# Procedural Building 2.5

V 2.5.0

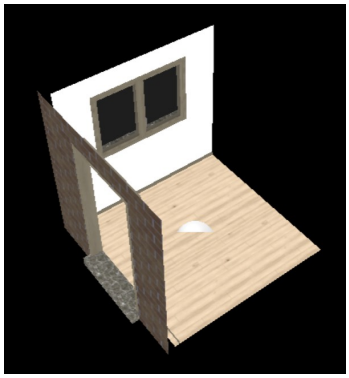
# Procedural Building

## Content

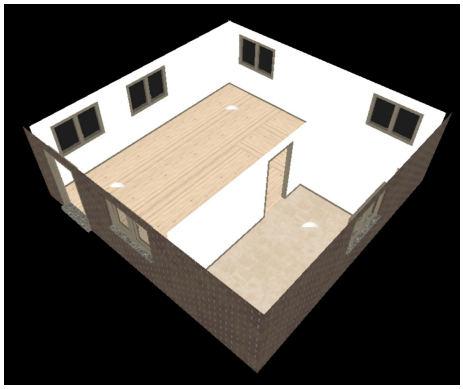
# 1 - Introduction

Procedural Building is a library that allows you to quickly generate buildings in your scene. This plugin also generates the interior areas of the building, including interior walls, doors, corridors and stairs. It has several parameters to configure the building to your liking, such as the floors it has, the number of corridors or if the building has foundations.

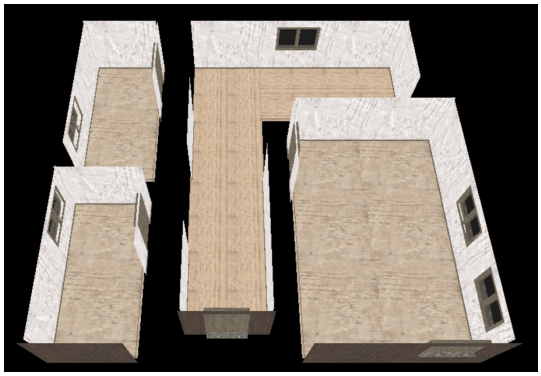
The building is made up of cells, to add them, rectangles with a configurable width and depth are created, you can also configure how many rectangles you want to be generated.
Cells contain parts of the building, such as walls, windows, and doors. A set of cells form a module. The modules can be of 4 types: Corridor, Room, Stairs, BaseRoofColumn.



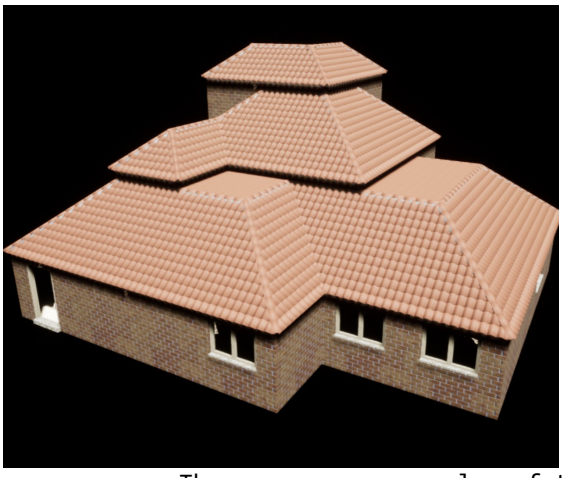
Cell

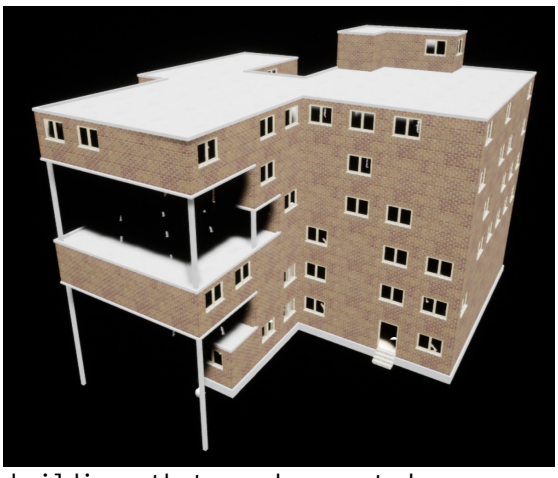


Rectangle 3x3 cells



Modules example
(1 Corridor, 3 rooms)





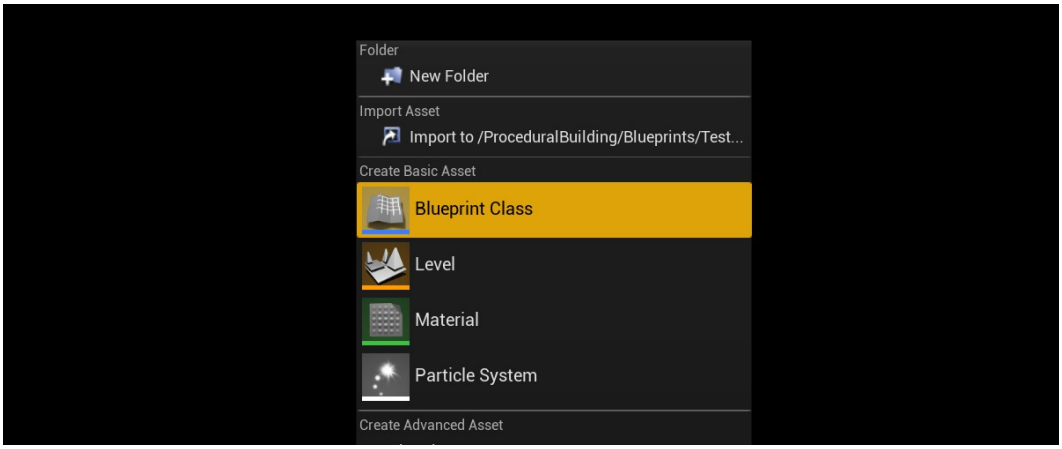
These are some examples of the buildings that can be created.

## 2 - First steps

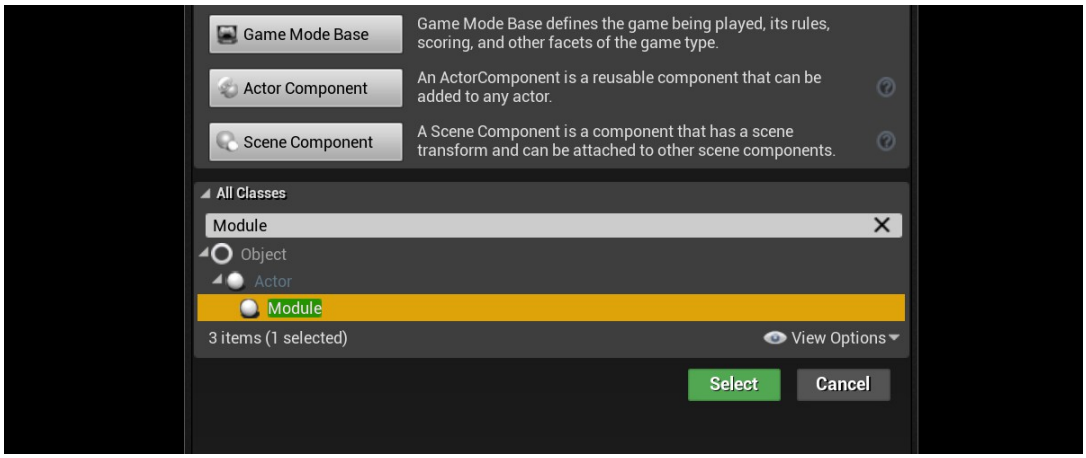
Create the modules that make up the building:

To generate the building, four module-type blueprints need to be created (Corridor, Room, Stair y BaseRoofColumn). To create the modules follow these steps:

1º
In the Content Browser, go to the folder where you want to create the blueprint.




