Session 2.3 2.4

How to Train Your Model with Project Refinery

Matt Jezyk, Tesla (formerly Autodesk)

Class Description

Project Refinery is an Autodesk generative design tool for the architecture, engineering and construction industry that gives users the ability to explore, analyze, and optimize their Dynamo and Revit designs.

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. This lab will feature pragmatic end-to-end workflows for architectural design option generation and evaluation using Project Refinery in Dynamo for Revit. We will teach you how to code 'Evaluators' and train your model using Project Refinery.

About the Speaker:

Matt Jezyk works as the Sr. Staff Software Engineer at Tesla in the group that designs and builds the 'Gigafactories'. He develops software to smooth out the design, fabrication and construction process. Matt's group is responsible for developing smart factories for upcoming Tesla products.

Prior to his current role, Matt was the Senior Engineering Manager for AEC Generative Design at Autodesk. He has been in the AEC industry for 23+ years and has spent the past 20 years developing Autodesk Revit and various other design tools. Matt helped build Revit Architecture and Revit Structure and led the team at Autodesk responsible for developing Dynamo (computational design) and Project

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Refinery (optimization and generative design). Dynamo is now being used around the world and has an amazing community of users who now teach these techniques at many events and conferences.

Matt's group also has explored new ways of building by connecting computational and generative design directly to digital fabrication tools and robots. Examples of this have been shown in the BUILD space in Boston and presented at conferences like ACADIA, SmartGeometry and Robots in Architecture

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Session Outline

Project Refinery is an Autodesk generative design tool for the architecture, engineering and construction industry that gives users the ability to explore, analyze, and optimize their Dynamo and Revit designs.

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. This lab will feature pragmatic end-to-end workflows for architectural design option generation and evaluation using Project Refinery in Dynamo for Revit. We will teach you how to code 'Evaluators' and train your model using Project Refinery.

We will focus on these workflows and focus on the Evaluators for each:

1. **3Box Massing Study Simple** – a simple workflow to explain concepts.
2. **2d Hill Climbing** – showing how to create define a goal or objective function and create Evaluators using either Dynamo nodes, DesignScript, Python or C# code
3. **Massing Study Complex** - retail and office distribution and configuration for a building on an urban site.
    a. We will also show how to take an evaluator from a different project (**StealthyRoofscapes**) and apply it to this one.
4. **Solar Analysis** – Using Simulation Results
5. **Office furniture layout** – desk layouts in a Revit room that provide the most desks but also the most private desks while maximizing desk area per person and minimizing unutilized space.

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

2.3 2.4 - How to Train Your Model with Project Refinery

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Header Style

Introduction

This class is about how to use generative design workflows to advance architectural processes.

Why learn about generative design workflows?

By 2050 there will be 10 billion people on earth. The global middle class is growing faster than at any time in our history. In 1990, only 23% of the global population had sufficient income to be classified as middle class. Today, 45% of us do. All of these people are going to drastically increase the population of our cities and this is going to require a lot of work: a lot of new buildings and infrastructure to support them. Recently Autodesk partnered with a firm called Statista to calculate just how much work. We're going to have to build 13,000 buildings every day and enough roads and rail to wrap around the earth six times every year. That's over 1.2 million km of roads and rail needed for our global infrastructure each and every year between now and 2050. I'm going to assert that there is a problem: *we need to produce more work at a higher quality with less resources*. In today's design processes, most people just evaluate a few designs and make decisions based on rules or thumb or instinct, generative design can let you study much more data driven designs to achieve better outcomes.

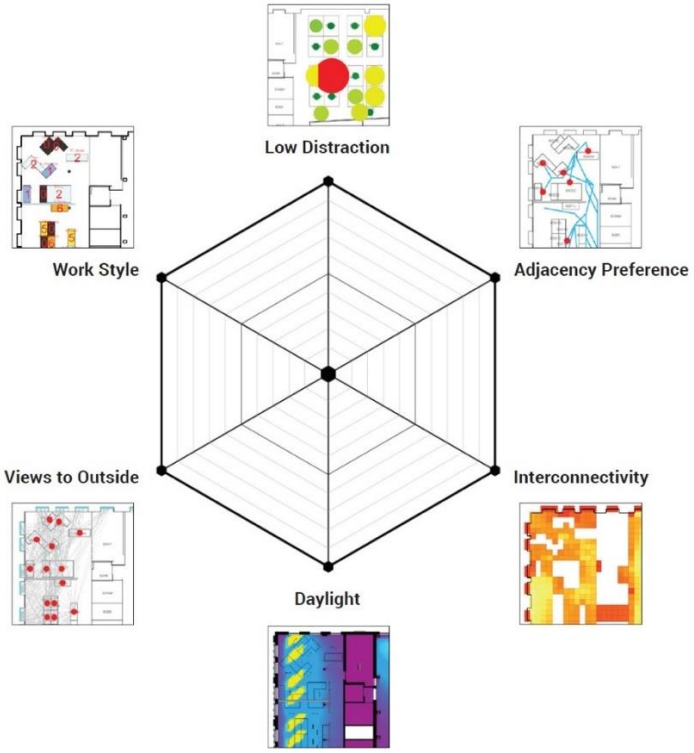
The traditional way of delivering buildings has been recording designs on paper and delivering the drawings to others who execute to build it. We've evolved to computerizing drawings making them more data rich and easier to update. But what we really want to do now is to combine intelligence of building professional with algorithmic intelligence of a computer to produce better buildings, faster. Autodesk Research has been experimenting with some ways to do this. The following studies are some examples of their work.

Research Example 1: Autodesk MaRs Office Design in Toronto

A few years ago, Autodesk moved our Toronto office to a new office in the MaRS innovation district. The Living, an architecture firm that Autodesk has incorporated into Autodesk Research to test out ideas, used a generative approach to design the new office's space layout. They began by surveying existing employees and asking them questions like "Who do you need to sit by?" "How much daylighting do you need to do

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

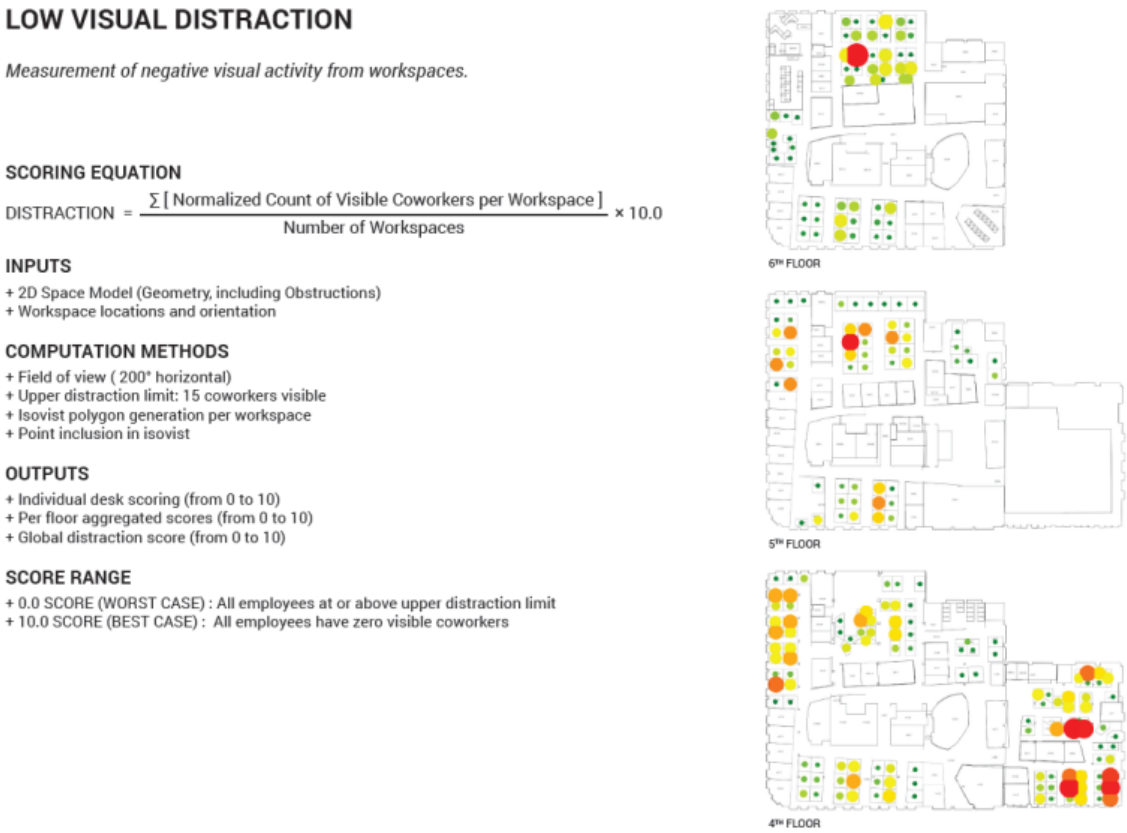
your job?" "How much distraction can you handle?" When they had all of this data they identified 6 goals that they felt would make a "good" office.



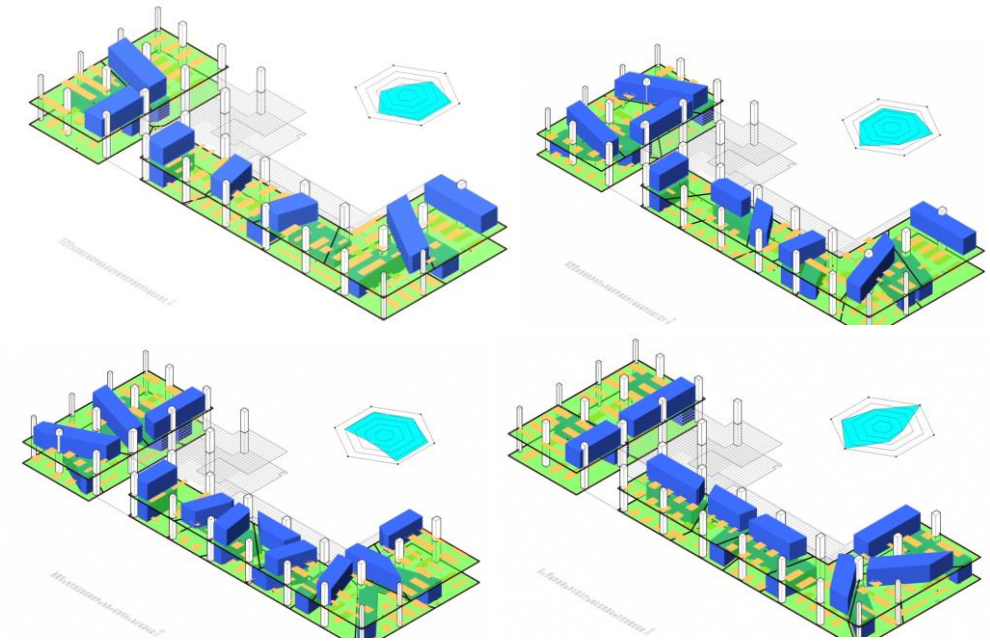
Autodesk MaRs office design goals

Algorithms were developed to measure these goals and tested the measurements on their existing office space. Here is an example of a way to measure low visual distraction. Note that the scoring equation is critical to the evaluation of the goals.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



A flexible model was built to generate designs that could be tested against the goals.



Autodesk Mars Office Design Project

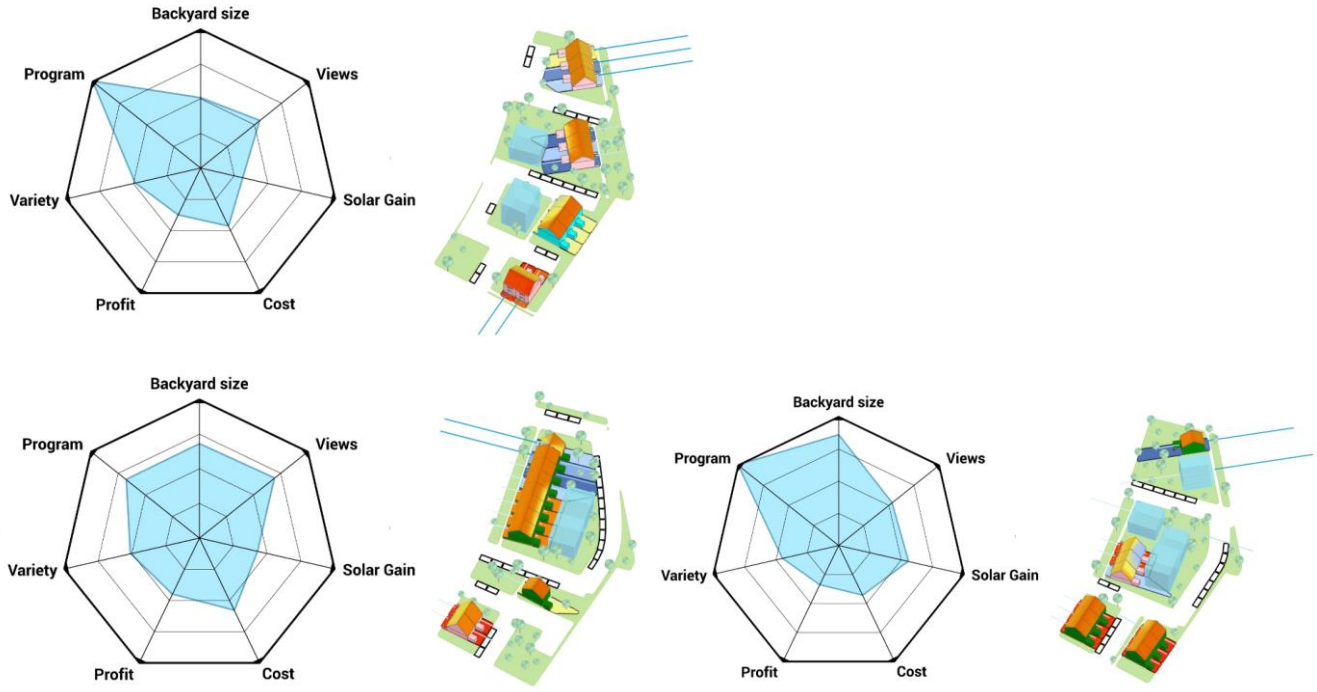
Different floor plan layouts were generated from the model including blue amenity bars of meeting rooms and yellow locations of desks. Each design option also feeds data into a dashboard showing performance across the six goals. Variables were set up so

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

that designers could flex the model and see what the effects of the variables are. 10s of 1000s of options were created and a methodology was developed to sort and help make decisions. Optimization algorithms were used to help evolve the designs towards identified goals. By clearly defining goals before the options were generated, the computer was able to assist the design team in producing not just more designs, but designs that better met the stated goals.

Research Example 2: Van Wijnen Neighborhood Design

Autodesk Research and The Living also worked on a project with the Dutch construction company Van Wijnen. In this project they followed a similar workflow where they identified certain goals, developed ways to measure those goals, developed a flexible model, and used the computer to help generate, evolve, and rank their designs.



Van Wijnen Neighborhood design project

"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

How can I use generative design workflows?

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

The tools that The Living used to do these studies were custom tools that were not available to the public at the time. Autodesk is interested in democratizing this methodology and in making them accessible to more people. Project Refinery is a beta application Autodesk is developing to help preform generative design workflows like these. Project Refinery:

- Automatically runs Dynamo created logic to create options
- Works on Dynamo for Revit or Dynamo Sandbox
- Created for one customer Van Wijnen in 2/2018
- Public Beta as of 11/2018
- Available at **autodesk.com/solutions/refinery-beta**

The outcome of an automatic design search is completely dependent on the process used to create the design space, as well as the specification of various measures and search parameters. So how do you create a good design space? The traditional architectural design process is similar to the generative design process in many ways. In the traditional architectural design process, you ask for requirements from your client – how big a building do you want? What kinds of spaces to you need in the building? What goals do you have for the design?

This traditional process of identifying goals and requirements is still important when incorporating generative design techniques. With generative design you usually can't design all aspects of a building in one automated process, you have to identify what kinds of problems might be good to look at in ways that the computer can help you to generate options and evaluate them, but you are still defining goals and measuring success. It's critical to design how the problem with be generated by the computer and how you will measure success. How are you going to make decisions on the results? The first step is good preparation:

- Understand what you want to achieve
- Define your problem
- Decide on ways to measure success
- Think about how to review the results

Traditional Design process is a little bit like a game of battleship. Ideas are tested one at a time. "Does this work?" "What about this?" You might get a hit, you might get a miss. Often times only a few designs can really be evaluated because you run out of time. Solutions may be presented based on cost only when you really want to show the

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

client more options. With generative design techniques, people and computers work together to created and evaluate more designs.

"The goal of generative design is not to automate the design process, or to replace human designers with artificial ones" – Danil Nagy, The Living

Example 1: Learning Resources

The examples for this class:

- https://github.com/tatlin/BiLT_NA_2019_How_To_Train_Your_Model

Refinery Primer (just released):

- https://refineryprimer.dynamobim.org/
- https://github.com/DynamoDS/RefineryPrimer

Refinery Toolkit Dynamo Package and Examples Files (still under development, but available for testing):

- https://github.com/DynamoDS/RefineryToolkits

Thank You To Our Partners:

- DesignTech/ENSTOA, ProvingGround, and Autodesk Research

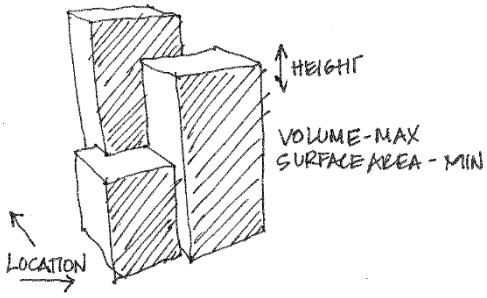
Example 1: Simple Massing Study

Let's see how some examples of design logic can be created in Dynamo and run through Refinery to learn about generative design workflows. If you are unfamiliar with Dynamo, you can learn more in this Dynamo Primer: https://primer.dynamobim.org/

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

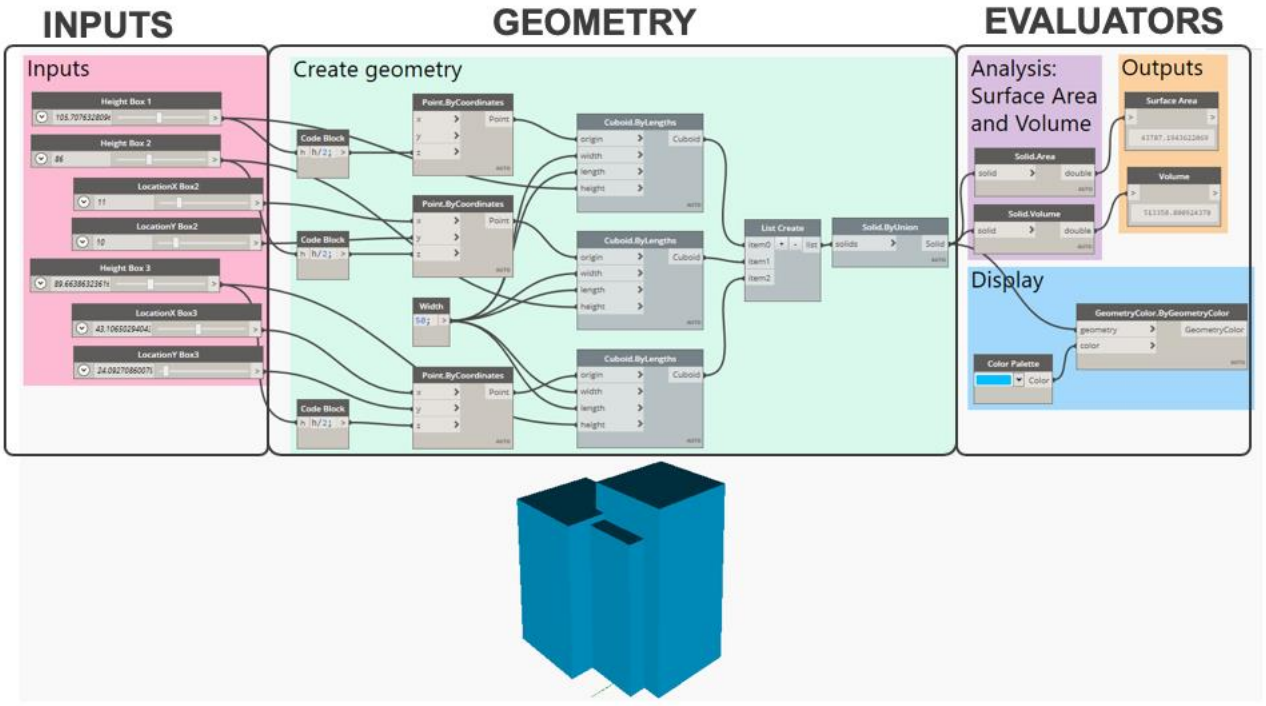
The first step with any generative design workflow is to identify what problem we are solving for. In this case we want to study a building mass to look at trade-offs between volume and surface area.

- Variable Inputs:
    1. Locations of 2 of the masses
    2. Heights of all 3 masses
- Goals:
    1. Maximum volume (rentable area)
    2. Minimum surface area (expensive façade)



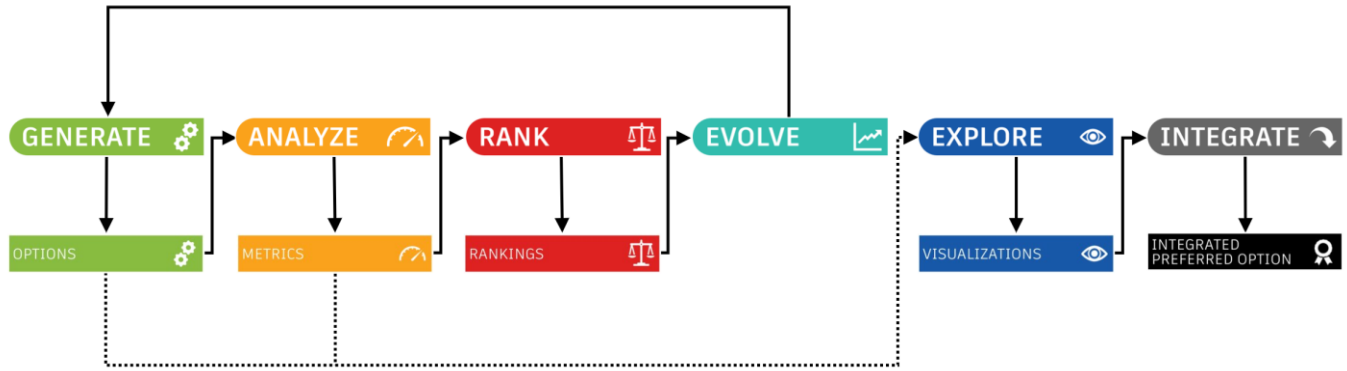
Open "\SampleFiles\ 01_3Box \ 3BoxVolumeSurfaceArea-SimpleBiLT.dyn" in Dynamo.

The Dynamo graph for this study is broken down into 3 sections:



Try moving the sliders in the Inputs section to understand how they are controlling the model. Next, look at the Geometry section to see how the geometry is being created with points and cuboid nodes. Finally, the Evaluator section contains nodes that measure surface area and volume, watches the outputs, and controls the display color of the geometry. There are several stages to the generative design workflow:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Generative design workflow

Let's look at some key concepts in each stage.

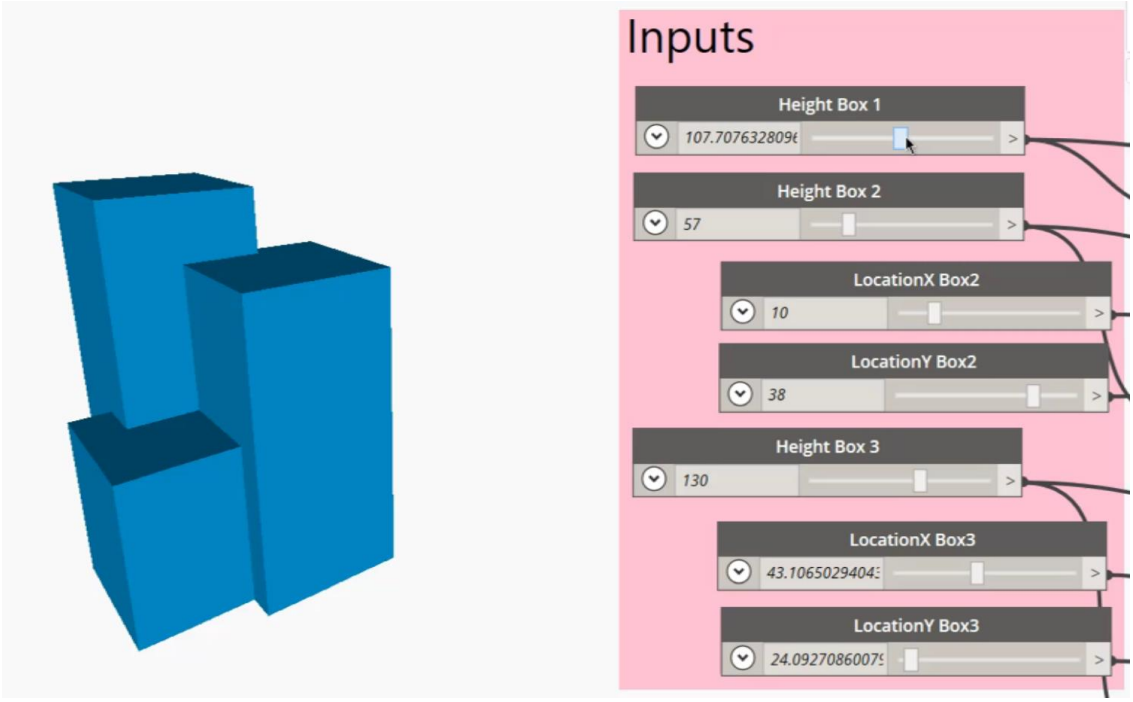
Generate

Create a flexible design system in Dynamo to generate designs.
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

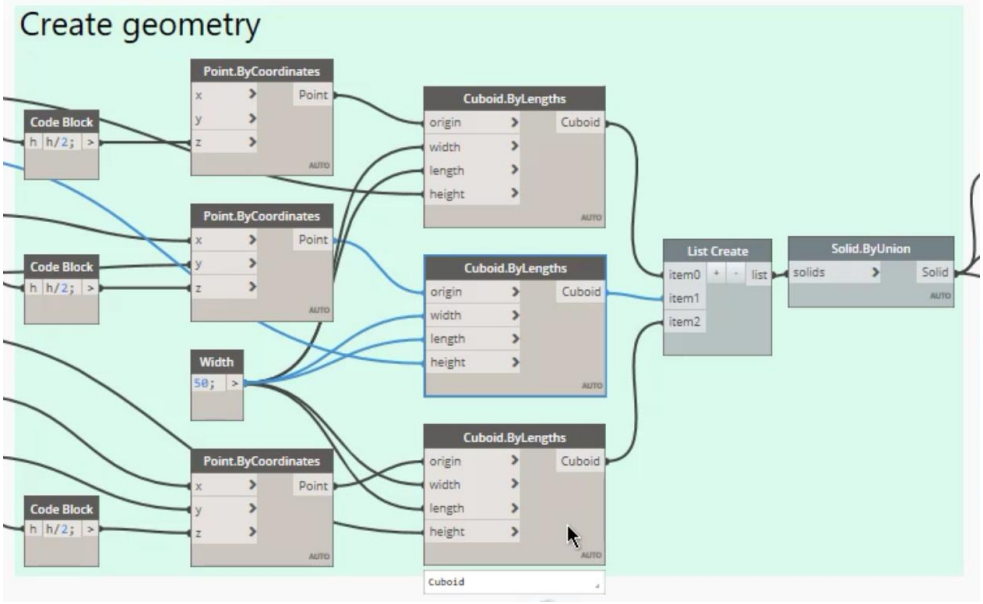
The flexible system in the 3BoxVolumeSurfaceArea-SimpleBiLT.dyn model is very simple. The Inputs in the pink group control point locations for the cuboids in the geometry section of green groups. As well as the cuboid locations, the inputs also control the height of the cuboids.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



3Box.dyn Inputs

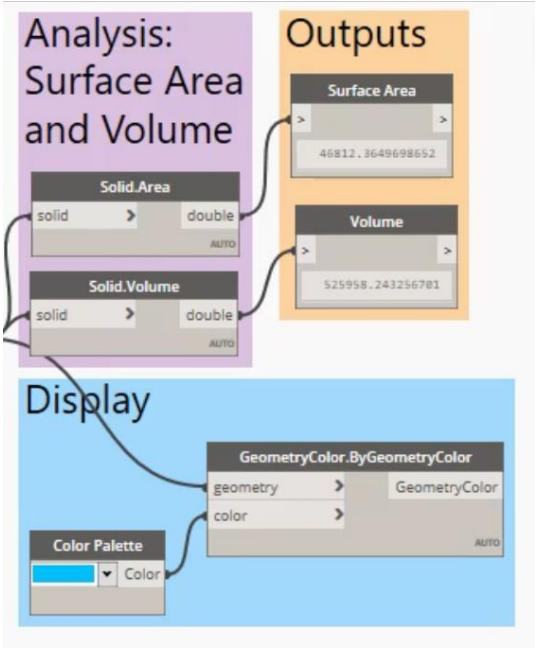
In the green geometry creation section, there are 3 points, that place 3 cuboids. Once placed, the 3 cuboids are unioned together to form a single piece of geometry.



