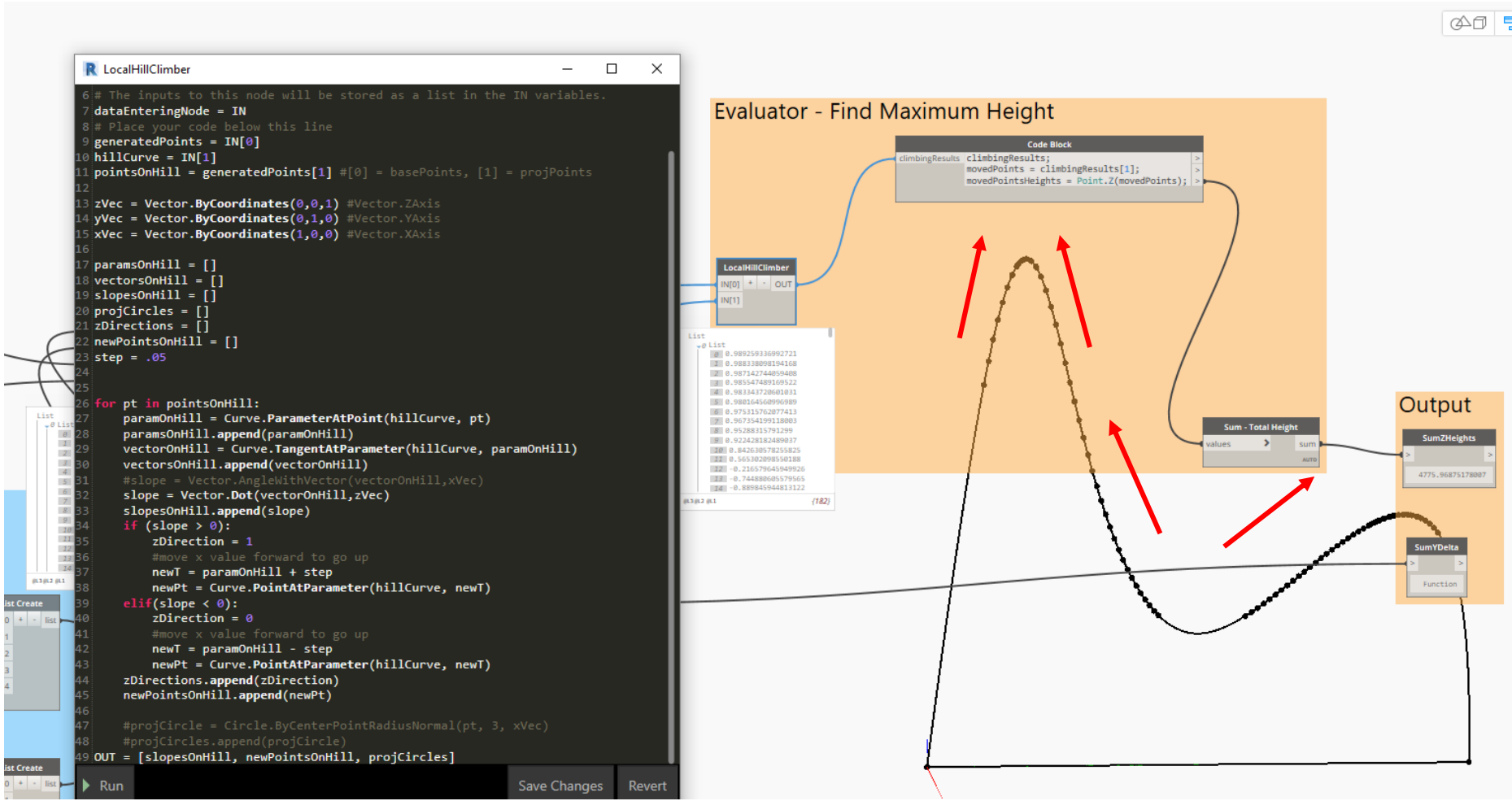
Measuring goals – DesignScript

- Find Maximum Height Z for Point P
  - z = f(c,p)
  - maximize z
- Variable Inputs:
  1. Curve C
  2. Point P
- Objective Function:
  - z = f(c,p)
  - Find maximum height
  - f(c,p) =
    - Projected Point P’ = P.Project(C)
    - Height Z = P’.z
    - x = P.x




Input - Vary Point Delta Y
Generator - Create Points
Evaluator - Find Maximum Height
Output
Evaluator - Max Delta Y
ProjectPoints
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN
# Place your code below this line
hillCurve = IN[0]
baseCurve = IN[1]
ptCount = IN[2]
ptSeed = IN[3]
zVec = Vector.ByCoordinates(0,0,1) #Vector.ZAxis
xVec = Vector.ByCoordinates(1,0,0) #Vector.XAxis
basePoints = []
projPoints = []
projCircles = []
for i in range(ptCount):
basePoint = Point.ByCoordinates(0,ptSeed+i,0)
basePoints.append(basePoint)
projPoint = Point.Project(basePoint, hillCurve, zVec)
projPoints.append(projPoint)
projCircle = Circle.ByCenterPointRadiusNormal(projPoint[0], 3, xVec)
projCircles.append(projCircle)
# Assign your output to the OUT variable.
OUT = [basePoints, projPoints, projCircles]
Run
Save Changes
Revert


LocalHillClimber
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN
# Place your code below this line
generatedPoints = IN[0]
hillCurve = IN[1]
pointsOnHill = generatedPoints[1] #[0] = basePoints, [1] = projPoints
#paramsOnHill = Curve.ParameterAtPoint(hillCurve, pointsOnHill)
#vectorsOnHill = Curve.TangentAtParameter(hillCurve, paramsOnHill)
zVec = Vector.ByCoordinates(0,0,1) #Vector.ZAxis
yVec = Vector.ByCoordinates(0,1,0) #Vector.YAxis
xVec = Vector.ByCoordinates(1,0,0) #Vector.XAxis
slopesOnHill = []
projCircles = []
for pt in pointsOnHill:
paramOnHill = Curve.ParameterAtPoint(hillCurve, pt)
vectorOnHill = Curve.TangentAtParameter(hillCurve, paramOnHill)
#slope = Vector.AngleWithVector(vectorOnHill,xVec)
slope = Vector.Dot(vectorOnHill,zVec)
slopesOnHill.append(slope)
projCircle = Circle.ByCenterPointRadiusNormal(pt, 3, xVec)
projCircles.append(projCircle)
OUT = [slopesOnHill, pointsOnHill, projCircles]
Run
Save Changes
Revert
Evaluator - Find Maximum Height
Code Block
climbingResults;
movedPoints = climbingResults[1];
movedPointsHeights = Point.Z(movedPoints);
LocalHillClimber
IN[0]
IN[1]
OUT
Sum - Total Height
values
sum
Output
SumZHeights
4178.396238859
SumYDelta
Function
List Create
Flat Line
points
NurbsCurve
Code Block


LocalHillClimber
# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN
# Place your code below this line
generatedPoints = IN[0]
hillCurve = IN[1]
pointsOnHill = generatedPoints[1] #[0] = basePoints, [1] = projPoints
zVec = Vector.ByCoordinates(0,0,1) #Vector.ZAxis
yVec = Vector.ByCoordinates(0,1,0) #Vector.YAxis
xVec = Vector.ByCoordinates(1,0,0) #Vector.XAxis
paramsOnHill = []
vectorsOnHill = []
slopesOnHill = []
projCircles = []
zDirections = []
newPointsOnHill = []
step = .05
for pt in pointsOnHill:
paramOnHill = Curve.ParameterAtPoint(hillCurve, pt)
paramsOnHill.append(paramOnHill)
vectorOnHill = Curve.TangentAtParameter(hillCurve, paramOnHill)
vectorsOnHill.append(vectorOnHill)
#slope = Vector.AngleWithVector(vectorOnHill,xVec)
slope = Vector.Dot(vectorOnHill,zVec)
slopesOnHill.append(slope)
if (slope > 0):
zDirection = 1
#move x value forward to go up
newT = paramOnHill + step
newPt = Curve.PointAtParameter(hillCurve, newT)
elif(slope < 0):
zDirection = 0
#move x value forward to go up
newT = paramOnHill - step
newPt = Curve.PointAtParameter(hillCurve, newT)
zDirections.append(zDirection)
newPointsOnHill.append(newPt)
#projCircle = Circle.ByCenterPointRadiusNormal(pt, 3, xVec)
#projCircles.append(projCircle)
OUT = [slopesOnHill, newPointsOnHill, projCircles]
Run
Save Changes
Revert
Evaluator - Find Maximum Height
Code Block
climbingResults
climbingResults;
movedPoints = climbingResults[1];
movedPointsHeights = Point.Z(movedPoints);
LocalHillClimber
IN[0]
IN[1]
OUT
Sum - Total Height
values
sum
Output
SumZHeights
SumYDelta
Function

PointGenerator

```python
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

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

# Place your code below this line

#PointGenerator
#How to Train Your Model BiLT NA 2019
#Jezyk

hillCurve = IN[0]
baseCurve = IN[1]
ptCount = IN[2]
ptOffset = IN[3]
zVec = Vector.ZAxis() #Vector.ByCoordinates(0,0,1)
xVec = Vector.XAxis() #Vector.ByCoordinates(1,0,0)

startY = baseCurve.StartPoint.Y
endY = baseCurve.EndPoint.Y
baseDist = endY - ptOffset - startY - ptOffset
segDist = baseDist/ptCount

basePoints = []
projPoints = []

for i in range(ptCount):
    basePoint = Point.ByCoordinates(0,segDist*i,0)
    basePoints.append(basePoint)

    projPoint = Point.Project(basePoint, hillCurve, zVec)
    projPoints.append(projPoint[0])

# Assign your output to the OUT variable.
OUT = [basePoints, projPoints, baseDist, segDist]
```

Run | Save Changes | Revert

Points

PointGenerator
IN[0] + - OUT
IN[1]
IN[2]
IN[3]

Step Size
0.13

Max Integer
0

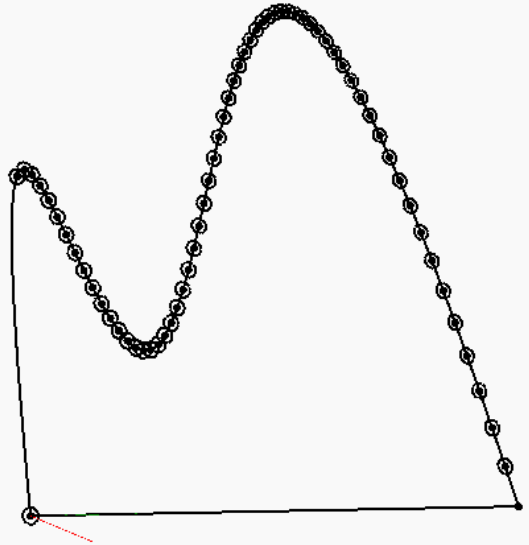
LocalHillClimber
IN[0] + - OUT
IN[1]
IN[2]
IN[3]

urbsCurve
AUTO

List Create
item0 + - list
item1

points
NurbsCurve
AUTO

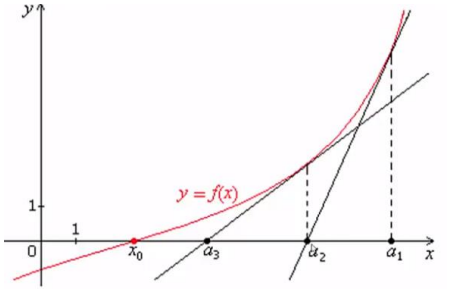
point 2
x
Point
y



Local Solver – More Advanced Evaluator

Global Solver vs Local Solver

- Python loop to move points closer to local peaks
  - Position — y = f(x) — *position function we have been using*
  - Tangent — dy = f'(x)dx — *f'(x) is the derivative of x*
  - Slope/Velocity — v = dy/dx — *slope of tangent line represents 'velocity'*
  - Acceleration — dv = f''(x) — change in slope of tan line is 'acceleration'



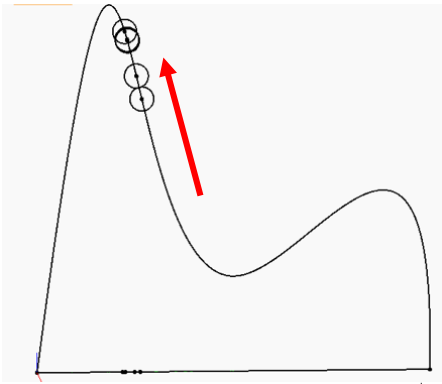
$$dy = f'(x)\,dx$$

$$\Delta y \approx dy$$

$$\Delta f = f(\mathbf{x} + \Delta\mathbf{x}) - f(\mathbf{x}).$$

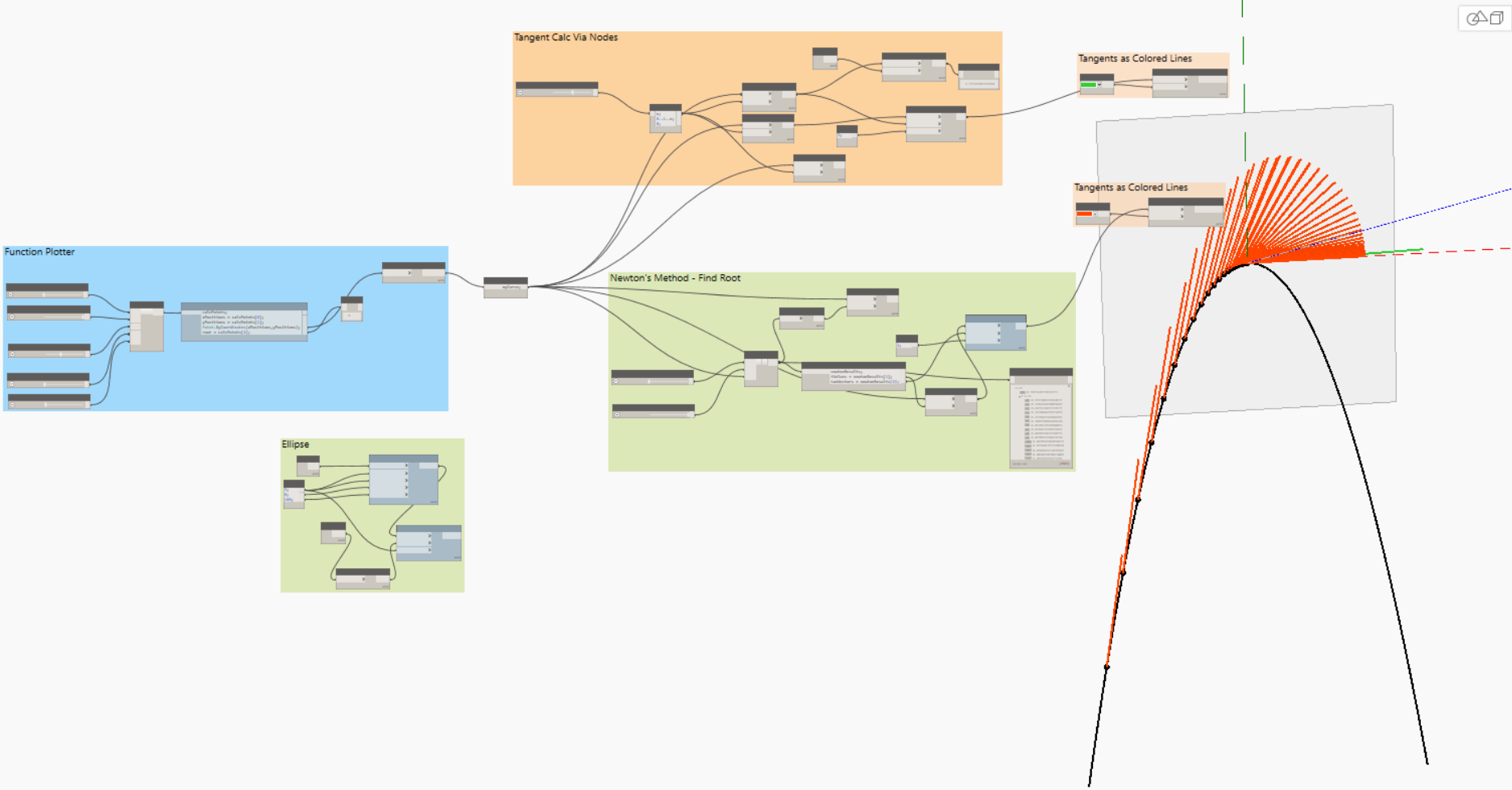
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

f'(x) = (f(x + h/2) - f(x - h/2))
h



Newton’s Method in Python

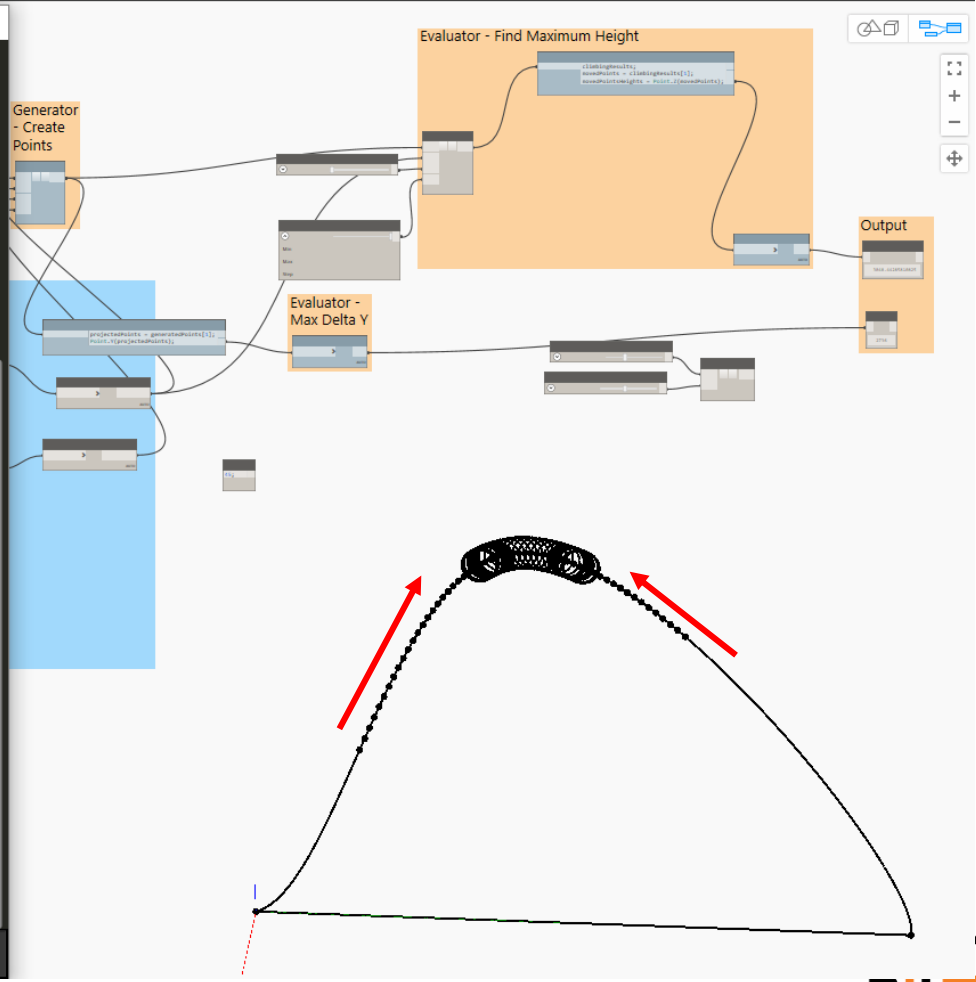
```
import sys

def f(x):
        return x**3 + x - 1

def f_prime(x):
        return 3*x**2 + 1

def newt(x,n):
        for i in range(n):
                if f_prime(x) == 0:
                        return x
                x = x - f(x)/f_prime(x)
        return x

def main(argv):
        if (len(sys.argv) != 3):
                sys.exit('Usage: newtons_method.py <x> <n>')

        print 'The root is: ',
        print newt(float(sys.argv[1]),int(sys.argv[2]))

if __name__ == "__main__":
        main(sys.argv[1:])
```


Tangent Calc Via Nodes
Tangents as Colored Lines
Tangents as Colored Lines
Function Plotter
Newton's Method - Find Root
Ellipse

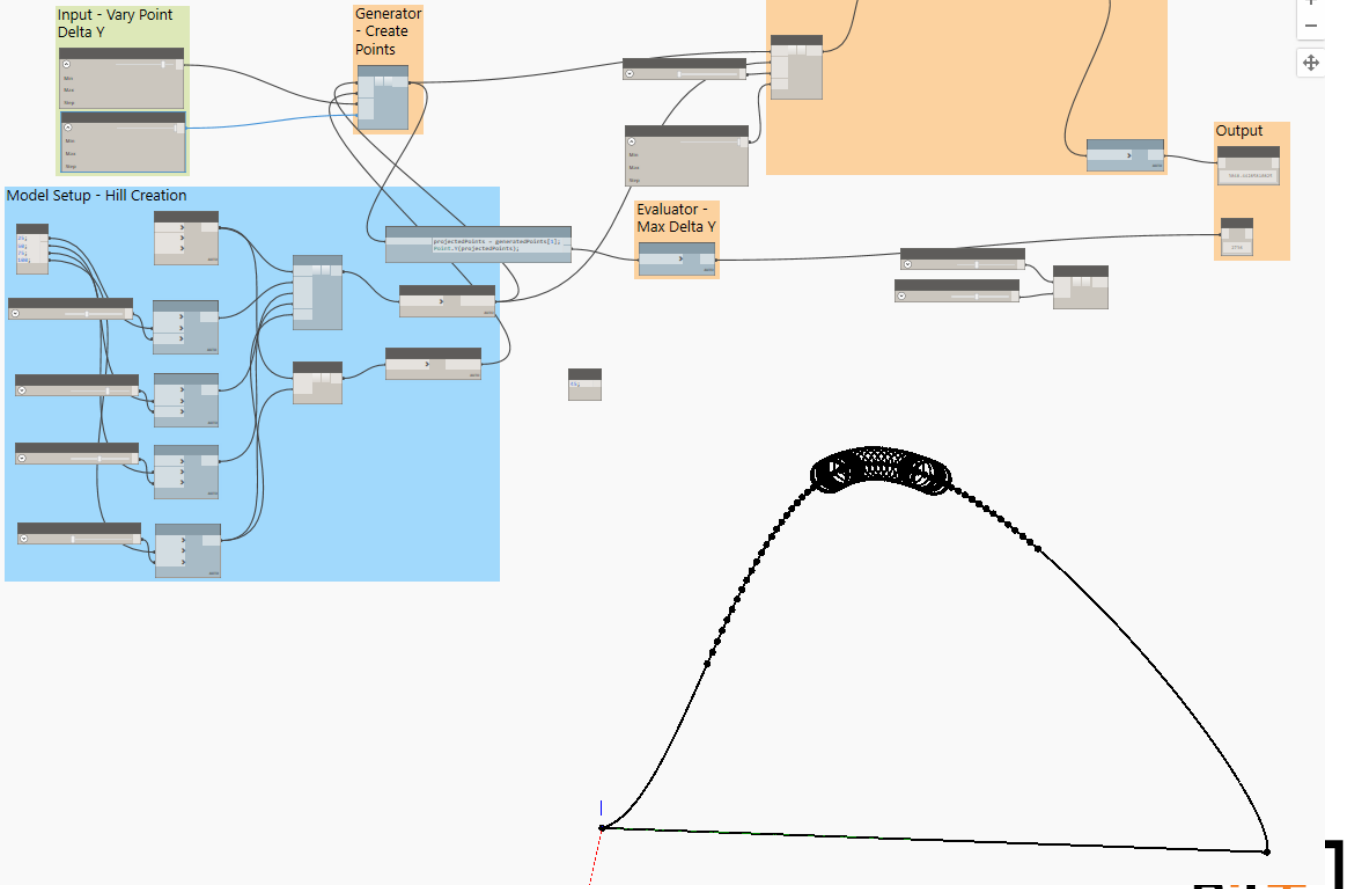
LocalHillClimber

```python
#Newton's Method

def calcHeight(tParam, hillCurve):
    calcPt = Curve.PointAtParameter(hillCurve, tParam)
    return calcPt.Z

def calcMaxHeight(hillCurve):
    heights = []
    for i in range(0, 100, 1):
        calcPt = Curve.PointAtParameter(hillCurve, i/100)
        heights.append(calcPt.Z)
    return max(heights)

def calcTangent(tParam, hillCurve):
    return Curve.TangentAtParameter(hillCurve, tParam)

def calcSlope(tangent):
    zVec = Vector.ByCoordinates(0,0,1)
    return Vector.Dot(tangent, zVec)

def newtCurve(tParam, n, hillCurve, flip, step):
    if flip:
        step = -step
    else:
        step = step
    for i in range(n):
        #SLOPE
        if abs(calcSlope(calcTangent(tParam, hillCurve))) < .5:
            return tParam
        tParam = tParam - (calcSlope(calcTangent(tParam + step, hillCurve)) - (calcSlope
        (calcTangent(tParam - step, hillCurve))))/calcSlope(calcTangent(tParam, hillCurve))

    return tParam

for pt in pointsOnHill:
    paramOnHill = Curve.ParameterAtPoint(hillCurve, pt)
    paramsOnHill.append(paramOnHill)
    vectorOnHill = Curve.TangentAtParameter(hillCurve, paramOnHill)
    vectorsOnHill.append(vectorOnHill)
    #slope = Vector.AngleWithVector(vectorOnHill,xVec)
    slope = Vector.Dot(vectorOnHill,zVec)
    slopesOnHill.append(slope)
    n = int(maxIterations)
    newT = newtCurve(paramOnHill, n, hillCurve, 0, step)
    pt = Curve.PointAtParameter(hillCurve, newT)

    projCircle = Circle.ByCenterPointRadiusNormal(pt, 3, xVec)
    projCircles.append(projCircle)
OUT = [slopesOnHill, pointsOnHill, projCircles] #
```

