Practical Design Computation

Dynamo for informed design and streamlined BIM

Autodesk University 2015

Nate Holland, nholland@nbbj.com
twitter: @_NateHolland

Nate Holland
nholland@nbbj.com

0_Class Description

Design computation is a way of thinking and approaching design that goes beyond the tools involved. This class will focus on how Dynamo fits into the design computation workflow at NBBJ and the larger architecture industry by walking through a series of example tools, processes and approaches to computation. Dynamo excels at bridging the gap between what Revit does really well and the ways designers want to work. The class will explore practical examples along two primary avenues: workflow automation, and rapid feedback. Key to both is data management. Revit holds a lot more information than it easily presents. By extracting data and synthesizing it with internal or external information, Dynamo can push that information back into the model or present it to the designer to drive informed decisions.

Learning Objectives

- Explain how Dynamo fits into the larger practice of Design Computation
- Automate actions and workflows in Revit using Dynamo
- Use Dynamo to provide rapid design feedback during the design process
- Recognize where Dynamo can improve the design and documentation of large scale projects

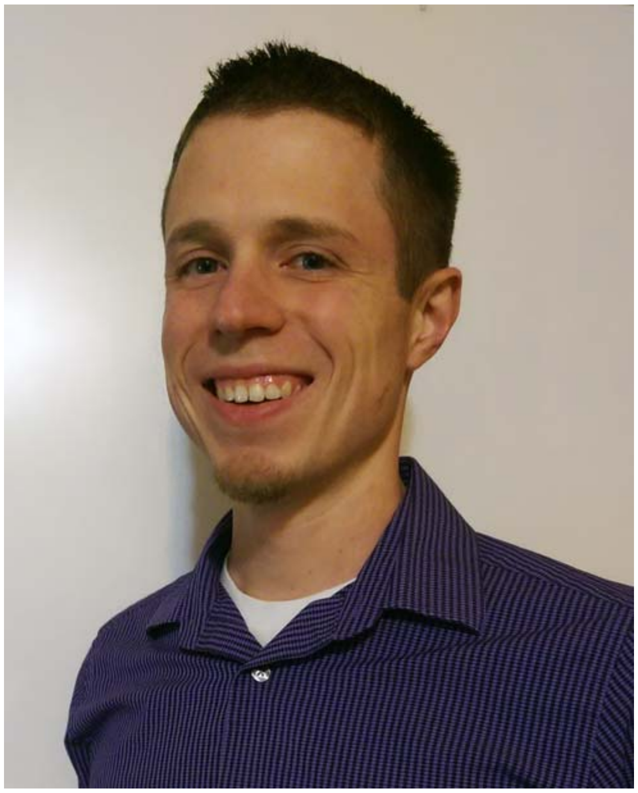
Class Agenda

Overview
- Dynamo and design computation
- plugins and prerequisites

Informed Design
- Preview geometry by data (Beginner level follow along)
- Measuring views with an isovist (Intermediate level overview)

Streamlined BIM
- Model maintenance with a purpose (Intermediate level follow along)
- Auto generating view numbers on sheets (Advanced level overview)

1_About the Speaker



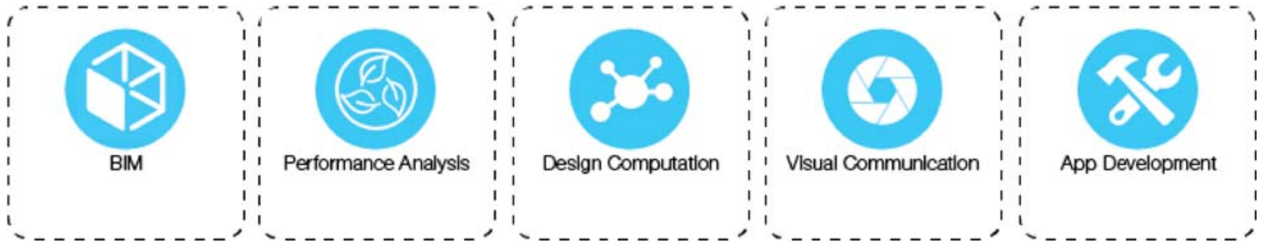
Nate is a Design Computation Leader and Designer at NBBJ. He is actively engaged in setting firm strategy for design computation, as well as developing custom tools and cross platform workflows to be deployed firmwide. He is the design computation lead for the Amazon.com Spheres project and the surrounding 3 block development where he developed custom tools for modeling, rationalizing, and fabricating the complex geometry of the Spheres' shell. He has master's degrees in both architecture and business administration and his work has been published in Fast Company, Wired, and Time Magazine. He has presented at conferences including: Autodesk's OTC, Facades+, Spaces and Flows, and ACADIA

2_Dynamo and Design Computation

2.01 Organization

- Digital practice group at NBBJ

nbbj digital practice

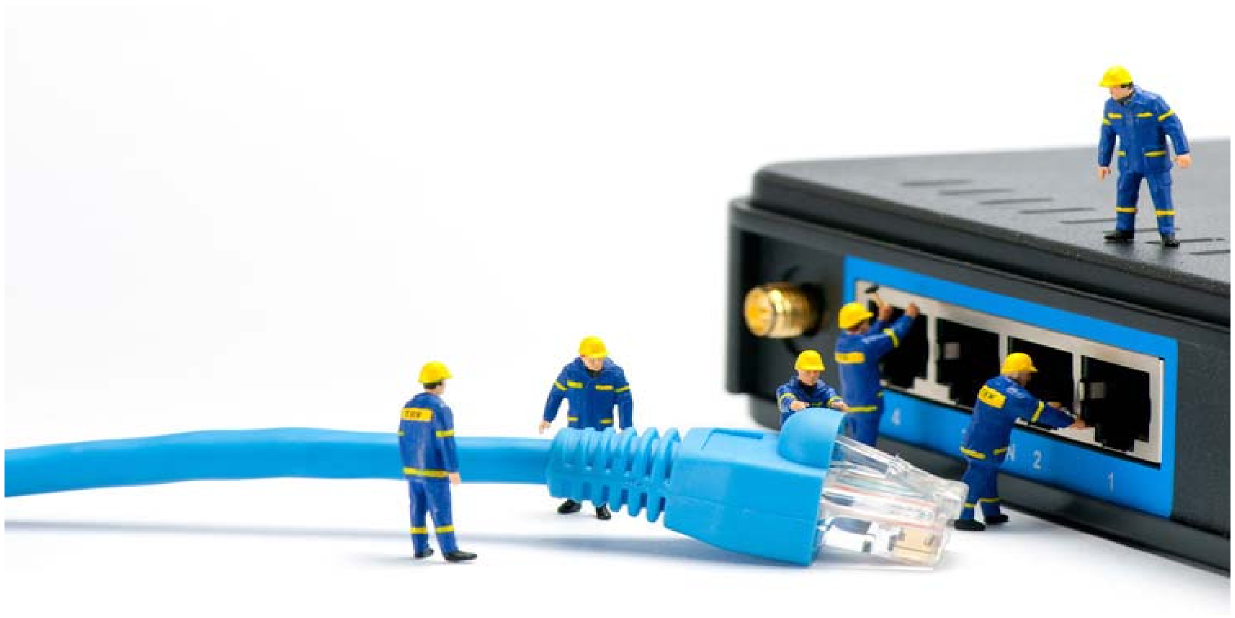


2.02 What is Design computation at NBBJ

- Design computation not computational design
- More than doubly curved geometry
- Design Computation augments a designer's skills and intuition
- Automating data and the visualization of that data
- Really all about data

2.03 Dynamo as a bridge

- Dynamo as a bridge
- Data centric approach
- Interoperability (geometry AND data transfer)
- Streamlining workflows



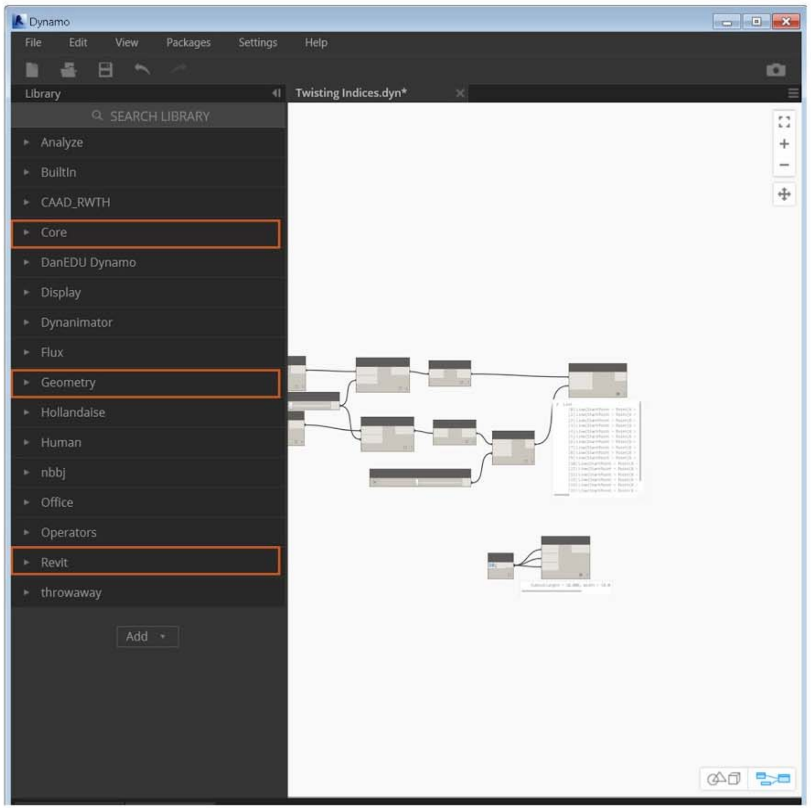
Nate Holland
nholland@nbbj.com

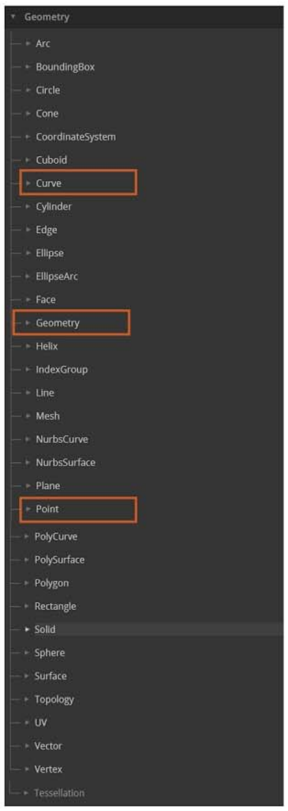
3_Getting on the same page

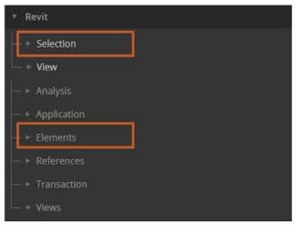
3.01 The Setup

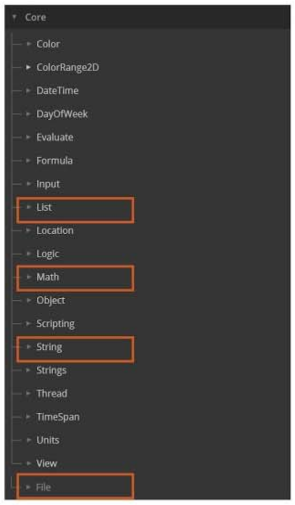
Revit Version: 2016 Service Pack 2
Dynamo version: 0.9.0 http://dynamobim.org/download/
Packages (plugins for dynamo): archi-lab 2015.11.11, Hollandaise 2015.11.12, Spring Nodes 82.1.2

3.02 The Dynamo interface and navigating the library









Most of the nodes I frequently use are under the Core, Geometry or Revit tabs in the library. Understanding fundamentally what goes where helps to navigate and browse the library. Core has things that deal largely with data: strings, lists math etc. Geometry deals with dynamo geometry which is completely different than Revit which deals with Revit elements. Yes Revit elements have geometry which you can extract but it is primarily operated with at the database level.

