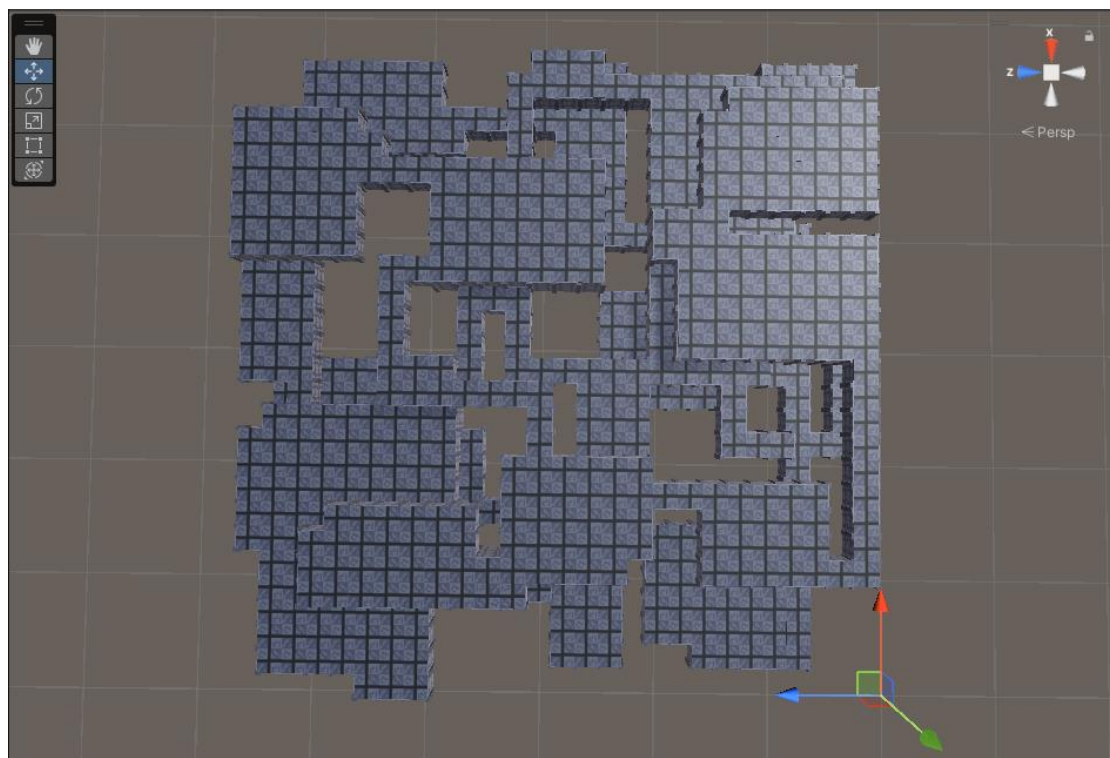
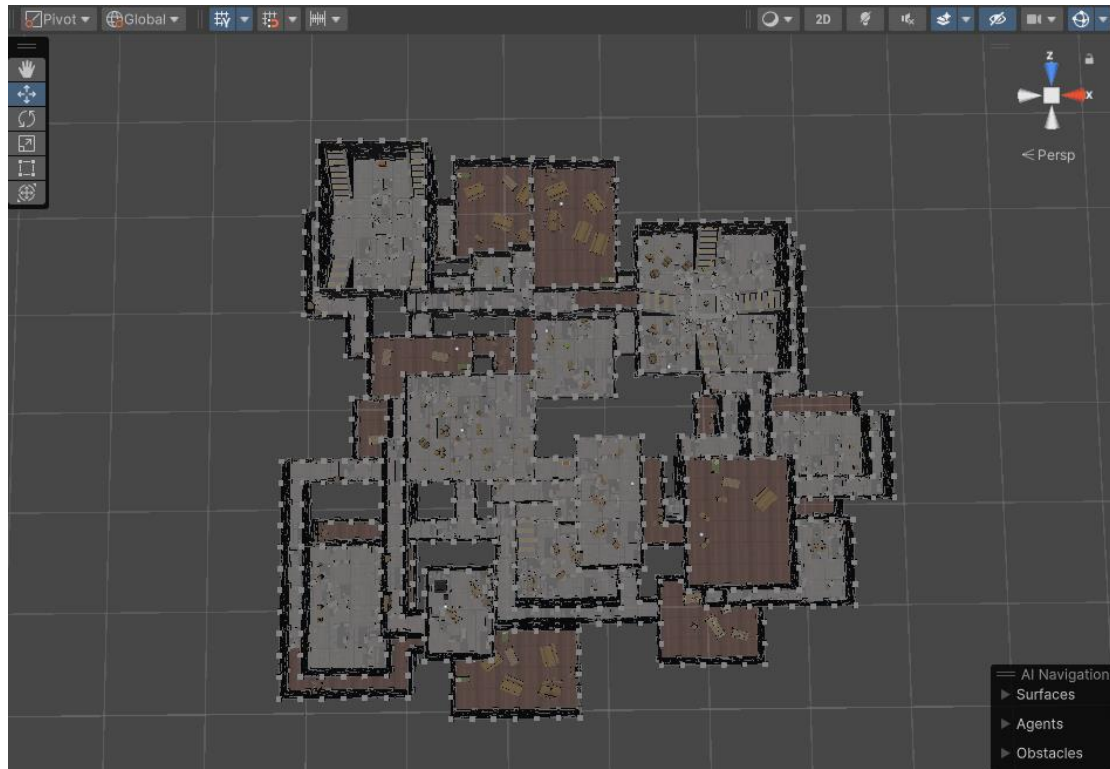


