RoomGen - Procedural Generator

First, thank you for your purchase! Being able to build software and tools to support other developers like you is one of my passions!

If at any time while using these tools you encounter issues, or have any suggestions or feature requests, please email me at **angularfoxdev@gmail.com**

Getting Started

RoomGen was intended to be as user-friendly as possible, saving you valuable time while giving you the creative power to customize as you like.

Right click in the Hierarchy window and create an empty gameObject.

Reset the transform and position so the RoomGenerator is at 0,0,0.

Add the RoomGenerator component.

You can see now there are a whole bunch of customization options so you can tweak your results as much as you'd like.

Expand the **Levels** foldout. Click the plus button to add a new level. RoomGen puts together buildings from the ground up. Level 0 will be your ground floor, level 1 will be the next floor up, etc.

We'll cover the customization options for levels later on. Next, we need a preset!

Creating Presets

Presets are the driving force behind this tool. Presets allow you to create rooms using a non-destructive workflow and easily return to combinations of items you frequently use. This means with a simple drag and drop, you can create as many room combinations you can imagine.

Right click in the Project window and select Create > Room Generator > Room Preset

Name the preset to whatever you'd like. This name will be copied to the saved prefabs later on.

Drag the preset you just created into both the Level 0 preset slot, and the Preset slot in the Preset Editor section of the RoomGenerator component in the scene.

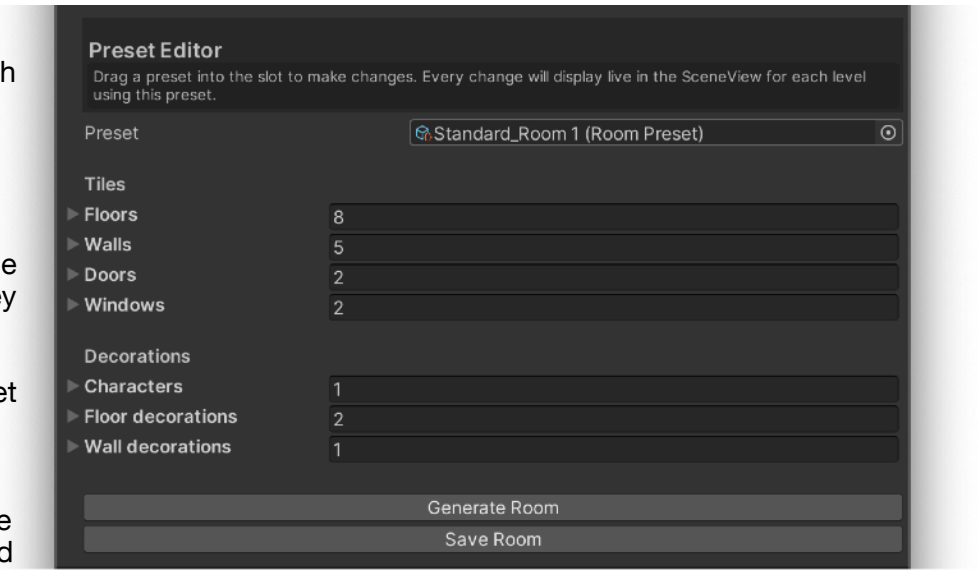
Now we are ready to customize!

Customizing Presets

Per-object settings within presets gives you complete control on how your room turns out.

The Preset Editor Component

Select the RoomGenerator object you created earlier. The **Preset Editor Component** gets automatically added to any GameObject with a generator component. The **Preset Editor** can be added to any type of RoomGen generator, including TargetGenerators. Changes made using the **Preset Editor** will make changes live to the generator component it is attached to. For other generators in your scene, you will just need to hit the "Generate" button to see the preset changes carry over to that object. You can have multiple preset editor components in a scene!



You will see a list of expandable options. Here, we will decide which game objects make up our room. As you fill these values in, you will see your room start to take shape instantly in the editor window. Making changes within the Preset Editor saves the preset at the same time, so as you make changes they will be saved for later use.

Any change you make in the Preset Editor will only effect the Levels that use this preset.

Expand the Floors section, and the Walls section. Click the plus to add a new element.

