QUBIC

User Guidance

by Star Twinkle

Installation

Built-In Project

- Install the package

URP/HDRP Project

- Install the package.
- Convert materials in folder Qubic/Demo/Materials to needed pipeline.

Quick Start

The easiest way to study Qubic and explore how it works - open the demo scene “01 - Basic” and play with it.

To use Qubic from scratch and for your prefabs, do the following:

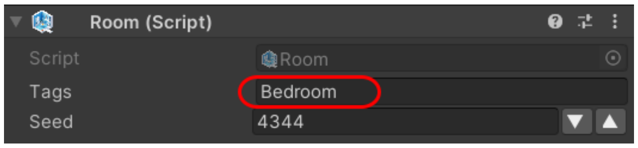
- Open Prefab Manager - via the main menu “Tools/Qubic/Prefab Manager”.
- In Prefab Manager create a new prefab database - the button “New Database”. Set fields Cell Size and Cell Height in the created database. To do it, select the database scriptable object in the Project window and set the fields in the Inspector window.
- Open a demo scene that contains your prefabs (preferable) or just create a new scene.
- Select prefabs in the demo scene or select prefabs in the Project window. For each prefab assign template in Prefab Manager. More details see in Prefab Templates.
- In parallel, create QubicBuilder on scene via the main menu “Tools/Qubic/Create Qubic Builder”
- Assign created database to PrefabDatabase field of QubicBuilder.
- As you add prefabs to the database, QubicBuilder will automatically rebuild and you will see how your prefabs appear in the building.
- You can adjust your prefabs selecting them inside Qubic: move/rotate/change bounds.

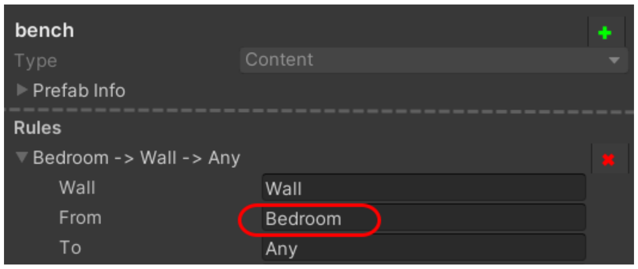
You can watch the whole process in the video tutorial: https://youtu.be/a1aMIOG8GsU

FAQ and Tips

❓How to create a room with different furniture?

Set specific tag in the Tags field of Room, and assign same tag to From field of prefab rules:


Room (Script)
Script: Room
Tags: Bedroom
Seed: 4344


bench
Type: Content
Prefab Info
Rules
Bedroom -> Wall -> Any
Wall: Wall
From: Bedroom
To: Any

As result the content prefab will be spawned in rooms with a specified tag.

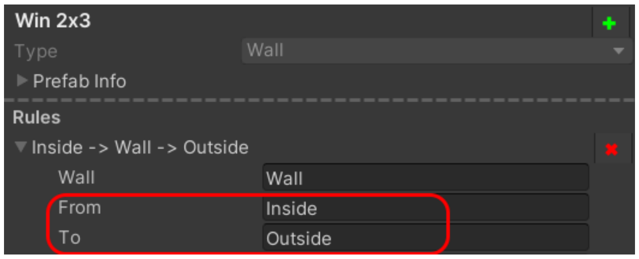
❓How to make room without windows (closet for example)?

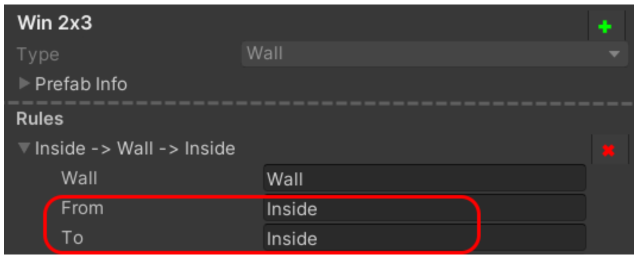
Simplest way - just use the Closet template when you create the room.

In common case you need to increase the priority of wall prefabs for specific room style. As a result all possible edges will be taken by walls and windows can not be spawned there.

❓How to avoid windows appearing on the outside/inside of a building?

Well, the conditions for window spawning are completely determined by the rules for windows. If you want to spawn windows from the inside out, then create a rule for this. Similarly in the opposite direction.

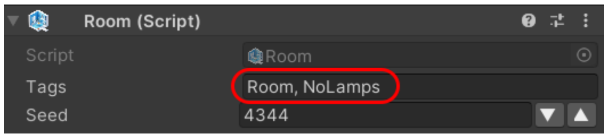

Win 2x3
Type: Wall
Prefab Info
Rules
Inside -> Wall -> Outside
Wall: Wall
From: Inside
To: Outside

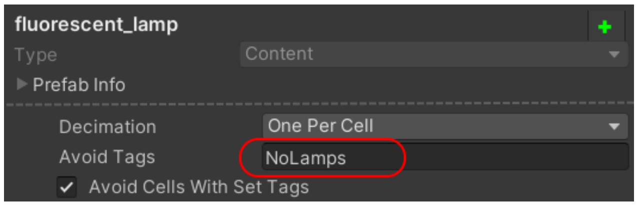

Win 2x3
Type: Wall
Prefab Info
Rules
Inside -> Wall -> Inside
Wall: Wall
From: Inside
To: Inside

Alternative way - increase/decrease priority of walls/windows between Inside and Outside cells (or Inside-Inside).
By default, if you use templates, the system makes windows outside of the building and avoids inside. Of course, if you're making a dungeon or a maze, you need to do the opposite.

❓How to switch off lighting/lamps in specific room?

In short - via specific tag and tag avoidance in prefabs.
For example, add tag NoLamps in the Tags field of Room, and avoid this tag in all lamp prefabs. As result - lamps will not be spawned in this room.


Room (Script)
Script: Room
Tags: Room, NoLamps
Seed: 4344


fluorescent_lamp
Type: Content
Prefab Info
Decimation: One Per Cell
Avoid Tags: NoLamps
✓ Avoid Cells With Set Tags

Same approach you can use to hide specific prefabs in the specific room.

❓How to avoid to spawn same furniture in neighbor cells?

There are several ways to achieve this. Each of them solves the problem a little differently. Simplest way - increase Decimation property of Prefab. Alternatively - decrease Chance in prefab rules.
And the most generic and powerful way - use SetTags, SetTagsRadius and AvoidCellsWithSetTags feature. More detailed: see chapter Content Collision Avoidance.

❓Prefab is spawned with wrong rotation, how to fix?

The Qubic grabs rotation from the prefab's Transform component. So, to change rotation, just change Rotation in the original prefab's transform.
Also you can change rotation around the Y axis, snapped to 90 degrees, directly in Prefab properties or on Scene View (button Rotation, or PrefabInfo in Inspector).

❓How to create a basement, or several underground levels?

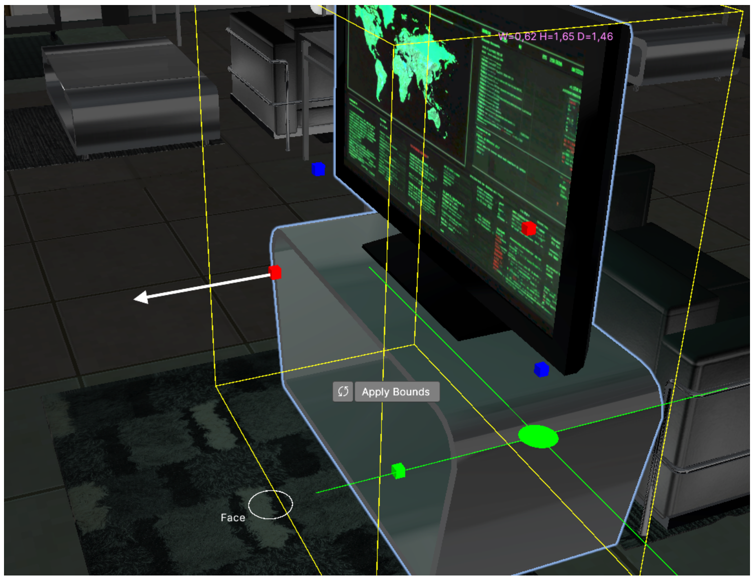
Well, just lower the starting point of the room below the parent QubicBuilder Y position. Qubic always counts floors relative to the position of QubicBuilder. So if the room starts below zero on the local Y axis - then those floors will be considered a basement or dungeon and you can set up prefabs for underground levels (via negative Levels property of prefab).

❓How can I prevent content from spawning where I placed my object?

If your object has a collider, use the property ContentBlockingLayers of PrefabSpawner. Another approach is to create a virtual room that spawns a high-priority empty object. This object will take up the entire cell and prevent other spawners from creating content there. You can also assign a tag to a room that does not have prefab content defined.

❓How to prevent to spawn some content in front of a picture or TV-set?

Increase bounds of TV-set in front direction. It forces the system to avoid this space to spawn other prefabs.



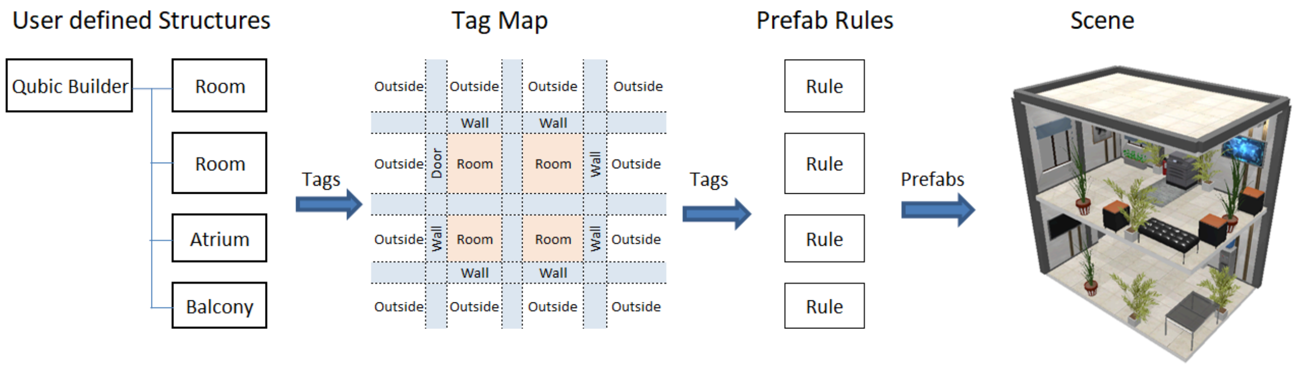
Overall Information

How the system works

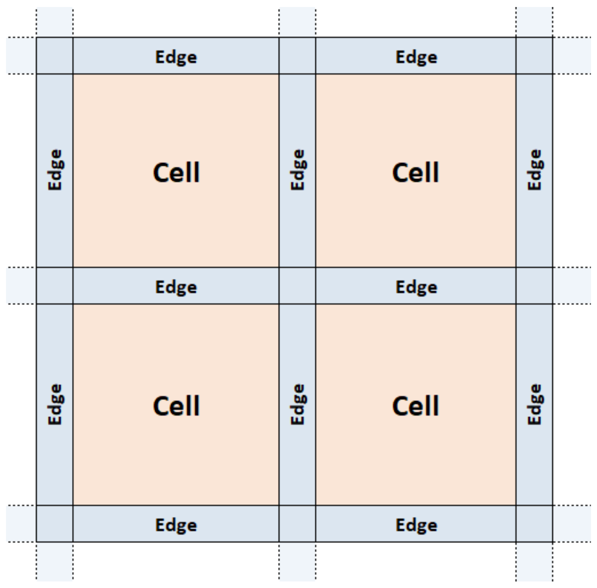
The construction of a building occurs in three main stages:

1. User defined structures (rooms, stairs, balconies, etc.) capture map cells, competing with each other.
2. Captured cells and their boundaries are marked with tags (Room, Closet, Wall, Floor, Steps, etc.) on the tag map.

3. Prefab rules check the tags on the map and place their prefabs where the conditions are met.



The map is a three-dimensional grid, with cells of different types. Large cells are called Cells, and the cells between them are called Edges.

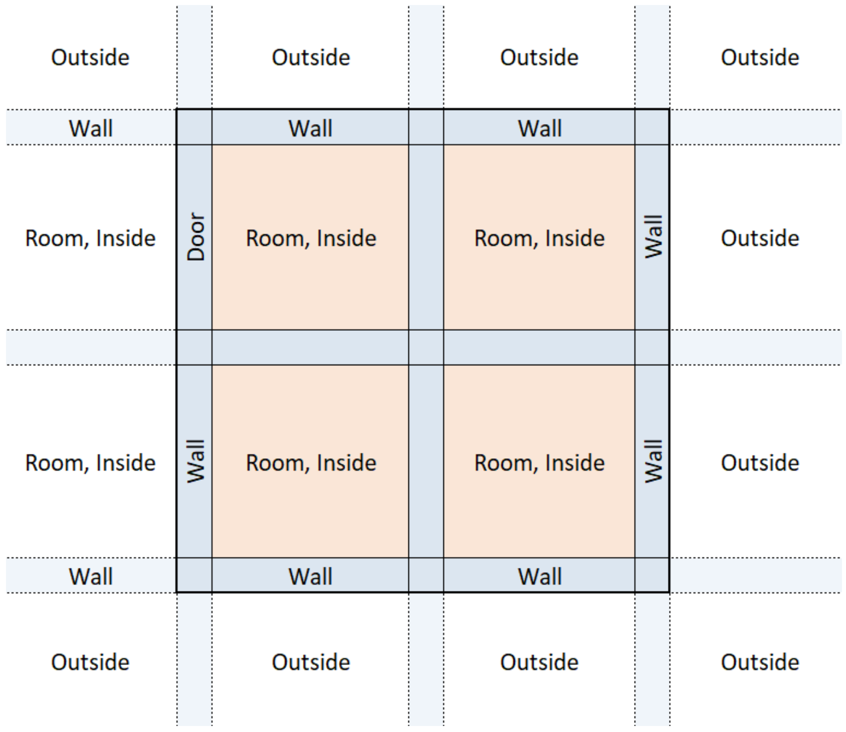


Note, the figure shows a flat map, but actually it is three-dimensional. The Cells are surrounded by Edges also at the top and bottom.

Cells of the **Cell** type contain tags with information about the room that captured these cells. Usually this is the name of the room style that the user specifies in the Room object. For example, Room, Closet, GlassRoom, Bedroom, and so on.
Each Cell can contain several tags at the same time.

Edge type cells contain information about the type of structural element that should be spawned in a given location. This is a set of standard tags: Wall, Door, Floor, Parapet, etc.
Usually, Edge contains exactly one tag (or is empty).

Example of map with marked tags:



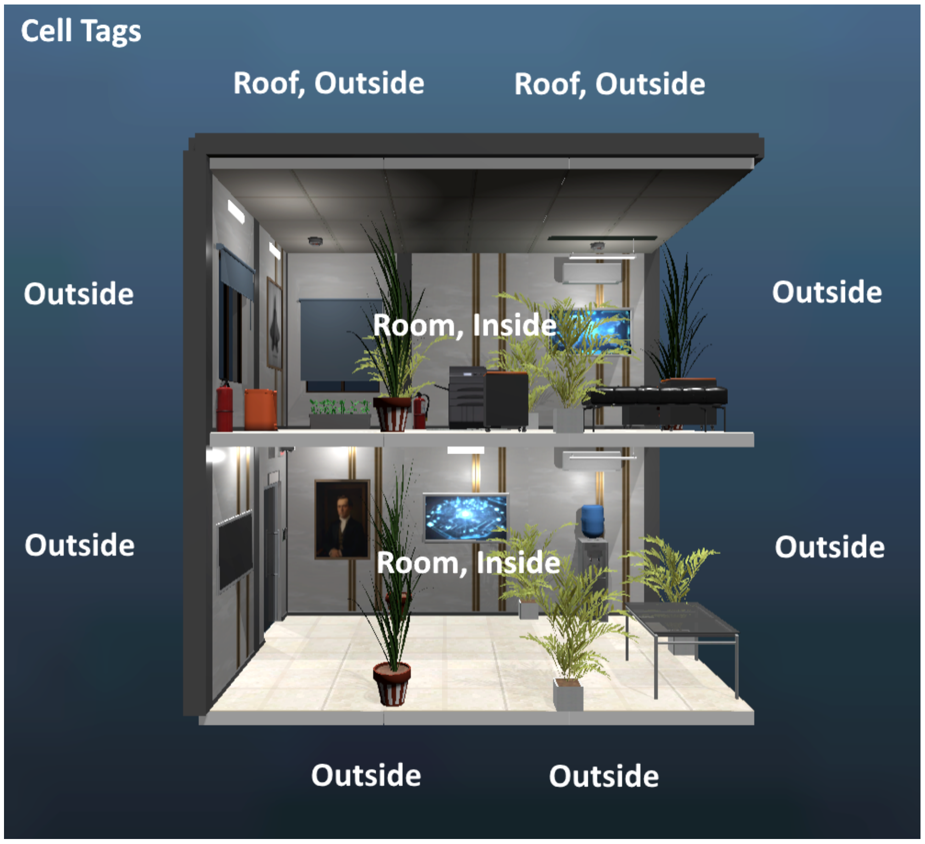
Standard Tags

Below is a list of system-built tags that are generated on the map by various structures - rooms, stairs, atriums, balconies, and so on.
You can use these tags in object spawn rules.
These tags are spawned automatically by the system, however, this is not an exhaustive list, because you can use your own custom tags for your specific markup.

Note: You should not be worried by the number and variety of tags. For basic use, you do not need to know the tags, because the prefab templates that you get out of the box already contain the predefined rules for all standard tags. The meaning of the tags is needed just for more fine-tuning the spawn.

Tags are divided by their place of origin. Cell tags spawn in cells, Wall tags - in the edge of cells, and Floor tags - in the edge below and above cells.
Note that one cell or edge can contain several tags at the same time. For example, Roof can spawn at the same time as Outside.

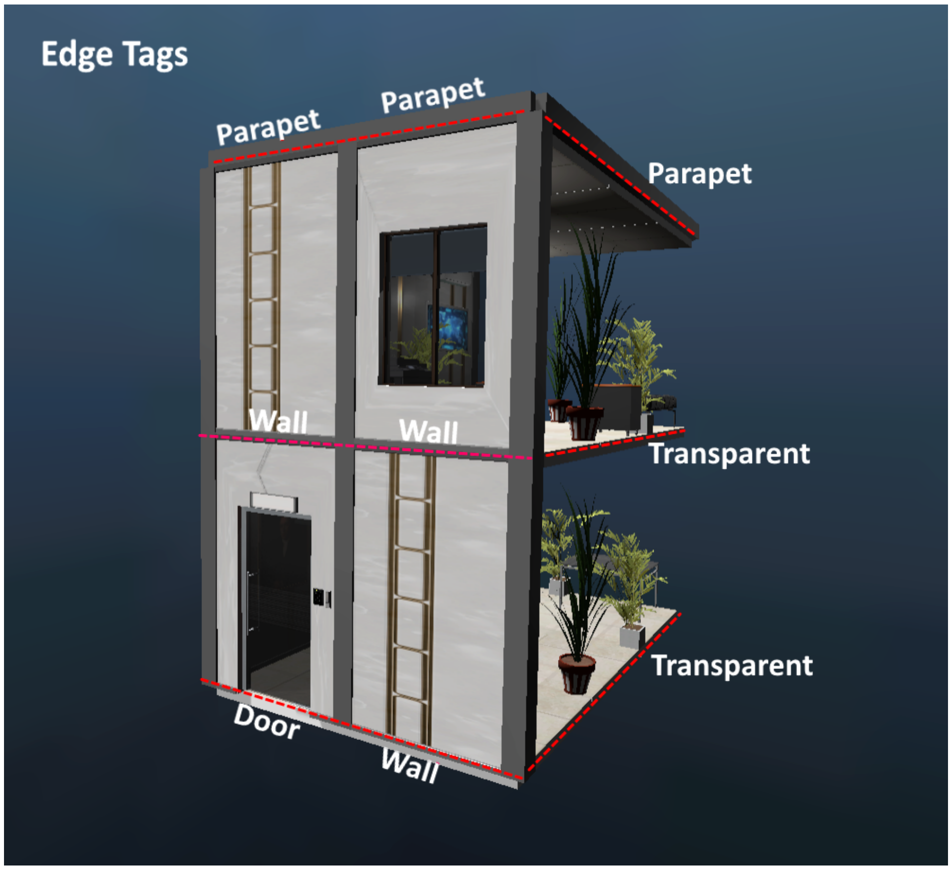
Cell standard tags



Note, on the figure shown tag **Room**, it is not a standard tag, but this tag usually spawned by Room structure and means regular room style.

- **Outside** - This tag automatically marks cells that are outside the building, outside any of the rooms.
  Note that the Outside tag can be forced to be spawned by some structures (such as a Fence or Bridge) to mark its area as outside of the building.
  Also, Outside tags sometimes go with other tags, for example Roof. This is necessary so that the spawn rules perceive these cells as external and build external walls in neighboring cells.
- **Inside** - This tag marks cells that have a room, as opposed to Outside.
  Use the Inside and Outside tags to customize the construction of the internal and external walls of a building. These tags can also be used to customize the orientation of a wall, for example if the wall has different external and internal sides.
- **Impassible** - Some structures (such as Stairs) mark cells with the Impassible tag, indicating that these areas are not passable and doors cannot be created there.
- **Roof** - A cell above a building is automatically tagged with the Roof tag if the corresponding room allows the creation of a roof and ceiling. You can use this tag to spawn roof prefabs.

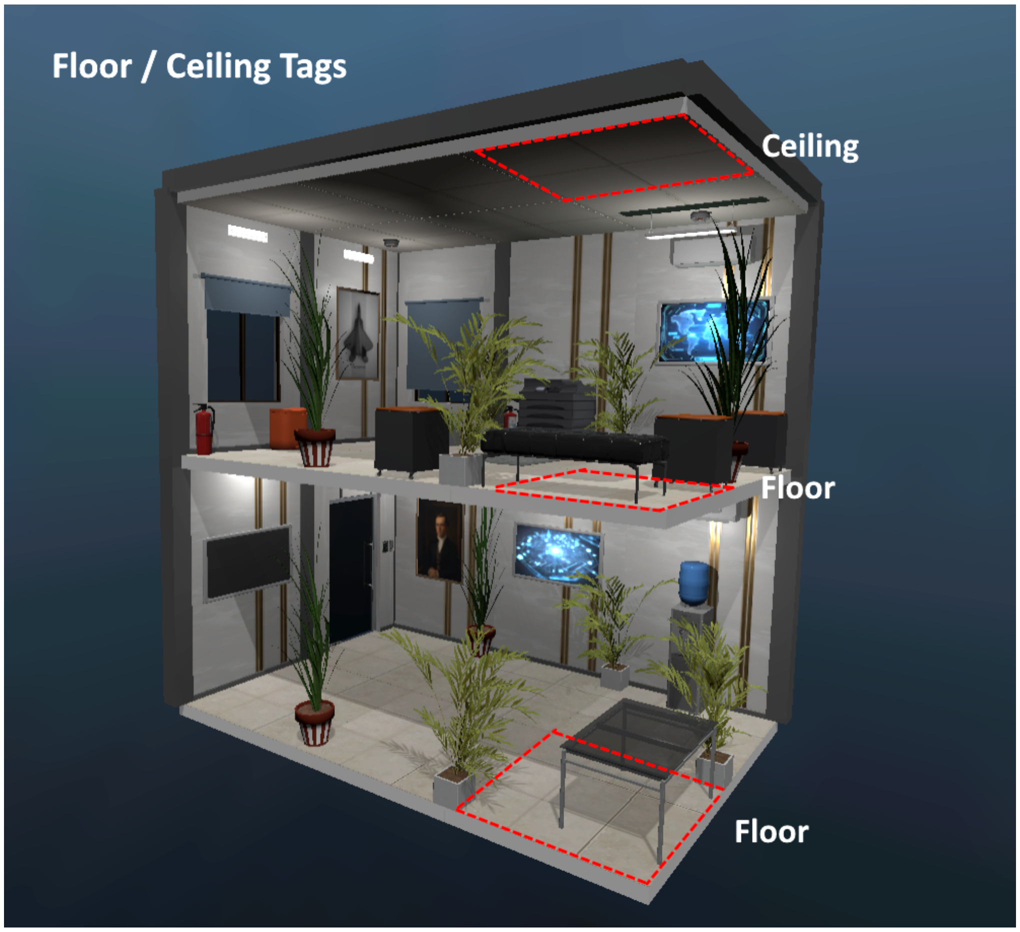
Wall / Edge standard tags



- **Wall** - A standard wall tag that spawns most structures (like Room) at its borders.
  Note that this tag does not necessarily mean a wall. It could also mean a window, an arch, or a rails, depending on which rule is triggered for that edge.
  The Wall tag is spawned simply to indicate the edge of the room, the specific kind of edge is determined by the spawn rules.
- **Steps** - This tag is spawned by the Stairs structure at the start of the stairs. This tag means that steps should be spawned at this location.
- **Passage** - This tag means an open passage. In meaning, this tag is the opposite of the Wall tag. The Passage tag is spawned by some structures (such as Stairs) that require the absence of a wall at some edge.
- **Door** - The tag indicates that a door needs to be spawned here. These tags are automatically placed by DoorsSpawner.
- **Parapet** - This is the edge type between the Roof and Outside cells. Here you need to spawn a small border that limits the roof.
- **Content** - This tag spawns inside the room as invisible walls along which content (furniture, lamps, carpets, etc.) needs to be spawned.
  Since the spawn rules only work with walls/edges, then for spawning content in the room interior you also need to generate virtual walls for the content.
  These tags are automatically placed by structures (Room mainly).

