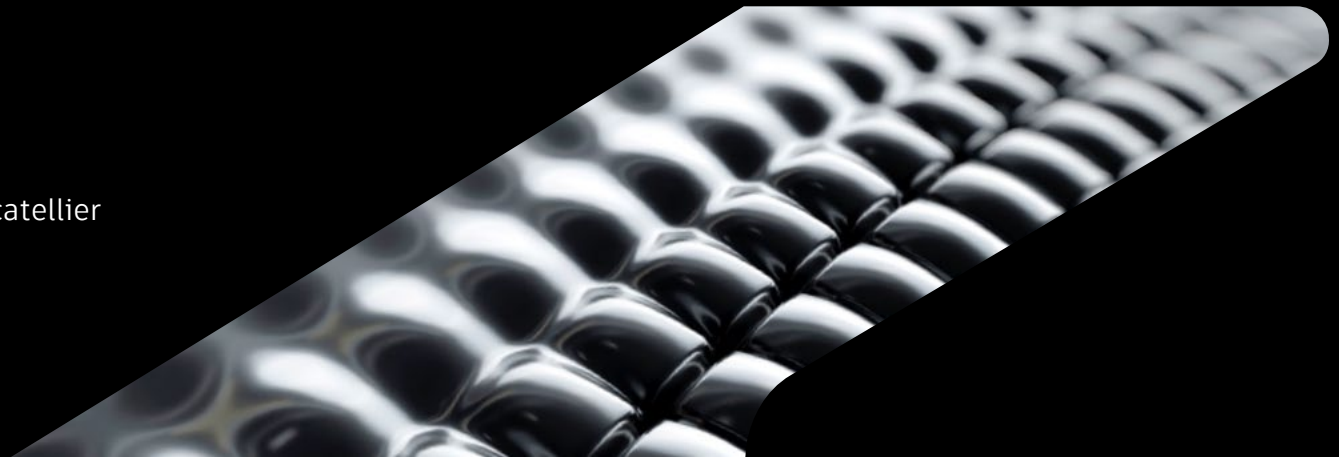
AUTODESK UNIVERSITY

Dynamo & Curtain Panels – A Wild Brick Pattern Workflow in Revit

Nicolas Catellier
Architect + BIM Specialist| @nicocatellier



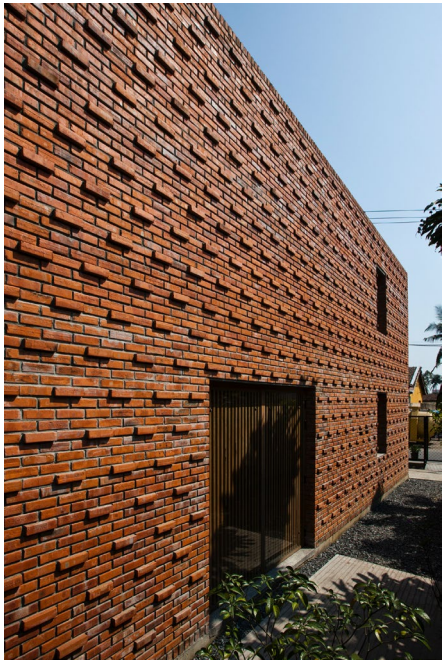
Nicolas Catellier – Revit Pure & BIM Pure

- Architect
- BIM Specialist
- Atelier 21 Architects (2012-2021)
- Founder & BIM Specialist at BIM Pure Productions (2021-…)
- Founder of revitpure.com (2016-…)
- Host of “Revit Pure Live”
- Speaker at BILT + AU
- Based in Quebec City, Canada

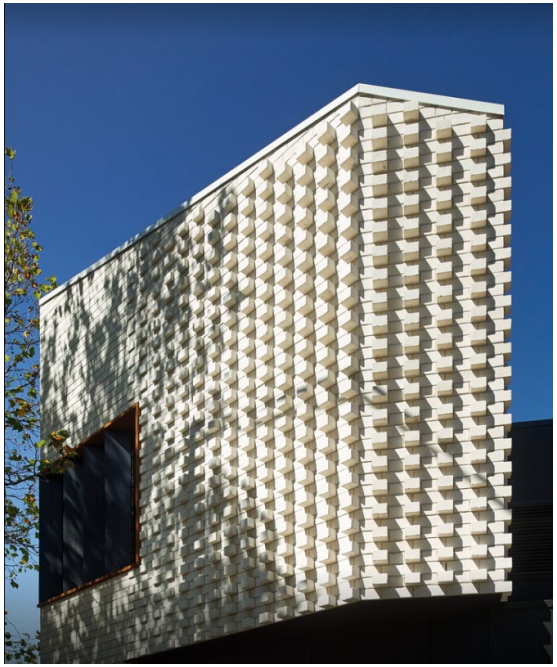
Bricks, Housing and Dynamo

- This presentation is based on an ongoing project I had the chance to work on.
- “Place Griffintown” housing project in Montreal by L’OEUF architects.
- Ambioner, L2C, Équipe Laurence, les Ateliers Ublo, Accès-Logis.

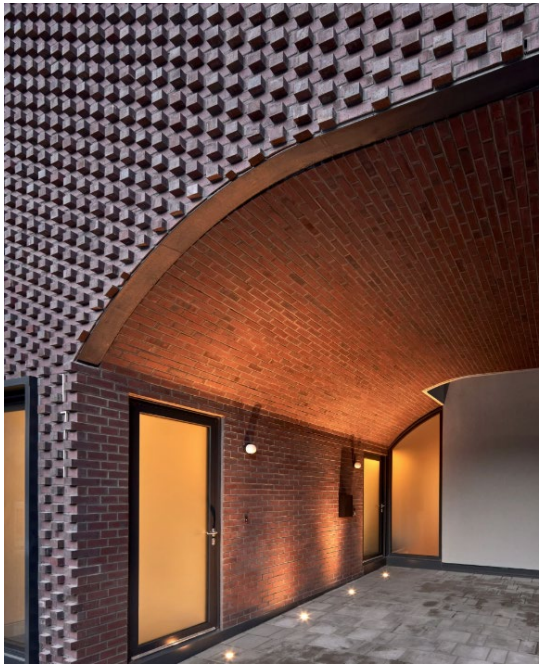
Examples of Extruded Brick Façades



Termitary House by Tropical Space

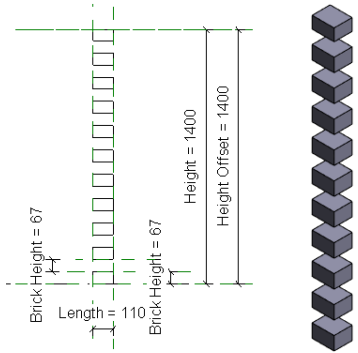


Little Brick Studio by Studio Bright

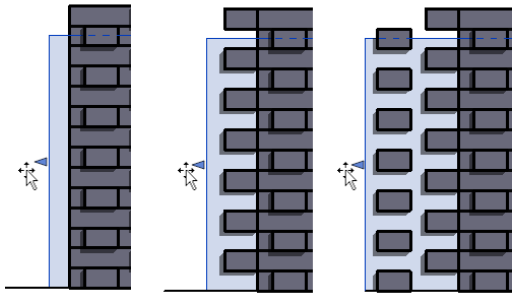


High Park Residence by Batay-Csorba Architects

Today We'll Talk About:



Creating a brick array family to place on walls.



Create a curtain panel brick family to prepare for automation.

Generate a brick façade design based on an image gradient

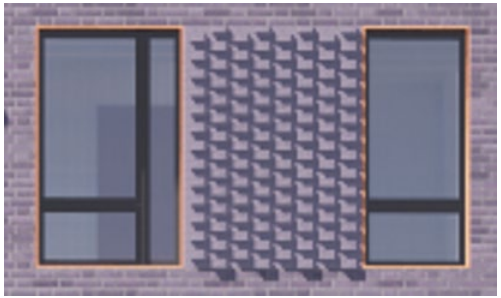
Download the Dataset

- Download the full dataset: https://revitpure.com/au2022

- Includes: families, sample file + Dynamo script.

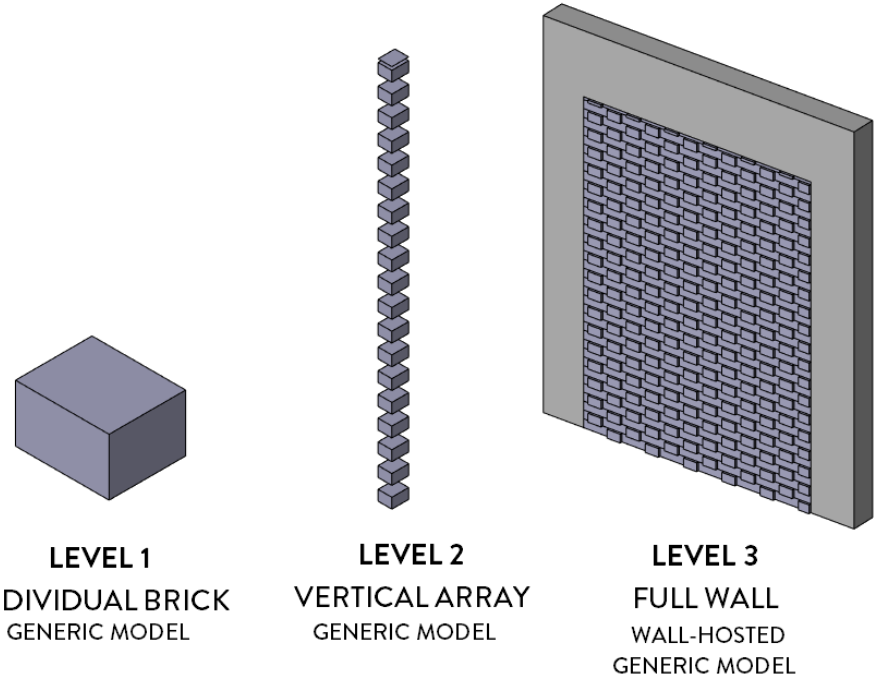
Protruding Brick Pattern: Generic Family

- First challenge: creating a rectangular-shaped brick pattern to place between windows.
- Workflow to avoid: model in-place.



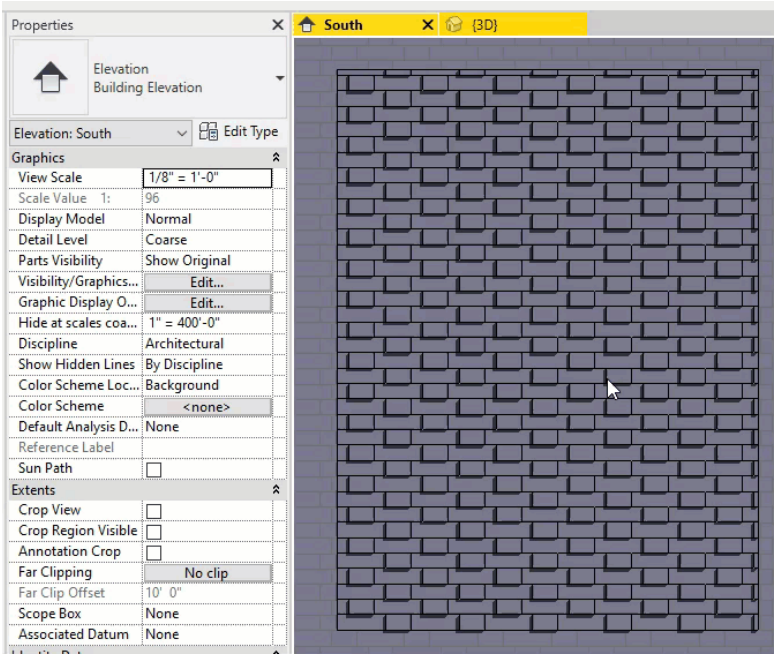
Brick Pattern: the Nested Family Technique

- This workflow is based on a generic model family.
- It requires three levels of nested families.
- To use arrays, nested families are better.



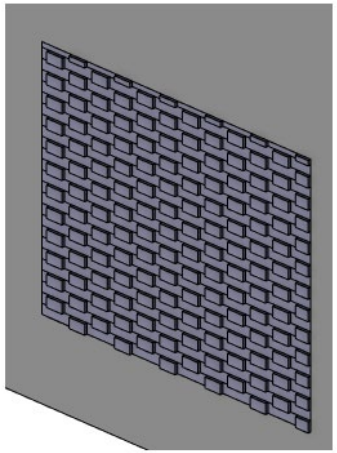
The End Result: Parametric Brick Wall

- This family is hosted on a wall.
- The height and width can be changed.
- Limitation: the origin is at the bottom left. That means full bricks will be placed there. Brick at the top and right side will be cut.

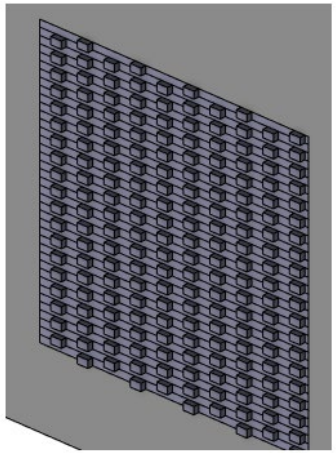


Customizable Brick Dimensions

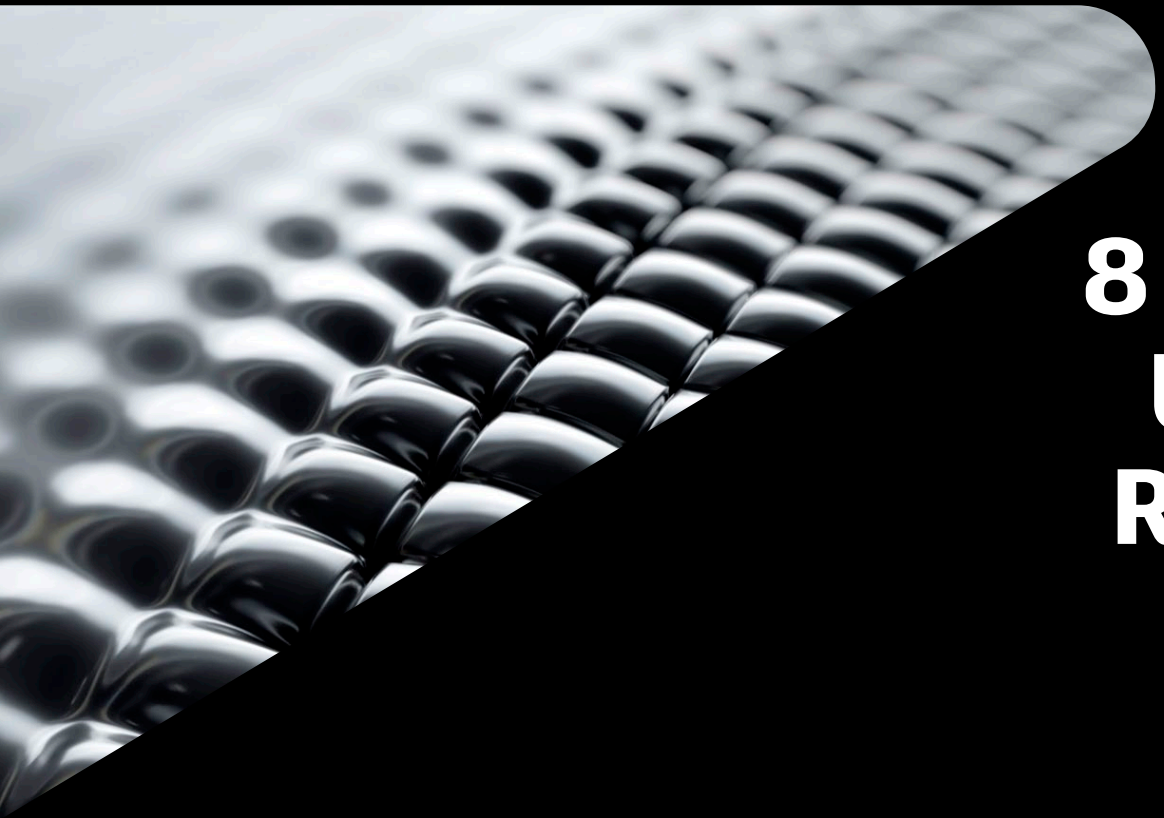
- This family is not limited to a specific brick dimension.
- You can change the dimension of the brick on the type properties.
- Limitation: the family is slow, caused by the arrays and 3-layers of nested families.



Dimensions	
Brick Depth	120.0
Brick Height	67.0
Brick Offset	19.0
L1	210.0
L2	110.0



Dimensions	
Brick Depth	120.0
Brick Height	50.0
Brick Offset	38.0
L1	300.0
L2	75.0



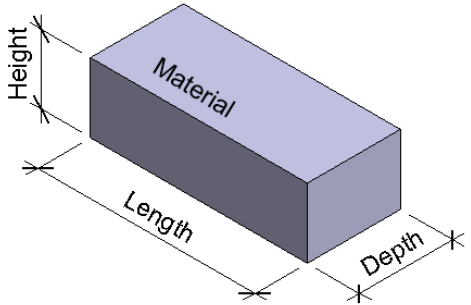
8 Principles to Use Arrays in Revit Families

What is an Array?

- An element that is repeated a number of time. In this context, the number of elements is determined by the family's length.

1- Use a Nested Family

- Trying to array a geometry created directly in the family can result in bugs.
- When there are multiple dimension parameters, it is better to have a nested family of the “single” element.

