MIT 4.567 Spring 2017
Prepared by: Takehiko Nagakura Rev.: 2017.04.17

Lab Session Note for Revit Architecture: Dynamo examples

This handout explains the attached .dyn sample file(s) which shows a few key methods of Dynamo for constructing a parametric assembly.

Versions tested: Revit 2015 and 2016 with Dynamo 0.9.2, Revit 2017 with Dynamo 1.3 (not a full test)

1. Getting started

Make sure to install most recent version of Dynamo (for Revit) on your pc.
http://dynamobim.org/
To learn Dynamo, the above site has links to many tutorials including
http://dynamoprimer.com/ (Dyanamo Primer: official manual)
To learn about individual nodes:
http://dictionary.dynamobim.com/#/

[Revit] > (open a project file) > Add-ins > {Visual Programming} > Dynamo x.x:
You need a Revit project file open for this menu to get activated.
If you have multiple versions of Dynamo, you can pull-down Dynamo menu for selecting one.

2. Some interface tips

A. 3D view vs Graphic view
Dynamo constructs its own graphics inside Dynamo which then is mirrored into Revit project. This graphics inside Dynamo often includes somewhat simplified representation of Revit objects, and is superimposed with the graph of parametric nodes and links. You can toggle the view control (zoom, pan, orbit, etc) of the graphics view and graph view through the icon on top-right of Dynamo window.

B. Adding comments in Graph view
You can select a group of nodes and right-click to choose Create Group. This adds green background for this group of nodes and allows you to add some comment in text.

3. Running computation Automatic or Manual

The switch between these two methods is available at the lower left corner of Dynamo window
Automatic mode allows the parametric computation to run automatically when you change nodes and links in the graph. However it is recommended to use Manual mode instead for designing the graph and click Run, as Automatic mode can crash or freeze easily.
Also, when you run the calculation in Dynamo, Revit needs to be in the state when
{Select} > Modify (arrow icon to the left of the menu icons)
is in effect. If it is in the middle of other operations, Dynamo may get frozen.

4. Packages

Nodes other than the default set are available by loading external packages. For this handout, load the following two packages.
Spring Nodes package: this provides
FamilyInstance.ByPoints: This can put a door/window in a wall, for example.

Element.SetLocation: This can change a parametric value for an object in Revit.
(this node is now included OOTB)

Clockwork package: this provides
Element.location : This can extract a line on a plan defining a wall, for example.
(this node is now included OOTB)

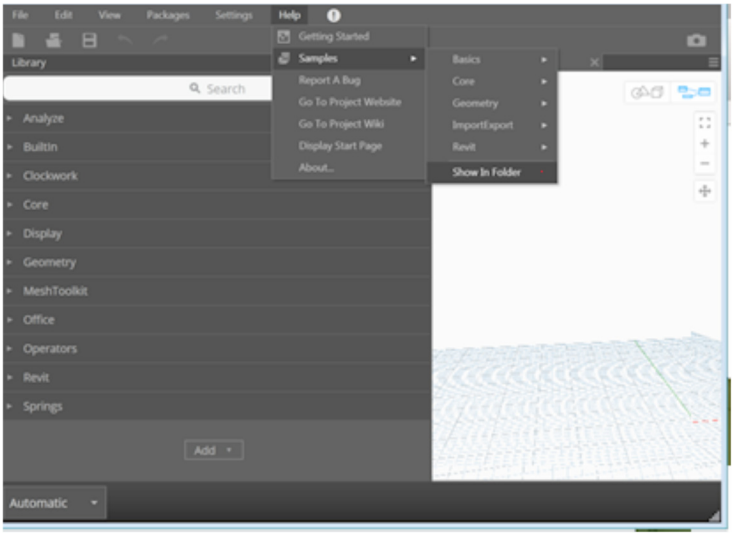
[Dynamo] > Packages > Search for a page > (type package name you want): This may take a while to initialize the search so be patient : Download one by clicking on the downward arrow.