Geometry creation section of 3Box.dyn

The next sections involve analyzing the resultant geometry for surface area and volume, watching the outputs, and controlling the display color of the geometry.

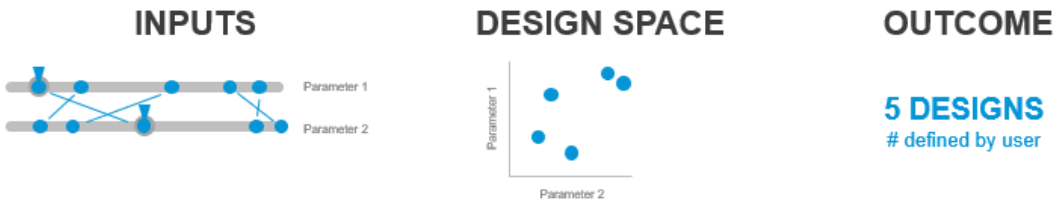
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Analysis, Display and Outputs of 3Box.dyn

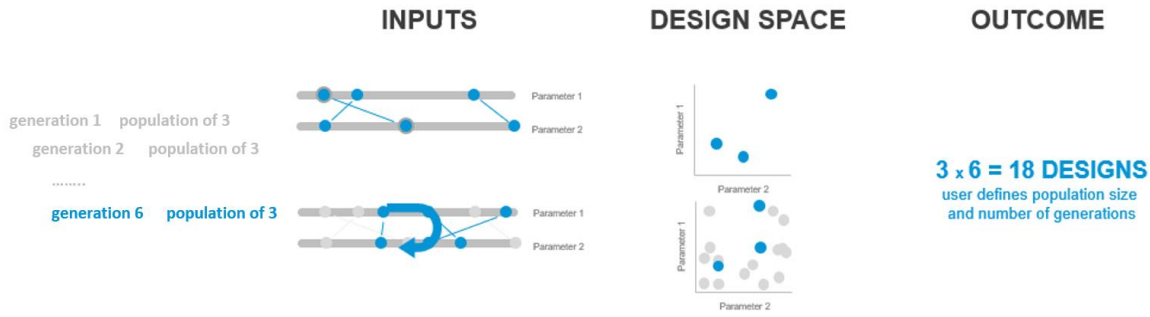
Now that we have a design system defined, we can use Refinery to automate or generate the creation of design alternatives that exist in the design system. Generation methods in Refinery include:

1. **Randomize** When Refinery uses the Randomize option it will generate a specified number of design options, by randomly assigning a value to each of the input parameters. This option is used when facilitating an optioneering process.

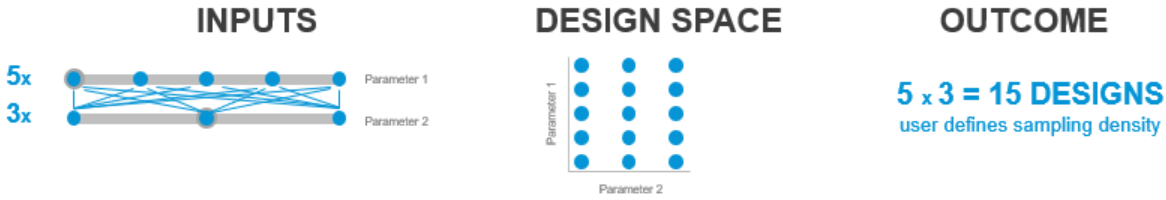


2. **Optimize** is the method for doing an optimization run with Refinery. During an optimization run Refinery will evolve the design based on the evaluators outputs. The optimization process works by running multiple generations of a design, each generation will use the input configuration from previous generations and from that optimize the new design options.

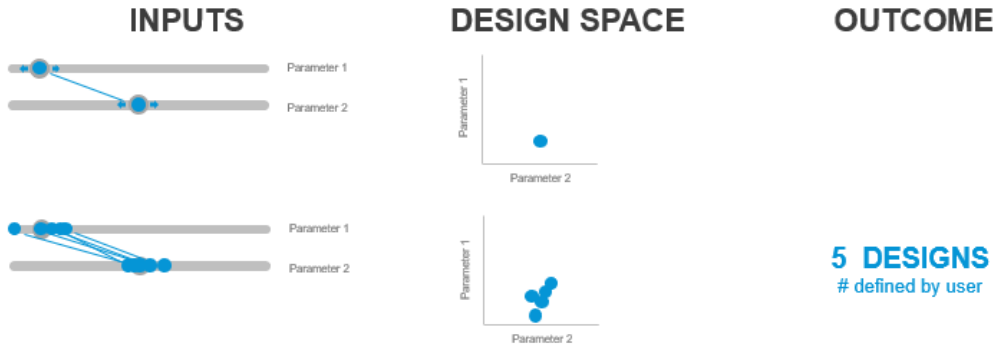
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



3. **Cross Product** method lets you explore the entire design space of your design, by combining each step of every parameter with the remaining parameters.



4. **Like This** will make Refinery apply slight variations to your current input configuration. Using this method lets you explore different variations of a design that you already like.



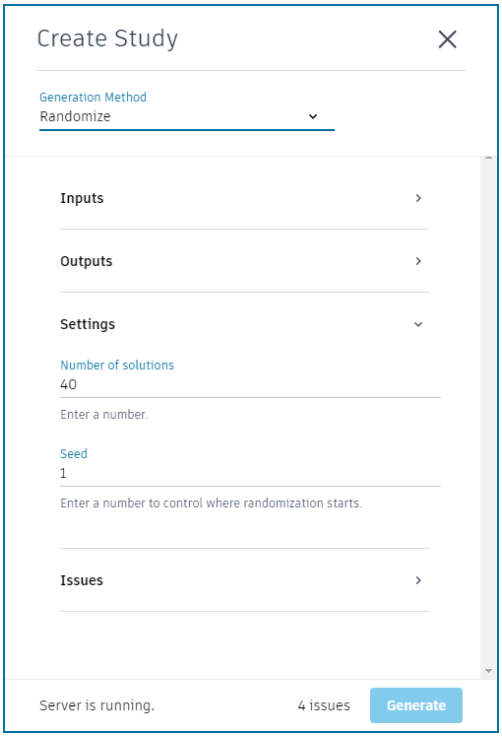
Refinery can be run using the different methods above. In the Refinery window you can choose from the four different methods.

How to run an optioneering process using Refinery

An optioneering process lets you explore all possible solutions that the graph can produce. This is helpful if you want to test the flexibility of the graph or you're unsure about what to optimize for. Refinery will generate the solutions based on the constraints that were defined in the Dynamo graph. To run an optioneering process in Refinery, follow these steps:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

1. Launch Refinery from the Refinery menu in Dynamo
2. Create a new study and select Randomize, Cross Product or Like This as the generation method
3. Under Inputs make sure that all the desired inputs are present
4. For inputs that should not change on each run, set the desired value and uncheck the box alongside it
5. Under Outputs ensure each output defined in the graph is listed
6. Under Settings choose how many options Refinery should create
7. Under Settings select a random seed (number) to initialize the randomization
8. Under Issues resolve any items
9. Click Generate

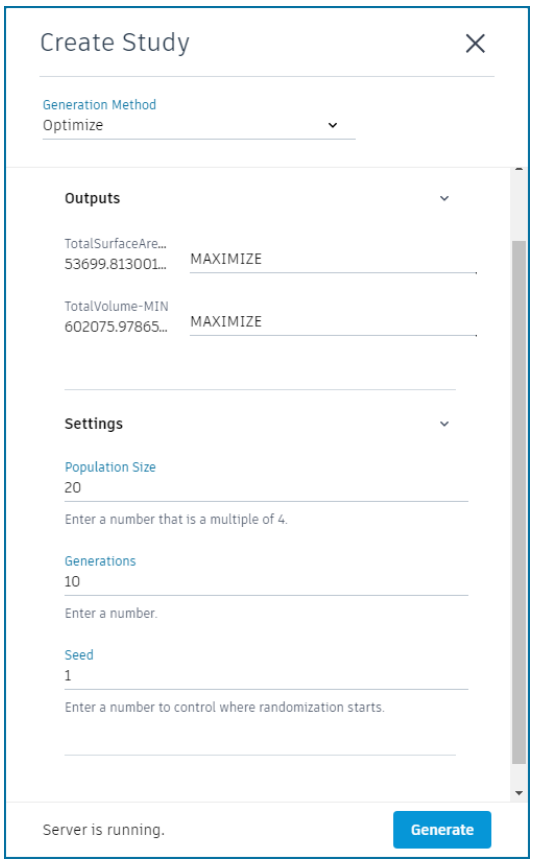


How to run an optimization process using Refinery

An optimization process lets Refinery evolve your design to find the most suitable options based on the constraints and goals provided. Refinery will run multiple generations of options and each time it will take the fittest (best) options of the generation and use them to create a new generation. Refinery is using NSGA-II, an 'elitist' multi objective genetic algorithm to optimize results. To run an optimization process in Refinery, follow these steps:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

1. Open Refinery from the Refinery menu in Dynamo
2. Create a new study and select Optimize as the generation method
3. Under Inputs make sure that all the desired inputs are present
4. For inputs that should not change on each run, set the desired value and uncheck the box alongside it
5. Under Outputs go through each objective and set the optimization goal you want to achieve - Maximize, Minimize, or Ignore
6. Under Settings set a population size, which represents the number of options that will be created in each generation.
7. Under Settings set the amount of generations you want to create. Each new generation is a range of options that falls between the two best designs of the previous generation
8. Click Generate



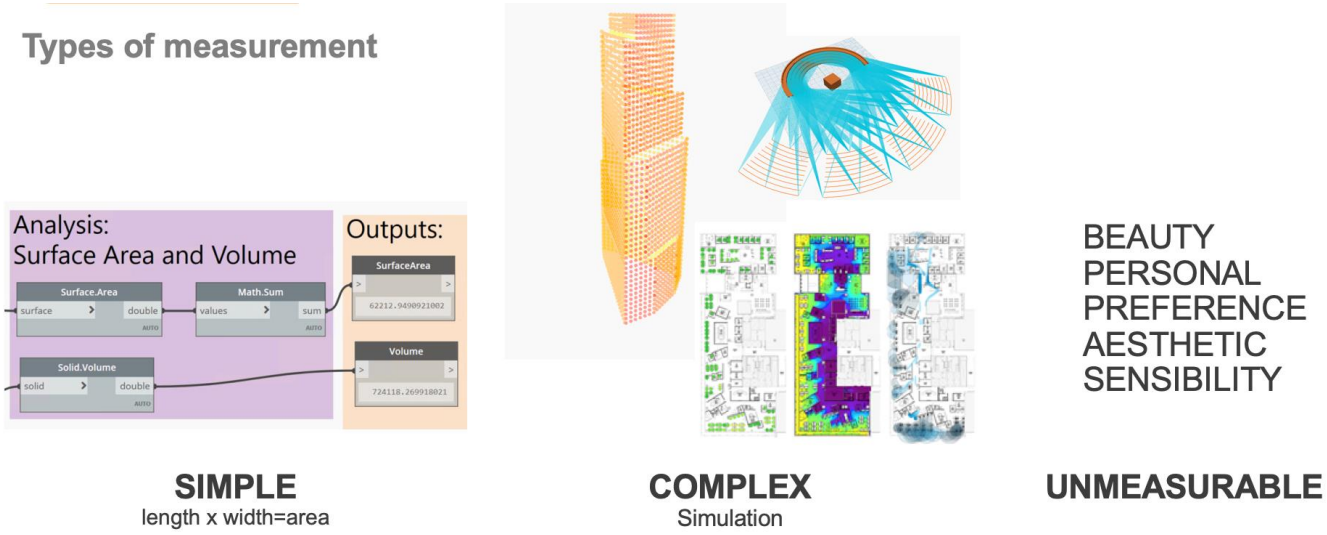
Analyze

Judging outcomes is the critical 2nd half of the problem. To do this you can use a set of measures that determine how the building is performing. Why?

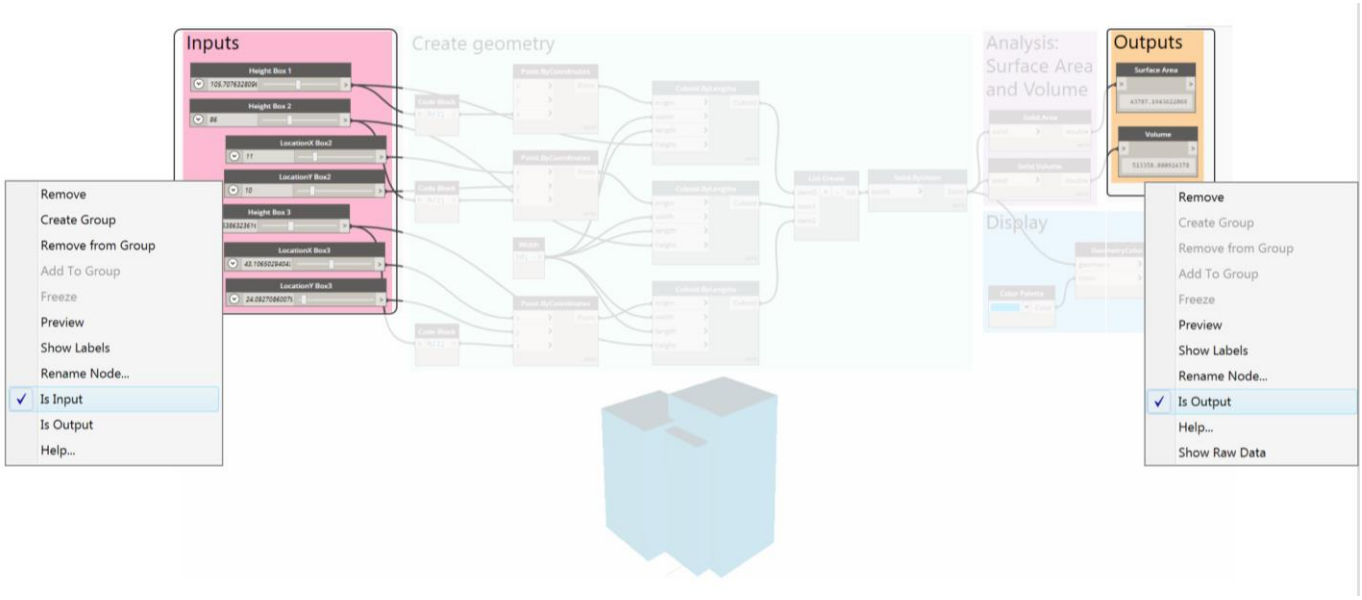
- Compare designs objectively (apples to apples)
- Evaluate designs based on non-intuitive measures
- Explore more designs than is possible manually

There are 3 types of measurement:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



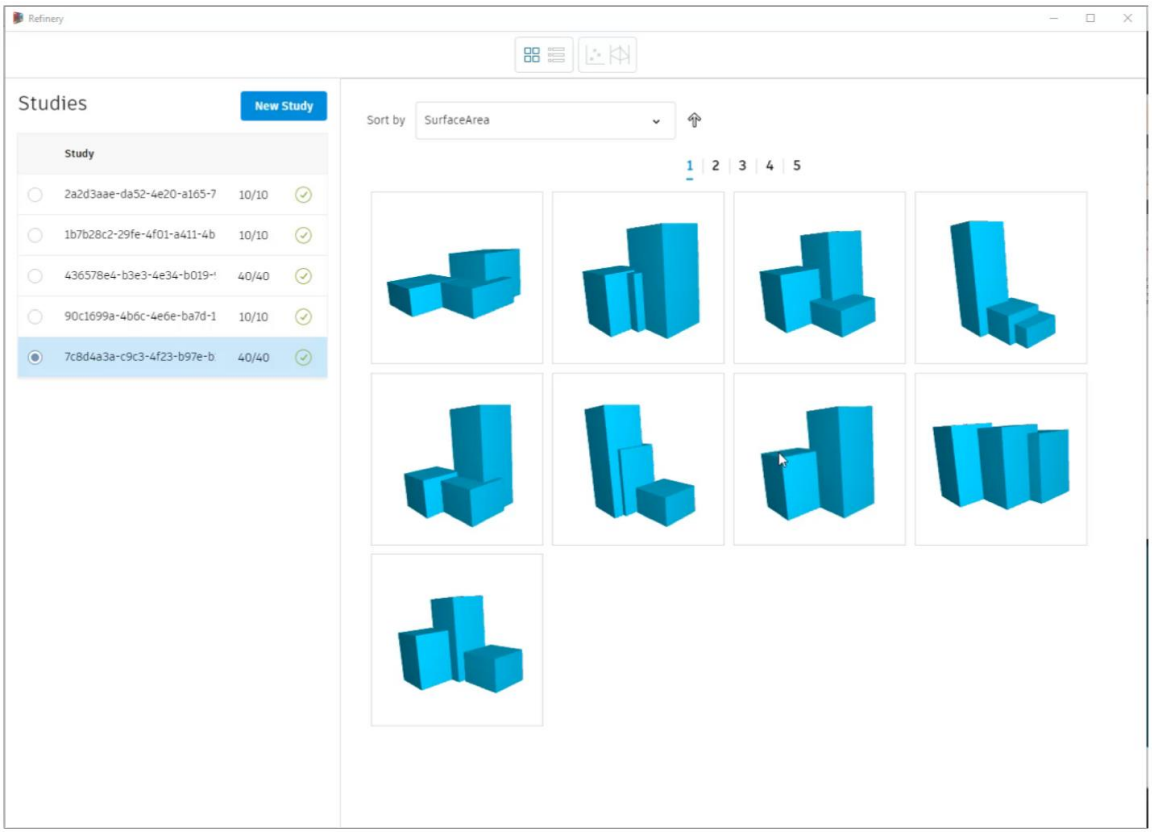
Designating inputs and outputs in Dynamo for use in Refinery:



Rank

In Refinery, you can use sorting and filtering to rank resultant design options and make decisions about the best options for your needs.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Refinery designs sorted by Surface Area

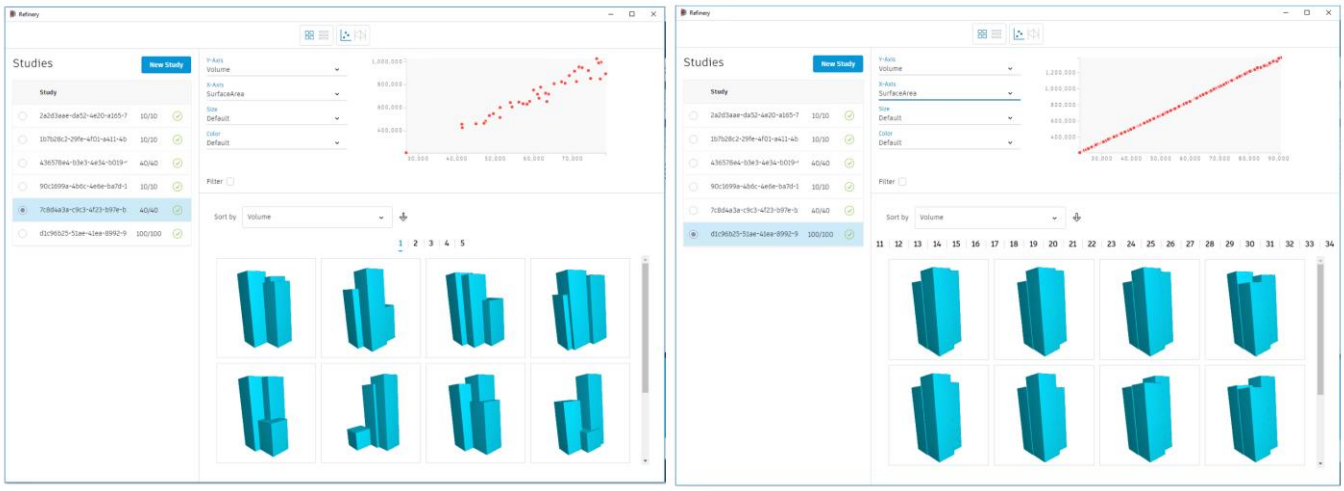
Evolve

After you have explored some possibilities in the design space that you have set up in Dynamo, you may want to set some goals for the design to guide the design generation and produce options that meet your goals. To do this, you can use the Optimize generation method. There are several parameters that effect the optimization

- Output goals (maximized or minimized)
- Population size (how many designs should be tested in each generation?)
- Generation size (how many generations should be tested?)
- Seed (initialization value)

Optimization runs can take longer, but will return only results that meet your goals. The runs below show the difference between the Random method and the Optimization method. The scatter plot charts below are both set to Volume on the Y-axis and Surface Area on the X-axis. Note that in the optimization run on the right, the designs return look much more similar and form a tighter line comparing volume and surface area.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



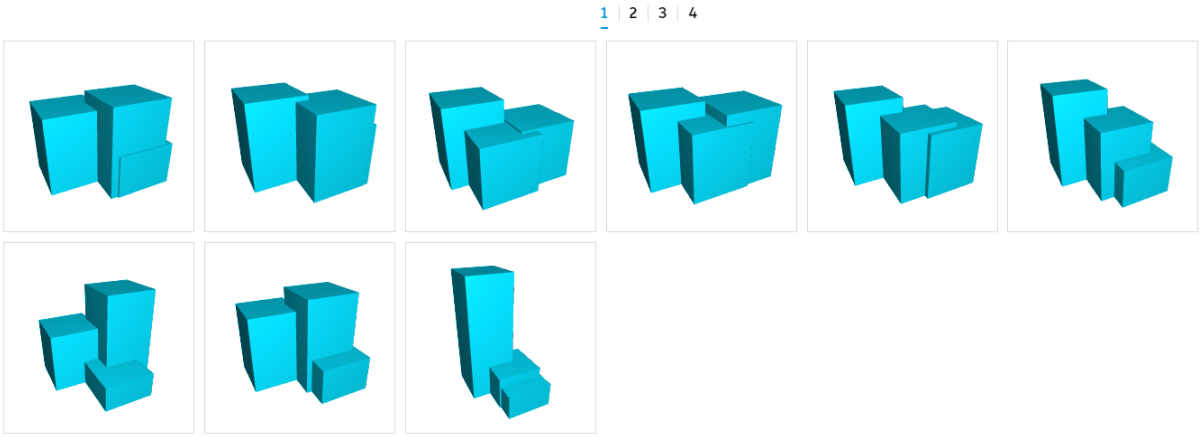
40 random runs – 10 sec

100x100 optimization – 4 min

Explore

After Refinery has run the generative process, the results are displayed in both geometric form and through a series of charts or tables. All of the resulting views are interlinked and selecting an option in one view will highlight it in the other views currently displayed. If Dynamo is running in Automatic mode in the background, selecting an option will also update the graph to show this design. There are several features of the explore interface:

1. **Design Grid** The design grid shows each option as a 3d geometrical thumbnail that can be individually rotated, zoomed and panned to explore the design in more detail. The order of the thumbnails can be sorted based on the inputs or outputs of the Dynamo script, with a toggle for both ascending and descending values.

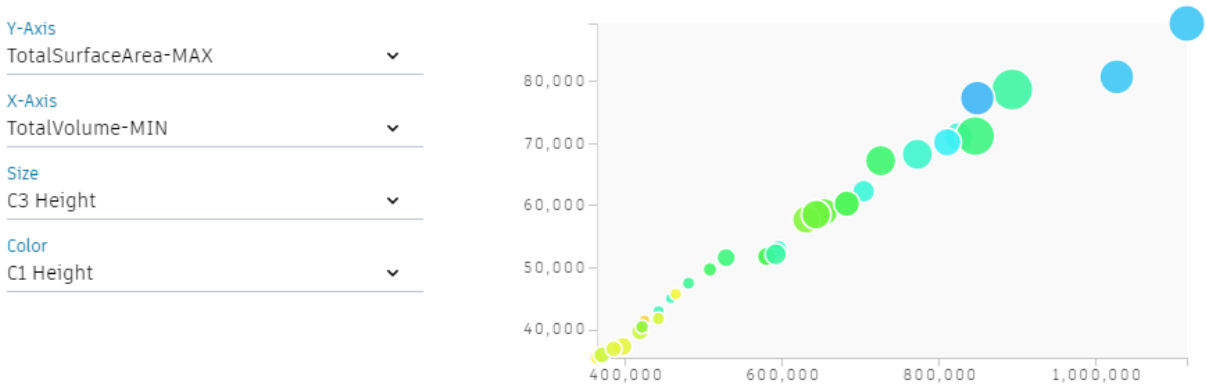


2. **Design Table T**he design table replaces the design grid, if chosen, and lists each option in a table form with each column representing the values for the inputs and outputs.

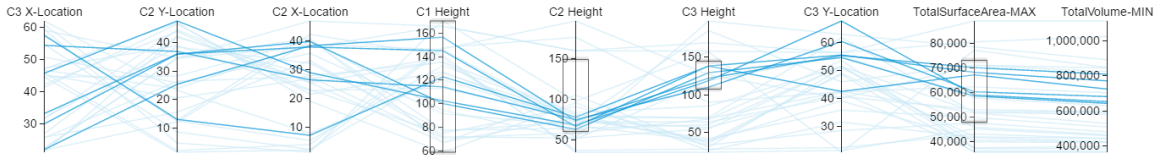
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

TotalSurfaceArea-MAX	TotalVolume-MIN	C3 X-Location	C2 Y-Location	C2 X-Location	C1 Height	C2 Height	C3 Height	C3 Y-Location
41456.712	454217.178	41.954	24.791	11.654	61.331	59.381	102.729	34.832
41578.589	424163.121	47.119	46.957	19.060	58.759	95.981	25.227	31.460
48569.928	529109.489	26.810	42.372	38.189	65.695	109.178	100.908	52.748
57784.215	630028.715	21.829	12.137	39.870	94.405	51.141	118.784	55.295
65351.505	805919.903	53.759	18.735	21.948	111.373	160.120	113.769	39.955
69265.414	876697.721	42.328	36.077	11.438	190.133	182.257	25.506	21.742
76891.400	986952.474	53.550	15.168	29.379	178.812	172.316	110.951	49.648
77314.139	848258.692	59.459	4.693	1.417	170.390	97.898	157.210	20.587
77508.007	993015.152	26.138	16.635	36.074	147.930	188.559	95.979	61.584
78710.577	892655.056	21.179	41.879	27.823	135.508	53.463	198.658	63.065

3. **Scatterplot** The first chart Refinery uses to visualize data is a Scatterplot : a type of mathematical diagram that uses cartesian coordinates to display values across a set of data. Refinery allows you to select what values are displayed along both the X and Y axis as well as which ones drive the size and color of the circles in this 4-dimensional view. The values can be chosen from the inputs or outputs you defined in the Dynamo graph in the previous steps. Selecting a circle from the graph space will also highlight the chosen option in the design grid or design table.