2º
Under All Classes type Module, below, select the Module class and click Select.

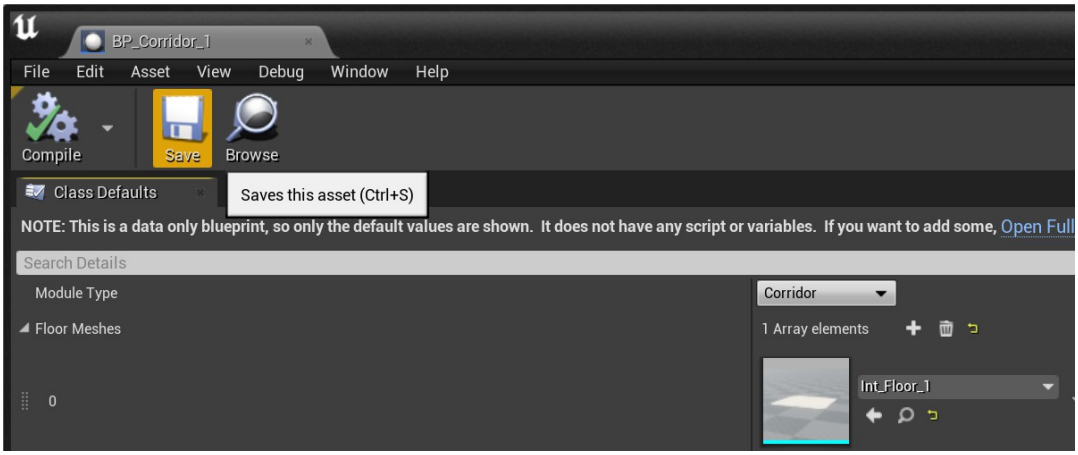



3º
Type a name for your new blueprint. I will call it BP_Corridor_1 since it will correspond to the corridors of the building.

4º
Double click on the created blueprint.

5º
In Module Type, we'll leave it at Corridor. We add the meshes that we need to Floor, Interior Wall, Interior Corner, Exterior Wall, Exterior Window, Exterior Corner and Ceiling. We also added the Blueprint for the front door.

6º
Once all the meshes are configured, click on Save and Compile.




7º
We repeat from the 3rd step until we have the necessary blueprints (Corridor, Room, Stair y BaseRoofColumn). In Module Type, we can select the type of module to which it belongs.

Create a building:

1º
Access the path ProceduralBuilding Content/Blueprints/ProceduralBuilding in the Content Browser.

2º
Drag the Blueprint BP_Building onto the scene.

3º
In the World Outliner, select BP_Building.

4º
In Config Building, we edit the parameters to our liking.

5º
In Corridors Module, Rooms Module, Stairs Module and Base Roof Column Module, we add the corresponding blueprints created earlier.

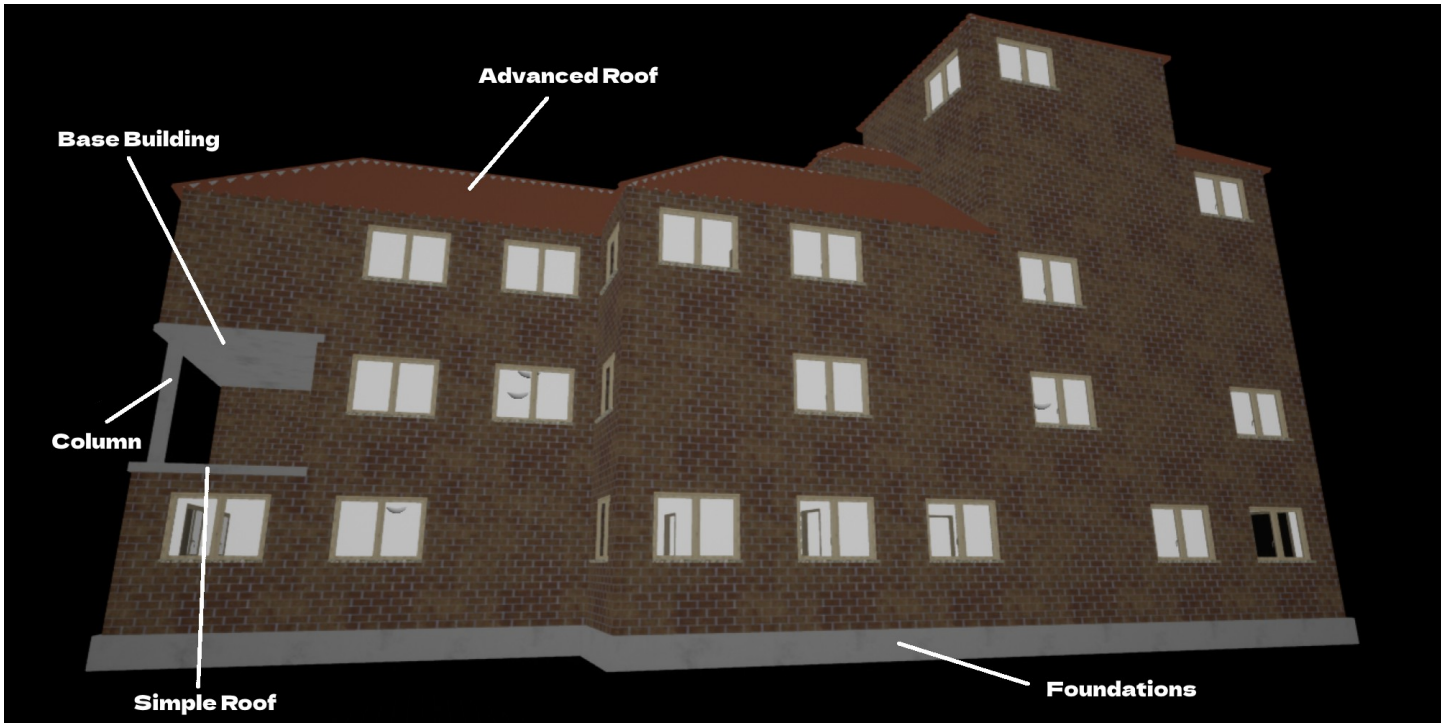
6º
We click on Build to create a building with the values assigned in step 4. We can also click on Random Building to generate a building with random values.

## 3 - Building buttons

- Build: Generates the building with the entered parameters.
- Delete Build: Delete all the content generated with the Build button.
- Random Build: Set the building parameters randomly and generate it. Modules must be included.
- Random Interior Seed: Assign a random seed to the building's interiors and generate it.
- Random Seed: Assign a random seed to the building and generate it.

