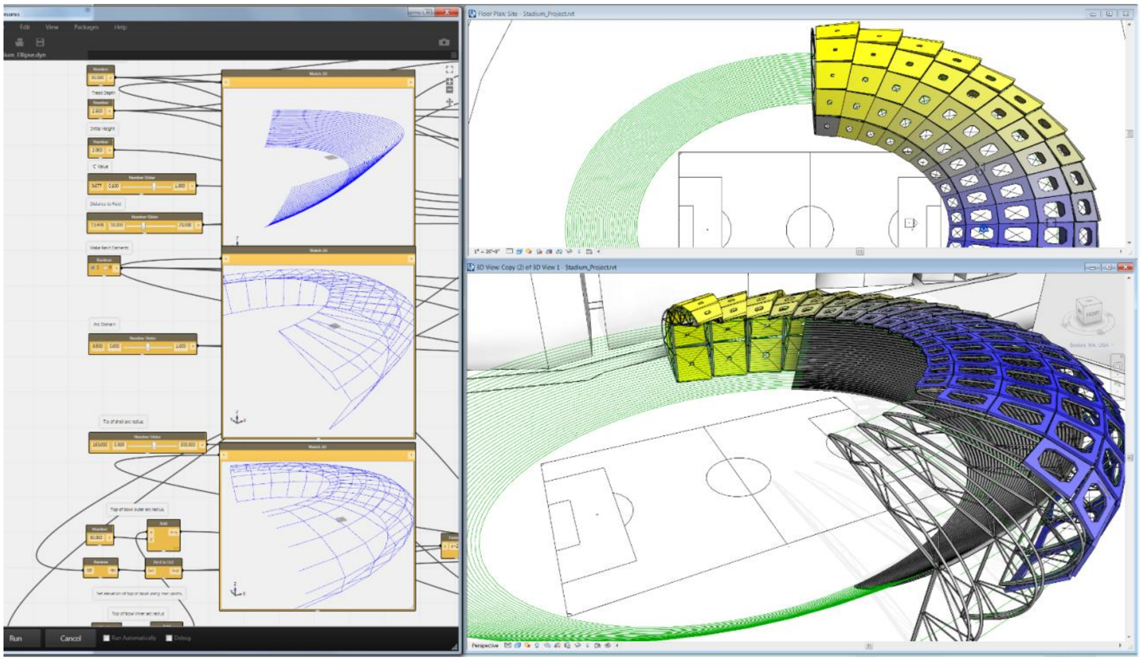
Dynamo: Visual Programming for Design



Contents

Description

These tutorial will demonstrate how to use Dynamo Visual Programming for Autodesk® Revit® software and Autodesk Vasari. The lab will provide users with resources and step-by-step examples for automating geometry creation, adjusting family parameters using external data, and sharing information with different design platforms.

Dynamo Visual Programming

Computational Design refers to the ability to link creative problem solving with powerful and novel computational algorithms to automate, simulate, script, parameterize, and generate design solutions. Computational Design has had a profound impact on Architectural practice in recent years. Design practices, large and small, have begun to invest in new computational capabilities that allow them to customize their process and pursue new, innovative design agendas. Computation might be leveraged for a variety of tasks such as automating a redundant production process or to construct an expressive form-generator. Regardless of the end-use, what is clear is that designers need frameworks that let them construct their own tools.

“Visual Programming Language” is a concept that provides designers with the means for constructing programmatic relationships using a graphical user interfaces. Rather than writing ‘code’ from scratch, the user is able to assemble custom relationships by connecting pre-packaged nodes together to make a custom algorithm. This means that a designer can leverage computational concepts, without the need to write code. Dynamo is an open source Add-in for Autodesk Vasari and Revit. Dynamo allows designers to design custom computational design and automation processes through a node-based Visual Programming interface. Users are given capabilities for sophisticated data manipulation, relational structures, and geometric control that is not possible using a conventional modelling interface. In addition, Dynamo gives the designer the added advantage of being able to leverage computational design workflows within the context of a BIM environment. The designer is able to construct custom systems to control Vasari Families and Parameters

Dynamo exposes a fundamentally new way of working with geometric information within Autodesk Vasari and Revit. Users can create control frameworks for creating, positioning, and visualizing geometry. The Visual Programming framework lets the user create unique systems and relationships and expand how BIM can be used to drive design ideation.

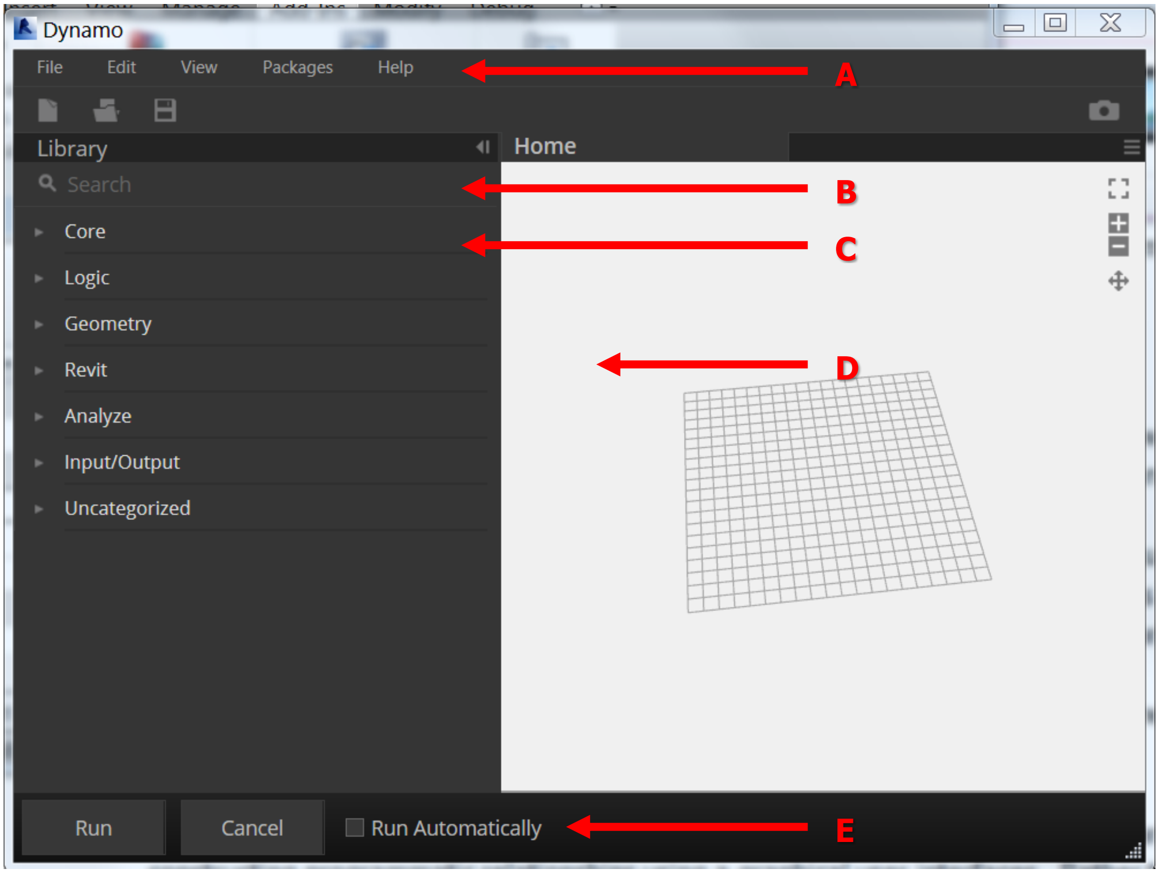
Getting Around in Dynamo

The Basics

Dynamo is primarily a plug-in for Autodesk Revit and Vasari. It works in Revit 2013, 2014 and Vasari Beta 3, and this tutorial requires that you have one of these applications installed. Dynamo can also run as a stand-alone application with all the list and logic functionality, and with some experimental geometry tools available using the Autodesk Shape Manager kernel. Work is also being done to port Dynamo functionality to other platforms.

Dynamo is open source under the Apache 2.0 lisence. The software can be downloaded from http://dynamoBIM.org, and source code is available at https://github.com/ikeough/Dynamo.

The Dynamo Interface



A. Pulldown Menus

Use the File pulldown to Open dynamo files, make new ones, Save-As a new file name, and export an image of your current workspace. Use edit to do copy/paste operations, create new custom nodes, and add comments. Use the View pulldown to activate background previews, view the console (log), and change wire appearance.

B. Search Bar

Use the Search Bar to find loaded Nodes

C. Node Library

Browser for picking nodes. Click on a node title to add to the Dynamo workspace

D. Workspace

The Workspace is the main environment for creation of Dynamo visual programs. The Home Workspace is the default workspace. When a user creates or edits a custom node, it will appear as a new tab.

E. Execution Bar

The Execution Bar allows the user to run or execute the current workspace
The "Run Automatically" checkbox will cause the workspace to run if the user changes the workspace or any of the watched Revit Elements in Revit or Vasari The "Debug" checkbox will invoke a more detailed method of execution

Concepts, Definitions, Terminology

Workspace

The active **Workspace** is the area where you interact with the elements of your visual program. When you start Dynamo you are in a blank **Home Workspace**. You start creating your visual program here by placing **Nodes** and connecting with **Wires**.

You can save the Workspace as a **.dyn**, or Dynamo file for later reuse. Dynamo files can be opened from the File menu or the **Samples** menu. Opening a Dynamo file clears the previous workspace and opens the file as the new active Workspace.
A Workspace is executed when you press the Run button or will execute automatically if you activate the "Run Automatically" check box
By default, the background of the workspace displays all geometric output of the graph. This background can be turned off in the view menu. To navigate the background, press Cntrl-G, to return to graph navigation, press Cntrl-G again.

Nodes

Nodes are the objects you place and connect together with **Wires** to form a visual program.
Nodes can represent Revit Elements like Model Lines or Reference Points.
Nodes can also represent operations like Math Functions.
Nodes have inputs and outputs.
The colors of Nodes change to indicate state.

a. Orange Nodes are well-connected and have all of their inputs successfully connected. They are part of the active Program Flow.

b. Grey Nodes indicate one of 2 states. Generally it means they are inactive and need to be connected with Wires to be part of the Program Flow in the active Workspace. Dynamo also allows for users to pass only partially connected nodes to downstream functionality, allowing for more complex interactions. This state is also represented in grey

c. Red Nodes are in an Error state. You can see what the error is by hovering your mouse over the node and reading the tooltip.

d. Currently selected Nodes have an aqua highlight. You can drag selected nodes around the workspace or right-click to see their properties.

e. Nodes have input **Ports** on the left side and output Ports on the right side. Directionality of execution and program flow usually goes left to right.

Wires

Wires connect between Nodes to create relationships and establish a program flow. You can think of them literally as electrical wires that carry pulses of information from one object to the next.

Wires connect the output Port from one Node to the input Port of another node.
You create a Wire using the mouse left-clicking on an output Port and dragging with the mouse button held down, then connect to the input port of another node. Wires appear as dashed while being dragged and solid lines when successfully connected. To disconnect a Wire, left-click on the output Node and pull the Wire away.

Ports

Ports are the light rectangular areas on Nodes, they are the receptors for Wires. Information flows through the Ports from left to right. a. Inputs Ports are on the left side of the Node. b. Outputs Ports are on the right side of the Node.

Ports are expecting to receive certain types of data, for example a Node might work on Point objects or Line objects. Try to connect like outputs to like inputs (XYZ->XYZ for example). Passing a Line object into the input Port of a Node that is expecting a Point will result in an error.

You can tell what a specific Node is expecting to receive or to return from the text label. If the name is cut off, simply hover over the port and a tooltip will appear.

Program Flow

Workspaces have a Program Flow that represents where to start program execution, what to do in the middle and how to know when program execution is complete. When you press “Run” Dynamo executes the visual program according to the established Program Flow.

Directionality of Execution

Typically Dynamo visual programs are executed from left to right and you can read the program that way to understand the Program Flow.

There are some exceptions as you get into advanced topics like recursion, but for the purposes of this class simply read from left to right.

Custom Nodes

You can create your own reusable Nodes in Dynamo without programming. These Nodes can be used in the current Workspace or in other Workspaces on your machine. You can also share these Nodes with others. Create new nodes from File>New Custom Node or by selecting existing nodes in the workspace and in Edit>Create Node From Selection (or right click in the canvas). Give the node a name and a category (selecting from the pulldown or by entering a new name. Notice the background color changes when you are editing a user-created node.
Custom Nodes appear in the **Node List** like the other nodes. You can double click to edit these nodes at any time or by selecting their name from the View menu pulldown.

Custom Nodes are graphically distinguished from other Nodes by looking like a stack of nodes. When you see a node with a dotted shadow underneath, it is an indication you can double-click on it to edit the contents.

You can share nodes you create with other colleagues by using Package Manager tools. By selecting a custom node, and going to the Package Manager in the dropdown menu, you can publish the node for others to use. Similarly, you can search the online catalogue for functionality published by other users.

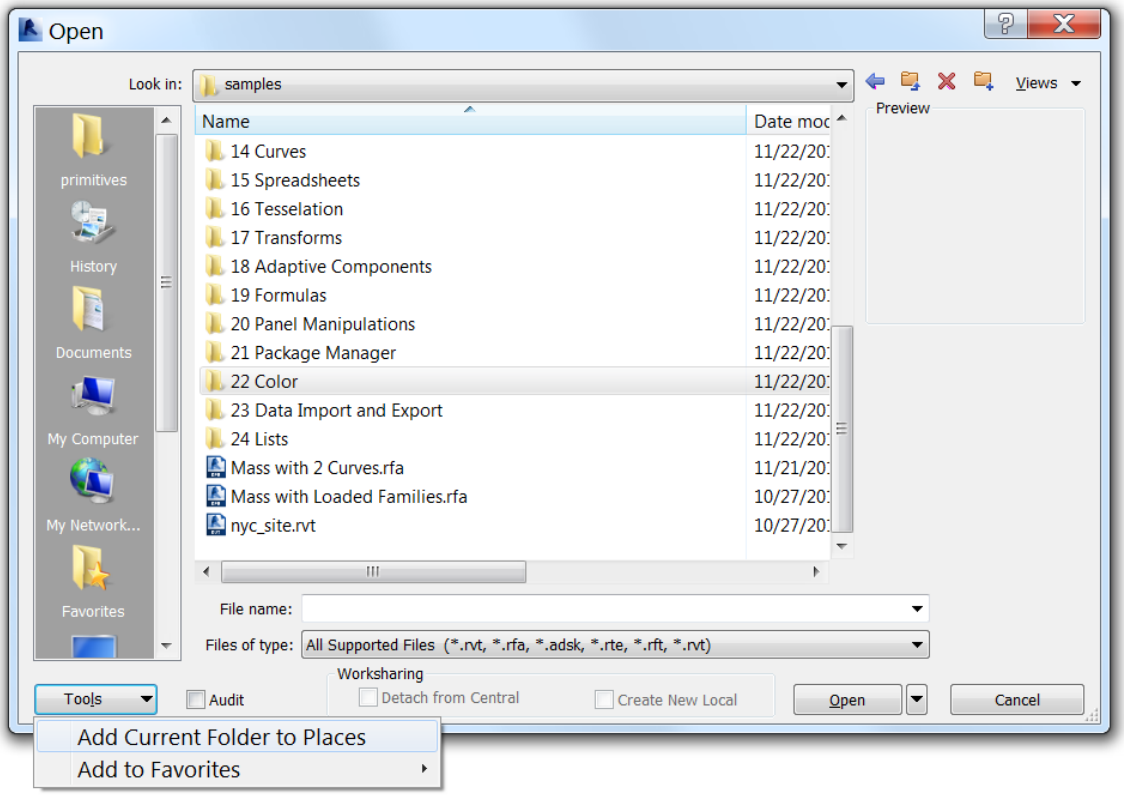
Custom Nodes can be nested inside of themselves to create recursive functionality, such as Fibonacci sequences or fractals.

Workflow The active environment that will be executed. The aggregation of the active workspace, python scripts and all dependent user-created nodes.

Examples

The following workflows use Dynamo files that are located in the Help menu of the application under Samples. Many have associated Revit family (rfa) and project (rvt) files that, by default, are located at C:\Autodesk\Dynamo\Core\samples. All the examples in this document can be used in either Revit or Vasari.

For ease of opening rfa and rvt files, you might want to add this folder to your Places open dialog.



Look in: samples

Name	Date moc
14 Curves	11/22/20
15 Spreadsheets	11/22/20
16 Tesselation	11/22/20
17 Transforms	11/22/20
18 Adaptive Components	11/22/20
19 Formulas	11/22/20
20 Panel Manipulations	11/22/20
21 Package Manager	11/22/20
22 Color	11/22/20
23 Data Import and Export	11/22/20
24 Lists	11/22/20
Mass with 2 Curves.rfa	11/21/20
Mass with Loaded Families.rfa	10/27/20
nyc_site.rvt	10/27/20

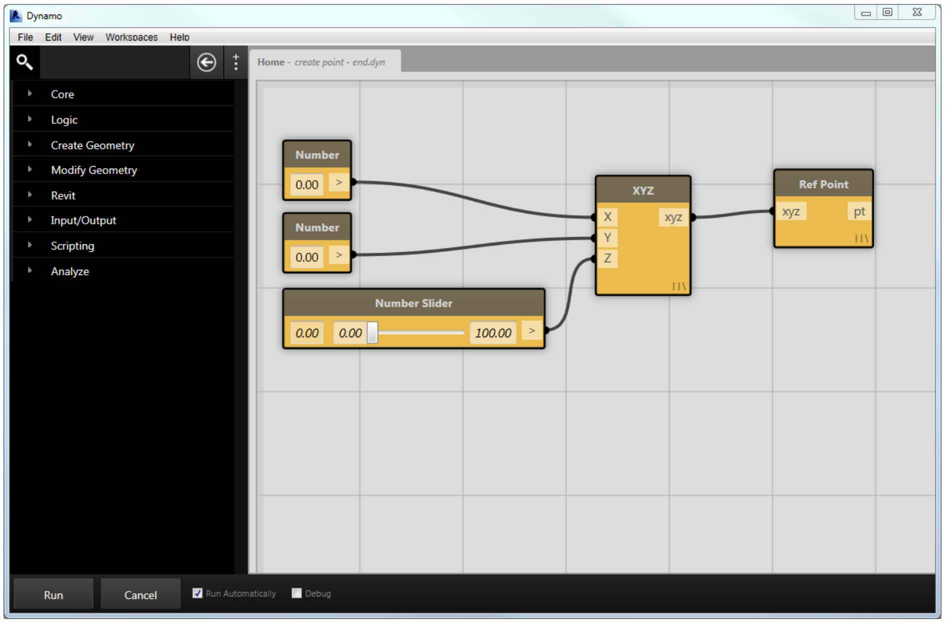
File name:

Files of type: All Supported Files (*.rvt, *.rfa, *.adsk, *.rte, *.rft, *.rvt)

Create a Point, or “Hello World!” In Dynamo

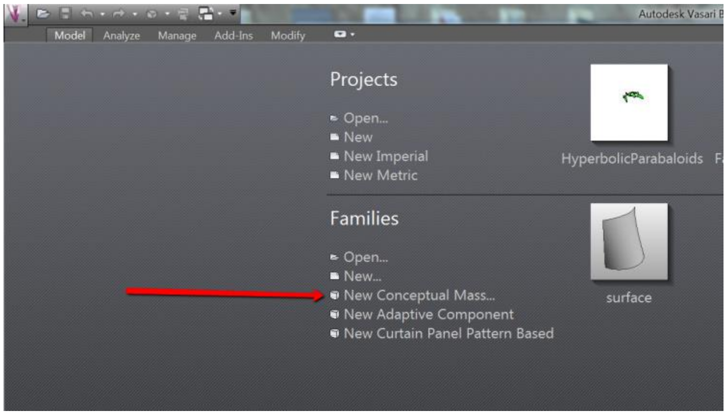
Creating a reference point is the most basic operation you could want to do in Dynamo. It’s the venerable “Hello World!” for the application. To get there, you will need to learn some high level concepts and understand some of the basic principles of working in the Revit and Vasari environment

- Learn how to launch Dynamo in the right environment for your needs
- Tour the User Interface, Understand search/browse, and navigate the Dynamo Workspace
- Place nodes and wire them together
- Learn about the difference between geometry and Revit Elements

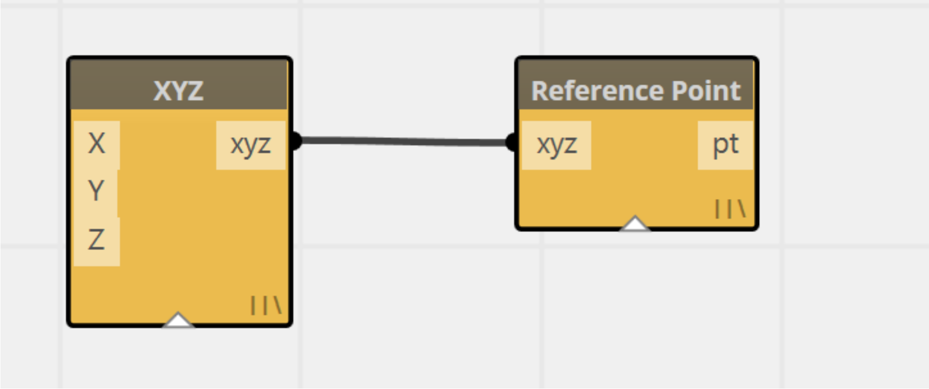


A workspace for creating a dynamically controlled Reference point

1. Launch Vasari.
2. We are going to be placing Reference Points, which can only be done in the Mass, Curtain Panel by Pattern, or Adaptive Component family environments (not the “project” or .rvt environment). Click > New > Family> Mass.rft. Or, in the recent documents screen, under Families, Click on “New Conceptual Mass”.

