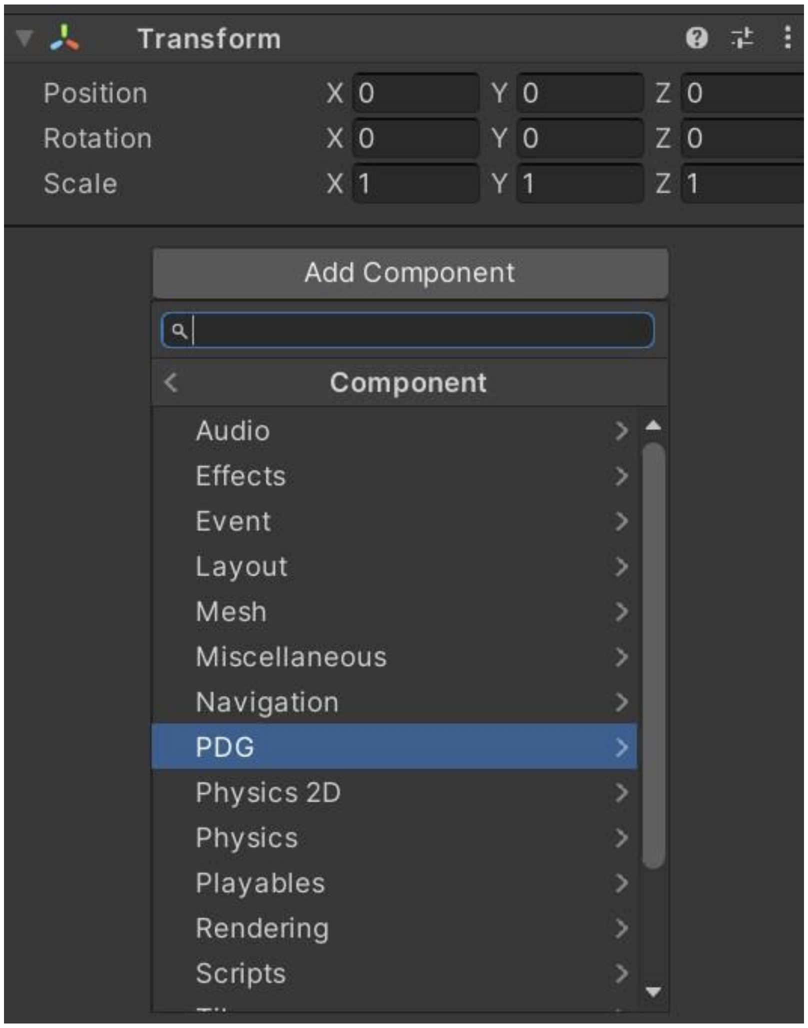
Quick start

This quick start guide shows fastest way to get Procedural Dungeon Generator working, and introduces basic customization options.

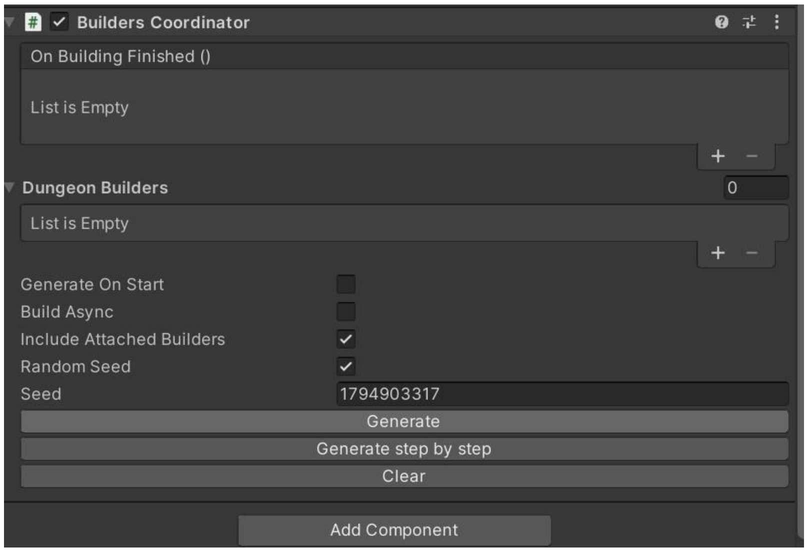
Minimal example

1. Create a new empty GameObject.
2. Add 3 components: RandomDungeon, TiledGameObjectDungeonBuilder, BuildersCoordinator.

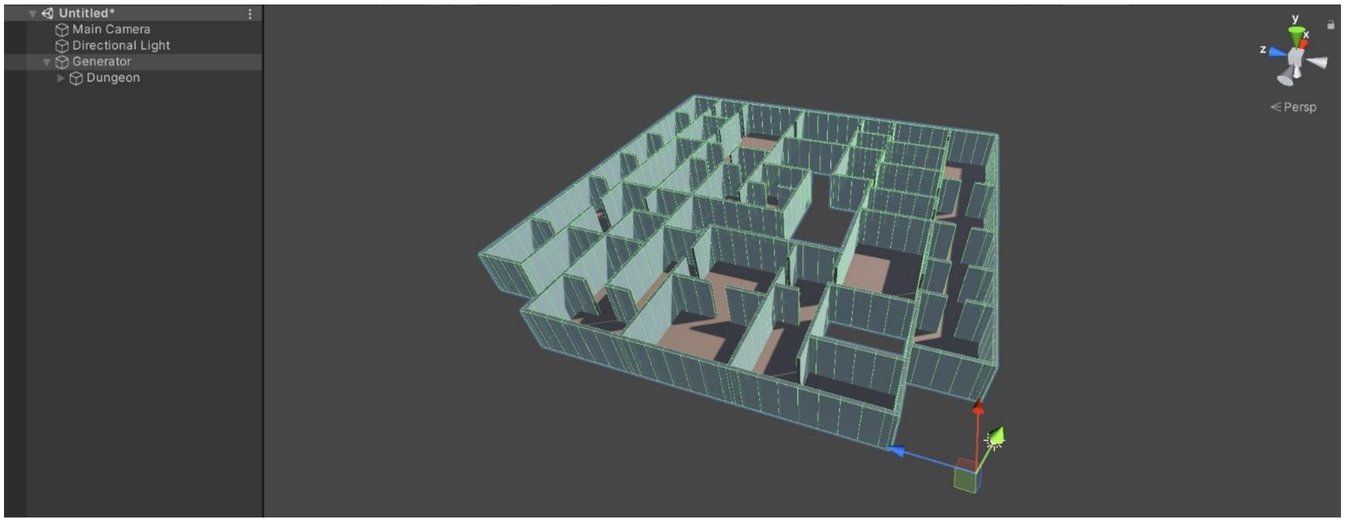
All of them are available in PDG menu.



3. Press "Generate" button on **BuildersCoordinator** component.


Builders Coordinator
On Building Finished ()
List is Empty
Dungeon Builders 0
List is Empty
Generate On Start ☐
Build Async ☐
Include Attached Builders ✓
Random Seed ✓
Seed 1794903317

4. The dungeon is ready.



You can also just launch the game and dungeon will be generated on start.

What happens when you press Generate button or start the game:

-**BuildersCoordinator** takes attached **LayoutGenerator**(**RandomDungeon** in this example) and uses it to generate **Layout.**

Layout is a container of rooms, where each room has its own specified position, size, tags, and attached connectors to other rooms.

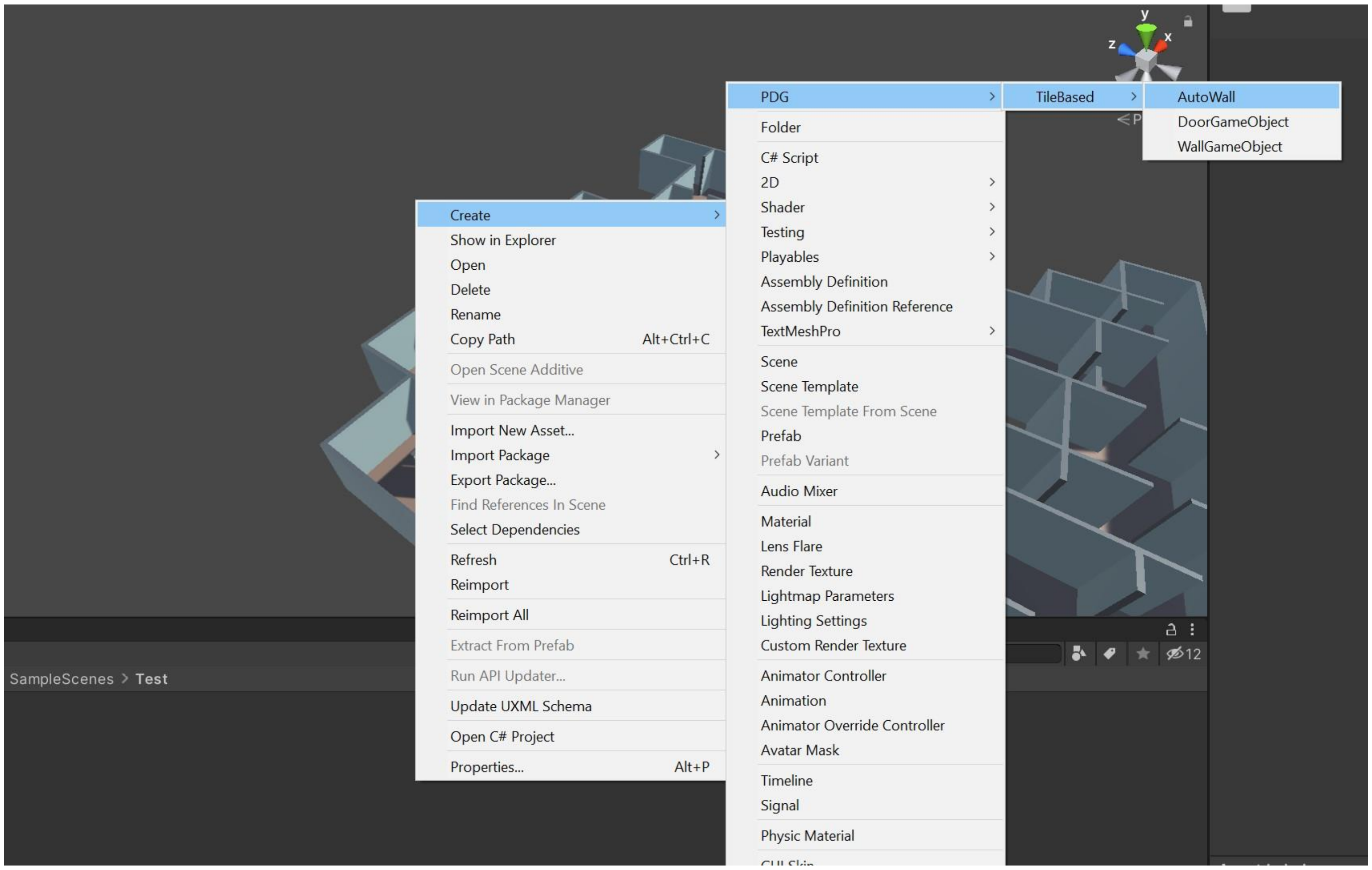
In other words, **Layout** specifies the structure of the dungeon.