Run | Save Changes | Revert



BiLT NA 2019
How To Train Your Model
2d Hill Climbing - 5 Points



Refinery

Studies

New Study

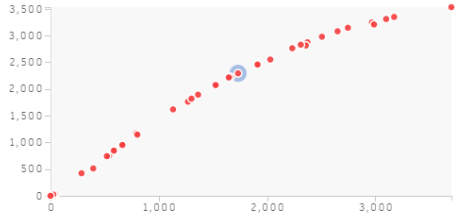
Study		
7add6c74-6bbc-41af-9a9d-...	10/10	✓
9e451c6e-d6e5-4431-86d1...	40/40	✓
46988a14-9114-4302-996f...	40/40	✓

Y-Axis
SumZHeights

X-Axis
SumYDelta

Size
Default

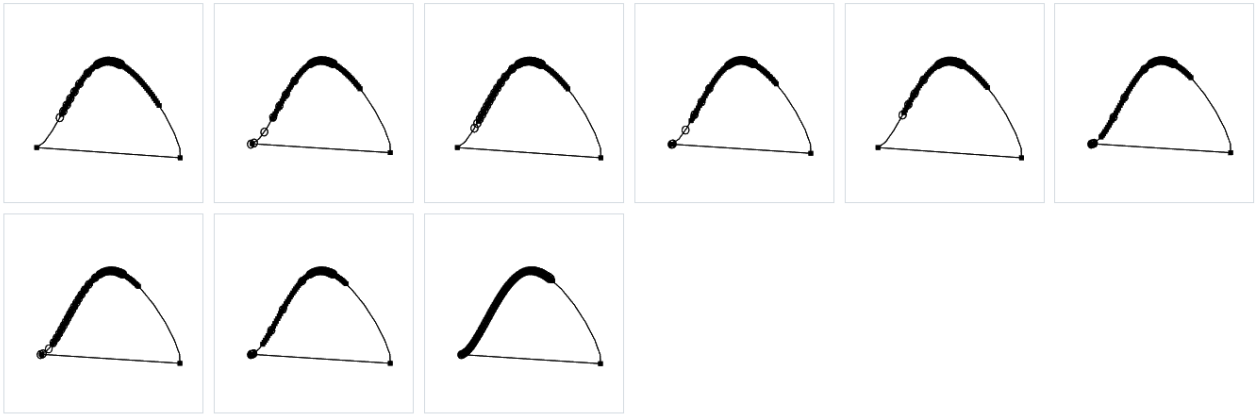
Color
Default



Filter ☐

Sort by SumZHeights

1 | 2 | 3 | 4 | 5



Example 3

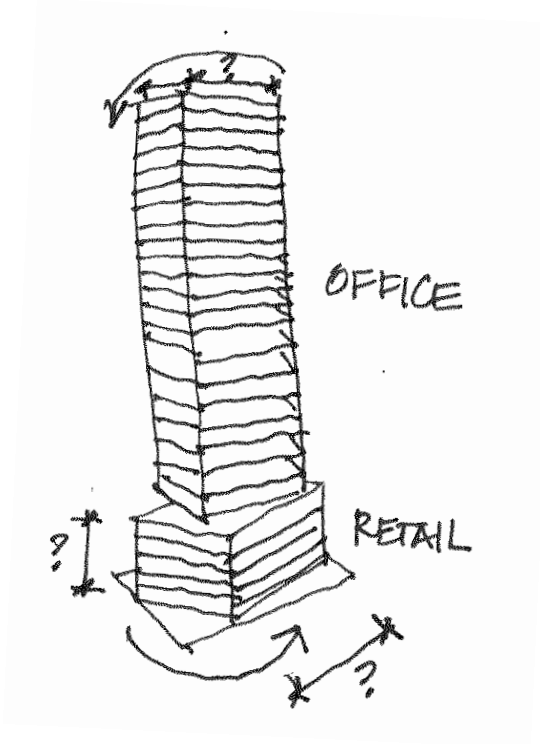
Building Massing

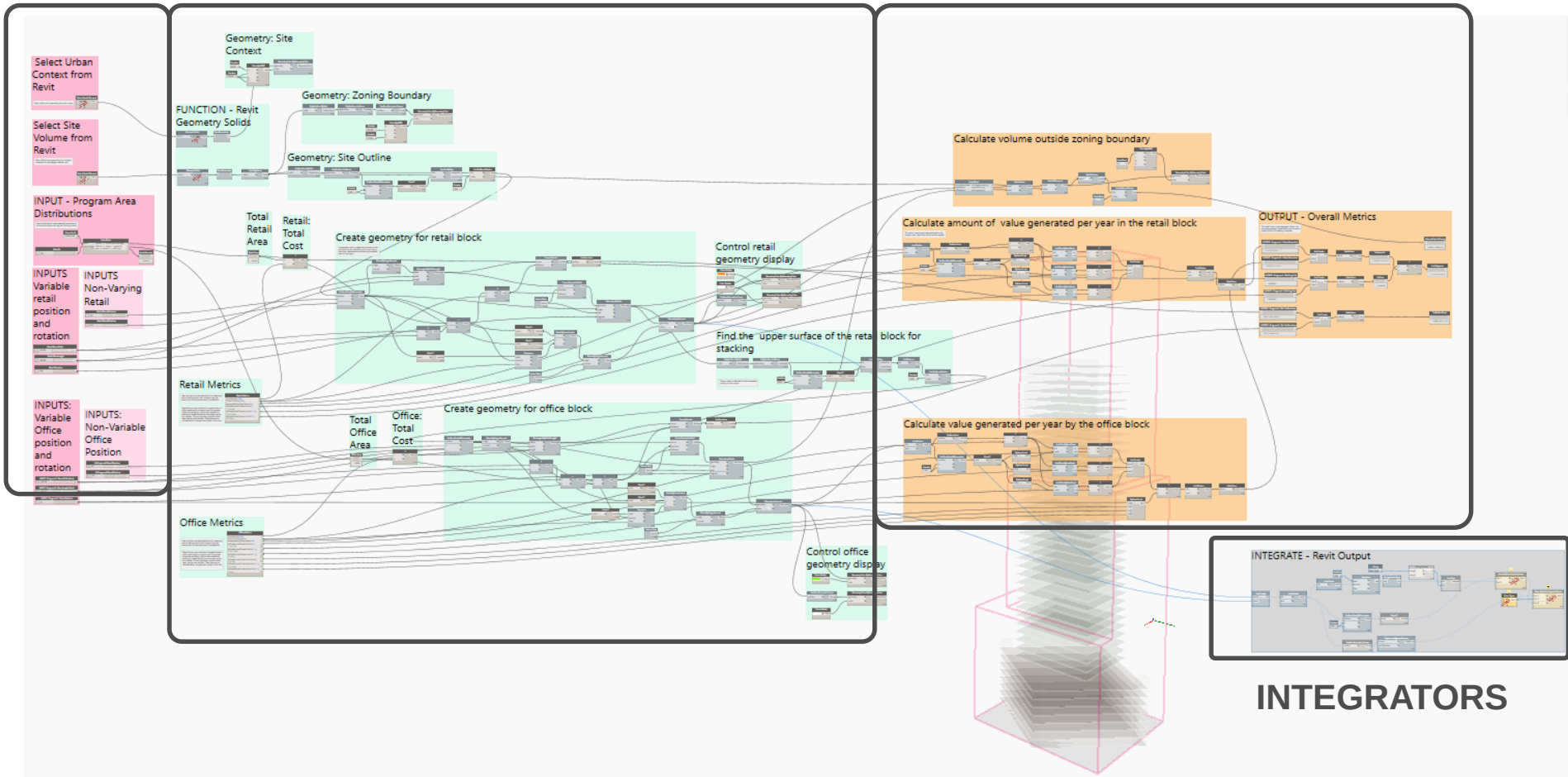
Goals

What are we solving for?

- Retail and office distribution and configuration for a building on an urban site.

- Variable Inputs:
  1. Ratio retail to office
  2. Program block size
  3. Program block rotation
- Goals:
  1. Minimize zoning envelop overlap
  2. Minimize cost
  3. Maximize total value per year




INPUTS
GEOMETRY
EVALUATORS
Select Urban Context from Revit
Select Site Volume from Revit
INPUT - Program Area Distributions
INPUTS Variable retail position and rotation
INPUTS Non-Varying Retail
INPUTS: Variable Office position and rotation
INPUTS: Non-Variable Office Position
Geometry: Site Context
FUNCTION - Revit Geometry Solids
Geometry: Zoning Boundary
Geometry: Site Outline
Total Retail Area
Retail: Total Cost
Create geometry for retail block
Control retail geometry display
Find the upper surface of the retail block for stacking
Retail Metrics
Total Office Area
Office: Total Cost
Create geometry for office block
Office Metrics
Control office geometry display
Calculate volume outside zoning boundary
Calculate amount of value generated per year in the retail block
OUTPUT - Overall Metrics
Calculate value generated per year by the office block
INTEGRATE - Revit Output
INTEGRATORS

Dynamo

File Edit View Packages Settings Help

BuildingMasser-2programsOTx.dyn*

Library

INPUT - Program Area Distributions

Users can set a gross square area and choose how to distribute the program among the 2 building masses

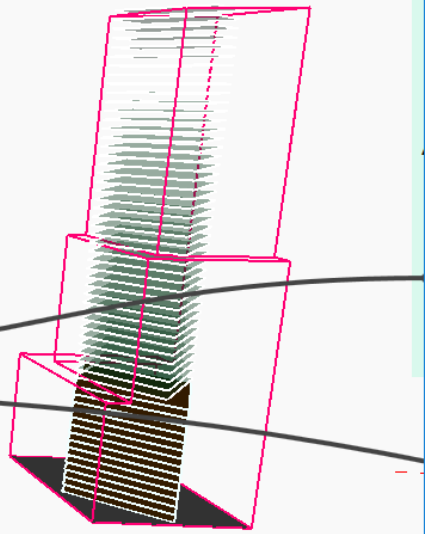
Gross Sq. Ft.
1200000.000 >

Retail%
0.3 >

Code Block

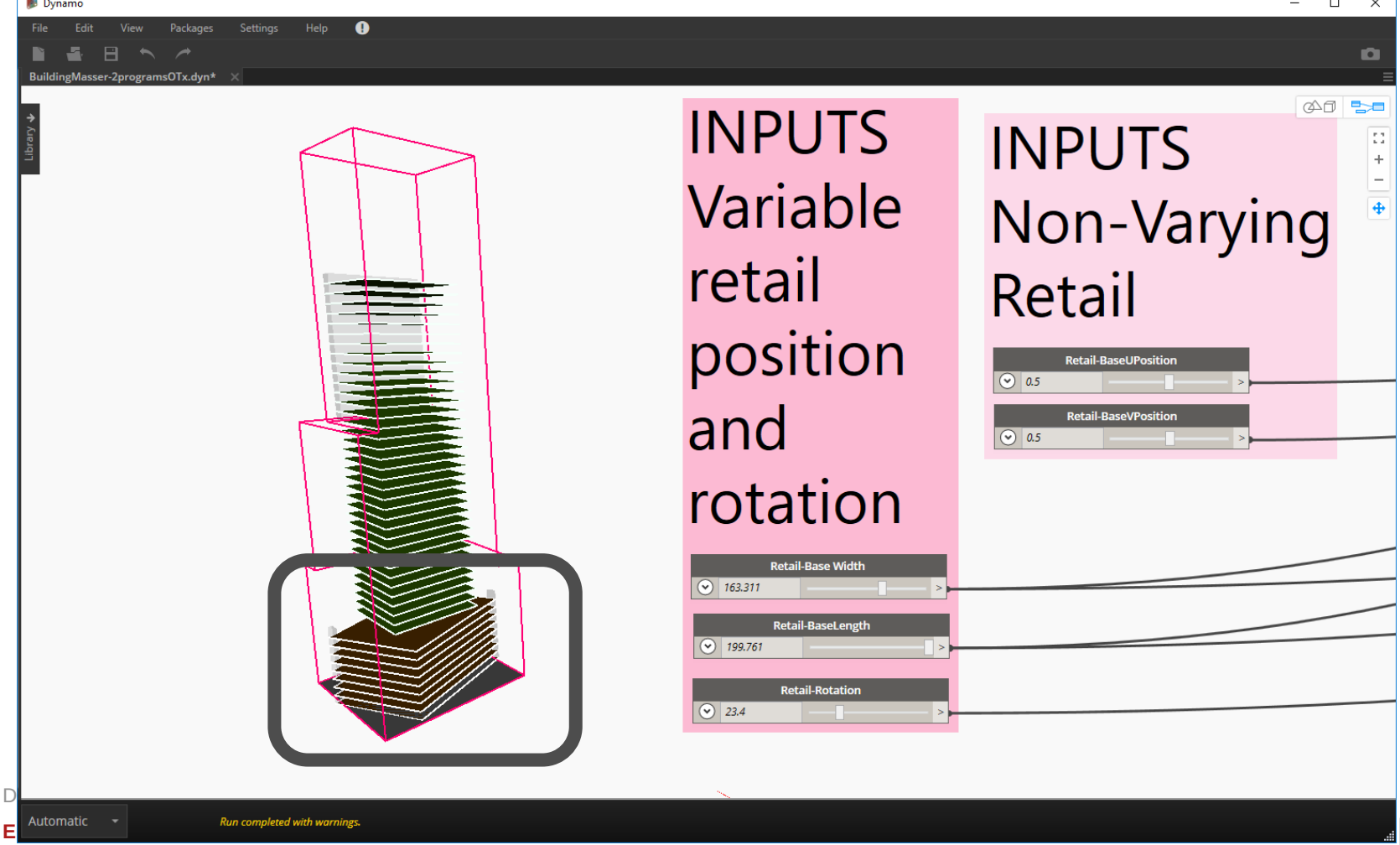
```
area
percentage
retail = area*percentage;
office = area - retail;
sum = retail + office;
```

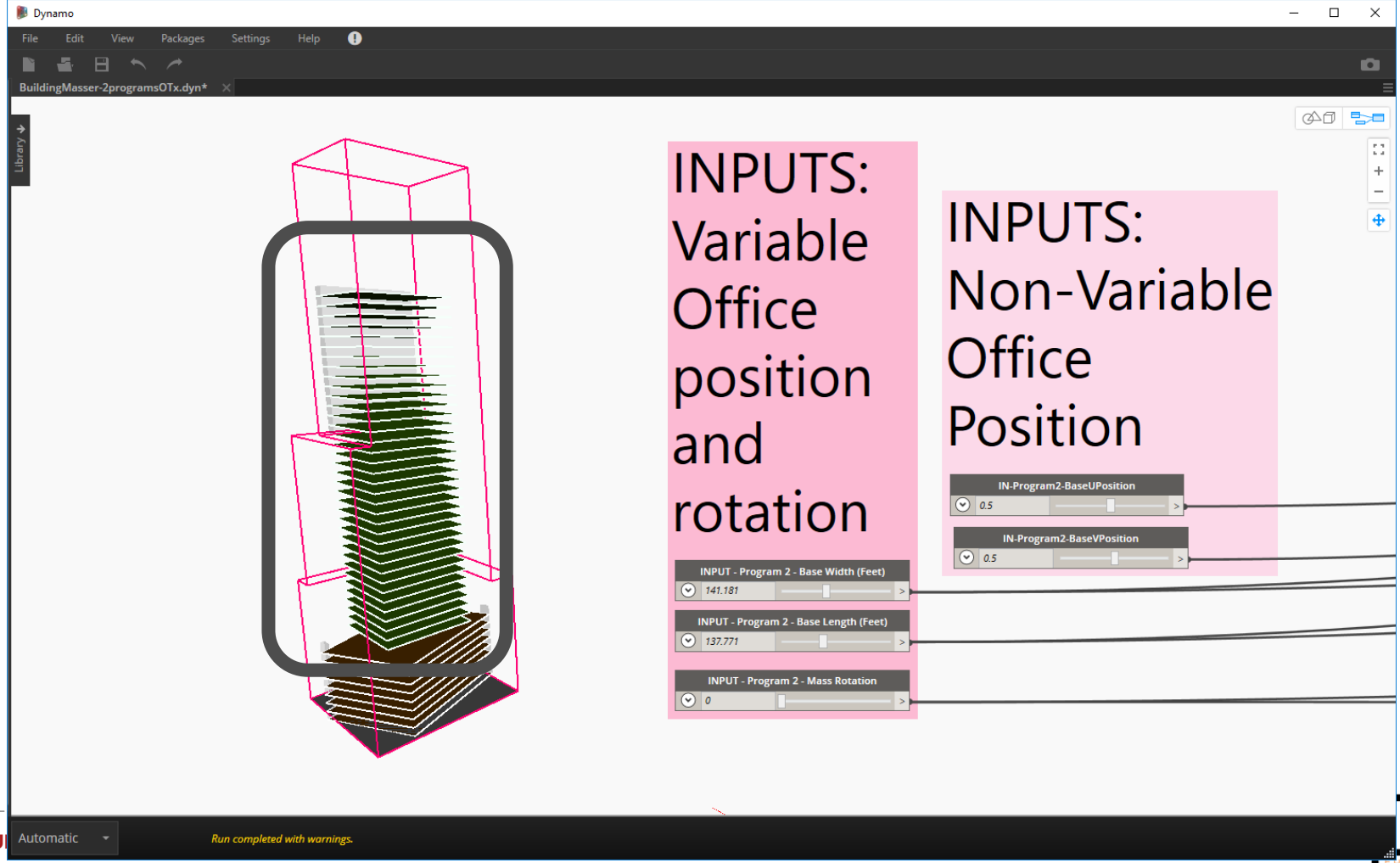
check the sum
1200000

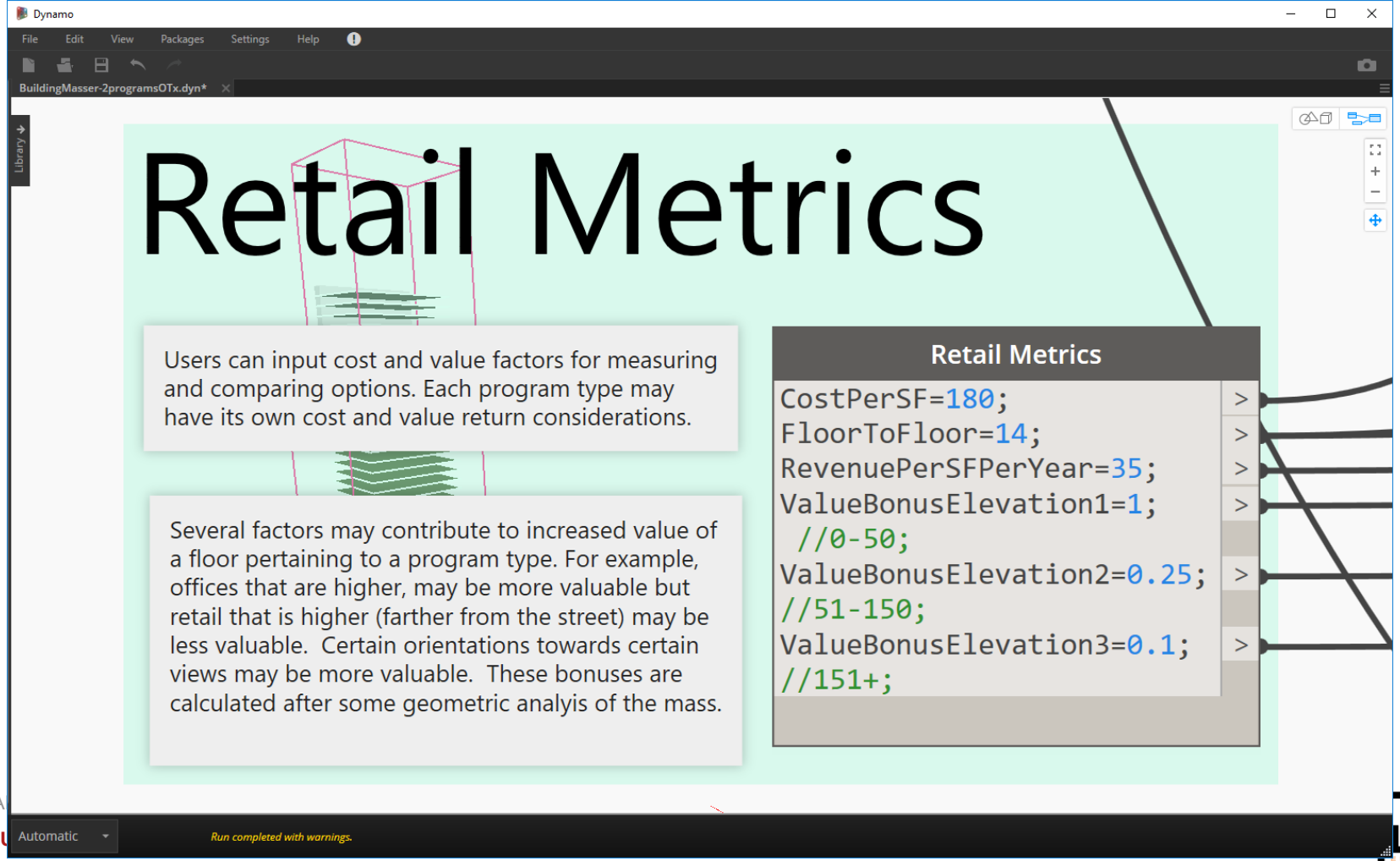


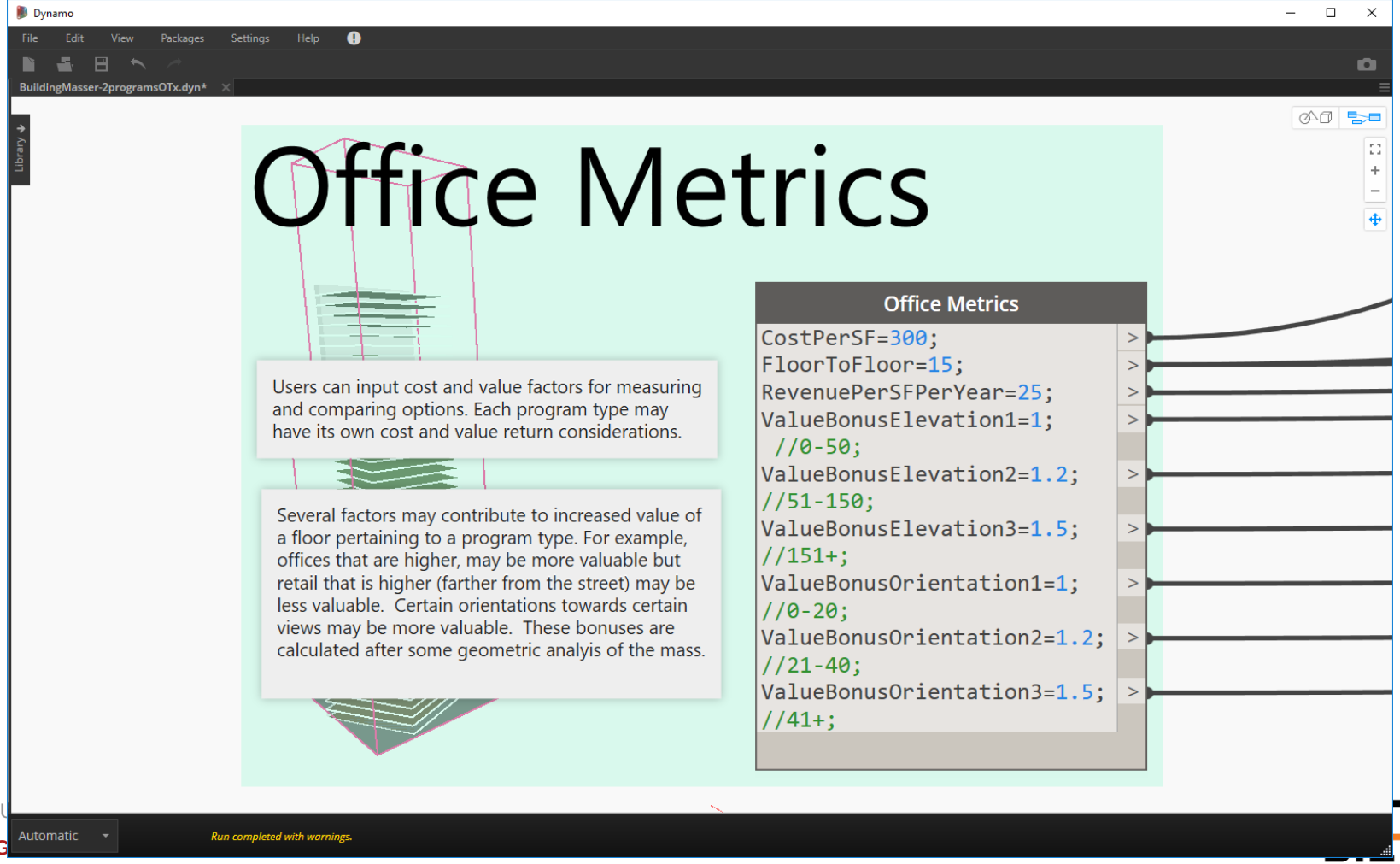
Automatic

Run completed with warnings.


Dynamo
File
Edit
View
Packages
Settings
Help
BuildingMasser-2programsOTx.dyn*
Library
INPUTS
Variable
retail
position
and
rotation
INPUTS
Non-Varying
Retail
Retail-BaseUPosition
0.5
Retail-BaseVPosition
0.5
Retail-Base Width
163.311
Retail-BaseLength
199.761
Retail-Rotation
23.4
Automatic
Run completed with warnings.


Dynamo
File
Edit
View
Packages
Settings
Help
BuildingMasser-2programsOTx.dyn*
Library
INPUTS: Variable Office position and rotation
INPUTS: Non-Variable Office Position
IN-Program2-BaseUPosition
0.5
IN-Program2-BaseVPosition
0.5
INPUT - Program 2 - Base Width (Feet)
141.181
INPUT - Program 2 - Base Length (Feet)
137.771
INPUT - Program 2 - Mass Rotation
0
Automatic
Run completed with warnings.

