

320057

Multidomain, Multiobjective Optimization Using Topologic, Dynamo, and Project Refinery

Wassim Jabi
Cardiff University

Learning Objectives

- Learn how to build generative and parametric design workflows using Topologic, Dynamo, and Project Refinery
- Learn how to build topological models that can be used to conduct energy performance simulation in the early stages of design
- Learn how to build topological models that can be used to analyze the spatial characteristics of a design in the early stages of design
- Learn how to conduct multidomain, multiobjective optimization to explore a large design solution space

Description

One of the challenges facing the industry is that optimization is usually offered only within a domain-specific software engine. One solution is to use general-purpose design environments such as Dynamo, coupled with optimization engines such as Project Refinery to enable multidomain, multiobjective optimization. This class will introduce Topologic, a new and free software kit (<https://topologic.app>). Topologic offers the ability to think spatially, topologically, and conceptually about a design project. Topologic is also tightly coupled with EnergyPlus for energy analysis within Dynamo and with Project Refinery. Thus, Topologic offers the possibility of multidomain, multiobjective optimization where the involved domains go beyond traditional building performance. This class will introduce the related concepts, illustrate the powerful features of the software through example workflows, and offer you hands-on experience to experiment with customized workflows.

Speaker

Dr. Wassim Jabi earned his B.Arch. from the American University of Beirut and his M.Arch. and Ph.D. from the University of Michigan. While in the U.S., he led the Association for Computer-Aided Design in Architecture (ACADIA), secured a \$250,000 National Science Foundation (NSF) grant, and taught at various universities before moving to Europe in 2008. He is currently a Reader (tenured Associate Professor) at the Welsh School of Architecture, Cardiff University where he leads the digital design area. Dr. Jabi has written a book titled "Parametric Design for Architecture" (Laurence King Publishing, London). His current research is at the intersection of parametric design, the representation of space, building performance simulation, and robotic fabrication in architecture

Introduction

In this class we introduce an example workflow that integrates several software platforms to enable users to explore and optimize their design projects using objectives from several domains at once. One of the disadvantages of using domain-specific software for multi-objective optimization is that they exclude the consideration of objectives from other domains. It would be difficult, for example, to optimize varied objectives such as the energy consumption of a building as well as the views it affords its residents using only thermal analysis software. In contrast, architects, designers, or engineers, do not only focus on one domain such as thermal comfort or structural integrity or sustainability. Instead, they look at the problem more holistically and try to optimize possibly conflicting objectives from several domains or areas.

The power of visual data flow programming environments, such as Dynamo, is that they allow plug-ins and other software libraries to extend its functionality. In this example workflow, we will illustrate this capability to integrate Dynamo with our own software, Topologic, OpenStudio and EnergyPlus and use Project Refinery for multidomain, multiobjective optimization.

Topologic is the result of a three-year research grant supported by the Leverhulme Trust and carried out by Cardiff University's Welsh School of Architecture in collaboration with UCL. Topologic provides an extensive modelling library for hierarchical and topological representations of space and other elements.

OpenStudio is “a cross-platform (Windows, Mac, and Linux) collection of software tools to support whole building energy modeling using EnergyPlus and advanced daylight analysis using Radiance. OpenStudio is an open source (LGPL) project to facilitate community development, extension, and private sector adoption. OpenStudio includes graphical interfaces along with a Software Development Kit (SDK).

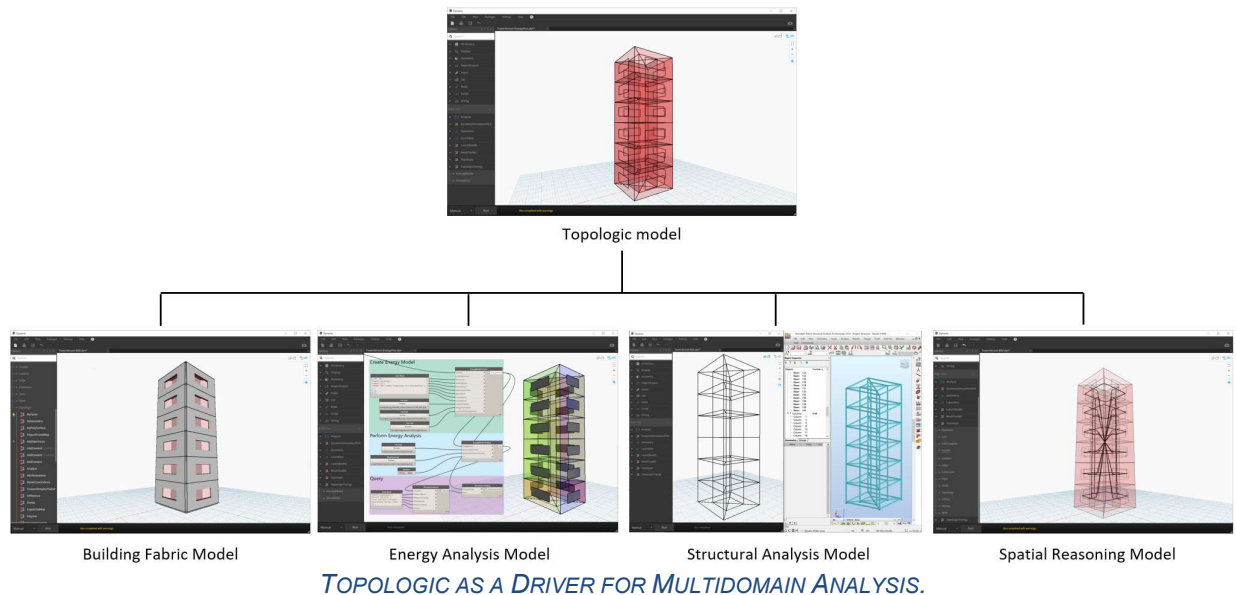
Project Refinery is “an Autodesk generative design beta for the architecture, engineering and construction industry that gives users the power to quickly explore and optimize their Dynamo designs.”

Since Topologic is the newest and least known of the used software, it is important to briefly introduce its abilities. The accompanying class presentation covers its features in more detail while this document will offer a detailed tutorial on an example workflow for exploring and optimizing an example building for shortest paths and energy consumption. More detailed information and the software itself can be found at <http://topologic.app>.

Topologic

Topologic is a software modelling library enabling hierarchical and topological representations of architectural spaces, buildings and artefacts through non-manifold topology (NMT). Topologic is designed as a core library and additional plugins to visual data flow programming (VDFP) applications and parametric modelling platforms commonly used in architectural design practice such as Dynamo and Grasshopper. These applications provide workspaces with visual programming nodes and connections for architects to interact with Topologic and perform architectural design and analysis tasks.

Topologic is well-suited to create a lightweight representation of a building as an external envelope and the subdivision of the enclosed space into separate spaces and zones using zero-thickness internal surfaces. Because Topologic maintains topological consistency, a user can query these cellular spaces and surfaces regarding their topological data and thus conduct various analyses. For example, this lightweight and consistent representation was found to be well-matched with the input data requirements for energy analysis simulation software. Because Topologic allows entities with mixed dimensionalities and those that are optionally independent (e.g. a line, a surface, a volume) to co-exist, structural models can be represented in a coherent manner where lines can represent columns and beams, surfaces can represent walls and slabs, and volumes can represent solids. In addition, non-building entities, such as structural loads can be efficiently attached to the structure. This creates a lightweight model that is well-matched with the input data requirements for structural analysis simulation software.

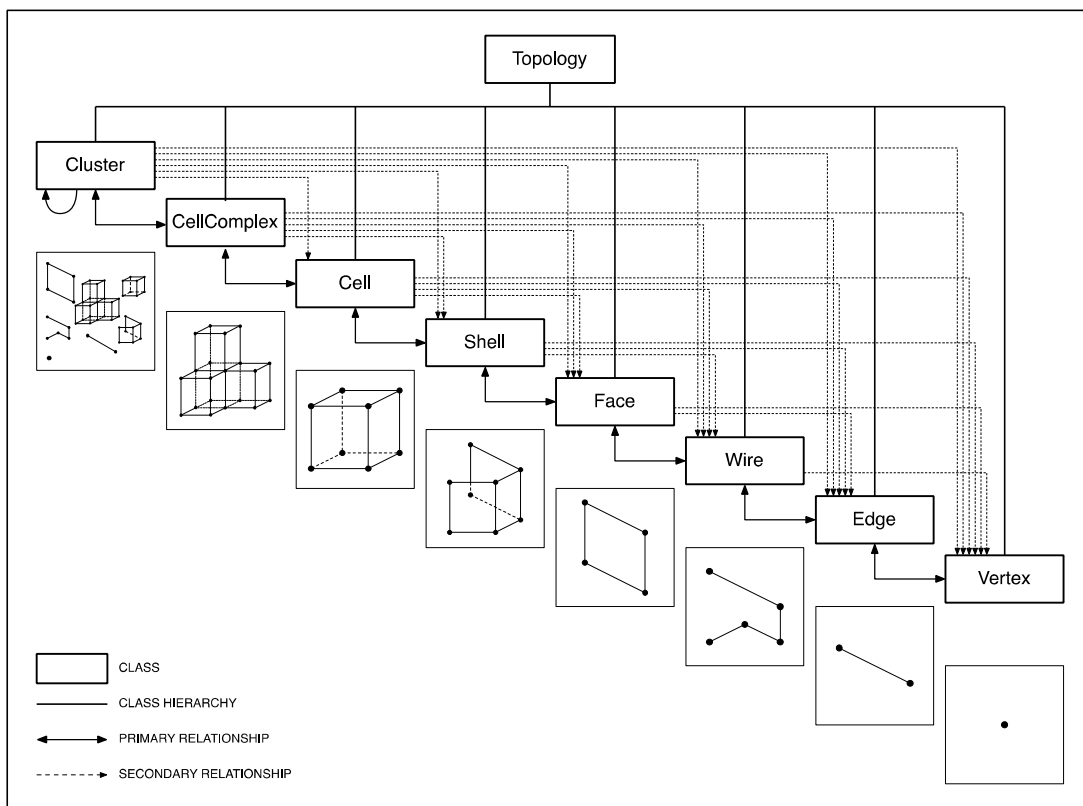


Class Hierarchy

Topologic can operate on and represent the following object categories:

- **Topology:** A Topology is an abstract superclass that stores constructors, properties and methods used by other subclasses that extend it.
- **Vertex:** A Vertex is a zero-dimensional entity equivalent to a geometry point.
- **Edge:** An Edge is a one-dimensional entity defined by two vertices. It is important to note that while a topologic edge is made of two vertices, its geometry can be a curve with multiple control vertices.
- **Wire:** A Wire is a contiguous collection of Edges where adjacent Edges are connected by shared Vertices. It may be open or closed and may be manifold or non-manifold.
- **Face:** A Face is a two-dimensional region defined by a collection of closed Wires. The geometry of a face can be flat or undulating.
- **Shell:** A Shell is a contiguous collection of Faces, where adjacent Faces are connected by shared Edges. It may be open or closed and may be manifold or non-manifold.

