Building Generator

Intent

To create a building procedurally with a variety of modular mesh pieces pertaining to certain family or style.

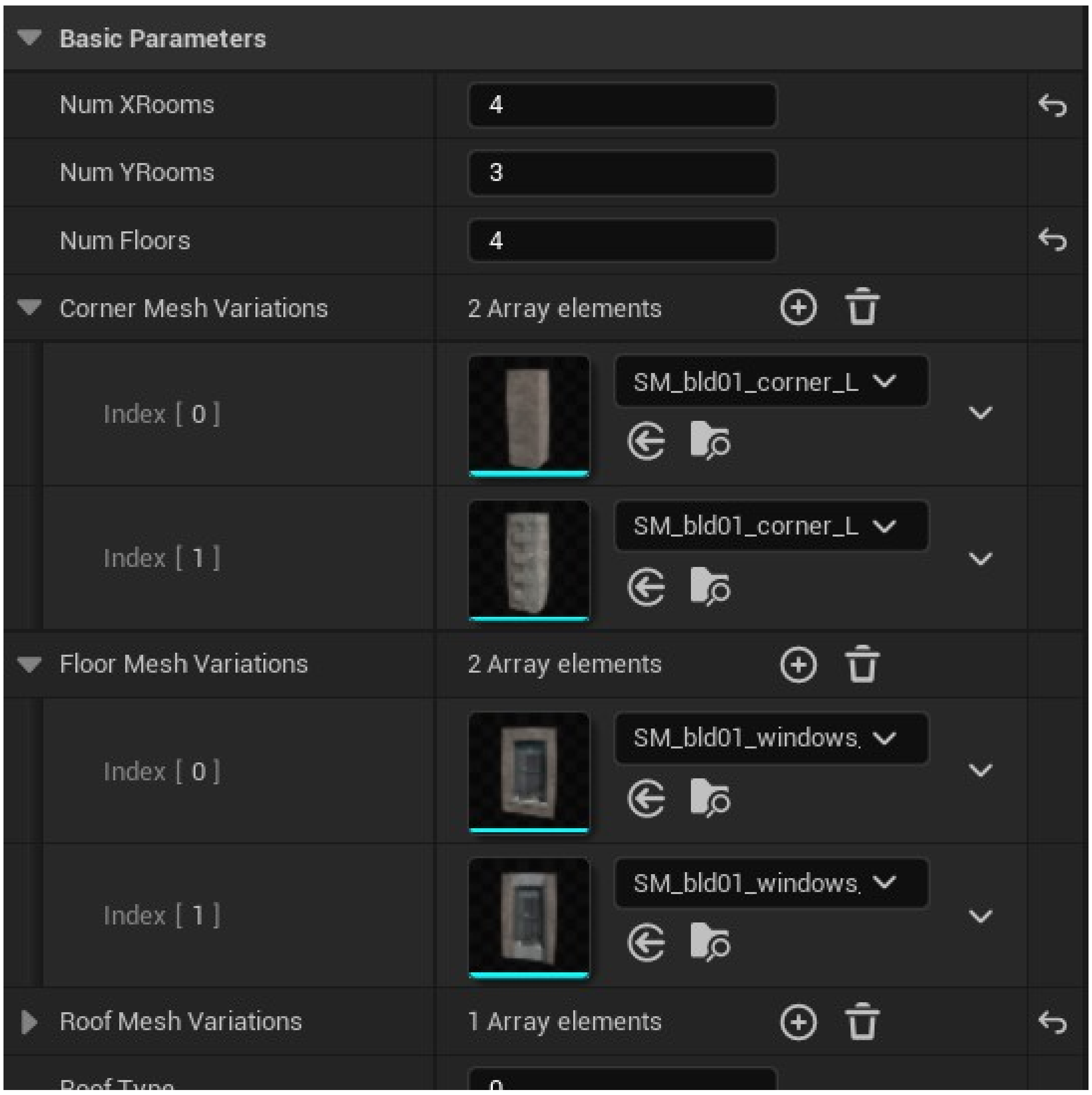
Once building is generated, should be easy to customize the look and feel just by changing the order/pattern of the rooms.

Overview

The key aspect of building generator is entirely based on the creation of a single floor.

The key parameters are divided into two categories:

- Basic Parameters

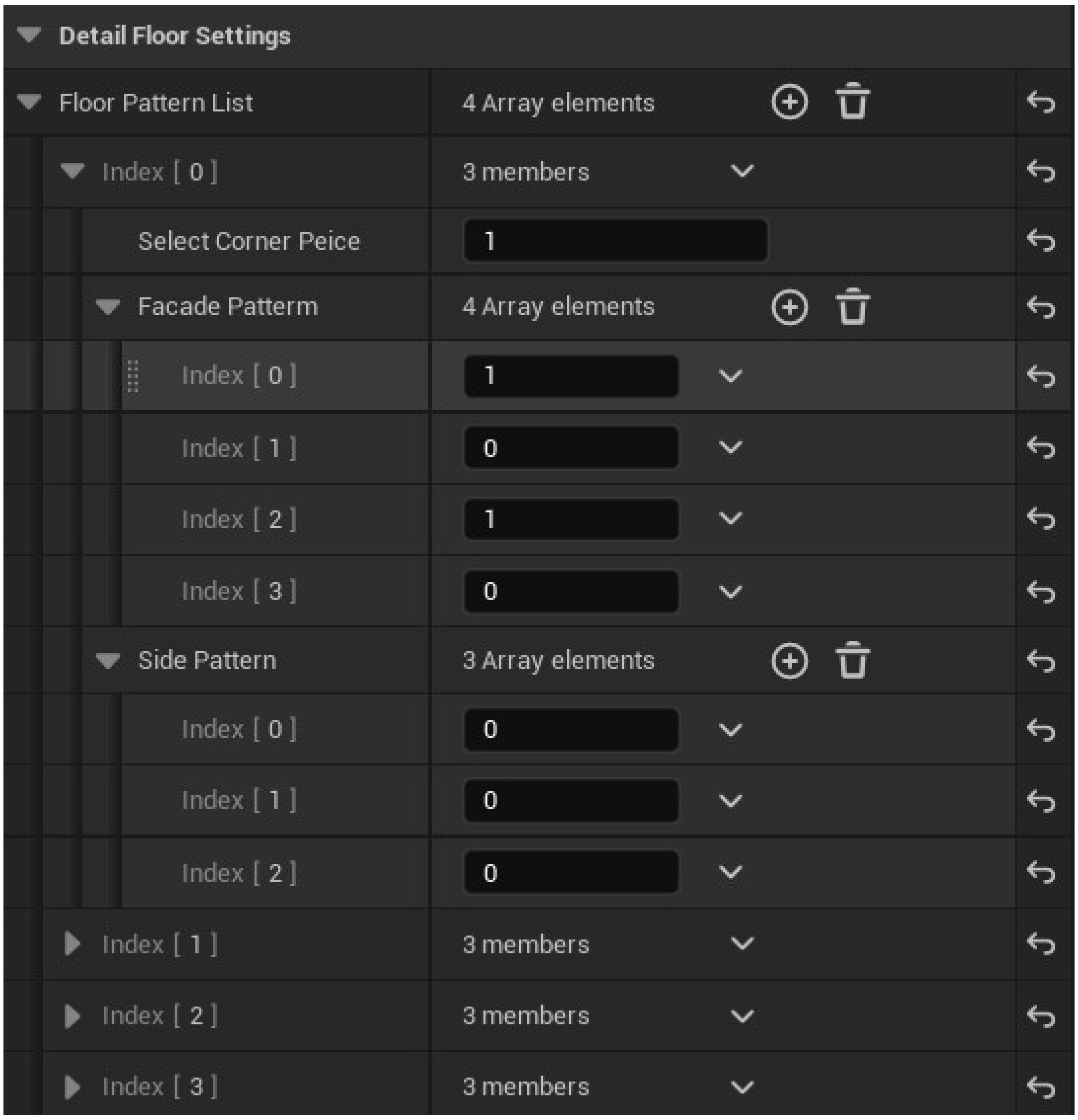


Basic Parameters

Num XRooms	4
Num YRooms	3
Num Floors	4
Corner Mesh Variations	2 Array elements
Index [0]	SM_bld01_corner_L
Index [1]	SM_bld01_corner_L
Floor Mesh Variations	2 Array elements
Index [0]	SM_bld01_windows.
Index [1]	SM_bld01_windows.
Roof Mesh Variations	1 Array elements

  - Setting Up number of rooms and floors
  - Adding modular mesh pieces

- Detailed Parameters



Detail Floor Settings

Floor Pattern List	4 Array elements	
Index [0]	3 members	
Select Corner Peice	1	
Facade Patterm	4 Array elements	
Index [0]	1	
Index [1]	0	
Index [2]	1	
Index [3]	0	
Side Pattern	3 Array elements	
Index [0]	0	
Index [1]	0	
Index [2]	0	
Index [1]	3 members	
Index [2]	3 members	
Index [3]	3 members	

  - providing options for selecting variants for modular mesh pieces.
  - Each room /unit is created with the information provided in the Facade and Side pattern. Similarly the corner pieces can be altered by changing the Select Corner Piece parameter.