Drag and Drop your floor/wall prefabs into the **Prefab** slots. You should see your room start to take shape.

Weight: This field determines the probability of that specific tile being generated in your room. Weights are relative to the other tiles in their category. A higher weight has a higher chance of being generated.

<u>These are two of the most important factors when creating your rooms. If for some reason your room isn't lining up, adjust these first!</u>

Position Offset: field can be used if you need to compensate for non-uniform object origins.
Rotation Offset: field can be used to rotate your tiles in case they are not lining up, or if they don't fit within the bounding box.

Random Rotation: Floor and roof tiles have the option to allow a random number of rotations. This value can be set from 0 to 3, where 0 means no rotation, 1 is one 90º rotation, 2 is two 90º rotations and so on.

Allow Decor: If you want to allow decor/props to be populated on this tile, check this box.

Align to Surface: If you have uneven shaped floor tiles like dirt mounds for instance, and you want to make sure props align with the ground, check this box. RoomGen will raycast across this tile to align props with the surface.

Tile Layer: This should match the layer you assigned to the floor tile prefab. This is only required if you have checked the Surface alignment box.

<u>Important:</u> RoomGen relies on the wall, floor and door **tiles** to be the same size. Floor and wall **decorations** can be any size however.

Next, expand the Window and Door Tiles Sections. These are optional. If you need, you can generate rooms with no doors or windows.

Click the plus to add a new element. Follow the same steps as before.

Tip: if you are using Synty Studios assets specify a -2.5 value in the x direction for your doors. (Half the door width). You should see them line up in the doorways.

Decorations

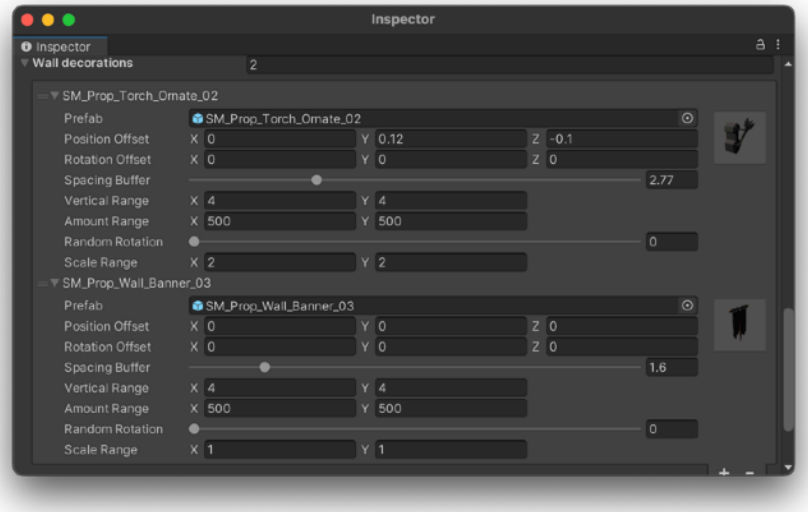
Here's where things get interesting!

Expand one of the foldouts in the decorations section. You will see a bunch of different options to control how each individual object type is placed. Drag and drop your prefab into the Prefab slot. You should see a preview appear of the prefab, and the item get populated in your scene.

RoomGen splits GameObjects up into 3 categories: Characters, Floor Decorations and Wall Decorations. These categories only apply to how these objects are placed, and their respective seeds.

Technically, any GameObject can be populated in each category. Be creative! If you have a character that needs to pop out of a wall, place it in the WallDecorations section and give it a little Z offset. This will make it look like its coming out of the wall!

Position Offset: Position spacing specific to this object. This helps if your object is clipping through walls or floors when the room is generated.
Experiment with these values! For instance, if you want chandeliers or hanging lights, create a floor decoration, and set the y value to a higher number, like 10, and it will create lights above the floor!



Rotation Offset: Apply a specific rotation offset to this object.