- **Cell:** A Cell is a three-dimensional region defined by a collection of closed Shells. It may be manifold or non- manifold.
- **CellComplex:** A CellComplex is a contiguous collection of Cells where adjacent Cells are connected by shared Faces. It is non- manifold.
- **Cluster:** A Cluster is a collection of any topologic entities. It may be contiguous or not and may be manifold or non- manifold. Clusters can be nested within other Clusters.
- **Aperture:** An aperture is an abstract class that is typically thought of as an opening in a face (e.g. window or door). However, in Topologic, an aperture can have any topology and placed on any other topology. For example, an aperture can be an Edge placed within another Edge to represent the width of a door or window. More commonly, however, an Aperture is a Face placed within another Face.
- **Content/Context:** Like Apertures, any Topology can hold within it any number of content Topologies. The Topology will then form the Context of these Content Topologies.
- **Graph:** A Graph is made of Vertices and Edges and implements many methods found in graph theory. A Graph can be derived from most Topologies and connect their centers of mass and optionally connect via shared Topologies, shared Apertures, and to external Topologies (e.g. outside Faces). A Graph can be asked for its topology in the form of a Cluster of Edges and Vertices, to return the shortest path from one Vertex to another, as well as provide extensive analytical information on its Vertices, Edges, and overall connectivity and structure.



TOPOLOGIC CLASS HIERARCHY.

Tutorial

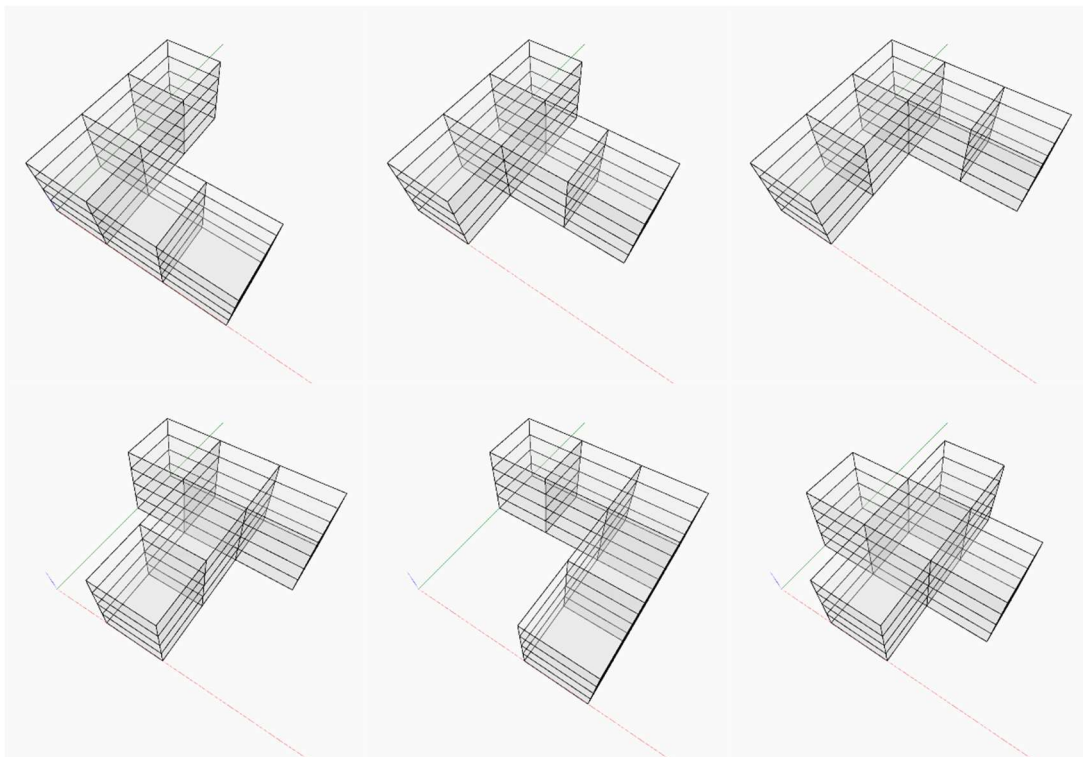
To follow this tutorial, you first need to install several pieces of software on a computer running Windows 10. For this tutorial, we used the following software and versions:

- Dynamo Sandbox version 2.3 <http://dynamobim.org>
- Topologic version 0.8.7 <https://topologic.app/software>
- Project Refinery Beta version 0.35.0 <https://www.autodesk.com/solutions/refinery-beta>
- OpenStudio version 2.8.0 <https://github.com/NREL/OpenStudio/releases/tag/v2.8.0>
- Additionally, example EPW and DDY weather files that are needed for the energy analysis are available as additional downloads or you can choose your own from the US Department of Energy website at: <https://energyplus.net/weather>
- The Dynamo definition (DYN file) is included with the downloadable resources for this file, but is also available from <https://topologic.app/learning>

This tutorial assumes moderate familiarity with Dynamo. *Save often!*

Getting Started

In this tutorial, we will explore various configurations of a sample building made of two wings that are perpendicular to each other. These wings can slide against each other in the X and Y directions thus forming either an L-shape, a T-shape, or a Cross-shape in various orientations.



BUILDING CONFIGURATIONS.

For the purpose of this tutorial, we will assume that you wish to explore two objectives that come from two different domains:

- The first objective is to minimize the average distance of the shortest path from various points in the building to a single exit point.
- The second objective is to minimize the average cooling load per square meter.

Step 1

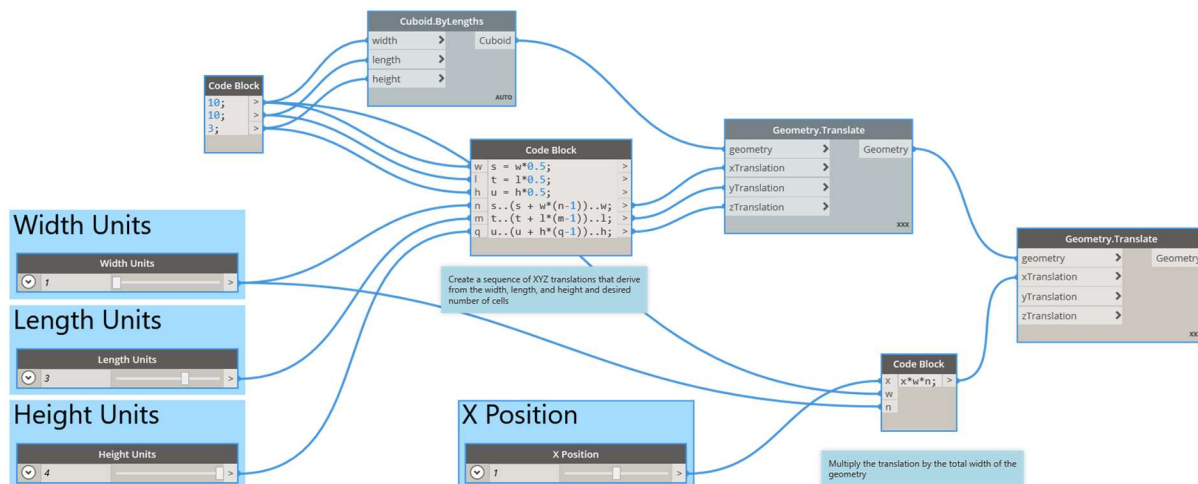
Let's get started. Open Dynamo and start a new Home Workspace. Drag and drop the connected nodes as you see below to create a three-dimensional array of cuboids and exercise the X position slider to test that the cuboids move together along the X axis. Notice that the **Geometry.Translate** node uses the **Cross Product** lacing. The code blocks contain the following lines:

```
s = w*0.5;
t = l*0.5;
u = h*0.5;
s..(s + w*(n-1))..w;
t..(t + l*(m-1))..l;
u..(u + h*(q-1))..h;
```

and

```
x*w*n;
```

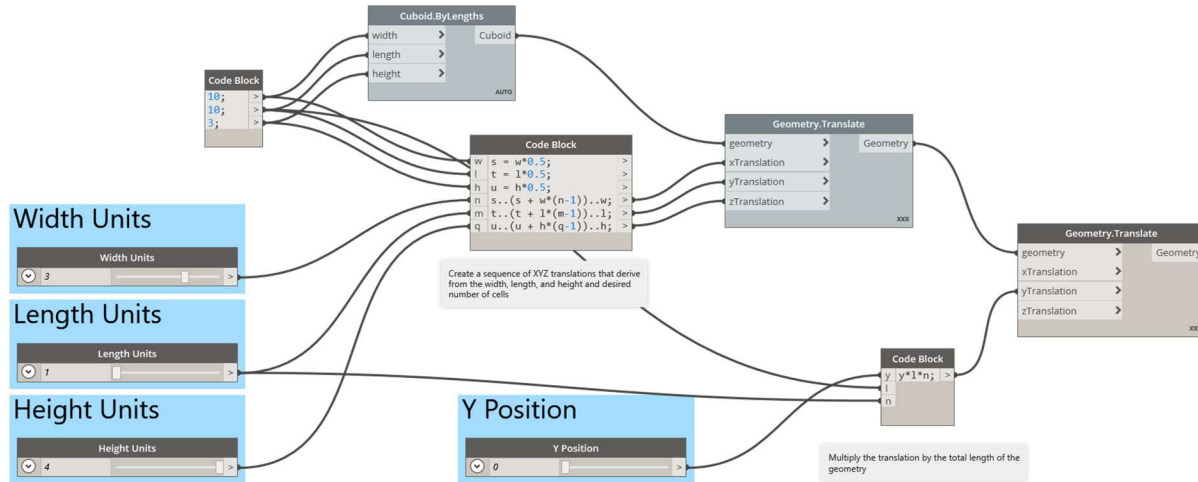
Set the **Width Units** slider and the **Length Units** slider to be integers that can vary between 1 and 4 in steps of 1. Set the **Height Units** slider to be an integer that can vary between 1 and 4 in steps of 1. Set the **X Position** slider to be a number that can vary from 0 to 2 in steps of 0.5.