# Documentation for Simple 3D Dungeon Generator

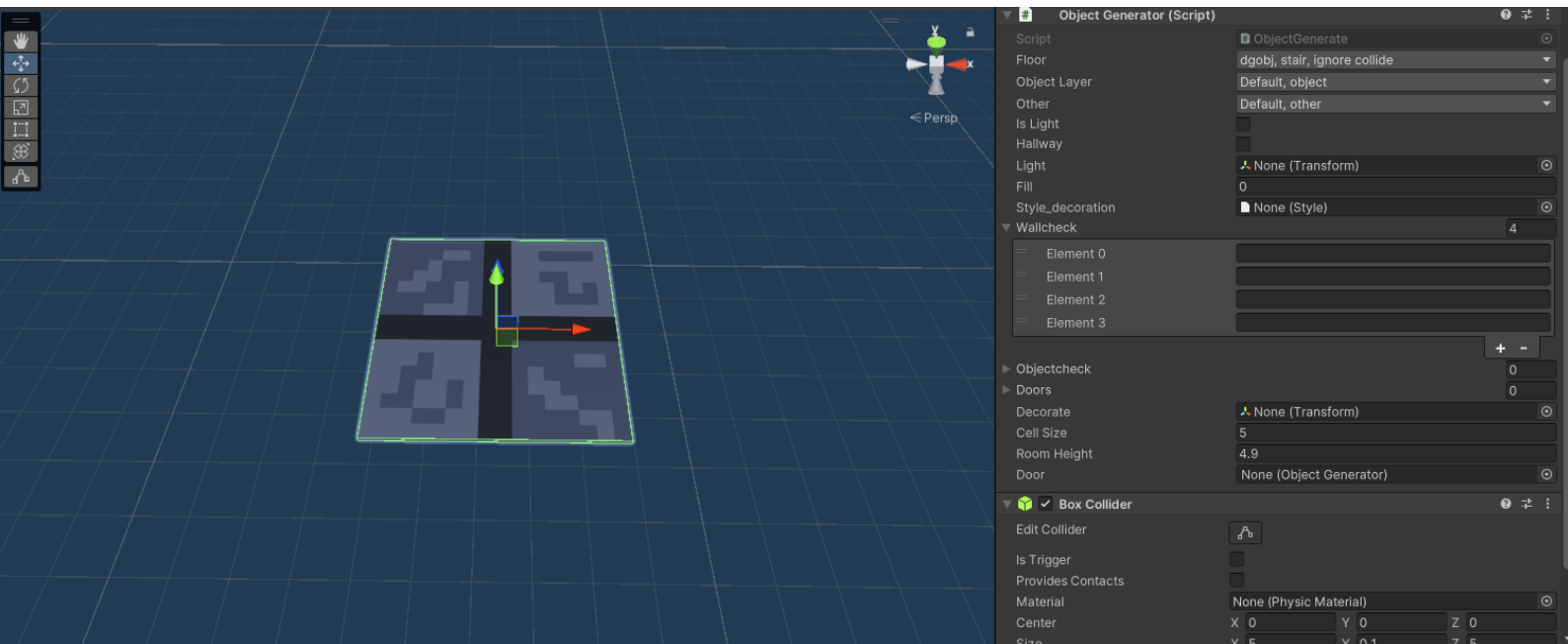
By bennynil



## Overview

The Simple 3D Dungeon Generator creates procedurally generated 3D dungeons by Using 2D triangulation, 2D path finding, 2D grid from [vazgriz's project](#) and add custom room to make 3D layout dungeon and room has multiple floors and can connect to each other possible and can control how many ways to go to other layers, also add room generation, random decoration and lock and keys generations.

