Tile Based Builders

Tile Based Builders are builders that use similar approach for dungeon generation and inner tilemaps for wall building and objects placement. Currently there are 2 such builders:

TiledGameObjectDungeonBuilder - creates dungeon from GameObjects
TilemapDungeonBuilder - builds dungeon on Unity's Tilemaps.

This page contains definiton of their common parameters, and setup guide for TIlemapDungeonBuilder.

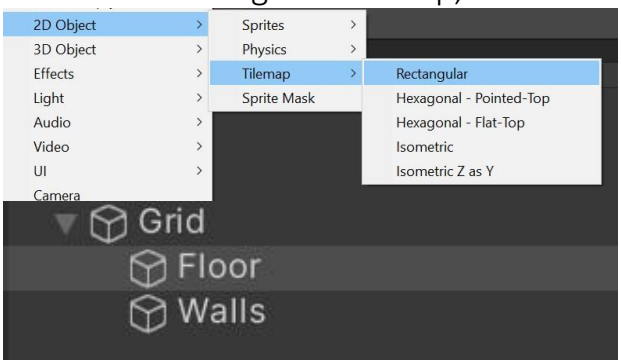
Getting started

TiledGameObjectDungeonBuilder

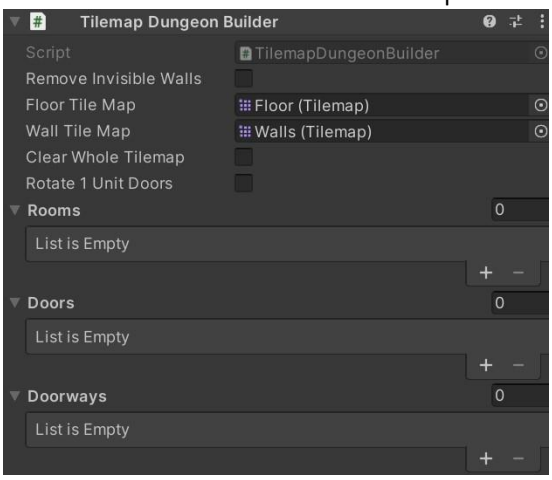
Guide available in the first section of Quick Start docs.

TilemapDungeonBuilder

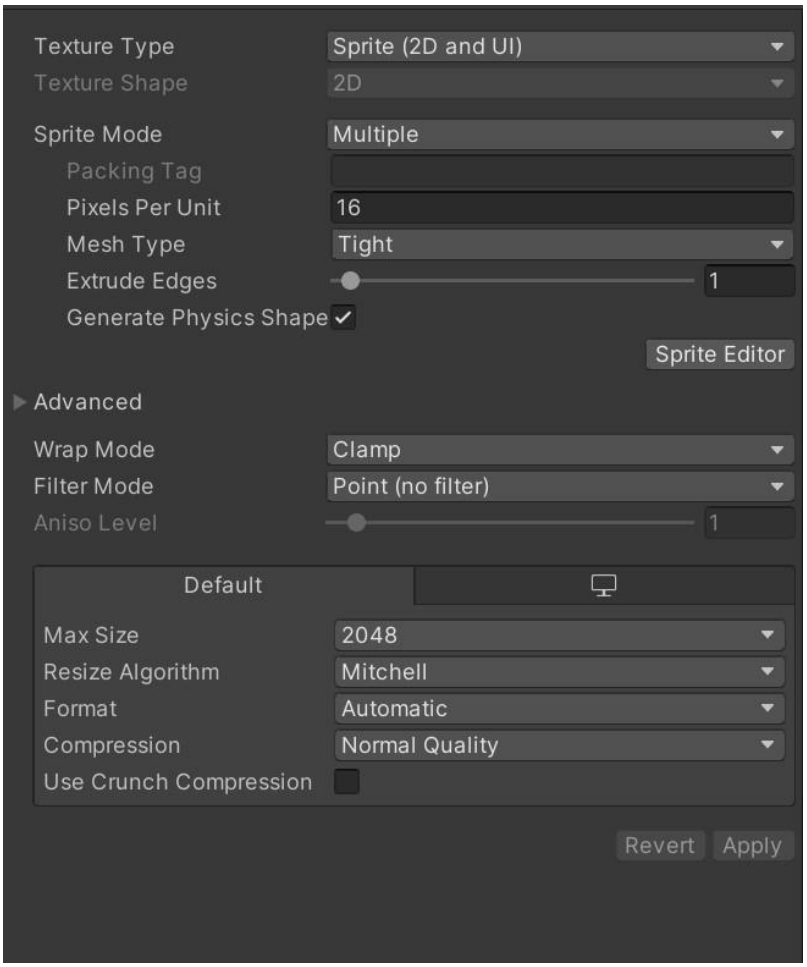
1. Create Rectangular Tilemap, with one layer for floor, and one layer for walls.



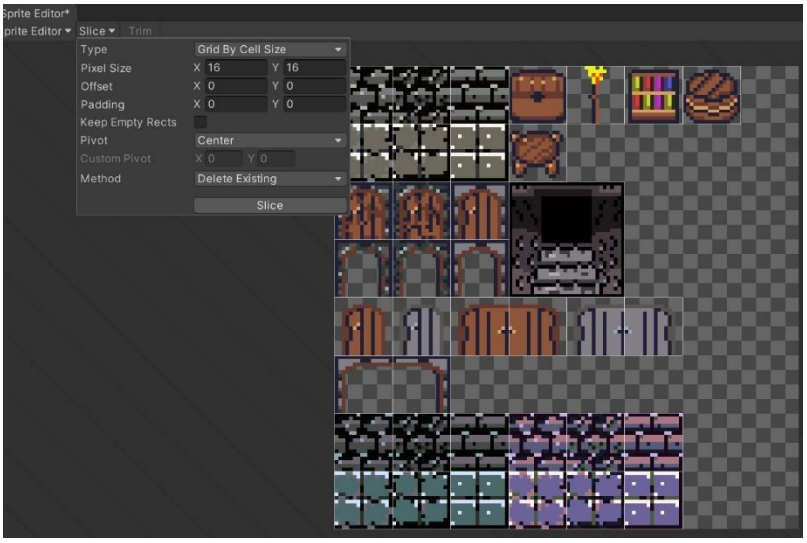
2. Create a new GameObject and add RandonDungeon, TilemapDungeonBuilder and BuildersCoordinator components. Set references to floor and wall layers.