In preparation for a Refinery study, right-click on the X Position node and select "Is Input"

Step 2

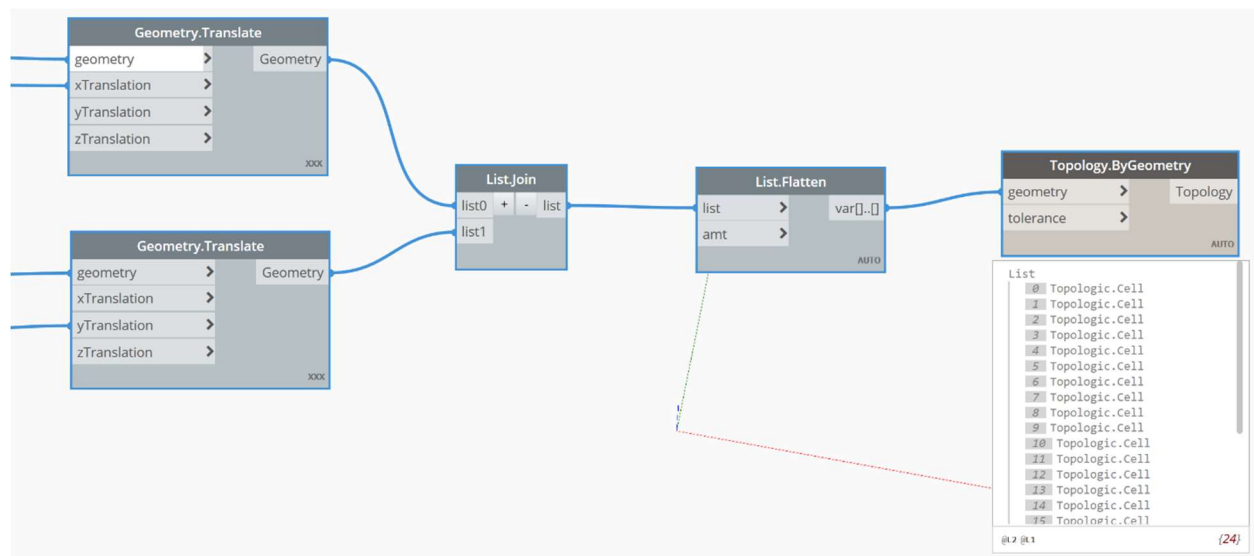
Copy and paste all the nodes from Step 1 and rename and rewire as you see below to create the second wing of the building. Notice that the **Width Units** and **Length Units** are reversed. The new wing needs to be perpendicular to the first wing and move along the Y axis.



In preparation for a Refinery study, right-click the Y Position node and select "Is Input"

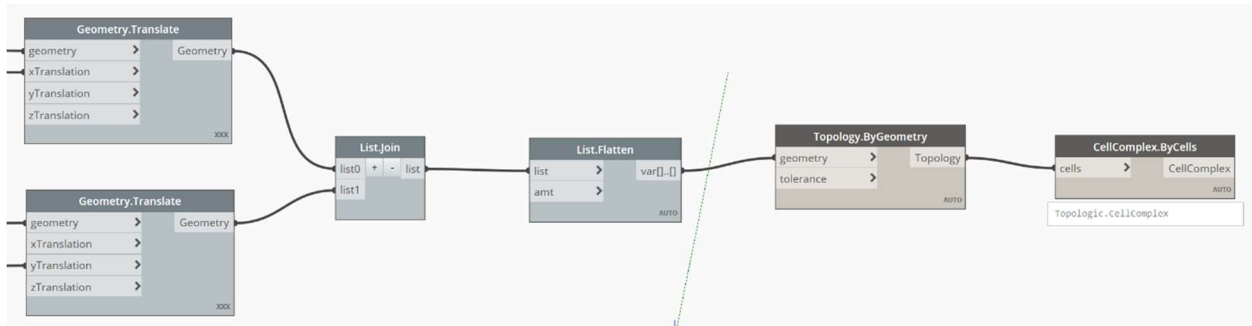
Step 3

Let's convert these cuboids to Topologic Cells. Use **List.Join**, **List.Flatten** and then **Topology.ByGeometry** to convert them into a list of Cells.



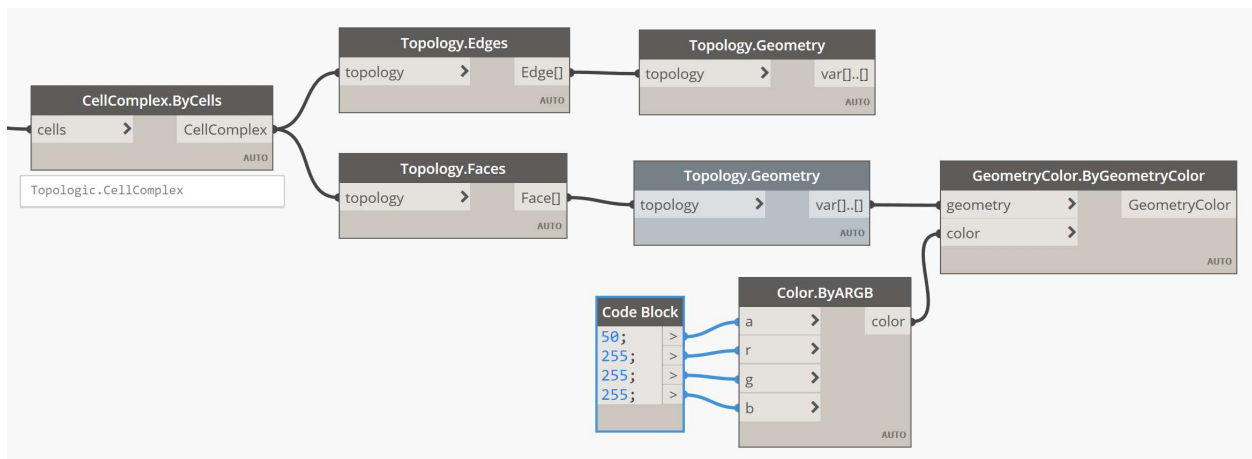
Step 4

Create a CellComplex from the Cells using **CellComplex.ByCells**.

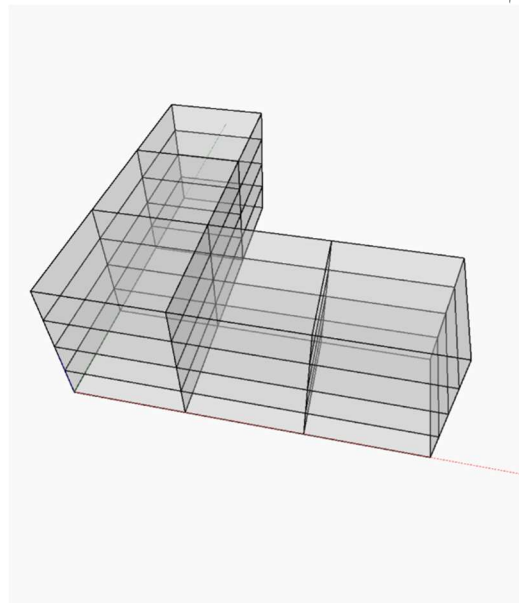


Step 5

Let us visualize the CellComplex. Hide all the geometries so that none show on the workspace by un-selecting **Preview** on each node that draws to the workspace. Connect the output of **CellComplex.ByCells** node to the **topology** input parameter of **Topology.Edges**. Then connect its output to **Topology.Geometry**. You should see a wireframe of the CellComplex. Connect the **CellComplex** output to **topology** input parameter of **Topology.Faces** and then to the **Topology.Geometry** node. Immediately un-select **Preview** on the **Topology.Geometry** node and connect it to **Geometry.ByGeometryColor**. Create a **Color.ByARGB** for a translucent color as you see below.