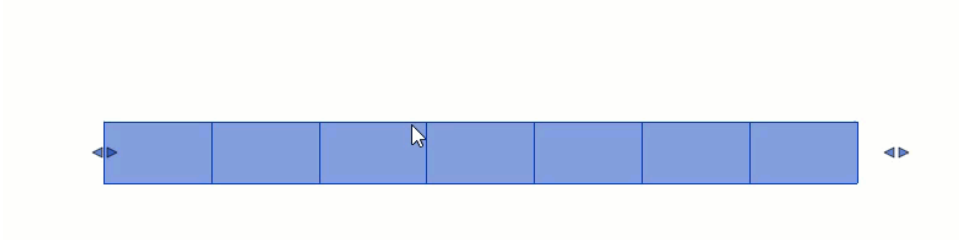


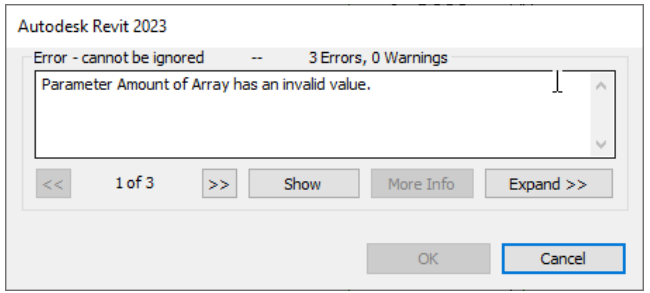
Materials and Finishes	
Brick Material	Brick-material
Dimensions	
Height	67.0
Length	210.0
Depth	90.0

SINGLE BRICK FAMILY

2- Number of Elements in Array Must Be ≥ 2

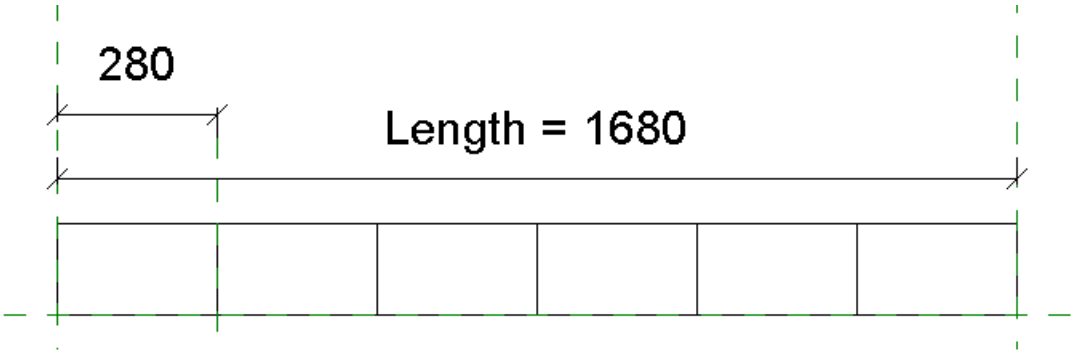
- In other words: an array of 1 or 0 elements won't work and will break the family.
- Possible workarounds with formulas and visibility parameters.





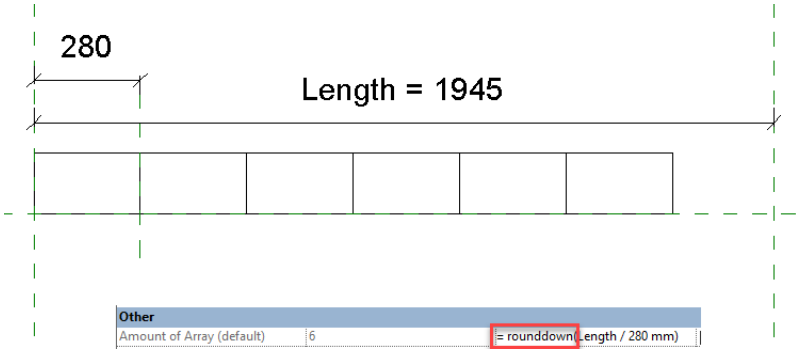
3- Basic Array Formula: Total Length/Element Length

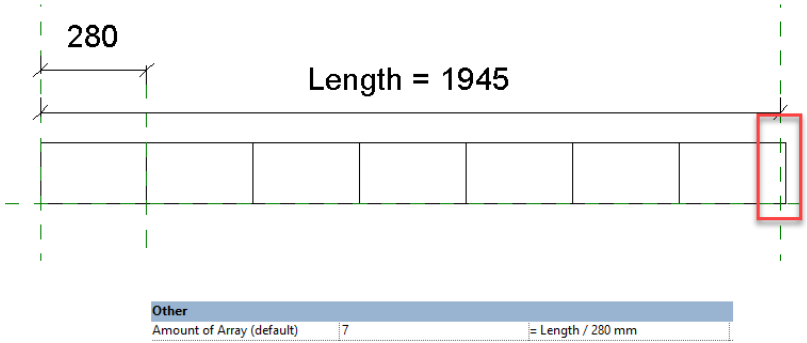
- The “Amount of Array” value will be rounded. For example, 5.3 will round to 5 and 5.8 will round to 6.



4- Use *rounddown* to Avoid Exceeding Length

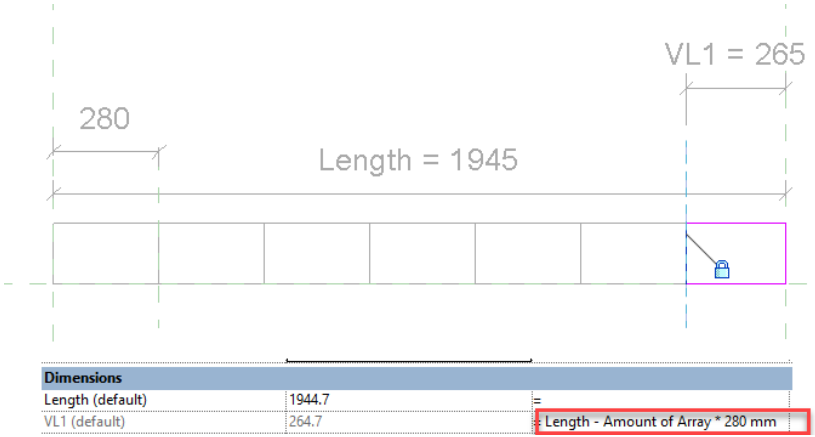
- Adding *rounddown* in the formula ensures the family won't exceed the length.
- For example, if Length/Element Length = 5.9, it will be rounded down to 5.





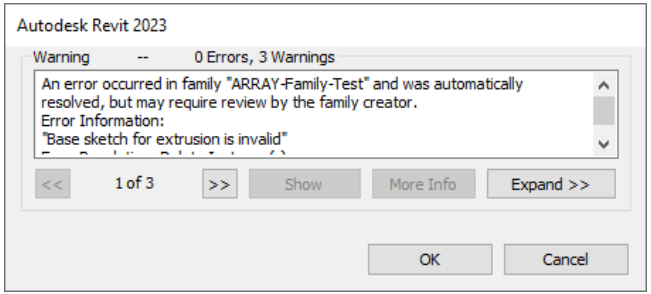
5- Add a Filler Element with Variable Length

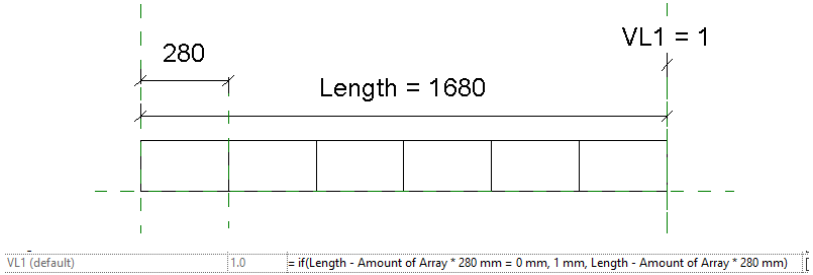
- Depending on the expected result, you can add an element with a variable length at the edge.
- Use the formula below to find the variable length value.



6- Elements with Dimension ≤ 0 Breaks

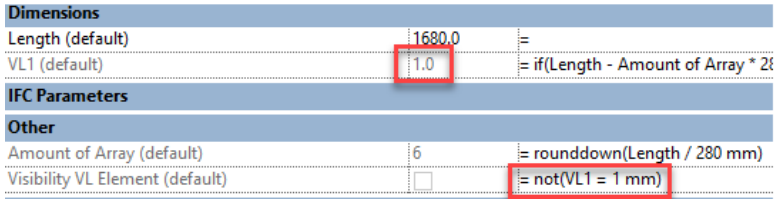
- If the calculated value of an element is equal or smaller than 0, the family will break. To prevent this, you can use a formula to ensure the dimension never goes below 0.
- Using an if formula: if(Condition,Result if true, Result if false)
- VL1 = *if(Length-Amount of Array * 280 < 0mm, 1mm, Length-Amount of Array*280)*



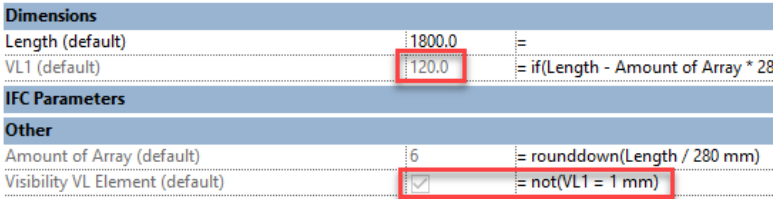


7- Make the Variable Element Invisible

- If the value of the Variable Element is less than 1mm, it should be invisible.
- You can use a **not** formula to make it disappear:



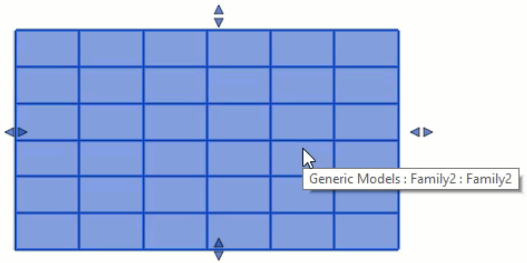
Dimensions		
Length (default)	1680.0	=
VL1 (default)	1.0	= if(Length - Amount of Array * 2
IFC Parameters		
Other		
Amount of Array (default)	6	= rounddown(Length / 280 mm)
Visibility VL Element (default)	☐	= not(VL1 = 1 mm)



Dimensions		
Length (default)	1800.0	=
VL1 (default)	120.0	= if(Length - Amount of Array * 28
IFC Parameters		
Other		
Amount of Array (default)	6	= rounddown(Length / 280 mm)
Visibility VL Element (default)	☑	= not(VL1 = 1 mm)

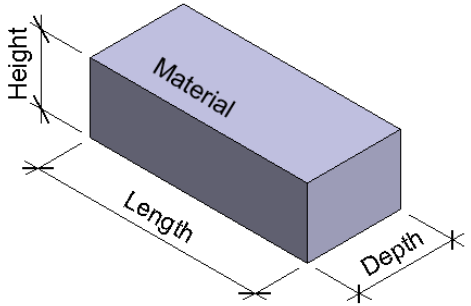
8- Be Careful with Horizontal + Vertical Arrays in the Same Family

- While technically possible, a vertical+ horizontal array in the same family can cause issues and slow down the family.
- More complexity in a family = more fragility
- Solution: nest one of the array in another family.



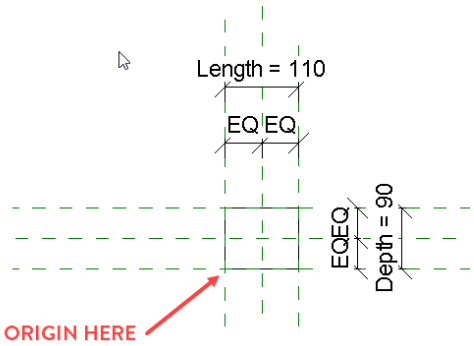
Level 1: Single Brick Family

- The first family is simple: a rectangular extrusion.
- Family category: Generic Model
- Important: set the origin of the brick at the bottom left.



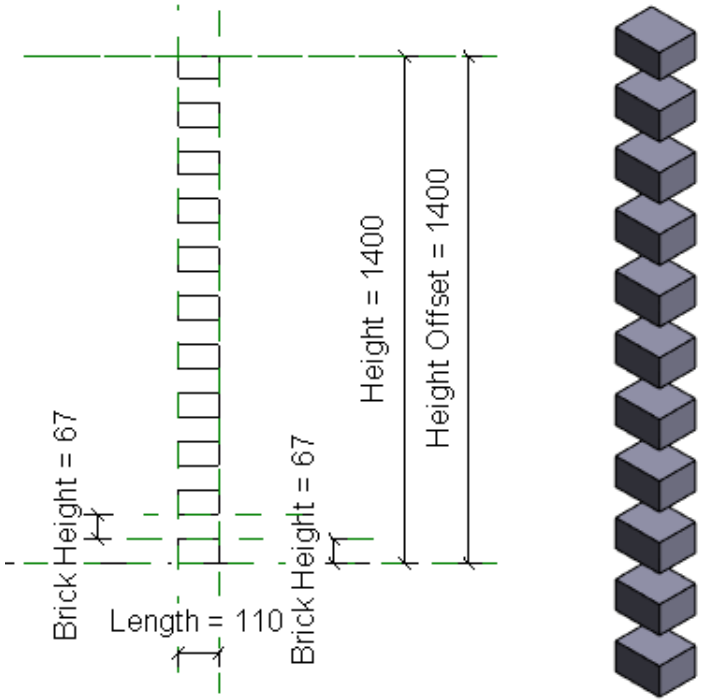
Materials and Finishes	
Brick Material	Brick-material
Dimensions	
Height	67.0
Length	210.0
Depth	90.0

SINGLE BRICK FAMILY



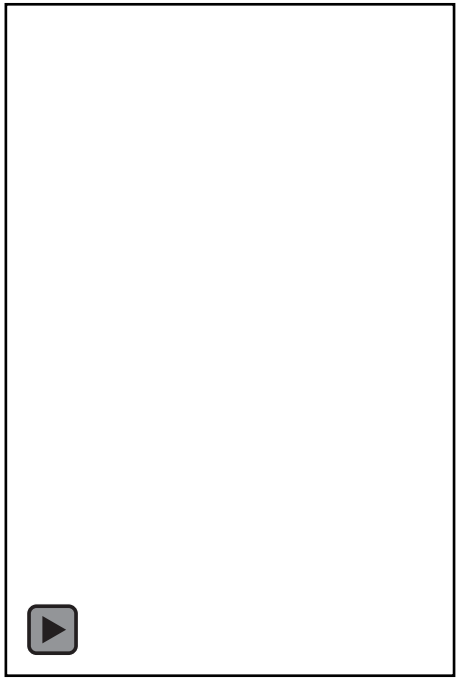
Level 2: Vertical Array

- This is the end result.
- Recreate all parameters from the single brick family and map them.



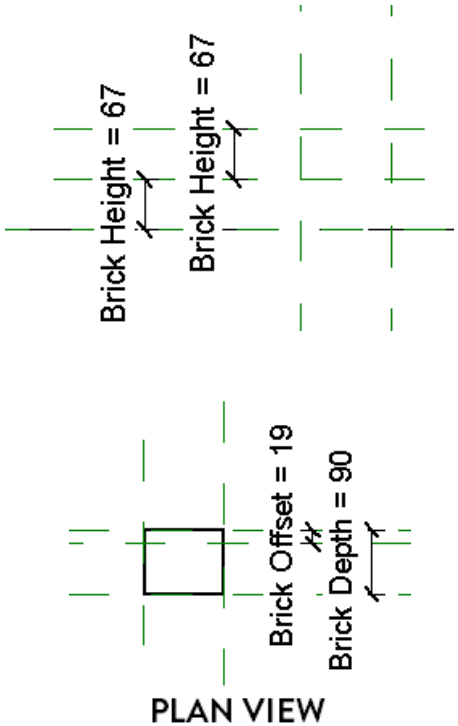
Level 2: Vertical Array

- Brick length, height are parametric.
- The height of the family can also be customizable.



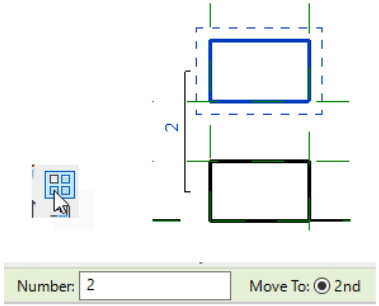
Create Vertical Array Family

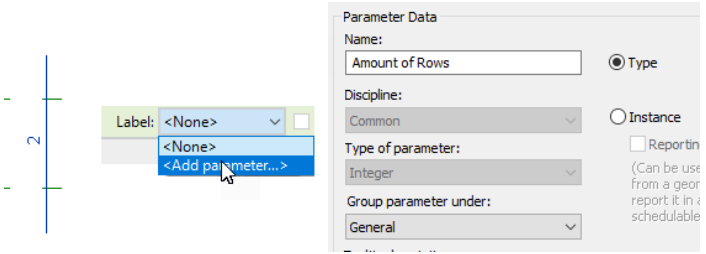
- In an elevation view, recreate this reference planes pattern and dimension parameters.
- Insert singe brick family into the array.
- In plan view, create the brick offset and brick depth parameters.
- Lock the single brick to the “brick offset” reference plane.