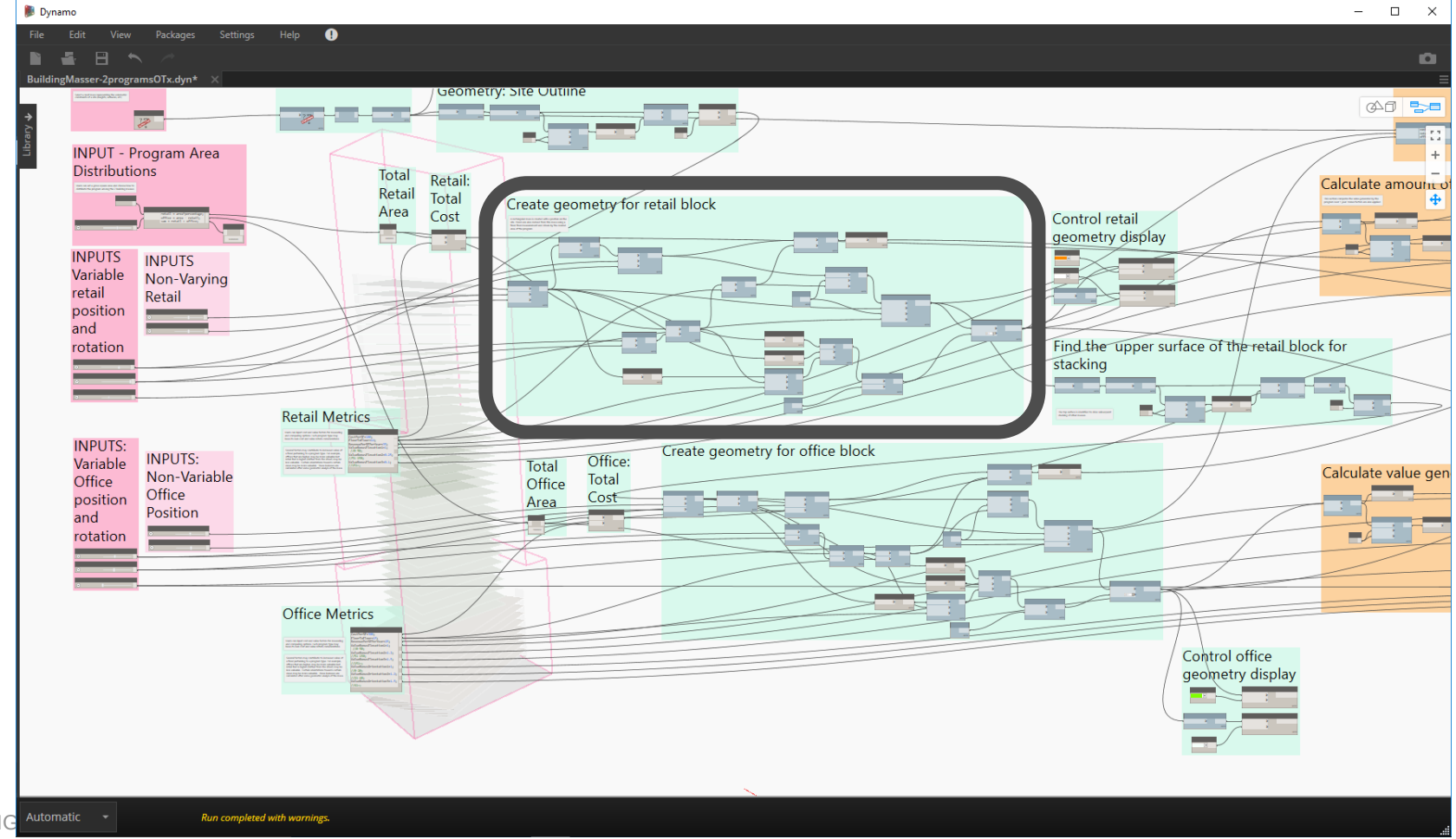

Dynamo
File
Edit
View
Packages
Settings
Help
BuildingMasser-2programsOTx.dyn*
Library
Retail Metrics
Users can input cost and value factors for measuring and comparing options. Each program type may have its own cost and value return considerations.
Several factors may contribute to increased value of a floor pertaining to a program type. For example, offices that are higher, may be more valuable but retail that is higher (farther from the street) may be less valuable. Certain orientations towards certain views may be more valuable. These bonuses are calculated after some geometric analyis of the mass.
Retail Metrics
CostPerSF=180;
FloorToFloor=14;
RevenuePerSFPerYear=35;
ValueBonusElevation1=1;
//0-50;
ValueBonusElevation2=0.25;
//51-150;
ValueBonusElevation3=0.1;
//151+;
Automatic
Run completed with warnings.

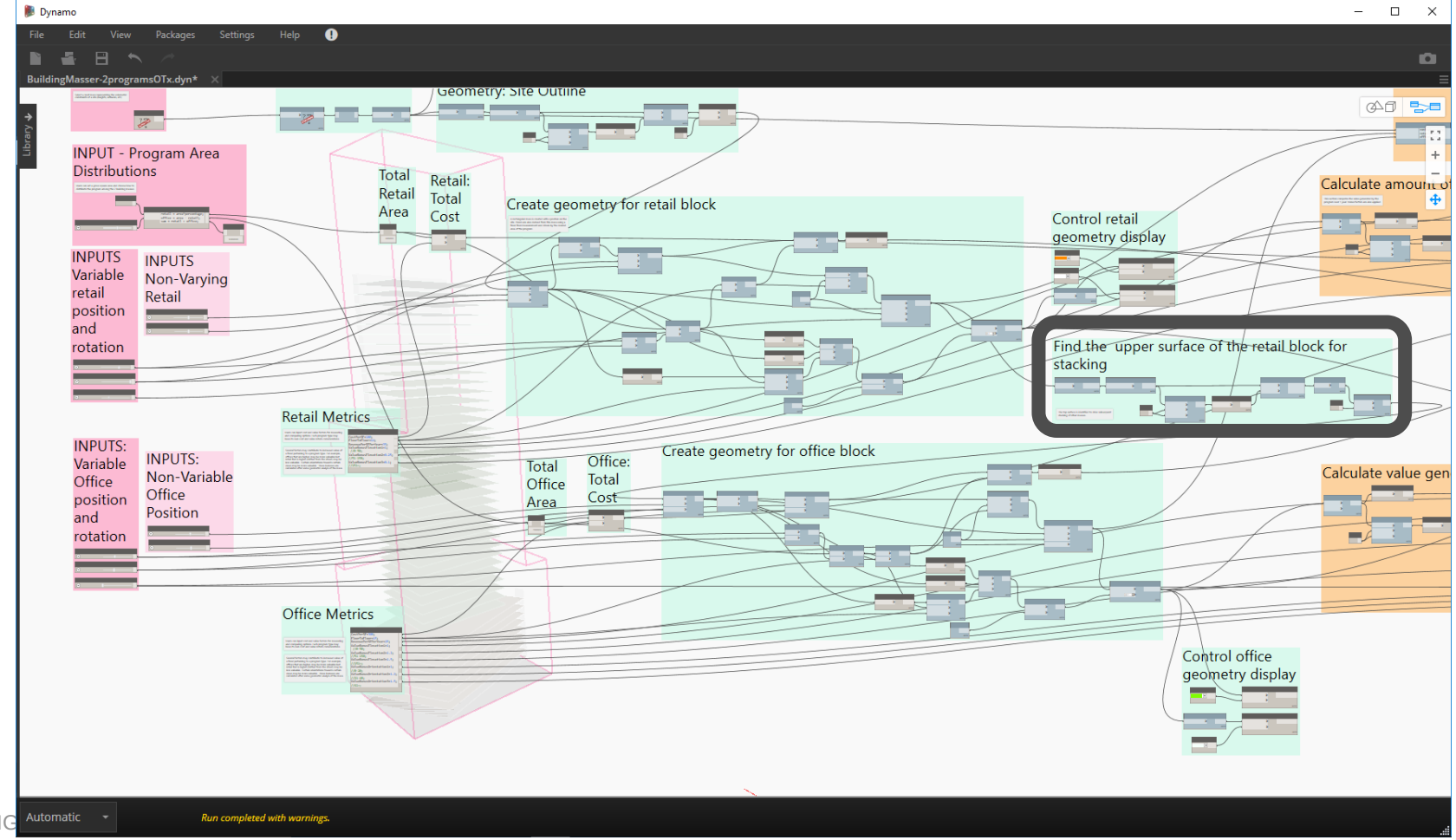

Dynamo
File
Edit
View
Packages
Settings
Help
BuildingMasser-2programsOTx.dyn*
Library
Office Metrics
Users can input cost and value factors for measuring and comparing options. Each program type may have its own cost and value return considerations.
Several factors may contribute to increased value of a floor pertaining to a program type. For example, offices that are higher, may be more valuable but retail that is higher (farther from the street) may be less valuable. Certain orientations towards certain views may be more valuable. These bonuses are calculated after some geometric analysis of the mass.
Office Metrics
CostPerSF=300;
FloorToFloor=15;
RevenuePerSFPerYear=25;
ValueBonusElevation1=1;
//0-50;
ValueBonusElevation2=1.2;
//51-150;
ValueBonusElevation3=1.5;
//151+;
ValueBonusOrientation1=1;
//0-20;
ValueBonusOrientation2=1.2;
//21-40;
ValueBonusOrientation3=1.5;
//41+;
Automatic
Run completed with warnings.

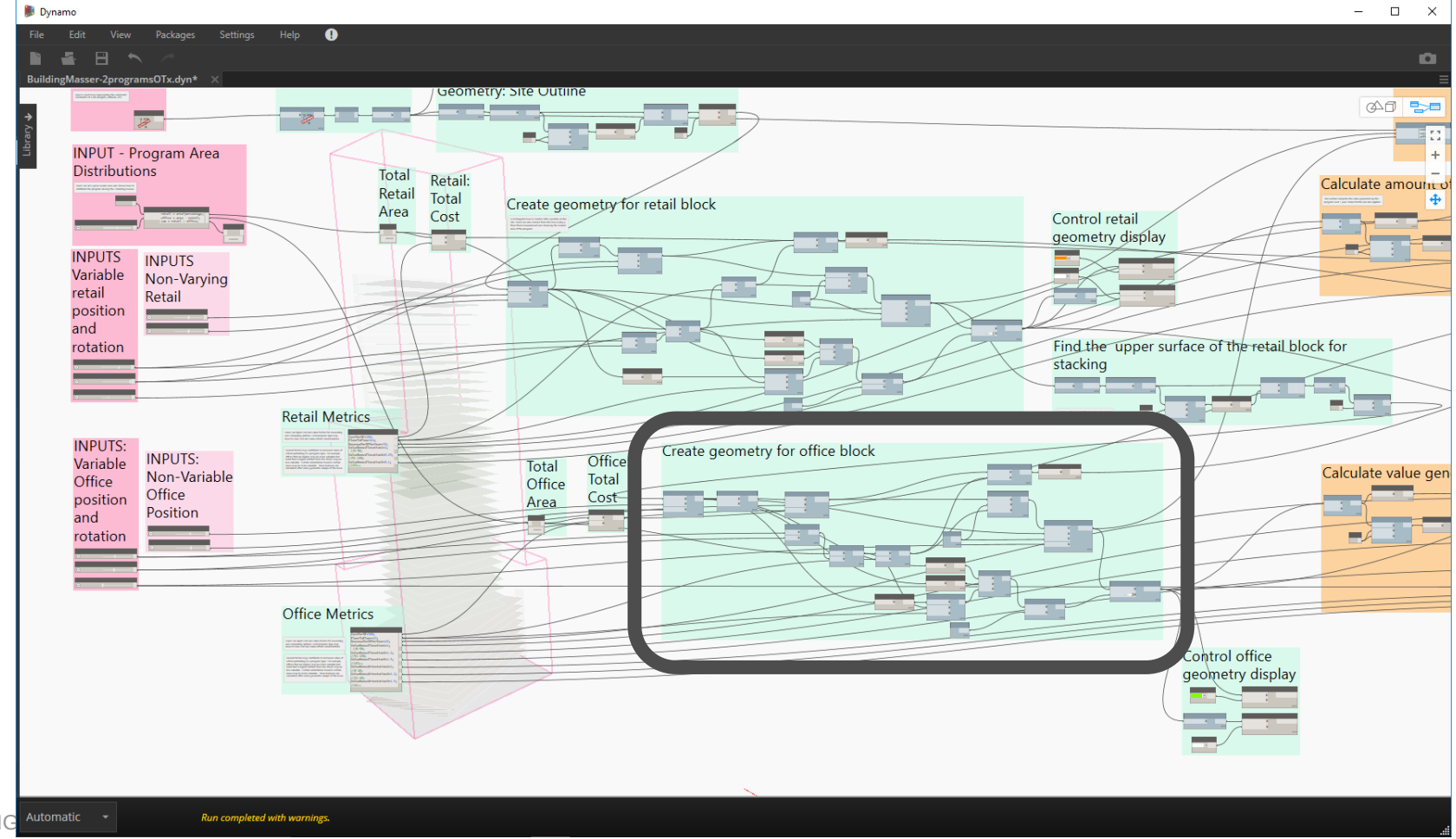
Metric Comparison

Building Massing Study

Metric	Retail	Office
Cost Per SF	$180	$300
Floor to Floor Height	14	15
Revenue/SF/YR	$35	$25
Elevation Bonus (0-50’)	1	1
Elevation Bonus 51-150’	0.25	1.2
Elevation Bonus 151’+	0.1	1.5
Orientation Bonus 0-20	n/a	1
Orientation Bonus 21-40	n/a	1.2
Orientation Bonus 41+	n/a	1.5


Dynamo
File
Edit
View
Packages
Settings
Help
BuildingMasser-2programsOTx.dyn*
Library
Geometry: Site Outline
INPUT - Program Area Distributions
Total Retail Area
Retail: Total Cost
Create geometry for retail block
Control retail geometry display
Calculate amount
INPUTS Variable retail position and rotation
INPUTS Non-Varying Retail
Find the upper surface of the retail block for stacking
Retail Metrics
INPUTS: Variable Office position and rotation
INPUTS: Non-Variable Office Position
Total Office Area
Office: Total Cost
Create geometry for office block
Calculate value gen
Office Metrics
Control office geometry display
Automatic
Run completed with warnings.


Dynamo
File
Edit
View
Packages
Settings
Help
BuildingMasser-2programsOTx.dyn*
Library
Geometry: Site Outline
INPUT - Program Area Distributions
Total Retail Area
Retail: Total Cost
Create geometry for retail block
Control retail geometry display
Calculate amount
INPUTS Variable retail position and rotation
INPUTS Non-Varying Retail
Find the upper surface of the retail block for stacking
Retail Metrics
INPUTS: Variable Office position and rotation
INPUTS: Non-Variable Office Position
Total Office Area
Office: Total Cost
Create geometry for office block
Calculate value gen
Office Metrics
Control office geometry display
Automatic
Run completed with warnings.


Dynamo
File
Edit
View
Packages
Settings
Help
BuildingMasser-2programsOTx.dyn*
Library
Geometry: Site Outline
INPUT - Program Area Distributions
Total Retail Area
Retail: Total Cost
Create geometry for retail block
Control retail geometry display
Calculate amount
INPUTS Variable retail position and rotation
INPUTS Non-Varying Retail
Find the upper surface of the retail block for stacking
Retail Metrics
INPUTS: Variable Office position and rotation
INPUTS: Non-Variable Office Position
Total Office Area
Office Total Cost
Create geometry for office block
Calculate value gen
Office Metrics
Control office geometry display
Automatic
Run completed with warnings.

Dynamo

File Edit View Packages Settings Help

BuildingMasser-2programsOTx.dyn*

Library

OUTPUT - Overall Metrics

The overall metrics show aggregate clashes, cost, and value generated. These metrics can be used as fitness functions for Refinery to evaluate.

OUTPUT - Program 1 - Value Generated
5994507.24329625

OUTPUT - Program 2 - Value Generated
25966614.480585

List Create
item0 + - list
item1

Math.Sum
values sum
AUTO

TotValue/Yr
31961121.7238813

VolumeOutsideZoning
567803.869391852

/
x
y
var[]..[]
AUTO

YrsToPayback
8.26003549815149

OUTPUT - Program 1 - Building Cost
54000000

OUTPUT - Program 2 - Building Cost
210000000

List Create
item0 + - list
item1

Math.Sum
values sum
AUTO

TotCost
264000000

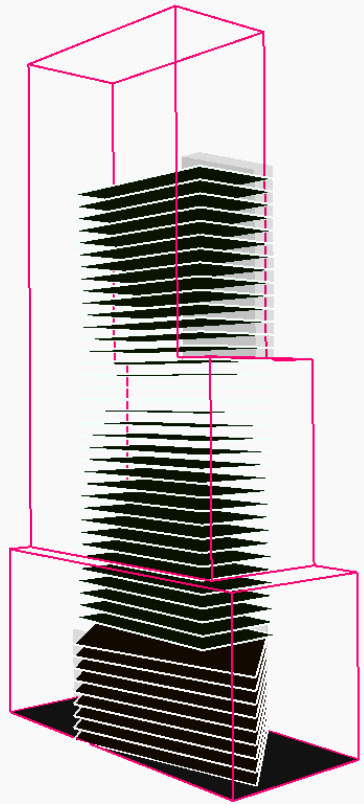
OUTPUT - Program 1 - Ext. Surface Area
93485.85450901

OUTPUT - Program 2 - Ext. Surface Area
301172.081013767

List Create
item0 + - list
item1

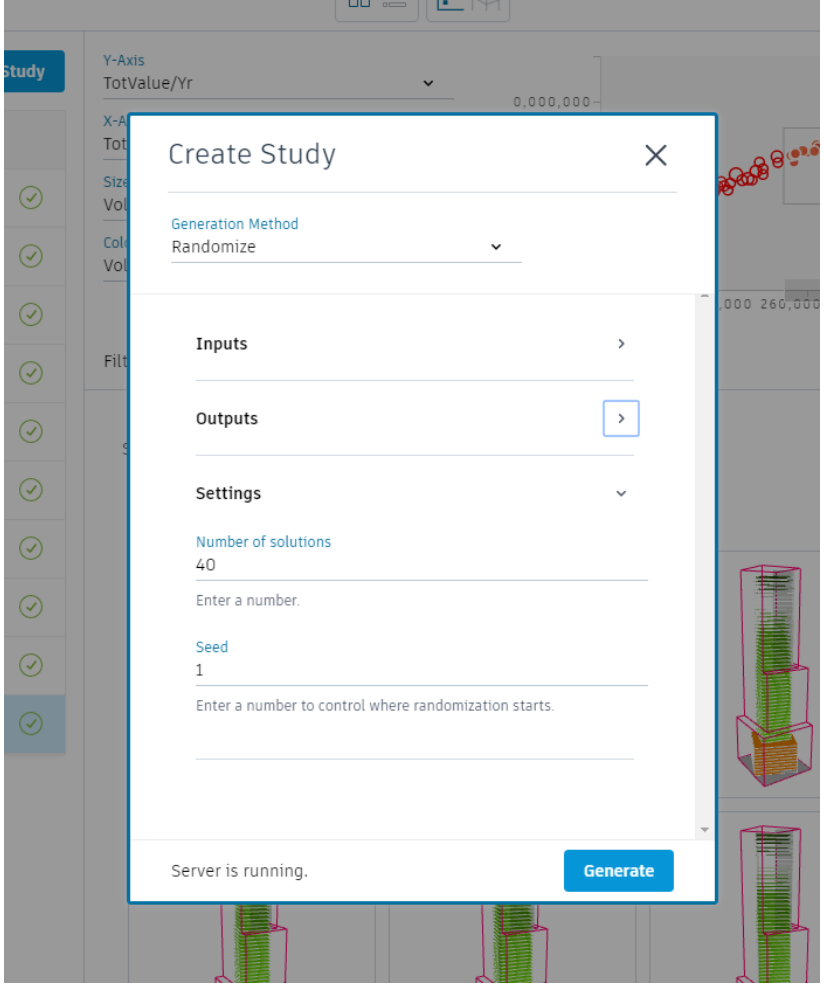
Math.Sum
values sum
AUTO

TotExtSurfArea
394657.935522777

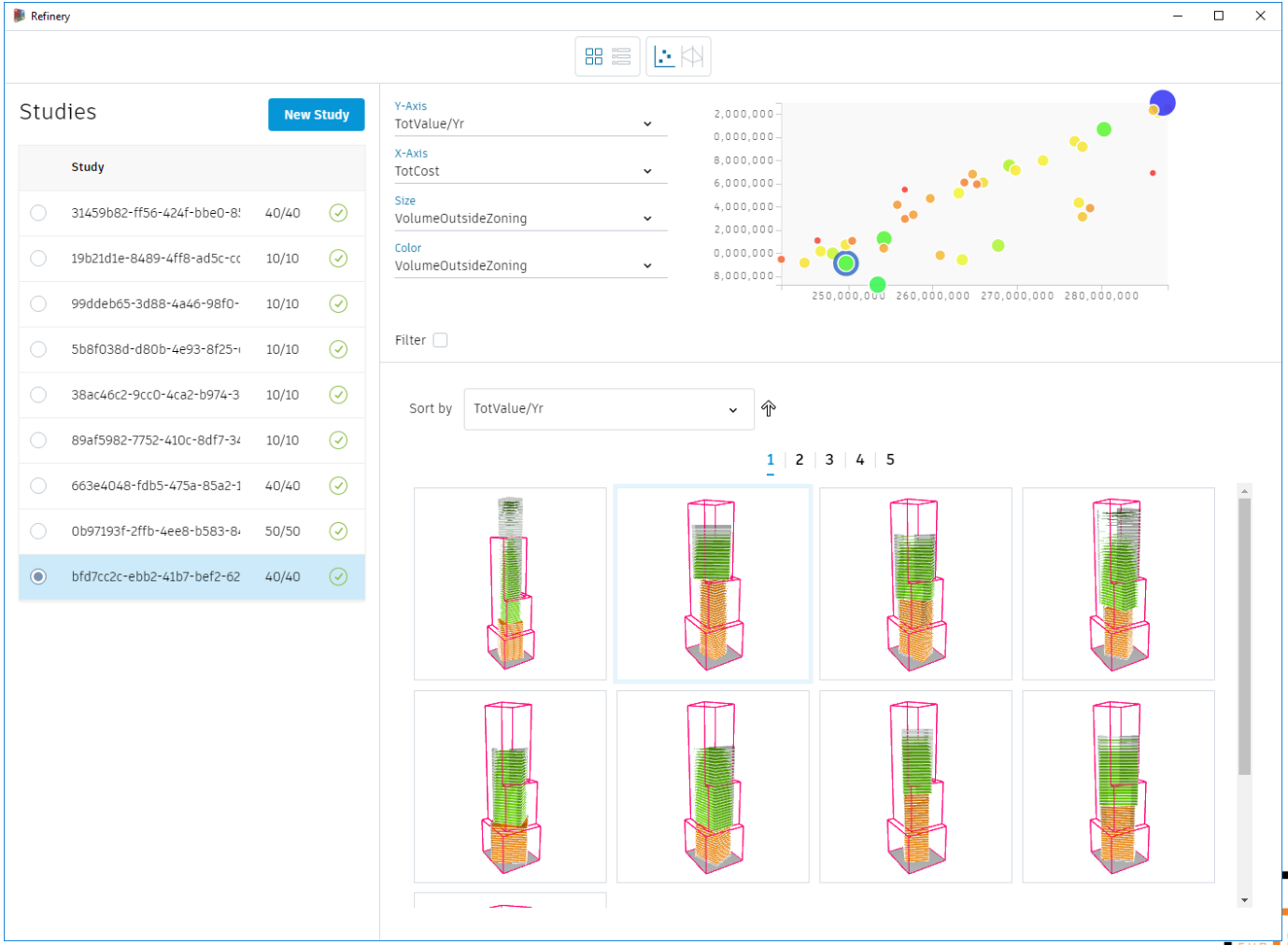


Automatic

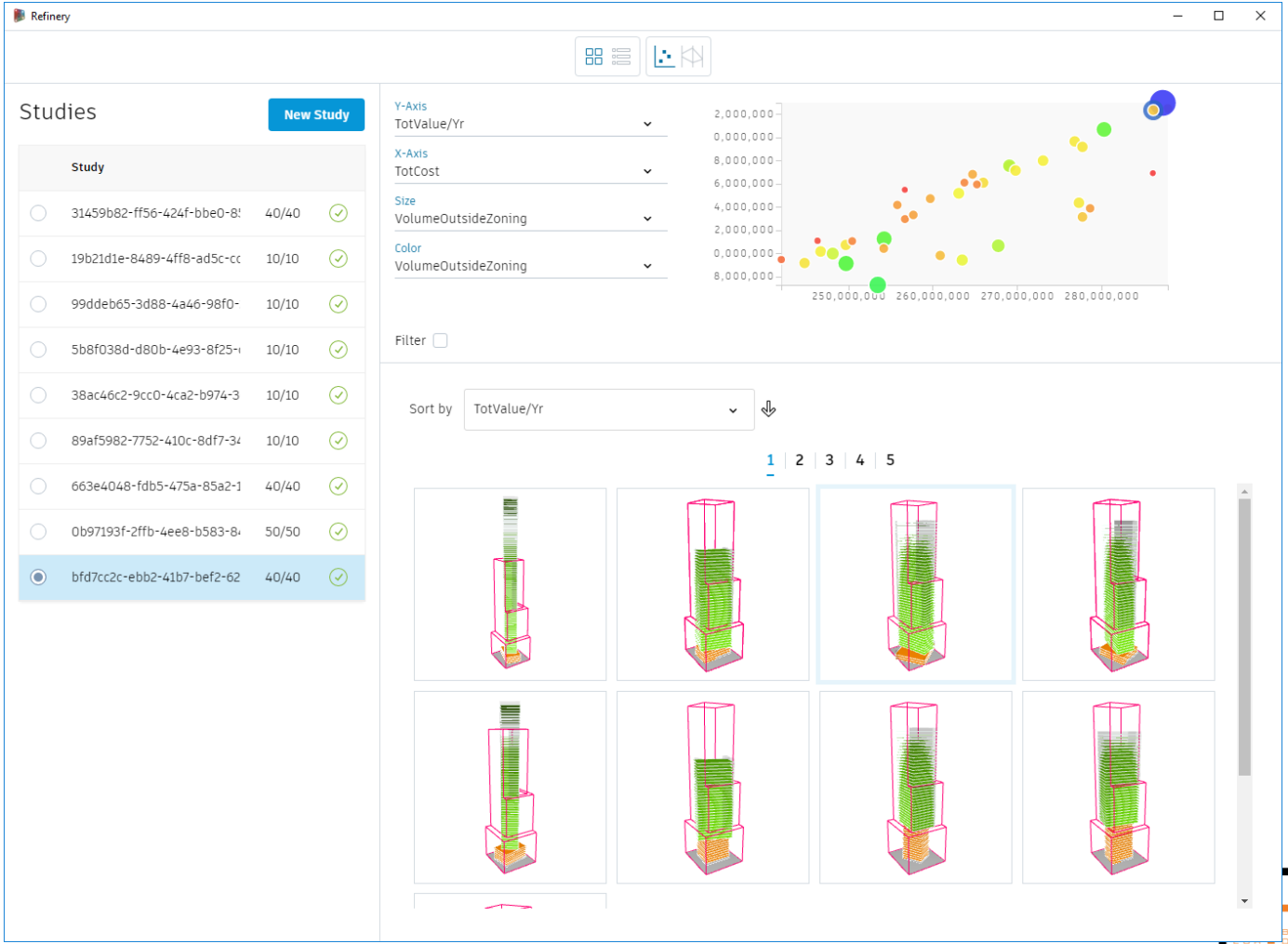
Run completed with warnings.


