

Advanced Dungeon Generator

USER GUIDE



[Creating a Dungeon](#)

[Dungeon Components](#)

[Dungeon Shape](#)

[Dungeon Brush](#)

[Dungeon Decoration](#)

[Dungeon Properties](#)

[Dungeon Generator](#)

[Dungeon Loader](#)

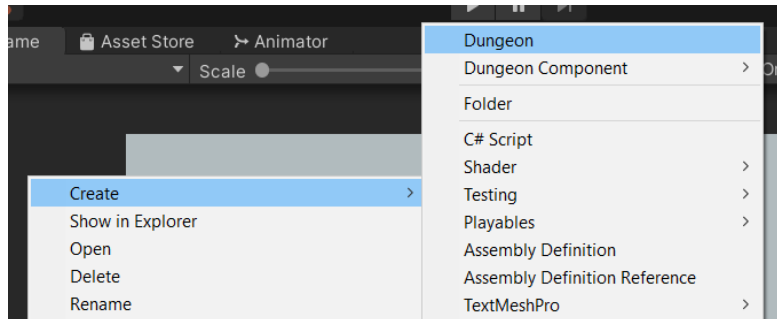
[Shape Visualizer](#)

[Security Check](#)

[Demos](#)

Creating a Dungeon

To create a new dungeon or any of its components, select the menu item **Assets > Create > Dungeon** or, in the Project Window, **Right Click > Create > Dungeon**.



Understanding the Dungeon System

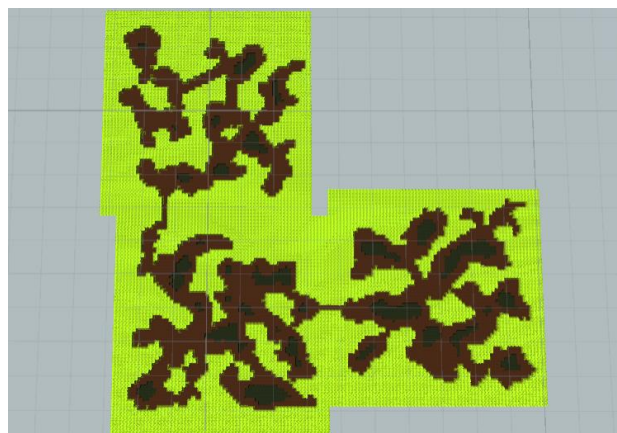
Before working with the Dungeons, we need to understand its fundamental components: **Dungeon Segments** and **Dungeon Rooms**.

- **Dungeon Room**

A **Dungeon Room** is an isolated and independent set of connected chambers.

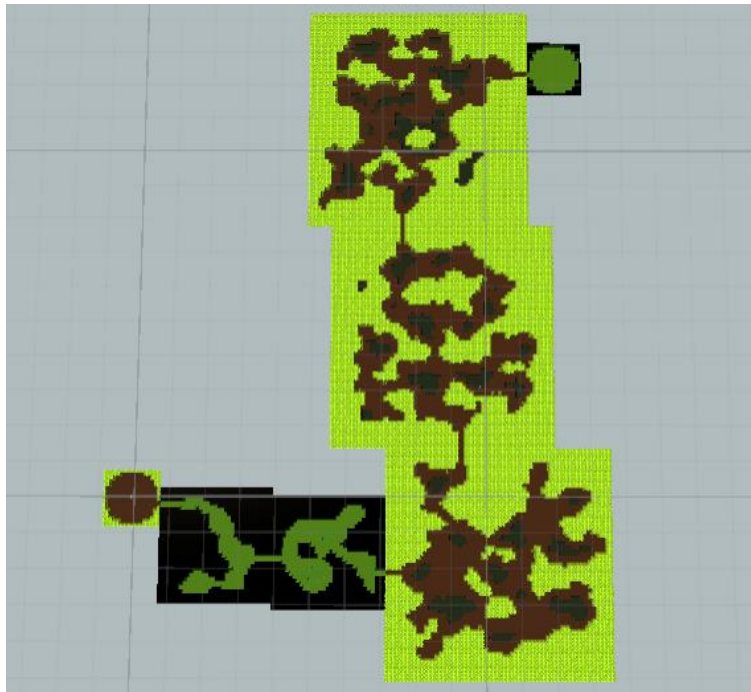


A **Dungeon** is a successive and connected collection of said rooms.



- **Dungeon Segment**

Dungeon Segments are groups of **Dungeon Rooms** generated under the same rules.



Each **Dungeon** can be divided into several **Dungeon Segments**.

Each **Dungeon Segment** can be assigned with different properties that will alter how its rooms will be generated.

With this system, each Dungeon can have a good composition variety without concatenating several dungeons.

Dungeon Components

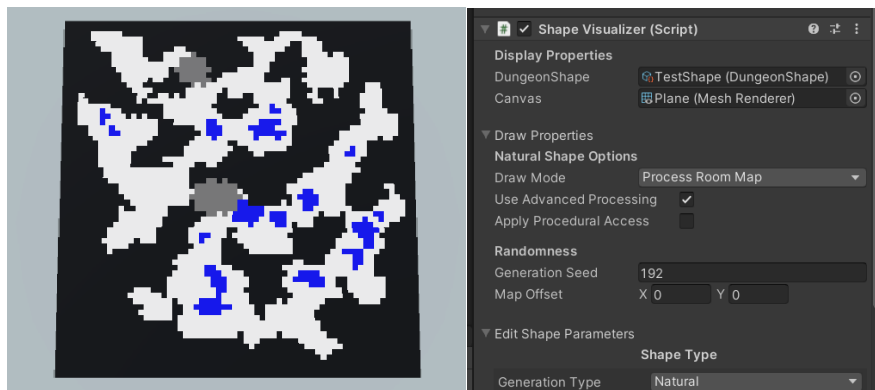
Each Dungeon Segment can be customized through the modification of these components:

- **Dungeon Shape**

The **Dungeon Shape** is in charge of creating the **Dungeon Rooms**. You can think of it as the set of instructions for creating the Dungeon Rooms.

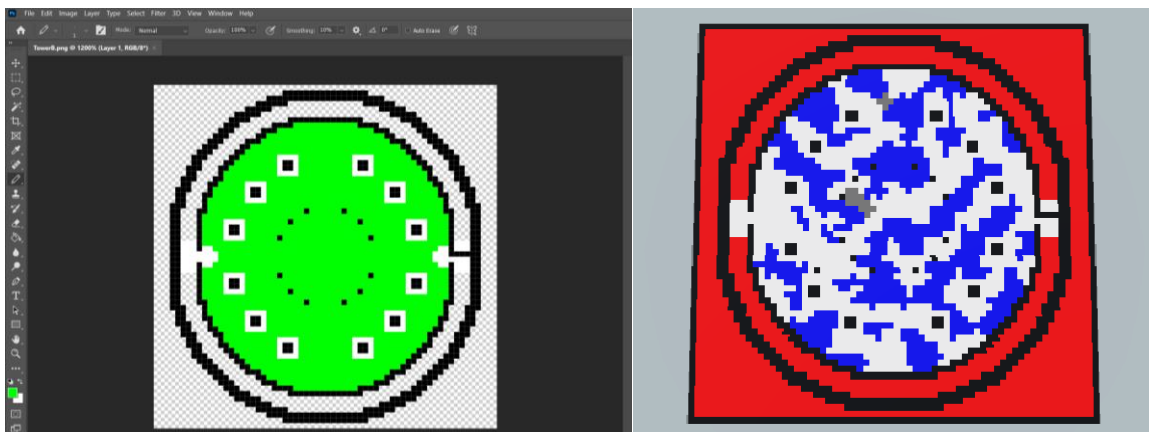
Through an algorithm, the Dungeon Shape values are transformed into integral matrices that will determine the physical properties of every tile in the room once they are generated.

You can see a recreation of the **Dungeon Rooms** of your **Dungeon Shapes** through the **Shape Visualizer** script.



The **Dungeon Shape** script will set 3 map tiles: a **Ground** tile, a **Wall** tile, and a multipurpose tile called **Structure** tile, which can be modified to be water, lava, void, rocks, a different type of wall, whatever you want.

Also, the **Dungeon Shape** script can set 2 types of generation for your **Dungeon Rooms**. The **Natural** is the default one and only requires modifying the generation properties. The **Artificial** one lets you import your own Dungeon Rooms as images and allows slight modifications.