4. **Parallel Coordinates T**he other chart available in Refinery is a Parallel Coordinates graph. This chart shows a set of vertical parallel lines, equally spaced, that represent the inputs and outputs. Each design option is represented as a polyline whose vertices sit on each parallel axis. The position of the polylines vertices on the axis corresponds to the value of the input or output. The graph can be filtered by dragging the selection on each vertical axis.

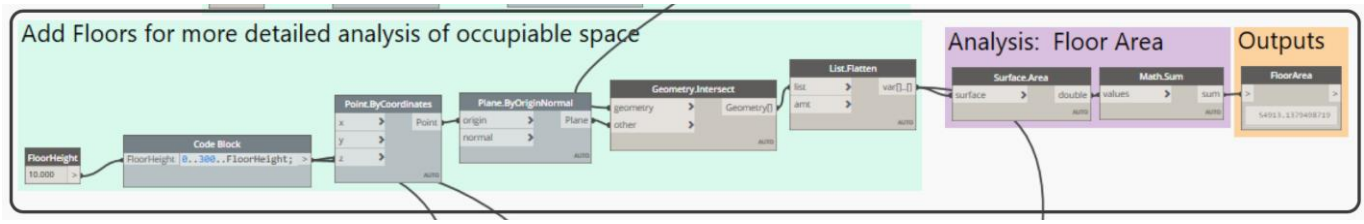


Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

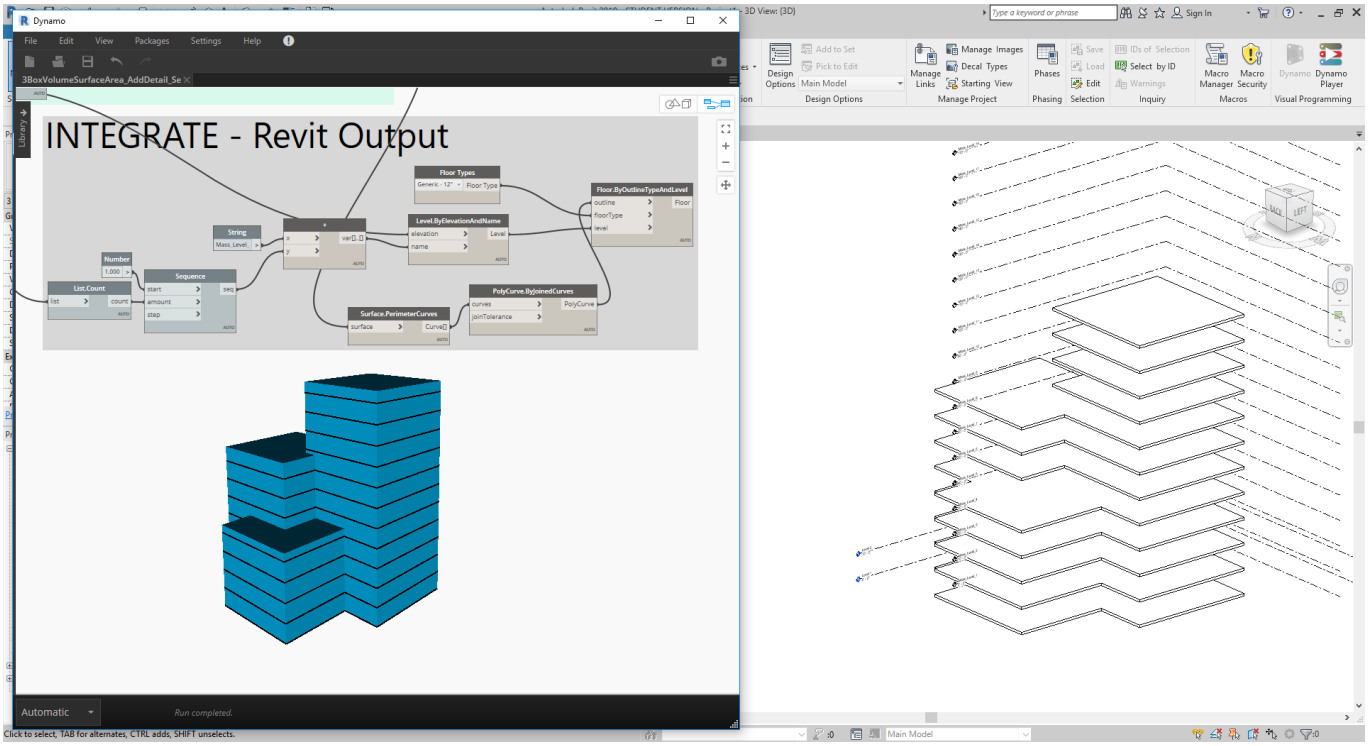
The kind of visualization you choose for your project may vary depending on what kind of process you are running. If you are running an optioneering process, it can be beneficial to visualize it using a parallel coordinates chart as it will be easier to filter options, while visualizing a multi-objective optimization using a scatterplot chart will make it easier to find the best trade-off between two objectives.

Integrate

The final step in the generative design workflow is to integrate your chosen design into a context where you can develop it further. One way to do this is to add a branch to your Dynamo graph which places Revit elements. In this example we're first adding floors to the Dynamo model for a more detailed analysis of the occupiable space.



And then we can add those same floors to the Revit model.



Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

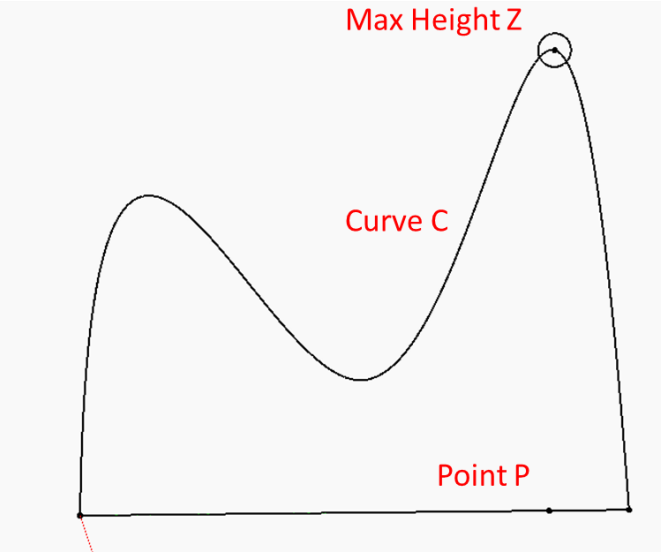
Integrate design data into Revit as floors.

Example 2: 2d Hill Climbing

Let's go back a step and focus goal-setting and how to make evaluators.

This study looks goes back to the basics and helps us define what an evaluator is, and how to describe objective functions to be maximized or minimized

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



So What is an Evaluator?

An evaluator (also known as a discriminator), is fed potential solutions from the generator and evaluates how good or bad these solutions are. Evaluators are a natural analogue of the objective and fitness functions outlined below. In a generative building design, an evaluator may be a function that describes the average amount of sunlight a façade will be exposed to over a year.

In design, evaluators must be specified mathematically, they must output a number that can be used to discriminate between solutions.

- https://refineryprimer.dynamobim.org/05-algorithms/05-03_evaluators

The Evaluation Phase

What do you mean by an 'objective function'?

An objective function is simply the criterion you want to maximize or minimize.

What do you mean by 'fitness'?

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

A fitness function is essentially the objective function for the genetic algorithm; it's the criterion you want to maximize or minimize, the thing you care most about achieving. A fitness function is used to evaluate how close (or far) a given design solution is from achieving the designer's goals. The genetic algorithm is designed to drive the model to higher and higher fitness, so defining a fitness function precisely and accurately is vital.

Some examples of fitness functions that could be used in a Generative Design context include:

- Maximize hours of daylight for each desk in an office
- Maximize circulation in common areas
- Minimize number of distinct types of parts needed to assemble an object
- Minimize number of total parts needed to assemble an object
- Maximize the structural strength of a critical component in a product
- Minimize the fuel needed to power a vehicle
- Minimize the weight of a design

You can see here that these suggestions are always framed as either a maximization or minimization problem. As discussed above, this formulation is crucial for an optimization approach to be effective.

Another consideration is that it is desirable for fitness functions to be calculated very efficiently by a computer. That is, a good fitness function can be calculated fast. With experience, a user comes to learn which kinds of fitness functions are likely to be particularly fast or slow.

One of the great strengths of a genetic approach is that the fitness function can be quite complicated. In fact, a single genetic model can have multiple competing fitness functions or even a single fitness function that is a combination of several others (minimize the weight of a design while also making it as structurally sound as possible, for example). This is also known as Multi Objective Optimization and Single Objective Optimization.

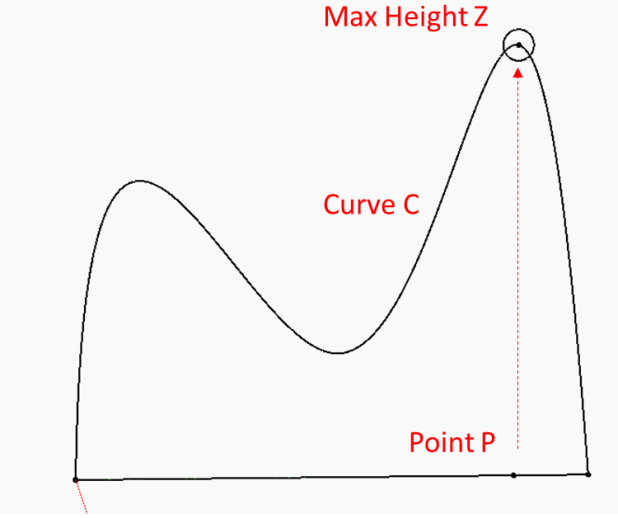
- https://refineryprimer.dynamobim.org/04-optimization/04-08_the-evaluation-phase

Here is the Objective Function for our 2d Hill Climber, and some pseudo-code for the logic we need:

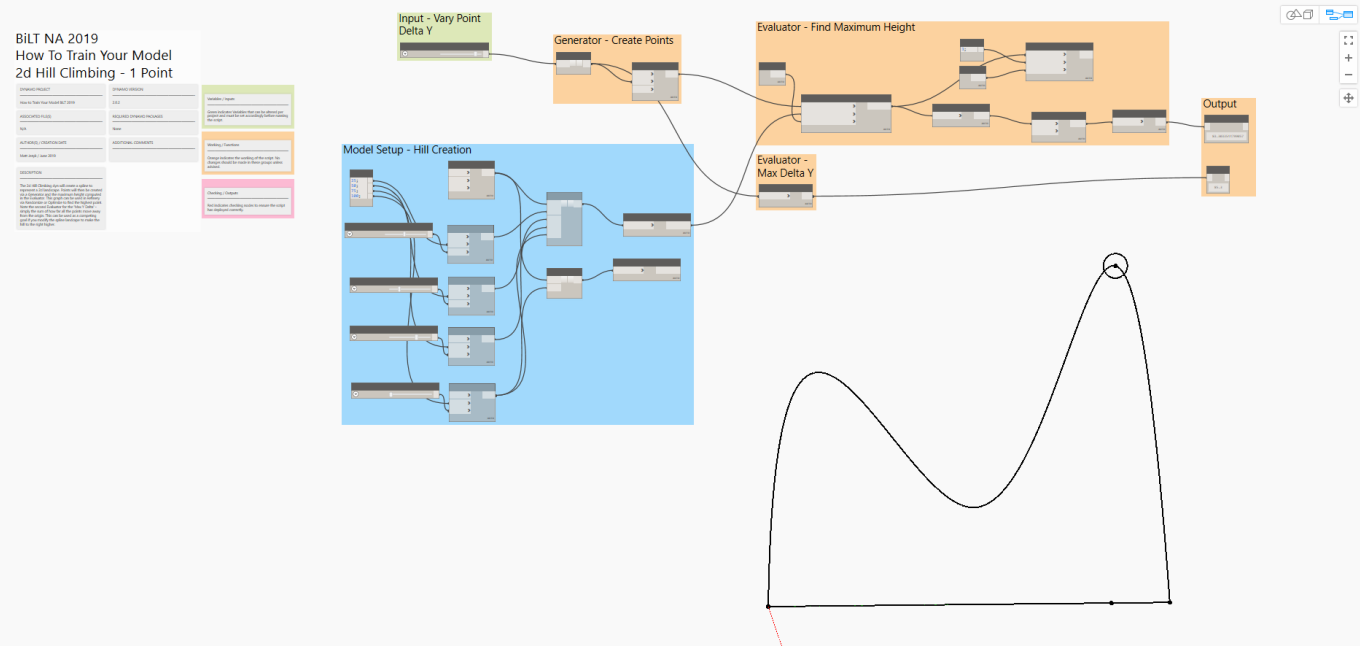
- Find Maximum Height Z for Point P

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

    - z = f(c,p)
    - maximize z
- Variable Inputs:
    - Curve C
    - 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



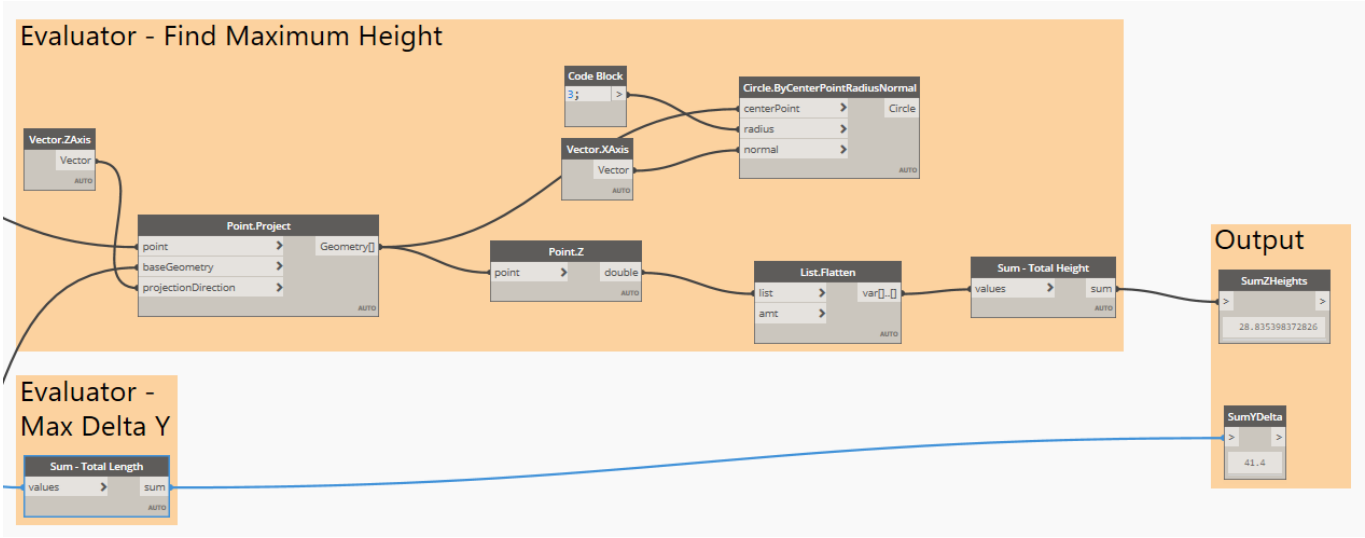
Here is the end state of the Dynamo graph we will create:



You can see the nodes in the blue group set up the 2d hill. We have a Generator that lays out points along the base line, simply by making a range of points and varying their Z value. The Evaluator is set up to project the base points up to the hill curve. It then adds up all the Z heights and passes that to the Output section.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

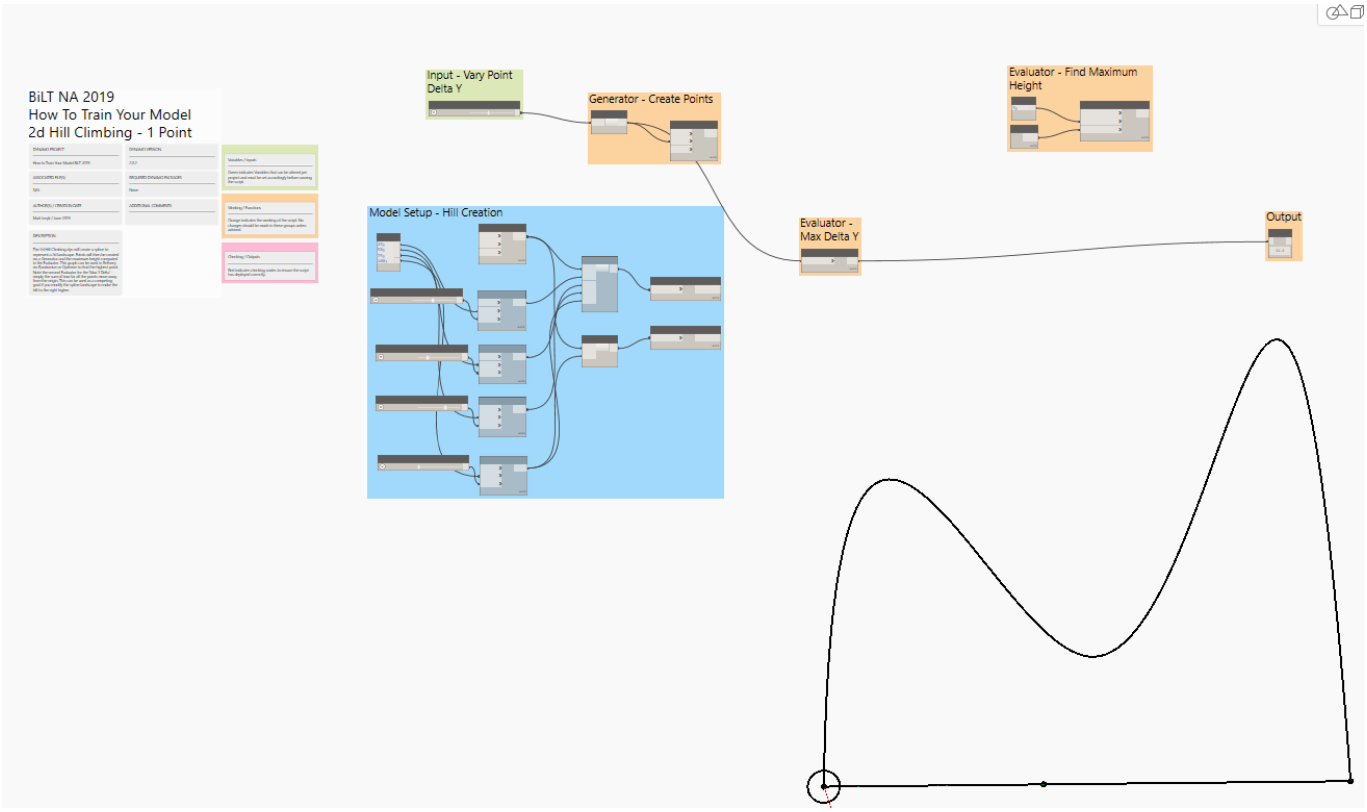
Open "\SampleFiles\ 02_2d Hill Climbing \ 01_2d Hill Climbing 1 point START.dyn" in Dynamo.
You should see the main part of the graph. Let's build up the evaluator together to look like the image below. Here is the Evaluator:



Also note the description in the notes on the left:

> The 2d Hill Climbing dyn will create a spline to represent a 2d landscape. Points will then be created via a Generator and the maximum height computed in the Evaluator. This graph can be used in Refinery via Randomize or Optimize to find the highest point. Note the second Evaluator for the 'Max Y Delta' - simply the sum of how far all the points move away from the origin. This can be used as a competing goal if you modify the spline landscape to make the hill to the right higher.

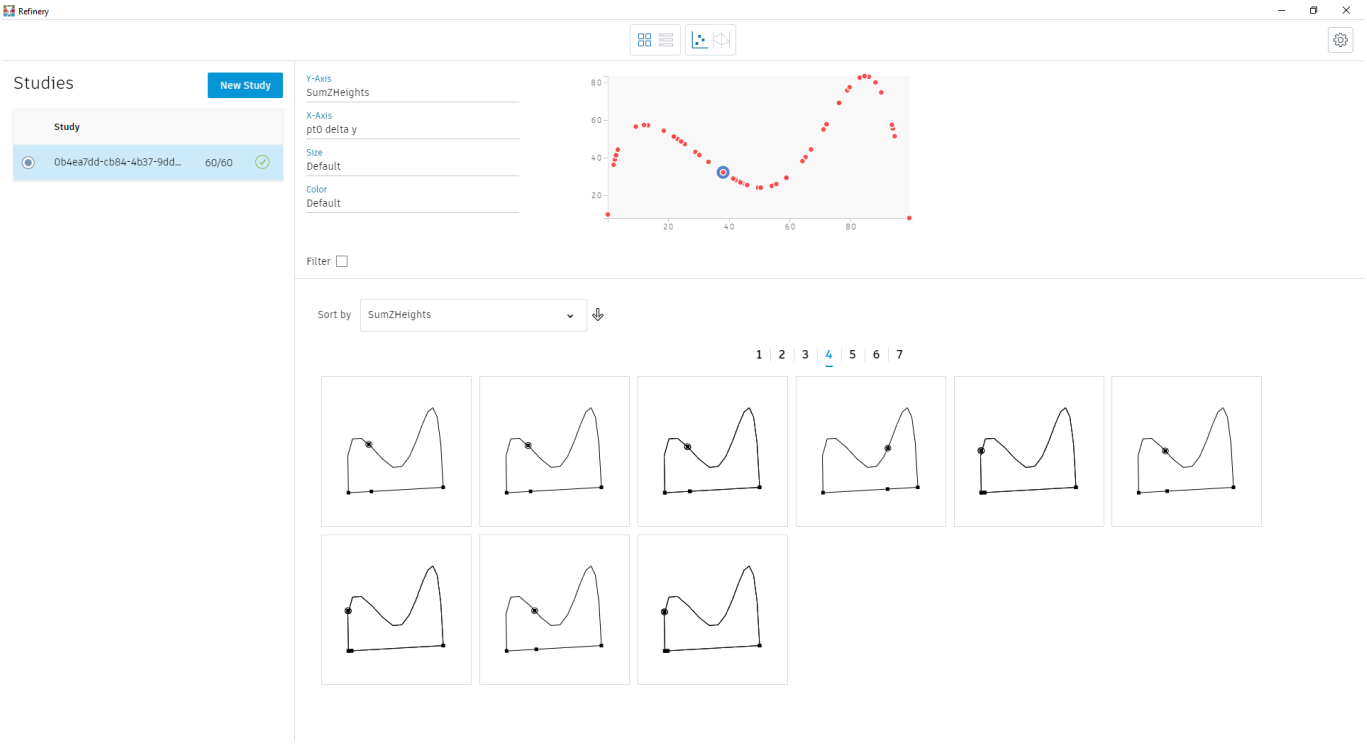
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



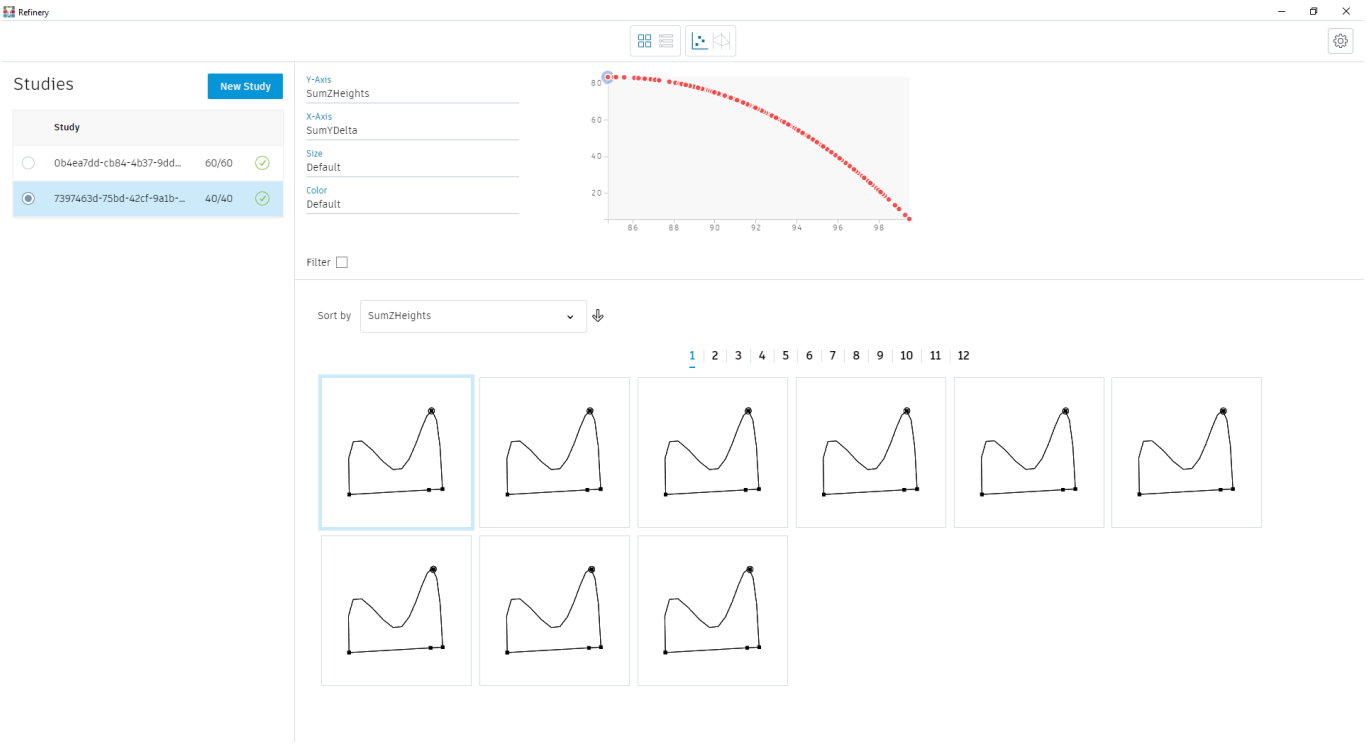
Build the Evaluator and also add a Watch node labelled “SumZHeights”, making sure to set the ‘IsOutput’ setting on the node so Refinery knows to use it. Once we have built the Evaluator and the Output we can open Refinery and Generate some options!

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Let's use Randomize first. You will notice the results look very similar to the 2d hill in dynamo because we are simply plotting y values for the range of x values being tested.

