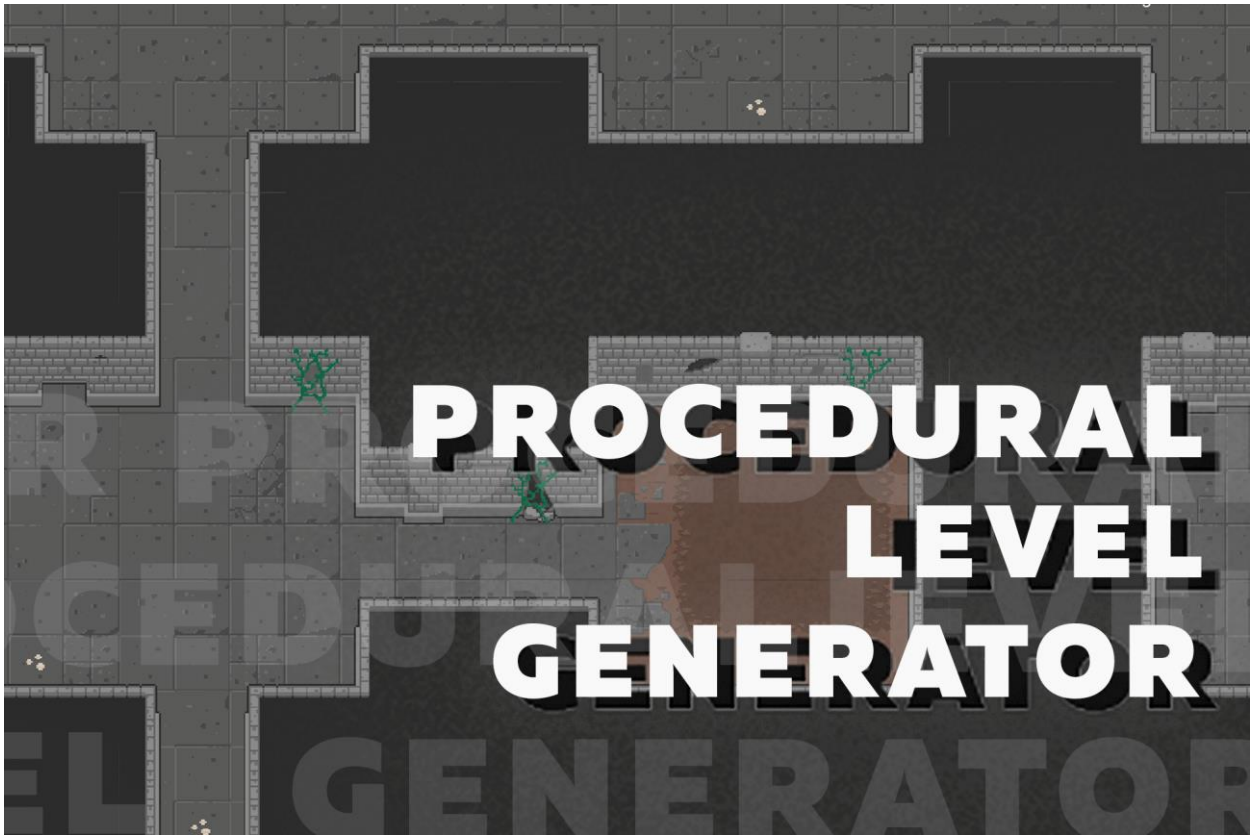
Dungeon Procedural Level Generator Documentation



Introduction

A universal algorithm for any game where procedural level generation is required. The generator can easily create up to 100 rooms without time delays.

Asset features:

1. fast and thought-out algorithm for generating dungeons based on a binary matrix;
2. conveniently create your own rooms using tile mappings;
3. ability to customize dungeon branching levels;
4. each generation has a unique dungeon.

Asset included:

1. One demo scene;
2. 16 prefabs of various rooms;
3. One demo tilemap

Example Scenes

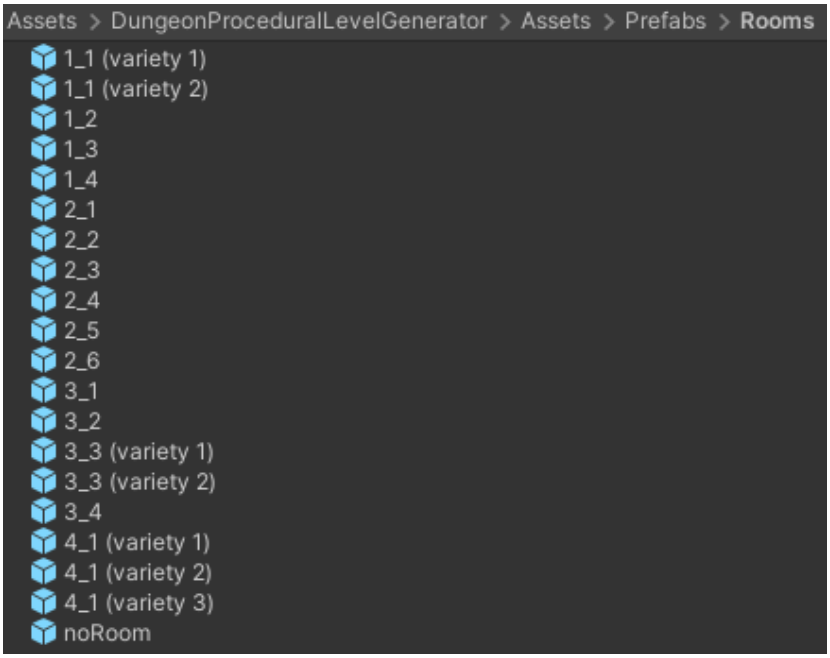
An example scene demonstrates the operation of the algorithm. In order to see the algorithm in action, you need to run the scene. A dungeon will be generated on your screen.

In addition to demonstrating the operation of the algorithm, an example of a scene with gameplay is also presented, where the character and camera control system has already been implemented.

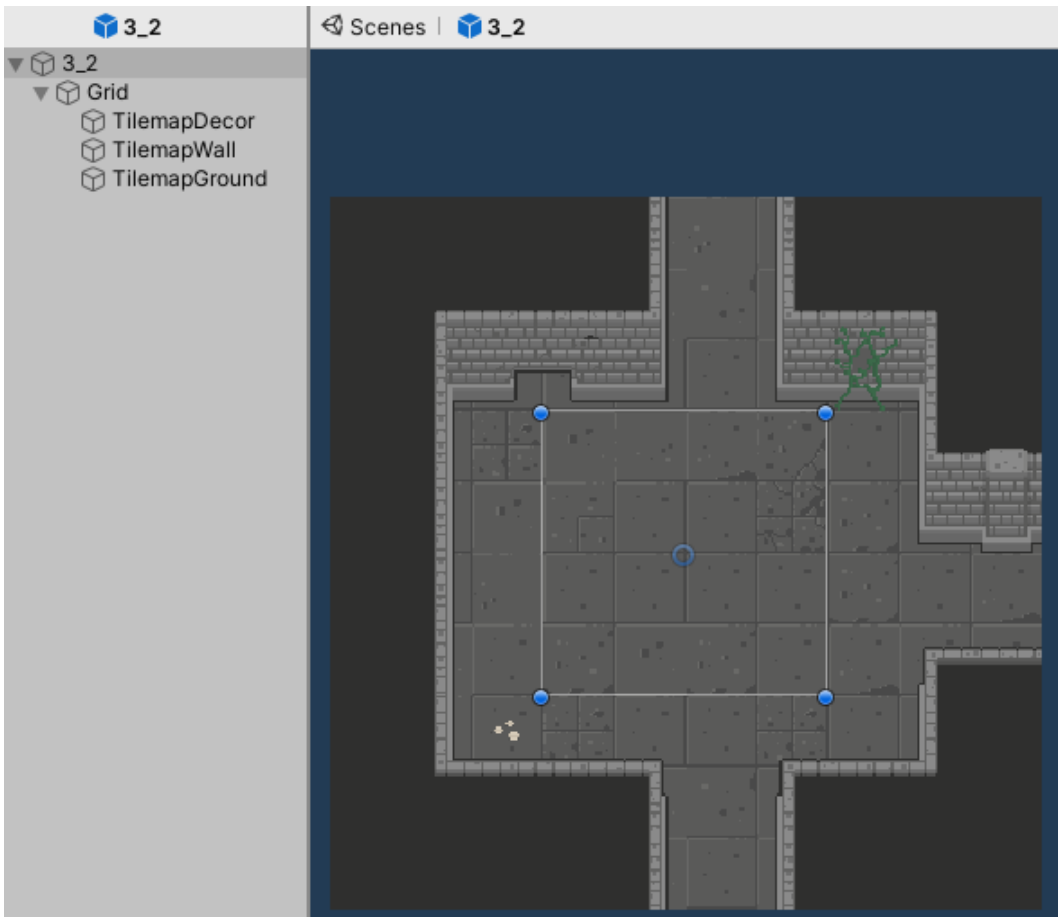
For new generation press «R».