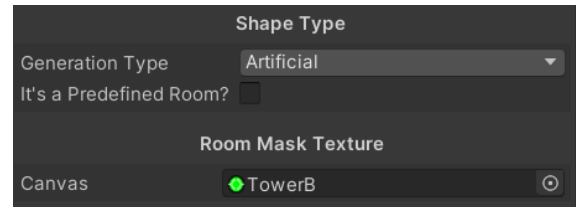
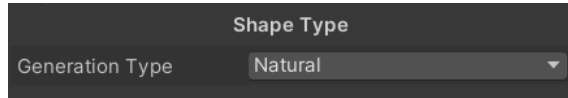


The next table could help you understand how the Dungeon Shape Script manages its values.

Tile	Description	Int Value	Write Color (For Image Import)	Read Color (As seen in the Visualizer Script)
Ground	The default walkable tile. The Dungeon Shape Script defines and limits the int tiles of every single chamber or sub-room with a common number. In this case, it goes from -1 to -inf.	(-inf, -1)	White #ffffff	White
<i>Ground (Passages Only)</i>	Specific tile that joins two sub-rooms. Has all the properties of the ground tile. The only change is the int value for better distinction.	0		Grey
Walls	The default obstruction tile.	1	Black #000000	Black
Structures	A multipurpose tile. Its properties can be set in their respective Dungeon Segment.	2	Blue #0000ff	Blue
<i>Empty Tile</i>	Artificial generation only Specifies the absence of a tile.	3	None Alpha must be zero	Red
<i>Tile to Overwrite</i>	Artificial generation only These tiles function as a mask for the natural procedurally generated dungeons.	4	Green #00ff00	Green will be replaced by the natural generation.
<i>Room Borders</i>	In-game generation only Specific tile for optional Room borders at Dungeon creation.	5		

Dungeon Shape Properties

Shape Type



Generation Type:

Defines the key properties needed for the creation of the Dungeon Rooms.

- **Natural:** A natural Dungeon Shape will use user modifications to alter a Perlin Noise Map used to create the Dungeon Rooms.
- **Artificial:** An artificial Dungeon Shape has nearly all the properties of a natural Dungeon Shape but will use a given image as a mask on top of all the natural generated Rooms.

It's a predefined Room? (Artificial Generation Only)

If checked, the masking process will be ignored, and all the Dungeon Rooms generated by this Dungeon Shape will be copies of the given image.

Room Mask Texture: (Artificial Generation Only)

The image used in the masking process of an Artificial Shape.

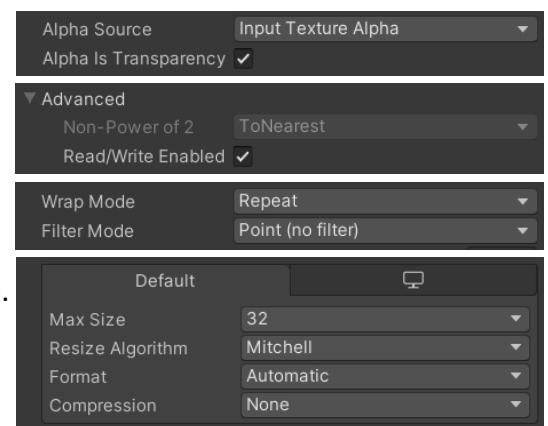
The mask image must have these properties:

In the Image Editor:

- Its dimensions must be multiple 16 pixels, and both must be the same length.
- It has to follow the set colors from the above table.

In Unity:

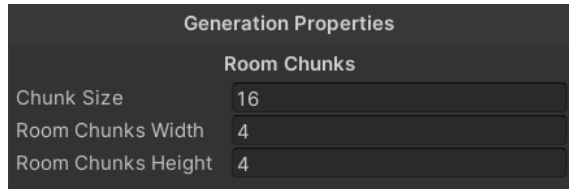
- **Alpha is Transparency** option must be checked.
- **Read/Write Enable** advanced option must be checked.
- **Filter Mode** option must be set to *Point (no filter)*.
- **Max Size** option must be set to the max size of the image.
- **Compression** option must be set to *None*.



Generation Properties

You can change the set limits for the Generation Properties at the top of the **DungeonShape** script.

Room Chunks (Natural Generation Only):

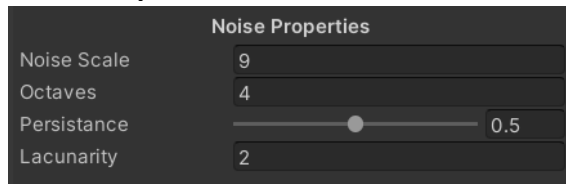


The screenshot shows a dark-themed window titled "Generation Properties". Inside, there is a sub-section titled "Room Chunks". It contains three input fields: "Chunk Size" with the value 16, "Room Chunks Width" with the value 4, and "Room Chunks Height" with the value 4.

The Dungeon Room tiles will be grouped in Chunks to improve game performance. In an Artificial Shape, these options are set by the dimensions of the given image.

- **Chunk Size:** Determines the width and height, in tiles, of the individual chunks.
(Limited from 4 to 32)
- **Room Chunks Width:** Determines the width, in chunks, of the Dungeon Rooms generated by this Dungeon Shape. *(Limited to a max of 8)*
- **Room Chunks Height:** Determines the height, in chunks, of the Dungeon Rooms generated by this Dungeon Shape. *(Limited to a max of 8)*

Noise Properties:

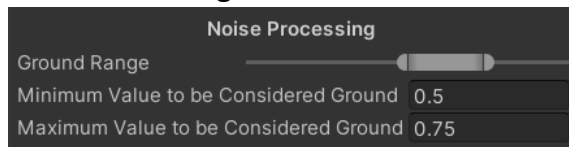


The screenshot shows a dark-themed window titled "Noise Properties". It contains four controls: "Noise Scale" with a value of 9, "Octaves" with a value of 4, "Persistence" with a slider set to 0.5, and "Lacunarity" with a value of 2.

Set the properties of the base Perlin Noise Map from where all Dungeon Rooms are generated.

- **Noise Scale:** Set the distance at what the noise map is seen. *(Limited to a max of 128)*
- **Octaves:** Set the levels of detail the Perlin Noise Map will have. *(Limited to a max of 8)*
- **Persistence:** Set how much each octave contributes to the overall shape.
- **Lacunarity:** Set how much detail is added or removed at each octave. *(Limited from -4 to 4)*

Noise Processing:



The screenshot shows a dark-themed window titled "Noise Processing". It contains three controls: "Ground Range" with a MinMax slider, "Minimum Value to be Considered Ground" with a value of 0.5, and "Maximum Value to be Considered Ground" with a value of 0.75.

The base Perlin Noise Map will be processed into an int matrix with only negative, 0, 1, or 2 values.

- **Ground Range:** This MinMax Slider represents the range of the 0 to 1 Perlin Noise Map values considered as Ground.
- **Minimum Value to be Considered Ground:** Any entrance of the 0 to 1 Perlin Noise Map below this value will be considered as a Wall.
- **Maximum Value to be Considered Ground:** Any entrance of the 0 to 1 Perlin Noise Map above this value will be considered as a Structure.

Room Processing:

Room Processing	
Minimum of Tiles to Form a Wall	20
Minimum of Tiles to Form a SubRoom	50
Failed SubRooms will be Converted to	Walls
Minimum of Tiles to Form a Structure	3
Maximum Possible Size for Passages	3

The processed Perlin Noise Map will be post-processed to comply with the following Dungeon Room generation rules.