> *The blueprint lacks support for roof generation and material variations support.*

Brakedown

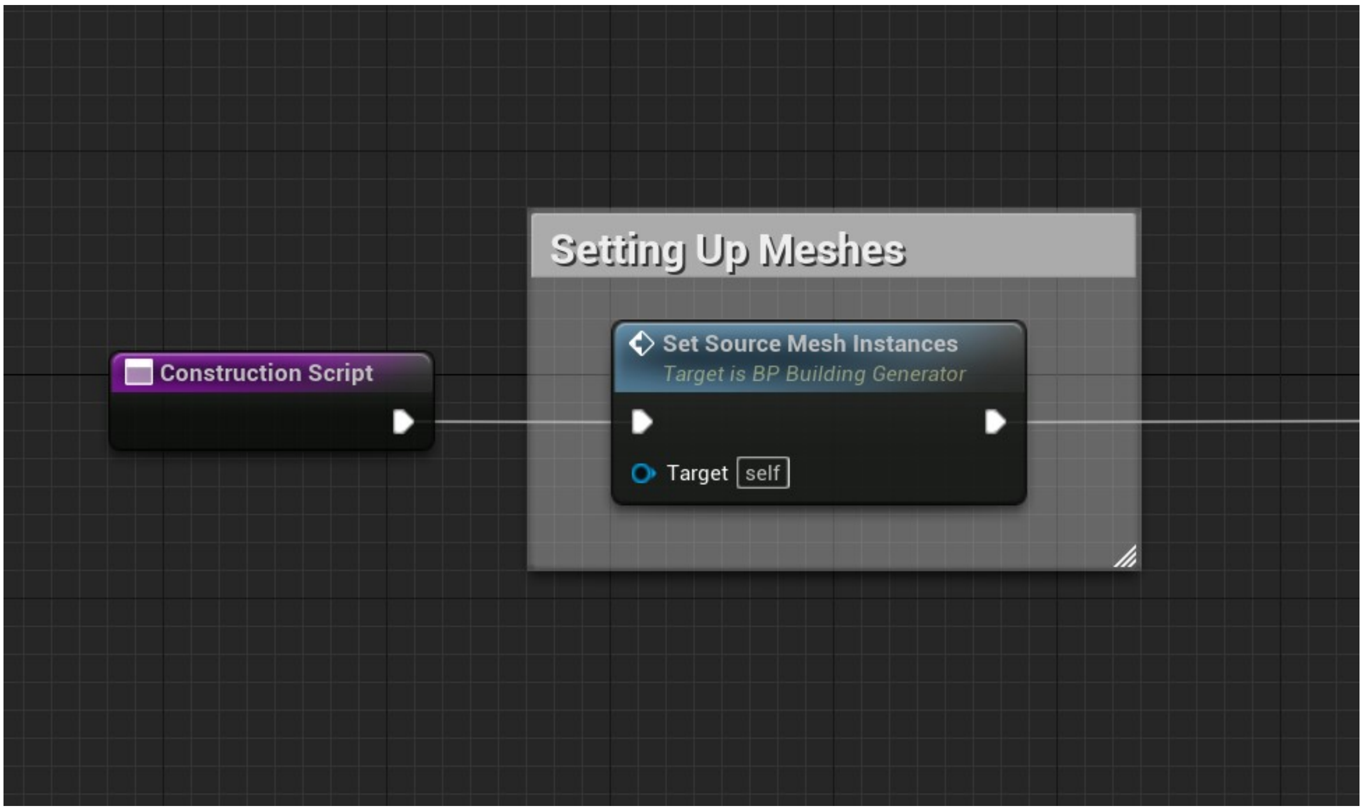
The functionality is divided into four main areas.

- Collection of modular mesh pieces for the room units, corner units and the roof units and setting up instances.
- Setting up arrays for the units along the facing units and units along the side.
- Calculating the over dimensions for the building and also identifying positions for each room unit and corner unit for a floor.
- Creating a function to generate one floor and iterate over the desired number of rooms and floors.

`Modular Pieces`

Storing moduring pieces for room,corners and roof in arrays in order to provide variations.

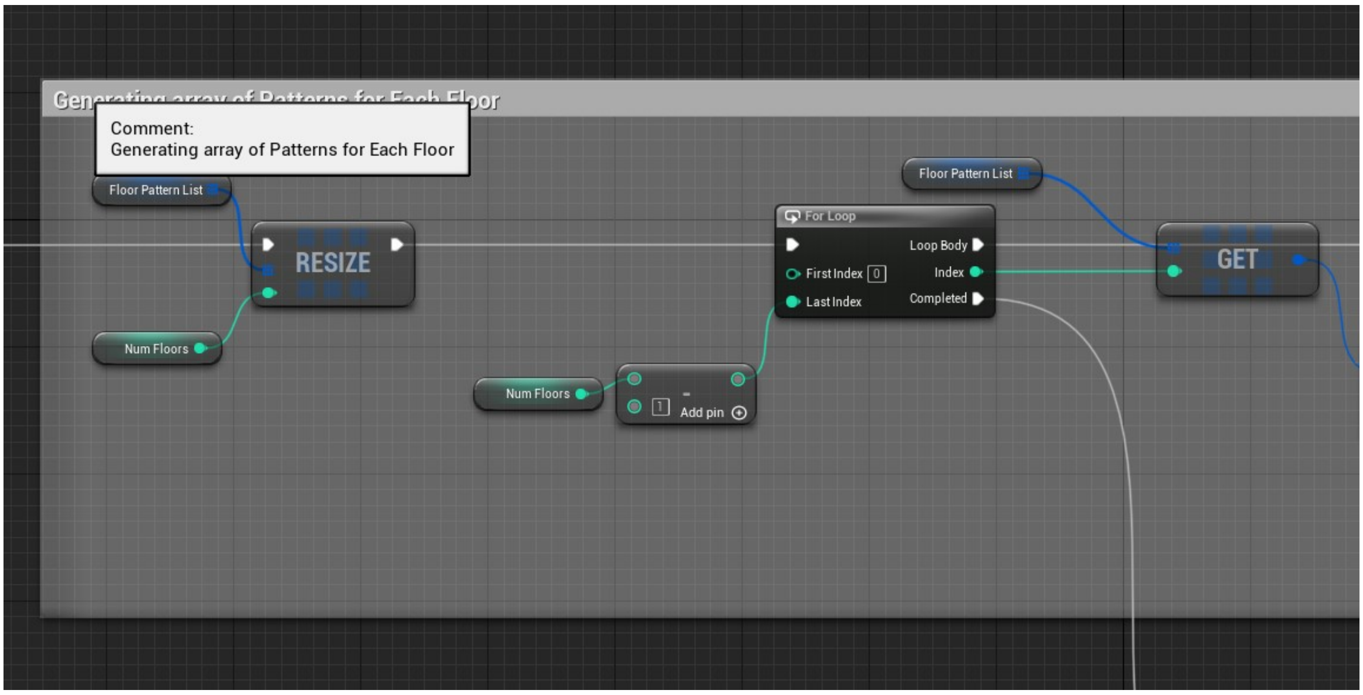
All mesh pieces in a floor need to be of certain height and width.