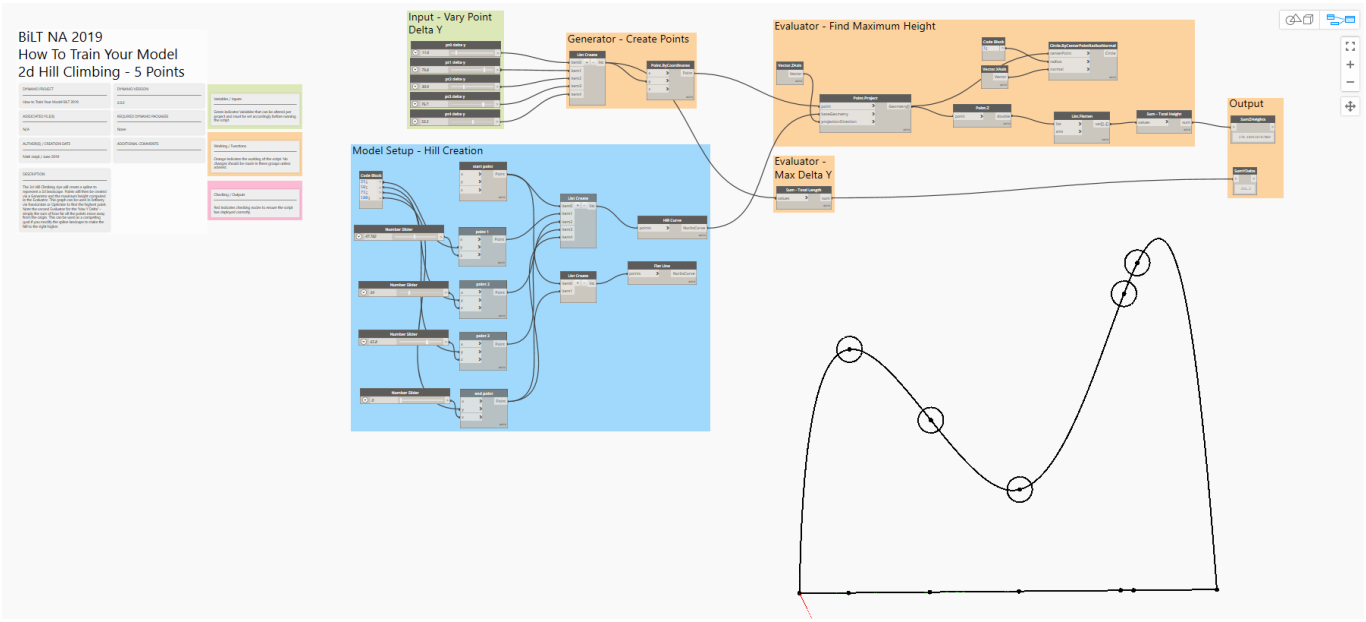


Now let's try an Optimize run, you should see results similar to this:



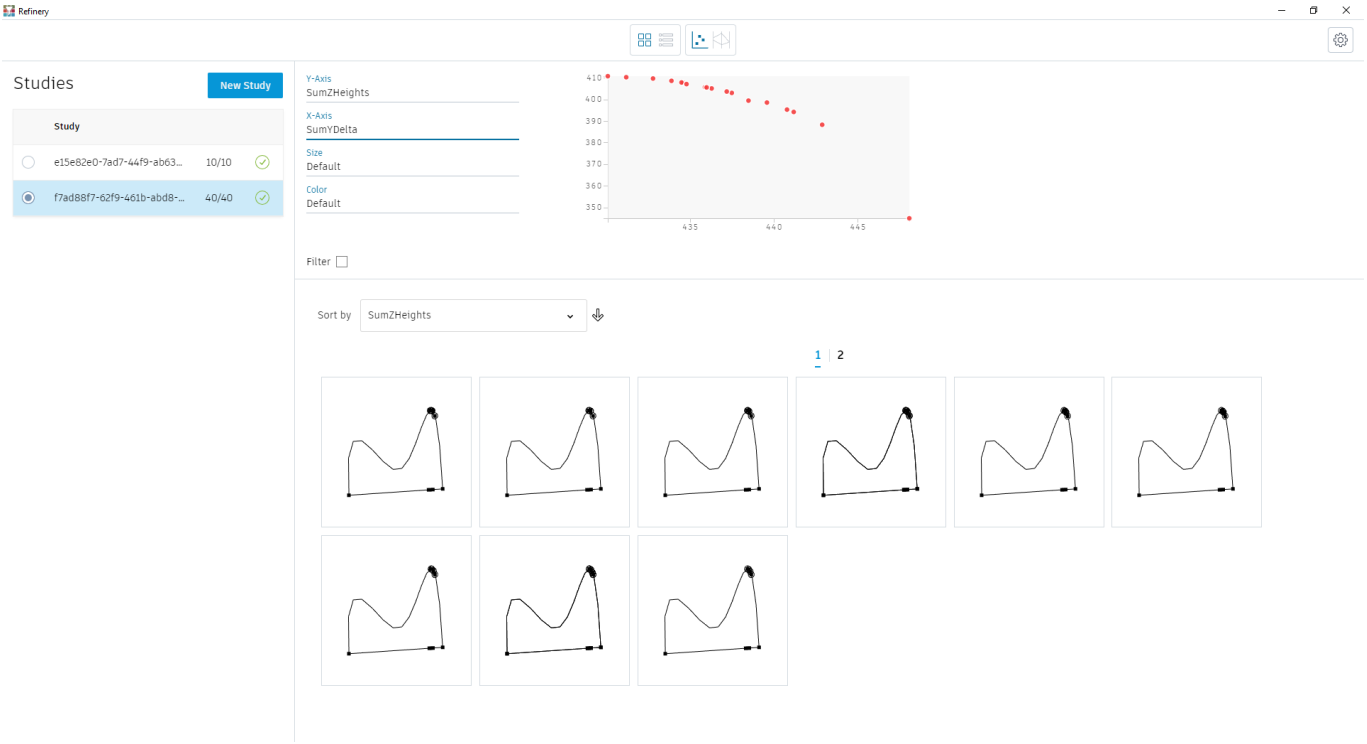
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

You can add more inputs to the graph, simply by adding more sliders and wiring them up to the List node. Make sure to set the sliders to 'IsInput' and give them unique names,

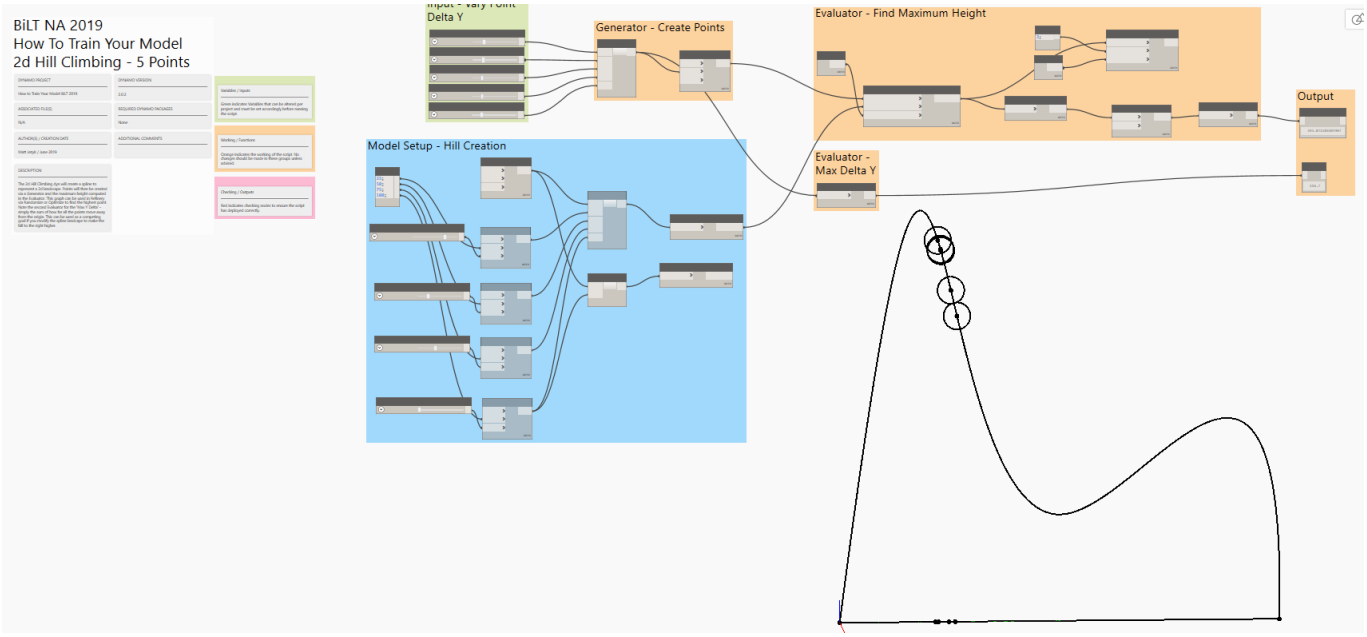


Notice what happens when you run Optimize on this graph? You should start to see the Pareto front develop.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

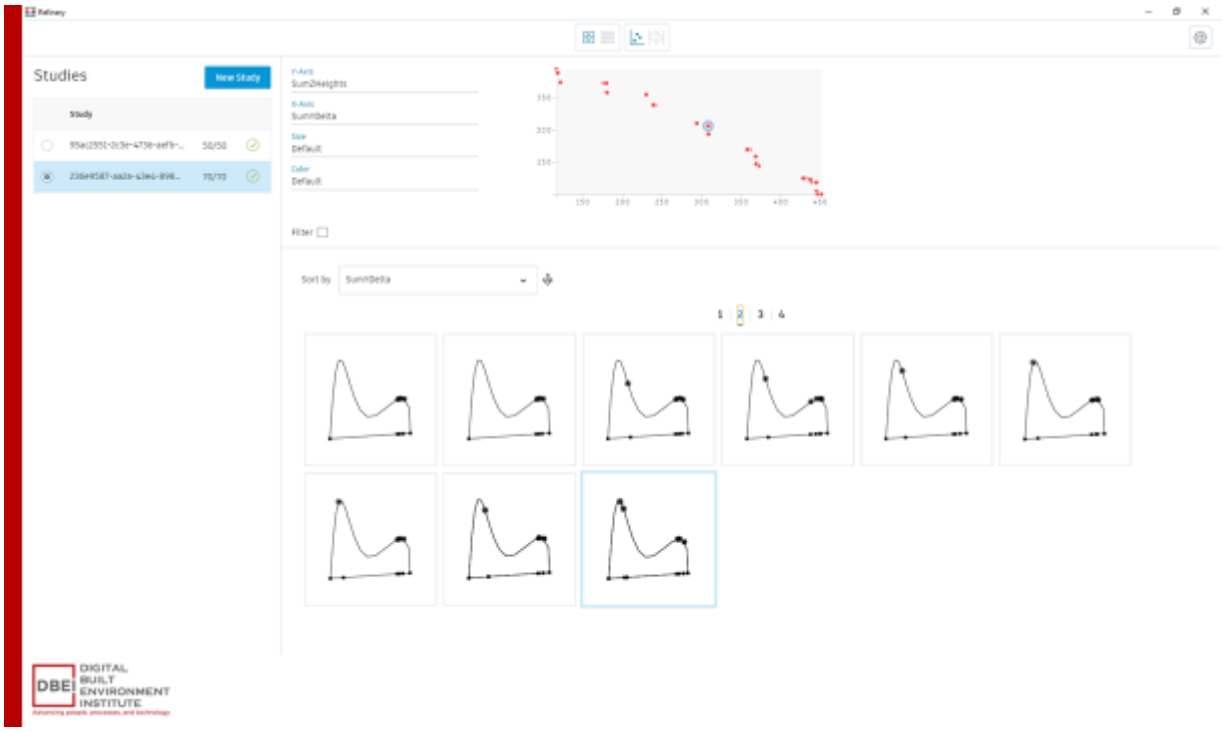


Now modify the shape of the hill by altering the sliders in the blue group.



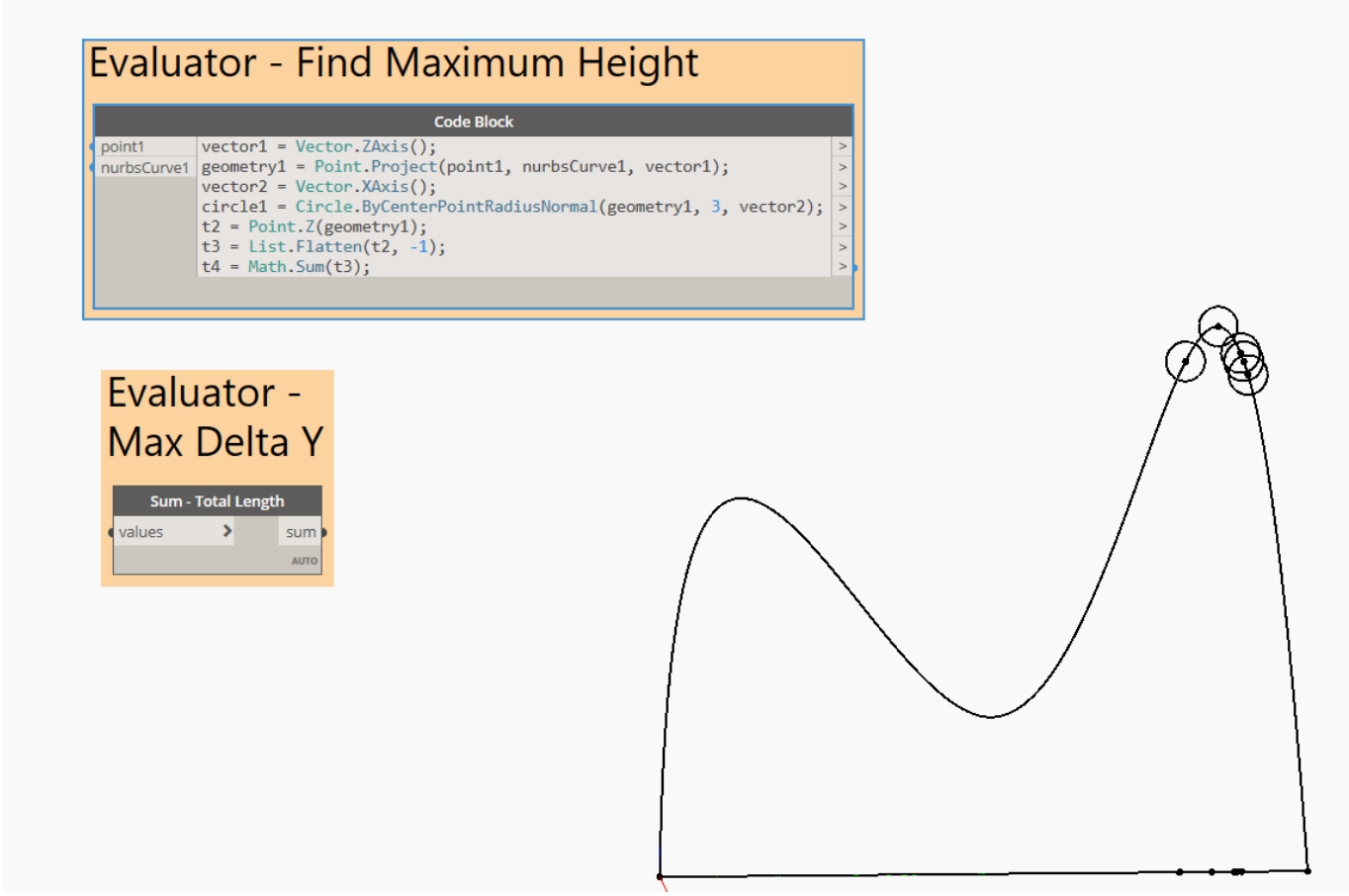
If you run Refinery again you can solve for the new hill condition. You can also see how competing goals can play out. The Maximize Z and Maximize Y goals are competing:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



We can create the same type of Evaluator using DesignScript and Python. For Designscript, simply in the interest of time, select the nodes and find the 'Node to Code' option in the right click menu:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



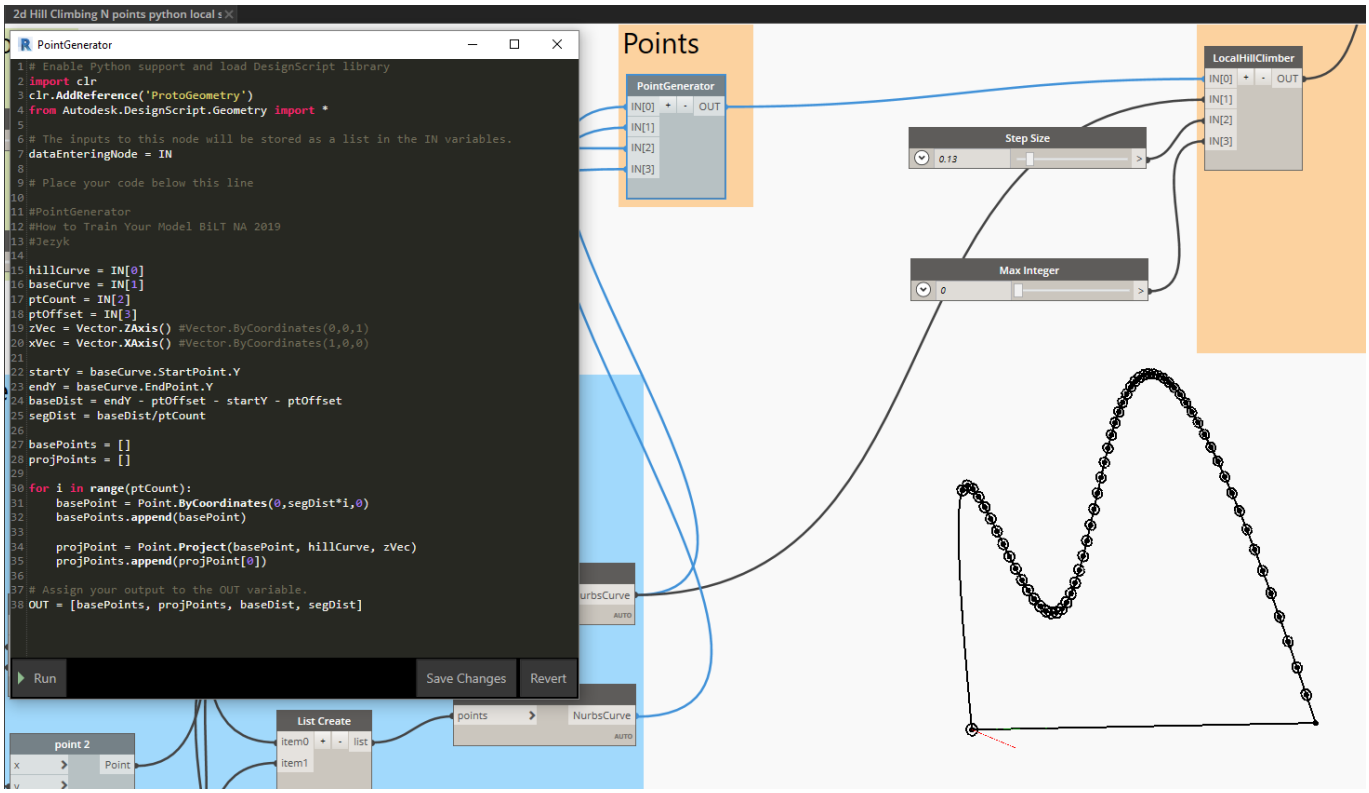
You can use DesignScript for much more than this, of course, but for the next excerize we will use Python instead.

We will use Python to make a new Generator to create many more points than we would want sliders for, then start to control them with a new Evaluator.

Open the file, "07_2d Hill Climbing N points python local solver slope START.dyn"

Start by double-clicking on the node called 'PointGenerator'. This is a Python node we will add code to.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Enter this code:

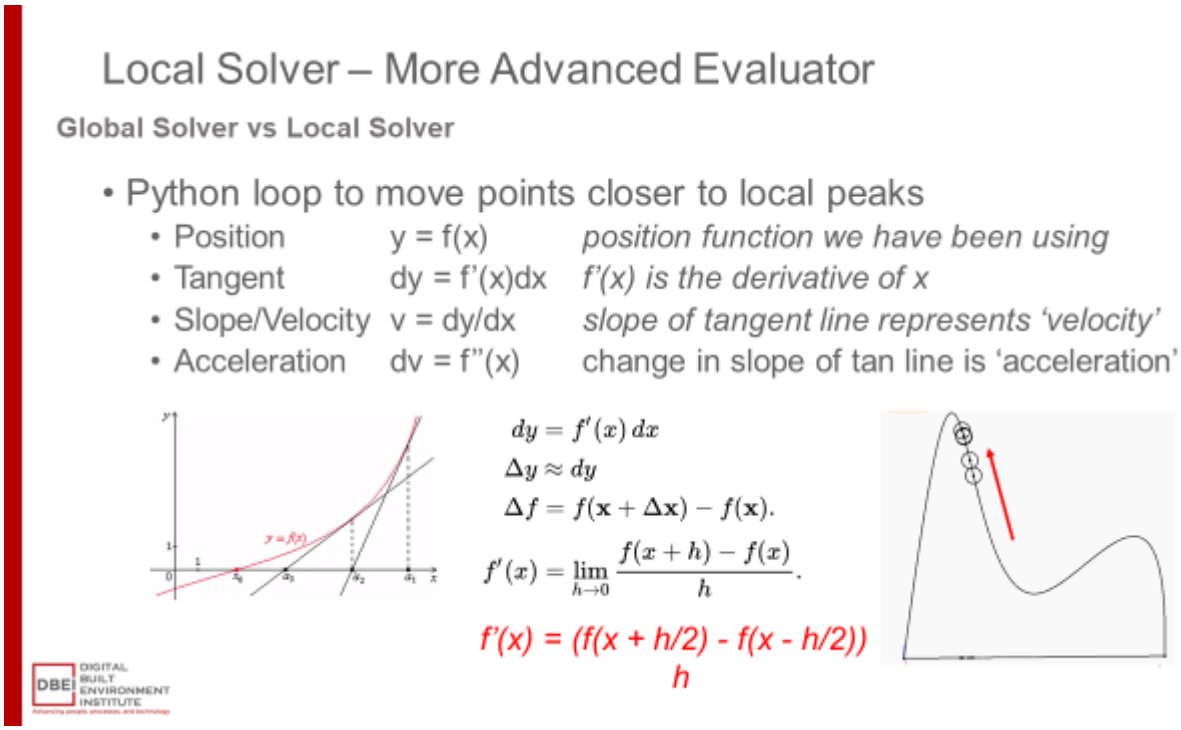
```
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# 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
```

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

```
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]
```

And then press Run, you should see points be laid out along the curve.

The Evaluator we want to add is going to simply move all the points from the Generator up the hill until they reach a horizontal slope. We will use a mathematical technique called discrete numerical differentiation to find the horizontal slope mathematically by performing a series of refined guess. This is using "Newton's Method" to 'step along the curve' and evaluate the curve tangent at a position, then refine that position and try to reduce the slope of the tangent to zero.



Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

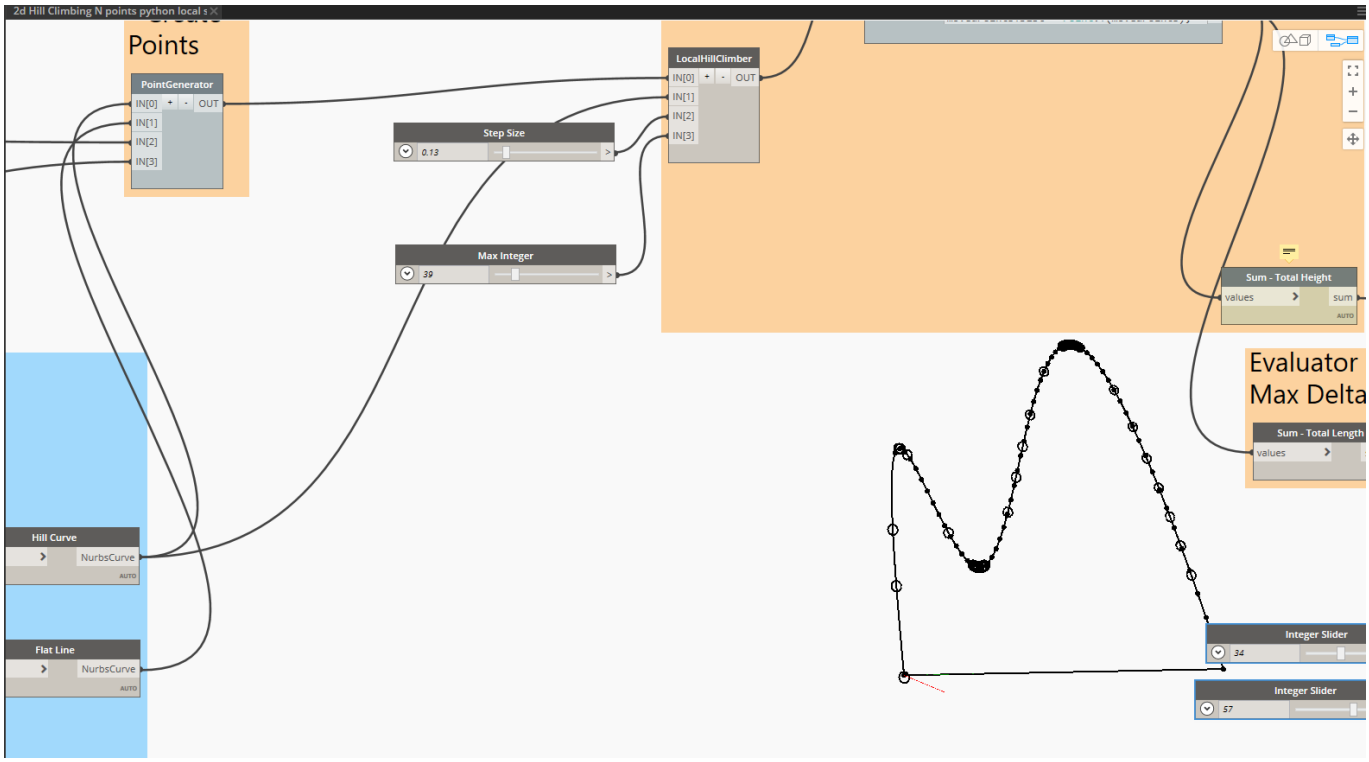
Now let's add the Evaluator, double-click on the node named "LocalHillClimber". This is a Python node we will add code to. Add this code:

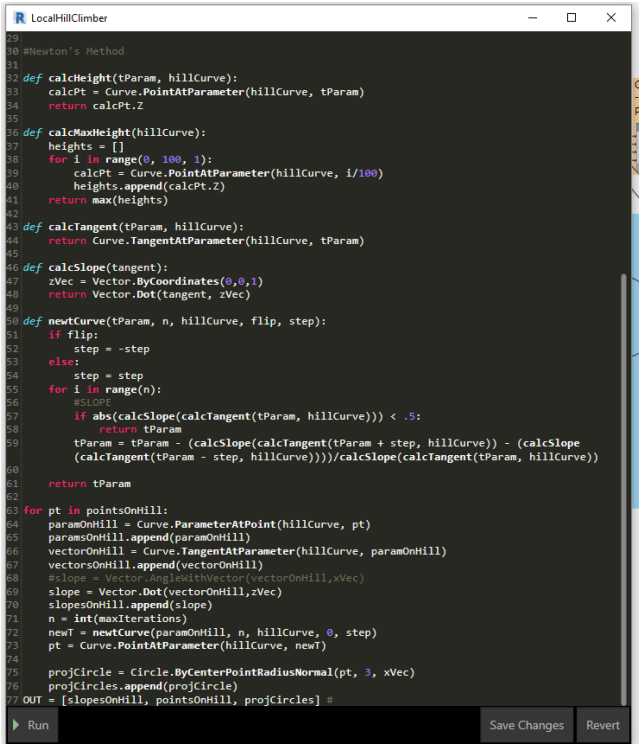
```
# Enable Python support and load DesignScript library
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
dataEnteringNode = IN
# Place your code below this line

#How to Train Your Model BiLT NA 2019 - Jezyk
#Using Newton's Method to find minimum slope along a curve,
#then moving point locations closer to the min slopes based on maxIterations
#you could make this a while loop that continues to iterate until all points are under a
min slope.

#Newton's Method in Python
#http://code.activestate.com/recipes/578419-newtons-method-to-solve-equations-in-python/


generatedPoints = IN[0]
hillCurve = IN[1]
pointsOnHill = generatedPoints[1] #[0] = basePoints, [1] = projPoints

step = IN[2]/100
maxIterations = IN[3]

zVec = Vector.ZAxis()
yVec = Vector.YAxis()
xVec = Vector.XAxis()

paramsOnHill = []
vectorsOnHill = []
slopesOnHill = []
projCircles = []
zValues = []
newPointsOnHill = []

#Newtons' Method
```