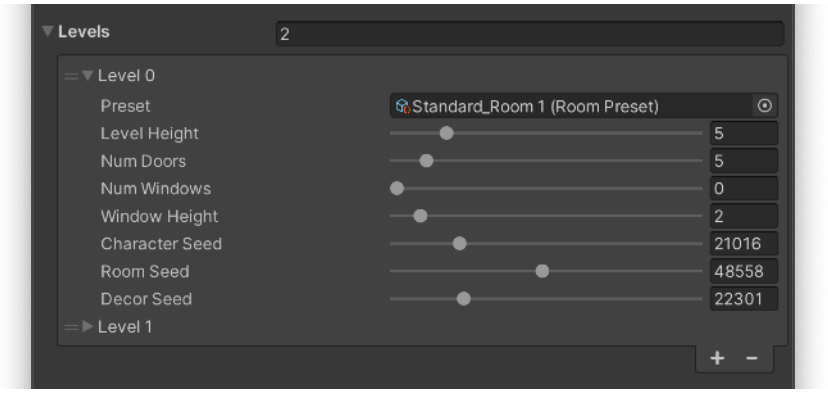
Spacing Buffer: This value determines the minimum amount of space between objects of **this same type**.
Say you want torches to line your walls, but you don't want them right next to each other. Increasing this value will space them apart.

Vertical Range: This determines the height range that this object can be placed on this Level. Each Level of your building starts at a value of zero and counts up from there. An easy way to set a max height is to turn on the **Debug** feature and count the number of points.

Amount Range: This range determines the MIN and MAX number of this object that will be placed into your scene, with the exception that there is a valid number of points. (ex. you have a 1x1 room, and set this to 10000, there won't be enough room for all of those objects, RoomGen handles this automatically.) If you want **an exact number of objects** set these two numbers to the same value.

Random Rotation: How much this object can be rotated. RoomGen sets a random value between 0 and this selected number.

Scale Range: The smallest (x) and largest (y) this object can be created in relation to its original size.



Levels

Levels make customization really easy! You can set specific presets for each level of your building. This is nice if you want a boss level on the lowest floor, and you want weak enemies only on the top floor.
If you decide you want to rearrange the building, simply drag the level above or below another and your room will reorganize itself.
Now that you've added walls, floors, and decorations, you should see a complete room in the SceneView. Let's dive into the Level parameters to further customize your room.

As you change property values in the inspector, you should see your room update in real-time in the editor.

Preset: This is where you will drag and drop your created presets. Every level can have its own preset!

Level Height: This is how tall this specific level will be. This effects how many windows vs doors will be placed if for instance you have a single story level.

Level Offset: This offsets the entire level by the specified amount. Use this if you have rectangular walls that are clipping with the next floor up, or if you want to give your building/ dungeon some horizontal variance.

Num Doors: This is the number of doors your room can have. Doors override windows in importance, so if for some reason your windows don't show up, you may have too many doors, or your window height should be set to a different level!

Num Windows: This is how many windows your room will have. RoomGen will only let you set a max number of windows per walls in your room. Ex. If your room is 2x2 your max windows will be 8 because you have 8 walls.

Window Height: This is the height of your windows in tiles. Ex. If you have a 2 story room, you can have windows up to 2 units high.