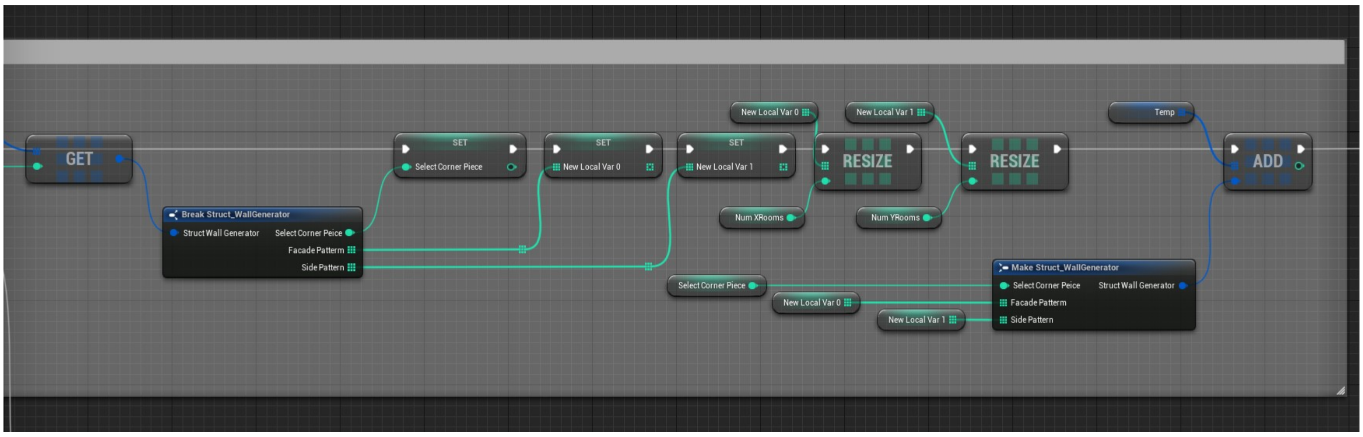
For every mesh in the array an Hierarchial InstanceMesh array is also generated in order have optimization. ![detail params](/images/img03-1.jpg)

Room Variation Arrays

Each floor has two arrays, One array is mapped to builing facing and rear and other is one for the sides of the building.

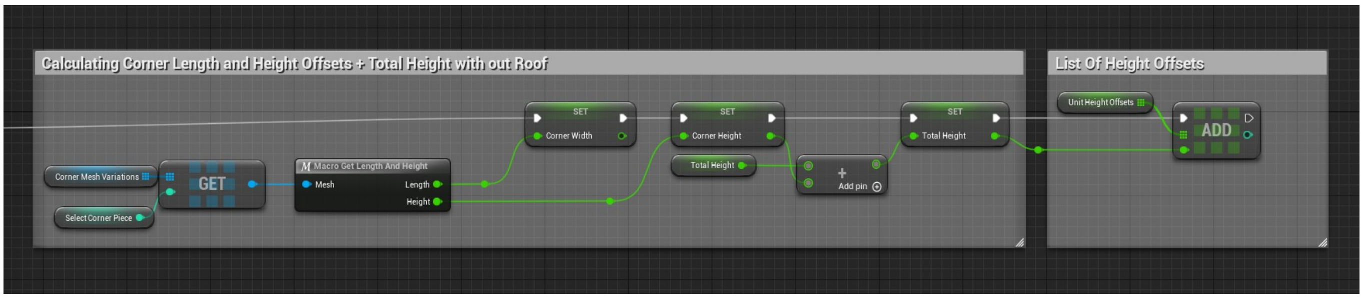


Each value in the array is mapped to the indices of the array where the list of meshes are stored.



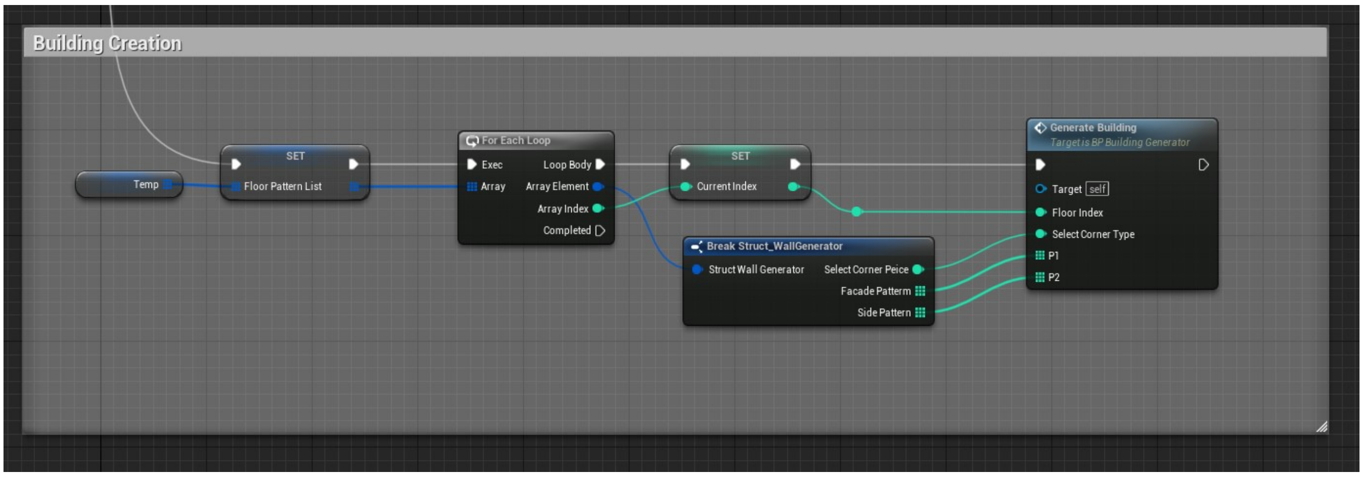
Calculating Unit and Total Dimensions

Calculating key dimension values required in further functions.

