CES226103-L

Dynamo for Civil Infrastructure

Valentin Belets
AECOM

Learning Objectives

- Learn how to set up inputs right, because it's half way to success
- Learn how to maintain the balance between Dynamo intelligence and Revit parametrics
- Learn how to make Dynamo scripts understandable and modifiable
- Learn how to establish solid data workflow from inputs all the way up to outputs

Description

What are the main benefits Dynamo can bring to civil infrastructure modeling? To what extent should visual scripting be used, and to which Revit elements' parametrics? This class will cover these areas and show the application of Dynamo to real civil infrastructure projects.

Speaker

Valentin's bridge engineer career started in 2011. He has worked on major projects in Moscow, Russia, for more than 5 years using AutoCAD at its maximum. Started getting into all things Information Modelling for Civil Infrastructure in 2014, he has been using a combination of Civil 3D, Revit and Dynamo since 2016. Moved to Sydney, Australia in 2017 in pursuit of career development. Valentin learned a lot from Autodesk University classes watching them online and contacting presenters with additional questions. This lab is an excellent chance for him to give something back to this vibrant community.

AUTODESK. UNIVERSITY

Contents

Intro

Using Dynamo in conjunction with Revit seems like the only way to do modelling for Civil Infrastructure (CI). I personally think, and many agree, that modelling for civil infrastructure is always a balancing act of using the right software to the right extent, keeping in mind transferability of the created content at each stage of design development.

Over the last several years I have seen massive Dynamo scripts written, often replicating functionality of professional Civil Design applications. Well, many of us are visual creatures that's why this kind of programming got such popularity in recent years.
In this hands-on lab I would like to share with you an approach which will try and ensure that you are using the right tool for the right job and establish the solid workflow for your everyday tasks.

Bringing the context (terrain, roads, earthworks, under- and above ground utilities) inside Dynamo is usually quite laboursome task, and often, unnecessary. Most of the context can be clearly seen in Civil Design software. Sometimes even dwg plan drawing is sufficient to understand what's going on. Not to mention the whole concept of Information Modelling going gradually through Level of Details, first of which is always a flat plan information.

Speaking from my experience, feeding Dynamo Script with 3D strings exported from a Civil Design software (Civil 3D and alike) causes lots of issues and warning messages in the process of script development. Even if in the end you manage to have a workable solution, chances are it's not going to work in each and every situation with other inputs/other projects. That's why I prefer keeping the model "live" within Civil Design software for as long as possible, and then exporting consistent outputs to feed Dynamo graph with. Dynamo graph should be as clear and understandable as possible and supplemented by smart and flexible Revit content.

In this hands-on lab we will have as an input an Excel Spreadsheet of the following format:

	A	B	C	D	E
1	Pile N	Easting (X)	Northing (Y)	Top of Pile Level (Z)	Wall Height
2					

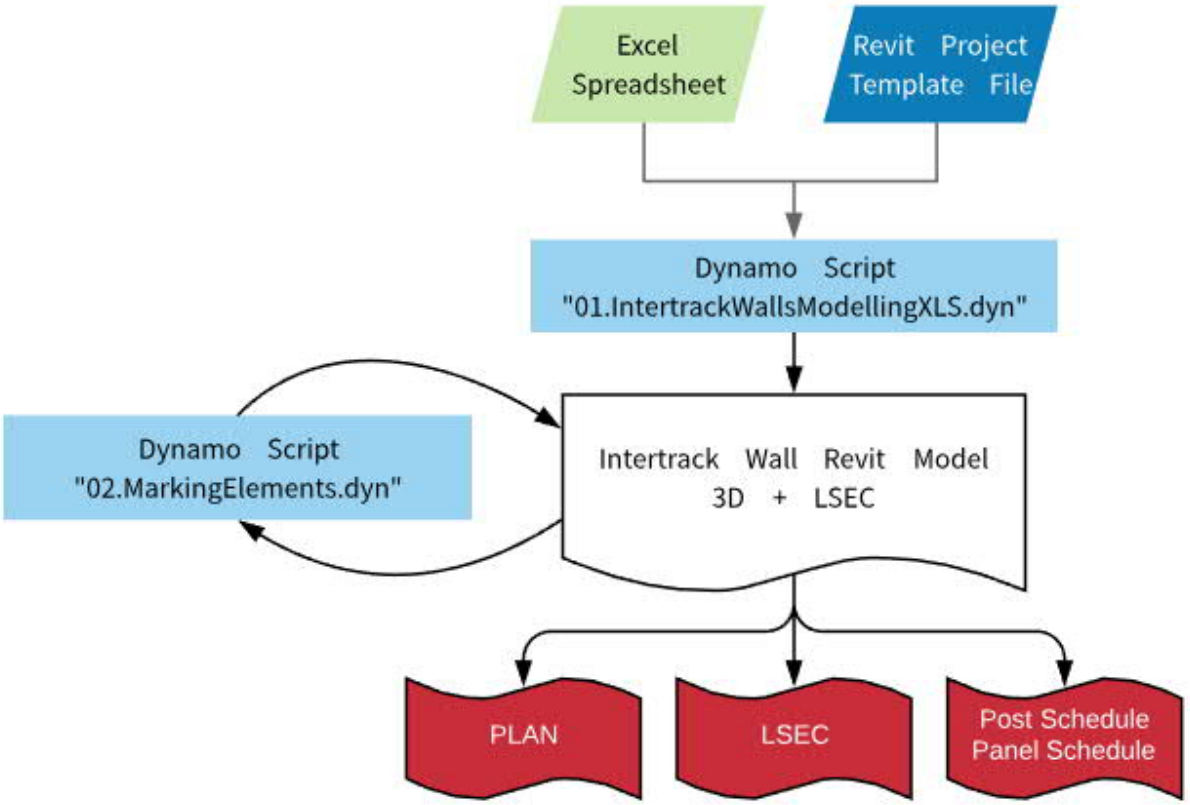
I got this outputs from specific to ANZ region Civil Design software. Getting this outputs from Civil 3D shouldn't be too complicated. There are numerous ways of doing it in Civil 3D, here is just one suggestion:

1. Spread out Pile dynamic blocks (if needed for clash detection check, even with 3D geometry) using path array and then project them onto Top of Pile Design Surface.
2. EATTEXT Easting, Northing and Z value. Half of the spreadsheet done.
3. Projecting the same blocks onto Top of Wall surface and EATTEXTing it will give the rest of information needed.

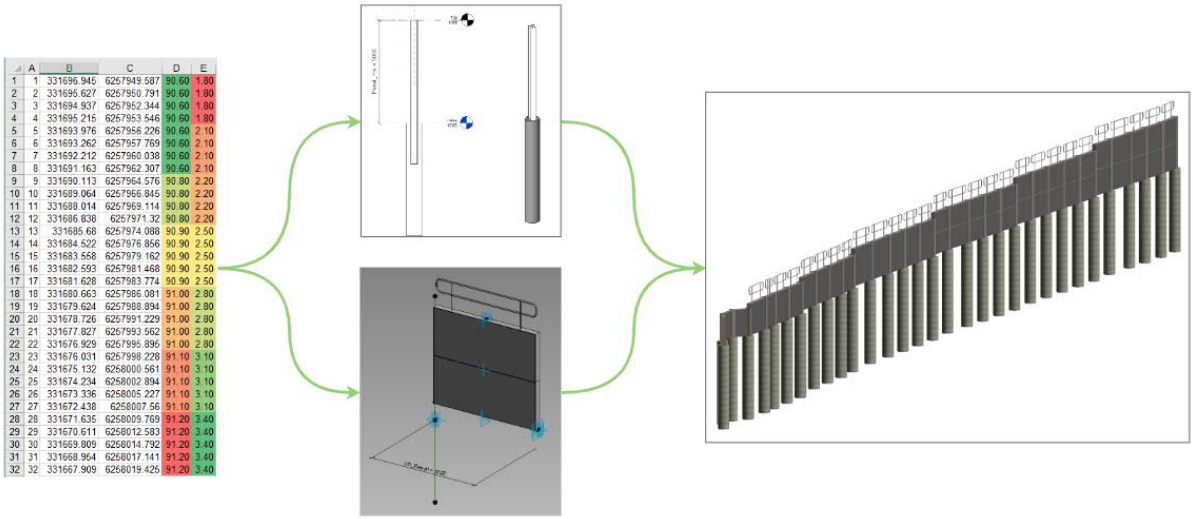
This excel spreadsheet will drive Revit content positioning through Dynamo Script.

Workflow overview

Mindmap of the process.