## 4 - Building parameters

- Tag Name: Assigns a tag to the building.
- Building Seed: Sets the building seed.
- Interior Seed: Set the building's interior seed.
- Cell Size XY: Allows you to set the size of the cell on the X axis.
- Cell Size Z: Allows you to set the size of the cell on the Z axis.
- Max Building Floors: Set maximum building floors.
- Iterations: It is the number of repetitions that the rectangles will have when they are generated.
- Min Rect Cells X: Minimum number of cells on the X axis that the rectangle will have.
- Max Rect Cells X: Maximum number of cells on the X axis that the rectangle will have.
- Min Rect Cells Y: Minimum number of cells on the Y axis that the rectangle will have.
- Max Rect Cells Y: Maximum number of cells on the Y axis that the rectangle will have.
- Corridors In Floor: Number of corridors that the building will have.
- Min Corridor Long: Minimum number of cells that will have the length of the corridor.
- Max Corridor Long: Maximum number of cells that will have the length of the corridor.
- Random Entry Orientation: Place the building's entry in a random orientation (North, South, East, or West).
- Orientation Entry: Set the orientation where the entry will be created. This option is only displayed if Random Entry Orientation is turned off.
- Corridors Module: Contains the different types of corridors that the building will have. Depending on the number of corridor modules that are added, take a random one for each floor.
- Rooms Module: Contains the different types of rooms.
- Random Rooms Module: Select a random module from the list and add it to the building. If this option is not enabled, modules are added sequentially.
- Room Generation Type: Set how rooms are created.
  - Random: Add cells to rooms randomly.
  - Rectangle: Attempt to add cells to rooms in a rectangular shape with a size specified in Max Room Size X and Max Room Size Y.
- Stairs Module: Contains the different types of stairs that the building will have. There is only one staircase for each floor to communicate this with the next. A stair module is randomly selected for each floor.
- Base Roof Column Module: Contains the module to be able to draw the foundations (if they are activated), the base of the floors when there is nothing below, the roof and the columns.




- Exclude Roof And Ceiling: If this option is enabled, it does not create roofs and ceilings.
- Add Foundations: Add foundations to the building. They must be included in BaseRoofColumn to be drawn.

## 5 - Modules

The building is made up of cells but to draw the building on the screen, the cells are grouped forming modules depending on the values of the cells..

Modules are blueprints of the Module class. To create a module in Unreal, within the content browser, we go to the folder where we want to create the module and follow these steps:

- Click with the right mouse button and select Blueprint Class.
- Under All Classes type Module, below, select the Module class and click Select.
- Write a name for your new blueprint.

### 5.1 Types

There are four types of modules, Corridor, Room, Stair and BaseRoofColumn. Each of these modules contain the data necessary to draw the building. When creating a module, in the Module Type dropdown, you can select the type of module.

### 5.2 Corridor parameters

- Floor Meshes: Contains the models for the floors that make up the corridor.
- Floors Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Floor Offsets: Use the same offsets for all added Floor meshes.
- Floors Meshes Offsets: Add an offset position to all added Floor meshes.
- Floor Meshes Offsets: Assign an independent offset position for each Floor mesh.

- Interior Wall Meshes: Contains the models for the interior walls that form the corridor.
- Same Interior Wall Offsets: Use the same offsets for all added Interior Wall meshes.
- Interior Walls Meshes Offsets: Add an offset position to all added Interior Wall meshes.
- Interior Wall Meshes Offsets: Assign an independent offset position for each Interior Wall mesh.

- Interior Corner Meshes: Contains the models for the interior corners that form the corridor.
- Same Interior Corner Offsets: Use the same offsets for all added Interior Corner meshes.
- Interior Corners Meshes Offsets: Add an offset position to all added Interior Corner meshes.
- Interior Corner Meshes Offsets: Assign an independent offset position for each Interior Corner mesh.

- Exterior Wall Meshes: Contains the models for the exterior walls that form the corridor.
- Same Exterior Wall Offsets: Use the same offsets for all added Exterior Wall meshes.
- Exterior Walls Meshes Offsets: Add an offset position to all added Exterior Wall meshes.
- Exterior Wall Meshes Offsets: Assign an independent offset position for each Exterior Wall mesh.

- Frequency Window Meshes: Probability of window spawn.
- Exterior Window Meshes: Contains the models for the exterior windows that form the corridor.
- Same Exterior Window Offsets: Use the same offsets for all added Exterior Window meshes.
- Exterior Windows Meshes Offsets: Add an offset position to all added Exterior Window meshes.
- Exterior Window Meshes Offsets: Assign an independent offset position for each Exterior Window mesh.
- Draw Window With Roof: Allows drawing windows facing the roof.

- Exterior Corner Meshes: Contains the models for the exterior corners that form the corridor.
- Same Exterior Corner Offsets: Use the same offsets for all added Exterior Corner meshes.
- Exterior Corners Meshes Offsets: Add an offset position to all added Exterior Corner meshes.
- Exterior Corner Meshes Offsets: Assign an independent offset position for each Exterior Corner mesh.

- Ceiling Meshes: Contains the models for the interior ceilings that form the corridor.
- Ceilings Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Ceiling Offsets: Use the same offsets for all added Ceiling meshes.
- Ceilings Meshes Offsets: Add an offset position to all added Ceiling meshes.
- Ceiling Meshes Offsets: Assign an independent offset position for each Ceiling mesh.
- Entry Door Blueprint: Contains the blueprint for the entrance door of the building.
- Entry Door Blueprint Offsets: Add an offset position to Entry Door Blueprint.

**5.3 Room parameters**

- Limit Rooms Generated: Set the maximum number of rooms in the building.
- Limit Cells in Room: Define the minimum and maximum number of cells that a room will have.

- Floor Meshes: Contains the models for the floors that make up the room.
- Floors Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Floor Offsets: Use the same offsets for all added Floor meshes.
- Floors Meshes Offsets: Add an offset position to all added Floor meshes.
- Floor Meshes Offsets: Assign an independent offset position for each Floor mesh.

- Interior Wall Meshes: Contains the models for the interior walls that make up the room.
- Same Interior Wall Offsets: Use the same offsets for all added Interior Wall meshes.
- Interior Walls Meshes Offsets: Add an offset position to all added Interior Wall meshes.
- Interior Wall Meshes Offsets: Assign an independent offset position for each Interior Wall mesh.

- Interior Corner Meshes: Contains the models for the interior corners that make up the room.
- Same Interior Corner Offsets: Use the same offsets for all added Interior Corner meshes.
- Interior Corners Meshes Offsets: Add an offset position to all added Interior Corner meshes.
- Interior Corner Meshes Offsets: Assign an independent offset position for each Interior Corner mesh.

- Exterior Wall Meshes: Contains the models for the exterior walls that make up the room.
- Same Exterior Wall Offsets: Use the same offsets for all added Exterior Wall meshes.
- Exterior Walls Meshes Offsets: Add an offset position to all added Exterior Wall meshes.
- Exterior Wall Meshes Offsets: Assign an independent offset position for each Exterior Wall mesh.

