Cozy House Generator

You are welcome to **Cozy House Generator** - a houses procedural generation system.

Also, please note that the author of this documentation is not a native speaker, so if you find some grammatical or semantic mistakes - please let me know! This will help **Cozy House** to become better!

Contact: CuddleCatsTeam@gmail.com.

What to expect?

- SEED number based random system.
- Customizable generation pipeline (propgramming skills don't necessary).
- Multiple floors.
- You can change size of the house.
- Anything can be replaced (including windows, walls, doors, etc...).
- Different interiors.
- Props depends on the interior.
- You can add your own props.
- You can make custom props place rules (propgramming skills don't necessary).
- You can add your own interiors also.
- Mesh combiner include (use Combined Builder).
- LODs include.
- Mesh combiner can also combine LODs if you need it.
- You can add some exterior parts like porch or air conditioning with Decorators system.
- Source code include.

Limitations

- No stairs. Stairs will come with v1.4 in Summer 2019.
- Windows and external doors generated by little strange way and don't look as good as it can. Will be fixed in the future.
- Current example cell didn't model good enough at all. Will be fixed.
- Undo Redo editor functions didn't work. Will be fixed.

Package structure

- **Assets** - contains 3D stuff and some settings.
  - **Animations**
  - **LODs**
  - **Materials**
  - **Meshes**
    - **Furniture** - props meshes.
    - **House Cell** - wall meshes, door meshes, window meshes, etc..
  - **Pipeline Presets** - contains pipeline settings presets.
  - **Textures**
- **Demo Files** - contains prefabs and scriptable objects with settings. Can be removed.
  - **Demo** - files for regular house demo.
    - **Interiors** - interior scriptable objects.
    - **Prefabs**
      - **Cell Parts** - walls, doors, windows etc.
      - **Furniture** - props prefabs.
      - ***Cell*** - cell prefab.
      - ***Wall*** - wall prefab.
    - **Props:** props scriptable objects with prefab link and place rules.
  - **Demo without LODs** - same as **Demo** but without LODs.
- **Editor** - inspector UI scripts and graphics.
- **Scripts**
  - **Core** - contains house blueprint scripts. Not include MonoBehaviour scripts.
  - **Demo**
  - **Gizmos**
  - **House Builders** - contains classes which build a house from a blueprint.
    - ***BuilderData.cs*** - data bus for builder (blueprint, list of interiors, size etc.).
    - ***CombinedBuilder.cs*** - optimized builder which combines meshes and LODs.
    - ***IHouseBuilder.cs*** - builder interface.
    - ***RawBuilder.cs*** - non-optimized builder which just instantiates cell predabs. Use it for debug.
  - **Scriptable Objects** - interiors, props, pipeline presets, etc.
  - ***Decorator.cs*** - this component uses to add some details for cell parts like cornice, porch etc.
  - ***HouseCell.cs*** - constructs the base cell of the house.
  - ***HouseGenerator.cs*** - generates a house.
  - ***LodsCombiner.cs*** - combines prop LODs into one LOD.
  - ***PropsRandomizer.cs*** - component that allows you to make props more chaotic.
  - ***SubProps.cs*** - makes prop child parts more chaotic. Required **Props Randomizer**.
  - ***Wall.cs*** - constructs a wall.

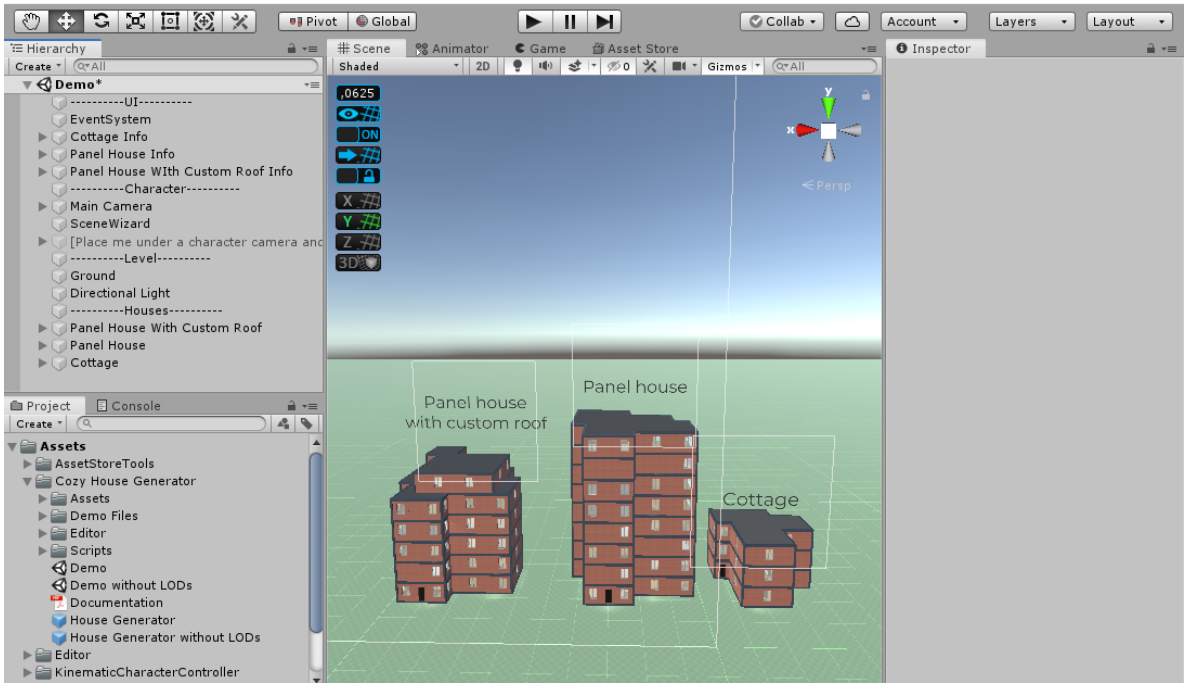
- ***Demo*** - the demo scene.
- ***Demo without LODs*** - the demo scene with houses without LODs.
- **Documentation**
- ***House Generator*** - House Generator prefab. Just drag it onto the scene and use.
- ***House Generator without LODs*** - House Generator prefab without LODs.

Quick start

To start using the Cozy House you can open the **Demo scene** or drug and drop the **House Generator prefab** onto your own scene.

> If you don't want to use LODs use **House Generator without LODs** prefab or open **Demo without LODs** scene.

After open the **Demo scene** you'll see this:



> If you want only houses you can remove all objects from scene except Houses.

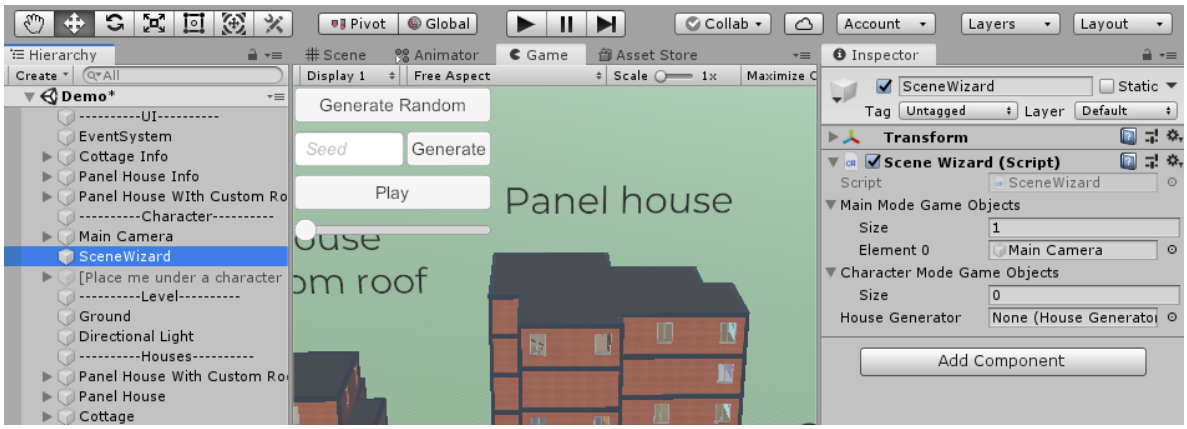
----------UI----------

Here are UI objects which shows an info about houses

----------Character----------

Main Camera - the main camera which shows houses from a top view. Also, includes Generation Settings UI panel as a child object.

Scene Wizard - switches Game Mode by disabling and enabling game objects.



> If current Game Mode is a Main Mode, after switching all Main Mode game objects will be disabled, and all Character Mode game objects will be enabled.
>
> **Play UI Button** just switches a Game Mode.
>
> To generate a new house during a play mode just press an "**R**" key.