- **Transparent** - This tag means invisible wall, for isometric view. IsometricSpawner replaces tags like Wall with Transparent, in order to hide some of the walls in an isometric view. You can use this tag to spawn semi-transparent walls or to spawn content allowed to be placed along invisible isometric walls.

Floor / Ceiling standard tags



- **Floor** - This tag is spawned by many structures (like Room or Stairs) in the edges under the cell, and means that the floor prefab should be spawned here.
  If the structure has multiple levels, the Floor tag will spawn on all floors (dependent on settings) except the topmost edge, where the Ceiling tag will spawn.
- **Ceiling** - The Ceiling tag spawns as structures on the top of the last level of the building.

Generic tags

- **Any** - This is a virtual tag, meaning any non-empty tag. Use this tag if the specific tag is not important to you, and can be any. This tag does not spawn on the map, but is used only in the user interface.

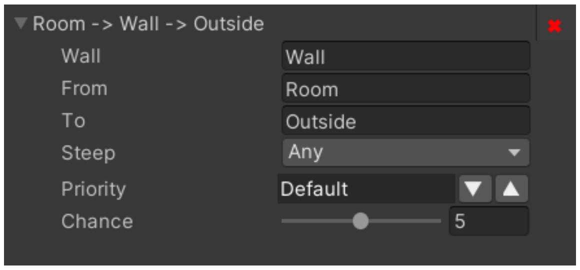
Qubic is a very flexible and extensible system. The tags is a general mechanism supported by the system. But the set of tags itself is not fixed and is determined by the user's rules. Therefore, you can create your own tags and your own spawn rules for these tags. Moreover, you can expand the logic of the system by adding your own fields to the rules and prefabs, adding your own scripts to implement the spawn logic, and creating your own spawners.

Rules

The rule is attached to a prefab and defines the conditions under which the prefab can spawn in a specific cell of the map.

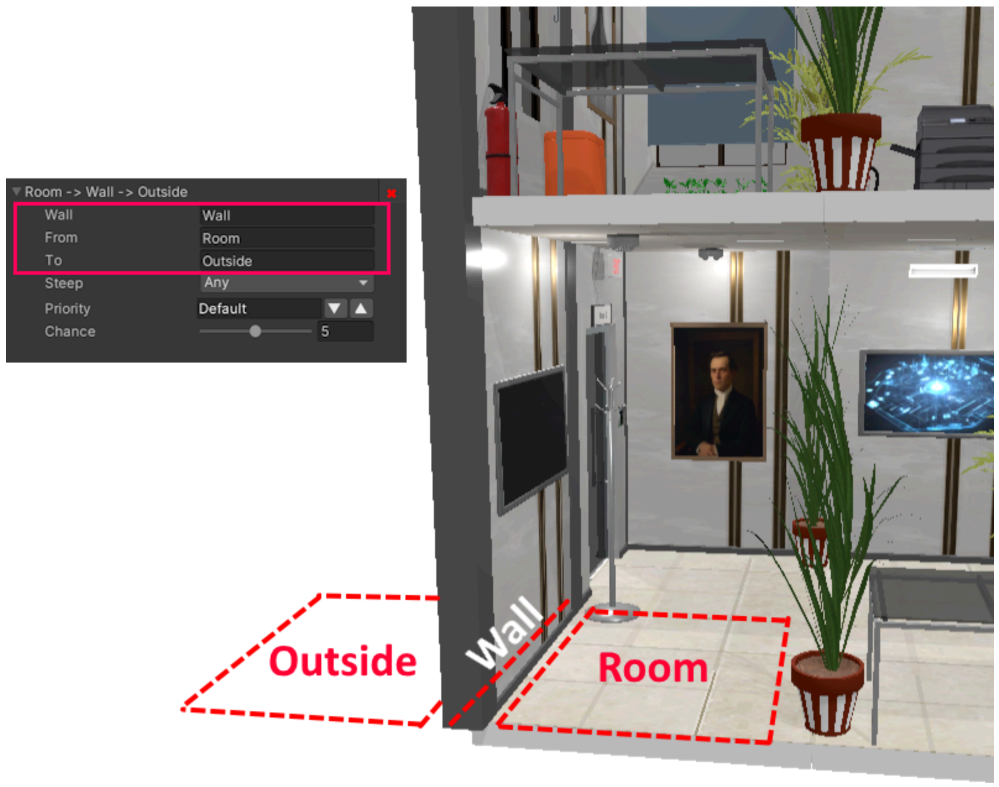
The rule contains three main conditions:

- **From** - The tag of the cell in which the object spawns.
  This tag is called “Style” sometimes in this document, because it usually contains the style of the owner's room.
- **Wall** - The tag of the wall near which the object spawns
- **To** - And the tag of the cell on the opposite side of the wall.


Room -> Wall -> Outside
Wall: Wall
From: Room
To: Outside
Steep: Any
Priority: Default
Chance: 5

For example, to spawn a prefab of the outer wall of a Room, the following rule can be used: Room->Wall->Outside

This means that this wall prefab can spawn in a cell with the Room tag, the wall tag must be Wall, and the opposite cell must be Outside.


Room -> Wall -> Outside

Wall	Wall
From	Room
To	Outside
Steep	Any
Priority	Default
Chance	5

The figure shows the rule to spawn a wall prefab.

Please note that the system spawns objects only based on walls. That is, the rules only work for Edge. If there is no wall/edge, the rules do not work.
Therefore, even for furniture created inside a room, the system creates invisible Content walls located inside the rooms.
Thus each rule for wall prefab works up to 4 times per cell - once for each of 4 non empty Edges of the cell.

Also, since the rule has a direction (defined by the From and To tags), the rule works twice for each Edge - from cell A to cell B, and the second time vice versa - from B to A.

Fields Wall, From and To of rule are multi-tag. This means that each field can contain several tags. For the rule to be triggered, it is enough for at least one of the tags of the multi-tag to match.
That is, if the Wall field in the rule has the form *Wall,Transparent* - this means that the rule will be triggered if the wall has the Wall tag or has the Transparent tag.

In addition to the tag condition, the rule also contains a condition on the floor height difference - **Steep**. This condition is usually used to spawn railings or parapets.

Also, rules have a **Priority** and a **Chance**.
Priority means the order in which the rules will be executed, and chance is the probability of the rule being executed among other rules with the same priority.

Note that the execution of prefab rules for walls and content are different. A wall can only contain one spawned object (i.e. just one Wall, Door, Window or Arch per edge). While content rules (i.e. furniture, curtains, carpets, etc.) can be triggered multiple times. Thus, a wall will spawn only one prefab, but there can be many content prefabs.

However, there is an exception, in IsOneSideWall mode it can spawn up to 3 parts per wall - one central part and two side parts.

Note also that the triggering of rules also depends on the spawn condition specified in the prefab itself.

Content Collision Avoidance

This section describes how the system avoids collisions and intersections of spawned content objects, as well as ways to avoid frequent spawning of identical prefabs in neighboring cells.

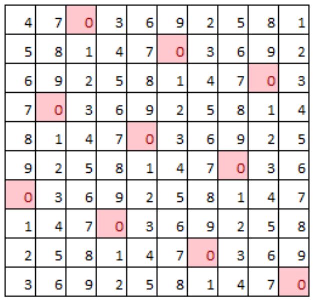
Two approaches are used - algorithmic and physical.
Algorithmic methods simply prohibit prefab spawning in some cells, according to the algorithms described below. The physical method checks for intersections of Bounds objects during spawning. The physical method only works within one cell.

In addition, in version 1.3, a check for intersection with user objects (which are already located on the scene) was introduced. This way, the system can avoid spawning content in places that the user has already occupied. See 🚩Content Blocking Layers

💡Decimation

Decimation filters cells and allows spawning only in selected cells.
In content prefabs, this is a way to spawn in every second cell, every third cell, and so on. This method ensures that the distance between spawned prefabs will be no less than the decimation step.

Example of decimation with step of 3:



4	7	0	3	6	9	2	5	8	1
5	8	1	4	7	0	3	6	9	2
6	9	2	5	8	1	4	7	0	3
7	0	3	6	9	2	5	8	1	4
8	1	4	7	0	3	6	9	2	5
9	2	5	8	1	4	7	0	3	6
0	3	6	9	2	5	8	1	4	7
1	4	7	0	3	6	9	2	5	8
2	5	8	1	4	7	0	3	6	9
3	6	9	2	5	8	1	4	7	0

Decimation is a very performant method, and it is recommended in most cases.
The disadvantage of this method is that it can result in a significant reduction in spawn density due to the fact that the selected cells may not match other spawn rule filters.

💡Chance

All spawn rules have filed Chance that means spawn probability relative to other pretenders to spawn in the Edge.
Reducing the chance results in fewer spawned instances of the prefab in the structure.
However, unlike Decimation, this method does not guarantee that spawned instances will not be nearby.

💡Max Content Per Edge

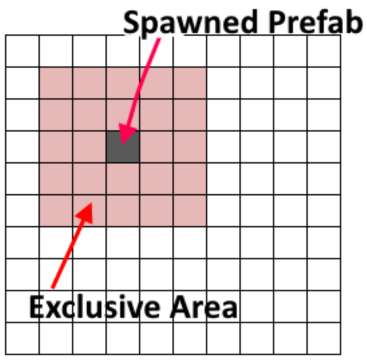
This is a simple method to decrease spawn prefabs in cells. PrefabSpawner just limits the count of spawned prefabs per edge.

💡Exclusive Tag Area

This is the most common method to avoid prefabs spawning nearby. Like Decimation, this method ensures a minimum distance between prefab spawns.

It works like this: when an object spawns, it fills its SetTags on the map in a given radius.
At the same time, other prefabs avoid spawning in cells where the specified tag is.

So after the first prefab has spawned, other prefabs with the same tags will not be able to be created in an exclusive radius around the first prefab.



This method is convenient because it allows you to spread different prefabs in space and not allow them to spawn nearby. For example, if you have ceiling lights, wall lights and spotlights, then using exclusive tags, you can ensure that all these prefabs will not spawn nearby and the area will not be too bright.

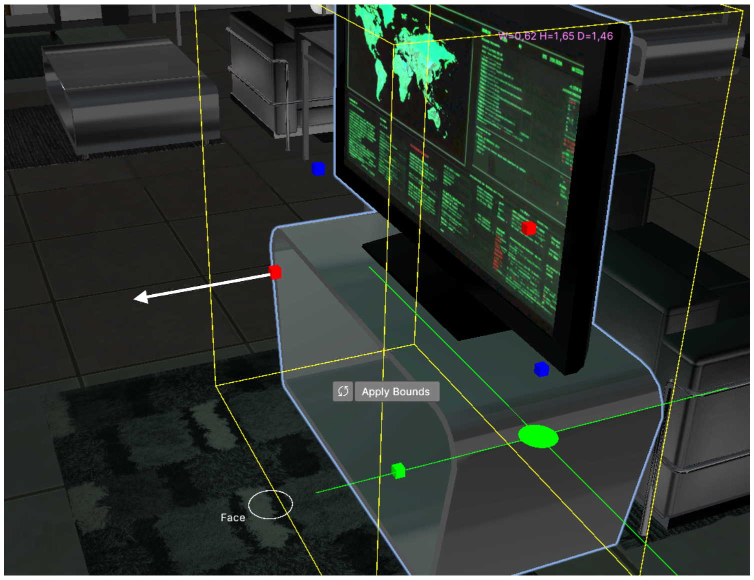
The disadvantage of this method is that it is quite slow compared to other algorithmic methods. Use it only if other methods cannot help you.

💡Checking Bounds Intersections

This method allows to avoid intersections of prefabs inside a cell by checking for intersections from Bounds. If an intersection is detected, the prefab will not spawn.

The system creates prefabs in such a way that their boundaries never intersect.

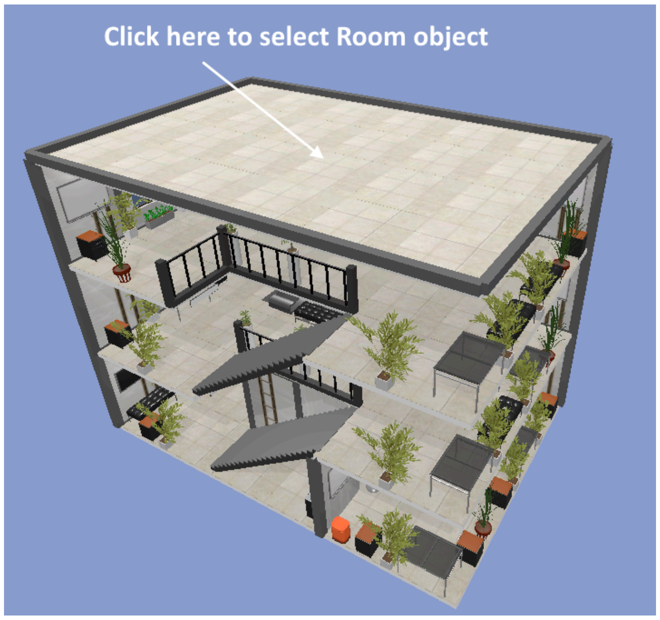
You can manually change prefab bounds, thus configuring collision avoidance more flexibly. For example, to prevent the system from spawning anything in front of a painting, in front of a chair, or in front of a TV, move the front side of prefab bounds.



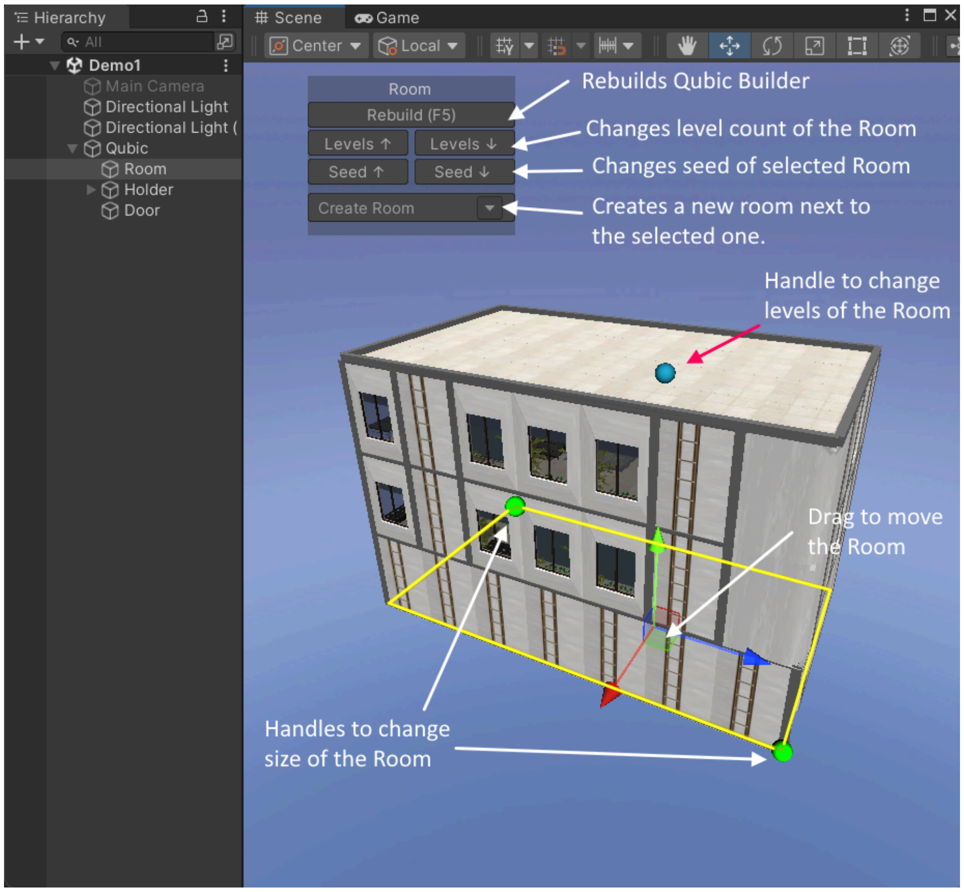
Note that Intersection borders only work within a single cell. Intersections between different cells are not checked.

User Interface

To select Room you can select the Room gameobject in the Hierarchy window, or click on the top level of spawned room.

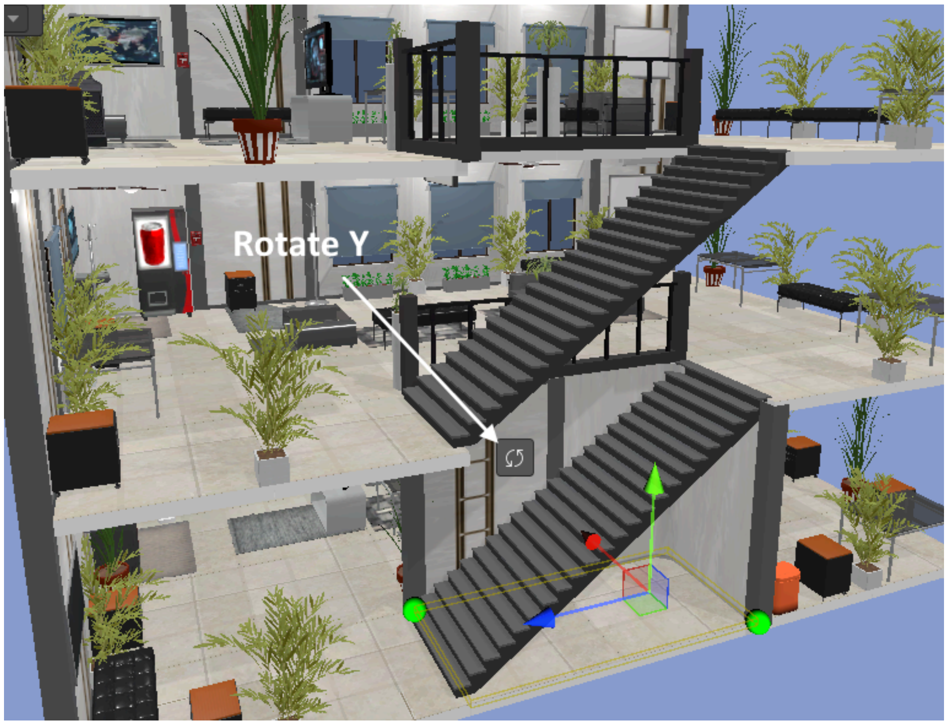


After Room selected you can change room position, rotation, size, levels and draw custom cells:

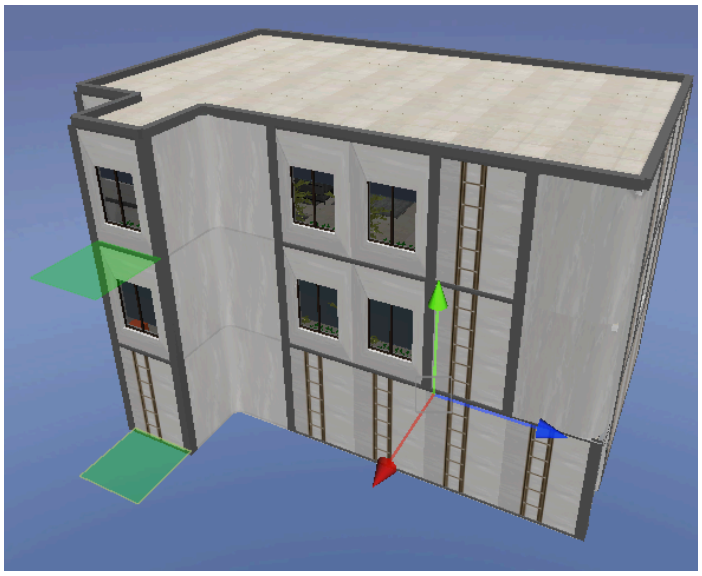


Note, room size and position are always snapped by cell size (specified in used PrefabDatabase).

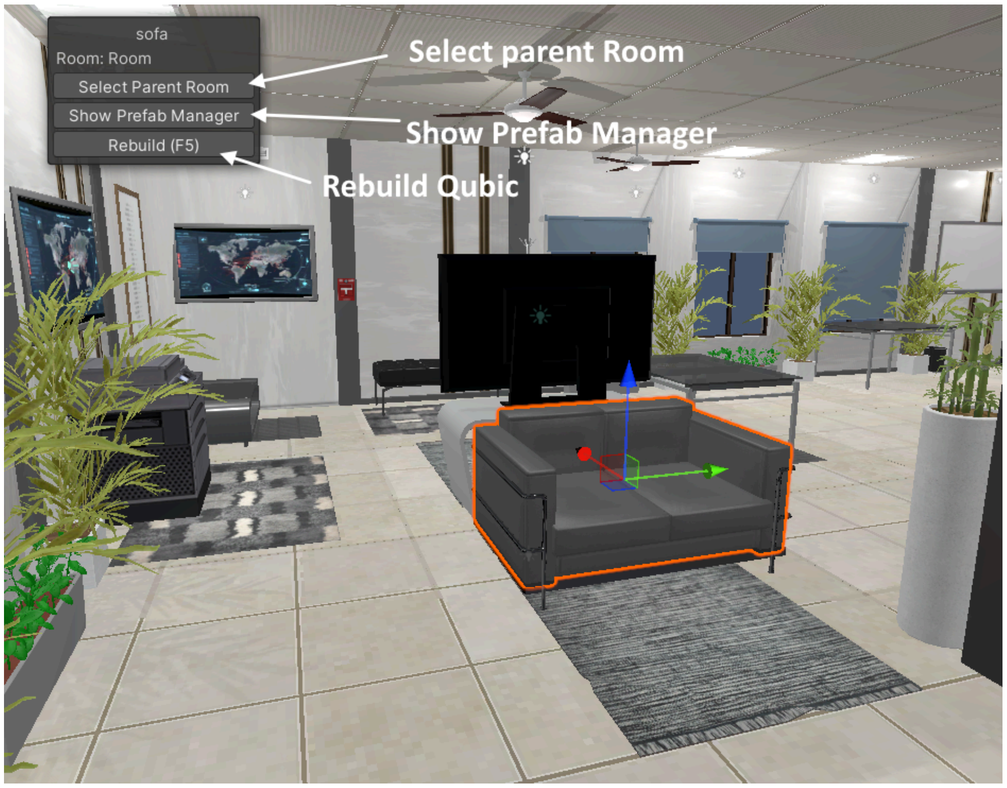
Some structures (Stairs for example) have orientation and can be rotated around the Y axis. This case button Rotate will appear. Also this button is shown if you draw custom cells in the Room.



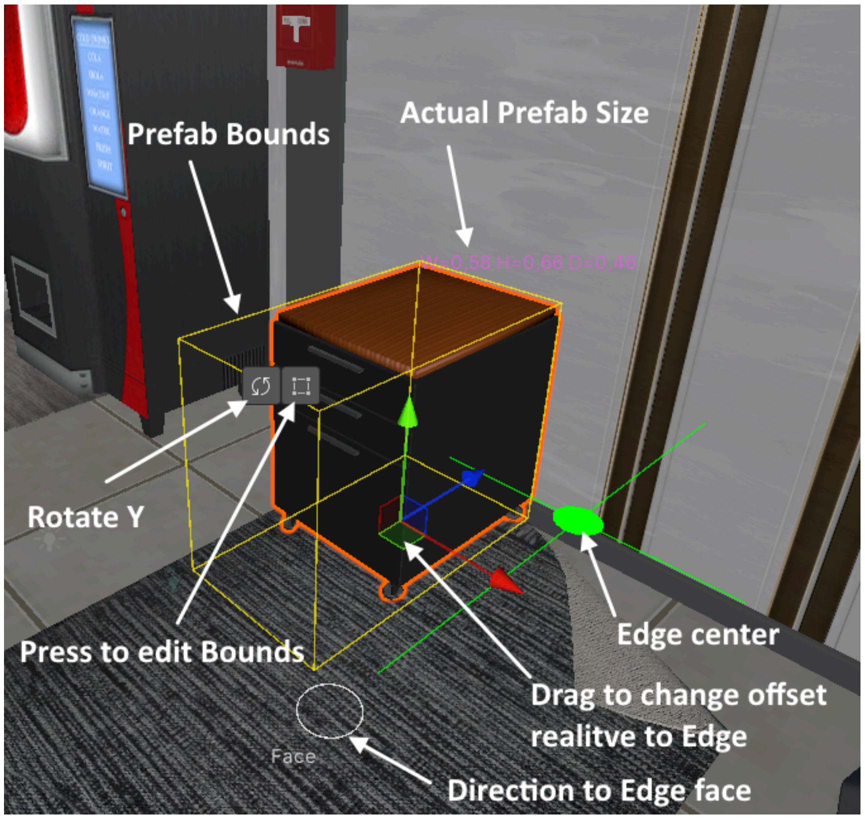
Press **Shift** while the room is selected to start drawing custom cells of the Room. Drawing place will be highlighted by green squares. Press **Ctrl** while the room is selected to remove cells of the Room.