- Frequency Window Meshes: Probability of window spawn.
- Exterior Window Meshes: Contains the models for the exterior windows that make up the room.
- Same Exterior Window Offsets: Use the same offsets for all added Exterior Window meshes.
- Exterior Windows Meshes Offsets: Add an offset position to all added Exterior Window meshes.
- Exterior Window Meshes Offsets: Assign an independent offset position for each Exterior Window mesh.
- Draw Window With Roof: Allows drawing windows facing the roof.

- Exterior Corner Meshes: Contains the models for the outer corners that make up the room.
- Same Exterior Corner Offsets: Use the same offsets for all added Exterior Corner meshes.
- Exterior Corners Meshes Offsets: Add an offset position to all added Exterior Corner meshes.
- Exterior Corner Meshes Offsets: Assign an independent offset position for each Exterior Corner mesh.

- Ceiling Meshes: Contains the models for the interior ceilings that make up the room.
- Ceilings Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Ceiling Offsets: Use the same offsets for all added Ceiling meshes.
- Ceilings Meshes Offsets: Add an offset position to all added Ceiling meshes.
- Ceiling Meshes Offsets: Assign an independent offset position for each Ceiling mesh.

- Door Blueprint: Contains the blueprint for the room door.
- Same Door Blueprint Offsets: Use the same offsets for all added Door Blueprints.
- Door Blueprints Offsets: Add an offset position to all added Door Blueprints.
- Door Blueprint Offsets: Assign an independent offset position for each Door Blueprint.

**5.4 Stair parameters**

- Floor Meshes: Contains the models for the floors that make up the stair.
- Floors Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Floor Offsets: Use the same offsets for all added Floor meshes.
- Floors Meshes Offsets: Add an offset position to all added Floor meshes.
- Floor Meshes Offsets: Assign an independent offset position for each Floor mesh.

- Interior Wall Meshes: Contains the models for the interior walls that form the staircase.
- Same Interior Wall Offsets: Use the same offsets for all added Interior Wall meshes.

- Interior Walls Meshes Offsets: Add an offset position to all added Interior Wall meshes.
- Interior Wall Meshes Offsets: Assign an independent offset position for each Interior Wall mesh.

- Interior Wall Up Meshes: Contains the models for the interior walls up that form the staircase.
- Same Interior Wall Up Offsets: Use the same offsets for all added Interior Wall Up meshes.
- Interior Walls Up Meshes Offsets: Add an offset position to all added Interior Wall Up meshes.
- Interior Wall Up Meshes Offsets: Assign an independent offset position for each Interior Wall Up mesh.

- Interior Corner Meshes: Contains the models for the interior corners that form the stair.
- Same Interior Corner Offsets: Use the same offsets for all added Interior Corner meshes.
- Interior Corners Meshes Offsets: Add an offset position to all added Interior Corner meshes.
- Interior Corner Meshes Offsets: Assign an independent offset position for each Interior Corner mesh.

- Exterior Wall Meshes: Contains the models for the exterior walls that form the staircase.
- Same Exterior Wall Offsets: Use the same offsets for all added Exterior Wall meshes.
- Exterior Walls Meshes Offsets: Add an offset position to all added Exterior Wall meshes.
- Exterior Wall Meshes Offsets: Assign an independent offset position for each Exterior Wall mesh.

- Exterior Wall Up Meshes: Contains the models for the exterior walls up that form the staircase.
- Same Exterior Up Wall Offsets: Use the same offsets for all added Exterior Wall Up meshes.
- Exterior Walls Up Meshes Offsets: Add an offset position to all added Exterior Wall Up meshes.
- Exterior Wall Up Meshes Offsets: Assign an independent offset position for each Exterior Wall Up mesh.

- Frequency Window Meshes: Probability of window spawn.
- Exterior Window Meshes: Contains the models for the exterior windows that form the staircase.
- Same Exterior Window Offsets: Use the same offsets for all added Exterior Window meshes.
- Exterior Windows Meshes Offsets: Add an offset position to all added Exterior Window meshes.
- Exterior Window Meshes Offsets: Assign an independent offset position for each Exterior Window mesh.

- Exterior Window Up Meshes: Contains the models for the exterior windows up that form the staircase.
- Same Exterior Window Up Offsets: Use the same offsets for all added Exterior Window Up meshes.
- Exterior Windows Meshes Up Offsets: Add an offset position to all added Exterior Window Up meshes.
- Exterior Window Meshes Up Offsets: Assign an independent offset position for each Exterior Window Up mesh.
- Draw Window With Roof: Allows drawing windows facing the roof.

- Exterior Corner Meshes: Contains the models for the exterior corners that form the stair.
- Same Exterior Corner Offsets: Use the same offsets for all added Exterior Corner meshes.
- Exterior Corners Meshes Offsets: Add an offset position to all added Exterior Corner meshes.
- Exterior Corner Meshes Offsets: Assign an independent offset position for each Exterior Corner mesh.

- Stair Type: Assigns the type of stair (StaticMesh/Blueprint).
- Stair Meshes: Contains the models of the stairs (StaticMesh only).
- Stair Blueprints: Contains the blueprints of the stairs (Blueprint only).
- Same Stairs Offsets: Use the same offsets for all added Stair meshes.
- Stairs Offsets: Add an offset position to all added Stair meshes.
- Stair Offsets: Assign an independent offset position for each Stair mesh.

- Ceiling Meshes: Contains the models for the interior ceilings that form the staircase.
- Ceilings Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Ceiling Offsets: Use the same offsets for all added Ceiling meshes.
- Ceilings Meshes Offsets: Add an offset position to all added Ceiling meshes.
- Ceiling Meshes Offsets: Assign an independent offset position for each Ceiling mesh.

**5.5 BaseRoofColumn parameters**

Config Module Foundation:

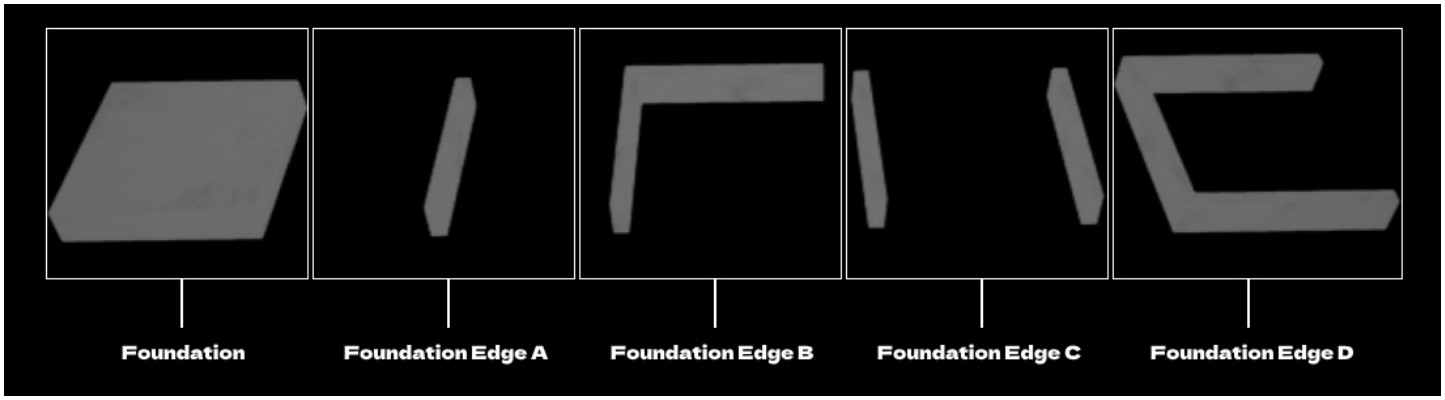
- Foundation Meshes: Contains the models for the foundations.
- Foundations Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Foundation Offsets: Use the same offsets for all added Ceiling meshes.
- Foundations Meshes Offsets: Add an offset position to all added Ceiling meshes.
- Foundation Meshes Offsets: Assign an independent offset position for each Ceiling mesh.