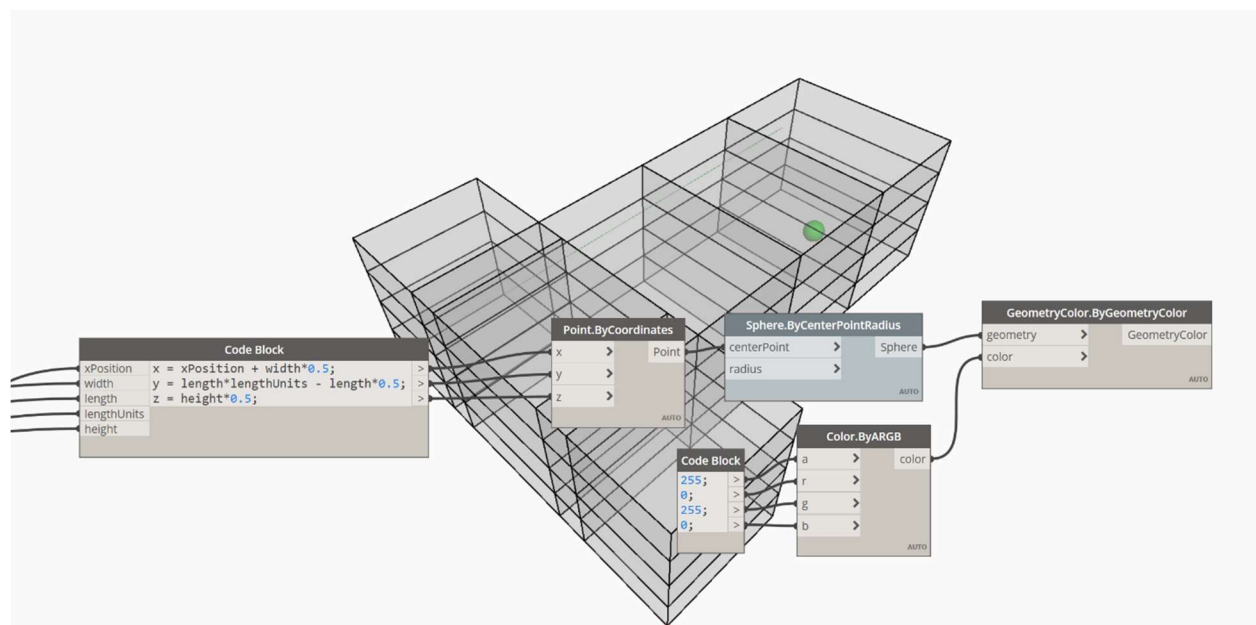


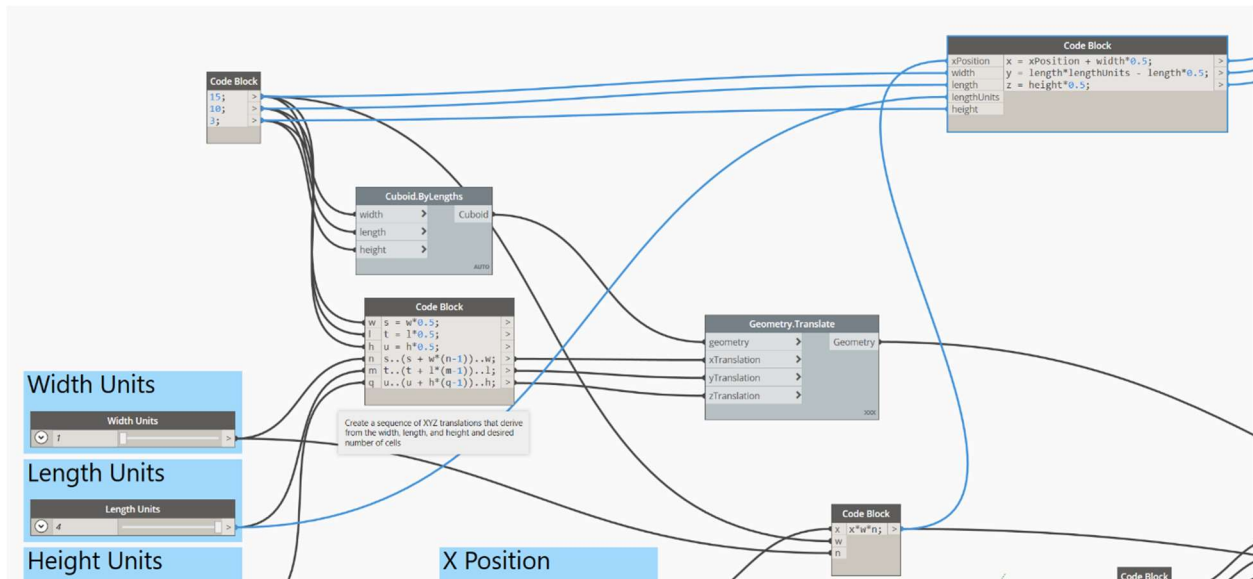
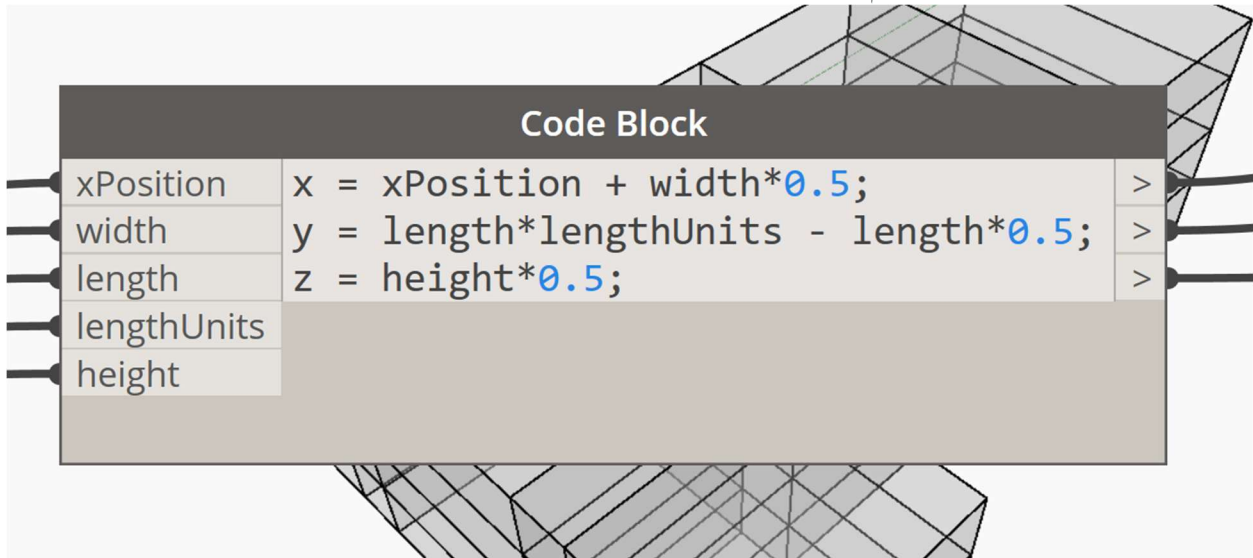
You should now see a CellComplex similar to below



Step 6

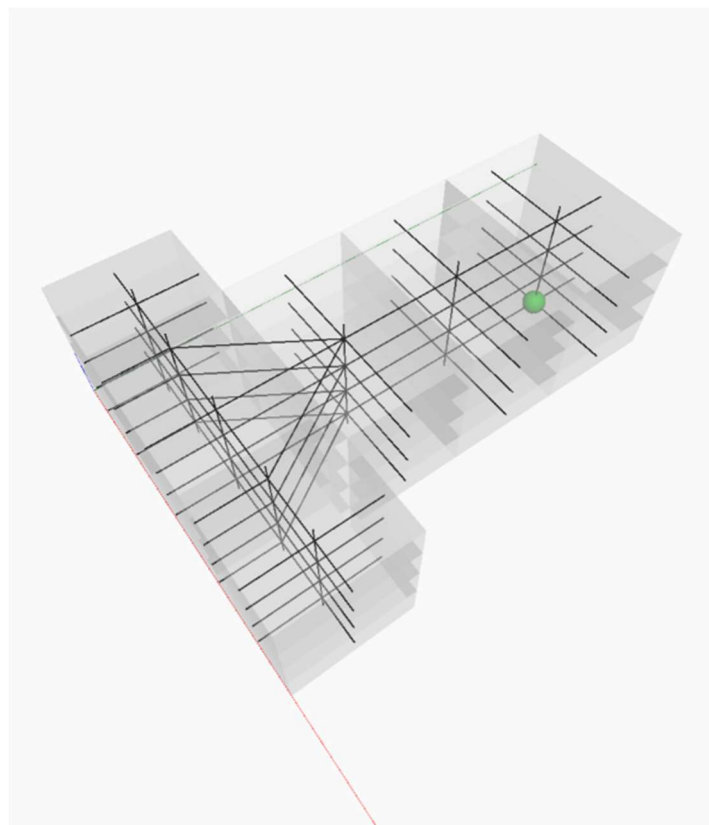
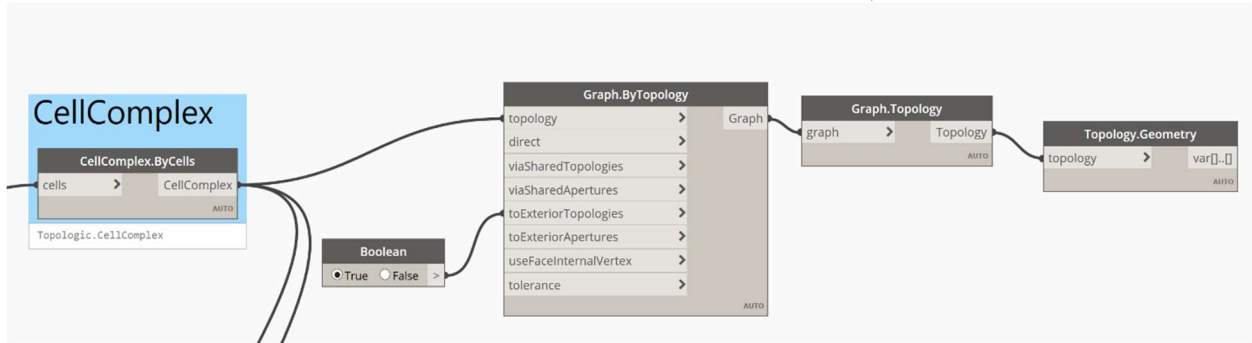
In this step we will choose a destination point that serves as the exit to the building. Let's start by choosing a point and marking it with a green sphere. We will choose the centroid of the bottom Cell in the first wing that is furthest on the Y axis. The images below show you what nodes need to be connected and the content of the Code block. When done, test that the green sphere moves when you move the **X Position** so it remains at the centroid of the furthest cell at the lowest level. Remember the x, y, z Code block as we will use these numbers for the end Vertex in the next step.





Step 7

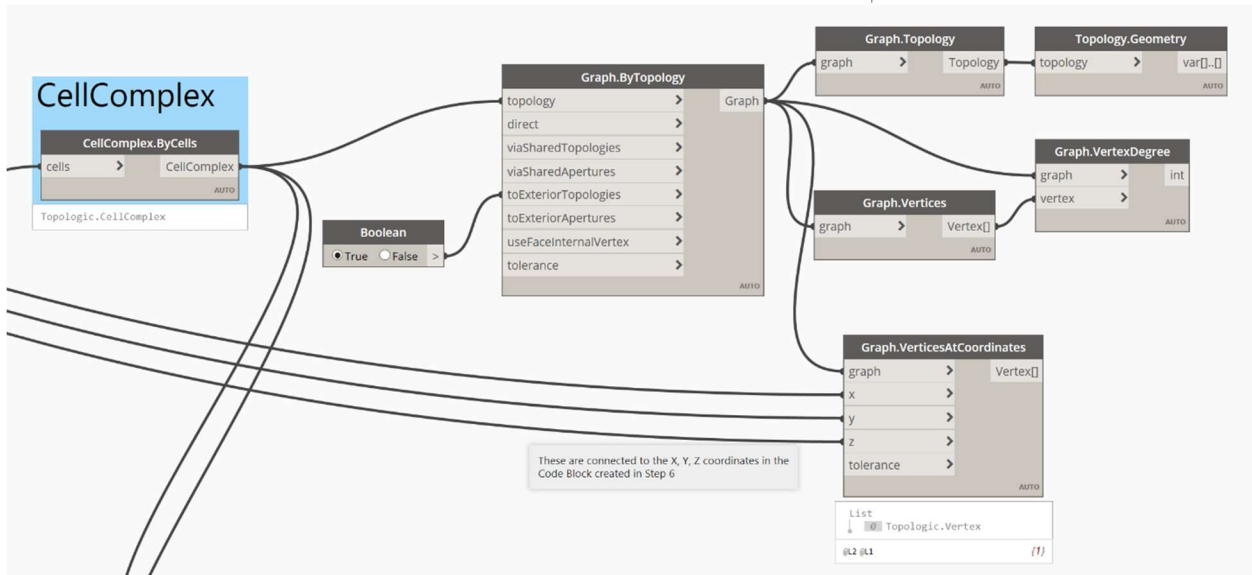
It is now time to create the dual Graph of this CellComplex. A Graph is a collection of edges and vertices. We have several options on how to create the dual Graph. For this example, we will create a Graph that connects the centroids of the Cells to each other as well as to the outside Faces. We will use this Graph to compute the average shortest path from each Vertex on an outside Face to a single Vertex of our choosing. Drag and drop a **Graph.ByTopology**. Drag and drop a **Boolean** node and set it to **True**. Connect it to **toExteriorTopologies**. Connect the output **Graph** to **Graph.Topology** and that to **Topology.Geometry** to visualize it.



