# Modular Asset Build Assistant

## Contents:



## Introduction:

Thank you for your interest in The Modular Asset Build Assistant! I genuinely hope you can utilize this set of blueprints to help speed up your level design and building to it's maximum potential. I developed these blueprints after using a handful of modular asset packs and realizing how much time I wasted just tediously trying to line up meshes, even with the help of snapping and the grid system.

At first glance, once you've acclimated yourself to the most efficient way of utilizing these blueprints, you'll see that the amount of time you save may only be seconds for each individual mesh, however the true impact on total design time in the long term can be hours if not days of repetitive mesh snapping depending on the size of your levels.

As a bonus, I've also included a blueprint called BP_PropRandomizer, which is a useful tool to use when you want to distribute random props around the level with varying degrees of randomness. This is useful for specific use cases that aren't necessarily handled neatly with the built in foliage tool in Unreal Engine.

There are quite a few moving parts behind the scenes and as such, some things may be a little difficult to wrap your head around. I strongly advise going into the demo level and experiment with each example blueprint to better understand how each setting affects the tiling. If you have any issues or questions, feel free to reach out to me at:


gamedev@ryanmcfate.com


I'm more than happy to help where I can.

# Common Variables

## Attach to Builder (obj ref):

Using the eye dropper, you can snap the currently selected blueprint to another builder blueprint at it's connect point.

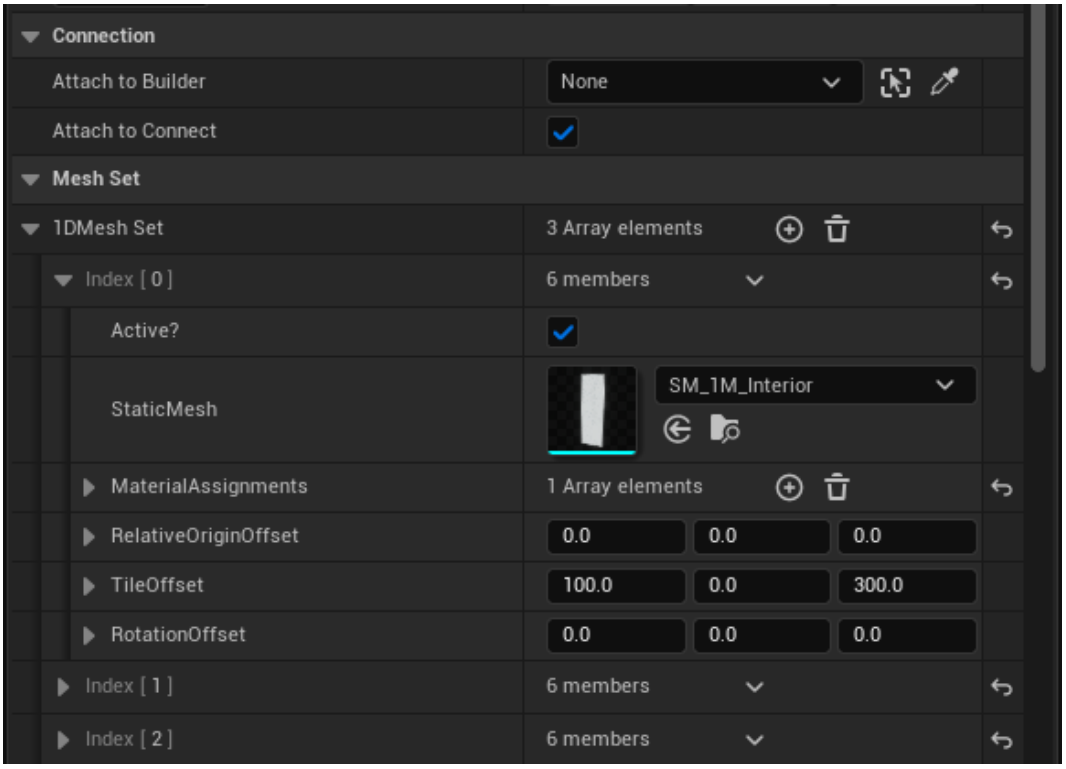
## Attach to Connect (bool):

When marked true, attaching to another builder will attach at the arrow connect point. For most builders, this is the origin point, however, 1D builders can attach to the end of the last tiled mesh to allow for easier wall tiling.