- Foundation Edge A Meshes: Contains the models for the type of Edge A for the foundations.
- Same Foundation Edge A Offsets: Use the same offsets for all added Foundation Edge A meshes.
- Foundations Edge A Meshes Offsets: Add an offset position to all added Foundation Edge A meshes.
- Foundation Edge A Meshes Offsets: Assign an independent offset position for each Foundation Edge A mesh.

- Foundation Edge B Meshes: Contains the models for the type of Edge B for the foundations.
- Same Foundation Edge B Offsets: Use the same offsets for all added Foundation Edge B meshes.
- Foundations Edge B Meshes Offsets: Add an offset position to all added Foundation Edge B meshes.
- Foundation Edge B Meshes Offsets: Assign an independent offset position for each Foundation Edge B mesh.

- Foundation Edge C Meshes: Contains the models for the type of Edge C for the foundations.
- Same Foundation Edge C Offsets: Use the same offsets for all added Foundation Edge C meshes.
- Foundations Edge C Meshes Offsets: Add an offset position to all added Foundation Edge C meshes.
- Foundation Edge C Meshes Offsets: Assign an independent offset position for each Foundation Edge C mesh.

- Foundation Edge D Meshes: Contains the models for the type of Edge D for the foundations.
- Same Foundation Edge D Offsets: Use the same offsets for all added Foundation Edge D meshes.
- Foundations Edge D Meshes Offsets: Add an offset position to all added Foundation Edge D meshes.
- Foundation Edge D Meshes Offsets: Assign an independent offset position for each Foundation Edge D mesh.



Example of models for the foundations

Config Module Base:

- Base Meshes: Contains the models for the foundations.
- Bases Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Base Offsets: Use the same offsets for all added Base meshes.
- Bases Meshes Offsets: Add an offset position to all added Base meshes.
- Base Meshes Offsets: Assign an independent offset position for each Base mesh.

- Base Edge A Meshes: Contains the models for the type A face for the base of the building.
- Same Base Edge A Offsets: Use the same offsets for all added Base Edge A meshes.
- Bases Edge A Meshes Offsets: Add an offset position to all added Base Edge A meshes.
- Base Edge A Meshes Offsets: Assign an independent offset position for each Base Edge A mesh.

- Base Edge B Meshes: Contains the models for the type B face for the base of the building.
- Same Base Edge B Offsets: Use the same offsets for all added Base Edge B meshes.
- Bases Edge B Meshes Offsets: Add an offset position to all added Base Edge B meshes.
- Base Edge B Meshes Offsets: Assign an independent offset position for each Base Edge B mesh.

- Base Edge C Meshes: Contains the models for the type C face for the base of the building.
- Same Base Edge C Offsets: Use the same offsets for all added Base Edge C meshes.
- Bases Edge C Meshes Offsets: Add an offset position to all added Base Edge C meshes.
- Base Edge C Meshes Offsets: Assign an independent offset position for each Base Edge C mesh.

- Base Edge D Meshes: Contains the models for the type D face for the base of the building.
- Same Base Edge D Offsets: Use the same offsets for all added Base Edge D meshes.
- Bases Edge D Meshes Offsets: Add an offset position to all added Base Edge D meshes.
- Base Edge D Meshes Offsets: Assign an independent offset position for each Base Edge D mesh.

Config Module Roof:

- Roof Type: Allows you to select between a simple and advanced roof.
  A simple roof is generated differently than an advanced roof. This type of roof adds a Roof Mesh randomly from the additions in Roof Meshes. All the cells that its upper part faces the outside, this model is added. Faces are added depending on the conditions needed to add them.
  An advanced roof needs more models to be able to be drawn, since each condition needs a specific roof model to be able to be drawn well.

- Roof Floor Meshes: Contains the models for the roof floor.
- Roof Floors Random Orientation: Rotates the mesh in the Yaw axis depending on the orientation. The orientation is generated randomly.
- Same Roof Floor Offsets: Use the same offsets for all added Roof Floor meshes.
- Roofs Floor Meshes Offsets: Add an offset position to all added Roof Floor meshes.
- Roof Floor Meshes Offsets: Assign an independent offset position for each Roof Floor mesh.

- Roof Edge A Meshes: Contains the models for the type A face for the roof of the building.
- Same Roof Edge A Offsets: Use the same offsets for all added Roof Edge A meshes.
- Roofs Edge A Meshes Offsets: Add an offset position to all added Roof Edge A meshes.
- Roof Edge A Meshes Offsets: Assign an independent offset position for each Roof Edge A mesh.

- Roof Edge B Meshes: Contains the models for the type B face for the roof of the building.
- Same Roof Edge B Offsets: Use the same offsets for all added Roof Edge B meshes.
- Roofs Edge B Meshes Offsets: Add an offset position to all added Roof Edge B meshes.
- Roof Edge B Meshes Offsets: Assign an independent offset position for each Roof Edge B mesh.

- Roof Edge C Meshes: Contains the models for the type C face for the roof of the building.
- Same Roof Edge C Offsets: Use the same offsets for all added Roof Edge C meshes.
- Roofs Edge C Meshes Offsets: Add an offset position to all added Roof Edge C meshes.
- Roof Edge C Meshes Offsets: Assign an independent offset position for each Roof Edge C mesh.

- Roof Edge D Meshes: Contains the models for the type D face for the roof of the building.
- Same Roof Edge D Offsets: Use the same offsets for all added Roof Edge D meshes.
- Roofs Edge D Meshes Offsets: Add an offset position to all added Roof Edge D meshes.
- Roof Edge D Meshes Offsets: Assign an independent offset position for each Roof Edge D mesh.

- Roof Flat Meshes: Contains flat roof models.
- Same Roof Flat Offsets: Use the same offsets for all added Roof Flat meshes.
- Roofs Flat Meshes Offsets: Add an offset position to all added Roof Flat meshes.
- Roof Flat Meshes Offsets: Assign an independent offset position for each Roof Flat mesh.

- Roof A Meshes: Contains the models of type A for the roof.
- Same Roof A Offsets: Use the same offsets for all added Roof A meshes.
- Roofs A Meshes Offsets: Add an offset position to all added Roof A meshes.
- Roof A Meshes Offsets: Assign an independent offset position for each Roof A mesh.

- Roof B Meshes: Contains the models of type B for the roof.
- Same Roof B Offsets: Use the same offsets for all added Roof B meshes.
- Roofs B Meshes Offsets: Add an offset position to all added Roof B meshes.
- Roof B Meshes Offsets: Assign an independent offset position for each Roof B mesh.