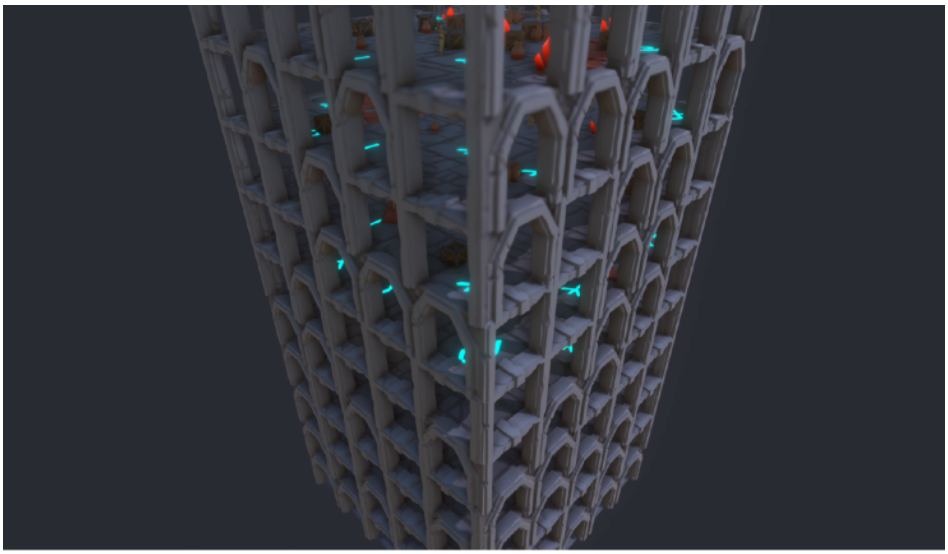
Character Seed: This lets you change the characters in a room without effecting the rest of the room settings. This is also useful if you have a particular room configuration you like, you can change and return to this seed to get the same character configuration.
Room Seed: This lets you change the walls/floors/doors/windows in a room without effecting the decorations. This is also useful if you have a particular room configuration you like, you can change and return to this seed to get the same room configuration.
Decor Seed: Same as the room seed, but this will change up the decorations in a room without effecting the room layout. Useful if you need different decor with the same door/window layout.

The Main Structure

Adjustments made to these parameters will effect every level within your building.

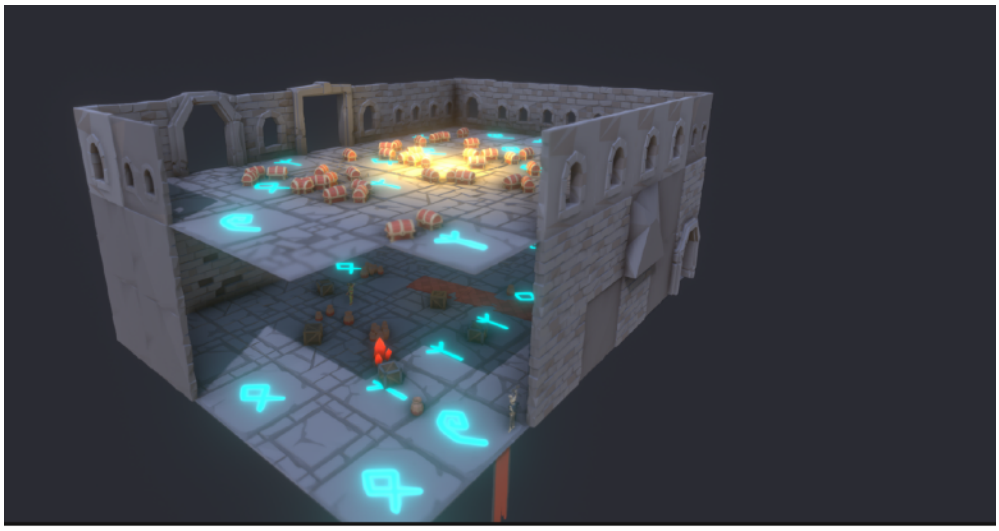
GridX: This is how many tiles large your room will be in the X direction.
GridZ: This is how many tiles large your room will be in the Z direction.

Tile Size: Set this value to how large your wall/floor/doors tiles are in units. For instance, Synty Studios tiles are typically 2.5 or 5 units large.

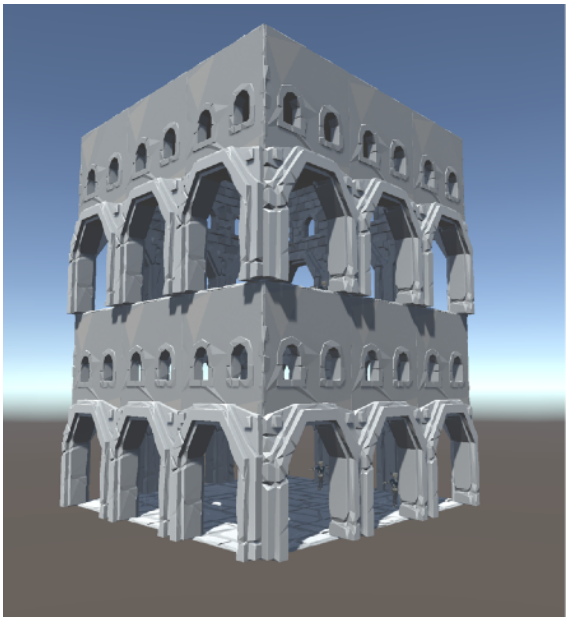


In the example below, the room has two Levels set up each with the same preset. The first floor is set to a level height of 2 to make it a little taller than the top floor.