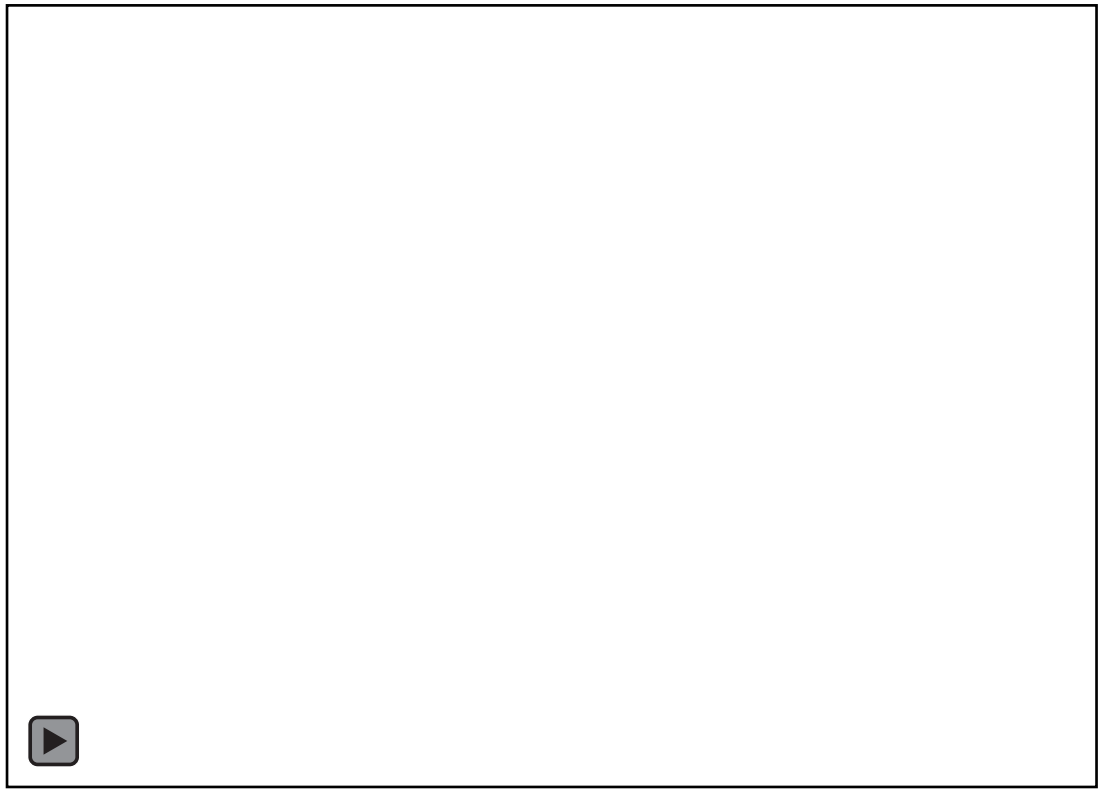
Create the Array

- In the elevation view, create an array. Select the “Move To: 2nd” option.
- Place the bottom of the brick on the third reference plane. Align the brick.
- Select the array and click on the number. Add a new parameter called “Amount of Rows” to it.



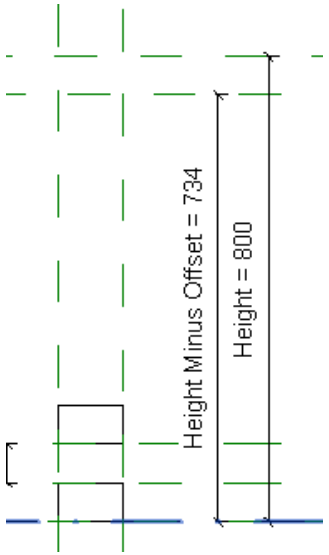


Create the Array (Video)



Place the Height Parameters

- Place the Height Minus Offset formula as in the image.
- Create two reference planes higher on the family: once with Height Minus Offset and Height parameters.
- The “Height Minus Offset” is used to make sure the family works with the second brick row.

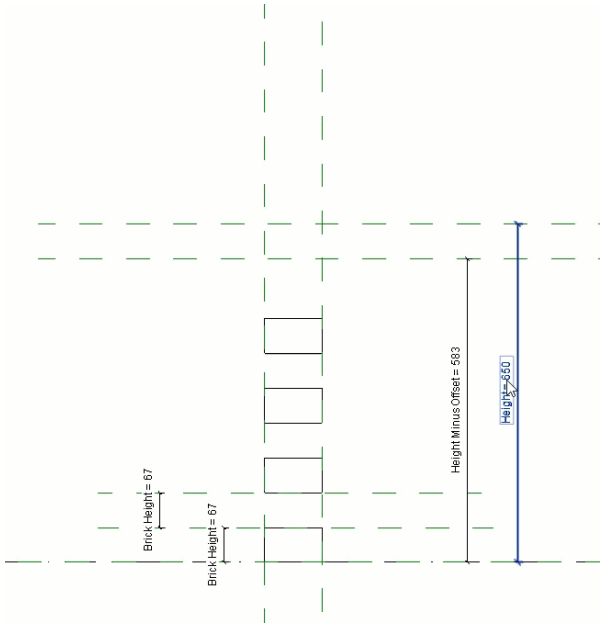


Brick Vertical Offset	67.0	=
Height	800.0	=
Height Minus Offset	733.0	= Height - Brick Vertical Offset

Place the Height Parameters

- Create the Amount of Rows formula.
- Test the family by changing the height parameter.

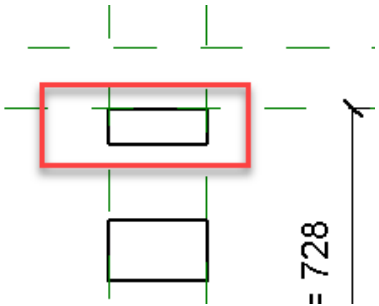
Other		
Amount of Rows	4	= Height Minus Offset / (Brick Height * 2)



Create the Top Brick

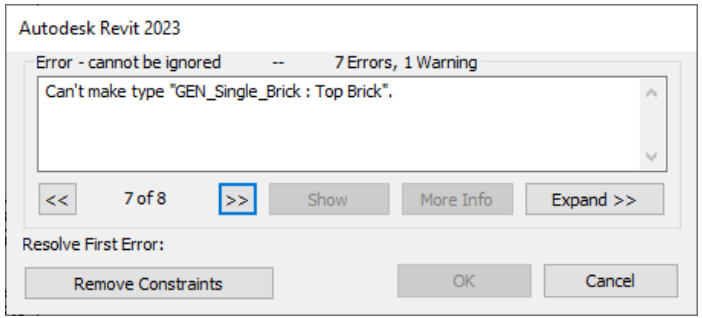
- The height of the top brick is variable, depending on the height of the array.
- The rest of the parameters need to be mapped as well.

Materials and Finishes	
Brick Material	Brick-nopattern
Dimensions	
Height	0.1
Length	110.0
Depth	90.0



Create the Top Brick: Avoid Dimension = 0

- You will get a warning if a dimension is changed to 0.



Create the Top Brick: Formulas

- If the height is too small, the family will break. Add a **if** formula to make sure the height doesn't go below 0.1mm.
- Formula logic: If the brick is smaller than 0.1mm, make it 0.1mm. Else, use the formula to find the height.
- If the height is 0.1mm, create a visibility parameter to hide the top brick.

BASIC IF LOGIC:

if(condition, value if true, value if false)

TOP BRICK HEIGHT FORMULA:

*if((Height Minus Offset - (Amount of Rows * (Brick Height * 2))) < 0.1 mm, 0.1 mm, (Height Minus Offset - (Amount of Rows * (Brick Height * 2))))*

TOP BRICK VISIBILITY FORMULA

Not(Top Brick Height = 0.1mm)

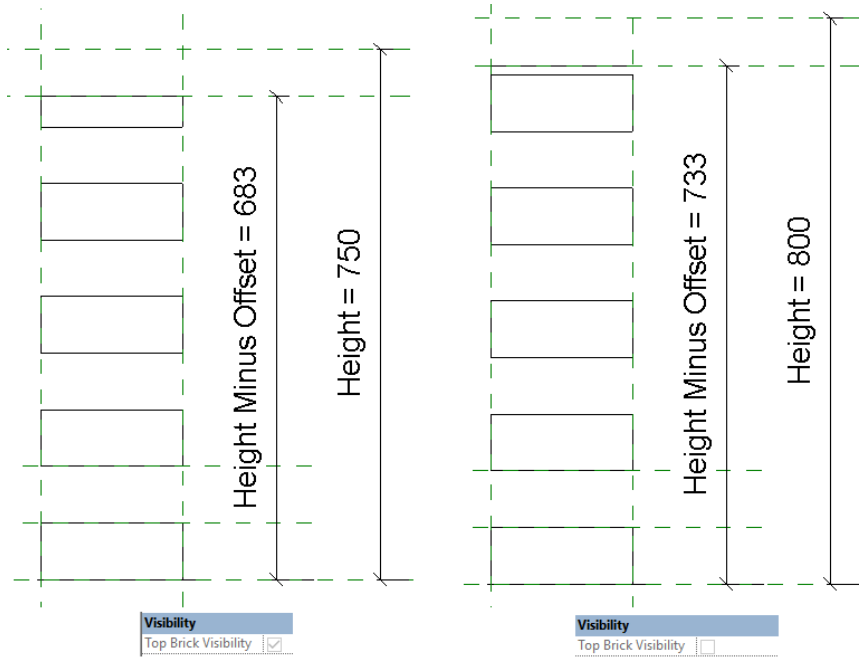
Array Parameters Overview

- Here are all the parameters and formulas you should have.

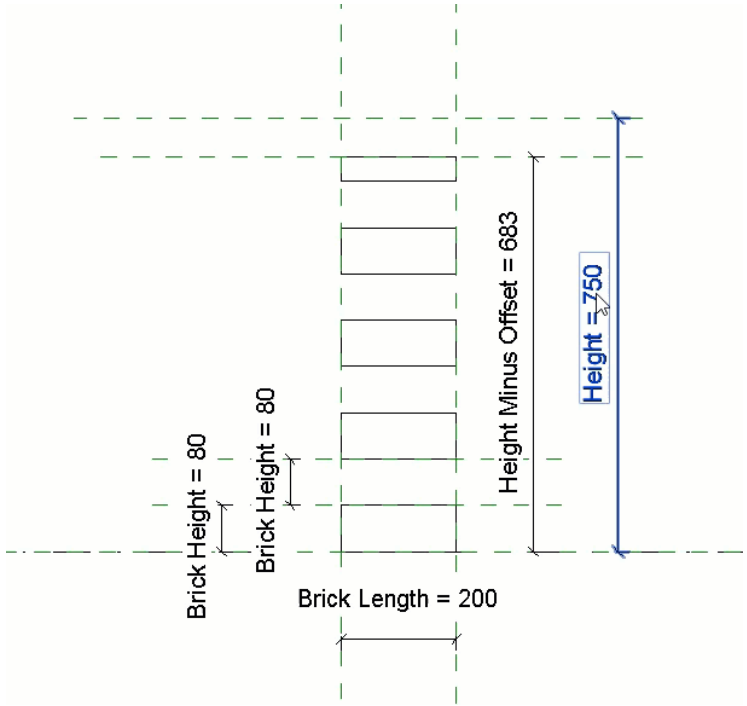
Dimensions		
Brick Depth	120.0	=
Brick Height	80.0	=
Brick Length	200.0	=
Brick Offset	19.0	=
Brick Vertical Offset	67.0	=
Height	1000.0	=
Height Minus Offset	933.0	= Height - Brick Vertical Offset
Top Brick Height	0.1	= if((Height Minus Offset - (Amount of Rows * (Brick Height * 2))) < 0.1 mm, 0.1 mm, (Height Minus Offset - (Amount of Rows * (Brick Height * 2))))
IFC Parameters		
Type IFC Predefined Type		=
Export Type to IFC As		=
General		
Amount of Rows	6	= (Height - Brick Vertical Offset) / (Brick Height * 2)
Visibility		
Top Brick Visibility	☐	= not(Top Brick Height = 0.1 mm)

Testing the Vertical Array Family

- The top brick visibility is automatically adjusted depending on the array's height.

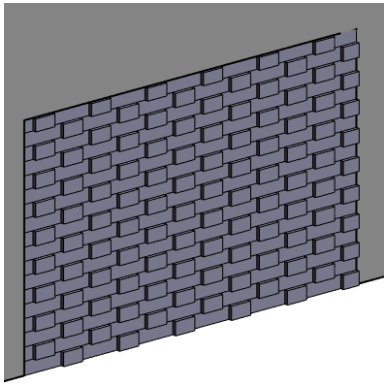


Testing the Vertical Array Family

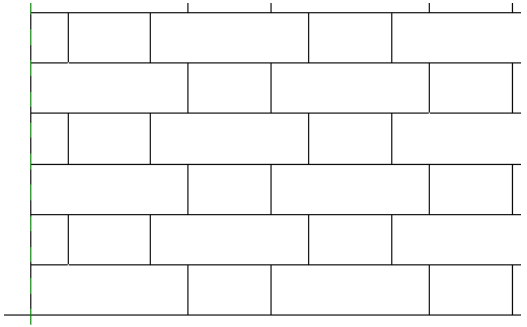


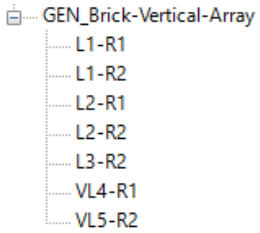
Level 3: Wall-Hosted Generic Family

- Create a new family using the “Wall-Hosted Generic Family” template.
- You want to recreate the brick pattern on the right.



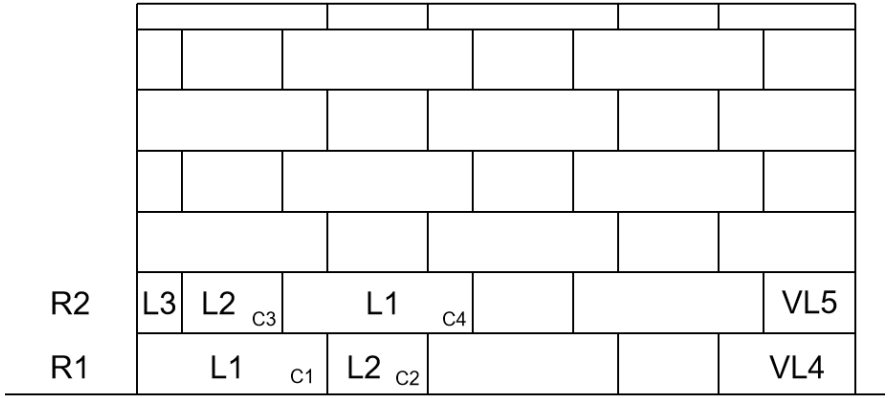
BRICK PATTERN
WALL-HOSTED FAMILY

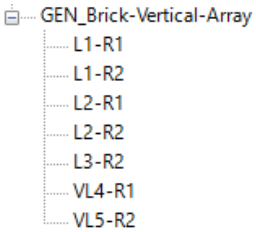




Creating a Brick Pattern Terminology

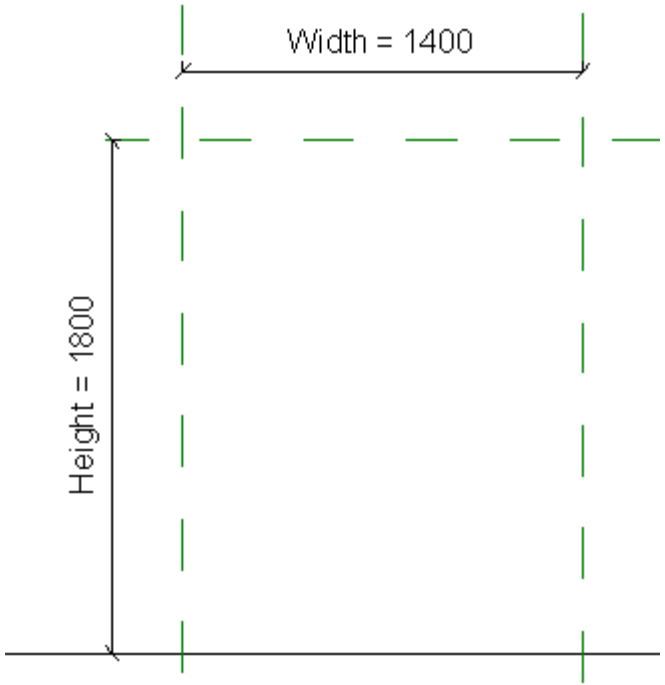
- There are two rows: R1 and R2.
- There are three “fixed-length” brick sizes: L1, L2 and L3.
- There are two variable-length bricks: VL5 and VL4.
- Therefore, you must create 7 types of the Vertical Array nested brick family.
- 4 of these types will be “arrayed” horizontally. They all have different formulas for the number of elements: C1, C2, C3 and C4.