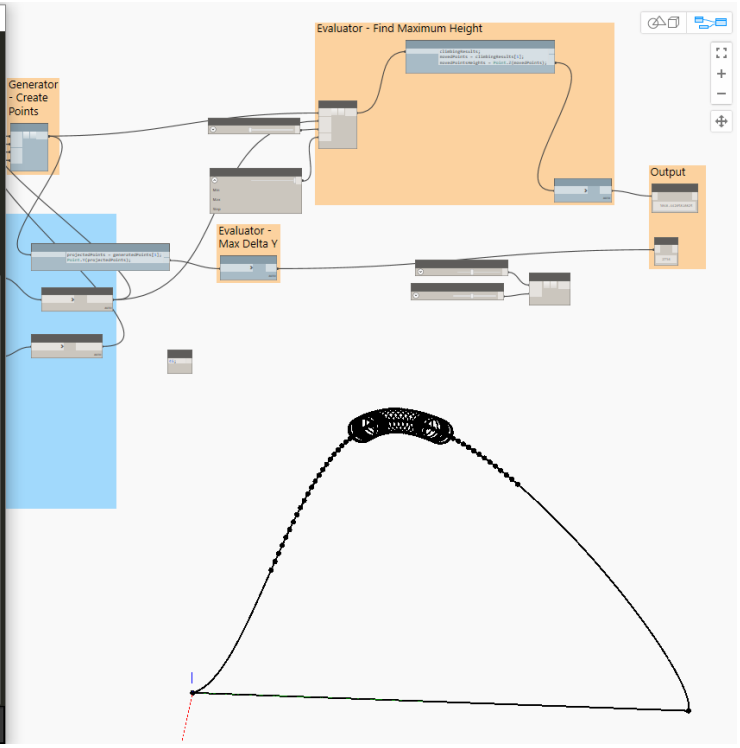
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

```
#Eval Position on Curve, Return Height
def f(t): # for t (t param of curve), return y height for the curve
    evalPoint = hillCurve.PointAtParameter(t)
    z = evalPoint.Z
    #zValues.append(y)
    return z

#Eval Tangent on Curve, Return Slope
def f_prime(t): # for t (t param of curve), calc tangent and return slope angle of
tangent line
    tanVector = hillCurve.TangentAtParameter(t)
    #tanVectors.append(tanVector)
    slopeAngle = Vector.Dot(tanVector, Vector.ZAxis()) #zVec #Vector.XAxis())
    return slopeAngle

#Use Newtons' Method to Step Along Curve
def newtCurve(tParam, n, hillCurve, step):
    for i in range(n):
        if abs(f_prime(tParam)) < 0.5:
            return tParam
        tParam = tParam - ((f_prime(tParam + step)) - (f_prime(tParam -
step)))/f_prime(tParam)
        #tParam = tParam - ((f(tParam + step)) - (f(tParam - step)))/step*10
        slopesOnHill.append(f_prime(tParam))
        zValues.append(f(tParam))
    return tParam

#Iterate Through All Points and Move Along Curve
for pt in pointsOnHill:
    paramOnHill = Curve.ParameterAtPoint(hillCurve, pt)

    n = int(maxIterations)
    newT = newtCurve(paramOnHill, n, hillCurve, step)
    pt = Curve.PointAtParameter(hillCurve, newT)

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

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)





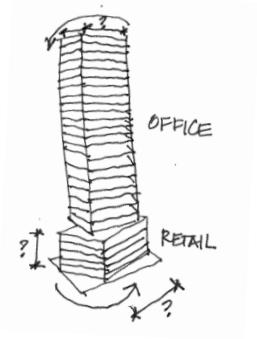


Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Example 3: Building Massing – Complex

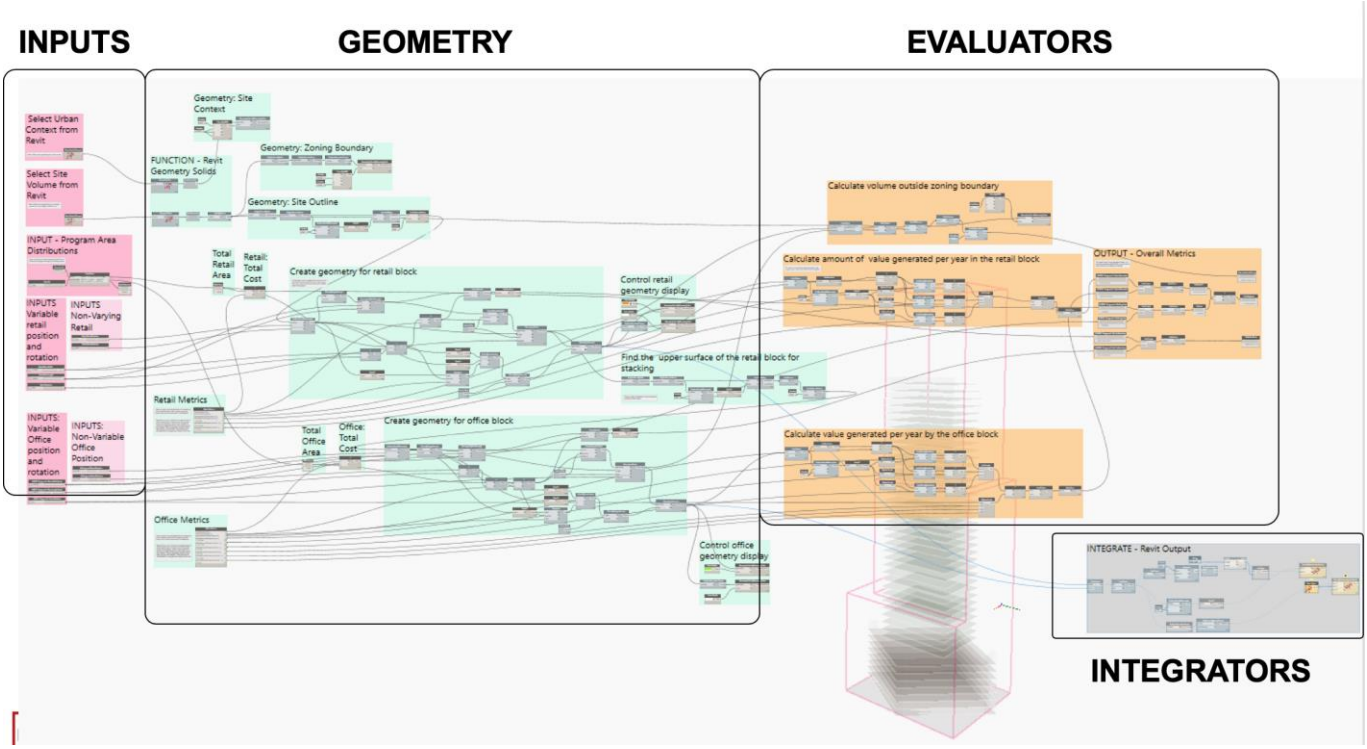
This study looks at the retail and office distribution and configuration for a building on an urban site.

- 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



Open the Revit file "\SampleFiles\03.0_BuildingMasserComplex\ BuildingMasser-UrbanContext2019.rvt" and then the Dynamo file file "\SampleFiles\03.0_BuildingMasserComplex\ BuildingMasser-2programsBiLT_NA2019-START.dyn"

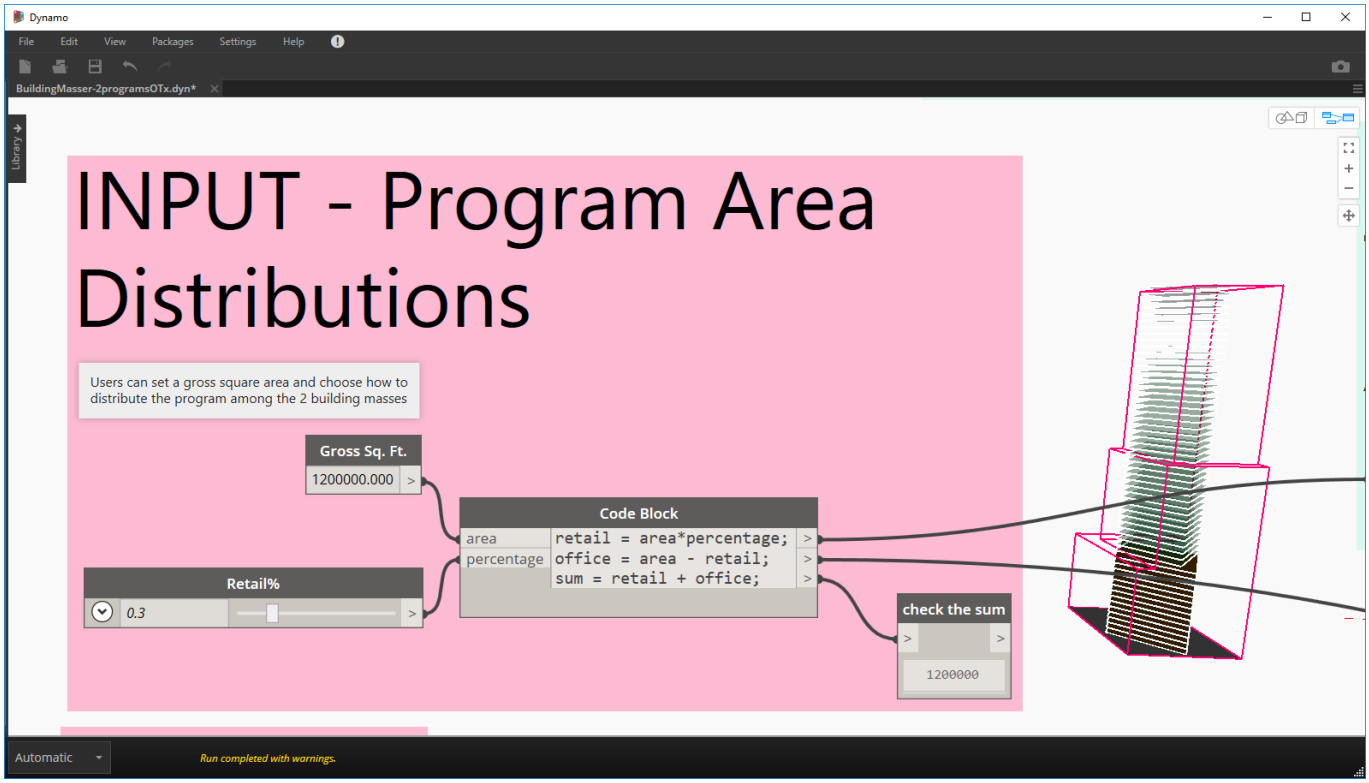
The graph contains the same groupings as our previous example, just with a bit more complexity.



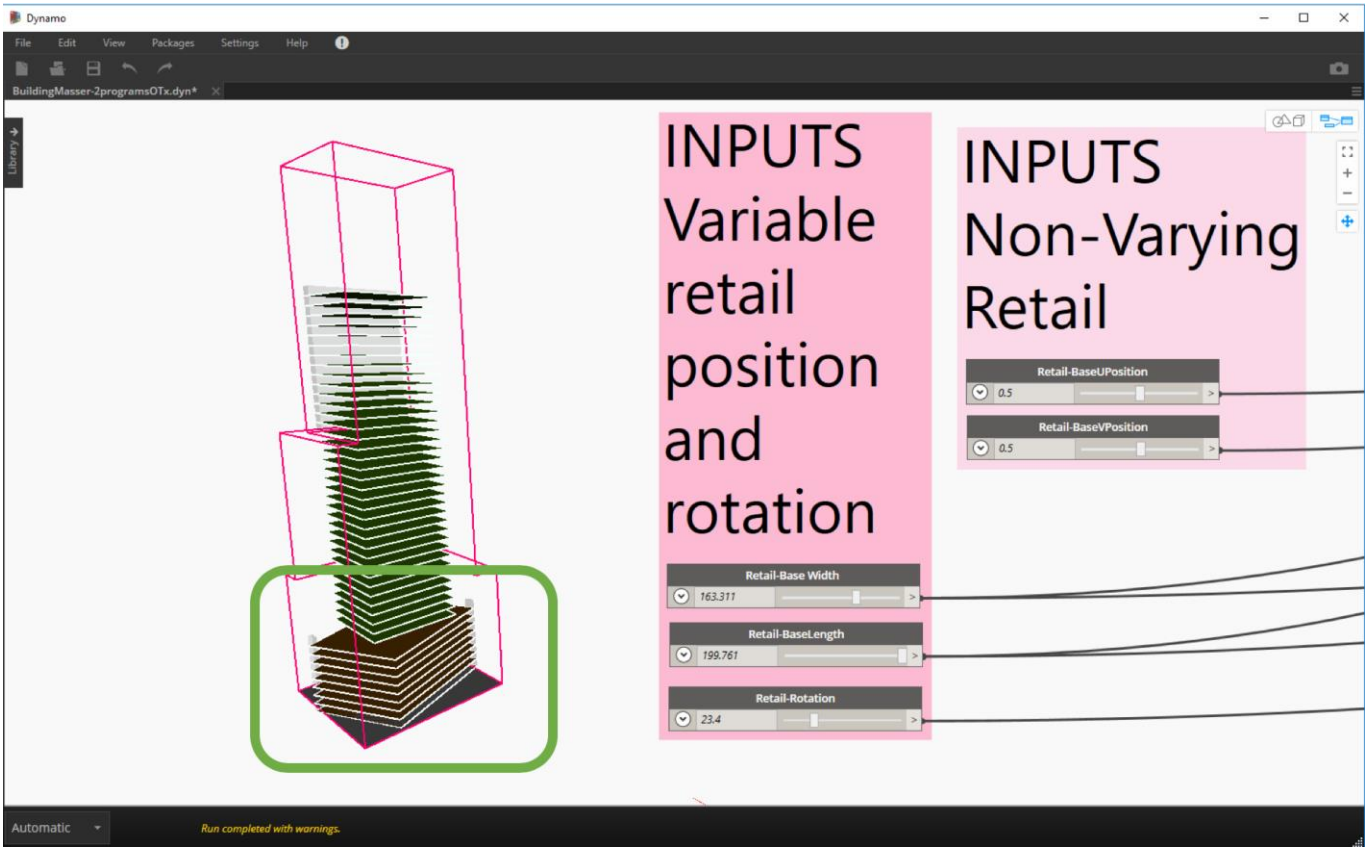
BuildingMasser-2programsBiLT_NA2019.dyn

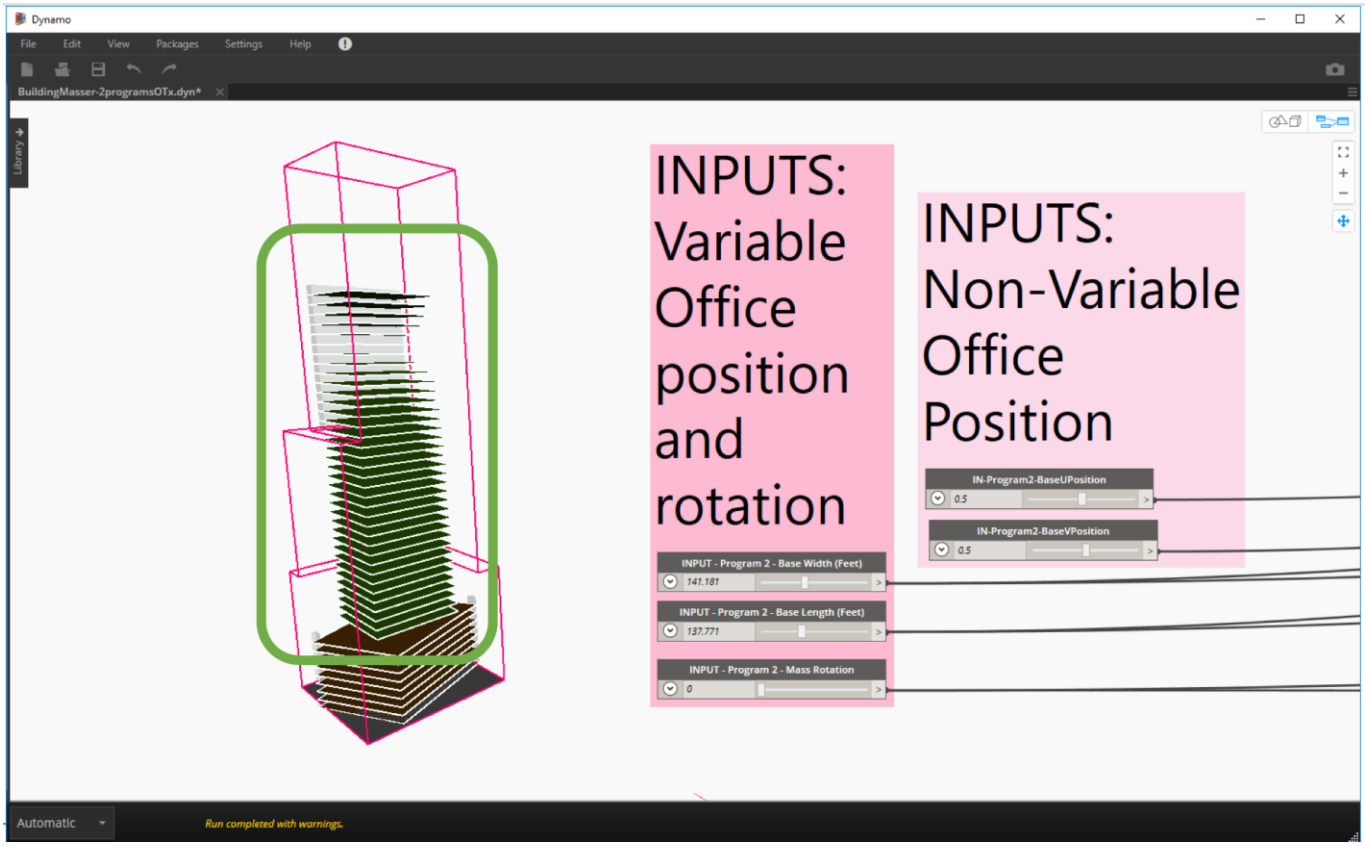
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

First let's look at the inputs. Users can set a gross square area and choose how to distribute the program among the 2 building parts of the building mass. Each building mass chunk can be controlled via length, width, and rotation with optional controls for position. These controls are set to not vary in Refinery, in order to focus the study on the size and rotation of the chunks.



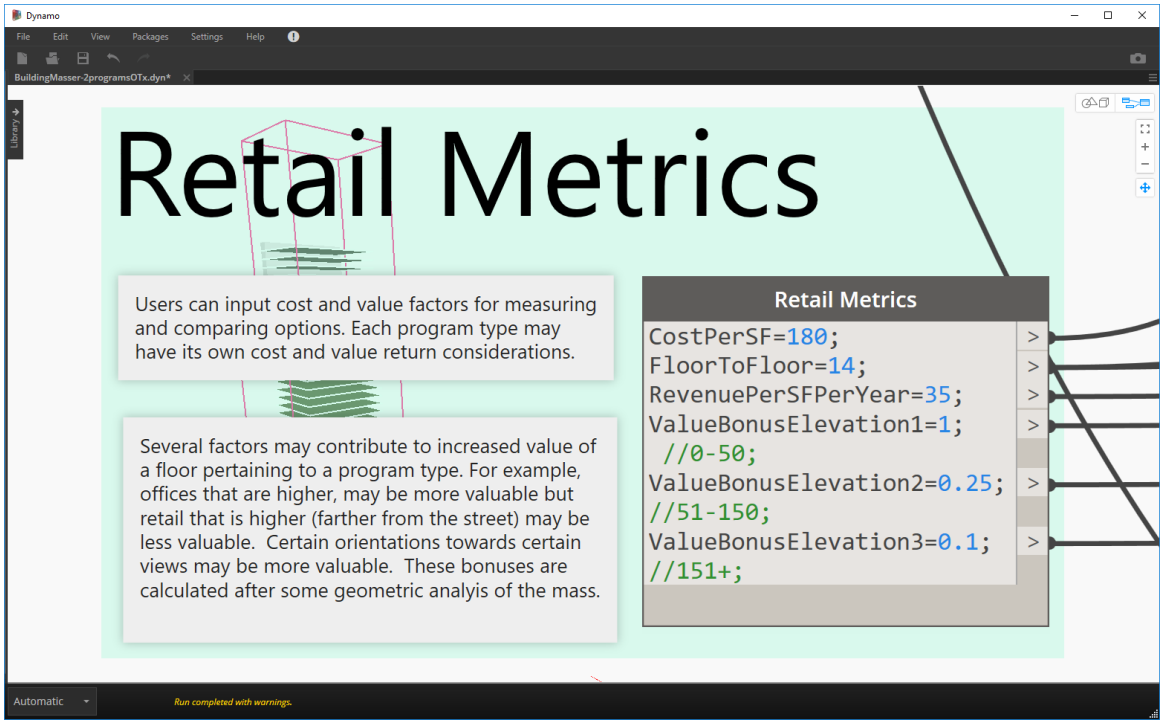
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

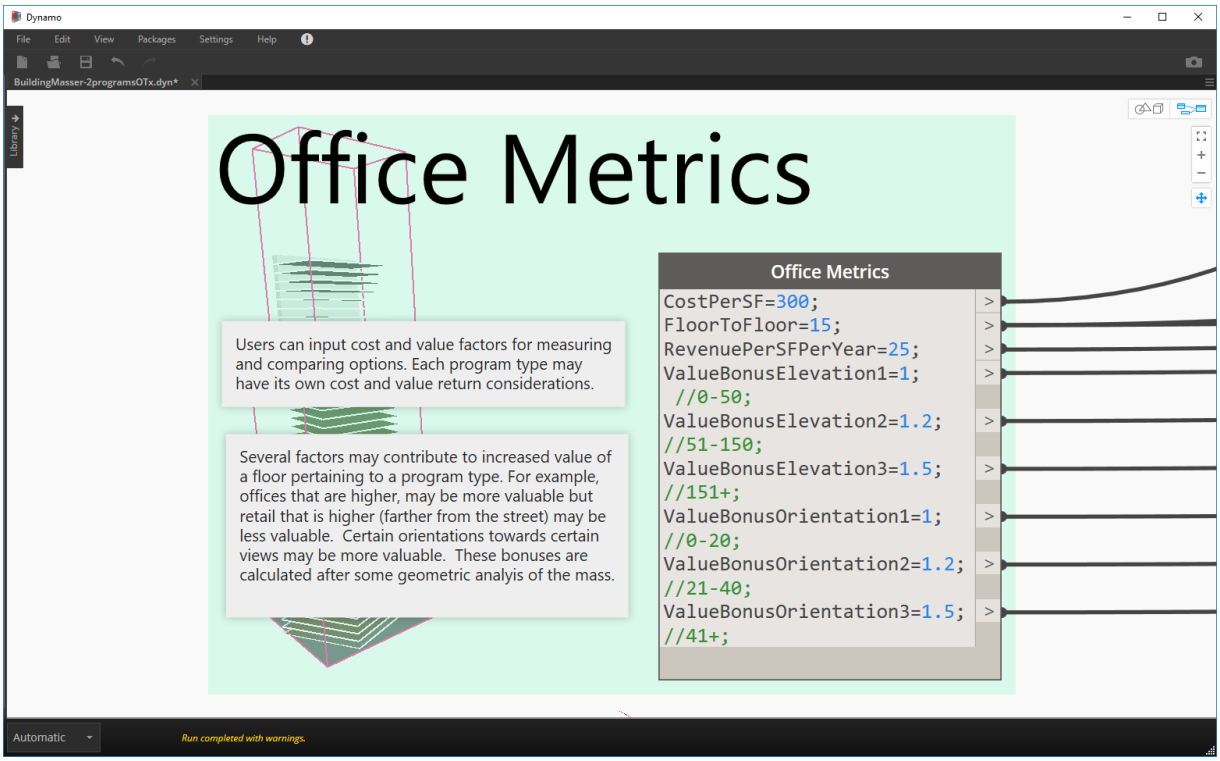




Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Next let's look at the metrics that are used to guide the sizes and rotation of the building mass.





Summarizing and comparing the metrics:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

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

The next steps is to use the inputs to create geometry that can be flexed. Open the script to explore in more detail, but the primary functions of geometry creation in the graph are grouped as follows:

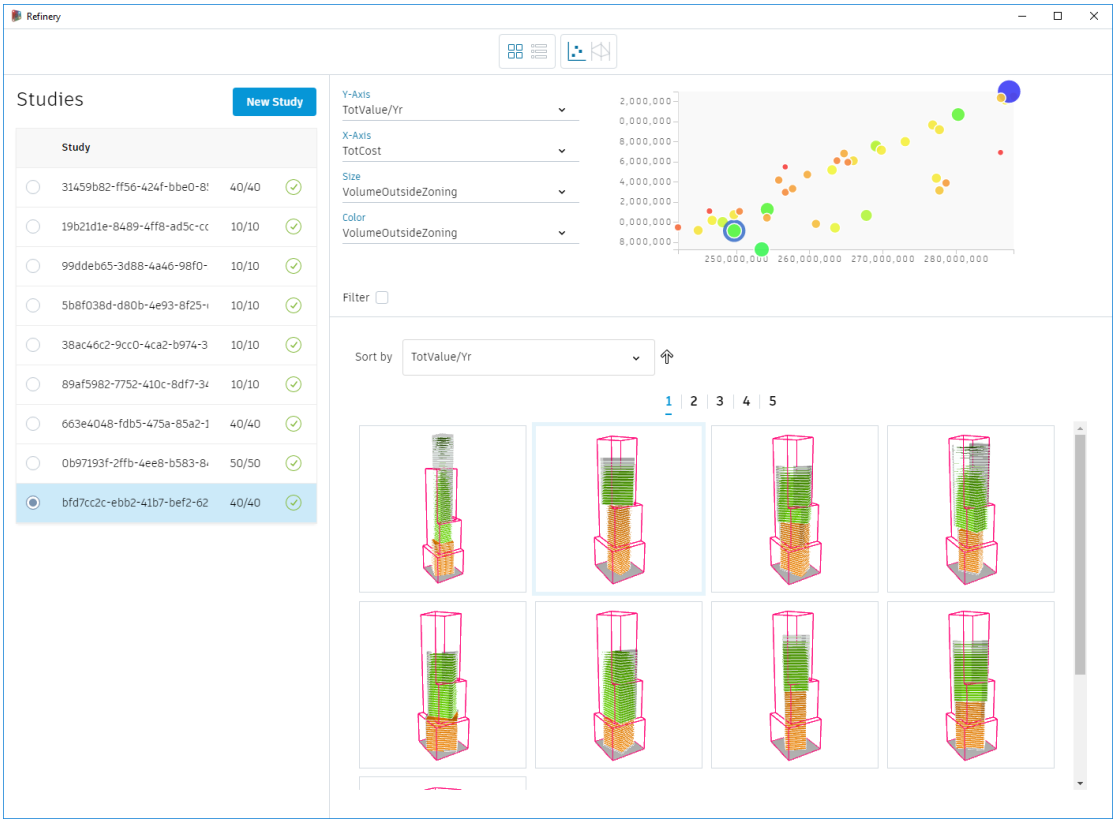
1. Create geometry for the retail block
2. Find the upper surface of the retail block for stacking
3. Create geometry for the office block (stacked on retail)

Once you have the geometry created and performance metrics designated, you can measure the outcomes via pretty basic mathematical calculations for:

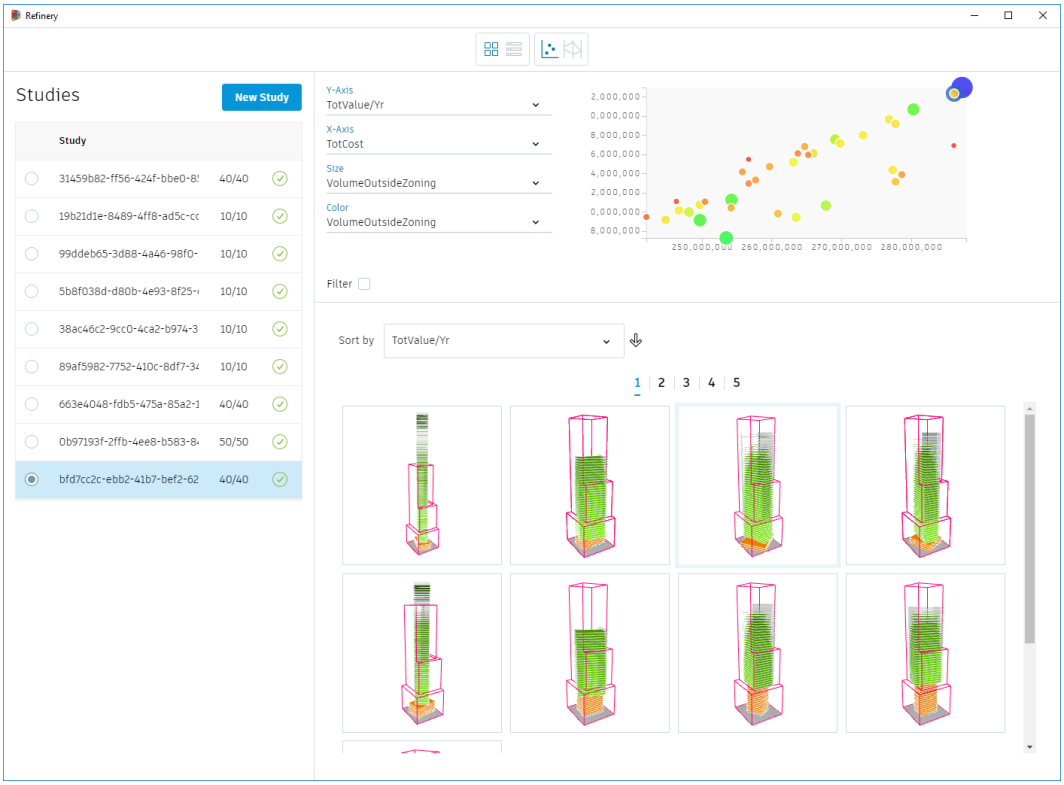
1. Volume Outside Zoning
2. Total Value / Year
3. Years to Payback
4. Total Cost
5. Total Exterior Surface Area

Once the logic is described in Dynamo, you can then automate the generation of design alternatives that you can measure and rank against each other. Start by using the "Randomize" generation method to get an idea of the design space.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

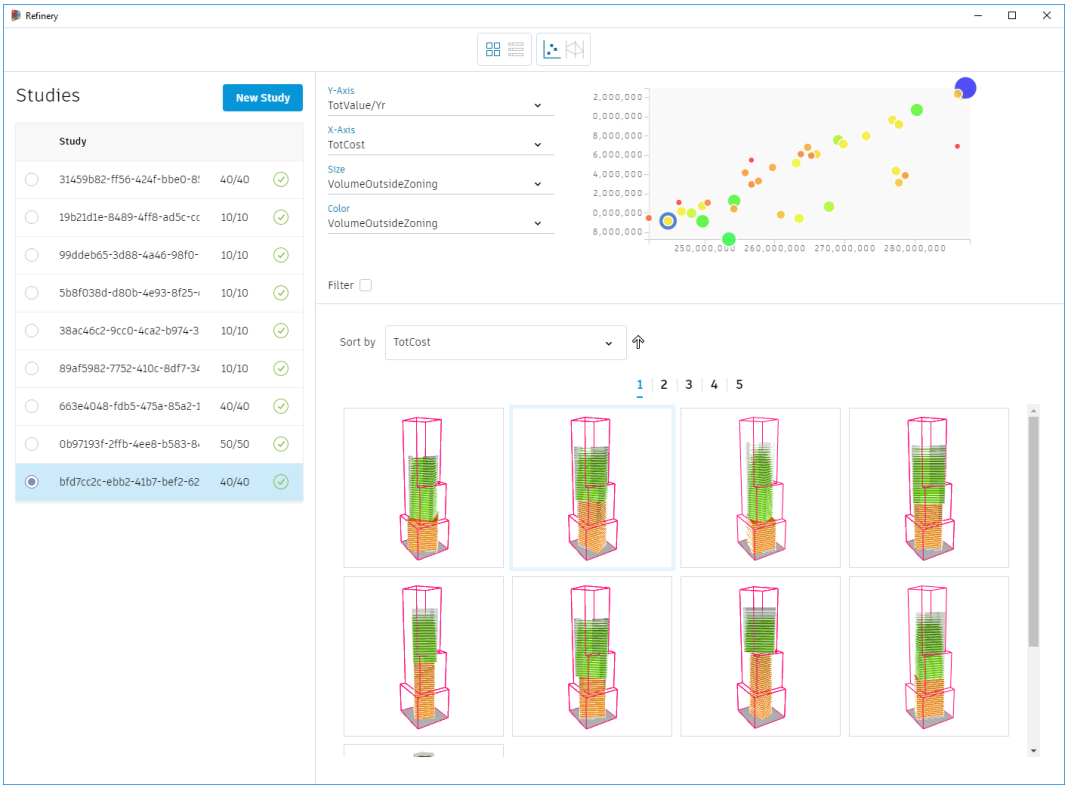


40 Randomized runs showing least TotalValue/Year

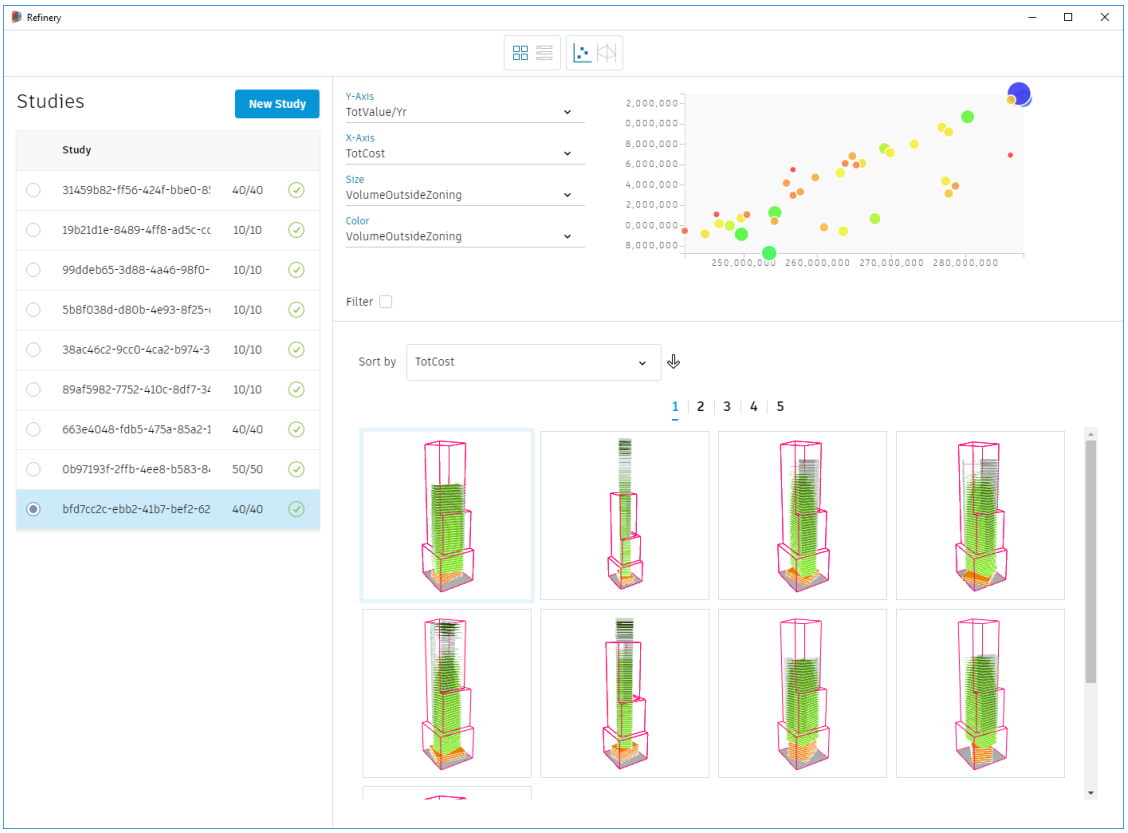


40 Randomized runs showing most TotalValue/Year

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

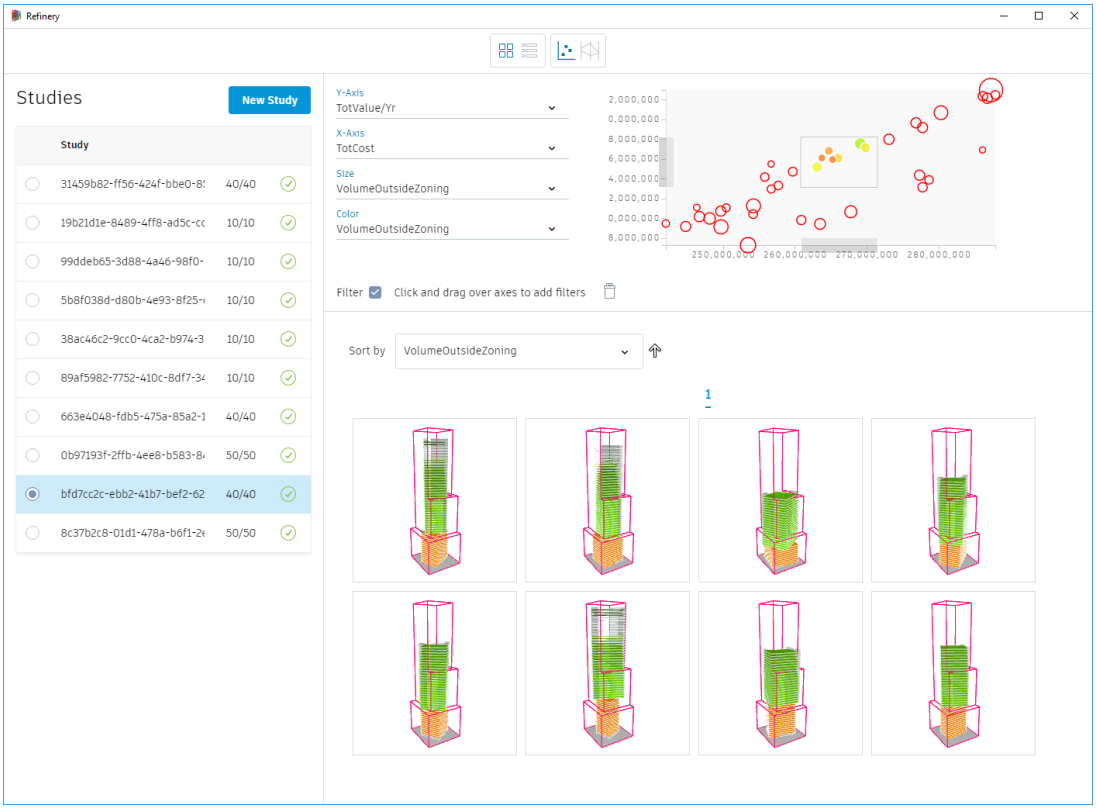