3. Import tiles as spirtes, with Mode set to Multiple, set pixels per unit to match 1 tile.

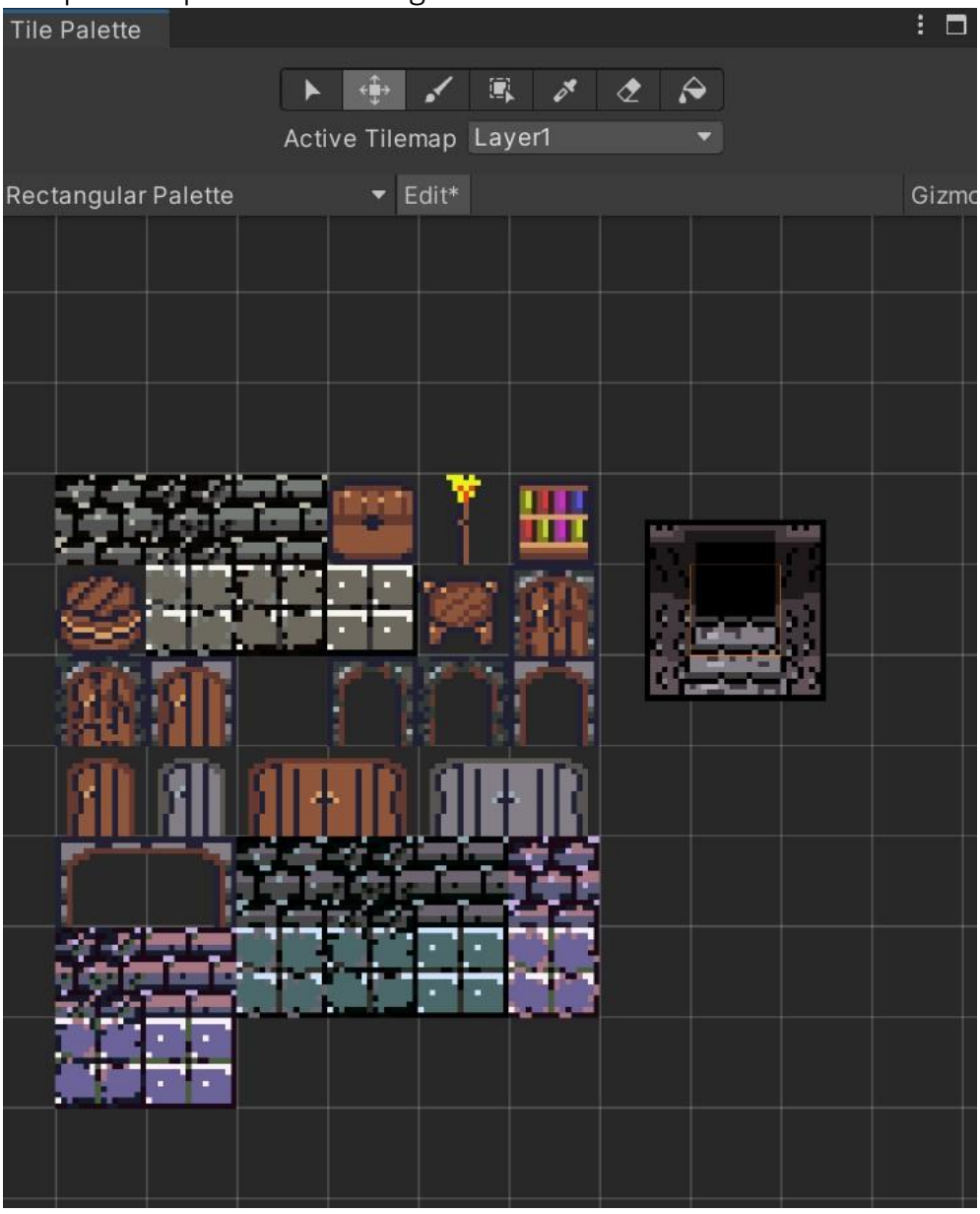


4. Slice tiles:

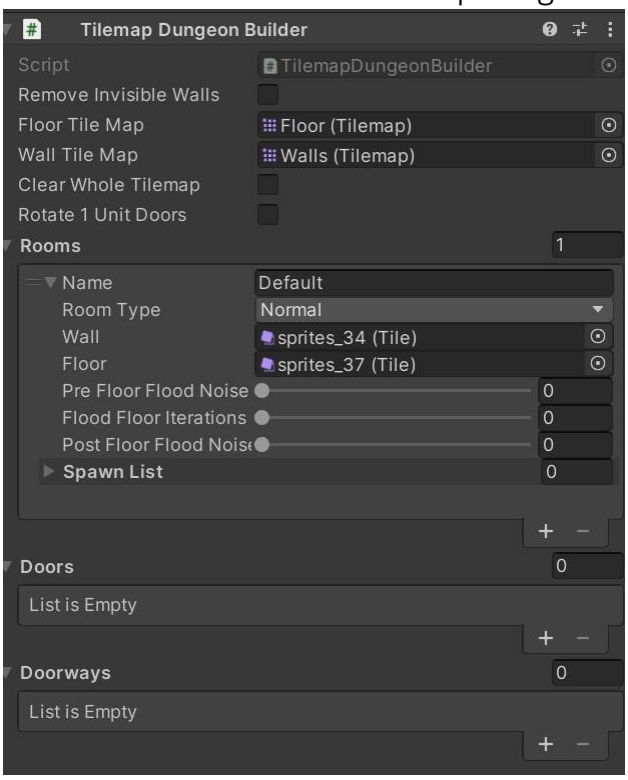


5. Create tile pallette:

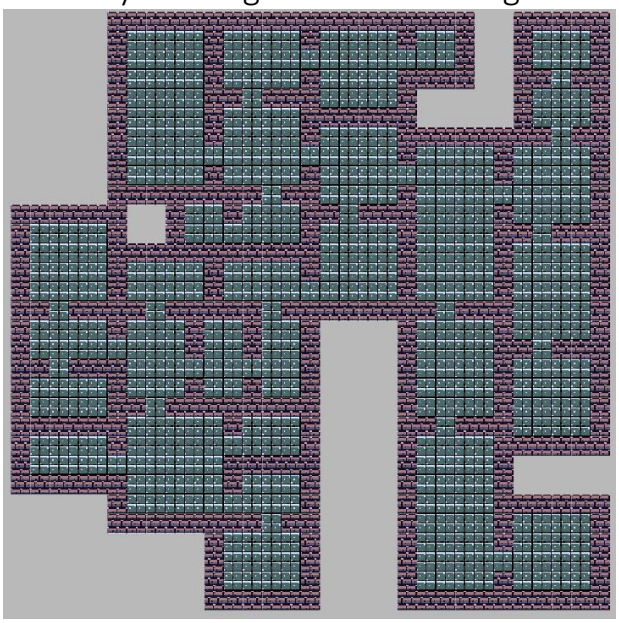
5. Open tile palette and drag tiles there:



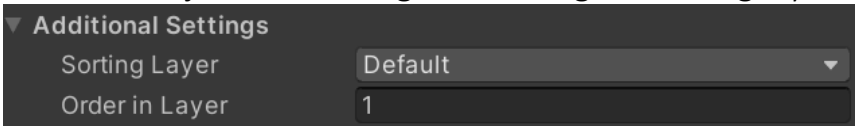
6. Add Default room to TilemapDungeon builder, set floor and walls:



7. Now you can generate the dungeon:

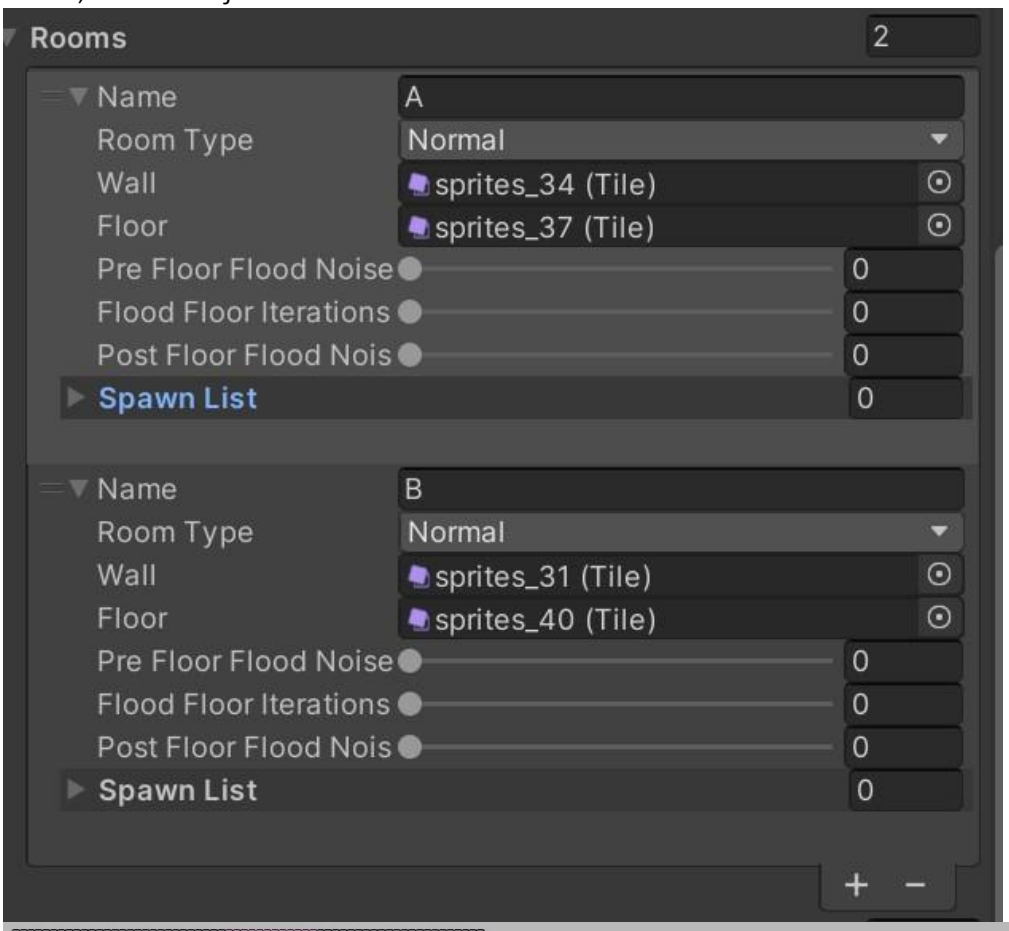


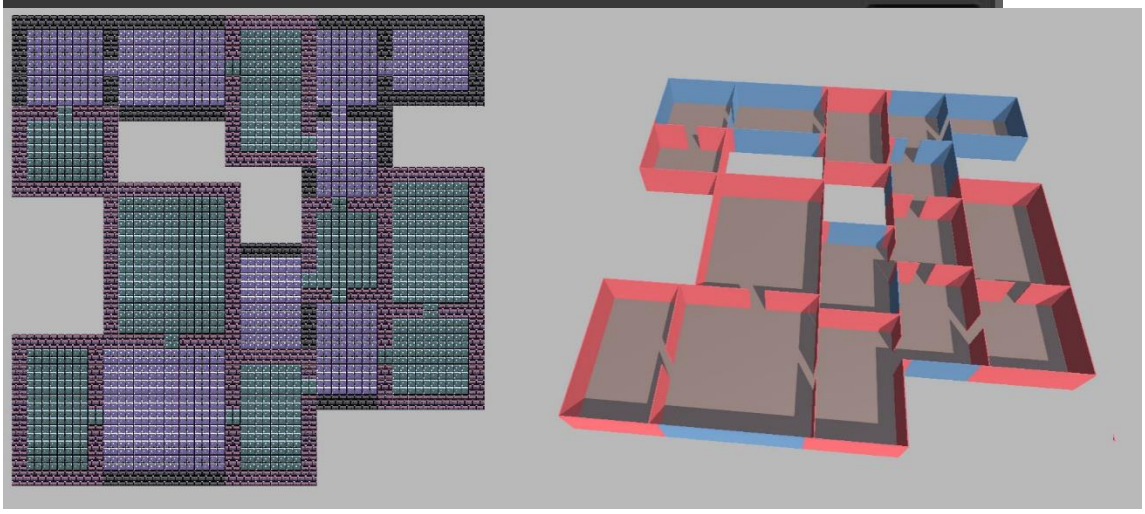
Note: for objects don't forget to set higher sorting layer than tilemaps:



Room Setup

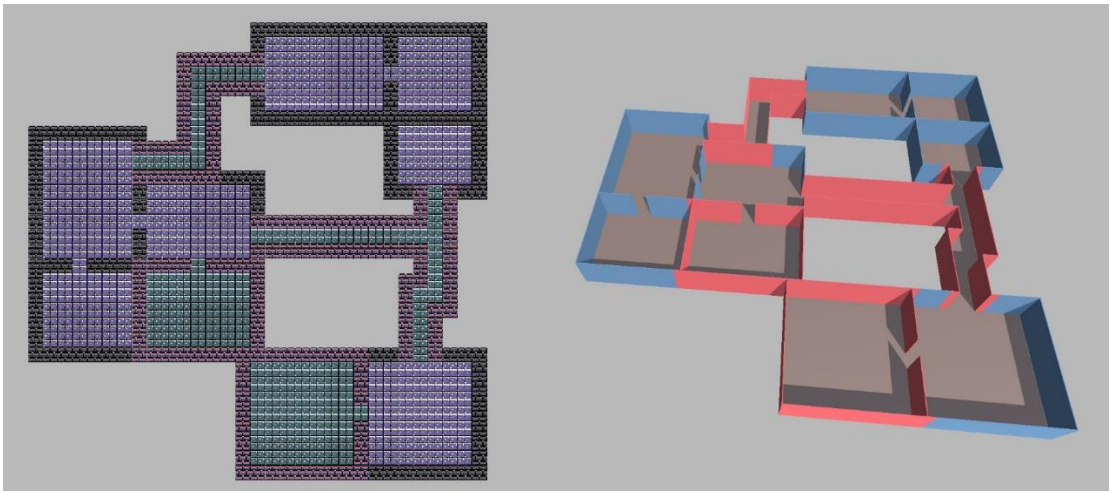
Room Setup determines how room must be built, and which objects should be spawned. Priority of building - from top to bottom, higher room setup overrides walls, when adjacent to other rooms.