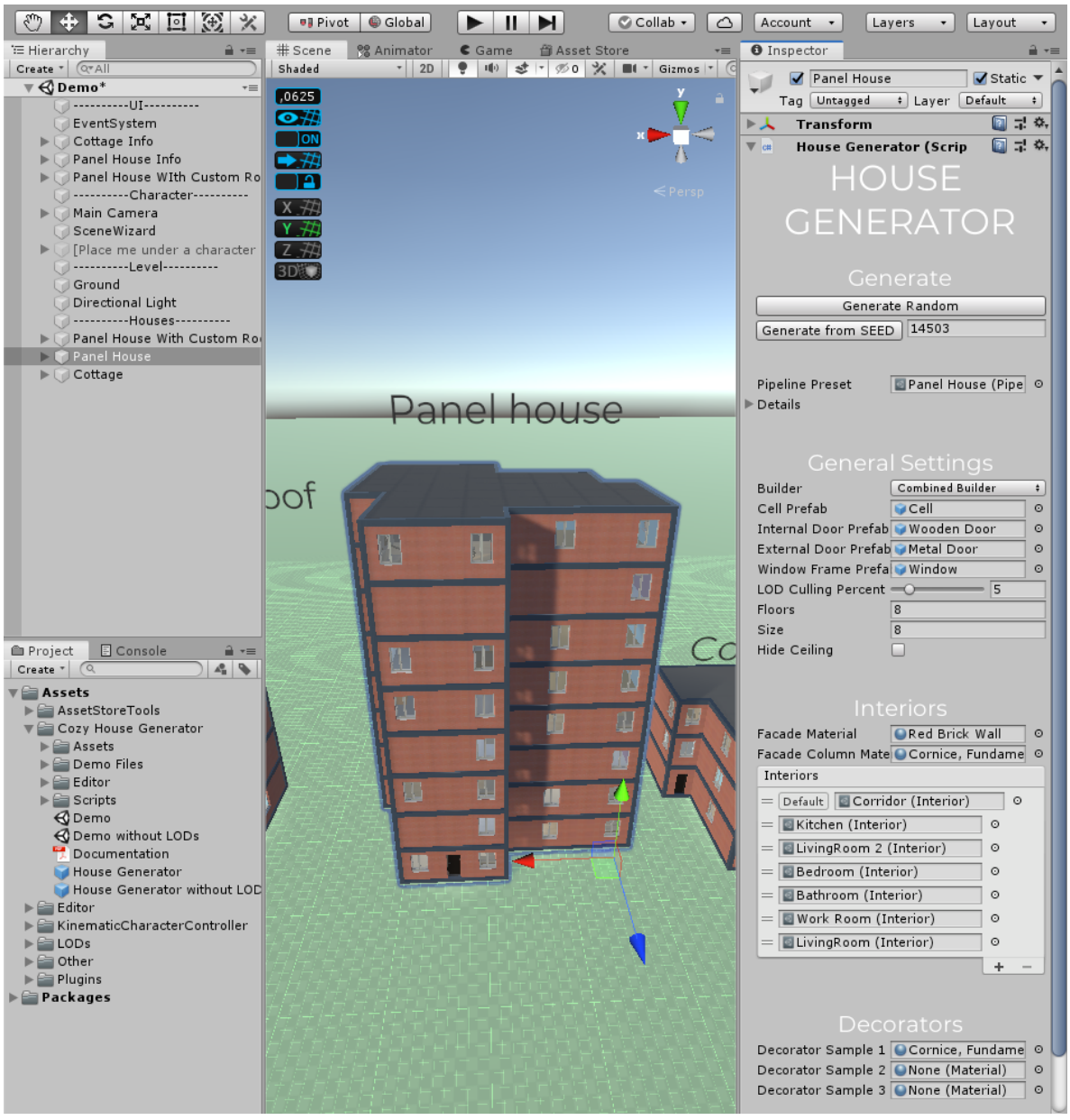
[Place me under a character camera and active it] - drag it under a character camera and active it to be able to open doors.

----------Level----------

Some level objects.

----------Houses----------

Configurated House Generators.
Select some of them an you'll see this panel:



Now you can press a **Generate Random button** to see the effect.

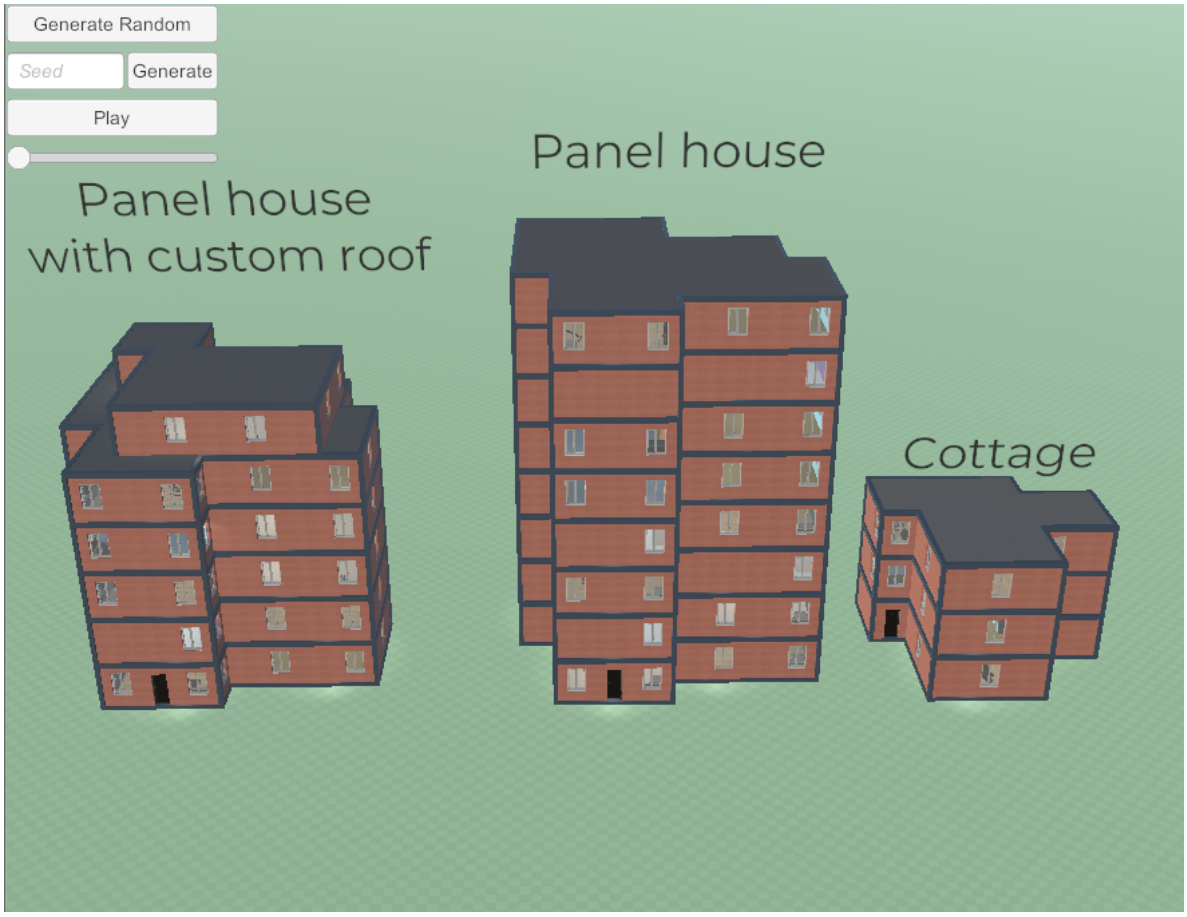
Also, you can write any number into SEED panel and press **Generate from SEED button** to generate a house from your own number. This house will be the same every time because the generation process is determined. You even can save it to use later or send to someone to build the same house. But if you change any generation settings the result will be another! Don't use it

to save the world, but use it to send a house in multiplayer.

Change **Pipeline Preset** to see the difference between them, and change **Size** and **Floors** to resize the house.

All other settings will be explained in the **House Generator** section below.

After pressing the **Play** button you'll see this:

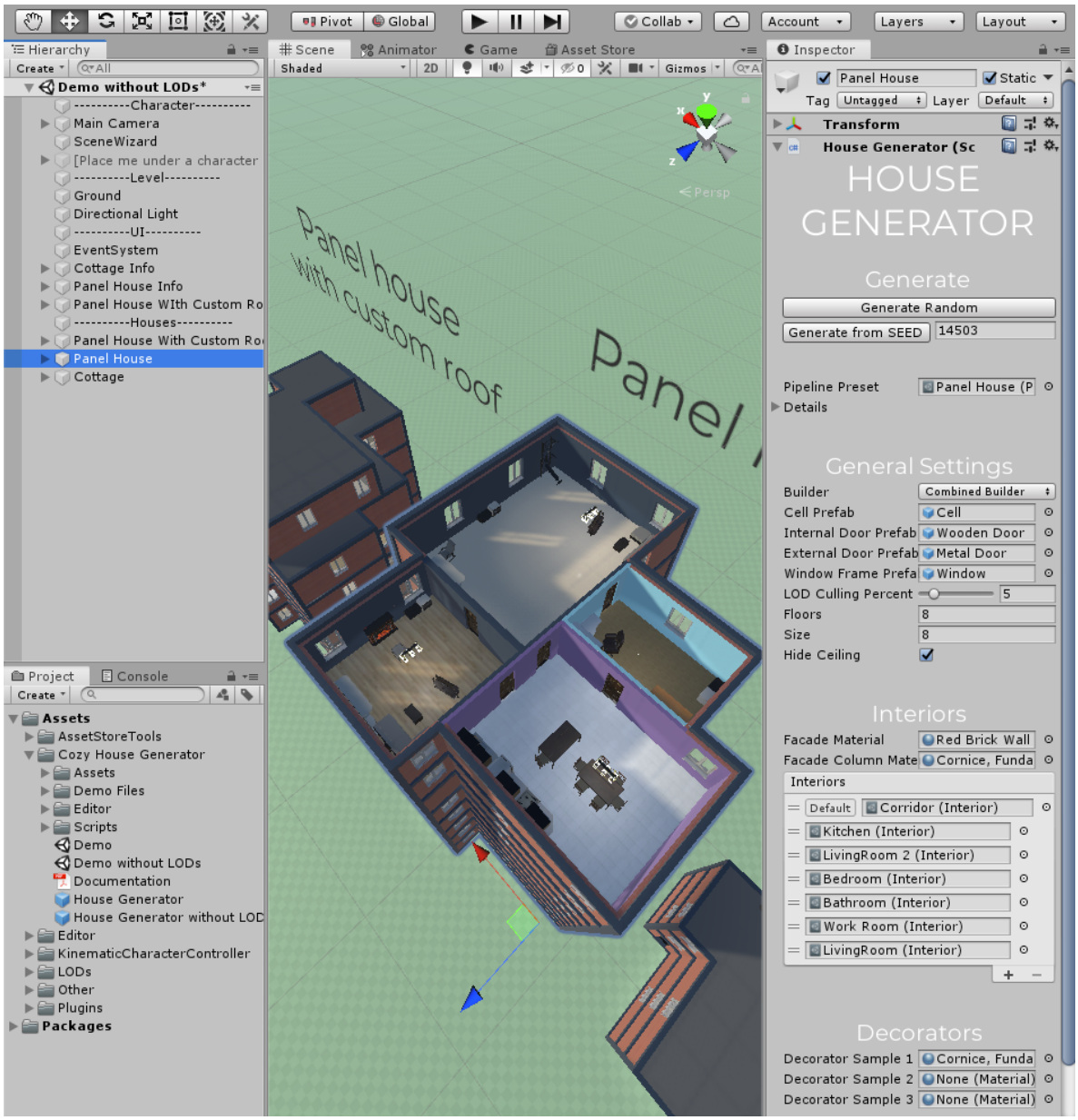


Get some fun with it!

How to configure?

House Generator

House Generator is the main component responsible for generating the house.