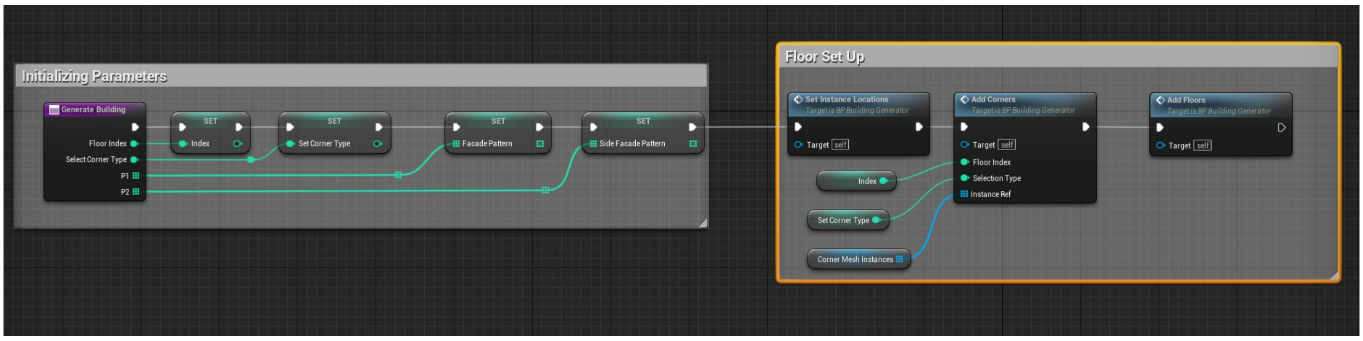


Generating Floors

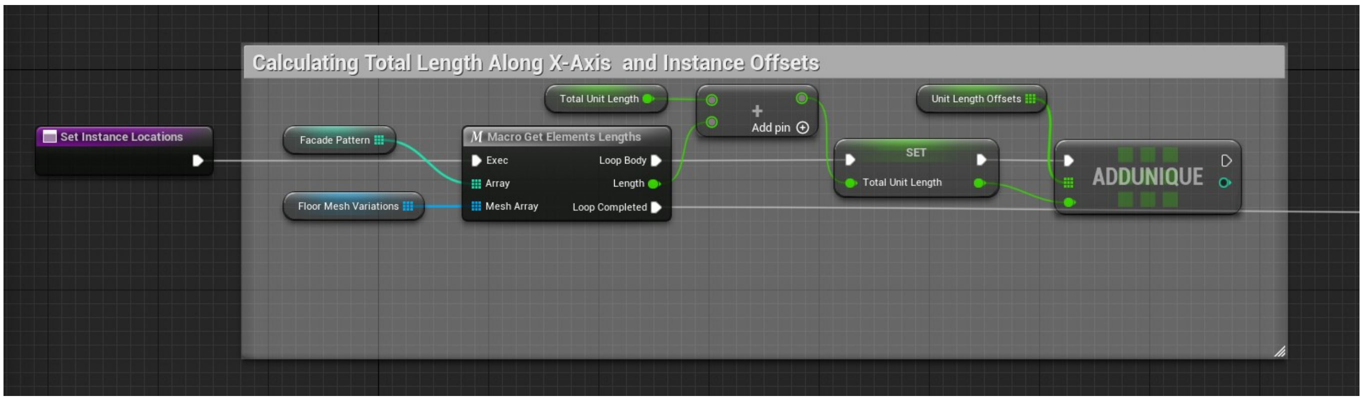
The floor variation parameters are stored in an array of struct and each then passed on to a floor generator function.

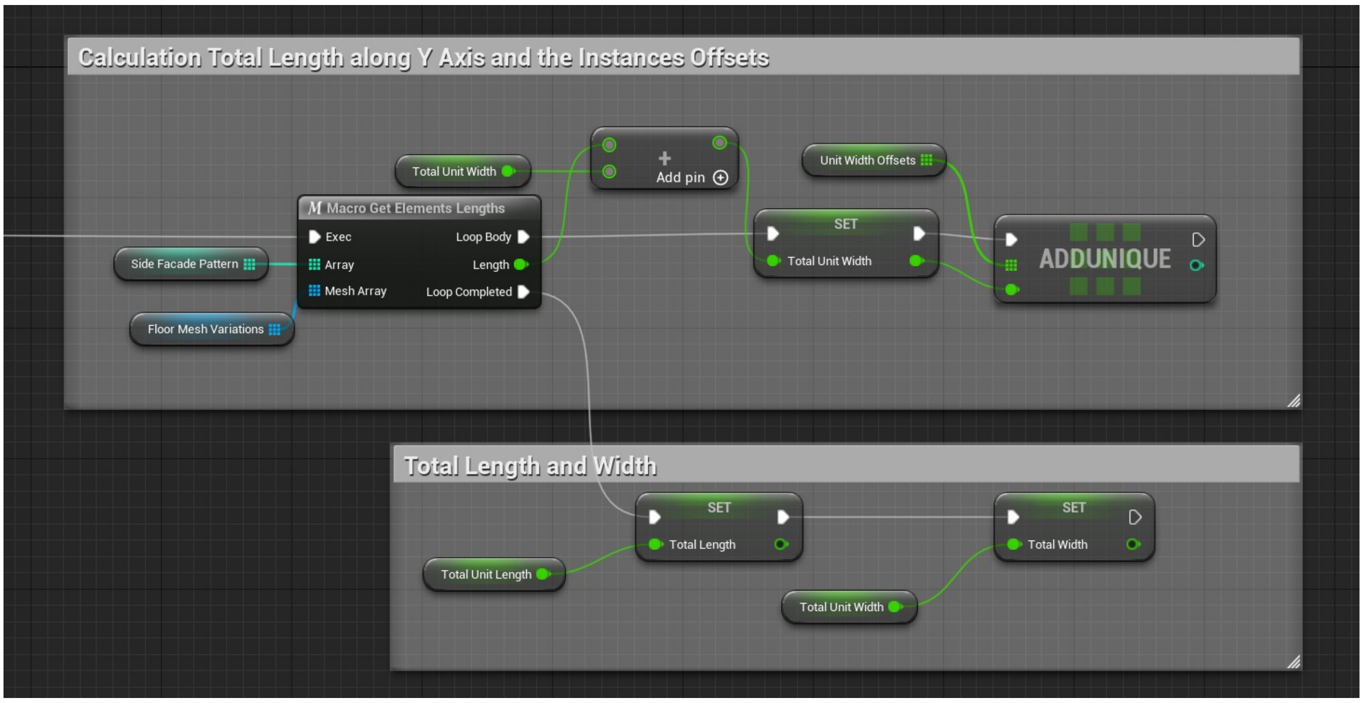


The floor generator function is futher divided into three functions.

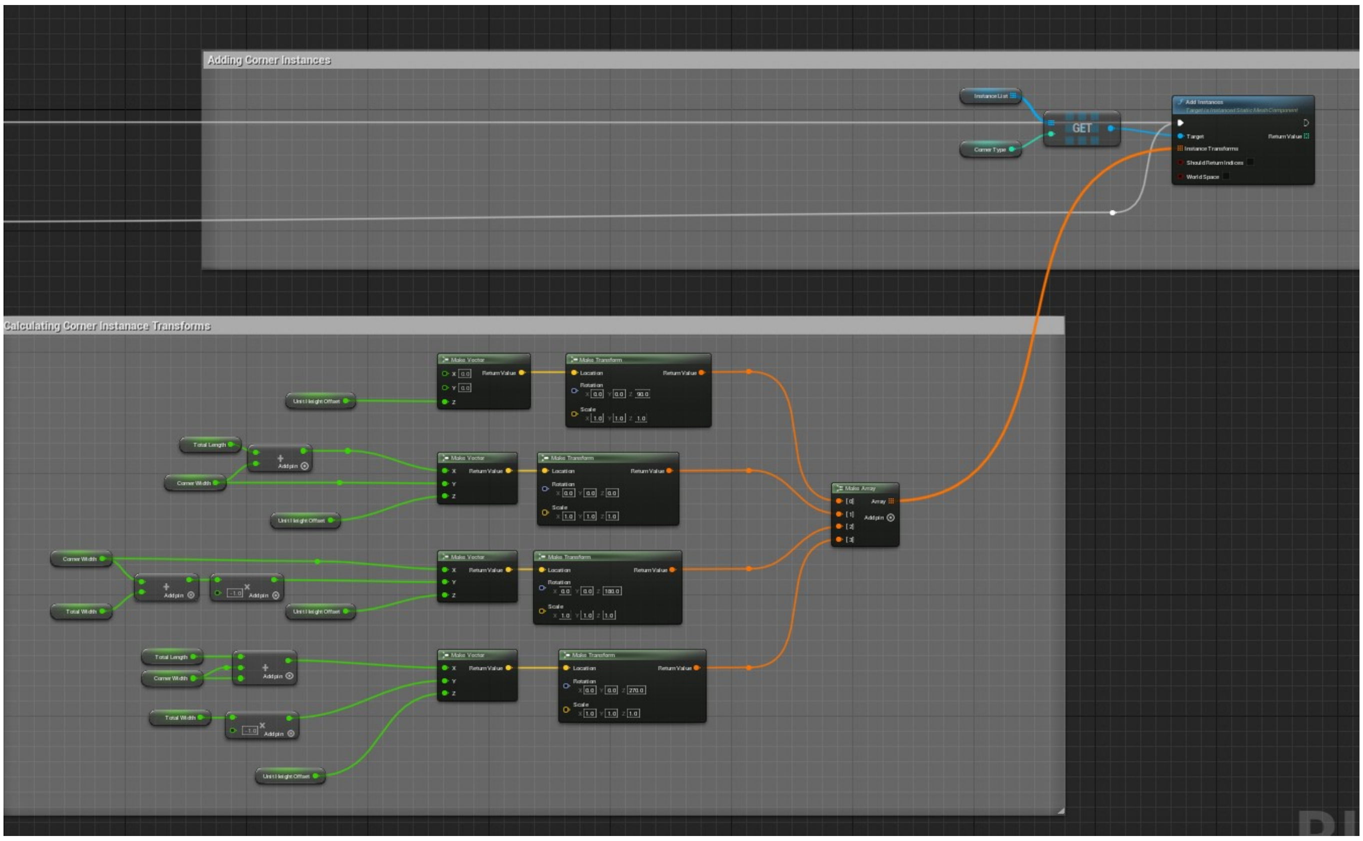


- Calculation positions for each room unit based on the pattern and storing it in an array.