Variety of rooms

All rooms are prefabs, in turn, each can be customized to taste:



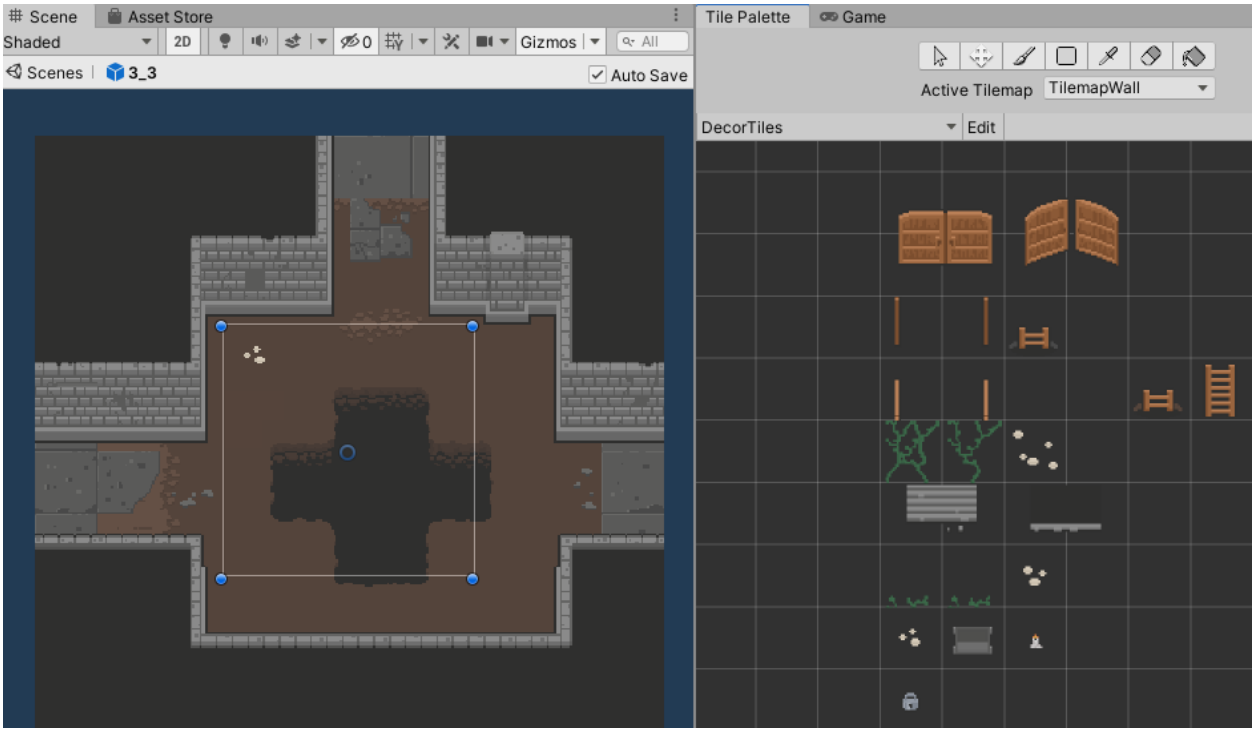
The first digit in the name indicates the number of exits from the room, the second is the serial number. For example, a prefab named «3_2» means that the room contains three exits:



*Rooms of the same type can have several varieties. They will be randomly generated in our dungeon. All room prefabs must be referenced to the object in the **ArrayRoom.cs** script. You will learn more about this in the next section below, "**Generator Settings**".*

A prefab named "noRoom" is needed to fill the empty space (where there are no rooms) when generating the level.

To edit rooms, you need to use the tilemap tool. But this is just an example, you can do it as you like:



Generator settings

For the functioning of the dungeon generator, you need to connect two scripts to the scene:

1. **GenerationMap** - this script, which generates a binary matrix, on the basis of which, our dungeon will be built.
2. **ArrayRooms** - this script is directly responsible for the visual display of the dungeon itself. Based on the generated matrix, it compares our rooms and places them on the stage

In the inspector, you need to set up the "**ArrayRoom**" script, add the prefabs of our room to the list of **RoomsStructure** objects in the **varietiesRooms** array field. This array contains all the varieties of the same room type so that as much variety of rooms as possible is used when generating.

In the **RoomsStructure** list, all rooms have a strict ordering that is linked between the matrix value and the room prefabs. For a better understanding of exactly which room is assigned in the array, each element of the list has a name with a visual designation of arrows, where arrows indicate the direction of exits from the room.