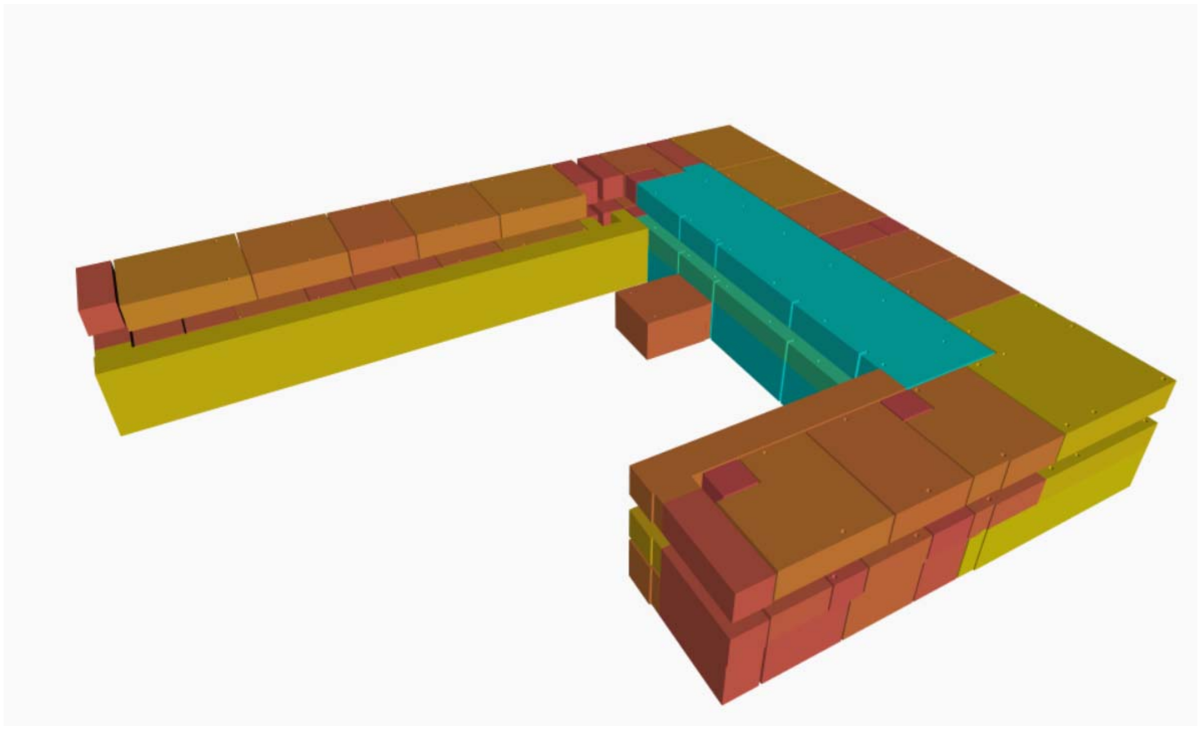
In addition to navigating and searching through the library structure you can type a node's name or keywords into the search bar about the library. Another way to access search a little quicker is to right click a blank spot on the canvas.

Nate Holland
nholland@nbbj.com

4_informed design through rapid feedback

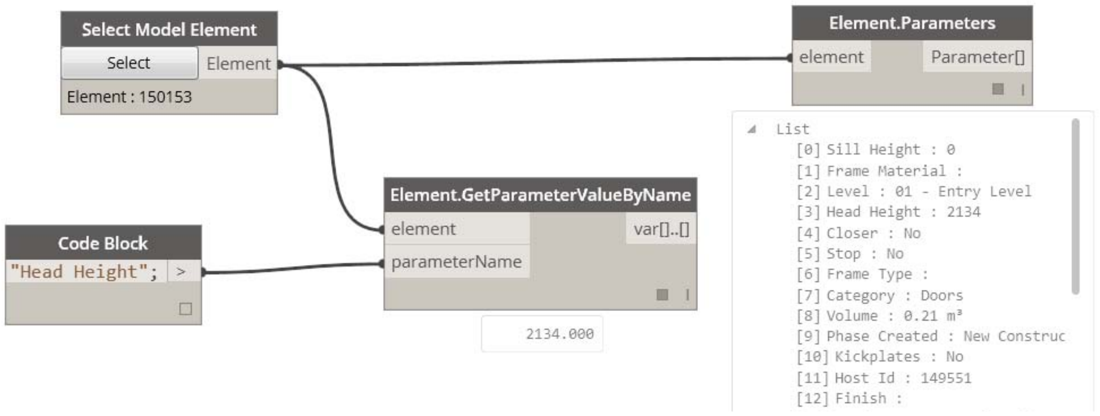
example one(walk through) Preview Geometry by Data

This example shows how to color code preview objects based on their parameter values. It will cover remapping a numeric value to a range and displaying the object in dynamo, as well as. grouping text parameters by their value and displaying the value by overriding the element's appearance in a revit view.



4.01 Get an element's parameter value

Use the Select Model Element Node to reference a revit element. Connect it to the Element.Parameters node to see what attributes Dynamo can access. Use a code block with "" around the parameter name to pass the desired parameter into an Element.GetParameterValueByName node.

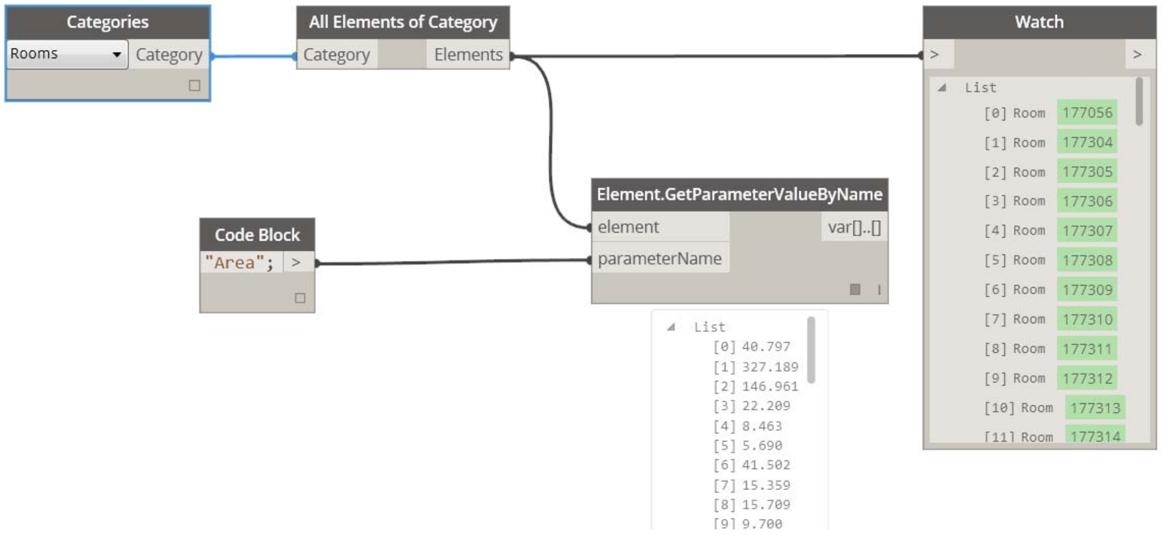


```
2134.000
```

```
List
  [0] Sill Height : 0
  [1] Frame Material :
  [2] Level : 01 - Entry Level
  [3] Head Height : 2134
  [4] Closer : No
  [5] Stop : No
  [6] Frame Type :
  [7] Category : Doors
  [8] Volume : 0.21 m³
  [9] Phase Created : New Construc
  [10] Kickplates : No
  [11] Host Id : 149551
  [12] Finish :
```

4.02 Select multiple elements

Not only can elements be selected manually one at a time, but Dynamo has a number of ways to filter and select multiple elements at the same time. Pass that whole list of elements into the GetParamValueByName node and notice it returns the values for each item in the list.

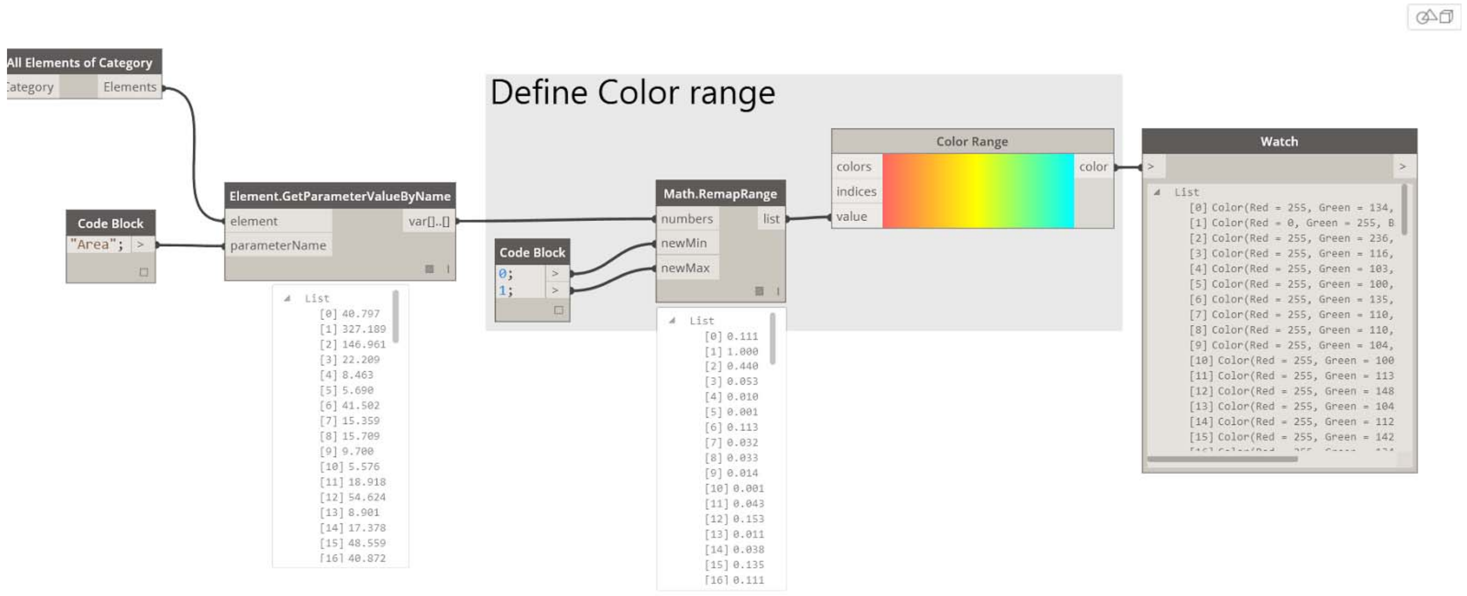


```
List
  [0] Room 177056
  [1] Room 177304
  [2] Room 177305
  [3] Room 177306
  [4] Room 177307
  [5] Room 177308
  [6] Room 177309
  [7] Room 177310
  [8] Room 177311
  [9] Room 177312
  [10] Room 177313
  [11] Room 177314
```

```
List
  [0] 40.797
  [1] 327.189
  [2] 146.961
  [3] 22.209
  [4] 8.463
  [5] 5.690
  [6] 41.502
  [7] 15.359
  [8] 15.709
  [9] 9.700
```

4.03 From Areas to Colors

The color range node takes a list of numbers between 0 and 1 and outputs a color based on how the numbers fall across the spectrum. Use the Math.RemapRange to change the range of room areas to a list of values between 0 and 1.



```
List
  [0] 40.797
  [1] 327.189
  [2] 146.961
  [3] 22.209
  [4] 8.463
  [5] 5.690
  [6] 41.502
  [7] 15.359
  [8] 15.709
  [9] 9.700
  [10] 5.576
  [11] 18.918
  [12] 54.624
  [13] 8.901
  [14] 17.378
  [15] 48.559
  [16] 40.872
```