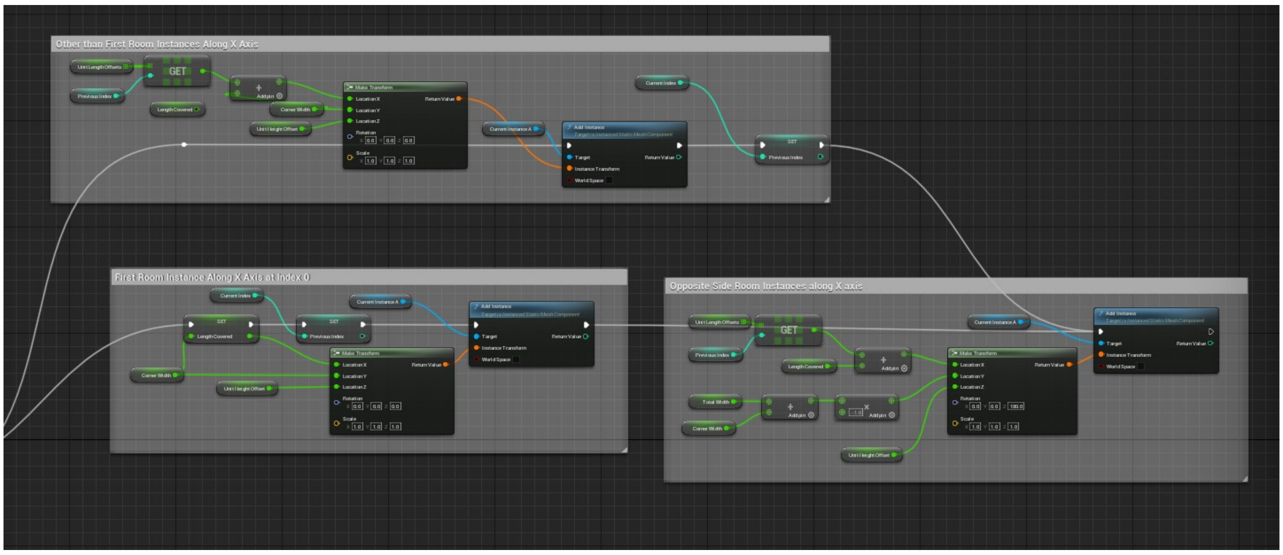




- Identifying corner positions and adding corner piece instances



- Finally passing the array of positions and adding instances for each unit. Also calculating position for opposites sides as well


Other than First Room Instances Along X Axis
First Room Instance Along X Axis at index 0
Opposite Side Room Instances along X axis
GET
Make Transform
Add Instance

Improvements for the Building generator

- Floor pattern generation could be automated and simplified for better end user experience.
- Need to accomodate material variations as well.
- Functions could have been more efficient and re-usable.

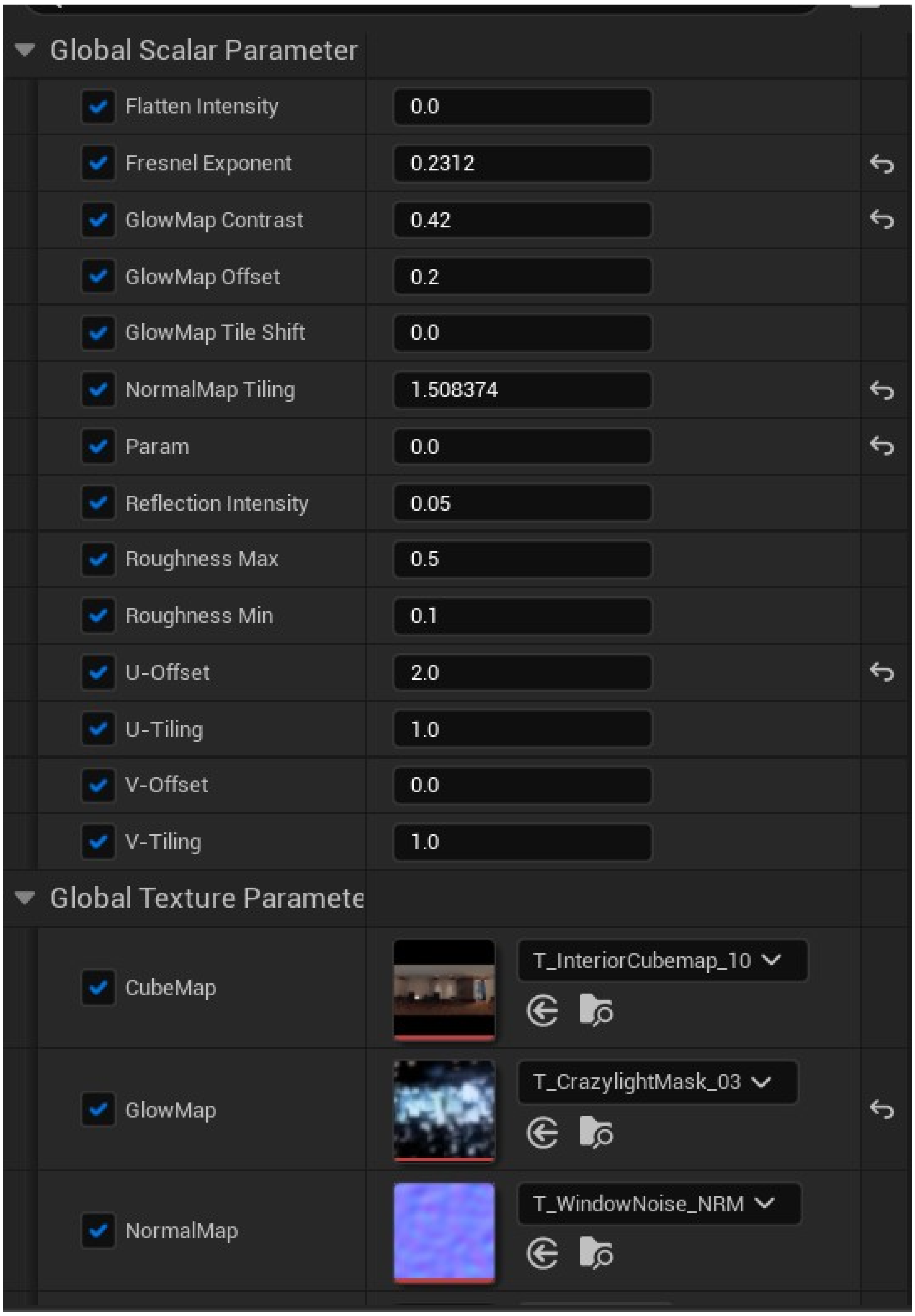
Storefront Material

Inspiration

> UE4 Live Training session by Ryan Brucks on Creation of Semi Procedural Materials.

Overview

The main aspect of the store front material is to render a fake interiors with the help of a baked cube map, and have parameters to control look and feel.