Level presets let you determine exactly what spawns on each floor. Above, a general preset was created, then duplicated and adjusted to add more treasure chests on the top floor.

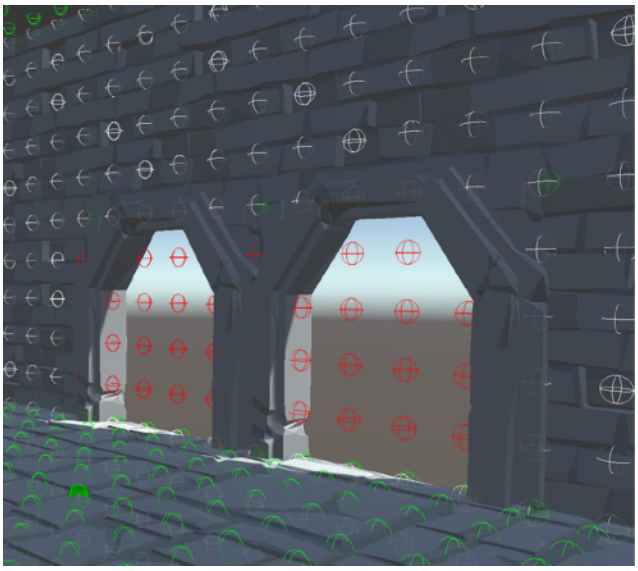


Floor Decoration Offset: Same as the wall decorations, this is a global variable that moves all the floor decorations upward, so they don't clip through the floor. This works in conjunction with the Y Offset within the decoration preset.
Wall Decoration Offset: If you notice your wall decorations are clipping through the walls, this is a global variable that will move all decorations away from the walls. This works in conjunction with the forward offset variable within the decoration preset.



Decor Safe Area: This is a useful variable that will move decorations away from the walls and more towards the center of the room.

Point Spacing: This value determines how closely packed your decorations will be in relation to your tile size. Normally a value of 1 will give adequate breathing room. If you have a small tile size, like 1, adjusting this up to 3 or more is a good idea in order to have more dense decorations. Changing this value effects the point density of



decorations, so your Vertical Range for each decoration will also become more compact.

Debug: This can be a very useful toggle if you want to see exactly where decor is going to be placed.

Green: floor points
Red: blocked decor points (eg, windows, doors)
White: Wall decor points
Blank: AllowDecor is set to false for that tile.

Generate Room: When changing the values above, the RoomGen will generate your room automatically. If for some reason your room doesn't generate automatically you can click this to regenerate it.