If you have selected some spawned object inside room, you can fast go to parent room object via panel placed in top left corner of scene view:

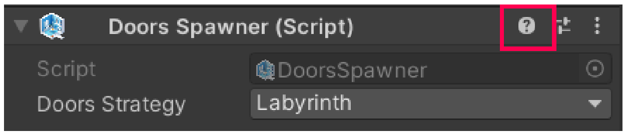


If you selected spawned prefab and Prefab Manager window is visible, you can edit prefab properties directly in Scene View:



All changes will be immediately applied, saved to the prefab, and the Qubic will be rebuilt live, so you will immediately see the changes in the scene.

Note: you can always use the help button in the inspector to open help for the component.



Doors Spawner (Script)

Script DoorsSpawner

Doors Strategy Labyrinth

Multi-tag fields

Many of the fields in the user interface are string fields for tags. In most cases they are multi tags fields. It means that you can write several tags in this field, separated by comma.

Components

Below are described the components that control the generation of the building.

Note that you do not need to add or create the components manually.

The QubicBuilder object will be created via the main menu (Tools/Qubic/Create QubicBuilder), and the rest of the structures and components are created automatically from templates, when you create rooms via the Scene GUI.

🧩Qubic Builder

Root class for building. Contains all spawners, structures, a constructed building, a map, and controls the calling of spawners.

🚩Lock

Blocks Qubic rebuilding.
After the structure is built, you will probably want to protect the structure from accidental rebuilding. This flag prevents rebuilding via menu, via F5, via button, in any way.

🚩Prefab Database

Link to scriptableobject contains prefabs to build Qubic structures. To create a new database - use the Tools menu.
You can change PrefabDatabase anytime to build the same building in a different style, from other prefabs.
Also PrefabDatabase contains the grid size for building cubes.

🚩Seed

Root seed of random generator.
Changing this value changes the appearance of all spawned structures inside QubicBuilder.

🚩Debug

A set of flags affecting the display of tags and cells in the scene. For debugging purposes.

🚩OnBuilt

This event is called after the build process is finished.

🚩Build and Clean Build

These buttons cause Qubic rebuild. Clean Build clears prefabs cache before building.

🧩Room

The basic structure for creating the building's premises.

🚩Tags

Tags that will be assigned to the Cells occupied by the room. This is a multi-tag field.
These tags are called Style sometimes in this document, because it usually contains the style of the room: Closet, Balcony, Bedroom, etc.

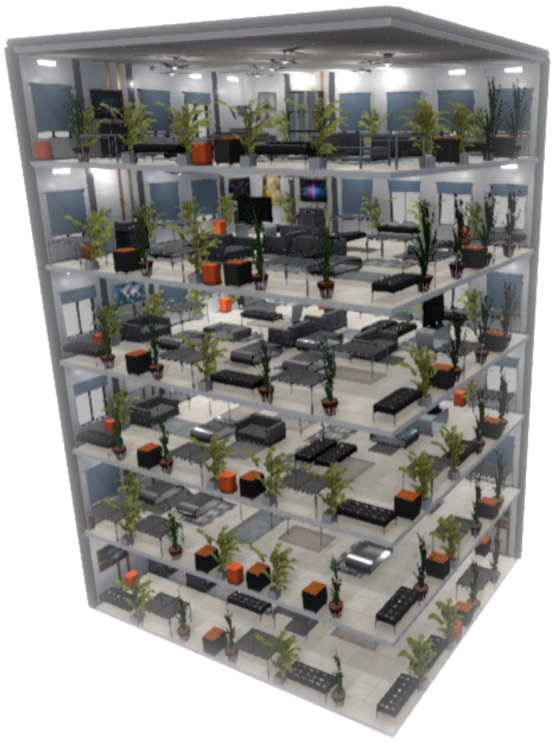
🚩Seed

Seed for random generator. Change this value to change the spawned prefabs of this room. The location of windows, doors, content - depends on the Seed.

🚩Level Count

Number of floors(levels) in the room.

Level Count = 1

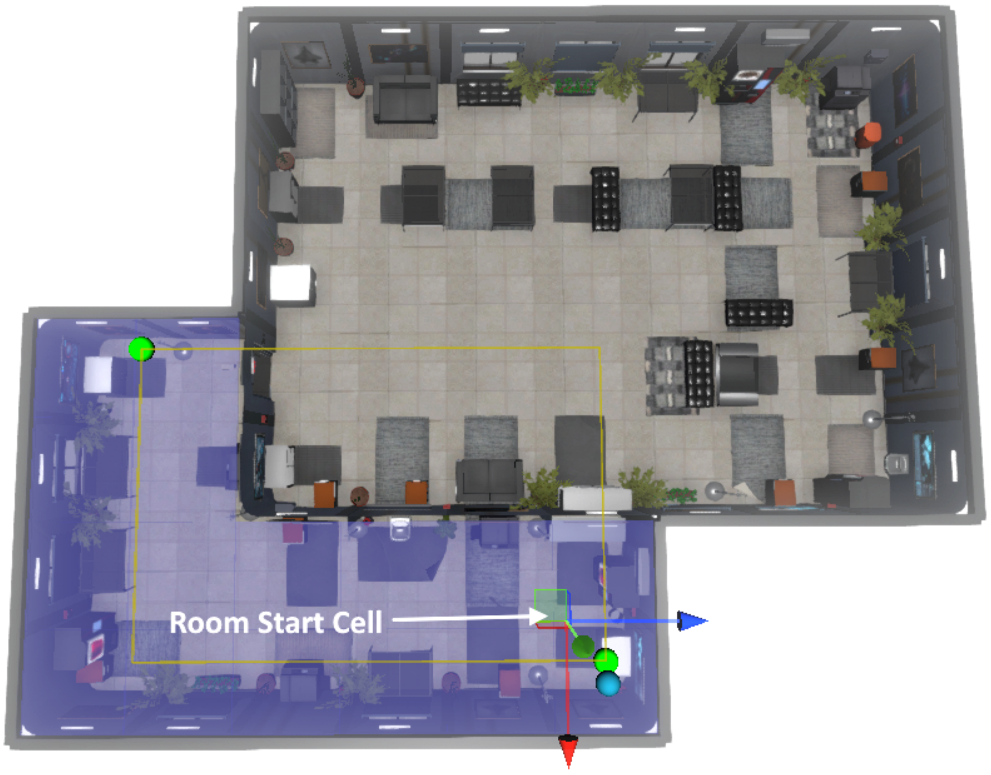

Level Count = 6

🚩Intersection Mode

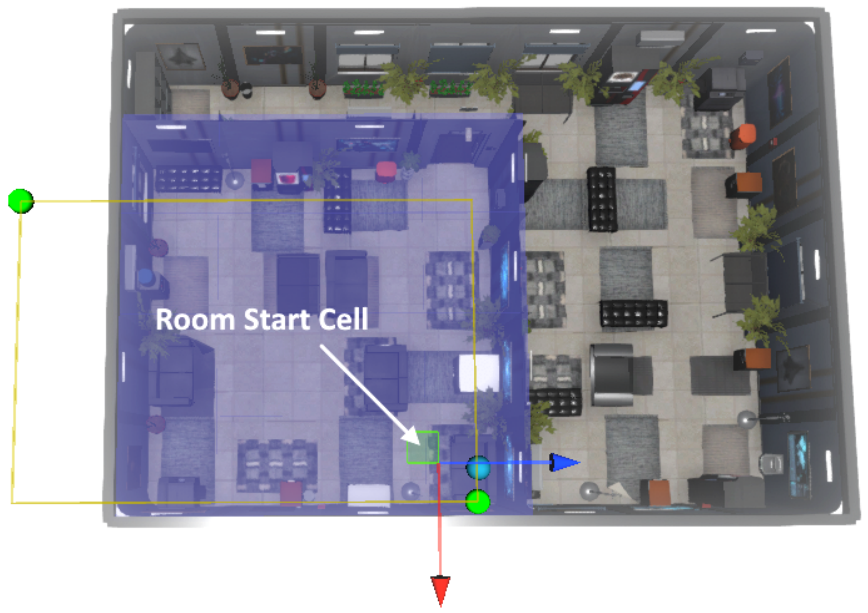
The method of occupying cells relative to other rooms. Building structures (rooms, atriums, staircases, balconies, etc.) compete with each other to take over cells. This property controls how the room will take cells.

Note: how rooms are intersected depends on the order of rooms in the Hierarchy window. Moving the room up/down in hierarchy causes a change of intersection. In general, a room that is higher in the hierarchy is considered more important and has priority over a room that is lower in the hierarchy.

- **Fill** - Fills available space inside or outside of other rooms.
  This method of filling depends on the location of the room's starting cell.
  If the starting cell is outside of other rooms, the room occupies all available free cells. In this case it behavior same as Complement method:

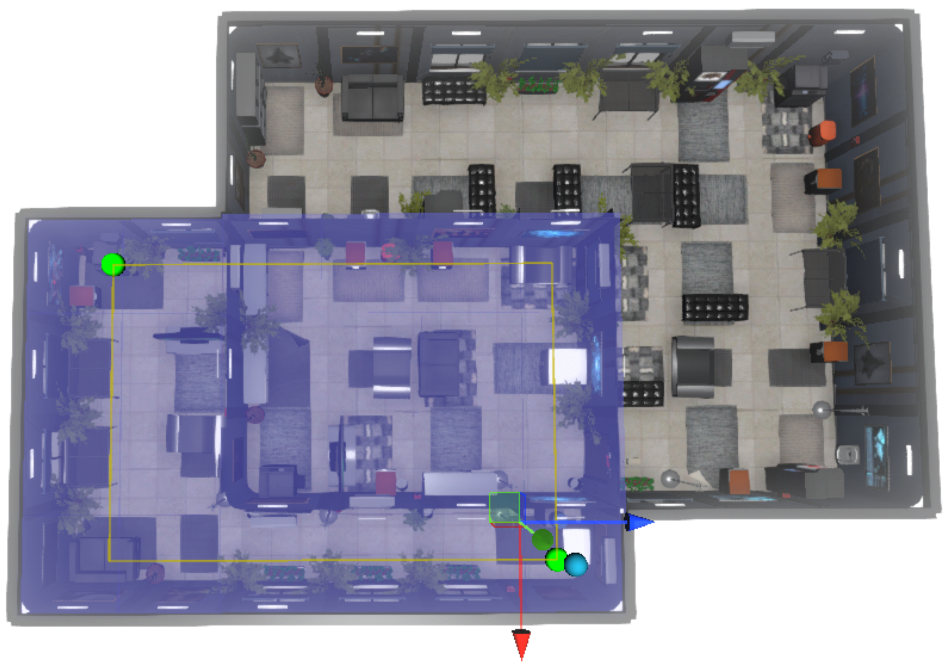


  If starting cell is inside another room (that is higher in hierarchy), the room will occupy cells only inside parents room:

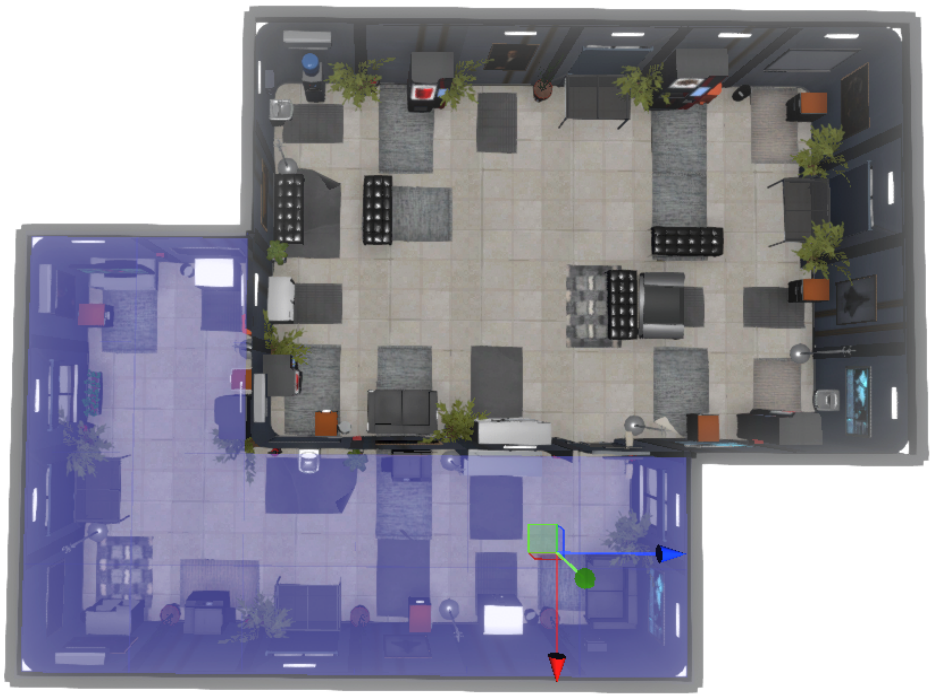


This method is useful for structures that need to be entirely within other spaces, such as atriums or staircases.

- **Overlap -** Occupies all requested cells, ignoring other rooms.This method does not divide the space into non-intersecting regions, so it is used only in special cases.



- **Complement** - Occupies only free cells that are not occupied by other rooms. This is the default method for Room structure.



- **Aggressive** - Occupies both free cells and forcibly takes cells from other rooms (even if they are higher in the hierarchy).

- **Weak** - Like the Overlap method, this method does not divide the space into non-intersecting regions. However, unlike Overlap, this method behaves independently of the position in the hierarchy. It is used in special cases: overlay areas inside rooms (content area, no doors area, etc).

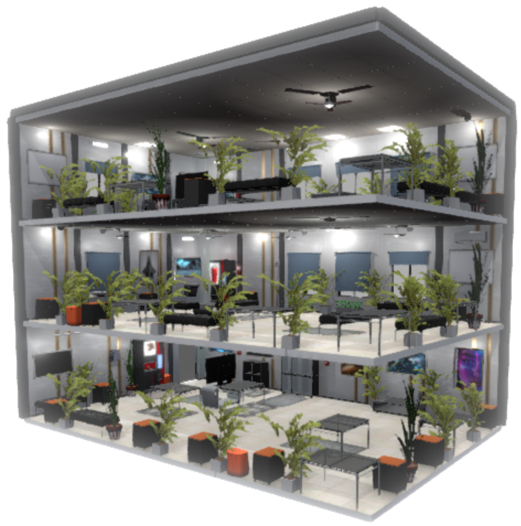
Note that all methods operate in three-dimensional space, and that occupied cells may differ from level to level. For example, the Complement method takes free cells per level basis:

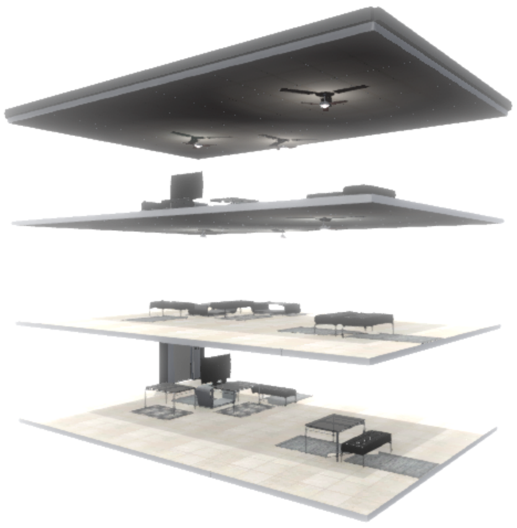
🚩Features

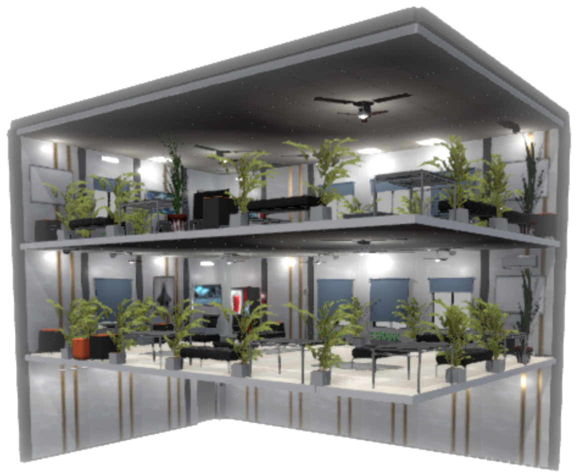
Set of fine-tuning settings.
Build section:

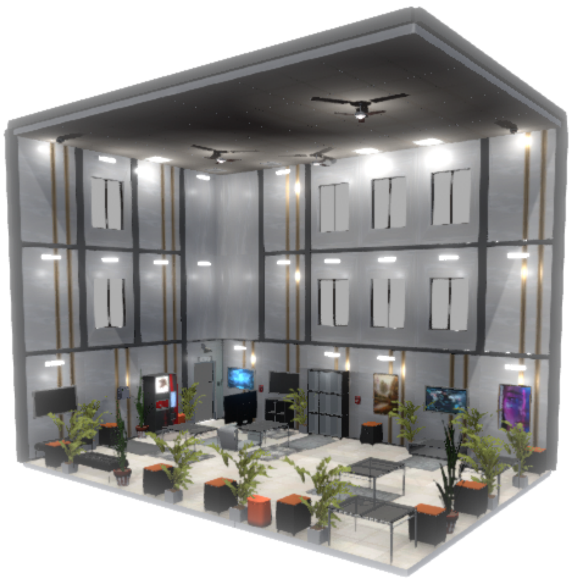
- **Walls** - allows to build walls.
  Walls are not always necessary, for example you can create a room that simply changes the cell tags (i.e. Style) in a certain area. Such a room does not require walls.
  Note that when we say "builds walls", it means that the structure creates Wall tags on the map. Remember that structures do not interact with prefabs directly, they only spawn tags on the map.
- **Floor On First Level** - allows to spawn floor on first level. This option spawns Floor tag on the map.
- **Floor On Intermediate Levels** - allows to spawn floor on mid levels - from second to last (exclude roof). This option spawns Floor tag on the map.
- **Roof / Ceiling** - allows to spawn ceiling on the last level of the room. This option spawns Ceiling and Roof tags on the map.

By default all flags are On.


All flags are On (isometric view)


Walls Off


Floor On First Level Off


Floor On Intermediate Levels Off

Roof / Ceiling Off

🚩Doors Strategy

Allows to override doors strategy defined in DoorsSpawner for this room.

- **Inherited** - use strategy pointed in DoorsSpawner (default value).
- **Fully Connected** - forces to build doors to all neighbor rooms.
- **No Doors** - forbid to build any doors in the room.
  Note that this is a special mode that affects not only the doors, but the behavior of the room as a whole. Only in this mode, the room does not capture cells. This means that only this mode allows you to build overlays - areas that change the properties of cells but do not build walls, and do not change the structure of the room.
- **Impassable** - forbid to build any doors in the room and mark this space as impassable (used mainly for stairs structures).
- **Custom** - allows you to manually specify a list of rooms with which the door will be created.

🚩Doors

Defines a list of rooms to which the current room will be connected by doors. This property is only available when DoorsStrategy = Custom.

🚩Wall Content Count

Property determines the number of prefabs that will be spawned near walls. This number also depends on the MaxContentPerEdge property of PrefabSpawner.

- **Zero** - disable content creation. Note that this property works for both walls and inside content. This property also works for overlays (NoDoors areas).
- **One** - spawn exactly one prefab per wall.

- **Two** - spawn exactly two prefabs per wall.
- **Max** - spawn prefabs defined in MaxContentPerEdge property of PrefabSpawner (default value).
- **From0toMax** - random prefabs count from 0 to Max.
- **From1toMax** - random prefabs count from 1 to Max.
- **ZeroOrMax** - 50% of cells (on average) will contain Max prefabs, other - no prefabs.
- **OneOrMax** - 50% of cells (on average) will contain Max prefabs, other - exactly one prefab.

Note: modes One, From1toMax, OneOrMax will spawn at least one prefab. It is suitable if you want to spawn one required prefab per cell - for example light, but avoid to spawn other prefabs in some cells.

🚩Set Tags Strategy

Defines how Tags property will be applied.

- **Replace** - replace any tags to Tags in the captured cell (default value).
- **Add** - adds Tags to captured cells. Thus this mode just adds tags, does not remove previously assigned tags.

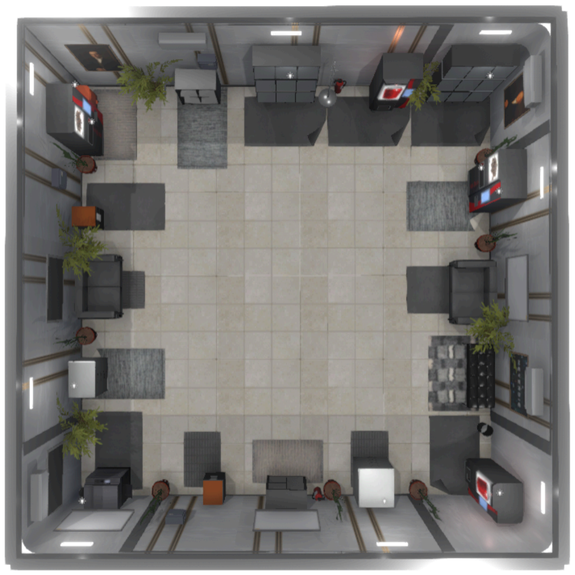
🚩Inside Content

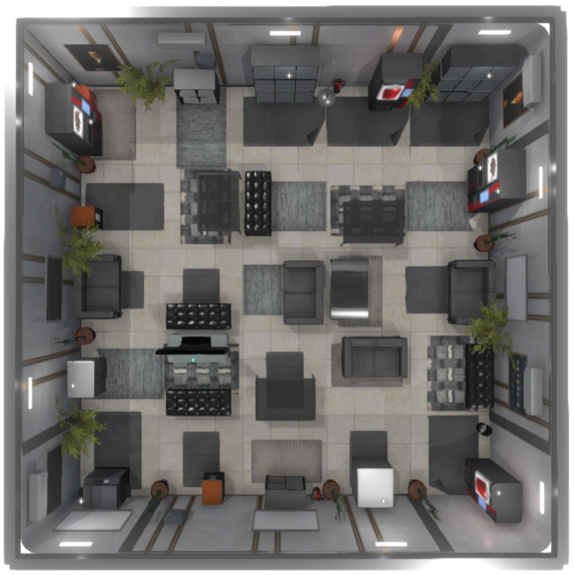
As mentioned in Overall Information, Qubic spawns prefabs only along Walls/Edges. Thus, to spawn interior content not only along outer walls, we need to spawn invisible walls inside the room. Such walls are called Content Walls and marked on a map with a standard Content tag. This section contains settings to spawn inside content walls.

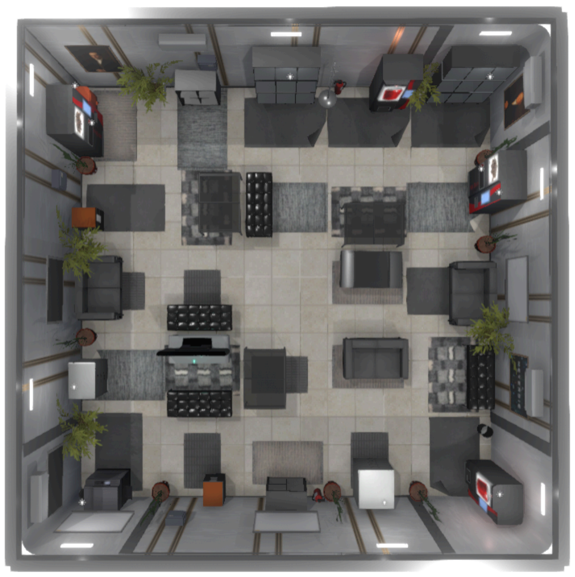
🚩Layout

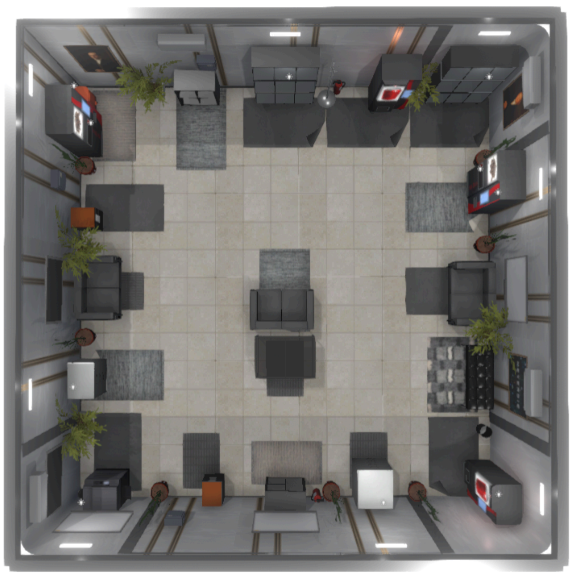
Type of content walls.

- **None** - does not spawn any inside content walls. This option disables content inside the room (excluding outer walls).
- **Full** - spawns content walls in all inside empty edges.
- **Island** - spawns inside content walls in the shape of square islands. This is the default value.
- Different layouts to achieve some patterns.