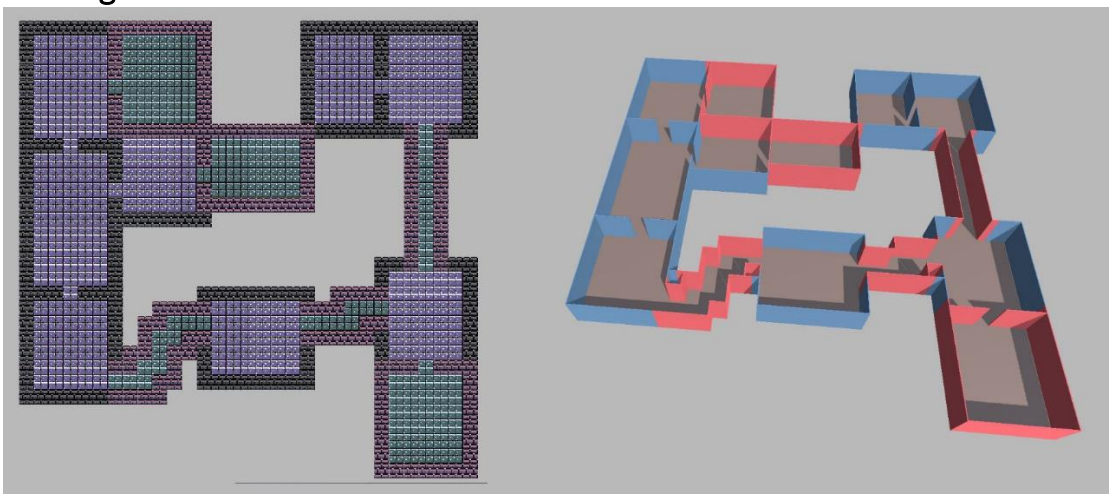


Room types

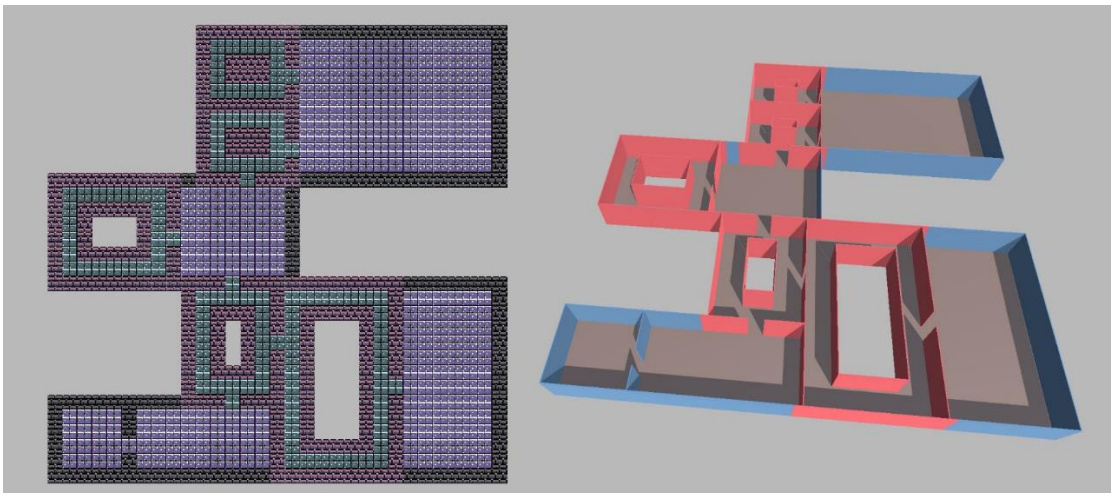
Corridors



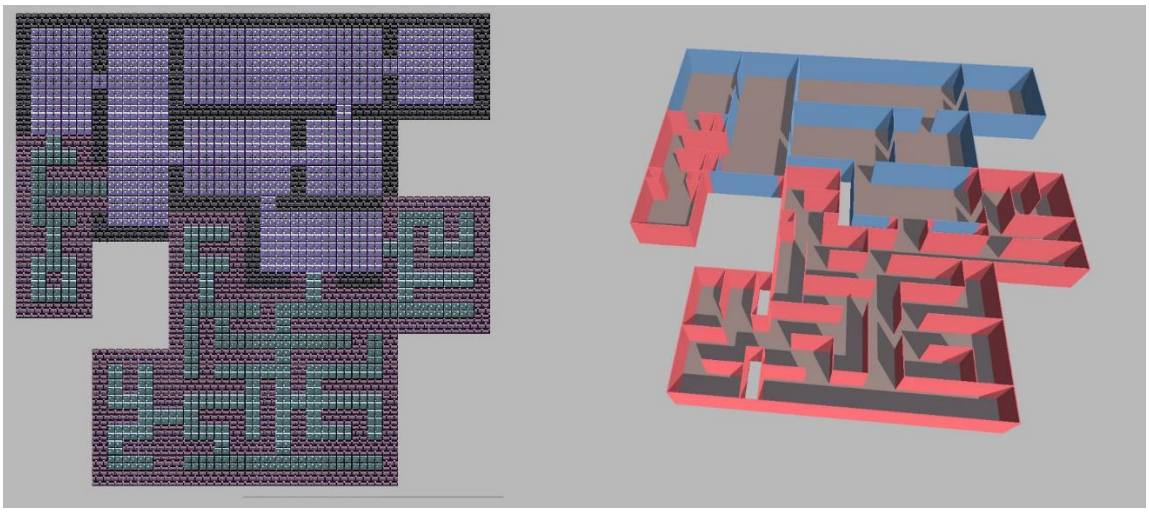
Turning Corridors



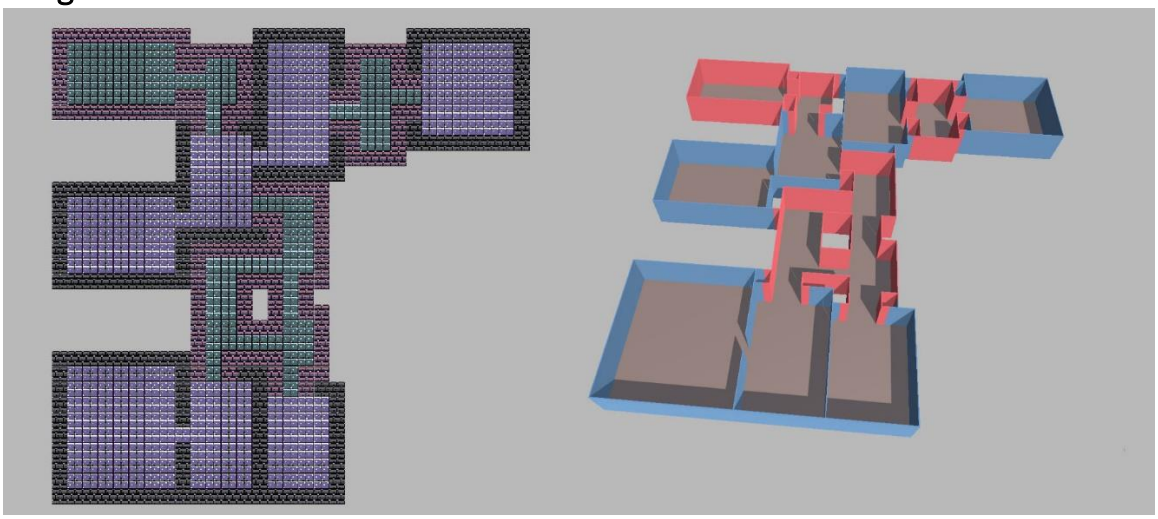
Border Path



Maze



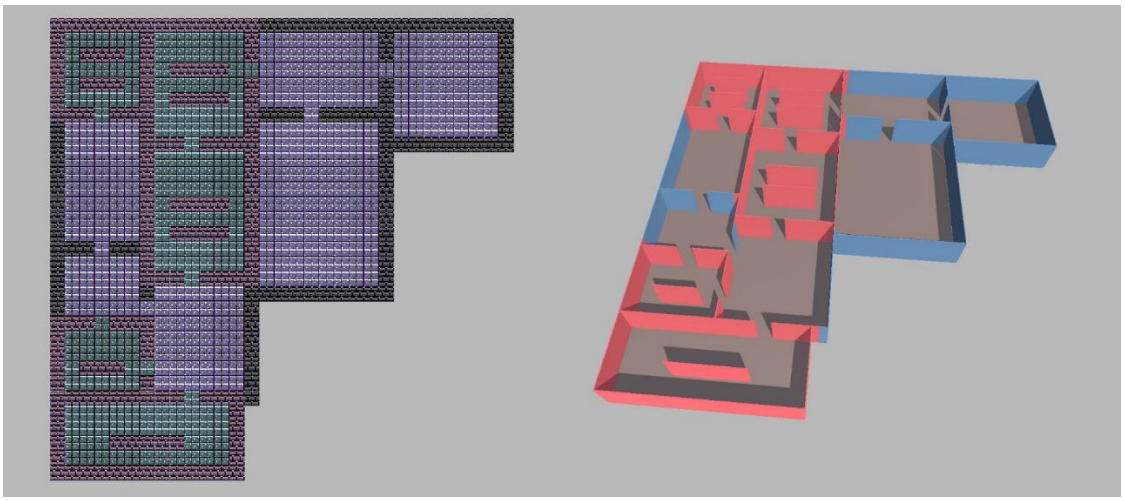
Longer Entrances



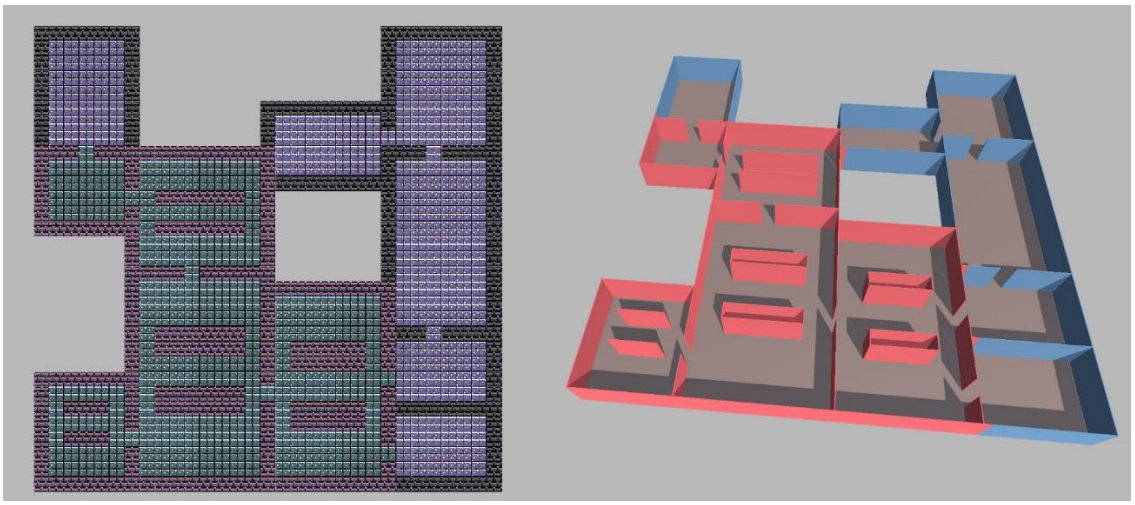
Custom – draws custom “picture” inside room.