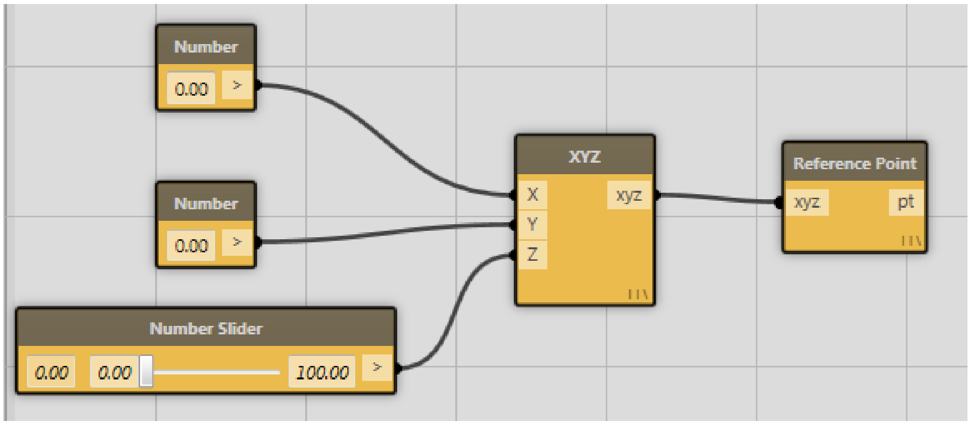


3. Dynamo will operate on the .rfa or .rvt file that is active at the time Dynamo is launched. Now that we have a Mass file open, go to Add-ins tab and launch Dynamo
4. From the Dynamo File Menu, go to **File/Samples/ 1. Create Point / create point.dyn**

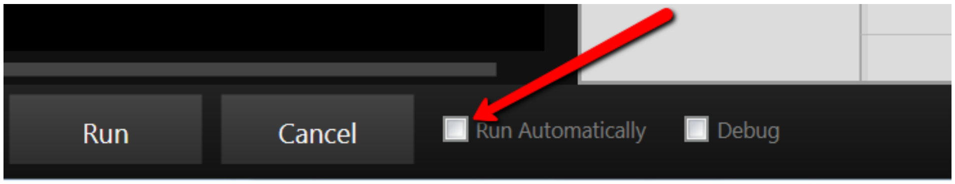


5. Notice a couple of nodes (**XYZ** and **Ref Point**) in the workspace. **Run** to create a single Reference Point at 0,0,0.
6. There is a difference between an XYZ and a Reference Point. An XYZ is a coordinate point in space, while a Reference Point is a full-fledged Revit Element with many aspects and associated meta-data.
   Abstract geometry, like this XYZ, is displayed in the background of the Dynamo workspace. You can toggle between navigating the background 3d space and the flat graph by pressing Control-G. You can also turn this preview off in the View menu> Background 3d Preview.
7. Select and move nodes by using the left mouse button.
   a. Type Delete in order to delete a node or right click and click Delete.

   b. The right click menu will also show a number of other functionalities. Click on the help button to see more information on a selected node
   c. Select all the nodes and right click to set their alignment.
   d. Zoom in and out using the mouse wheel and pan using middle click and hold
8. Now we will learn how to re-wire the workspace to add more inputs:
   a. Type "Number" into the search bar to find the **Number** node to add it into the Workspace. This can be done by either typing enter with the number node selected in Search or clicking the node in the node lib
   b. Find the **Number Slider** node and add that as well
   c. On the **XYZ** node, select the end of the wire connecting to the **Y port**. Drag it off into space to disconnect. Do the same for the **Z port**.
   d. Connect the new **Number** node to the **Y port** and the **Number Slider** node to the **Z port**.



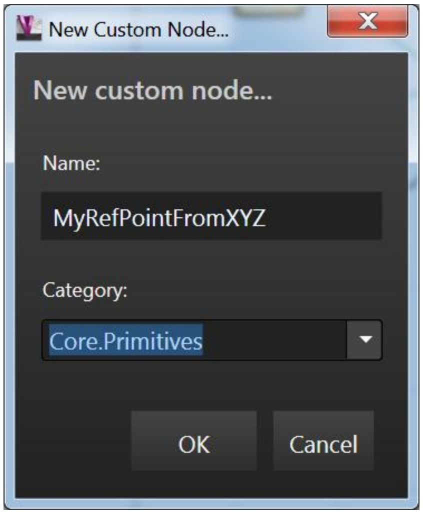
   e. At the bottom of the canvas, check "Run Automatically".



   f. Move the slider to see the point move around
   g. In the Vasari toolbar, pick Model> Draw>Spline Through Points, and draw a spline with one of the points using the Dynamo created Reference Point. Move the Number Slider and see both Dynamo created stuff and manually created stuff update.

   h. Select the Dynamo created Reference Point and, in Dynamo, right click in the canvas and pick "Find Nodes from selected elements"

9. Create a Custom Node by Selecting the XYZ and Reference Point nodes, then right click out in the canvas, and pick New Node from Selection. Name your custom node something meaningful.

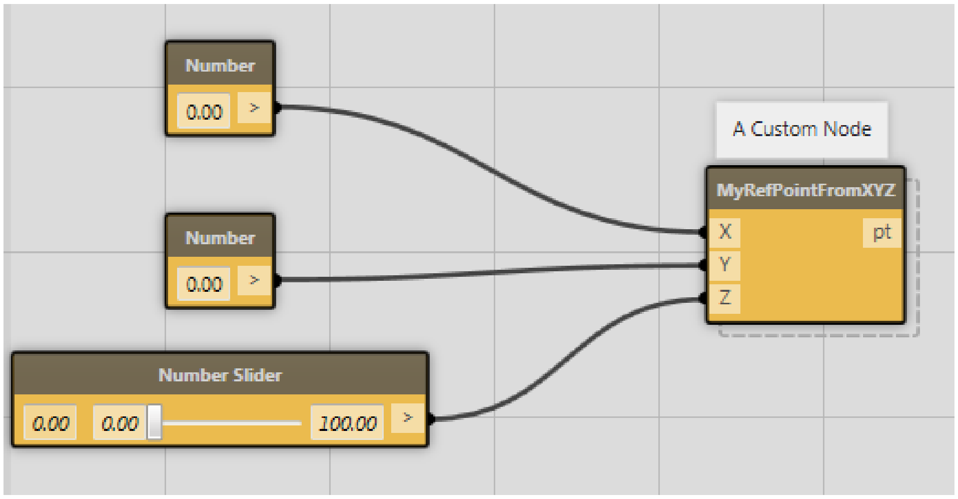


New custom node...

Name: MyRefPointFromXYZ

Category: Core.Primitives

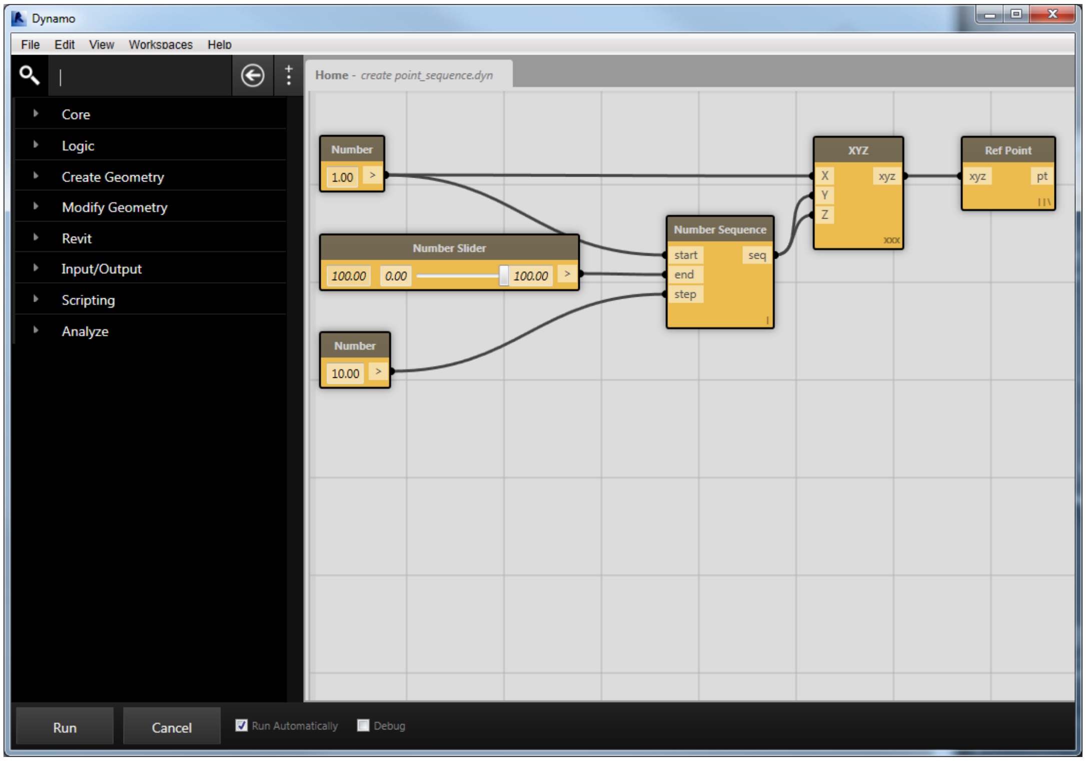
10. In the Edit Menu, pick Create Note to annotate your workflow (or type Cntrl-W). Double click on the Note to edit it (or edit from the right click menu)



Placing Families with Sequences, Ranges, Lines and Grids

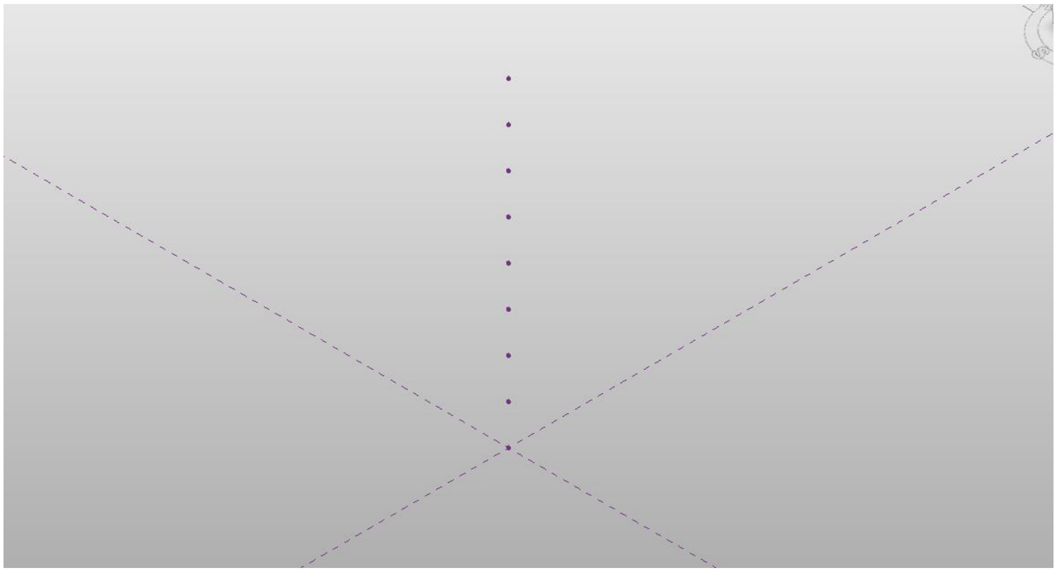
This tutorial aims to introduce the following:

- How to use node **Lacing** to evaluate the members of a list in different ways
- Generate **lines, grids, and lattices** of Reference Points
- **Place family instances** with Dynamo



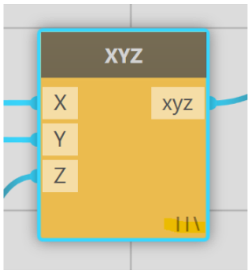
Dynamo after opening the create point_sequence.dyn file

1. Close any other open Vasari files
2. From Vasari, Click > 🗁 (Open).
3. Navigate to **C:\Autodesk\Dynamo\Core\samples**
4. Open **Mass with Loaded Families.rfa** from the **Samples** directory
5. From the Dynamo Help menu, go to **Samples/ 1. Create Point / create point_sequence.dyn**
6. Hit the "Run" button to see the following:



A sequence of points created by running this example

7. Right click on the **Number Sequence** and select **Help…** to see what kinds of inputs and outputs the number sequence expects.
8. Notice the little icon in the bottom right corner of the XYZ node. This indicates the *Lacing* for this particular node. Lacing allows you to automatically apply the node to the sequence created by the Number Sequence node.

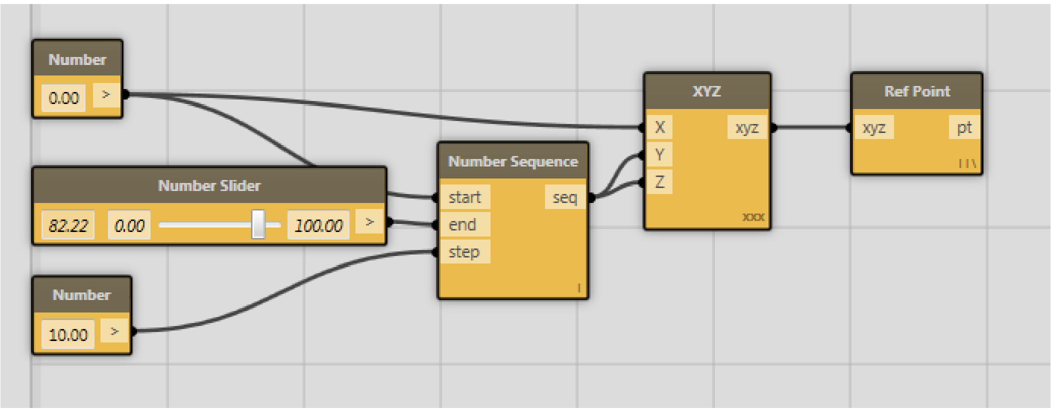


The XYZ node with lacing set to "Longest"

9. Try changing the Lacing strategy to **First** by right clicking on the XYZ node and selecting First. You should see the icon in the bottom right corner of the node change.

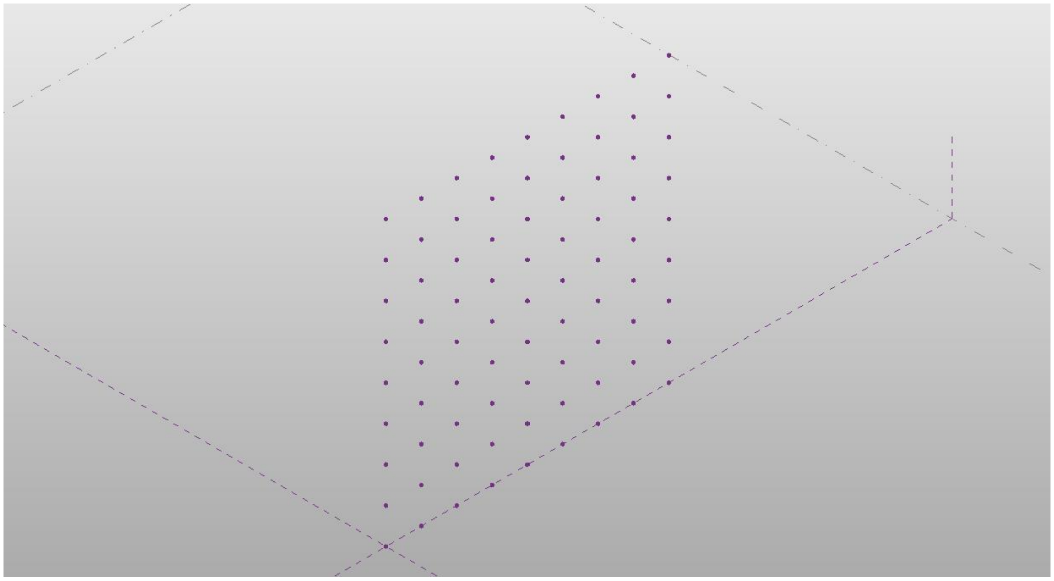
10. Hit Run again. You should see a single point located at the origin. This is because the XYZ node is only evaluating the first element in the list created by the Number Sequence node.

11. Change the lacing on the XYZ node to **Cross Product**. If you hit Run again, you should again see a vertical line of regularly spaced points.
12. Click the output port button on the Number Sequence node and connect it to the XYZ node by clicking on the Y input. Your workspace should look like this:



A workspace for creating a grid of XYZ points in the YZ plane

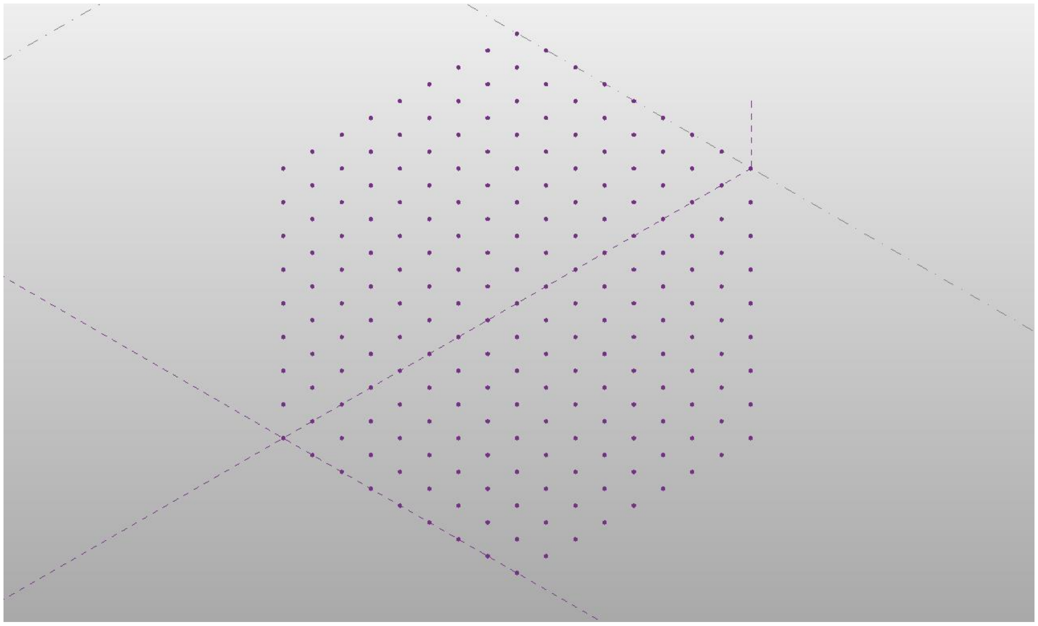
13. By running again, you should see an orthogonal grid in the YZ plane, like this:



A square grid of XYZ points.

You can experiment by connecting the sequence to the X and Y ports to get a plane in XY plane.

14. Connect all 3 input ports of the XYZ node to the output of the Number Sequence Node and Hit Run. You should get a cubic 3d lattice:

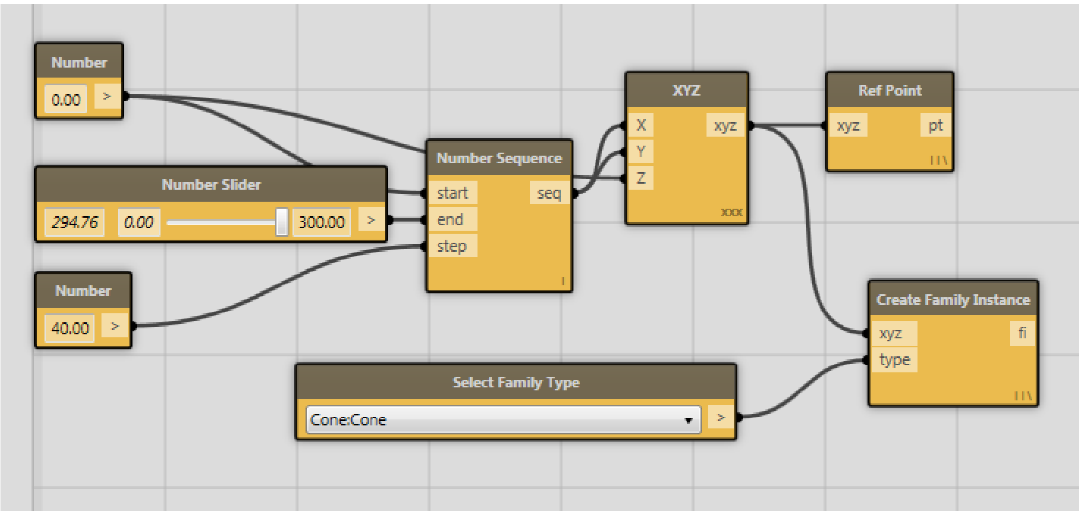


A cubic lattice of XYZ points.