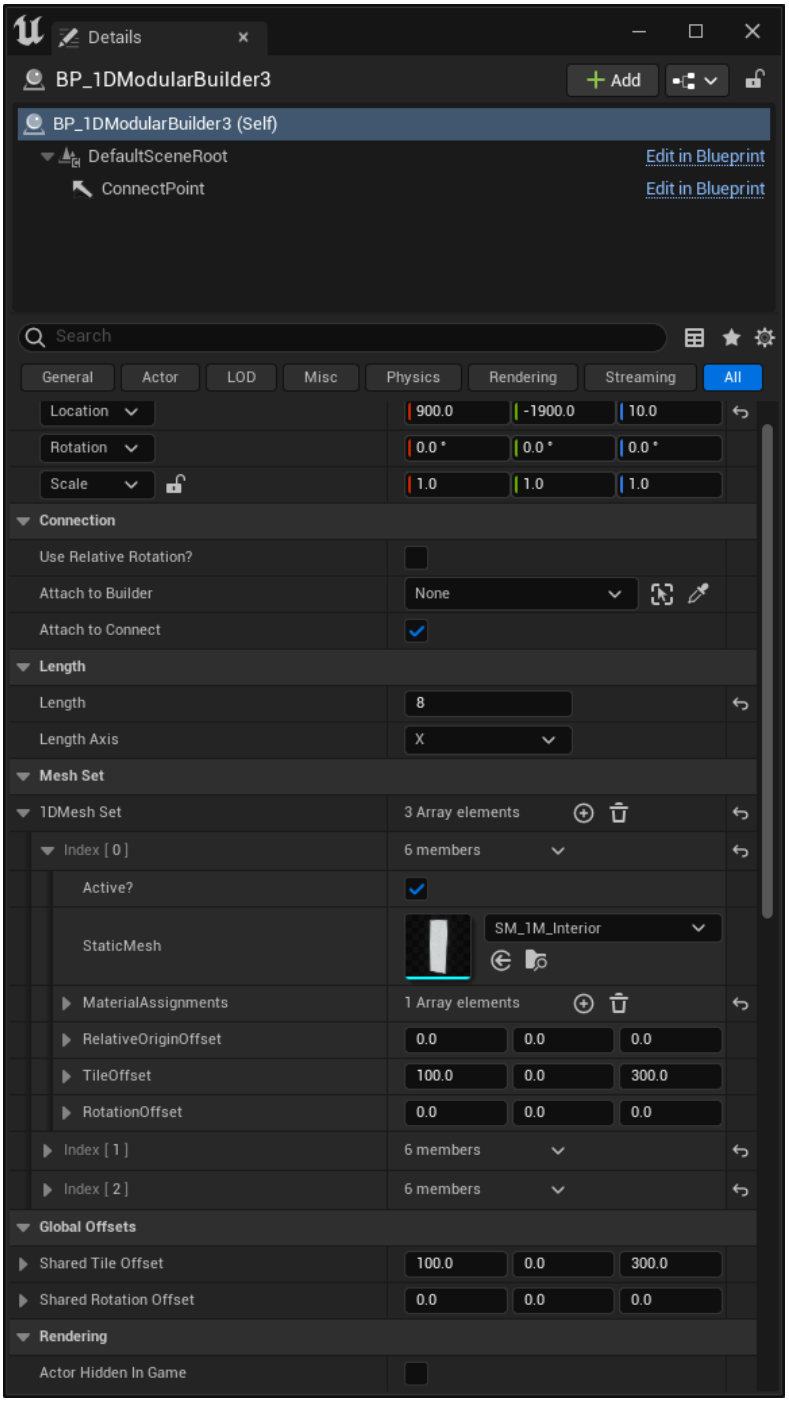
## [1D][2D][3D][Complex] Mesh Sets (user structs):

Mesh sets are custom structure classes that contain all the information needed to tile modular meshes.