- **Minimum of Tiles to Form a Wall:** Minimum number of connected wall tiles required to be considered as an appropriate wall.
- **Minimum of Tiles to Form a SubRoom:** Minimum number of connected ground tiles required to be considered as an appropriate chamber or sub-room.
- **Failed SubRooms will be Converted to:** SubRooms with less than the minimum required tiles will be converted to this specification.
- **Minimum of Tiles to Form a Structure:** Minimum number of connected structure tiles required to be considered as an appropriate structure.
- **Maximum Possible Size for Passages:** Maximum size for the passages that will connect distant chambers or sub-rooms. *(Limited to a max of 16)*

Falloff Map Mask (Natural Generation Only):

Falloff Map Mask	
Concentration	3
Remoteness	10

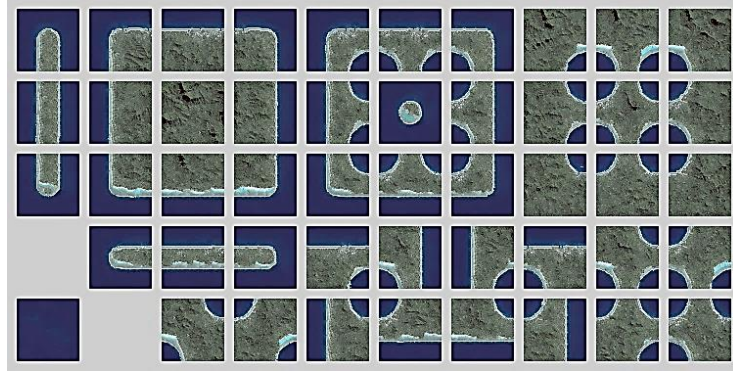
Before processing the 0 to 1 Perlin Noise Map, a Fallow Map will mask it to avoid open borders. In an Artificial Shape, the user is responsible for setting the Room Borders in the given image.

- **Concentration:** Set how condensed the Fallow Map will be. *(Limited to a max of 128)*
- **Remoteness:** Set how remote to the center the Fallow Map will be. *(Limited to a max of 128)*

- **Dungeon Brush**

The **Dungeon Brush** is the physical tileset of the **Dungeon**, and it can be changed to alter the aesthetics of the maps defined by the **Dungeon Shapes**.

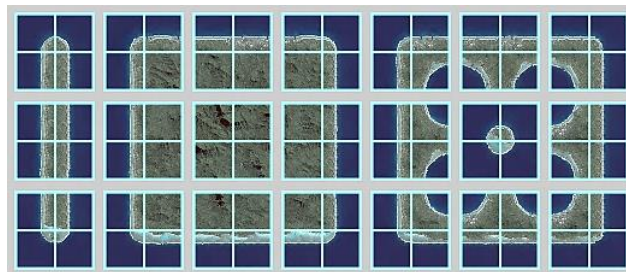
If you have worked with tilesets, you are already used to working with these kinds of guides:



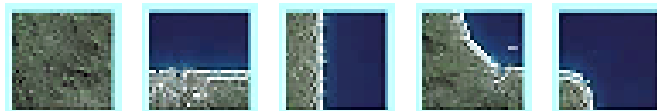
Advanced Dungeon Generator was born from my total and absolute lack of desire to do multiples of those, while working with traditional tile generation.

So, maybe you saw my previous claims and wonder how **Advanced Dungeon Generator** can create entire Dungeons with just 5 building blocks? Well, if you look closely:

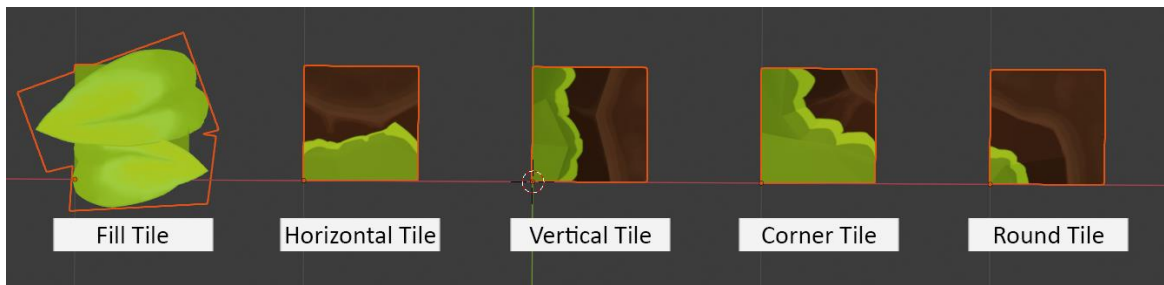
You can see that every previous tile can be divided into several components.



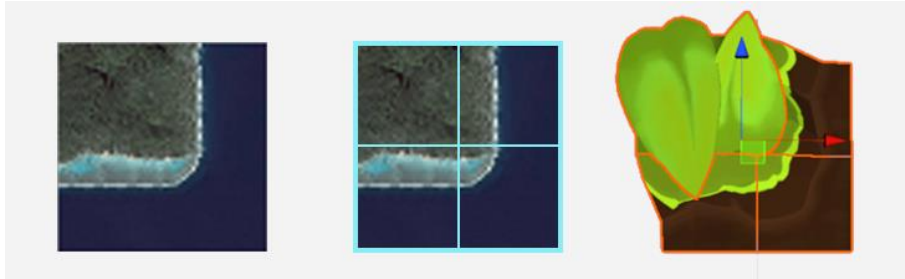
And that those components are shared between tiles; they are only rotated.



These **5 quarter blocks** are the **5 building blocks** that the **Dungeon Generator** uses as a **Dungeon Brush**.

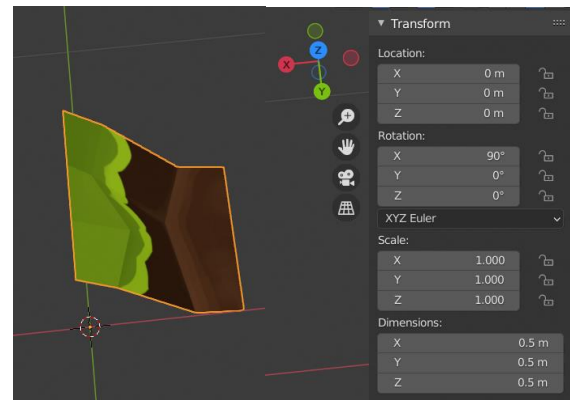
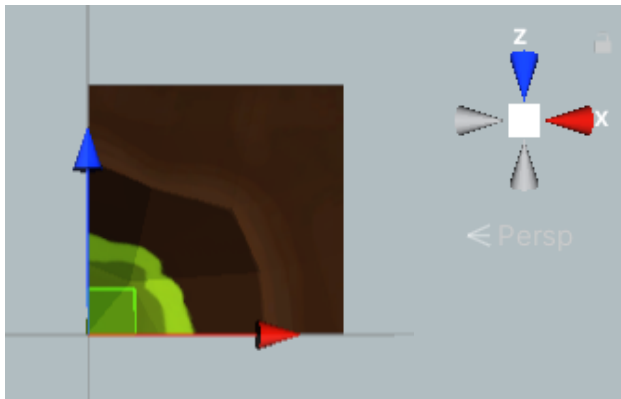


Advanced Dungeon Generator uses the traditional tiles but generates them on the fly. It creates them using those quarter blocks, right upper corner first and then rotates the rest clockwise.



To accomplish this, the given meshes of a **Dungeon Brush** must have these specifications:

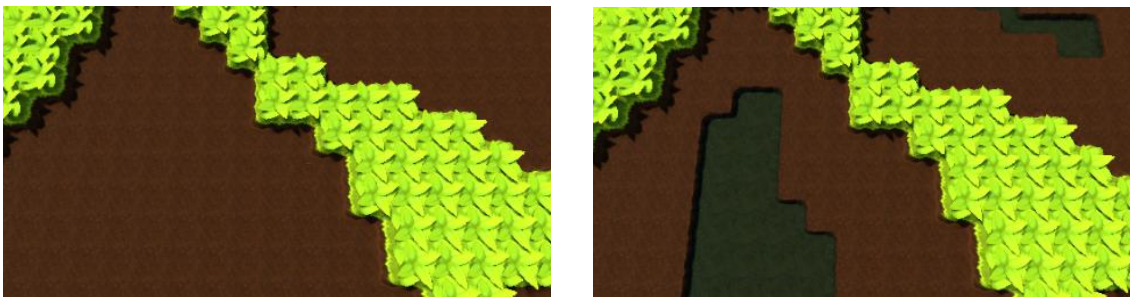
- The pivot point of the mesh must be in (0, 0, 0).
- The actual mesh can only be in the Unity X+, Z+ quarter.
- The mesh X, Z dimensions should be 0.5 x 0.5 maximum.
- The meshes had to have radial symmetry.



As you may have foreseen, this approach has an actual problem. Having so few blocks can make the results look extremely artificial.

This radial tiling can extremely limit the developer's creativity and the style of the end products. Fortunately, **Advanced Dungeon Generator** has systems that try to counter this problem.

