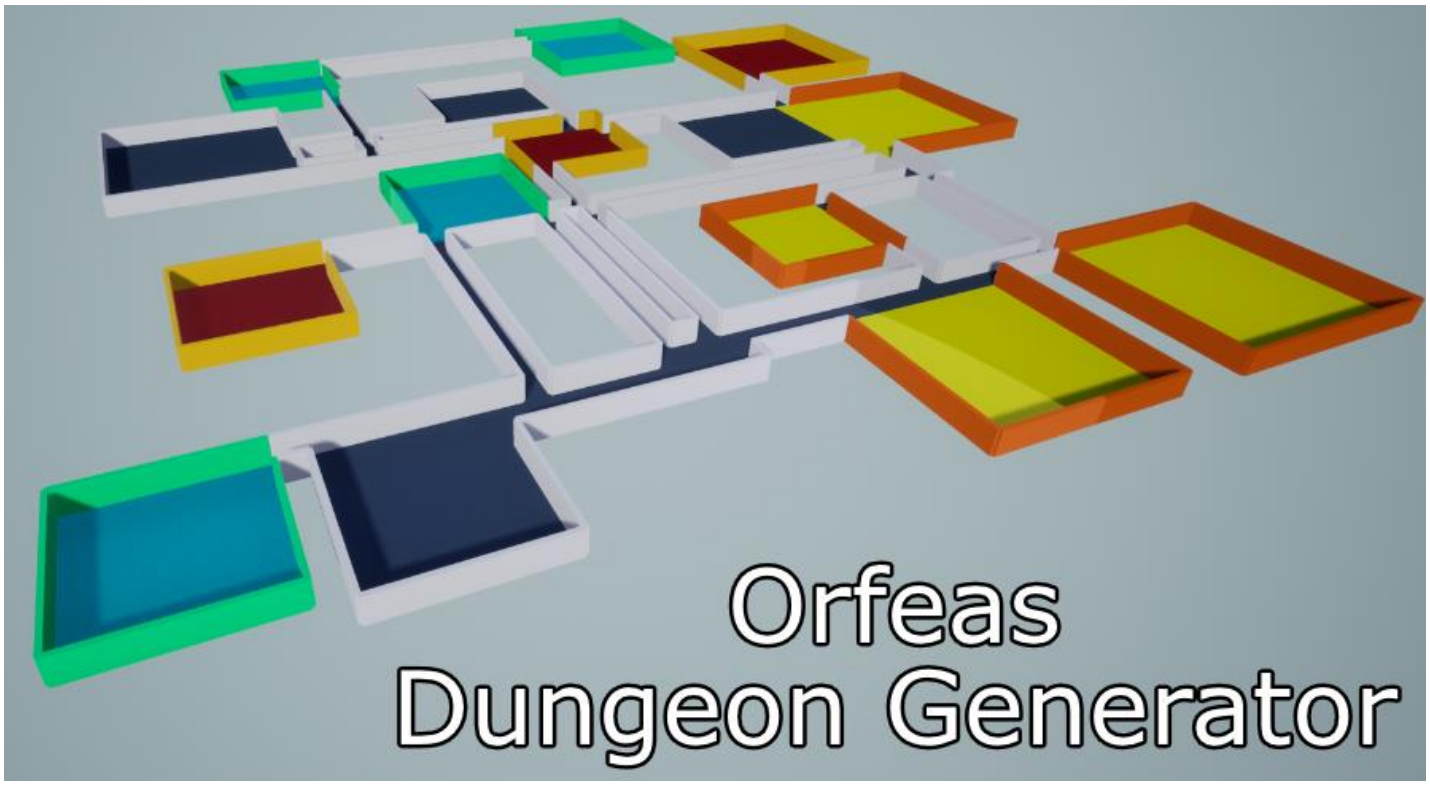
Orfeas Dungeon Generator



A handy dungeon generator for Unreal Engine 4. Can be used in editor and at runtime.