-After that **BuildersCoordinator** collects all **DungeonBuilders**(**TiledGameObjectDungeonBuilder** in this example) attached to the same GameObject, as well as all builders from **DungeonBuilders list**, and passes the same layout to each of them.

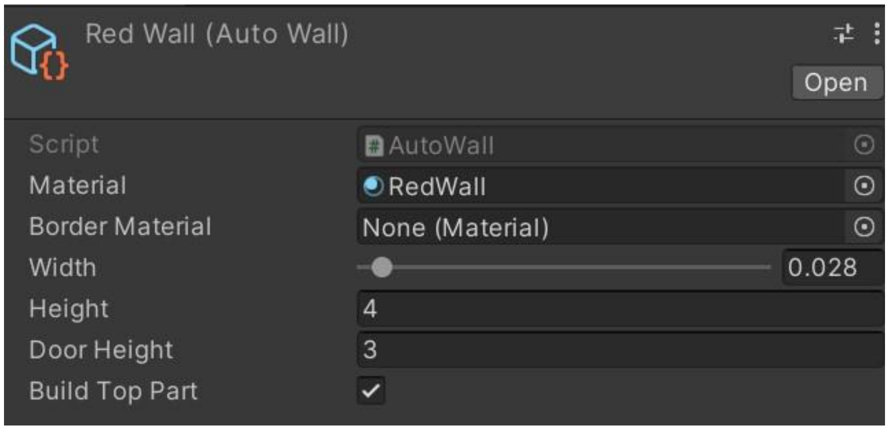
-After processing the Layout, each **DungeonBuilder** produces real GameObjects or places tiles, depending on its type.

Changing walls, floor and room type

1. Create an **AutoWall** from PDG->TileBased menu.

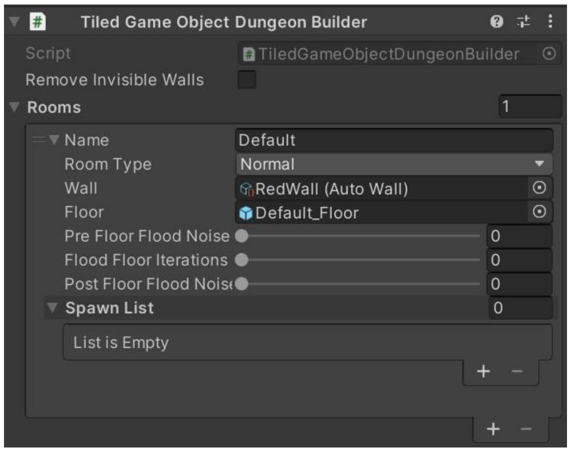


2. Assign some material and adjust parameters. Note: preferably set very small width.


Red Wall (Auto Wall)

Script	AutoWall
Material	RedWall
Border Material	None (Material)
Width	0.028
Height	4
Door Height	3
Build Top Part	✓

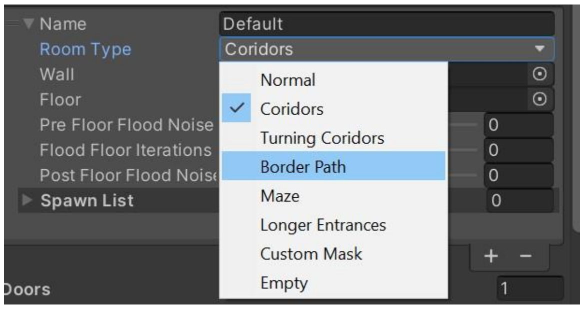
3. Go to **TileBasedGameObjectBuilder**, unfold Default room preset, and replace Default wall with newly created wall.



Tiled Game Object Dungeon Builder

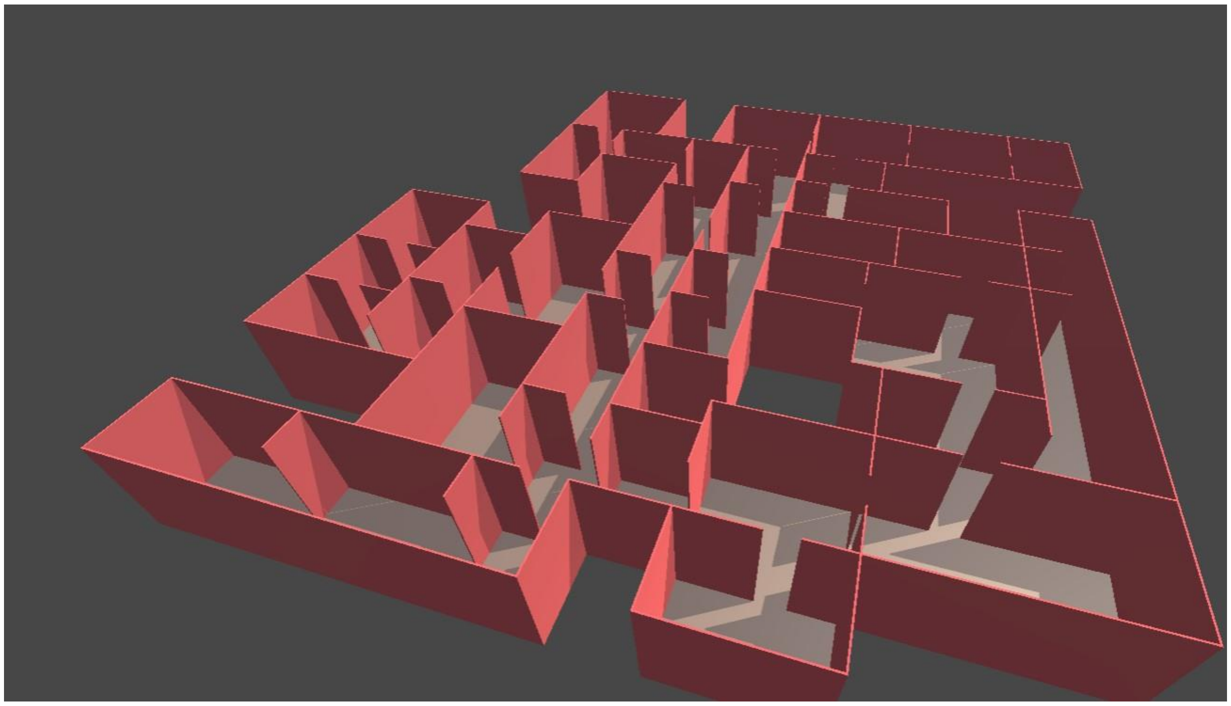
Script: TiledGameObjectDungeonBuilder
Remove Invisible Walls: ☐
Rooms: 1
Name: Default
Room Type: Normal
Wall: RedWall (Auto Wall)
Floor: Default_Floor
Pre Floor Flood Noise: 0
Flood Floor Iterations: 0
Post Floor Flood Nois: 0
Spawn List: 0
List is Empty

4. Here you can change the room type as well.



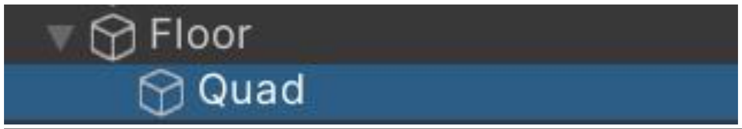
Name: Default
Room Type: Coridors
Wall
Floor
Pre Floor Flood Noise: 0
Flood Floor Iterations: 0
Post Floor Flood Noise: 0
Spawn List: 0
Doors: 1

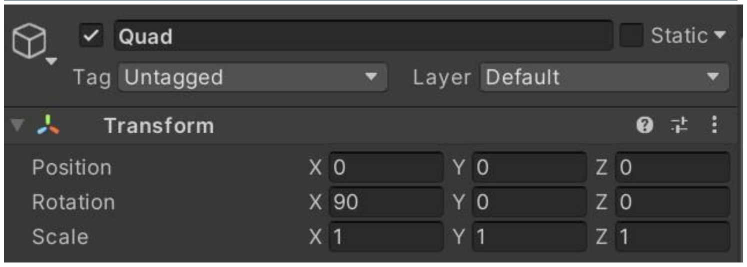
5. Now you can generate dungeon again to see the difference.



6. To change the floor you need to provide your own game object.

The easiest way to create custom floor - create empty GameObject, attach quad, and rotate quad by 90 degree, and optionally add another quad hiher to create a roof.

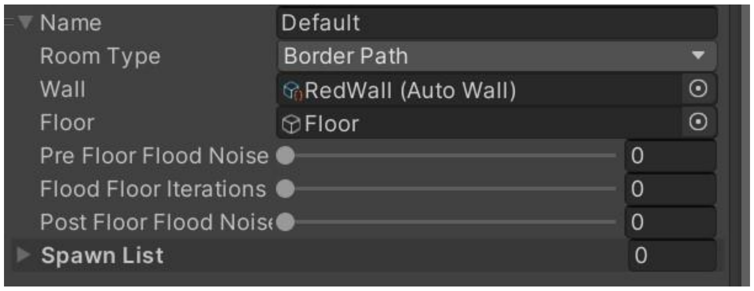




Quad
Static
Tag Untagged
Layer Default
Transform

	X	Y	Z
Position	0	0	0
Rotation	90	0	0
Scale	1	1	1

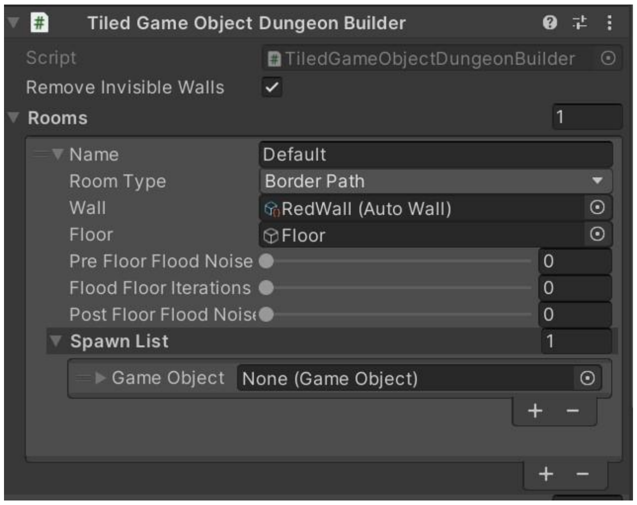
Then drag prefab or GameObject from scene to the floor field.