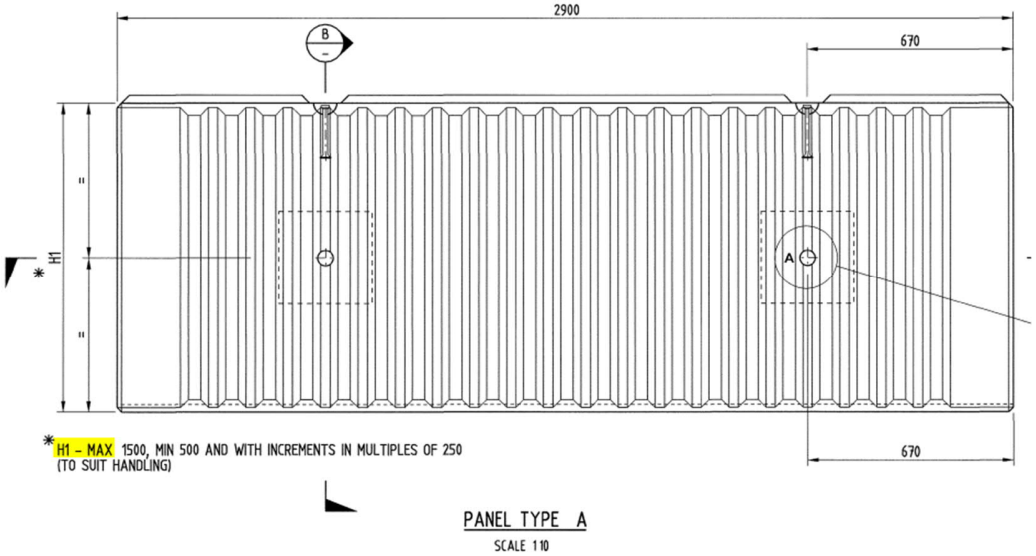


And a more visual representation of it.

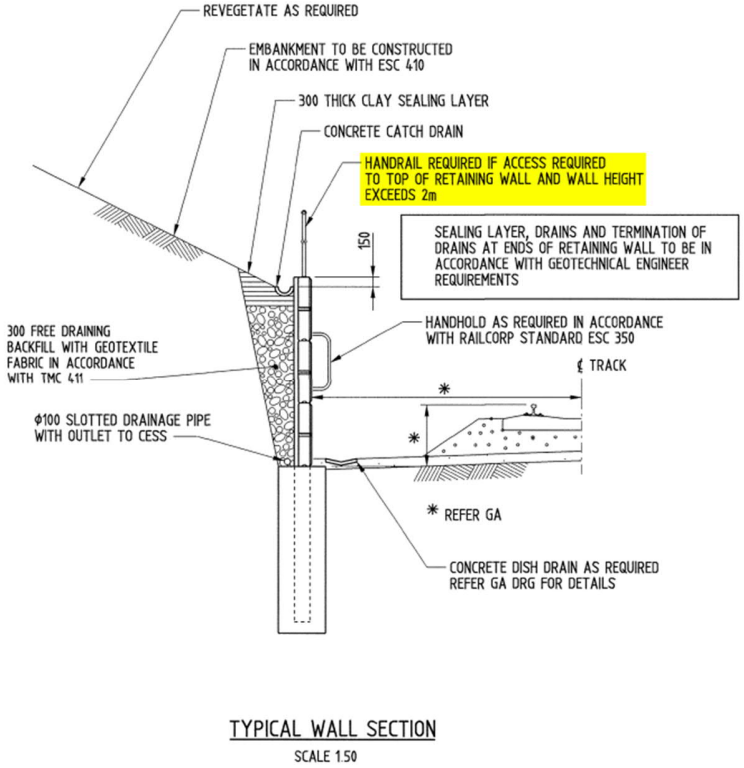


Design constraints

Individual panel maximum height is driven by family type parameter **Panel_Hts_MAX**.



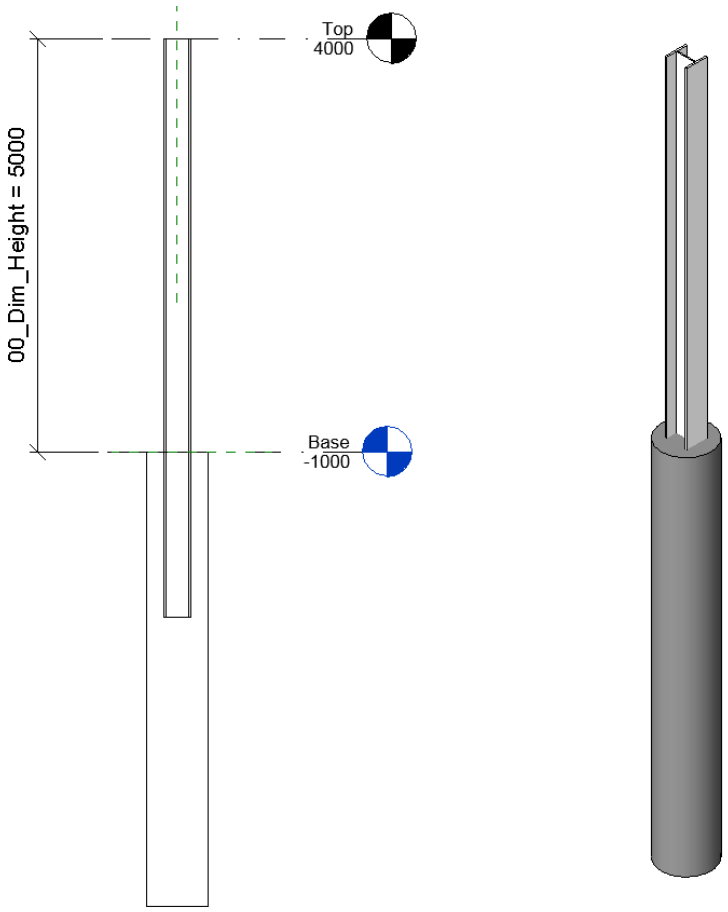
Railing application requirements. Driven by family type parameter **Railing_Hts_MIN**.



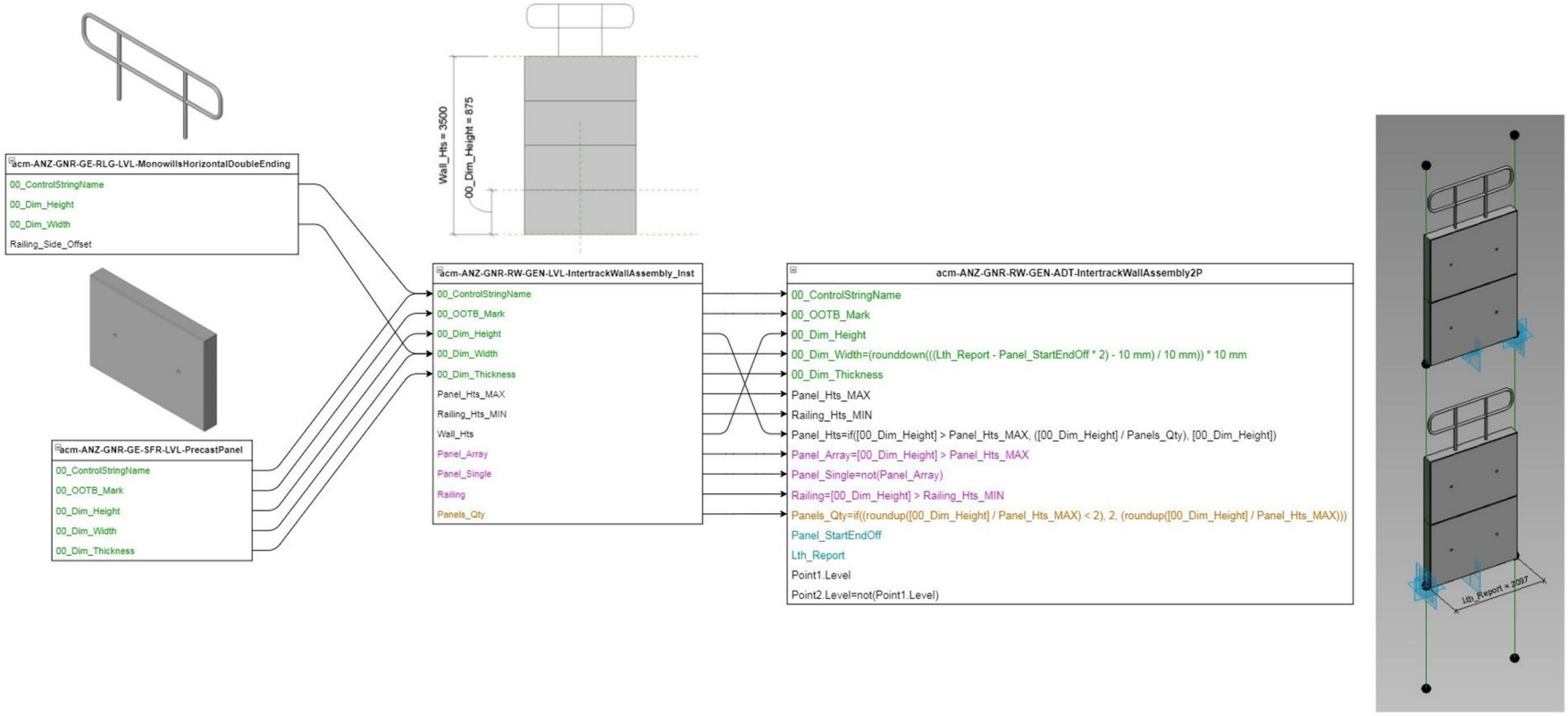
Revit families' creation

Intertrack Wall Column

This family was initially created as a single level based one, with shared parameters driving top of the pile and top of cast-in I-beam levels. This allowed those parameters not only to be scheduled but also to have them in Revit Tags. I thought it was an elegant solution which keeps a dynamic link between element positioning and schedules/tags (The user just need to place them with Offset=0 to Height Datum level). But then I understood that this family won't work in every situation, simply because sometimes these levels might have negative (below Datum) levels. So, I decided to rebuild the family and make it two-level based. I thought I was genius at that moment, cause now I can place a family setting both Base and Top Levels as Height Datum and then assign Top and Base Offset to whatever values I want! Yes, we can even have these values in Revit Schedules! But the negative side of this approach was that the only way the user can tag it is by faking these parameters with duplicate shared ones and then make Dynamo "magic" work to read actual offset values and write them to the fake ones. But I don't think it's a good idea, so I left it as it is.