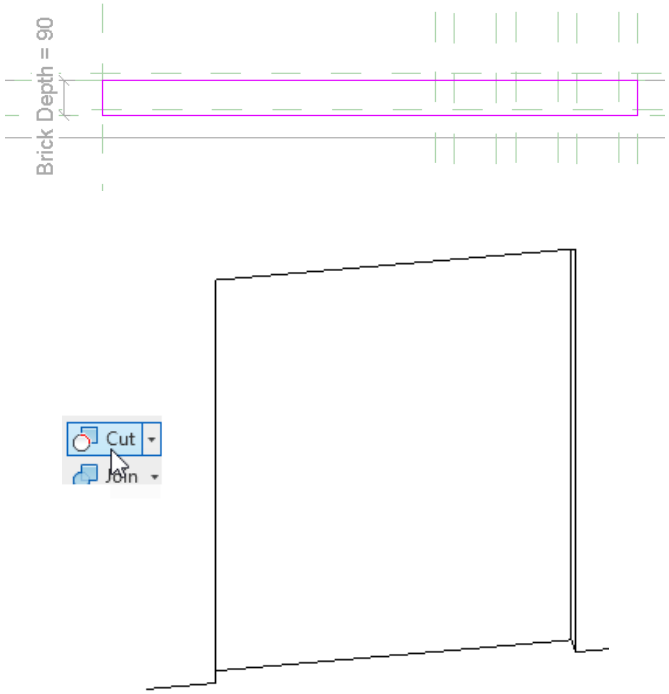
Create Basic Dimensions

- Create the Height and Width dimensions. Create them as **Instance** parameters.

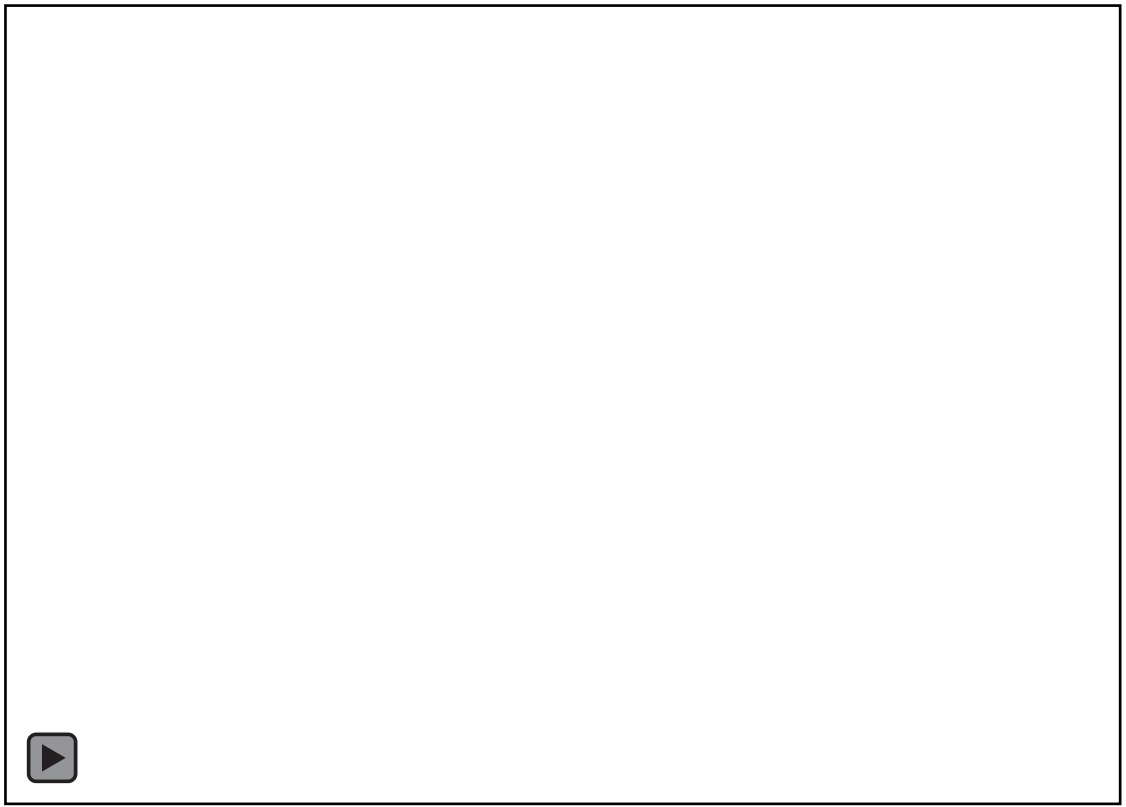


Cut the Wall

- In plan view, create a brick depth parameter.
- Then, create a void extrusion to cut the wall for the whole height, width and depth.
- Use the Cut tool. Select the wall, then click on the void extrusion. The wall is cut like this.

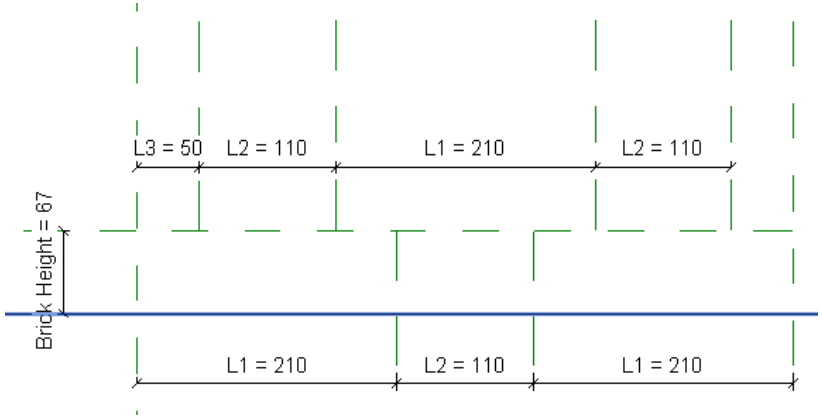


Cut the Wall (video)



Create Brick Length Pattern in Elevation

- Create a reference planes pattern like in this image.
- Create multiple length parameters: L1, L2 and L3.
- L3 length is driven by a formula. It is half the difference between L1 and L2.

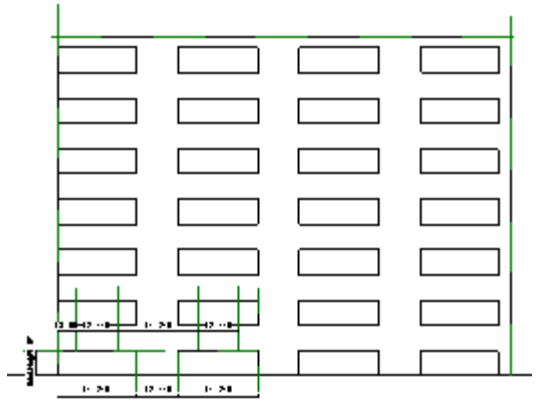


L1	210.0	=
L2	110.0	=
L3	50.0	= (L1 - L2) / 2

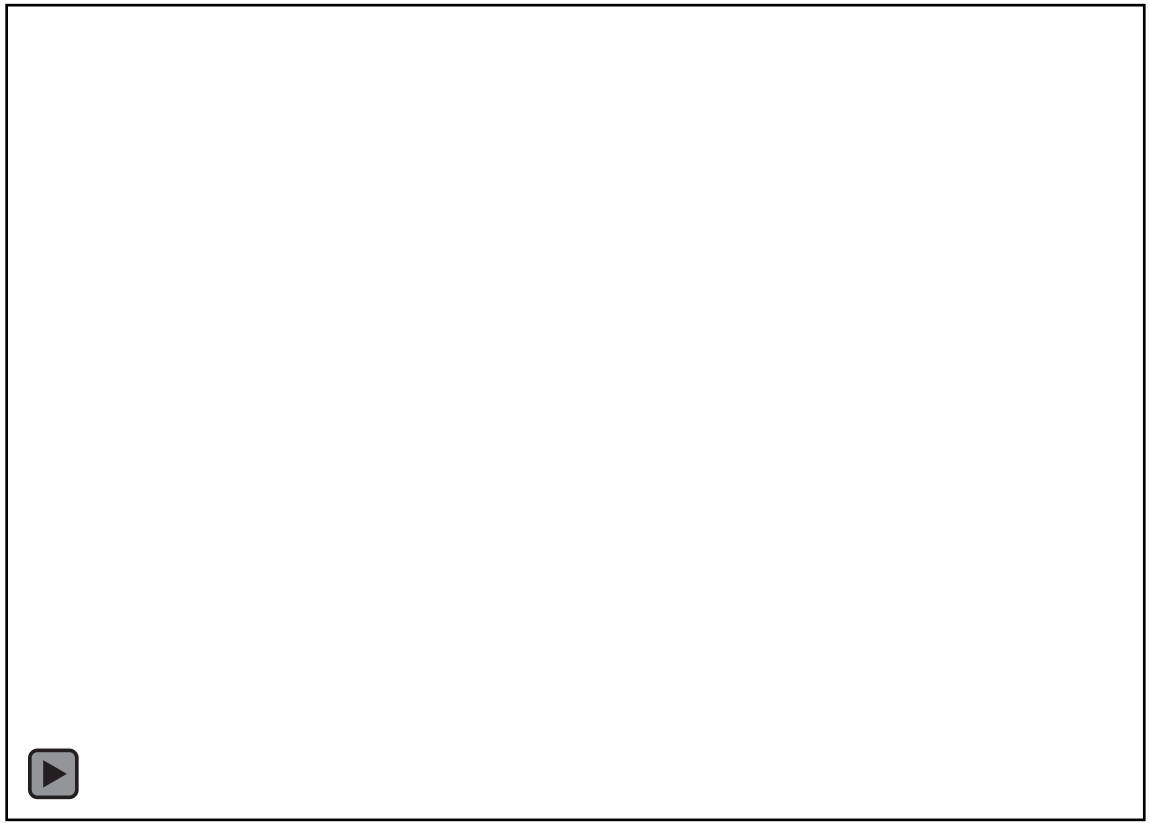
Map the Nested Family

- Place an instance of the vertical array family. Map all these parameters.
- The brick length varies from one type to another. In this case, you can use L1.
- Place the vertical array family in the plan view. Lock it to the planes. In the elevation view, create an array.

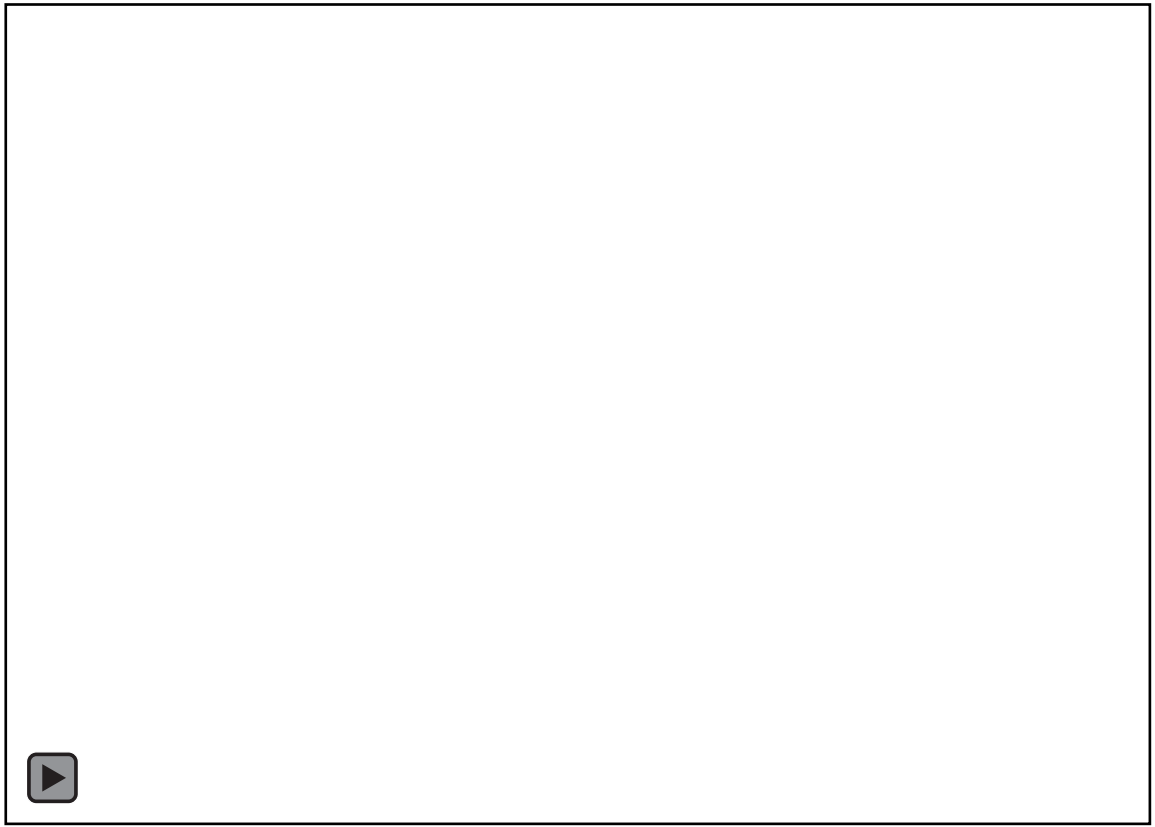
Dimensions	
Brick Depth	90.0
Brick Height	67.0
Brick Length	210.0
Brick Offset	0.0
Brick Vertical Offset	0.0
Height	1800.0



Map and Array the Nested Family

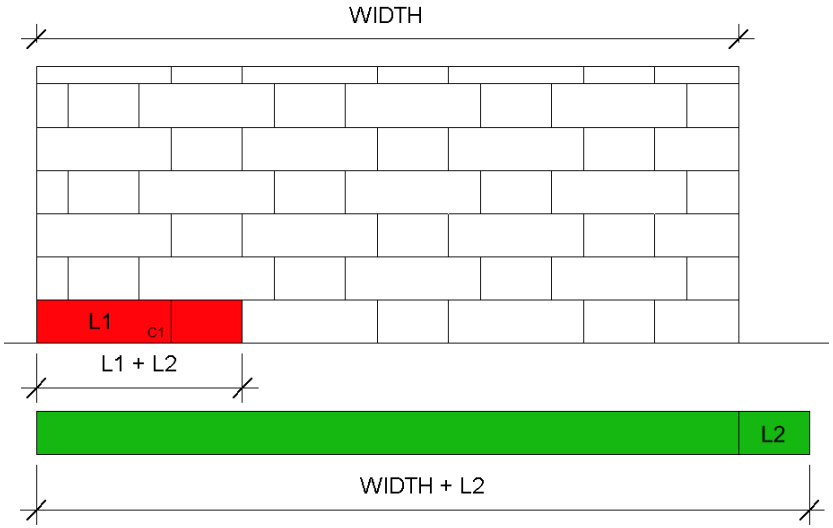


Array the Nested Family



Finding the C1 Formula

- C stands for « column ».
- Here is the visual guide to find out the Array formula for C1.
- C2, C3 and C4 have similar formulas, but slightly different to account for the brick placement offset.



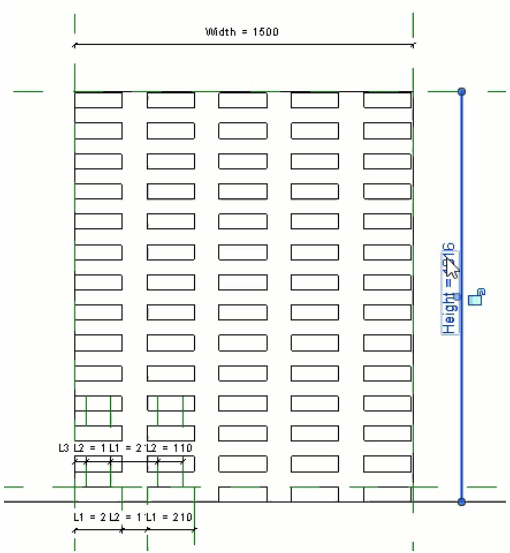
C1 ARRAY FORMULA: rounddown((Width+L2) / (L1+L2)

Other		
C1 (default)	5	= rounddown((Width + L2) / (L1 + L2))
C2 (default)	5	= rounddown((Width) / (L1 + L2))
C3 (default)	6	= rounddown((Width + (L2 + L3)) / (L1 + L2))
C4 (default)	5	= rounddown((Width - L3) / (L1 + L2))

Assign the C1 Formula

- You can see that after applying the formula, the family does flex.

Other		
C1 (default)	5	= rounddown((Width + L2) / (L1 + L2))



Place the Protruding Header

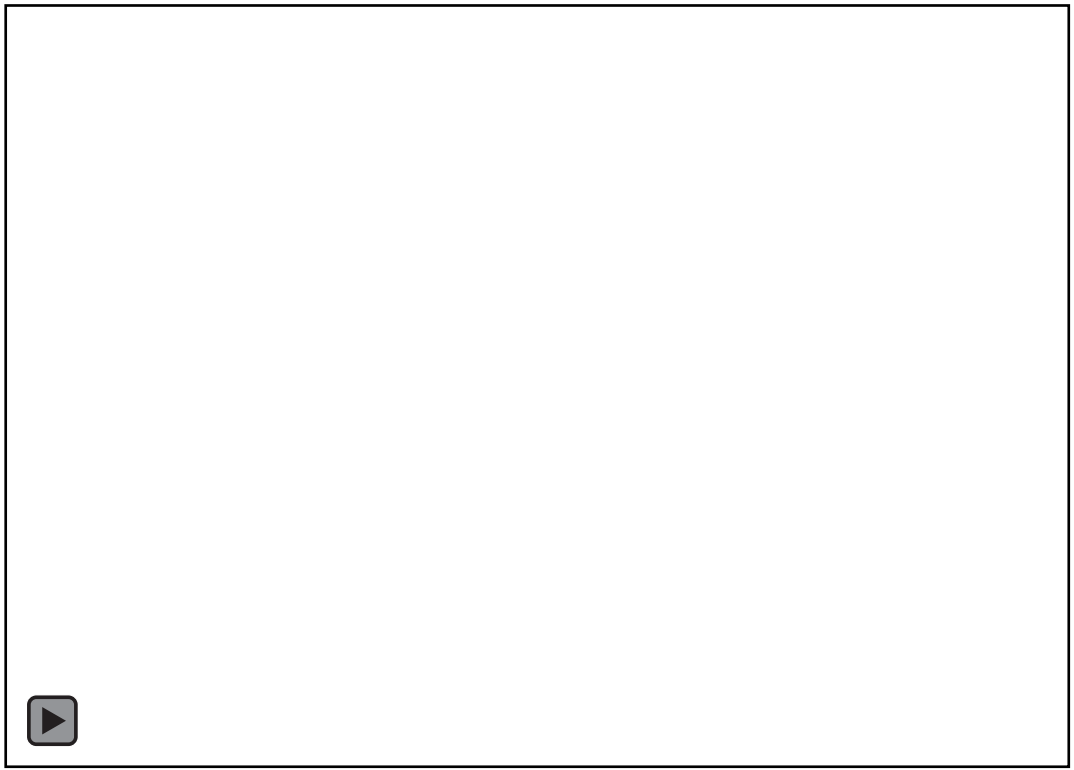
- Create another vertical array nested family type. This time, the length = L2 and the Brick Offset is mapped since this brick is protruding.