Name	Default
Room Type	Border Path
Wall	RedWall (Auto Wall)
Floor	Floor
Pre Floor Flood Noise	0
Flood Floor Iterations	0
Post Floor Flood Nois	0
Spawn List	0

Spawning GameObjects

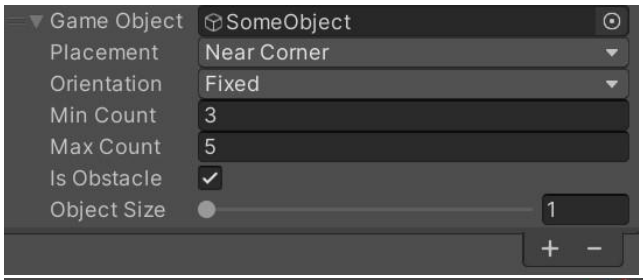
1. Unfold SpawnList in Default room preset and add a new element.



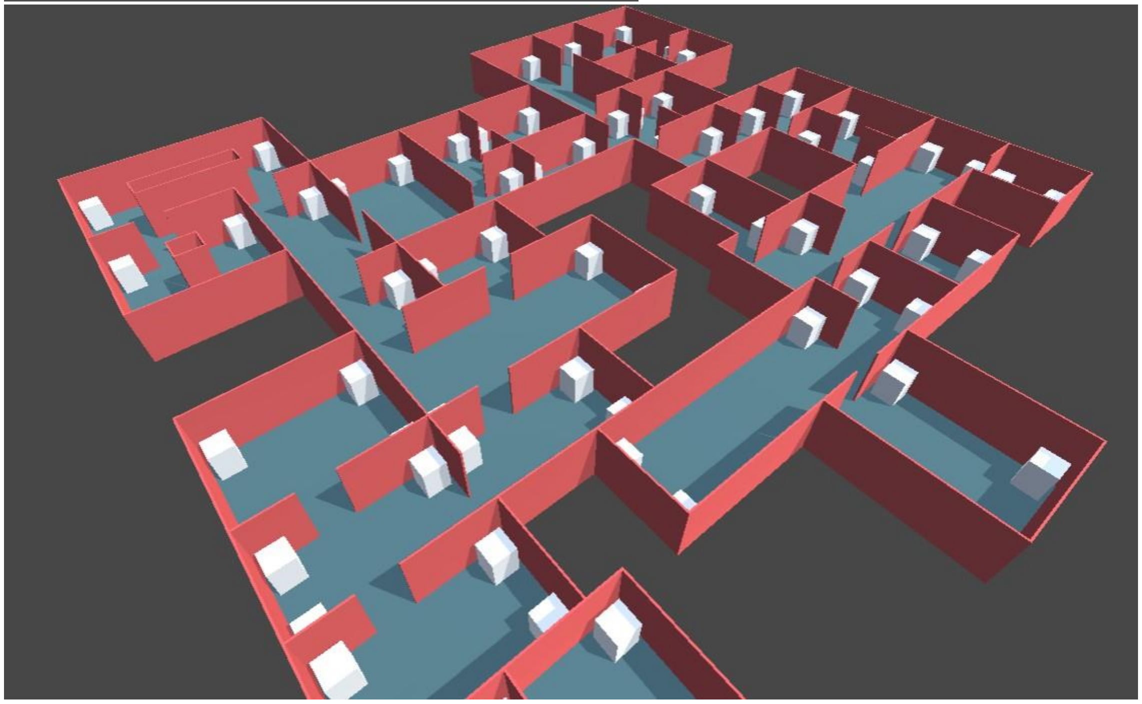
Tiled Game Object Dungeon Builder

Script	TiledGameObjectDungeonBuilder
Remove Invisible Walls	✓
Rooms	1
Name	Default
Room Type	Border Path
Wall	RedWall (Auto Wall)
Floor	Floor
Pre Floor Flood Noise	0
Flood Floor Iterations	0
Post Floor Flood Nois	0
Spawn List	1
Game Object	None (Game Object)

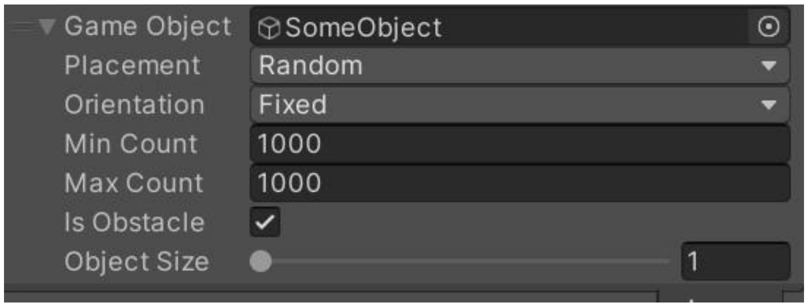
2. Assign any GameObject, and set it's parameters..



Game Object	SomeObject
Placement	Near Corner
Orientation	Fixed
Min Count	3
Max Count	5
Is Obstacle	✓
Object Size	1

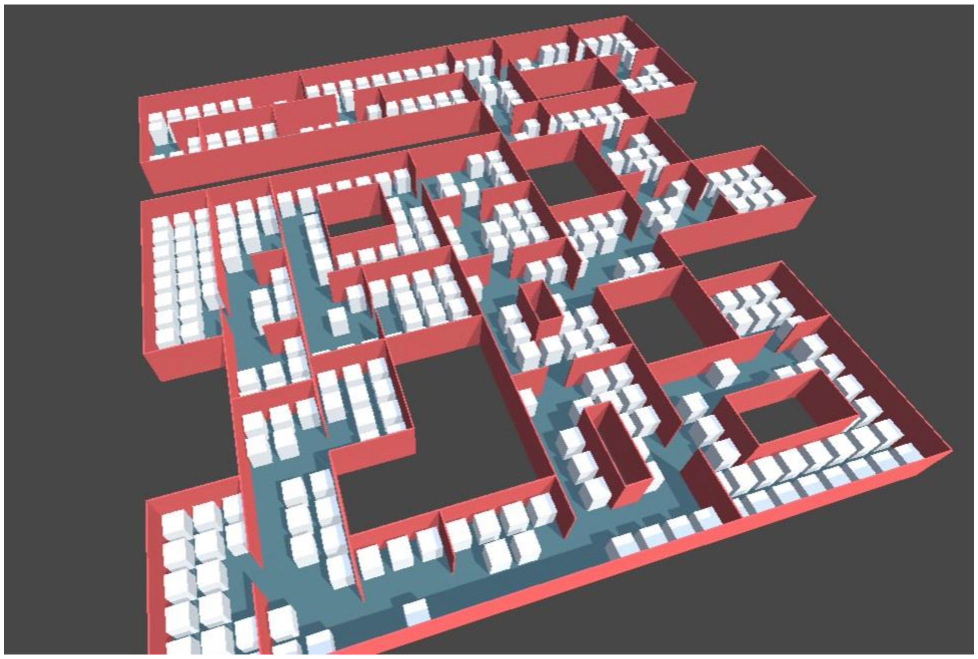


Note:
There will always be a way to get to any point of the room, without going through objects which are marked by IsObstacle.
For example let's set object count to 1000(it doesn't affect performance - process will be stopped as soon as room runs out of space), and enable IsObstacle mode

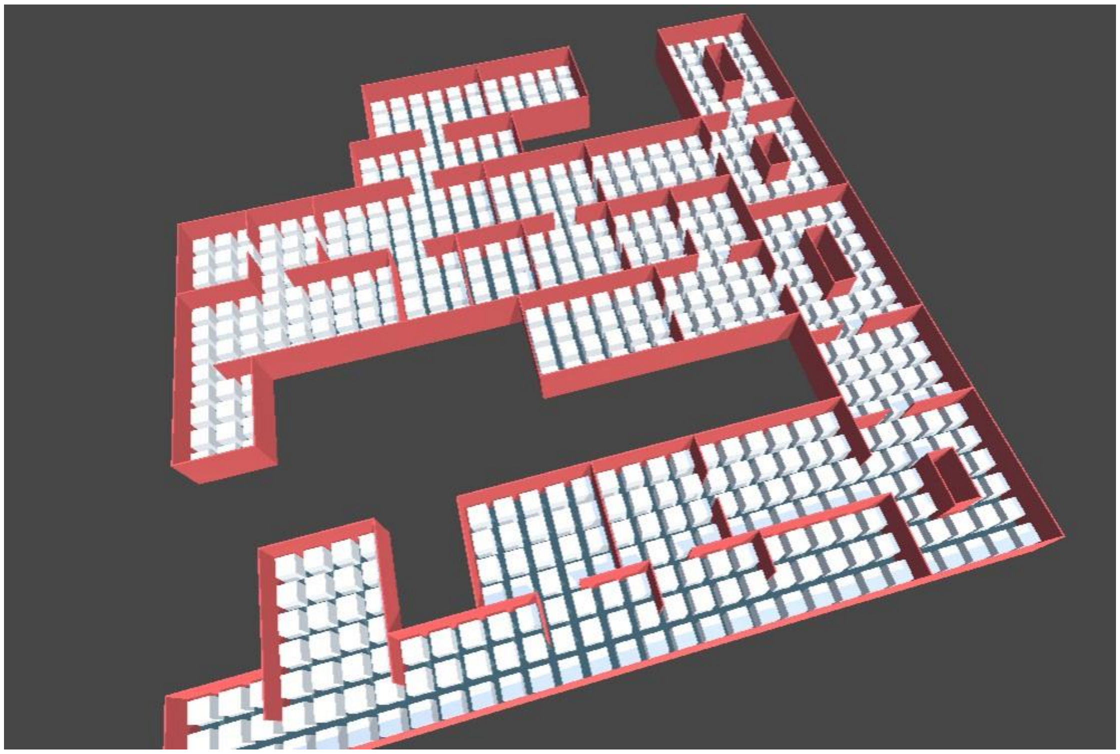


Game Object	SomeObject
Placement	Random
Orientation	Fixed
Min Count	1000
Max Count	1000
Is Obstacle	✓
Object Size	1

Each room left some free space to ensure a path to each connected room.