Global Scalar Parameter

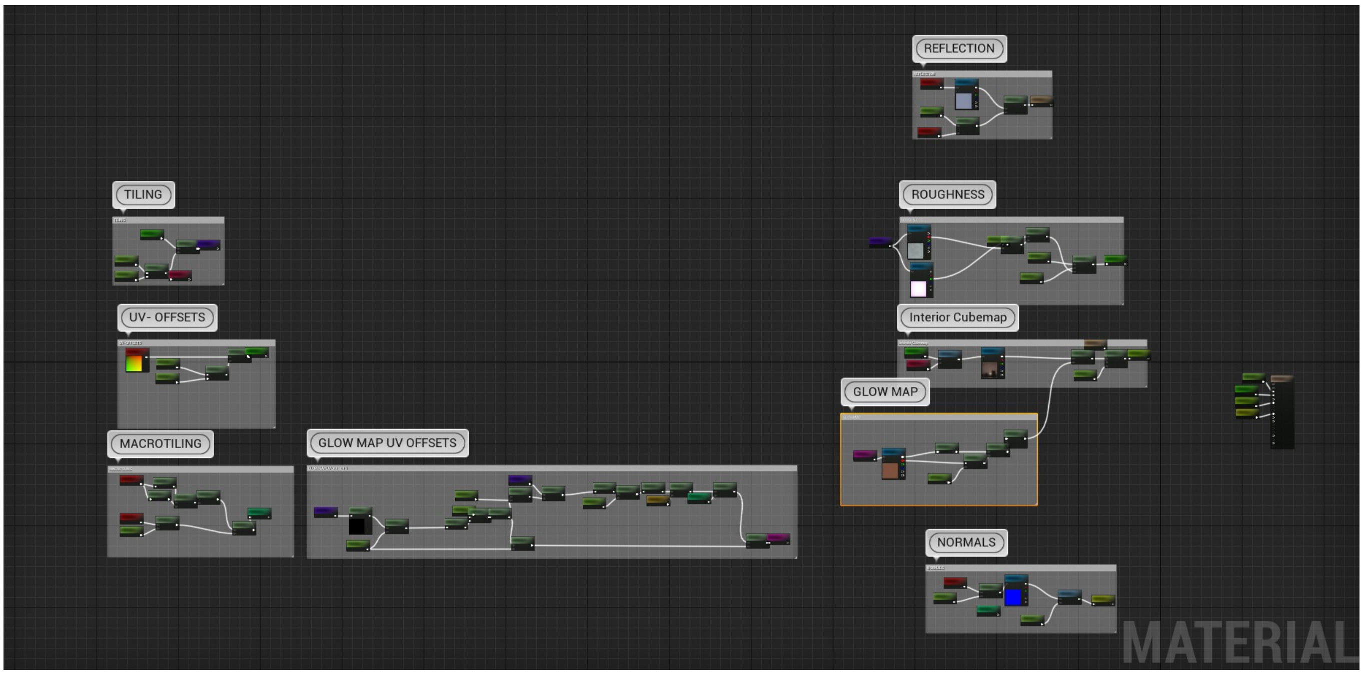
✓ Flatten Intensity	0.0
✓ Fresnel Exponent	0.2312
✓ GlowMap Contrast	0.42
✓ GlowMap Offset	0.2
✓ GlowMap Tile Shift	0.0
✓ NormalMap Tiling	1.508374
✓ Param	0.0
✓ Reflection Intensity	0.05
✓ Roughness Max	0.5
✓ Roughness Min	0.1
✓ U-Offset	2.0
✓ U-Tiling	1.0
✓ V-Offset	0.0
✓ V-Tiling	1.0

Global Texture Paramete

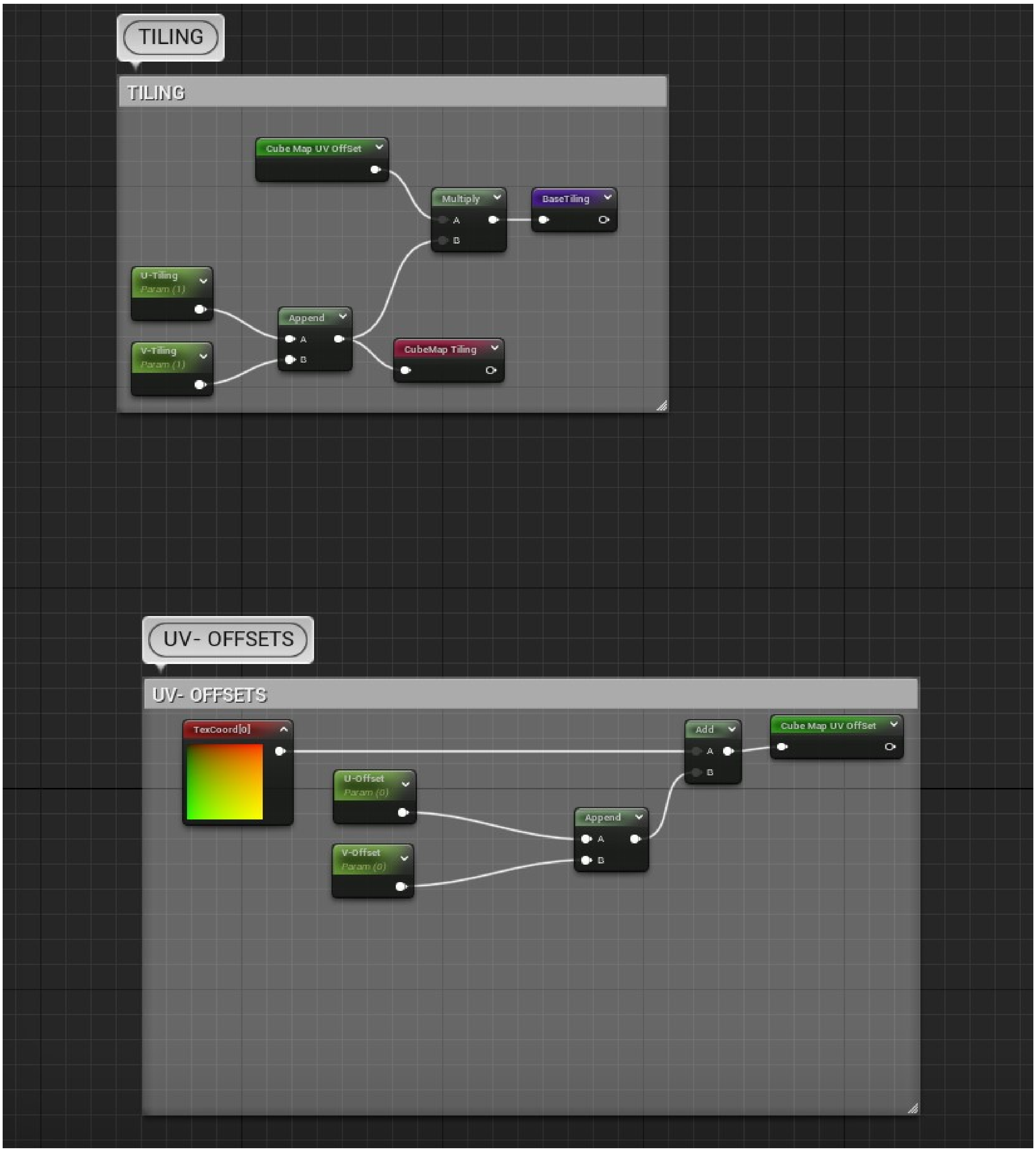
✓ CubeMap	T_InteriorCubemap_10
✓ GlowMap	T_CrazylightMask_03
✓ NormalMap	T_WindowNoise_NRM