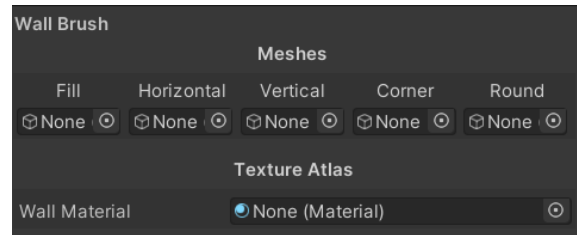
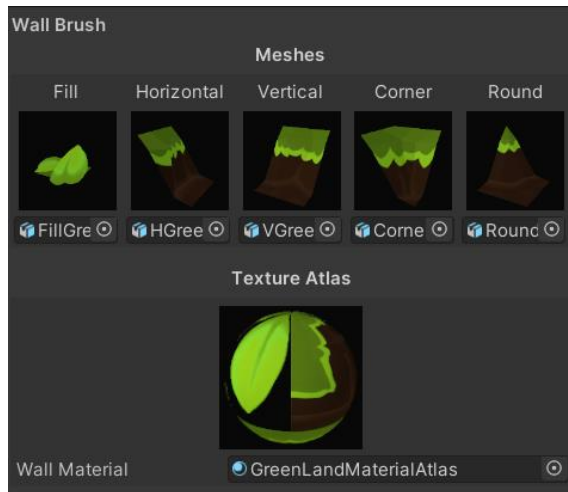
First, the **Dungeons** accept the use of a secondary **Dungeon Brush** to serve as **Structures**.



And, you can also add **Dungeon Decorations**.

Dungeon Brush Properties

Wall Brush



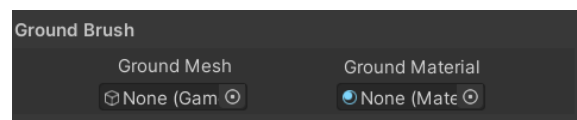
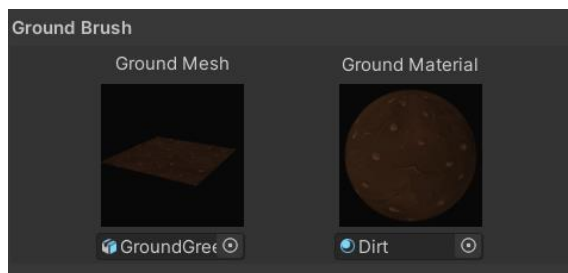
Meshes:

The actual heart of the Dungeon Brush. Each tile has its own selection box for your prefabs. The **Dungeon Generator** doesn't instance the actual **Game Objects**, it only takes the meshes (**Mesh Filter**) of given Game Objects to work in conjunction with the **Dungeon Shape** chunk settings for later mesh combination.

Texture Atlas (Wall Material):

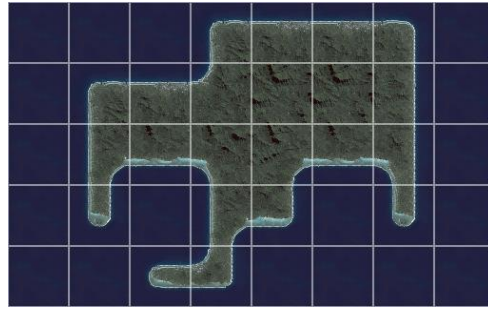
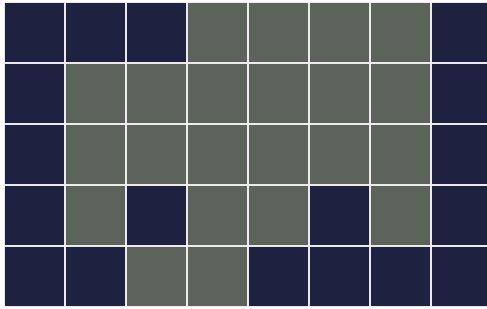
The **Dungeon Generator** only takes the meshes of the Game Objects, so none of the prefabs materials are retained for the building process. This material is the only one applied to the generated chunk combination of the five previous meshes. This means that the **UV Maps** of your five meshes must be in the same **Texture**.

Ground Brush

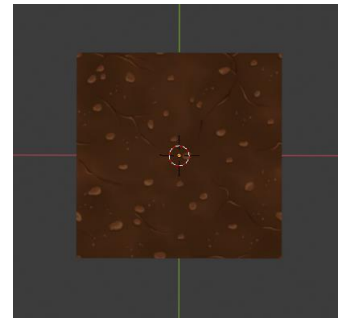
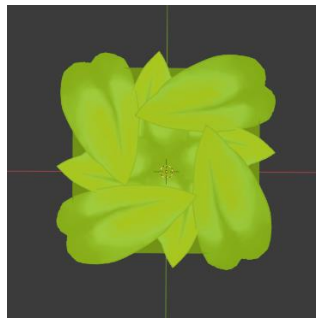
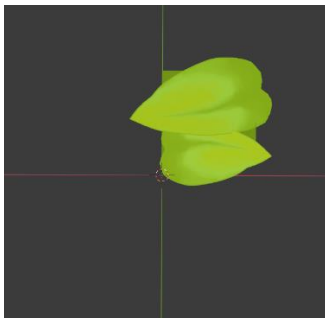


Ground Mesh:

So, there is something that I haven't said yet. If you have worked with tiles, you know that the whole purpose of these tilesets is to have a smooth transition between two states on a set grid. In these examples, the transition between a **High State** and a **Low State**.



The system is focused on creating the transition tiles on the fly. That's why it asks you for the state tiles; **why only one of them, you ask?** Because for now, and I have to clearly state this, for now, the second state tile is also created on the fly by rotating four **Fill** quarter tiles.



So again, for now, in every single dungeon brush, one of your state tiles has conventional tiling, and the second one has to have radial tiling. These are its properties compared with the **Wall** ones.

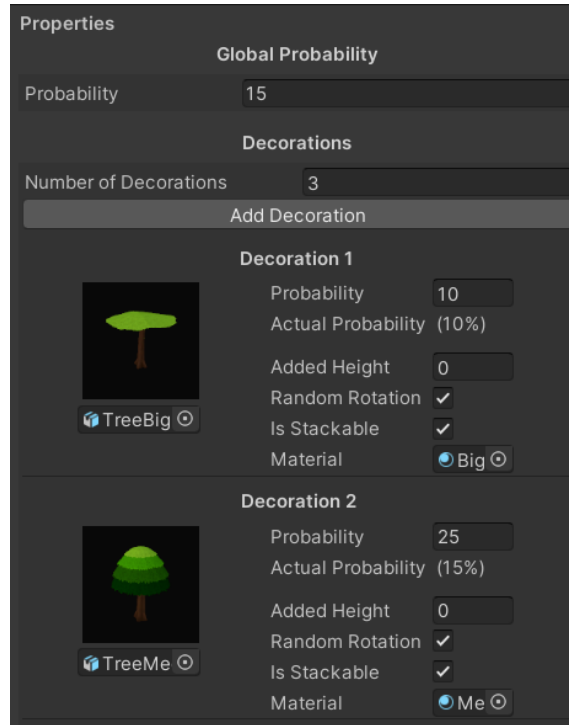
Wall Quarter Meshes	Ground Mesh
The pivot point of the mesh must be in (0, 0, 0)	Even though this mesh will not be rotated, its pivot point must also be in (0, 0, 0)
The actual mesh can only be in the Unity X+, Z+ quarter	Its recommended for this mesh to be a standard 1 x 1 plane.
The mesh X, Z dimensions should be 0.5 x 0.5 maximum	The mesh X, Z dimensions must be 1 x 1
The meshes had to have radial symmetry	The mesh and its material must be tileable

Ground Material:

The material is applied to the chunk combination of the ground mesh.

- **Dungeon Decoration**

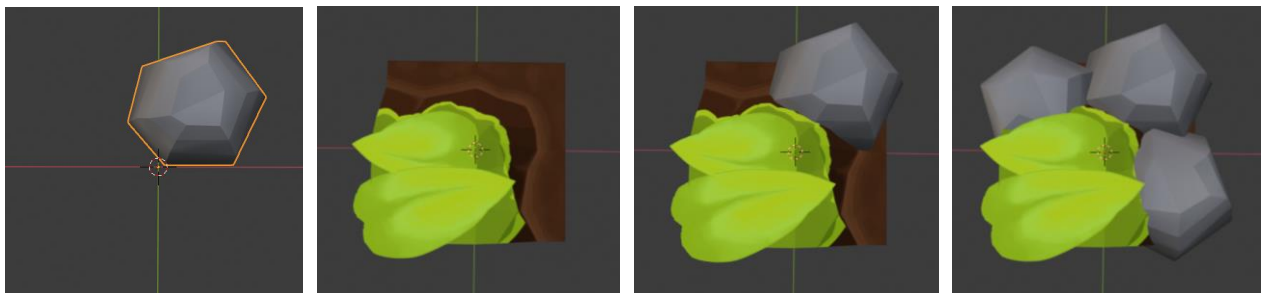
The **Dungeon Decorations** are a collection of **GameObjects** and spawn properties that define how they will be instantiated, or mesh combined in the **Dungeon**. You can use these sets to add more life to your created Dungeons.