Layout = None

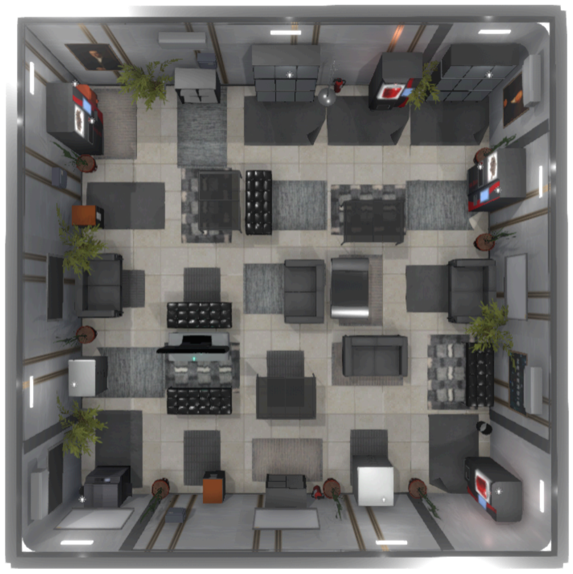

Layout = Full

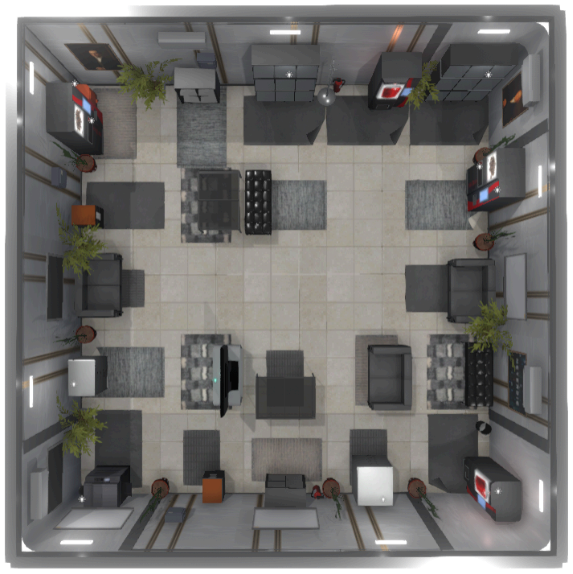

Layout = Islands

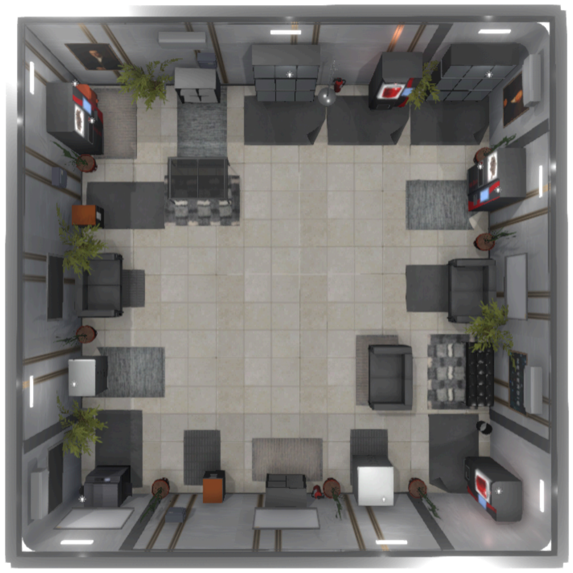

Layout = One Along X

🚩Decimation

Step of decimation of content walls.
Increase this value to reduce the amount of content walls.
Below examples of content with different decimation value (Layout = Full):

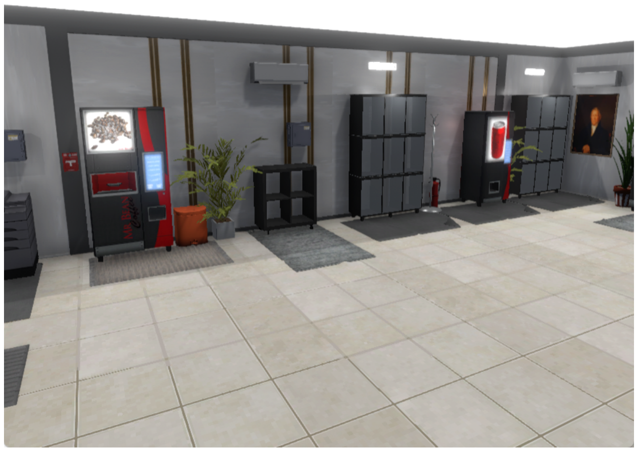

Decimation = 1

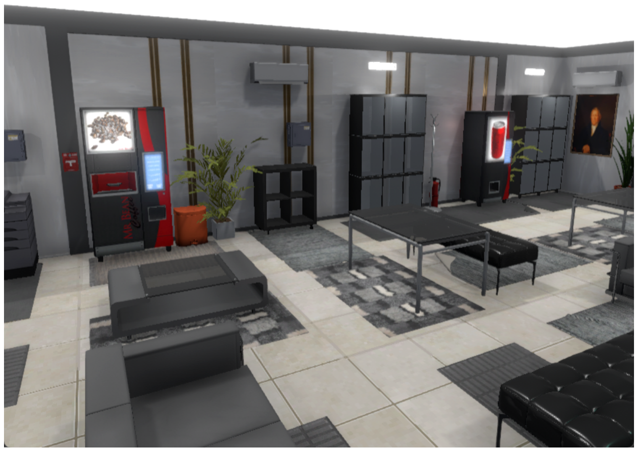

Decimation = 5

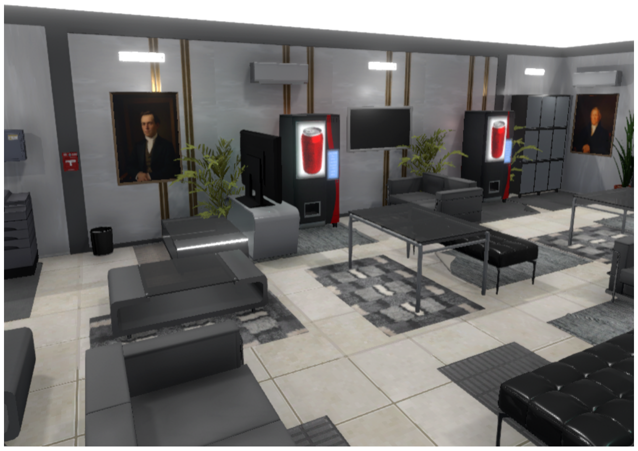

Decimation = 9

🚩Do Not Affect Walls

This flag prevents the content wall from being placed next to the outer walls of the room. By default - True.
If you look closely at the images above, you will see that the internal content does not affect the content located along the outer walls of the room. This is because the content walls are not built next to the walls of the room. If you turn off this mode, the content walls will also affect the content located along the outer walls:


Layout = None


Layout = Full, DoNotAffectWalls = True


Layout = Full, DoNotAffectWalls = False

🧩Stairs

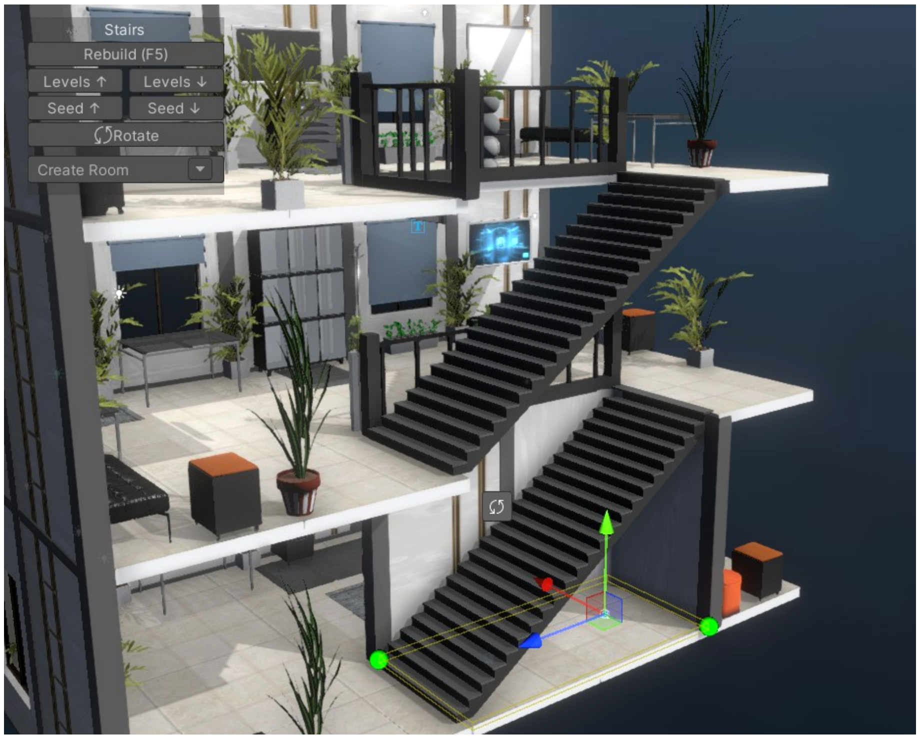
The structure for creating the staircase.
This component is inherited from Room, so all properties of Room are available for Stairs too.

Note: Stairs component does not generate prefabs or meshes for steps. It just makes tags markup. You should use your own prefabs for steps mesh.

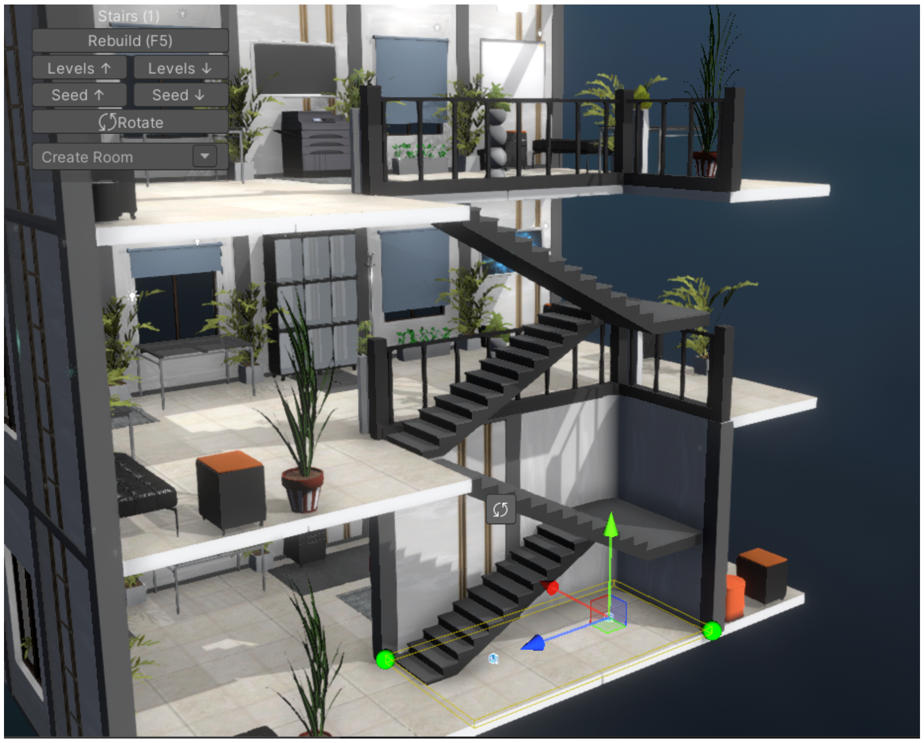
🚩Type

Type of stairs.

- Straight - A type of staircase in which flights of stairs go in one direction.



- U Shaped - Staircase with U-shaped two directional flights.



🧩Atrium

The structure for creating open space inside a building, without floor.
This component is inherited from Room, so all properties of Room are available for Atrium too.

🧩Wall

Structure for forcing the creation of an edge with certain tags.
Unlike other structures inherited from Room, the Wall structure does not spawn Cell tags, but an Edge tag on one of its sides. The structure can be rotated to select the desired Edge.
This structure is designed to force the spawning of an Edge of a certain type - a door, a window, an arch, etc.

🚩Tags

Tags that will be assigned to the Edge occupied by the wall. This is a multi-tag field.

🧩Prefab

Contains information about the prefab and spawn rules for it. Prefabs are stored in PrefabDatabase scriptable objects.

Note properties of selected prefab are displayed in the PrefabManager window, not in Inspector. Because Qubic does not attach any scripts to prefabs, it stores info about prefabs in PrefabDatabase.

🚩Type

Type of prefab. This is a readonly field that is assigned when you create a Prefab object.
List of allowed types:

- **Wall** - structural unit of building: wall, door, window, rails, parapet, steps and so on.
- **Corner** - same as Wall, but L-shaped structure to spawn in corners of building.
- **Column** - column type connector to spawn between wall prefabs.
- **Content** - prefabs to spawn inside building: furniture, lamps, carpets, curtains and so on.

🚩Prefab Info

Contains a reference to the prefab object, as well as additional information about the object's placement and rotation.

- **Prefab** - link to Prefab object. This field is assigned automatically during creation of the Prefab object, but you can update this field directly, dragging here a new prefab. You can drag only Unity's prefabs (ie object with blue icon) here - from project view or from scene.
  Do not drag non-prefabs GameObject here!
- **Anchor** - offset of object when spawned, relative to edge center. This value is updated automatically, when you drag the spawned object in the scene view. But here you can set the value manually and more precisely.
- **RotationY** - sets the rotation of the object around the Y axis when spawning. This value is automatically updated when you click the rotate button in Scene View.

- **Alternatives** - alternative versions of the prefab. You can specify any number of additional versions of the prefab by simply dragging them here. They will spawn with equal probability, instead of the main prefab specified in the Prefab field.
  This is convenient if you have a large number of almost identical objects, with the same spawn rules and the same sizes (for example, a set of vases with flowers, or a set of different versions of curtains). Then you can not create a Prefab object for each of them, but simply create one prefab that will spawn all the sets of similar objects.

🚩Levels

The range of valid floors of the building where the prefab can spawn. First level of the building (i.e. the ground floor) has number 0.

You can adjust this value if you want the prefab to spawn only on the ground floor, or only on the upper floors. This is often useful for spawning walls and columns, for example if you want stone walls on the ground floor and wooden walls on the upper floors.
Also this property is useful to build different styles of rooms in basement or underground levels (in this case level should be negative).

🚩Set Tags

Tag that will be set to Cell or Edge after the prefab is spawned. If the field is empty - it does not affect map tags. This is a multi-tag field.

Setting the tag works differently for different prefab types:
- For Wall - replaces tag in Edge.
- For Corner - replaces tag in two adjacent Edges taken by the corner prefab.
- For Column - does nothing.
- For Content - adds (but does not replace) tag to Cell where prefab spawned. You can also specify the tag spawn radius in this case.

SetTags is a very useful tool that allows some prefabs to influence other prefabs. For example, to spawn curtains on windows, the window prefab spawns in the Edge with the Wall tag, and after spawning, it changes the tag to Window. And the curtain prefab spawns in the Edge with the Window tag. Thus, curtains will always spawn where windows spawn and will not appear in other types of walls.

Note, first the system spawns prefabs of the Wall and Corner type, then Column and lastly - Content. Thus, the tags set during the spawn of Wall, Corner affect the spawn of Column and Content, but not vice versa.

🚩Set Tags Radius (for Content prefabs)

The radius in which the SetTags will spawn (in Cell units). This property works only for Content prefabs.

- Value 0 or less - does not affect tags map.
- Value 1 means spawn SetTags exactly in the cell where the prefab was spawned.
- Value 2 means to spawn the tag in the cell and in the surrounding cells (in fact, in a 3x3 square).
- Value 3 and more - force spawn tag in square of specified radius around cell (4x4, 5x5 and so on).

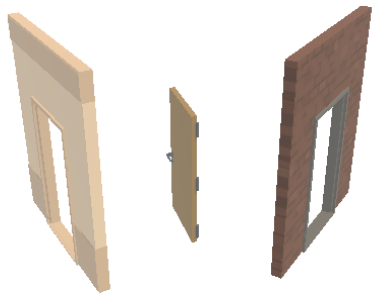
Note, radius affects only in XZ plane, it does not affect on other levels of building.

In combination with AvoidTags and AvoidCellsWithSetTags, this property is useful if you want to prevent the same content spawning within a certain radius. More detailed - see AvoidCellsWithSetTags property and Content Collision Avoidance chapter.

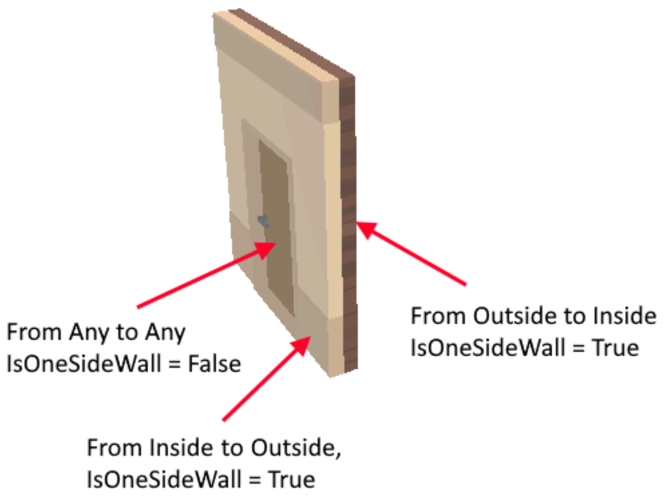
🚩Wall Features (for Wall prefabs)

Set of settings for non-content prefabs.

- **Content Padding** - For Wall prefabs: sets the offset from the wall when spawning Content (meters). This is useful if you have wall prefabs with different thicknesses. This allows you to automatically move furniture or wall prefabs away from spawned walls with different thicknesses. This value will be added to the Z component of the Anchor property of Content prefab, when it will be spawned.
  For Floor prefabs: sets offset along Y axis when spawning Content (meters).
- **Requires Column** - This wall prefab requires columns to be created on both sides.
- **Is One Side Wall** - This option means that this prefab is only half a wall on one side. This is useful if your walls consist of different parts (usually an inner and an outer part). As in the picture:

  Then you can mark the interior and exterior part as IsOneSideWall, and spawn them in opposite directions (one from Outside to Inside, another - from Inside to Outside). Door - without IsOneSideWall flag.
  As result the Qubic will spawn three different part of wall:



Prefabs of two sides of the wall are selected based on the coincidence of SetTags values. Therefore, if you need two halves to spawn in one Edge, they need to have the same SetTags.

Also see tutorial video: https://youtu.be/3R3fmek9zEY

🚩Column Features (for Column prefabs)

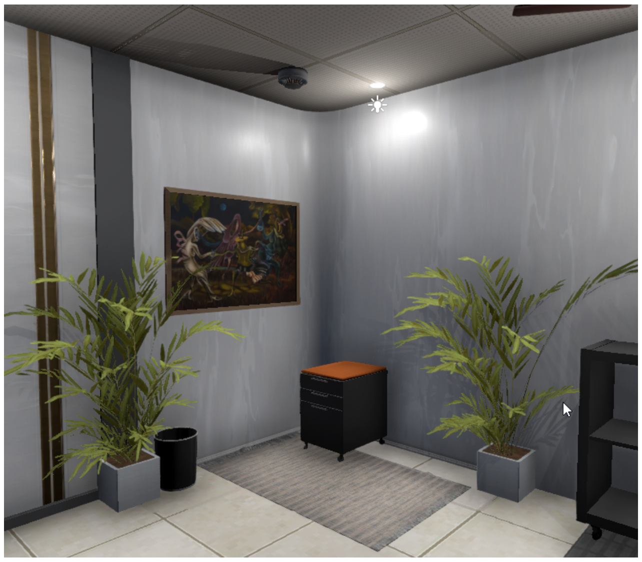
- **Corner Type** - corner type where the column can be spawned.
  Note that corner type depends on the spawn rule and orientation there. For example L-Corner can be Convex from Outside to Inside, and at the same time this corner can be Concave from Inside to Outside rule. The corner type is determined by the right end of the wall.
  In common case you should not use corner type for Any-Any rules, because in this case orientation is not defined.
  [TODO:figure]

🚩Content Features (for Content prefabs)

- **Collision Layer** - name of collision layer that used to check content intersection. By default - Default.
  Typical cause of change is carpets. Usually, the carpet intersects with the legs of the furniture. As a result, the system detects the intersection and either does not spawn the furniture or does not spawn the carpet. To avoid this, make a separate collision layer for the carpet (for example, call it Carpets). Then the furniture will spawn regardless of the intersection with the carpet. And at the same time, the carpets themselves will check for collisions with each other and not spawn in one place.
- **Avoid Narrow** - avoid narrow corridors to spawn.
  Cell is narrow if current wall has wall in opposite side of cell:

To avoid spawn in such a place - set value to **On**.
Value **Auto** - the system measures prefab size and does not spawn big objects in a narrow place.

- **Avoid Convex Corners** - avoid to spawn in convex corners:



- **Avoid Concave Corners** - avoid to spawn in concave corners:

- **Need Ceiling** - requires ceiling to spawn this prefab. Set this flag for lamps, ceiling fans, vents and other content that is placed on the ceiling.
- **Need Floor -** to spawn, requires floor in this cell. Be default - true. Any content that stands on the floor requires this flag. But you can switch off for wall content (wall lamps, curtains, pictures) or for ceiling.
- **Need Column** - a left or right column is required to create a prefab. The column side (left or right) is determined by the offset of the prefab relative to the edge center.
- **Ignore Wall Padding** - forces to skip wall padding. The prefab will always spawn on a predefined distance from the edge. This useful checkbox for some complex assembled construction like moldings, tubes, air ducts, etc.
- **Ignore Floor Padding** - forces to skip floor Y padding. The prefab will always spawn on a predefined distance from the floor.
- **Ignore Collisions** - forces to skip checking of bounds intersection with other prefabs. So this prefab can be spawned even if it intersects with others. It can be useful for self-intersections constructions like moldings.
  Note, this does not cancel checking of intersections of other prefabs with this. So, other prefabs will not be able to spawn in bounds of this prefab.
- **Check Walls Around** - Forces rules to check not only the current Edge tag, but also the tags of all Edges in the cell.
  For example, if you spawn a bed and don't want it to spawn near a window, you need to check not only for a wall in the current Edge, but also for walls in neighboring Edges.
  To check around Edges - set value to **On**.
  Value **Auto** - the system measures prefab size and will check around Edges for big content objects.
- **Decimation** - Decreases count of spawned instances via decimation of allowed to spawn cells. In other words, the option spawns objects in each 2-nd, in each 3-d, and so on cell. This option allows to avoid spawning the same prefab in neighbor cells, while at the same time it allows to decrease total instance count in building.

Value None allows prefab to spawn in the same cell, other values - prohibits to spawn the same prefab in the same cell.
More details - see Content Collisions Avoidance.

- **Avoid Tags** - cell or edge tags where prefab spawning is prohibited. This is a multi-tag field.
- **Avoid Cells With Set Tags** - avoid to spawn in cells tagged as SetTags property. Actually, this option just adds value from the SetTags property to AvoidTags property. More details - see Content Collisions Avoidance.

🚩Rules

A set of rules that control the spawning of a prefab. The prefab will spawn if at least one of the rules is met.

Note, for non-Content prefabs, the prefabs can be spawned once per cell's Edge (ie for each of 4 Edges of cell). But for Content prefabs the system spawns no more than one prefab instance per Cell.

🧩Prefab Rule

Contains the tag-based rule for spawning this prefab.
Before writing rules, please read the How the system works and Rules chapters to understand the logic of the system, tags and the structure of the map.

Note that rules of selected prefab are displayed in the PrefabManager window, not in Inspector. Because Qubic does not attach any scripts to prefabs, it stores info about prefabs in PrefabDatabase.

🚩Wall

Tags of Edge required for prefab spawning.
At least one of the listed tags must be present in Edge for the rule to be triggered. This is a multi-tag field.

🚩From

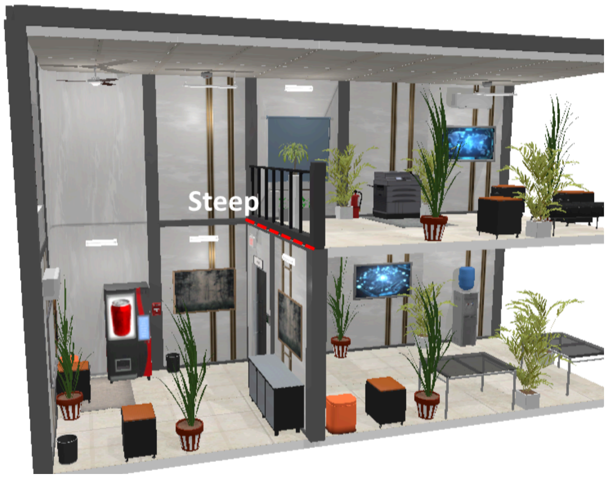
Cell tags required for prefab spawning. At least one of the listed tags must be present for the rule to be triggered. This is a multi-tag field.
The From tag is called "Style" sometimes in this document.

🚩To

Adjacent Cell tags required for prefab spawning. At least one of the listed tags must be present for the rule to be triggered. This is a multi-tag field.

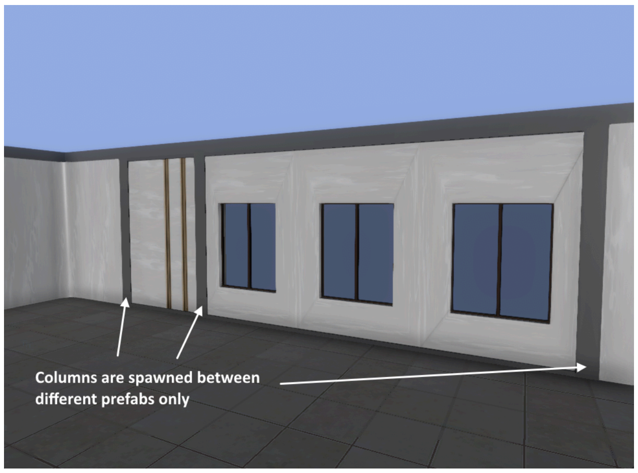
🚩Steep

Forces the prefab to spawn only in places where there is a height difference on the floor. Usually used for spawning railings.