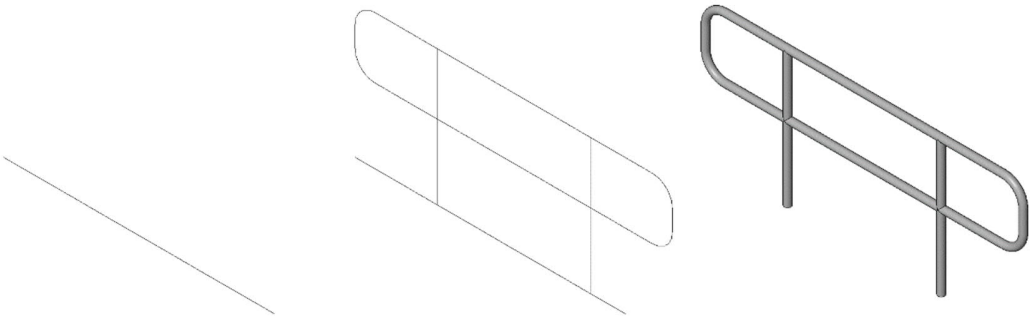


Intertrack wall assembly parameter linking structure

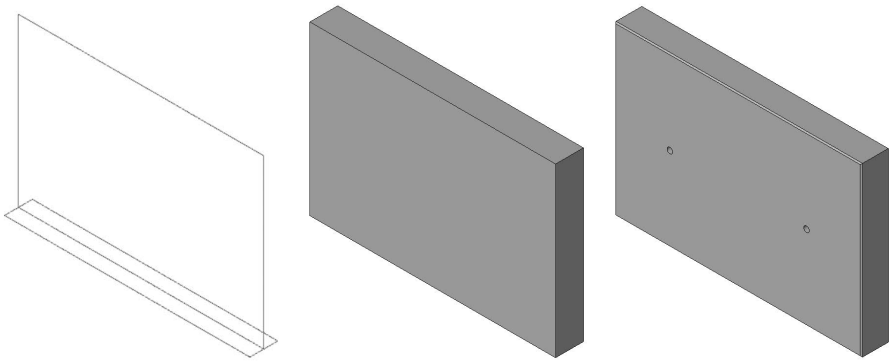


Subcomponents are:

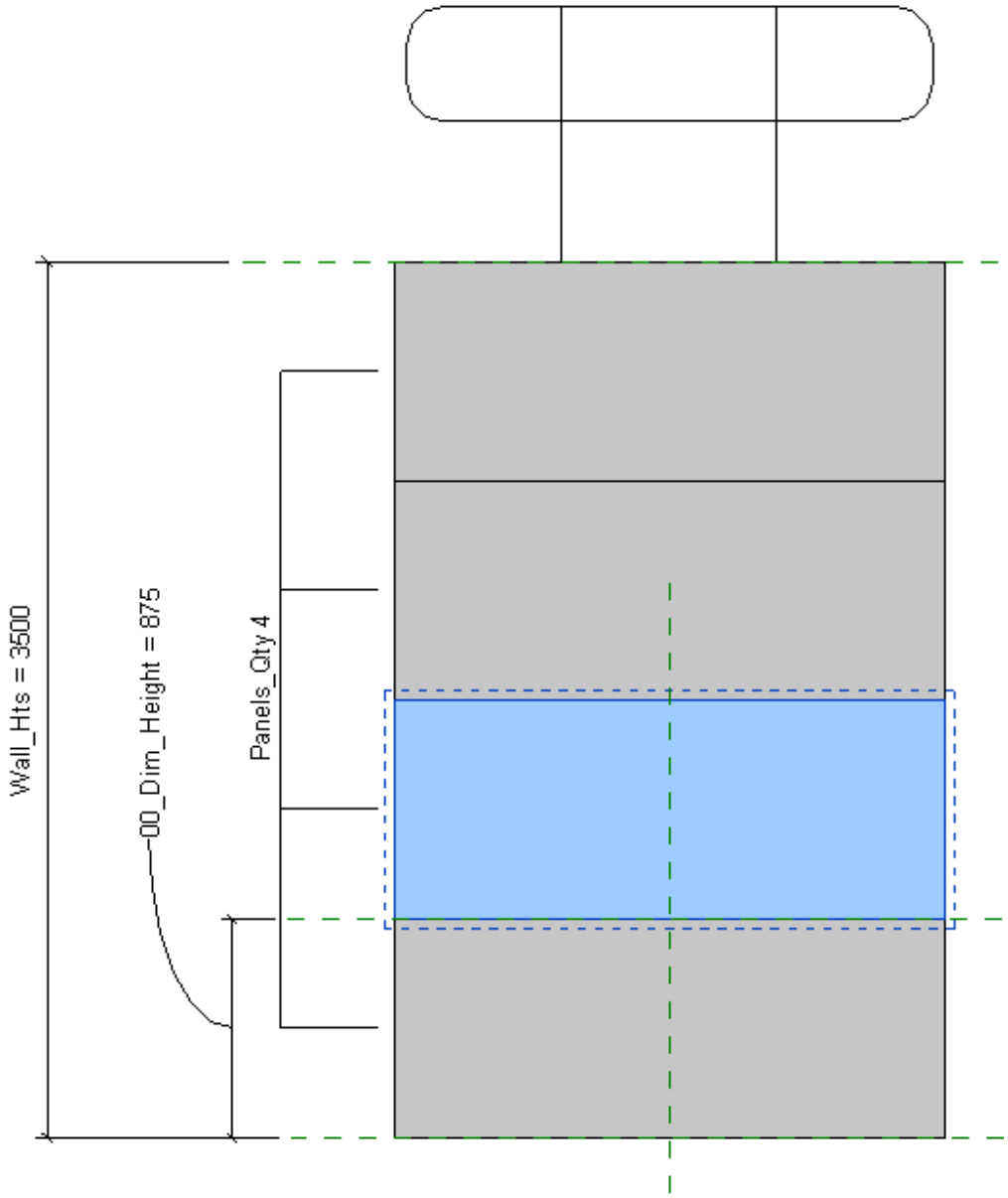
- Railing is a simple family modelled in different Levels of Detail.



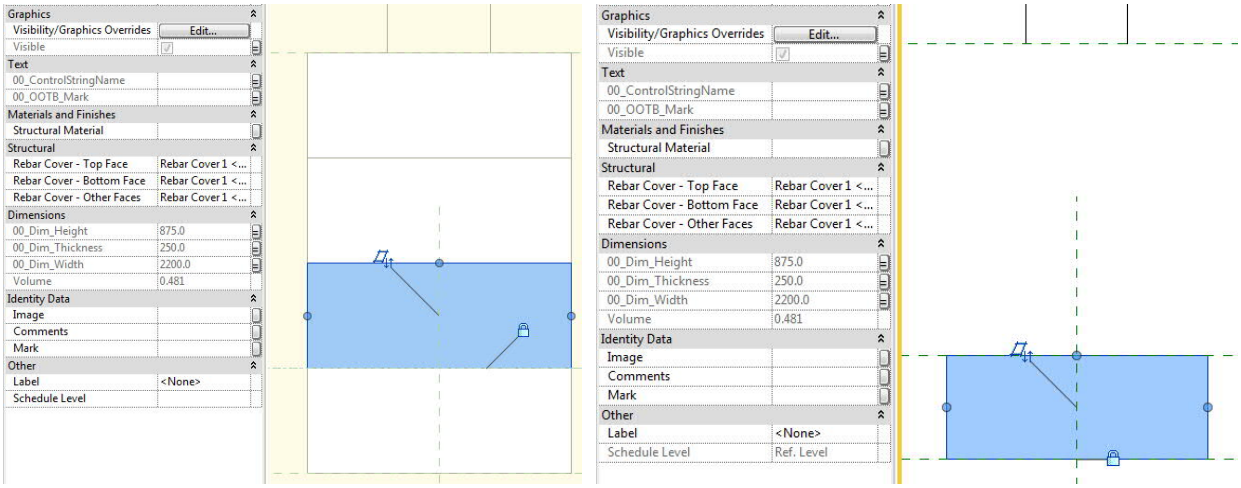
- Precast panel, also modelled in different Levels of Detail.