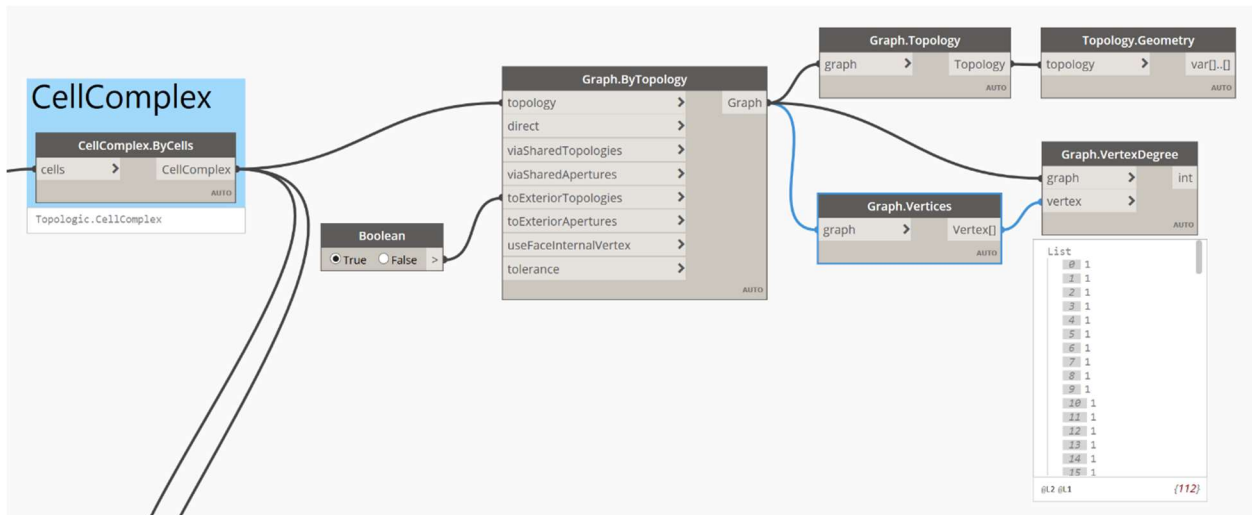
THE DUAL GRAPH OF THE CELL COMPLEX.

Step 8

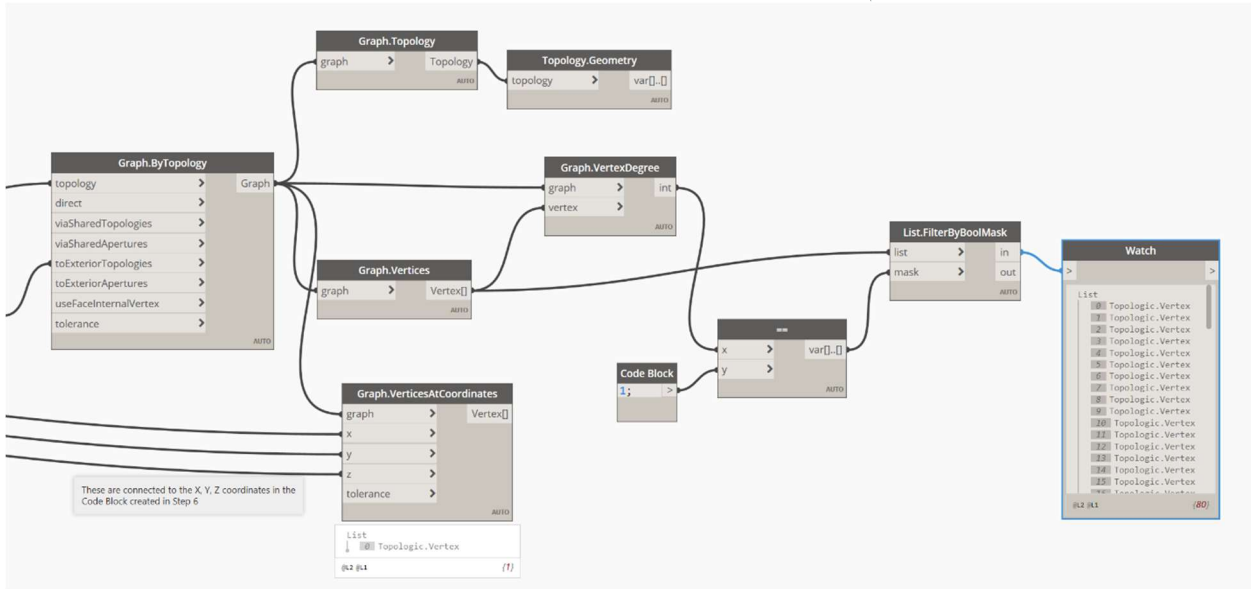
Next, we need to find the Graph Vertex that matches the X, Y, Z coordinates of the Green sphere. For that, we will use the **Graph.VerticesAtCoordinates** node. The **X, Y, Z** inputs should be connected to the X, Y, Z variables in the Code Block created in Step 6.



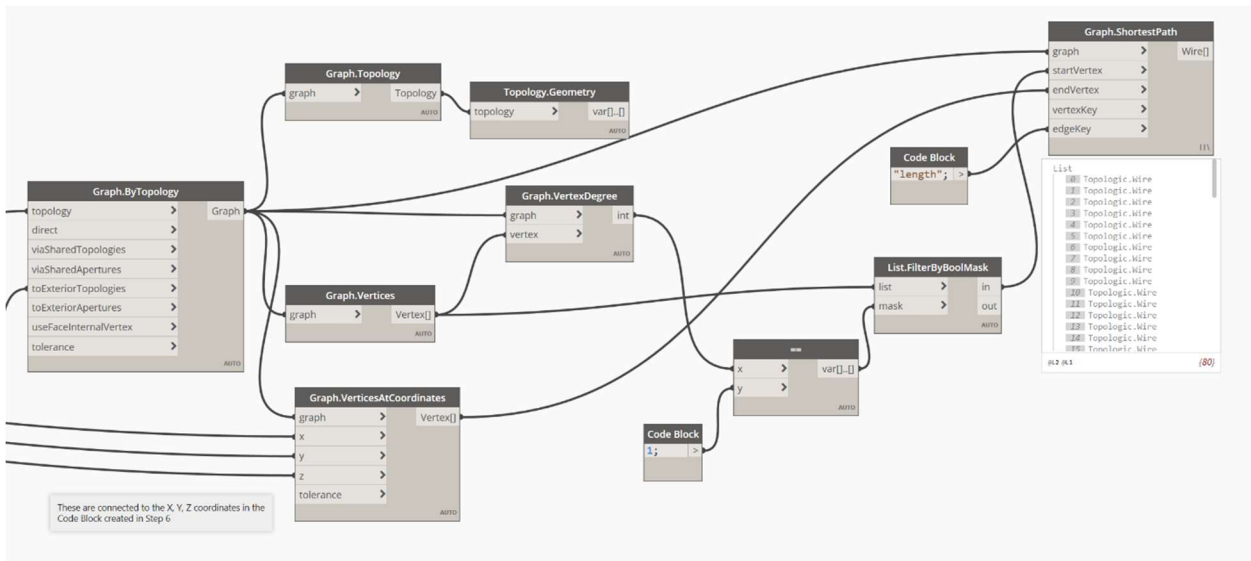
Next, get all the Vertices of the Graph using **Graph.Vertices**. Then let's find out the degree of each Vertex. The degree of a Vertex is simply the number of edges connected to it. The Vertices that we are interested in are the ones with a degree of 1. These are the exterior vertices as you can see in the dual graph figure above.



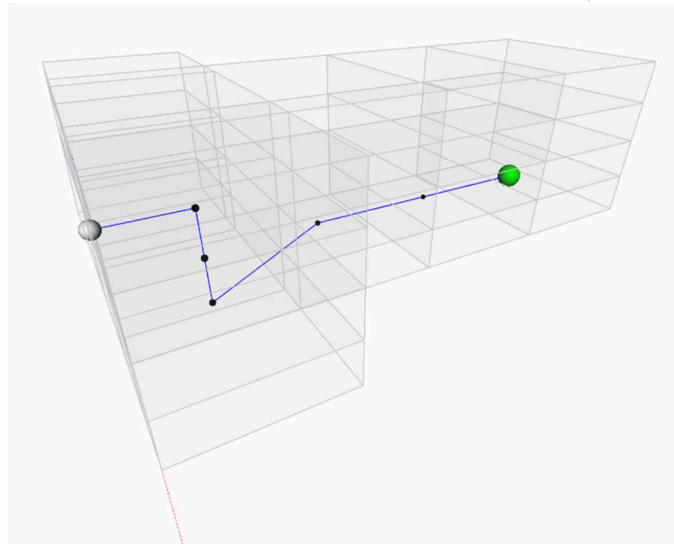
We will use the **==** node and a **List.FilterByBooleanMask** node to filter the ones with a degree of 1.