40 Randomized runs showing lowest cost



40 Randomized runs showing highest cost

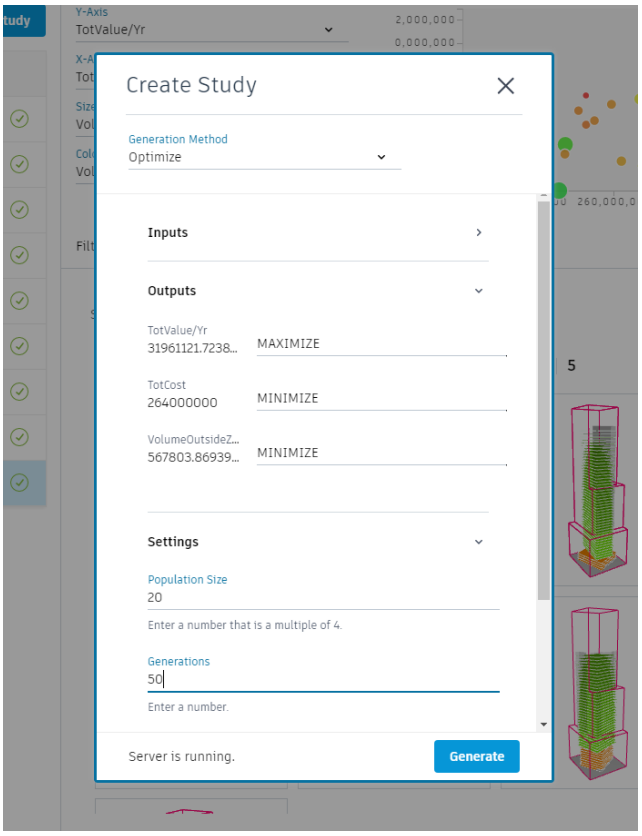
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



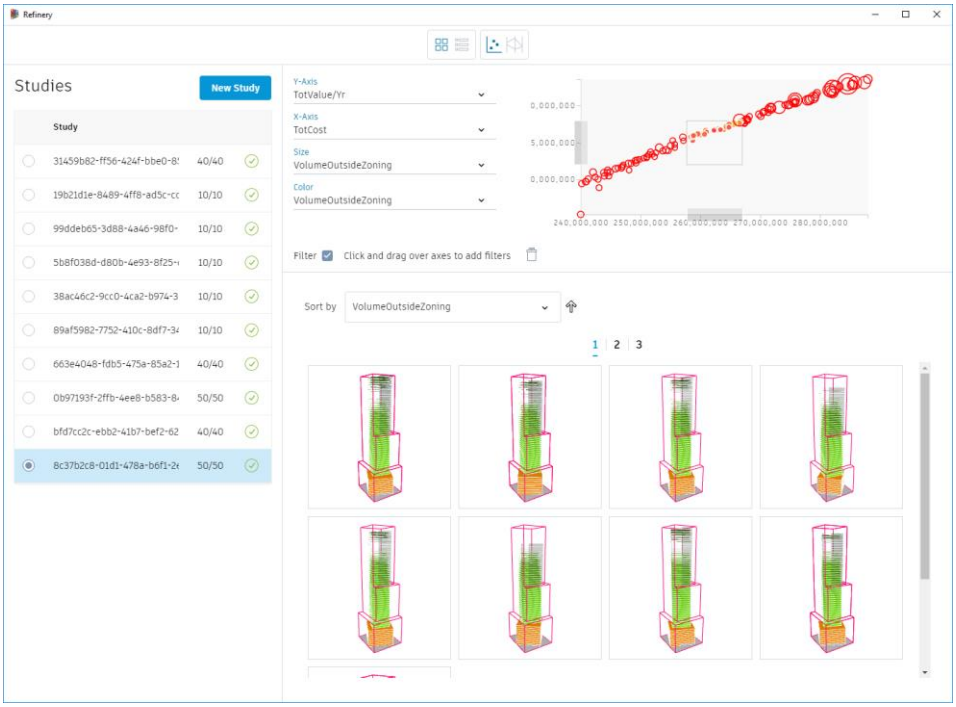
Use filtering to select options with median cost and value, sort by lowest Volume Outside Zoning

Finally, you can use the Optimize method to set goals before creating runs. If you set the Outputs to maximize Total Value/Yr, minimize Total Cost and minimize VolumeOutsideZoning, the computer will evolve the solution set towards designs that meet your stated goals.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Setting up for an optimization run in Refinery



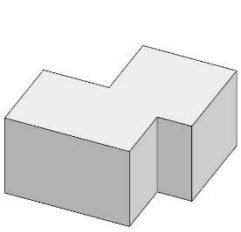
20x50 Optimization run filtered for medium Cost, medium Value, and sorted for lowest VolumeOutside Zoning

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

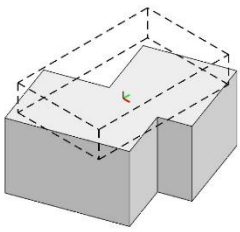
Example 3.5: Stealthy RoofScapes

In this example we will show how to re-purpose an Evaluator from one Dynamo graph and re-use it in a new project.

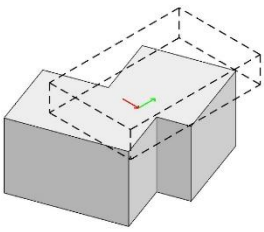
Specifically there was a Dynamo Hackathon in London where a project was developed called Stealthy Roofscapes. The project team was composed of Bogdan Davydov, Wiktor Kidziak, and Thomas Corrie. The main project goal was to develop a system that tried to optimize for conflicting goals: Maximizing Building Volume but at the same time keeping the visible profile of the building at street level at a minimum. The project team developed a method of testing views from sample points on the street and determining how much of the building façade is visible.



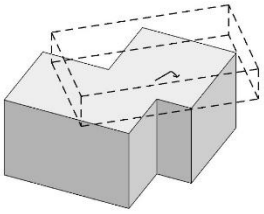
Existing Building



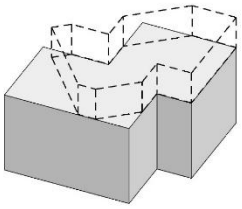
Cuboid placed at centroid of roof aligned with coordinate system
- Length
- Width
- Height



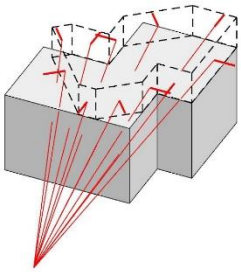
Cuboid moved in x and y
- XOffset
- YOffset



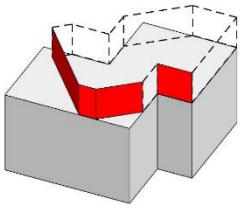
Cuboid rotated
- Rotation



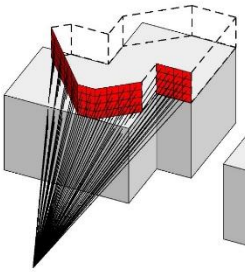
Cuboid trimmed to extent of existing roof boundary



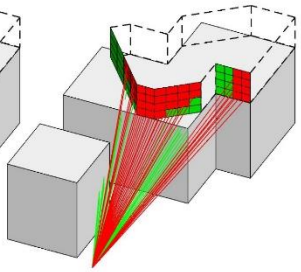
For each viewpoint, establish the angle between the normal of each face of the addition (ignoring top and bottom faces) and the sightline



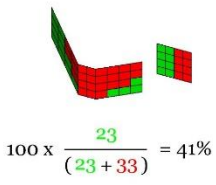
Faces with angles to the sightline of less than 100° face towards the viewpoint and will be considered



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



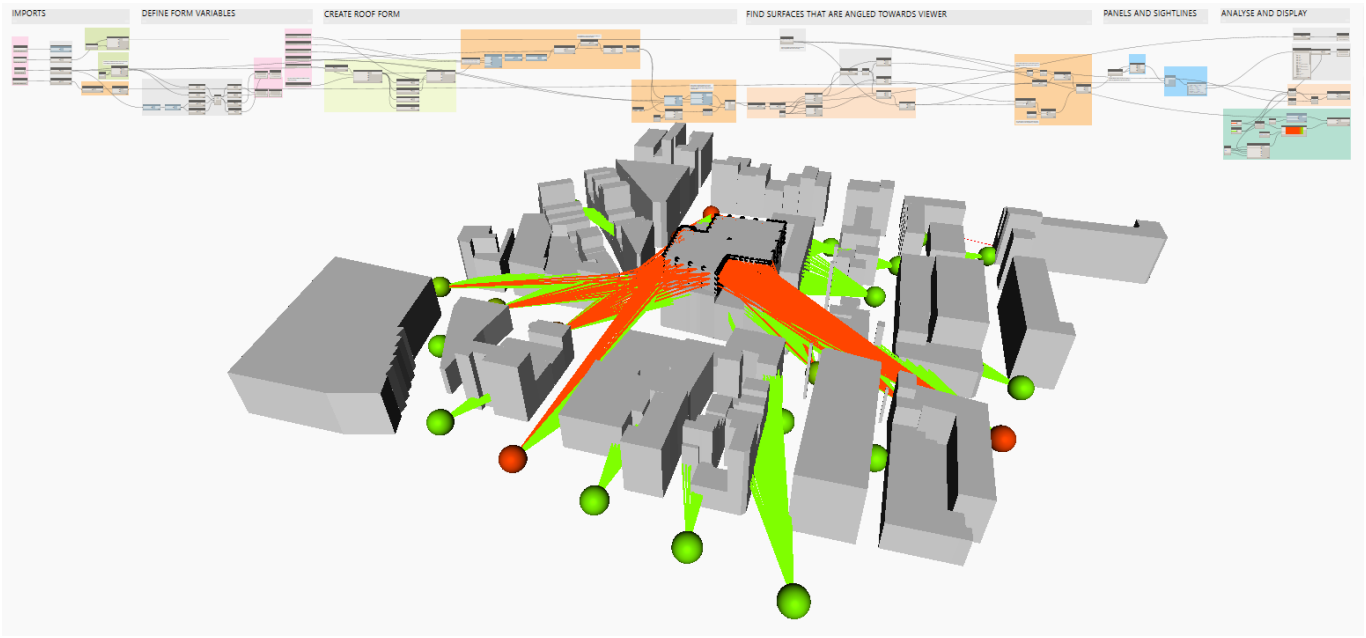
The sightline is tested for intersections with the surrounding buildings (including the existing building below) and if it intersects it the panel is considered not to be seen (green)



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

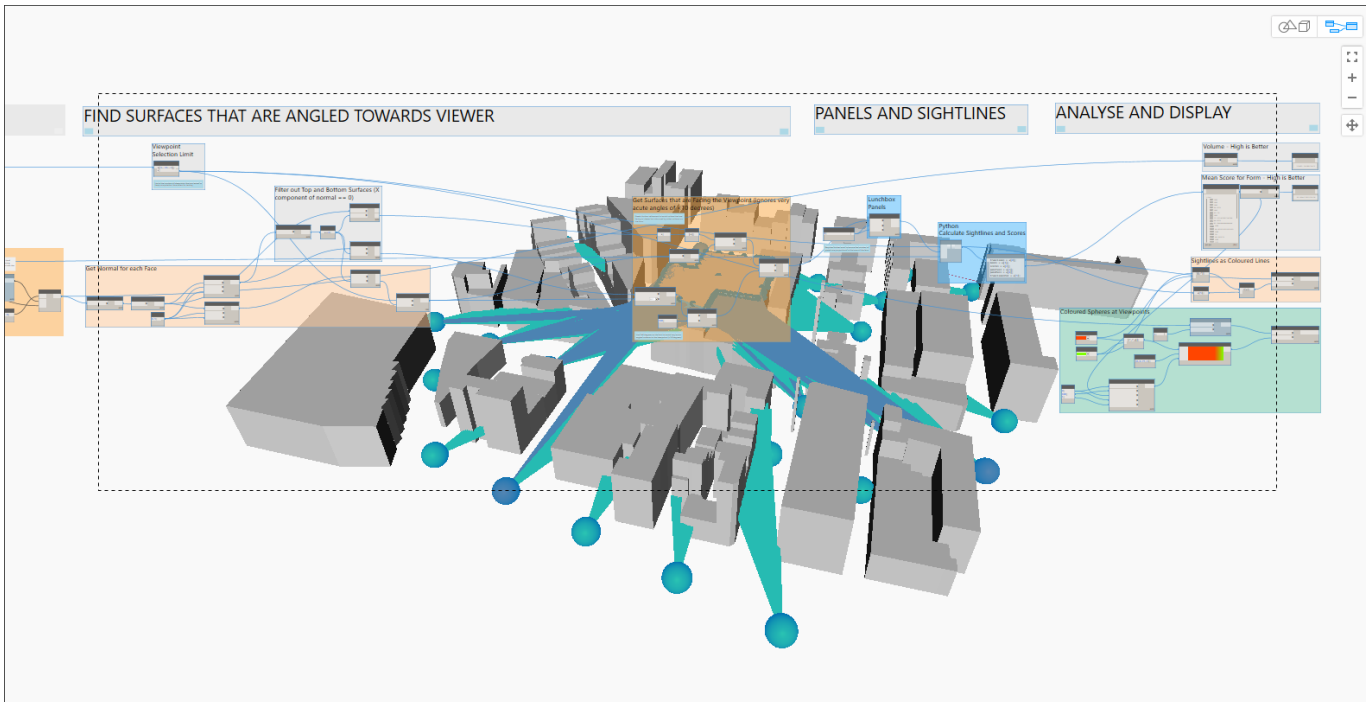
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Let's take a look at their work. Open "SampleFiles\03.5_StealthyRoofscapes\dyn\StealthyRoofscapes_CreateForm-START.dyn", it should look like this:

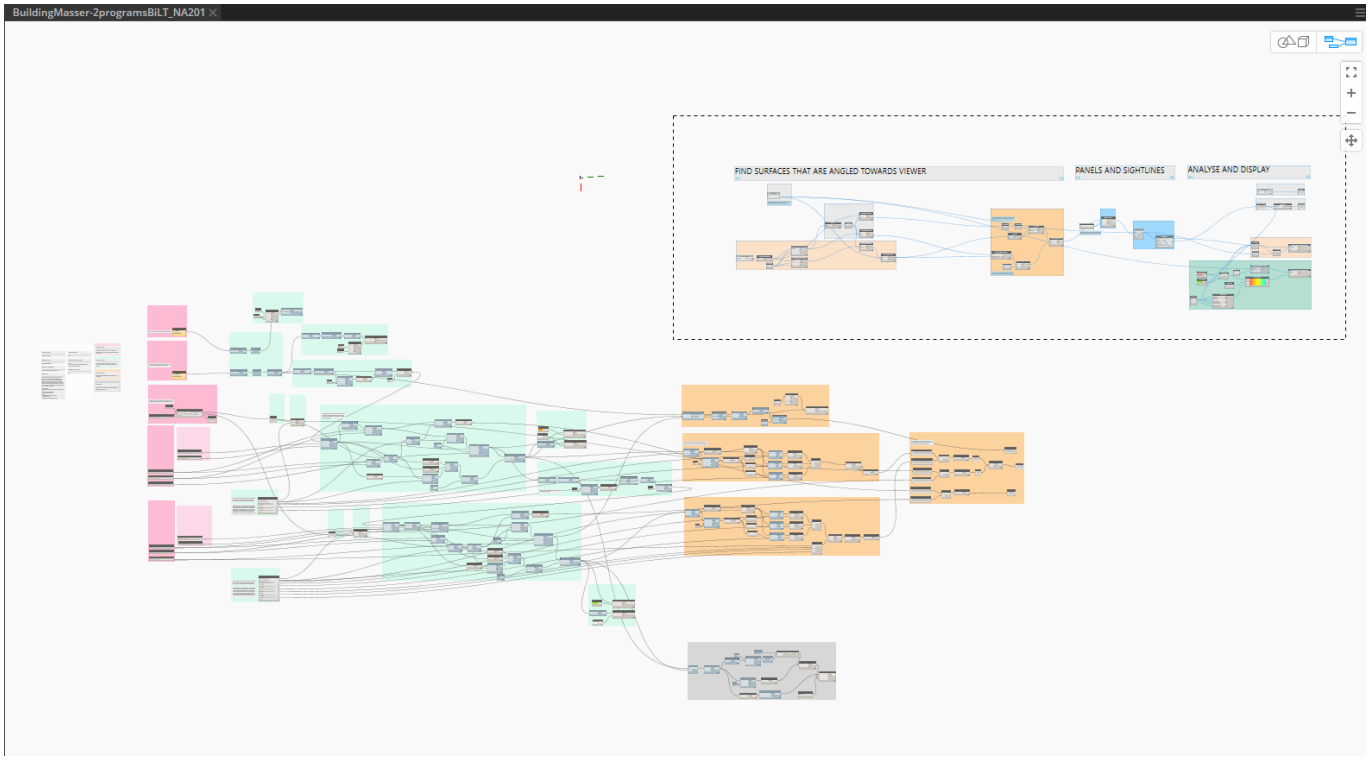


- Box Select this area of the graph
- CTRL-C to Copy to the Clipboard

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

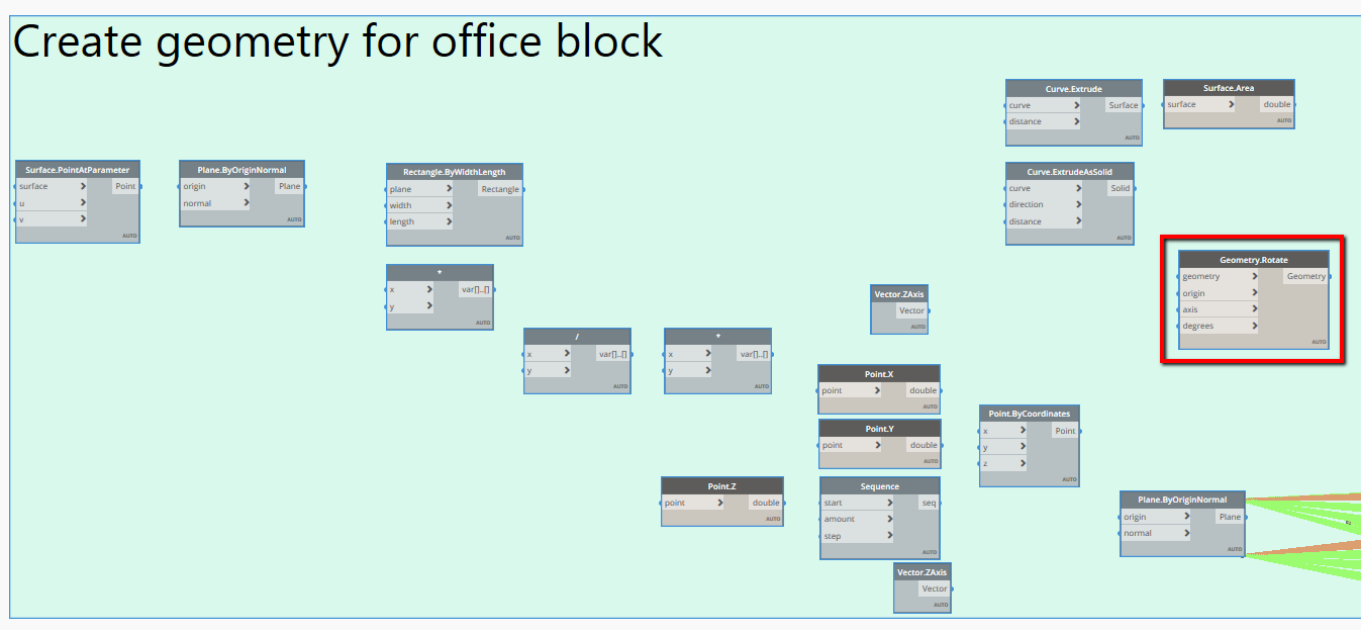


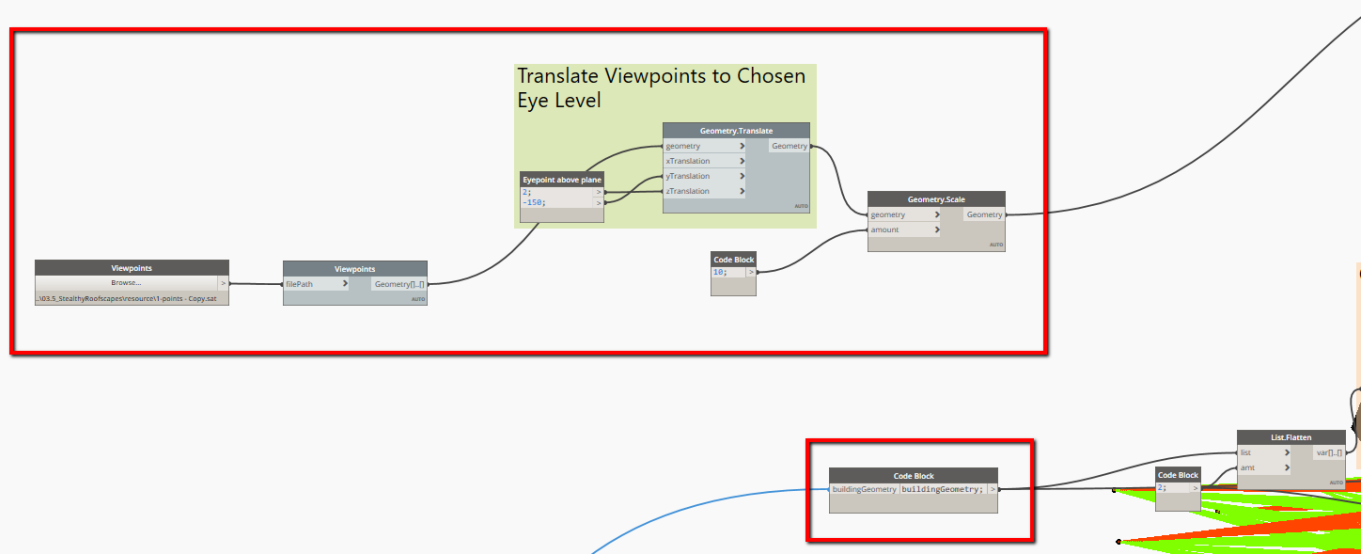
- In Same Dynamo/Revit Session
- Open Building Massing RVT/DYN from the previous example
- “BuildingMasser-2programsBiLT_NA2019.dyn”
- CTRL-V to Paste the copied nodes from the clipboard



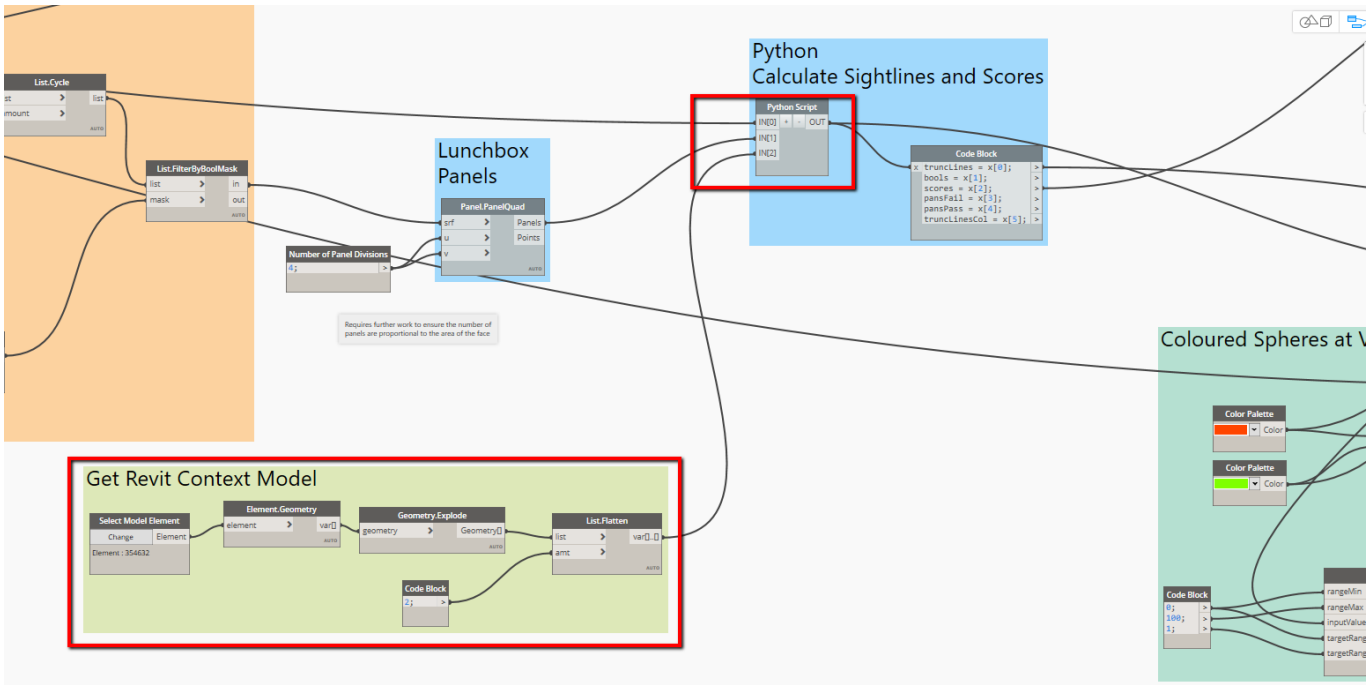
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Now let's wire up the inputs to the Evaluator. We need three main inputs: the sample points, the building geometry, and the context geometry.



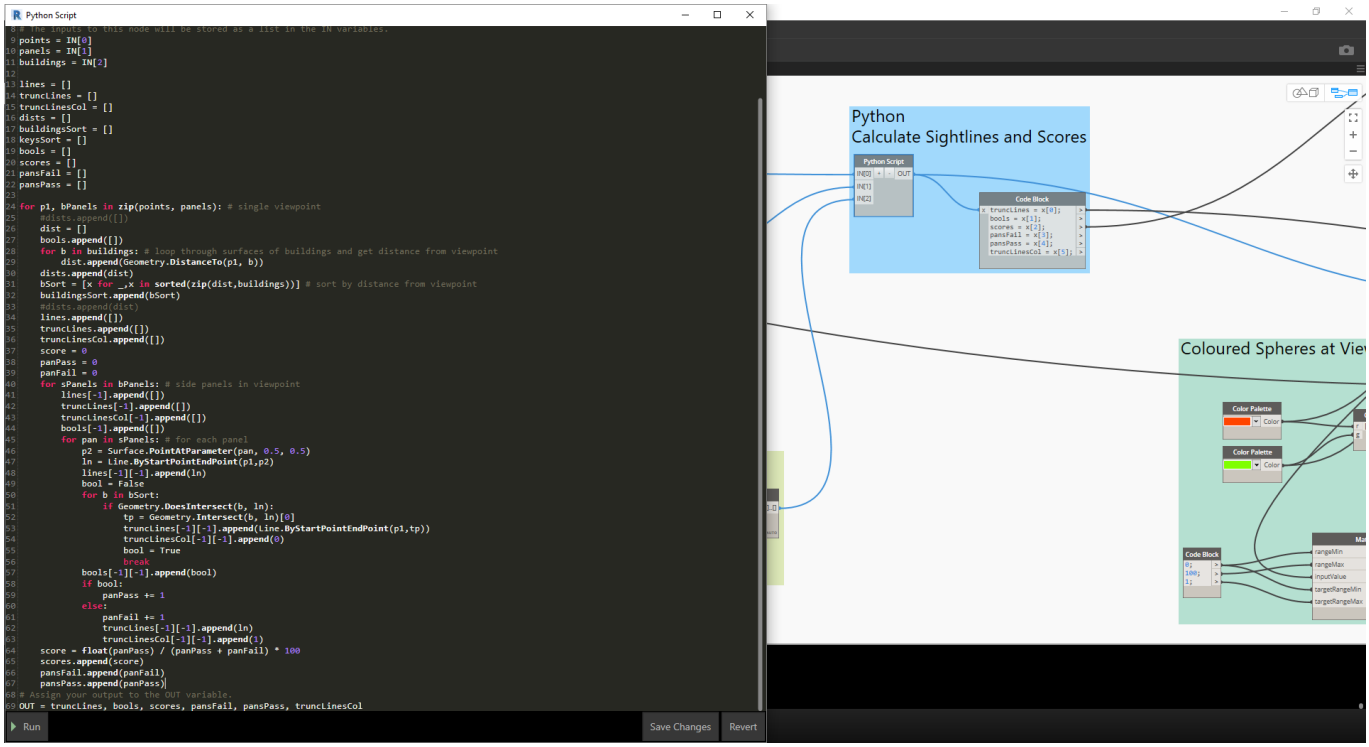


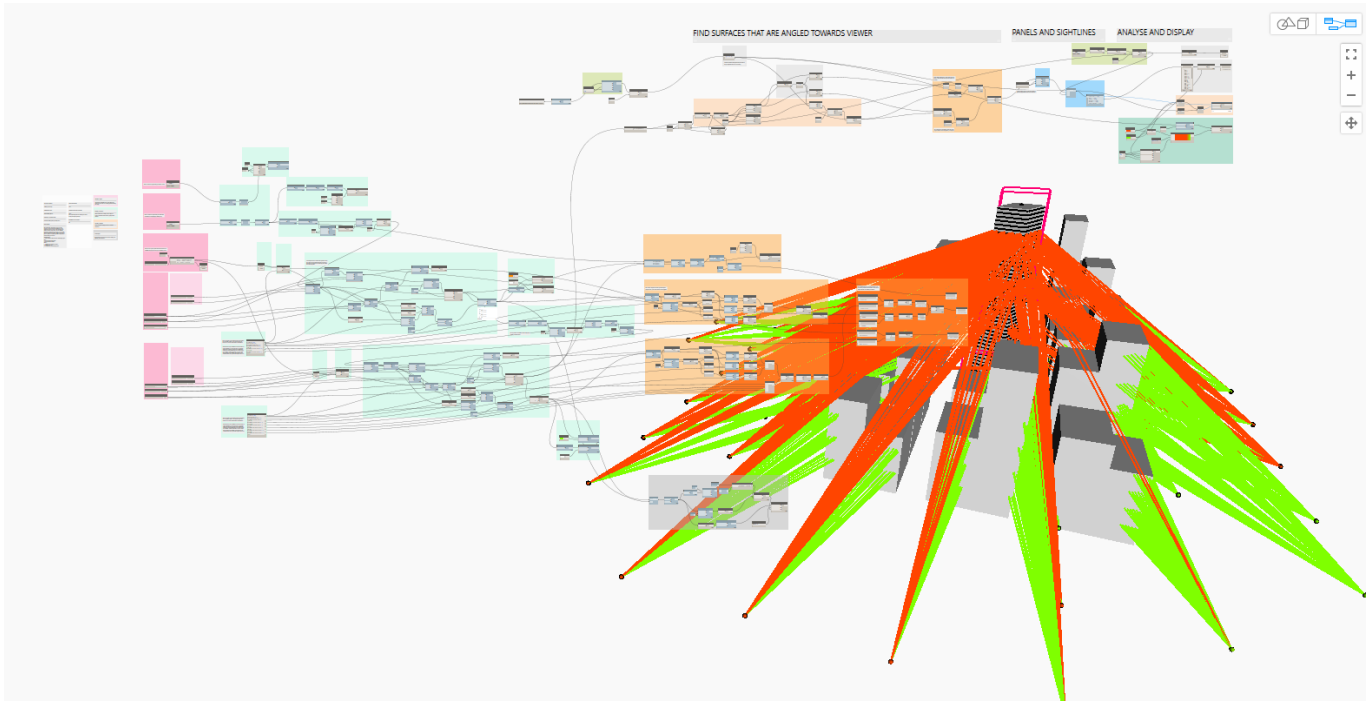
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



We can take a look at the Python code that does the main evaluation loop. It basically performs ray casting between each of the points on the façade and each of the sample points. If the ray intersects the context geometry it is considered obstructed. If it does not there is a clear view between the sample point and that part of the building. A weighted average is computed to create a score of how much building is visible, this can be used in Refinery to prefer options that are less visible from the street.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)