There are several ways to use the **House Generator**:

1. Using the **Generate panel**. This panel includes the **Seed** field, the **Generate from SEED** button and the **Generate Random** button. Pressing one of the buttons will start the generation at house. **House Generator** uses a special number, called **Seed**.

   > A **random seed** (or **seed state**, or just **seed**) is a number (or vector) used to initialize a pseudorandom number generator.

> For a seed to be used in a pseudorandom number generator, it does not need to be random. Because of the nature of number generating algorithms, so long as the original seed is ignored, the rest of the values that the algorithm generates will follow probability distribution in a pseudorandom manner.
>
> Wikipedia

Using the same **seed** number we will always get the same result. To create a random seed number before each generation, use the **Generate Random button**.

> Also, an extremely useful property of **seed** is the fact that you can use it for storage and transfer a house. This can be useful in multiplayer games, because in order to transfer the house generated on the server to the client, all you have to do is to transfer the **seed** number, and the client will generate an absolutely similar house on its basis.

2. By calling the **Generate()**, **GenerateWithSeed(int seed)** or **GenerateWithRandomSeed()** function on the **House Generator** component. It also works as a Generate panel.

```
public HouseGenerator house;

public void Generate()
{
     house.Generate();
}
```

> If you'll use a Generate() function the seed will be got from the house.seed field.

or

```
public HouseGenerator house;

public void Generate()
{
     house.GenerateWithRandomSeed();
}
```

or

```
public HouseGenerator house;

public void Generate()
{
     house.GenerateWithSeed(1337);
}
```

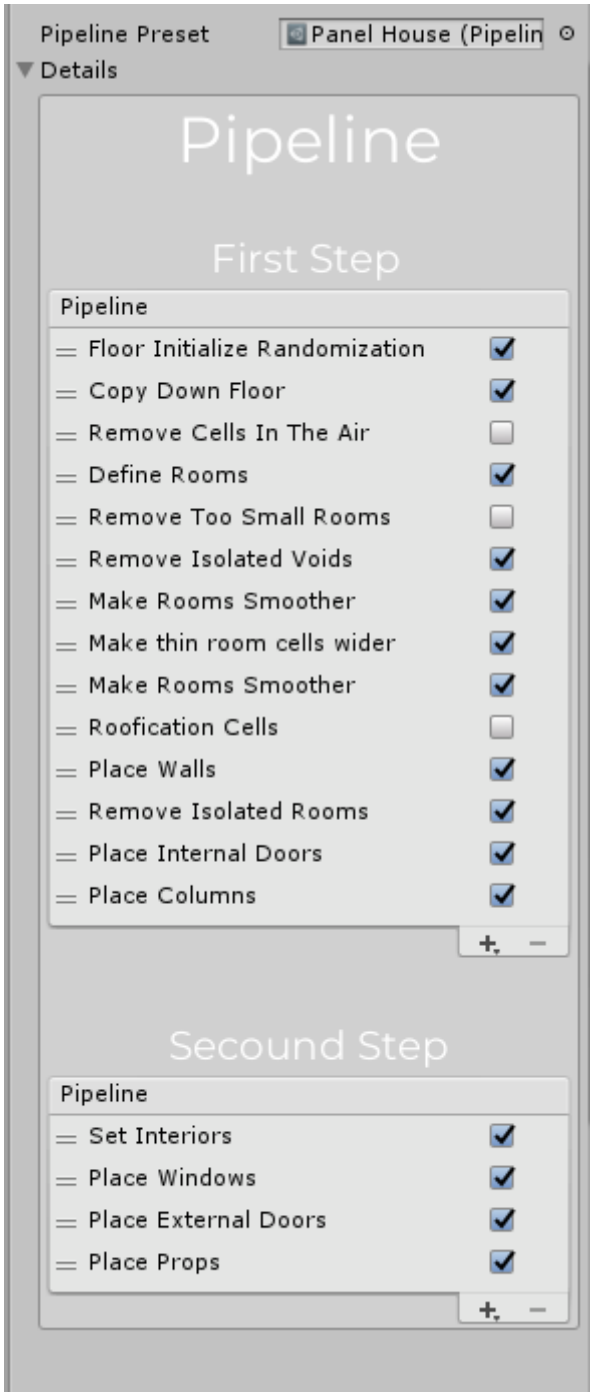
3. Using the **Example Generator UI** panel.

The following panels are used to set up generation:

Pipeline Preset

Pipeline Preset is a scriptable object which contains **a list of pipes** used to process the **blueprint** of the house.

To see more click **Details**.



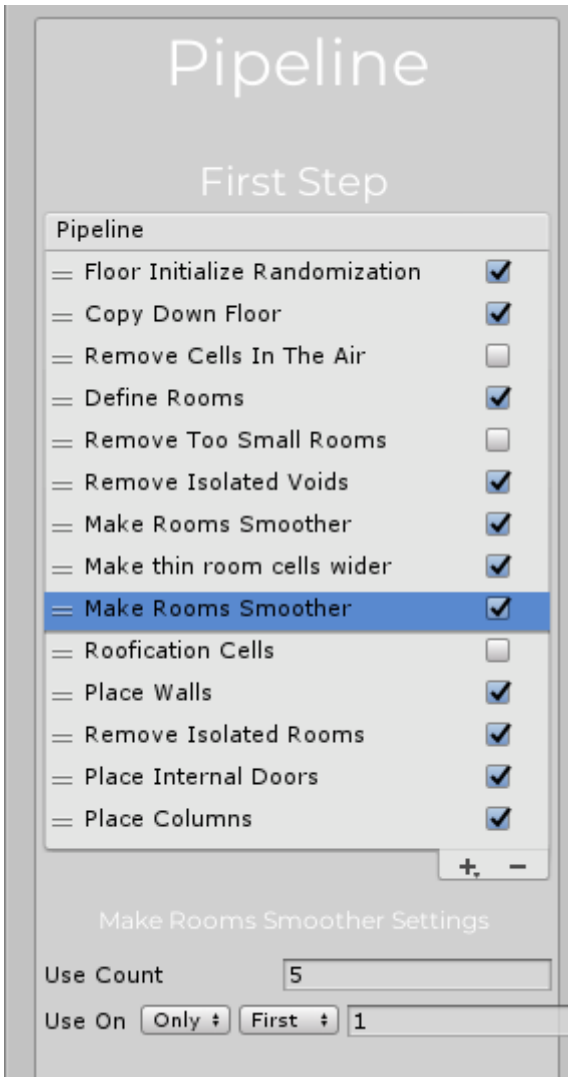
The house blueprint generates by two steps.

The first step usually uses for make a house geometry.

The secound step uses for pipes which should be used when the house geometry is done.

You can disable, enable and reorder them. If you want to make some experiments better to make a new pipeline (or dublicate one of existed) and drag and drop it to the Pipeline Preset field.

After click on the pipe you'll see a panel with additional settings:

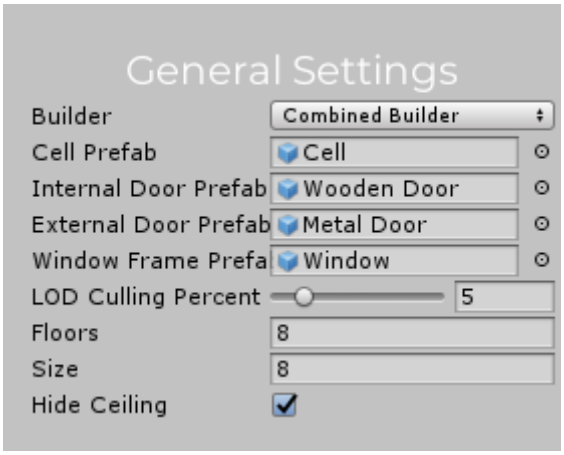


Use Count - how much this pipe will be used.

Use On - sets limits to use on floors.

> **Make rooms smoother** uses only on the first floor because all other floors will be just copied from the first.

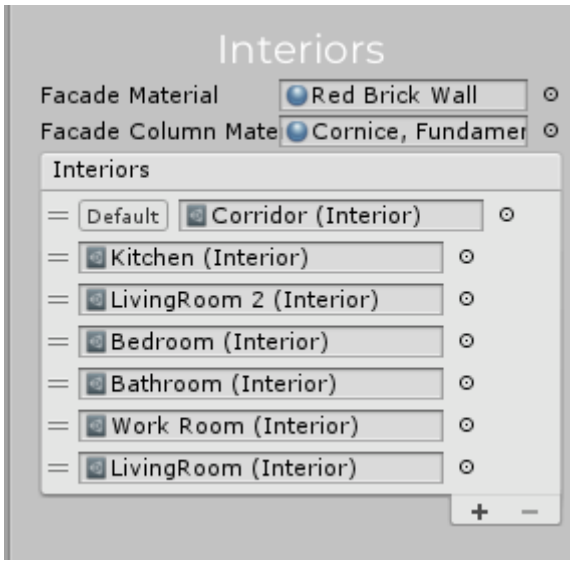
General Settings



Here are the basic settings for HouseGenerator:

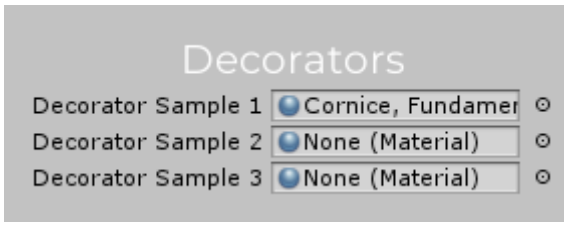
1. **Builder** - type of builder for consctruct of a house.
    - **Raw Builder** - makes a grid from cell prefabs. Not optimized but pretty good for debug.
    - **Combined Builders** - combines all cells into one mesh and tries to combine props and LODs. Very optimized, but you can't get an information from any cell.
2. **Internal Door Prefab** - a door between rooms. Necessary for the combined builder.
3. **External Door Prefab** - a facade door. Necessary for the combined builder.
4. **Window Frame Prefab** - just a window frame prefab. Necessary for the combined builder.
5. **LOD Culling Percent** - culling percent for all prefab LODs. It's like a distance to disable props renderers.
6. **Cell Prefab** - contains the prefab of the **cell** of the house.
7. **Floors** - a count of floors.
8. **Size** - maximum size of the house.
9. **Randomizer** - affects the **raw blueprint**. If you want to see more clearly how it works, turn off all Pipes and generate a couple of houses using different Randomizer values.
10. **Hide Ceiling** - indicates whether to hide the roof when generating the house.