We now can find the shortest path from each of the Vertices with a degree of 1 to the exit Vertex (the green sphere) by using **Graph.ShortestPath**. Create a **Code block** with the string “length”; and connect it to **edgeKey**. **Make sure you set the Lacing to “Longest” on the Graph.ShortestPath node.**

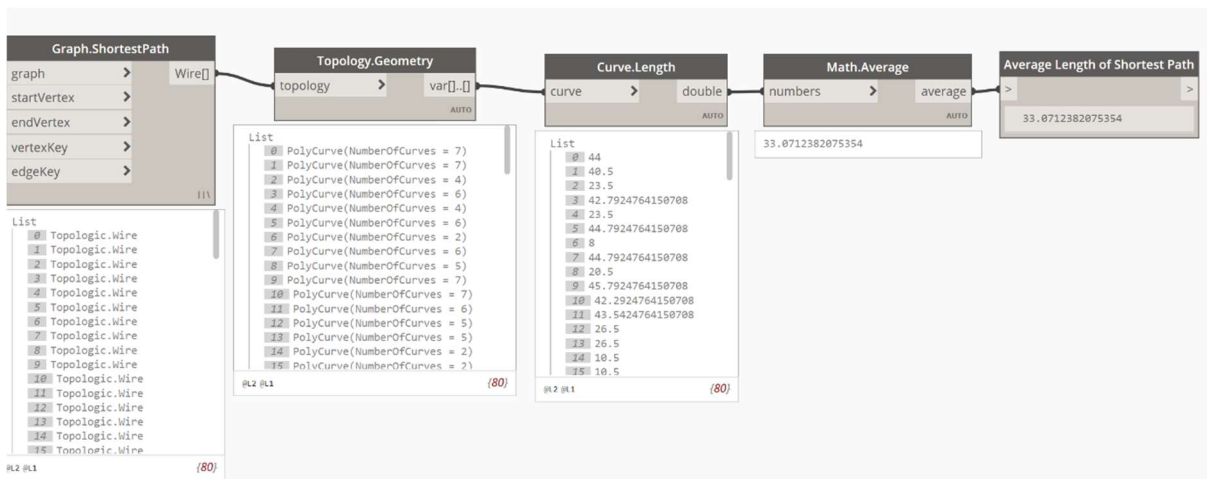


If you wish to visualize a single path, then hide the overall Graph and select one vertex from the list of **Graph.Vertices** and then connect the **Graph.ShortestPath** to **Topology.Geometry**.



Step 9

In this step we will compute the average length of each path in **Graph.ShortestPath**. Connect that node to **Topology.Geometry**. It will result in a list of Dynamo Polycurves. Compute the length of each Polycurve using **Curve.Length** and then take the average of these lengths using **Math.Average**. Connect that average to a **Watch** node and rename it to “Average Length of Shortest Path”



In preparation for a Refinery study, right-click on the Watch node and select “Is Output”

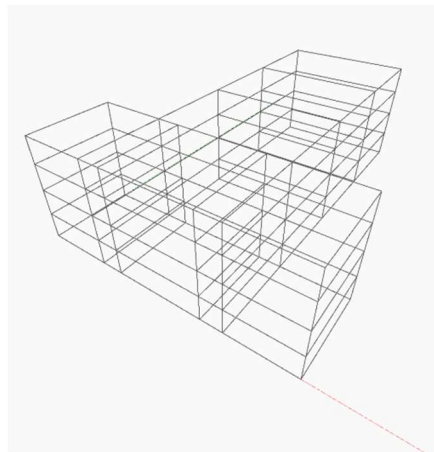
Congratulations on making it this far. This represents the first objective: to explore configurations that produce the shortest average path from several locations in the building to one exit location. Next, we will create a workflow that tries to achieve a very different objective: to explore configurations that create the lowest average cooling load per area in the building.

Energy Analysis

In this section, we will need to use TopologicEnergy nodes. Make sure you have installed OpenStudio v 2.8.0 from the website listed at the start of this document. In addition, create a special folder that will store the TopologicEnergy outputs. We will assume that this folder is called "Energy Analysis Output". Finally, make sure you have the .EPW file and the .DDY file at a known location. These are included with this document.

Another thing to note is that we will assume that the two wings of the building use the same floor to ceiling height and have the same total height.

Hide all displayed geometry except for the wireframe display of the CellComplex. Your building should appear like the image below.



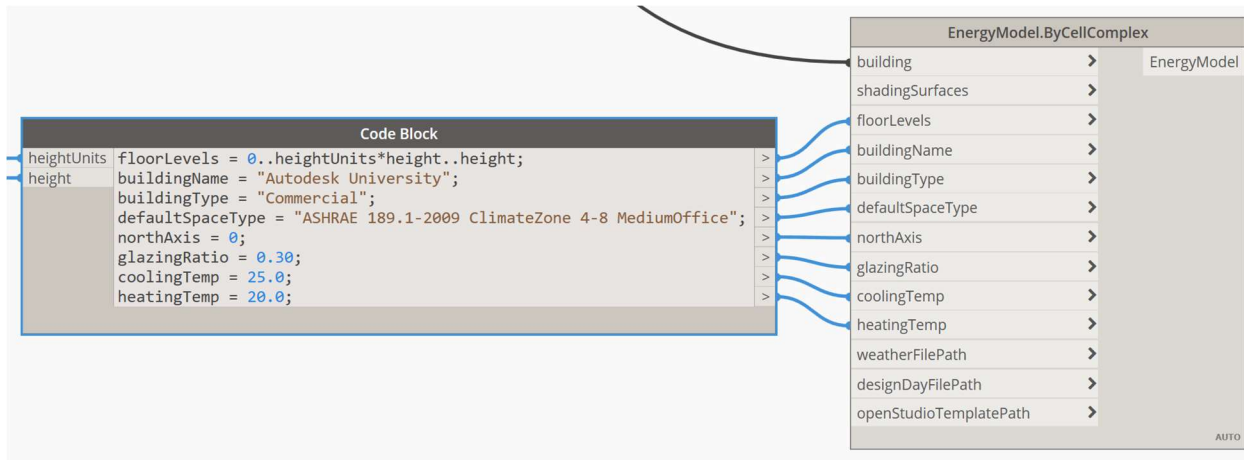
Step 10

Drag and drop an **EnergyModel.ByCellComplex** node. Connect the **CellComplex** to its **building** input parameter. Create a Code block and type the following lines into it. You may find it convenient to copy and paste from this document into the Code block. You need to connect the **heightUnits** (i.e. number of floors) and the **height** variables (i.e. floor to ceiling height of one floor) into this code block. For example, if the **heightUnits** are 4 and the **height** of one unit is 3, then the correct resulting floor levels should be [0,3,6,9,12]. The floor levels include the ground level (at 0) and the roof of the building (at 12). Note that we chose the north axis to be in the positive Y direction (i.e. 0 degrees clockwise from North) and that we set the glazing ratio to 30% (i.e. 0.3).

```

floorLevels = 0..heightUnits*height..height;
buildingName = "Autodesk University";
buildingType = "Commercial";
defaultSpaceType = "ASHRAE 189.1-2009 ClimateZone 4-8 MediumOffice";
northAxis = 0;
glazingRatio = 0.30;
coolingTemp = 25.0;
heatingTemp = 20.0;

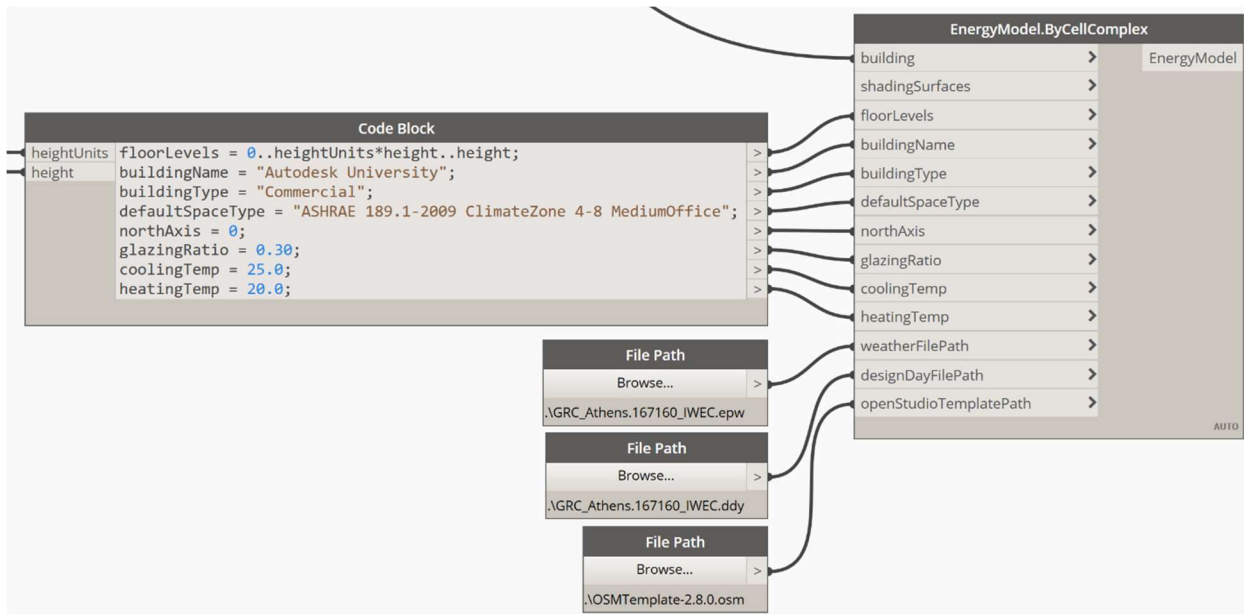
```



Step 11

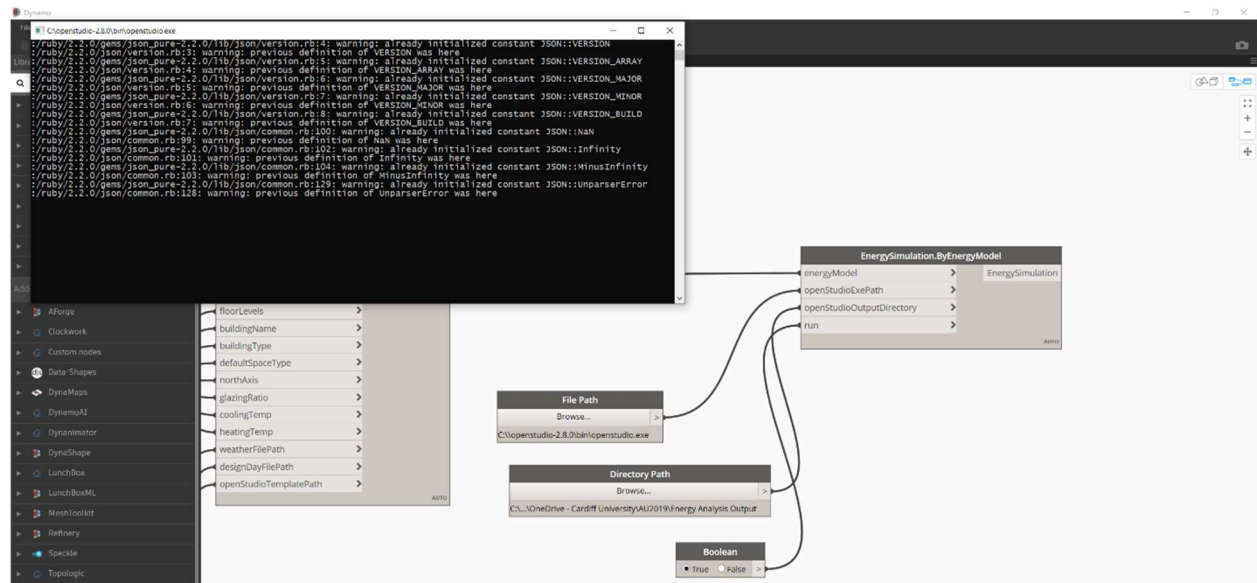
Create three File Path nodes, click on Browse on each and select the Weather EPW file, the design day (DDY) file, and the OpenStudio Template file (OSM) respectively. In our example, the file names are:

- Weather File: GRC_Athens.167160_IWEC.epw
- Design Day File: GRC_Athens.167160_IWEC.ddy
- OpenStudio Template: OSMTemplate-2.8.0.osm



Step 12

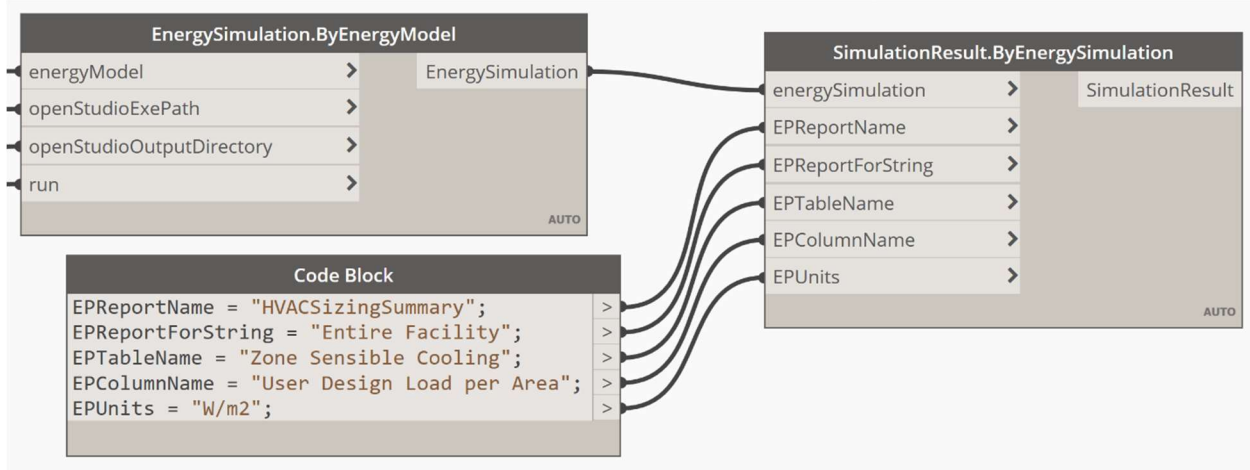
Drag and drop an **EnergySimulation.ByEnergyModel** node. Connect the **EnergyModel** output from the previous node to its **energyModel** input. Drag and drop a **File Path** node, Browse, and select the openstudio.exe file located at: C:\\openstudio-2.8.0\\bin\\openstudio.exe. Make sure you have created a folder for the energy analysis folder. Drag and drop a **Directory Path** node and point it to that folder. Lastly, create a **Boolean** node, set it to **True** and connect it to the **run** input parameter to run the analysis. If all is correct, you will see a black window appear with some warnings. This is usual and is of no concern. It means that OpenStudio has been launched and is analyzing the building.



Step 13

Now that the energy simulation is complete, we can quickly read back in the results of the analysis. Drag and drop a **SimulationResult.ByEnergySimulation** node. Connect the output of the previous node to its **energySimulation** input parameter. Create a CodeModel block and type into it the following lines to retrieve the cooling load per area:

```
EPReportName = "HVACsizingSummary";
EPReportForString = "Entire Facility";
EPTableName = "Zone Sensible Cooling";
EPColumnName = "User Design Load per Area";
EPUnits = "W/m2";
```

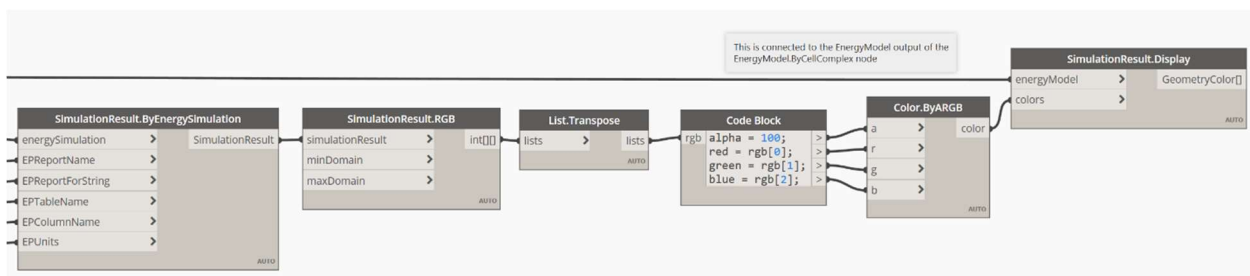


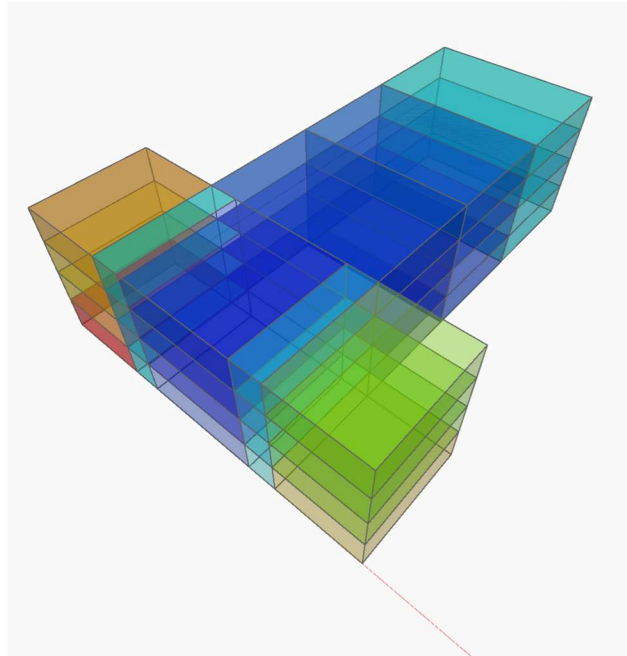
Step 14

The time has come to display the results as colors mapped unto the Cells of the CellComplex. Drag and drop a **SimulationResult.RGB** and connect the output of the previous node to its **simulationResult** input parameter. Drag and drop a **List.transpose** node and connect the output of the previous node to its **lists** parameter. Create a Code block and type in the following lines:

```
alpha = 100;
red = rgb[0];
green = rgb[1];
blue = rgb[2];
```

Connect the output of the previous node to its **rgb** input parameter. Drag and drop a **Color.ByARGB** node and connect the outputs of the previous node to its **a, r, g, b** input parameters respectively. Drag and drop a **SimulationResult.Display** node and connect the **EnergyModel** output of the **EnergyModel.ByCellComplex** node to its **energyModel** input parameter. Connect the output of the **Color.ByARGB** to its **color** input parameter. You should now see the fully colored building where each color represents the cooling load per area of that Cell.

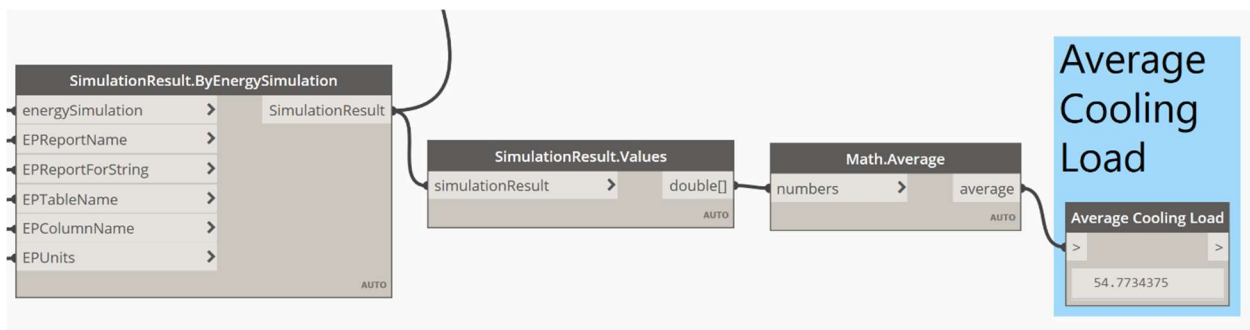




Step 15

In this step we will calculate the average cooling load per area for the whole building. To do that, we will collect the value for each space and divide by the number of spaces. Drag and drop a **SimulationResult.Values** node and connect the **SimulationResult** output of the **SimulationResult.ByEnergySimulation** node to its **simulationResult** input parameter. Drag and drop a **Math.Average** node and connect the output of the **SimulationResult.Values** to its **numbers** input parameter. Drag and drop a **Watch** node and connect it to the previous node. Rename it as “Average Cooling Load”

In preparation for a Refinery study, right-click on the Average Cooling Load node and select “Is Output”



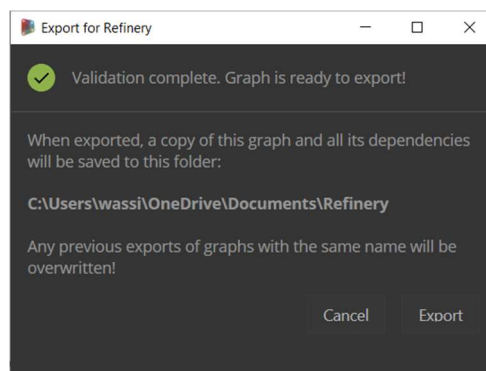
Congratulations! You have just completed the computation of the second Refinery analysis objective. In the next section, we will explore the effect of positioning the two building wings on the average shortest path and the average cooling load. Two objectives that derive from two very different domains of exploration: Spatial and Thermal.