For mask:

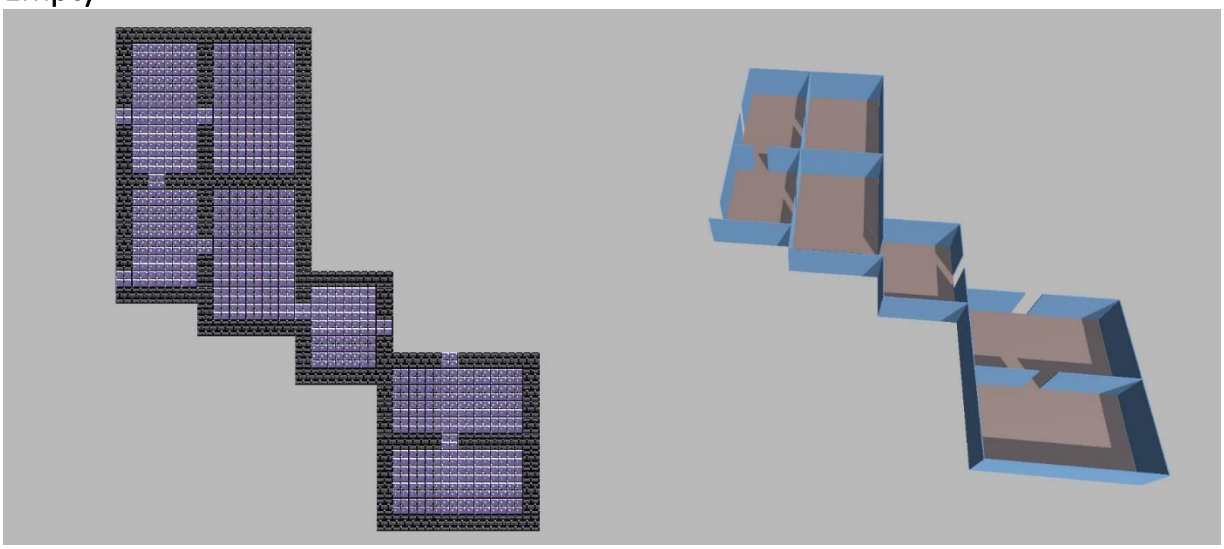
Without Rescale



With Rescale

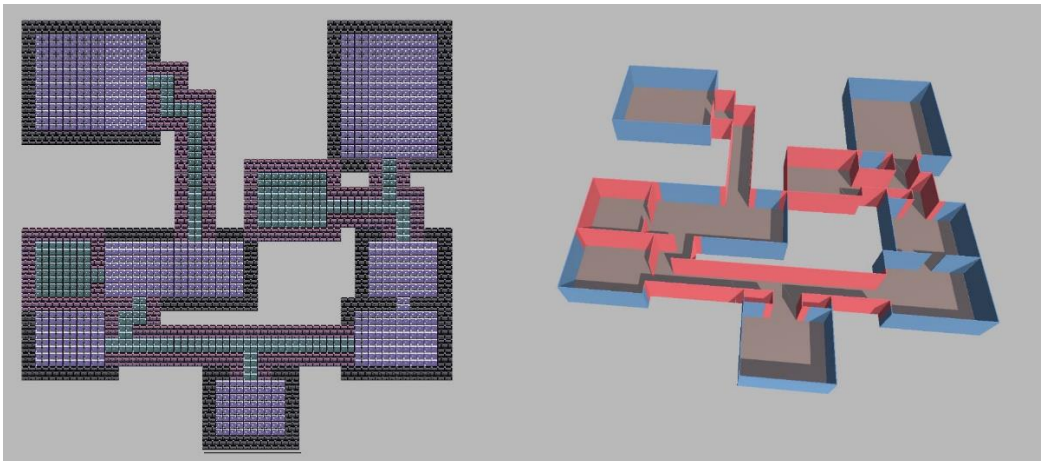


Empty

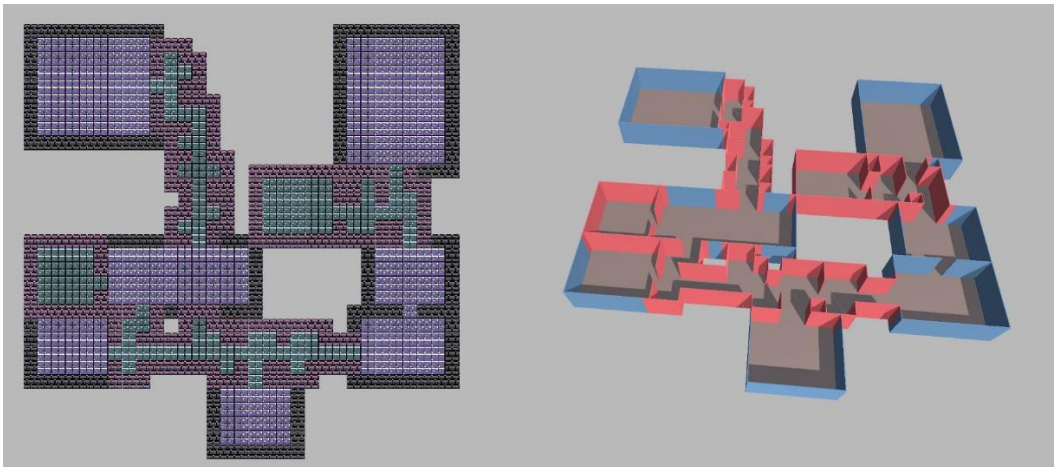


Room postprocessing

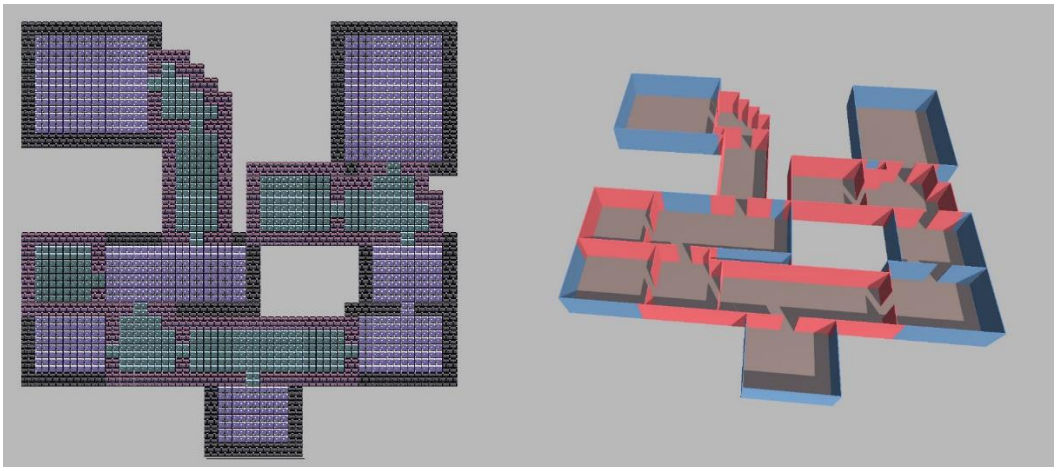
Initial room



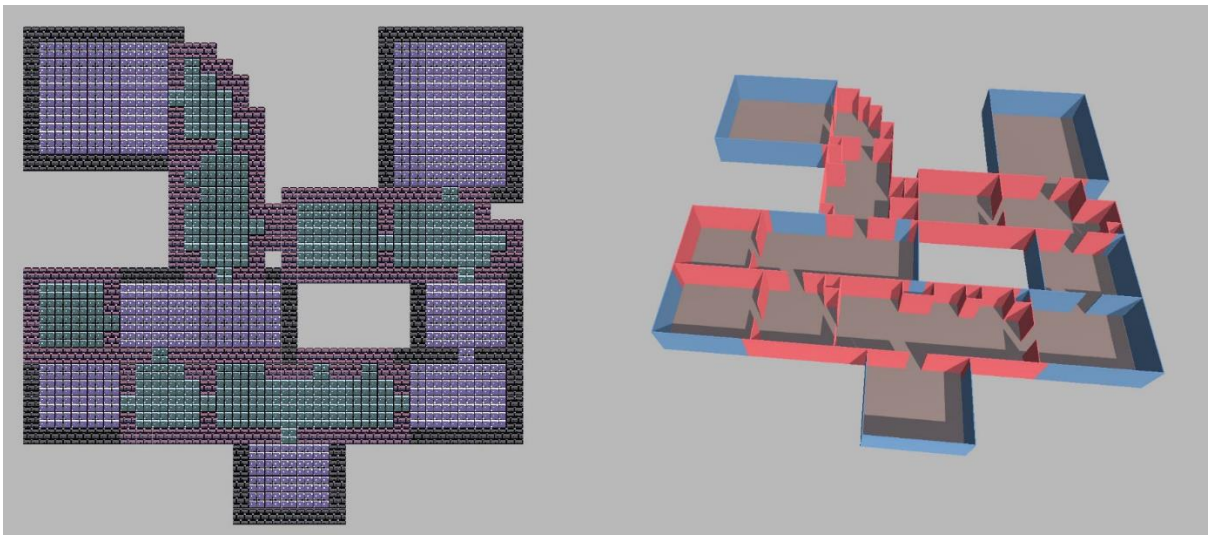
Pre Floor Flood Noise = 0.3, this noise applies after room was generated



Flood Floor Iteraions = 1, applies after room was generated, and after previous nosie(if it was > 0)



Flood Floor Iteraions = 1, Post Floor Flood Noise = 0.4

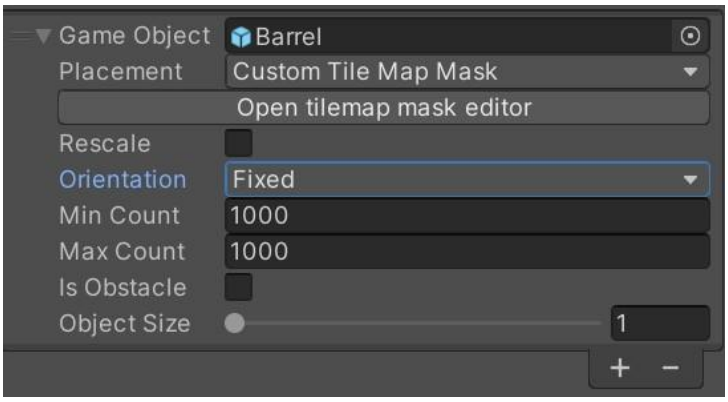


Layout conversion differences

TilemapDungeonBuilder and TiledGameObjectDungeonBuilder convert layout into dungeon differently. Prior uses standard tilemap approach, the latter inserts walls between object tiles.