The next step is to nest panel and railing into a Generic Model family and create an array of panels with the railing on top. All the available parameters of these nested families should be “mirrored” with the ones in this assembly family.

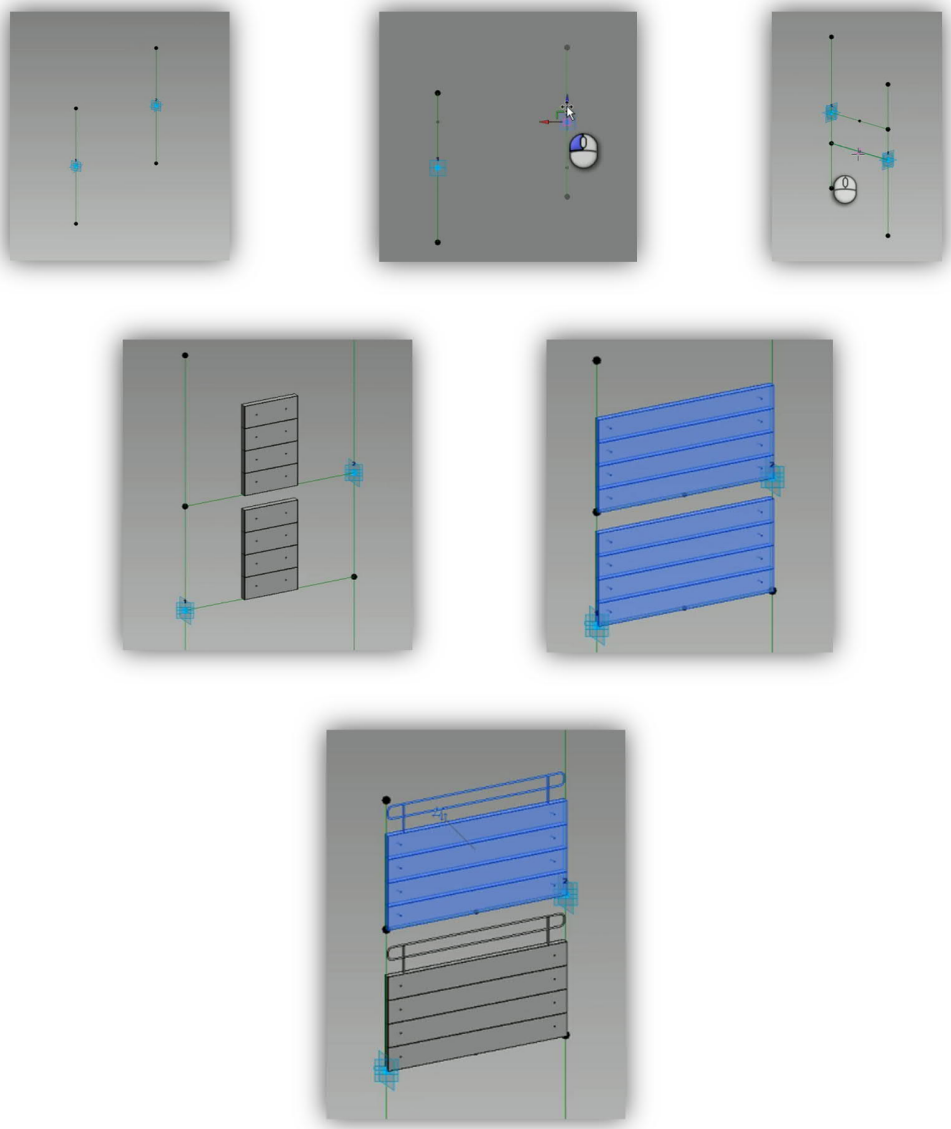


As always with arrays in Revit, Single panel family is placed in addition to array one so that the family doesn't break in single panel scenario.



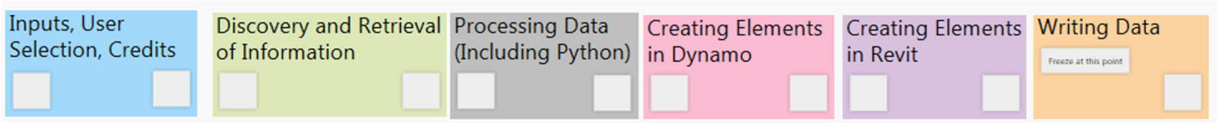
Next step is to assign correspondent visibility parameters to single panel, array panel and railing.

This panel assembly then is imported into 2 points adaptive family. Here is the screencast showing how it's done:
https://knowledge.autodesk.com/community/screencast/57d84de4-017c-4b34-9474-e344bc3a4c4c



Resulting family adapts to setout points and have main design and manufacturing criteria assigned via Type Parameters. This allows design inputs to be easily changed even after the model is built (providing plan setout stays the same). If the plan setout changes, then it's always better to get back to Civil Design application and re-do the arrangement simply because there are too many things designer needs to take into consideration, which none of the scripts out there can capture and interpret.

Dynamo script overview



Dynamo Graph Template

Each input is marked with the input type and has prefix in its name denoting in which part of the script it is used. Numeric naming convention is used for data processing and manipulation. Alphabetic – for placing Revit families and setting their parameters based on calculations.

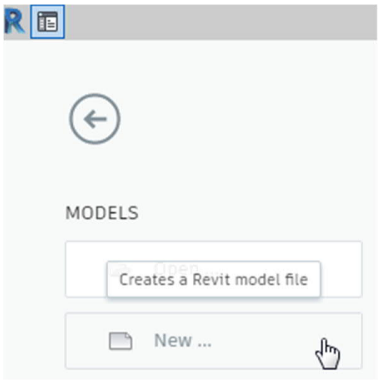
The main processing parts are:

- 01_Getting data from Excel;
- 02_Transforming World Coordinate System into Project one;
- 03_SetOut Points from XLS;
- 04.1_Extracting Control String information from the file name;
- 04.2_Pile Number from Excel to Mark family parameter;
- 04.3_Python Script_^_Post Rotation Calculation;
- 04.4_Panel Placement Determination;
- 04.5_Panel Height Calculation;
- 04.6_LSEC Processing;
- 04.6_LSEC_SetOut points;
- A_Panels placement_3D;
- B_Posts Placement and rotation_3D;
- A_Panels Placement_LSEC;
- B_Posts Placement_LSEC.

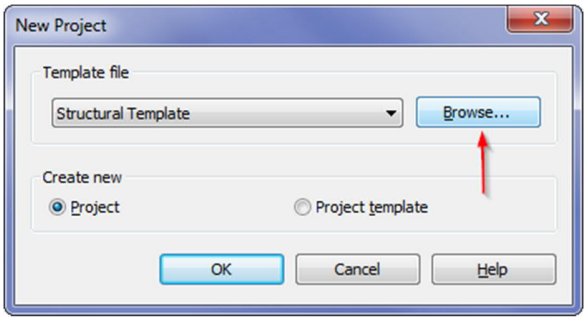
Dynamo script is built so that it should be able to guide the user through it by itself, and you don't need to move your eyes away from the script. The following pages are for those who are not yet comfortable enough with either Revit or Dynamo or both of them.