Y-Axis
TotValue/Yr
Create Study
Generation Method
Randomize
Inputs
Outputs
Settings
Number of solutions
40
Enter a number.
Seed
1
Enter a number to control where randomization starts.
Server is running.
Generate

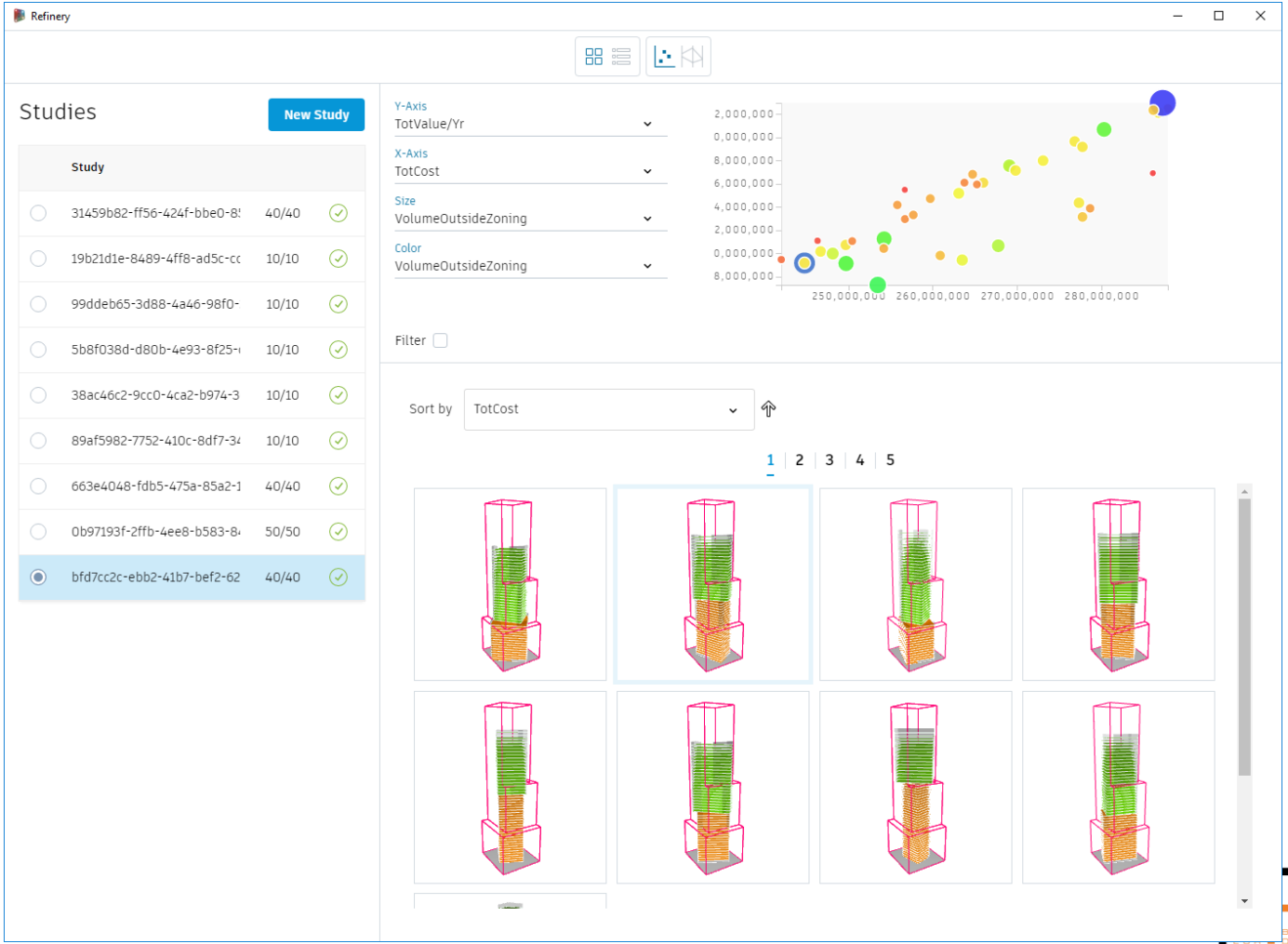
- 40 Random Runs
- Least Value/Yr



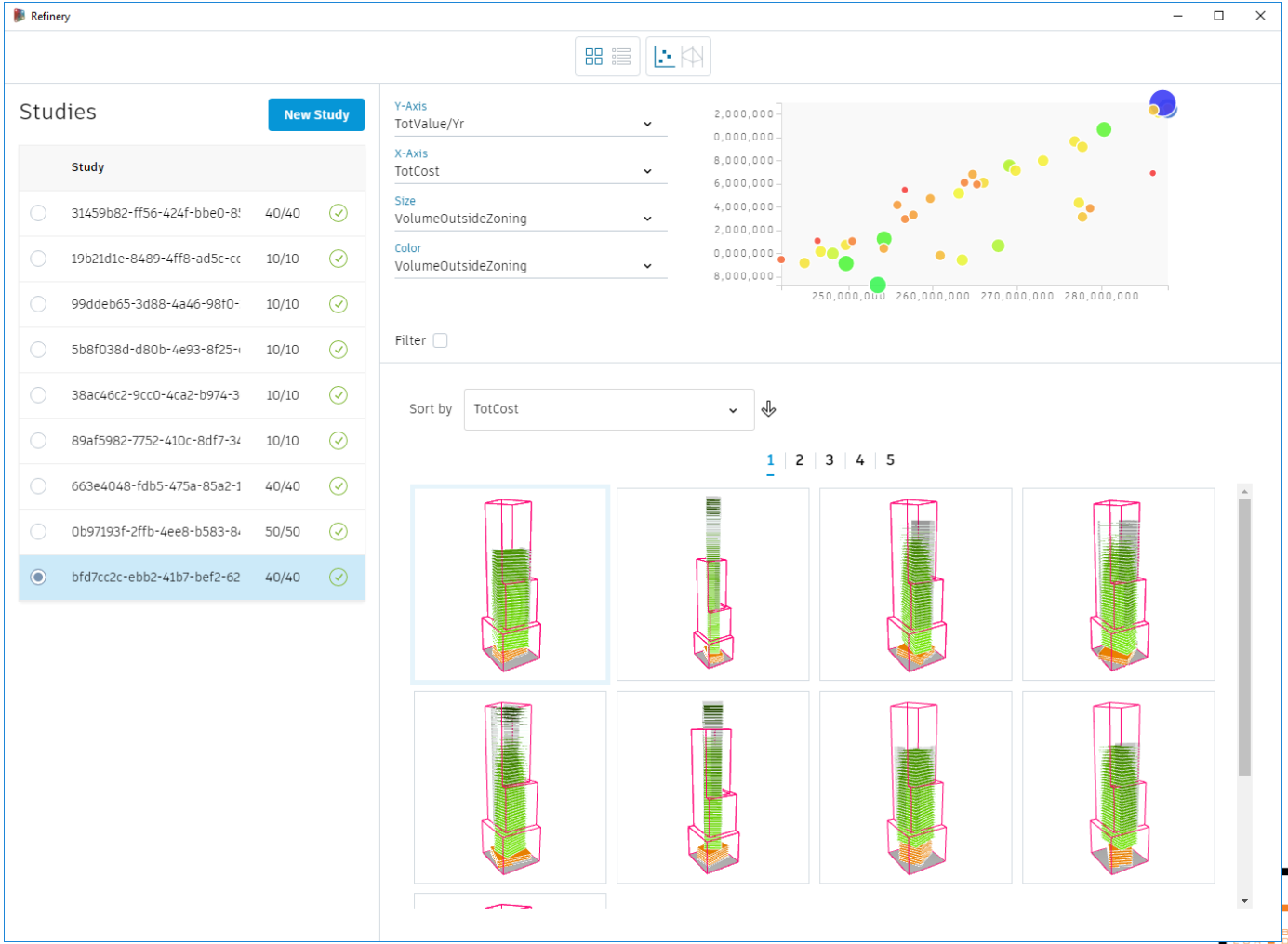
- 40 Random Runs
- Most Value/Yr



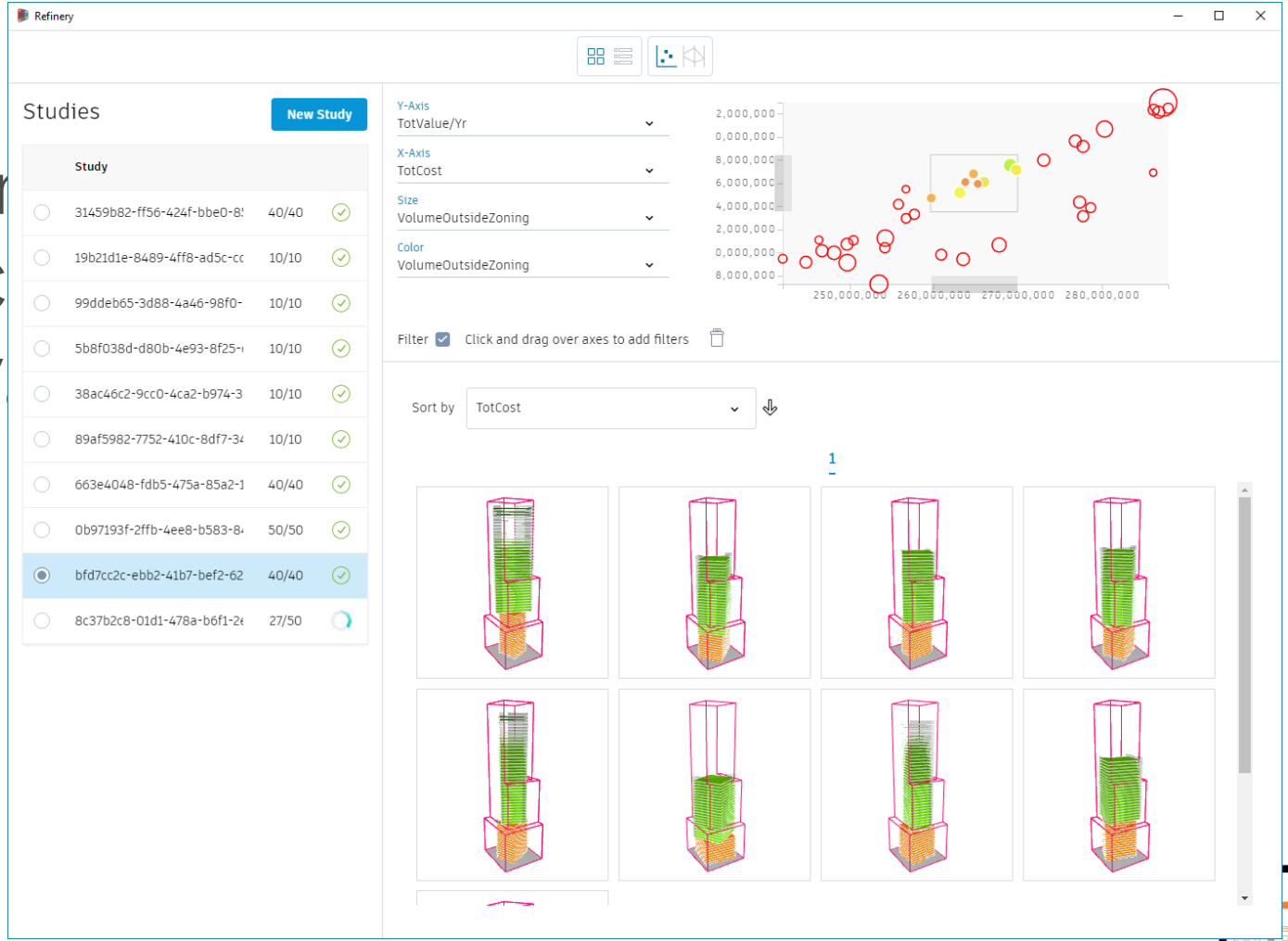
- 40 Random Runs
- Lowest Cost



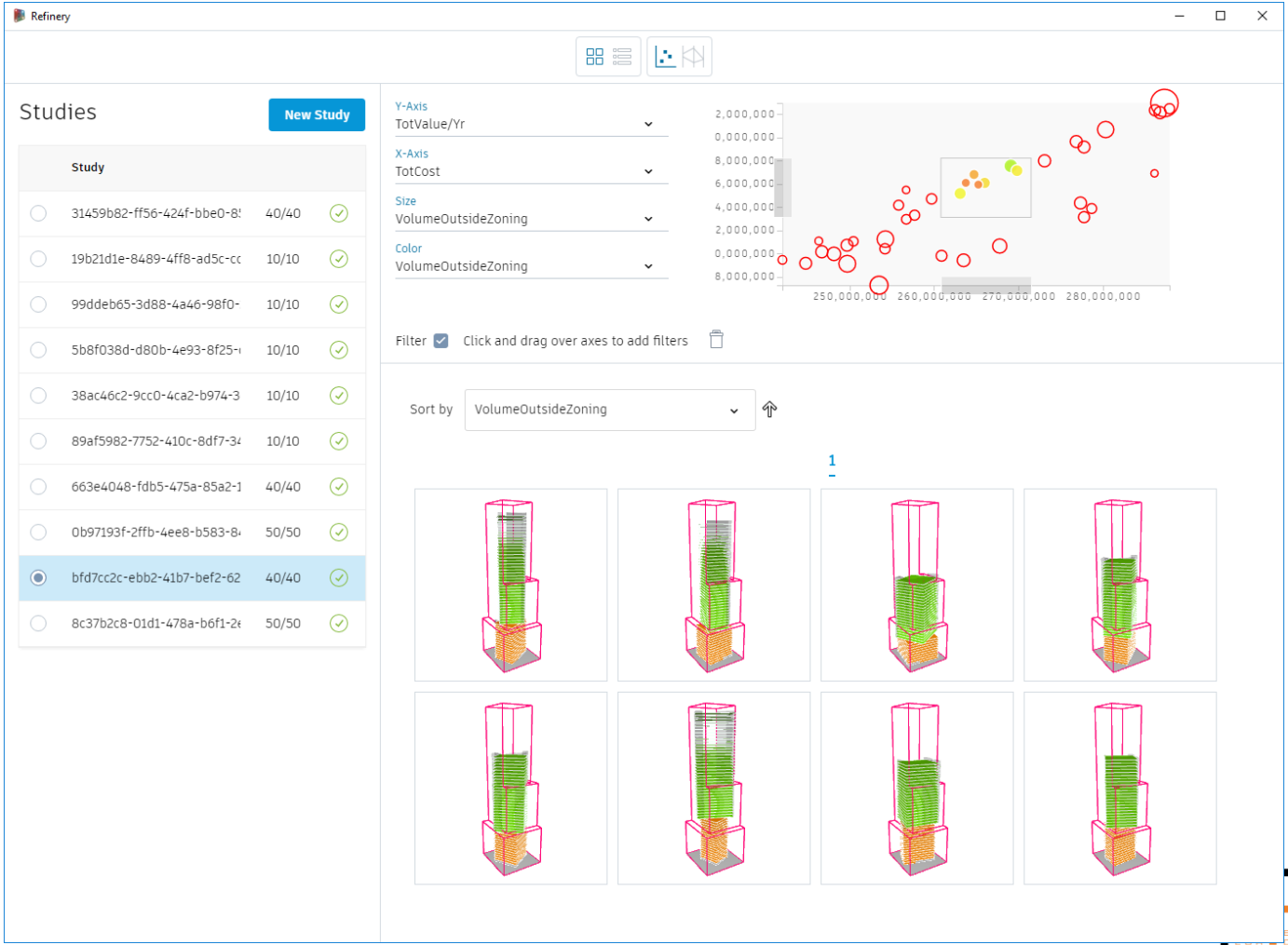
- 40 Random Runs
- Highest Cost



- 40 Randor
- Medium C
- Medium V



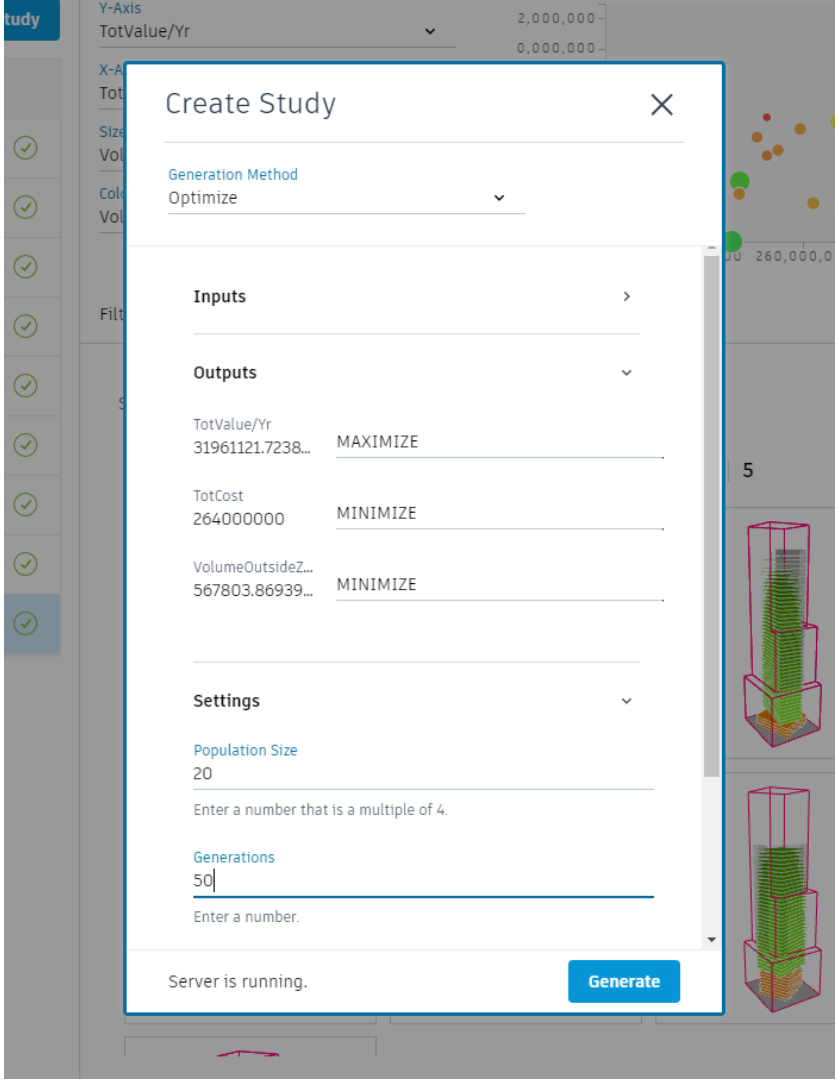
- 40 Random Runs
- Medium Cost
- Medium Value
- Lowest volume
- Outside zoning



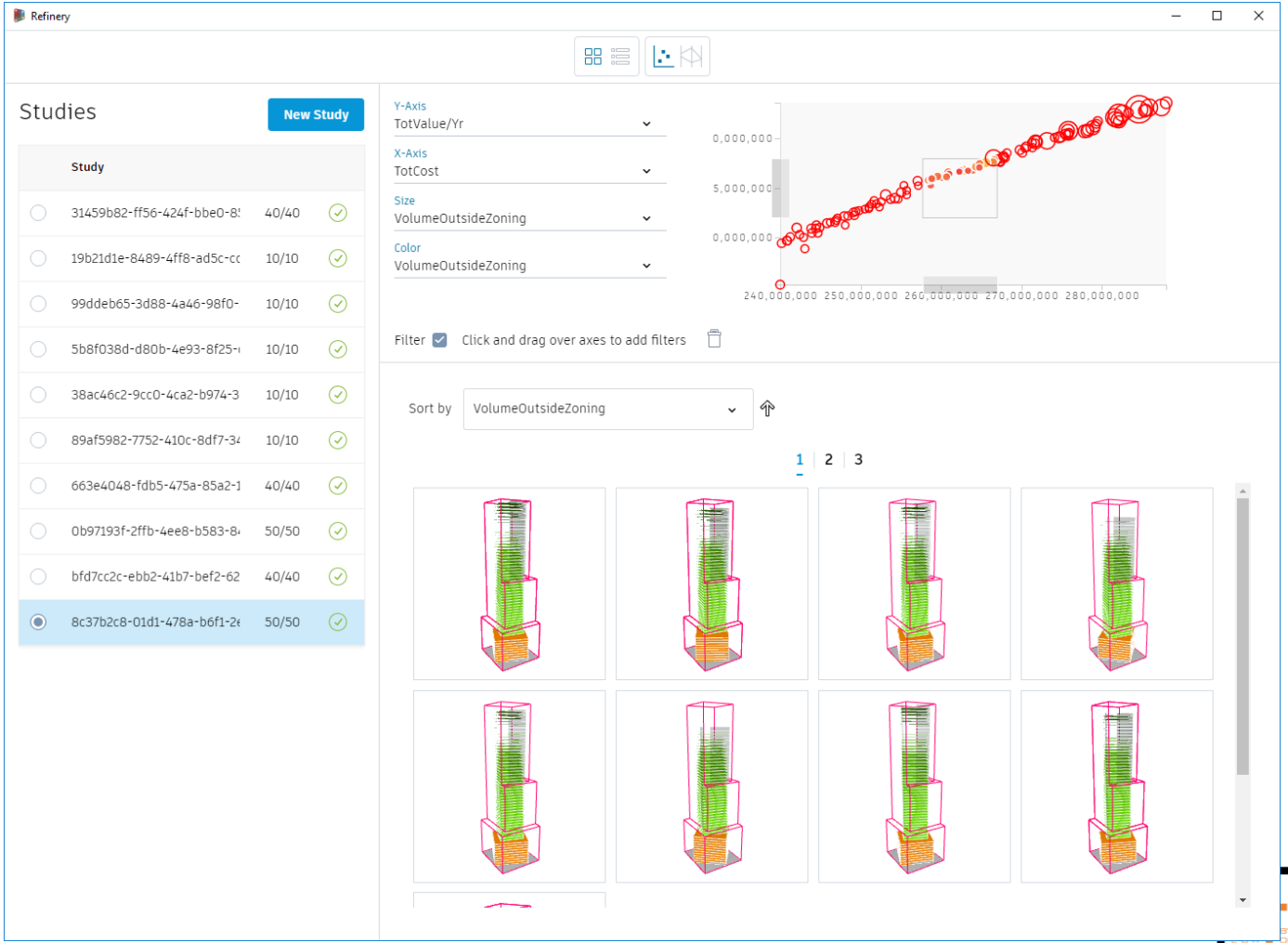
DIGITAL BUILT WEEK EUROPE

EDINBURGH

10 – 12 OCT 2019


Y-Axis
TotValue/Yr
2,000,000
0,000,000
260,000,0
Create Study
Generation Method
Optimize
Inputs
Outputs
TotValue/Yr
31961121.7238...
MAXIMIZE
TotCost
264000000
MINIMIZE
VolumeOutsideZ...
567803.86939...
MINIMIZE
Settings
Population Size
20
Enter a number that is a multiple of 4.
Generations
50
Enter a number.
Server is running.
Generate

- 20x50 Optimization
- Medium Cost
- Medium Value
- Lowest volume
- Outside zoning

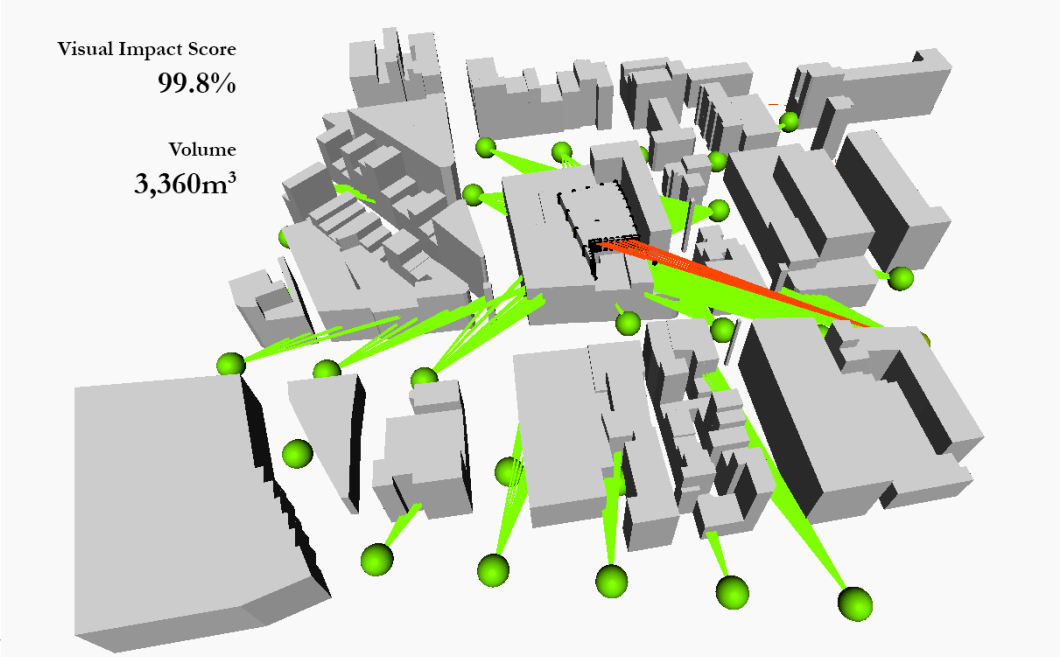


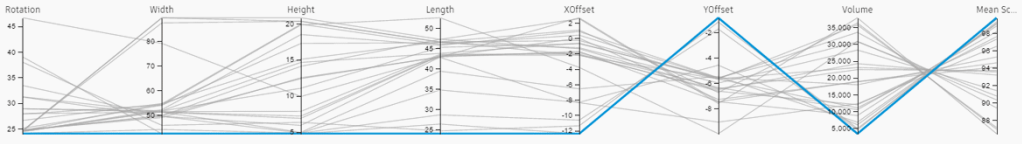
Example 3.5

Adding New Evaluator to Building Massing

Stealthy RoofScapes!