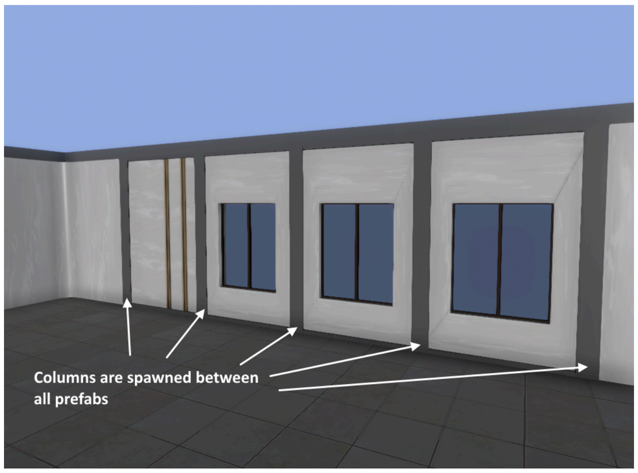


🚩Between Same Prefabs (for Column prefabs only)

Forces to spawn columns between the same prefabs. By default - false.
By default the system spawns columns only between different prefabs. But this option forces a spawn column between any neighbor prefabs.



Between Same Prefabs = Off



Between Same Prefabs = On

🚩Priority

The priority of the rule. The rule with the higher priority will always be checked first.
Note that the rules for the Wall and Corner prefabs are triggered first, then the rules for Column, and then for Content. That is, the priority affects the execution order only within prefabs of a given type.

🚩Chance

Chance is the probability of the rule being executed among other rules with the same priority. By default - 5.

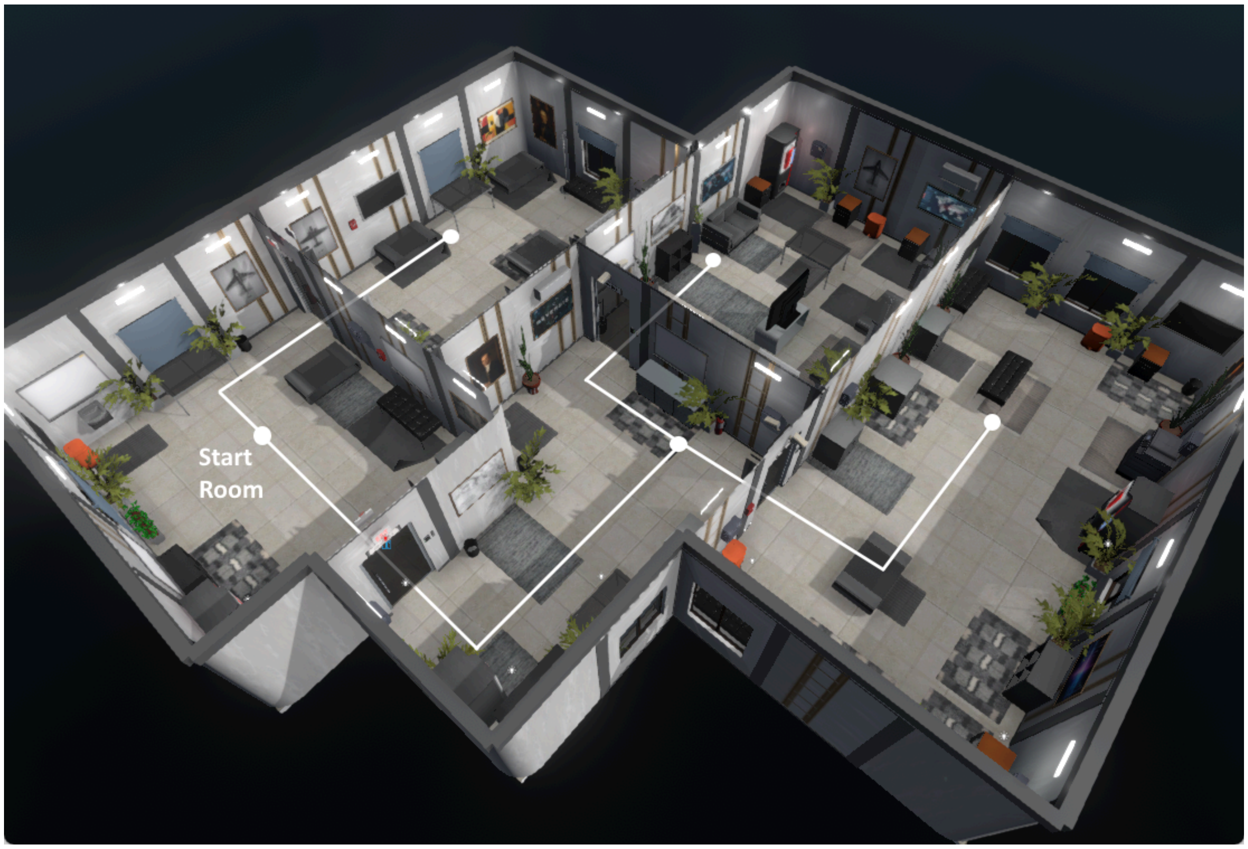
🧩Doors Spawner

The spawner creates doors to ensure that all rooms are connected. It spawns standard Door tag in
Edges.
This script is automatically attached to QubicBuilder gameobject.

🚩Doors Strategy

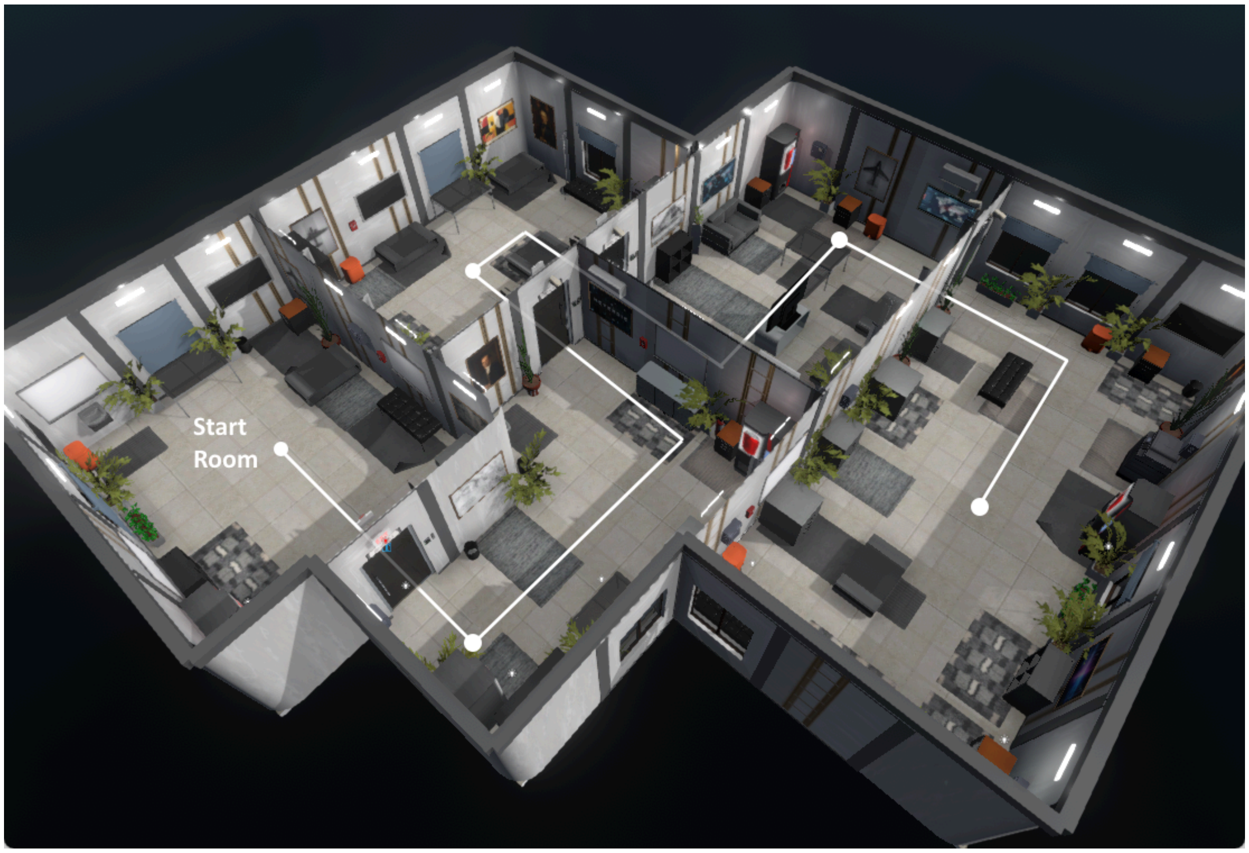
Specifies the method for rooms connecting.

- **Tree** - this method creates doors in such a way that from the starting room (the top most room in the hierarchy) you can get to any room by the **shortest** route. This is the default method.
  This method is suitable for most proposed buildings.

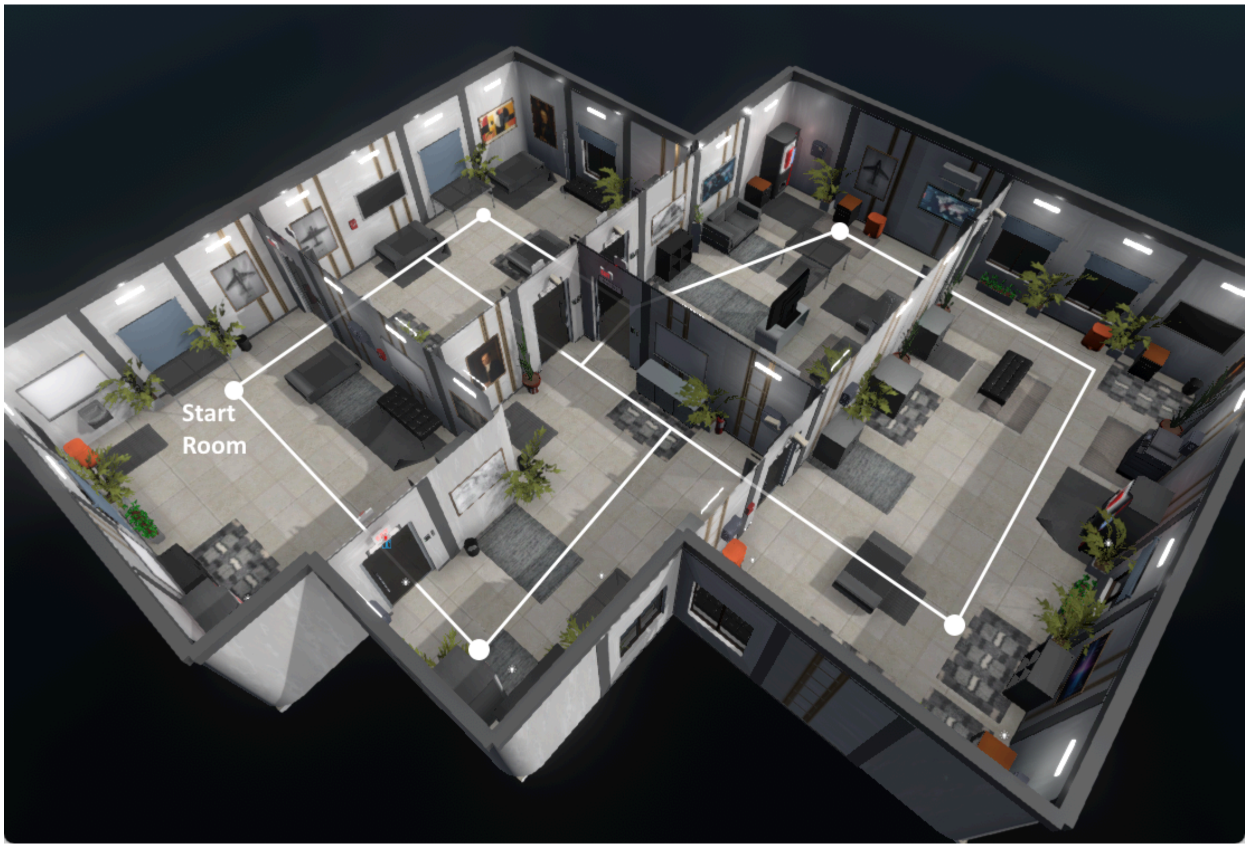


- **Labyrinth** - this method creates doors in such a way that from the starting room you can get to any room by the **longest** route.

This method will always lead the player along the longest route. Suitable for dungeons or mazes.



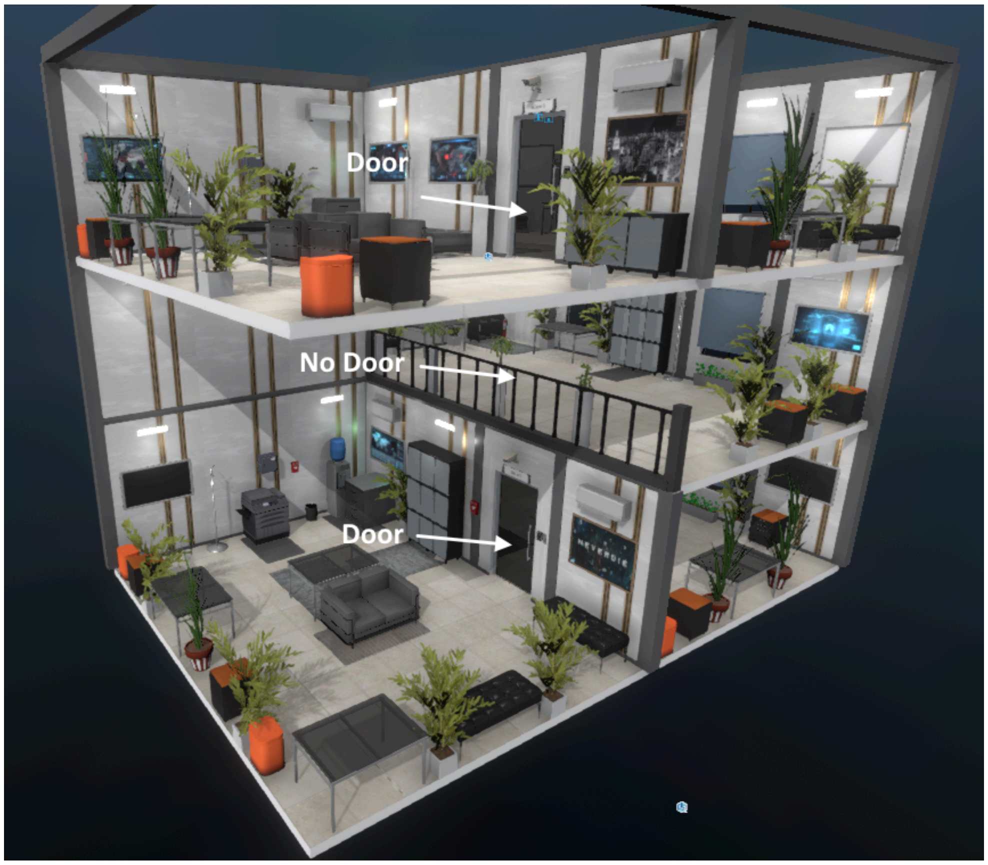
- **Fully Connected** - this method creates doors between any neighboring rooms.



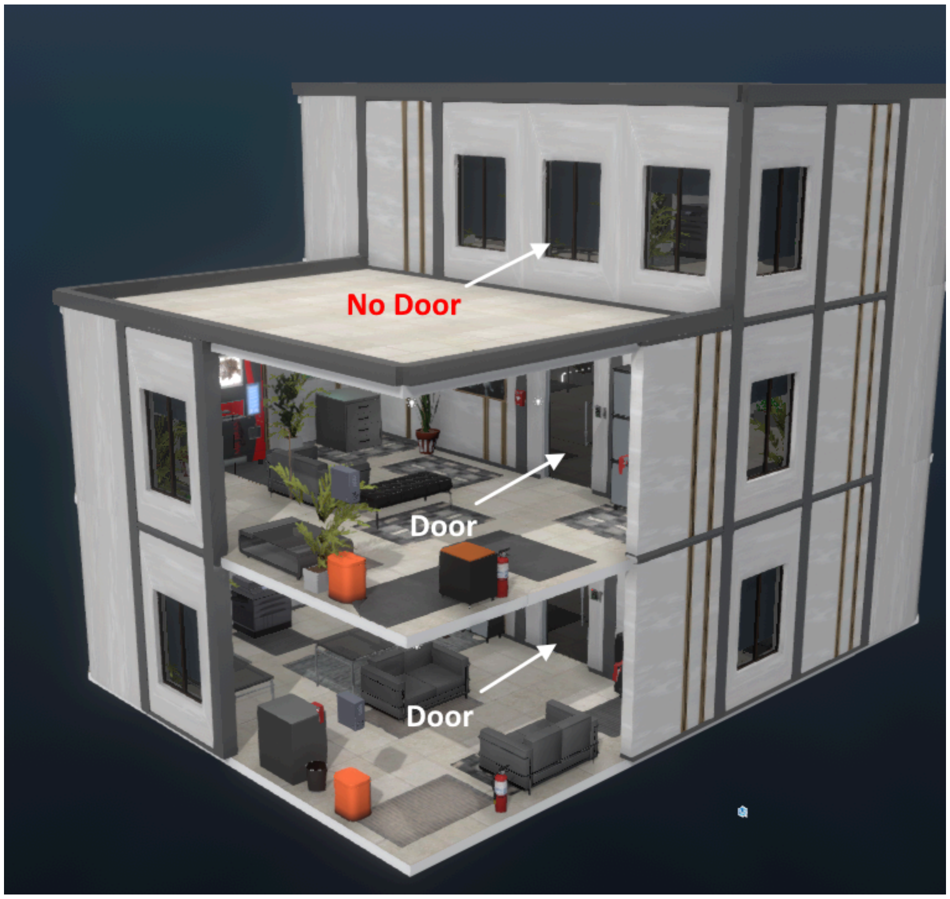
Since buildings are multi-story structures and rooms can have different numbers of floors, the location of doors and movement paths may differ depending on the floor. But in any case, DoorsSpawner guarantees connectivity of all rooms on every building level.

DoorsSpawner does not pay attention to stairs. It provides connectivity only per level basis. Therefore you should ensure vertical level-to-level connectivity yourself (via Stairs).

To spawn a door, DoorSpawner requires a floor in both adjacent rooms. If no such Edge is found, doors will not be created.
In the example below, doors do not appear on the second floor because one of the rooms does not have a floor at that level:



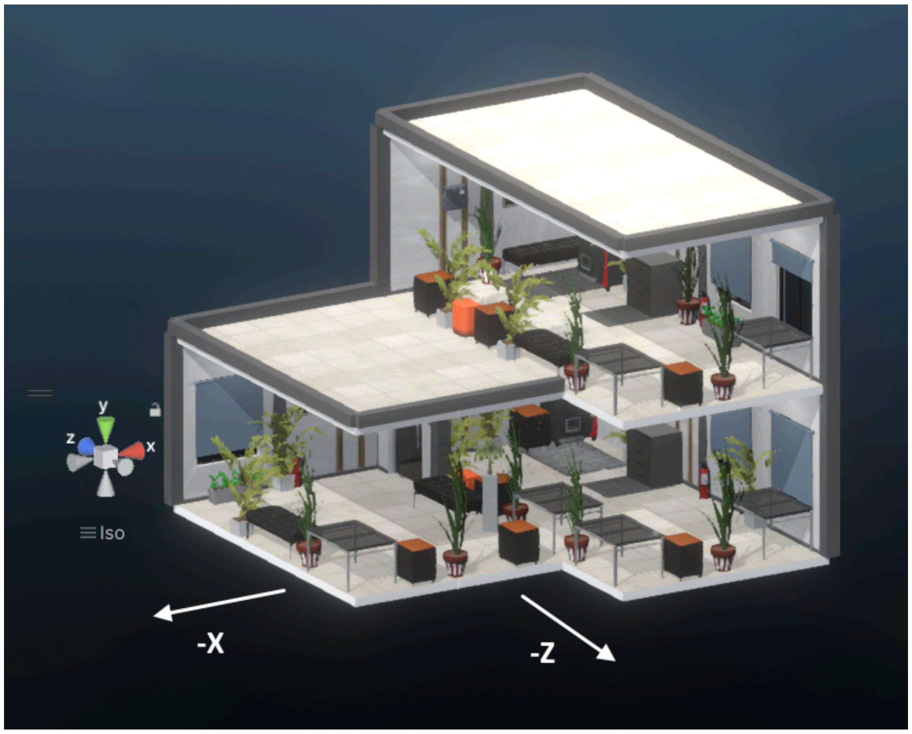
Also, DoorsSpawner doesn't spawn doors from room to roof/ceiling, because ceiling is not a passable area from DoorsSpawner viewpoint:



Note: you can override doors strategy for specific rooms in property DoorsStrategy of Room component (include custom Doors).
Note: you can force spawn doors in arbitrary places by Forced Door template (Wall component based).

🧩Isometric Spawner

Enables isometric view of building: makes walls transparent in negative X and Z directions.
This script is automatically attached to QubicBuilder gameobject.



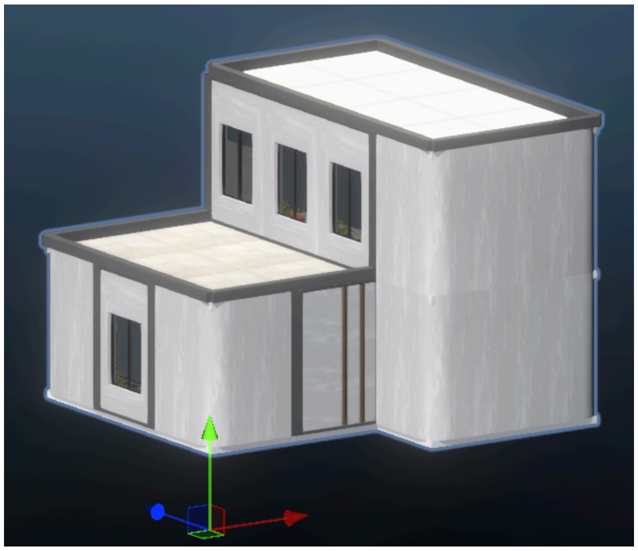
Note, IsometricSpawner does not remove the roof and ceiling. To remove them, disable the flag Build Ceiling/Roof in Features of the room object.

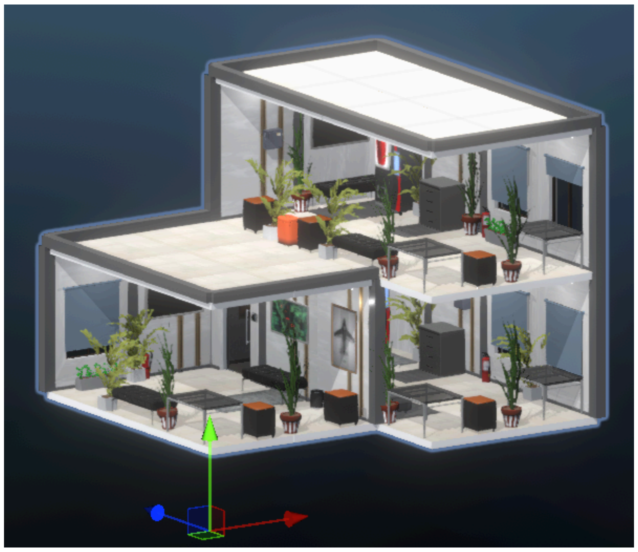
Note, for debug purposes that isometric mode can be forcibly enabled by property Forced Isometric View in Debug of QubicBuilder (or via hotkey F6).

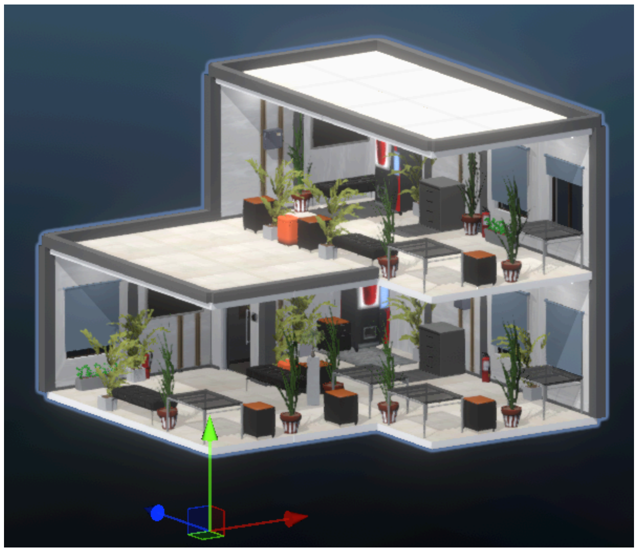
🚩Mode

Specifies the isometric mode.

- **None** - do not use isometric view.
- **Hide Outside Walls** - replaces outer walls with standard tags Transparent.
- **Hide Outside and Inside Walls** - replaces outer and inside walls with standard tags Transparent.
- **Hide Content** - does not touch walls itself, but hides content along isometric walls.