Brakedown

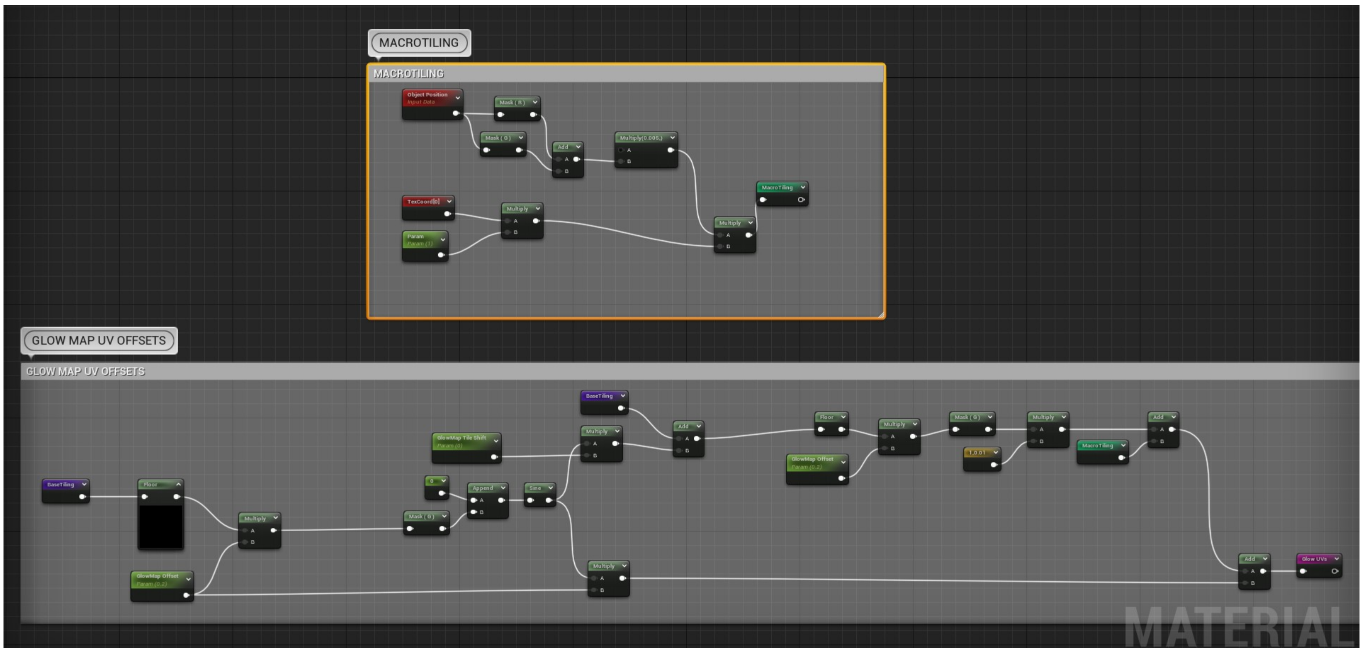
The cube map is manipulated with UV tiling and UV offsets and on top the variations like random values of the brightness,contrast, glow are overlayed with Roughness and Reflections.


REFLECTION
TILING
ROUGHNESS
UV- OFFSETS
Interior Cubemap
GLOW MAP
MACROTILING
GLOW MAP UV OFFSETS
NORMALS
MATERIAL

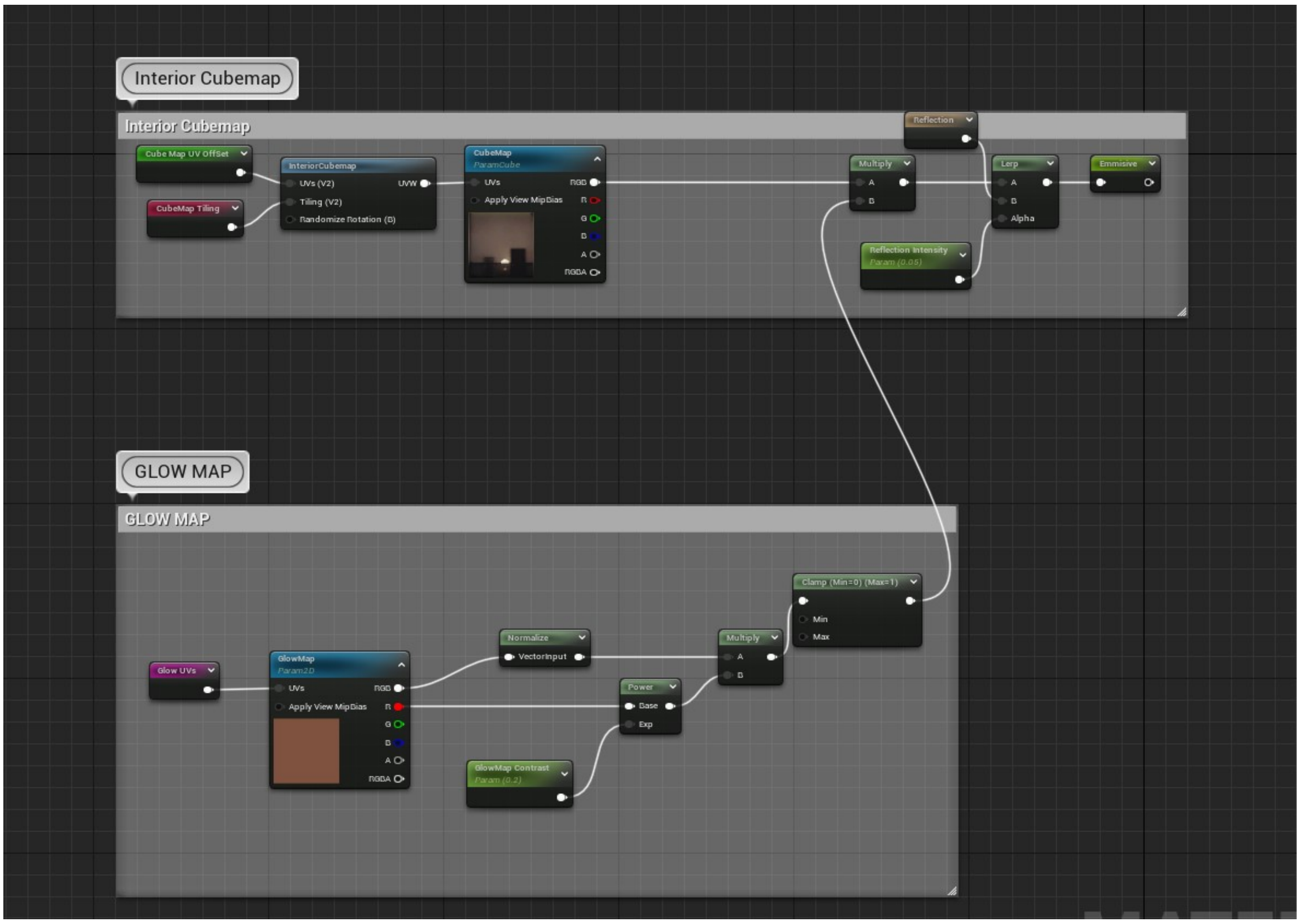
Creating UV Tiling and UV Offsets for the interior Cube map



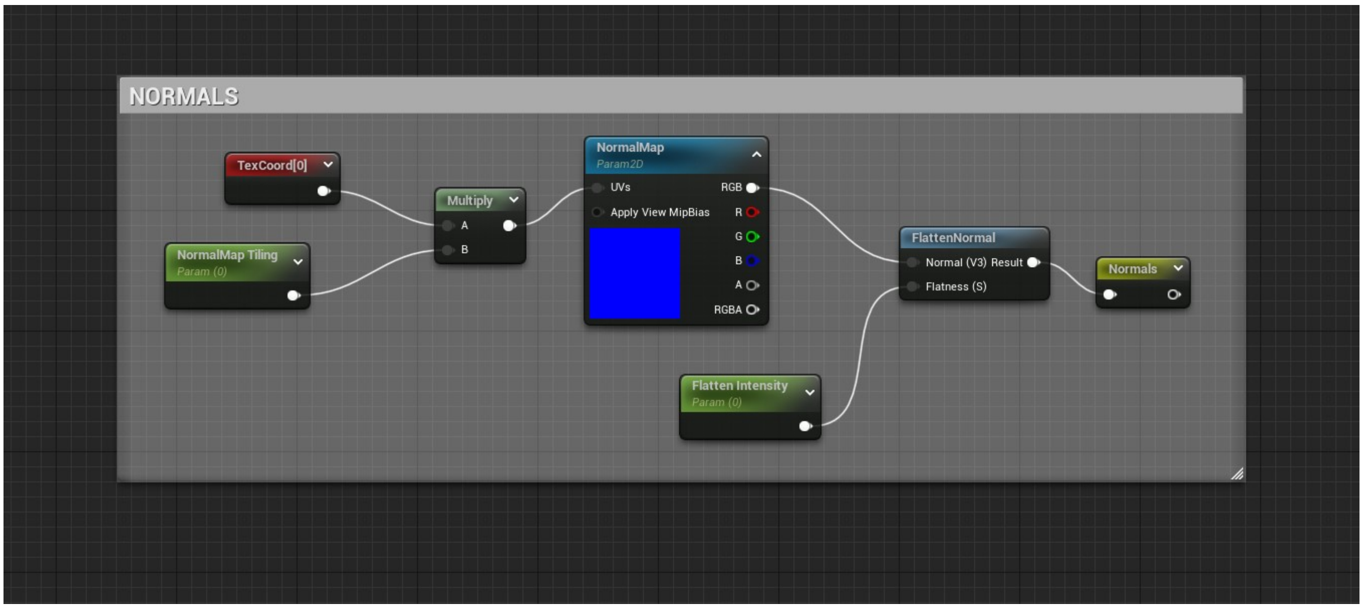
Creating UVs for Color Overlay Variation