## Floor Element



### 1. Pivot Point

- **Positioning:** The pivot point of the floor element should be centered. Its vertical alignment (up or down) is flexible but should be consistent to ensure proper alignment with other dungeon components.

### 2. Object Generator Component

Each floor element must include an Object Generator component with the following functionalities:

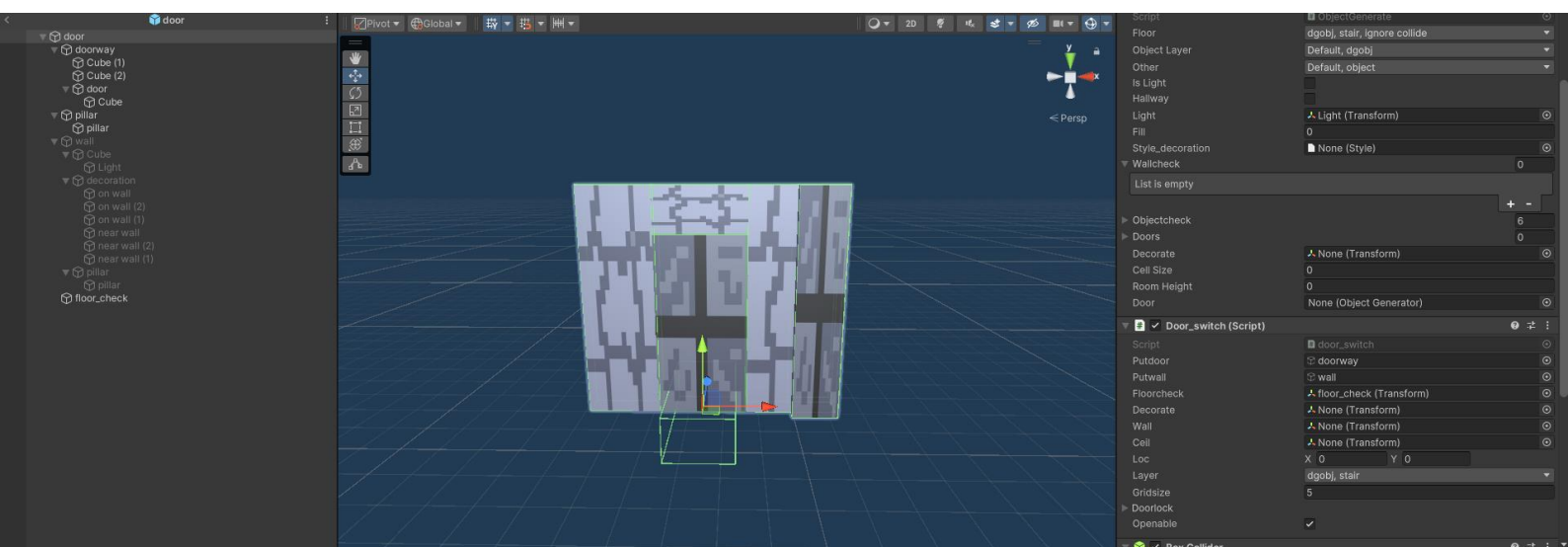
- **Floor Layer:**
  - **Purpose:** Determines the placement of walls or doors based on adjacent structures such as stairs.
- **Object Layer:**
  - **Purpose:** Ensures objects do not collide with one another.
- **Additional Layer:**
  - **Purpose:** Provides additional flexibility for object placement, allowing overlaps with the Object Layer.
- **Object Check Array:**

- **Purpose:** Defines potential spawn points for objects on the floor.
- **Functionality:** An array of transforms indicating where objects can be placed. The actual placement is often determined by the room configuration, and some spots may be left empty.
- **Cell Size:**
  - **Purpose:** Defines the 2d size of each floor cell.
- **Room Height:**
  - **Purpose:** Sets the vertical size of the room.
  - **Recommendation:** Set the room height slightly lower than the actual required height to prevent potential overlap issues with rooms positioned above.

### 3. Box Collider and Layer

- **Box Collider:**
  - **Requirement:** Each floor element must have a box collider.
  - **Purpose:** Ensures physical interaction and collision detection within the dungeon environment.
- **Layer:**
  - **Requirement:** Each floor element must be assigned to the “dgobj” layer.
  - **Purpose:** Facilitates proper categorization and interaction with other dungeon components.