If you choose to use these Decorations, the Dungeon Generator will go through every **Dungeon Tile** and try to add a Decoration to it. You can also assign a **Dungeon Decoration** to a specific tile type.

For now, you can set fixed decorations for both **State Tiles** (Wall Tiles and Ground Tiles), and the **Transition Tiles** (Corner Tiles).

- The **Wall** and **Ground** decorations will be spawned in the center of the 1 x 1 tile.
- The **Corner** decorations will try to spawn in each quarter tile and must have the same specifications as the quarter blocks of a **Dungeon Brush**.



Dungeon Decoration Properties

Global Probability



Global Probability	
Probability	15

Probability:

The probability of even spawning the set decorations. The **Dungeon Generator** will generate a number between 0 and 100. If that number is lower than the set here, a decoration will be generated.

Decorations

You can change the set limits for the Decorations Properties at the top of the **DungeonDecoration** script.

Decorations	
Number of Decorations	3
Add Decoration	
Decoration 1	
	Probability 10
	Actual Probability (10%)
	Added Height 0
	Random Rotation <input checked="" type="checkbox"/>
	Is Stackable <input checked="" type="checkbox"/>
	Material 

Number of Decorations:

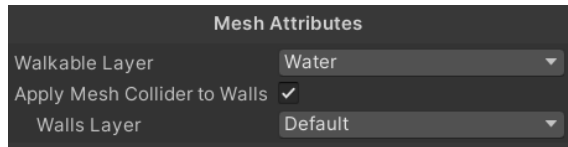
The number of decorations in this set. You can add a new decoration by pressing the **Add Decoration** button, or you can set the specific number in the options box. *(Limited to a max of 8)*

Decoration Properties:

- **Mesh:** The left box is for the decoration prefab.
- **Probability:** A representation of the overall probability of the Dungeon Decoration.
- **Actual Probability:** The actual probability for this item to be generated. The probability of this item spawning is the probability of this decoration minus the previous one.
- **Added Height:** Height to be added to this decoration.
- **Random Rotation:** If checked, when generated, the decoration will be randomly rotated on the Y axis. Otherwise, the rotation of the prefab will be used.
- **Is Stackable:** If checked, the mesh of this decoration will be combined with similar ones to improve performance, otherwise these decorations will be individually instantiated. All other object components will also be ignored when the decoration is generated.
- **Material:** The decoration material, only necessary for stackable decorations.

Dungeon Properties

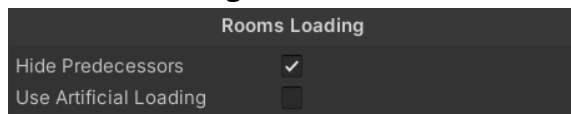
Mesh Attributes



The Dungeon Generator will build the Dungeon chunks into meshes; you can decide the properties of these meshes.

- **Walkable Layer:** To help you with in-game collisions, a Mesh Collider and this Layer will be applied to all the generated Dungeon Ground.
- **Apply Mesh Collider to Walls:** If checked, a mesh collider will be added to all Dungeon Walls. This option is a default for the Dungeon Ground.
- **Walls Layer:** The layer to which all Dungeon Walls will be assigned.

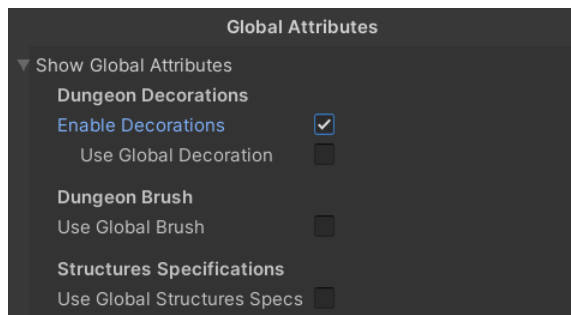
Rooms Loading



The Dungeon Generator will load and unload, build, and destroy Dungeon Rooms as needed, and you can specify certain aspects of this. You can learn more about it in the **Dungeon Loader** section.

- **Hide Predecessors:** If checked, the rooms no longer visible on the screen will be hidden.
- **Use Artificial Loading:** If checked, the user will define the proper load and distribution of the Dungeon Rooms, otherwise, the Dungeon Generator will do it independently.

Global Attributes

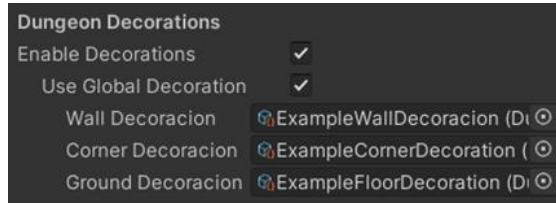


The Dungeons can be divided into segments. Each segment can have its own Dungeon Brush, Dungeon Decorations and Structures Specifications. The Global Attributes section will help you set these attributes to all segments of your Dungeon.

Show Global Attributes:

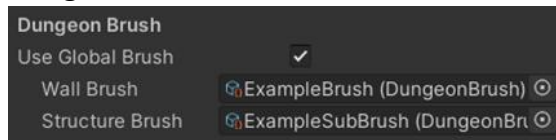
A fold-out label that lets you see the global attributes of the Dungeon.

Dungeon Decorations



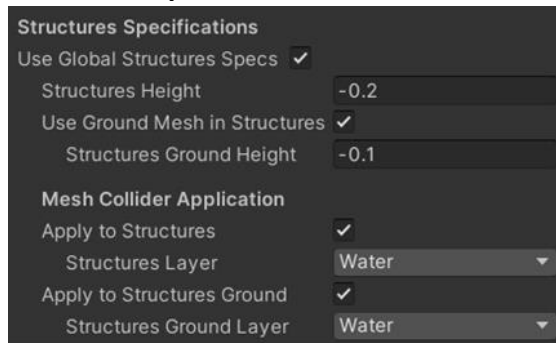
- **Enable Decorations:** If checked, you can add decorations to the Dungeon.
- **Use Global Decoration:** If checked, these decorations will be assigned to all dungeon segments.
 - **Wall Decoration:** This decoration will be placed on the Wall Tiles.
 - **Corner Decoration:** This decoration will be placed on the Transition Tiles.
 - **Ground Decoration:** This decoration will be placed on the Ground Tiles.

Dungeon Brush



- **Use Global Brush:** If checked, these brushes will be assigned to all dungeon segments.
 - **Wall Brush:** This brush will be applied to the Walls and Ground of the Dungeon Rooms.
 - **Structure Brush:** This brush will be applied to the Structures of the Dungeon Rooms.

Structures Specifications



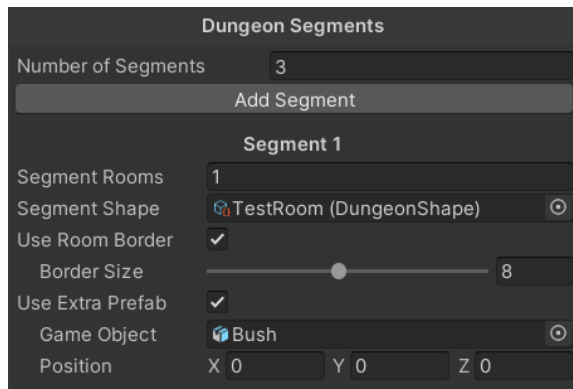
- **Use Global Structures Specs:** These specifications will be applied to all dungeon structures if checked.
 - **Structures Height:** Height to be added to the structures.
 - **Use Ground Mesh in Structures:** If checked, the Ground part of the structure brush will also be added to the structures tiles. You can see a use for this function in the Demos.
 - **Structures Ground Height:** Height to be added to the structure's ground.