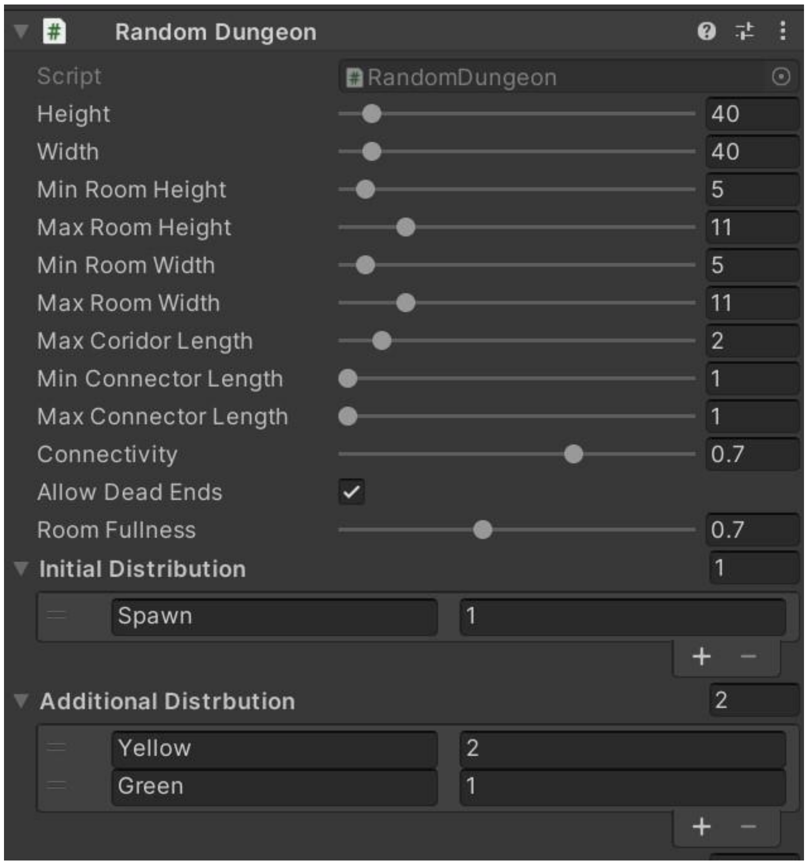


If you disable it - whole dungeon will be filled.



Setting up more room presets

1. Go to RandomDungeon Component and add new element to InitialDistribution and Additional Distribution.

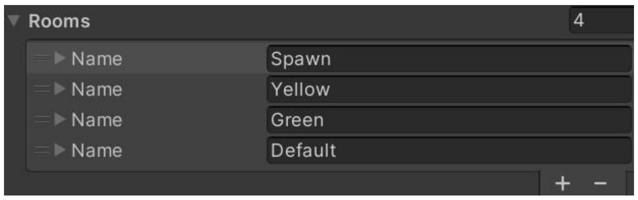


Random Dungeon

Script	RandomDungeon
Height	40
Width	40
Min Room Height	5
Max Room Height	11
Min Room Width	5
Max Room Width	11
Max Coridor Length	2
Min Connector Length	1
Max Connector Length	1
Connectivity	0.7
Allow Dead Ends	✓
Room Fullness	0.7
Initial Distribution	1
Spawn	1
Additional Distrbution	2
Yellow	2
Green	1

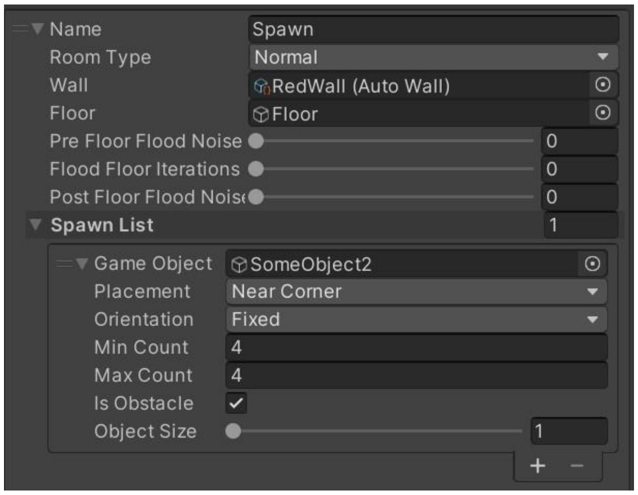
These parameters add tags to random rooms.
Initial distribution precisely specified required amount of rooms. In this case - there will be only one Spawn room.
Additional distribution takes the rest of the rooms, and distributes tags according to specified weights. In this case - there will be 2x more Yellow rooms than Green rooms.
if you press Generate button now - nothing will change, because parameters for new rooms were not specified, and they will use Default preset.

2. Add room parameters for rooms with new tags.

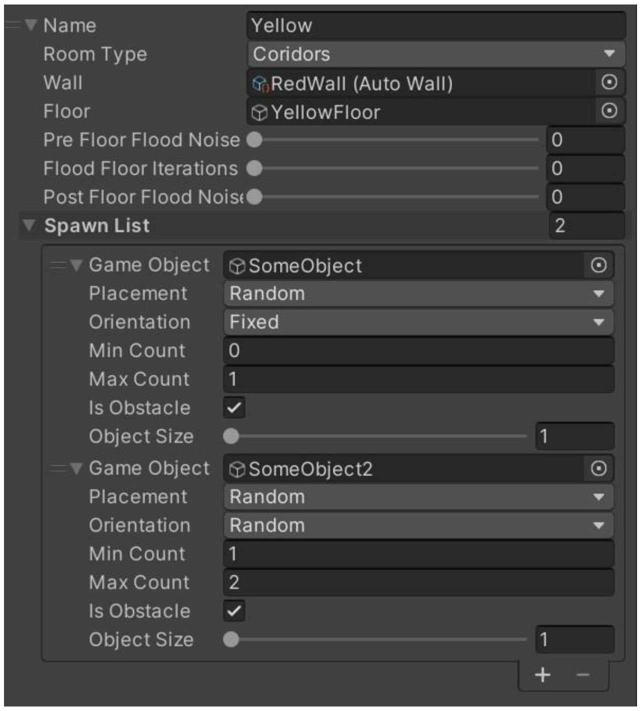


Rooms	4
Name	Spawn
Name	Yellow
Name	Green
Name	Default

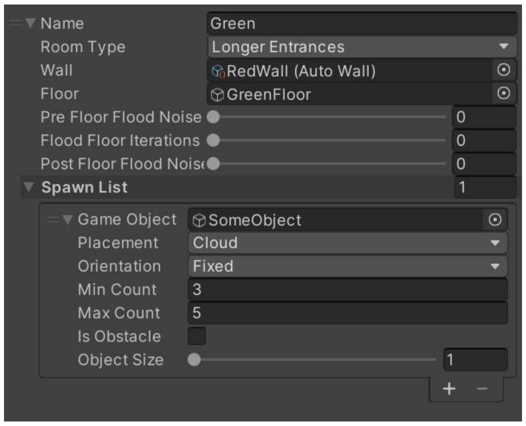
Parameters for following example:



Name	Spawn
Room Type	Normal
Wall	RedWall (Auto Wall)
Floor	Floor
Pre Floor Flood Noise	0
Flood Floor Iterations	0
Post Floor Flood Nois	0
Spawn List	1
Game Object	SomeObject2
Placement	Near Corner
Orientation	Fixed
Min Count	4
Max Count	4
Is Obstacle	✓
Object Size	1

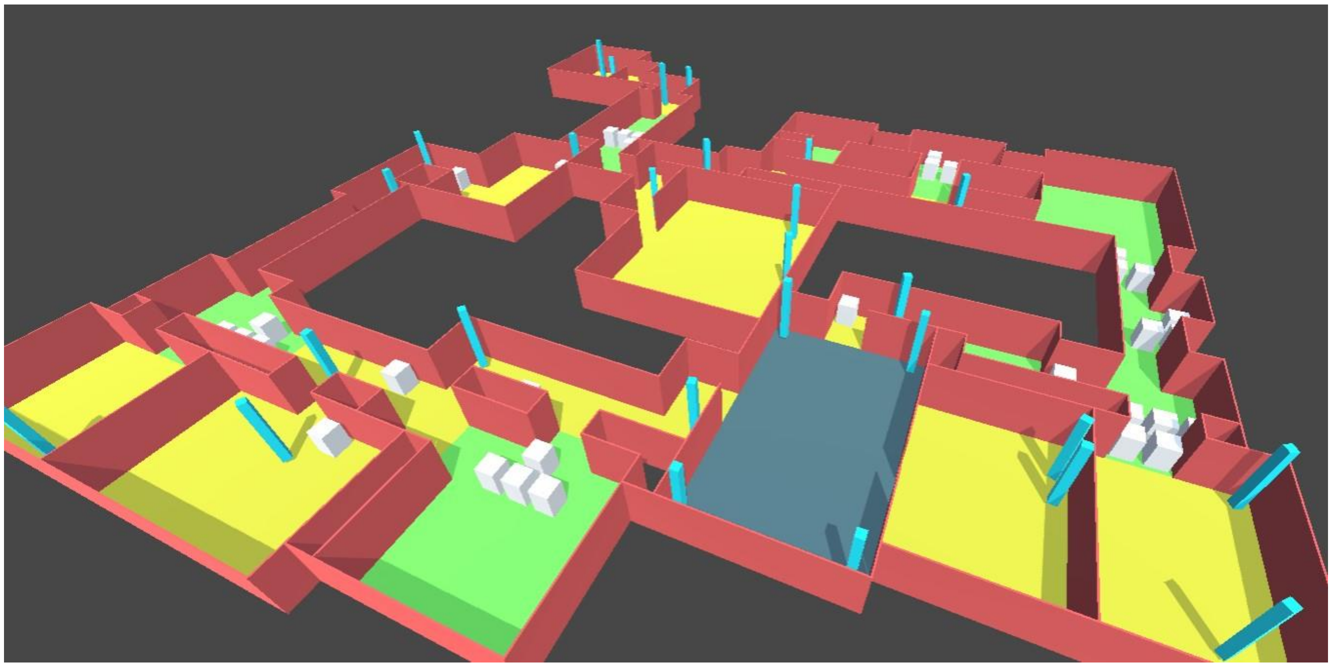


Name	Yellow
Room Type	Coridors
Wall	RedWall (Auto Wall)
Floor	YellowFloor
Pre Floor Flood Noise	0
Flood Floor Iterations	0
Post Floor Flood Nois	0
Spawn List	2
Game Object	SomeObject
Placement	Random
Orientation	Fixed
Min Count	0
Max Count	1
Is Obstacle	✓
Object Size	1
Game Object	SomeObject2
Placement	Random
Orientation	Random
Min Count	1
Max Count	2
Is Obstacle	✓
Object Size	1