- Active (bool): Toggles whether the mesh is rendered or not.
- StaticMesh (mesh ref): The mesh to be tiled.
- MaterialAssignments (array of user struct): Material Instance and it's Material Slot Name if the mesh has the option of different materials.
- Relative Origin Offset (vector): An offset applied to the mesh. Useful when paired with other meshes whose origin is not the same but needs to be lined up. Usually stays 0,0,0.
- Tile Offset (vector): The distance in units that the mesh tiles in each direction. This vector represents the size of the mesh in X, Y, and Z. Depending on the direction of the mesh, some of these values may need to be negative to tile correctly. Blueprint assumes tiling in the positive X direction.
- Rotation Offset (rotator): A rotational offset for the mesh.



# BP_1DModularBuilder




## Use Relative Rotation? (bool):

If marked true, tile and rotation offset are based on the previous mesh's location and rotation, allowing you to make more complex patterns by leveraging relative stacking of tiled meshes. If marked false, the location is calculated using mesh count and is constrained to the axis in a linear fashion.

Same Settings, Absolute vs. Relative:


## Length (int):

Number of meshes to tile.

## Length Axis (User Enum):

Which axis to tile against.

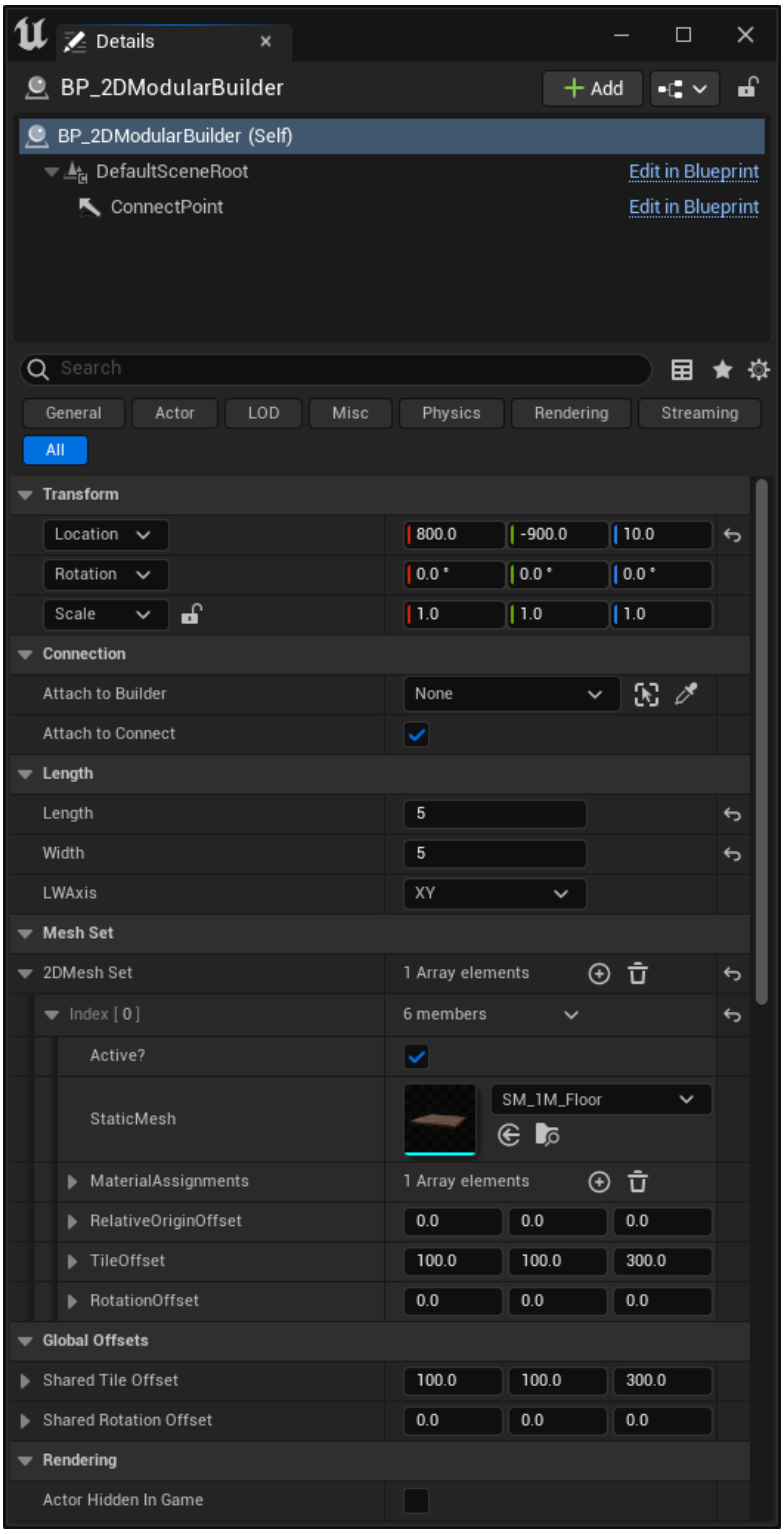
## Shared Tile Offset (vector):