Object placement

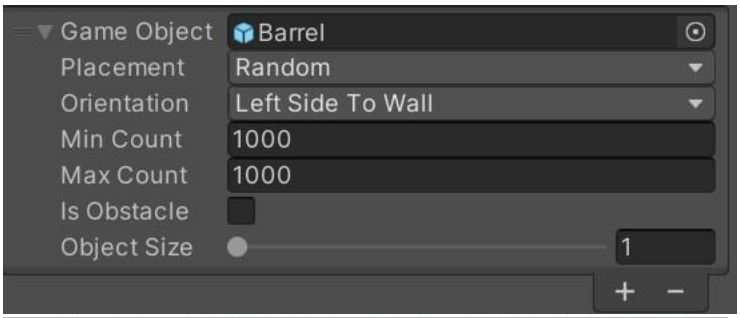
To spawn objects you need to add elements to SpawnList.

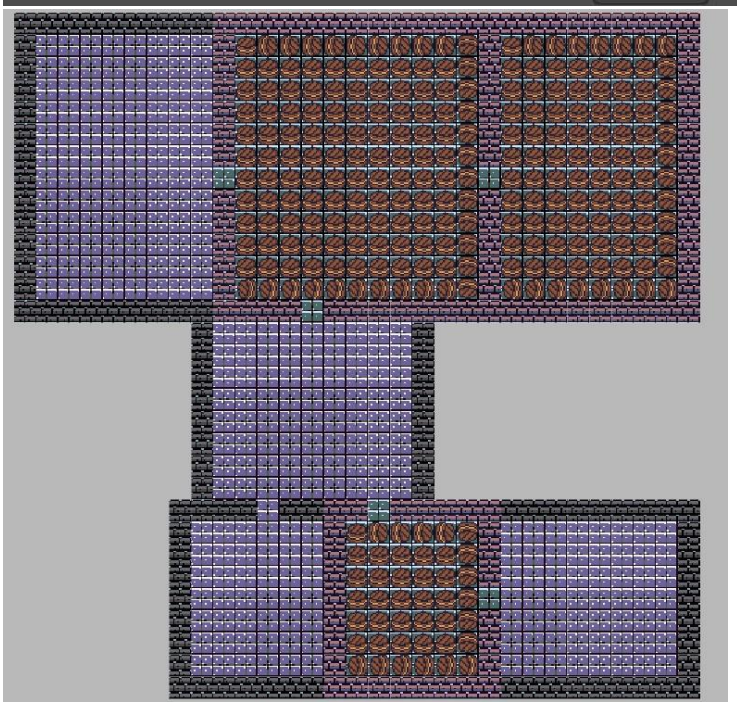


Parameters:

Placement - determines where object can be placed.

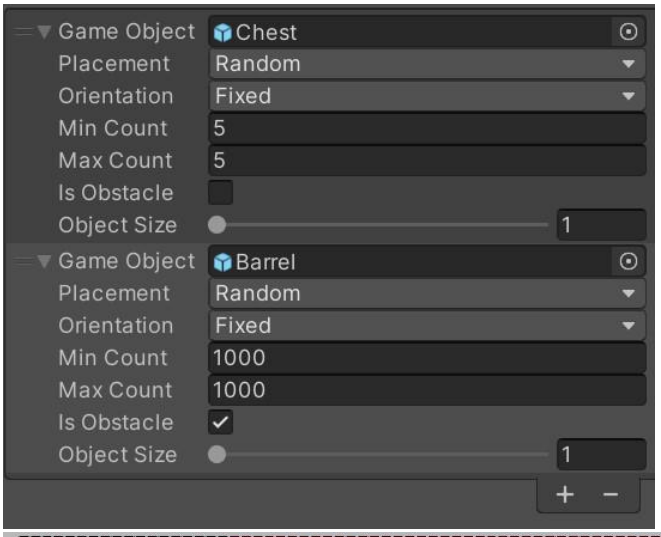
Orientation - determines object rotation.

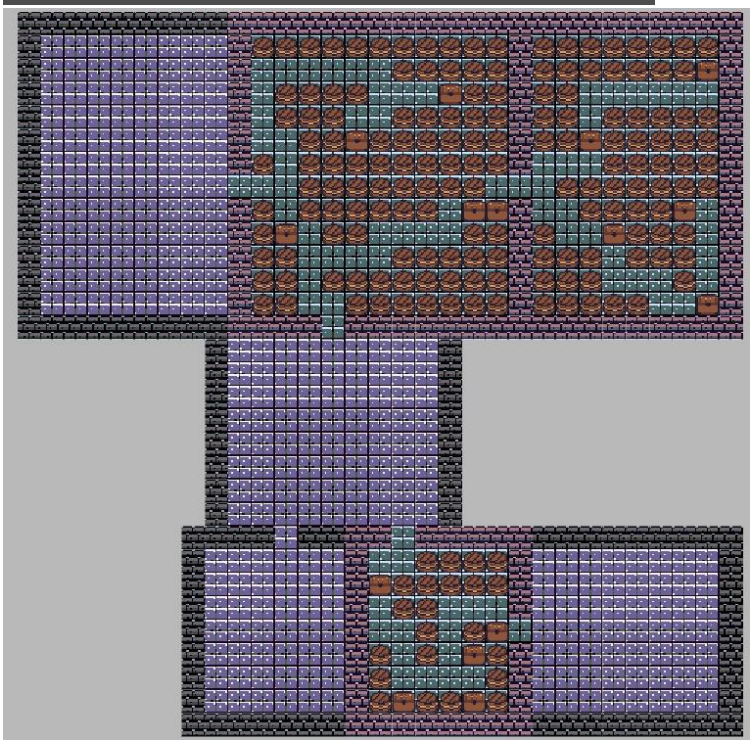




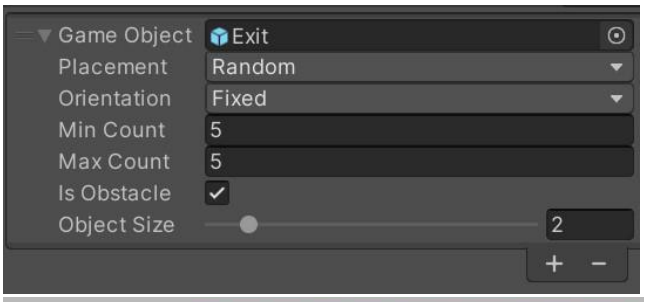
MinCount, MaxCount - how many objects must be placed. If there is not enough space(among suitable tiles according to Placement) in the room - as much as possible will be spawned.

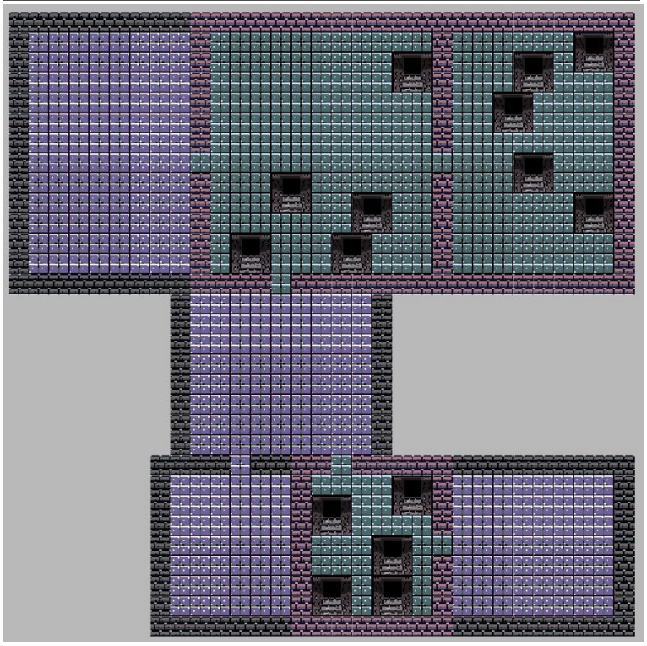
IsObstacle - if set, object will be considered obstacle, and will not be placed on any tile, that can block the road between connectors, and to other Non-obstacle objects.
For example, if we place 5 non-obstacle chests, and a lot of obstacle barrels - barrels will not block the road.





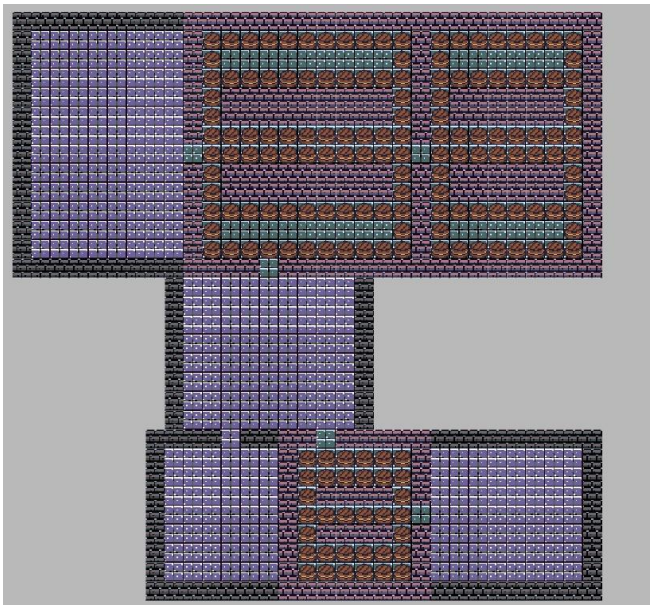
Object Size - size of the object.