Mode = None


Mode = Hide Outside Walls


Mode = Hide Outside and Inside Walls

🚩Remove Wall Tags

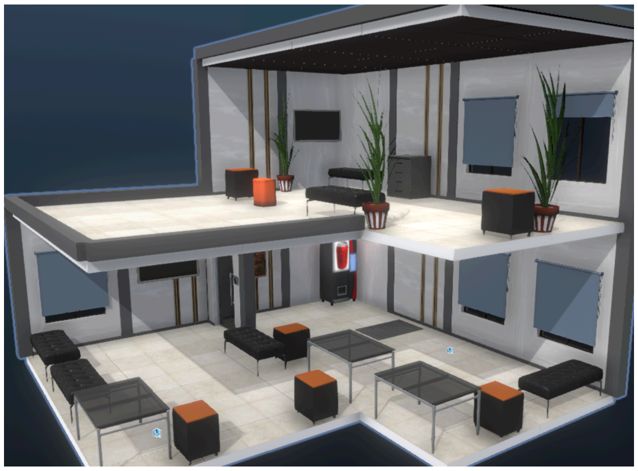
Defines a set of tags that will be replaced on tag Transparent. This is a multi-tag field. By default - Wall.
You can add other tags to replace, for example Door.

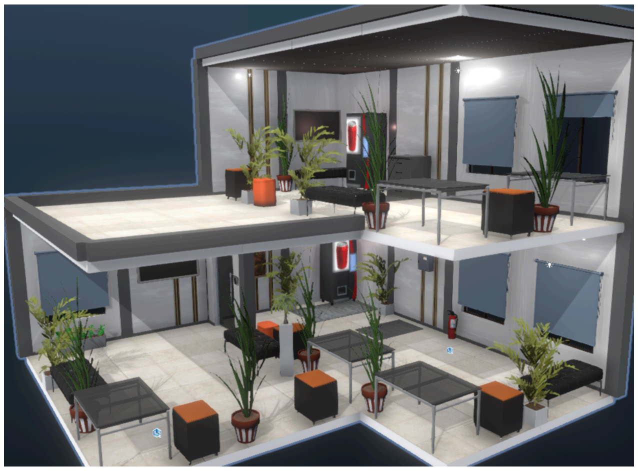
🧩Prefab Spawner

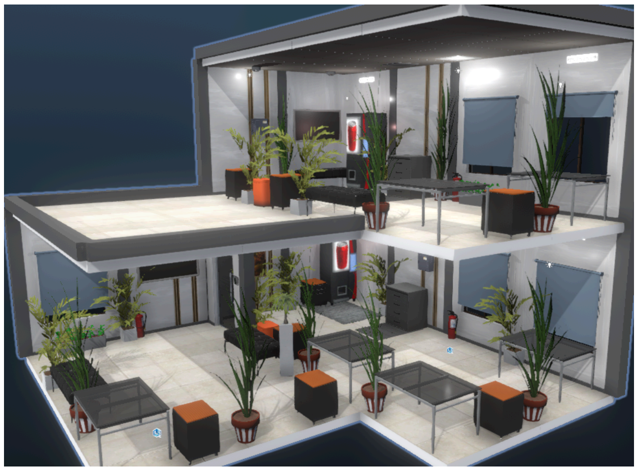
This component spawns prefabs based on map tags, following the prefab rules.
This script is automatically attached to QubicBuilder gameobject.

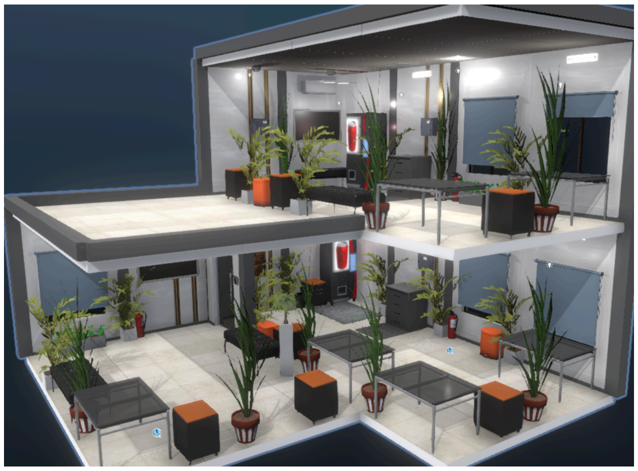
🚩Max Content Per Edge

Limits the maximum number of spawned Content prefabs on the Edge. This property affects the density and number of spawned content-type prefabs.


Max Content Per Edge = 1


Max Content Per Edge = 2

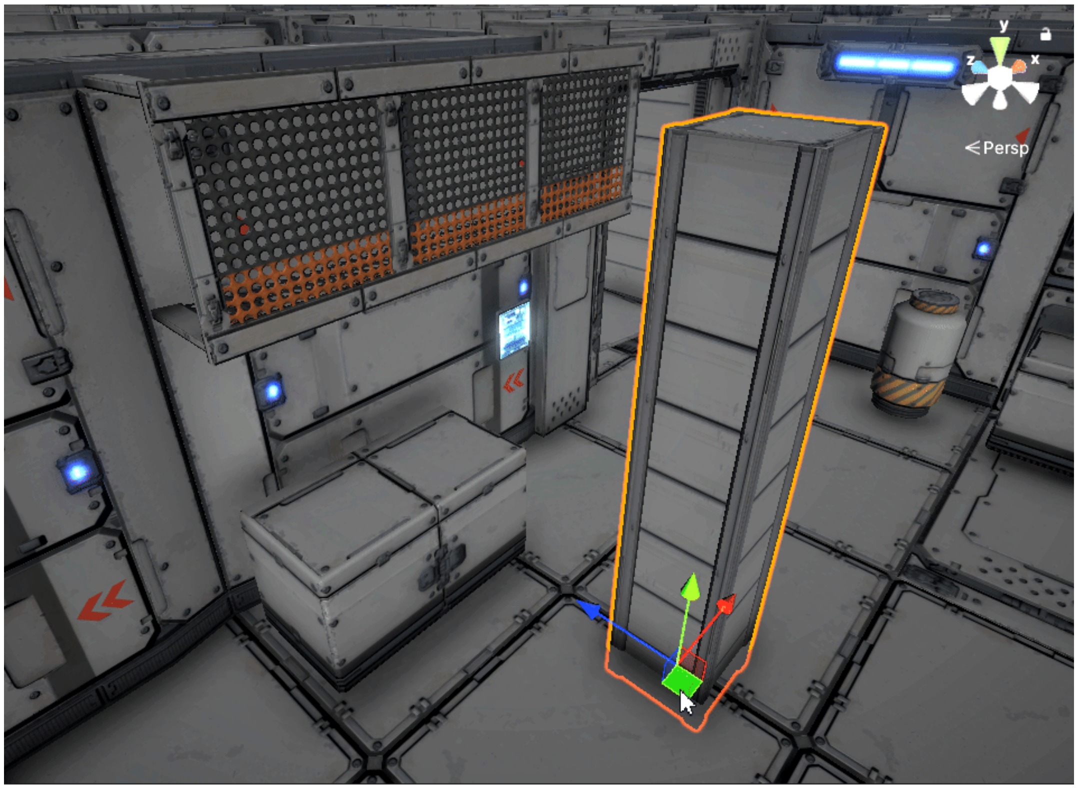

Max Content Per Edge = 4


Max Content Per Edge = 8

Note, after a certain value, the actual number of spawned prefabs stops growing. This happens because the number of prefabs is naturally limited by the intersection of bounds objects and the culling of intersecting objects. See Content Collision Avoidance.

🚩Content Blocking Layers

Defines which physics layers block content spawning in this cell. If any collider within the cell belongs to one of these layers, the cell will be considered occupied, and content will not be spawned there. This is useful for avoiding placement in areas with user defined content, obstacles, reserved space, or restricted zones.



🚩Content Blocking Mode

Defines how exactly content spawning will be blocked.

- **Intersection** - Only those prefabs that intersect with the custom object will be blocked from spawning. As shown in the animation above.
- **Whole Cell** - The entire cell that contains the custom object will be blocked from spawning.

🧩Tags Post Processing

This component adds some tags on map after all other spawners (Room, Stairs etc) spawned their tags. Based on the map, TagsPostProcessing adds such standard tags as: Outside, Inside, Roof, Parapet.
This script is automatically attached to QubicBuilder gameobject.

🧩Prefab Database

A scriptable object that saves information about prefabs.

🚩Cell Size

Size of cell in grid (meters). Usually this value is the typical width of wall prefabs.

🚩Cell Height

Height of cell in grid (meters). Usually this value is the typical height of wall prefabs.

🚩Features

- **Default Content Padding** - Default content padding for Content Walls, in meters.
- **Column Coeff** - The factor that is used to determine whether an object will spawn on a column, along X, in fractions of CellSize, from 0 to 1. By default - 0.1.
- **Full Depth Coeff** - A coefficient that is used to automatically determine "large" furniture, in fractions of CellSize, from 0 to 1. By default - 0.25.

🚩Prefabs

List of prefabs of the database. More details - see Prefab and Prefab Rule.

Preferences

Set of global preferences for Qubic. You can find these preferences in the window "ProjectSettings/Qubic", or open via the main menu "Tools/Qubic/Preferences".

🚩Auto Rebuild

Enables automatic rebuild of Qubic when you change any properties in components inside the Qubic.

🚩Highlight Cells

Highlights the cells of all rooms in the current Qubic, even if they are not selected.

🚩Highlight Cells Of SelectedRoom

Highlights the cells of the selected room.

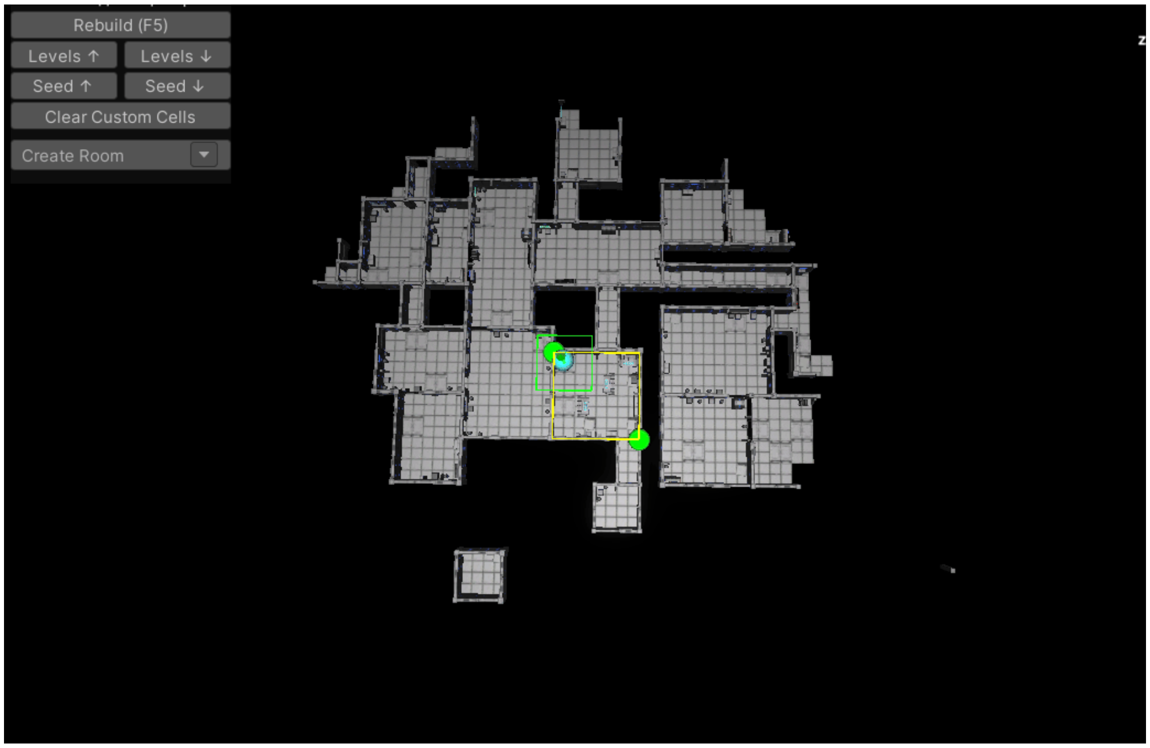
🚩Fast Mode

This option enables a super fast way to rebuild the map, where the map is rebuilt only around the selected room. This provides very fast dynamic rebuilding of the map when you work with a single room. In this mode, the speed of rebuilding is independent of the size of the map. Therefore, in this mode, you can comfortably work with a map of any size.
At the same time, pressing F5 or the Rebuild button still rebuilds the entire map.

By default - true.

Fast mode settings:

- **Radius to Rebuild** - the rebuild radius around selected room (in cells).
- **Max Level Difference to Rebuild** - the maximum spread of floors that will be built around the first floor of the selected room.
- **Min Map Cells Count** - the minimum number of cells on the map for FastMode to be enabled (5000 by default).



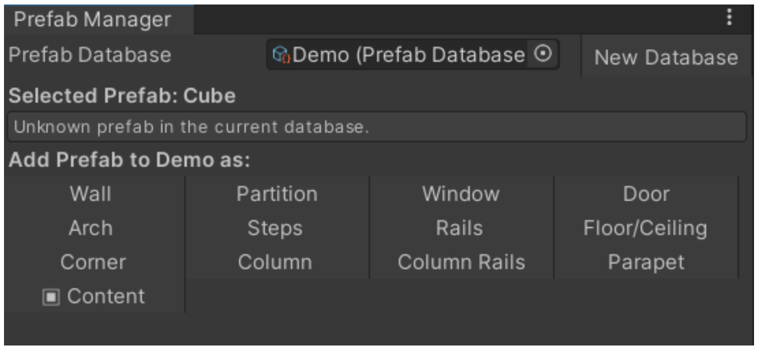
Prefab Templates

The Qubic contains a set of templates for prefabs, which will allow you quickly to add prefab to the database, assign specific settings and spawn rules for a prefab in one click. You need just select the role of prefab.

Note, this is not an exclusive list and templates do not contain any magical components. Everything that is in the templates you can do manually, simply by creating rules or by changing the rules created from the template. Templates only serve the purpose of setup prefabs with the desired properties as conveniently and quickly as possible. Also, you can expand this list, for more details see the API chapter.

To use prefab templates, you need to select prefab on scene or in Project window and open Prefab Manager window (from main menu Tools/Qubic/Prefab Manager). Note, selected object must be prefab, with blue icon: Curtain

Then, in Prefab Manager you will see list of prefab templates:



- **Wall** - A regular wall for building construction.
- **Partition** - A wall inside a building (i.e. wall between Inside and Inside cells). If you do not defined such prefab, Wall prefab will be used as inside building wall.
- **Window** - A wall with a window. By default, the window is built only for external walls. If you need to build windows inside the building, replace the Inside-Outside tag pair in the rule with Inside-Any, or Any-Any tags.
- **Door** - A wall with a door.
- **Arch** - A wall with an arch.
- **Steps** - Prefab of steps, used inside stairs structures. By default, the template assigns it as for Straight Stairs. If you want to use the prefab as U-shaped Stairs, replace wall tag Stairs with UStairs.
- **Rails** - Railing/Fence prefab that used in balconies, fences, bridges, stairs, etc.
- **Floor/Ceiling** - Prefab for floor.
- **Corner** - A prefab of L-shaped corner wall.
- **Column** - A column connector that appears at the corners of a L/T/X-shaped form, or to join walls of different types.
- **Column Rails** - A column connector for railings.
- **Parapet** - A wall prefab to spawn as a parapet on roof.
- **Content** - Prefab is content, that is: furniture, carpets, curtains, lamps, paintings and so on.

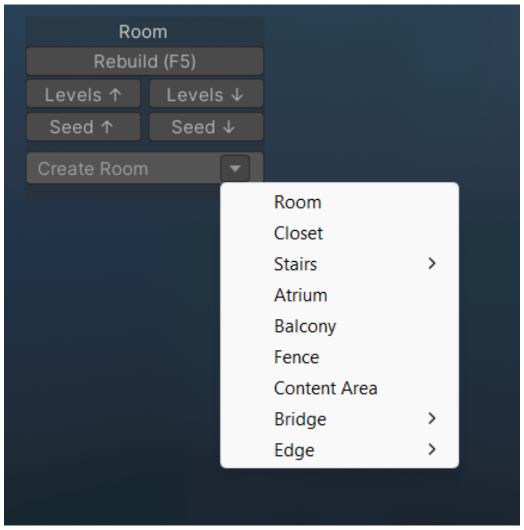
Settings for prefabs see here: Prefab and Prefab Rule.

Room Templates

Room templates are a set of ready-made structures that you can create on the scene.
They cover the basic needs for constructing a building.

Note that templates are not an exclusive list and are not magical components. Everything that is in the templates you can do manually, simply by creating components on the scene or by changing the rooms created from the template. Templates only serve the purpose of creating rooms with the desired properties as conveniently and quickly as possible. Also, you can expand this list, for more details see the API chapter.

Room templates list is appearing when you select any room inside Qubic:

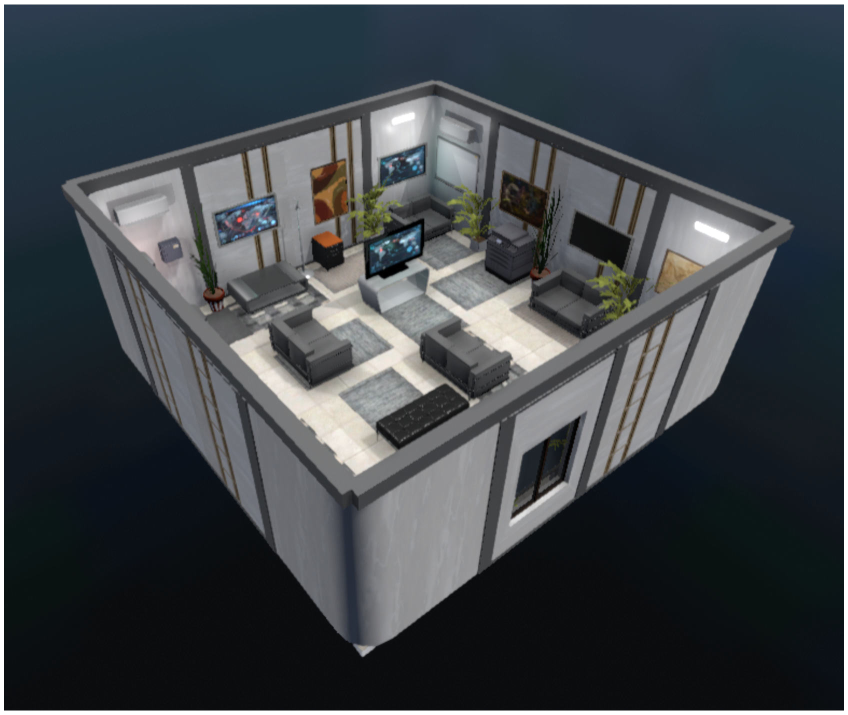


A new room will be created next to the current room, and the number of levels will be copied from the current room.

Below is a brief description of the templates:

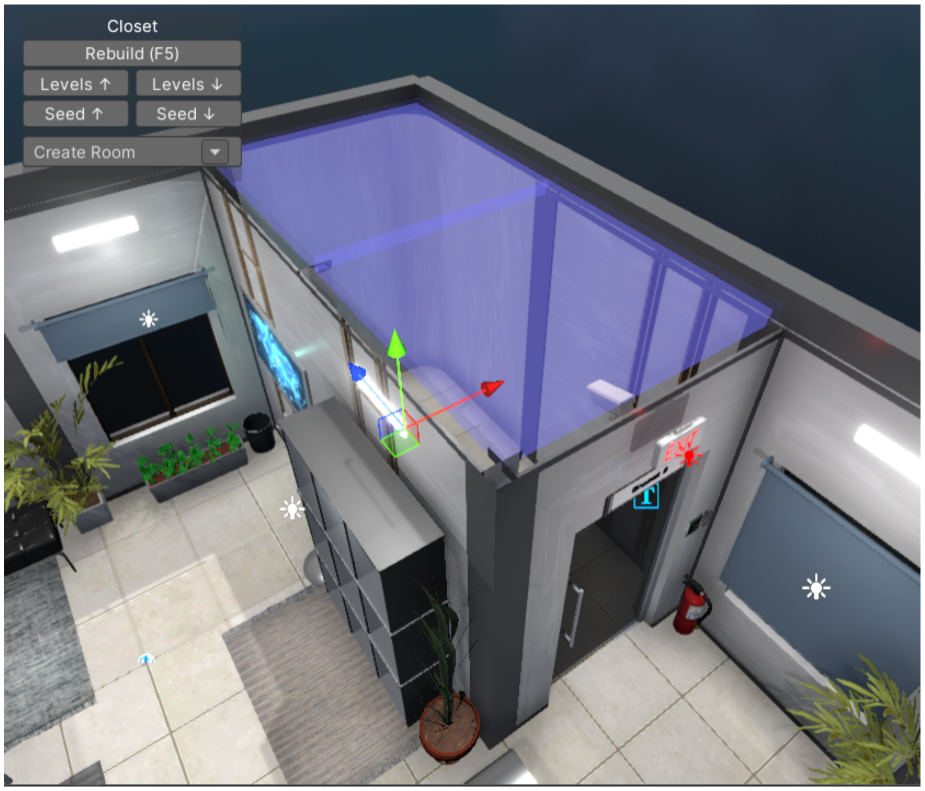
📋Room

A regular room for building construction. The most commonly used general purpose structure. By default spawns a style Room in owned cells. It is based on the Room component.



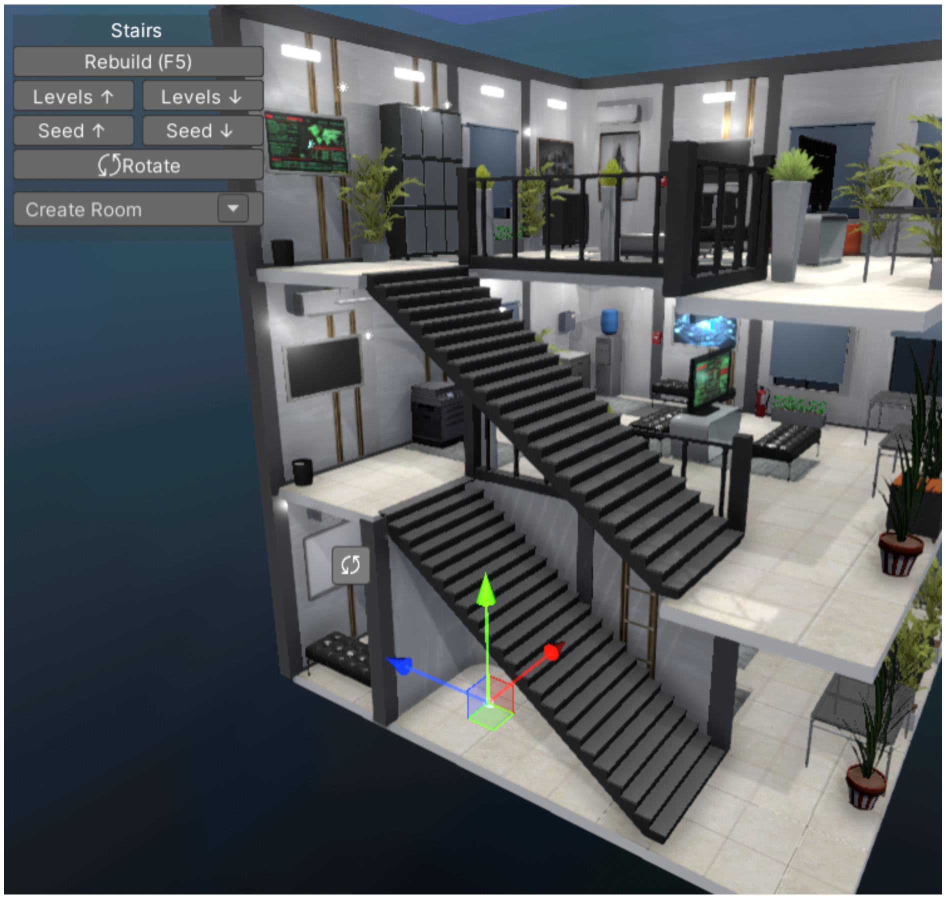
📋Closet

A room without windows. Can be used for small windowless rooms - closets, server rooms, toilets, utility rooms, etc. By default, it spawns a style Closet in owned cells. It is based on the Room component.