- Roof C Meshes: Contains the models of type C for the roof.
- Same Roof C Offsets: Use the same offsets for all added Roof C meshes.
- Roofs C Meshes Offsets: Add an offset position to all added Roof C meshes.
- Roof C Meshes Offsets: Assign an independent offset position for each Roof C mesh.

- Roof D1 Meshes: Contains the models of type D1 for the roof.
- Same Roof D1 Offsets: Use the same offsets for all added Roof D1 meshes.
- Roofs D1 Meshes Offsets: Add an offset position to all added Roof D1 meshes.
- Roof D1 Meshes Offsets: Assign an independent offset position for each Roof D1 mesh.

- Roof D2 Meshes: Contains the models of type D2 for the roof.
- Same Roof D2 Offsets: Use the same offsets for all added Roof D2 meshes.
- Roofs D2 Meshes Offsets: Add an offset position to all added Roof D2 meshes.
- Roof D2 Meshes Offsets: Assign an independent offset position for each Roof D2 mesh.

- Roof E Meshes: Contains the models of type E for the roof.
- Same Roof E Offsets: Use the same offsets for all added Roof E meshes.
- Roofs E Meshes Offsets: Add an offset position to all added Roof E meshes.
- Roof E Meshes Offsets: Assign an independent offset position for each Roof E mesh.

- Roof F Meshes: Contains the models of type F for the roof.
- Same Roof F Offsets: Use the same offsets for all added Roof F meshes.
- Roofs F Meshes Offsets: Add an offset position to all added Roof F meshes.
- Roof F Meshes Offsets: Assign an independent offset position for each Roof F mesh.

- Roof G Meshes: Contains the models of type G for the roof.
- Same Roof G Offsets: Use the same offsets for all added Roof G meshes.
- Roofs G Meshes Offsets: Add an offset position to all added Roof G meshes.
- Roof G Meshes Offsets: Assign an independent offset position for each Roof G mesh.

- Roof H Meshes: Contains the models of type H for the roof.
- Same Roof H Offsets: Use the same offsets for all added Roof H meshes.
- Roofs H Meshes Offsets: Add an offset position to all added Roof H meshes.
- Roof H Meshes Offsets: Assign an independent offset position for each Roof H mesh.

- Roof I Meshes: Contains the models of type I for the roof.
- Same Roof I Offsets: Use the same offsets for all added Roof I meshes.
- Roofs I Meshes Offsets: Add an offset position to all added Roof I meshes.
- Roof I Meshes Offsets: Assign an independent offset position for each Roof I mesh.

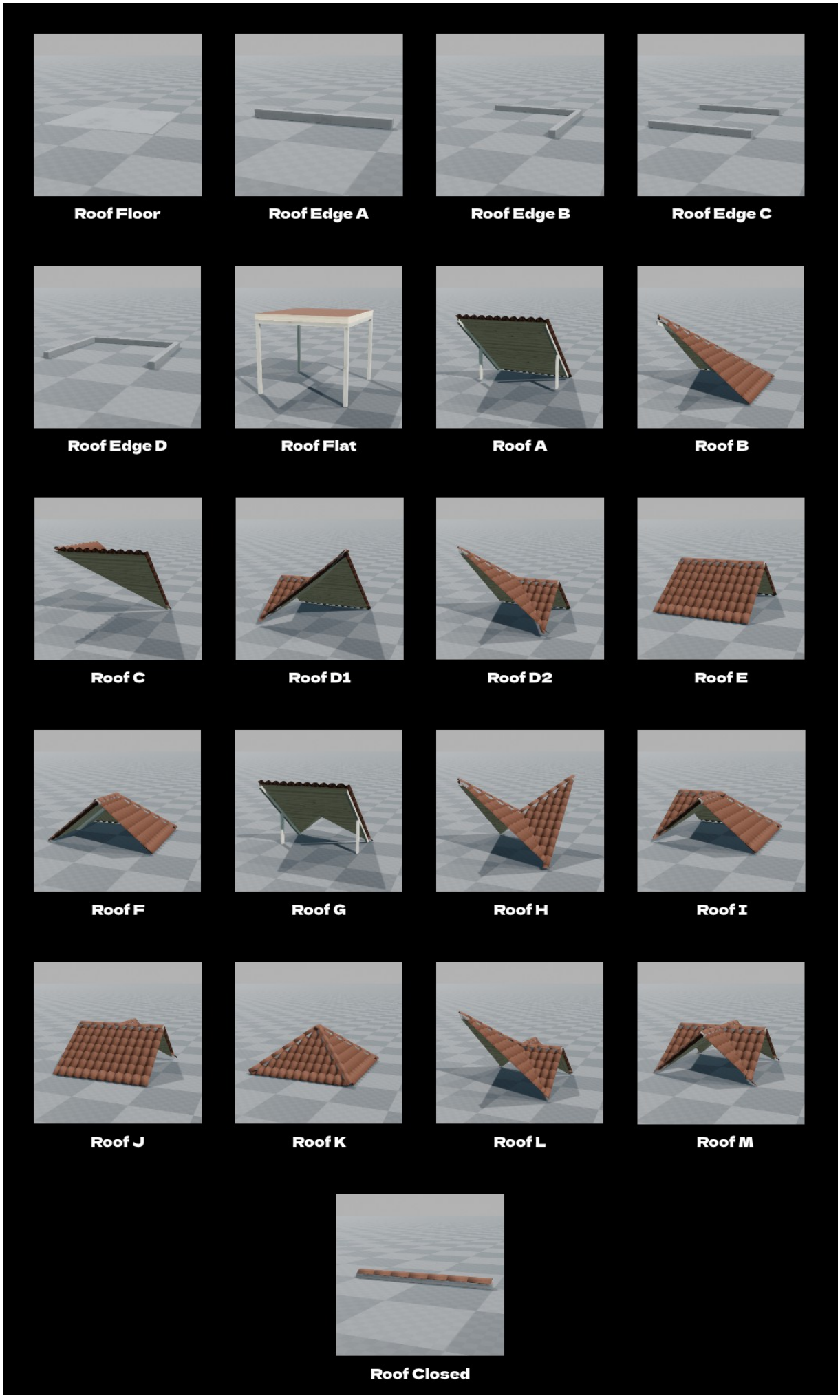
- Roof J Meshes: Contains the models of type J for the roof.
- Same Roof J Offsets: Use the same offsets for all added Roof J meshes.
- Roofs J Meshes Offsets: Add an offset position to all added Roof J meshes.
- Roof J Meshes Offsets: Assign an independent offset position for each Roof J mesh.

- Roof K Meshes: Contains the models of type K for the roof.
- Same Roof K Offsets: Use the same offsets for all added Roof K meshes.
- Roofs K Meshes Offsets: Add an offset position to all added Roof K meshes.
- Roof K Meshes Offsets: Assign an independent offset position for each Roof K mesh.

- Roof L Meshes: Contains the models of type L for the roof.
- Same Roof L Offsets: Use the same offsets for all added Roof L meshes.
- Roofs L Meshes Offsets: Add an offset position to all added Roof L meshes.
- Roof L Meshes Offsets: Assign an independent offset position for each Roof L mesh.