## Wall Element



### 1. Pivot Point

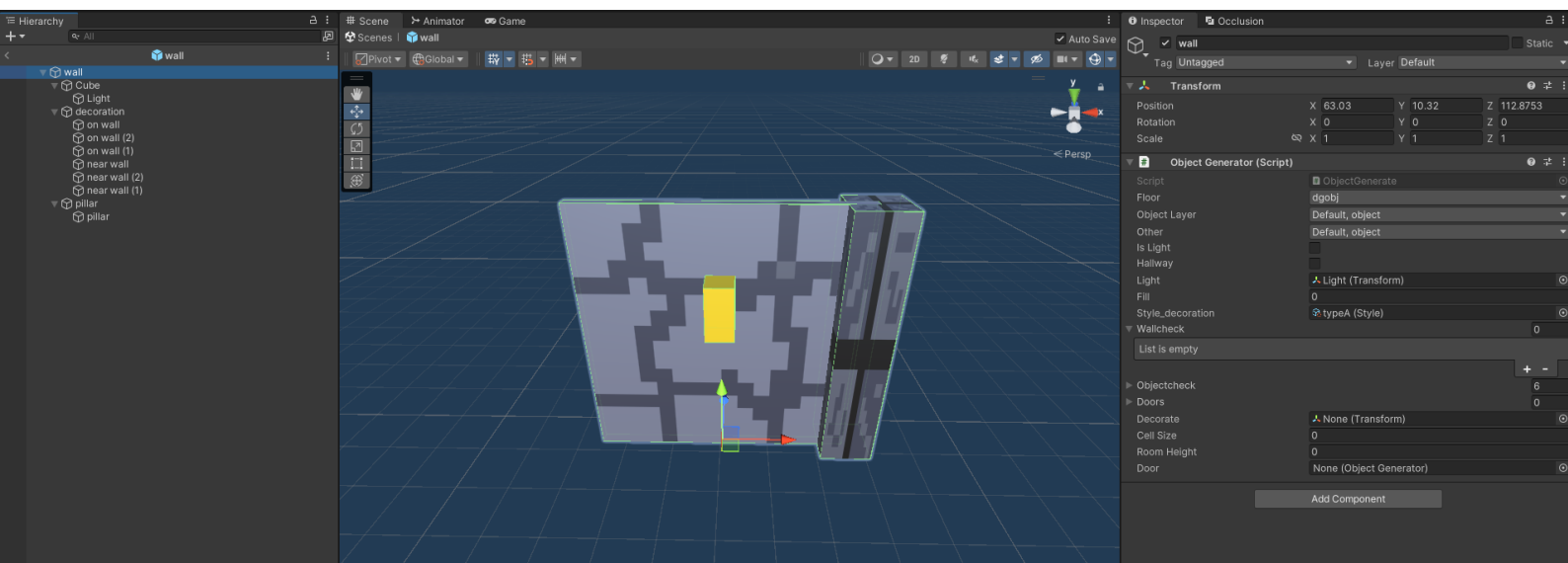
- **Positioning:** The pivot point of the wall element must be positioned in the middle and aligned at the bottom.

### 2. Object Generator Component

The wall element must include an Object Generator component with the following functionalities:

- **Light Management:**
  - **Purpose:** Manages lighting for walls based on their location.
  - **Functionality:** Automatically turns off lights for wall sections where lighting is not needed.
- **Decoration Tag Rules:**
  - **Object on Floor:** Objects placed on the floor should have the tag "OBJECT".
  - **Object on Wall:** Objects placed on the wall should have the tag "WALLOBJECT".
  - **Object Near Wall:** Objects located near the wall should have the tag "NEARWALLOBJECT".

## Door Element



### 1. Basic Structure

- **Setup:** The door element is an empty GameObject that contains both the door and wall components. Initially, the wall component should be inactive.

### 2. Object Generator Component

- **Requirement:** The door element must have an Object Generator component.
- **Configuration:** The wall component associated with the door must also have an Object Generator component. Ensure the door's Object Generator component is referenced correctly in the wall's configuration.