Interiors



1. **Facade Material** - material for the facade.
2. **Facade Column Material** - material for columns that are on the outside of the house.
3. **Interiors** - list of interiors to be used to generate the house. The order of the interiors matters, if there are two interiors with the same requirements for the room, then the first one that came up will be used if it has not exhausted its use limit. The **first** interior in this list is a **default** interior and it'll be chosen if others do not fit.

Decorators

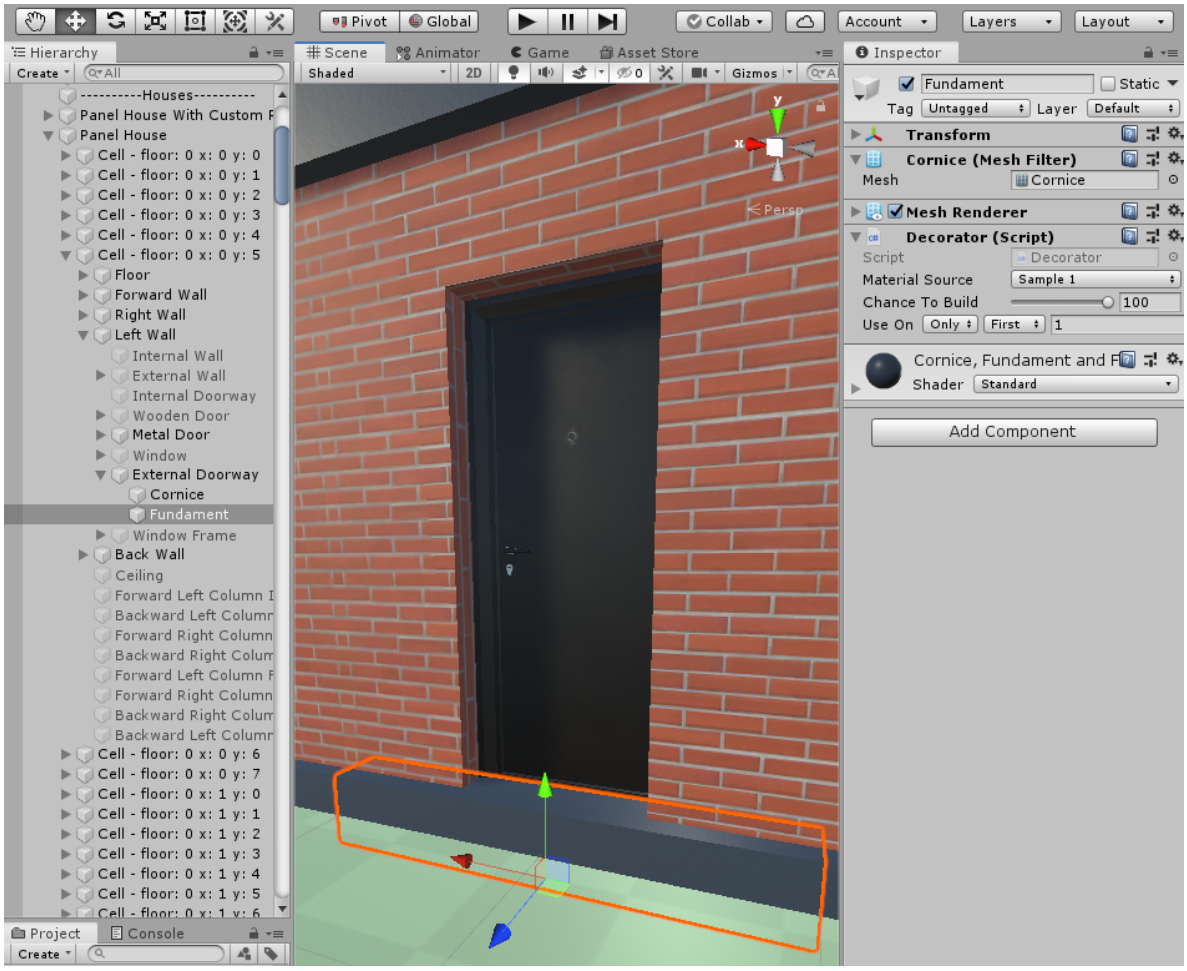


Decorator samples are materials which decorators can use instead of their own materials.

Decorator

Decorator is a some additional part of cell.

Use it for something like a cornice, porch etc.

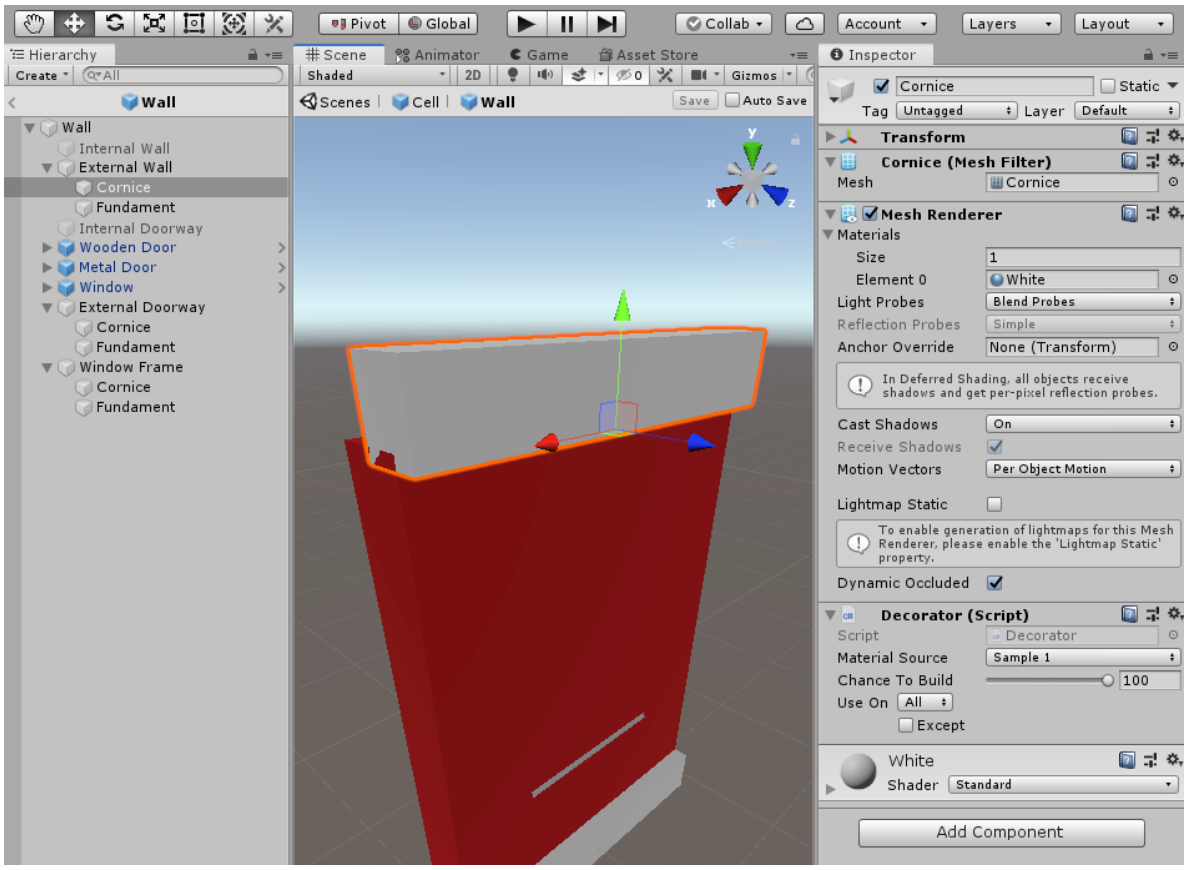


1. **Material Source** - which material you want to use for this decorator
    - **Sample** - use a material from the house generator sample.
    - **Own** - use its own material.
2. **Chance To Build** - probability of occurrence.
3. **Use On** - sets limits to use on floors.

> Fundament should be used only on the first floor because it's an... fundament.

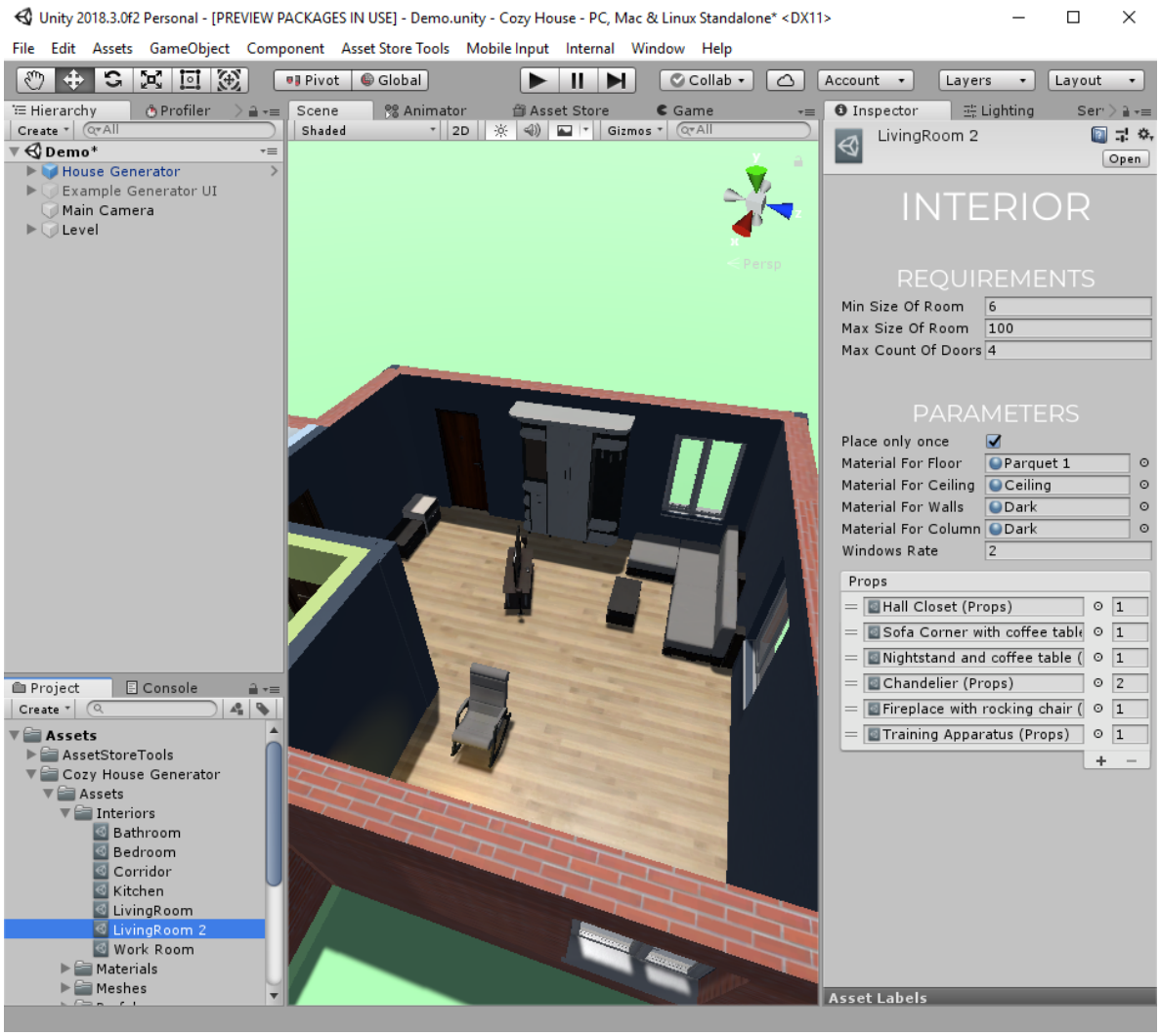
To add a new decorator open the cell prefab and add it under some part of cell.