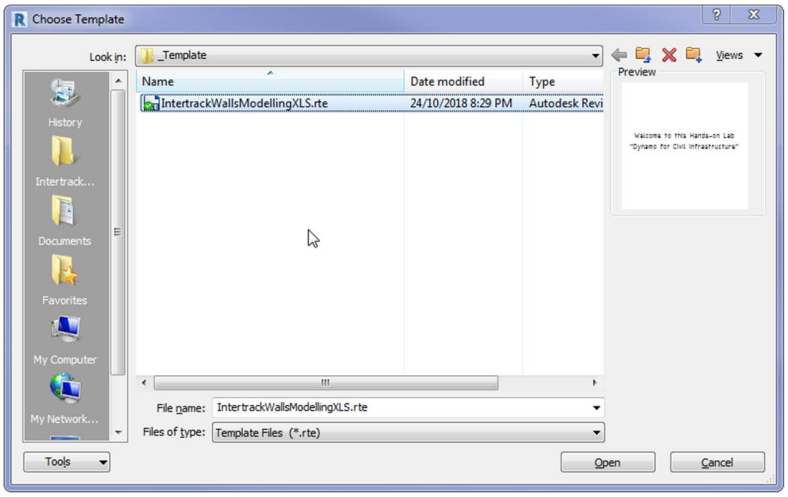
Getting started with the script

1. Open Revit, hit “New” on the home screen:



2. Browse for a template file in the DataSet:

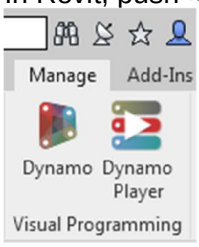




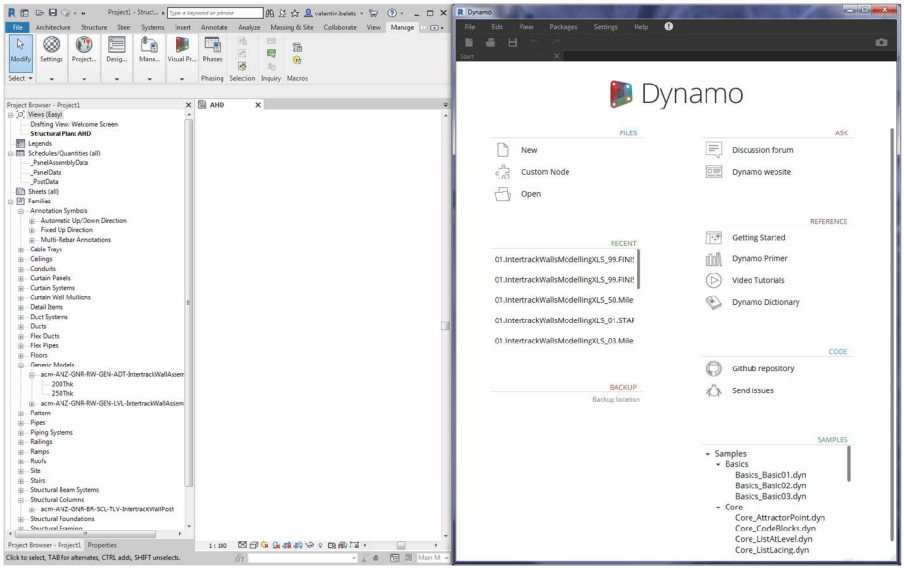
Hit “OK”.

3. Save the file in the root folder of the dataset, if possible give it a meaningful name.

4. In Revit, push ⊞ + Left Arrow. Then go to the Manage tab in the Ribbon, open Dynamo.



5. Dynamo window might be off the screen. Find it by pressing ⊞ + Right Arrow several times.
6. The result should be as following (both Revit and Dynamo windows are on the screen):

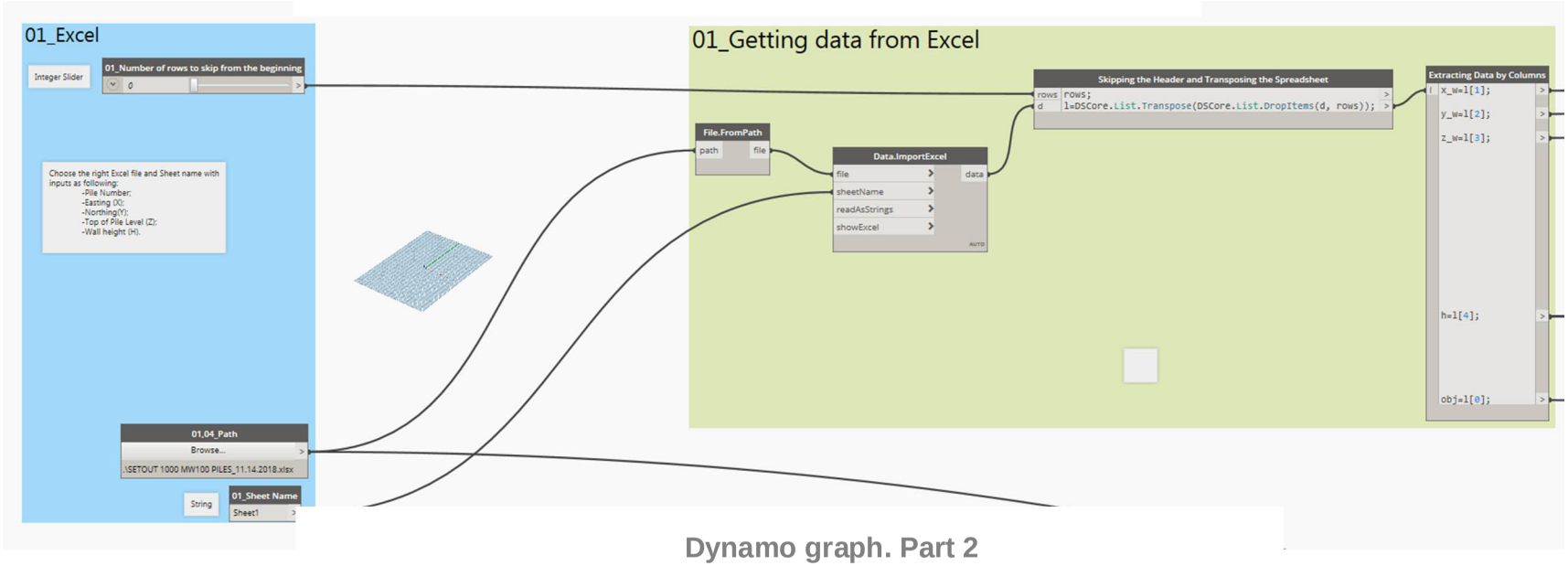


7. Now we are ready to open the script itself. Which is named the same as the Revit template file, just with the suffix “START” and located in the DataSet root folder.

Note: Depending on your level, you might choose to open the one with suffix “FINISH” for a complete script or “FINISH_FROZEN” to be able to follow along thawing nodes and exploring the interim results the nodes are outputting.

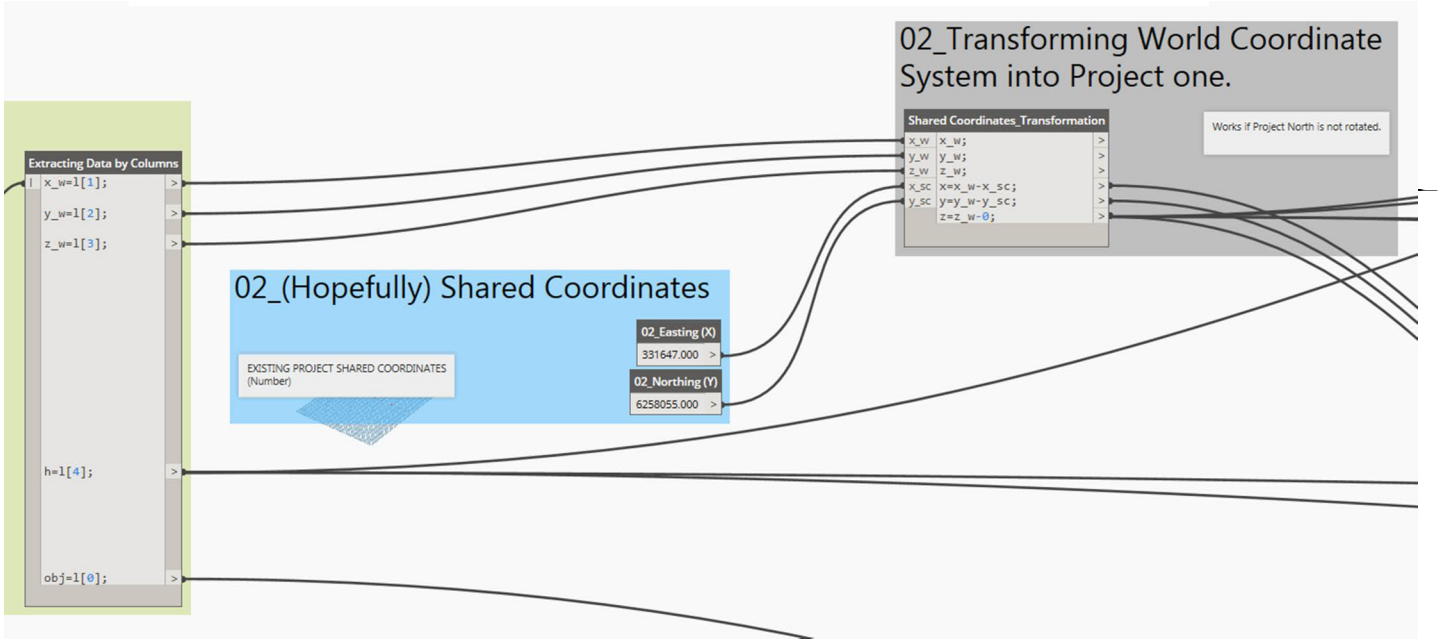
Dynamo graph. Part 1

Getting data from linked Excel file with Setout Information, skipping the heading rows if needed and extracting information from the columns.