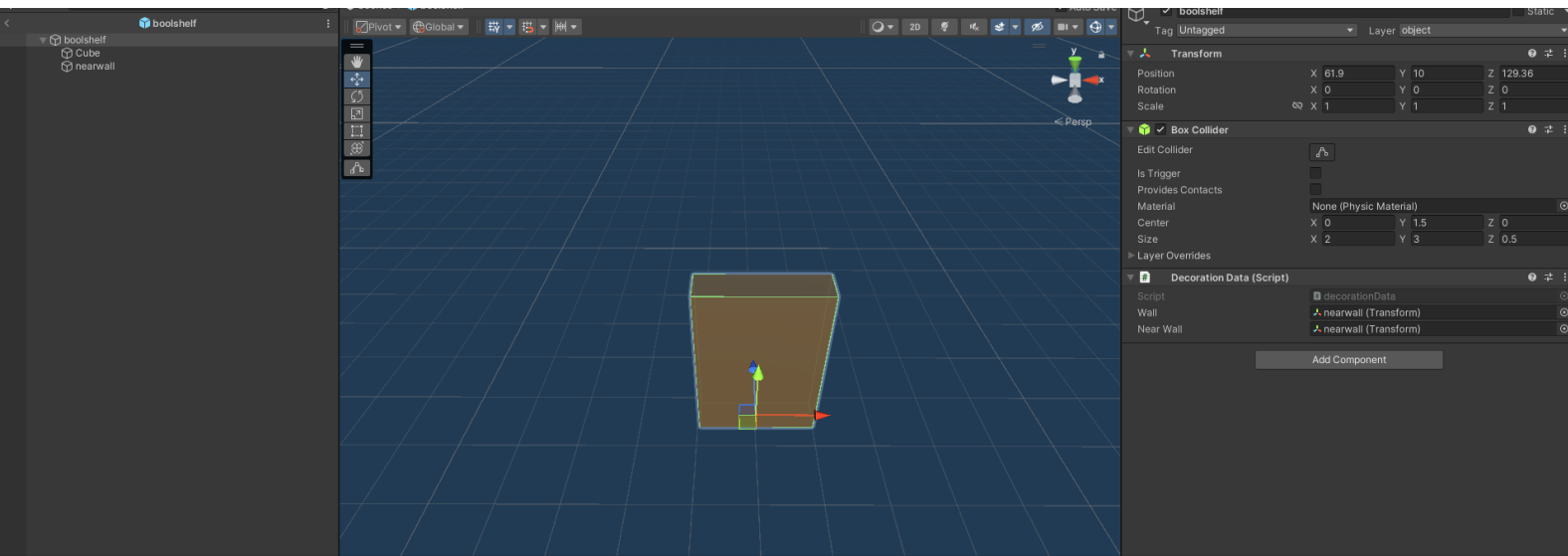
### 3. Door Switch Component

- **Components:**
  - **Put Wall:** Assign the wall object to the “putwall” field.
  - **Put Door:** Assign the door object to the “putdoor” field.
  - **Floor Check:** Use an empty GameObject positioned on the opposite side of the door (outside the room) to ensure proper placement.
- **Layer:** Set the layer for the door switch component to match the floor layer.

### 4. Collider Configuration

- **Collider Switch:** Ensure the object has a collider switch and assign box collider to box field to handle interactions properly.
- **Box Collider:** Attach a box collider to the door object.
  - **Purpose:** The box collider prevents other floors from incorrectly placing walls in the door's space.

## Decoration Element



### 1. Layer Assignment

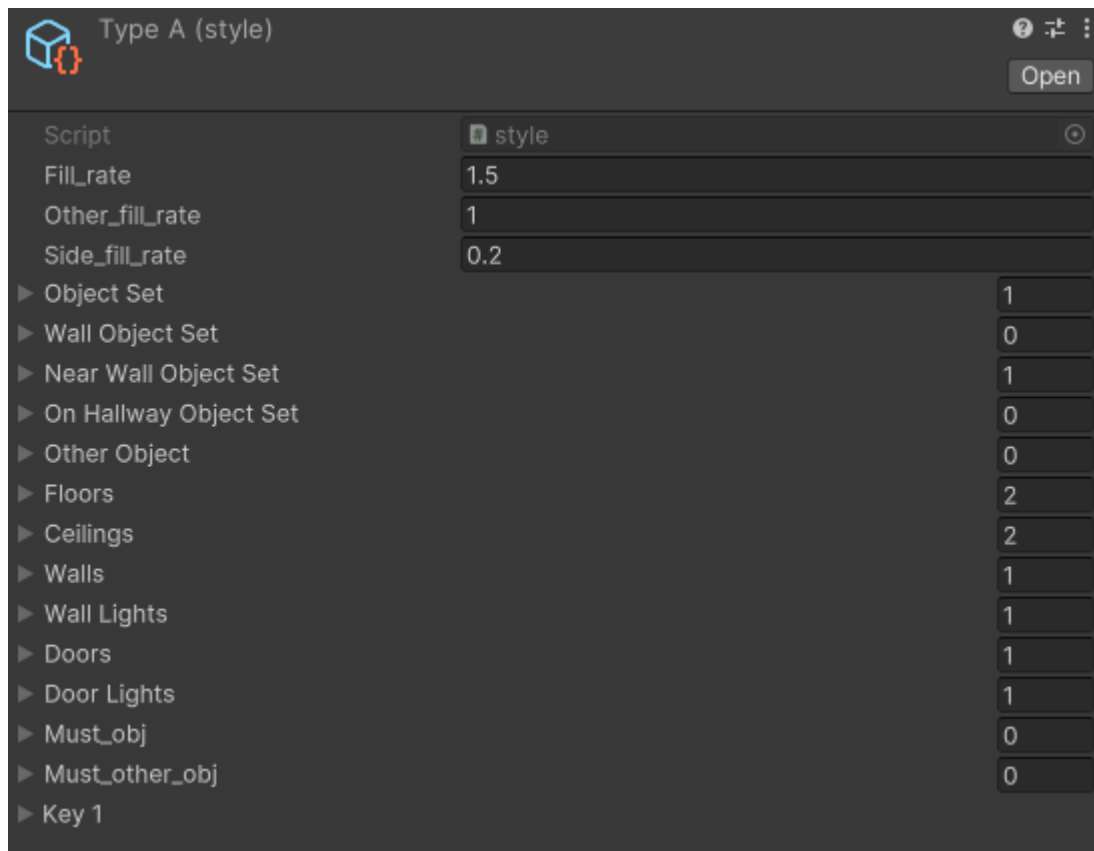
- **Layer:** Set the layer of the decoration to "object" unless you want to set it as "other" layer.