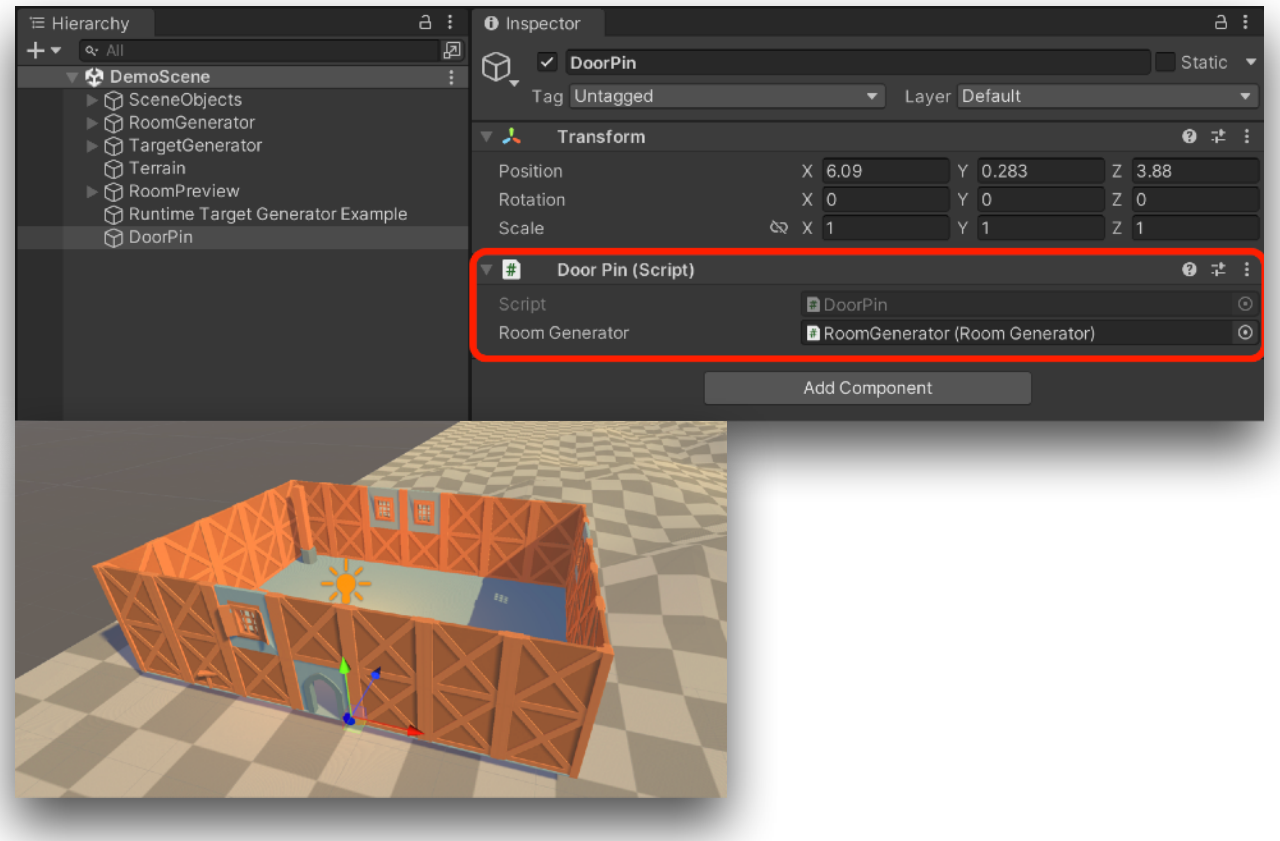
Save Room: This button will save your current room to a prefab in the assets folder. Useful to quickly create a whole bunch of room prefabs!

Door Pinning

Quickly and easily place doors where you want them.

One important part of creating rooms is being able to place doors where you need them. RoomGen can either place doors randomly for you, or you can create a DoorPin to tell RoomGen exactly where you want a door to be placed.

To create a DoorPin, right click in your Hierarchy and navigate to RoomGen > DoorPin. This will create a new GameObject in your scene with a DoorPin component. Drag your RoomGenerator component into the slot, and that's it! Simply drag this game object over the top of a wall in your generated room and you will see a door appear.



DoorPins work alongside your existing generation. Meaning if you have specified to have 3 doors in your room and you have created 1 DoorPin, 2 of your doors will be placed randomly and 1 will be placed at your door pin location.
DoorPins are not tied to a specific Level in your Room. You can move a DoorPin anywhere in your room and to any level and it will align your door for you.

Runtime Generation and API's

Modify generation at runtime with only a few lines of code.

Modifying RoomGen variables at runtime is fairly straightforward. The RoomGenerator component should add an *EventSystem* component when it is first added to the scene. (If it doesn't have one, you can add one easily through the *Add Component* menu).

The RoomGen EventSystem is responsible for delivering events to a specific RoomGen component. The ID parameter in each method corresponds to a RoomGen component in your scene.