Name	Green
Room Type	Longer Entrances
Wall	RedWall (Auto Wall)
Floor	GreenFloor
Pre Floor Flood Noise	0
Flood Floor Iterations	0
Post Floor Flood Nois	0
Spawn List	1
Game Object	SomeObject
Placement	Cloud
Orientation	Fixed
Min Count	3
Max Count	5
Is Obstacle	
Object Size	1

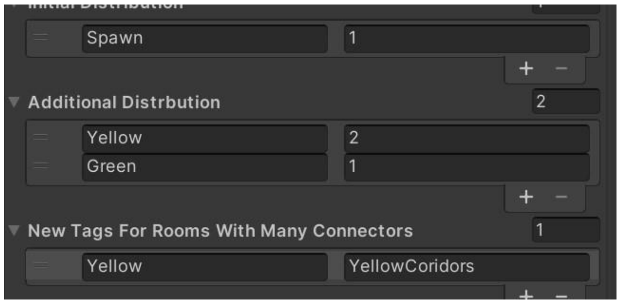
Now let's see the result:



Yellow rooms were specified to be Coridors, however, some of them were turned into normal rooms.

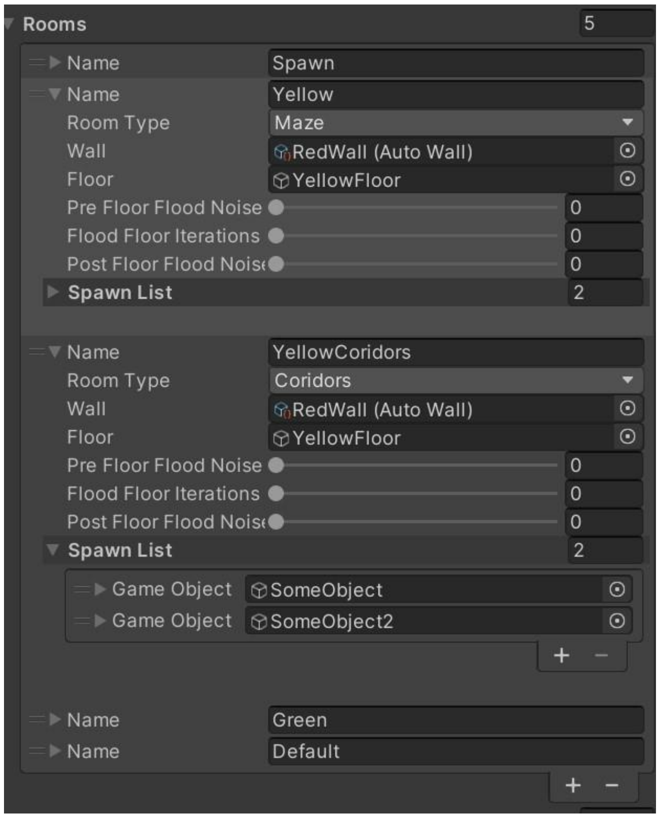
This happens, because when room has only 1 connector, it turns into normal room.

To control this behavior, add new element to **New Tags For Rooms With Many Connectors**



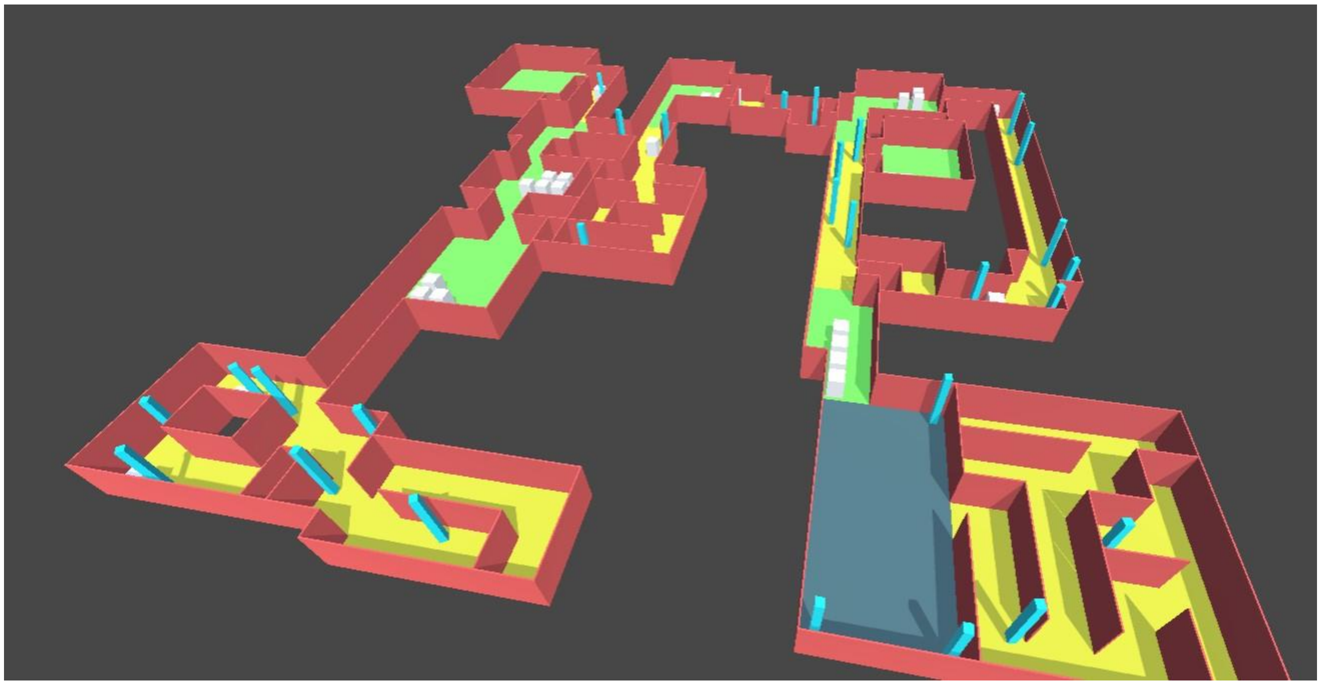
Spawn	1
Additional Distrbution	2
Yellow	2
Green	1
New Tags For Rooms With Many Connectors	1
Yellow	YellowCoridors

Each room with many connectors and Yellow tag will be converted into YellowCoridor.
Let's set Yellow room type to Maze, and YellowCoridor type to Coridors.



```
Rooms                          5
  Name                         Spawn
  Name                         Yellow
    Room Type                  Maze
    Wall                       RedWall (Auto Wall)
    Floor                      YellowFloor
    Pre Floor Flood Noise      0
    Flood Floor Iterations     0
    Post Floor Flood Noise     0
    Spawn List                 2
  Name                         YellowCoridors
    Room Type                  Coridors
    Wall                       RedWall (Auto Wall)
    Floor                      YellowFloor
    Pre Floor Flood Noise      0
    Flood Floor Iterations     0
    Post Floor Flood Noise     0
    Spawn List                 2
      Game Object              SomeObject
      Game Object              SomeObject2
  Name                         Green
  Name                         Default
```

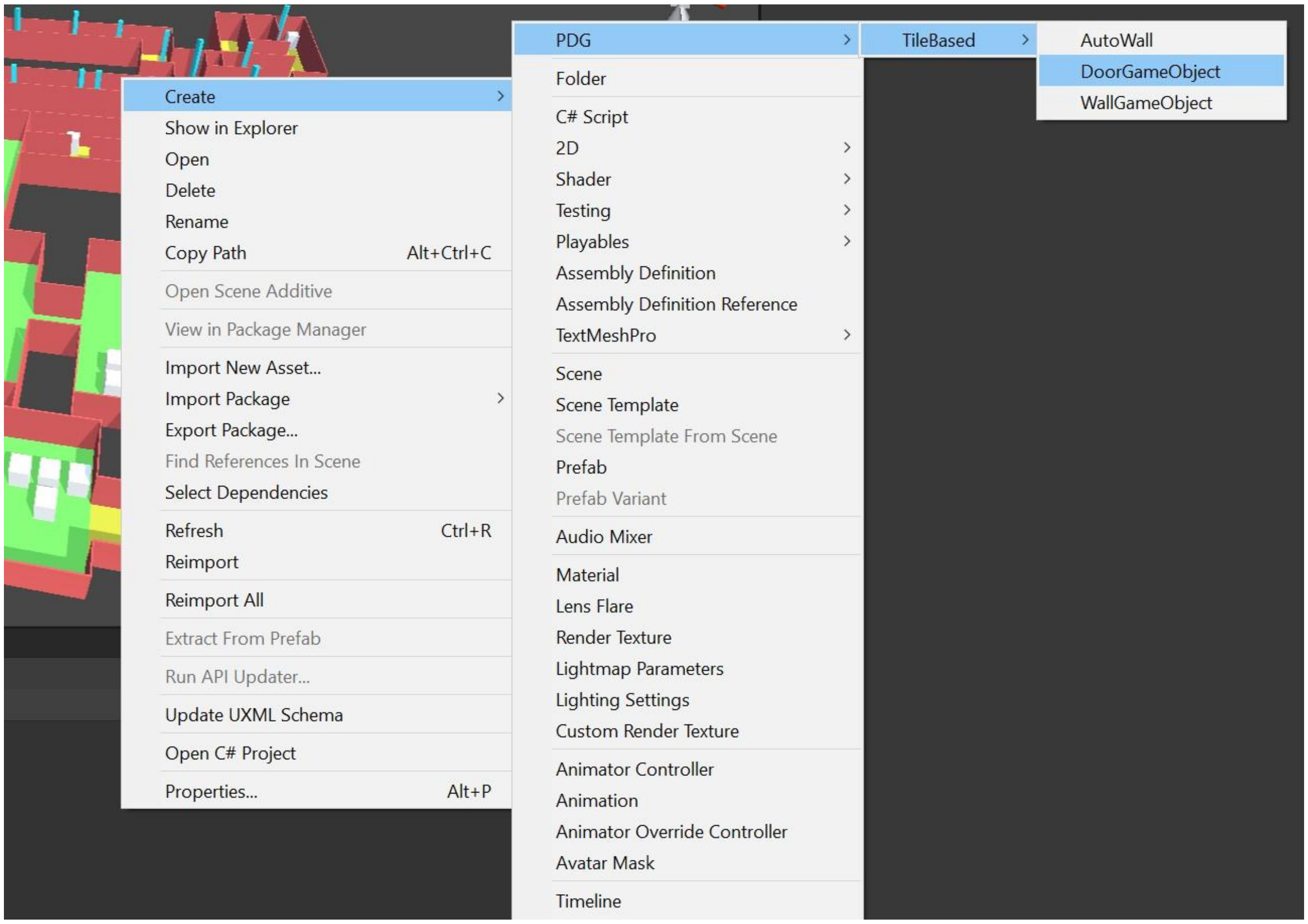
The result will look like this:



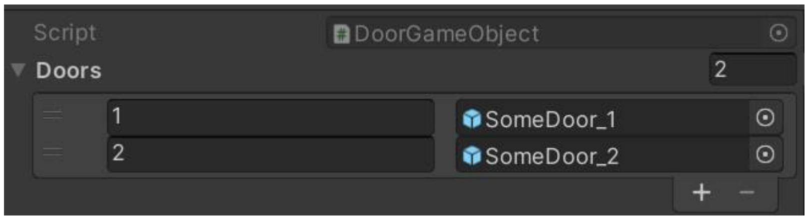
Note: walls between two rooms will be chosen according to their priority in the list(higher = higher priority).

Adding doors

1. Create DoorGameObject.

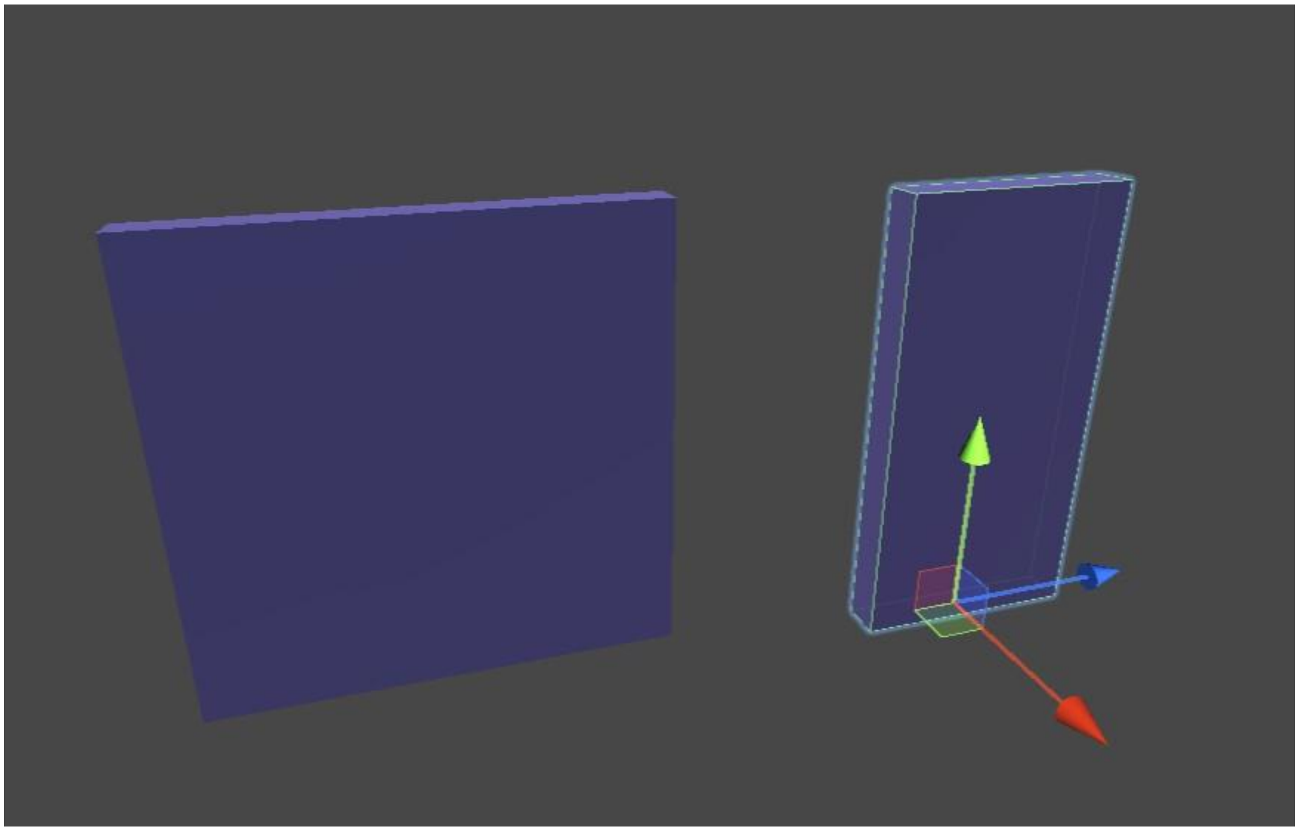


2. Here you can select door prefabs for different door lengths.

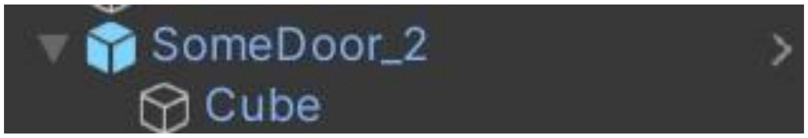


Script	DoorGameObject
Doors	2
1	SomeDoor_1
2	SomeDoor_2

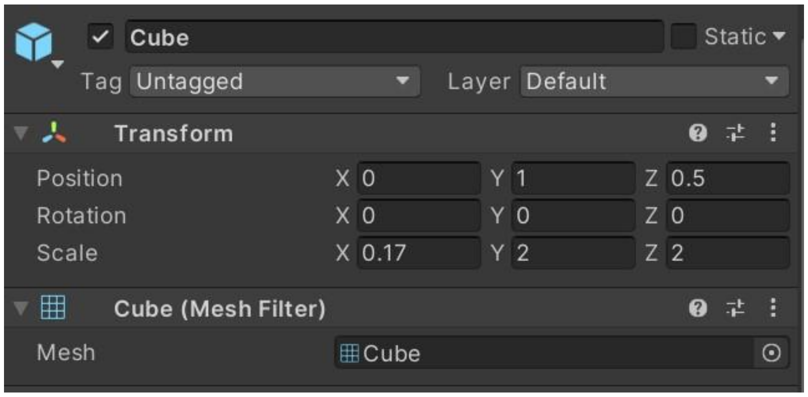
3. Door prefabs must face X direction, and aligned around center.



For this example, we can create empty GameObject and attach Cube to it.



And then set Cube's transform like this:



Cube

Static

Tag Untagged

Layer Default

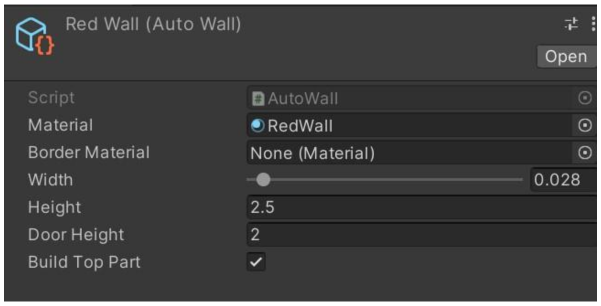
Transform

	X	Y	Z
Position	0	1	0.5
Rotation	0	0	0
Scale	0.17	2	2

Cube (Mesh Filter)

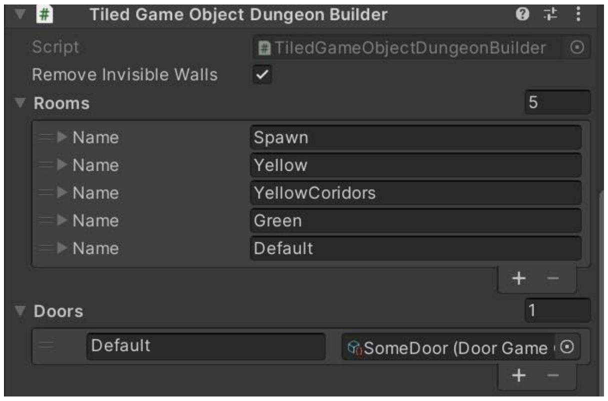
Mesh Cube

4. Optionally, adjust wall parameters to create proper doorway.


Red Wall (Auto Wall)