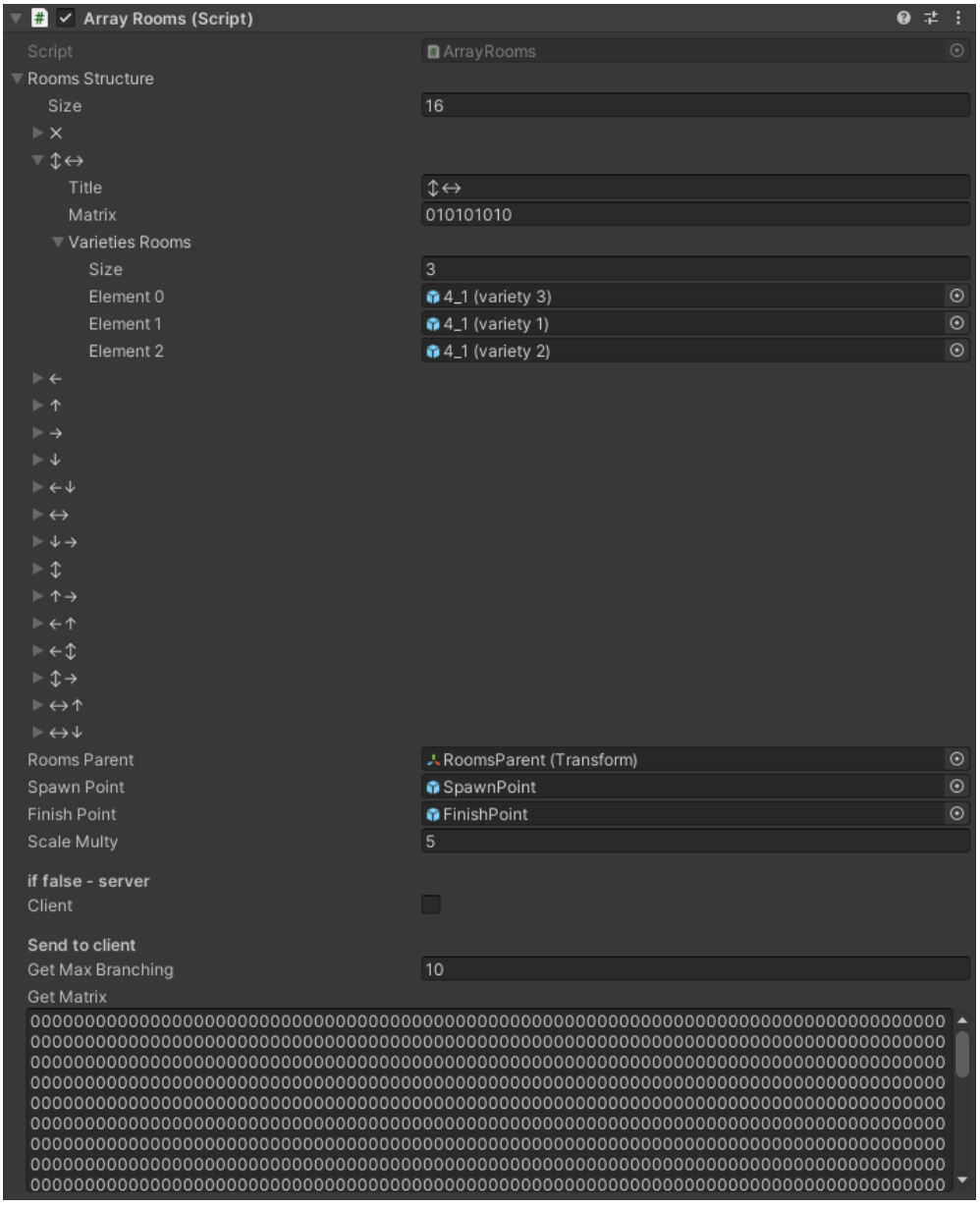
It is also necessary to indicate the link to the parent object in the «**Rooms Parent**» field, in which the generated rooms will be placed according to the hierarchy of the object. It is necessary that everything is grouped.

The "**Scale Multy**" field is a multiplier for increasing the scale parameter for the generated maze.