For example, if you want to add a cornece, place it under all external walls, but if you want to add a porch, you should place it only under an external door.


Pivot
Global
Collab
Account
Layers
Layout
Hierarchy
Create
All
Wall
Wall
Internal Wall
External Wall
Cornice
Fundament
Internal Doorway
Wooden Door
Metal Door
Window
External Doorway
Cornice
Fundament
Window Frame
Cornice
Fundament
Scene
Animator
Game
Asset Store
Shaded
2D
Gizmos
Scenes
Cell
Wall
Save
Auto Save
Persp
Inspector
Cornice
Static
Tag
Untagged
Layer
Default
Transform
Cornice (Mesh Filter)
Mesh
Cornice
Mesh Renderer
Materials
Size
1
Element 0
White
Light Probes
Blend Probes
Reflection Probes
Simple
Anchor Override
None (Transform)
In Deferred Shading, all objects receive shadows and get per-pixel reflection probes.
Cast Shadows
On
Receive Shadows
Motion Vectors
Per Object Motion
Lightmap Static
To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.
Dynamic Occluded
Decorator (Script)
Script
Decorator
Material Source
Sample 1
Chance To Build
100
Use On
All
Except
White
Shader
Standard
Add Component

Interior

Interior is a scriptable object that defines the appearance of rooms.



To create a new interior, right-click on any folder => **Create** => **Cozy House Generator** => **Interior**

To use the interior, add it to **the list of interiors in the House Generator**.

The following panels are used to configure the interior:

Requirements

Here are the requirements for the room. If the room meets the requirements, this interior will be applied to it.

Min Size Of Room: if the number of cells in the room is less than this number, this interior wont be applied to the room.

Max Size Of Room: if the number of cells in the room is greater than this number, this interior wont be applied to the room.

Max Count Of Doors: if the doors in the room are larger than this number, then this interior wont be applied to the room. This field is very useful for interiors of small rooms such as bathroom.

Parameters

It contains a description of the interior.

Place only once: if true, this interior will be used only once for the whole house. ***WIP***

Material For Floor: this material will be applied to the floor in the room.

Material For Walls: this material will be applied to the walls.

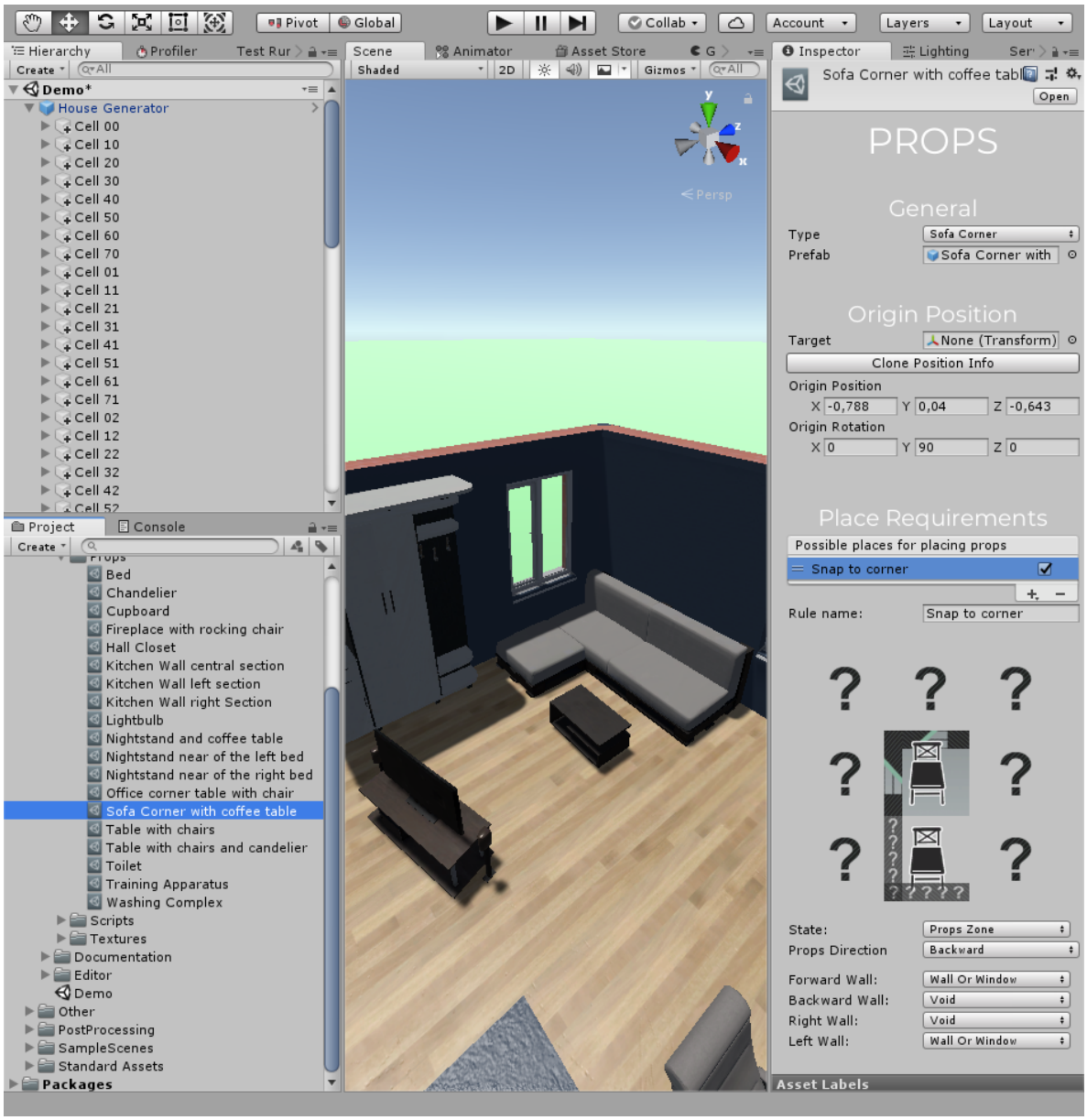
Material For Columns: this material will be applied to the columns.

Windows Rate: the gap between the windows in the house. 0 - no windows, 1 - the whole wall is covered with windows, 2 - the gap between the windows is one cell and so on.

Props: A list of props that may be in this interior. The maximum number of copies is indicated to the right of the prop.

Props

This scriptable object provides the creation of furniture objects during the generation of the house.
It contains information about the type of **props**, a link to the prefab, data on the position inside the **cell**, and a list of space requirements around the **cell**.



To create a new **props**, right-click on any folder => **Create** => **Cozy House Generator** => **Props**

To use **props**, add this **props** to the **list of props in the interior**, for which is are best suited. For example, the toilet is only suitable for the bathroom, but the table can be added to the interior of the kitchen or living room.

The following panels are used to configure the props:

General

Type: type of props. Used to help props find nearby props of the type they need. If your props does not match any existing type, you can add your own in **PropsType.cs**, which is located in the directory ...Assets\Generator\Scripts\Core\DataTypes\Enums

```
namespace Generator.Scripts.Core.DataTypes.Enums
{
//////////////////////////////////////////////////////////////////////////////////
///////
    /// <summary>  The type of props  </summary>
//////////////////////////////////////////////////////////////////////////////////
////
    [Serializable]
    public enum PropsType
    {
        None , // Custom name for "Nothing" option
        Chair          = 1 << 0 ,
        Sofa           = 1 << 1,
        Table          = 1 << 2,
        Toilet         = 1 << 3,
        Fridge         = 1 << 4,
        KitchenWall    = 1 << 5,
        Chandelier     = 1 << 6,
        Stove          = 1 << 7,
        Bed            = 1 << 8,
        Cupboard       = 1 << 9,
        Nightstand     = 1 << 10,
        SofaCorner     = 1 << 11,
        MedicineChest  = 1 << 12,
        ShowerBath     = 1 << 13,
        WashBasin      = 1 << 14
    }
}
```

Prefab: link to props prefab.

Origin Position

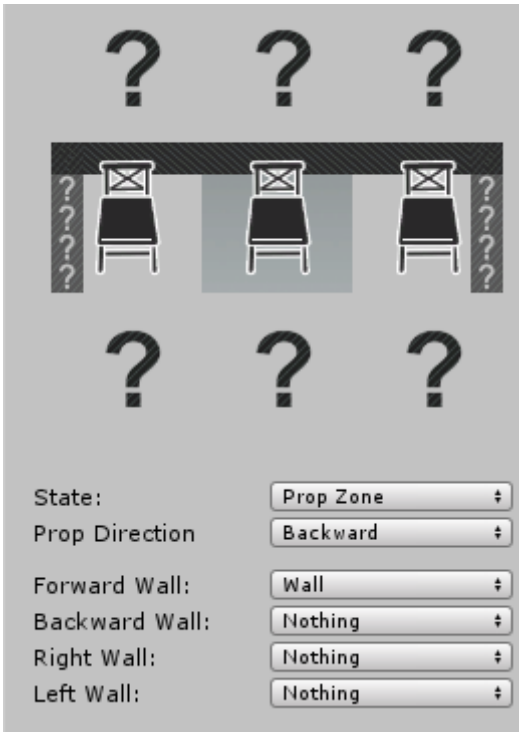
During generation, props are instantiated inside **Cell** => **PropsContainer** with a **local position** and **local rotation** from this panel. Next, the props turns in the direction that is recorded in the **Prop Direction** field in the **Place Requirements** panel. If the prop is turned in the wrong direction, try changing the rotation values in this panel.