Date: November 16, 2021

Changelog

- Added description for material overrides in Data Table

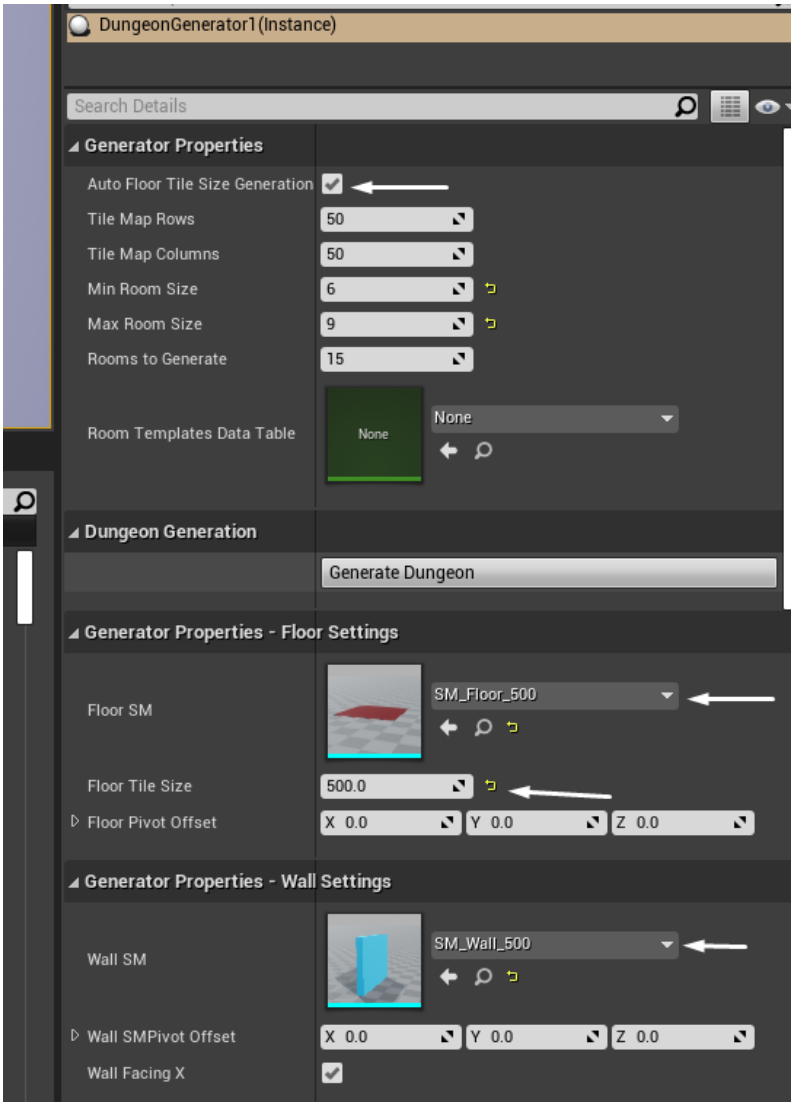
Contents

1. How to use

In order to use the dungeon generator:

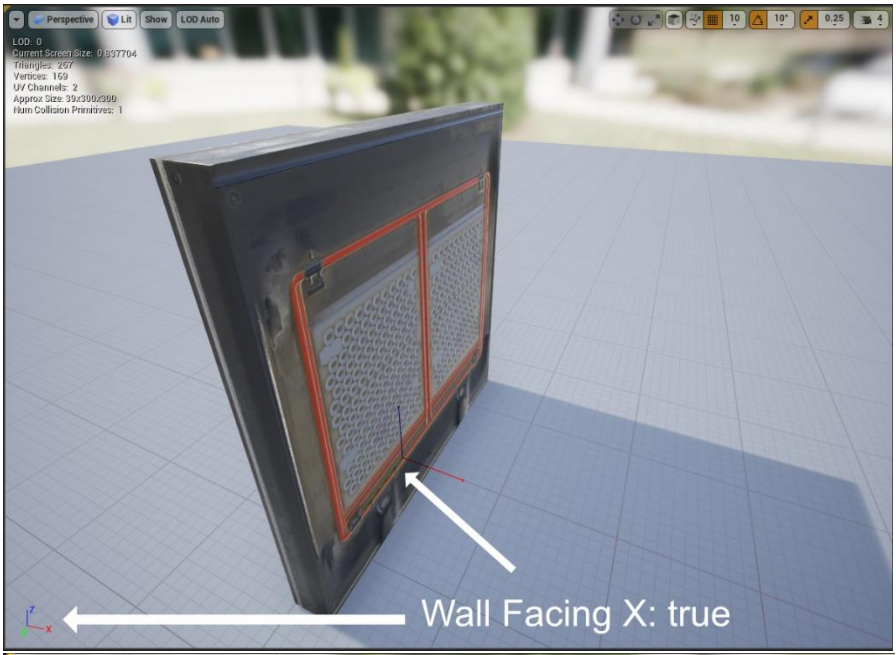
- Place a Dungeon Generator actor from the “Place Actors” panel
- From the details panel:
    - Assign a static mesh to FloorSM (the static mesh for each floor tile)
    - If the plugin doesn’t automatically detect the correct size, turn off the Auto Floor Tile Size Generation property and manually assign the correct value to Floor Tile Size
    - Assign a static mesh to WallSM (the static mesh for each wall). **The plugin assumes that wall extents match your assigned tile** meaning that if your tile size is 500 then your wall’s static mesh width should be 500 units as well.
- Click the Generate Dungeon button