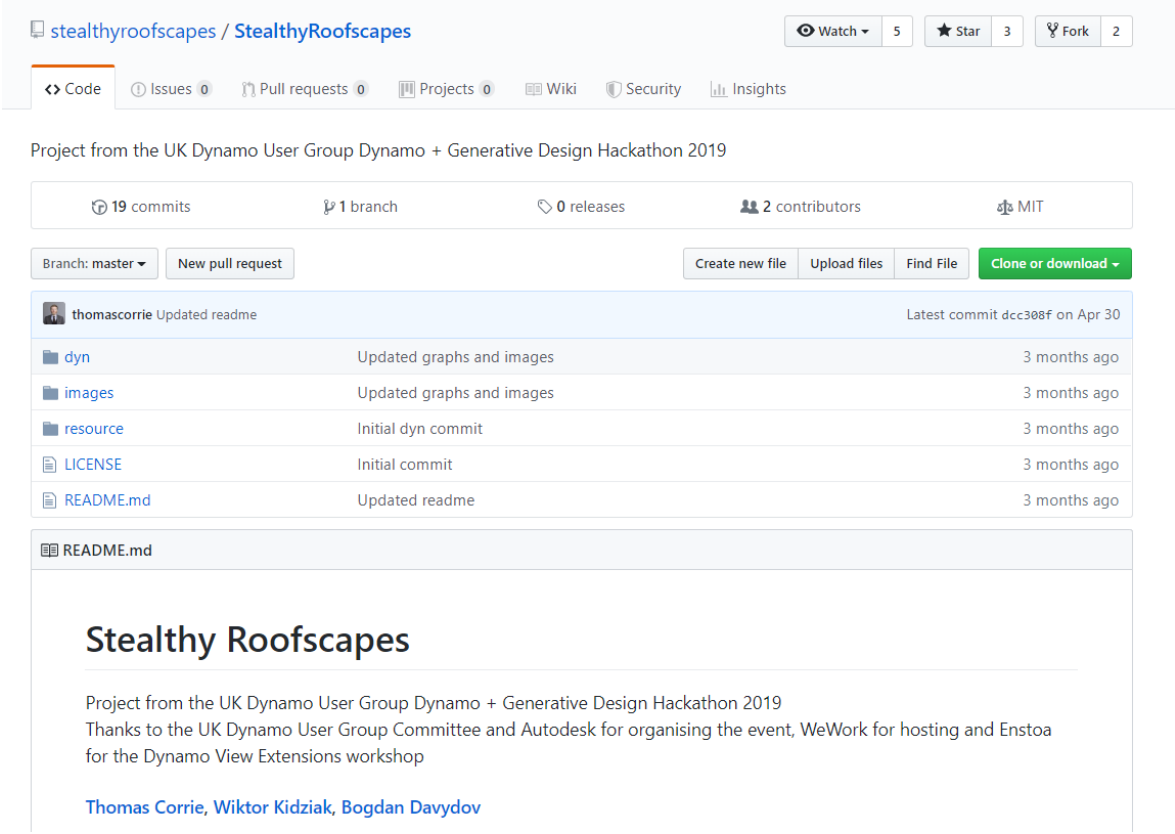
UK Dynamo User Group's Dynamo + Generative Design Hackathon 2019





- Bogdan Davydo
- Wiktor Kidziak
- Thomas Corrie

DIGITAL BUILT WEEK EUROPE
EDINBURGH
10 – 12 OCT 2019



stealthyroofscapes / StealthyRoofscapes

Watch 5 | Star 3 | Fork 2

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights

Project from the UK Dynamo User Group Dynamo + Generative Design Hackathon 2019

19 commits | 1 branch | 0 releases | 2 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

thomascorrie Updated readme — Latest commit dcc308f on Apr 30

dyn	Updated graphs and images	3 months ago
images	Updated graphs and images	3 months ago
resource	Initial dyn commit	3 months ago
LICENSE	Initial commit	3 months ago
README.md	Updated readme	3 months ago

README.md

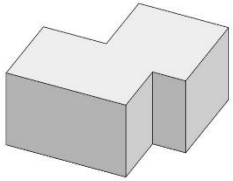
Stealthy Roofscapes

Project from the UK Dynamo User Group Dynamo + Generative Design Hackathon 2019
Thanks to the UK Dynamo User Group Committee and Autodesk for organising the event, WeWork for hosting and Enstoa for the Dynamo View Extensions workshop

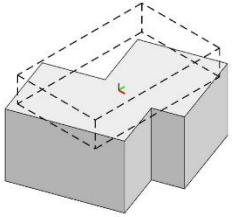
Thomas Corrie, Wiktor Kidziak, Bogdan Davydov

Stealthy RoofScapes!

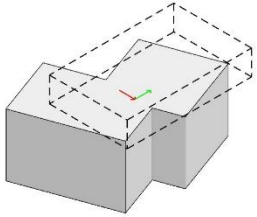
UK Dynamo User Group's Dynamo + Generative Design Hackathon 2019



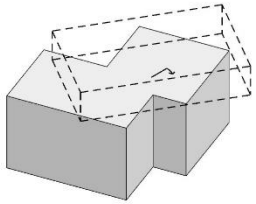
Existing Building



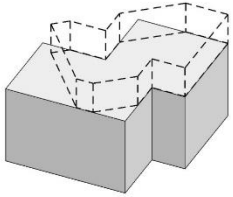
Cuboid placed at centroid of roof aligned with coordinate system
- Length
- Width
- Height



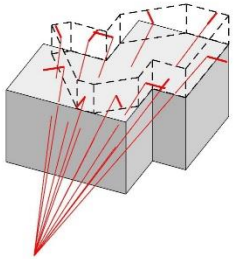
Cuboid moved in x and y
- XOffset
- YOffset



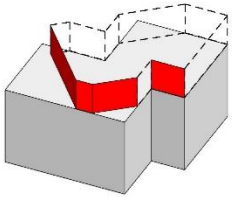
Cuboid rotated
- Rotation



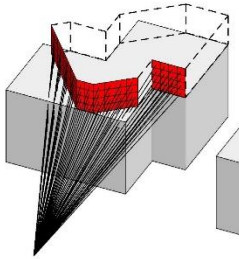
Cuboid trimmed to extent of existing roof boundary



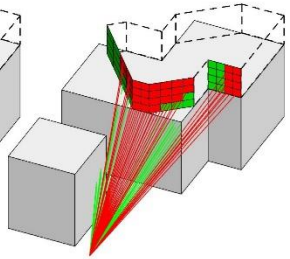
For each viewpoint, establish the angle between the normal of each face of the addition (ignoring top and bottom faces) and the sightline



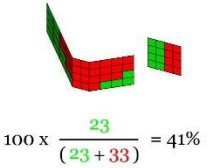
Faces with angles to the sightline of less than 100° face towards the viewpoint and will be considered



Each face is divided into panels and a sightline established from the panel centre to the viewpoint



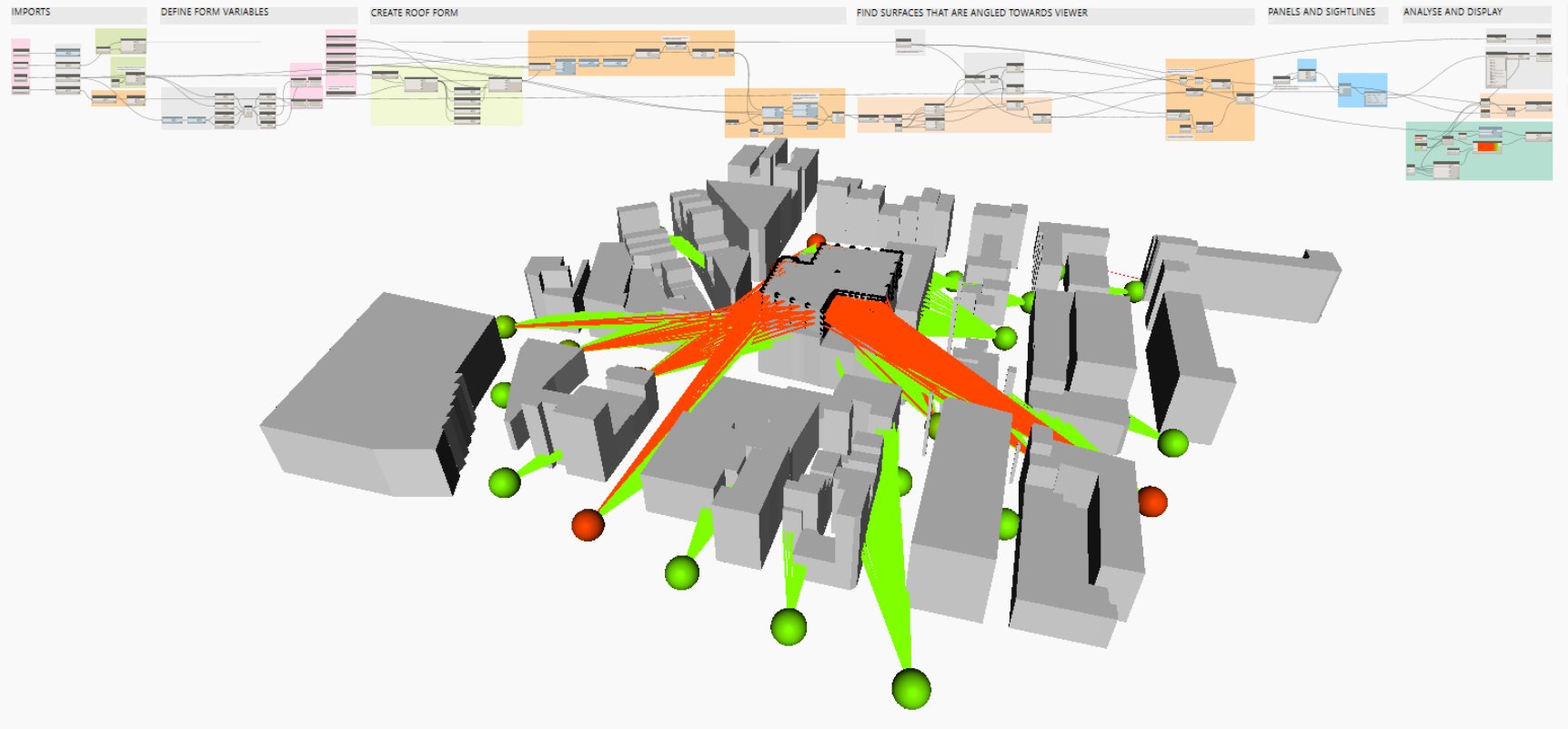
The sightline is tested for intersections with the surrounding buildings (including the existing building below) and if it intersects it the panel is considered not to be seen (green)



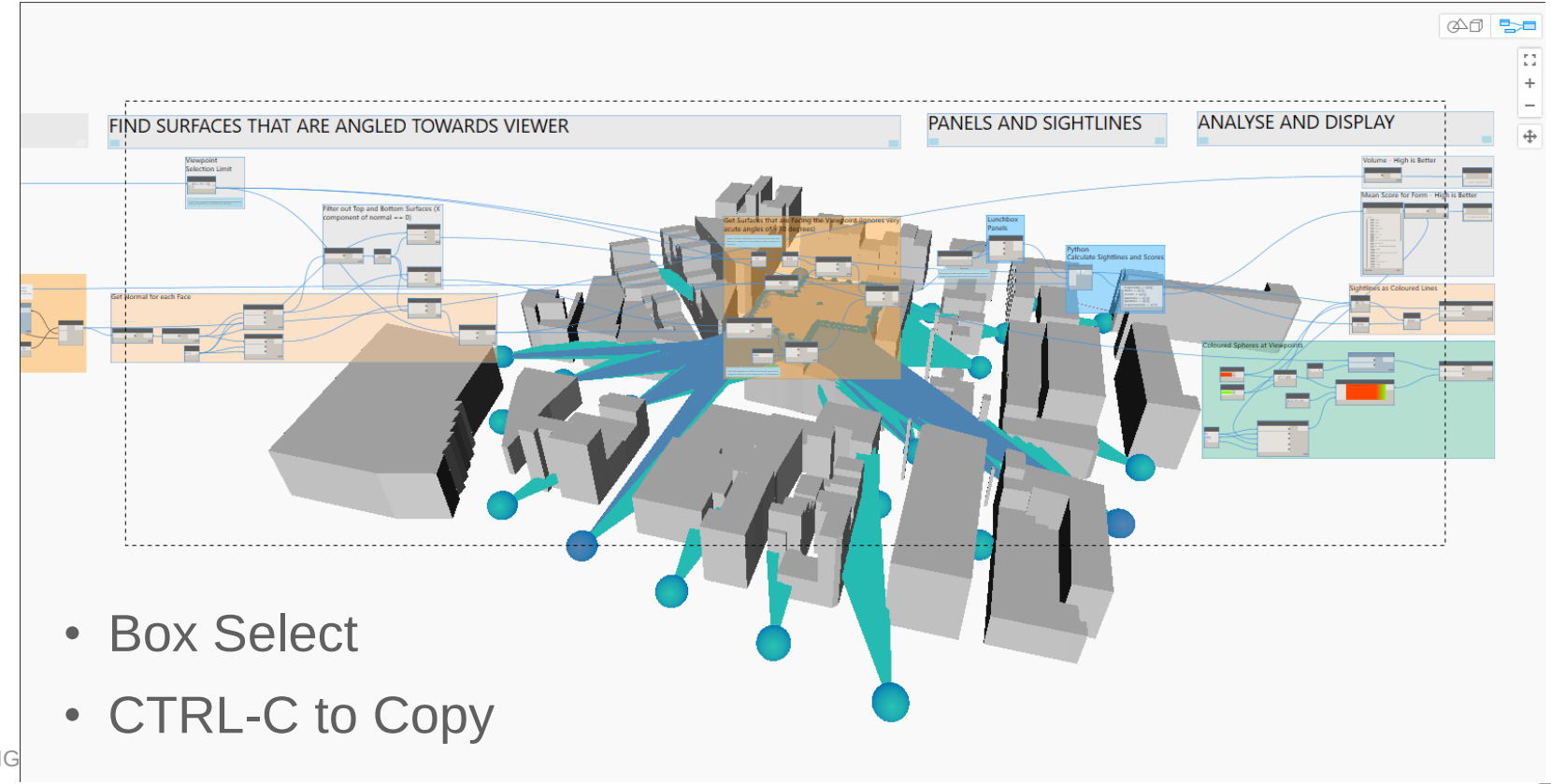
The score for the viewpoint is the percentage of panels not visible from the viewpoint. The score for the whole is the mean percentage from all viewpoints

Stealthy RoofScapes!