### 2. Decoration Data Component

- **Requirement:** Each decoration must have a Decoration Data component.
- **Pivot Point:**
  - **Transform:** Assign the transform to specify the pivot point for placing the decoration. Use the original pivot if the decoration is placed on the floor.
- **Data Assignment:** Ensure the Decoration Data component is not left empty. If necessary, assign default values or the same values across decorations for consistency.

## Room and Style and Room Game Object

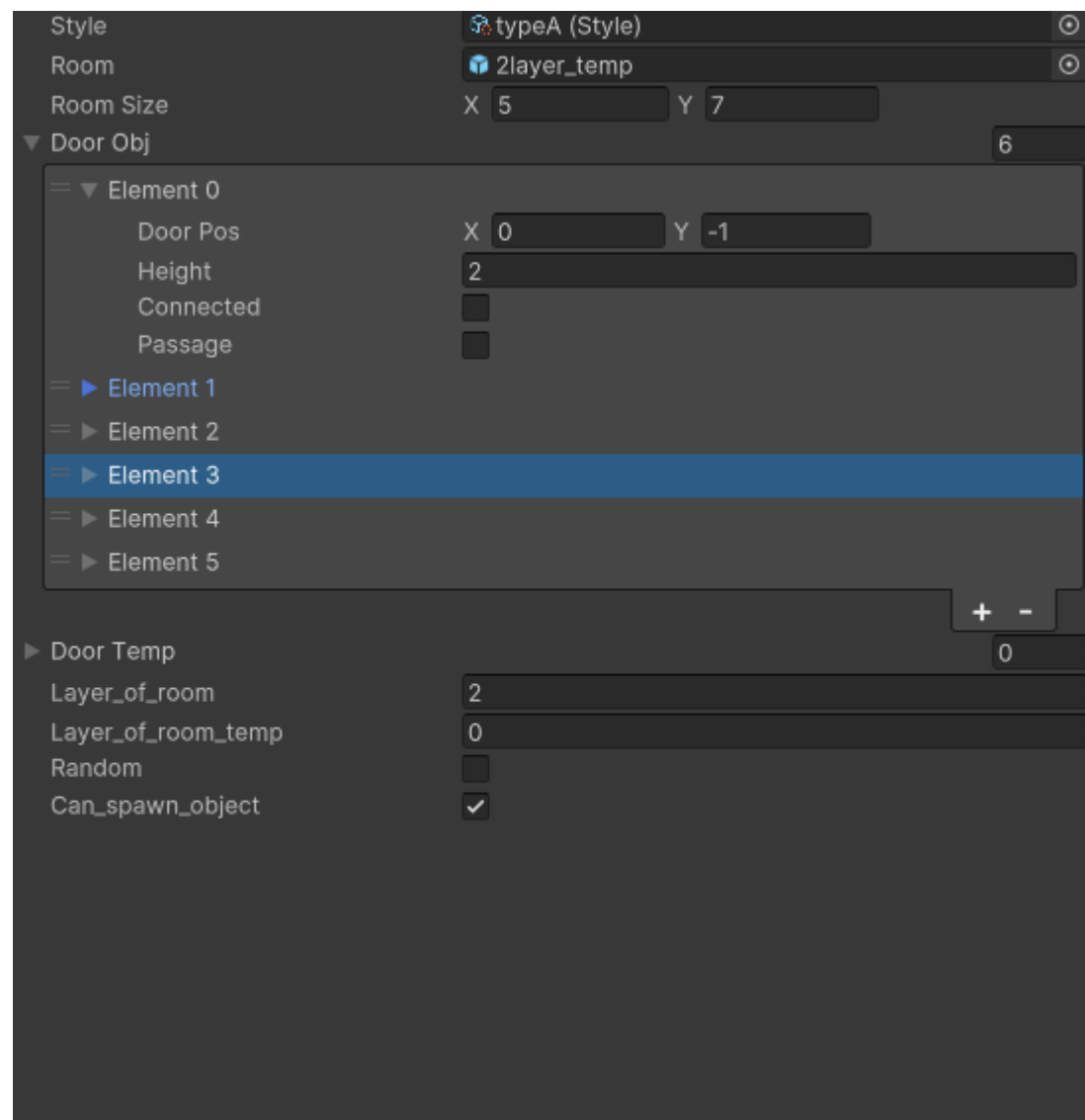
### Style



### Creating a Style

- **Creation:** Right-click and select in menu to generate a new style.
- **Components:**
  - **Fill Rate:** Defines how much of the room's space is occupied by objects. The fill rate is calculated based on the size of the objects and their placement. Due to randomization in location and rotation, the actual fill might not be the specified value. So set it to more than one doesn't mean there will be full of objects.
  - **Other Fill Rate:** Similar to the fill rate but applies to other objects.