Mesh Collider Application

- **Apply to Structures:** If checked, a mesh collider will be applied to the dungeon structures.
- **Structure Layer:** The GameObject layer that will be applied to the dungeon structures.
- **Apply to Structures Ground:** If checked, a mesh collider will also be applied to the ground.
- **Structures Ground Layer:** The GameObject layer that will be applied to the structure's ground.

Dungeon Segments

You can change the set limits for the Dungeon Segments Properties at the top of the **Dungeon** script.

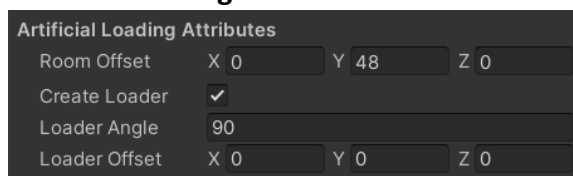


Number of Segments:

The number of segments of this Dungeon. You can add a new segment by pressing the **Add Segment** button, or you can set the specific number in the options box. *(Limited to a max of 32)*

- **Segment Rooms:** The number of rooms in this segment. *(Limited to a max of 64)*
- **Segment Shape:** The Dungeon Shape for these segment rooms.
- **Use Room Border:** The rooms of this segment will have a smart border, very useful to avoid showing the raw edges of the room.
 - **Border Size:** The size, in tiles, of the room border *(Limited to the Segment Shape Chuck Size)*
- **Use Extra Prefab:** Will instantiate a given GameObject in each room of this segment.
 - **Game Object:** The object to be instantiated.
 - **Position:** Position of the extra prefab relative to the room center.

Artificial Loading Attributes



These options will only be displayed if the Dungeon is set to **Use Artificial Loading**. You can learn more about how it works in the **Dungeon Loader** section, or you can see it in use at the Demo Dungeons.

- **Room Offset:** The displacement a room of this segment has relative to the previous one. To clarify, this movement moves the center of this room from the center of the previous one.
- **Create Loader:** Will instantiate a custom loader in all the rooms of the segment.
- **Loader Angle:** The Y angle at which the loader will be.
- **Loader Offset:** The displacement the loader will have relative to the room center.

If the Dungeon is not using **Global Attributes**, each Dungeon Segment will ask you for its respective Dungeon Brush, Dungeon Decorations and/or Structure Specifications.

Dungeon Segments

Number of Segments3

Add Segment

Segment 1

Segment Rooms1

Segment ShapeTestRoom (DungeonShape)

Use Room Border☒

Border Size1

Use Extra Prefab☐

Brushes

Wall BrushExampleBrush (DungeonBrush)

Structure BrushExampleSubBrush (DungeonBr)

Decoration

Wall DecorationExampleWallDecoracion (Dung)

Corner DecorationExampleCornerDecoration (Dur)

Ground DecorationExampleFloorDecoration (Dung)

Structures Specifications

Structures Height-0.2

Use Ground Mesh in Structures☒

Structures Ground Height-0.1

Mesh Collider Application

Apply to Structures☒

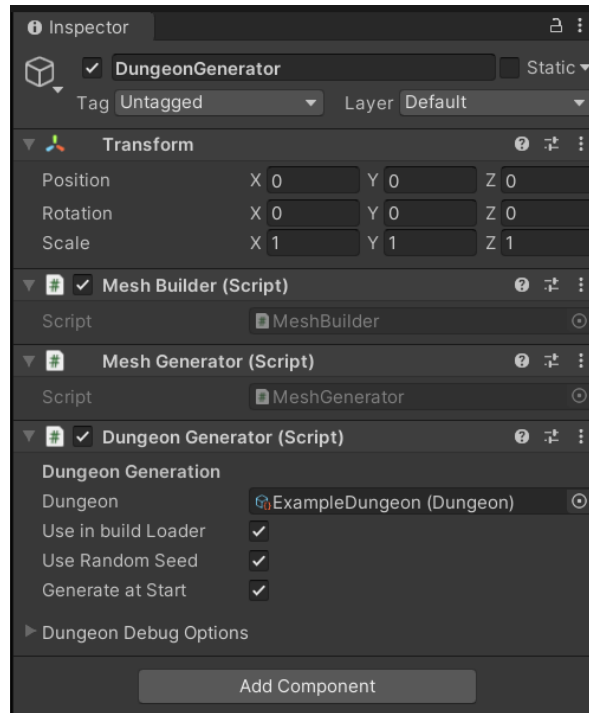
Structures LayerWater

Apply to Structures Ground☒

Structures Ground LayerWater

Dungeon Generator

The **Dungeon Generator** script is **Advanced Dungeon Generator's** heart and soul, as it is the tool in charge of creating the dungeons.

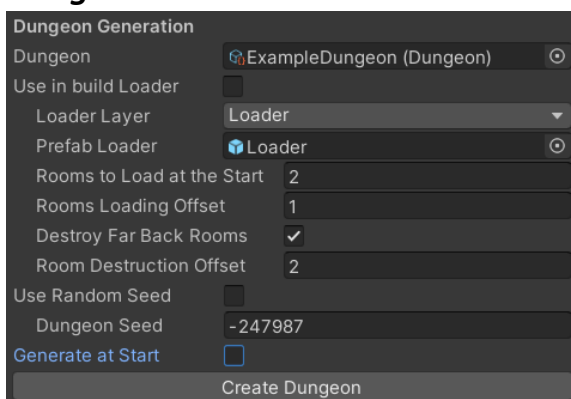


To begin creating a dungeon, create a GameObject and add the **Dungeon Generator** script to it. You will notice that two other scripts will be added as well. The **Mesh Builder** script and the **Mesh Generator** script are support construction scripts and need to be present in the same GameObject.

Note **there can only be one Dungeon Generator for Scene**. If you have multiple GameObjects with the Dungeon Generator scripts, **then the extras will be destroyed at runtime**.

Dungeon Generator Properties

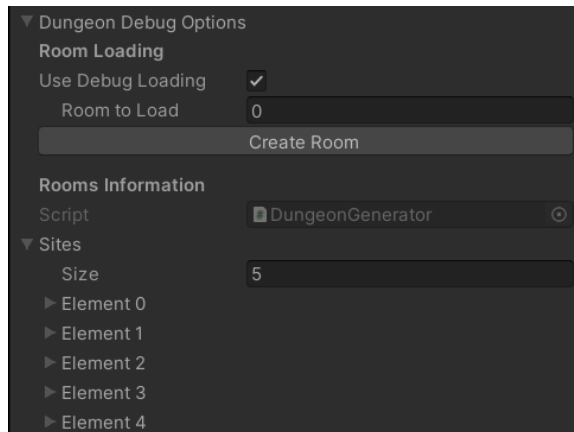
Dungeon Generation



- **Dungeon:** The Dungeon to be build.
- **Use in build Loader:** If checked, the Dungeon Generator will create the standard loaders of the Dungeon on its own. Otherwise, the user will specify the prefab loader to use.
 - **Loader Layer:** The layer assigned to all Dungeon Loaders.
 - **Prefab Loader:** The loader instantiated at every room. This prefab needs to have the Dungeon Loader script attach, you can learn more about it in the Dungeon Loader section.
 - **Rooms to Load at the Start:** The number of rooms loaded at the start of the dungeon creation (2 by default).
 - **Rooms Loading Offset:** When the loader triggers, the room to load will be the current one plus this number (1 by default).
 - **Destroy Far Back Rooms:** Rooms too far back will be destroyed (True by default).
 - **Room Destruction Offset:** When the loader triggers, the room to destroy will be the current one minus the Room Loading Offset and this number (2 by default).
- **Use Random Seed:** If checked, the Dungeon Generator will use a random seed when creating the Dungeon.
 - **Dungeon Seed:** The seed being or to be used at dungeon generation.
- **Generate at Start:** If checked, the Dungeon will be generated at the start of the game.
 - **Create Dungeon:** Clicking this button will generate the Dungeon, you can call this function in the Dungeon Generator script by the ***DungeonStartTrigger*** name.

```
DungeonGenerator.Instance.DungeonStartTrigger();
```

Dungeon Debug Options



Room Loading

- **Use Debug Loading:** Gives you the option to load a specific room from the Dungeon.
- **Room to Load:** The ID of the room you want to load.
- **Create Room:** Clicking this button will generate the specified room.