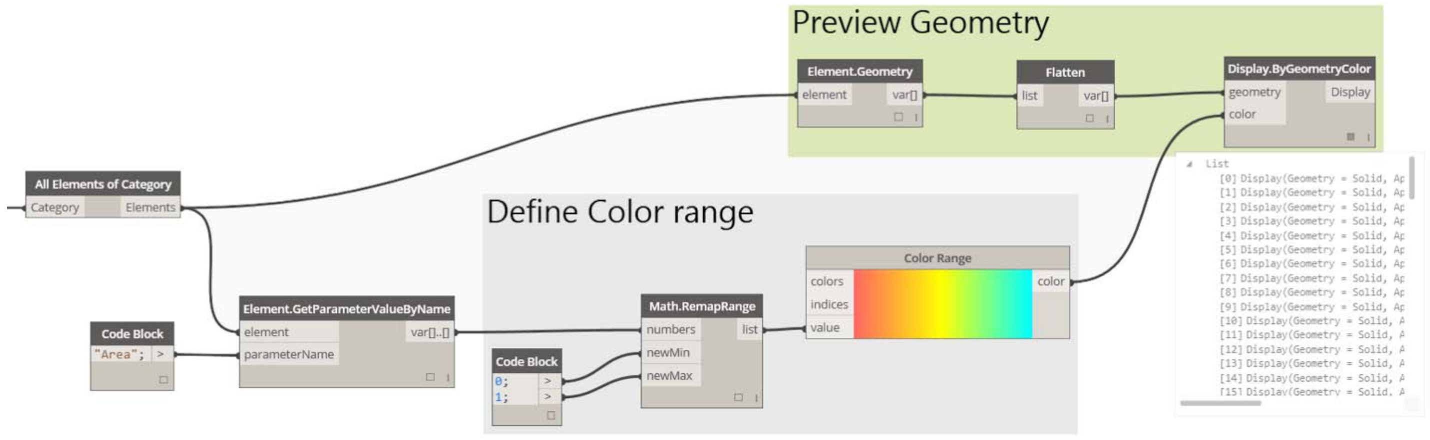
```
List
  [0] 0.111
  [1] 1.000
  [2] 0.440
  [3] 0.053
  [4] 0.010
  [5] 0.001
  [6] 0.113
  [7] 0.032
  [8] 0.033
  [9] 0.014
  [10] 0.001
  [11] 0.043
  [12] 0.153
  [13] 0.011
  [14] 0.038
  [15] 0.135
  [16] 0.111
```

```
List
  [0] Color(Red = 255, Green = 134,
  [1] Color(Red = 0, Green = 255, B
  [2] Color(Red = 255, Green = 236,
  [3] Color(Red = 255, Green = 116,
  [4] Color(Red = 255, Green = 103,
  [5] Color(Red = 255, Green = 100,
  [6] Color(Red = 255, Green = 135,
  [7] Color(Red = 255, Green = 110,
  [8] Color(Red = 255, Green = 110,
  [9] Color(Red = 255, Green = 104,
  [10] Color(Red = 255, Green = 100
  [11] Color(Red = 255, Green = 113
  [12] Color(Red = 255, Green = 148
  [13] Color(Red = 255, Green = 104
  [14] Color(Red = 255, Green = 112
  [15] Color(Red = 255, Green = 142
```

Nate Holland
nholland@nbbj.com

4.04 Preview the geometry

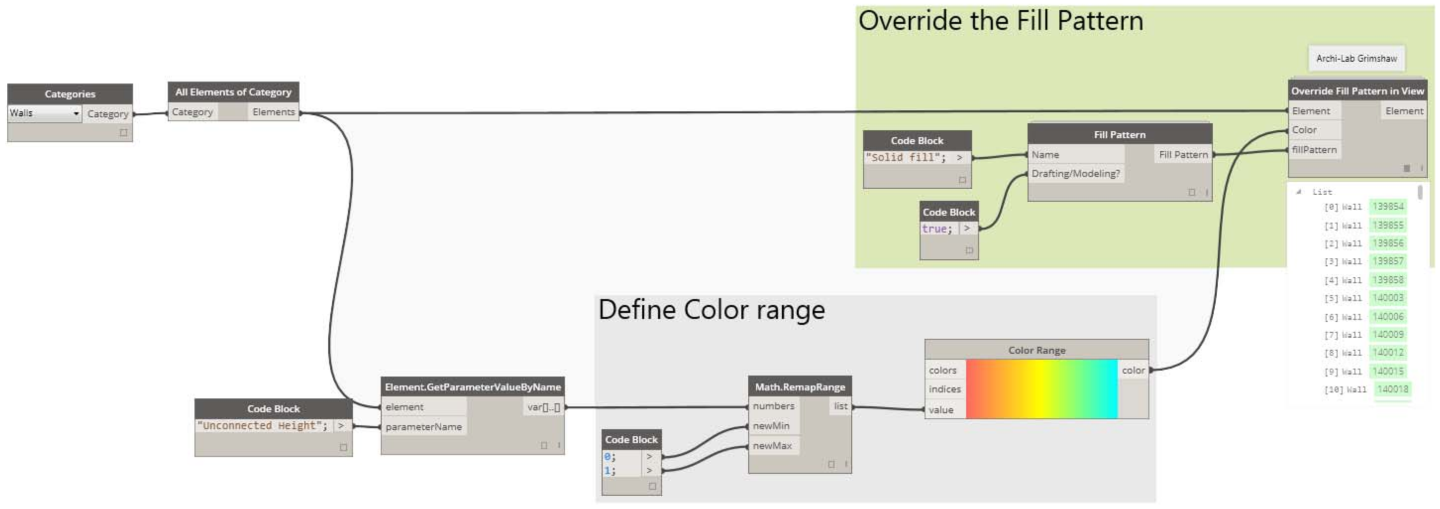
Use Element.Geometry to extract the geometry from the revit elements. Because each element might have multiple pieces of geometry the geometry is placed in separate sublists. Use the flatten node to smash all of the geometry down into one list before feeding it into the display.ByGeometryColor.



```
List
  [0] Display(Geometry = Solid, Ap
  [1] Display(Geometry = Solid, Ap
  [2] Display(Geometry = Solid, Ap
  [3] Display(Geometry = Solid, Ap
  [4] Display(Geometry = Solid, Ap
  [5] Display(Geometry = Solid, Ap
  [6] Display(Geometry = Solid, Ap
  [7] Display(Geometry = Solid, Ap
  [8] Display(Geometry = Solid, Ap
  [9] Display(Geometry = Solid, Ap
  [10] Display(Geometry = Solid, A
  [11] Display(Geometry = Solid, A
  [12] Display(Geometry = Solid, A
  [13] Display(Geometry = Solid, A
  [14] Display(Geometry = Solid, A
  [15] Display(Geometry = Solid, A
```

4.05 Preview wall data instead

Change the Category drop down to Walls and change the code block to "Unconnected Height". Rather than previewing the walls in Dynamo we'd like to override their graphics in the active view so that we can print the plan. Use the Override Fill Pattern in View node from the Archi-Lab package instead of the Display.ByGeometryColor

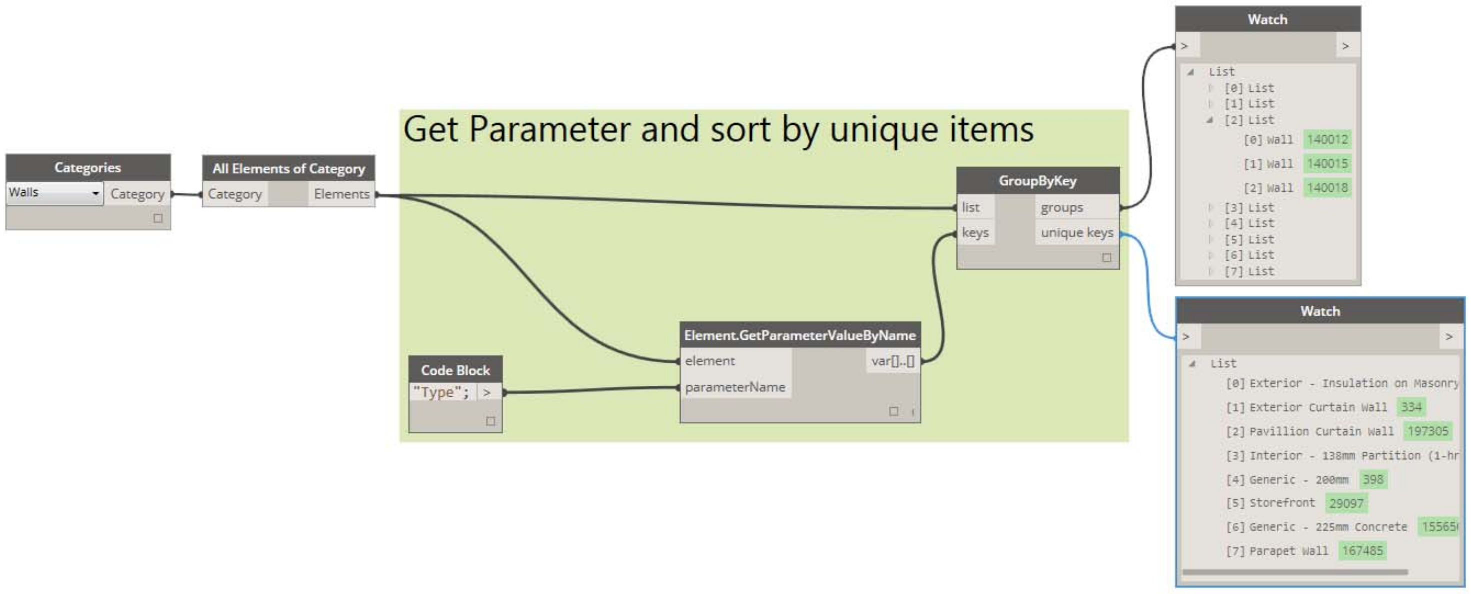


```
List
  [0] Wall  139854
  [1] Wall  139855
  [2] Wall  139856
  [3] Wall  139857
  [4] Wall  139858
  [5] Wall  140003
  [6] Wall  140006
  [7] Wall  140009
  [8] Wall  140012
  [9] Wall  140015
  [10] Wall  140018
```

Nate Holland
nholland@nbbj.com

4.06 Group the walls by type

Not every parameter is numeric and can be mapped to use with the color range node.Using the Group by Key node you can sort the elements into a list of lists based on the type values passed into the keys