Overrides the TileOffset on every mesh set. This is required for consistent tiling of all meshes in the set, which means all meshes in the set should be the same size. If for some reason you need to have each mesh be a different size, consider using the ComplexBuilder.

## Shared Rotation Offset (rotator):

Overrides the RotationOffset on every mesh set. This is required for consistent tiling of all meshes in the set, which means all meshes in the set should be the same size. If for some reason you need to have each mesh be a different size, consider using the ComplexBuilder.

# BP_2DModularBuilder




## Length (int):

Number of meshes to tile on first axis.

## Width (int):

Number of meshes to tile on the second axis

## LW Axis (User Enum):

Which two axes to tile against.

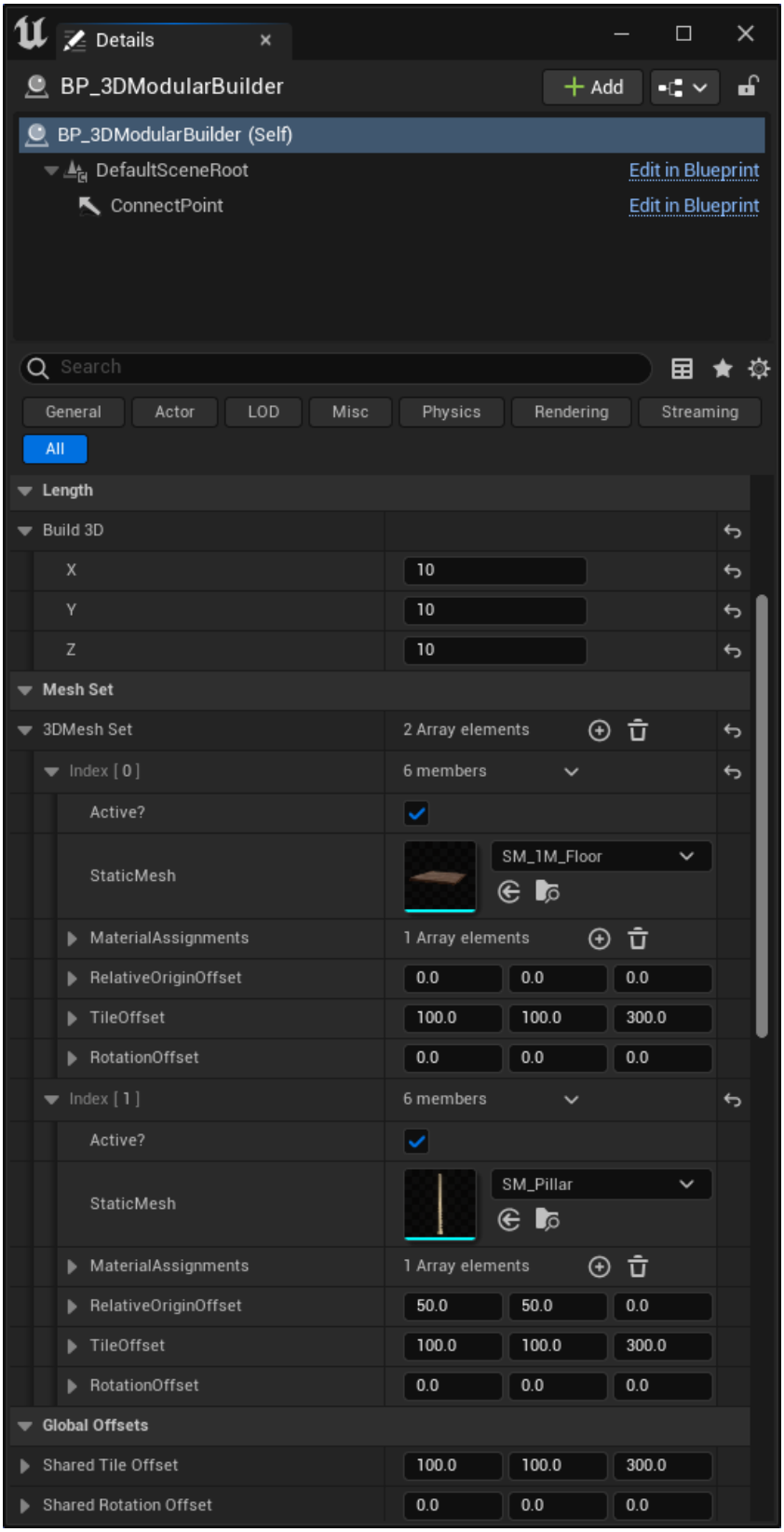
## Shared Tile Offset (vector):