  - **Side Fill Rate:** Determines the likelihood of spawning wall objects or objects on walls.
  - **Other Values:** where you put prefabs in list.
  - **Must Object:** Specifies objects that must be placed in the room.
  - **Must Other Object:** Objects that must be included in the room but are not strictly required.

## Room



Style typeA (Style)

Room 2layer\_temp

Room Size X 5 Y 7

Door Obj 6

- Element 0
  - Door Pos X 0 Y -1
  - Height 2
  - Connected ☐
  - Passage ☐
- Element 1
- Element 2
- Element 3
- Element 4
- Element 5

+ -

Door Temp 0

Layer\_of\_room 2

Layer\_of\_room\_temp 0

Random ☐

Can\_spawn\_object ☒

### Creating a Room

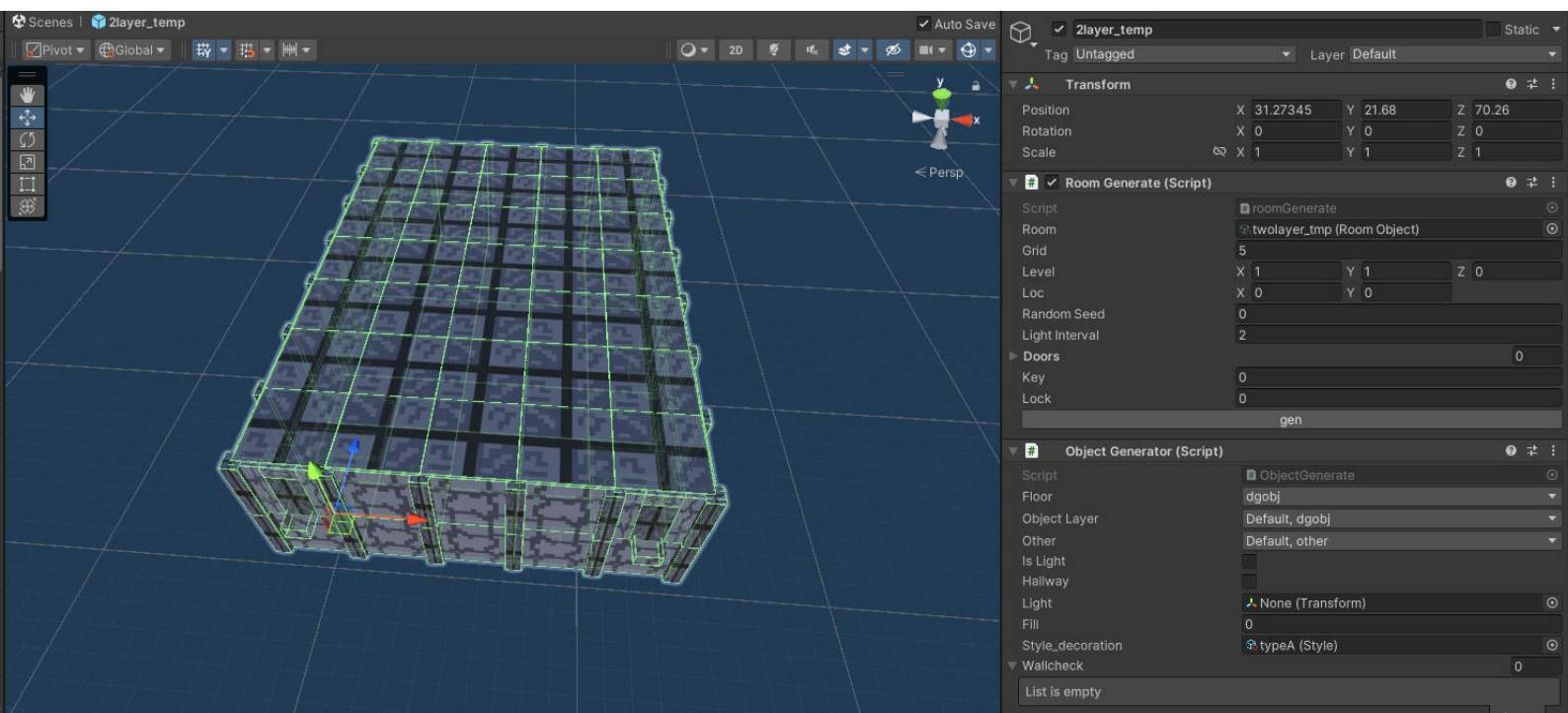
- **Creation:** Right-click and select in menu to generate a new room object.
- **Components:**
  - **Style Assignment:** Apply a style to the room by placing the style object in the style field of the room configuration.
  - **Room Object:** Place the room gameobject in the designated room field.