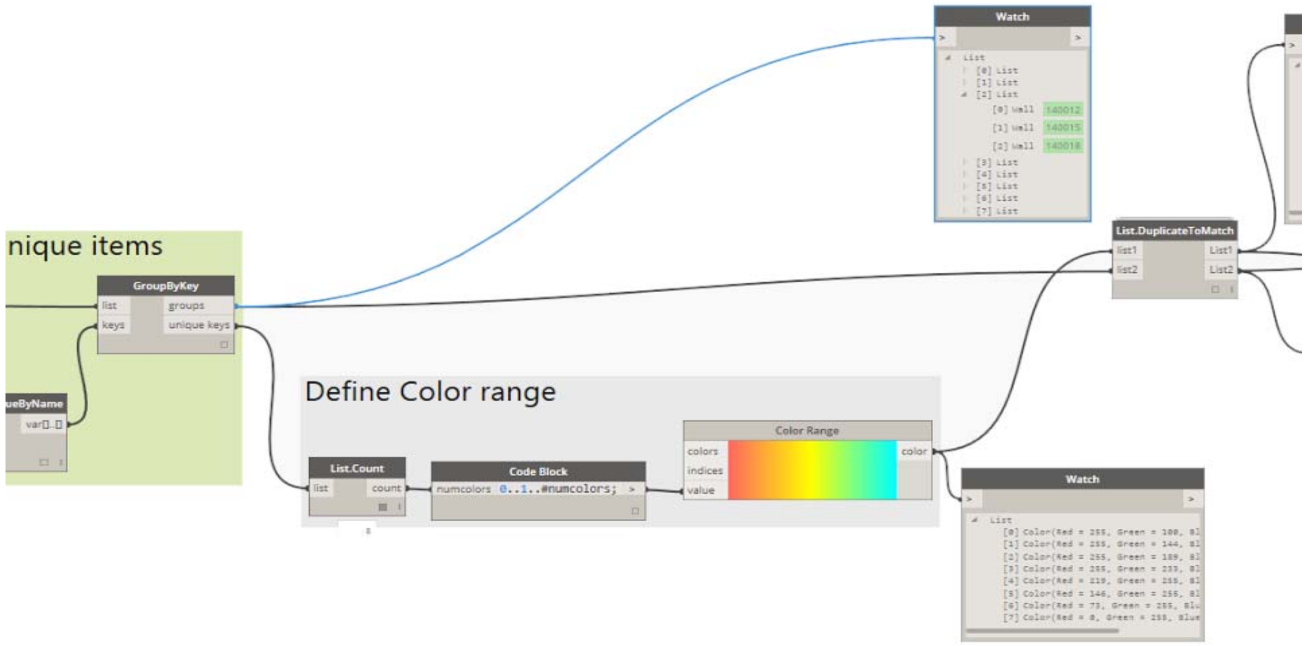


```
List
  [0] List
  [1] List
  [2] List
    [0] Wall 140012
    [1] Wall 140015
    [2] Wall 140018
  [3] List
  [4] List
  [5] List
  [6] List
  [7] List
```

```
List
  [0] Exterior - Insulation on Masonry
  [1] Exterior Curtain Wall 334
  [2] Pavillion Curtain Wall 197305
  [3] Interior - 138mm Partition (1-hr
  [4] Generic - 200mm 398
  [5] Storefront 29097
  [6] Generic - 225mm Concrete 15565
  [7] Parapet Wall 167485
```

4.07 Create the colors by type

With a list count node you can identify how many items are in the unique keys output of the group by keys node. Pass that count into a code block to create a series between 0 and 1 with that many steps to pass into the color range.



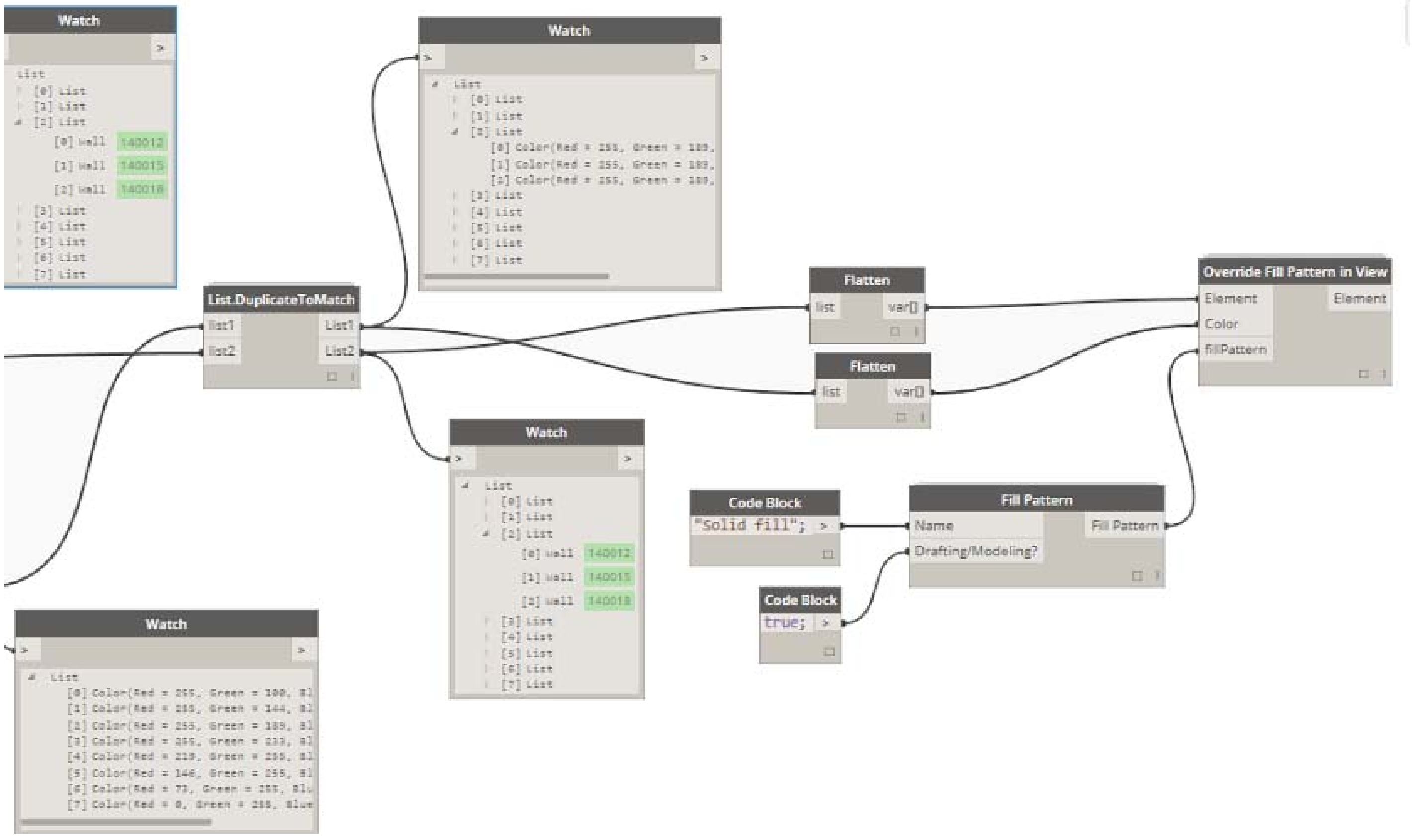
```
List
  [0] List
  [1] List
  [2] List
    [0] Wall 140012
    [1] Wall 140015
    [2] Wall 140018
  [3] List
  [4] List
  [5] List
  [6] List
  [7] List
```

```
List
  [0] Color(Red = 255, Green = 100, Bl
  [1] Color(Red = 255, Green = 144, Bl
  [2] Color(Red = 255, Green = 189, Bl
  [3] Color(Red = 255, Green = 233, Bl
  [4] Color(Red = 219, Green = 255, Bl
  [5] Color(Red = 146, Green = 255, Bl
  [6] Color(Red = 73, Green = 255, Blu
  [7] Color(Red = 0, Green = 255, Blue
```

Nate Holland
nholland@nbbj.com

4.08 Match the Color and Element Data structures

Use the List.DuplicateToMatch from Hollandaise to duplicate the colors to match the list of list data strcture coming out of the group by keys node. Then flatten both lists and pass tehm into an Override Fill Pattern in View node from Archi-Lab along with the appropriate fill pattern.



```
List
  [0] List
  [1] List
  [2] List
    [0] Wall  140012
    [1] Wall  140015
    [2] Wall  140018
  [3] List
  [4] List
  [5] List
  [6] List
  [7] List
```

```
List
  [0] List
  [1] List
  [2] List
    [0] Color(Red = 255, Green = 189,
    [1] Color(Red = 255, Green = 189,
    [2] Color(Red = 255, Green = 189,
  [3] List
  [4] List
  [5] List
  [6] List
  [7] List
```

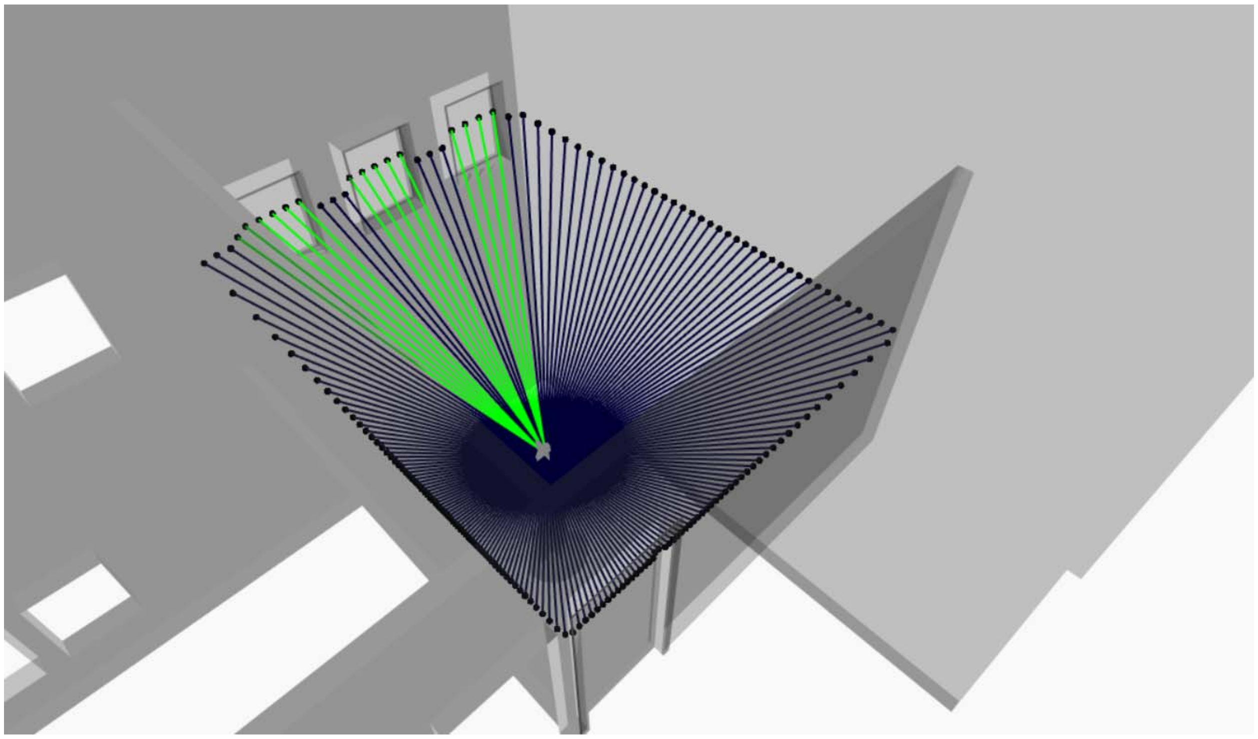
```
List
  [0] List
  [1] List
  [2] List
    [0] Wall  140012
    [1] Wall  140015
    [2] Wall  140018
  [3] List
  [4] List
  [5] List
  [6] List
  [7] List
```