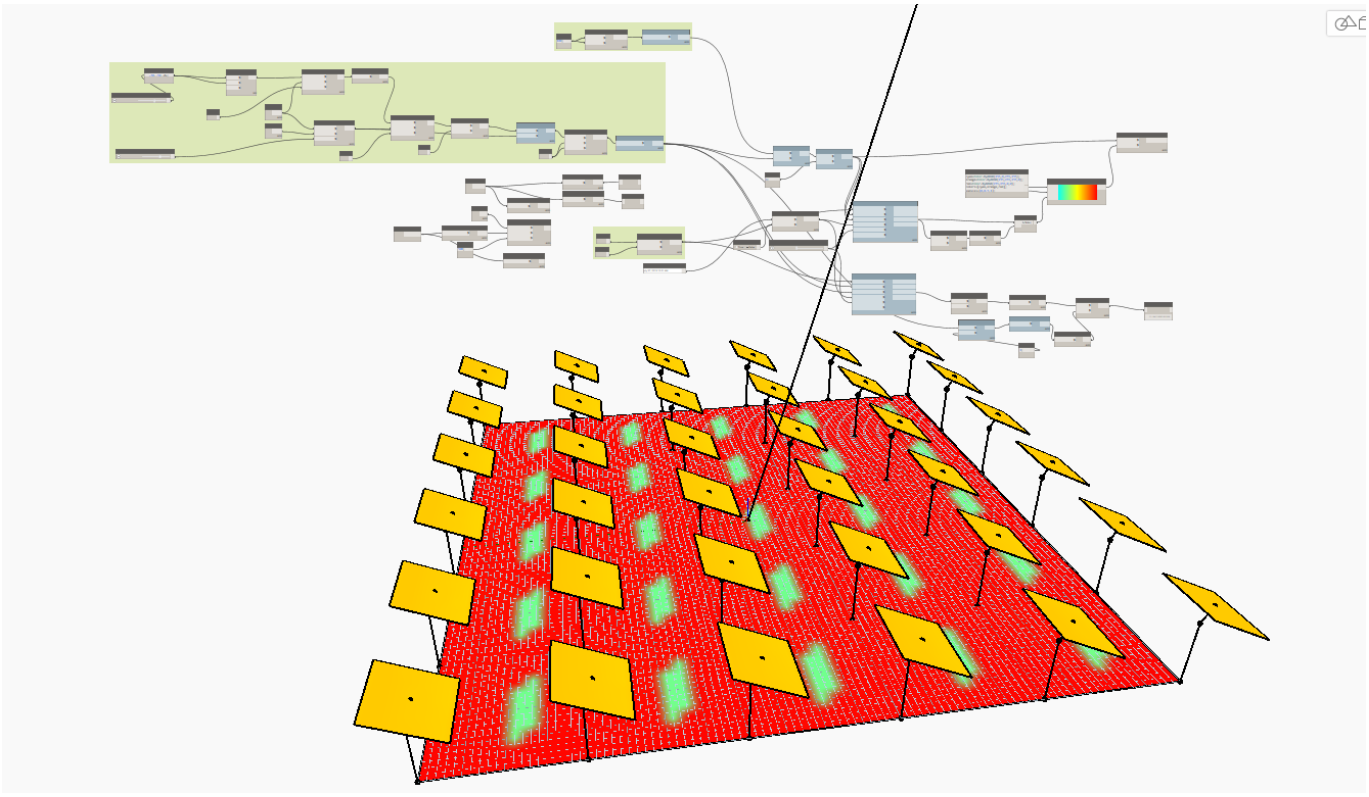
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Example 4: Solar Radiation

This study looks at how leverage solar radiation analysis data coming from the Solar Radation for Dynamo package.

- Variable Inputs:
    1. PV Collector Tilt Angle
- Goals:
    1. Maximize Average Intensity

Open "\SampleFiles\04_Solar Analysis\ 04_Solar Analysis.dyn"

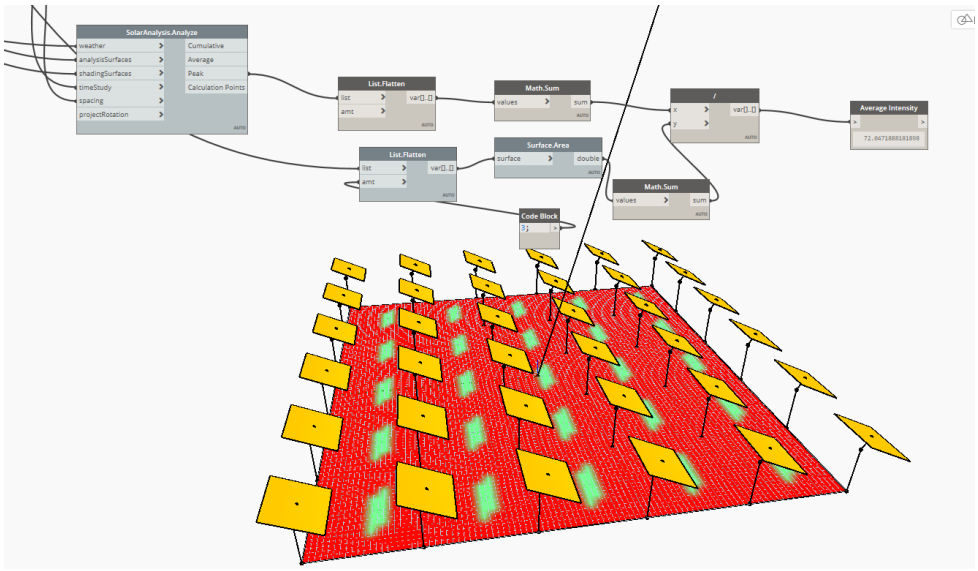


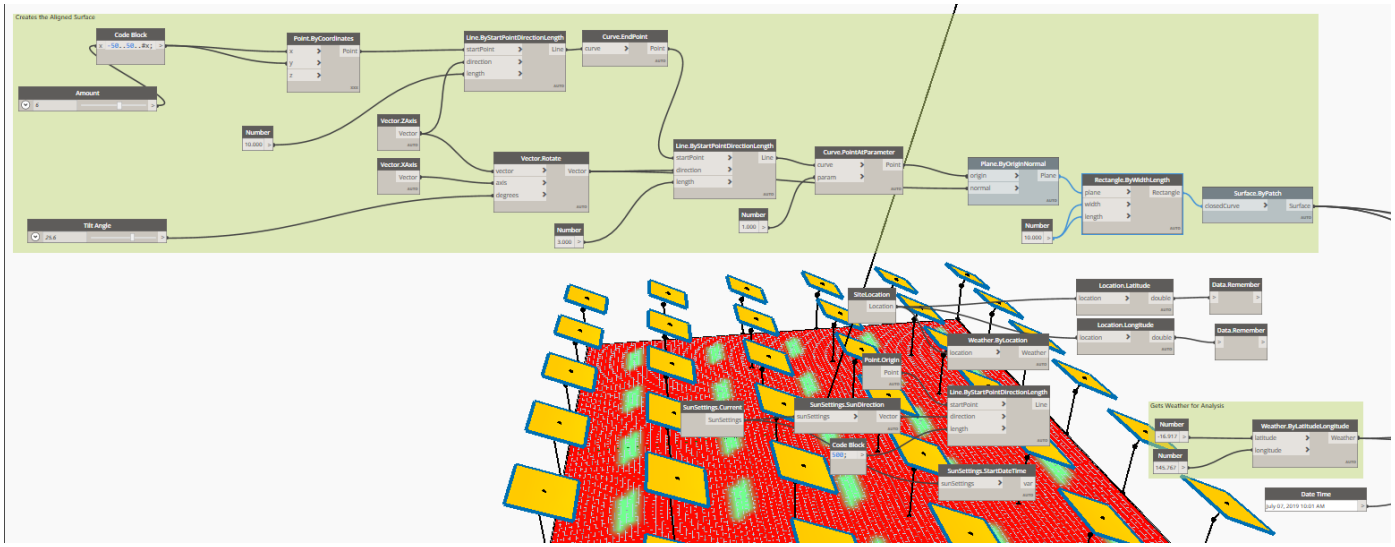
You will notice a set of photovoltaic panels arrayed. They have a 'tilt angle' in one axis and the goal of this exercise is to find the optimal angle to capture the most solar radiation for a given time span.

We are using the Solar Radiation for Dynamo package to perform the analysis and then are tabulating the recorded outputs for all the sample points. Let's review the graph to see how it works:

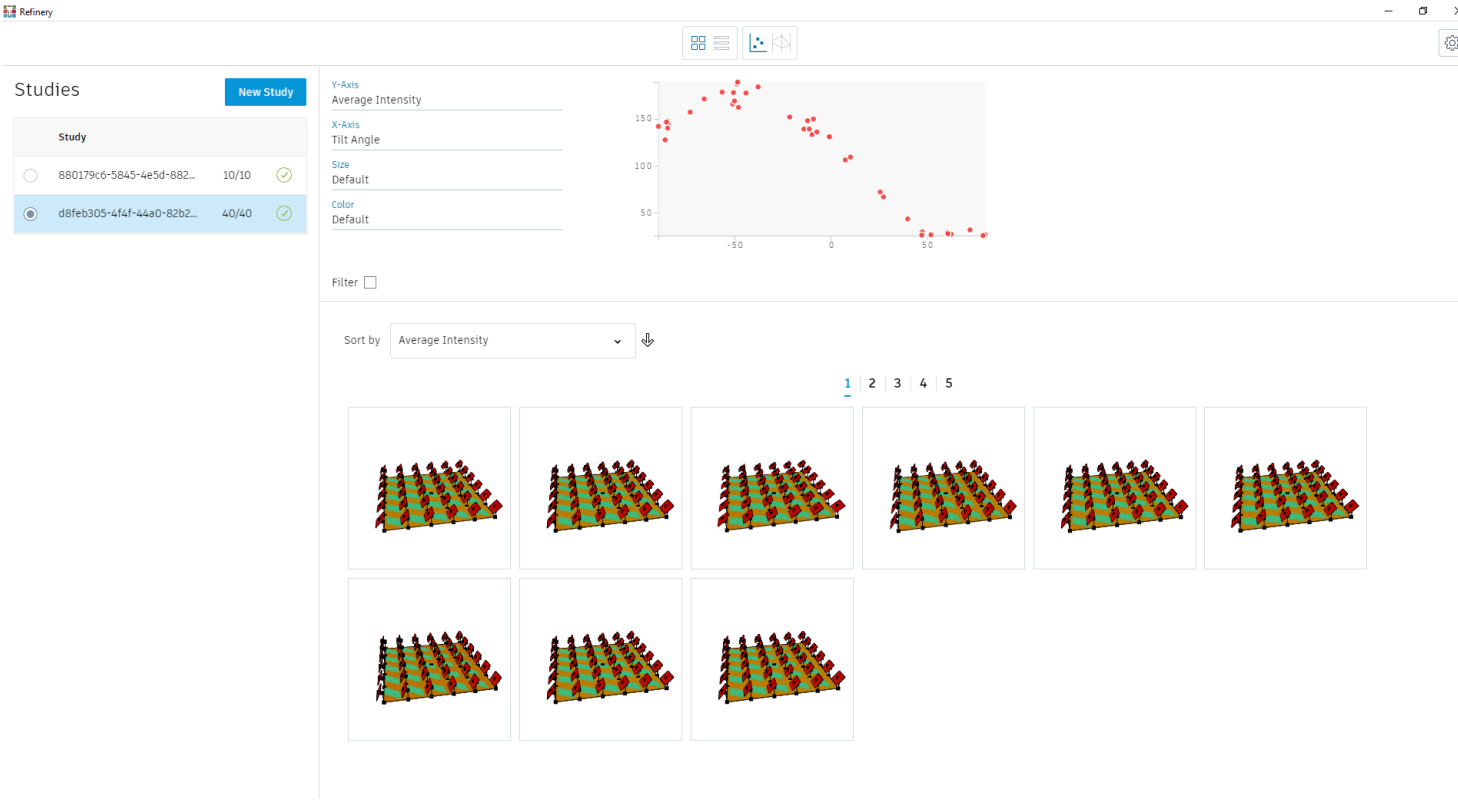
2.3 2.4 - How to Train Your Model with Project Refinery

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)





Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



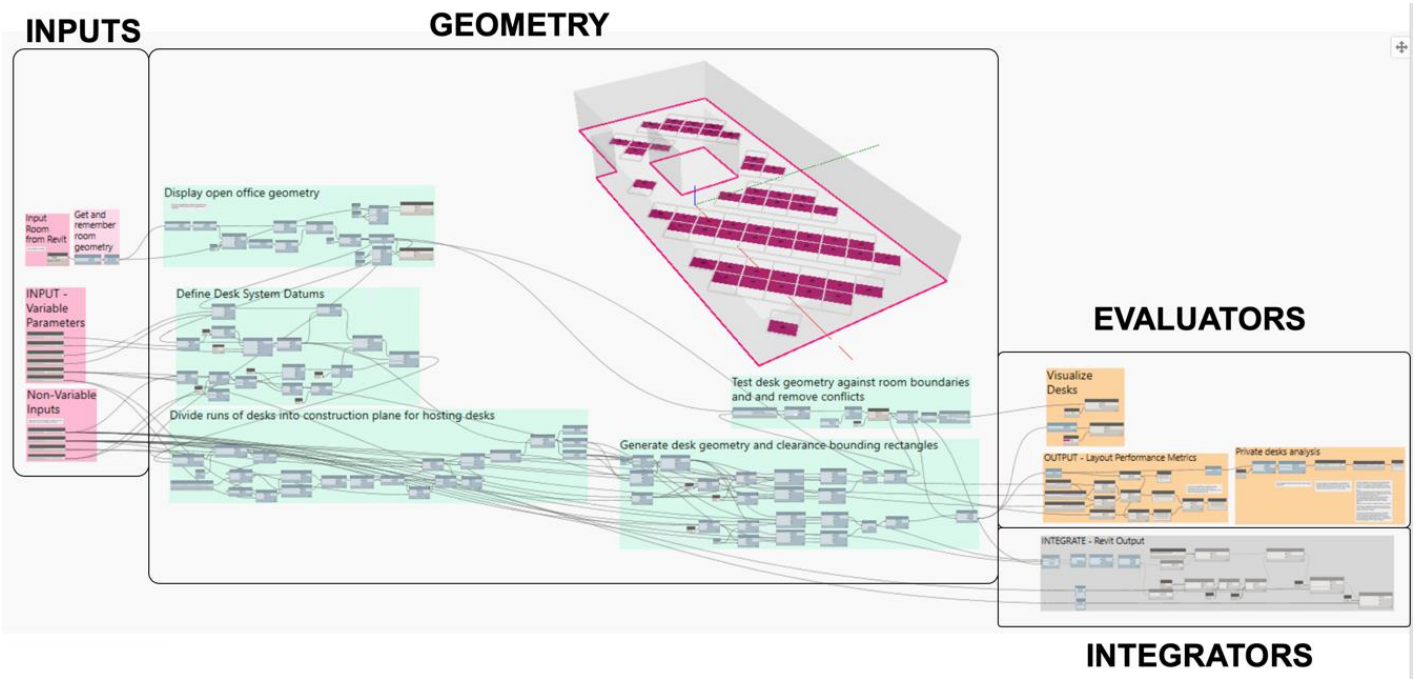
Example 4: Open Office Layout

This script lays out desks given a Room element coming from Revit. In this study we will look at desk layouts that gives us the most desks but also the most private desks while maximizing desk area per person and maximizing space utilization.

- 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. Minimize unutilized space
    4. Maximize number of private desks

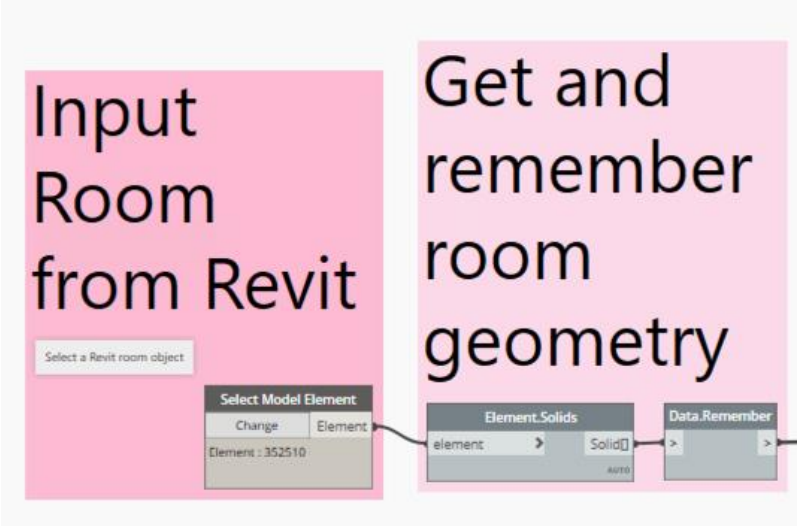
Once again, the graph contains the same sections as our previous examples:

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



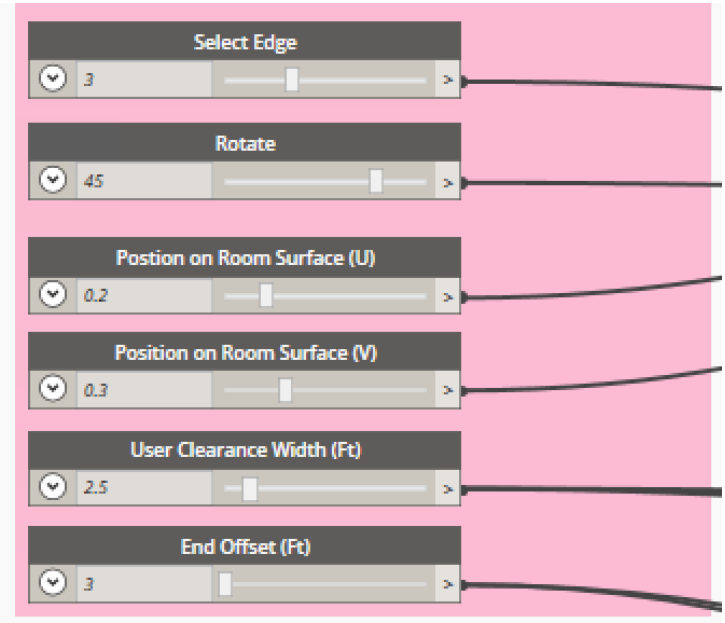
Let's look at each section:

1. Inputs: The first input is a Room element coming from Revit. This gets remembered for use in Refinery.



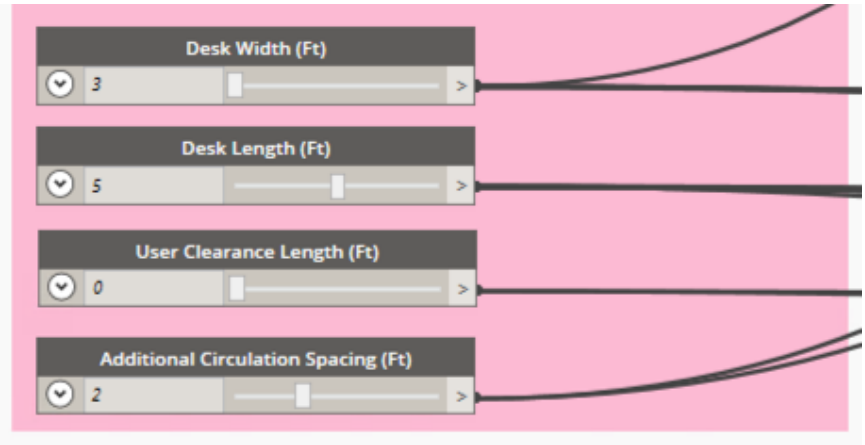
The next 2 groups of inputs are divided into variable and no variable inputs for clarity. The sliders in the first group are set to "Is Input" **on** via the context menu. These are the inputs that we want to vary in our study.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Variable Input Parameters

The sliders in the second group are set to "Is Input" **off** via the context menu. These are inputs that we want to remain constant during the study. Inputs can be changed to vary or stay constant by setting the "Is Input" flag in Dynamo or changing the variation option in Refinery.

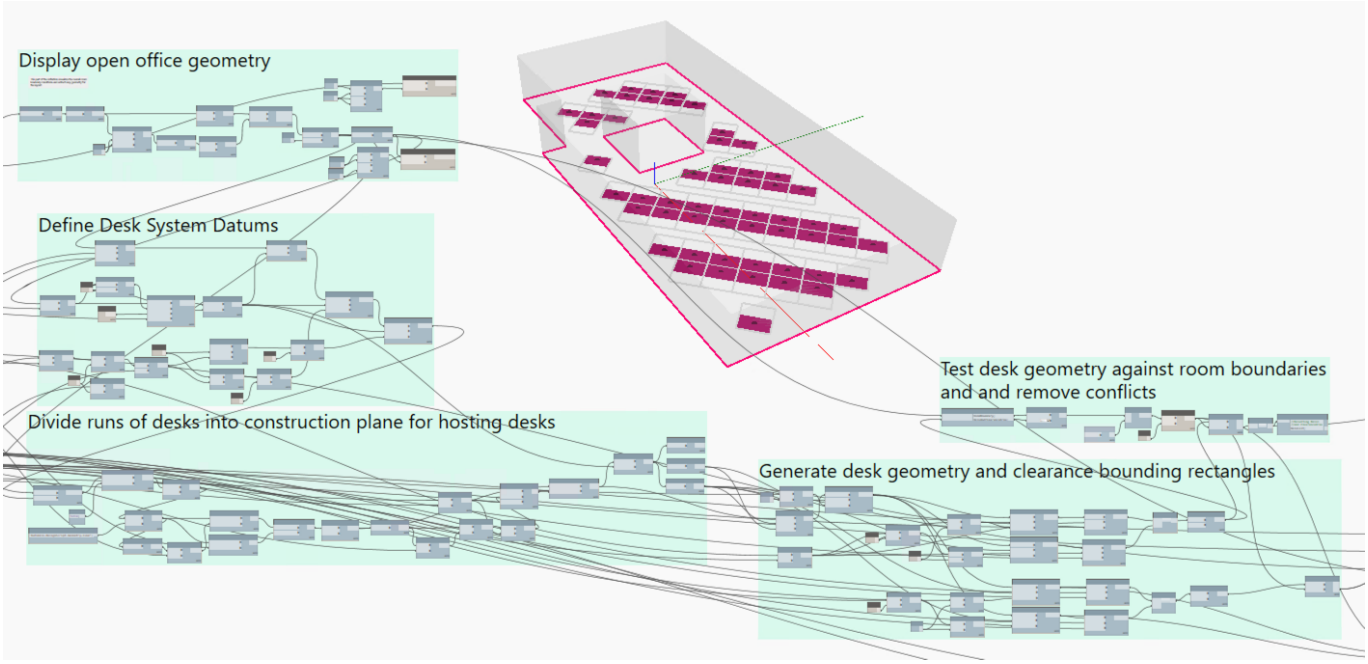


Non-variable Input Parameters

Be sure to consider how the maximum, minimum and step values are set for the range of your sliders so that you can limit the results to desirable values. For example, here, the rotation step value is set to 15 with a range of -90 to 90 so that the system won't return many options with tiny rotation changes.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

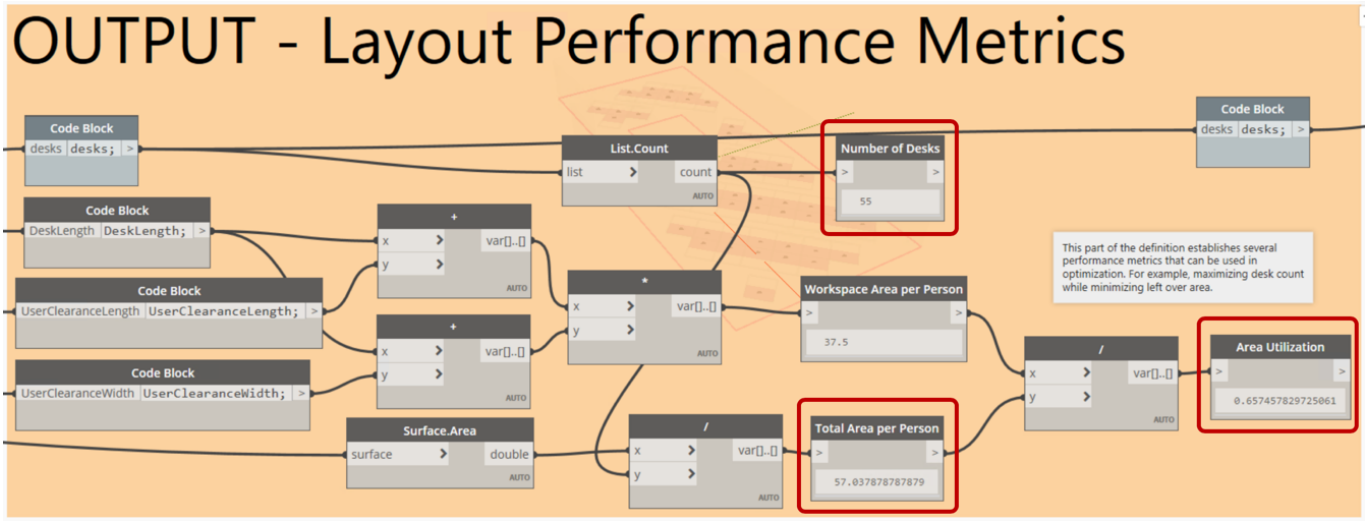
2. Geometry – The green section of the graph contains nodes which create the desk layouts based on the room geometry coming from Revit. Open the graph to explore in more detail, but the basic idea is to process and display the room geometry coming from Revit. Then define a system of datums based off of this geometry. Next, runs of desks can be divided into construction planes for hosting each desk. Then the desk geometry and clearance rectangles can be drawn. Finally, the geometry is compared against the room boundaries and any conflicts are removed.



Geometry section of OpenOfficeLayout.dyn

3. Evaluators. In the orange section of the scripts, the designs are evaluated against our established goals. The first orange section counts the number of desks, calculates the Total Area per Person and the Area Utilization. These nicknamed watch nodes are all set as “Is Output” so that Refinery will recognize them as outputs.

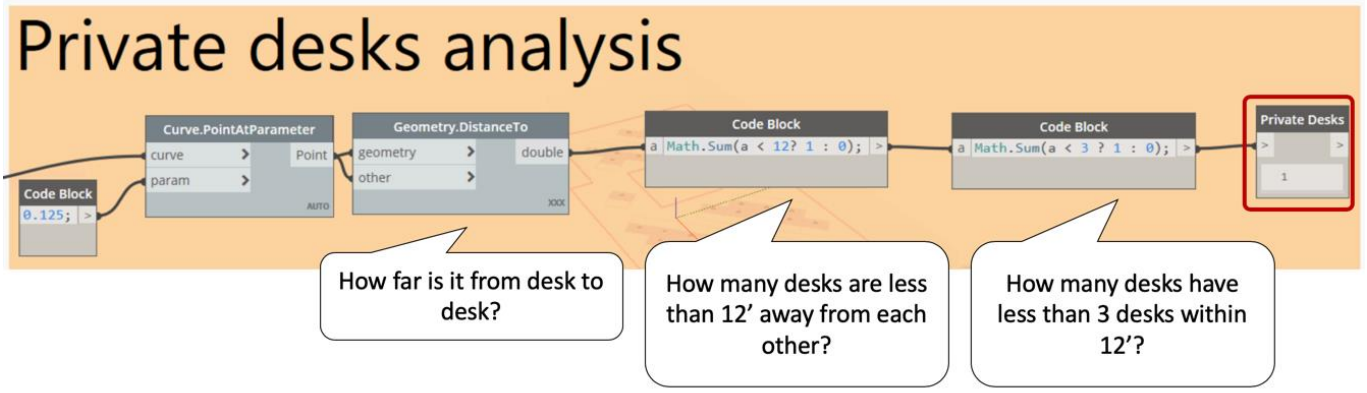
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Desk Layout metrics

In the second orange section, we perform some calculations to count the number of "private desks". This calculation does the following:

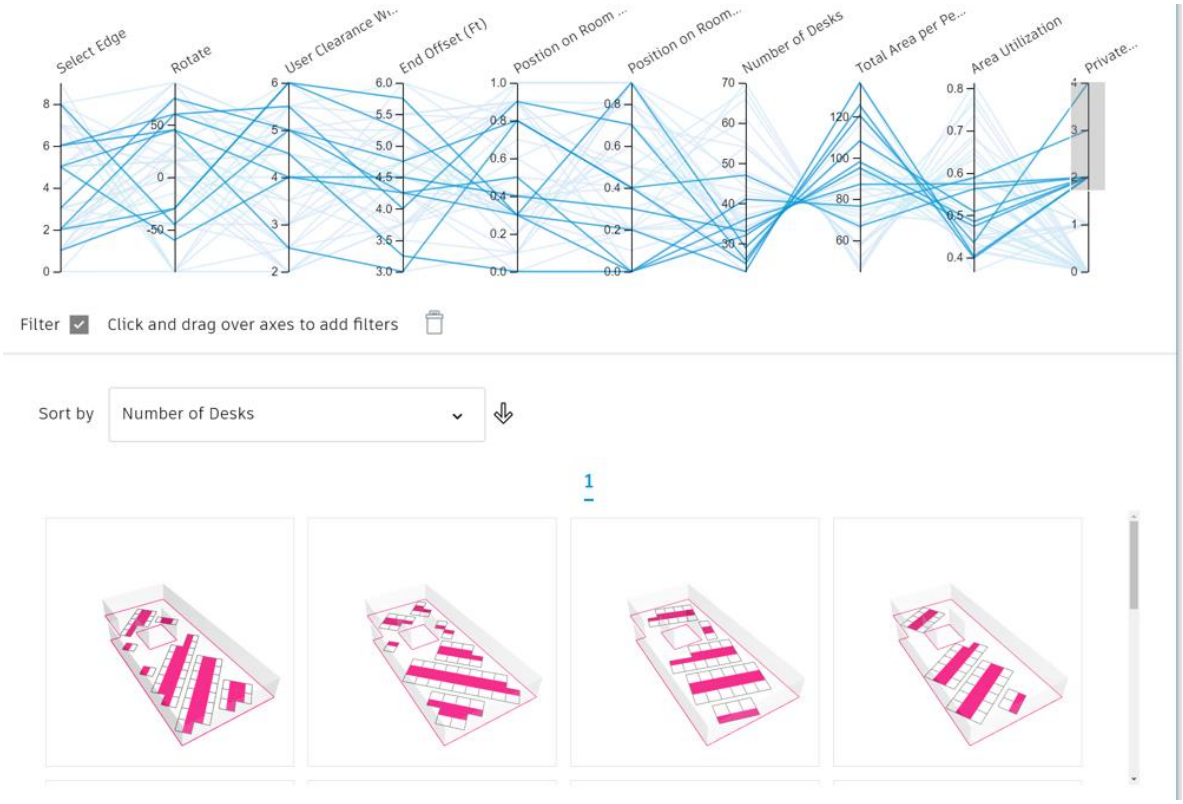
1. Measure how far each desk is from the other.
2. Count the number of desks that are less than 12' from each other
3. Count the number of desks that have less than 3 desks that are less than 12' away.