📋Stairs Straight

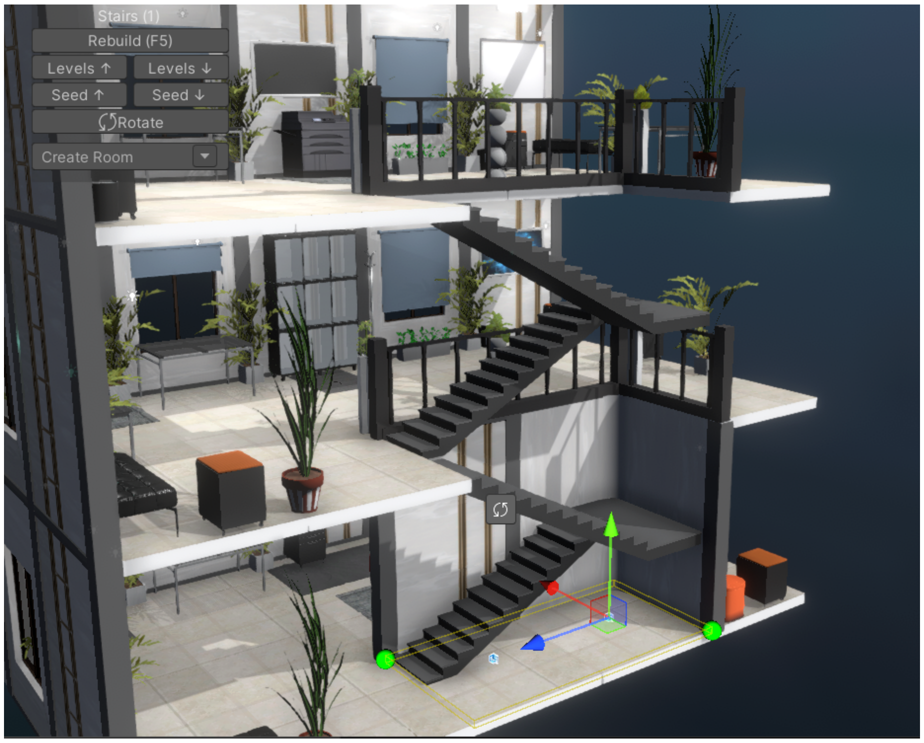
A staircase structure of straight type, based on the Stairs component. It spawns Stairs tag in owned cells. Also it spawns Steps and Passage standard tags in specific edges.



By default the Stairs template spawns walls around steps on the first level and rails on other levels.

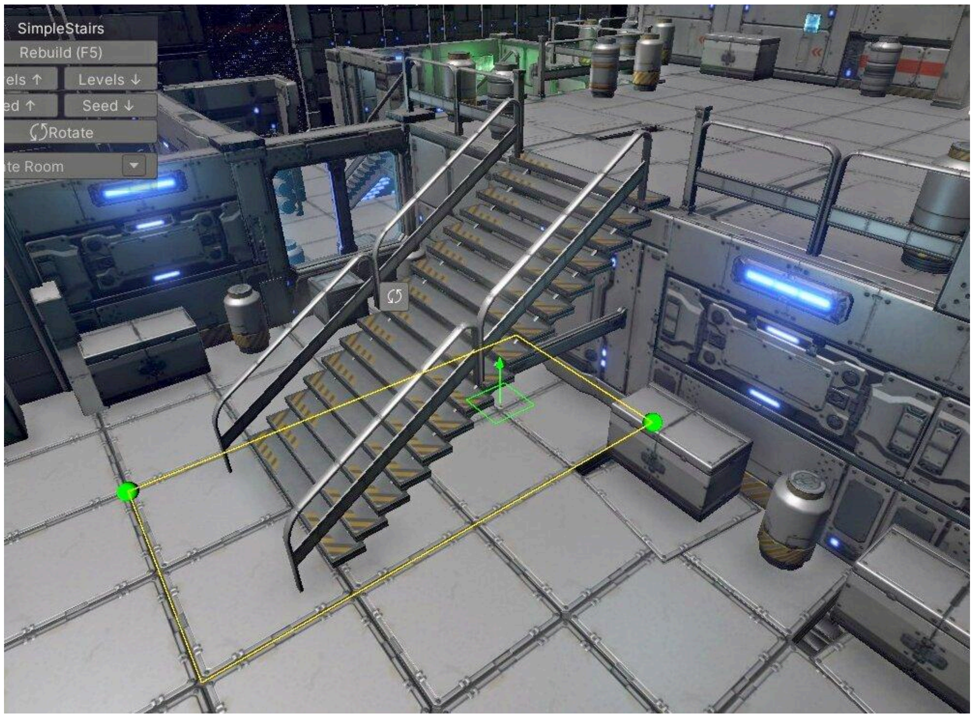
📋Stairs U-Shaped

A staircase structure of U-shape type, based on the Stairs component. It spawns UStairs tag in owned cells. Also it spawns Steps and Passage standard tags in specific edges.



📋Stairs Simple

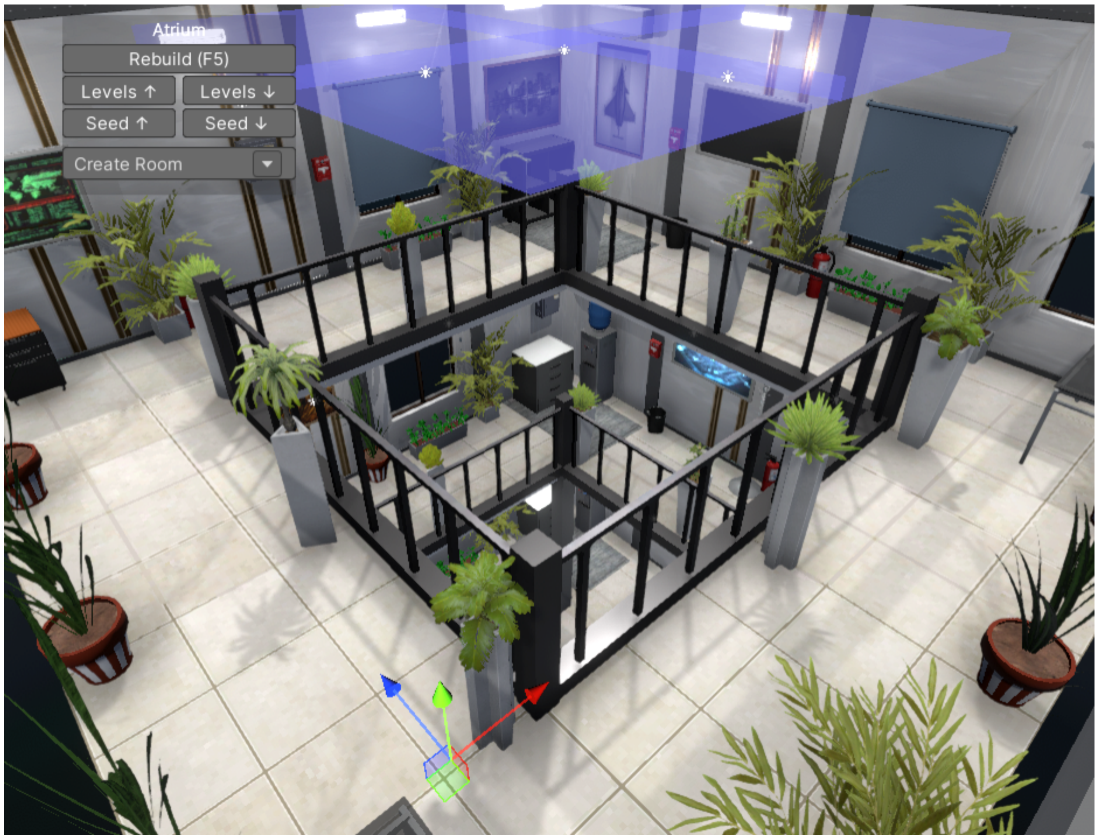
A simple staircase structure without walls around. It spawns Stairs tag in owned cells. Also it spawns Steps and Passage standard tags in specific edges.

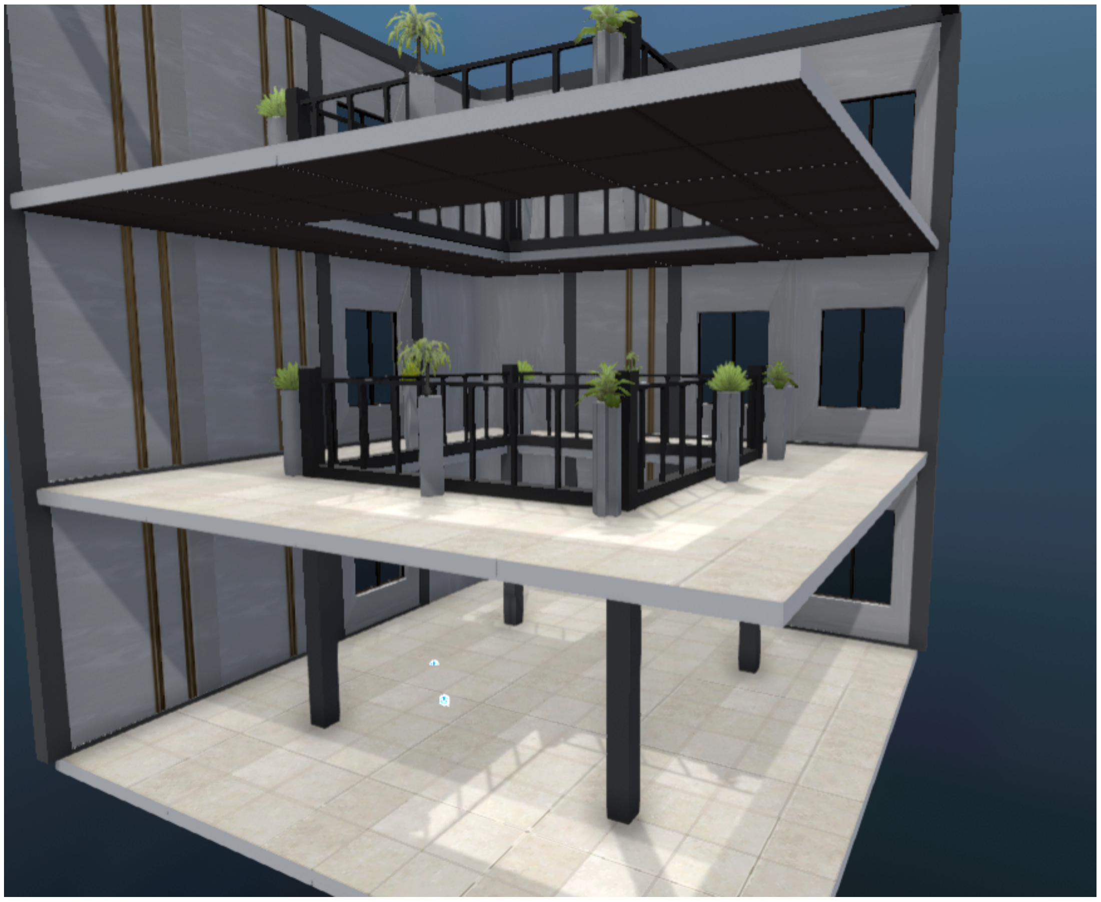


Unlike Straight Stairs, this template does not create walls around the steps. It is well suited for single-story buildings, isometric or top-down games.

📋Atrium

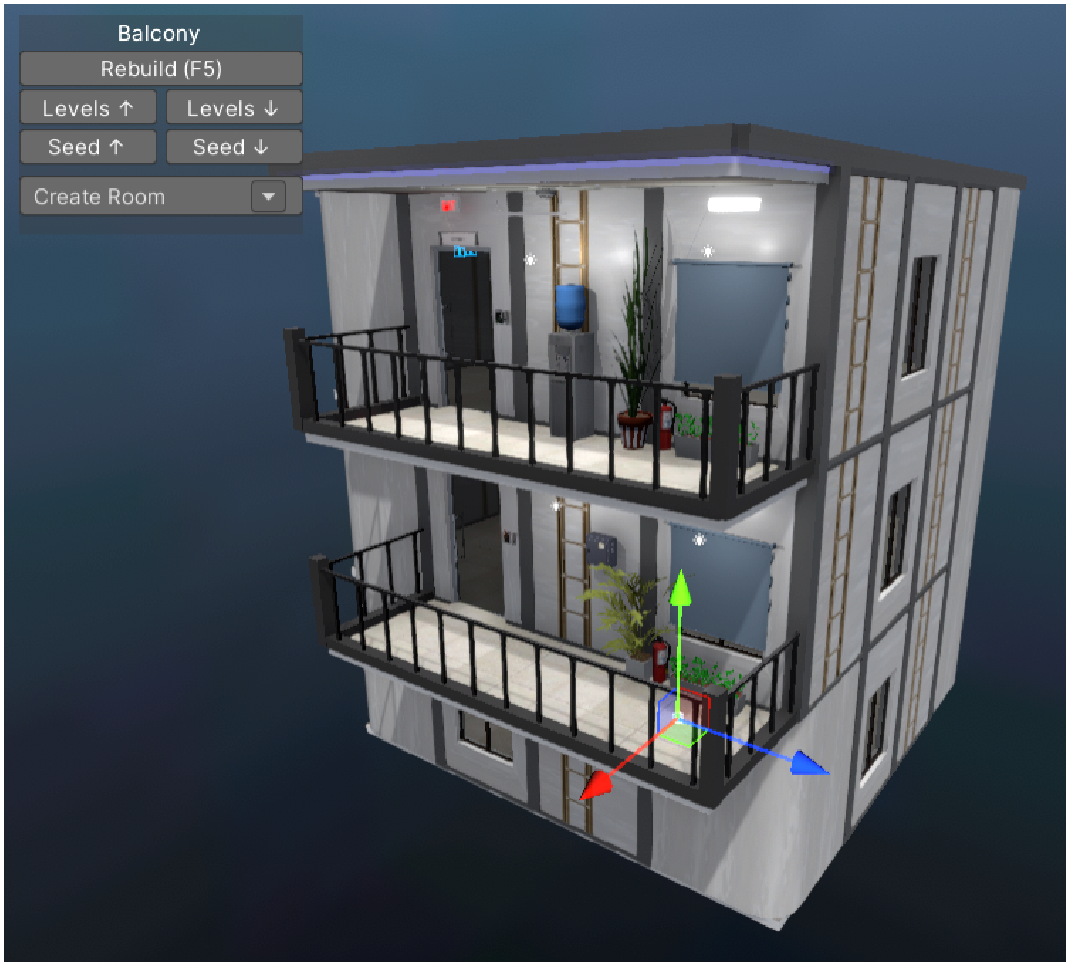
Structure to create an open space without floor inside a building. Based on the Atrium component. It spawns Atrium tag in owned cells.

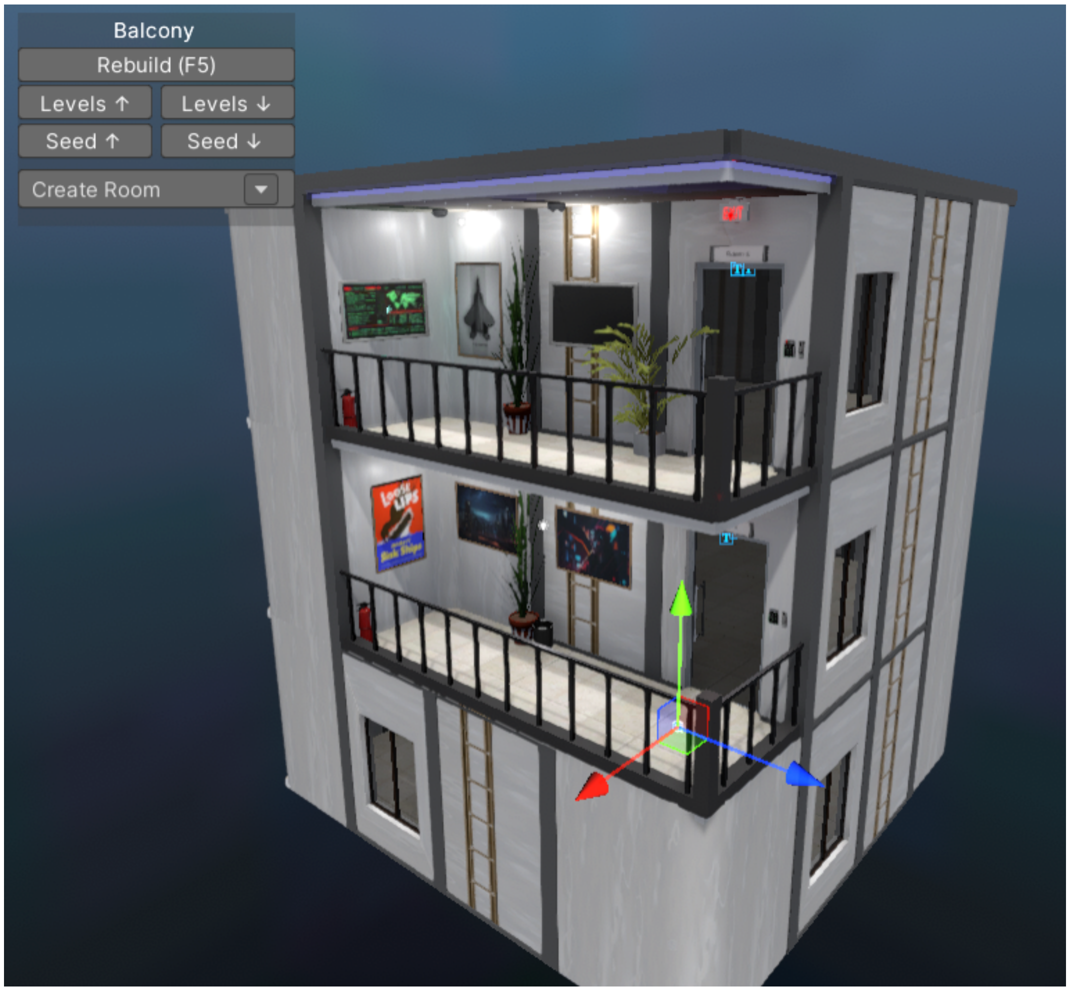



By default Atrium spawns Arches on first level and railings on upper levels.

📋Balcony

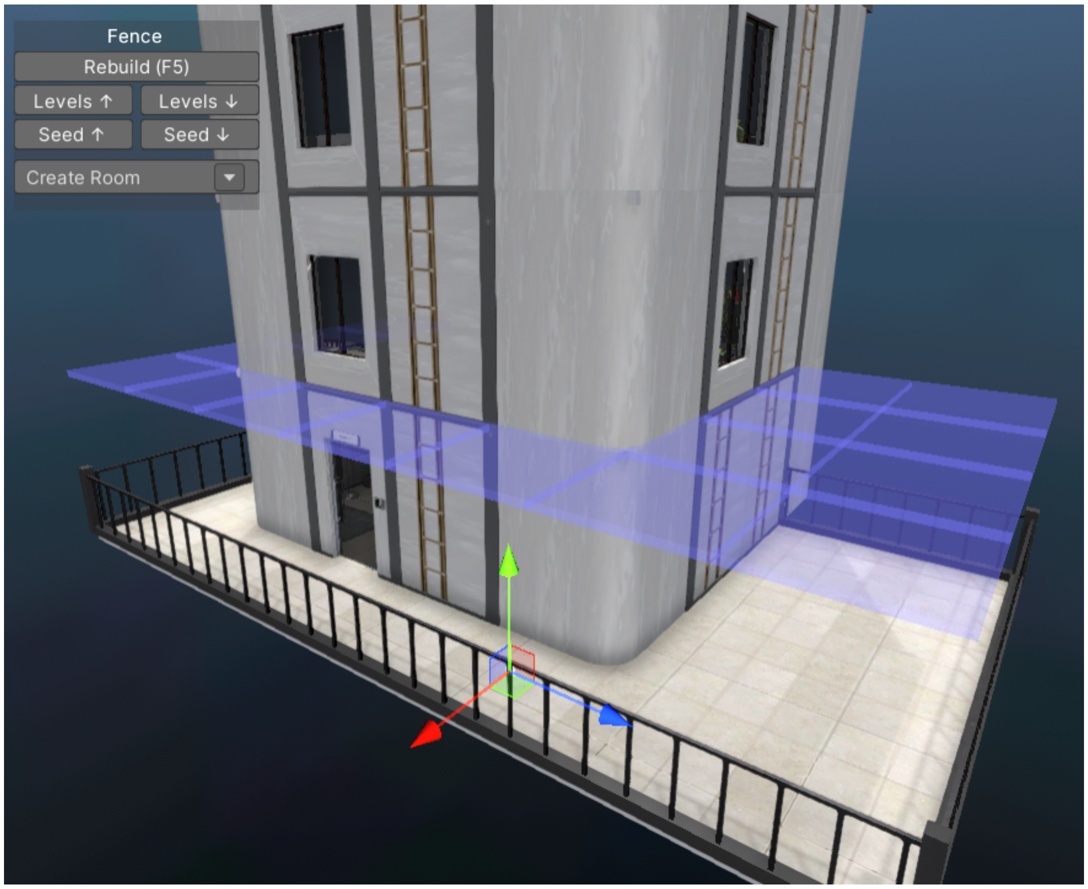
The structure with railings on outer edges. It spawns tags Room,Balcony,Outside in owned cells. It is based on the Room component.


Balcony
Rebuild (F5)
Levels ↑
Levels ↓
Seed ↑
Seed ↓
Create Room



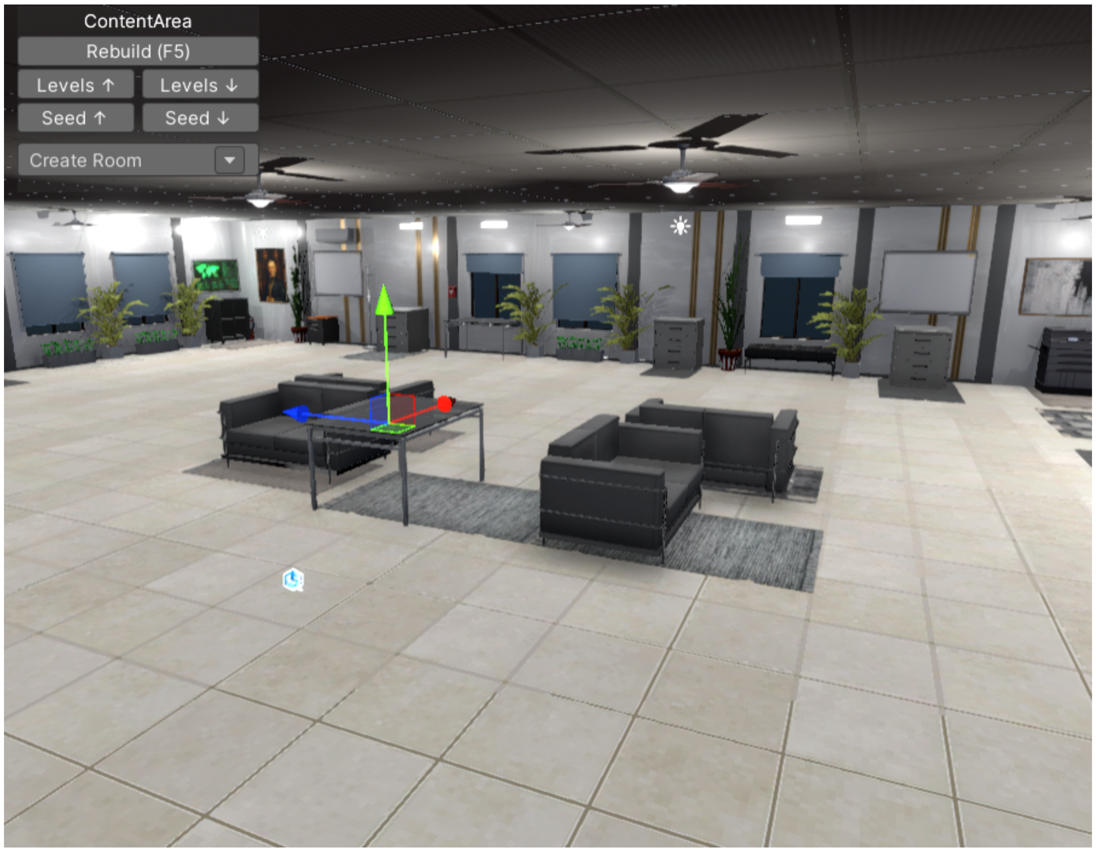
📋Fence

The structure with railings intended to make fences around buildings. It spawns tags Fence,Outside in owned cells. It is based on the Room component.



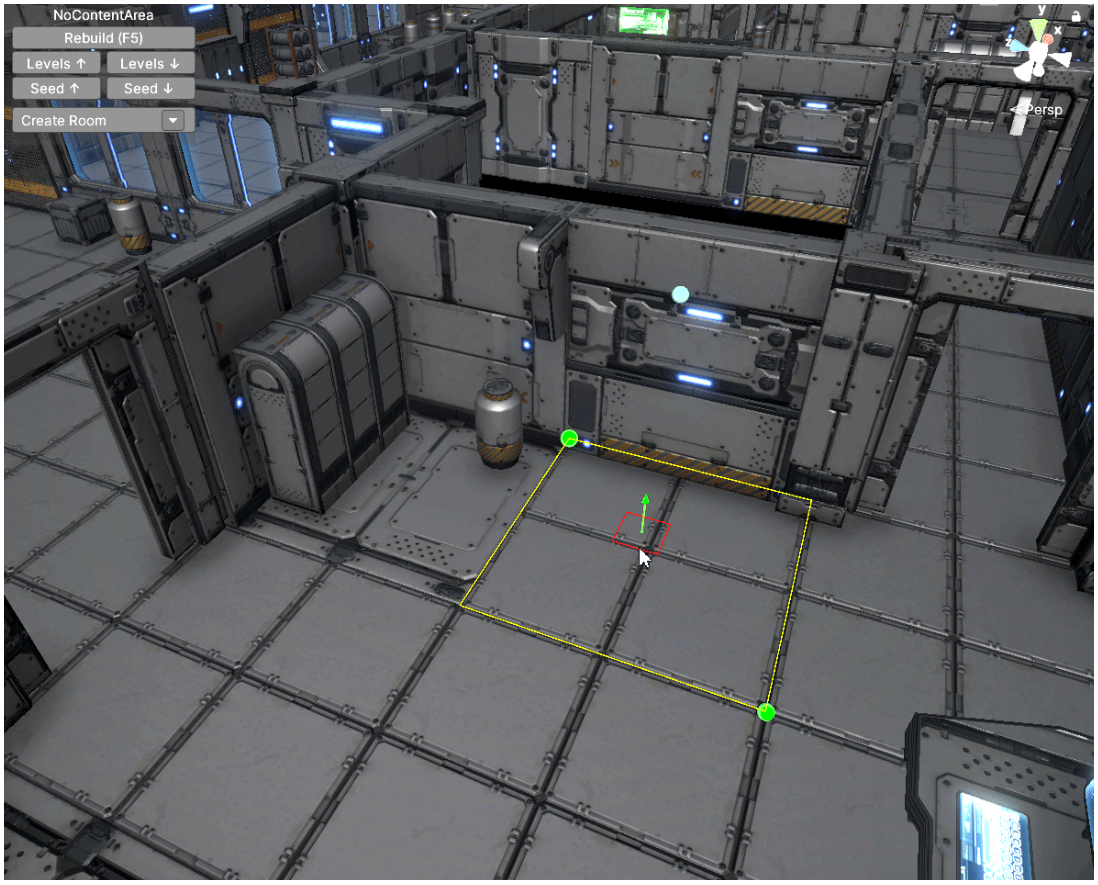
📋Content Area

A structure without outer walls designed to force content to be created in some places. It creates inside content walls with the default Room style. It is based on the Room component. See also Forced Content template.