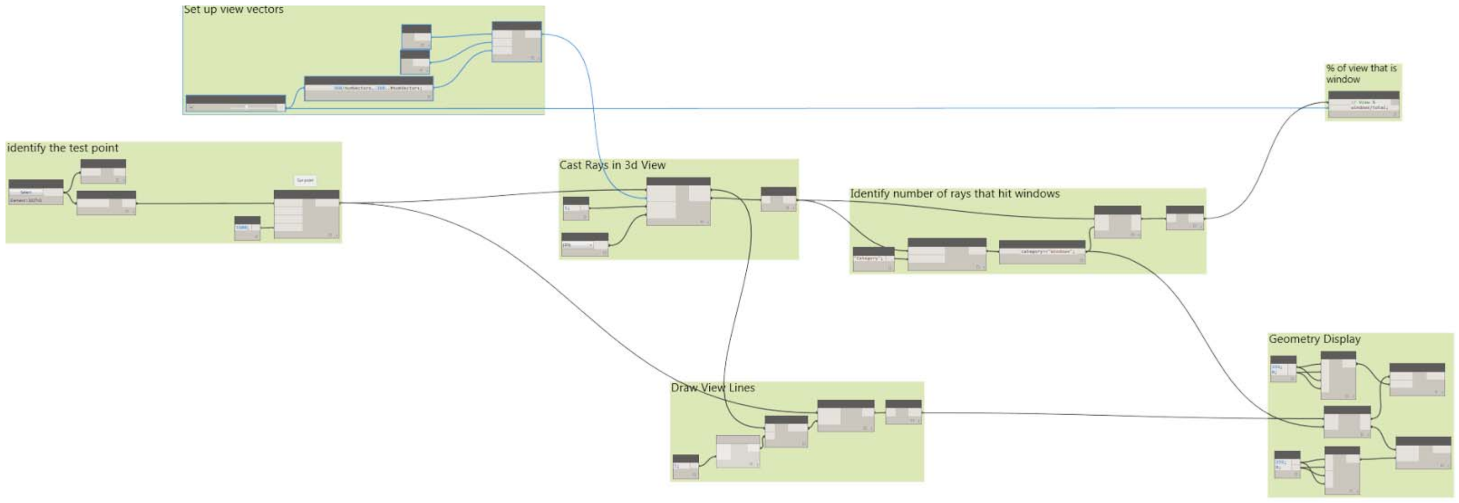
```
List
    [0] Color(Red = 255, Green = 100, Bl
    [1] Color(Red = 255, Green = 144, Bl
    [2] Color(Red = 255, Green = 189, Bl
    [3] Color(Red = 255, Green = 233, Bl
    [4] Color(Red = 219, Green = 255, Bl
    [5] Color(Red = 146, Green = 255, Bl
    [6] Color(Red = 73, Green = 255, Blu
    [7] Color(Red = 0, Green = 255, Blue
```

Nate Holland
nholland@nbbj.com

example two(description) Views by Isovist

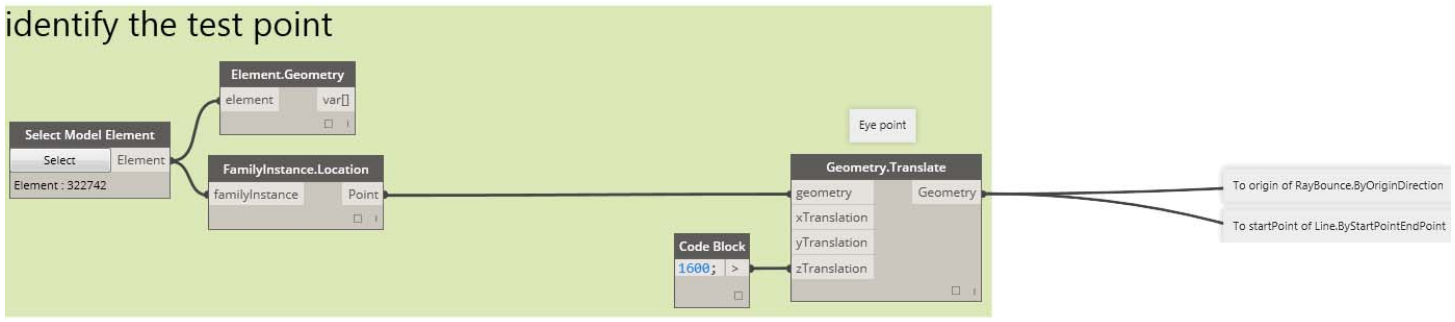
Understanding what people can see within a space is an important aspect of architectural design from hospital patient rooms to restrooms or office layouts. This example shows how to calculate what portion of a person's view from a given point is a window as a percentage and displays that graphically as colored lines.





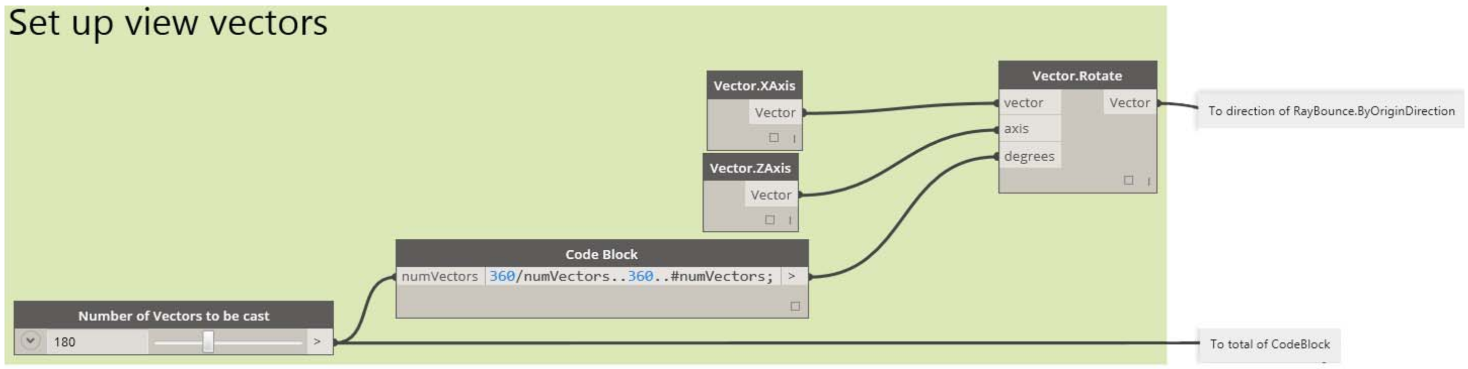
4.09 Identify the point for casting the view rays

Select the Entourage person and extract its location point. Move that location point up in the z direction.



4.10 Set up the array of view lines as vectors

To create a variable number of evenly spaced view rays extending radially from the view point use the Vector.Rotate node to rotate the x vector around the z vector. Use a codeblock to create a series with a variable number of steps and pass the resultant values in as the degrees to rotate.

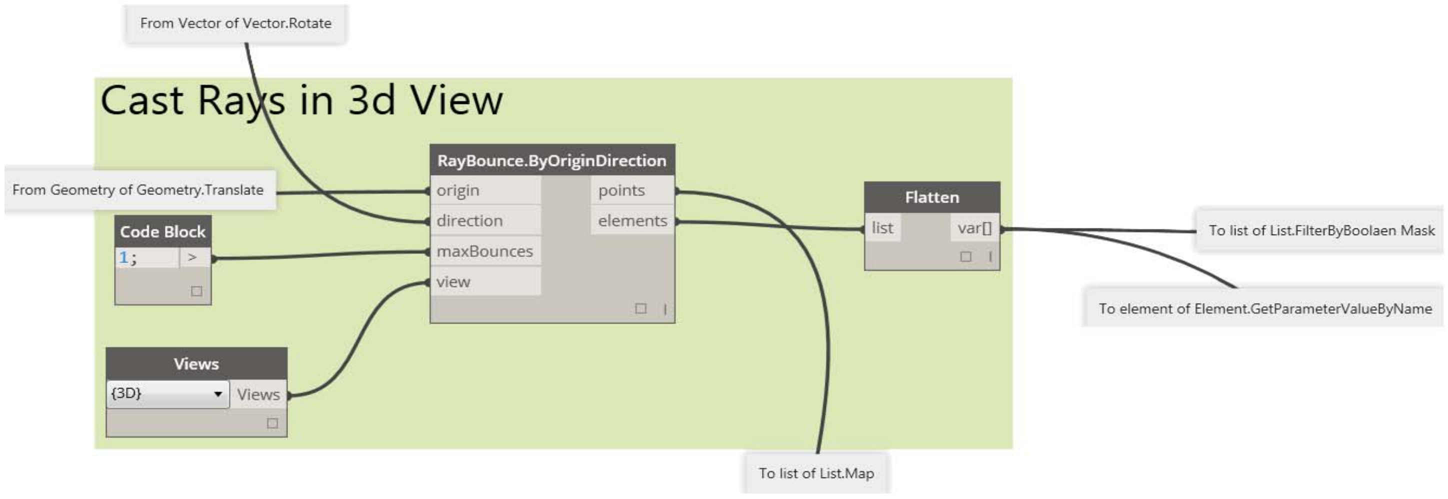


```
360/numVectors..360..#numVectors;
```

Nate Holland
nholland@nbbj.com

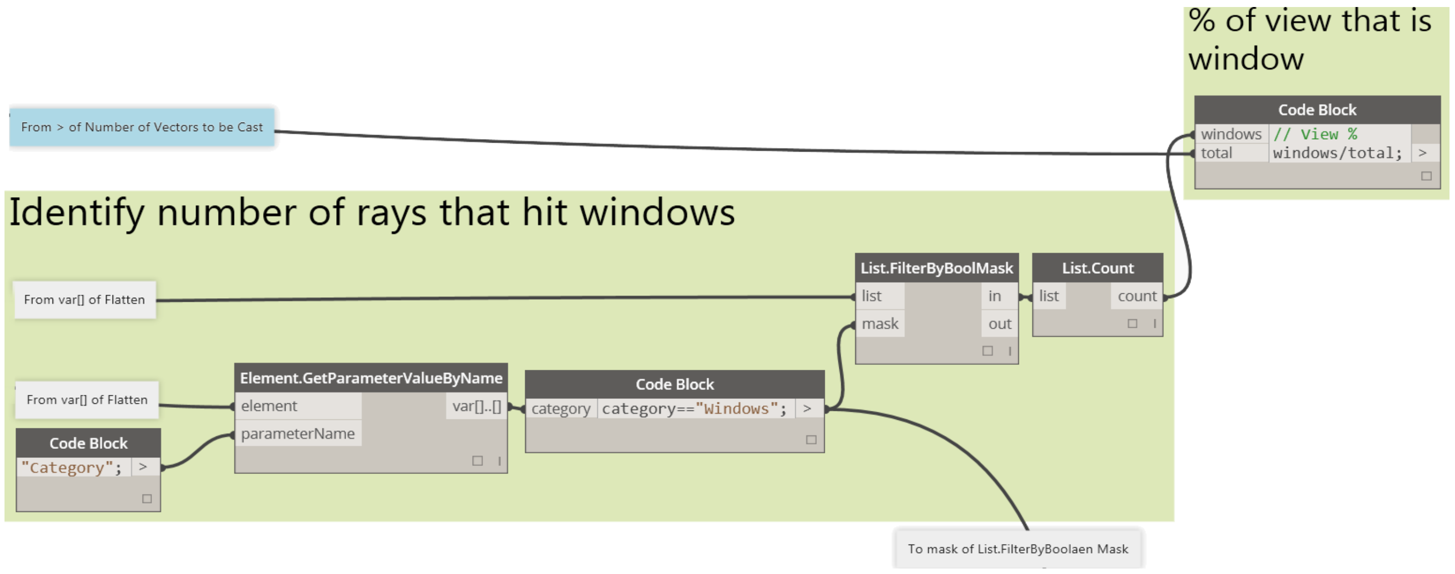
4.11 Use the RayBounce node to see what objects in the 3d view are hit by the rays

The RayBounce node does most of the work for this example. Give it an origin point and the direction the rays are traveling away from that point. Set the number of bounces to 1 because we only care about direct line of site for this example and then pass in the default 3d view so Revit knows what geometry to use. The output of this node is a list of the elements hit by the rays and a list of the intersection points.