The available methods are:

```
EventSystem.instance.SetGridSize(int id, int x, int z)

EventSystem.instance.SetPreset(int id, int x, int z)


EventSystem.instance.SetRoomSeed(int id, int levelNumber, int seed);
EventSystem.instance.SetDecorSeed(int id, int levelNumber, int seed);
EventSystem.instance.SetCharacterSeed(int id, int levelNumber, int seed);

EventSystem.instance.SetDoorCount(int id, int levelNumber, int count);
EventSystem.instance.SetWindowCount(int id, int levelNumber, int count);

EventSystem.instance.SetLevelOffset(int id, int levelNumber, Vector3 offset);
EventSystem.instance.SetLevelHeight(int id, int levelNumber, int height);

EventSystem.instance.Generate(int id);
```

Using the API's

To use these in your scripts, first include the RoomGen namespace to your script by adding ***using RoomGen*** to the inclusions.

Every call to the RoomGen API requires you to specify the **id** of the generator you want these changes to apply to, as there can be multiple RoomGen instances in a scene.

In this example we will update the size of the room, and then update the character seed to get some more randomization. It's important to note that the ***levelNumber*** parameters in these methods start at an index of zero. If you aren't sure which levelNumber, expand the ***Levels*** section of the RoomGen component and the level number will be listed there.

Add the methods to your script:

```
EventSystem.instance.SetGridSize(0, 5, 5)
EventSystem.instance.SetCharacterSeed(0, 0, 1000);
```

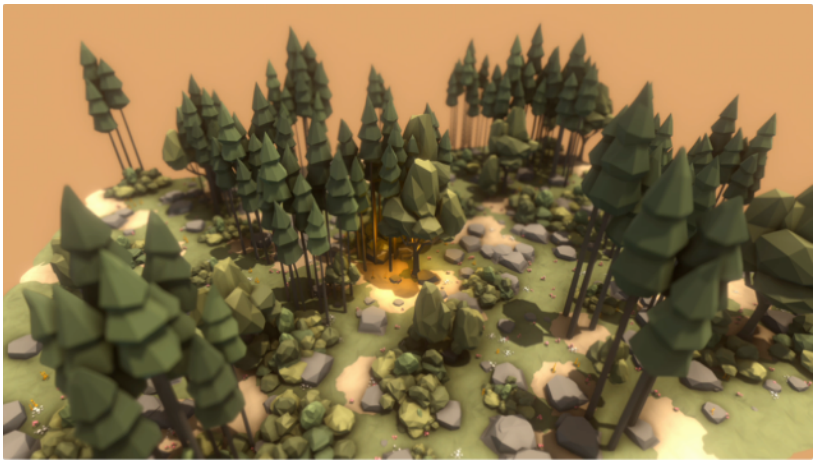
Next, we add the generate method. This is very important to get your scene to regenerate. Without this call you won't see any changes at runtime.

```
EventSystem.instance.Generate(0);
```

And that's it! With just those few lines of code you can regenerate your entire RoomGen scene during runtime. Use these methods to generate new levels programmatically, adjust character and decor seeds, and more.

Landscapes

Generating landscapes is just as easy as creating rooms!



Create a new preset for your desire landscape theme.

Add your ground objects to the floor tiles section. This can be dirt tiles, grass tiles etc.

Add your trees, grass, bushes and vegetation to the Floor Decorations section of RoomGen.

Specify probabilities for each type of vegetation and set the random rotation to 360 for more natural landscapes.

Play around with the GridX, GridZ values and the decor seeds! You can generate a whole bunch of landscape prefabs very quickly this way.

Target Generators

These are great for quickly placing props along a surface. Target generators distribute props within a circular radius.

Target generators work with RoomGen presets, similar to how the RoomGenerator component works.

Add the Target Generator component to an empty GameObject in your scene.

<u>Target Generators only use the Floor Decorations within a preset.</u>

Drag a preset into the slot.

If you want to make changes to a preset and see them reflected in the Target Generator, drag the same Preset into the preset editor component.