- The Array formula for C2 is similar, but slightly different to account for the width difference.

Materials and Finishes	
Brick Material	Brick-nopattern
Dimensions	
Brick Depth	90.0
Brick Height	67.0
Brick Length	110.0
Brick Offset	19.0
Brick Vertical Offset	**0.0**
Height	1900.0
Height Minus Offset	1900.0
Top Brick Height	24.0

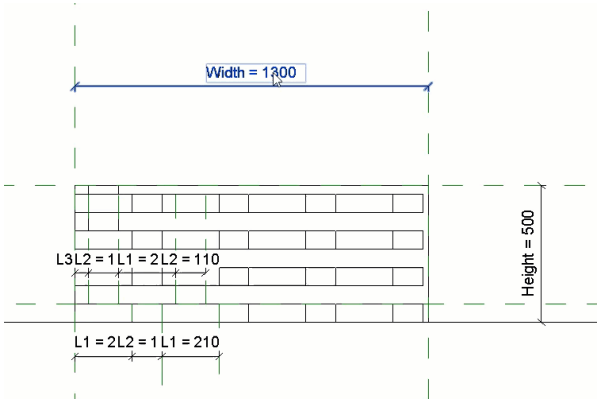
C2 (default)	5	= rounddown((Width) / (L1 + L2))

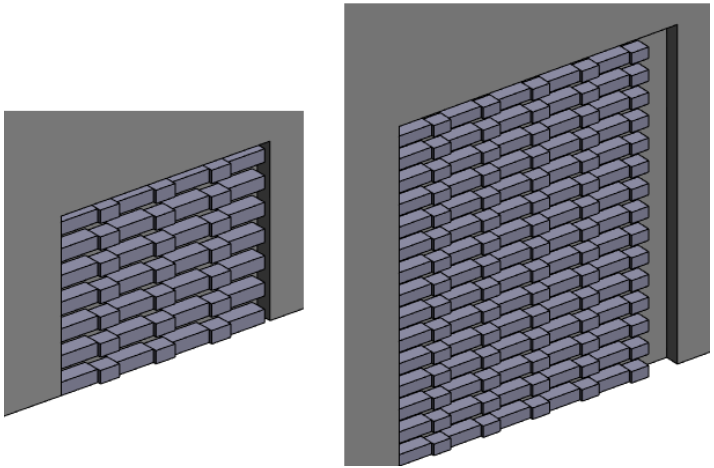
Place the Protruding Header (video)



First Row Almost Complete

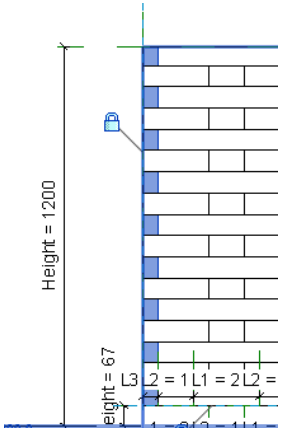
- The first row is almost complete.
- Test multiple dimensions to see if the family works.

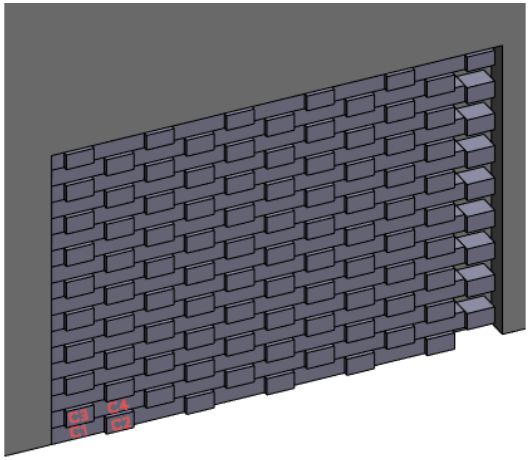




Create the Second Row

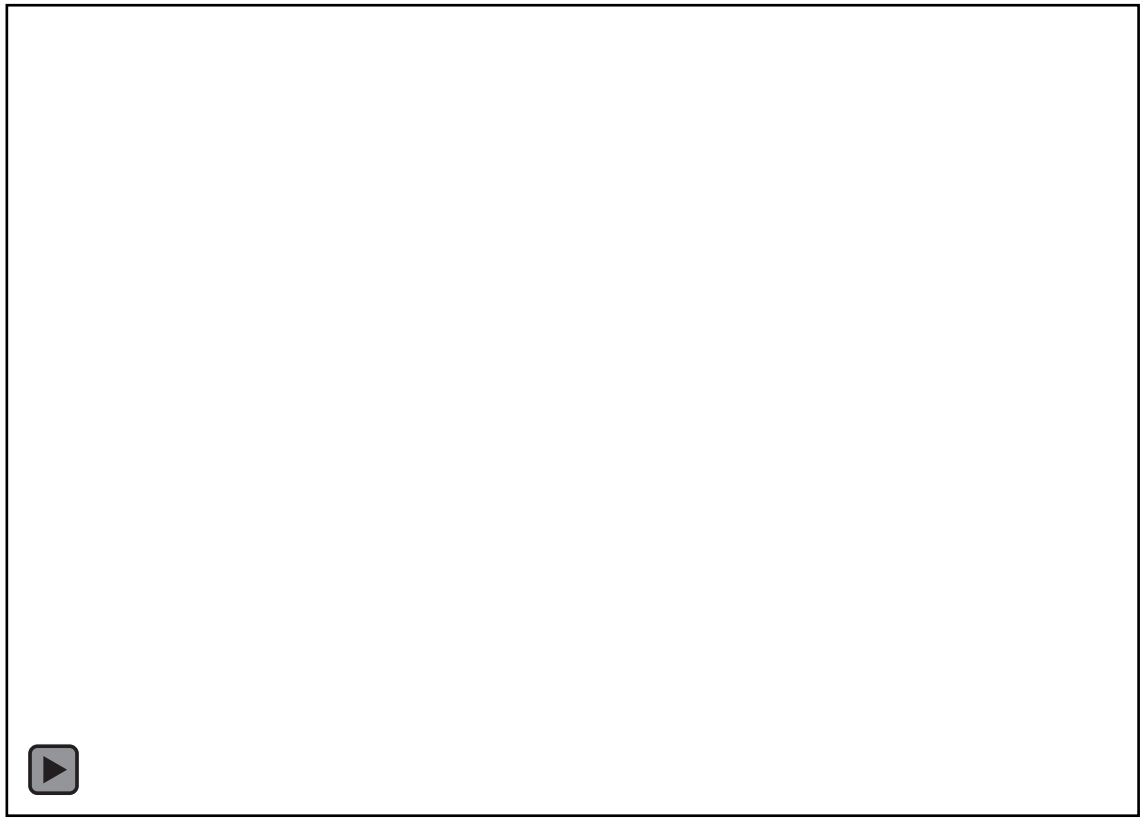
- The first vertical array of the second has a length of L3. It is **not** arrayed horizontally.
- The two next bricks are like the first rows. Use these formulas for C3 and C4.
- The second row “Vertical Brick Offset” should be mapped to the brick height.





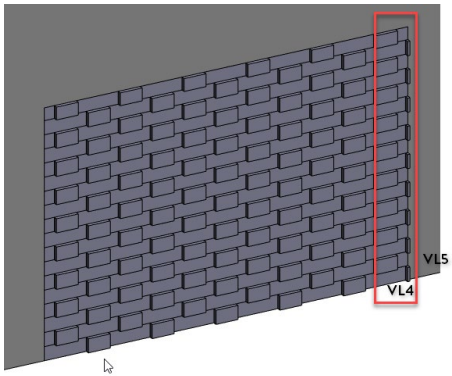
Other		
C1 (default)	5	= rounddown((Width + L2) / (L1 + L2))
C2 (default)	5	= rounddown((Width) / (L1 + L2))
C3 (default)	6	= rounddown((Width + (L2 + L3)) / (L1 + L2))
C4 (default)	5	= rounddown((Width - L3) / (L1 + L2))

Create the Second Row (video)



Create the Right-Side Bricks Width

- The right-side bricks have a variable length, based on the width of the family. The 1st row variable brick is VL4 and the 2nd row variable brick is VL5.
- The formula ensure that the width of the brick never go below 0.1mm so the family doesn't break.



VARIABLE LENGTH FORMULAS:

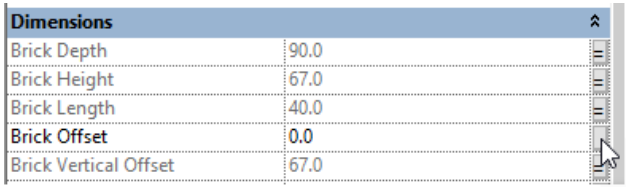
VL4: *if((Width - ((C1 * L1) + (C2 * L2))) < 0.1 mm, 0.1 mm, (Width - ((C1 * L1) + (C2 * L2))))*

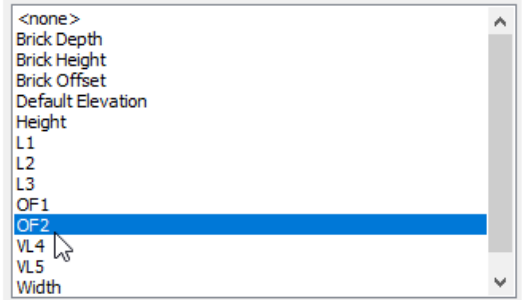
VL5: *if((Width - ((C3 * L2) + (C4 * L1)) - L3) < 0.1 mm, 0.1 mm, (Width - ((C3 * L2) + (C4 * L1)) - L3))*

Create the Right-Side Offset Formula

- Another tricky part of this family is the fact that the right-side brick is sometimes protruding, sometimes it is not.
- We must create an if formula to find out if it is the case or not. These parameters are called OF1 and OF2.
- Map these parameters to the vertical array types, in this case called VL4-R1 and VL5-R2.

OF1 (default)	19.0	= if(C1 = C2, 0 mm, Brick Offset)
OF2 (default)	0.0	= if(C3 = C4, Brick Offset, 0 mm)



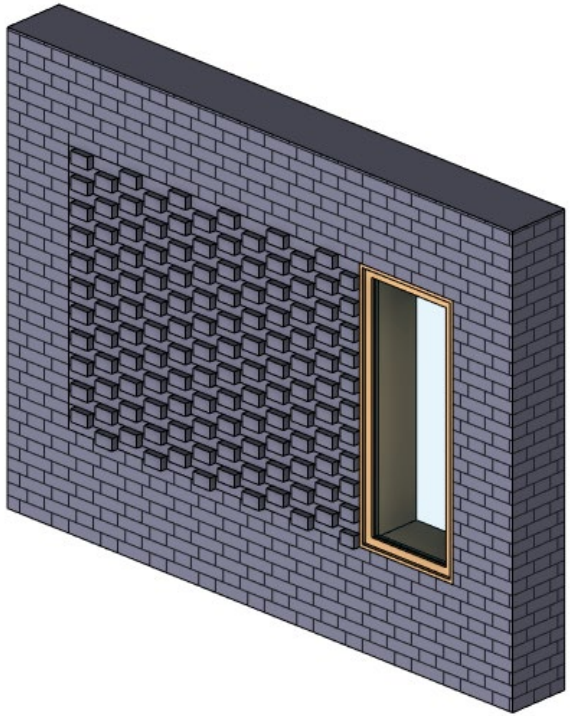


All Parameters

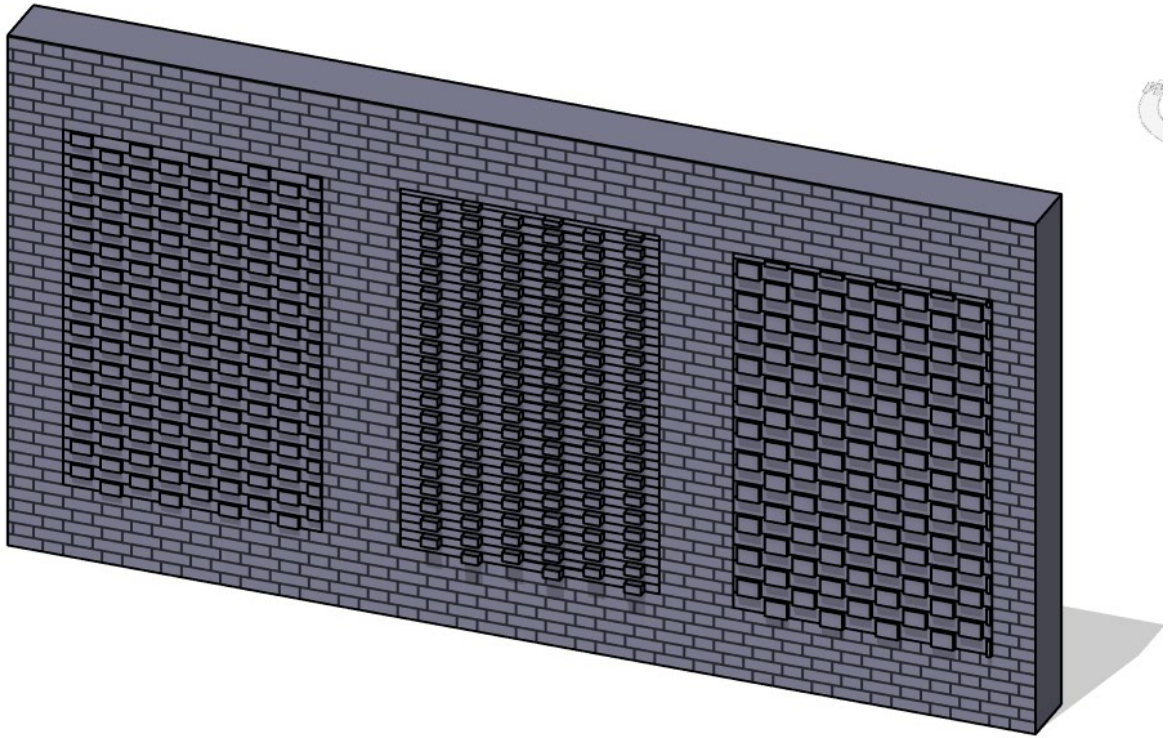
Constraints		
Default Elevation	0.0	=
Materials and Finishes		
Brick Material	Brick-nopattern	=
Dimensions		
Brick Depth	120.0	=
Brick Height	67.0	=
Brick Offset	19.0	=
Height (default)	1200.0	=
L1	210.0	=
L2	110.0	=
L3	50.0	= (L1 - L2) / 2
OF1 (default)	0.0	= if(C1 = C2, 0 mm, Brick Offset)
OF2 (default)	0.0	= if(C3 = C4, Brick Offset, 0 mm)
VL4 (default)	200.0	= if((Width - ((C1 * L1) + (C2 * L2))) < 0.1 mm, 0.1 mm, (Width - ((C1 * L1) + (C2 * L2))))
VL5 (default)	40.0	= if((Width - ((C3 * L2) + (C4 * L1)) - L3) < 0.1 mm, 0.1 mm, (Width - ((C3 * L2) + (C4 * L1)) - L3))
Width (default)	1800.0	=
IFC Parameters		
Other		
C1 (default)	5	= rounddown((Width + L2) / (L1 + L2))
C2 (default)	5	= rounddown((Width) / (L1 + L2))
C3 (default)	6	= rounddown((Width + (L2 + L3)) / (L1 + L2))
C4 (default)	5	= rounddown((Width - L3) / (L1 + L2))

Load and Test the Family

- Bring the family to a test project and make sure it properly works. Try to modify all the parameters.
- Current limitation: minimum width of L3+ (2*L1)+(2*L2). Else, the family breaks.
- Possible workarounds: create visibility formulas with arrays.



Load and Test the Family



Alternative: Skip the Level 2 Vertical Array

- It is possible to create this family with only 2 levels of families.
- The problem: the main family is more complex = more chances to break.
- This version is included in the dataset for you to compare.