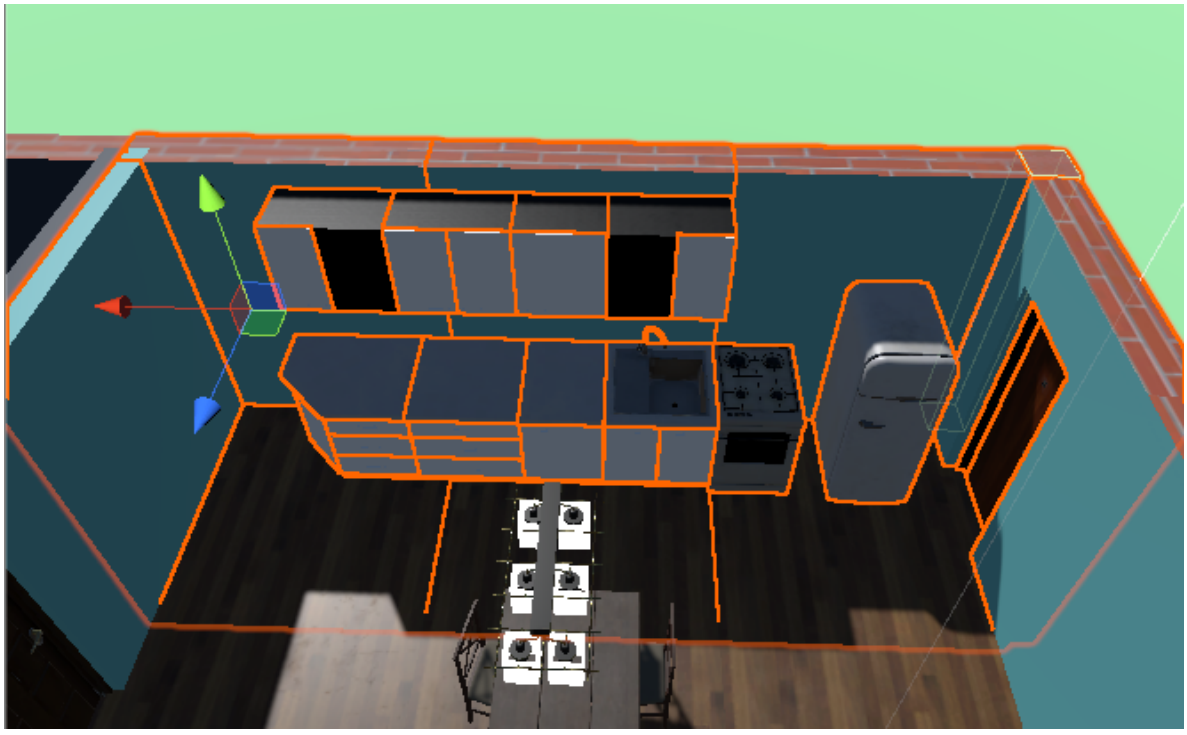
For the convenience of setting the rotation, there is also the **Target** field and the **Clone Position Info** button. It is assumed that you place a copy of the props in this field, then place it in the Props Container inside the cell, and you will rotate and move the props until you get a satisfactory result, then record the result by clicking on the **Clone Position Info** button.

Place Requirements

This panel contains information about what should be the space around the props.

Possible places for placing props: This list contains descriptions of how the space around the props can be. If the space around the props corresponds to one of the descriptions, the props prefab will be instantiated. To add a new description, click + in the lower right corner. You can also copy an existing description. To customize the description, simply click on it, and the settings panel will appear from below.





In fact, this panel schematically shows how the space around the props can be.
In the image above, you can see how a schematic description of the space and its actual looks.

To customize the description of the space around the props, simply click on the cell and use the cell settings panel shown below.

This panel consists of the following elements:

- **State:** cell state. Possible states:

- **Any:** means that a given cell can have any state and will not be checked when generating.
- **Floor:** means that this cell belongs to any room and can have any props or not have it at all.
- **Other Props:** means that the cell must have one of the specified props.

> When you select the status of **Other Props**, you will be available to the setting of **Possible Props**, in which you can specify which types of props can be in this place.

- **Prop Zone:** this cell is overlapped by the current pops.

  > When you select the **Prop Zone** state, or if you select the center cell, you will have the **Prop Direction** settings available, which indicates in which direction the props prefab should be rotated after the instantiation.

- **Outside Of The Room:** means that this cell is outside the room. It can be a part of another room, or it can be outside the house.

When selecting any state other than **Any** or **Outside Of The Room**, the wall settings panel opens.

This panel contains a list of walls that can be around the cell.

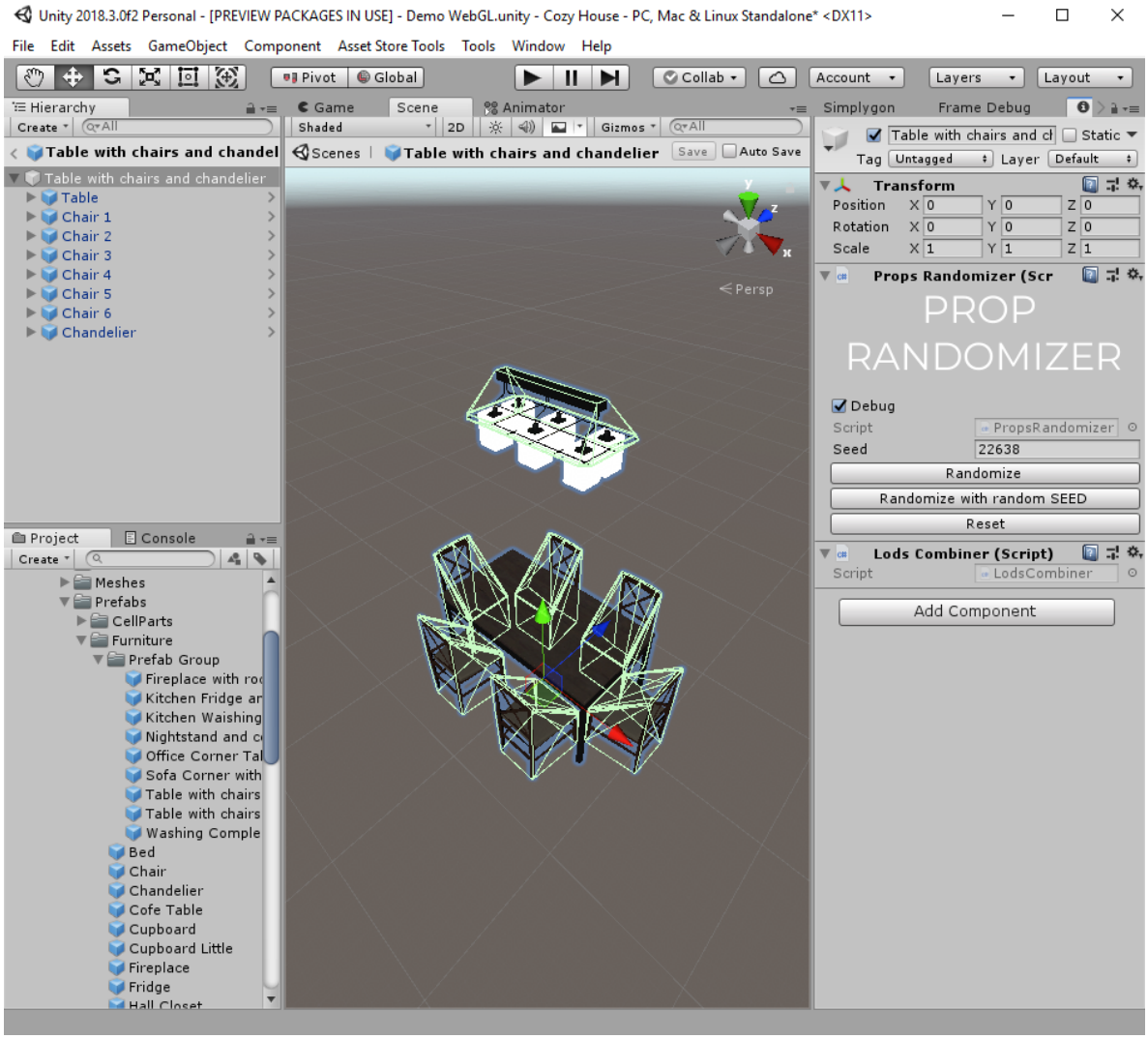
Wall types:

- **Nothing:** means that there should be no wall in this place.
- **Any:** means that this wall can have any state, including **Nothing**, therefore it is not checked when generating.
- **Wall:** this is an ordinary wall.
- **Door:** this is a doorway.
- **Window:** this is a window.

> Note that you can combine several types of walls!

Props Randomizer

Props Randomizer allows you to make your props more random with each generation.



To use the **Props Randomizer**, add the **Props Randomizer component** to the prefab which contains child objects which should be randomized, and then add the **Sub Props component** to these child objects. How to configure the **Sub Props component** will be described below.

A good example is a table with chairs. Add the **Props Randomizer component** to the table. Make chairs children of the table. Add **Sub Props component** to the chairs and customize to your taste. Everything, now the table is ready for randomization.

Props Randomizer is automatically used when instantiating a props prefab.

Debug

The **Debug** panel allows you to check how well **Sub Props** has been configured without the need to generate a house.

Seed - the seed number.