Rooms Information

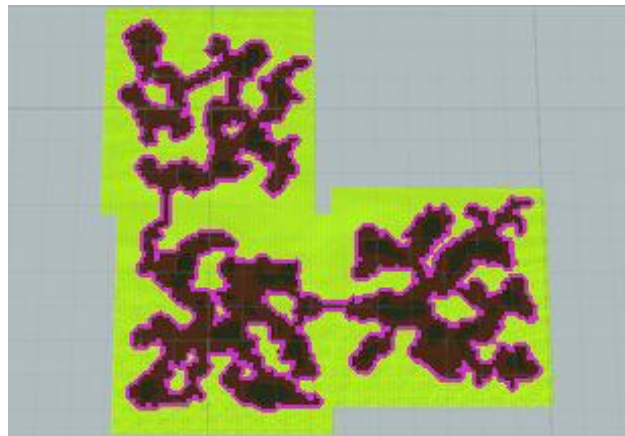
- **Sites:** A visual representation of the generation values of all the dungeons rooms.

Dungeon Loader

Dungeon Rooms are created as isolated and independent sets of connected chambers.

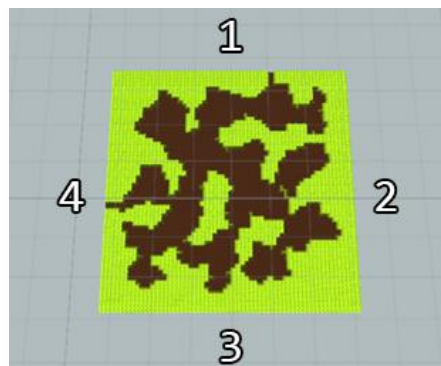


But they need to be connected to form an actual Dungeon

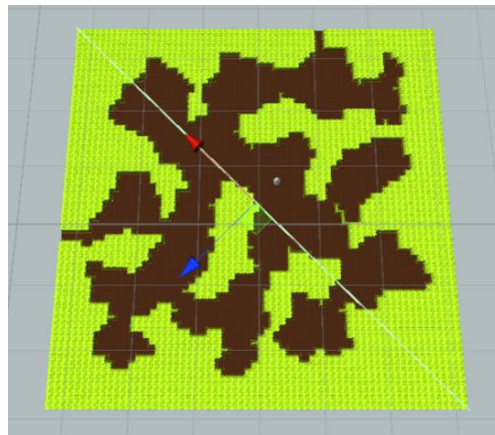


So, before the **Dungeon Generator** starts the in-game construction of the **Dungeon Rooms**, it has to create its entrances and exits. The Dungeon Rooms can be generated as squares or rectangles, so the Dungeon Generator will choose one side to create an entrance and a different one to create an exit.

For reference, the top side of the room takes the value of < 1 >, and then it increases clockwise.

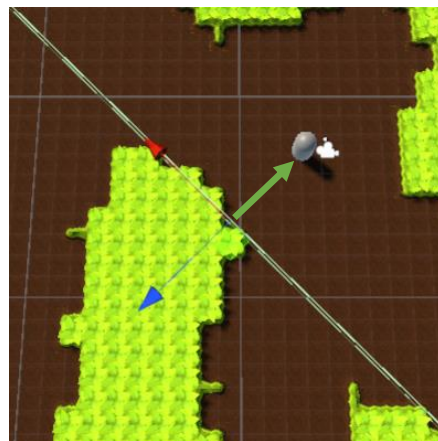


As you may have seen before, the **Dungeon Generator** will only create some Dungeon Rooms when it is called. The rest will be generated dynamically. To ensure this, the generator will create **Loaders**, room-size loaders between the entrance and the exit of the Dungeon Rooms.



These **Loaders** are **Box Colliders** that will use their position and the position of their triggers to decide which rooms are to be loaded and which are to be unloaded.

The rotation of these colliders is important, the loaders are **always looking towards the entrance** of the rooms, and they are always on the lookout for other colliders trying to go through them.

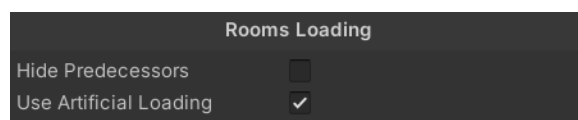


The dot product of these two vectors is the value that will determine the rooms to load.

If these vectors are **opposite**, the room following the exit will be loaded, and the room preceding the entrance will be unloaded.

If these vectors are **parallel**, the room preceding the entrance will be loaded, and the room following the exit will be unloaded.

Take this into account when creating **Artificially Loaded Dungeons**.



Making the entrances and exits of the rooms, placing rooms one after the other, and placing the room loaders in the right way is done procedurally in the standard dungeons. But, in Artificially Loaded Dungeons, the user must do these.

In Artificially Loaded Dungeons:

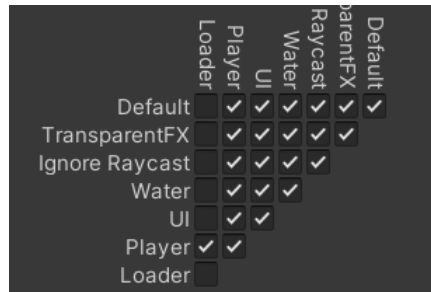
- The entrances and exits of the rooms should also be mapped in the **Mask Texture** of your **Dungeon Shapes**, whether those shapes are artificial or not.
- The rooms offset is calculated by the user and set at every Dungeon Segment in the **Artificial Loading Attributes**.
- The loader, its angle (the loader front, **the blue arrow**, should point to the entrance of the rooms), and its offset are calculated for each room by the user and set at every Dungeon Segment in the **Artificial Loading Attributes**.

This new option complicates the creation of dungeons, but don't worry, it will allow you to create amazing things, and you have a demonstration of how to do it in the **Demo Dungeons**.

Loaders Behavior

The **Loaders** will react to any collider that tries to go through it, you can select the layer assigned to all Dungeon Loaders in the **Dungeon Generator**, or set a custom Loader.

I recommend you to put the Player in a specific layer and then modify the **Unity Layer Collision Matrix** (Edit > Project Settings > Physics) so they can only interact with each other.

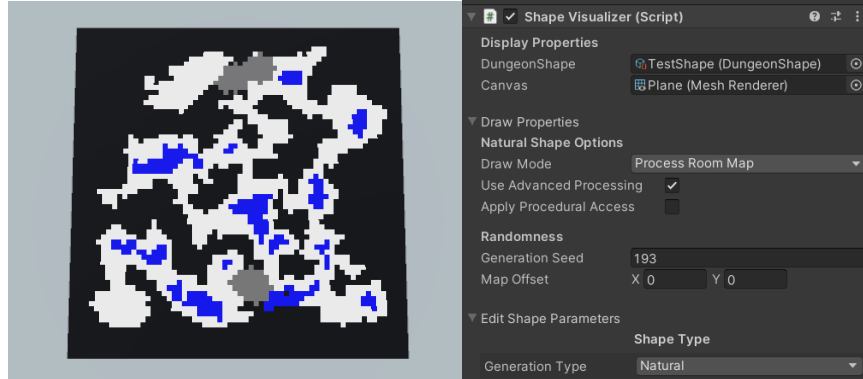
A screenshot of the Unity Layer Collision Matrix. It shows a grid of checkboxes for interactions between different layers. The layers listed on both the horizontal and vertical axes are: Default, TransparentFX, Ignore Raycast, Water, UI, Player, and Loader. The 'Loader' layer has a checkmark in the 'Player' column, indicating it interacts with the Player layer. The 'Player' layer has a checkmark in the 'Loader' column, indicating it interacts with the Loader layer. All other interactions are unchecked.

	Default	TransparentFX	Ignore Raycast	Water	UI	Player	Loader
Default	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TransparentFX	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ignore Raycast	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Water	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
UI	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Player	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Loader	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

If you don't want to use the **Dungeon Generator** in-built loader system you can add your own loaders (There is an example Loader prefab in the Demos), by doing this you can also change the default loading settings and set how many rooms will be generated at the start, the rooms unloading offset, if the rooms too far back can be destroyed, etc.

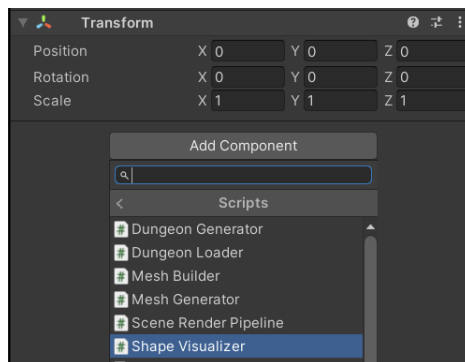
Shape Visualizer

The **Shape Visualizer** script is an extra tool that lets you preview the rooms that a **Dungeon Shape** will generate.