Samples:
Class tutorial page includes "dynamo sample.rvt" that you can try, or use your own simple file to get started. You can also find other samples here:

C:\ProgramData\Dynamo\Dynamo Revit\1.x\Samples\Data

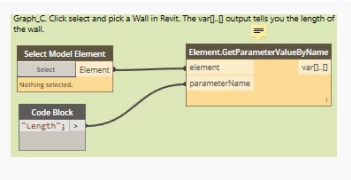
or

[Dynamo menu] : Help > Samples > Show-in Folder



5. **Example 1** (See 4561_dynamo_sample_tn01_v01.dyn)

Graph_A: shows how to list parameter names of an object
Graph_B: shows how to get Wall height.
Graph_C: shows how to get Wall width (length)
Graph_D: shows how to change Wall height
Graph_E: shows how to change Wall elevational location



- **[Dynamo] Revit > Selection > SelectModelElement : Make Node-A**
  (select) : Click the button and pick a wall in Revit model
  (output) Element: Wall is made available here

- **[Dynamo] Revit > Elements > Element > Element.Parameters:**
  (input): Connect output from Node-A
  (output) Parameter: This lists all available named parameters of Node A

- **[Dynamo] Spring Node > Revit > Elements > Element > GetParameterValueByName**
  (input) element: Connect output from Node-A
  (input) parameterName: Connect output from Code Block
  For wall, use "Length"; or "Unconnected Height"; in Code Block
  The label is case sensitive! Carefully type the exact name.

- **[Dynamo] Spring Node > Revit > Elements > Element > SetParameterByName**
  (input) element: Connect output from Node-A
  (input) parameterName: Connect output value from Code Block

- **[Dynamo]** Core > Input > Code Block: Use this to specify values for other nodes.
  The value should be terminated with ; (colon) which signifies end of a code line.

  parameterName for **GetParameterValueByName**: Double quote the name as a string value
  value for **SetParameterByName**: type a number

- **[Dynamo]** Core > Input > NumberSlider: Use this to specify a value interactively for other nodes.
  Pull-down the hat mark to the left of the slider value to set min, max and step.

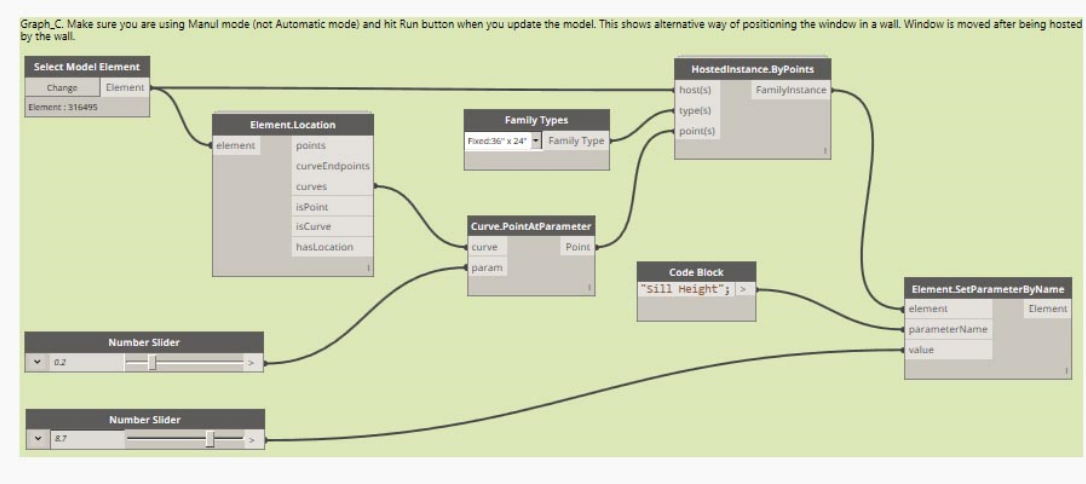
- **[Dynamo]** Core > View > Watch: Use this to check the output of nodes.

6. **Example 2** (See 4561_dynamo_sample_tn02_v01.dyn)

   To test this example, make sure to use **Manual calculation mode** and Run it manually. Automatic mode will likely to freeze/crash.

   These example show how to place a window/door in a wall.
   Graph_A: shows how to place a window in a wall.

Graph_B: shows how to set the height of the window.
Graph_C: shows how to set the height of the window in another way.