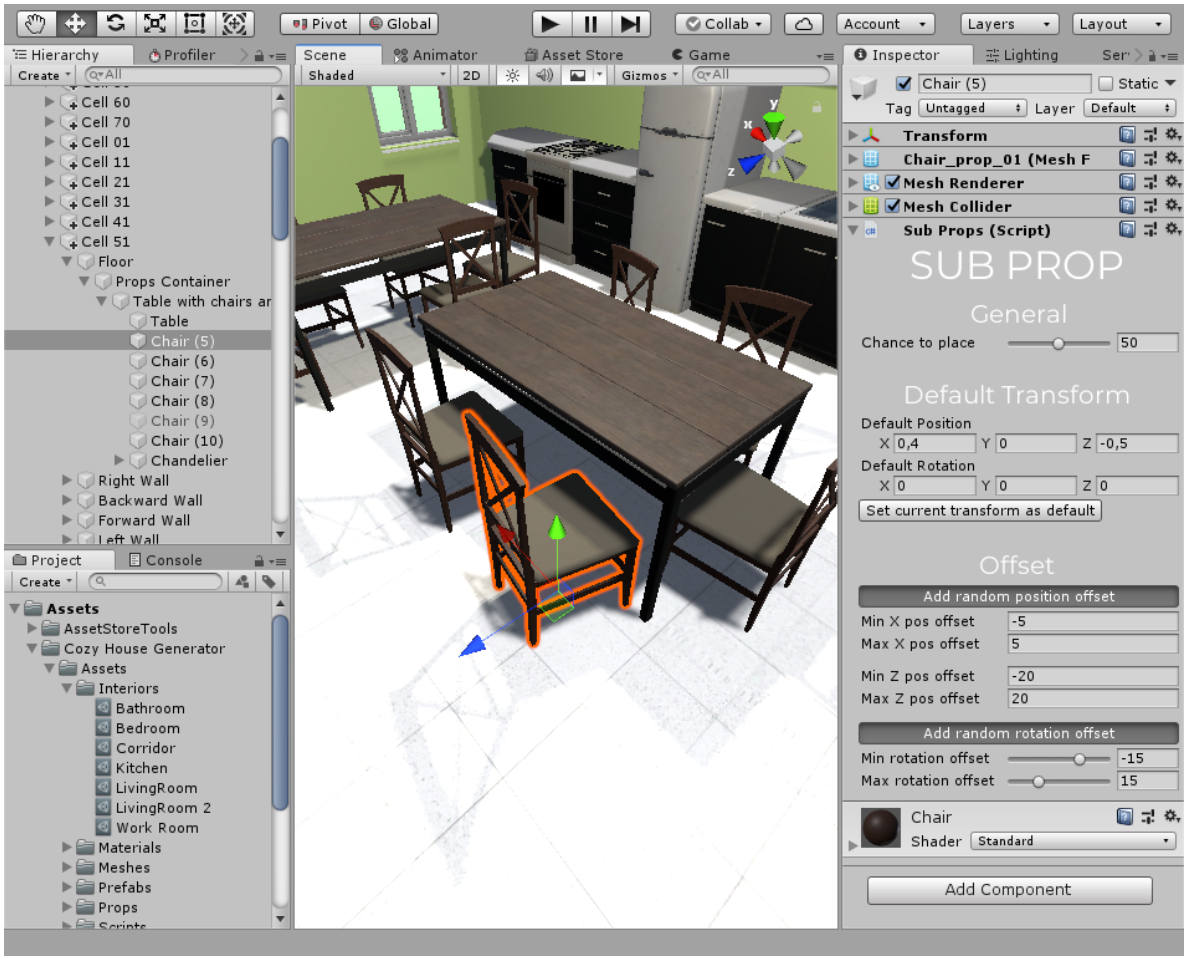
Randomize - this button starts seed-based generation.

Randomize with random SEED - This button generates a random seed number, writes it to the seed field and starts seed-based generation.

> Note that if **Sub Props** aren't configured, the chairs won't move.

Sub Props

Sub Props component allows you to rotate, move or turn off the object at random.



To use **Sub Props**, add the **Sub Props component** to the prefab's child object that you want to randomize.

Sub Props is automatically used when instantiating a props prefab.

The following panels are used to configure **Sub Props**:

General

Chance to place - the percentage that determines the probability of the object being turned on.

Default Transform

This panel setups the starting position and the initial rotation of the object. The displacement of the object occurs relative to this position.

Set current transform as default button - writes the current position and the current rotation of the object in the appropriate fields.

> Note that if the **Add random position offset switch** is not turned on, the **Default Position** will not be used, and if **Add random rotation offset** is also not turned on, **Default Rotation** will not be used either.

Offset

This setting allows you to specify the limits of an object's random displacement.

Add random position offset: adds a random offset along the X and Y axes. When generating, the offset value is chosen randomly between the minimum and maximum values.

- **Min X pos offset:** minimum offset along the X axis.
- **Max X pos offset:** maximum offset along the X axis.
- **Min Y pos offset:** minimum offset along the Y axis.
- **Max Y pos offset:** maximum offset along the Y axis

> Position offset is written in centimeters.

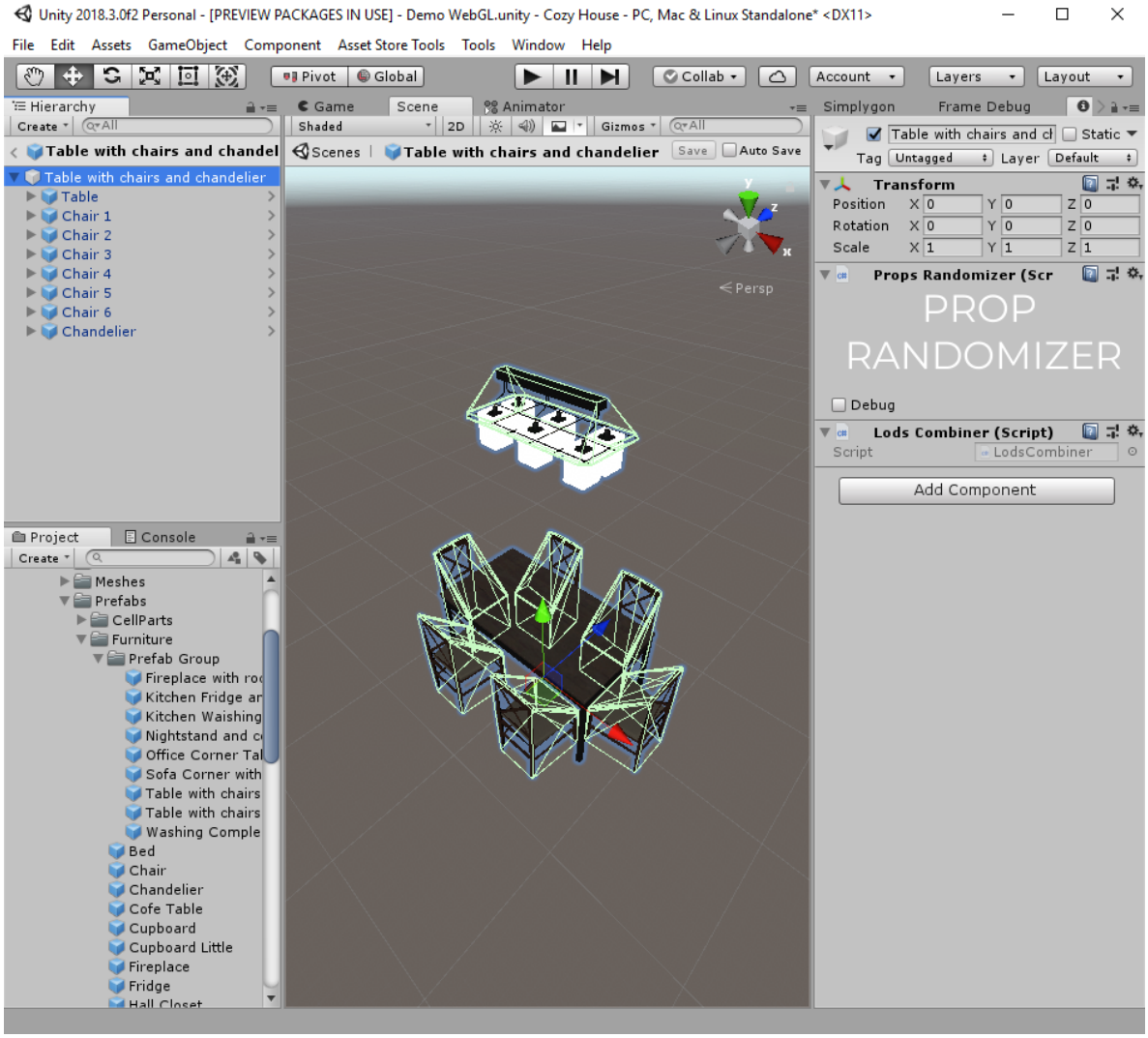
Add random rotation offset: adds a random rotation. When generating, the amount of rotation is chosen randomly between the minimum and maximum values.