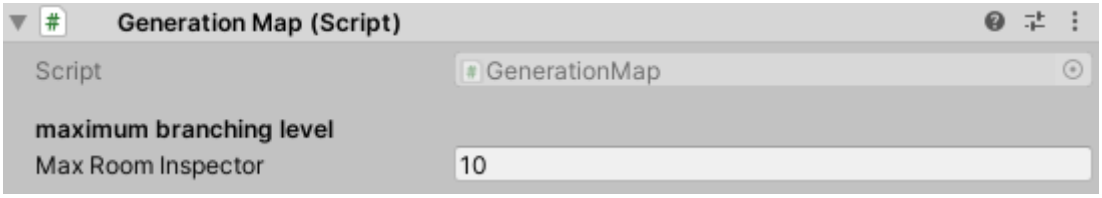
Boolean value "**Client**" - if this option is activated, the maze will be generated by the given matrix, in this case the value of the matrix is taken from the value of the "**GetMatrix**" field. This feature is implemented for network games when the server generates a dungeon and sends a matrix to all clients. If you need a new generation every time you run the level, leave the "**Client**" field false.

Field "**Get Max Branching**" - takes the value of the number of branches of the maze from the center. For example, if the value is 10, then in each direction (left, right, top, bottom) there will be a permissible probability of generating 10 rooms.

The "**Spawn Point**" and "**Finish Point**" fields contain the prefab of the corresponding points, which according to a special algorithm will be located on the generated dungeon, while at the same time unraveling at maximum distance from each other. This opportunity gives the developer to place his character in the right place in the dungeon.

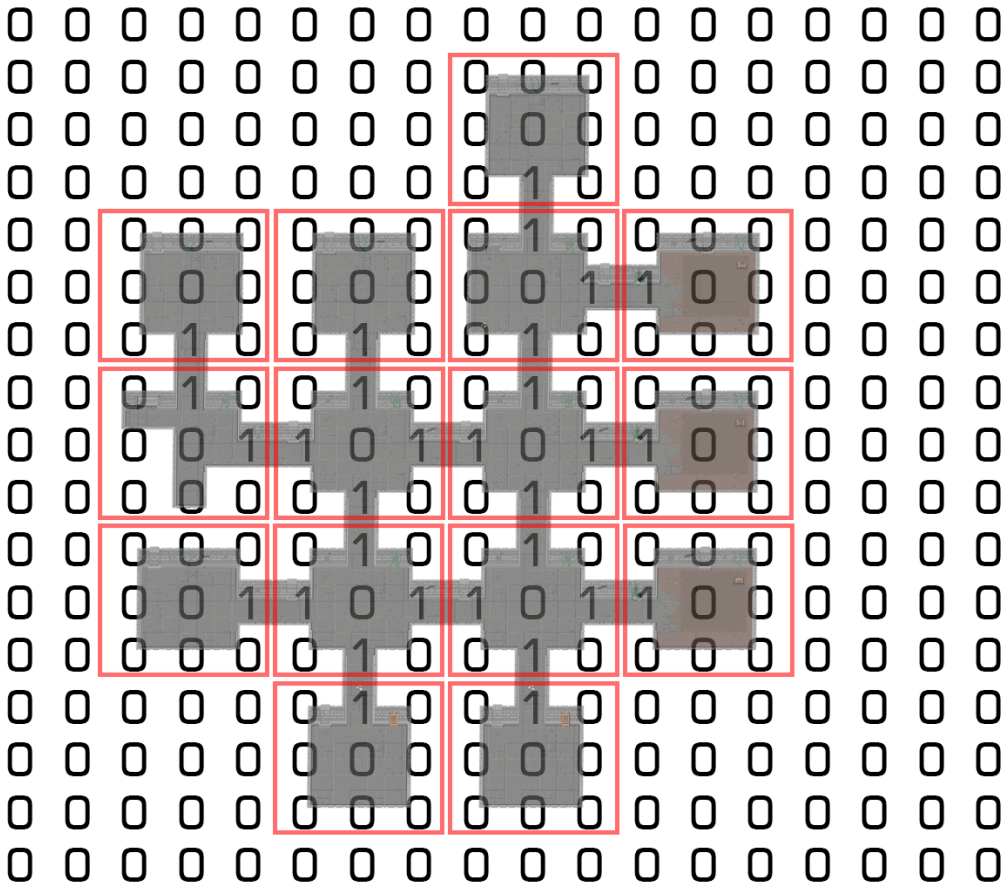


In the «GeneratorMap» script inspector, specify the maximum number of branches for generating rooms in the «MaxRoomInspector» field:



For example, if you specify a value of 3, then three rooms can be generated from the center in every four directions.

How the algorithm works



The room has a 3x3 matrix dimension, where
1 - door
0 - floor

Support

If you are confused by something, you can write to denis@proger.xyz and we will do our best to contact you immediately. We hope you enjoy our asset and good luck in creating your games!