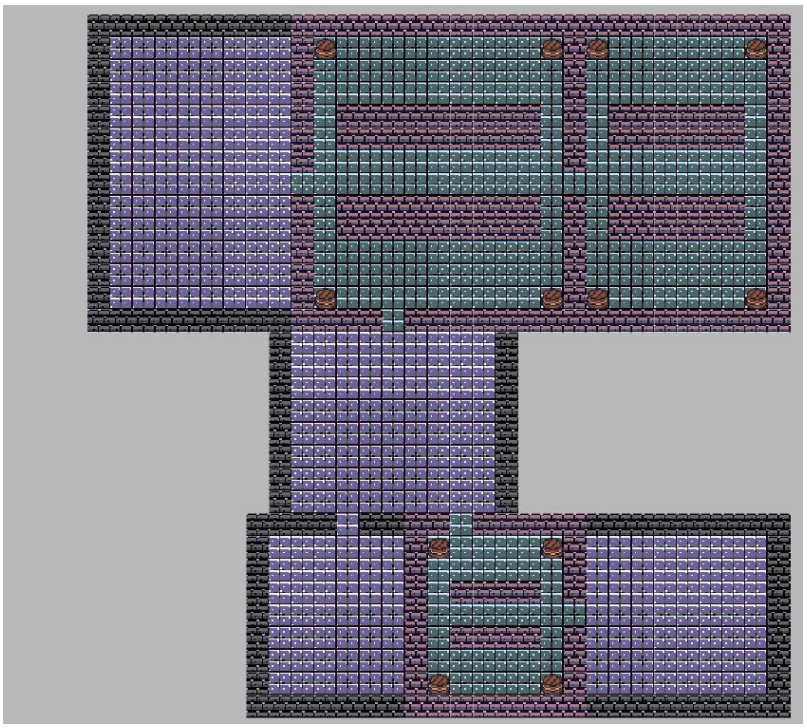


Placement types

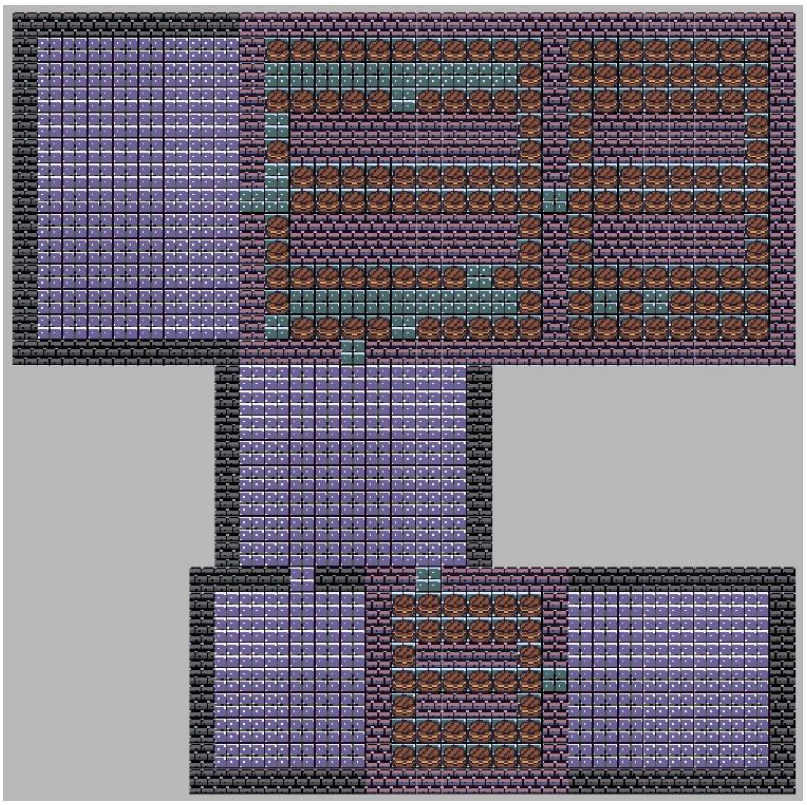
Near wall



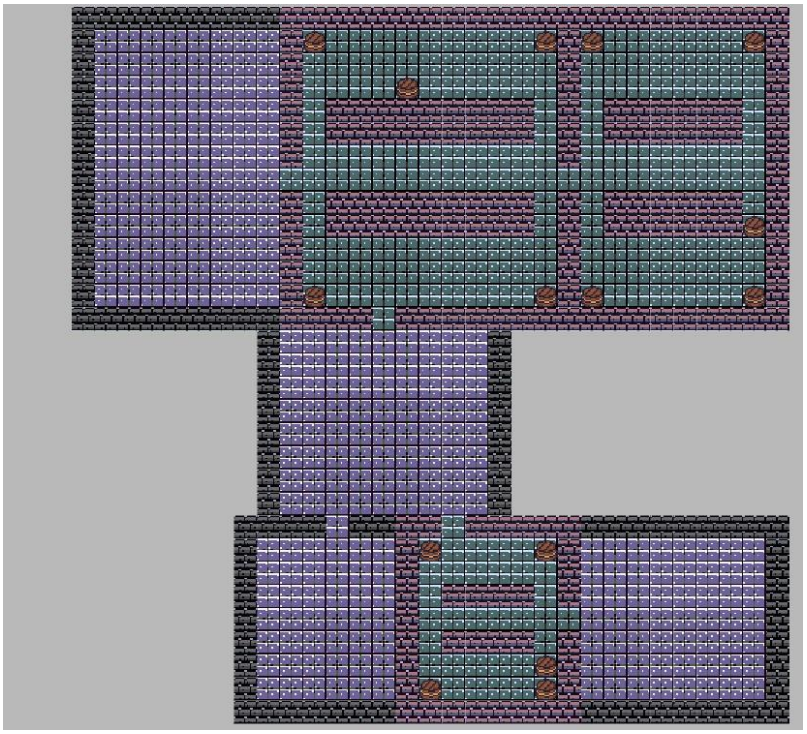
Near corner



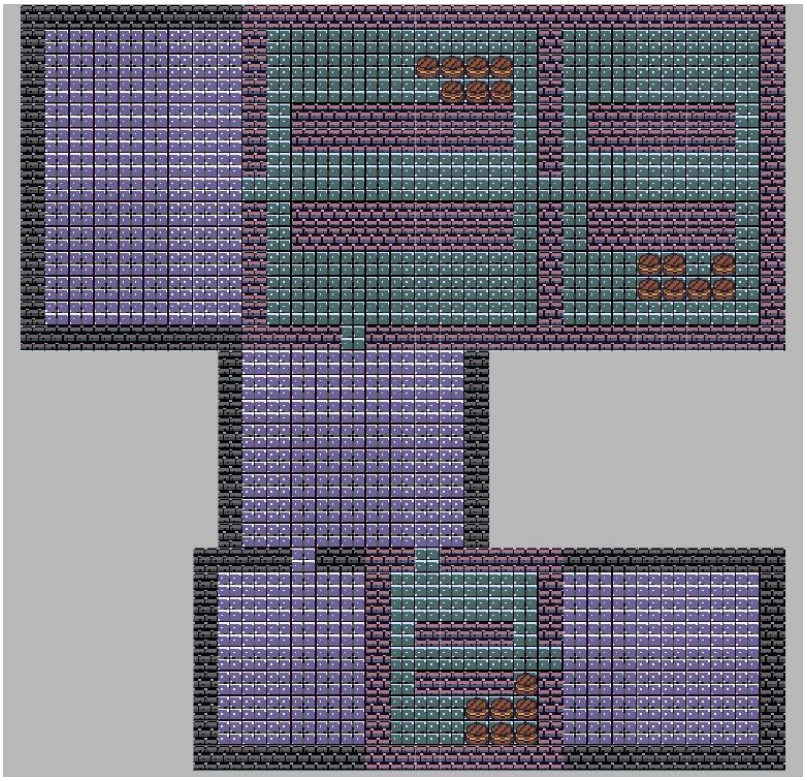
Near wall or random



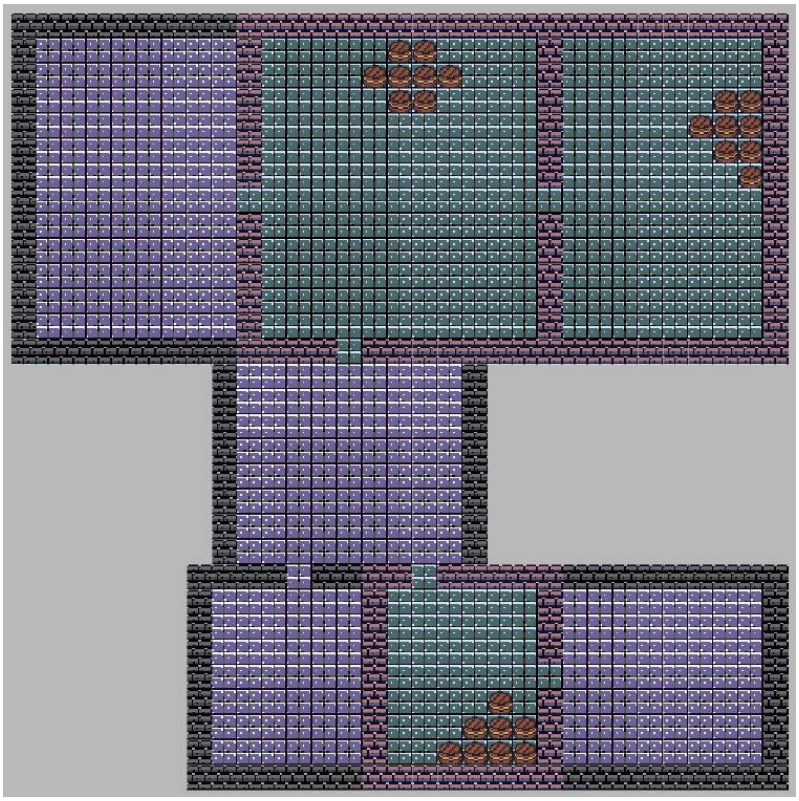
Near corner or random



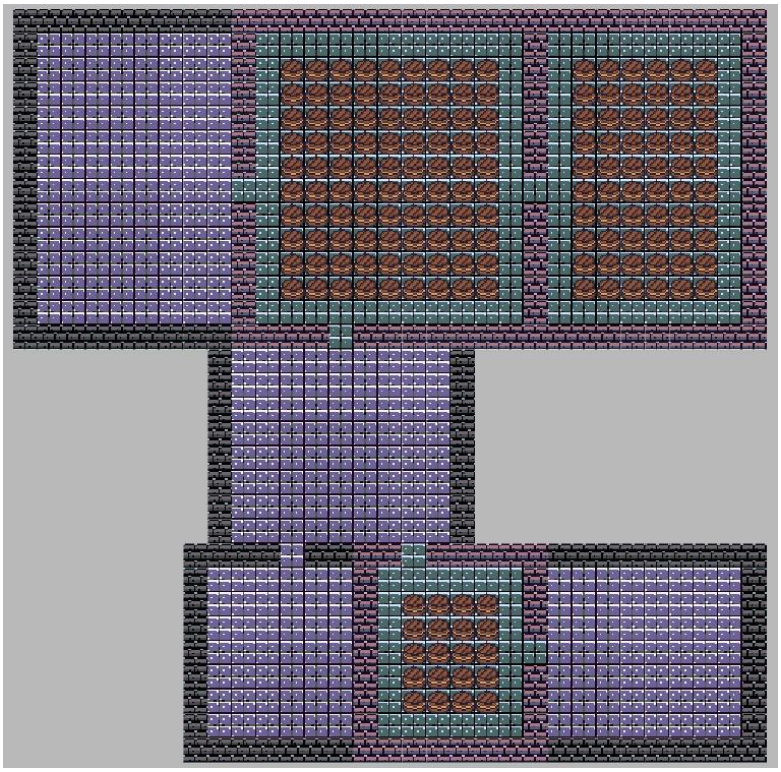
Cloud



Circle

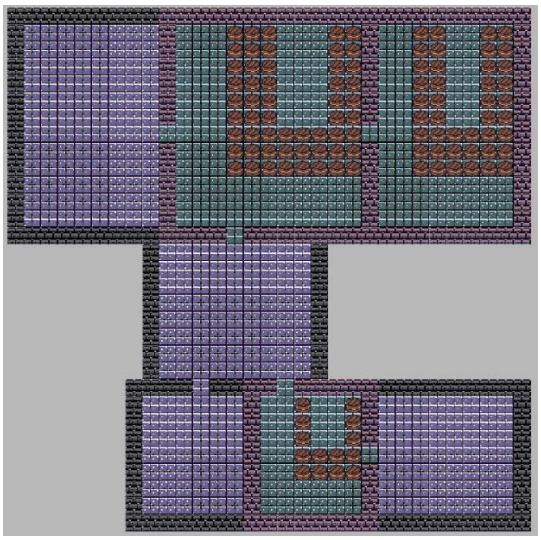


Not near wall

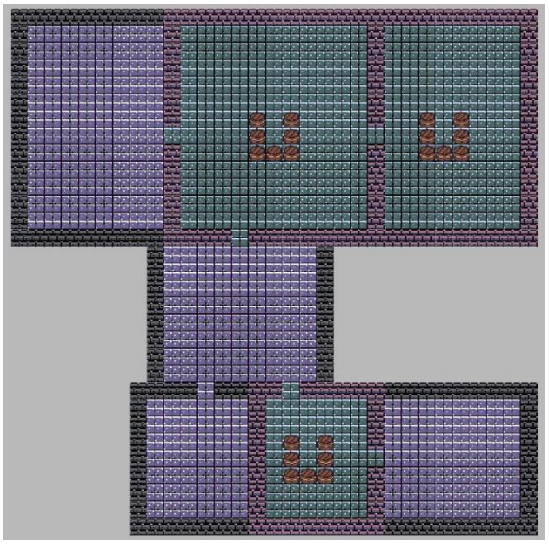


Custom tilemap mask

With rescale

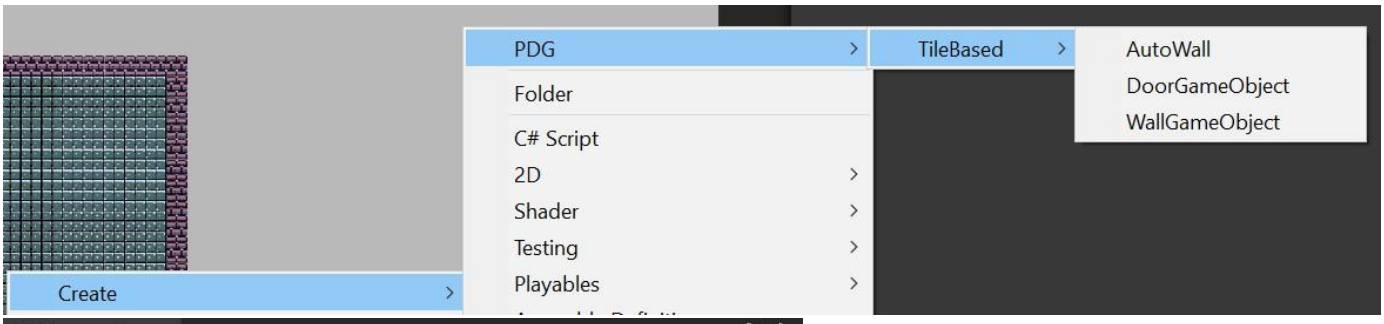


Without rescale

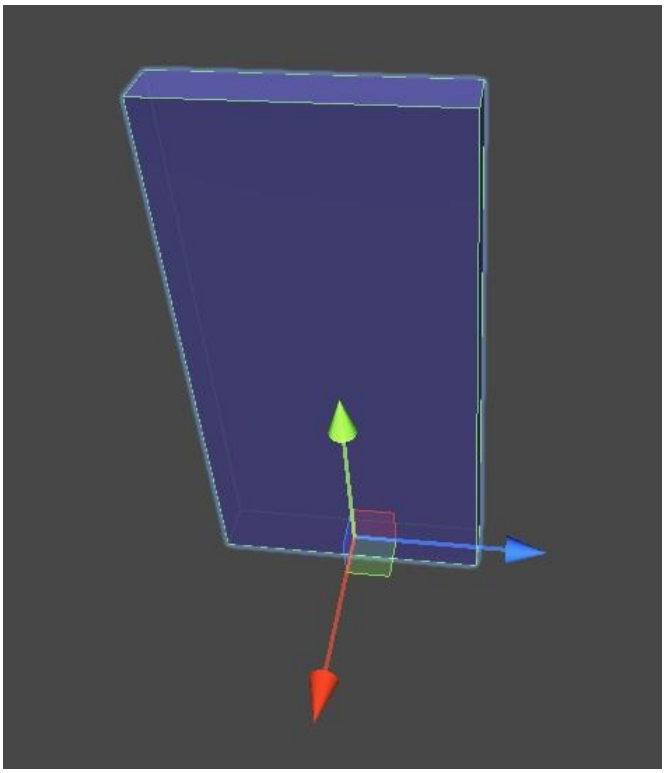


Custom walls, doors, and doorways