Note that both the "12' distance" and "3 desks within 12' " evaluator rules are somewhat arbitrary. I noticed I could hear my neighbours if they were at a desk less than 12' away, but I was oblivious to whoever was at a desk more than 12' away. But….privacy at work is relative. Is there white noise in your office? What kind of work are you doing? Are you workers doing loud things or talking at their desks? Someone who only cares about what shows on screen might be fine sitting within 12' of 3 or more people, but someone who needs audio privacy might not be ok with being within 12' of even 1 person. For this study, we have defined "Private Desks" as described above. Let's see how we can use it to guide design decisions.

Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

The first step is to do a run using the "Randomize" method. After generating 40 designs, we can use the Parallel Coordinates chart view to filter for alternatives that have the most number of "private desks" and then rank that result by alternatives that have the most desks. Design alternatives that rotate the desks seem to give us some interesting options along the lines of what we're looking for.

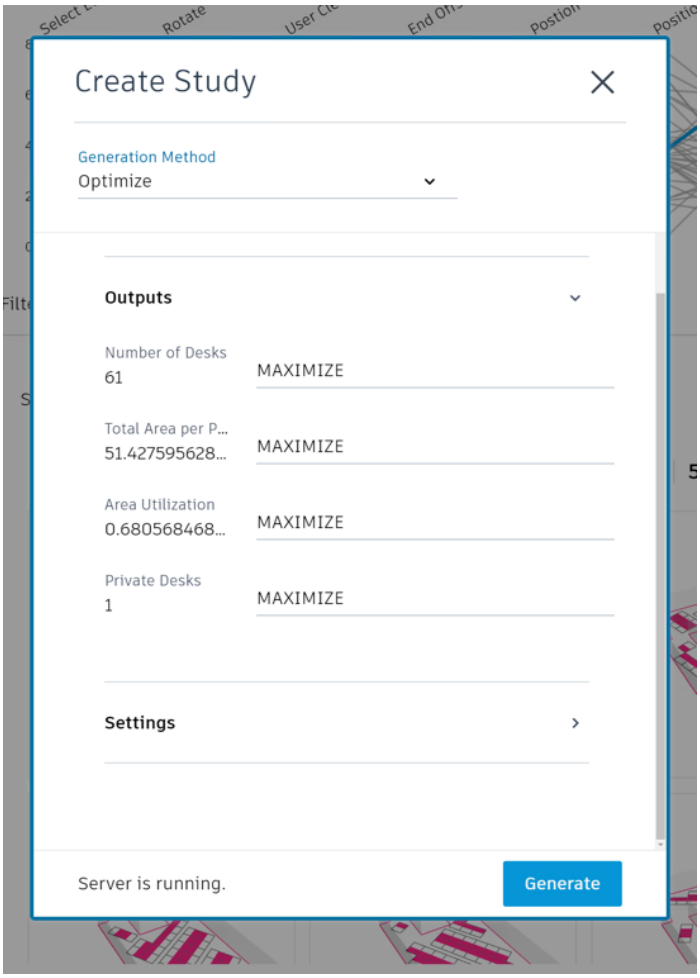


Number of Desks↓	Total Area per Person	Area Utilization	Private Desks
47.0	66.746	0.562	2.0
41.0	76.514	0.588	3.0
36.0	87.141	0.574	2.0
33.0	95.063	0.473	2.0

40 Random runs, filtered by most Private Desks and sorted by most Desks

Let's see if an optimization run will get us even closer to our goals. Using Refinery, create a new Study using the Optimize generation method and the following output settings. The most private desks we get are 3 with 41 total desks.

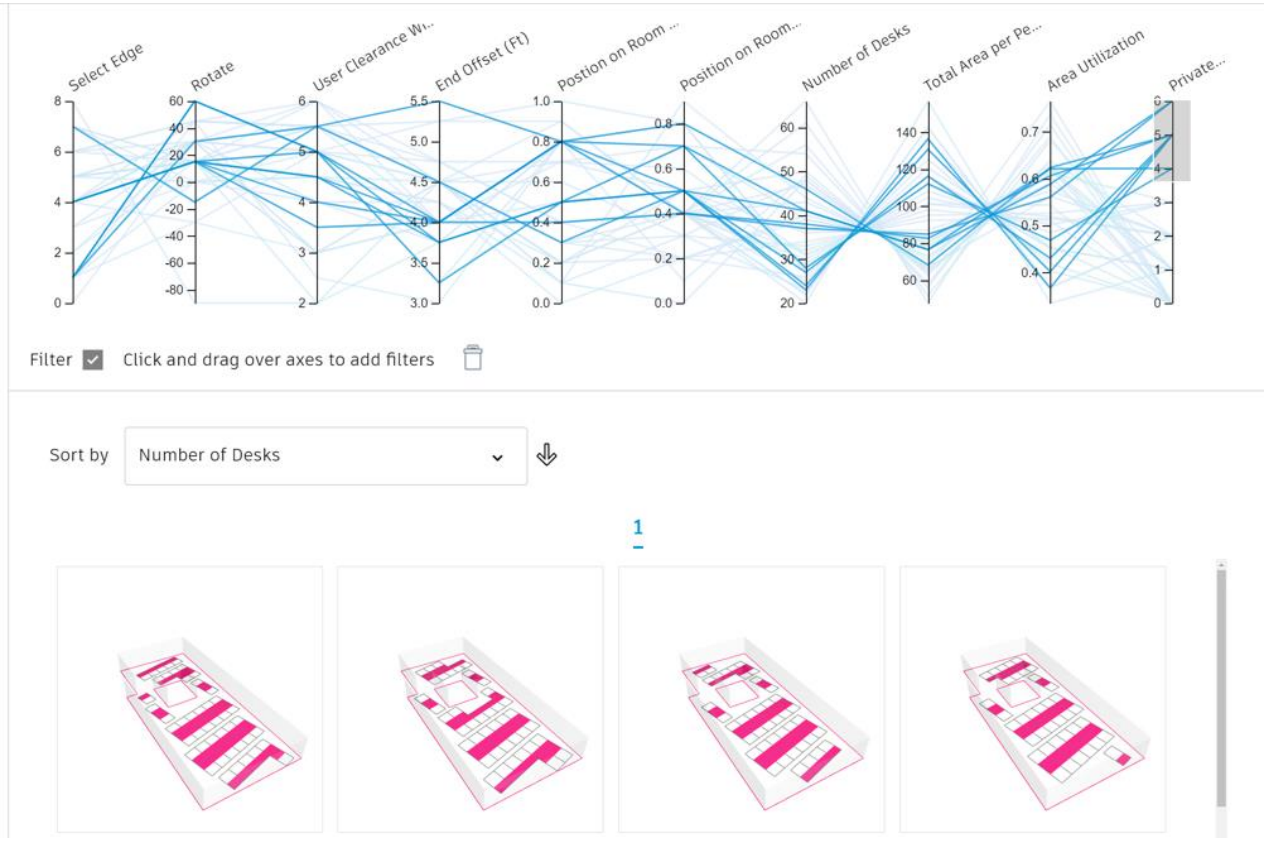
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Optimization settings in Refinery

The optimization run does give us some even better results. Now we see options with 4, 5, and 6 private desks and total desk counts of 41, 46, and 41 respectively. We can continue to sort, filter and rank the results to evaluate the different designs quickly. If the underlying room geometry changes, we can simply run the study again.

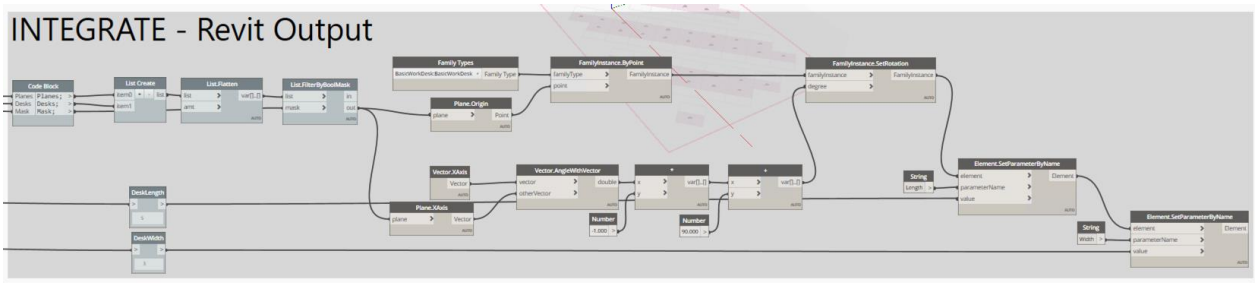
Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)



Number of Desks↓	Total Area per Person	Area Utilization	Private Desks
46.0	68.197	0.623	5.0
41.0	76.514	0.621	4.0
41.0	76.514	0.588	6.0
38.0	82.555	0.606	5.0

10x20 Optimization run filtered by most Private Desks and sorted by most desks

4. Integrators. The final step is to take the chosen design and return it to Revit for further development. The Integrate group takes desk location and rotation and place Revit families. It also sets Revit parameters for desk width and length.



Matt Jezyk, Tesla (formerly Autodesk) Tesla (formerly Autodesk)

Conclusion

We hope that these examples will give you some ideas about how you can start to use generative design workflows with Dynamo and Revit. Keep in mind that there are many different scales of workflows where you can use generative techniques from very complex examples like the Mars Toronto Autodesk offices (see https://www.keanw.com/2019/06/project-rediscover-is-now-available-for-download.html for more details and the Dynamo graph) to very simple examples like 3Box which can be extended and adapted for specific workflows – see https://www.youtube.com/watch?v=uqFgg2Dorec for example. We encourage you to share your work with #ProjectRefinery, ask question on the Dynamo and Refinery forums or contact us at refineryfeedback@autodesk.com.

Additional Resources

1. Getting Started with Dynamo: https://primer.dynamobim.org/
2. Dynamo Questions, inspiration: https://forum.dynamobim.com/
3. Design Script: https://dynamobim.org/wp-content/uploads/forum-assets/colin-mccroneautodesk-com/07/10/Dynamo_language_guide_version_1.pdf
   Or http://designscript.io/DesignScript_user_manual_0.1.pdf
   Or https://dynamobim.org/wp-content/links/DesignScriptDocumentation.pdf
   Or https://github.com/Amoursol/dynamoDesignScript
4. Refinery: https://www.autodesk.com/solutions/refinery-beta
5. Refinery Primer:
   a. https://refineryprimer.dynamobim.org/
   b. https://github.com/DynamoDS/RefineryPrimer
6. Refinery Toolkit Dynamo Package and Examples Files:
   a. https://github.com/DynamoDS/RefineryToolkits
7. Generative Design education: https://medium.com/generative-design