From Vector of Vector.Rotate
Cast Rays in 3d View
From Geometry of Geometry.Translate
RayBounce.ByOriginDirection
origin
direction
maxBounces
view
points
elements
Code Block
1;
Views
{3D}
Views
Flatten
list
var[]
To list of List.FilterByBoolaen Mask
To element of Element.GetParameterValueByName
To list of List.Map

4.12 test to see if the hit objects are windows and calculate %

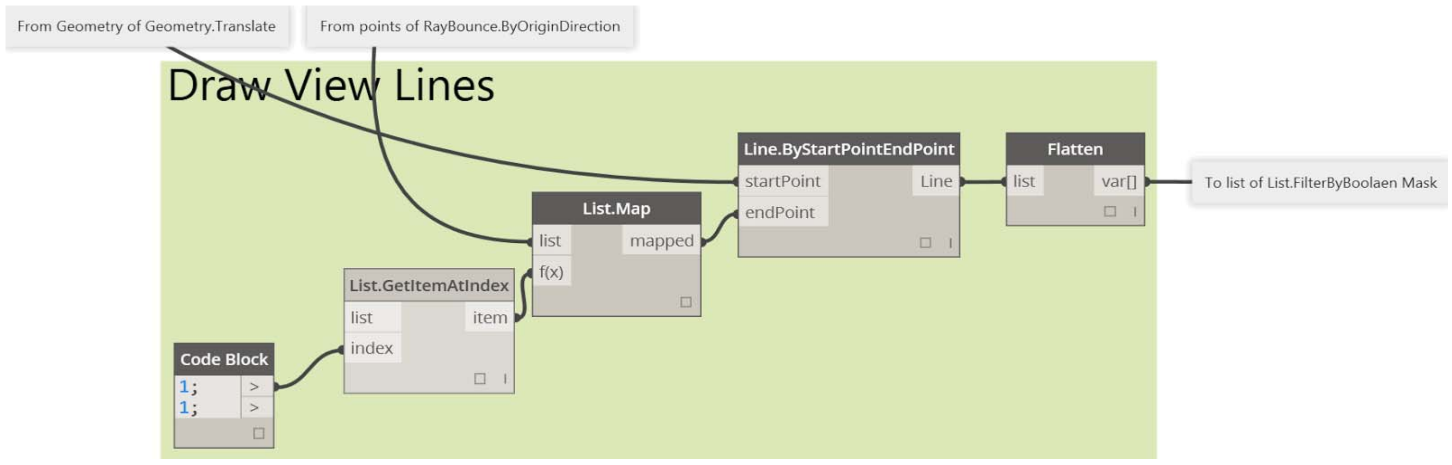
Get the category of the hit elements and set up a code block to test if they are windows. Use the List.FilterByBooleanMask to split the elements into two lists. Count the number of rays that hit windows and divide by the total number of rays cast.



Nate Holland
nholland@nbbj.com

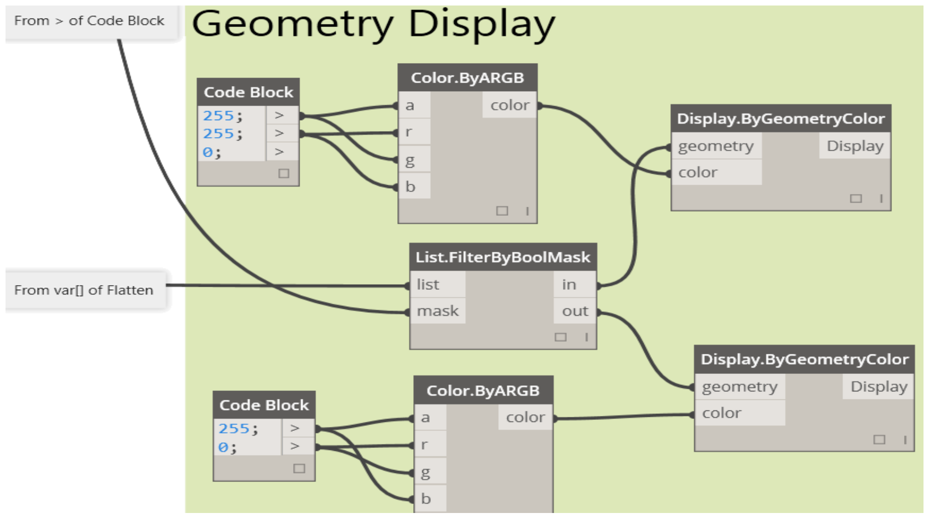
4.13 Create lines from the eye point to each of the intersecting points

To visualize the results create a line between the original eye point and each of the intersection points from the ray bounce.


From Geometry of Geometry.Translate
From points of RayBounce.ByOriginDirection
Draw View Lines
Line.ByStartPointEndPoint
startPoint
Line
endPoint
Flatten
list
var[]
To list of List.FilterByBoolaen Mask
List.Map
list
mapped
f(x)
List.GetItemAtIndex
list
item
index
Code Block
1;
1;

4.14 Filter the lines by their category and display windows as green

The same list of boolean values that split the list of elements above can be used to split the lines into two lists, color the lines that hit windows green and those that hit anything else blue with the Display.ByGeometryColor node.

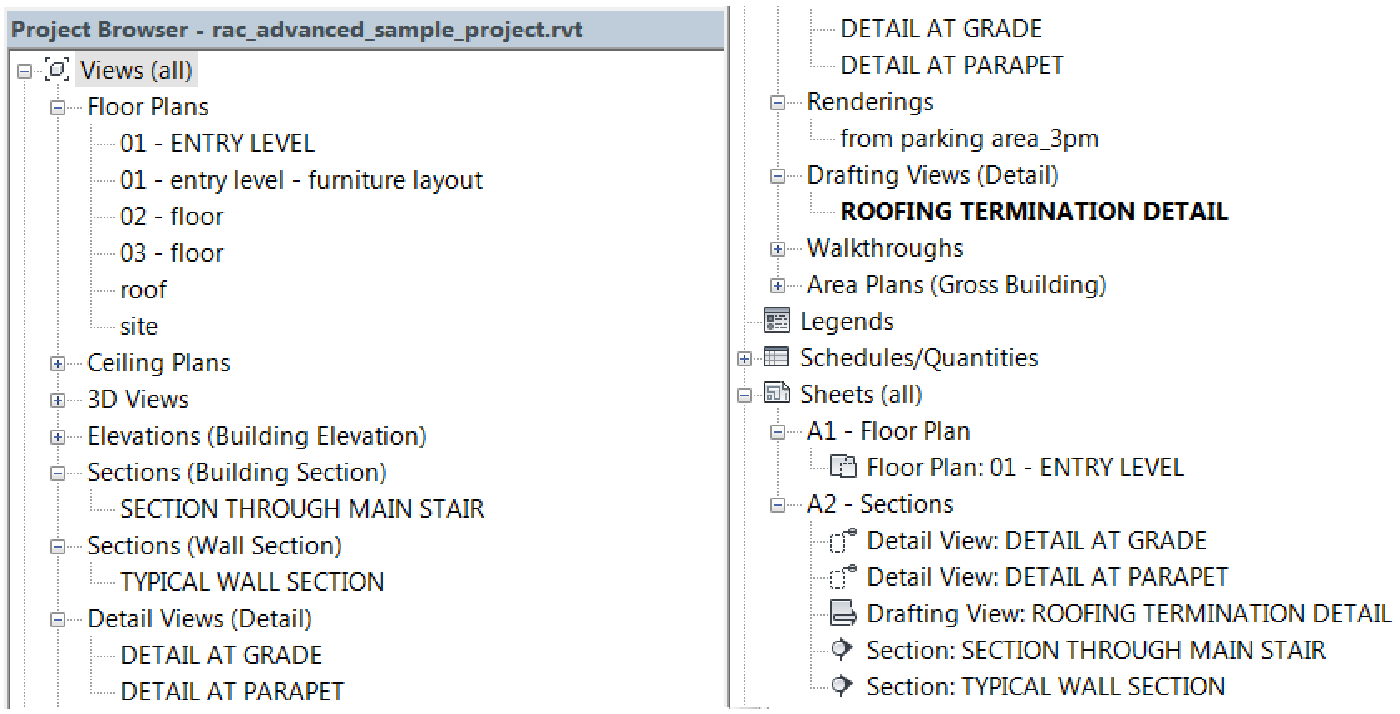


Nate Holland
nholland@nbbj.com

5_Streamlined BIM

example three(walk through) Model Maintenance with a Purpose

The prupose of this graph is to cycle through all of the views in the document and identify which ones are placed on sheets by setting the characters in their name to uppercase while setting the names of all the unplaced views to lowercase. This helps ensure that the placed views are formatted consistently with project standards while also indicating in the project browser which views are not currently part of the document set.



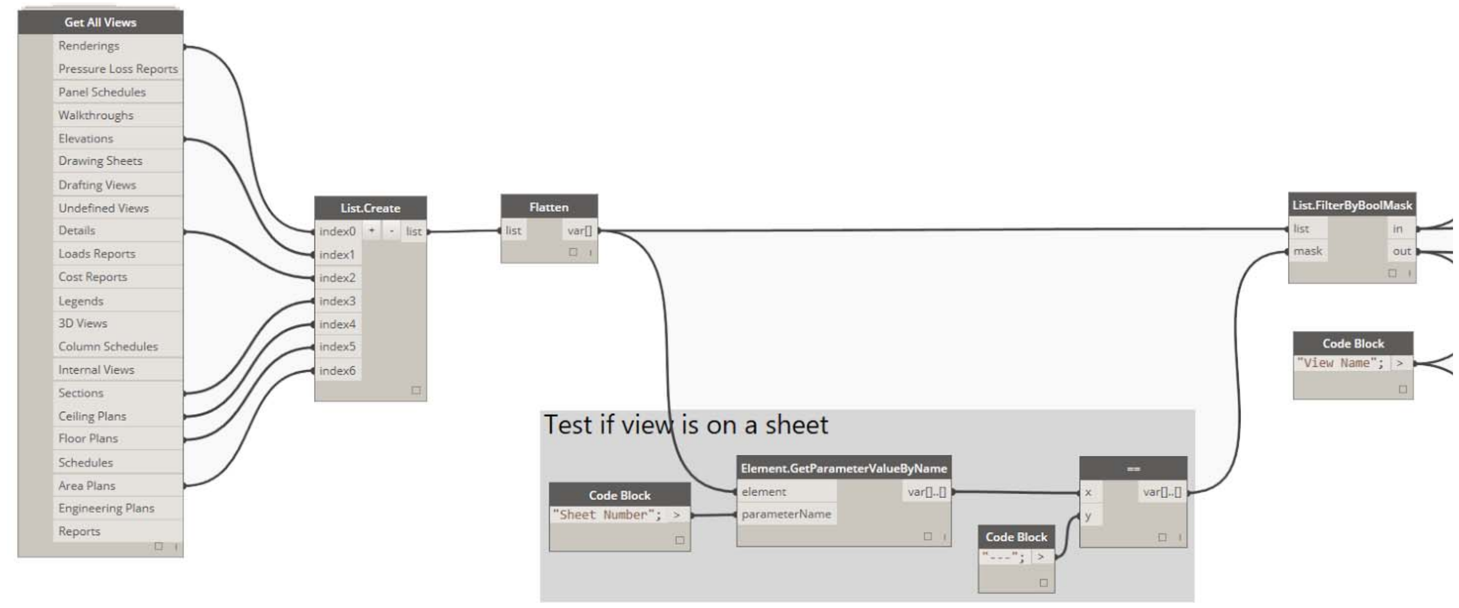
```
Project Browser - rac_advanced_sample_project.rvt
Views (all)
    Floor Plans
        01 - ENTRY LEVEL
        01 - entry level - furniture layout
        02 - floor
        03 - floor
        roof
        site
    Ceiling Plans
    3D Views
    Elevations (Building Elevation)
    Sections (Building Section)
        SECTION THROUGH MAIN STAIR
    Sections (Wall Section)
        TYPICAL WALL SECTION
    Detail Views (Detail)
        DETAIL AT GRADE
        DETAIL AT PARAPET

        DETAIL AT GRADE
        DETAIL AT PARAPET
    Renderings
        from parking area_3pm
    Drafting Views (Detail)
        ROOFING TERMINATION DETAIL
    Walkthroughs
    Area Plans (Gross Building)
Legends
Schedules/Quantities
Sheets (all)
    A1 - Floor Plan
        Floor Plan: 01 - ENTRY LEVEL
    A2 - Sections
        Detail View: DETAIL AT GRADE
        Detail View: DETAIL AT PARAPET
        Drafting View: ROOFING TERMINATION DETAIL
        Section: SECTION THROUGH MAIN STAIR
        Section: TYPICAL WALL SECTION
```