Overrides the TileOffset on every mesh set. This is required for consistent tiling of all meshes in the set, which means all meshes in the set should be the same size. If for some reason you need to have each mesh be a different size, consider using the ComplexBuilder.

## Shared Rotation Offset (rotator):

Overrides the RotationOffset on every mesh set. This is required for consistent tiling of all meshes in the set, which means all meshes in the set should be the same size. If for some reason you need to have each mesh be a different size, consider using the ComplexBuilder.

# BP_3DModularBuilder




## Build 3D (User Struct):

Number of meshes in each axis.

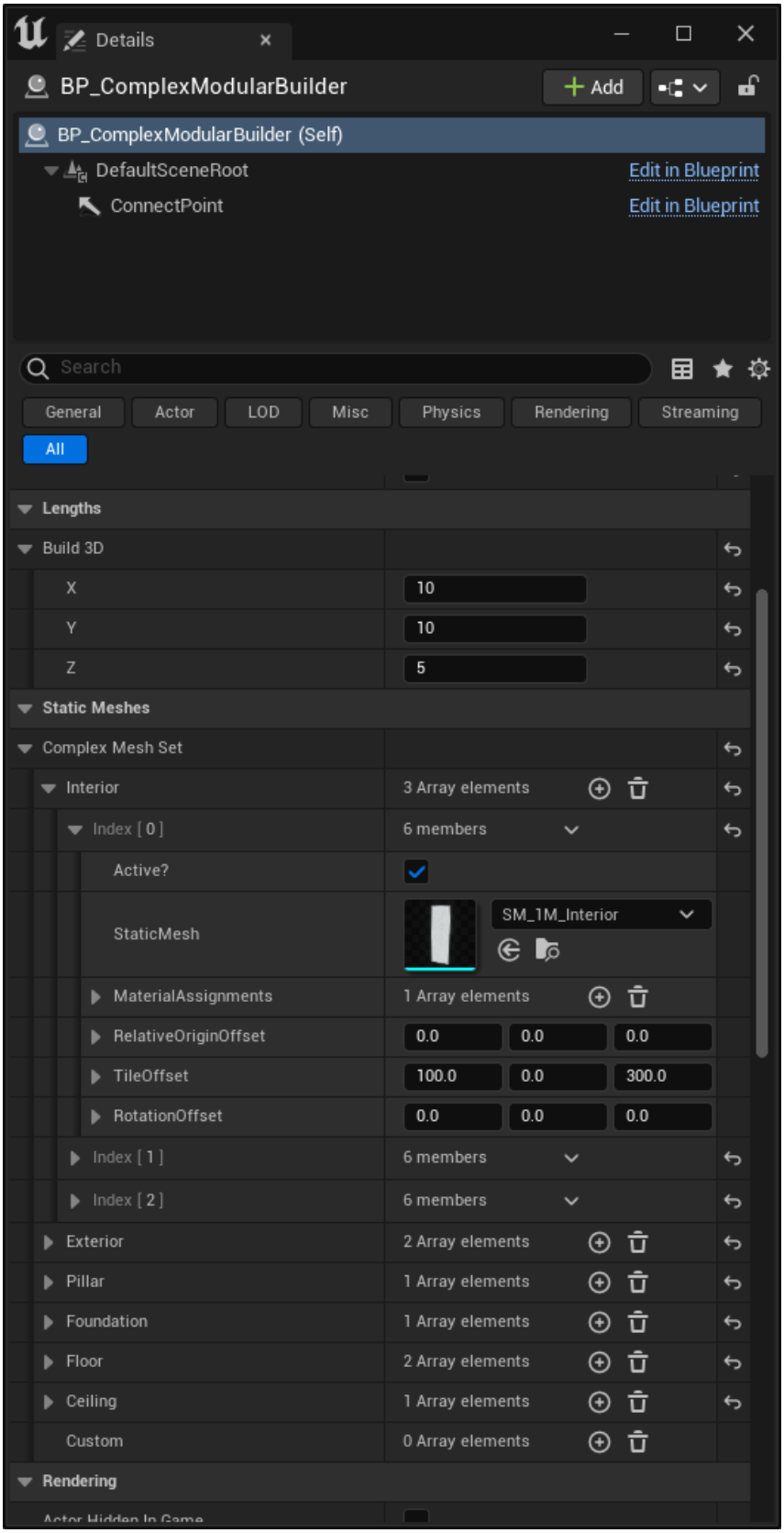
## Shared Tile Offset (vector):

Overrides the TileOffset on every mesh set. This is required for consistent tiling of all meshes in the set, which means all meshes in the set should be the same size. If for some reason you need to have each mesh be a different size, consider using the ComplexBuilder.

## Shared Rotation Offset (rotator):

Overrides the RotationOffset on every mesh set. This is required for consistent tiling of all meshes in the set, which means all meshes in the set should be the same size. If for some reason you need to have each mesh be a different size, consider using the ComplexBuilder.