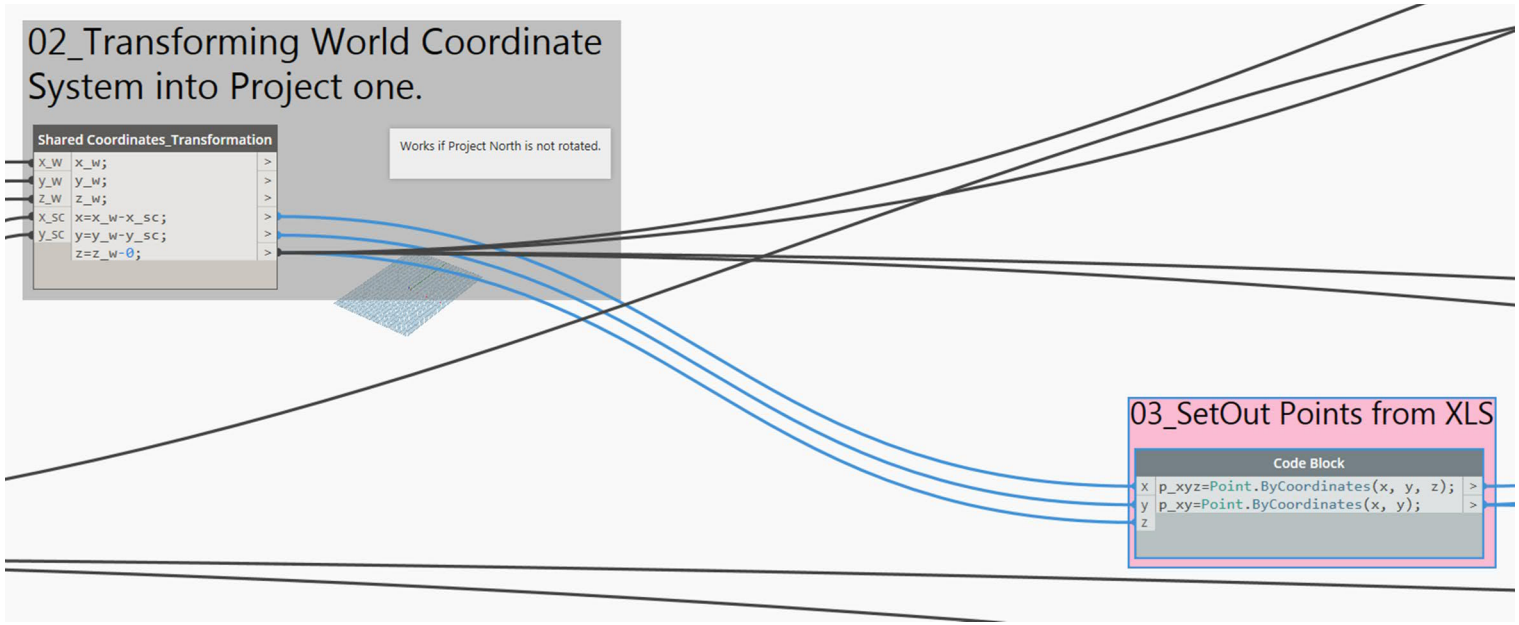
Dynamo graph. Part 2

Transforming real-world coordinates into local coordinates within Revit file.



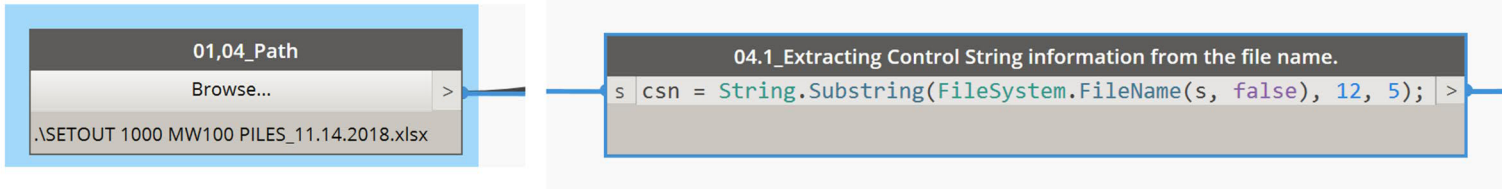
Dynamo graph. Part 3

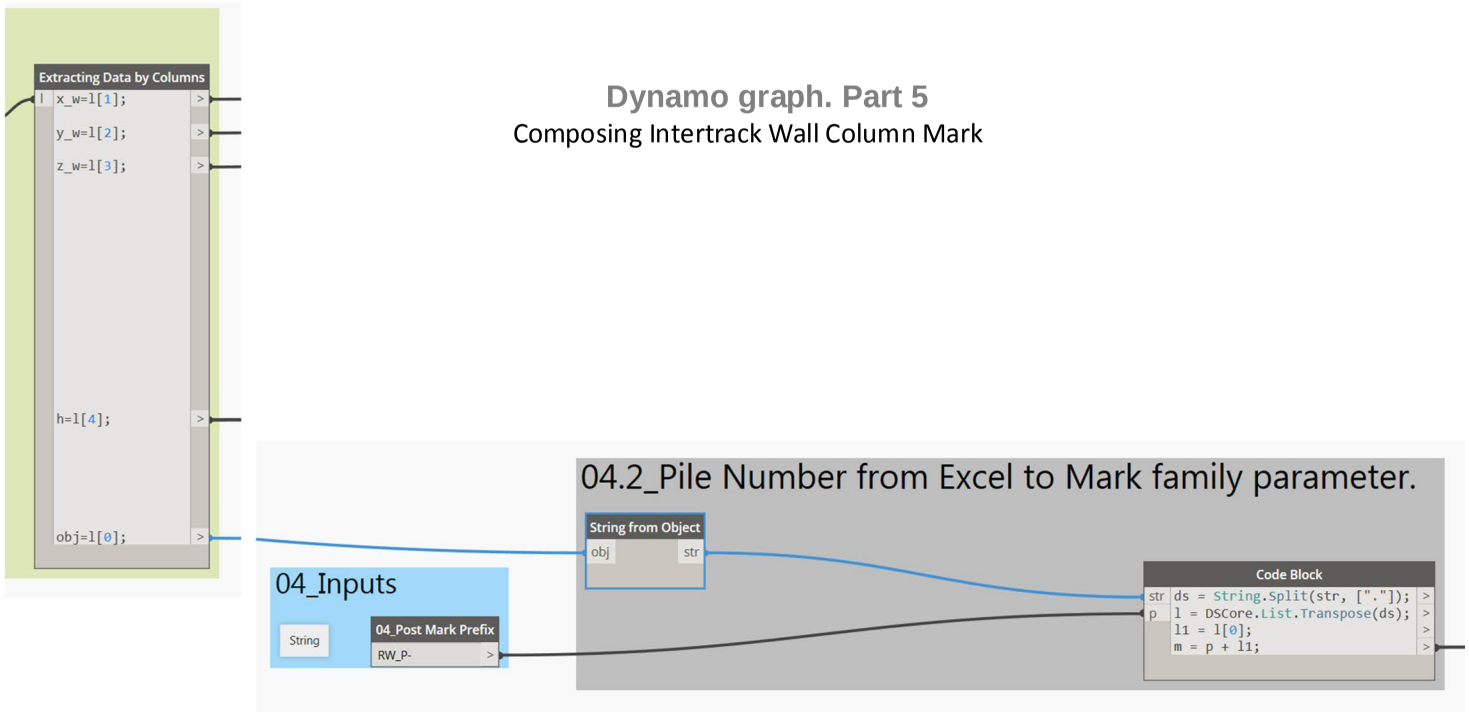
Creating Dynamo Points. Flattened version and 3D.



Dynamo graph. Part 4

Extracting Control String Name from the input file name. Yes, it can be that simple.