UK Dynamo User Group's Dynamo + Generative Design Hackathon 2019

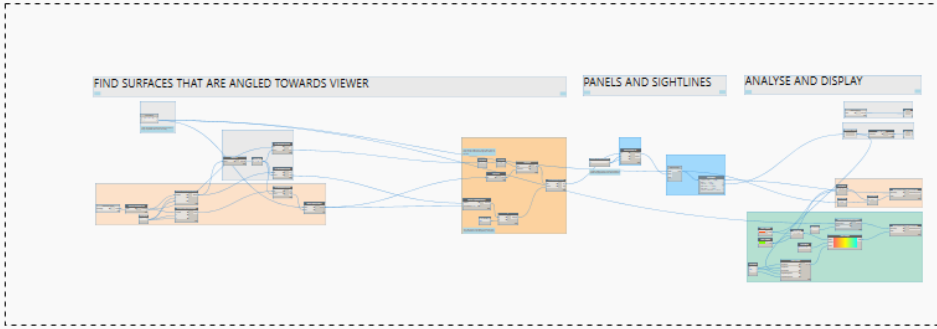


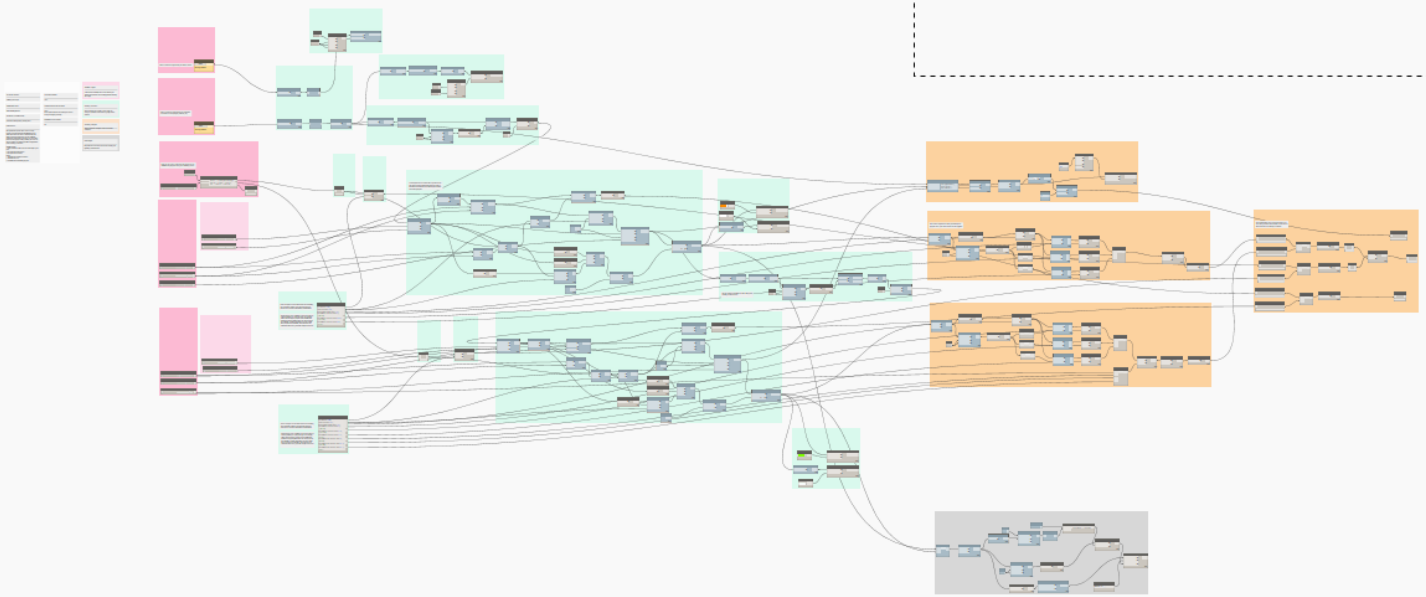
Re-using and Re-applying Evaluators



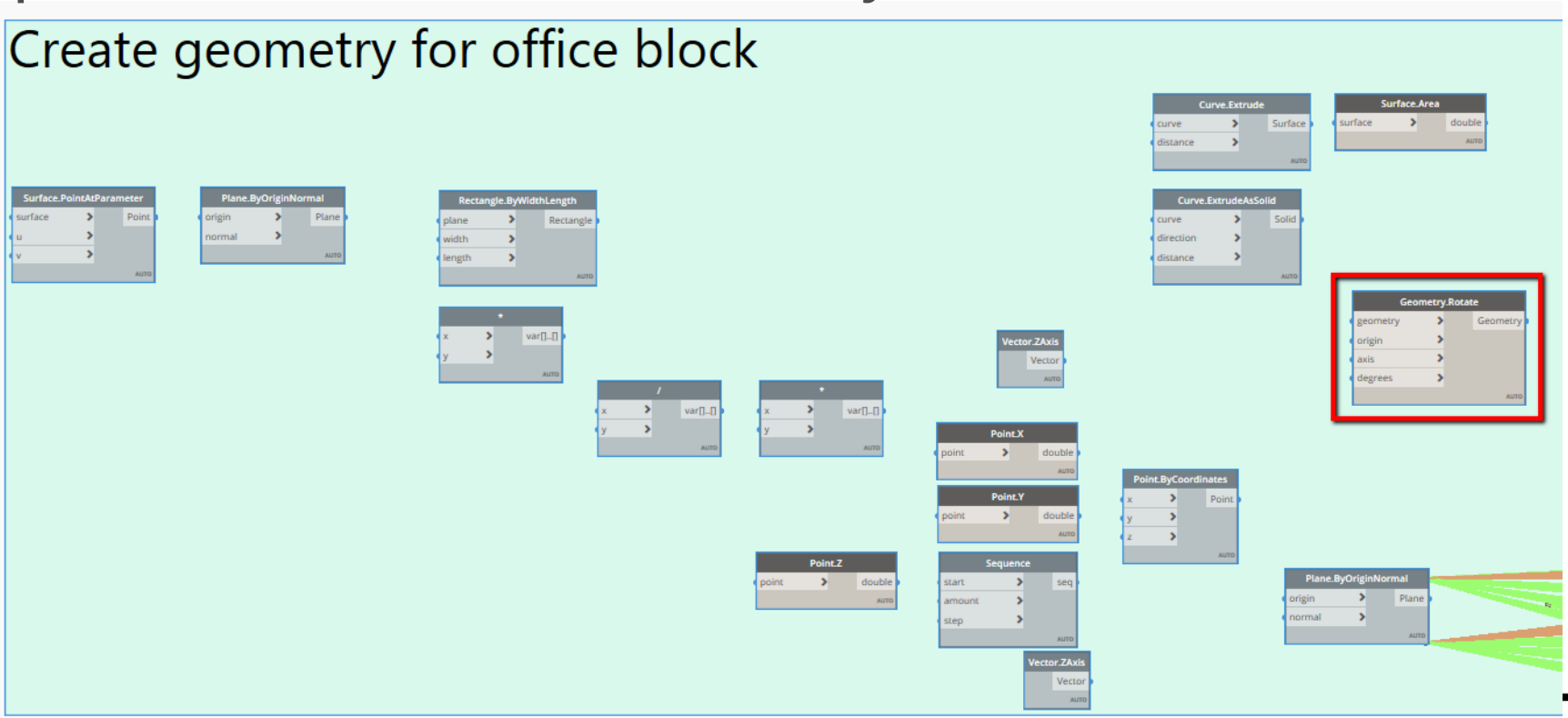
- Box Select
- CTRL-C to Copy

- In Same Dynamo/Revit Session
- Open Building Massing RVT/DYN
- CTRL-V to Paste

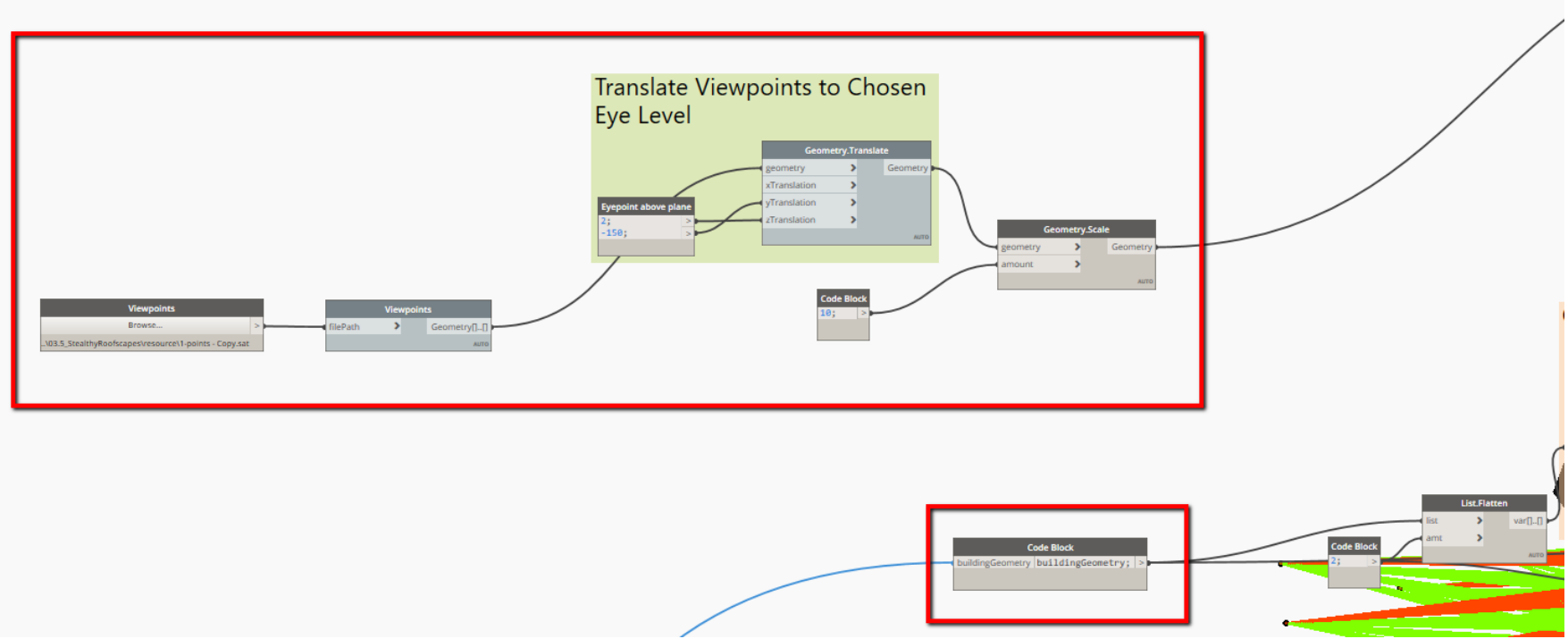


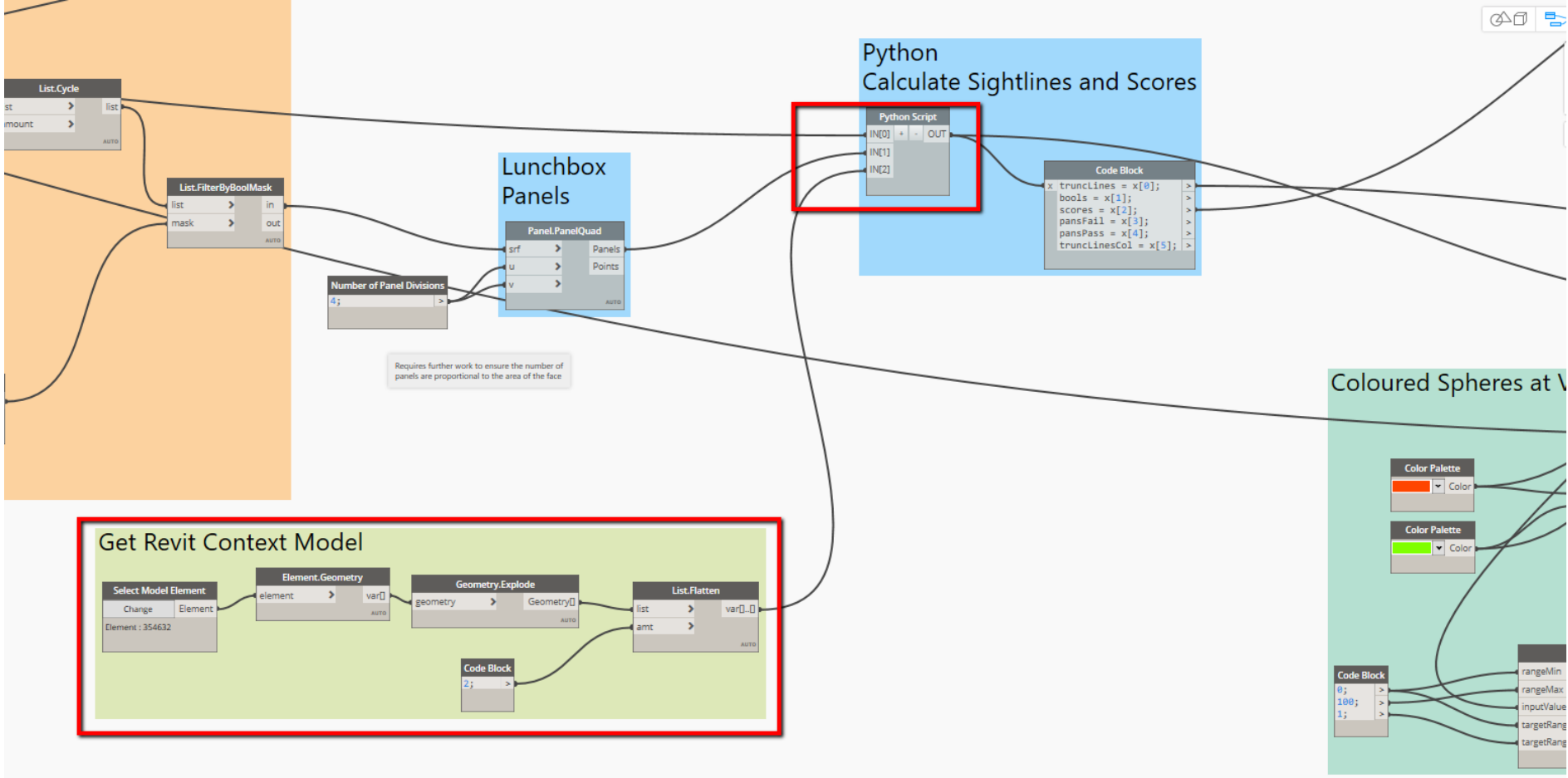


Inputs – Office Block Geometry



Inputs – Sample Points

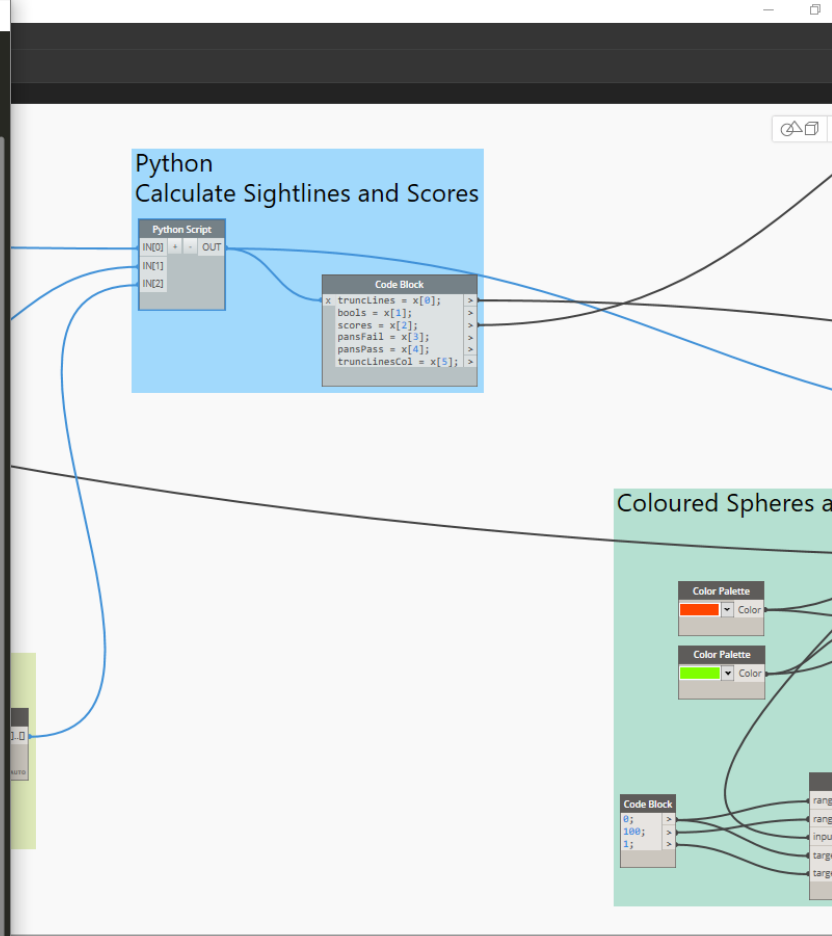


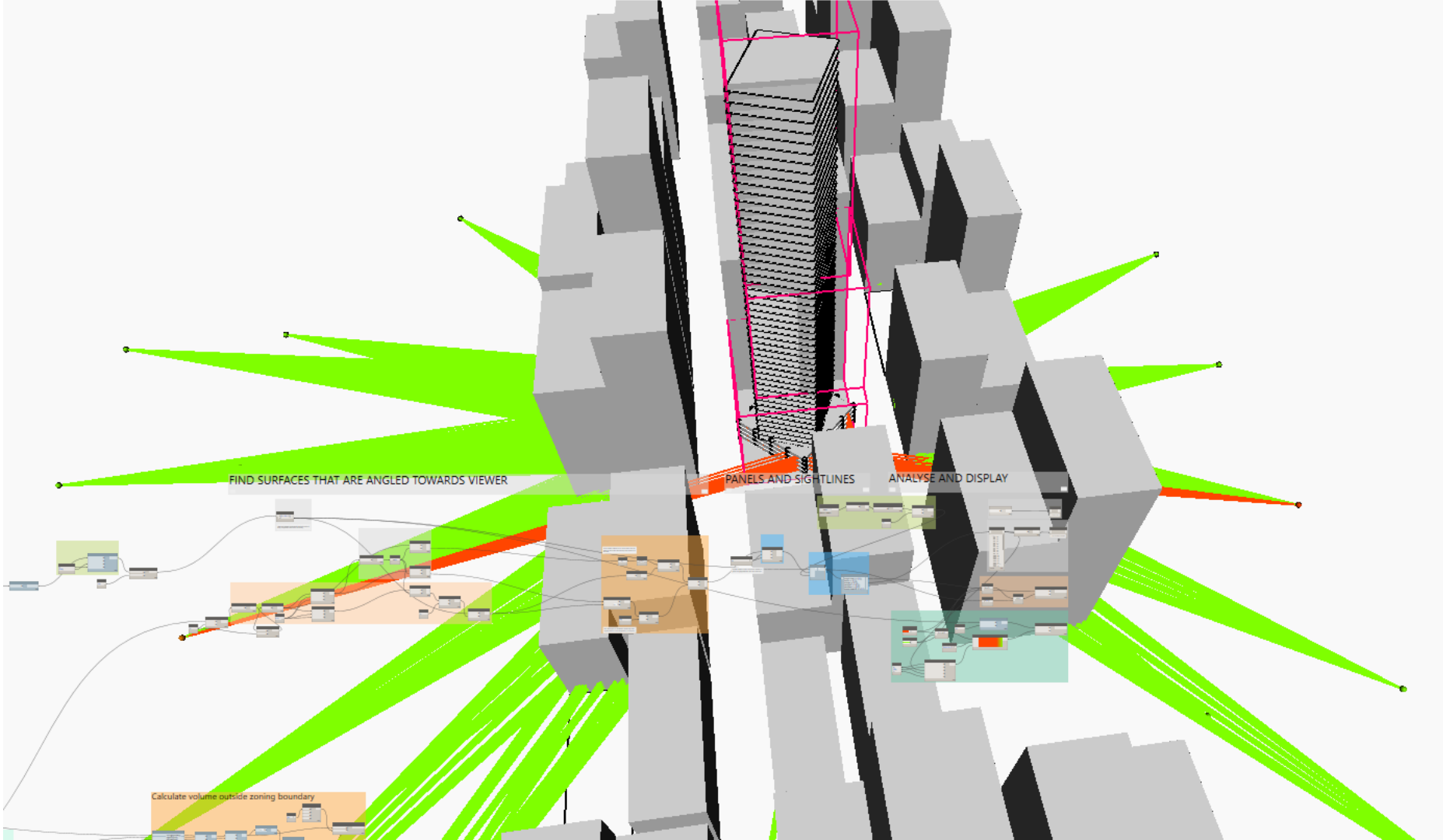

List.Cycle
List.FilterByBoolMask
Number of Panel Divisions
4;
Lunchbox
Panels
Panel.PanelQuad
Requires further work to ensure the number of panels are proportional to the area of the face
Python
Calculate Sightlines and Scores
Python Script
Code Block
truncLines = x[0];
bools = x[1];
scores = x[2];
pansFail = x[3];
pansPass = x[4];
trunclinesCol = x[5];
Get Revit Context Model
Select Model Element
Element.Geometry
Geometry.Explode
List.Flatten
Code Block
2;
Coloured Spheres at V
Color Palette
Color Palette
Code Block
0;
100;
1;

Python Script

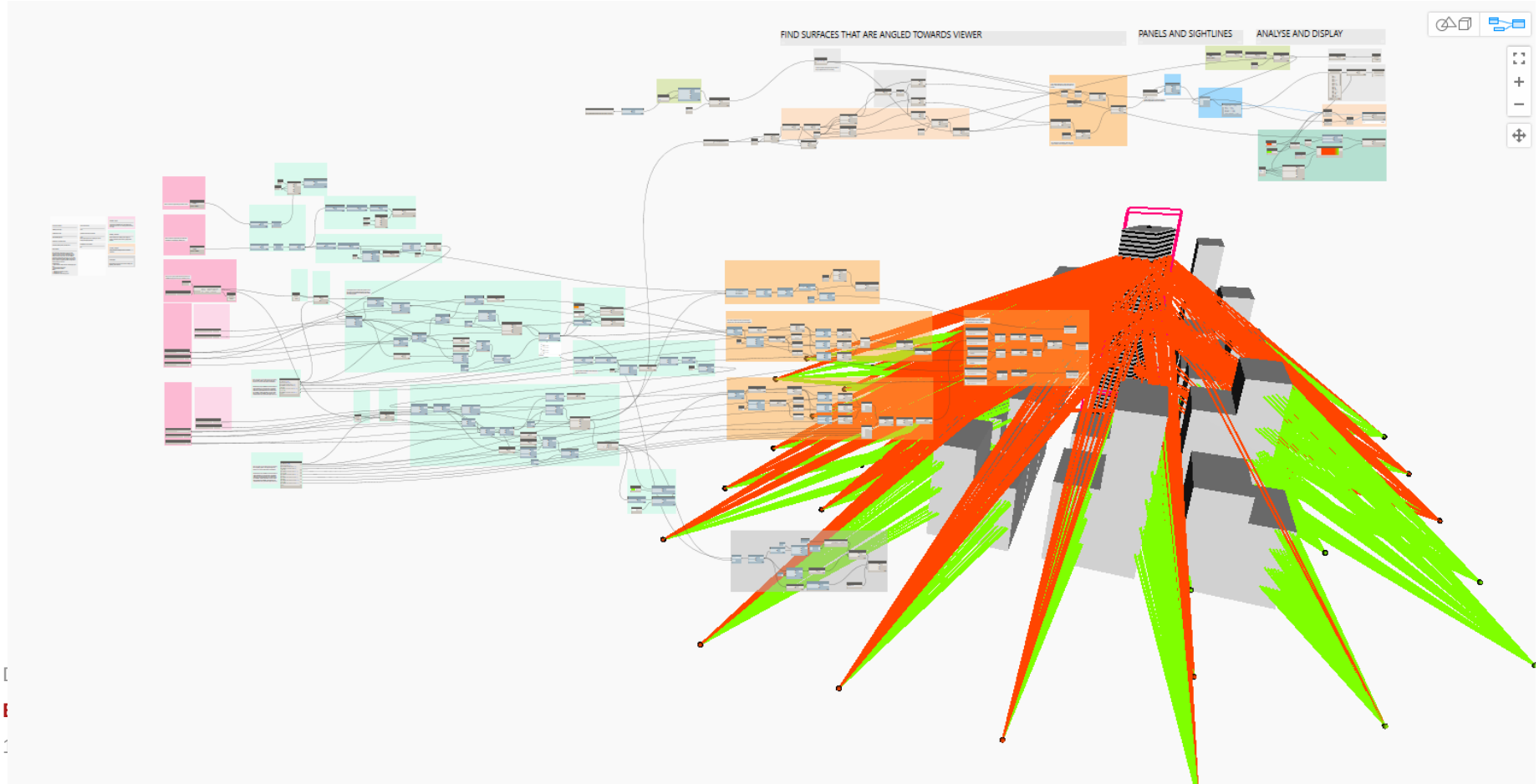
```python
# The inputs to this node will be stored as a list in the IN variables.
points = IN[0]
panels = IN[1]
buildings = IN[2]

lines = []
truncLines = []
truncLinesCol = []
dists = []
buildingsSort = []
keysSort = []
bools = []
scores = []
pansFail = []
pansPass = []

for p1, bPanels in zip(points, panels): # single viewpoint
    #dists.append([])
    dist = []
    bools.append([])
    for b in buildings: # loop through surfaces of buildings and get distance from viewpoint
        dist.append(Geometry.DistanceTo(p1, b))
    dists.append(dist)
    bSort = [x for _,x in sorted(zip(dist,buildings))] # sort by distance from viewpoint
    buildingsSort.append(bSort)
    #dists.append(dist)
    lines.append([])
    truncLines.append([])
    truncLinesCol.append([])
    score = 0
    panPass = 0
    panFail = 0
    for sPanels in bPanels: # side panels in viewpoint
        lines[-1].append([])
        truncLines[-1].append([])
        truncLinesCol[-1].append([])
        bools[-1].append([])
        for pan in sPanels: # for each panel
            p2 = Surface.PointAtParameter(pan, 0.5, 0.5)
            ln = Line.ByStartPointEndPoint(p1,p2)
            lines[-1][-1].append(ln)
            bool = False
            for b in bSort:
                if Geometry.DoesIntersect(b, ln):
                    tp = Geometry.Intersect(b, ln)[0]
                    truncLines[-1][-1].append(Line.ByStartPointEndPoint(p1,tp))
                    truncLinesCol[-1][-1].append(0)
                    bool = True
                    break
            bools[-1][-1].append(bool)
            if bool:
                panPass += 1
            else:
                panFail += 1
                truncLines[-1][-1].append(ln)
                truncLinesCol[-1][-1].append(1)
    score = float(panPass) / (panPass + panFail) * 100
    scores.append(score)
    pansFail.append(panFail)
    pansPass.append(panPass)
# Assign your output to the OUT variable.
OUT = truncLines, bools, scores, pansFail, pansPass, truncLinesCol
```

Run | Save Changes | Revert



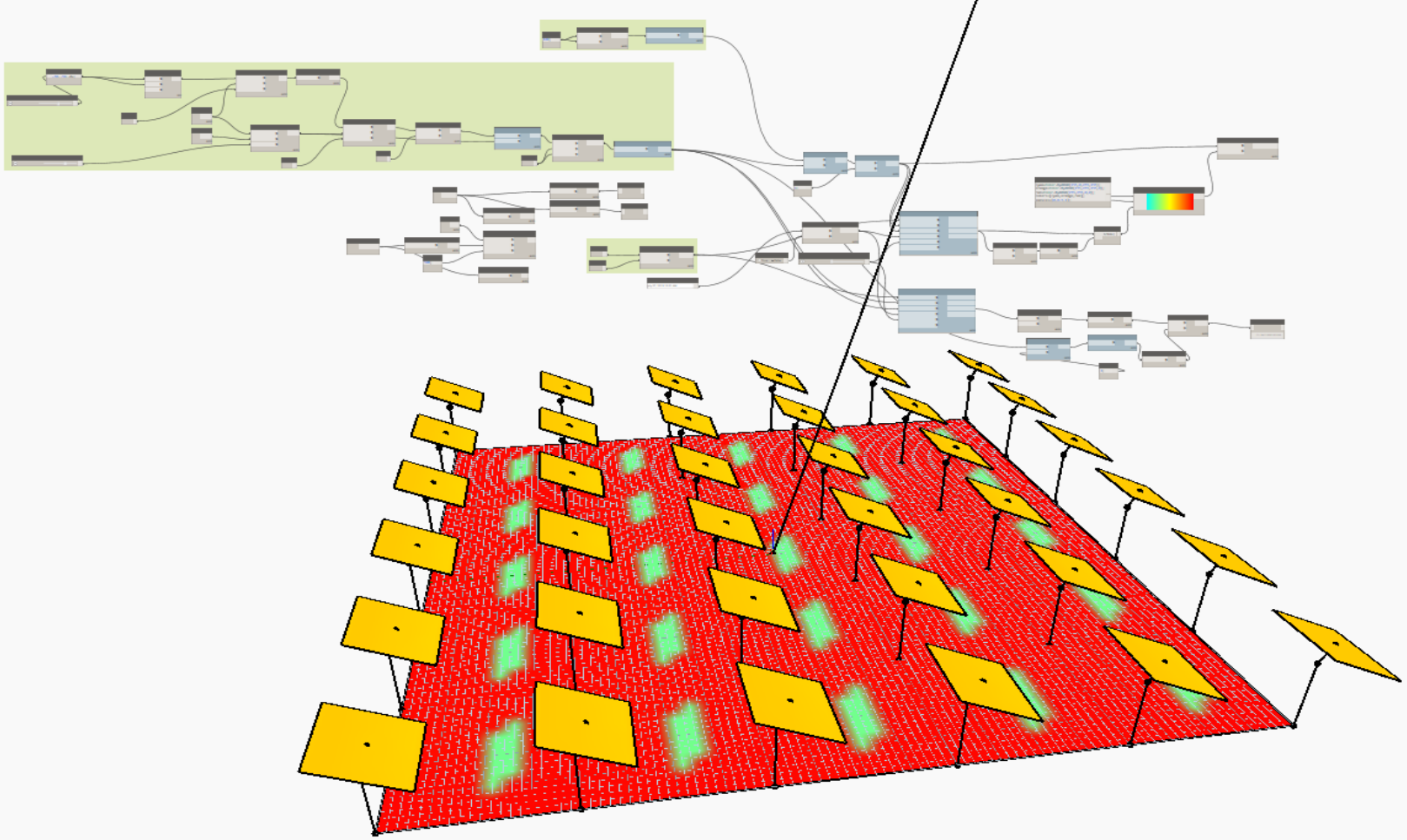

FIND SURFACES THAT ARE ANGLED TOWARDS VIEWER
PANELS AND SIGHTLINES
ANALYSE AND DISPLAY
Calculate volume outside zoning boundary

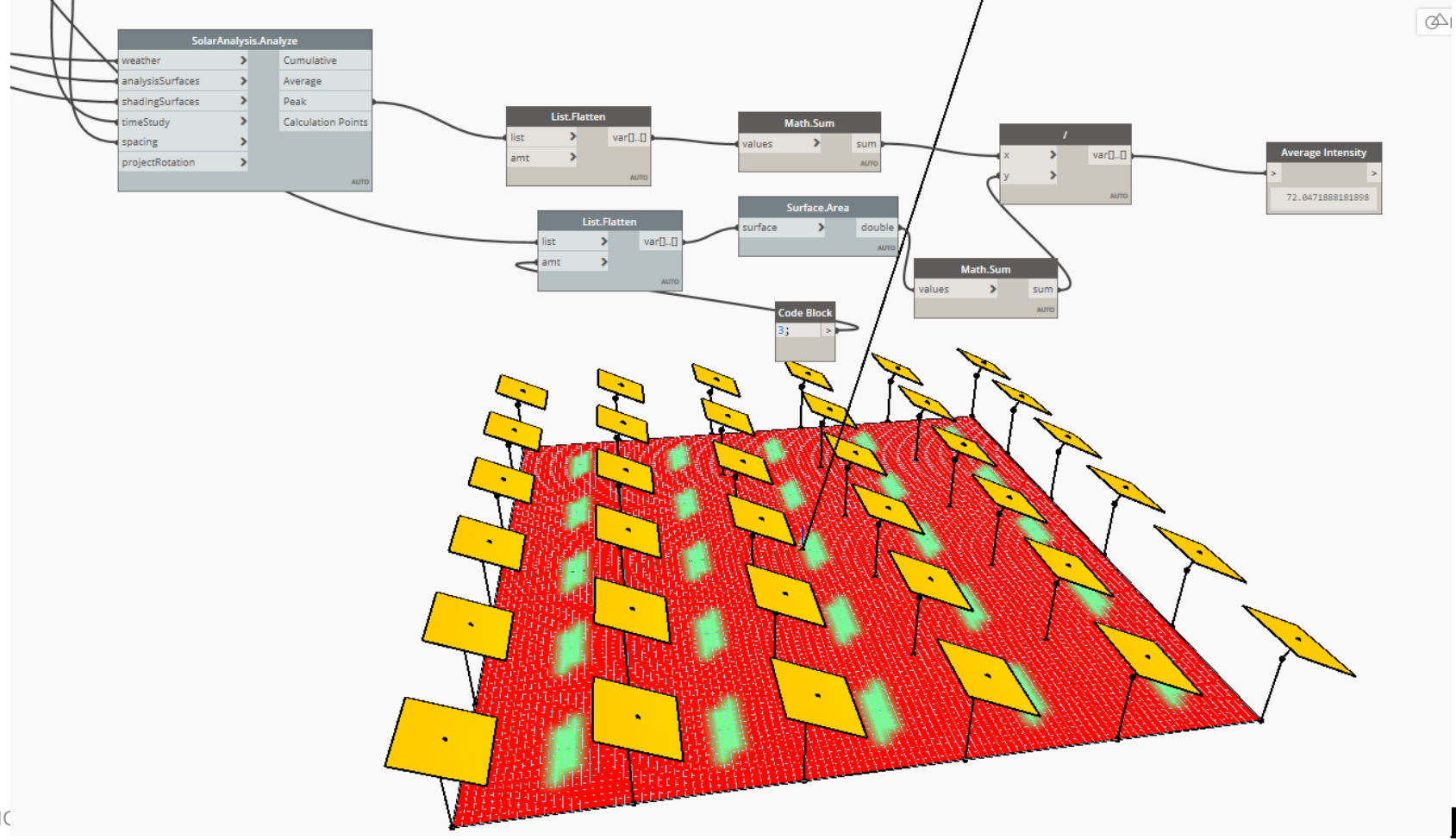
StealthyRoofScape Evaluator On Building Massing