15. Let's scale back a bit and go back by connecting the Number node, set to 0, to the Z port of the XYZ node. We'll **have just a grid in the XY plane**.

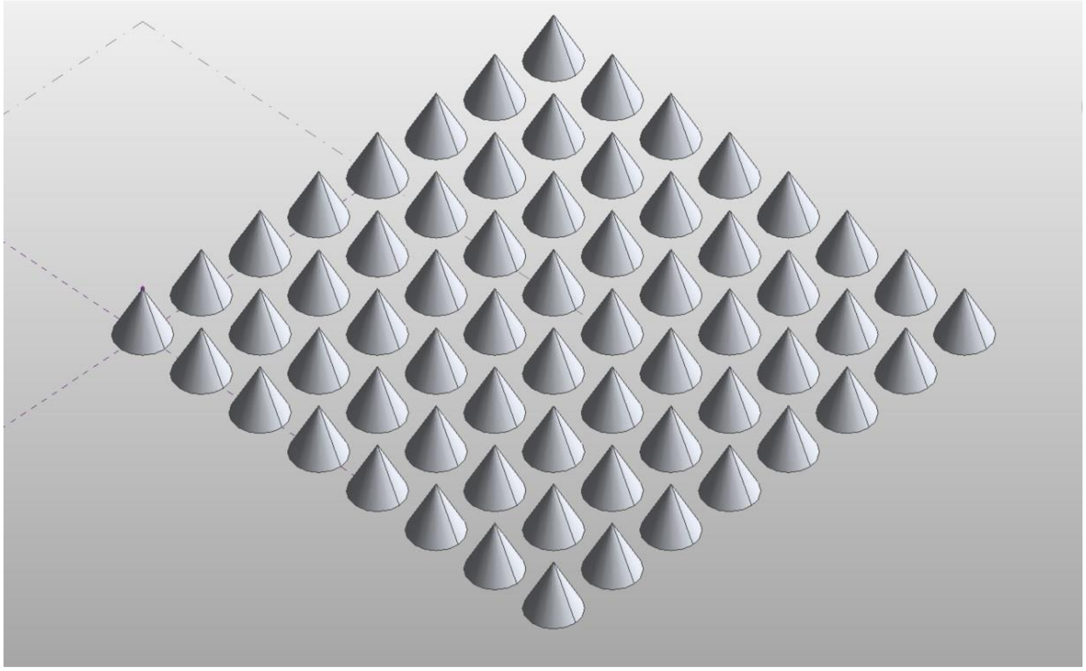
Now we will extend this workspace to do something more useful than just creating points. We'll place Family Instances!

16. Go to the **Search Bar** and type in **"Family",** this filters the available nodes down and allows easier access from the **Node List.** You should now only see nodes related to "Family".
17. Add one **Create Family Instance** node and one **Select Family Type** node into the workspace. Look at the Help menu for both of these nodes by right clicking.
18. Connect the **Select Family Type** output to the **Create Family Instance** "typ" input port.
19. Now select the **Cone** Family Type from the pulldown on Select Family Type and connect the XYZ output from the **XYZ Node** to the XYZ input of the **Create Family Instance** node. Your workspace should look like this:



The workspace after adding the Select Family Type and Create Family Instance nodes

20. Hit **Run** and you should see something like this:



A grid of cone families laid out on the XY plane.

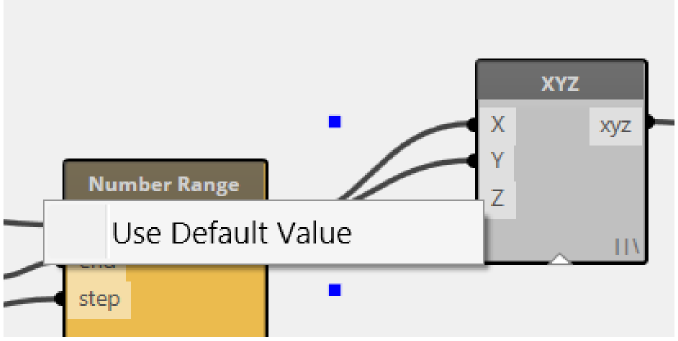
21. Experiment with different values for the number sliders or different family types. By turning on **Run Automatically** you can do this interactively.

Nested Lists and Basic Data Management

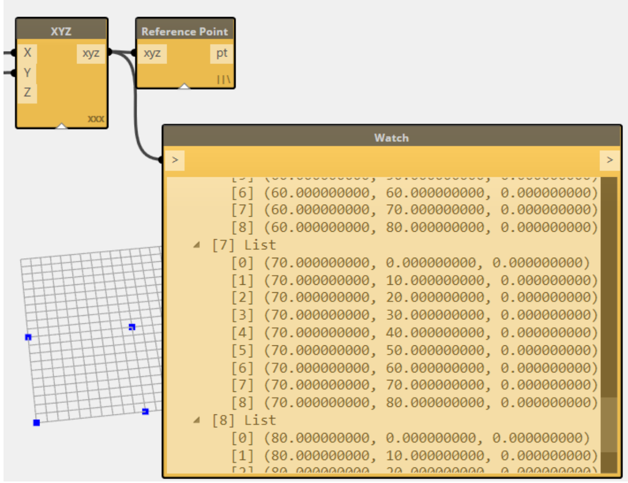
- Understand the importance of lists in any standard workflow
- Explore some of the tools for making data do what you want it to do

This tutorial gives a very brief introduction to the process of sorting data in lists. Indexed lists are the backbone of algorithmic design, and getting control of both the tools and concepts is essential for achieving even moderate complexity.

1. From Vasari, Click > (Open).
2. Navigate to **C:\Autodesk\Dynamo\Core\samples**
3. Open **Mass with Loaded Families.rfa** from the **Samples** directory
4. Go to Add-ins tab and launch Dynamo. If you already had it open, close and re-open it to re-associate Dynamo with the newly opened .rfa file.
5. From the Dynamo Help menu, go to **Samples/ 1. Create Point / create point_sequence.dyn**
6. Running the workspace will create a vertical line of xyzs, we will now adjust this to make a horizontal grid of xyz.
7. Unplug the z input for the xyz node, and wire the Number Range node into the X and Y ports of the XYZ node. Right click on the Z port of the XYZ and activate "Use Default Value":

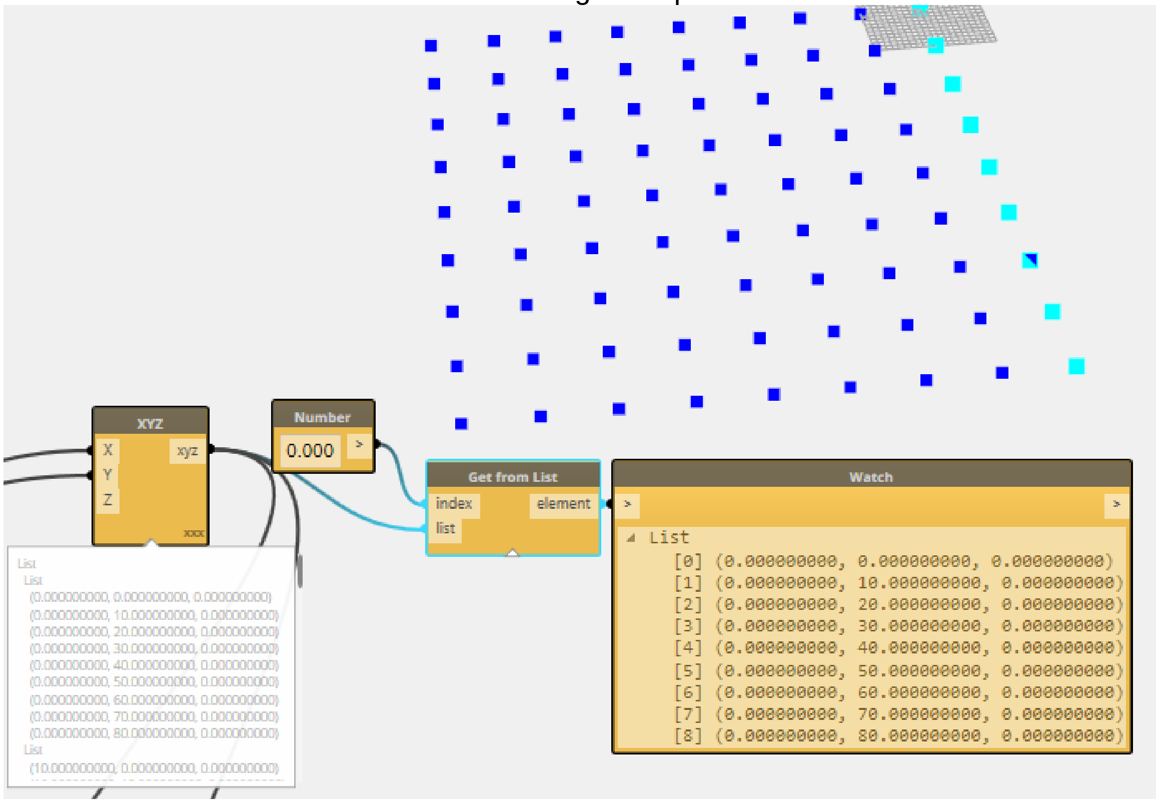


8. Right click on the XYZ Node and set Lacing to Cross Product. Run the workspace and inspect the results, a nested list or list of lists.



```
    [6] (60.000000000, 60.000000000, 0.000000000)
    [7] (60.000000000, 70.000000000, 0.000000000)
    [8] (60.000000000, 80.000000000, 0.000000000)
 ◢ [7] List
    [0] (70.000000000, 0.000000000, 0.000000000)
    [1] (70.000000000, 10.000000000, 0.000000000)
    [2] (70.000000000, 20.000000000, 0.000000000)
    [3] (70.000000000, 30.000000000, 0.000000000)
    [4] (70.000000000, 40.000000000, 0.000000000)
    [5] (70.000000000, 50.000000000, 0.000000000)
    [6] (70.000000000, 60.000000000, 0.000000000)
    [7] (70.000000000, 70.000000000, 0.000000000)
    [8] (70.000000000, 80.000000000, 0.000000000)
 ◢ [8] List
    [0] (80.000000000, 0.000000000, 0.000000000)
    [1] (80.000000000, 10.000000000, 0.000000000)
```

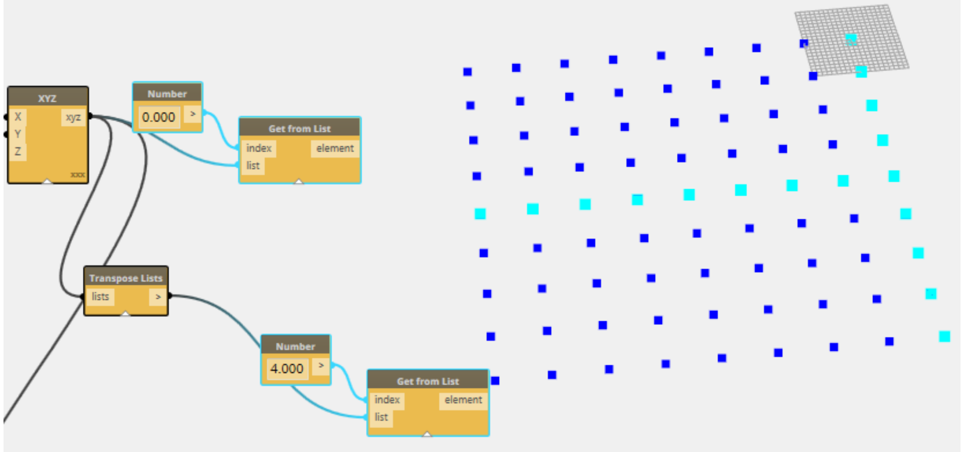
9. Now we will extract a single row of information from the nested list. In the search bar, find Get From List, or browse to the node in Core>Lists>Query. Extract the first row of data by passing the list of lists into the Get From List "List" port, and choosing the first index row by passing in a 0 number node. Pick the Get From List node to see the row isolated in the background preview.



```
List
  List
    (0.000000000, 0.000000000, 0.000000000)
    (0.000000000, 10.000000000, 0.000000000)
    (0.000000000, 20.000000000, 0.000000000)
    (0.000000000, 30.000000000, 0.000000000)
    (0.000000000, 40.000000000, 0.000000000)
    (0.000000000, 50.000000000, 0.000000000)
    (0.000000000, 60.000000000, 0.000000000)
    (0.000000000, 70.000000000, 0.000000000)
    (0.000000000, 80.000000000, 0.000000000)
  List
    (10.000000000, 0.000000000, 0.000000000)
```

```
◢ List
    [0] (0.000000000, 0.000000000, 0.000000000)
    [1] (0.000000000, 10.000000000, 0.000000000)
    [2] (0.000000000, 20.000000000, 0.000000000)
    [3] (0.000000000, 30.000000000, 0.000000000)
    [4] (0.000000000, 40.000000000, 0.000000000)
    [5] (0.000000000, 50.000000000, 0.000000000)
    [6] (0.000000000, 60.000000000, 0.000000000)
    [7] (0.000000000, 70.000000000, 0.000000000)
    [8] (0.000000000, 80.000000000, 0.000000000)
```

10. Copy/Paste your Number and Get From List nodes, and place a Transpose node in between the XYZ and Get From List. Adjust the index input. Now you have swapped rows for columns and have a different set of geometry to use.



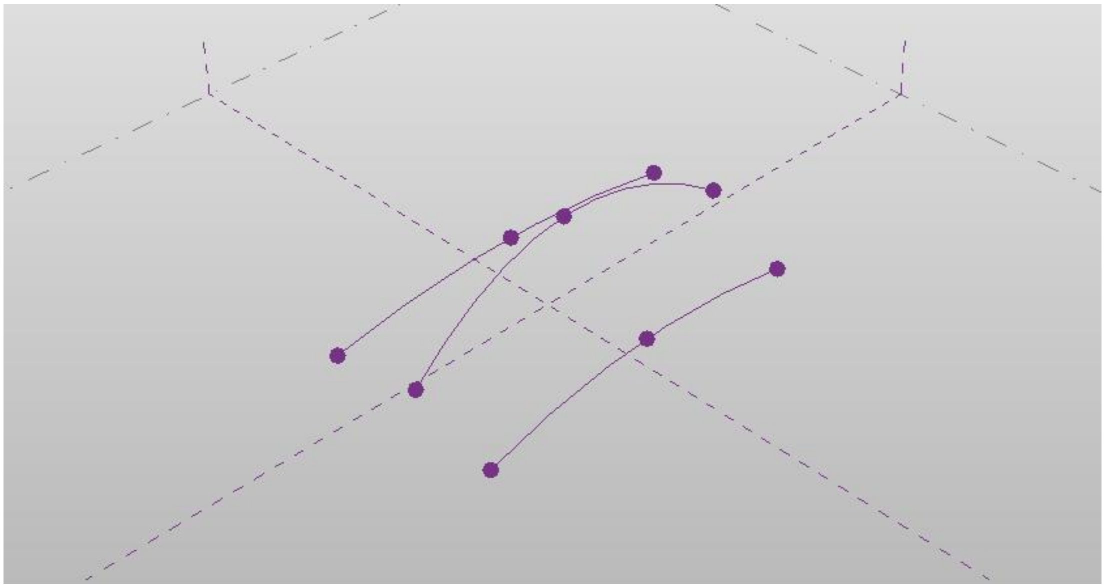
For more examples of list operations, take a look at the other files in samples > 24 Lists. There is also an extensive set of List samples for advanced data management at https://github.com/ikeough/Dynamo/tree/master/test/core/list

Advanced Family Placement: Adaptive Components

Dynamo can use geometry extracted from the Revit environment and perform lots of different operations on that geometry. That geometry can be manipulated in various ways using list operations. Finally, you can take that information to place adaptive components. In this example, you'll learn how to:

- Select and use Revit geometry in Dynamo
- Place Adaptive Components.

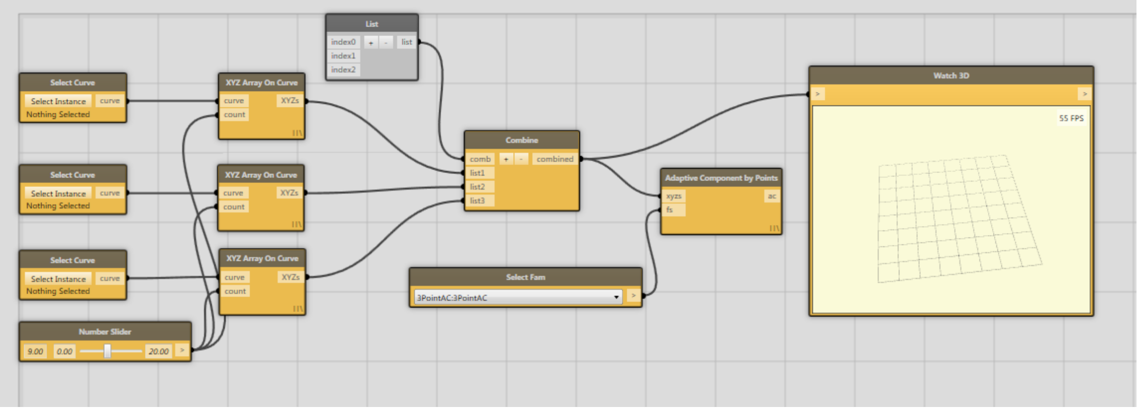
1. Close any Vasari files that you have open.
2. From Vasari, Click V > 📂 (Open).
3. Navigate to:
   C:\Autodesk\Dynamo\Core\samples\18 Adaptive Components
4. Open **Adaptive Component Placement.rfa**



The contents of Adaptive Component Placement.rfa.