# BP_ComplexBuilder



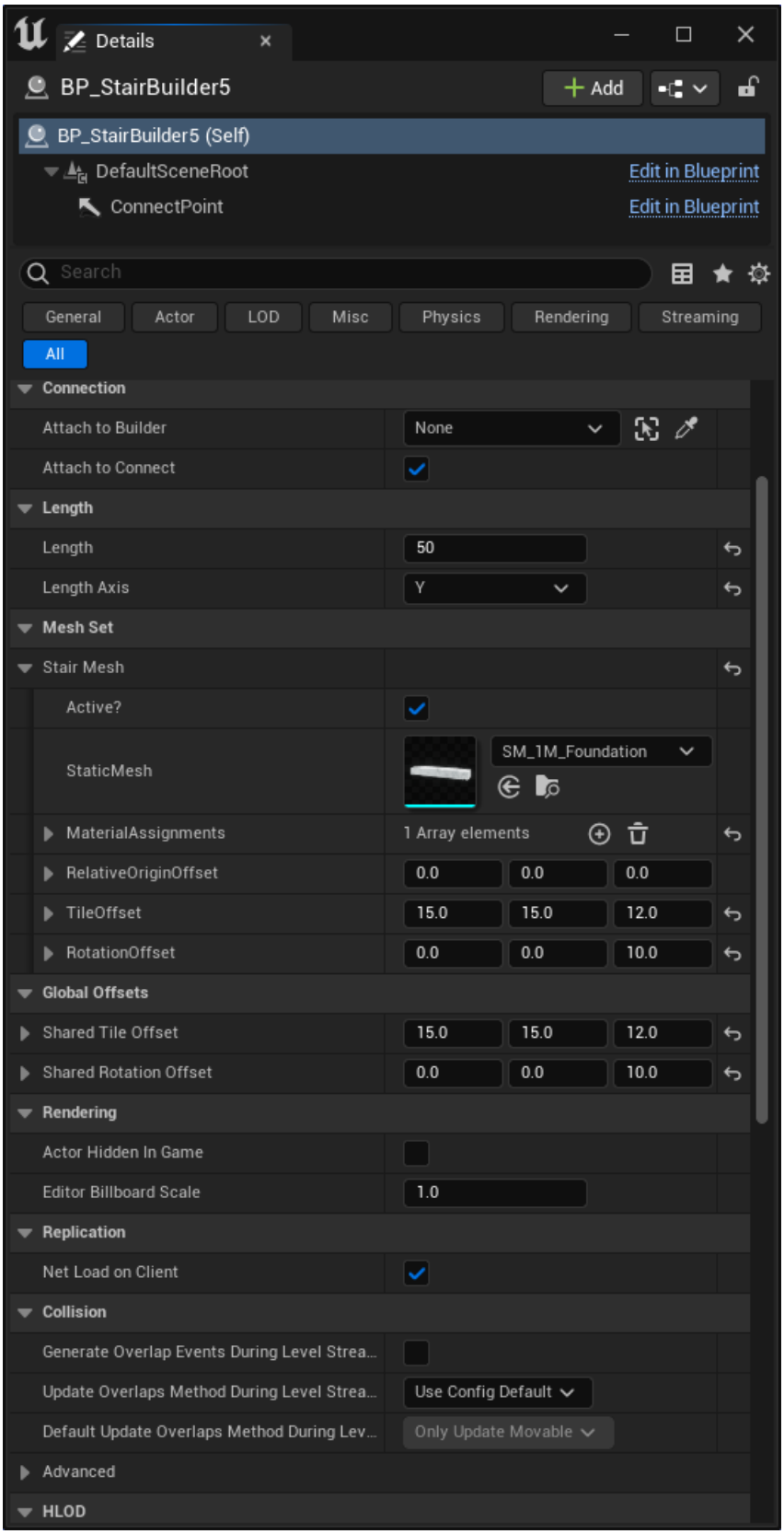

## Build 3D (User Struct):

Number of meshes in each axis.

## Complex Mesh Set (User Struct):

The complex mesh set is a custom array of mesh sets with named sections. You may add multiple types of meshes with different materials to each array (for example, alternating colored interior walls under the “Interior” array). This will alternate the different meshes in a pattern. For 2D types like floors, having 2 meshes will create a checkerboard pattern (assuming the floor tiles are the same size), having more will create more complex patterns. This is the only blueprint type where tile offsets and rotations are not constrained to one global value for each, which allows for more customized tiling, however meshes that aren’t the same size will cause odd tiling behavior and is usually not recommended.

# BP_StairBuilder




## Length (int):

Number of meshes in the tile set.

## Length Axis (user enum):

Which axis to tile.