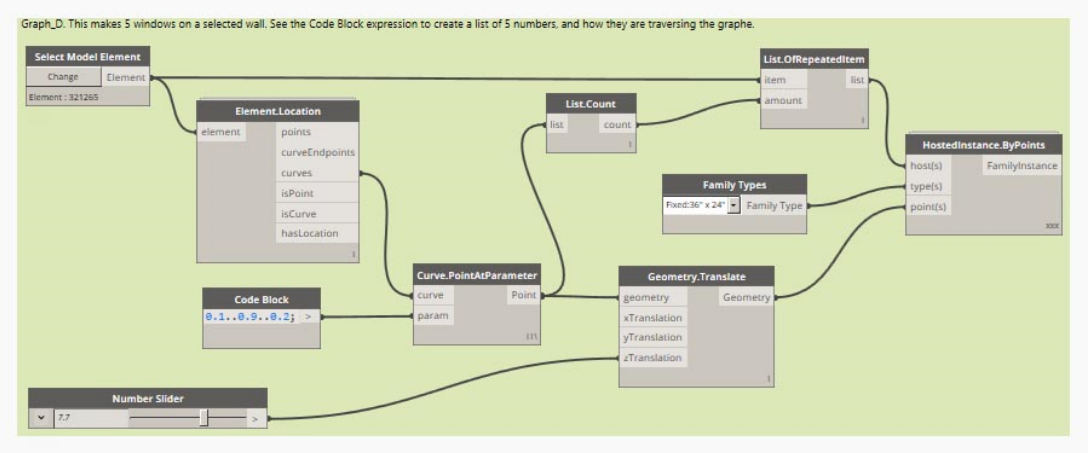
- **[Dynamo] SpringNodes > Revit > FamilyInstances > HostedInstance.ByPoints**
  (input) host(s): Connect output of SelectModelElement (such as Wall)
  (input) type(s): Connect output of Family Types (such as a Door type or Window type)
  You can make your own Window/Door in Revit Family editor and use it here.
  (input) point(s): xyz coordinates of the Window.
  In this example, this coordinates is made from the defining line in the Plan of the Window by **PointAtParameter** node.

- **[Dynamo]** Revit > Selection > Family Types
  (pull-down) Family Type: All Window/Door types from Revit including your custom ones are made available here (e.g. Fixed:16"x24" for a window)
  (output) : This FamilyType is used for input of **HostedInstance.ByPoints**

- **[Dynamo]** Clockwork > Revit > Elements > Element > Element.Location
  (input) element: Connect output of SelectModelElement
  (output) curves: For a wall, this extracts the defining line of the wall in plan.

- **[**Dynamo**]** Geometry > Curve > PointAtParameter
  (input) curve: We use the defining line of the wall in plan.
  (input) param: 0 and 1 represents the ends of the curve, and a value in between represents a point on the curve.
  (output) point: This node creates the coordinate of the point on the curve.

- **[**Dynamo**]** Geometry > Geometry > Geometry.Translate (xTrans, yTrans, zTrans)
  (input) geometry: We just use a point.
  (input) zTranslation: The point will be moved in z direction by the specified amount.
  (output) geometry: The coordinates of the new moved point.



Graph_D: shows how to make a repetition of windows on a wall.
In this example, there are a few tricks.

- Code Block syntax for number series
  The code block in this example creates a list of 5 numbers. Here are the syntax you can use.
  x..y..d; : this creates numbers between x and y with interval of d
  x..y..#n; : this creates numbers between x and y by dividing the difference by n

  * For more details, check DynamoPimer.com

- Repeating the same element in a list
  Each input of **hosts** and **points** of **HostedInstance.ByPoints** node receives a list of 5 elements. For the host(s) input, this is achieved by repeating the same wall 5 times and make a list of them in List.OfRepeatedItems. And the number of repetition is calculated by using **List.Count** and checking the list of points produced by **Curve.PointAtParameter**.

- Lacing option
  Also, another trick is to set the **PointAtParameter** node with the option of Lacing: Longest. This can be done by right-clicking the node and select Lacing:Longest option from the pop-up menu. This will tell this node that list of 5 values given to the parameter input should produce a list of 5 points for output. Without this, the default may just take the first point and discard other 4 points.

  * For more details, check DynamoPimer.com

- **[**Dynamo**]** Core > List > OfRepeatedItem
  (input) item: The point is a wall.
  (input) amount: Number of repetition.
  (output) list: The output is a list of the wall repeated for the specified times.

- **[**Dynamo**]** Core > List > Count
  (input) list: Any list. We input a list of points.
  (output) count: The output is number of elements in the list.