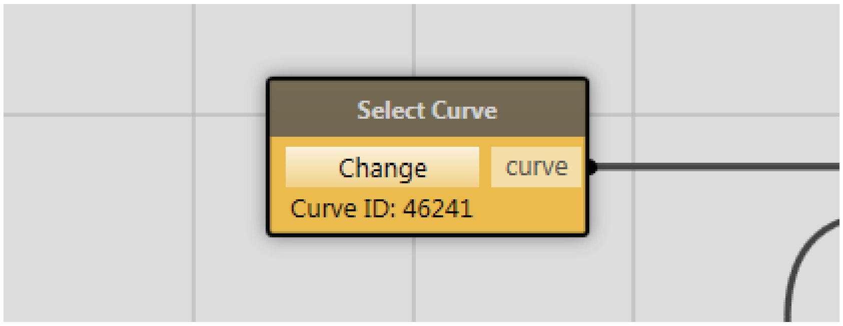
5. From the Dynamo **Help** menu, go to:

Samples/ 18 Adaptive Components / Adaptive Component Placement



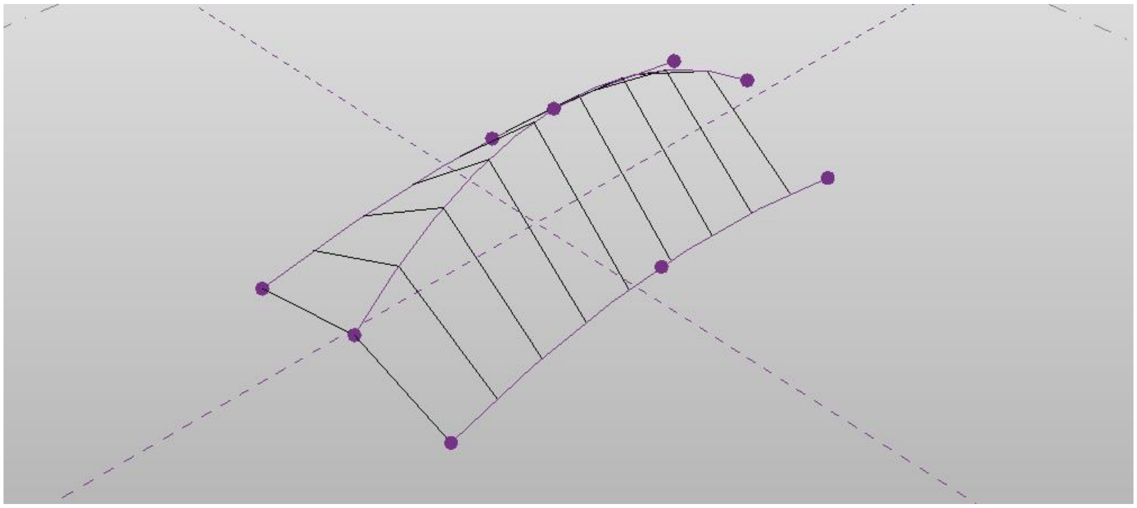
The nodes in Adaptive Component Placement.dyn

6. You'll notice that each of the Select Curve nodes has a button in it. That button can be used to select a curve inside of Revit. Arrange your Dynamo window so that you can see the Vasari canvas. Click the first Select Curve nodes "Select Instance" button and select the first curve (Leftmost on the preceding image). You should see the node change to this:



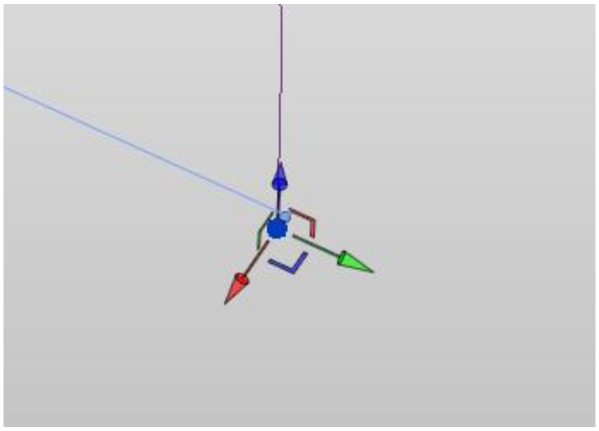
A curve has been selected

7. Do the same on the other two nodes, continuing with first with the middle curve. Now, we have selected three curves for use in Dynamo.
8. Now, hit Run. You should see the following:



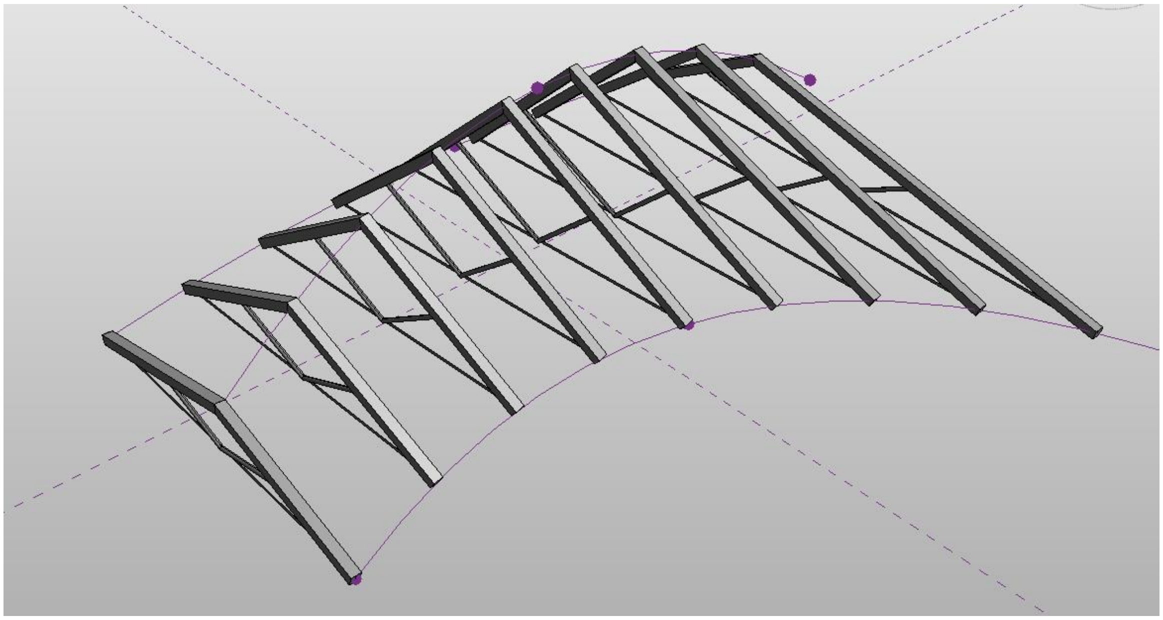
A series of three point adaptive components placed along the three curves.

9. Change the family type on the Select Family Type node to "3PointAC_wireTruss" to place a series of trusses along these three curves.
10. Select one of the reference point controlling the three curves and edit it.



Edit the reference point

11. After editing the curves, re-run the calculation in Dynamo. Note that the adaptive components have changed shape to match new location of the curve:



The adaptive components arrayed onto the three curves

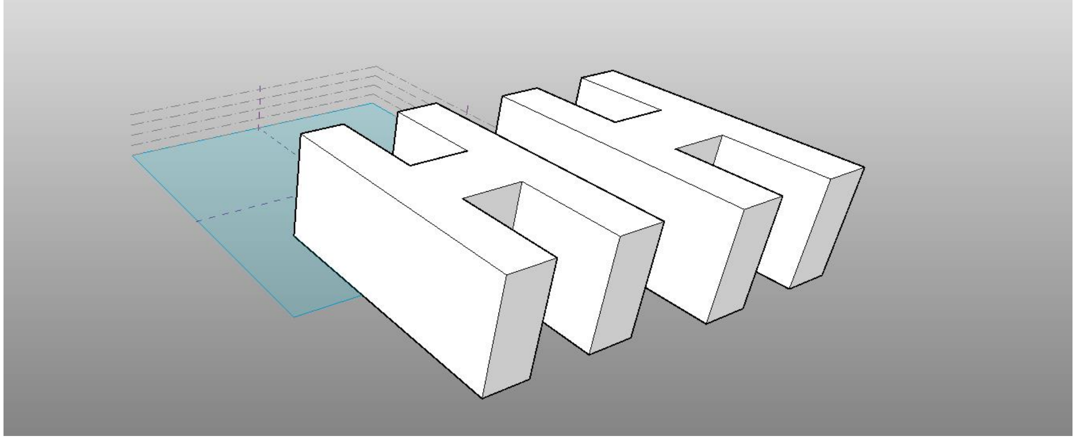
This workspace begins by taking three curves from the **Select Instance** and the **XYZ Array On Curve** node to get a regular list of XYZ's along the curve. Then, we use the **List** node to combine the XYZ's into a nested list. This list contains three lists of XYZ's, a 2D list, each one sampled from the curves we selected at the beginning. Then, we use the **Transpose Lists** node. This node returns a list of length-three lists of XYZ's. This is the result of replacing the rows with the columns in our original 2D list. This workspace gives us the three XYZ's that are necessary to place the Adaptive Component's.

Get and Set Family Instance Parameters

- Use Dynamo to select **Family Instances**
- Synchronize **Family Instance** parameters across multiple instances using **Run Automatically**

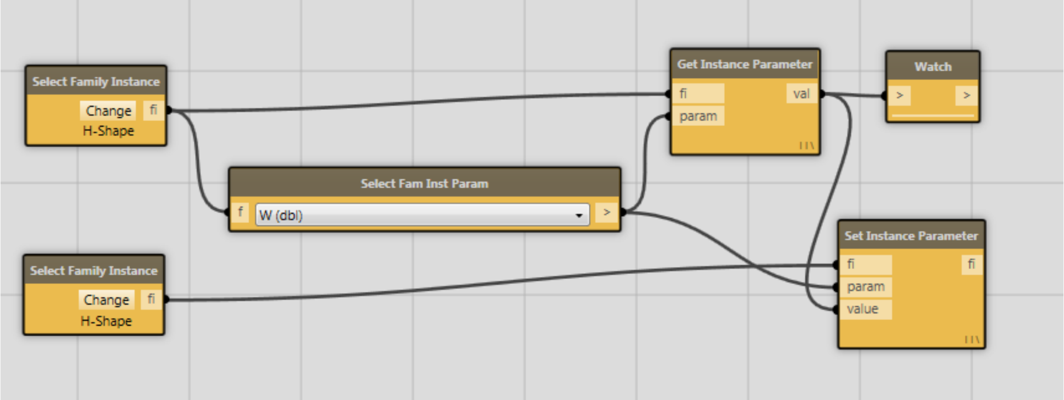
1. Close any Vasari files that you have open
2. Click > (Open).
3. Navigate to:
   C:\Autodesk\Dynamo\Core\samples\08 Get Set Family Params\ inst param mass families.rvt

   You should see the following:



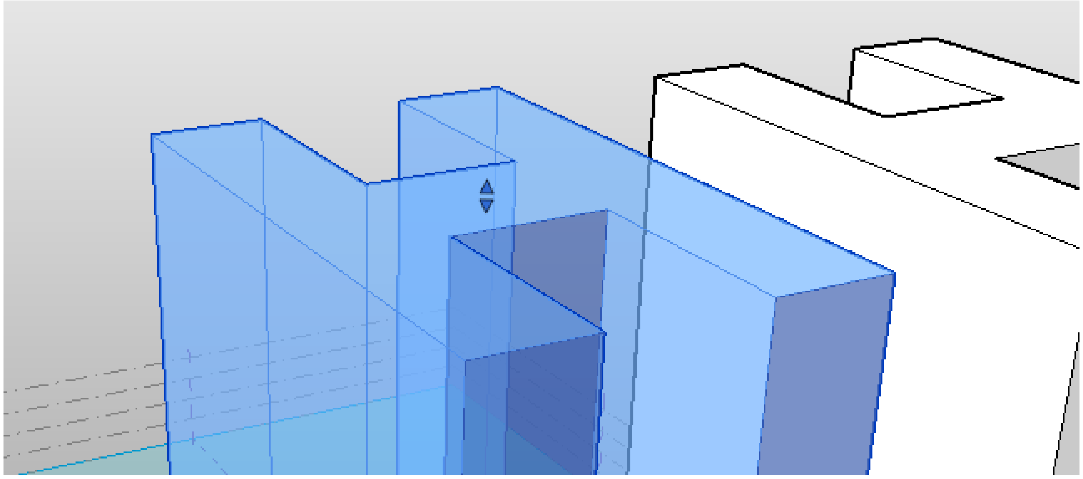
The contents of inst param mass families.rvt

4. From the Dynamo Help menu, go to:
   Samples/ 08 Get Set Family Params / inst param 2 masses driving each other
5. You should see the following in the workspace:



Dynamo after opening the create point_sequence.dyn file

6. Note that the two **Select Family Instance** nodes have automatically associated themselves with the two family instances (a pair of H-shaped building masses). This file was saved after having associated these nodes with Revit geometry, so you don't have to pick the families every time you open the file.
7. Set the **Select Family Instance Parameter** node to H (dbl).
8. Click **Run**. You should see the two instances match in height.
9. In Vasari, edit the height of the H instance closer to the origin.



The manipulator to change the height of the family instance

10. Click **Run,** again. You should see the family instances update.
11. Set Dynamo to **Run Automatically.**
12. Edit the same Family Instance's height again. You should see the other instance update.

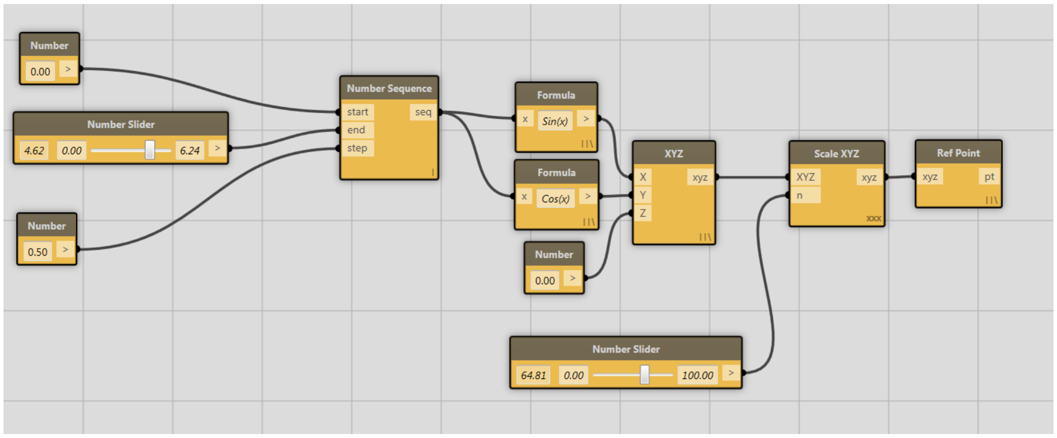
Now, you know how to synchronize Family Instance parameters!

Basic Math with the Formula Node

Dynamo has many useful tools for doing math. This tutorial aims to demonstrate how to:

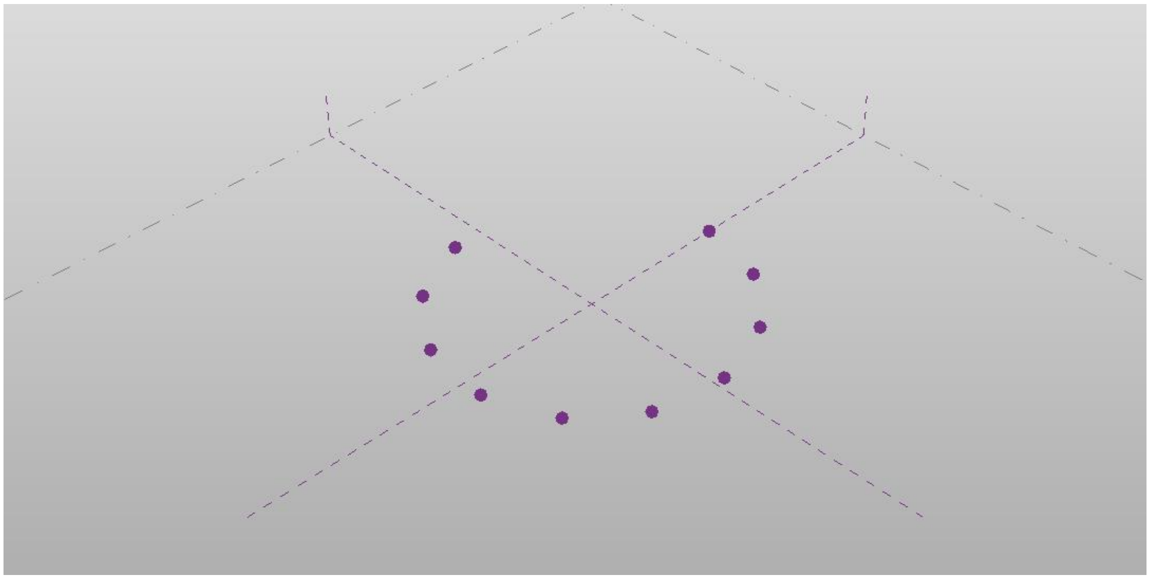
- Use the Formula node to simplify writing mathematical expressions
- Generate points that follow circular or elliptical paths from a mathematical formula.

1. Close any Vasari files that you have open
2. From Vasari, Click V > 📂 (New) > Family.
3. Use Mass.rft in the Conceptual Mass folder.
4. From the Dynamo File Menu, go to **Samples/ 19 Formulas / Scalable Circle.** You should see the following in your workspace:



The nodes in Scalable Circle.dyn

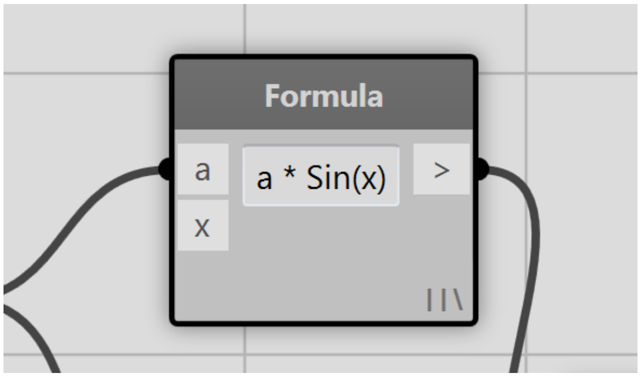
5. To get an idea of what this workspace does, hit **Run.** You should see an arc of XYZ points centered at the origin:



The result of running Scalable Circle.dyn

We're going to edit this workspace to demonstrate a few concepts of the **Formula node**.

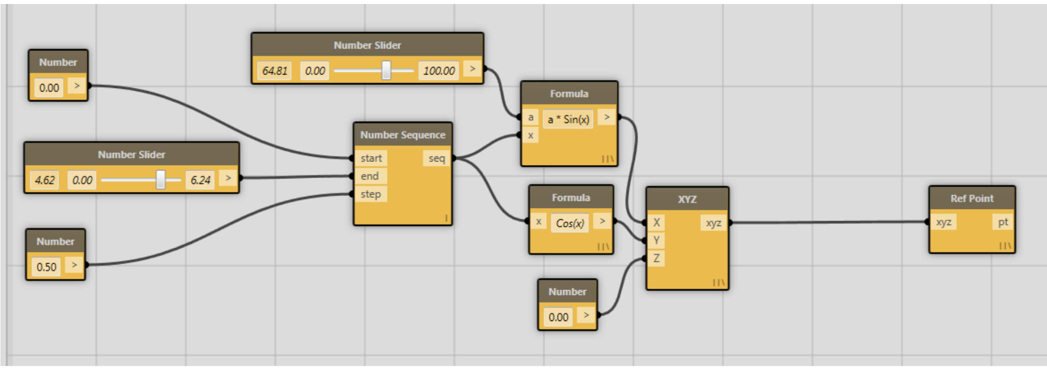
6. First, select and delete the Scale XYZ node. You should see the XYZ and Ref Point node turn grey.
7. Let's edit the formula in the first Formula node, which currently contains "Sin(x)". Change the formula so it says "a * Sin(x)" and hit enter. You should see a new port on the Formula node called (appropriately) "a". You could've called this variable anything. It should look like this:



The Formula node after editing the expression

8. Connect the output of the Number Slider node to the "a" input of your freshly edited Formula node and reconnect the output of the Number Sequence node

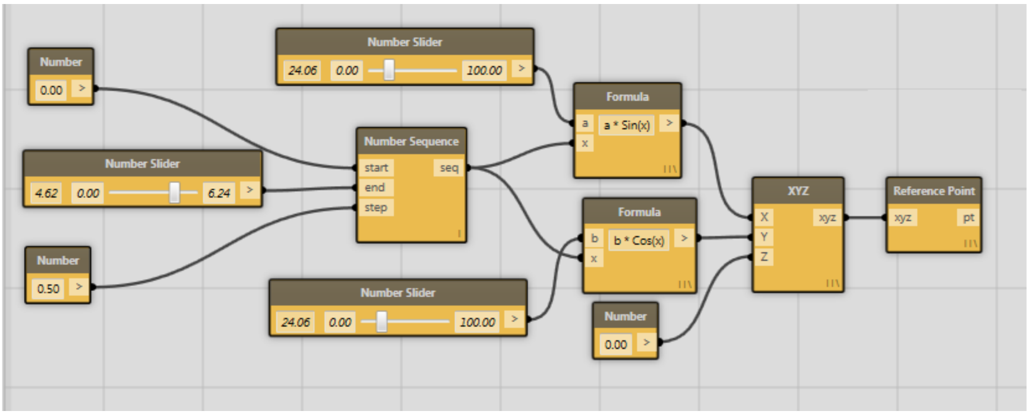
to the x input of the same node. Connect the XYZ output again to the Reference Point node. Your workspace should look like this:



The new configuration of the workspace after editing the first Formula node.

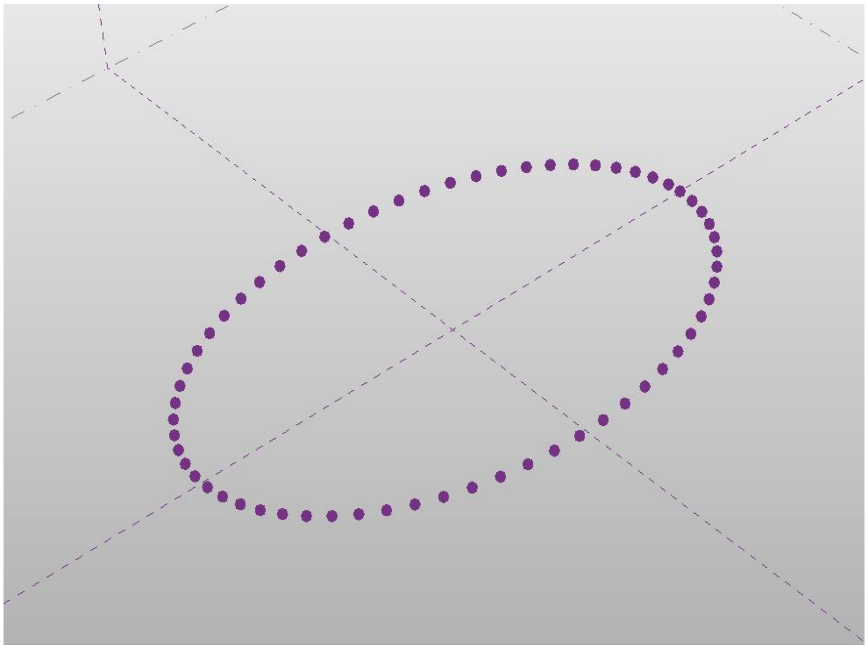
9. Hit **Run**. You should see your circle of points disappear and a very narrow ellipse will replace it. Play with the sliders to change it.