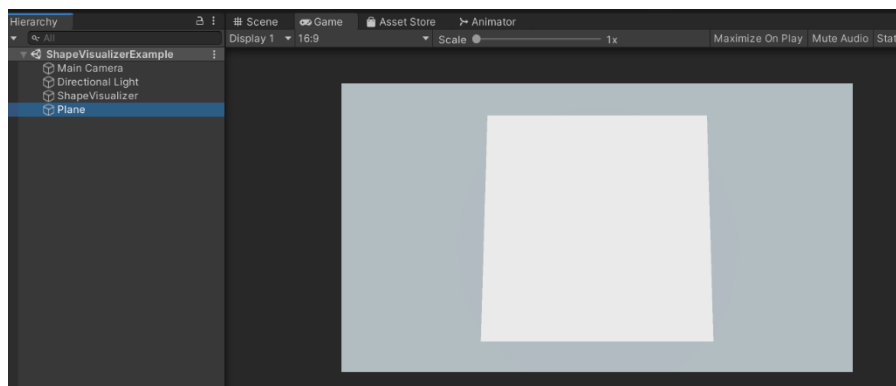


Setting the parameters of a Dungeon Shape can become a difficult task, especially if you don't have active feedback on how your changes affect the generation of the room. With the Shape Visualizer, you can see how your changes in the Dungeon Shapes affect their rooms on the fly.

To use the **Shape Visualizer** tool, create a GameObject and add the Shape Visualizer script.

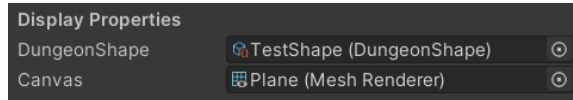


You will also need another GameObject to use as a canvas. Remember that this will change the texture of his material, so create a material only for visualization.



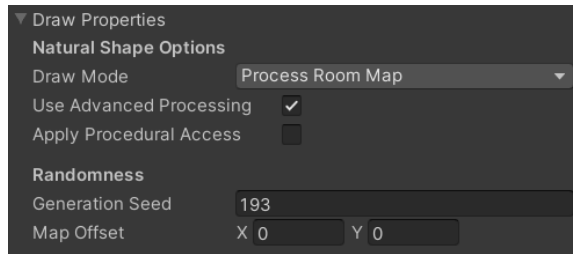
Shape Visualizer Properties

Display Properties



- **Dungeon Shape:** The Dungeon Shape that will be tested.
- **Canvas:** The GameObject on which the rooms will be drawn.

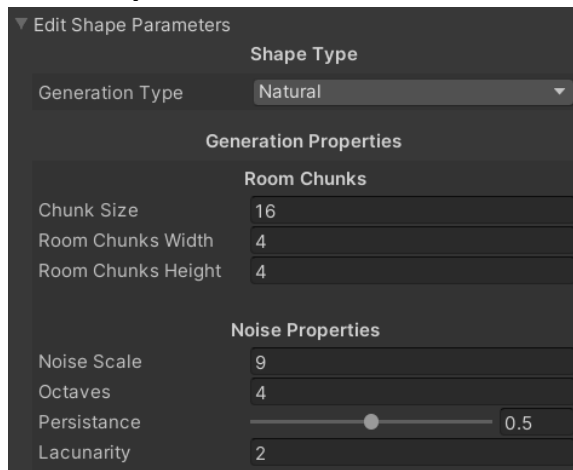
Draw Properties



The generation type of the Dungeon Shape will determine what drawing options will be available.

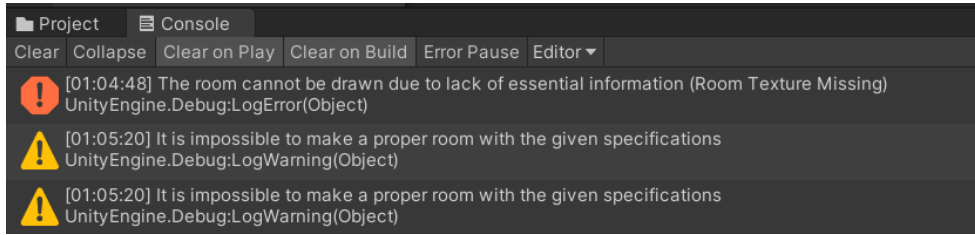
- **Draw Mode:** You can decide on a specific component of the rooms to draw. Depending on the Dungeon Shape, you can select between the Processed Room Map, the Basic Noise Map of the room, the Falloff Map of the rooms, and the image given for the Masking Process of an Artificial Shape.
- **Use Advanced Processing:** Will show the Post-processed room maps. A Processed room map will only show the raw Perlin Noise smoothing process, a Post-Processed room map will show the sub-rooms now cleaned and joined.
- **Apply Procedural Access:** Will apply random entrances and exits to the room maps.
- **Generation Seed:** The seed of the base Perlin Noise map.
- **Map Offset:** The offset of the base Perlin Noise map.

Edit Shape Parameters

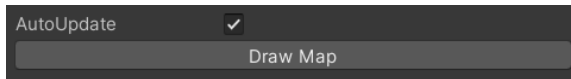


Here you can change all the properties of the selected Dungeon Shape.

The Shape Visualizer script will give you a custom error if the generator cannot create a room with the given specifications.



Update Options

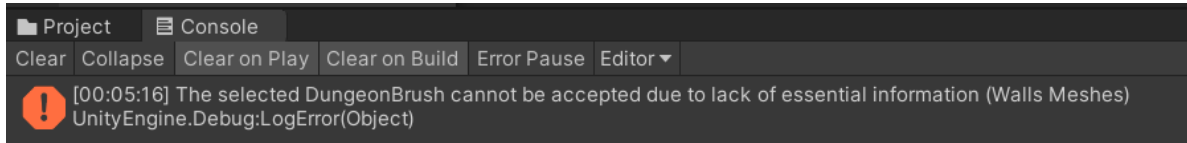


- **Auto Update:** If checked, the canvas will be updated every time you change the Shape Parameters
- **Draw Map:** The canvas will be updated when this button is pressed.

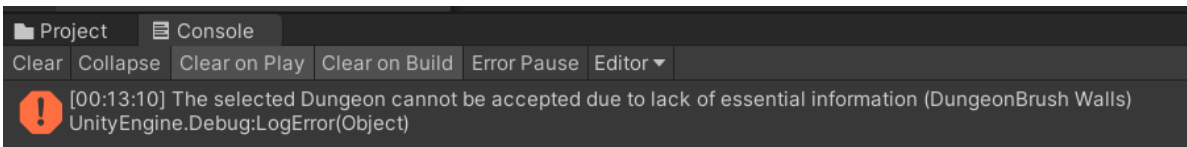
Security Check

Security Check is a small script that will help you create your Dungeons. It will behave as a background element in all your Dungeon Components and **will prevent you** from adding key elements that **lack essential information**.

For example, when creating a Dungeon, it will prevent you from adding any Dungeon Brush that lacks the Meshes and/or Materials.



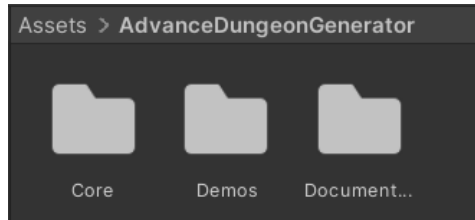
It also works with the Dungeon Generator, so it will not accept a Dungeon that lacks Dungeon Brushes or Dungeon Shapes.



A quick review of the Security Check script will help you understand which elements are considered key.

Demos

Advanced Dungeon Generator comes with several folders. In one of them is this Documentation. There is also a folder full of Demonstrations. The only folder required for Advanced Dungeon Generator to work is the **Core folder**; the rest can be safely deleted if you don't want them.



The **Demos folder** comes with a set of extra prefabs, materials, textures, and scripts that will help you understand how the Dungeons would work in a real scenario.

This folder also includes examples of Dungeons and Dungeon Components, example Scenes for Natural Dungeons, Artificial Dungeons, the Shape Visualizer, and a full-on Example Dungeon.