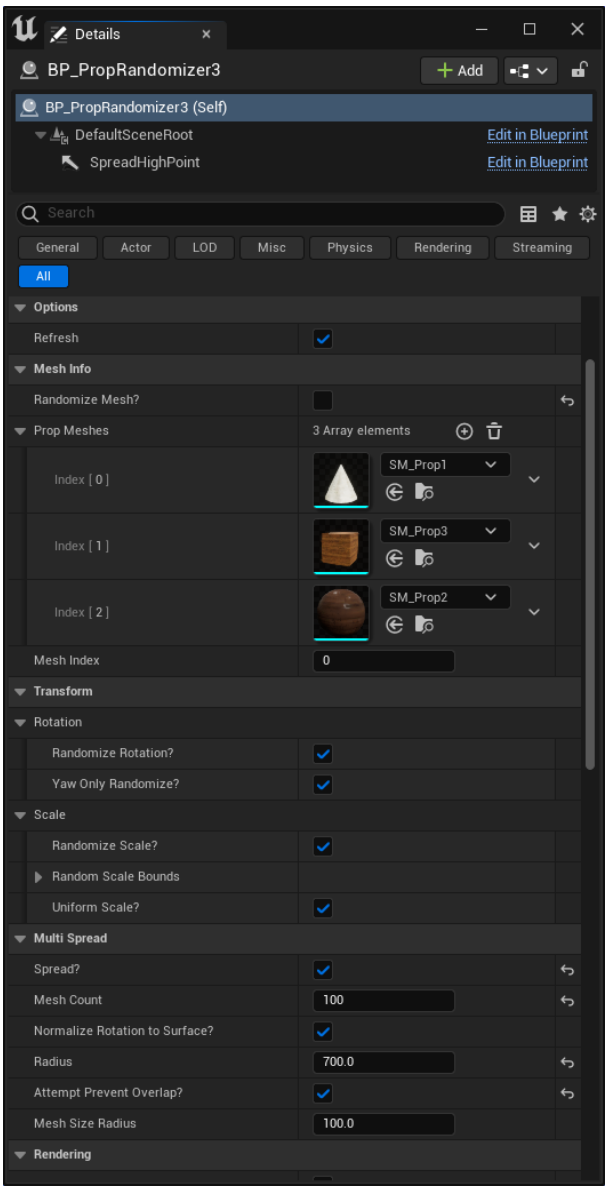
## Shared Tile Offset (vector):

Overrides the TileOffset on the mesh. The Z value should be the height of the mesh to create proper stepping height.

## Shared Rotation Offset (rotator):

Overrides the RotationOffset on the mesh. The Yaw value being set to a small angle creates spiral staircases. Changing roll and pitch will allow you to create non-conventional patterns that don't really function as real stairs and is useful for creating sculpture-like structures.

# BP_PropRandomizer




## Refresh (bool):

Refreshes the randomization pattern to get a different result.

## Randomize Mesh? (bool):

If marked true, will randomize the mesh(es) used from the Prop Meshes array. If marked false, use the mesh at array index “Mesh Index” of the Prop Meshes array.

## Mesh Index (int):

If RandomizeMesh? is marked false, this is the index of the mesh in the Prop Meshes array to use. Doesn’t do anything if RandomizeMesh? is marked true.

## Randomize Rotation? (bool):

If marked true, randomize the rotation of the mesh on all axes.

## Yaw Only Randomize? (bool):

If marked true, only randomized the rotation on Yaw.

## Randomize Scale? (bool):

If marked true, randomize the scale of the mesh using the Random Scale Bounds minimum and maximum values.

## Randomize Scale Bounds (user struct):

A minimum and maximum value to keep within when randomizing the scale of meshes.

## Spread? (bool):

If marked true, create a spread of prop meshes instead of one singular prop mesh.

## Mesh Count (int):

Number of meshes to create. Final number of meshes may be less if AttemptPreventOverlap? Is marked true.

## Normalize Rotation to Surface? (bool):

If marked true, attempt to rotate the mesh so that it appears to lay flat on the surface it is touching. Good for spreading props across a non-flat or sloped surface.

### Radius (float):

Half-size of the area to spread the meshes around. This area is actually a square and not a circle.

### Attempt Prevent Overlap? (bool):

If marked true, attempt to prevent spawning meshes if it collides with other prop meshes from the same blueprint. Needs to be used in conjunction with “Mesh Size Radius” below to estimate the size of the props because the actual mesh is not used when calculating the size.

### Mesh Size Radius (float):

Used in conjunction with Attempt Prevent Overlap, this value is roughly the half size of the largest mesh used in the prop mesh array pool. This value is used to calculate whether or not meshes will overlap with each other.