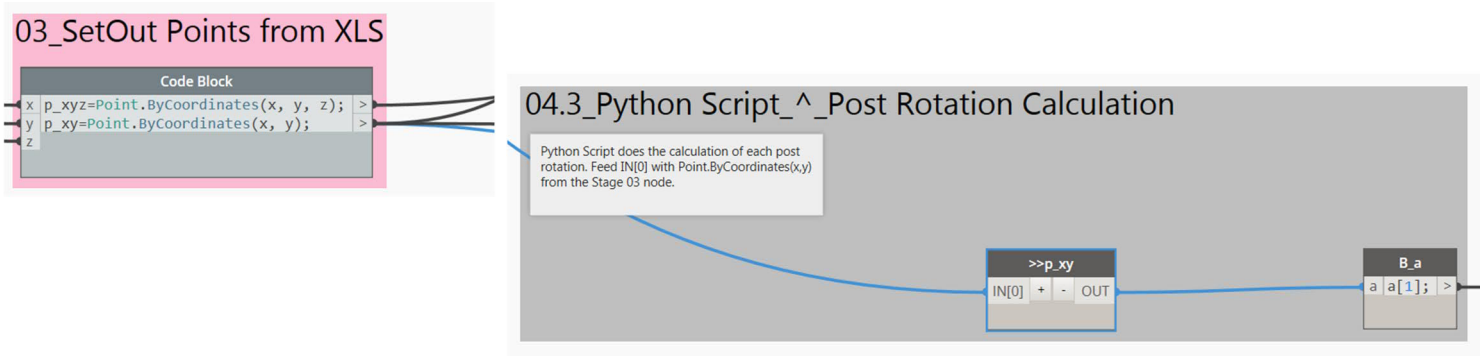


Dynamo graph. Part 5

Composing Intertrack Wall Column Mark

Dynamo graph. Part 6

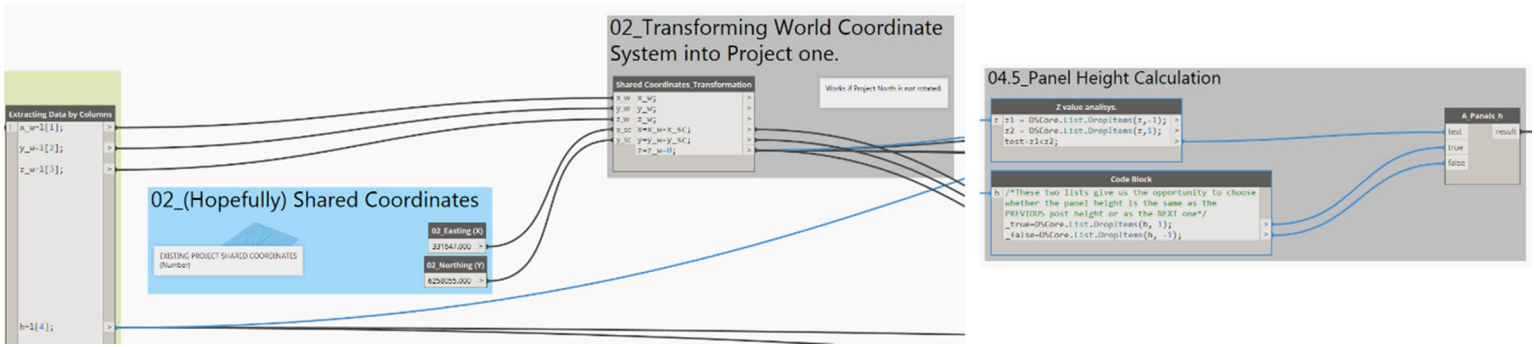
Python script calculating each post rotation so that both coming and the following panels can fit inside the I-beam.



AUTODESK. UNIVERSITY

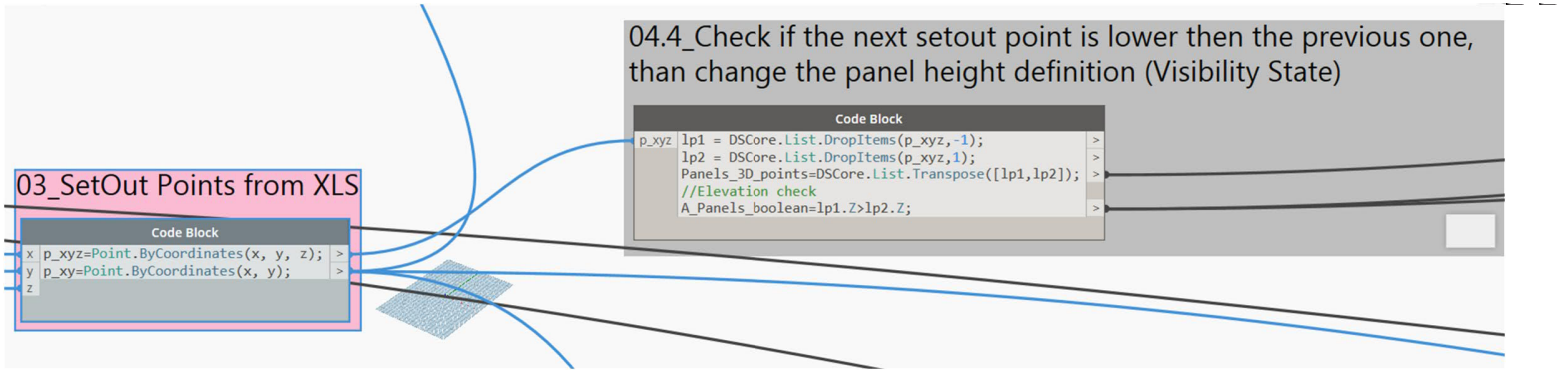
Dynamo graph. Part 7

Panel height not necessarily equals the previous post height. These nodes assign panel height sequence of setout points going, whether it is ascending or descending.



Dynamo graph. Part 8

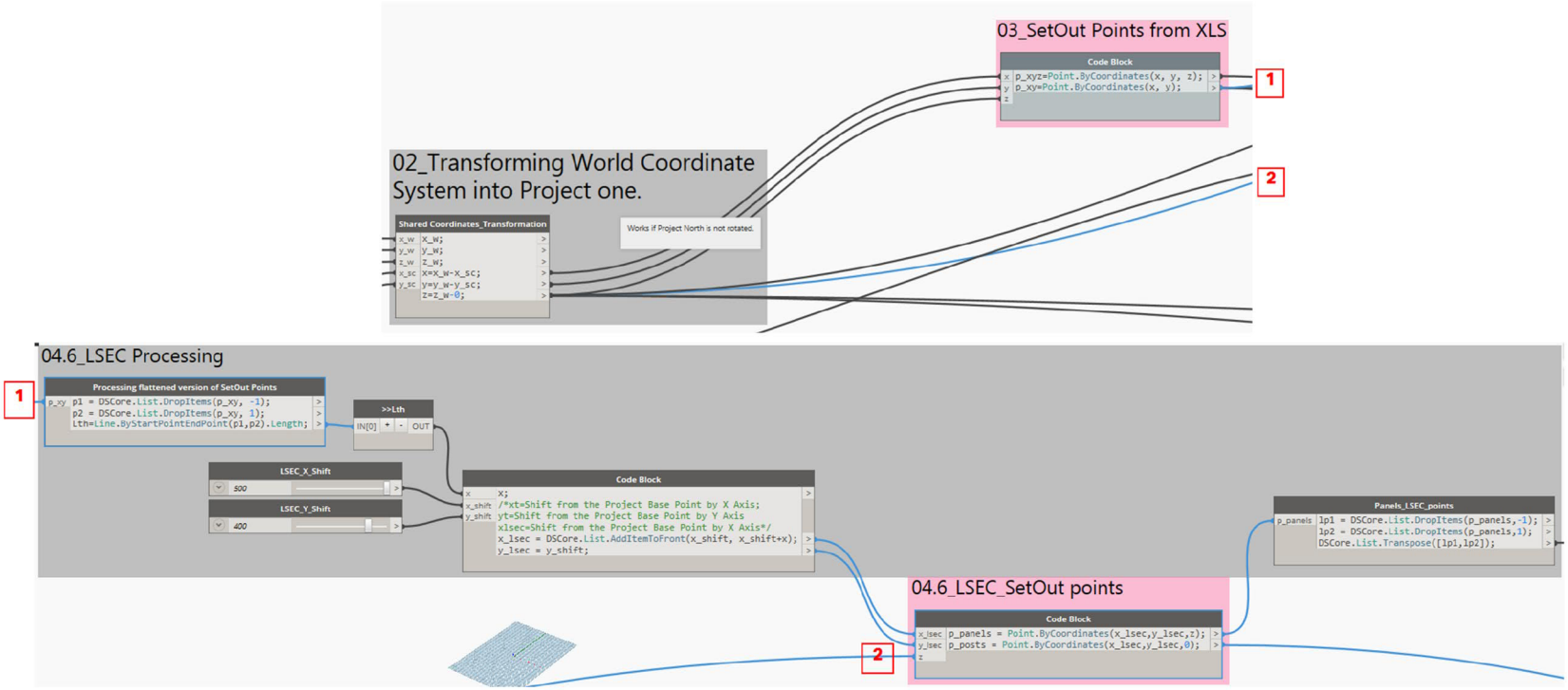
These nodes do similar thing to the previous one, but the resulting output defines whether the panel should “sit” on the previous post or on the next one.



AUTODESK. UNIVERSITY

Dynamo graph. Part 9

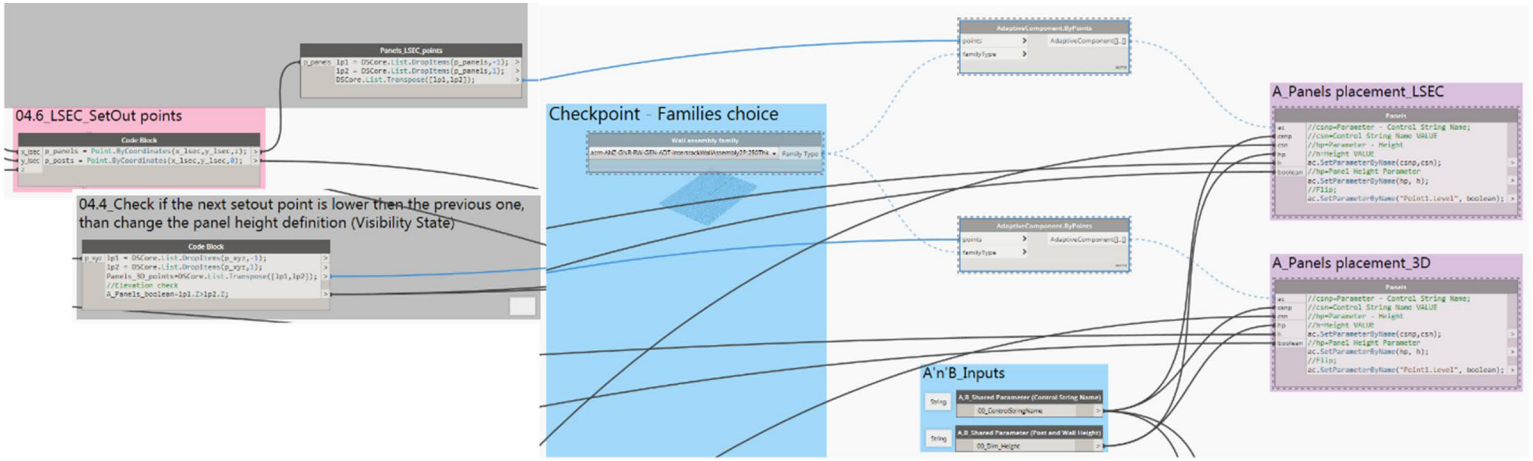
Building LSEC taking the distance between setout points and then placing the same components along the line. I've been told that Revit absolutely loves this part.