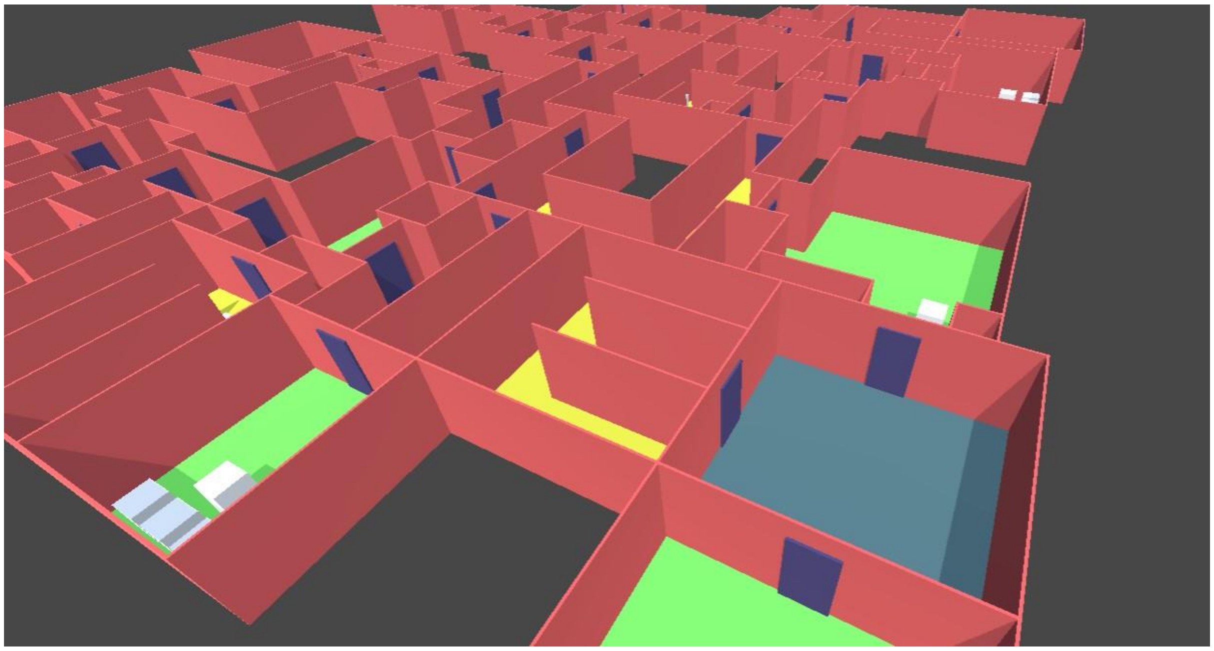
Script	AutoWall
Material	RedWall
Border Material	None (Material)
Width	0.028
Height	2.5
Door Height	2
Build Top Part	✓

5. Add Default door to our TiledGameObjectDungeonBuilder.

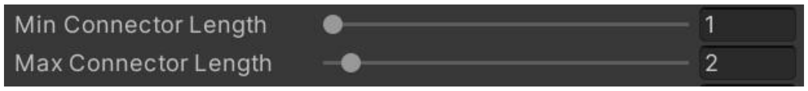

Tiled Game Object Dungeon Builder

Script	TiledGameObjectDungeonBuilder
Remove Invisible Walls	✓
Rooms	5
Name	Spawn
Name	Yellow
Name	YellowCoridors
Name	Green
Name	Default
Doors	1
Default	SomeDoor (Door Game

If you press Generate - you will see the doors.

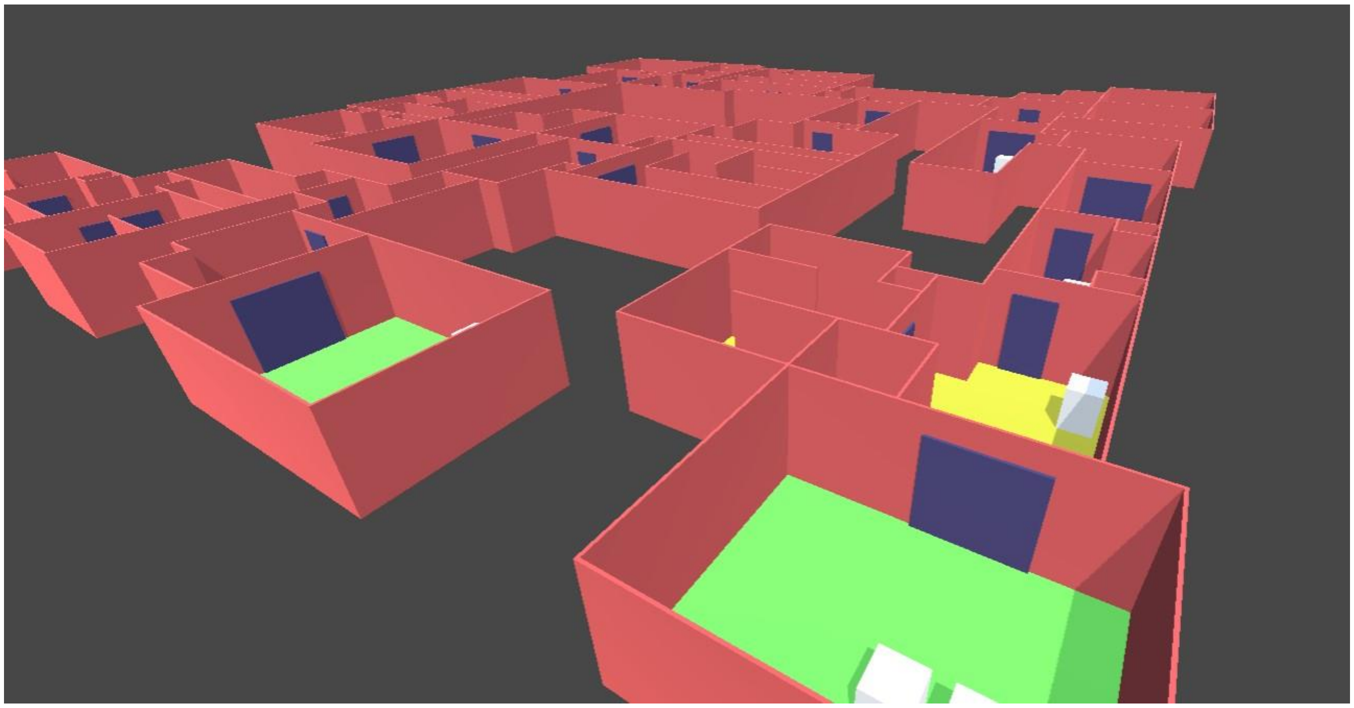


6. Let's also add bigger doors to our dungeon. Go to RandomDungeon and set MaxConnectorLength to 2.



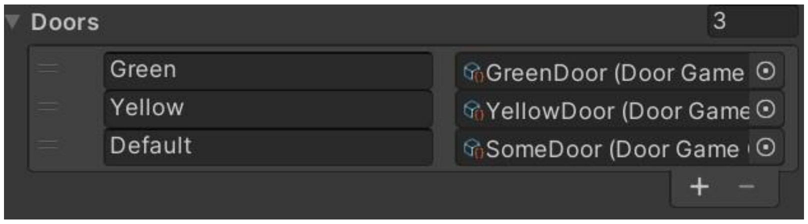
Min Connector Length	1
Max Connector Length	2

Now you can see that some bigger doors are generated as well.



Note: don't set MinimumConnectorLength to value bigger than 1, unless you are sure that all rooms can be connected with door longer than 1.

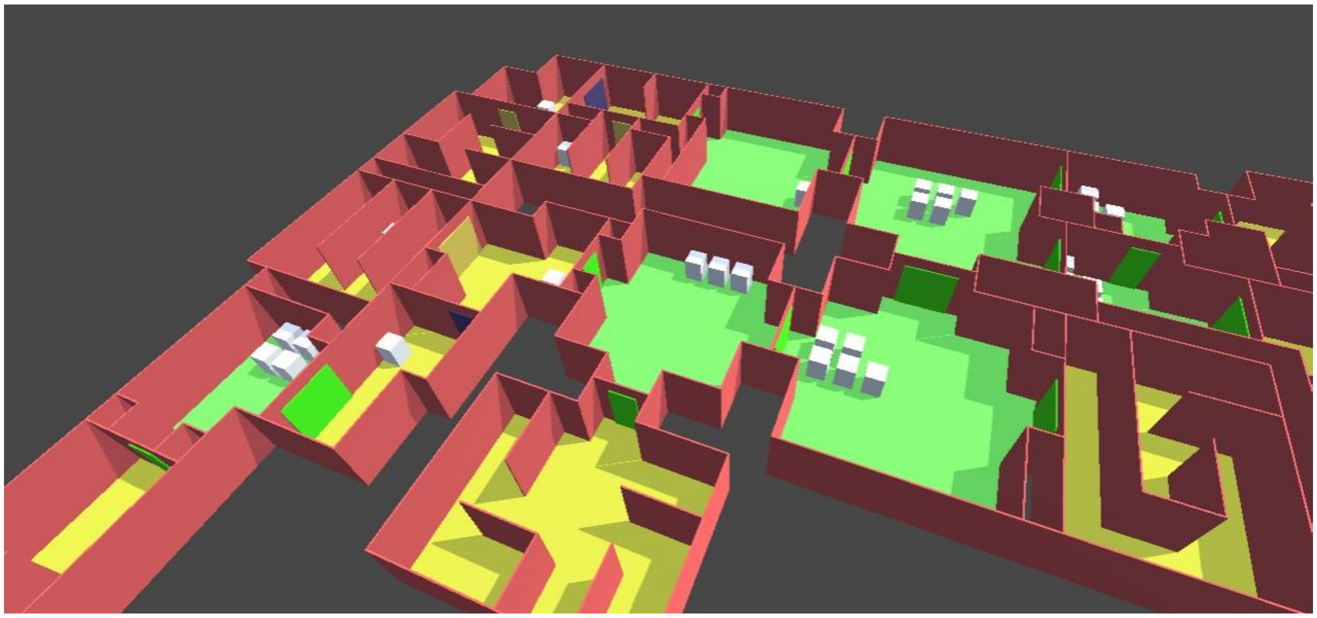
7. Let's add more door types



Doors 3

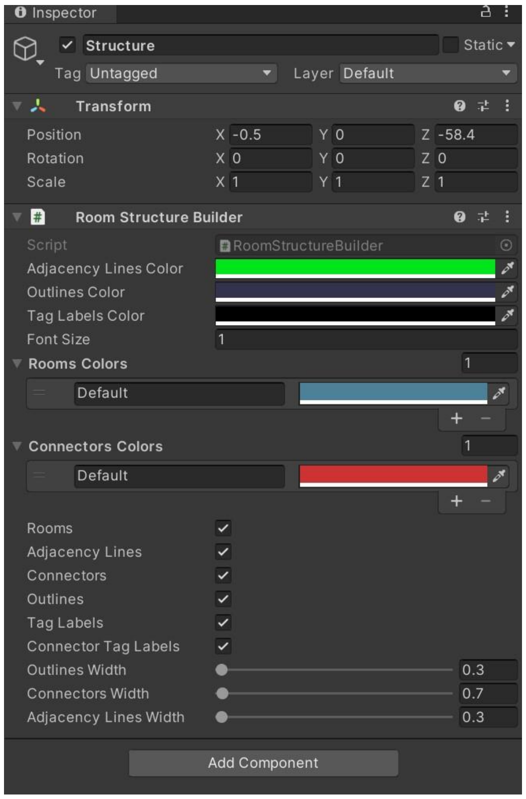
Green	GreenDoor (Door Game
Yellow	YellowDoor (Door Game
Default	SomeDoor (Door Game

Now all doors near Green and Yellow rooms will be selected from this list.
Doors between Green and Yellow rooms will be selected according to doors order(higher - higher priority).