10. Let's do the same thing to the other formula node which currently has "Cos(x)" in it. That is, let's add a multiplier to the second Formula node and connect a number slider to that. Now, your workspace should look like this:



The new configuration of the workspace after editing both Formula nodes.

11. Now, turn on Run Automatically and experiment!



An ellipse created using two Formula nodes.

Note: The formula node is based on the open source **NCalc** library. It has an amazing amount of features – it includes many mathematical operators, functions, and you can even pass your own functions into the node! For a full description of operators, functions, and more check out**:** **http://ncalc.codeplex.com/**

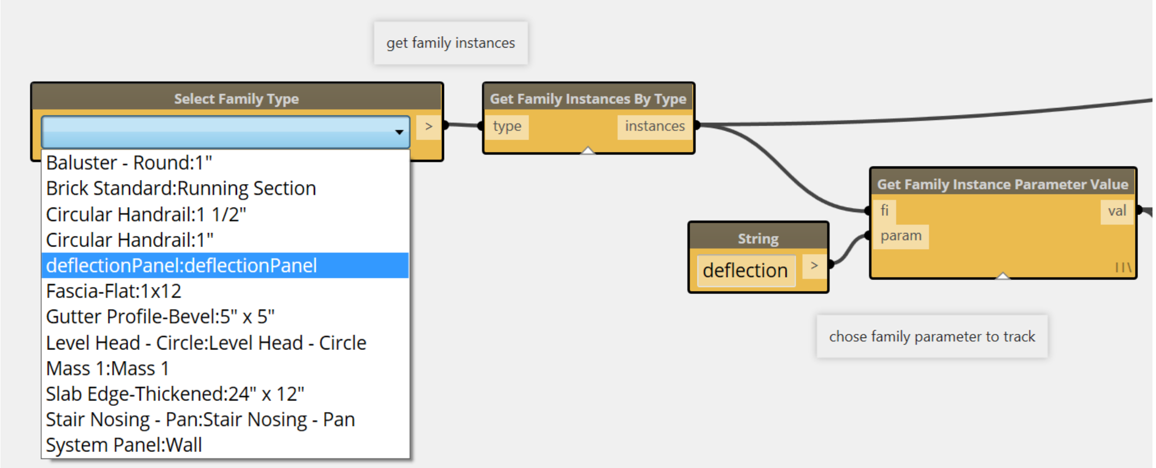
Color

This tutorial aims to introduce how the user can:

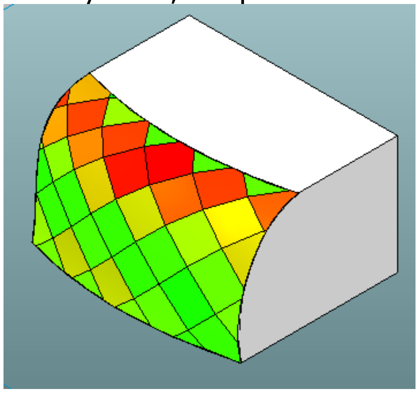
- Control color in Revit for Analysis and other purposes
- Sort data to get the information you want
- Gain familiarity with the Revit concept of View Specific operations

Revit and Vasari treat color in a different way than most other applications. In general, Revit elements have a particular color associated with what they are. That is, color is a very literal representation of the building data . . . brick looks like brick, shingles like shingles. If an element's surface characteristics are messed with, it can also alter quantity takeoffs and other aspects of the model and therefore compromise the integrity of the building data. Instead, users can arbitrarily alter the manner in which a particular element is visualized in a particular view without having any impact on the larger data model. This is called a View-Specific change, is different than changing the element itself, and is purely graphical in nature.

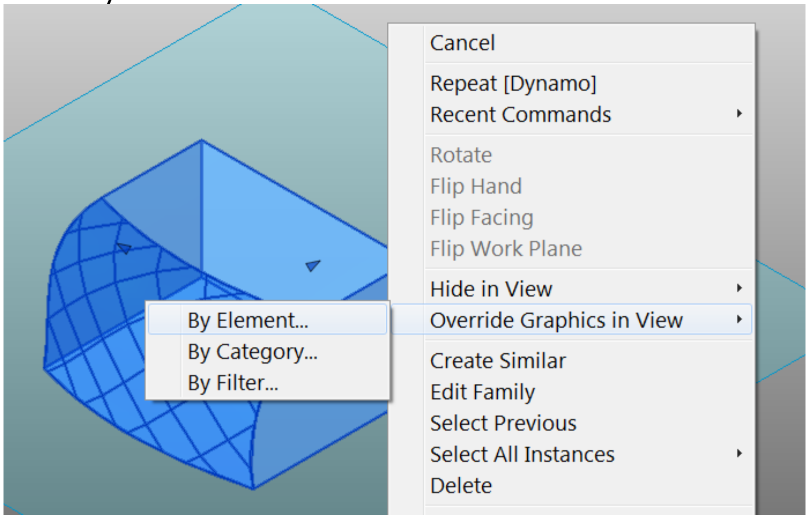
1. Close any open files. In Vasari, Click > 📂 (Open).
2. Open: **C:\Autodesk\Dynamo\Core\samples\22 Color\Override Graphic in View.rvt**
3. In Dynamo, in Help menu, browse to **Samples>22 Color> OverrideColorsInViewFamiles.**
4. The Override Graphics in View project file contains an In-Place mass, with a curtain panel by pattern applied to a double-curved surface. If you TAB>select any of the panels, and inspect the properties. In each panel family, called deflectionPanel, you will see that there is a parameter called "deflection". This is a measurement that was manually added to the panel family to indicate how far out of plane (how not-flat) the panel geometry is in each placement. Most panels that are slightly out of plane can be manufactured with negligible increases in price, but significantly out of plane, or panels that show a large amount of deflection, will be too expensive or difficult to manufacture. The Dynamo workflow is going to inspect this parameter for each panel and color it based on some threshold criteria
5. Dynamo first need to find the right family to look at. In Dynamo, locate the Select Family Type node. In the dropdown, pick "deflectionPanel:deflectionPanel". Dynamo can now pass this specific famlily type to Get Family Instance By Type, and query it for the value of "deflection" with a Get Family Instance Parameter Value node.


get family instances
Select Family Type
Baluster - Round:1"
Brick Standard:Running Section
Circular Handrail:1 1/2"
Circular Handrail:1"
deflectionPanel:deflectionPanel
Fascia-Flat:1x12
Gutter Profile-Bevel:5" x 5"
Level Head - Circle:Level Head - Circle
Mass 1:Mass 1
Slab Edge-Thickened:24" x 12"
Stair Nosing - Pan:Stair Nosing - Pan
System Panel:Wall
Get Family Instances By Type
type
instances
String
deflection
Get Family Instance Parameter Value
fi
param
val
chose family parameter to track

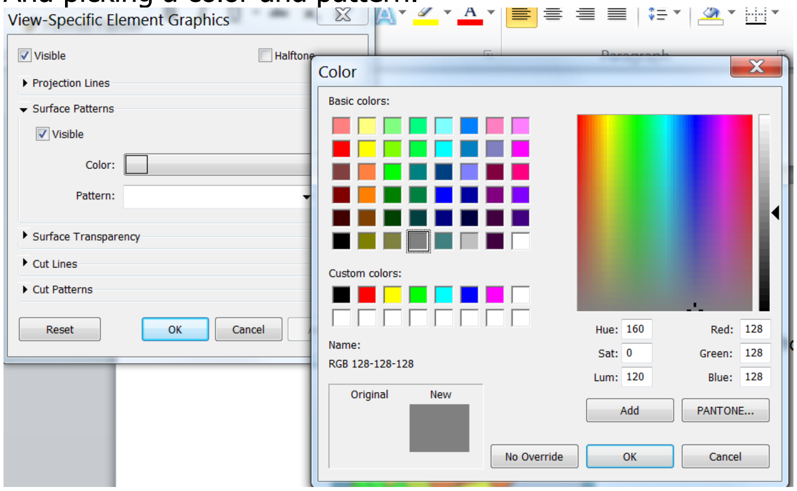
6. Run Dynamo, the panels will be colored in a gradient from green to red.



7. A number things are being done in the graph to get this outcome. Starting from the end: There is an Override Element Color in View node. This node looks at whatever is the "active view" in the Revit model and will now override whatever the default visibility is of any element with a given value. You can simulate this manually by right clicking on a Revit Element and selecting Override Graphics in View>By Element.



And picking a color and pattern.



View-Specific Element Graphics

☑ Visible

□ Halftone

Projection Lines

Surface Patterns

☑ Visible

Color:

Pattern:

Surface Transparency

Cut Lines

Cut Patterns

Color

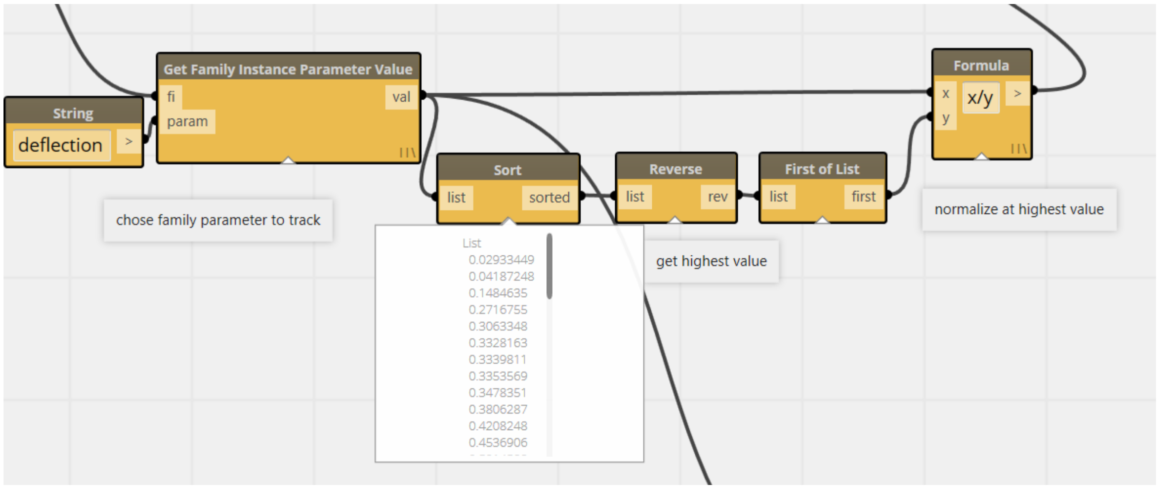
Name:
RGB 128-128-128

Hue: 160
Sat: 0
Lum: 120

Red: 128
Green: 128
Blue: 128

Keep in mind that this is not changing anything about the Revit element itself, it is just changing how this specific element is treated in this specific view. This setting will persist in this view when Dynamo is not present, and Dynamo will remember what panels were changed if they need to be adjusted at a later time. Right Click on one of the panels and you can inspect the setting for its color override.

8. Coloration in this workspace is currently set to give an even range of green to red, with a sorting algorithm that finds the value of the most deformed panel and scales all the colors based on this value.



String: deflection

chose family parameter to track

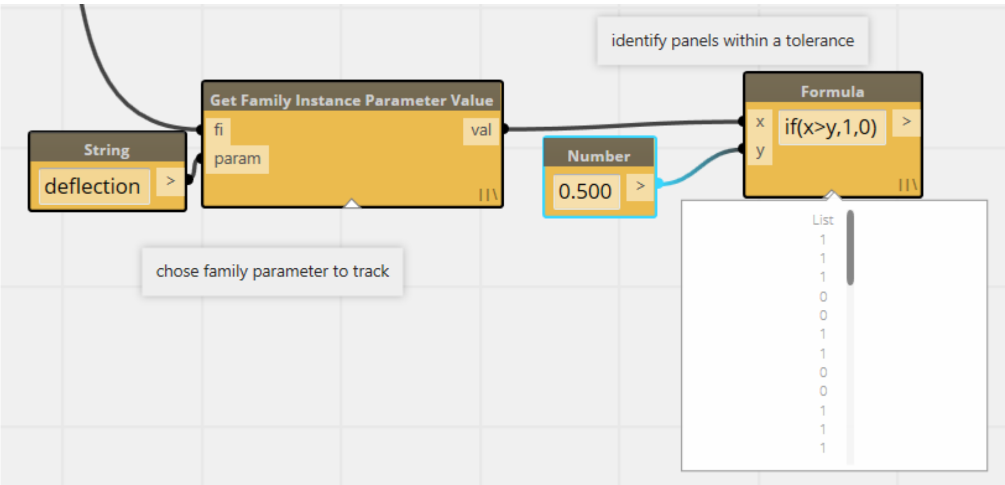
get highest value

normalize at highest value

Formula: x/y

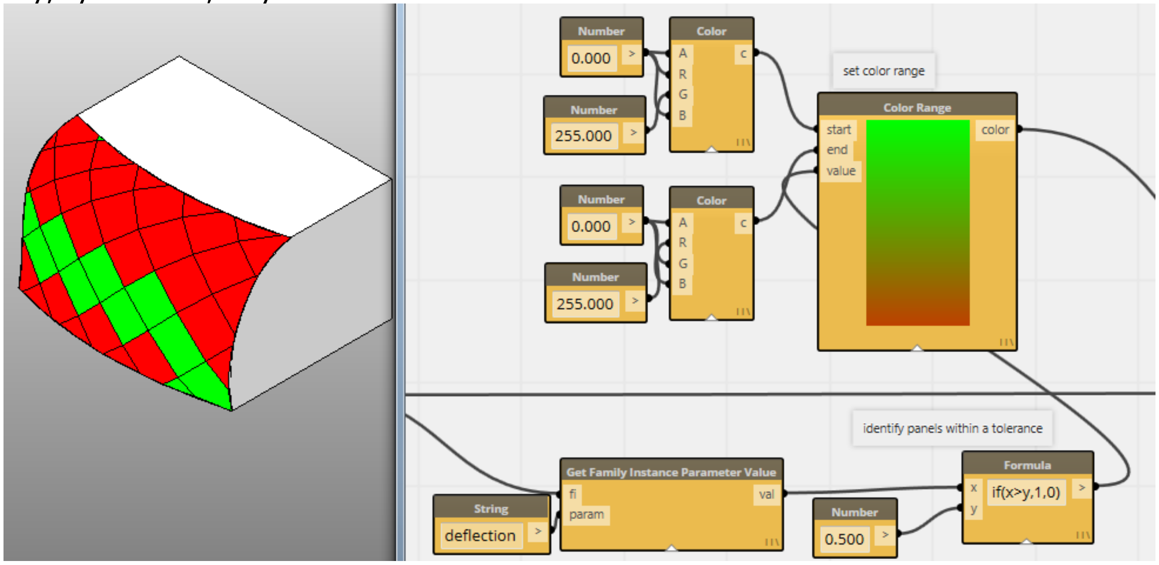
```
List
  0.02933449
  0.04187248
  0.1484635
  0.2716755
  0.3063348
  0.3328163
  0.3339811
  0.3353569
  0.3478351
  0.3806287
  0.4208248
  0.4536906
```

9. However, maybe your coloration needs to simply identify a threshold and indicate what panels are above and which are below. This is a bit easier. By passing the deflection values into a formulat node with an “if” conditional statement, you can detect which values are greater than the acceptable value and which ones are less than the acceptable value. (Find more function syntax for the formula node at http://ncalc.codeplex.com)


identify panels within a tolerance
Get Family Instance Parameter Value
fi
param
val
String
deflection
Number
0.500
Formula
x
y
if(x>y,1,0)
chose family parameter to track

```
List
1
1
1
0
0
1
1
0
0
1
1
1
```

10. By passing this formula output of 0 or 1 to the Color Range node, the color output will either pass on completely red or green values. As Heidi Klum would say, "you are in, or you are OUT!"



Try changing the values of the threshold (the number node feeding into the Formula node), and shifting the ARGB value for the Color Range node.

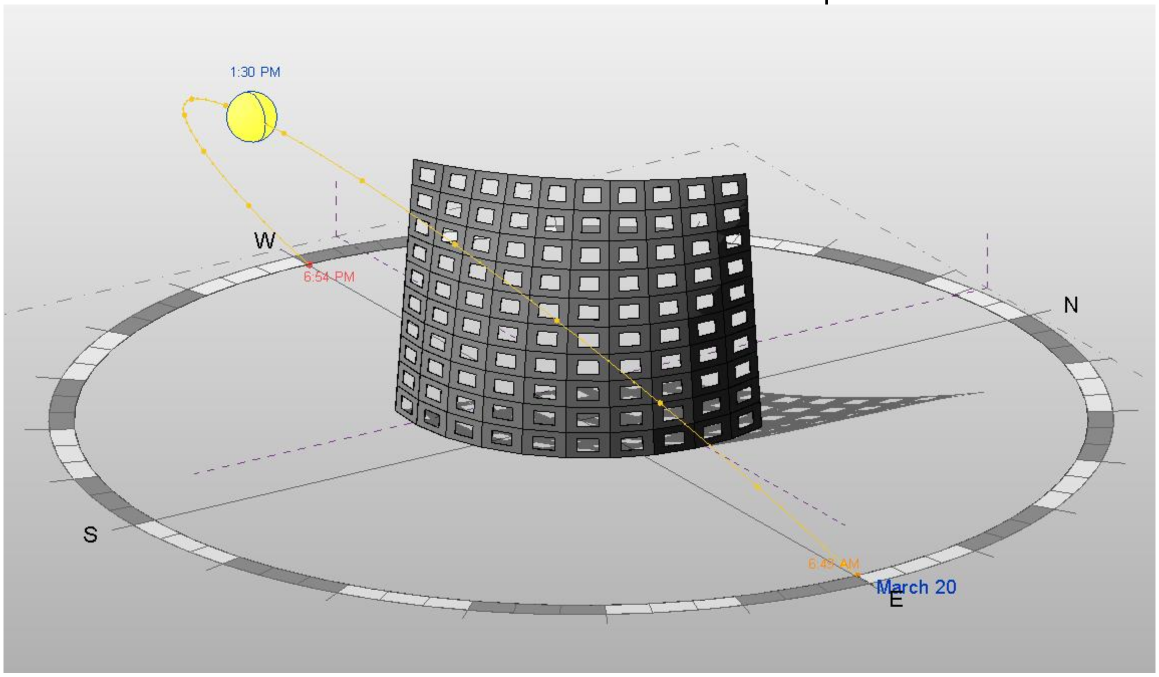
For more examples of coloration, you can also experiment with the wall sample in 22 Color samples. There are also a number of examples using the Analysis Visualization Framework to do view specific coloring, but that is another exercise.

Data Interop

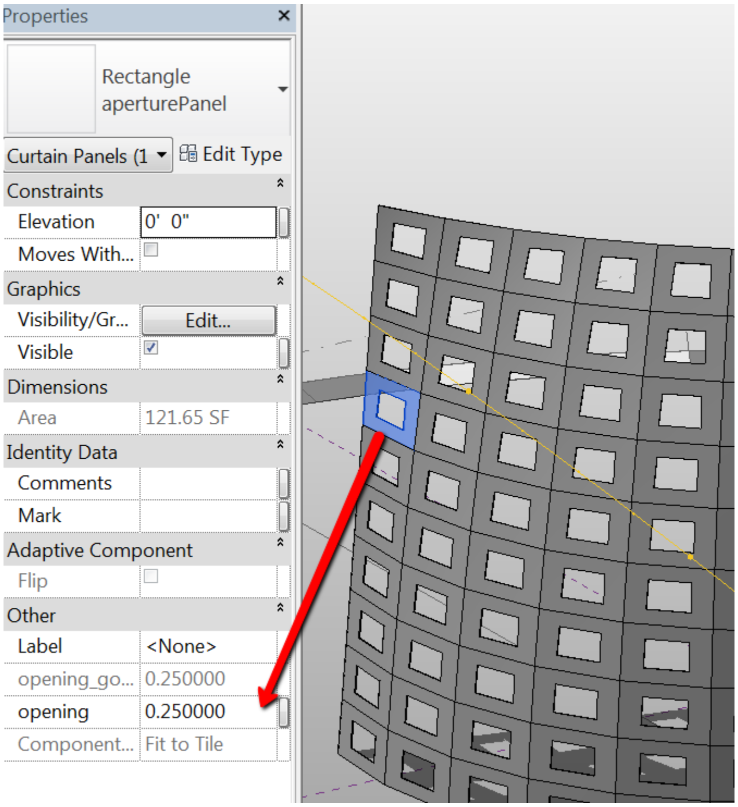
- Bring Data into Dynamo from external sources like spreadsheets and images
- Drive any kind of data into Dynamo and Revit processes
- Export Data to other formats

There are lots of Data out there in the world. Sometimes it comes in the form of number in spreadsheet, or colors in an image. Sometimes it is data that is already in your BIM. This tutorial will show how to move any of those kinds of data to meaningful places in your model.