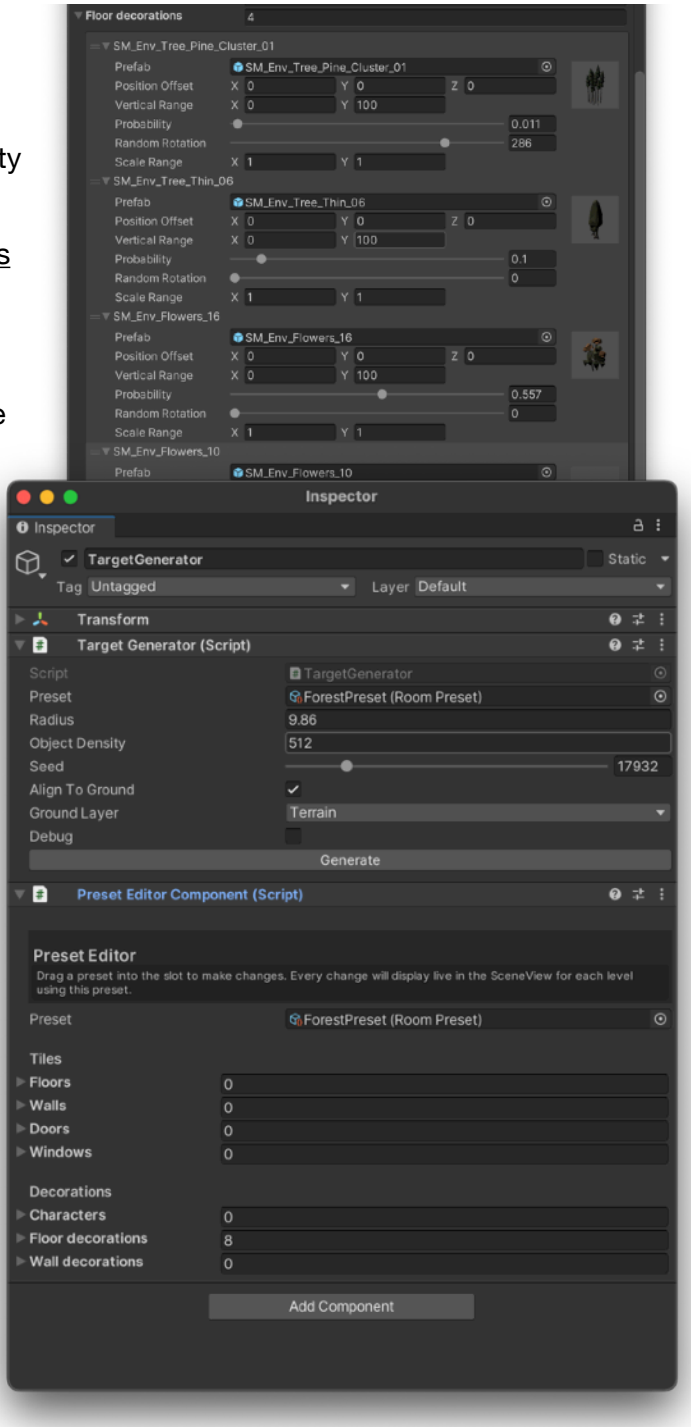
Adjust the radius to the desired size.

Adjust the object density to the desired amount. This determines how many objects will be placed, and how dense they will be packed.

If you would like the objects aligned to a surface like a terrain, uneven ground GameObjects, etc. Check the Align to Surface box.
Make sure to specify which layer your surface is on.

Check the **Debug** box if you want to visualize the object placement points.

You should see your objects being placed within your scene as you make changes in the inspector. If for some reason a change doesn't happen live, you can click "Generate".



Troubleshooting

If your wall/door tiles aren't lining up, rotated strangely or your wall decorations don't match the walls even after adjusting the position offsets try the following steps to get things lined up.

Nothing is happening in the scene:
Check to make sure you have added at least one Level, and that Level has a preset.

Walls or floors are rotated strangely:
Go to the object in the dropdown list. Try adjust the rotation until your tiles line up.

Walls or floors have gaps:
Go to the object in the dropdown list. Try adjust the position offset until your tiles line up.

Decorations aren't appearing:
Make sure that in the item settings you have set correct probabilities, scale ranges and that your vertical range is large enough. To test, set this range to something large like x: 0 and Y:100

If both your walls and floors have gaps between them, try adjusting the tile size variable. This value should match your prefab size.

Have fun and happy creating!


Angular Fox Dev