Dimensions		
Brick Depth	90.0	=
Brick Height	67.0	=
Brick Offset	19.0	=
Height (default)	1450.0	=
L1	210.0	=
L2	110.0	=
L3	50.0	= (L1 - L2) / 2
Length (default)	2150.0	=
Offset	56.0	=
VH (default)	43.0	= if(VH Calculated < 0.1 mm, 0.1 mm, VH Calculated)
VH Calculated (default)	43.0	= Height - ((H1 + H2) * Brick Height)
VL1 (default)	20.0	= if(VL1 Calculated < 0.1 mm, 0.1 mm, VL1 Calculated)
VL1 Calculated (default)	20.0	= Length - ((C1 * L1) + (C2 * L2))
VL1-Offset (default)	19.0	= if(C1 = C2, 0 mm, Brick Offset)
VL2 (default)	70.0	= if(VL2 Calculated < 0.1 mm, 0.1 mm, VL2 Calculated)
VL2 Calculated (default)	70.0	= Length - ((C3 * L2) + (C4 * L1) + L3)
VL2-Offset (default)	0.0	= if(C3 = C4, Brick Offset, 0 mm)
IFC Parameters		
Other		
C1 (default)	7	= rounddown((Length + L2) / (L1 + L2))
C2 (default)	6	= rounddown((Length) / (L1 + L2))
C3 (default)	7	= rounddown((Length + (L2 + L3)) / (L1 + L2))
C4 (default)	6	= rounddown((Length - L3) / (L1 + L2))
H1 (default)	11	= Height / (Brick Height * 2)
H2 (default)	10	= (Height - Brick Height) / (Brick Height * 2)
R1-VH-Height (default)	☑	= not(VH Calculated = 0.1 mm)
R1-VH-Visibility (default)	☐	= C3 = C4
R1-VH-Visibility_Overall (default)	☐	= and([R1-VH-Height], [R1-VH-Visibility])
R2-VH-Height (default)	☑	= not(VH Calculated = 0.1 mm)
R2-VH-Visibility (default)	☑	= not(C3 = C4)
R2-VH-Visibility_Overall (default)	☑	= and([R2-VH-Height], [R2-VH-Visibility])

AUTODESK UNIVERSITY

Creating a Brick Pattern with Curtain Panels

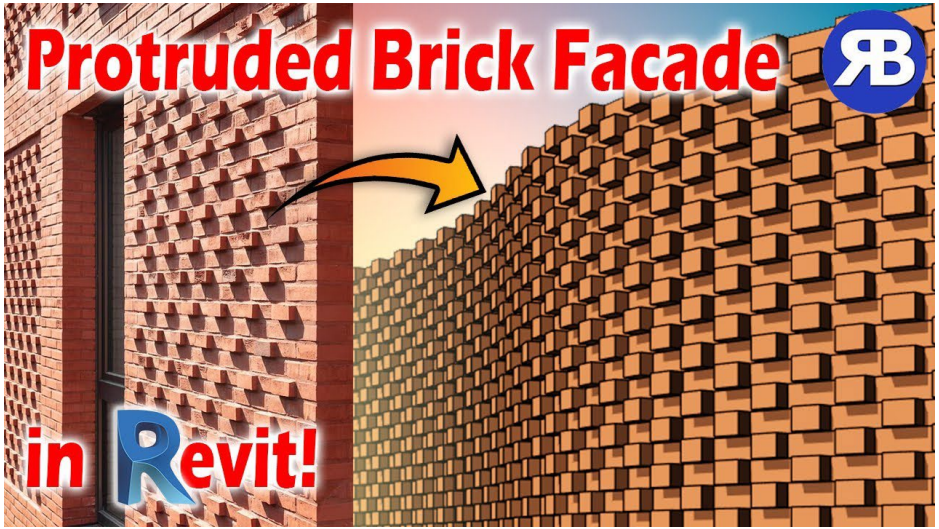
Creating a “Gradient” Brick Design

- When the design expanded, it went from being only between windows to having a “gradient” like effect on the façade.

- Big time constraint to present the design. Dynamo to the help!

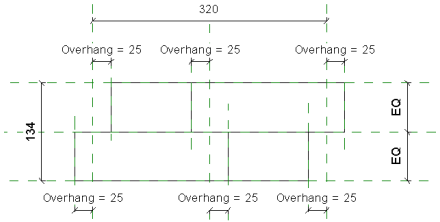
YouTube Tutorial with Curtain Walls

- Very simple technique that quickly creates an interesting result.
- Link: https://www.youtube.com/watch?v=q6OJHA43FKY



Bricks with Curtain Walls Strategy

- Create a curtain panel.
- The curtain wall properties must have fixed distance matching the dimension of L1 + L2.



CURTAIN WALL PANEL ELEVATION

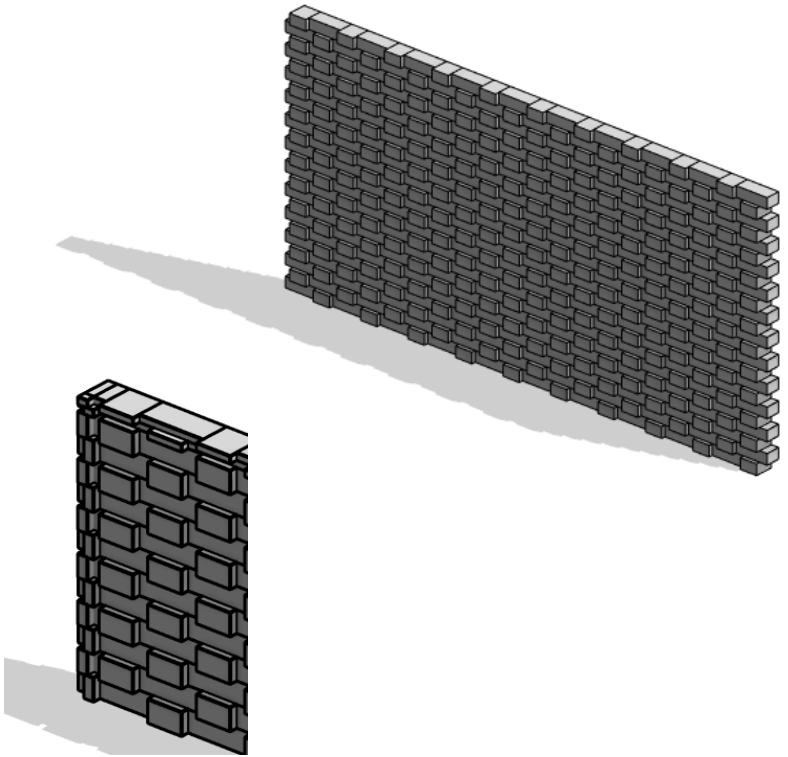
Type Parameters

Parameter	Value
Construction	
Function	Exterior
Automatically Embed	☐
Curtain Panel	CP_Brick_Panel : CP_Brick_Panel
Join Condition	Not Defined
Materials and Finishes	
Vertical Grid	
Layout	Fixed Distance
Spacing	320.0
Adjust for Mullion Size	☐
Horizontal Grid	
Layout	Fixed Distance
Spacing	134.0
Adjust for Mullion Size	☐

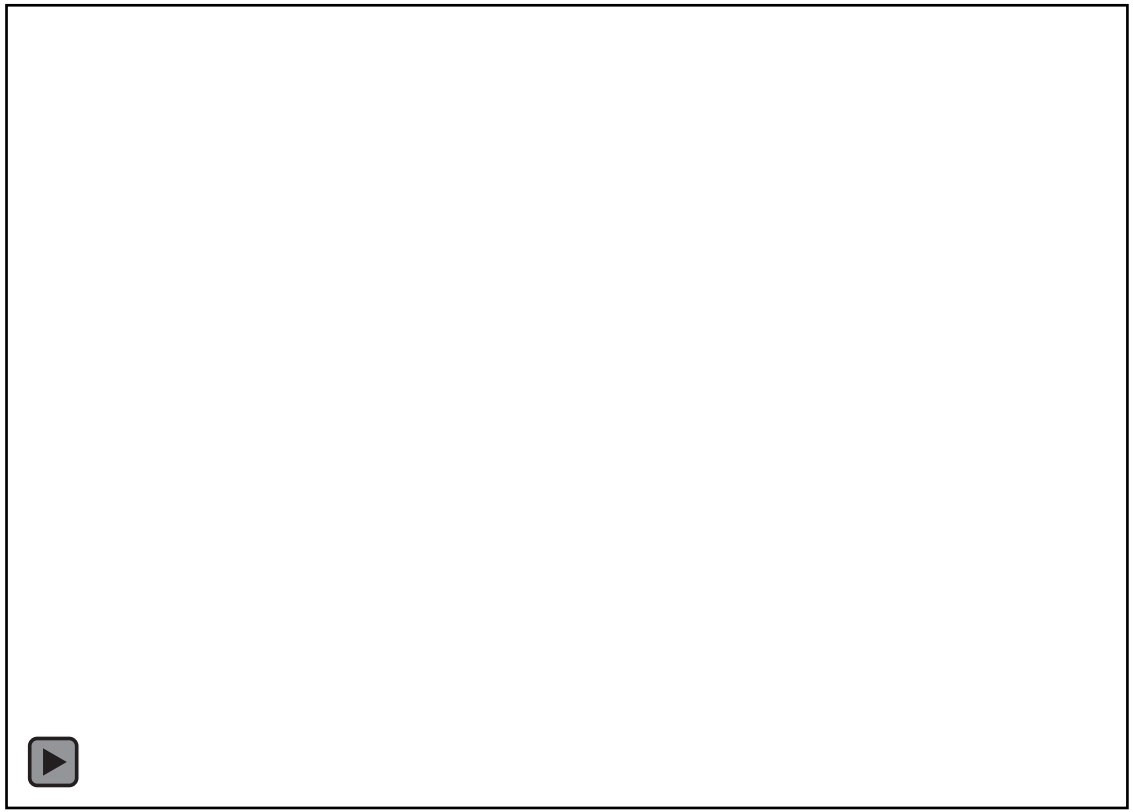
SET CURTAIN WALL TYPE PROPERTIES

Bricks with Curtain Walls Strategy

- This is very fast and easy to create.
- It works great for performance. Much faster than the nested family technique.
- However, there are problems at the edge of the walls and at openings.

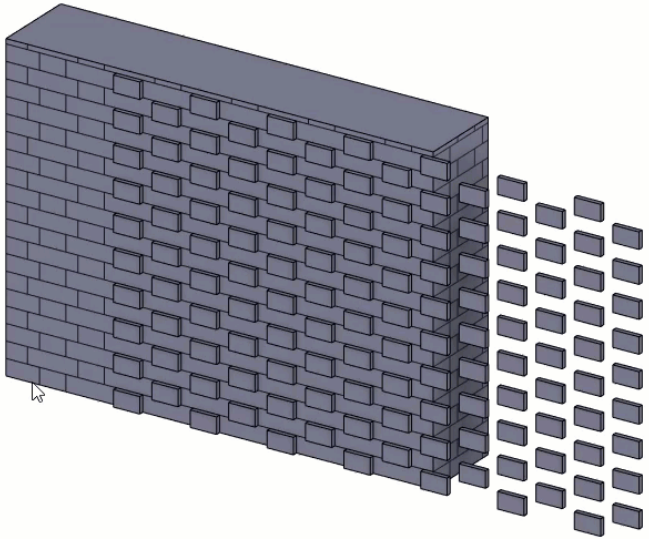


Bricks with Curtain Walls Strategy (video)



Alternative: Only Model Headers

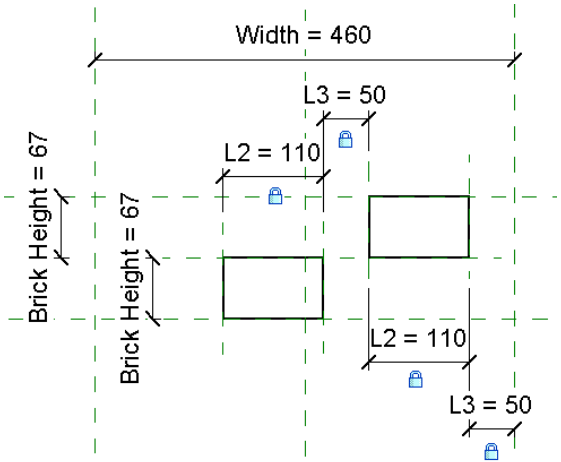
- A simplified curtain panel family can be created and placed over a regular brick wall.
- This will be easier to manage and won't interfere with the "regular" wall used in the Revit model.
- Curtain wall can be placed in a separate workset or even linked model to improve performance.



Create a Modified Curtain Panel Family

- Reproduce the pattern as in this elevation. Make sure to lock the dimension so the brick aren't deformed.

- Set a formula to L3.

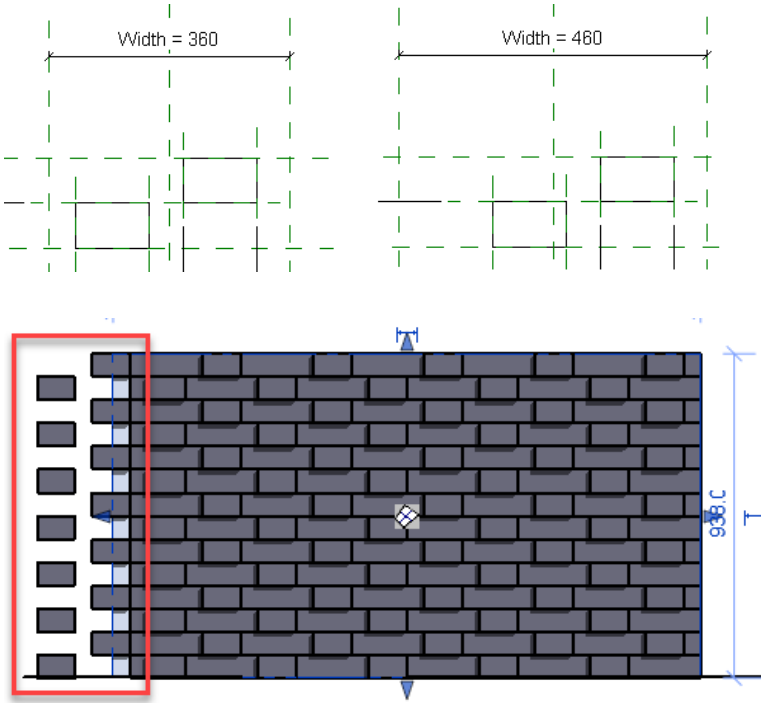


SIMPLIFIED CURTAIN PANEL

Dimensions		
L1	210.0	=
L2	110.0	=
L3	50.0	= (L1 - L2) / 2

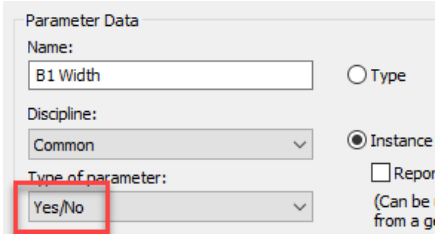
Create a Modified Curtain Panel Family

- When the width is modified, the bricks should always stay on the **right side** of the family.
- If you load the family back into the project and use it as a curtain panel, you will get this result. Current issue: the bricks extend beyond the curtain wall limit.

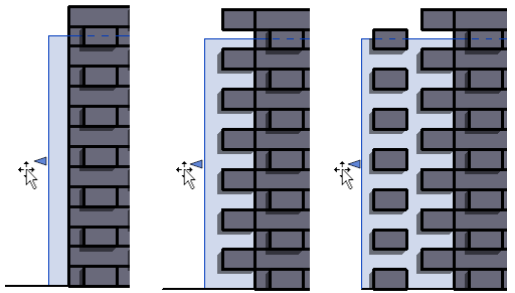


Create Visibility Parameters to the Bricks

- Add a “reporting” width parameter.
- Add visibility yes/no parameters to make the bricks disappear if they are beyond the curtain wall limit.
- B1 and B2 bricks will only be visible depending on the width of the curtain panels.



Visibility		
B1 Width (default)	☑	= not(Width < (L3 * 2 + L2 * 2))
B2 Width (default)	☑	= not(Width < (L3 + L2))



Create Visibility Parameters to the Bricks

- We potentially have the same issue at the top of the curtain panels. Add visibility parameters to make the bricks invisible.
- Finally, add a “Manual Visibility” that can be turned on and off.
- Add an overall visibility parameter for each brick. Use the **and** formula to make sure all parameters are checked.