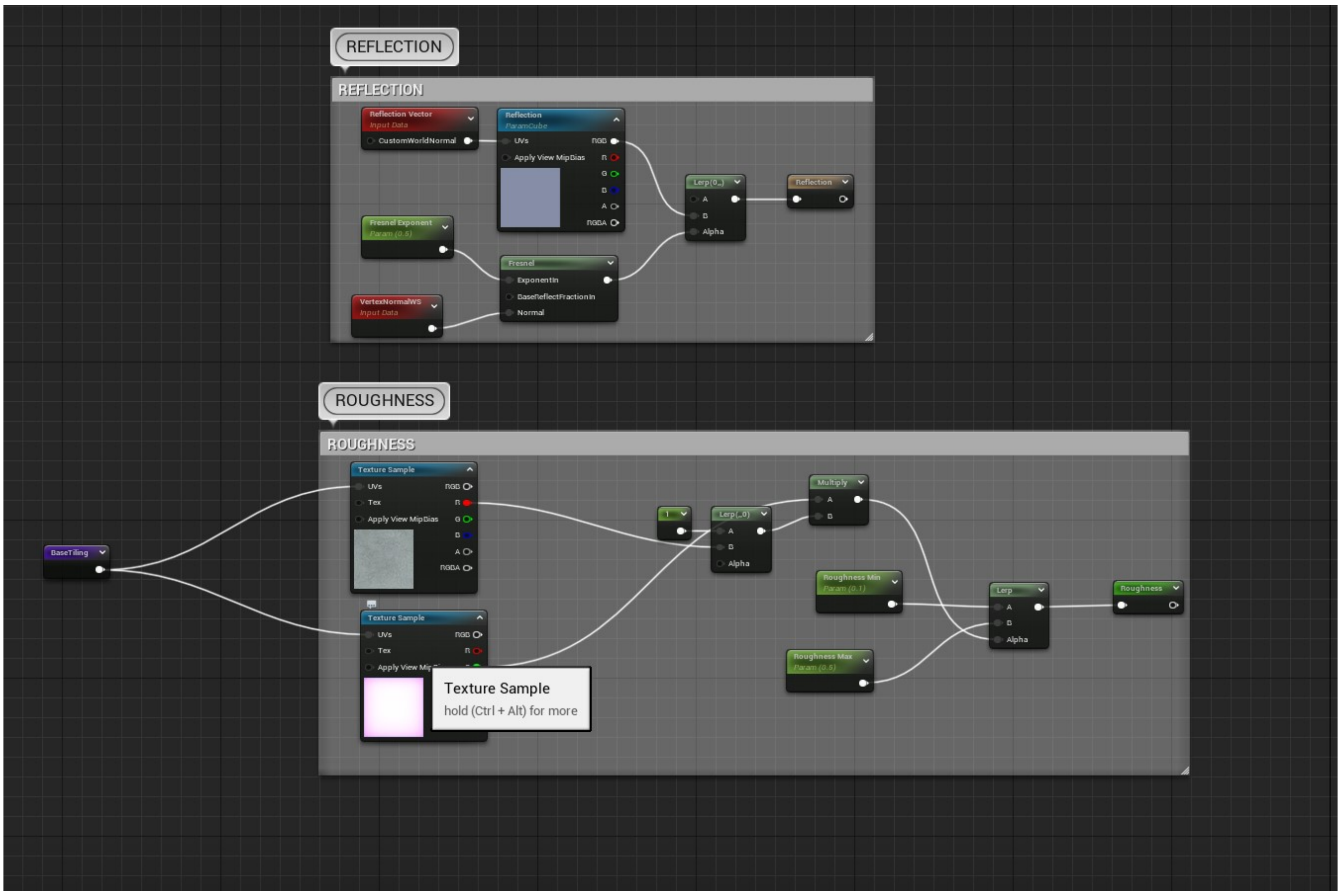


Setting up Cube Map Textures and GlowMap Textures

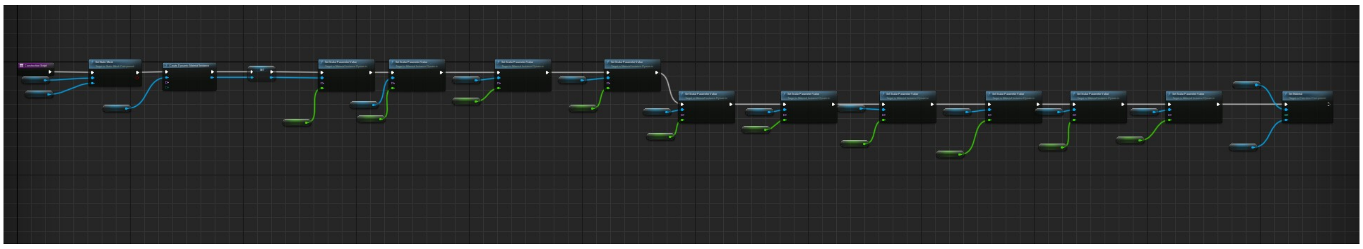


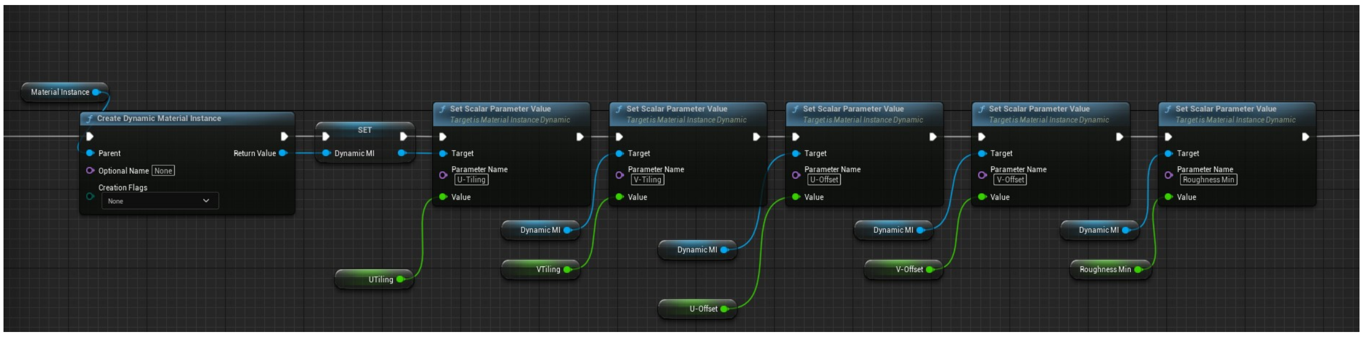
Setting up Normals,Roughness and Reflections to simulate reflection on the Glass





Mapping Scalar Parameters to a blueprint




Material Instance
Create Dynamic Material Instance
Parent
Optional Name None
Creation Flags
None
Return Value
SET
Dynamic MI
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name U-Tiling
Value
UTiling
Dynamic MI
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name V-Tiling
Value
VTiling
Dynamic MI
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name U-Offset
Value
U-Offset
Dynamic MI
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name V-Offset
Value
V-Offset
Dynamic MI
Set Scalar Parameter Value
Target is Material Instance Dynamic
Target
Parameter Name Roughness Min
Value
Roughness Min

Improvements

- Features like adding curtains or blinds.
  Needed more understanding and experimentation

- Adding Glass grunge around the borders of the glass