Multidomain, Multiobjective Optimization Using Project Refinery

Now that we have a Dynamo definition with two inputs (X Position, and Y position) and two outputs (Average Shortest Path and Average Cooling Load), we can export the definition to Refinery for analysis.

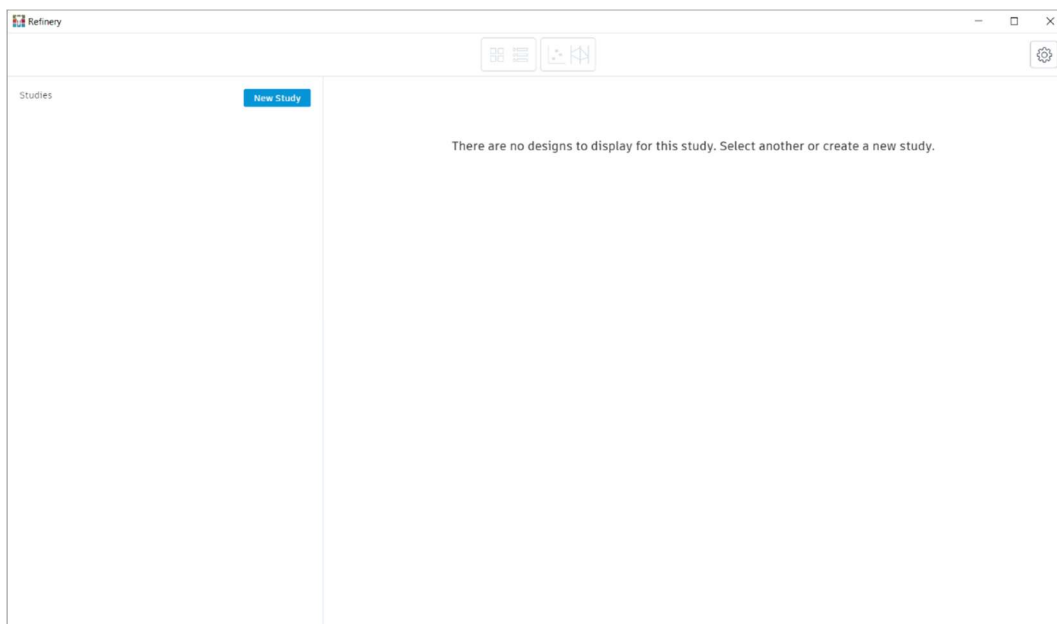
Step 16

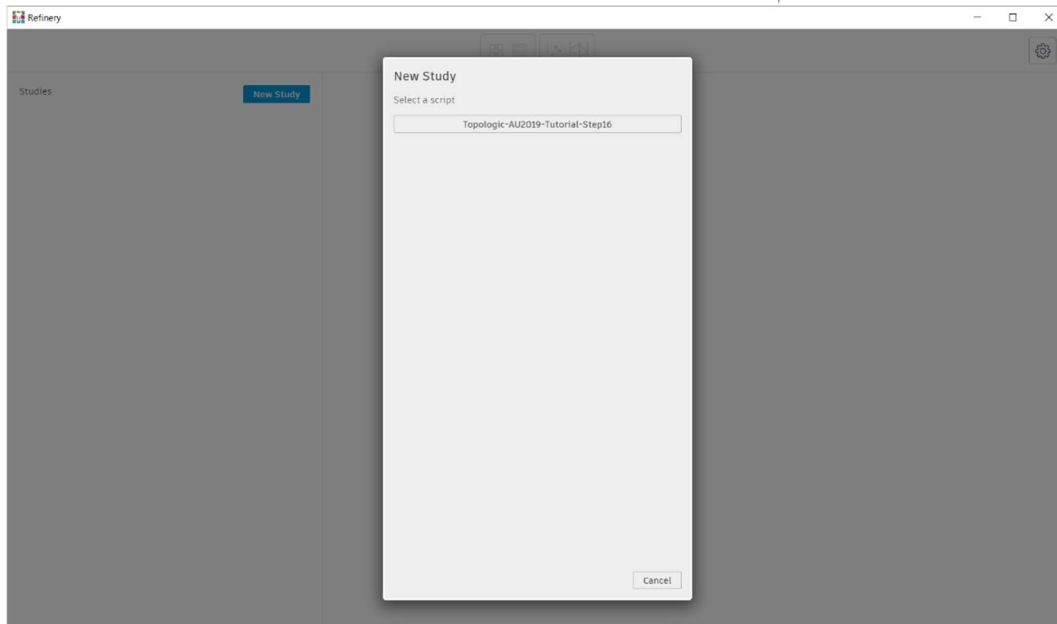
From the top menu in Dynamo, choose **Export for Refinery**. If all is correct, you should see the window below.



Step 17

From the top menu in Dynamo, choose **Launch Refinery**. A refinery window will appear. Press the New Study and choose the file name that you just exported.





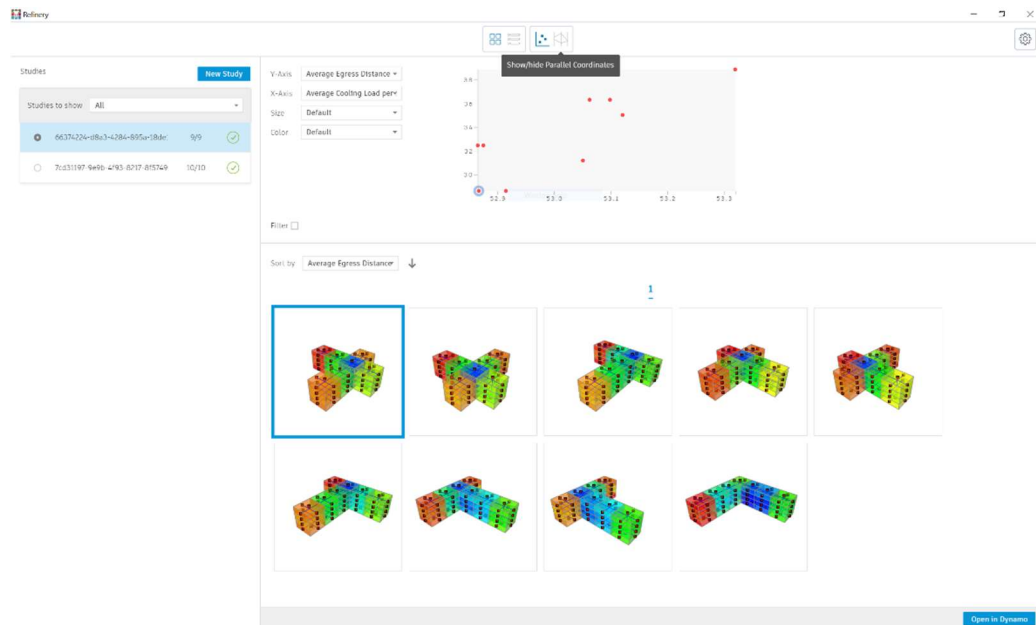
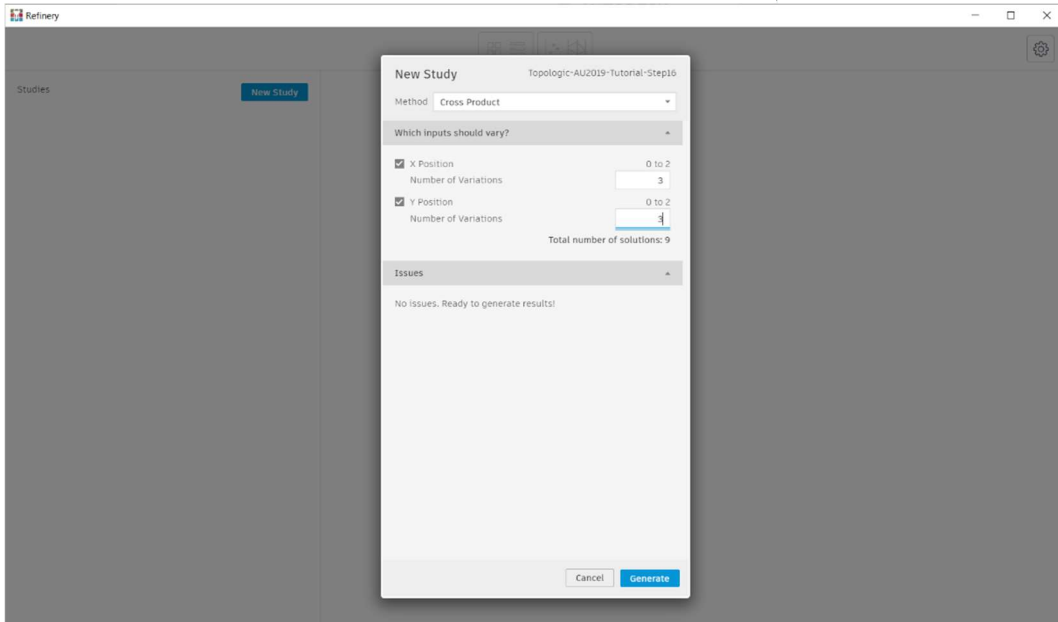
Refinery can explore possible design solutions using four methods:

- Optimize
- Cross Product
- Randomize
- Like This

The **Optimize** method is the most advanced. It uses a genetic algorithm to automatically explore the solution space, improve the result iteratively and choose a set of optimized solutions at the end. This advanced functionality also means it is time consuming. For this tutorial, we will use the **Cross Product** method which simply laces every step in the input parameters with every other step in all the other input parameters. For example, if you have two parameters each with 3 steps, this method would then create 3x3 (i.e. 9) possible solutions. It is then up to the user to choose the most appropriate solution. **Randomize**, as the name implies, chooses random steps from the input parameters and allows the user to specify the desired number of output solutions. The **Like This** option allows you to choose which inputs should vary and specify the desired number of output solutions.

For this tutorial, choose the **Cross Product** method and specify 3 for the **Number of Variations** for both the **X Position** and the **Y Position** input parameters. This will create 9 solutions.

Press the **Generate** button to generate the nine solutions. This will take some time. So kindly be patient. Once done, set the **X Axis** to be the **Average Shortest Path** and the **Y Axis** to be the **Average Cooling Load**. You can now use the buttons on top of the window to explore the solutions in various ways.



Congratulations on completing this tutorial! You are now versed in multidomain, multiobjective optimization. I hope you have enjoyed this tutorial and learned valuable techniques. We hope you will apply these skills and techniques in your own workflows and please feel free to get in touch to let us know what you have been up to.