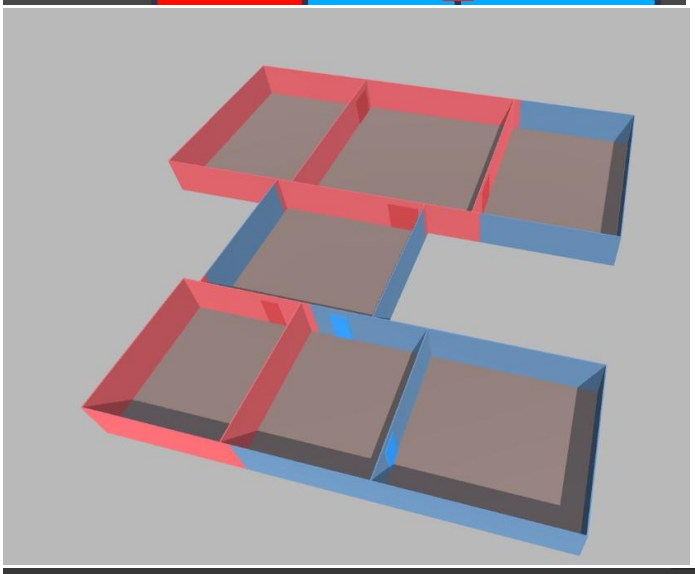
TiledGameObjectDungeonBuilder supports custom doors and custom walls. To add a custom door create DoorGameObject, and set prefabs for different door lengths.

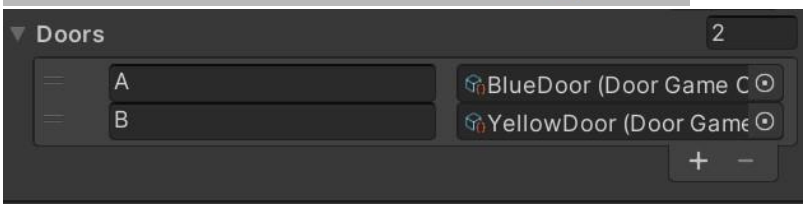


Door prefab should face X axis, and be aligned by it's bottom center.

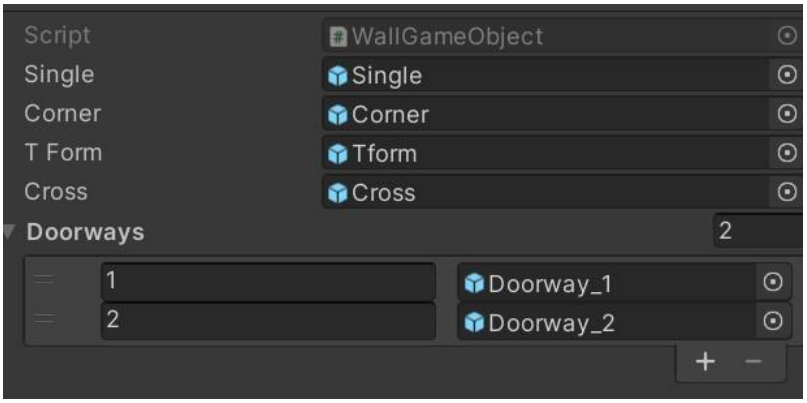


If connector has more than 1 tag - higher door will be used with higher priority. For example, let's take a look at a dungeon generated by RandomDungeon - each connector has tags of adjacent rooms, and there are 2 types of rooms. Therefore, on the border between them A room will be used.