1. Close any open files. In Vasari, Click V > 🗁 (Open).
2. Open: **C:\Autodesk\Dynamo\Core\samples\20 Panel Manipulations\DataToFamilies.rfa**
3. In Dynamo, in Help menu, browse to **20 Panel Manipulations>DataToFamilies**
4. You should see a collection of Curtain Panel families and a sun path.



If you TAB>select one of these panels, you will see a parameter for adjusting the size of the opening:



Properties

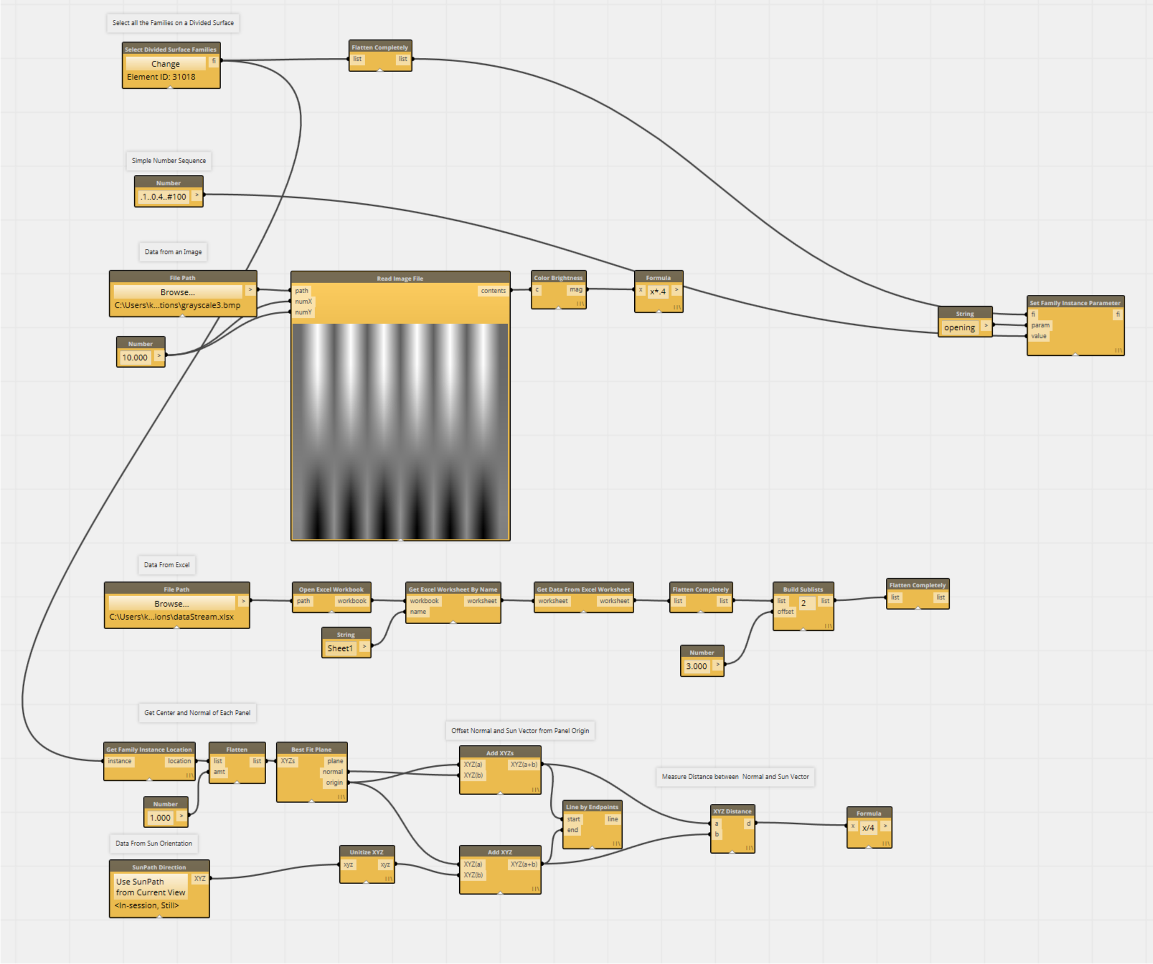
Rectangle
aperturePanel

Curtain Panels (1 | Edit Type

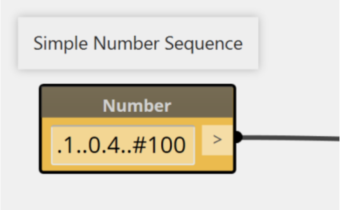
Constraints	
Elevation	0' 0"
Moves With...	☐
Graphics	
Visibility/Gr...	Edit...
Visible	☑
Dimensions	
Area	121.65 SF
Identity Data	
Comments	
Mark	
Adaptive Component	
Flip	☐
Other	
Label	<None>
opening_go...	0.250000
opening	0.250000
Component...	Fit to Tile

If you adjust this number between 0 and .5 it will open and close this aperture. Over .5 it will stay at the .5 value. We will be driving various kinds of data into this parameter.

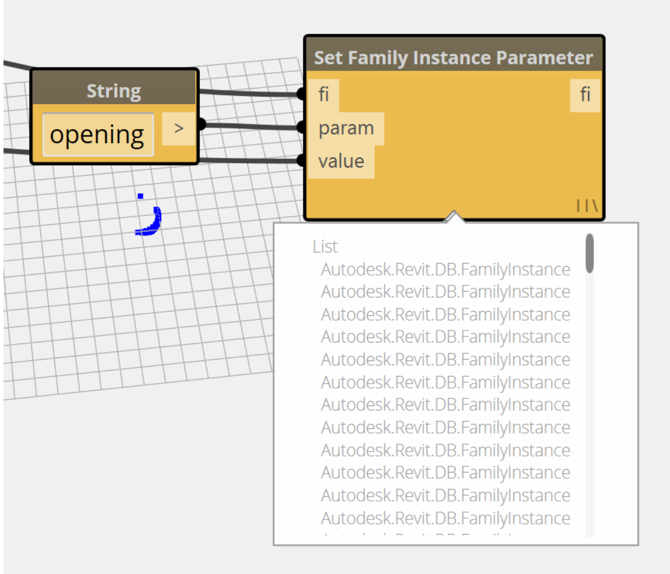
5. In Dynamo, there is a node that pulls out a list of Panel families from the model, and 4 different data sources that we will direct to these panels. Some of these data sources need to be identified. Find the SunPath Direction node and Click on Use SunPath from Current View. This will identify the active view. Also Click on the File Path node and make sure that it is pointing at the dataStream.xlsx file in the samples/PanelManipulations folder.
6. Run Dynamo and there will be a short pause while all the data sources are initialized and the first one passes it's data to the Panel families. You will notice that Excel has started and a spreadsheet is now open, and the Read Image File node has a preview image of the data it is passing.



7. The first data source is just a number sequence passing 100 values between 0 and.04

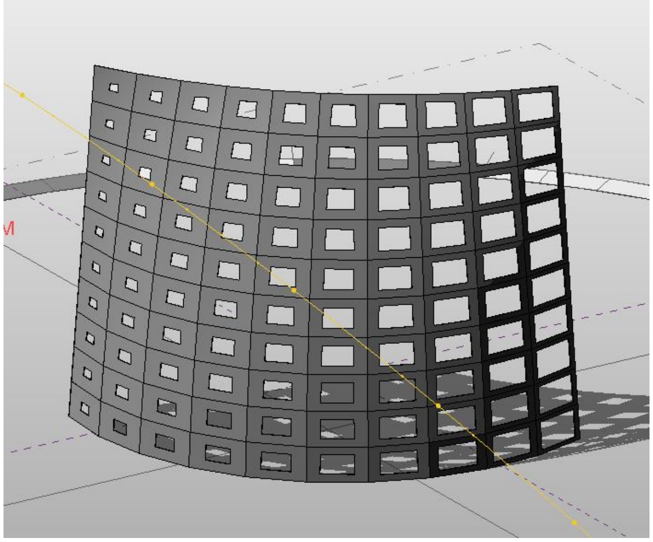


This is then passed into a Set Family Instance Parameter node that adjusts the parameter called “opening” that we inspected before.

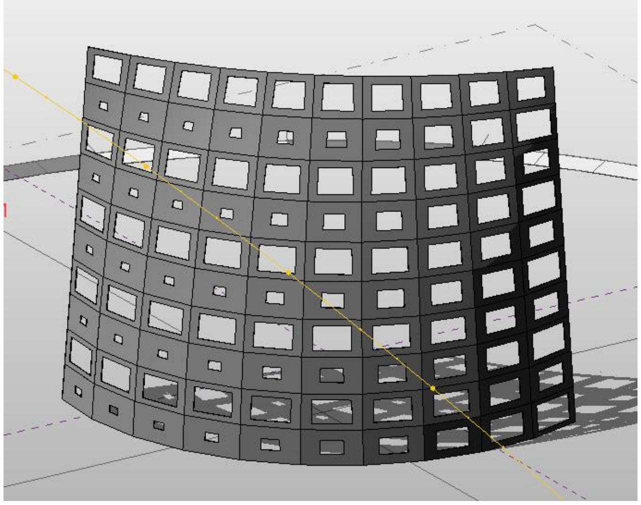


```
List
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
  Autodesk.Revit.DB.FamilyInstance
```

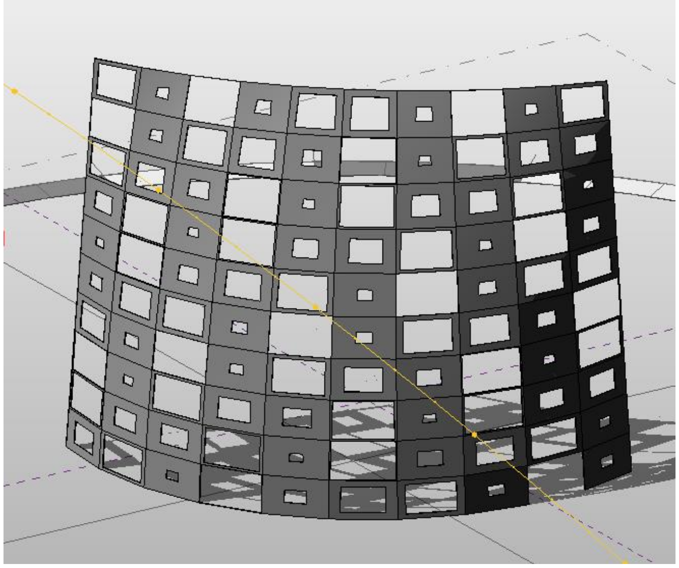
The result is openings between the ranges of .1 and .4



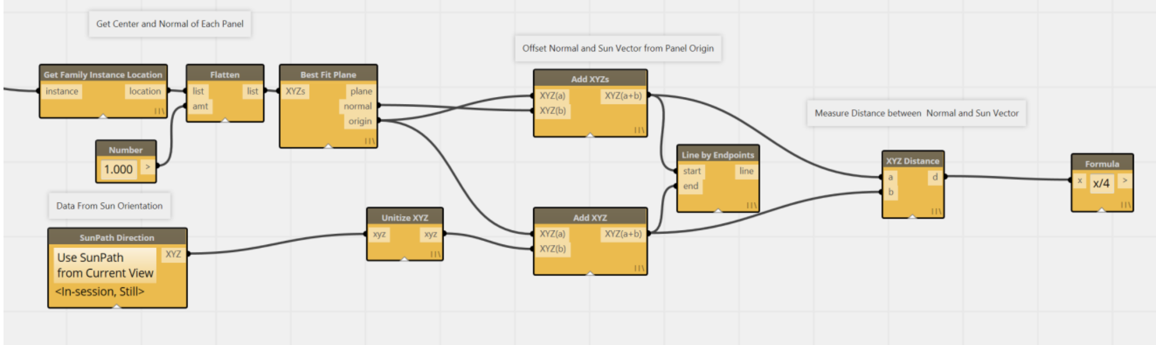
8. Next is an Image Reader node, which extracts the RGB value of a bitmap. It then runs through a Color Brightness node which gets a value from 0-1 for brightness, and then a Formula node that adjusts the values to conform to our 0-.4 range we have been using. If you wire this into the Set Family Instance Parameter's value port and re-execute the graph, you can now pass image data to the panels.

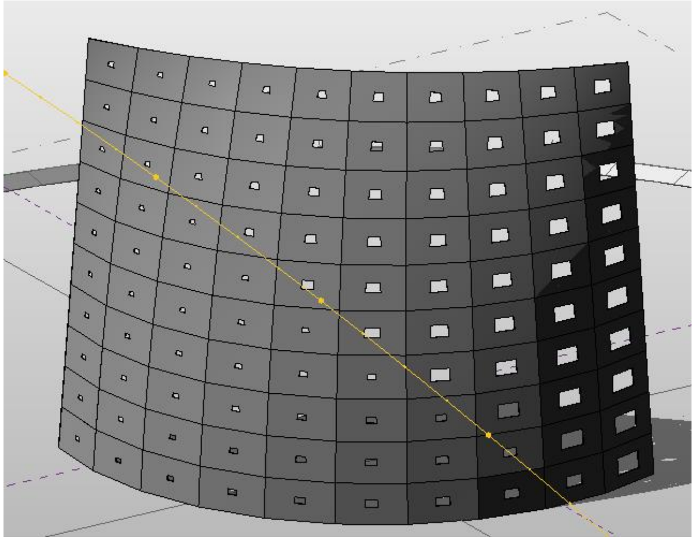


9. Next is an Excel spreadsheet reader, that extracts tabular data, parses the rows to get one number from each (a sine function). Pass this to the families.



10. Finally, we have data that is internal to the model, the position of the sun. However, there is no built in way with Revit to move this data from one place in the model to another. This takes a little work, as you need to extract the direction each panel is facing from the panels, then compare it to the position of the sun, and turn this into a number in our 0-.4 range.

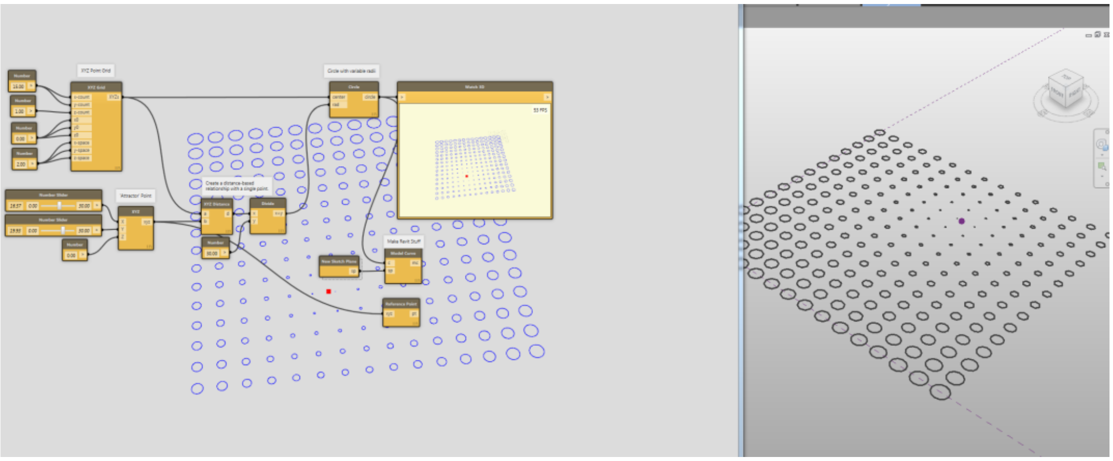




But it does the trick! The more the panel is pointed at the sun, the smaller the aperture. Drag around the sun position and run again.

For more examples of moving data around, Also look at sample 23 Data Import and Export. These demonstrate Excel specific tools for reading in data, parsing, and exporting back out.

Attractor Pattern



This exercise will guide the user through a simple computational design problem using Dynamo. The user will construct an "attractor" logic wherein the distance between points will be used to drive a geometric variation.

- Learn how to compute relationships between XYZ geometry.
- Learn how to construct a basic mathematical relationship with Dynamo Nodes.
- Learn how to create variations in geometry with distance-driven values.
- Convert abstract Geometry to Model Lines
- Visualize geometry in a number of ways

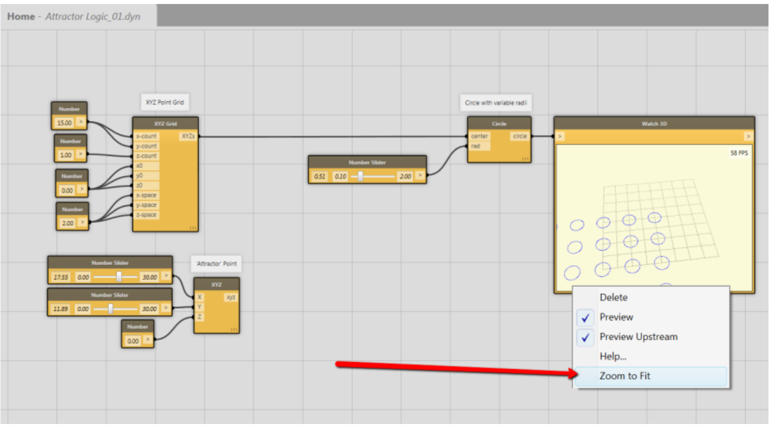
This example creates Revit elements that can be made in any environment, so you can work in any Revit or Vasari context.

1. Close any Vasari files that you have open
2. Navigate to

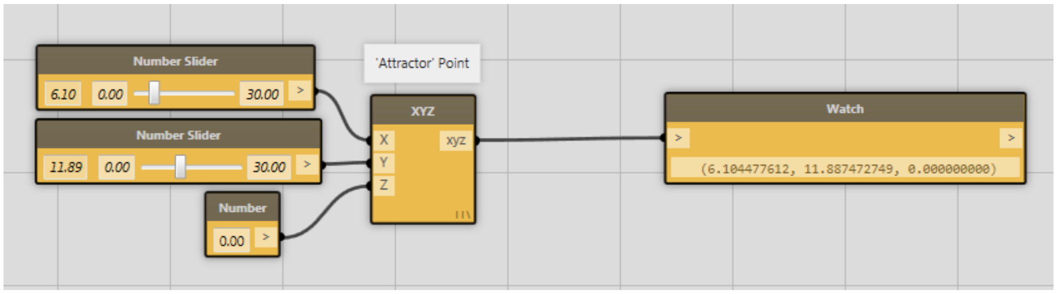
C:\Autodesk\Dynamo\Core\samples\10 Attractor

3. Set to Run Automatically, and hit Run

4. Zoom to Fit Watch 3d

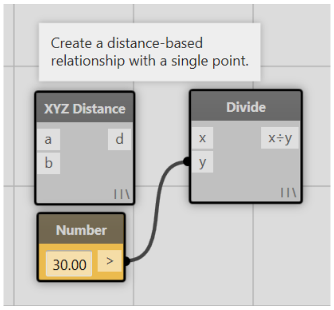


5. Adjust the slider wired into the Circle generating component. Notice that although you are getting different sized circles, you are not creating Revit geometry.
6. Adjust the Sliders for the 'Attractor" Point. Notice that there is no geometric representation although the Watch node registers a change

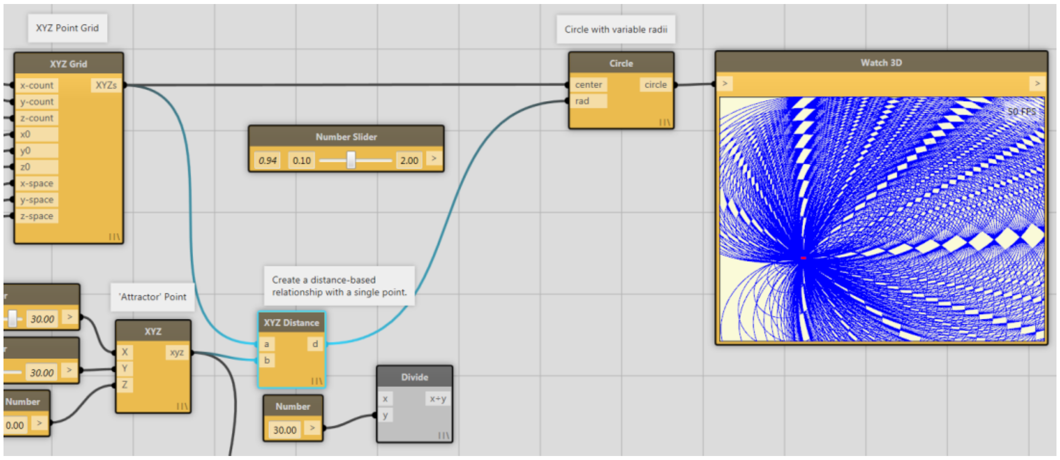


```
(6.104477612, 11.887472749, 0.000000000)
```

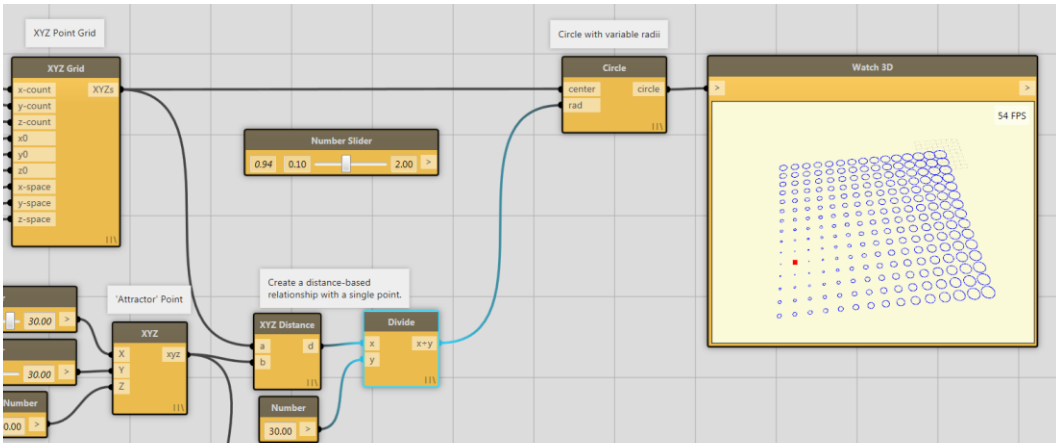
7. In the View menu, click on Preview Background. All geometric entities are rendered in the canvas.
   a. Toggle Navigation by clicking cntrl-G. Zoom out with the scroll wheel, orbit with right click, and pan with middle mouse
   b. Hit cntrl-G to exit navigation
   c. Change the Attractor Point sliders, see XYZ and Circles in the same view.
8. Now we need to connect the proximity of the Attractor Point to the radius of the circles. In the lower portion of the screen, find a cluster of nodes: XYZ Distance, Divide, and a Number.