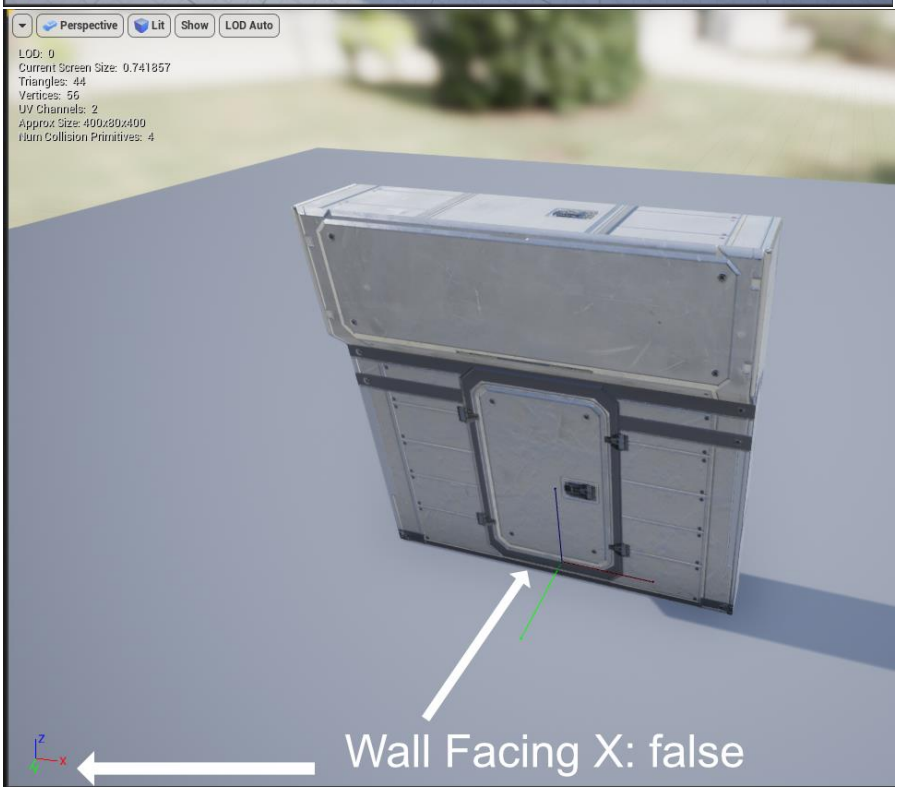
Here’s a screenshot summing up the following steps:



1.1 Configuring wall meshes

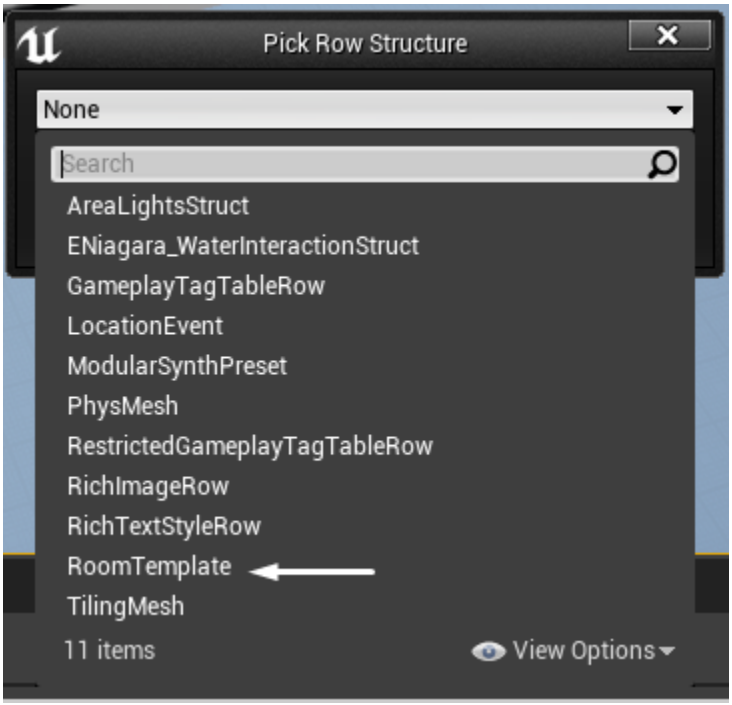
Depending on the imported wall mesh, you may have to modify the Wall Facing X property, found under Generator Properties – Wall Settings. To determine the correct value for your particular case, open up your wall mesh and see if your wall is facing the X axis. If that's the case, make sure to mark this option as true (false otherwise). An example of two meshes using this option:



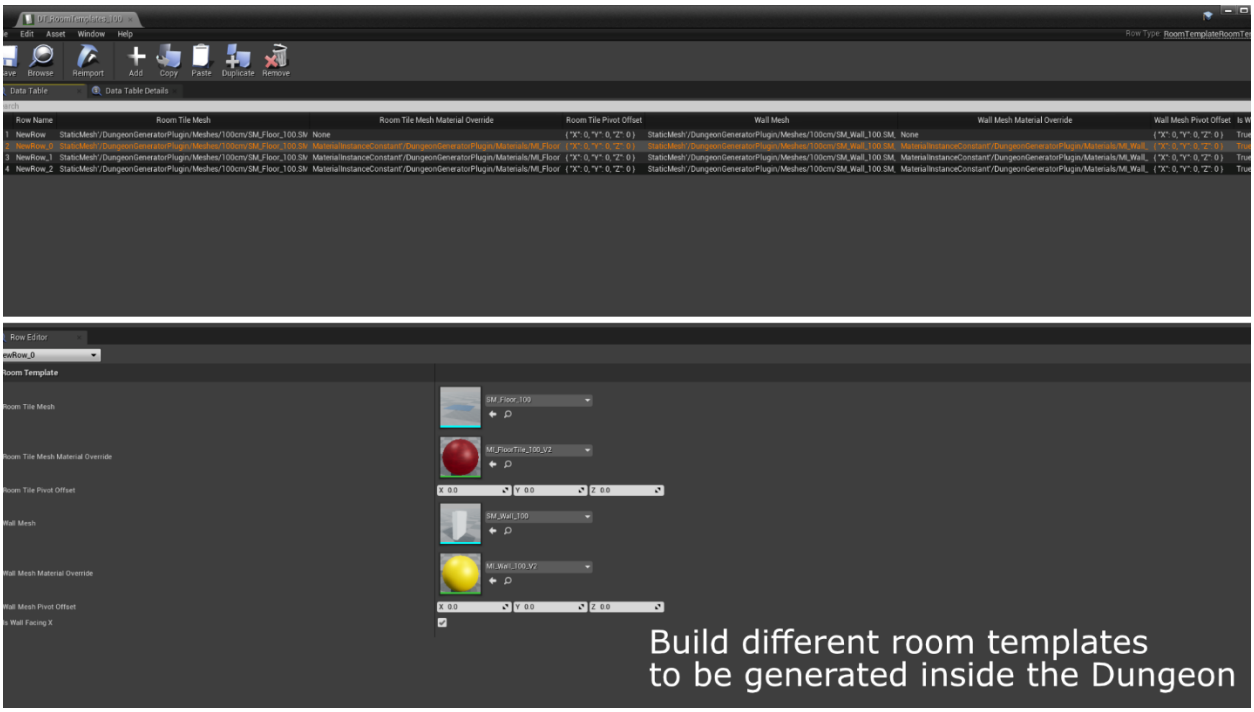


2. Using the Room Templates Data Table

This dungeon generator offers the option to use different floor and wall meshes for different rooms. To use this, create a new data table and choose the RoomTemplate type:



Then, proceed on adding your floor static mesh as well as the walls for this particular room:



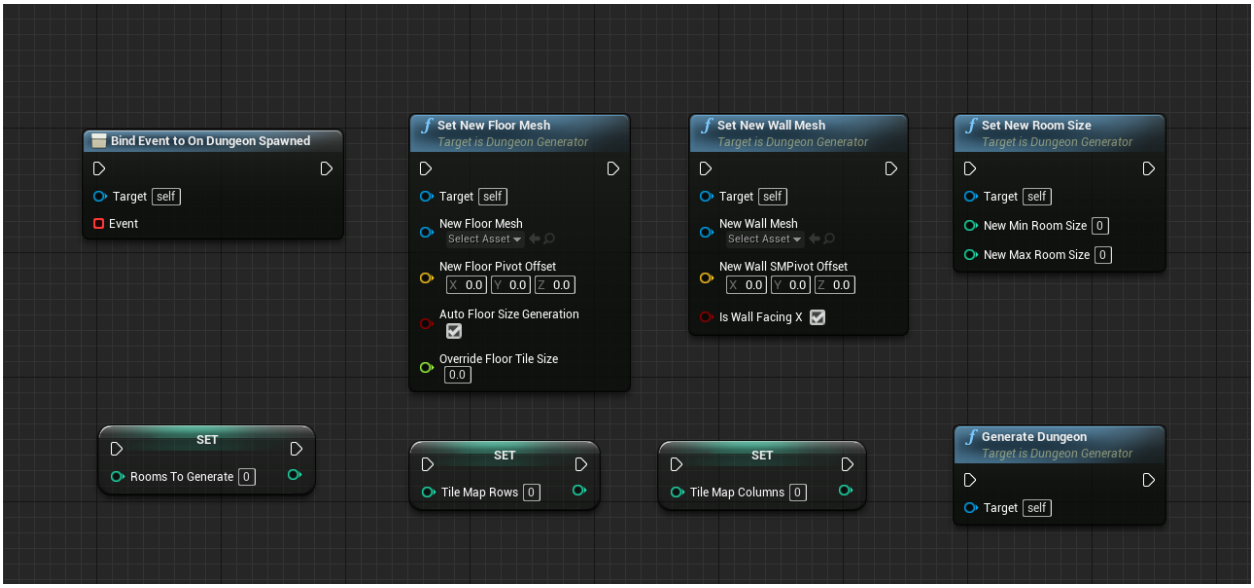
You can assign a material override for each individual mesh. This is used to generate different looking rooms with the same mesh!

The generator assumes that every floor tile is matching the "FloorSM" tile size you added on the details panel.

Once you have configured your data table, assign it to the details panel and generate a new dungeon. The generator will automatically try to use the assigned data table. Moreover, all the corridor floor tiles and walls will fall back to the default "FloorSM" and "WallSM" properties so **don't forget to assign these properties whether you're using the Data Table or not**.

3. Configuring the generator at Runtime

The dungeon generator actor provides the following BP functions and properties:



Moreover, you can use the OnDungeonSpawned callback to call any custom code when the generator has finished spawning various meshes.

4. A brief overview of what's happening behind the scenes

The plugin contains two C++ classes:

- A Tile Matrix class, responsible for generating locations of rooms and walls (therefore designing the whole dungeon)
- A Dungeon Generator class, which takes all the input from the Tile Matrix class and spawns the assigned meshes in the world, bringing the whole dungeon to life!

The tile matrix class initially creates a NxY matrix of Booleans. Afterwards, it randomly assigns true values to various elements on the matrix. Whenever a new element of the matrix is selected, the tile matrix tries to "fit" a room into the nearby tiles if they're not already occupied. If a room can fit in the selected space, the tile matrix proceeds on marking all the involved elements as true, otherwise the elements of the matrix remain unchanged and a room isn't spawned.

Each time the Tile Matrix creates a new room somewhere, it also connects it automatically to the previous added room using the shortest available path.

Once all required tiles in the matrix have been marked as “true” (which means at this point that we have generated all the rooms and the connections between them), we’re going through each occupied tile and peek its nearby tiles. For the nearby tiles that aren’t occupied, we need to spawn walls (since the tile matrix didn’t generate floor to that location), therefore the tile matrix also stores these locations to an array and also marks the orientation of the walls. The orientation comes in handy when we need to rotate walls based on their world location (see section 1.1)

Once the Tile Matrix completes the whole process, it provides a list of locations for all the floor tiles and wall locations to the Dungeon Generator Actor. Then, the dungeon generator removes any previously spawned meshes and creates the new meshes on the provided locations.