Dispaying layout structure

1. Add empty GameObject and add RoomStructureBuilder(available in PDG->DungeonBuilders menu) component.



Inspector

Structure ✓ Static

Tag Untagged Layer Default

Transform

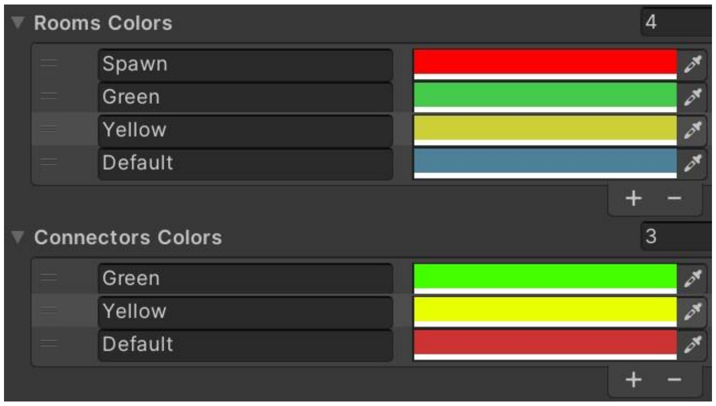
Position	X -0.5	Y 0	Z -58.4
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Room Structure Builder

Script	RoomStructureBuilder
Adjacency Lines Color	🟩
Outlines Color	⬛
Tag Labels Color	⬛
Font Size	1
Rooms Colors	1
Default	🟦
Connectors Colors	1
Default	🟥
Rooms	✓
Adjacency Lines	✓
Connectors	✓
Outlines	✓
Tag Labels	✓
Connector Tag Labels	✓
Outlines Width	0.3
Connectors Width	0.7
Adjacency Lines Width	0.3

Add Component

2. Add colors for differnt room and doors tags.



Rooms Colors 4

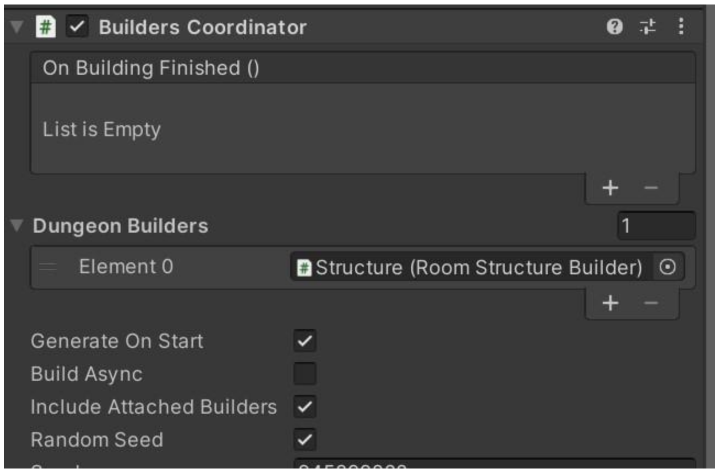
Spawn	🟥
Green	🟩
Yellow	🟨
Default	🟦

Connectors Colors 3

Green	🟩
Yellow	🟨
Default	🟥

Note: higher placement in list = higher priority.

3. Go to main generator object, and add reference to this new GameObject on BuildersCoordinator component.



Builders Coordinator

On Building Finished ()

List is Empty

Dungeon Builders 1

Element 0 Structure (Room Structure Builder)

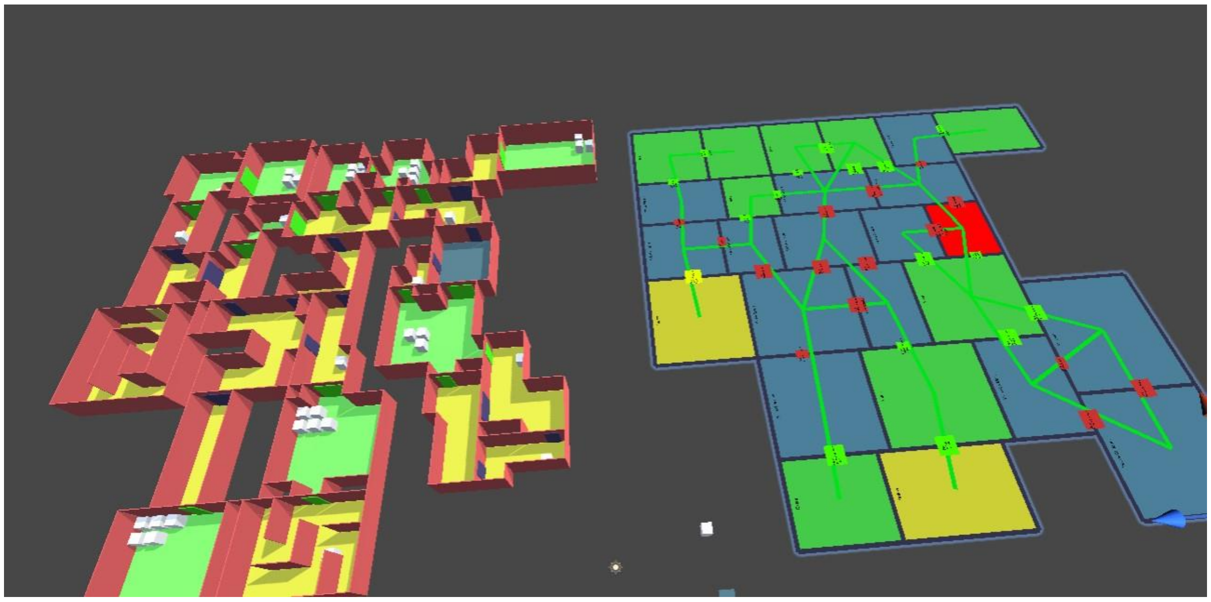
Generate On Start ✓

Build Async

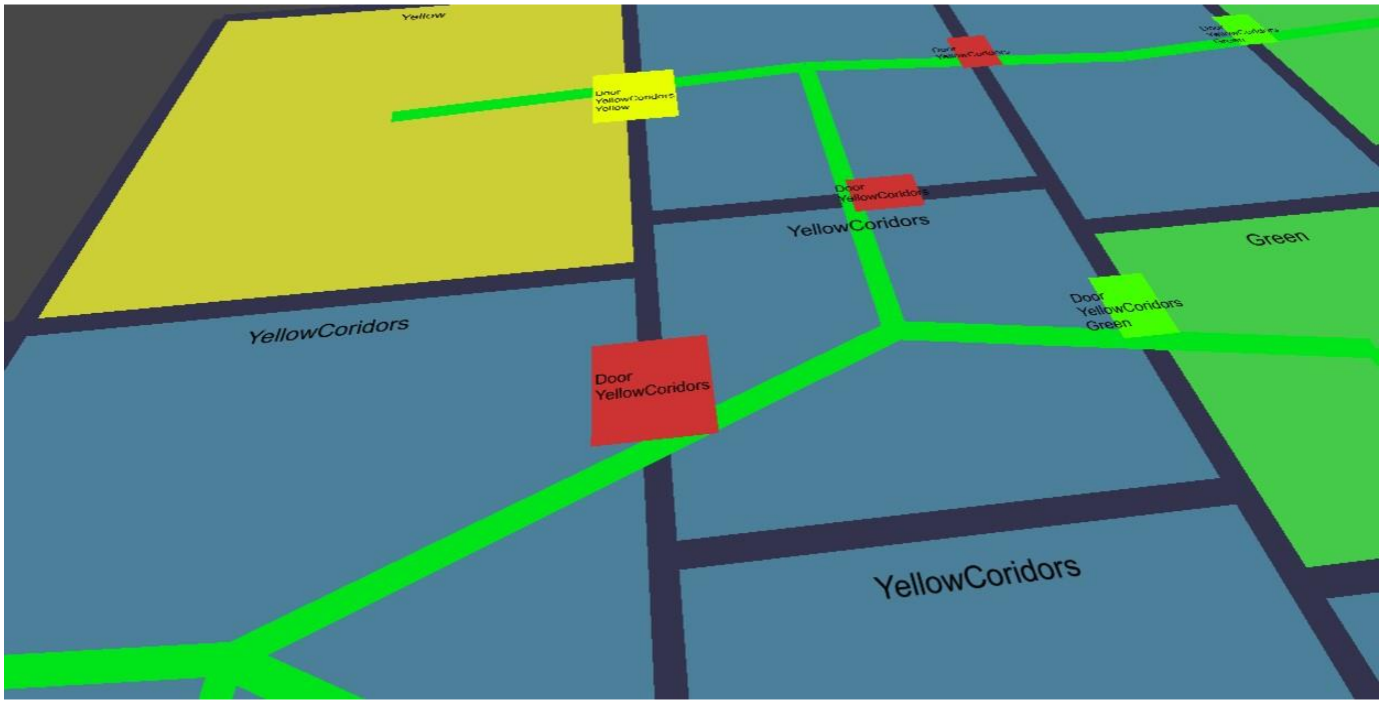
Include Attached Builders ✓

Random Seed ✓

4. When you press Generate button - you will see that structure was built as well.



If we inspect generated structure closer - we can see that each room has tag, and some connectors have multiple tags.



These tags are used by TiledGameObjectDungeonBuilder when it is building the dungeon.
Rooms can have multiple tags as well, but RandomDungeon makes sure to remove all unnecessary tags.

RoomStructureBuilder is meant to be used while prototyping the dungeon structure, before setting up building details of actual rooms.

This concludes the quickstart guide.
To learn more details about Random Dungeon, read Random Dungeon docs.
To learn more about other Layout Generators, you can read AdjacentRules Dungeon, Route Dungeon or MultiStageGenerator docs.
To learn more about dungeon building(how to set up custom walls) and workflow with Unity Tilemaps, visit the Tile Based Builders docs.