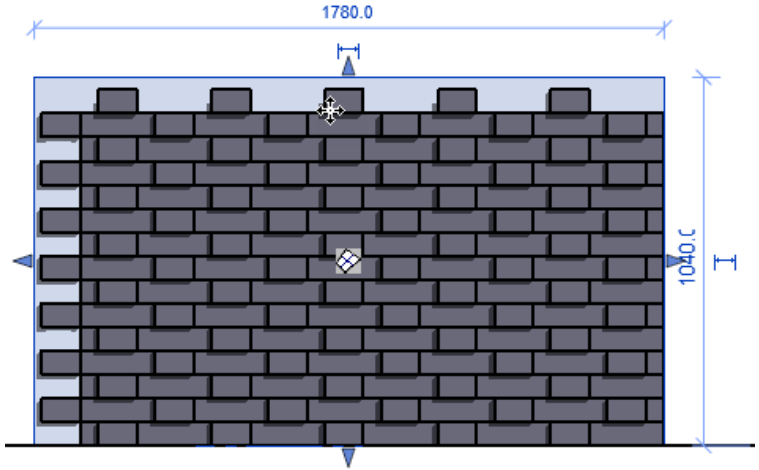
Visibility		
B1 Height (default)	☑	= not(Height < Brick Height)
B2 Height (default)	☑	= not(Height < Brick Height * 2)

Visibility	
B1 Manual Visibility (default)	☑
B2 Manual Visibility (default)	☑

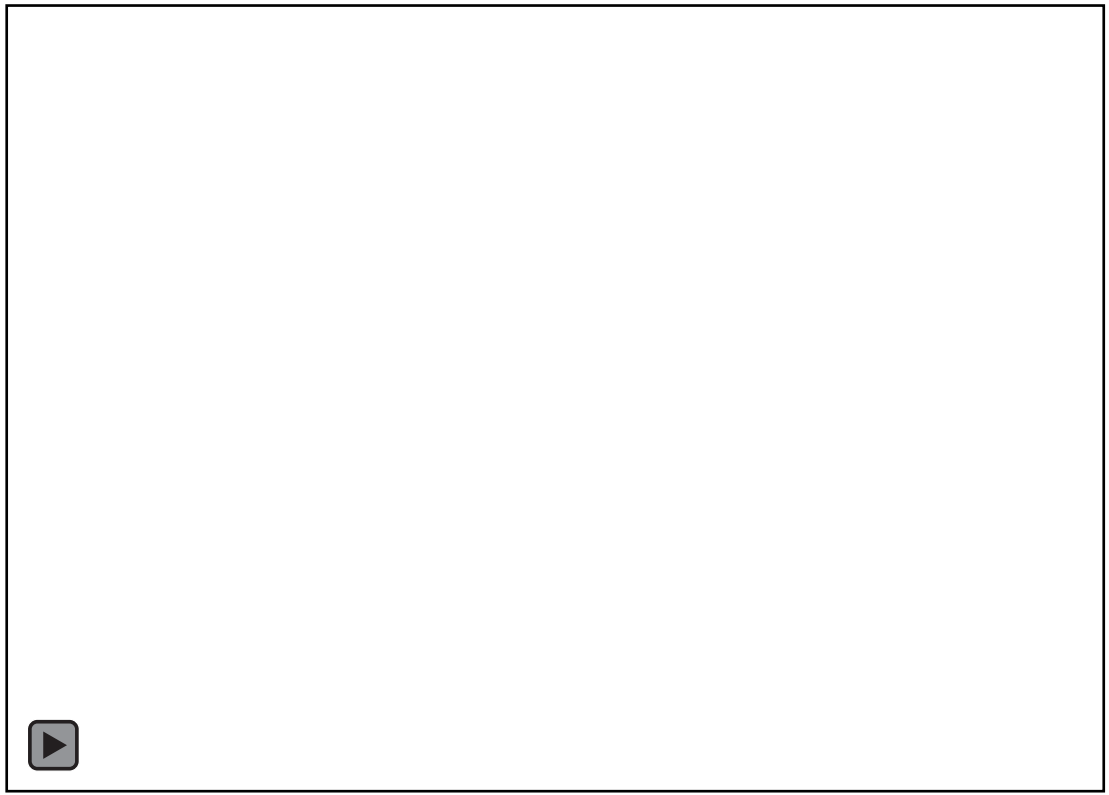
Visibility		
B1 Height (default)	☑	= not(Height < Brick Height)
B1 Manual Visibility (default)	☑	=
B1 Visibility (default)	☑	= and(B1 Height, B1 Width, B1 Manual Visibility)
B1 Width (default)	☑	= not(Width < (L3 * 2 + L2 * 2))
B2 Height (default)	☑	= not(Height < Brick Height * 2)
B2 Manual Visibility (default)	☑	=
B2 Visibility (default)	☑	= and(B2 Height, B2 Width, B2 Manual Visibility)
B2 Width (default)	☑	= not(Width < (L3 + L2))

Test the Curtain Wall

- Play with the height and width to ensure the family properly works.

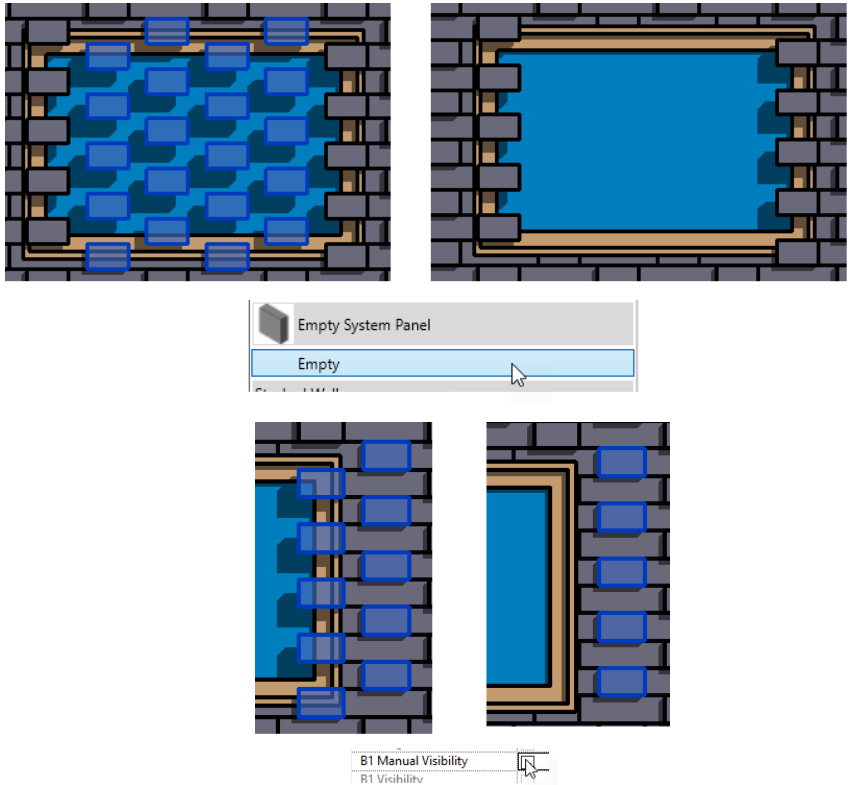


Test the Curtain Wall (video)

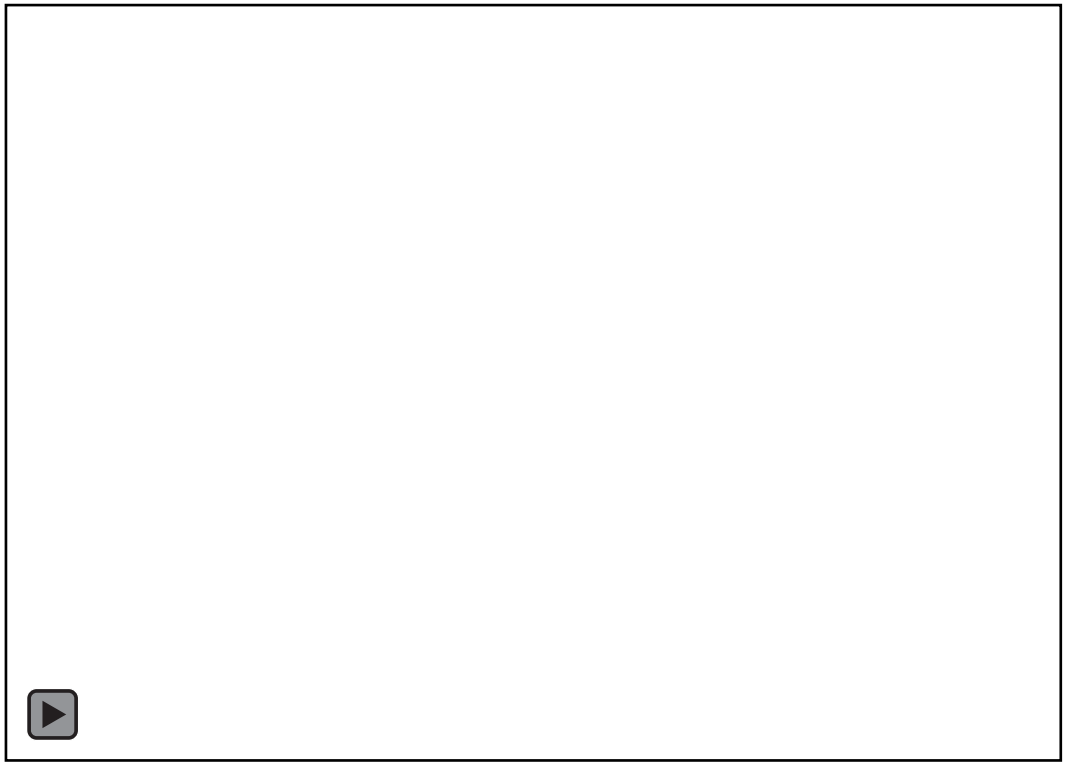


Dealing with Openings

- The curtain panels that are completely over the window should be switched to “Empty System Panel”.
- For the panels that only have 1 brick that should be removed, use the Manual Visibility parameter.
- Ideally, this can be done after the Dynamo script pass.

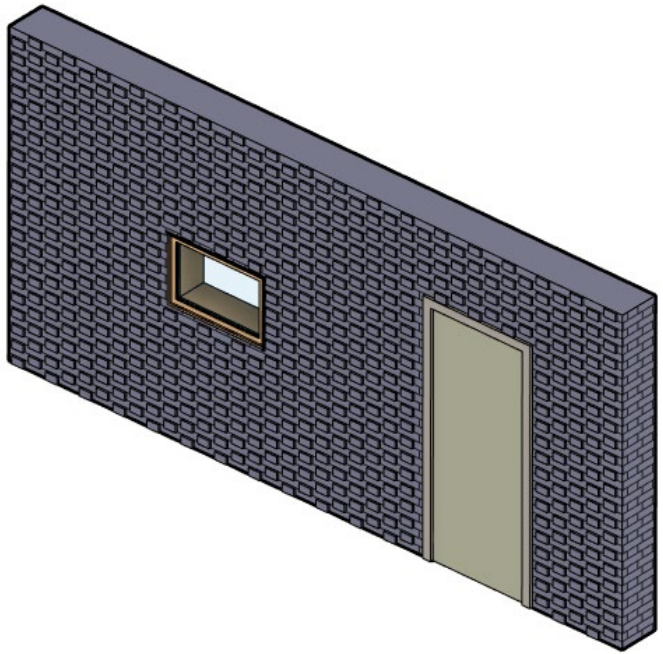


Dealing with Openings (video)



Ready for Dynamo?

- The curtain wall seems to be properly working.
- Time for some Dynamo magic.



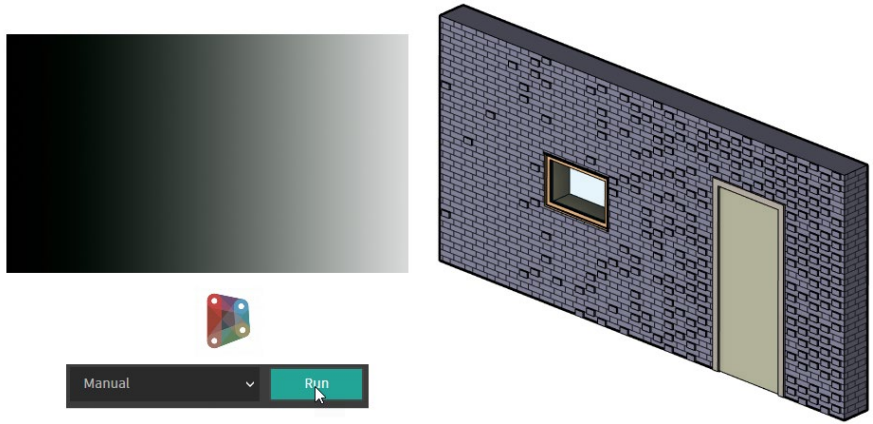
AUTODESK UNIVERSITY

Use Dynamo to Create a Brick Pattern

The Script in Action

- Here is the end result of the script. An image is mapped into the wall. White = 100% chance of protruding. Black = 0% chance of protruding. Everything grey is in between.

- Basic script idea by Niko G. Click here to see it in action.

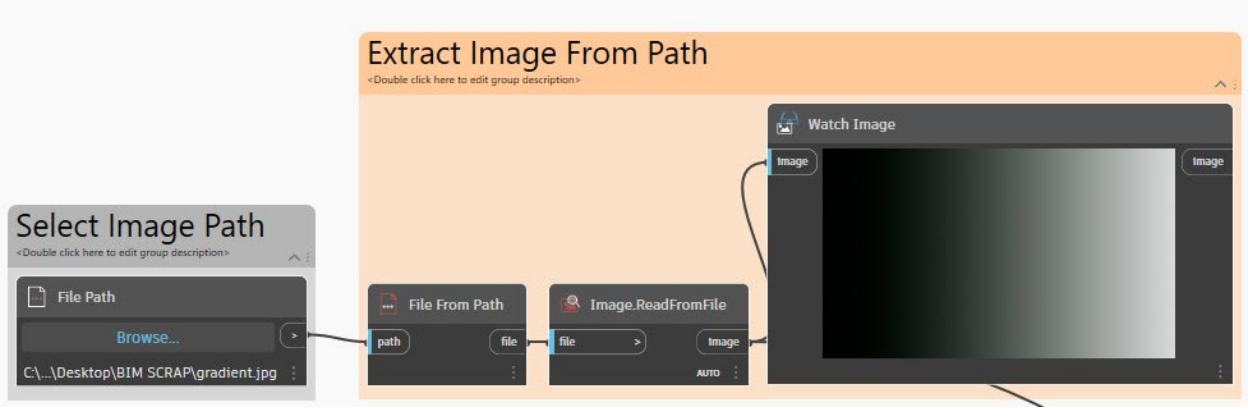


The Script Steps

1. Select a curtain wall, select all curtain panels inside of it, grouped by columns.
2. Select a .jpeg image to be used as the reference.
3. Map the brightness of the image to a grid matching the curtain panels.
4. In this case, white = yes and black = no. Everything in between is a gradient.
5. Take the brightness of the pixel and add it to a random number. If the result is higher than a threshold value, the brick is protruding.

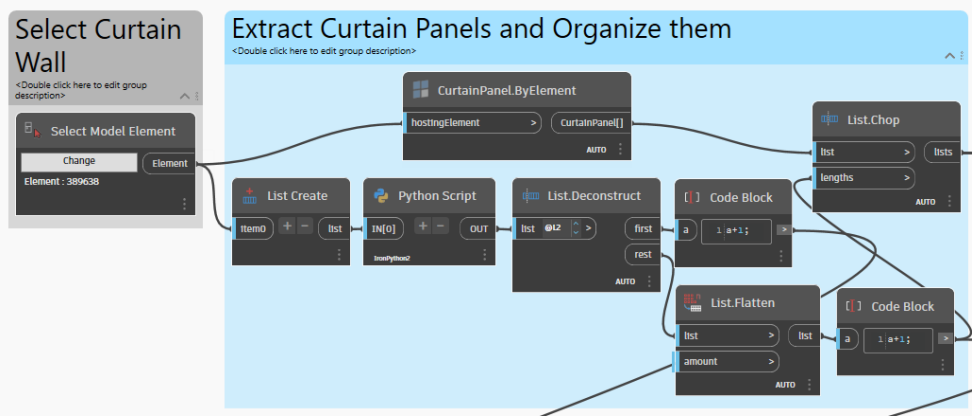
Extract the Image

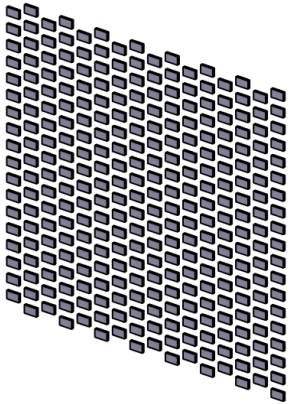
- Use these nodes to select an image on your computer.

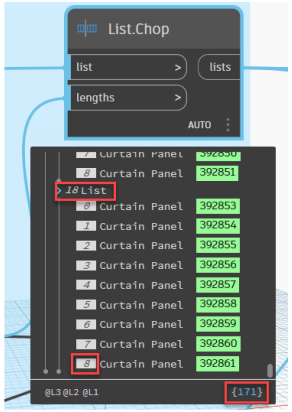


Select the Curtain Wall and Extract the Panels

- Select the curtain walls. These nodes will reorganize the curtain panels in lists. Each sublist contains all panels in a row. The Python node extracts the number of grids: you must add 1 to this number to get the number of panels in rows/columns.