- Roof M Meshes: Contains the models of type M for the roof.
- Same Roof M Offsets: Use the same offsets for all added Roof M meshes.
- Roofs M Meshes Offsets: Add an offset position to all added Roof M meshes.
- Roof M Meshes Offsets: Assign an independent offset position for each Roof M mesh.

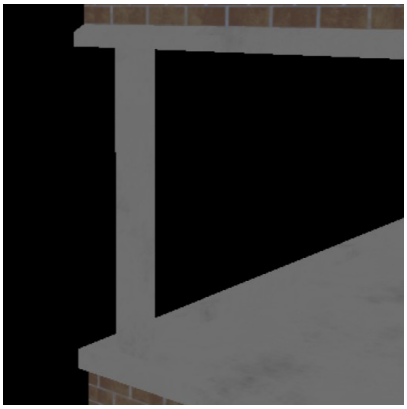
- Roof Closed Meshes: It contains the models of the closing type for the roof. This type of model may not be necessary as it serves to cover the joint between some roofs. The need for it depends on the modeling of the roof.
- Same Roof Closed Offsets: Use the same offsets for all added Roof Closed meshes.
- Roofs Closed Meshes Offsets: Add an offset position to all added Roof Closed meshes.
- Roof Closed Meshes Offsets: Assign an independent offset position for each Roof Closed mesh.



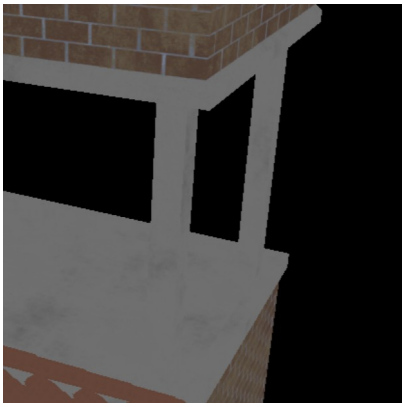

Example of advanced roofs

Config Module Column:

- Column A Meshes: Contains the models for the columns that are added to the corners.
- Column B Meshes: Contains the models for the columns that are added in the closures.



Example column type A



Example column type B

**5.6 Module interiors**

Each module has a section to configure its interior (Config Module Interior), where you can set the interior type, appearance type, quantity, and more.

Interiors are added to a cell, and each cell has 9 free spaces to add interiors: 1 general (cell), 4 for the walls (1 for each orientation), and 4 for the windows (1 for each orientation).

**5.7 Interior parameters**

- Interior Type: Specifies the type of interior to generate (Static mesh, blueprint particle system, decal).
- Interior Meshes: Contains the models to be added to the interior; one will be randomly selected. This parameter is only active if the Static Mesh interior type is selected.
- Interior Blueprints: Contains blueprints to be added to the interior; one will be randomly selected. This parameter is only active if the Blueprint interior type is selected.
- Interior Particles System: Contains particle systems to be added to the interior; one will be randomly selected. This parameter is only active if the Particle System interior type is selected.
- Interior Decals: Contains decals to be added to the interior; one will be randomly selected. This parameter is only active if the Decal interior type is selected.
- Spawn Method: Determines where the interior will appear.
  - **Cell:** The most basic method to add an interior; it doesn't consider the orientation of the walls and windows. This method can be used to add interiors to the floor and ceiling.
  - **Walls:** Adds the interior only to the walls, considering their orientation.
  - **Windows:** Adds the interior only to the windows, considering their orientation.
  - **Wall and Windows:** Adds the interior to both the walls and windows, considering their orientations.
- Spawn Type: Specifies when the interior should be added.
  - **Only Free:** Adds the interior when it finds an empty space, ignoring interiors already in the cell where it's added.
  - **Only Used:** Adds the interior when it finds a space already used by another interior, ignoring interiors in the cell where it's added.
  - **FreeAndUsed:** Adds the interior regardless of whether the space is free or used, ignoring the cell where it's added.
  - **AnyUsedInCell:** Adds the interior when it finds an empty space and the cell already has another interior added.
  - **OnlyFreeCell:** Adds the interior when it finds a cell with all its spaces free.
  - **OccupyAllCell:** Adds the interior when it finds an empty space in the cell; it occupies any remaining free spaces.
- Spawn in Door: Allows adding the interior if the cell contains a door. If disabled, it won't add the interior if a door is present in the cell.
- Occupy Interior: Determines if the added interior occupies a free space in the cell.
- Spawn Ammount: Specifies the quantity of the interior to add to the module.
  - **All:** Adds the interior until its conditions prevent adding more. The conditions depend on Spawn Method and Spawn Type.
  - **Random Spawned:** Selects a random number between a minimum and maximum, which determines the quantity of interior to generate.
  - **MaxInModule:** Sets a limit on the quantity of interior to be generated in the module.
  - **MaxInBuilding:** Sets a limit on the quantity of interior to be generated in the building.

- Spawn Restriction: Defines the conditions for adding the interior.
  - **No Restriction:** No restrictions are placed on adding the interior.
  - **SameModuleOneOrientation:** Checks the neighboring cell's orientation; if it belongs to the same module, the interior is added.
  - **SameModuleOneInverseOrientation:** Checks the neighboring cell's inverse orientation; if it belongs to the same module, the interior is added.
  - **SameModuleTwoOrientations:** Adds the interior if the cells on both sides of the cell where the interior is added belong to the same module.
  - **SameModuleArround:** Adds the interior if the surrounding cells belong to the same module.
  - **DiferentModuleTwoOrientations:** Adds the interior if the cells on both sides of the cell where the interior is added do not belong to the same module.
  - **OnePerCell:** Adds only one interior per cell in the module.
  - **FrontInDoor:** Adds the interior if there is a door in front of a wall or window. It's added only if the door belongs to the same cell as the added interior.
- Offset Position: Adds a position offset to the interior.
- Offset Rotation: Adds a rotation offset to the interior.
- Random Yaw Rotation: Adds a random rotation on the Yaw axis.
- Scale: Sets the size of the interior.

## 6 - Building functions (Blueprint)

- GenerateBuilding(): Generates the building with the values assigned to its variables.
- GenerateRandomSeed(): Generates the building with a seed randomly keeping the value of its variables.
- GenerateRandomInteriorSeed(): Generate the interior building with a seed randomly.
- GenerateRandomBuilding(): Assign random values to your variables and generate a building.
- ClearBuilding(): Remove all children from the building. It only affects the visual part.
- CreateFloorMapTexture(int cellSize, int floor, int marginX, int marginY):
  Creates a texture that contains the map of a specific floor of the building.
  - **cellSize:** Sets the size of each building cell in the map texture. By default it is 8.
  - **floor:** Select the floor of the building to generate the map texture. By default it is 0.
  - **marginX:** Assigns a margin for the left and right side of the map texture. By default it is 4.
  - **marginY:** Assigns a margin for the top and bottom of the map texture. By default it is 4.
- Set_FloorsMapColor(FColor color): Assigns a color to the building map texture for the floors.
- Set_WallsMapColor(FColor color): Assigns a color to the building map texture for the walls.
- Set_EntryDoorMapColor(FColor color): Assigns a color to the building map texture for the entry door.
- Set_DoorsMapColor(FColor color): Assigns a color to the building map texture for the doors.
- Set_StairsMapColor(FColor color): Assigns a color to the building map texture for the stairs.
- Set_CellsLockedMapColor(FColor color): Assigns a color to the building map texture for the cells locked.
- Set_BackgroundMapColor(FColor color): Assigns a color to the building map texture for the background.