Example 4

Solar Analysis

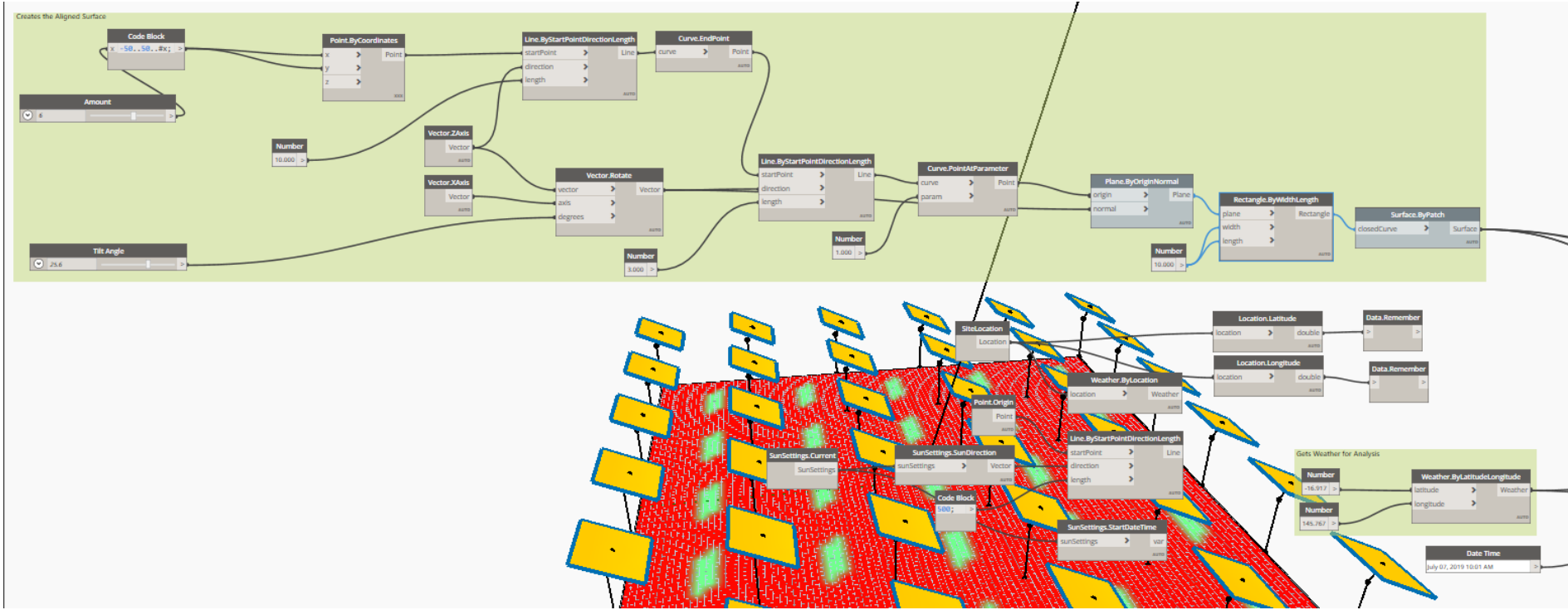




DIC

EDINBURGH

10 – 12 OCT 2019



DIGITAL BUILT WEEK EUROPE

EDINBURGH

10 – 12 OCT 2019

Refinery

Studies

New Study

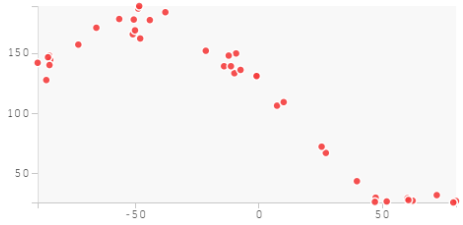
Study		
880179c6-5845-4e5d-882...	10/10	✓
d8feb305-4f4f-44a0-82b2...	40/40	✓

Y-Axis
Average Intensity

X-Axis
Tilt Angle

Size
Default

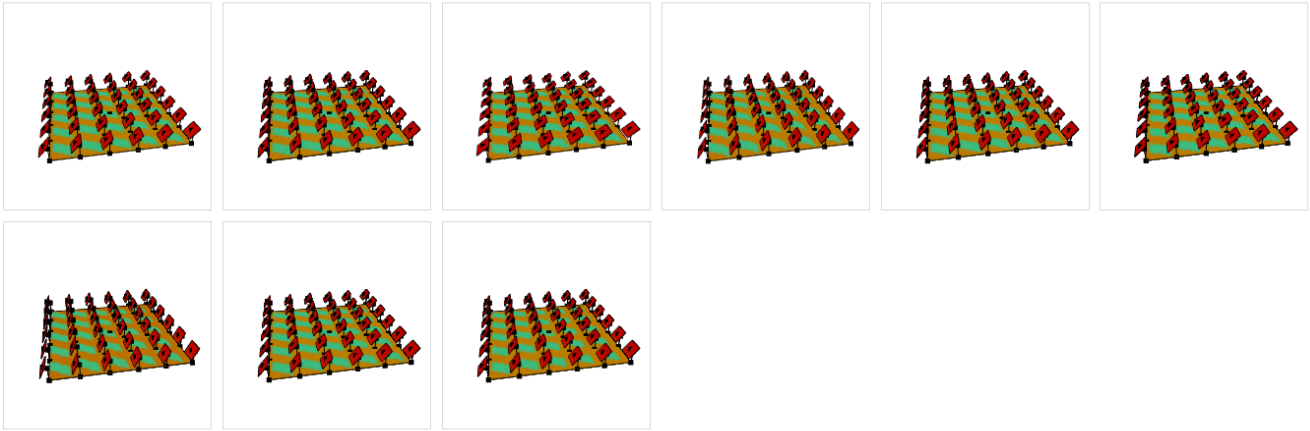
Color
Default

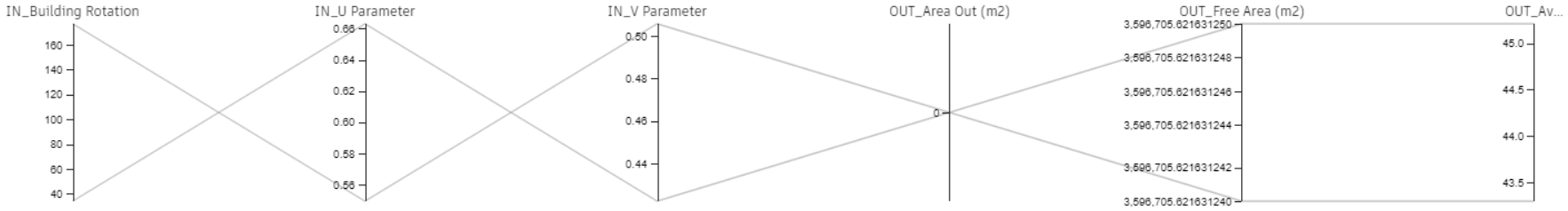


Filter

Sort by Average Intensity

1 | 2 | 3 | 4 | 5

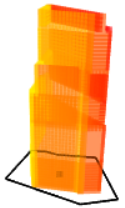




Filter

Sort by OUT_Avg. kWh/m2

1



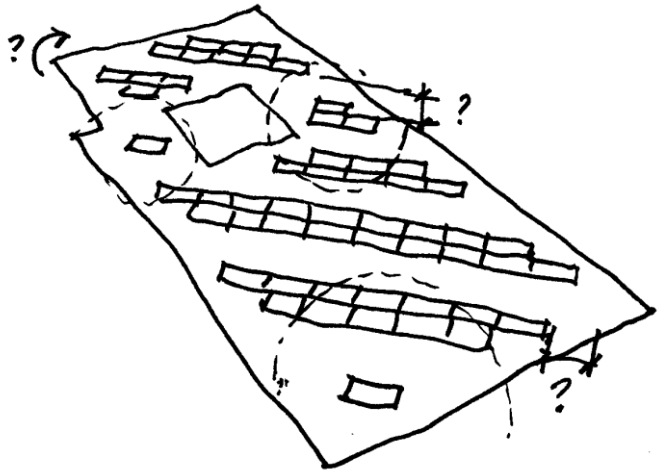
Example 5

Office Layout

Goals

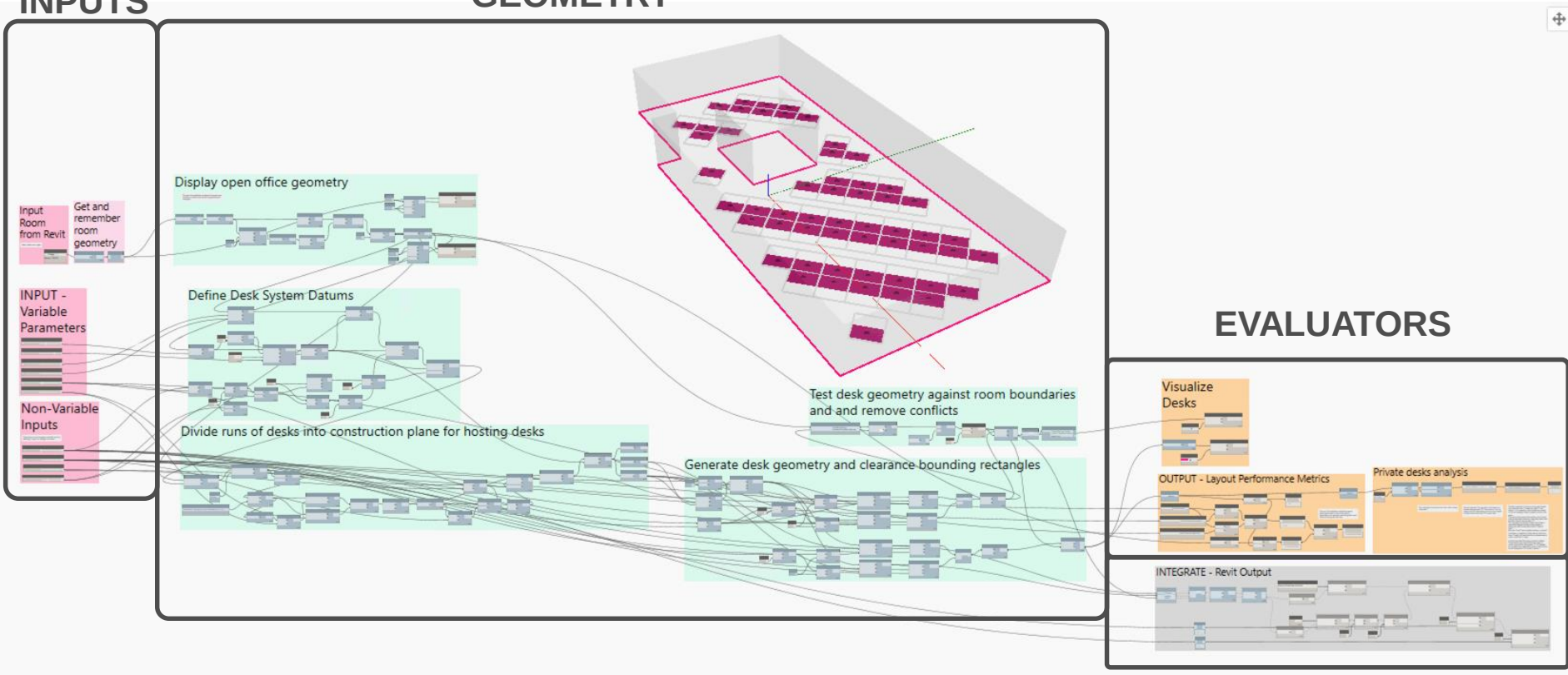
What are we solving for?

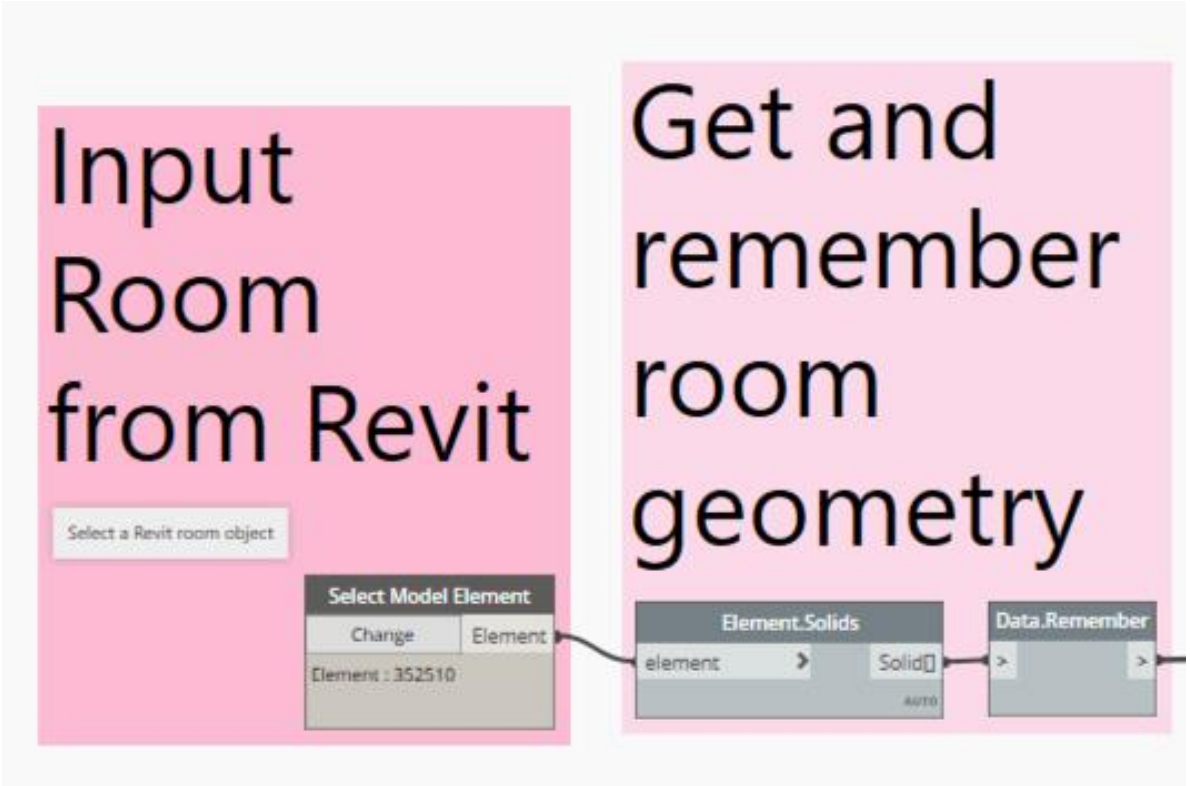
- Look at desk layouts that gives us the most desks but also the most private desks while maximizing desk area per person and minimizing unutilized space.
- Variable Inputs:
  1. Edge to start layout
  2. Desk rotation
  3. Position on room surface
  4. Desk clearance
  5. End offset
- Goals:
  1. Maximize number of desks
  2. Maximize desk area per person
  3. Maximize space utilization
  4. Maximize number of private desks



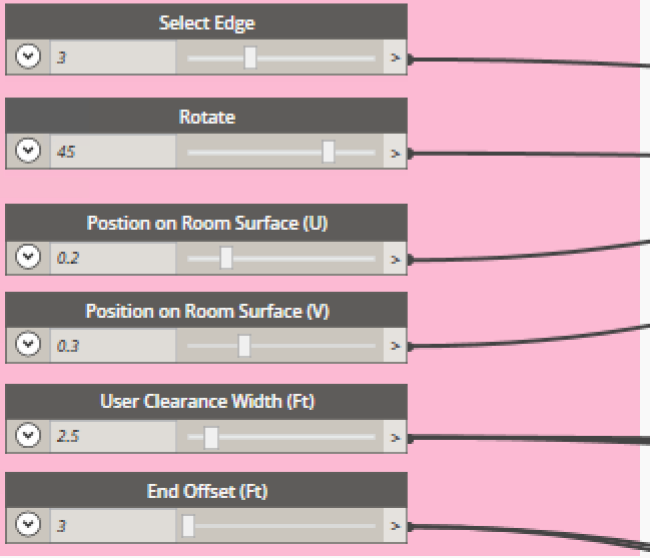
INPUTS

GEOMETRY

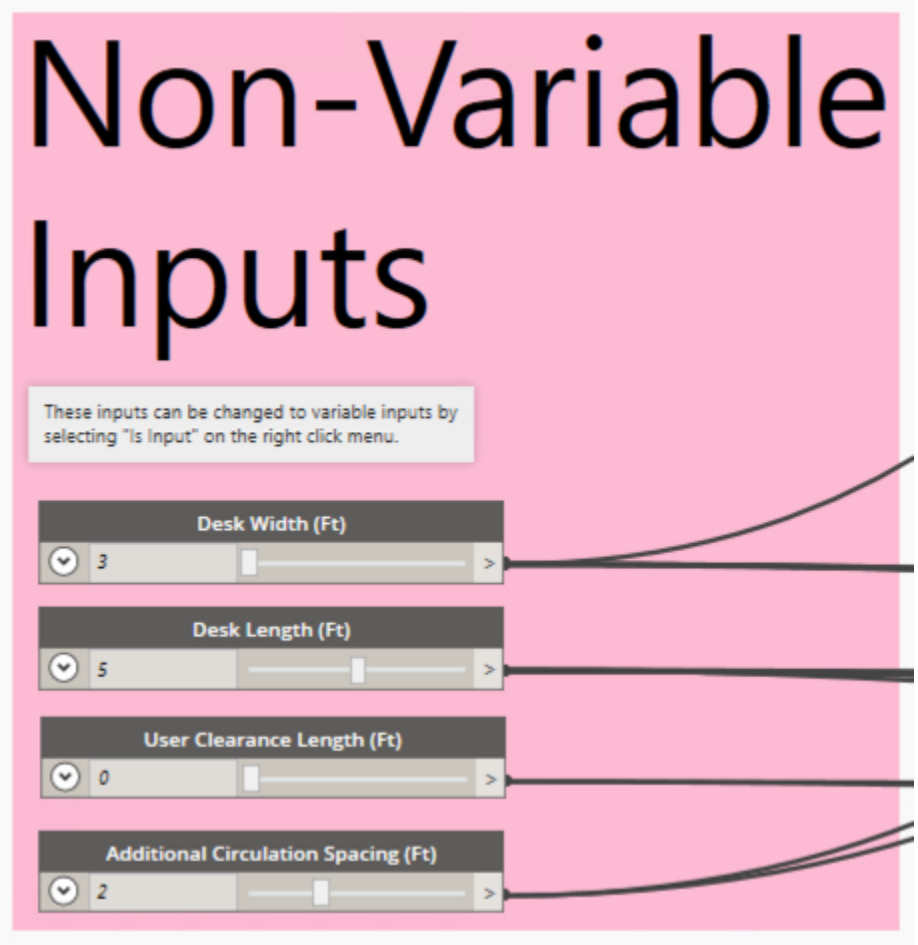


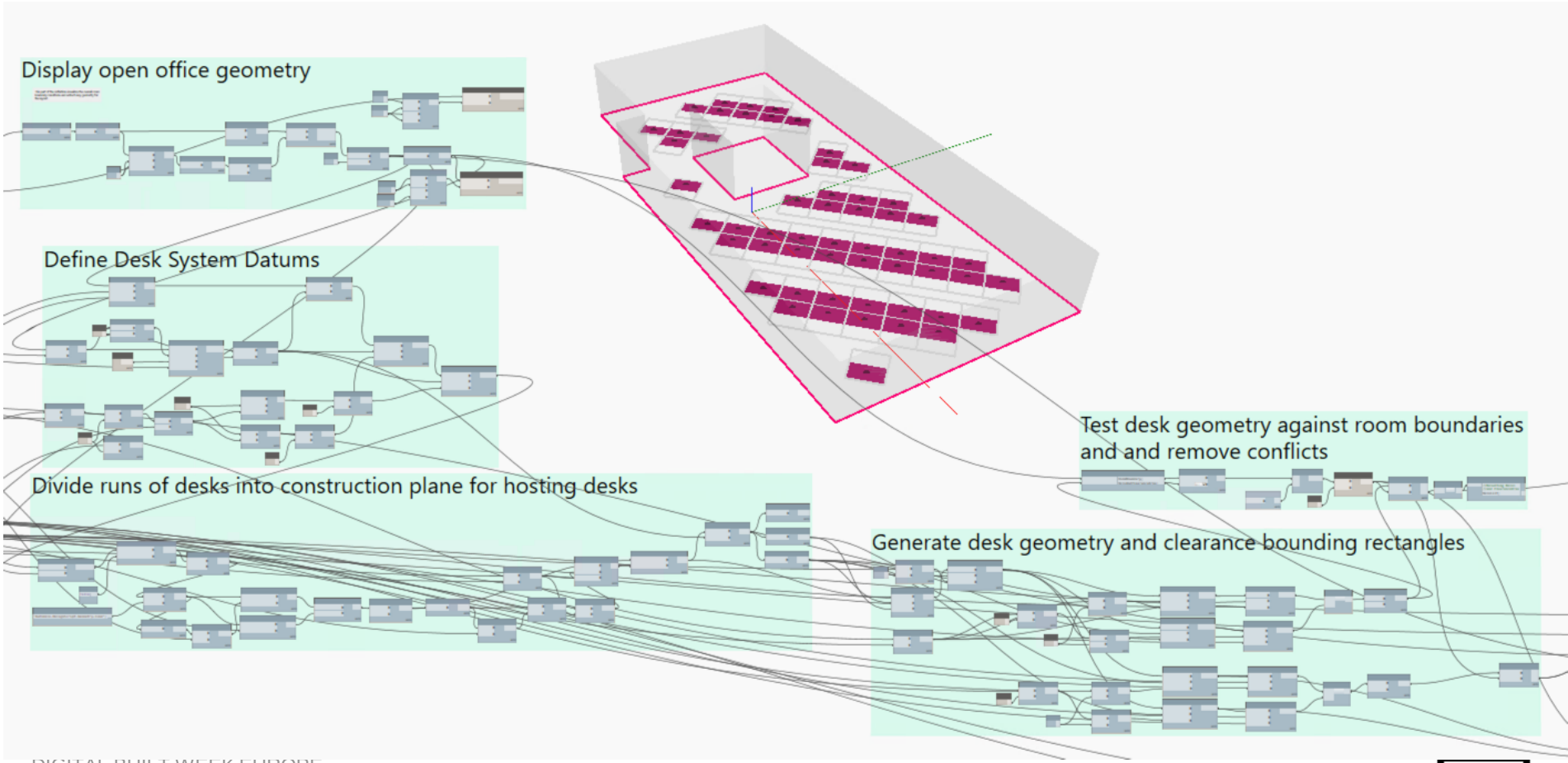

Input Room from Revit
Select a Revit room object
Select Model Element
Change
Element
Element : 352510
Get and remember room geometry
Element.Solids
element
Solid[]
AUTO
Data.Remember

INPUT - Variable Parameters

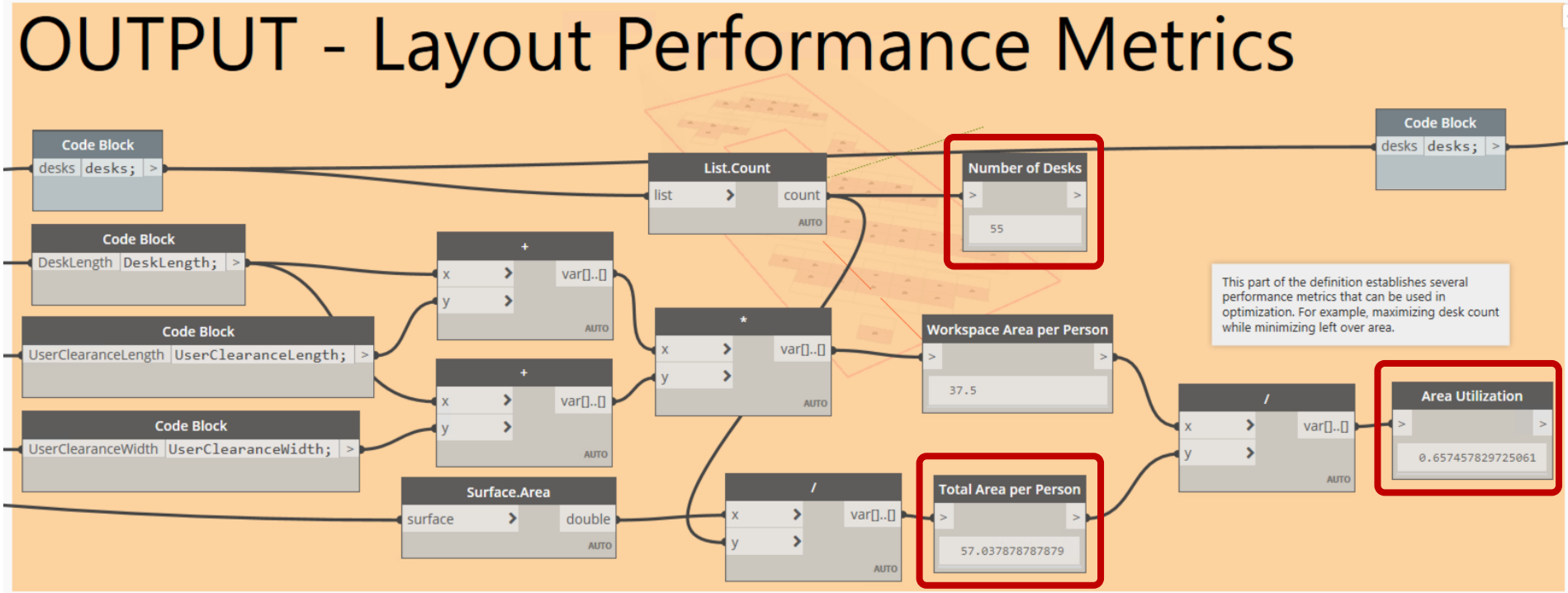


Non-Variable Inputs

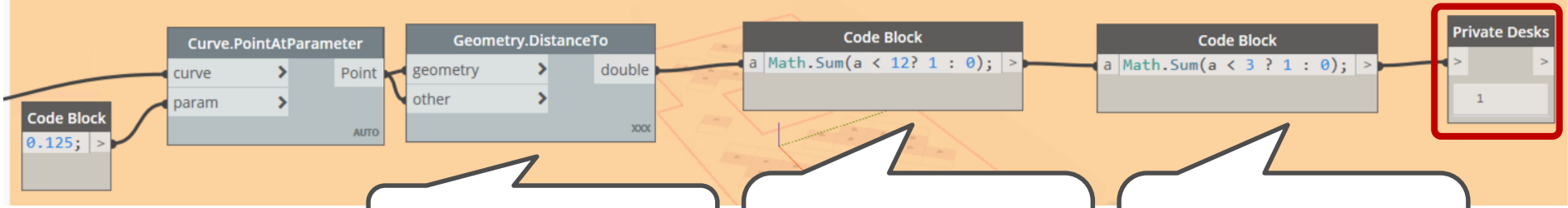



Display open office geometry
Define Desk System Datums
Divide runs of desks into construction plane for hosting desks
Test desk geometry against room boundaries and and remove conflicts
Generate desk geometry and clearance bounding rectangles