### Door Configuration

- **Door Object:** Defines the door's location in the room, typically representing an exit (e.g., position where a player going outside).
- **Layer:** Determines the height of the room by setting the layer.



## Room Generation and Crafting



**Room Generation:** Rooms can be generated randomly or crafted manually.

- **Random Generation:** Utilize random settings to generate room layouts and door placements. Ensure to check that objects do not spawn in passages where they are not needed.
- **Crafting:** For manually crafted rooms, set up an empty GameObject named "Decoration" and assign it to the decoration field in the Object Generator component.

### Room Components

- **Room Generator Component:**
  - **Room Object:** Assign the room object.
  - **Grid:** Set the grid size to match the floor size or cell size.
  - **Light Interval:** Specify the distance between walls and light sources to control lighting placement.
- **Object Generator Component:**
  - **Style:** Assign a style to the room to control object placement and decoration.

## Dungeon Generator

The screenshot shows the 'Generator 2D (Script)' interface with the following settings:

- Script:** Generator2D
- Size:** X 30, Y 30
- Room Max Size:** X 6, Y 6
- Room Min Size:** X 3, Y 3
- Rooms:** 9
- Randint:** 1
- Cost Function\_longer:** 0
- Height\_floor:** 5
- Interval\_source:** 2
- Layers:** 3
  - Element 0:**
    - Rooms:** 6
    - Number\_for\_rooms:** 6
    - Num\_of\_room:** 12
  - Element 1:**
    - Rooms:** 6
    - Number\_for\_rooms:** 6
    - Num\_of\_room:** 10
  - Element 2:**
    - Rooms:** 4
    - Number\_for\_rooms:** 4
    - Num\_of\_room:** 5
- Grid Size:** 5
- Door Chance:** 0
- P:** holder
- Keys In Room:** 1

### Layered Structure

- **Building Order:** The dungeon is constructed from the lowest layer to the top. This means the initial layers set the foundation for the dungeon, and subsequent layers build upon them.
- **Room Placement:** Rooms are defined by their type and quantity per layer. Each layer can contain different types of rooms, and the number of rooms specifies how many of each type should be placed. Since not every room can be placed and the multiple layer rooms also count as a room in layer so the value "num of room" should be smaller than combination of number for rooms array.

### Room Configuration

- **Room Types:** Define what types of rooms you want in each layer.
- **Number of Rooms:** Indicates how many instances of each room type you

want in the layer. Since not every room type will be placed every time, the number should be less than the total number of possible rooms.

#### **Grid Size**

- **Cell Size:** The grid size for the dungeon layout is defined by the cell size. This ensures consistency in the dungeon's spatial arrangement and alignment.

#### **Floor Height**

- **Height Configuration:** Specify the height of each floor layer.
  - **First Layer:** Height is set to 0.
  - **Second Layer:** Height is set to 5.
  - **Third Layer:** Height is set to 10.
  - **Subsequent Layers:** Continue increasing the height incrementally (e.g., by 5 units) for each additional layer.

#### **Keys in Room**

- **Maximum Keys:** Define the maximum number of keys that can be placed in a room. This helps in managing the dungeon's complexity and ensuring that rooms are appropriately populated with items.

#### **Player Prefab**

- **Prefab Configuration:** The player prefab (P) represents the player's starting point or spawn location in the dungeon.