## 7 - Utils functions (C++)

- GetExistAroundCells(TArray<FS_CellBuilding>* cellsInBuilding, int posX, int posY, int posZ):
  Returns an array of booleans with a value of true or false depending on whether or not the cell around the specified position exists.

- GetExistAroundCells(TArray<FS_CellBuilding>* cellsInBuilding, FS_CellBuilding* cell):
  Returns an array of booleans with a value of true or false depending on whether or not the cell around the specified cell exists.

- GetAroundCells(TArray<FS_CellBuilding>* cellsInBuilding, int posX, int posY, int posZ):
  Returns an array of cells around the specified position.

- GetAroundCells(TArray<FS_CellBuilding>* cellsInBuilding, FS_CellBuilding* cell):
  Returns an array of cells around the specified cell position.

- GetSympleAroundCells(TArray<FS_CellBuilding>* cellsInBuilding, int posX, int posY, int posZ):
  Returns an array of cells that are north, east, south, west of the specified position.

- GetSympleAroundCells(TArray<FS_CellBuilding>* cellsInBuilding, FS_CellBuilding* cell):
  Returns an array of cells that are north, east, south west of the specified cell.

- GetCellInBuilding(TArray<FS_CellBuilding>* cellsInBuilding, int posX, int posY, int posZ):
  Returns the cell at the specified position.

- GetCellInBuilding(TArray<FS_CellBuilding>* cellsInBuilding, FS_CellBuilding* cell):
  Returns the cell starting at the specified cell position.

- GetCellInBuilding(TArray<FS_CellBuilding>* cellsInBuilding, FS_CellBuilding* cell, int offsetX, int offsetY, int offsetZ):
  Returns the cell from the specified cell position by adding the assigned offsets.

- GetCellsInModule(TArray<FS_CellBuilding>* cellsInBuilding, int idModule):
  Returns an array with the cells contained in the module with the indicated id.

- SetCorridorConectedModule(TArray<FS_CellBuilding>* cellsInBuilding, int idModule, bool corridorConnected):
  Assigns from the module id if it is connected to the corridor.

- SetCorridorConectedModule(TArray<FS_CellBuilding*> cellsInModule, bool corridorConnected):
  Sets whether or not it is connected to all the cells that belong to the module.

- AddCellRectInBuilding(FVector buildingPosition, int cellSizeX, int cellSizeY, int cellSizeZ, TArray<FS_CellBuilding>* cellsInBuilding, int orientation, int xStartPos, int yStartPos, int widthRect, int longRect, int floor):
  Adds a set of rectangle-shaped cells with the specified parameters.

- AddCellBuilding(TArray<FS_CellBuilding>* cellsInBuilding, int cellSizeX, int cellSizeY, int cellSizeZ, int orientation, FVector buildingPosition, int xPos, int yPos, int floor):
  Add a cell with the specified parameters.

- AddCellBuilding(TArray<FS_CellBuilding>* cellsInBuilding, int idModule, int cellSizeX, int cellSizeY, int cellSizeZ, int cellOrientation, FVector buildingPosition, int xPos, int yPos, int floor, bool entry, int entryOrientation, E_CellType cellType, E_ModuleType moduleType, int stairOrientation, bool drawStair, bool corridorConnected):
  Add a cell with the specified parameters.

- AddCorridor(int idModule, TArray<FS_CellBuilding>* cellsInBuilding, TArray<FS_CellBuilding*>& cellsCorridors, int xStartPos, int yStartPos, int floor, int orientation, int longCorridor, bool entry = false, int entryOrientation):
  Adds the cells to form a corridor with the specified parameters.

- GetCellsInFloor(TArray<FS_CellBuilding>* cellsInBuilding, int floor):
  Returns an array with the cells found on the specified floor.

- GetCellsInCorridors(TArray<FS_CellBuilding>* cellsInBuilding, int floor):
  Returns an array with the cells corresponding to the corridors of the specified floor.

- GetCellStair(TArray<FS_CellBuilding>* cellsInBuilding, int floor):
  Returns the cell that corresponds to the stair on the specified floor.

- GetNumberCellsFromOrientation(TArray<FS_CellBuilding>* cellsInBuilding, int maxRectCellsX, int maxRectCellsY, int xPos, int yPos, int zPos, int orientation):
  Returns the number of cells from a specified position to a specified orientation.

- DupicateCellsFloor(TArray<FS_CellBuilding>* cellsInBuilding, int floor, int floorDestination):
  Duplicates cells on a floor to assign them to another specified floor.

- GetCellBorderFromOrientation(FRandomStream streamSeed, TArray<FS_CellBuilding>* cellsInBuilding, int orientation, int floor):
  Returns the cell that is on the edge of the building based on orientation (0 north, 1 east, 2 south, 3 west).
- CheckAnyUpCell(TArray<FS_CellBuilding>* cellsInBuilding, FS_CellBuilding* cell, int totalFloors):
  Returns true or false depending on whether the cell that is on top of the specified cell exists.

- AddBlueprint(FRandomStream* streamSeed, UWorld* world, AActor* actor, FString name, FVector position, int orientation, TSubclassOf<class AActor>* blueprint, TArray<AActor*>* containerBlueprints):
  Add a blueprint with the specified parameters.

- AddBlueprint(FRandomStream* streamSeed, UWorld* world, AActor* actor, FString name, FVector position, int orientation, TArray<TSubclassOf<class AActor>>* blueprints, TArray<AActor*>* containerBlueprints):
  Add a blueprint with the specified parameters.

- AddMesh(FRandomStream* streamSeed, USceneComponent* rootComponent, AActor* actor, FString name, FVector position, TArray<UStaticMesh*>* meshes, TArray<UStaticMeshComponent*>* containerMeshes):
  Add a static mesh with the specified parameters.

- AddMesh(FRandomStream* streamSeed, USceneComponent* rootComponent, AActor* actor, FString name, FVector position, int orientation, TArray<UStaticMesh*>* meshes, TArray<UStaticMeshComponent*>* containerMeshes):
  Add a static mesh with the specified parameters.

- PrintTextValue(FString text, int Value):
  Prints a text on the screen and the value of a variable.

- DestroyAllChildrens(AActor* Actor):
  Removes all children of the specified actor.

- GetNumCellsInX(TArray<FS_CellBuilding>* cellsInBuilding, int floor):
  Returns the maximum number of cells that are on the X axis of the building on a specific floor.

- GetNumCellsInY(TArray<FS_CellBuilding>* cellsInBuilding, int floor):
  Returns the maximum number of cells that are on the Y axis of the building on a specific floor.

- CreateCellMapTexture(FS_CellBuilding* cellInFloor, int cellSize, int fixPosX, int fixPosY, int offsetX, int offsetY, FColor floorsMapColor, FColor wallsMapColor, FColor entryDoorMapColor, FColor doorsMapColor, FColor stairsMapColor, FColor cellsLockedMapColor):
  Creates the cell data to use in creating the map texture.

## 8 - Contanct

Email: contact@broskygames.com
Web: https://www.broskygames.com