OUTPUT - Layout Performance Metrics



Private desks analysis

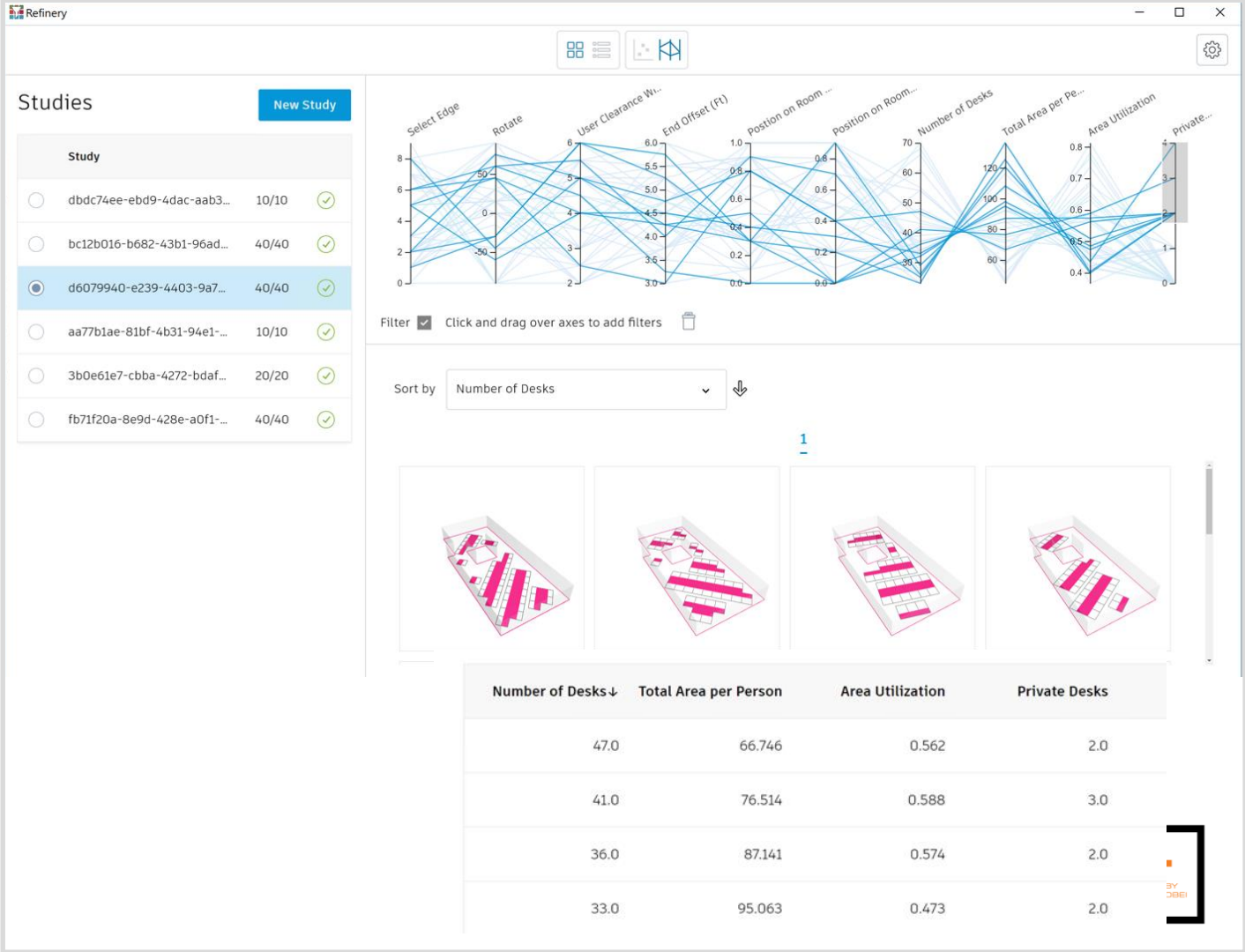


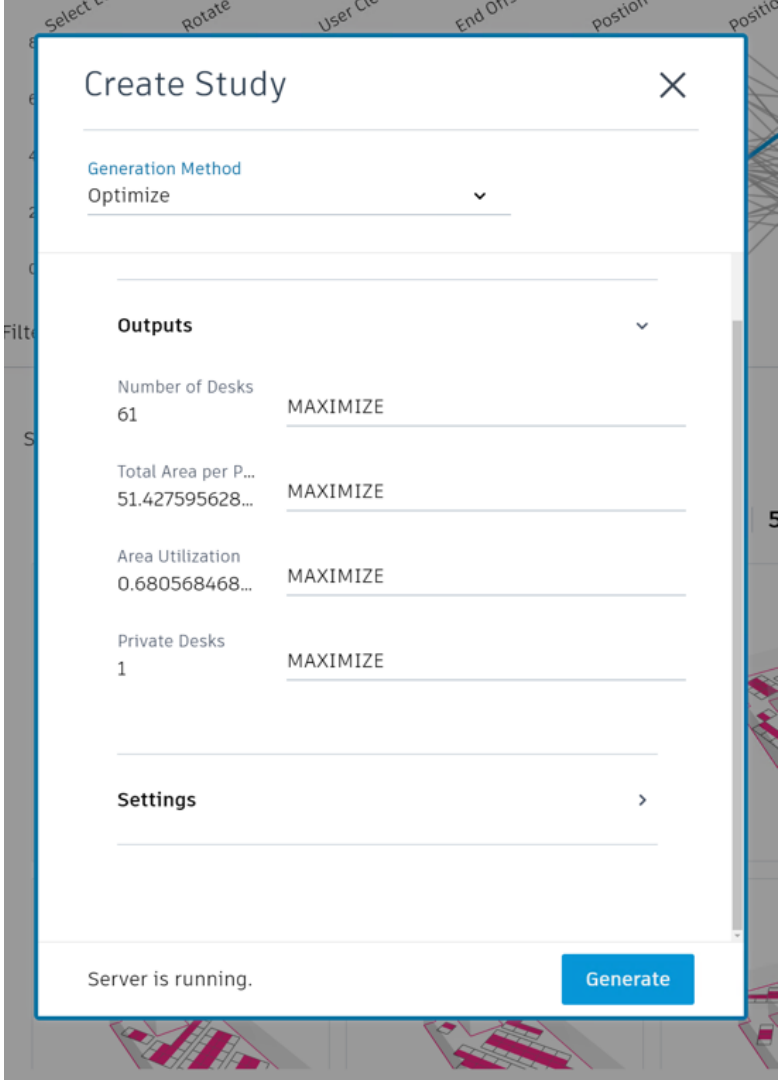
How far is it from desk to desk?

How many desks are less than 12' away from each other?

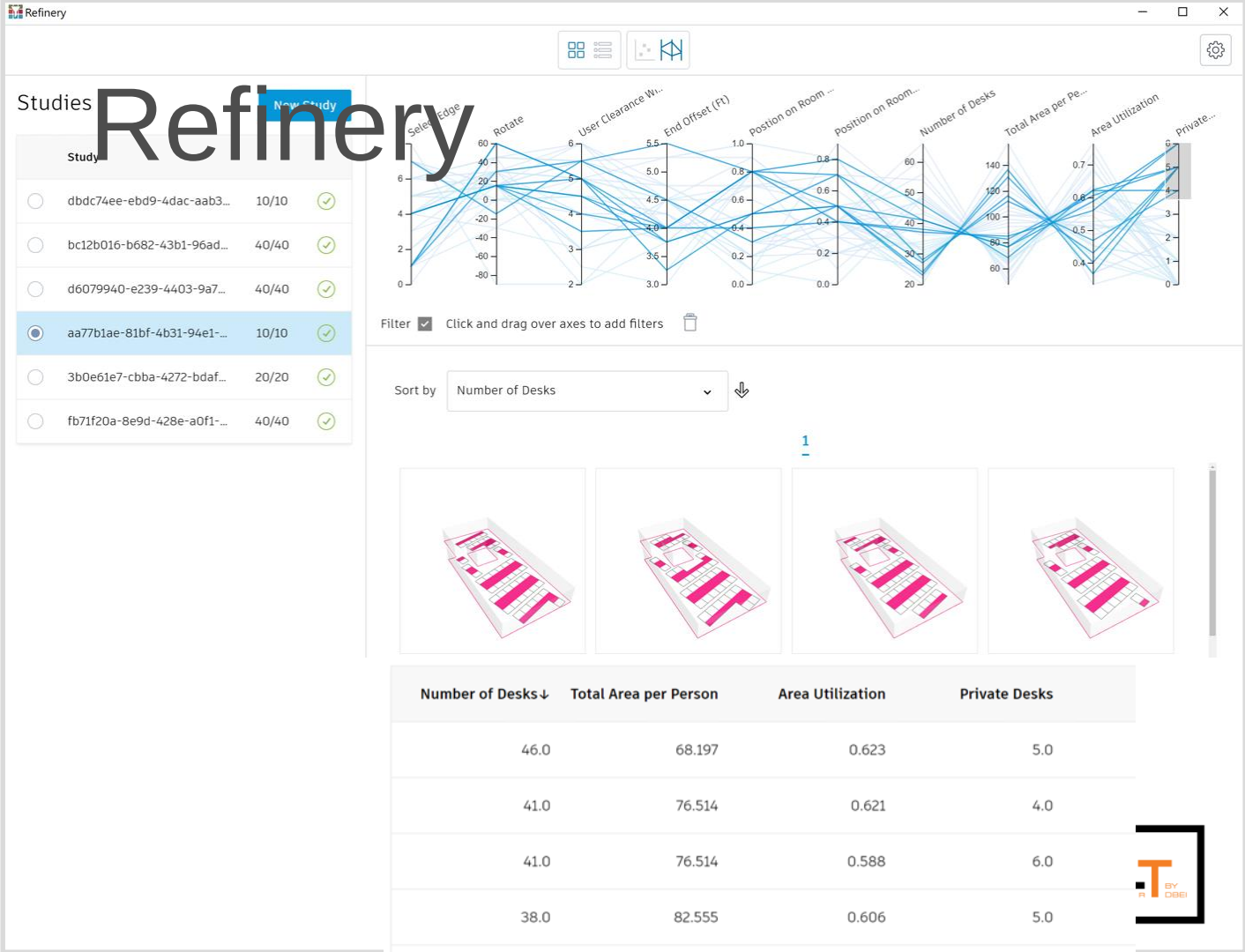
How many desks have less than 3 desks within 12'?

- 40 Random runs
- Most Private Desks
- Most Desks




Create Study
Generation Method
Optimize
Outputs
Number of Desks
61
MAXIMIZE
Total Area per P...
51.427595628...
MAXIMIZE
Area Utilization
0.680568468...
MAXIMIZE
Private Desks
1
MAXIMIZE
Settings
Server is running.
Generate

- 10x20 Optimization
- Most Private Desks
- Most Desks

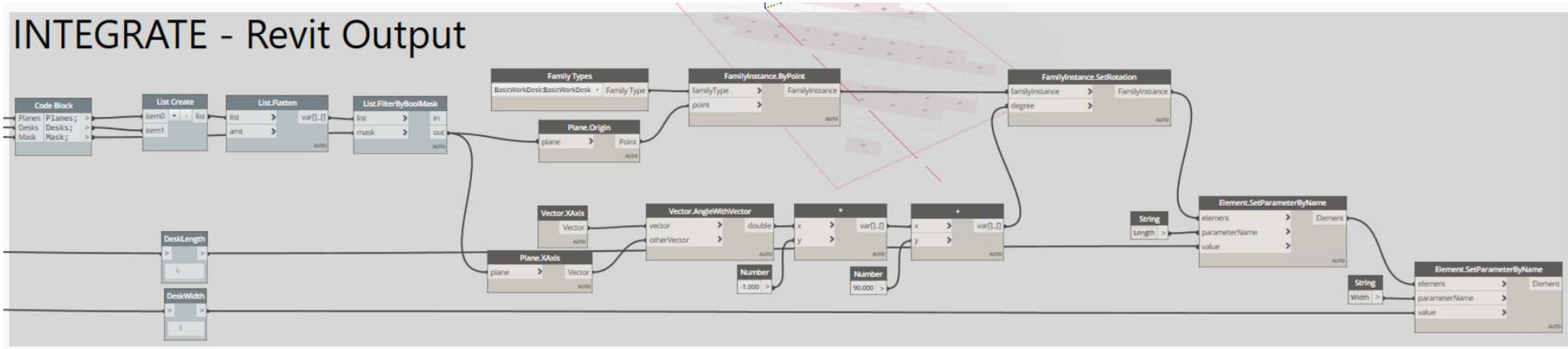


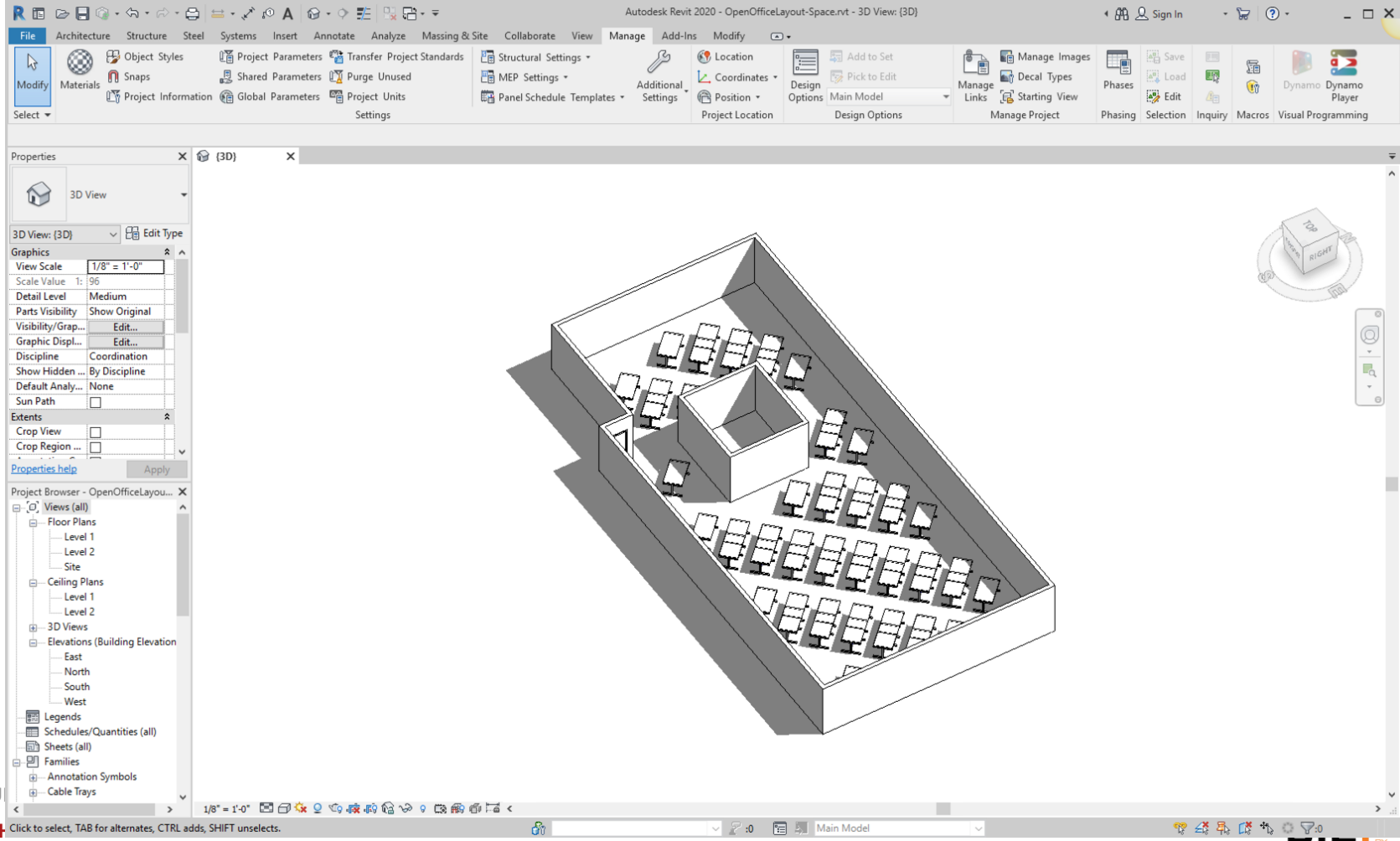
DIGITAL BUILT WEEK EUROPE

EDINBURGH

10 – 12 OCT 2019

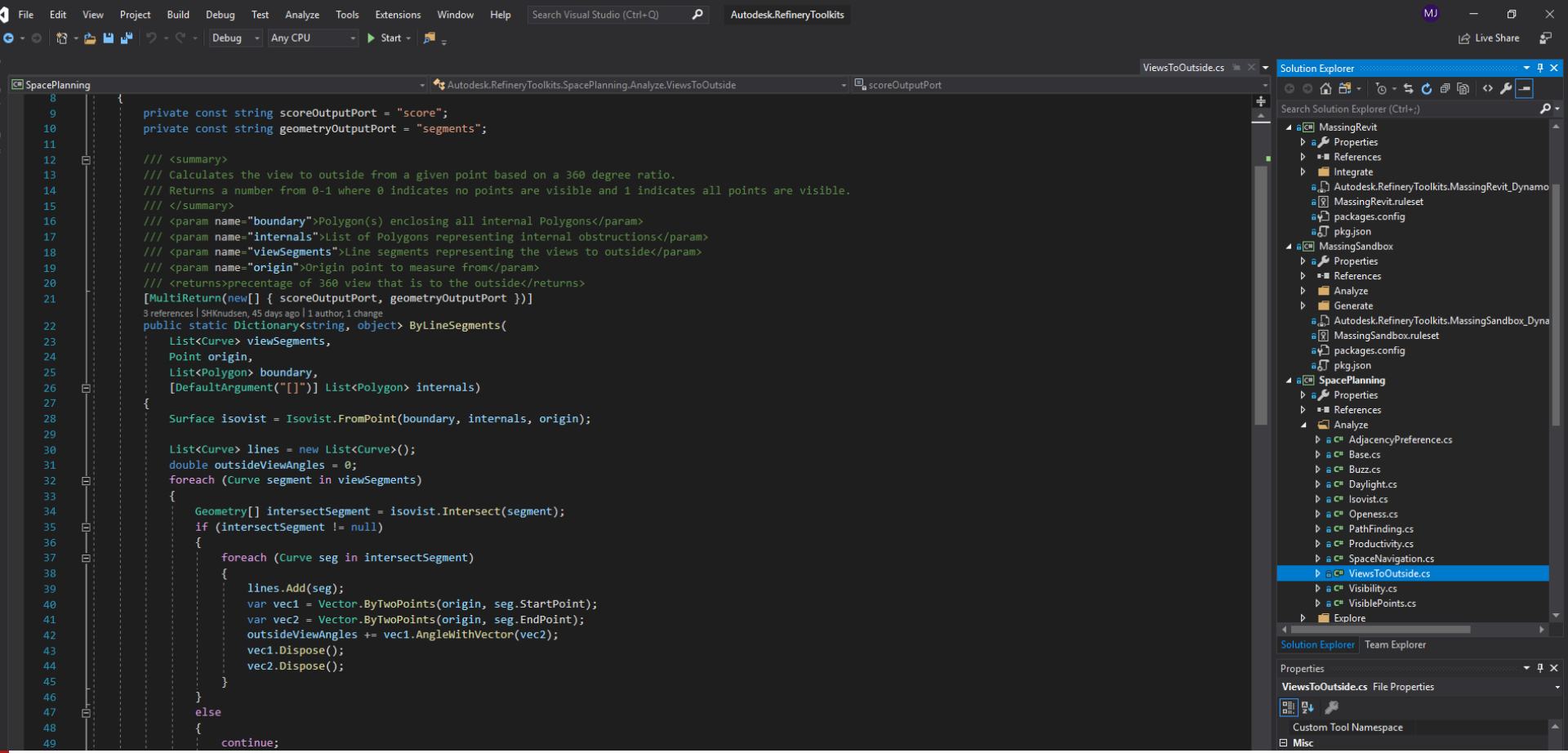
INTEGRATE - Revit Output




Autodesk Revit 2020 - OpenOfficeLayout-Space.rvt - 3D View: {3D}
File
Architecture
Structure
Steel
Systems
Insert
Annotate
Analyze
Massing & Site
Collaborate
View
Manage
Add-Ins
Modify
Modify
Materials
Object Styles
Snaps
Project Information
Project Parameters
Shared Parameters
Global Parameters
Transfer Project Standards
Purge Unused
Project Units
Structural Settings
MEP Settings
Panel Schedule Templates
Additional Settings
Settings
Location
Coordinates
Position
Project Location
Design Options
Add to Set
Pick to Edit
Main Model
Design Options
Manage Links
Manage Images
Decal Types
Starting View
Manage Project
Phases
Phasing
Save
Load
Edit
Selection
Inquiry
Macros
Dynamo
Dynamo Player
Visual Programming
Properties
3D View
3D View: {3D}
Edit Type
Graphics
View Scale
1/8" = 1'-0"
Scale Value 1:
96
Detail Level
Medium
Parts Visibility
Show Original
Visibility/Grap...
Edit...
Graphic Displ...
Edit...
Discipline
Coordination
Show Hidden ...
By Discipline
Default Analy...
None
Sun Path
Extents
Crop View
Crop Region ...
Properties help
Apply
Project Browser - OpenOfficeLayou...
Views (all)
Floor Plans
Level 1
Level 2
Site
Ceiling Plans
Level 1
Level 2
3D Views
Elevations (Building Elevation
East
North
South
West
Legends
Schedules/Quantities (all)
Sheets (all)
Families
Annotation Symbols
Cable Trays
{3D}
1/8" = 1'-0"
Click to select, TAB for alternates, CTRL adds, SHIFT unselects.
Main Model

Example 7 – Overflow Samples

Refinery Primer Samples


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
Autodesk.RefineryToolkits
Debug
Any CPU
Start
Live Share
ViewsToOutside.cs
SpacePlanning
Autodesk.RefineryToolkits.SpacePlanning.Analyze.ViewsToOutside
scoreOutputPort
private const string scoreOutputPort = "score";
private const string geometryOutputPort = "segments";
/// <summary>
/// Calculates the view to outside from a given point based on a 360 degree ratio.
/// Returns a number from 0-1 where 0 indicates no points are visible and 1 indicates all points are visible.
/// </summary>
/// <param name="boundary">Polygon(s) enclosing all internal Polygons</param>
/// <param name="internals">List of Polygons representing internal obstructions</param>
/// <param name="viewSegments">Line segments representing the views to outside</param>
/// <param name="origin">Origin point to measure from</param>
/// <returns>precentage of 360 view that is to the outside</returns>
[MultiReturn(new[] { scoreOutputPort, geometryOutputPort })]
3 references | SHKnudsen, 45 days ago | 1 author, 1 change
public static Dictionary<string, object> ByLineSegments(
List<Curve> viewSegments,
Point origin,
List<Polygon> boundary,
[DefaultArgument("[]")] List<Polygon> internals)
{
Surface isovist = Isovist.FromPoint(boundary, internals, origin);
List<Curve> lines = new List<Curve>();
double outsideViewAngles = 0;
foreach (Curve segment in viewSegments)
{
Geometry[] intersectSegment = isovist.Intersect(segment);
if (intersectSegment != null)
{
foreach (Curve seg in intersectSegment)
{
lines.Add(seg);
var vec1 = Vector.ByTwoPoints(origin, seg.StartPoint);
var vec2 = Vector.ByTwoPoints(origin, seg.EndPoint);
outsideViewAngles += vec1.AngleWithVector(vec2);
vec1.Dispose();
vec2.Dispose();
}
}
else
{
continue;
Solution Explorer
Search Solution Explorer (Ctrl+;)
MassingRevit
Properties
References
Integrate
Autodesk.RefineryToolkits.MassingRevit_Dynamo
MassingRevit.ruleset
packages.config
pkg.json
MassingSandbox
Properties
References
Analyze
Generate
Autodesk.RefineryToolkits.MassingSandbox_Dyna
MassingSandbox.ruleset
packages.config
pkg.json
SpacePlanning
Properties
References
Analyze
AdjacencyPreference.cs
Base.cs
Buzz.cs
Daylight.cs
Isovist.cs
Openess.cs
PathFinding.cs
Productivity.cs
SpaceNavigation.cs
ViewsToOutside.cs
Visibility.cs
VisiblePoints.cs
Explore
Solution Explorer
Team Explorer
Properties
ViewsToOutside.cs File Properties
Custom Tool Namespace
Misc

A SPECIAL THANKS TO:

ENSTOA

proving ground

DBEI DIGITAL BUILT ENVIRONMENT INSTITUTE

designtech.io

AUTODESK. RESEARCH

BUILDING CONTENT SUMMIT by DBEI

BiLT EUR BY DBEI

REMINDER:

Speaker Feedback is appreciated

Fill in the Survey on the Event App

DIGITAL BUILT ENVIRONMENT INSTITUTE

Reference Links

- https://refineryprimer.dynamobim.org/
  - https://refineryprimer.dynamobim.org/02-getting-started/02-01_anatomy-of-a-good-generative-design-process
  - https://refineryprimer.dynamobim.org/06-using-revit-alongside-refinery/06-03_detailed-example-workflow
- https://github.com/DynamoDS/RefineryPrimer
- https://www.keanw.com/
  - https://www.keanw.com/2019/06/project-rediscover-is-now-available-for-download.html
- https://dynamobim.org/math-transit/ 2d point plotting
- https://scipy-lectures.org/advanced/mathematical_optimization/index.html
- https://en.wikipedia.org/wiki/Numerical_differentiation
- https://aroberge.blogspot.com/2005/04/computing-derivatives-using-python.html

One on One Meeting

I will be at the **Speaker Lounge** in the Exhibition Hall for further conversations. Please join me there at this time:

Friday 11 October

12:35pm-1:35pm (Lunch)

4:15pm-4:45pm (Afternoon tea)

REMINDER

Session materials are available on the Conference App

Session 2.3 2.4 How to Train Your Model with Project Refinery

Matt Jezyk

Questions?

How to Train Your Model with Project Refinery

Matt Jezyk, Senior Staff Software Engineer, Tesla

Formerly Senior Engineering Manager
Autodesk AEC Generative Design

@MattJezyk

DIGITAL BUILT ENVIRONMENT INSTITUTE

BUILDING CONTENT SUMMIT by DBEI

BILT EUR BY DBEI

Example 10 – Overflow Samples

Autodesk MaRS Office

Example 11 – Overflow Samples

Tiling

Example 12 – Overflow Samples

Refinery Beta Samples