Nate Holland
nholland@nbbj.com

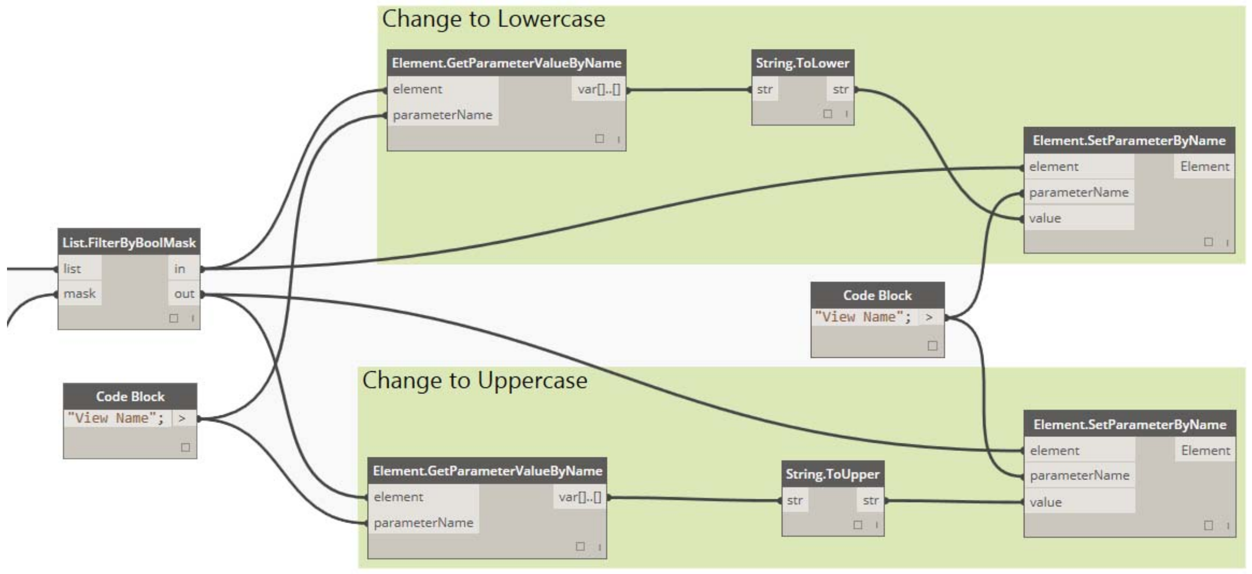
5.01 Get the views and their sheet numbers

Use the Archi-Lab Get All Views node in combination with a list.create to select all of the view categories that you'd like to manage and then flatten the list. Use the Element.GetParameterValueByName to get each view's sheet number. Views that are not on a sheet will return three dashes "---". We can use an == node to test if the parameter value is --- and return a list of true or false boolean values.



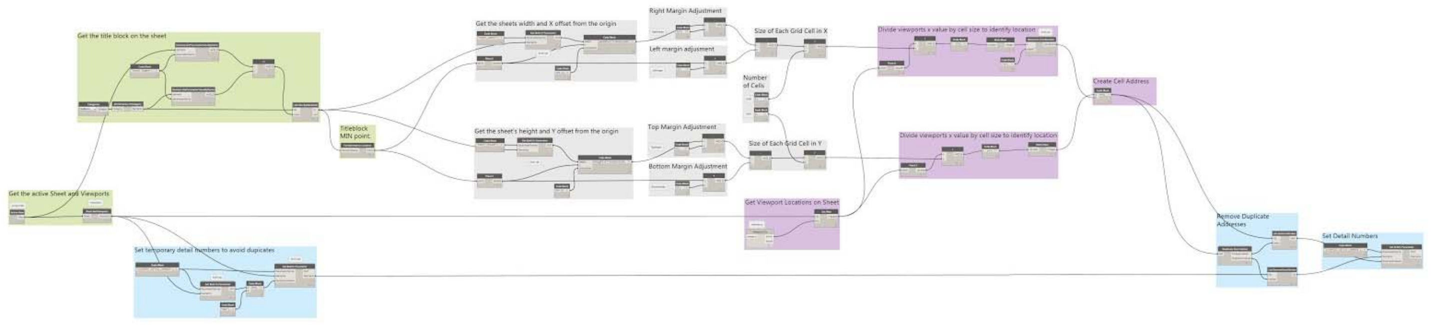
5.03 Change the case names

Use the results from the == node to filter the list of views, passing all of the views on a sheet to the out output and all of the views without sheets to the in node. Get the names from the sheets using the Element.GetParameterValueByName node, use the String.ToUpper or ToLower to change the case of the text and then reassign it with the Element.SetParameterByName



example four (description) View Number on Sheets

Many company and national BIM standards use an address format for numbering views and details on sheets. When done manually the process of numbering and renumbering views as the project progresses can be time consuming. This example shows how to set up a dynamo graph to automatically renumber all of the views based on the grid cell that their lower left corner is in. This example requires significantly more nodes and therefore will be covered at more of a conceptual level. The completed file along with a high resolution image will be included with the course materials.

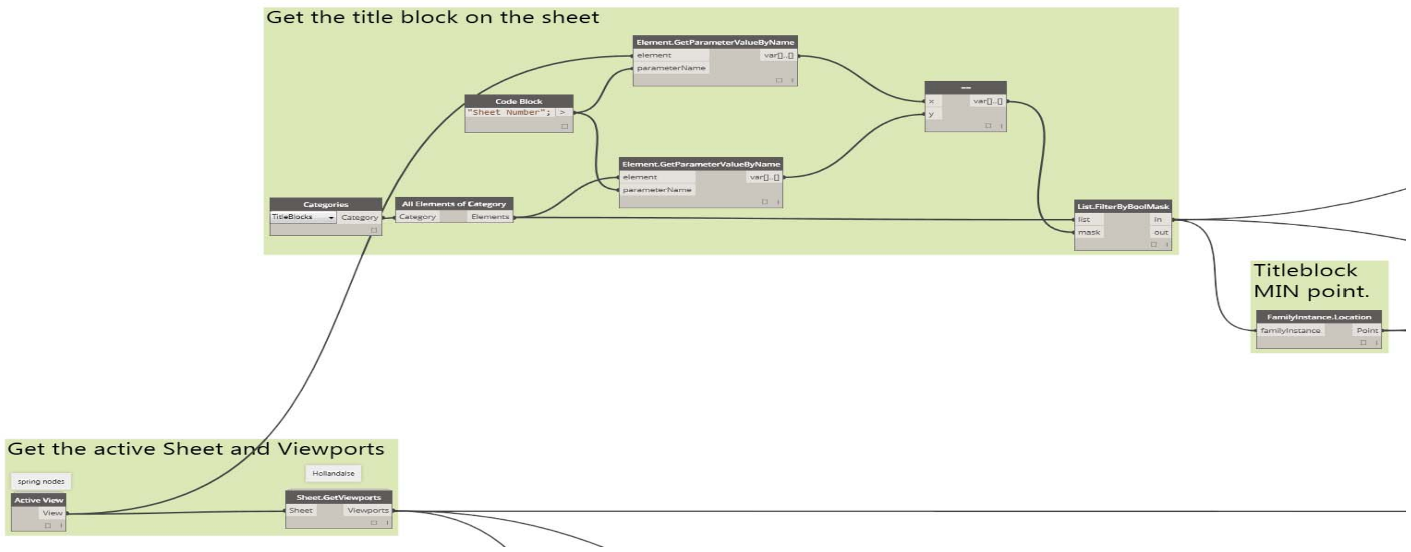


5.04 Overview

When assembling larger or more complex graphs it helps to first identify the primary steps necessary to achieve the goal before jumping in. This graph can be broken into 4 parts identified by the colors above, 1) Extracting the sheet and viewport information from the model, 2) identifying the grid cell size, 3) Determining what cell each view is in, and 4) Changing the viewport detail numbers

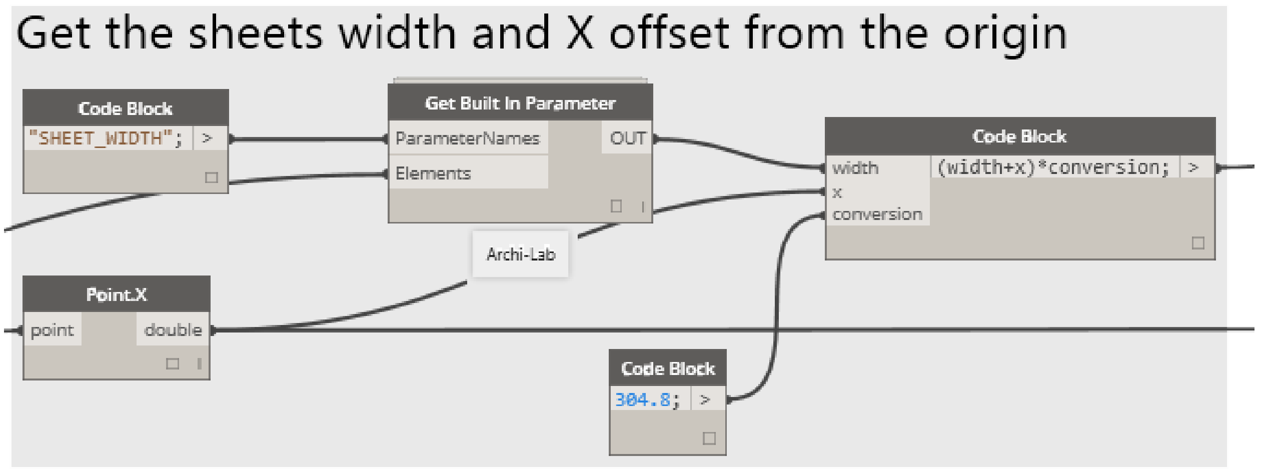
5.05 Extracting Sheet and Viewport Info

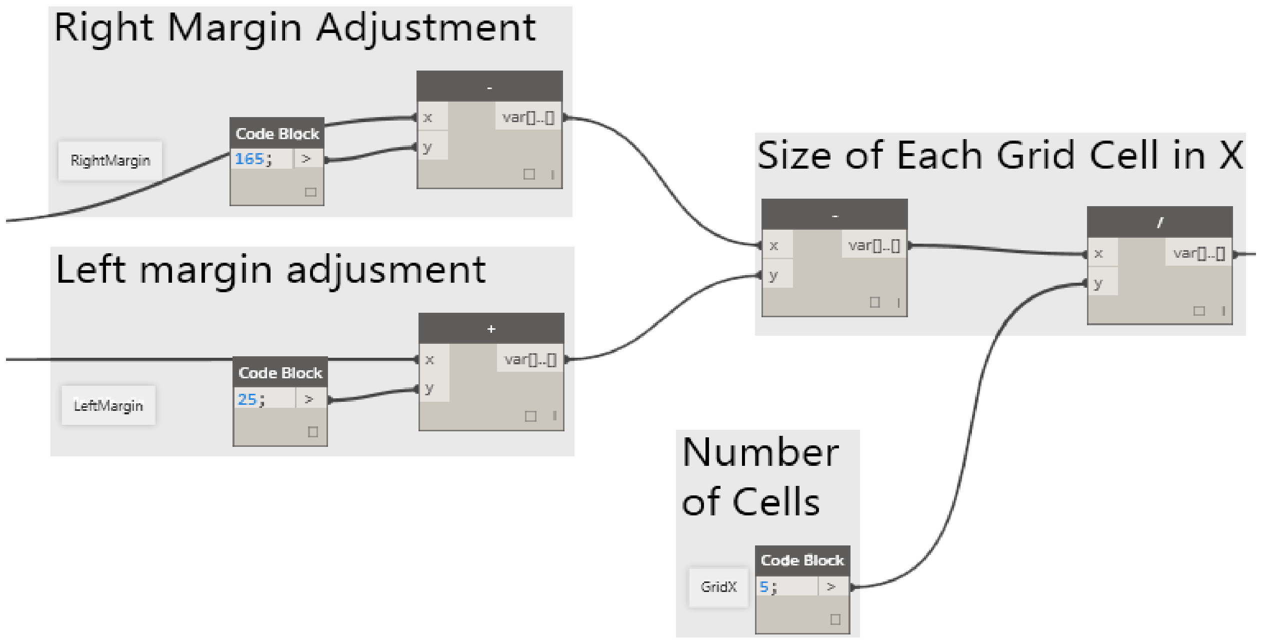
The active view node from Spring Nodes will load in the sheet if it is open as the active view in Revit. The Sheet.GetViewports from the Hollandaise package will let you extract the viewports on that sheet. To get the title block load in all of the title blocks by category and compare the sheet number parameter to match the Title Block with the active sheet. The FamilyInstance.Location will identify where the titleblock has been placed in the sheet's XY coordinate system.