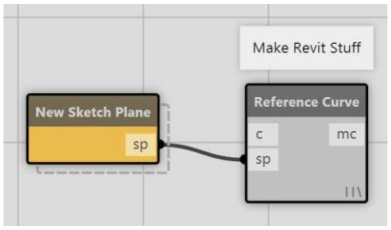
9. Measure the distance between each grid point and the attractor point. Pass the position of the Attractor point and the position of each XYZ Point grid to the XYZ Distance node and then connect the resulting distance to the circle’s radius.



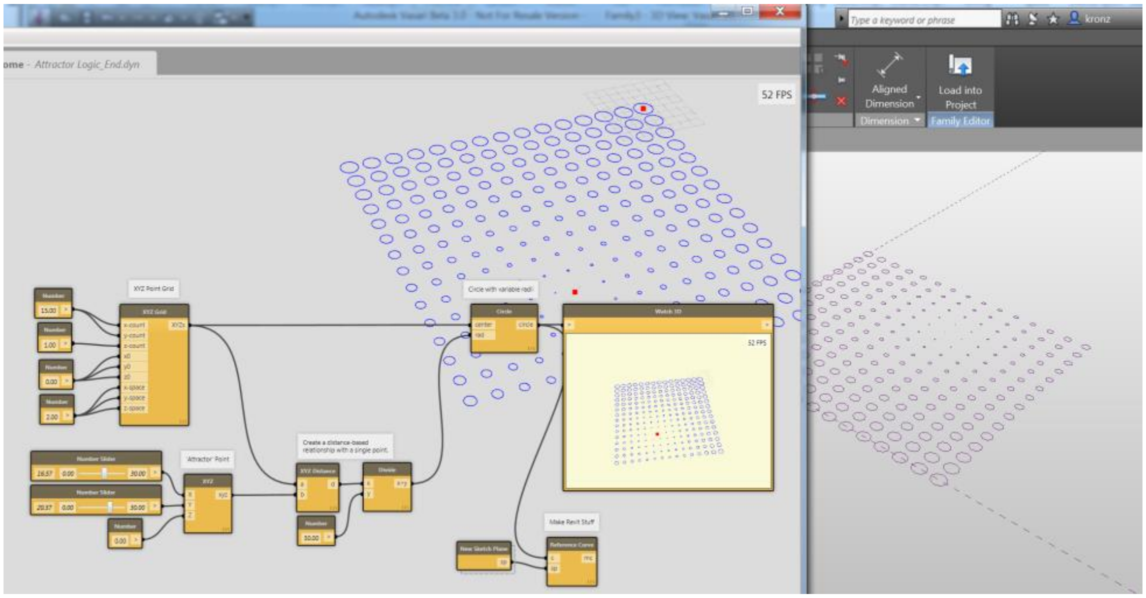
10. The resulting Geometry is a bit of a mess, as each circle has the same radius as the distance of the attractor point. Let’s moderate this by passing the XYZ Distance Node through a Divivision operator first



11. So far we have just made abstract geometry. We can dump out this data into Revit Model Lines by attaching the Circle or Watch 3d outputs to Reference Curve and New Sketch Plane nodes



12. The resulting workflow results in this Revit Geometry which is still associated with the graph.
    a. Experiment with the sliders to get a feel for changes in model update with and without Revit geometry.
    b. If you save the Dynamo file and the Vasari file, the Dynamo file will "remember" the geometry it has previously created and manipulate its parameters later, not create new stuff.

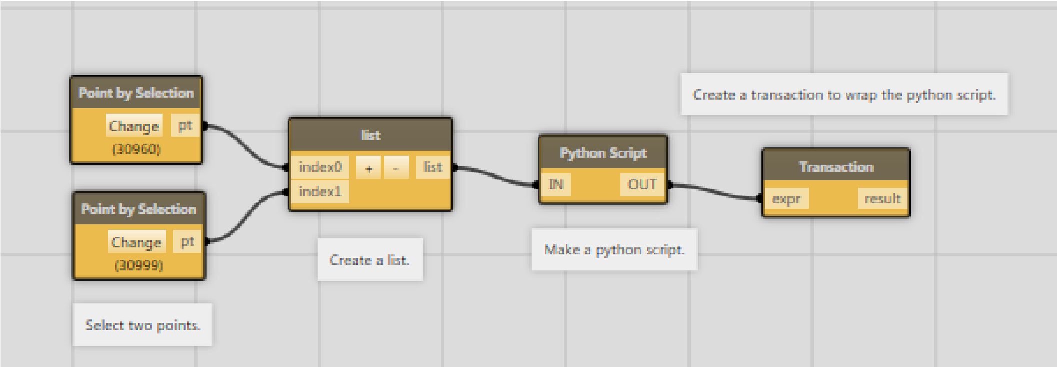


Python: Script a Sine Wave

The aim of this tutorial is to show you how to:

- **Edit and run a python script** in Dynamo
- Show how to **use changes in the document to update your Python script.**

1. Close any Vasari files that you have open
2. From Vasari, Click > 📂 (New) > Family.
3. Use Mass.rft in the Conceptual Mass folder.
4. Go to Add-ins tab and launch Dynamo. If you already had it open, close and re-open it to re-associate Dynamo with the newly conceptual mass
5. From the Dynamo File Menu, go to **Samples/ 06 Python Node / create sine wave from selected points.** You should see the following in your workspace:



Create a transaction to wrap the python script.

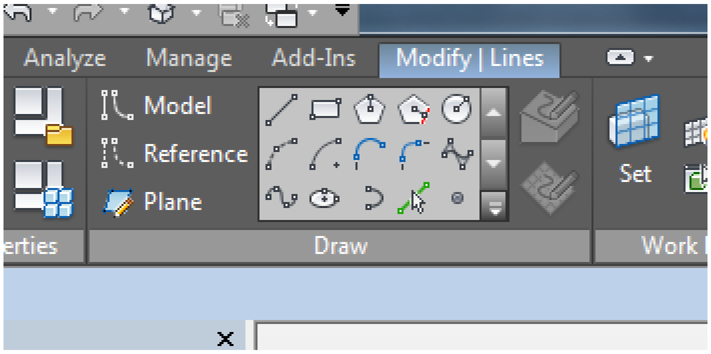
Create a list.

Make a python script.

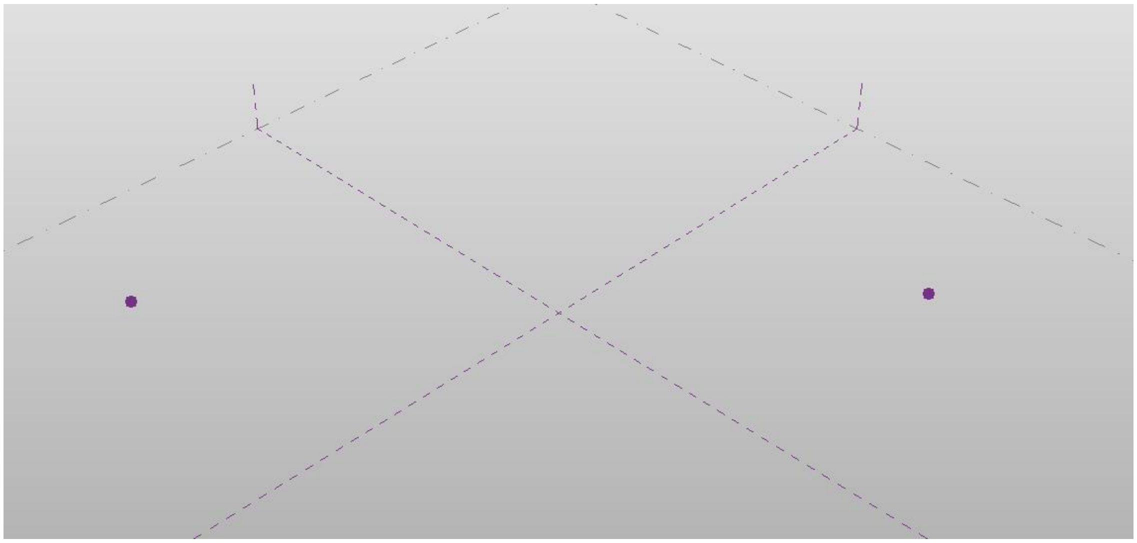
Select two points.

A python node in a workspace.

6. Now place two points using the Vasari Sketching Gallery. The reference point button is in the bottom right corner.

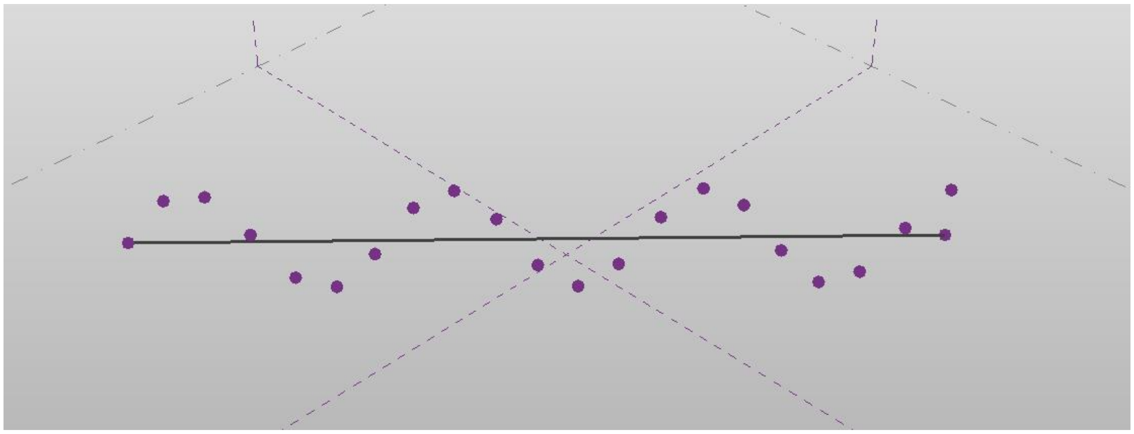


7. You should have two Reference points in the Vasari view:



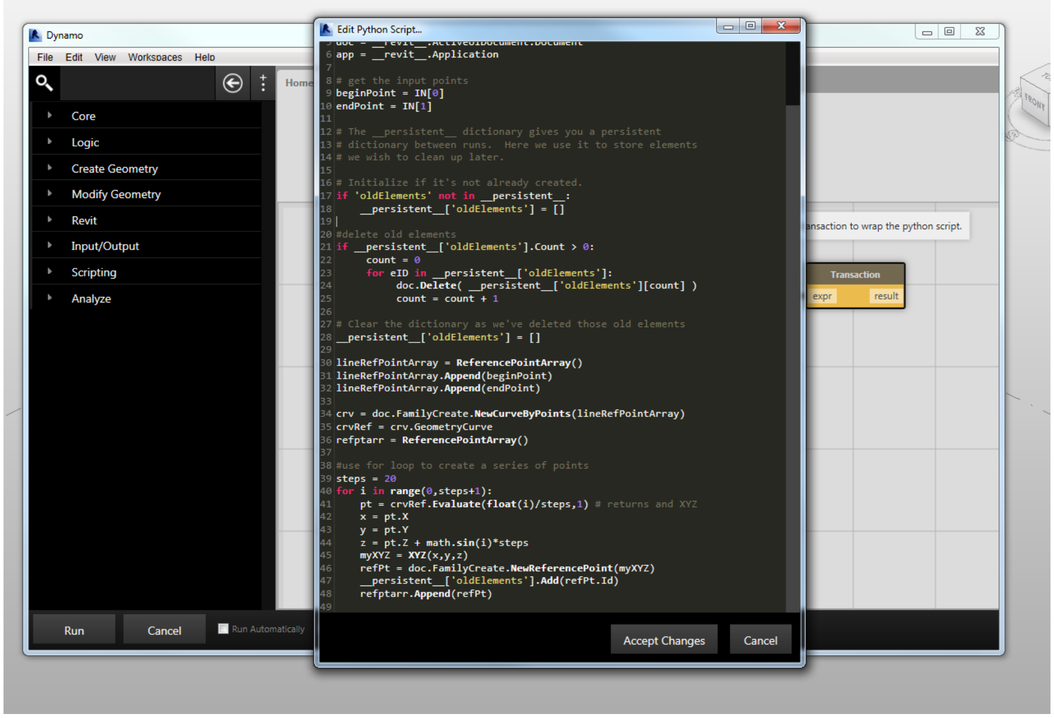
Two reference points

8. Select each point using the **Select Point** nodes in Dynamo.
9. Press **Run**. You should see a line and a sine wave plotted between the two entered points. These elements were created by a Python script!



The output of the Python script

10. Set Dynamo to **Run Automatically**
11. Experiment with moving the points or the straight line, notice how the sine wave is updated to follow the new position? This is because **Dynamo and the Python node can watch specific elements and then re-run the workspace to keep things in sync based on your changes.**
12. Right click on the Python node and click Edit… to show the script editor:

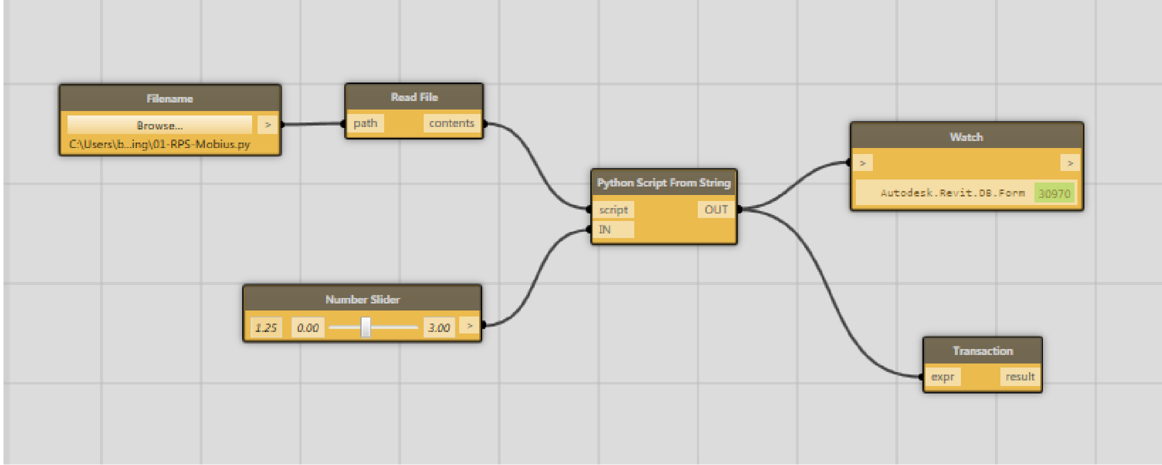


```
 5 doc = __revit__.ActiveUIDocument.Document
 6 app = __revit__.Application
 7
 8 # get the input points
 9 beginPoint = IN[0]
10 endPoint = IN[1]
11
12 # The __persistent__ dictionary gives you a persistent
13 # dictionary between runs.  Here we use it to store elements
14 # we wish to clean up later.
15
16 # Initialize if it's not already created.
17 if 'oldElements' not in __persistent__:
18     __persistent__['oldElements'] = []
19
20 #delete old elements
21 if __persistent__['oldElements'].Count > 0:
22      count = 0
23      for eID in __persistent__['oldElements']:
24           doc.Delete( __persistent__['oldElements'][count] )
25           count = count + 1
26
27 # Clear the dictionary as we've deleted those old elements
28 __persistent__['oldElements'] = []
29
30 lineRefPointArray = ReferencePointArray()
31 lineRefPointArray.Append(beginPoint)
32 lineRefPointArray.Append(endPoint)
33
34 crv = doc.FamilyCreate.NewCurveByPoints(lineRefPointArray)
35 crvRef = crv.GeometryCurve
36 refptarr = ReferencePointArray()
37
38 #use for loop to create a series of points
39 steps = 20
40 for i in range(0,steps+1):
41     pt = crvRef.Evaluate(float(i)/steps,1) # returns and XYZ
42     x = pt.X
43     y = pt.Y
44     z = pt.Z + math.sin(i)*steps
45     myXYZ = XYZ(x,y,z)
46     refPt = doc.FamilyCreate.NewReferencePoint(myXYZ)
47     __persistent__['oldElements'].Add(refPt.Id)
48     refptarr.Append(refPt)
49
```

The python script editor in Dynamo

13. We will not dissect it in detail now, but take a look at how the **beginPoint** and **endPoint** variables, get assigned objects from **IN**. The **IN** and **OUT** ports map to variables in the script.
14. In the script editor, go to line 39 and edit the value of steps.
15. Click **Run** again. You'll notice that the sine wave has changed. Feel free to experiment with this script.

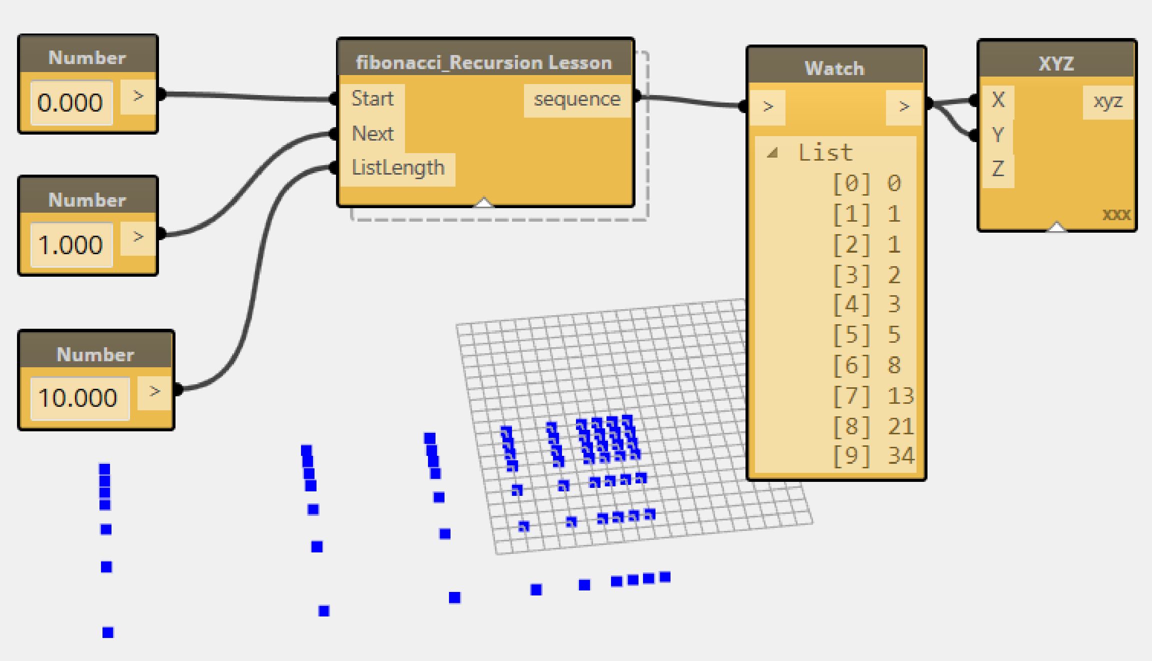
You don't have to use Dynamo to edit your python scripts. If you feel more comfortable in a different text editor, you can use the Python Script From String node. Combined with the File Path node and Read file, you can read your files from a text file and Dynamo will automatically update. Check out the DynamicMobius example under Help > Samples > Dynamic Python Editing for an example of this.