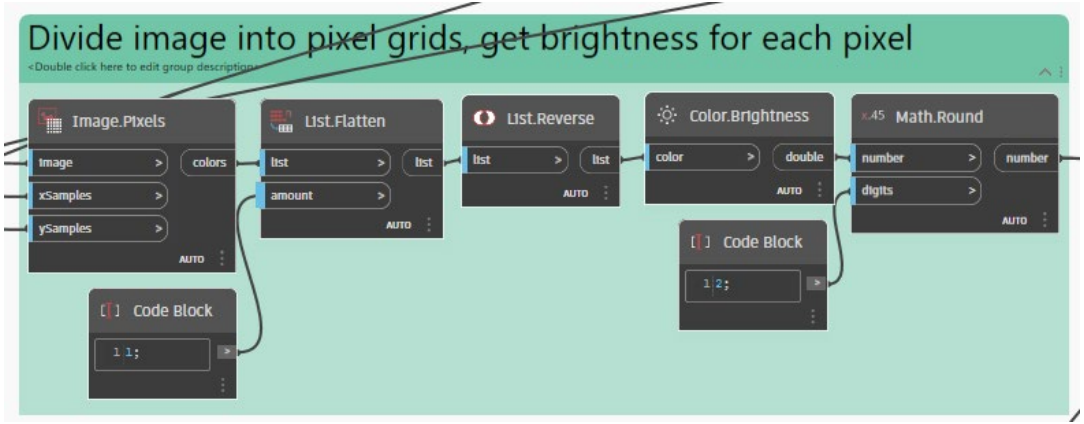


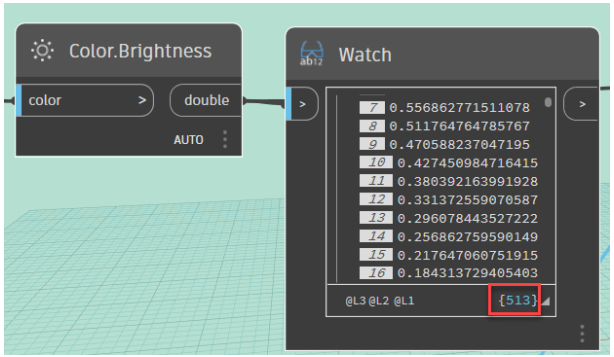




Divide the Image into Pixel Grids, get Brightness

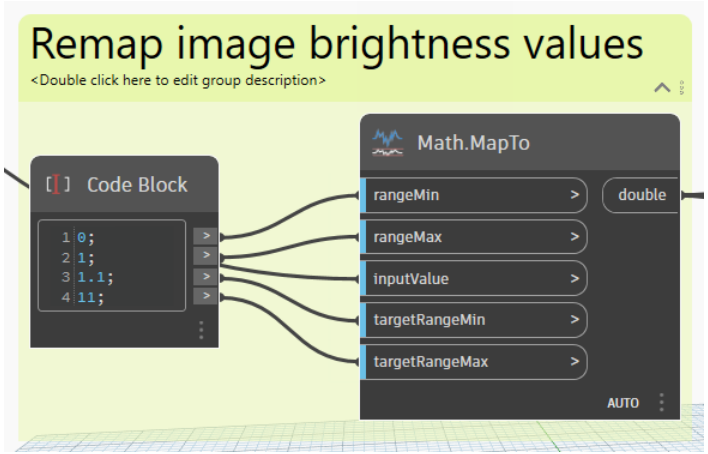
- These nodes allow you to “resample” the image into a specific number of pixels. This number matches the amount of curtain panels. The brightness of each pixel is extracted.





Remap the Brightness Values

- We want these brightness values to be “remapped” between 1.1 and 11. This will be helpful when creating the probability with “random” nodes.

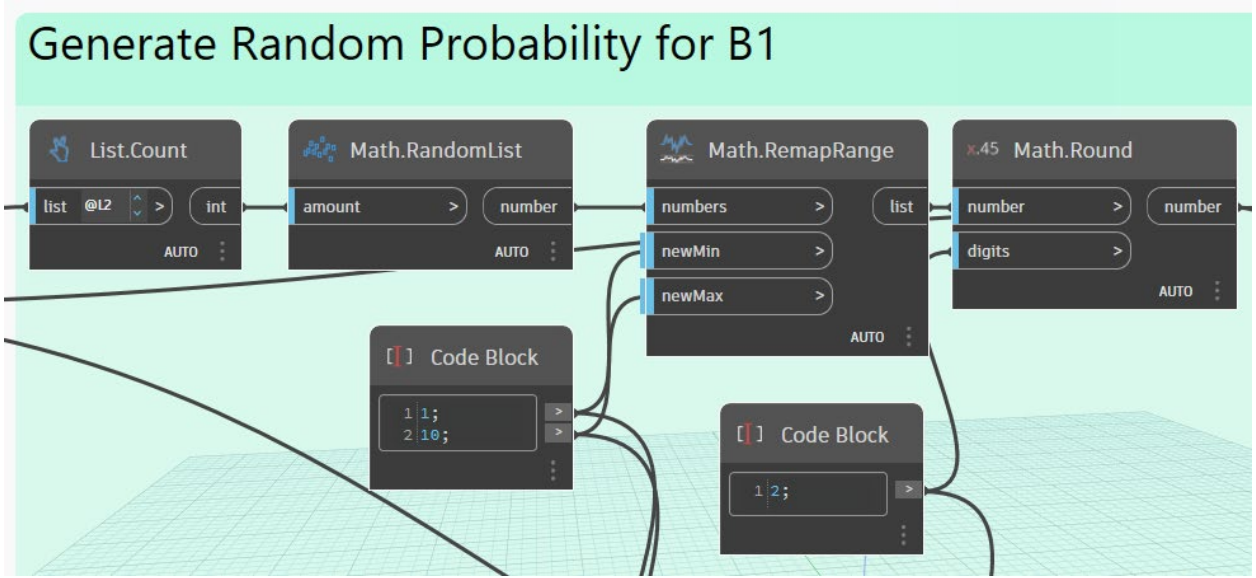


Probability & Threshold Value

- The Pixel Brightness is between 1.1 and 11.
- A random value is added to this number.
- The Pixel Brightness + Random Number must be above 11 for the brick to be protruding. It means the **Threshold value is 11**.
- There, the random number generated should be between 1 and 10.

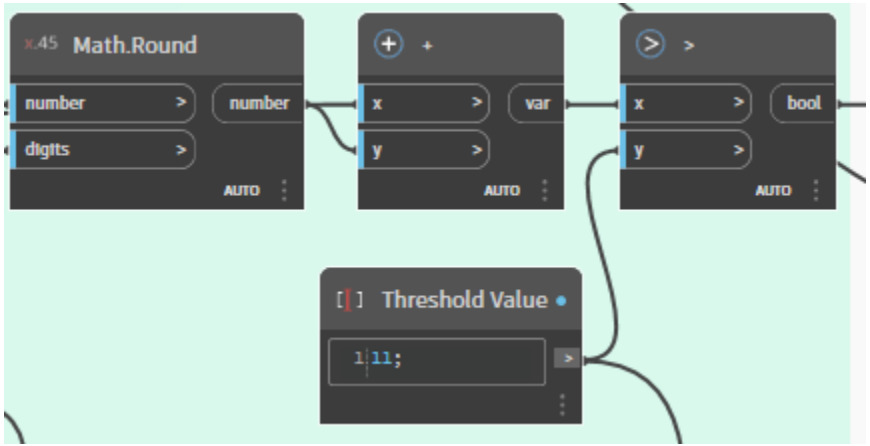
Generating Random Probability

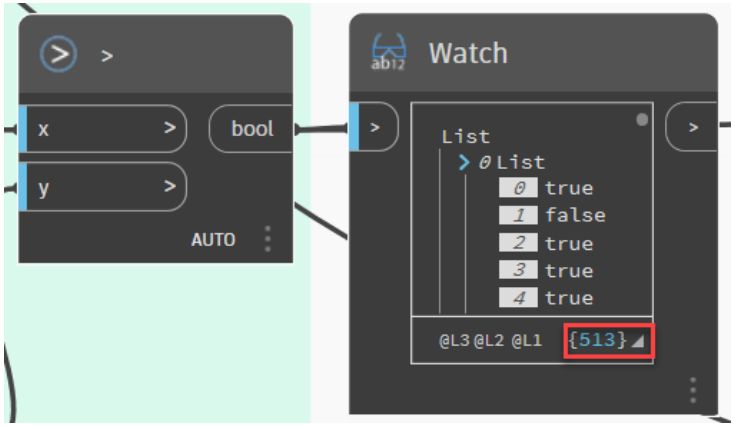
- In the node below, we generate a list of numbers between 1 and 10. The amount of number generated is equal to the amount of curtain panels.



Generating Random Probability

- The random number is added to the brightness value. We set a threshold value of 11. Numbers below this number make the brick invisible. Numbers above it make the brick visible. White has almost 100% chance, while black has 0% chance. The result is a list of true (visible) or false (invisible).



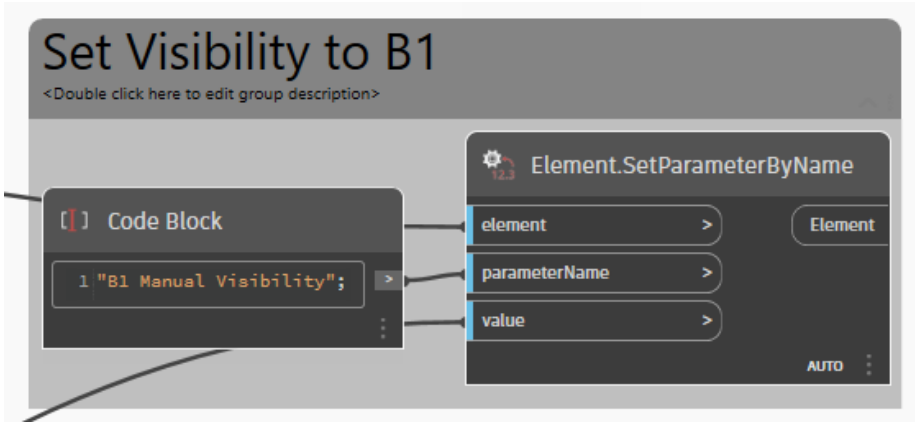


Threshold Value

- Let’s give an example. Here are three pixels brightness value:
- Pixel A has a brightness of 3.5 (dark grey). The random generated number for this pixel is 6. The addition of these numbers is 9.5. It doesn’t pass the threshold value of 11. The brick will not be protruding.
- Let’s put it in math terms. The equation for the brick to be protruding is: ***Pixel Brightness + Random Number > 11***
- In the case of Pixel A, the equation is: ***3.5 + 6 < 11***
- Each brick has: a pixel brightness and a random number.

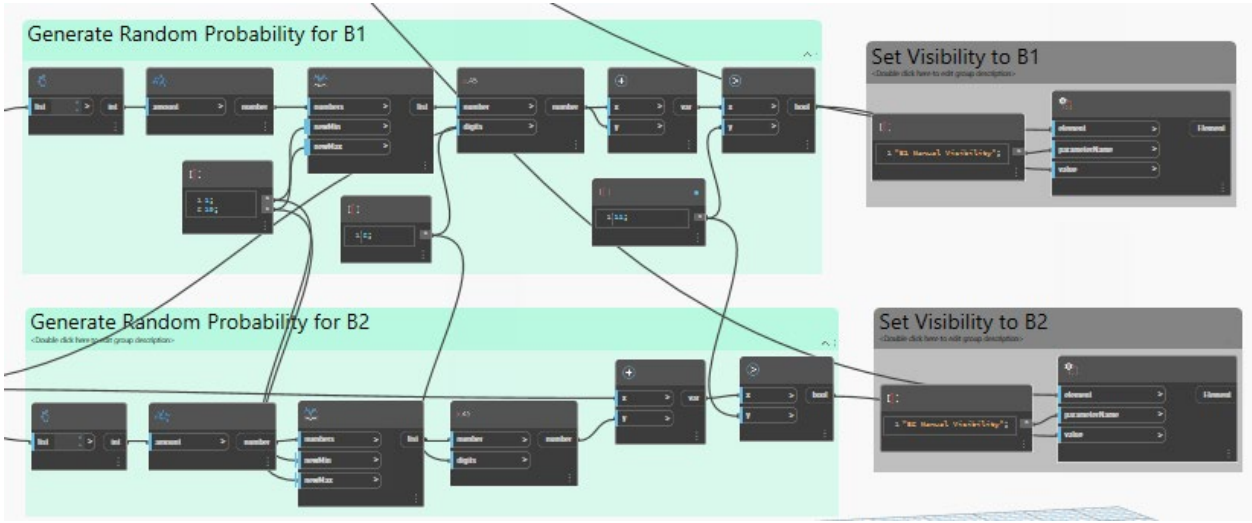
Assign Visibility to Each Brick

- This true false value is assigned to the “B1 Manual Visibility” parameter.



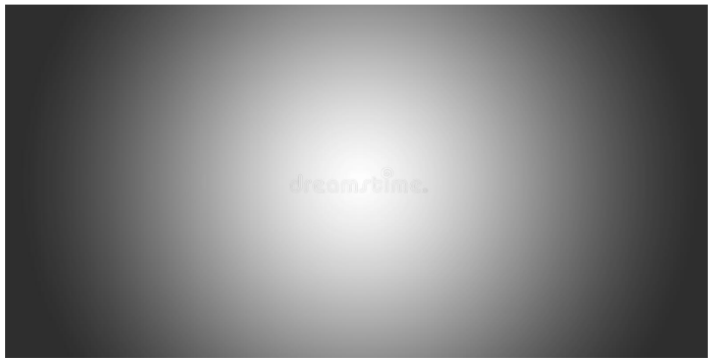
Repeat process for B2

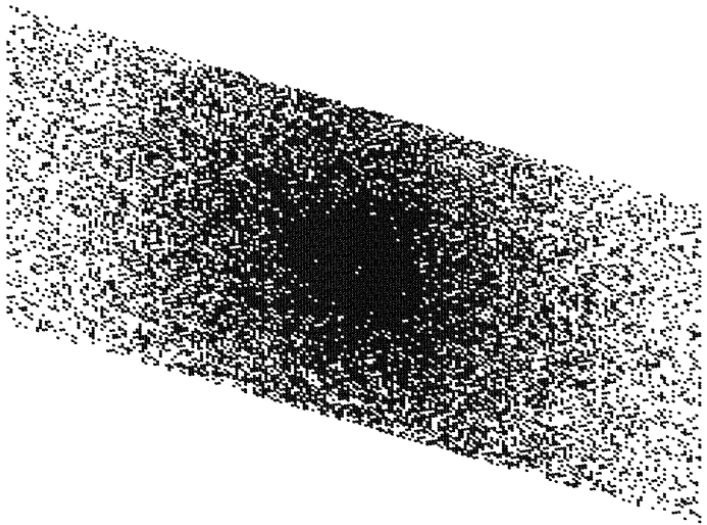
- Repeat the random probability nodes for the B2 brick.



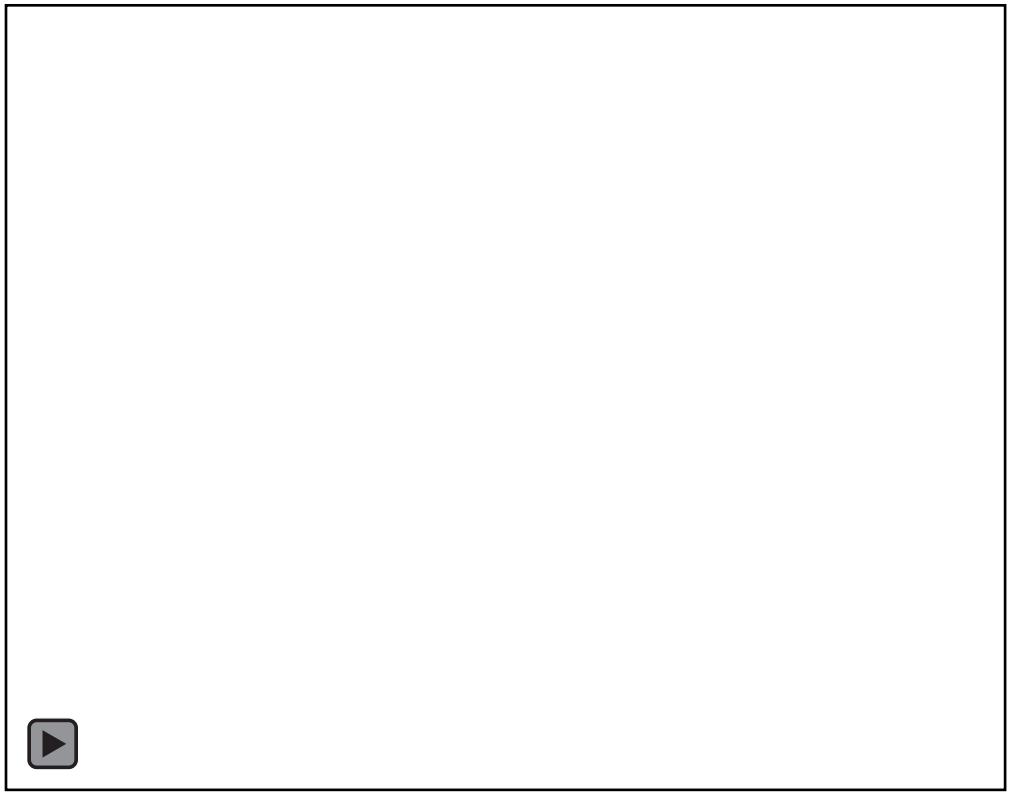
Script is done!

- You can use any image with this script.





Script demo



Completed Design on Griffintown Project

- Rendering done with Enscape.
- Lead designer: Ben Wareing

Thank you!

- Don’t forget to rate the course.
- Download the full dataset: https://revitpure.com/au2022

Revit Pure Pamphlets

- Free quarterly PDF guides focused on specific Revit topics, since 2016.
- Download them all here: revitpure.com/pamphlets

Revit Pure Live

- Seasonal weekly shows with various BIM experts
- Watch the episodes here: youtube.com/revitpure

Revit Pure Courses

- BASICS for beginners
- DESIGN for architect + designers
- MANAGE for BIM managers and advanced users.
- learn.revitpure.com