5.06 identifying the grid cell size

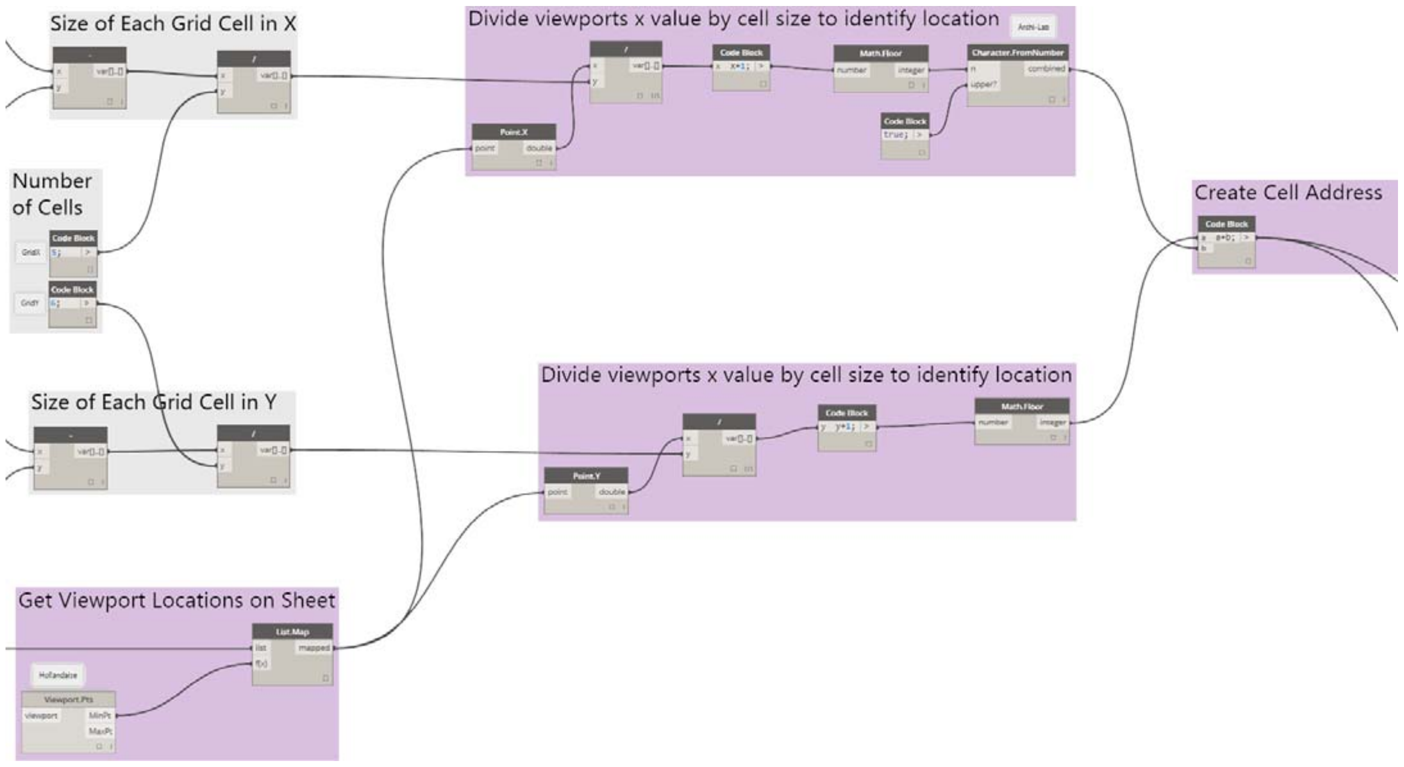
Use the Get Built In Parameter Node from Archi-Lab to get the width of the title block and add it to the X value of the placement point of the title block to get the total range of x values within the titleblock. Watch out for unit conversions, for some reason titleblocks measure in feet even when the document is in mm. Next subtract the width of the right margin and add the width of the left margin. This will give you the domain of the usable portion of the title block. The codeblock acts as a user input to enter the number of cells in the x direction. Divide the domain by this number to find the width of the cells. Repeat this process for the height of the cells Y direction.

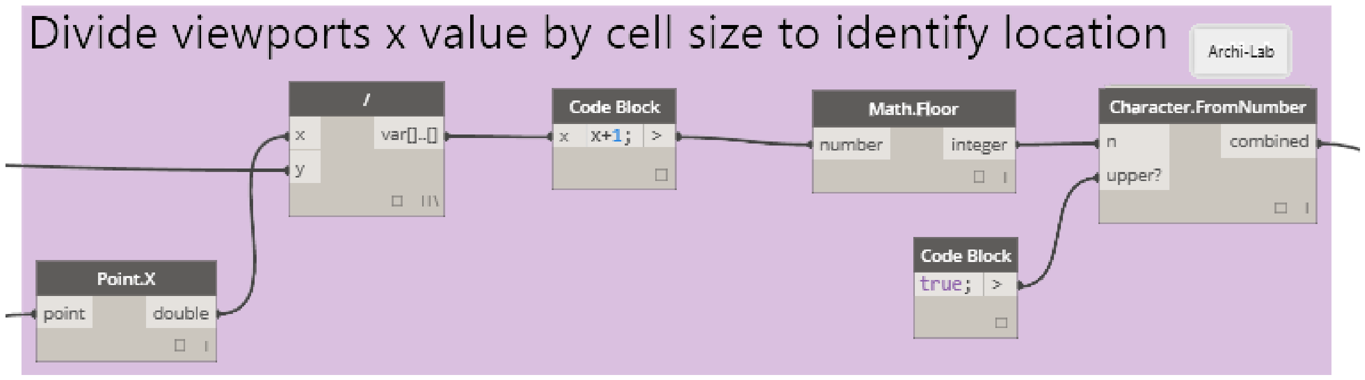




5.07 Determining what cell each view is in

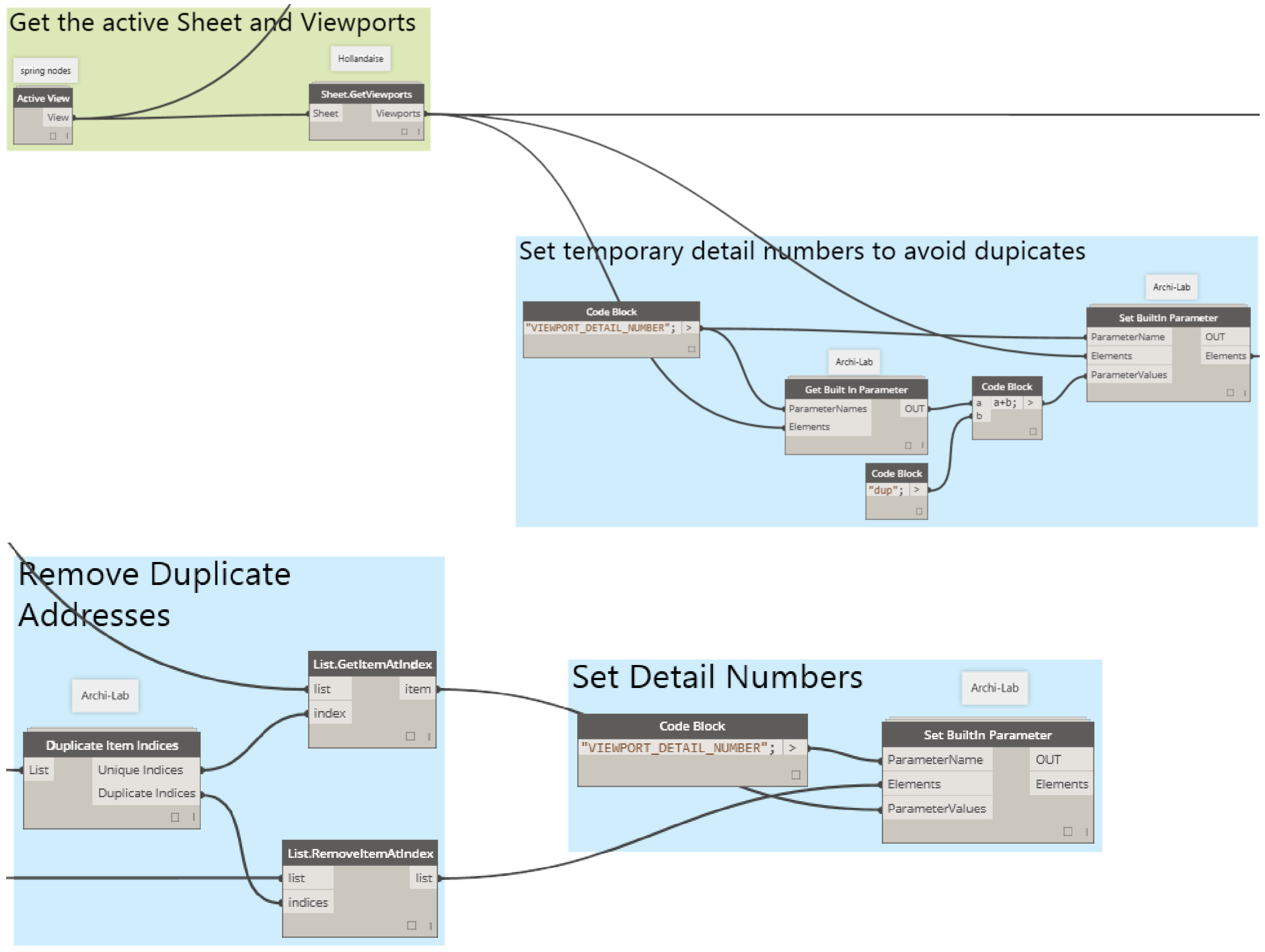
List.Map the Viewport.Pts node to get the bottom left corner point of each of the viewports on the sheet. Divide the x value of the bottom left corners by the grid size in the x direction. To identify the cell round the number down to the next lower integer. Because Dynamo starts counting from 0 and people count from 1, be sure to add 1 to that number. In the X direction the cell addresses have a letter instead of a number. Use the Character.FromNumber from Archi-Lab to convert it. Repeat the process for the Y direction (without converting the number to a letter) and then use a codeblock to add the two characters together to create the address.


Size of Each Grid Cell in X
Divide viewports x value by cell size to identify location
Archi-Lab
Code Block
Math.Floor
Character.FromNumber
Point.X
Number of Cells
Create Cell Address
Size of Each Grid Cell in Y
Divide viewports x value by cell size to identify location
Point.Y
Get Viewport Locations on Sheet
List.Map
Viewport.Pts



5.08 Changing the viewport detail numbers

Setting the detail numbers is a two part process. Revit refuses to allow two views to have the same number at the same time, rather than bother with what order the views are renumbered in this example changes all of the views to numbers that can't possibly conflict by adding the letters "dup" to the end of their current number. After the rest of the graph determines their new numbers a second check has to be run to make sure there aren't any duplicates, ie two views in the same cell. The solution this example uses is to exclude changing their names the second time which means they will have the dup extension and can easily be filtered out and corrected through another process.



```
"VIEWPORT_DETAIL_NUMBER";
```

```
a+b;
```

```
"dup";
```

```
"VIEWPORT_DETAIL_NUMBER";
```