An example of using Python Script From String to edit a python script with a different text editor

Sharing and Reusing Algorithms with the Package Manager

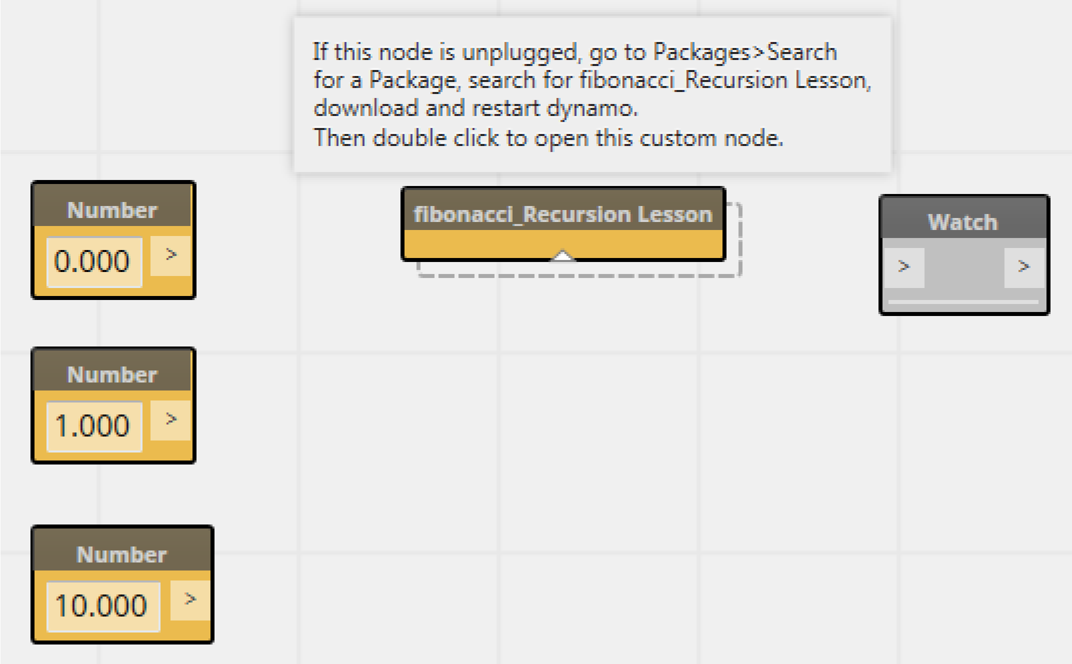
- Re-use code
- Simplify large graphs
- Create a recursive function
- Access advanced workflows made by others



```
List
  [0] 0
  [1] 1
  [2] 1
  [3] 2
  [4] 3
  [5] 5
  [6] 8
  [7] 13
  [8] 21
  [9] 34
```

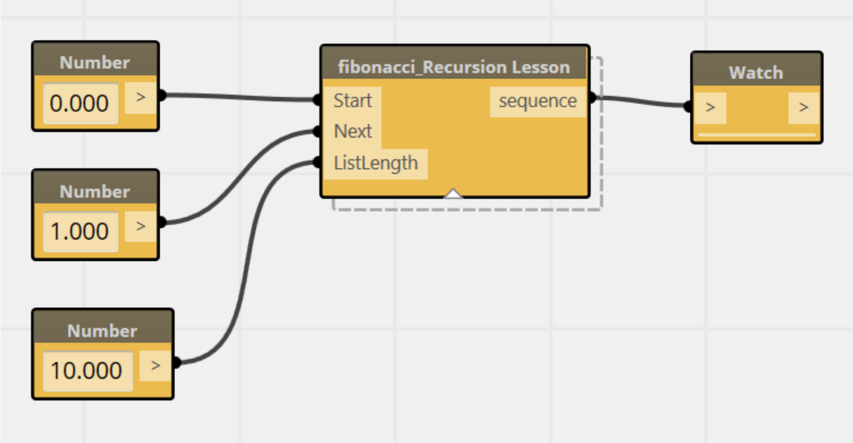
As shown in the first example, Custom node creation allows users to make compact representations of groups of functionality. Users can identify modular sections of their workflows for reuse either in the same project, or in other projects. But Custom nodes also allow for more advance functionality in Dynamo by accessing recursive functionality.

1. In the Help Menu, Open sample 21 Package Manager>Recursion
2. The workspace will look a little strange, with a handful of nodes that are disconnected

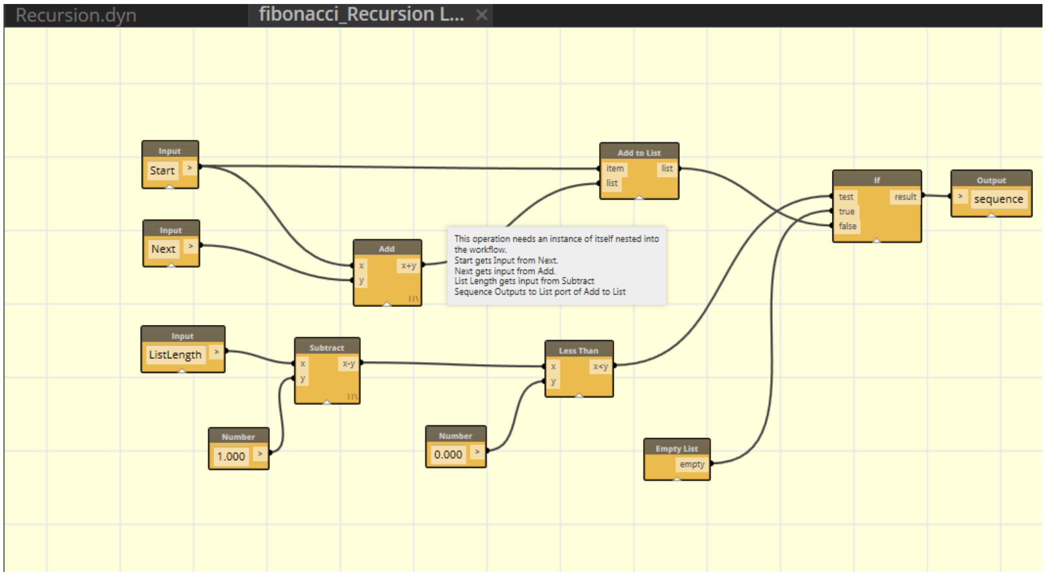

If this node is unplugged, go to Packages>Search for a Package, search for fibonacci_Recursion Lesson, download and restart dynamo.
Then double click to open this custom node.

The "fibonacci_Recursion Lesson" node is a Custom Node that was saved with the Recursion.dyn, but you don't have that Custom Node! We are considering saving Custom nodes in the dyn file itself, but for now, you can download this missing bit of functionality from the Package Manager.

3. In the pull down Menu, go to Packages>Search for a Package, and locate fibonacci_Recursion Lesson. Install it.
4. Close Dynamo, Re-start Dynamo, and open sample 21 Package Manager>Recursion. This time your workspace will look like this:

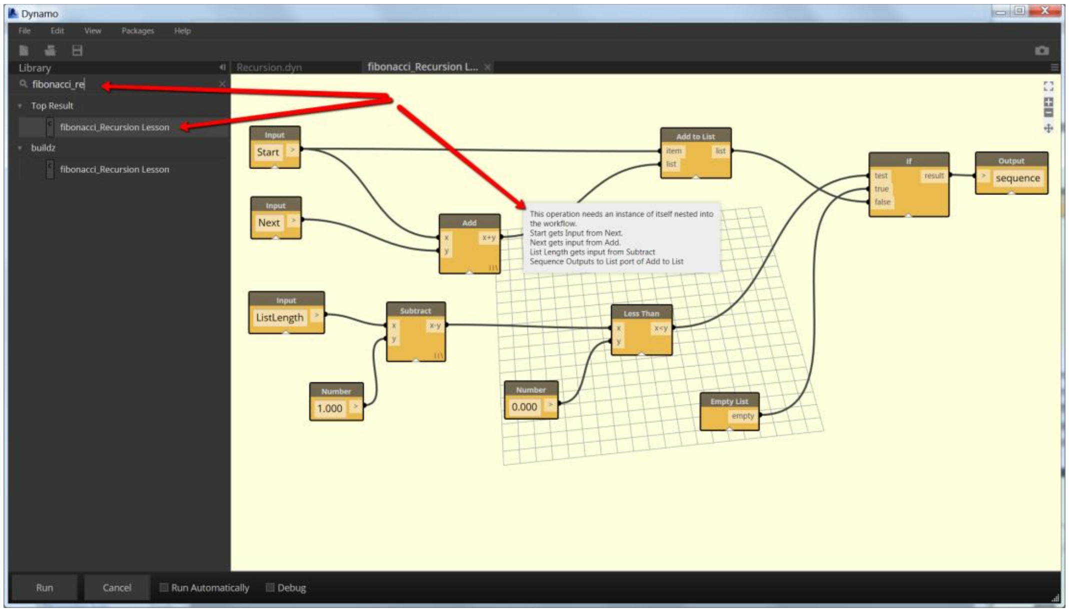


5. Usually a package that you download will be a complete piece of functionality. In this case, we will use the custom node to show how to contruct advanced functionality. Double click on the node and you will now have a second tab open which shows the contents of the node.



This operation needs an instance of itself nested into the workflow.
Start gets Input from Next.
Next gets input from Add.
List Length gets input from Subtract
Sequence Outputs to List port of Add to List

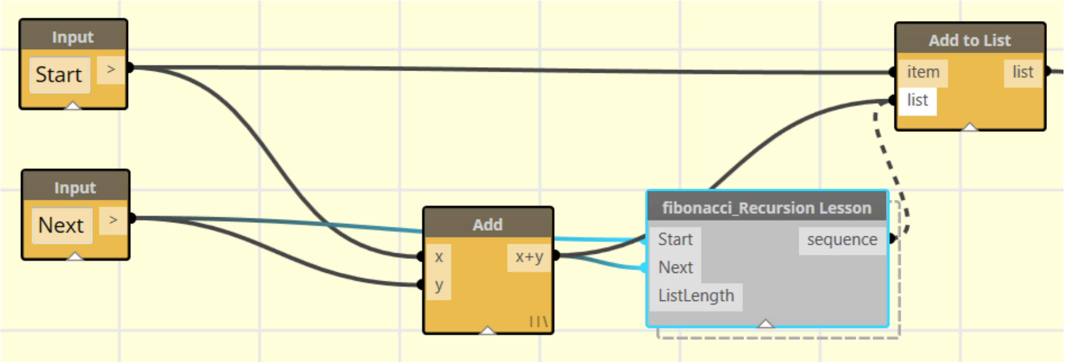
6. The logic of this node is incomplete. To create a fibonnaci sequence, the computation needs a starting sequence (usually 0,1) which is then recursively added. The second number is added to the first, then the resulting third number is added to the second, and so on. In the search bar, enter "fibonnaci" and you will now see your loaded custom node appear. Place it INSIDE of the active fibonacciRecursion Lesson.dyf workspace.



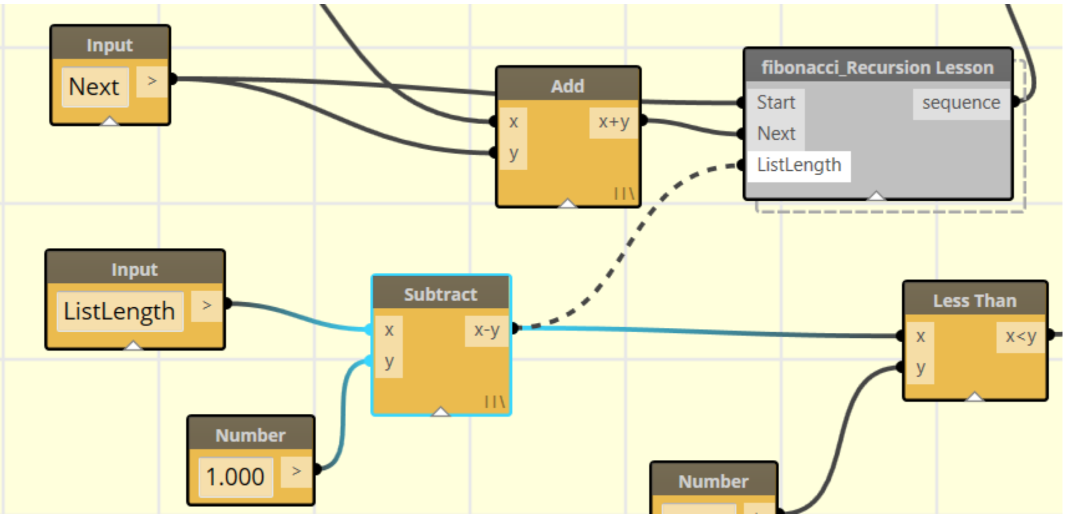
This operation needs an instance of itself nested into the workflow.
Start gets Input from Next.
Next gets input from Add.
List Length gets input from Subtract
Sequence Outputs to List port of Add to List

7. The second number in the Fibonacci sequence becomes the starting number for the next calculation. We can now pass the "next" input to the "start" input of the recursive node. The additive result of Start and Next becomes the

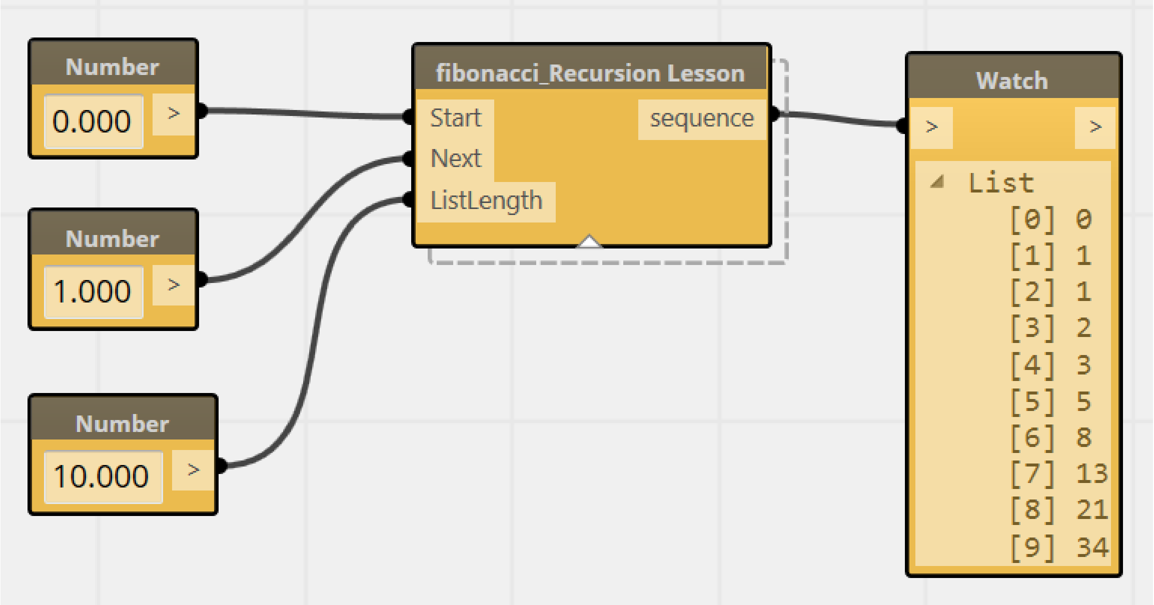
subsequent calculation's "next" input, and the "Result" of this calculation gets added to a growing list of numbers.



8. This calculation would continue infinitely if there was not condition specified to stop it. The ListLength creates a countdown, subtracting one from every loop of the recursive function, until it becomes zero, at which point a conditional statement tells the loop to cease. Wire the countdown ListLength into the recursive node. See how the "IF" node creates a condition wherein the appended list is passed on until the ListLength passes a value less than zero.



9. Your Node does not need to be saved to experiment and see if it is working. Click on the Recursion.dyn tab and try it out



```
List
  [0] 0
  [1] 1
  [2] 1
  [3] 2
  [4] 3
  [5] 5
  [6] 8
  [7] 13
  [8] 21
  [9] 34
```

If you have the behavior you want, you can now save your node for later use like this. However, keep in mind that if you Save As a different name . . . you will need both the top level node and the nested (recursive) node to have the same name!

Many More Examples

These tutorials have only covered a very small subset of the sample workflows that are available. There are more examples within the application, in the Help>Samples dropdown (keep in mind that many have accompanying rvt and rfa files that are by default kept in c://Autodesk/Dynamo/Core/samples. This will be most obvious where there are nodes that select Revit elements or where they place specific families.) For even more examples, please take a look at the hundreds of simple .dyn files that are used in the automated testing process on Github.
https://github.com/ikeough/Dynamo/tree/master/test
This folder is divided up into core, pkgs, and Revit.

The core functionality deals with nodes and operations that are independent of Revit, things like list management, math, conditional statements. You may also note that there are samples here that deal with geometry. These examples are using the Autodesk Shape Manager geometry kernel (ASM), and reflect development we are working on to be able to expand beyond the geometric capabilities of Revit. This geometry is still pretty experimental, so it does not yet connect up to Revit (although you can export/import with the .sat file format.)

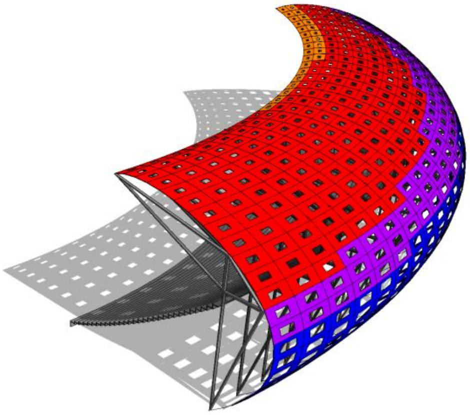
Pkgs folder is entirely concerned with making sure that the package manager continues to work. You can download any samples here from the package manager already.

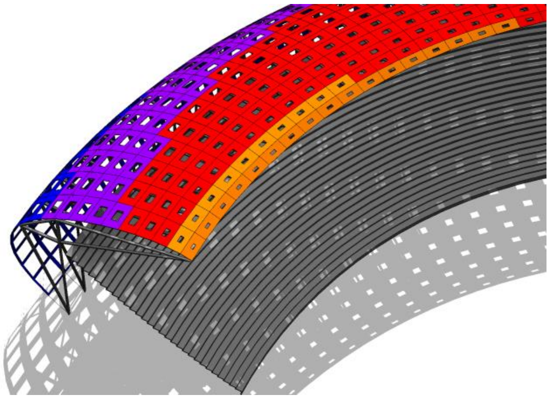
The Revit folder houses samples that are entirely about operations specific to Revit. Revit solids creation, family placements, view manipulations, etc. All of these tests involve some amount of interaction with the Revit document.

In theory, we should have samples that reference EVERY node that is available in Dynamo . . . we aren't there yet, but we are working on it. If you are looking for a particular node and you can't find an sample for it in this repo, please drop us a note. You need it, and we need a test to cover it.

What Else Can Dynamo Do?

The above examples all demonstrate many of the individual capabilities of Dynamo working on top of either Vasari or Revit. By combining these and other techniques, users can build up complex and robust parametric building systems. The following images demonstrate one such project, a Stadium enclosure and seating bowl. The seating bowl is formed by a recursive algorithm that optimizes the view for each seat based on the seat below it. The enclosure creates a self-adjusting shading system based upon the orientation of the panels relative to the sun. The structural system is a combination of manually defined and Dynamo placed truss systems, leveraging both the rule based and hand modelled strengths of Dynamo and Vasari.





Where to Learn More about Dynamo

Dynamobim.org
Official builds, works in progress, forum discussions, and learning resources

github.com/ikeough/Dynamo
Watch development as it takes place

github.com/ikeough/Dynamo/issues
Submit bugs, comments, and improvements for Dynamo