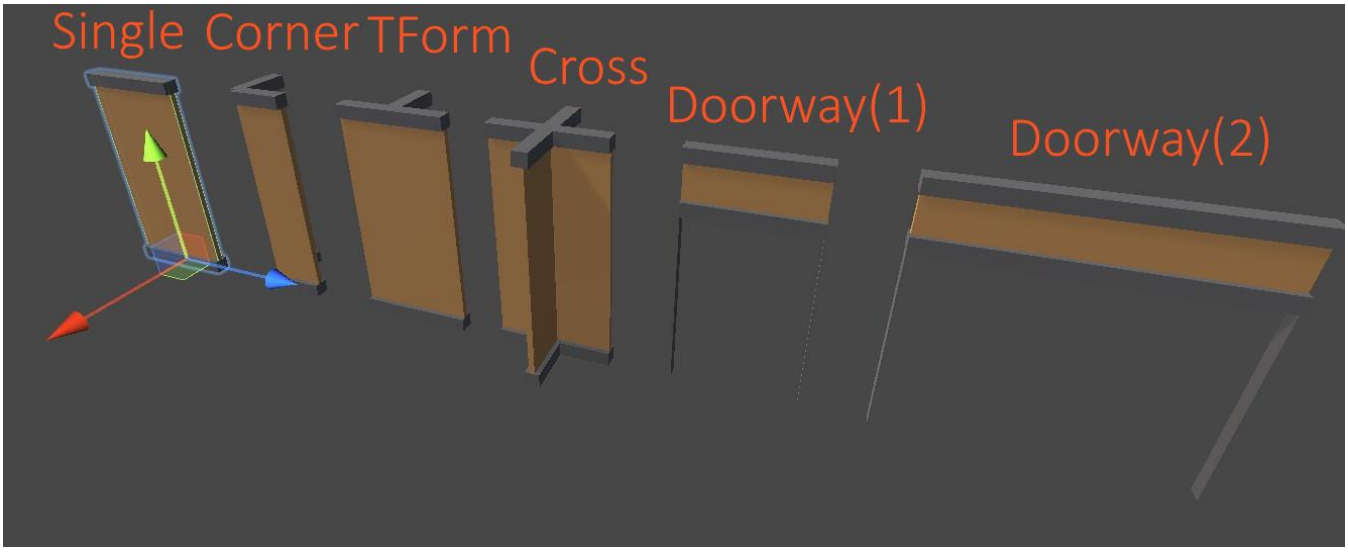




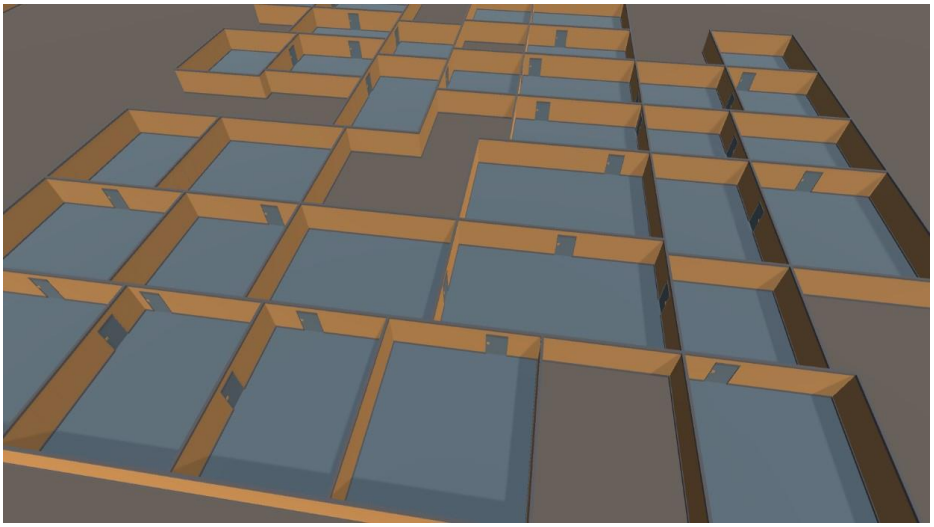
To create a custom wall - create **WallGameObject** or **AutoWall**.
For WallGameObject you need to specify certain GameObjects.



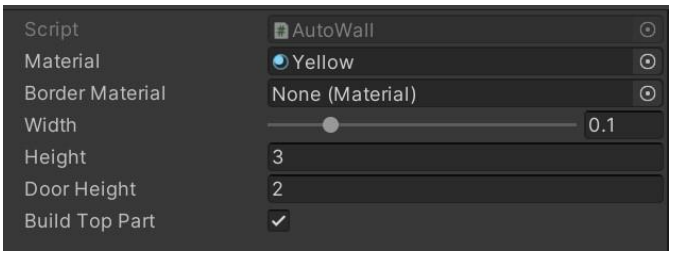
And they must be aligned like this:



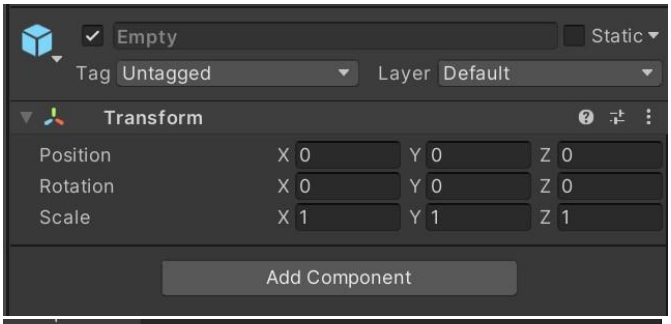
Result of using this wall:

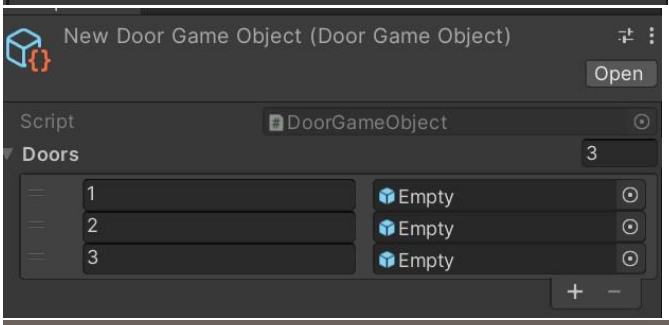


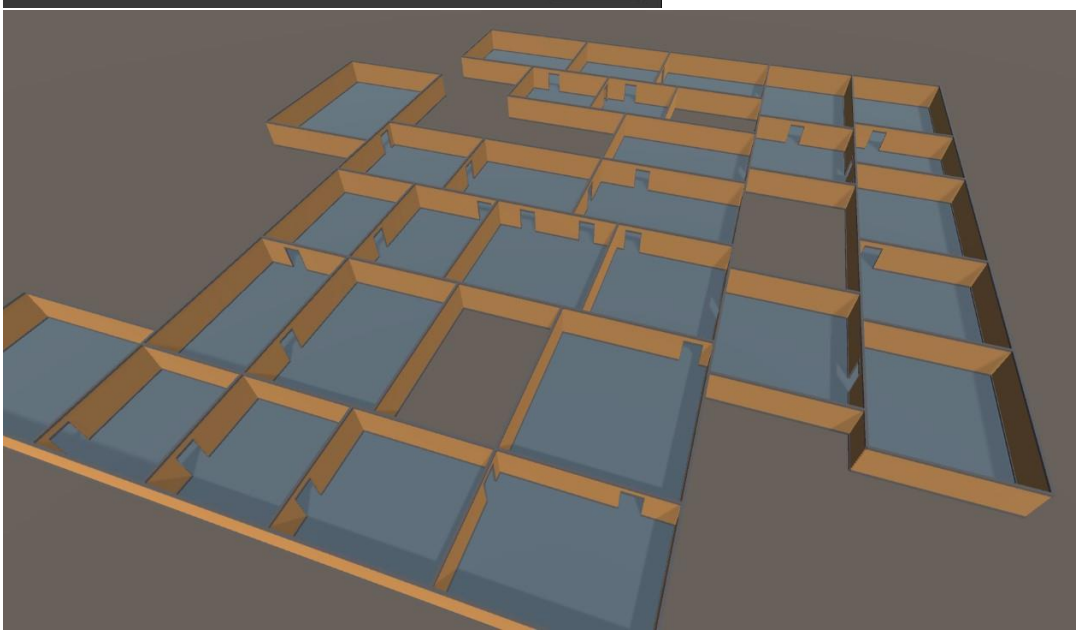
Autowall automatically generates wall with mesh and collider according to parameters:



Another important detail: doorway will be generated only when door was spawned. If you need doorway without a door - pass empty GameObject.







TilemapDungeonBuilder uses similar approach to work with doors, however walls can be specified only with single tile. To support doorways TilemapDungeonBuilders provides a doorways list, where each doorway for any room tag can be specified with DoorGameObject

Using pathfinding

To use pathfinding you need to manually call FindPath method from buildere component.

```
_dungeonBuilder = FindObjectOfType<TiledGameObjectDungeonBuilder>();
```

```
_path = _dungeonBuilder.FindPath(gameObject.transform.position, newDestination, blockedConnectors);
```

It will return a list of positions on the path.
First 2 arguments specify start and target position.
The third argument takes array of strings, they specify which doors whould not be used.
Similar method is available to TilemapDungeonBuilder

```
_tilemapBuilder.FindPath(new Vector2Int(5, 5), new Vector2Int(20, 25), "YellowDoor", "BlueDoor");
```

Note: pathfinding doesn't support dungeon modification. If you want to remove obstacle game objects - you can access Pathfinder object, and call removeObstacle method.

```
_dungeonBuilder.Pathfinder.RemoveObstacle(new RectShape());
```

Argument specifies which tiles object occupies on tilemap.