AUTODESK. UNIVERSITY

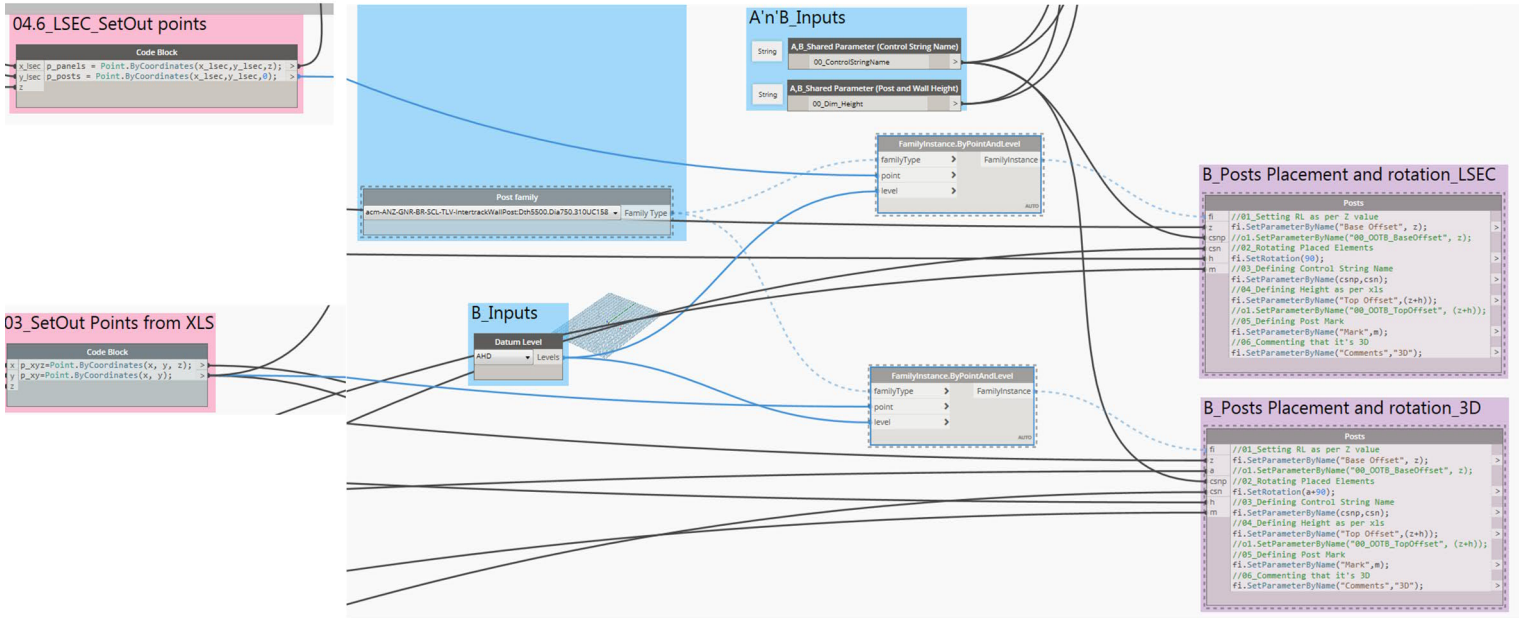
Dynamo graph. Part 10

Feeding the final Code Blocks with all the processed inputs.



Dynamo graph. Part 11

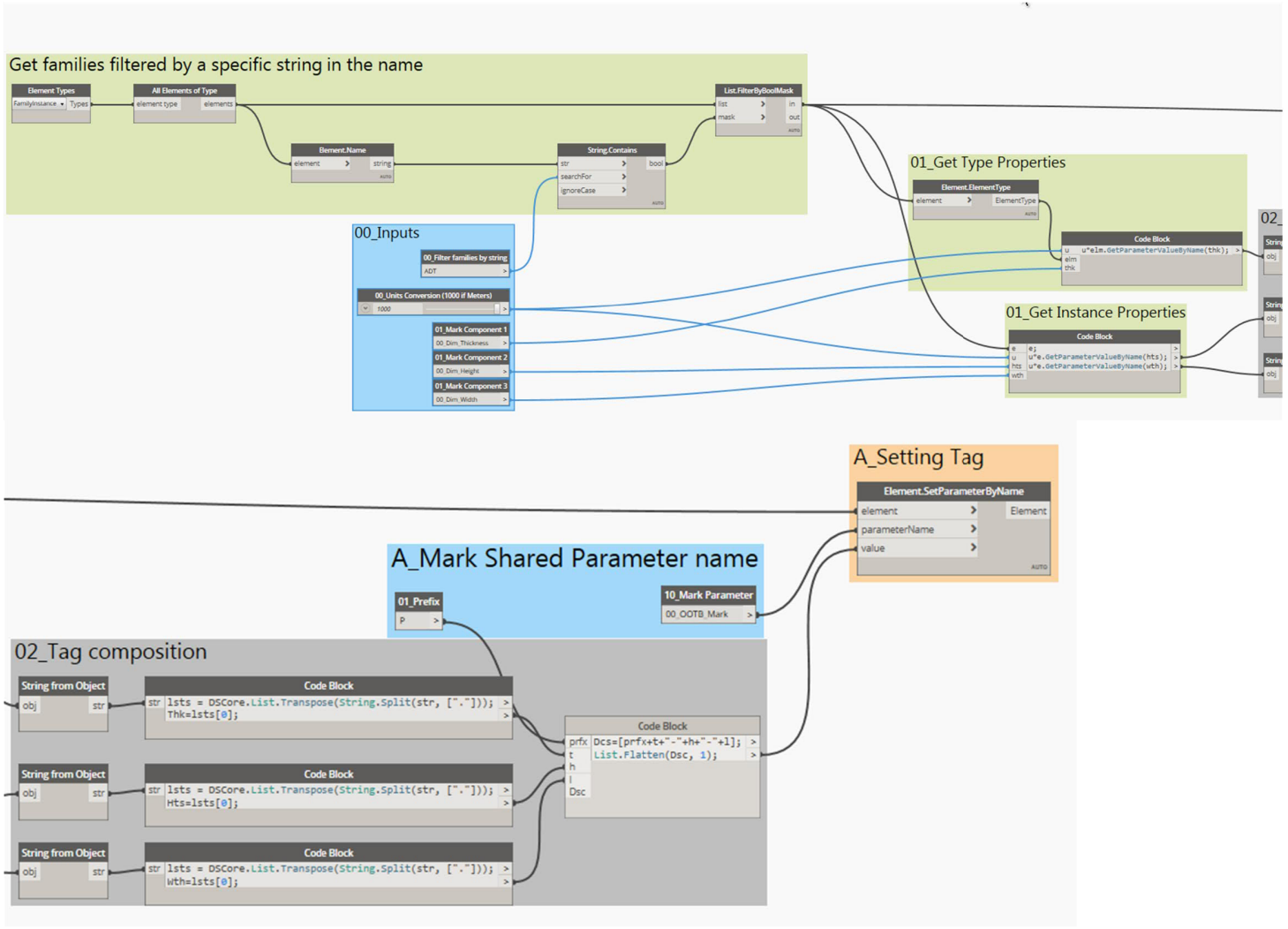
Same here.



AUTODESK. UNIVERSITY

Dynamo Graph “Marking Elements”

It’s a separate dyn file which optionally can be executed to mark elements based on their geometry.
00_**OOTB_Mark** parameter is used in order to mark individual panels assigning the mark to adaptive assembly family.



AUTODESK. UNIVERSITY

Conclusion

In my opinion, visual programming is a much broader definition than it sometimes considered to be. It's also intelligently built Revit Families which reflect some crucial Design Codes or Manufacturing constraints. A properly built family is the one which is programmed to follow design standards, minimizing human error in tweaking its parameters.

Visual programming is so infectious that it's too easy to get carried away by it. For example, panel height calculation or railing application could have been processed through Dynamo script, but then chances are Revit will feel disappointed and will start crashing simply because the one is not using its strengths accordingly.

What potentially could be done with Dynamo is analysis of the setout and suggesting better post layout and heights to minimize overall number of panel sizes, so that it will be cheaper and faster to manufacture them. It is worth developing this sort of algorithm for a massive scale project, which might be just around the corner.

To wrap this all up, maintaining the right balance between the range of tools and their strengths is a much more complex, but at the same time much more rewarding thing to do. Hopefully this lab has given you another way of looking at Information Modelling in this day and age.