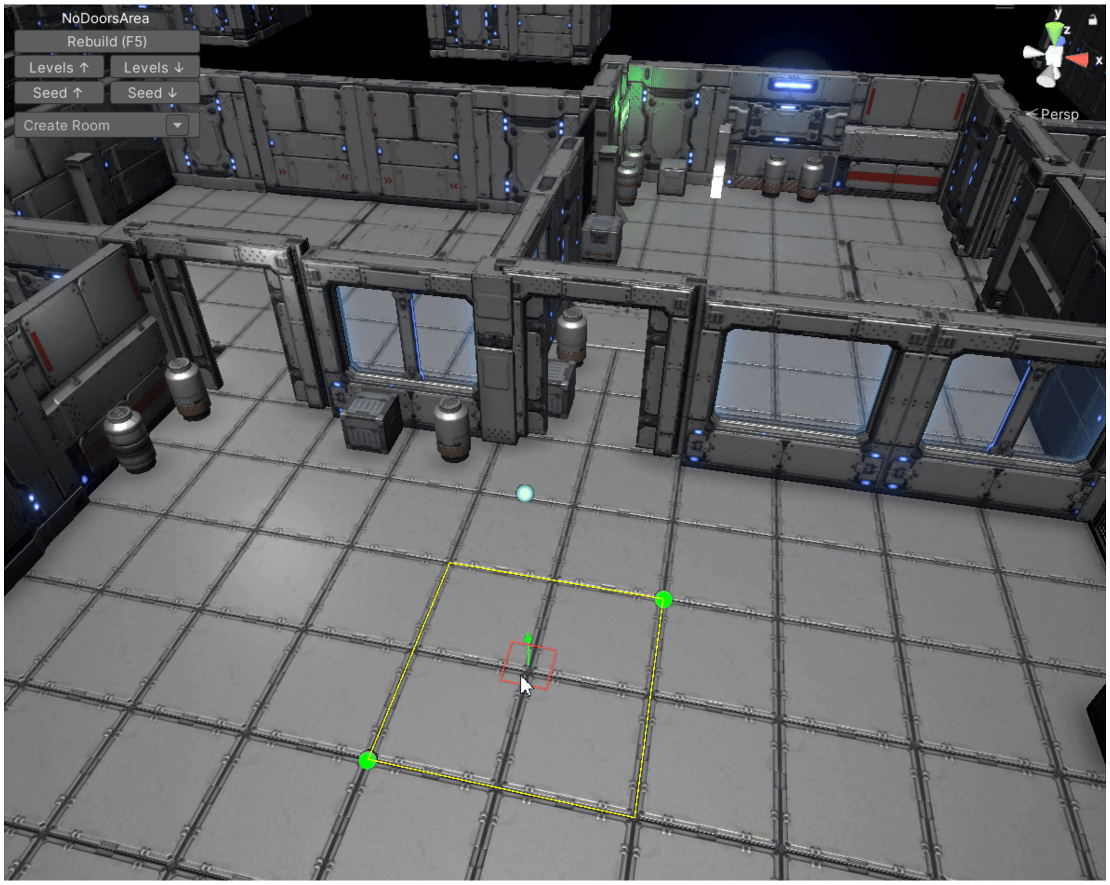
📋No Content Area

A structure designed to make it impossible to create content (either wall or inside).



📋No Doors Area

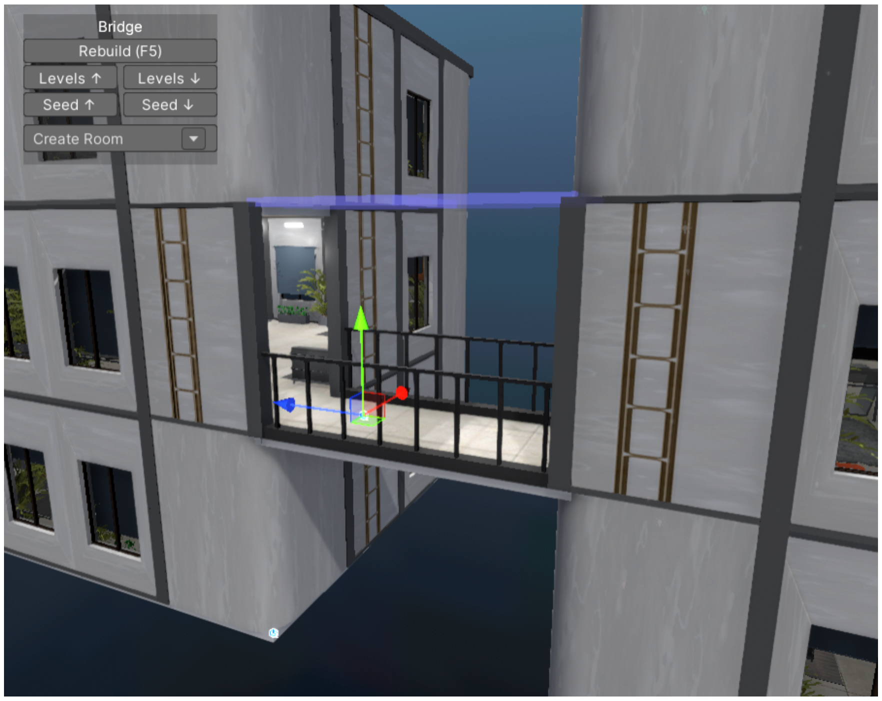
A structure designed to disable doors to be created in defined cells.

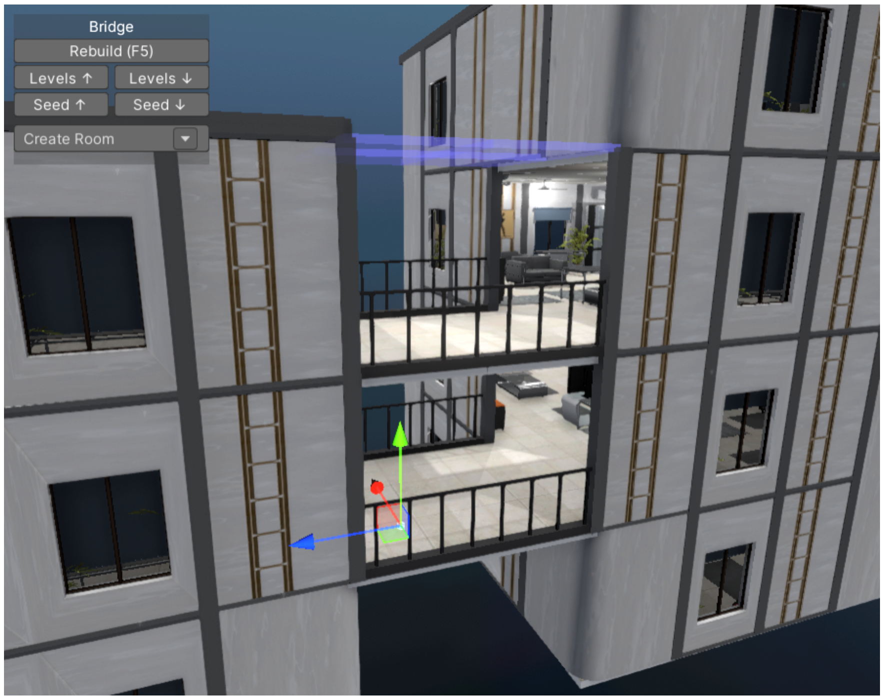


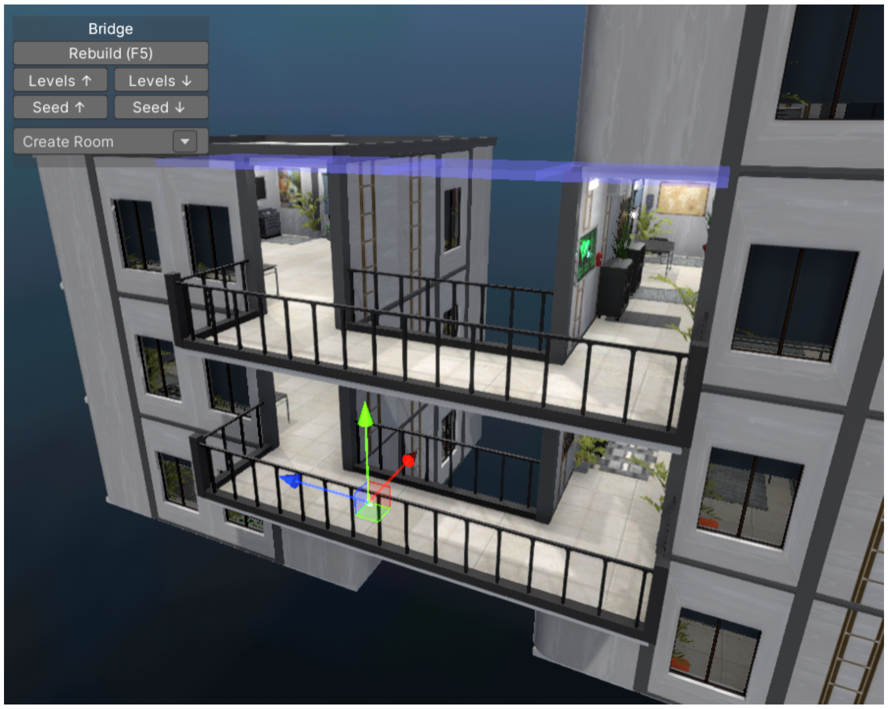
📋Bridge Regular

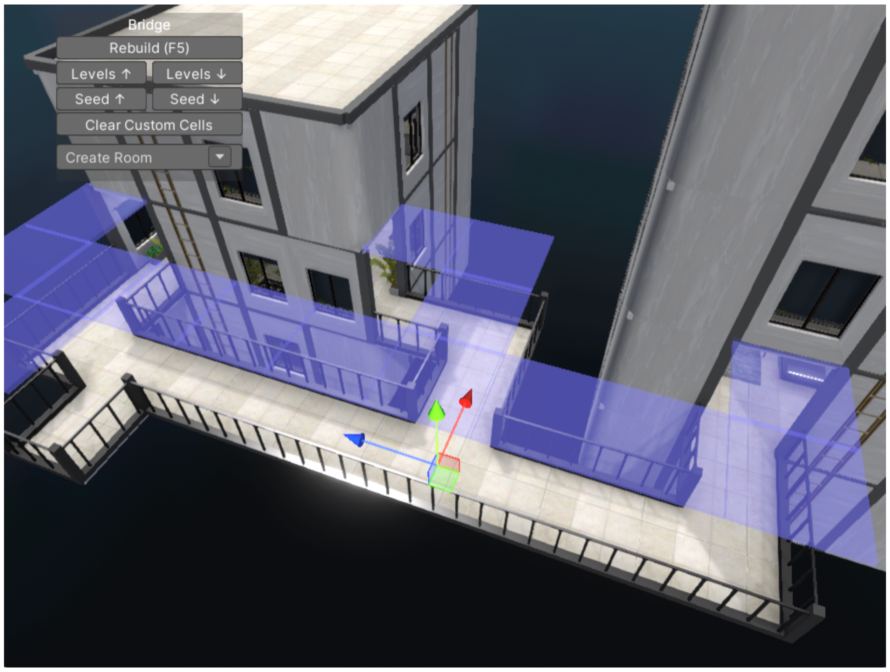
A structure designed to make bridges and passages between buildings. It spawns tags Bridge,Outside in owned cells. It is based on the Room component.

The bridge automatically creates an arch where it connects to the building. External walls are spawned as railings.
The bridge can be either single-story or multi-story.


Bridge
Rebuild (F5)
Levels ↑
Levels ↓
Seed ↑
Seed ↓
Create Room

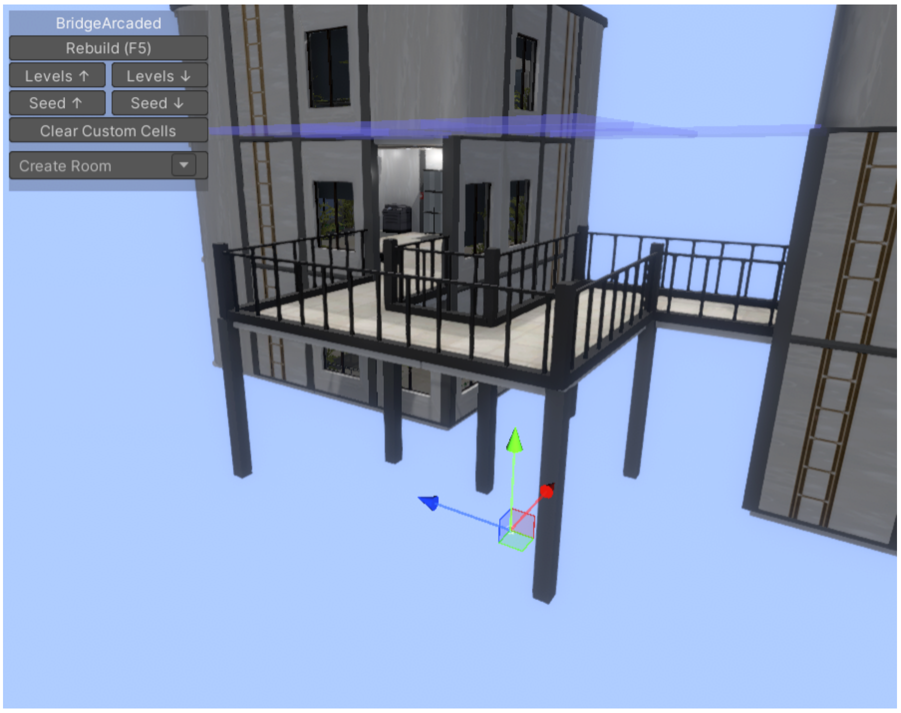

Bridge
Rebuild (F5)
Levels ↑
Levels ↓
Seed ↑
Seed ↓
Create Room





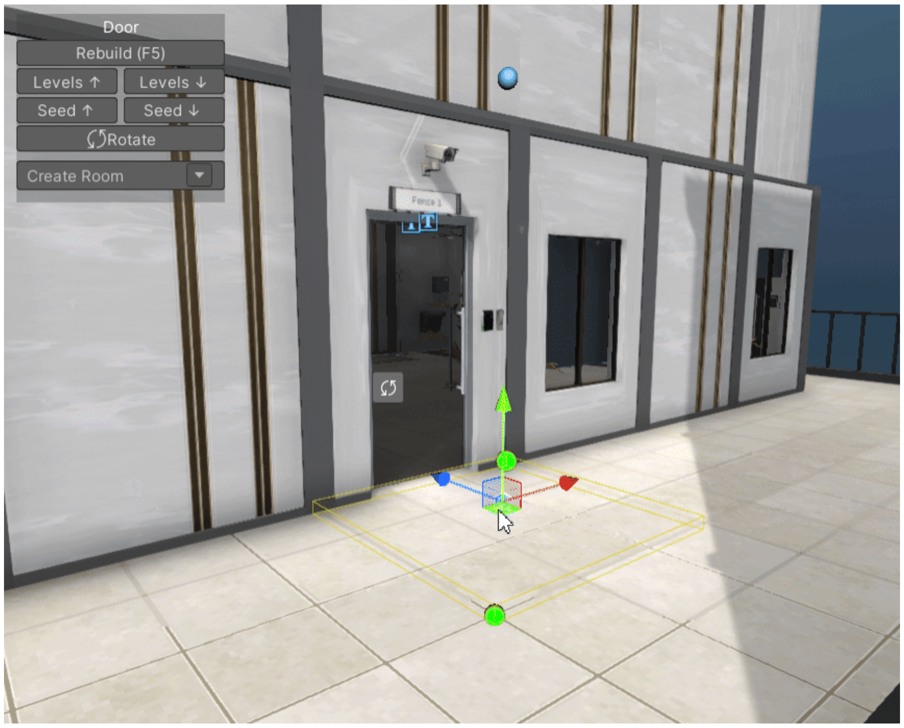
📋Bridge Arcaded

A structure similar to a Bridge, but the first floor of the structure spawns arcs and columns.



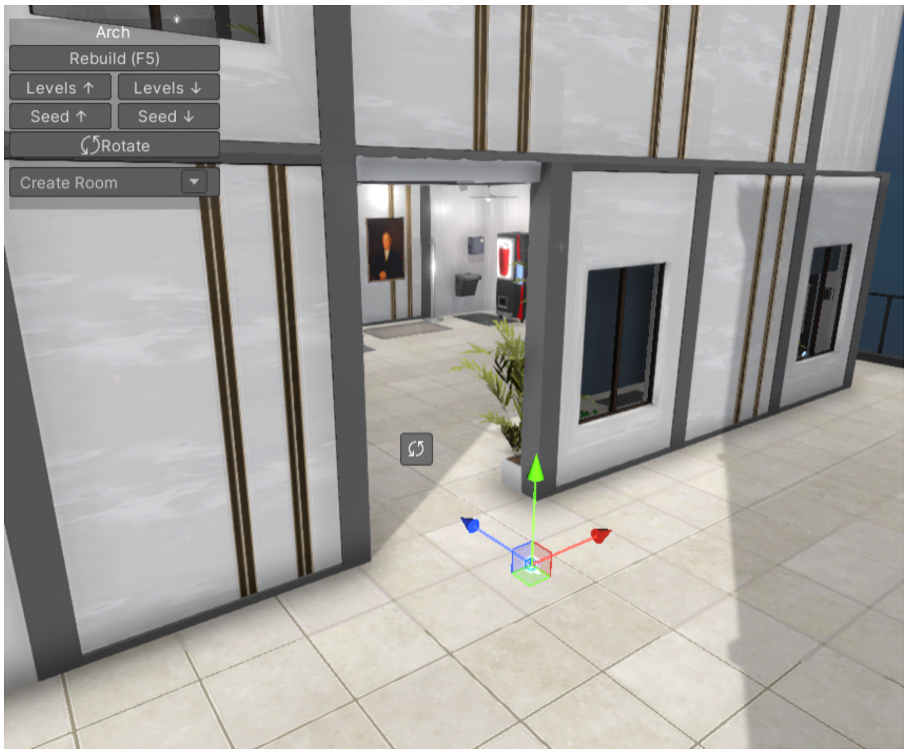
📋Forced Door

This template will force a door to spawn at the specified location. It is based on the Wall component and spawns standard Door tag in Edge.



📋Forced Arch

Like Forced Door, this template spawns Arch at the specified location. It is based on the Wall component and spawns the Arch tag in Edge.

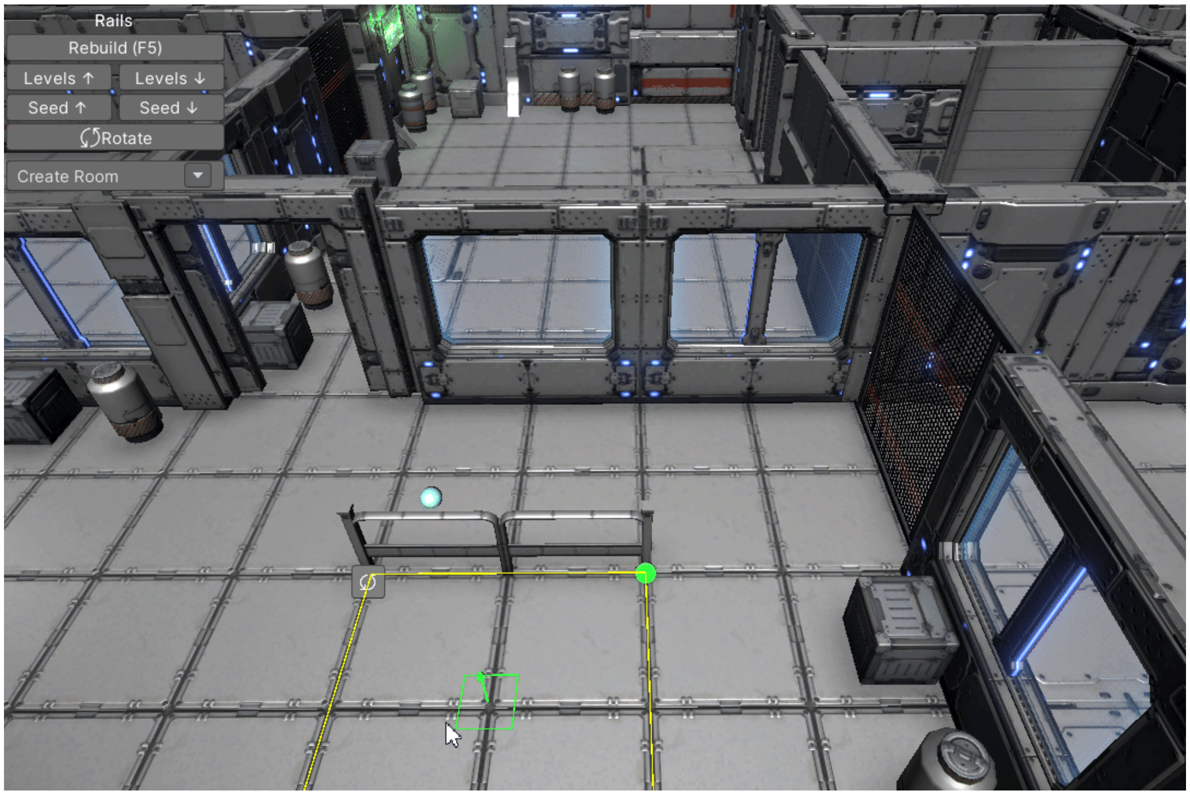


📋Forced Window

Like Forced Door and Arch, this template spawns Window at the specified location. It is based on the Wall component and spawns the Window tag in Edge.

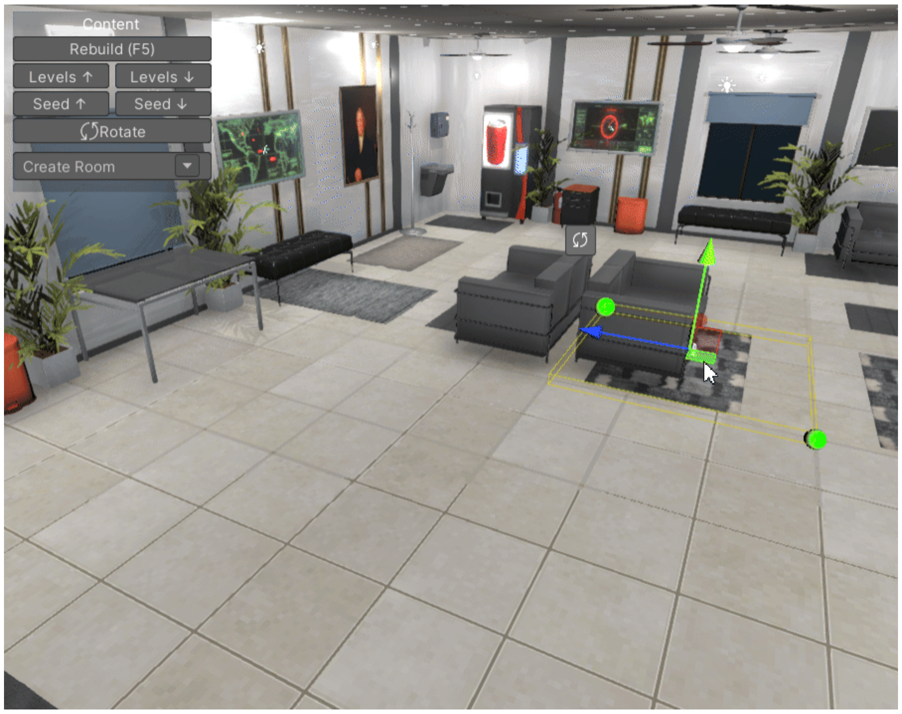
📋Forced Rails

This template spawns Rails at the specified location. It is based on the Wall component and spawns the Rails tag in Edge.



📋Forced Content

This is similar to Content Area structure, but spawns exactly one content wall in specified direction. It is based on the Wall component and spawns a standard Content tag in Edge.



Addons

Just copy addon file into Qubic folder:

RandomRotation.cs - Allows you to add random Y-axis rotation to content prefabs.

🆘Support

Discord: https://discord.gg/BPFFCXcmka
E-mail: tps99tps@gmail.com