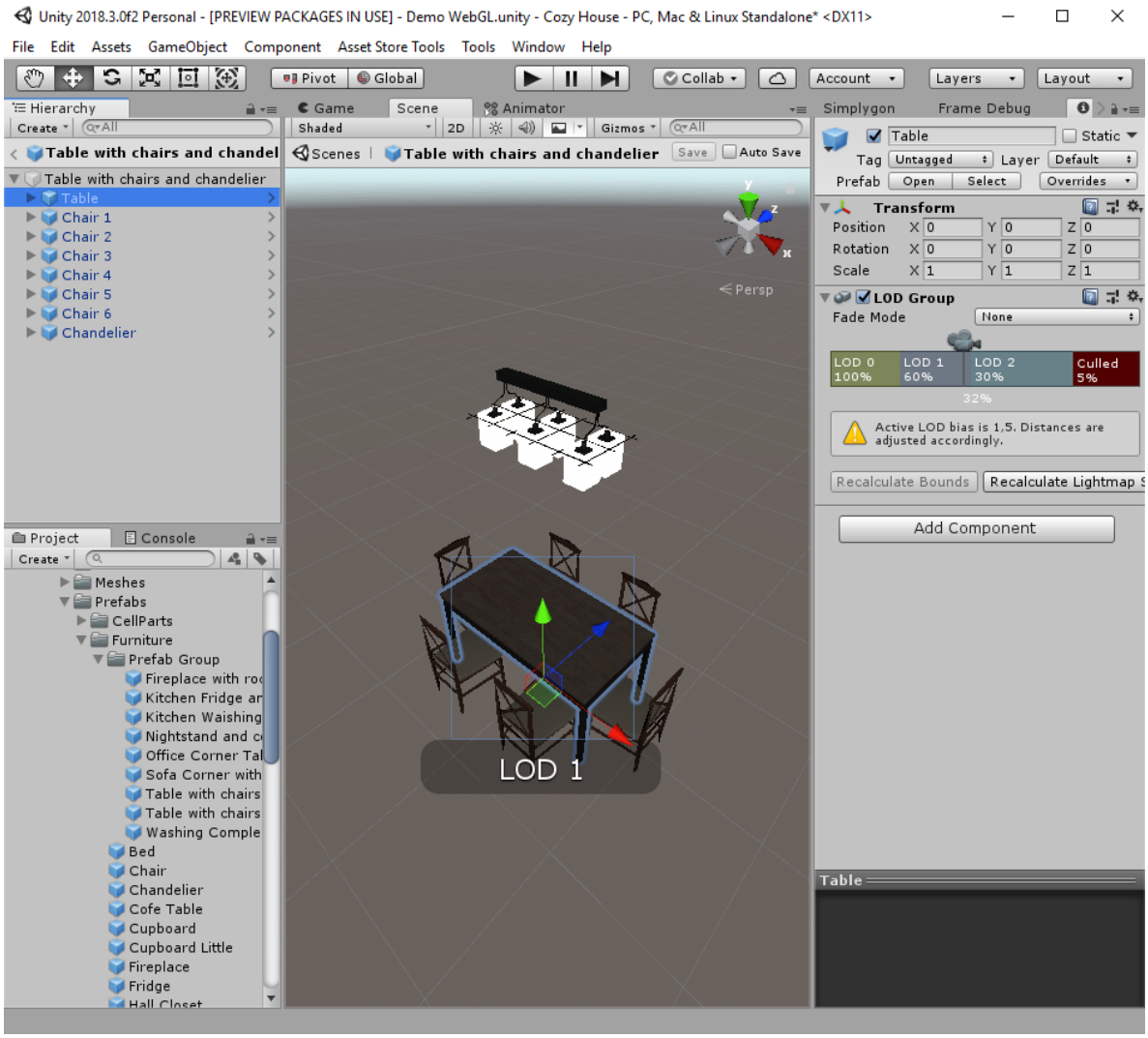
- **Min rotation offset:** minimum amount of rotation.
- **Max rotation offset:** maximum amount of rotation.

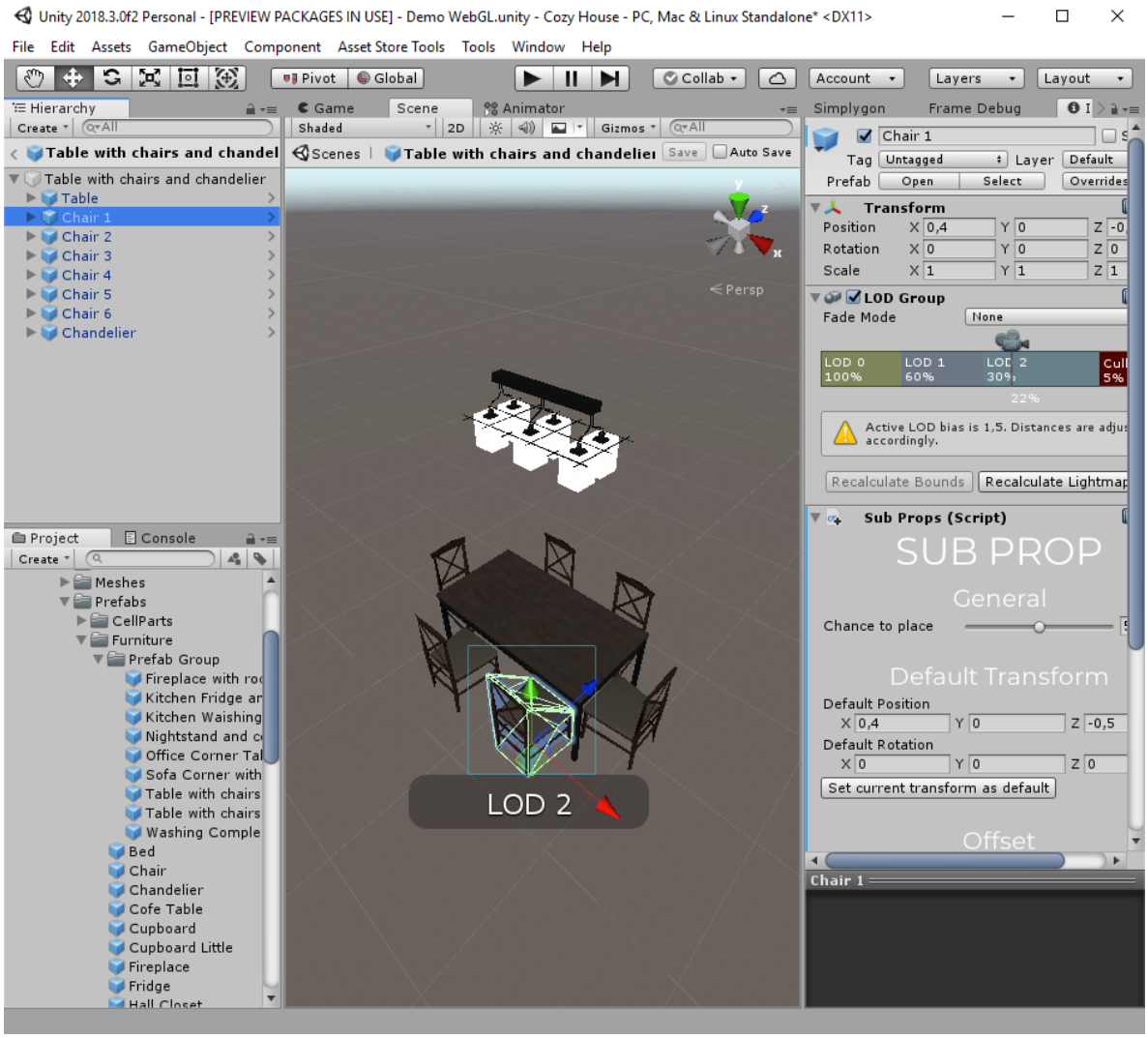
> The rotation is written in degrees.

Lods Combiner

If you have several configurated LOD groups inside props prefab, you can add **Lods Combiner** component on root gameobject and they will be combined by **Combined Builder**.

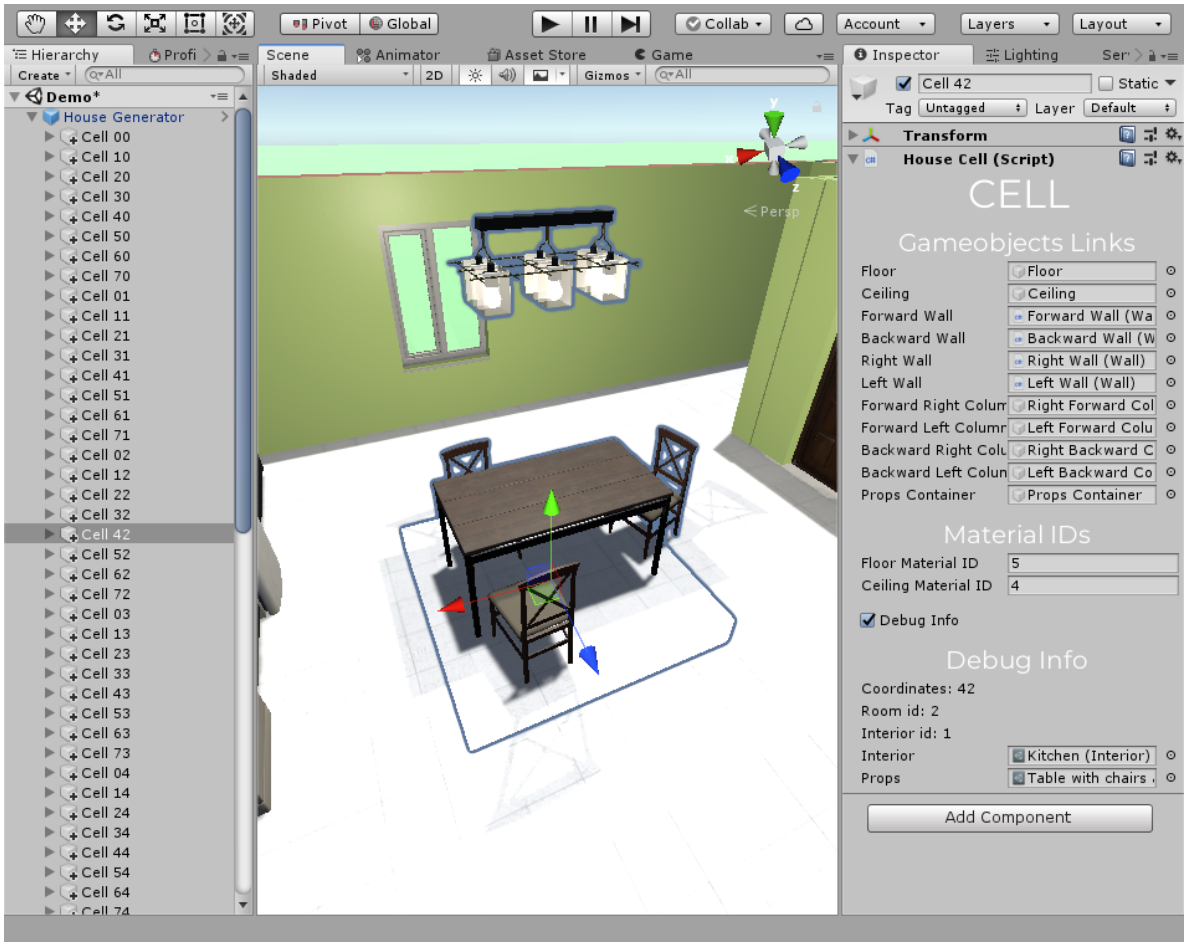



Unity 2018.3.0f2 Personal - [PREVIEW PACKAGES IN USE] - Demo WebGL.unity - Cozy House - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Asset Store Tools Tools Window Help
Pivot Global
Collab Account Layers Layout
Hierarchy
Create All
Table with chairs and chandel
Table with chairs and chandelier
Table
Chair 1
Chair 2
Chair 3
Chair 4
Chair 5
Chair 6
Chandelier
Game Scene Animator
Shaded 2D Gizmos All
Scenes | Table with chairs and chandelier Save Auto Save
Persp
LOD 1
Simplygon Frame Debug
Table Static
Tag Untagged Layer Default
Prefab Open Select Overrides
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
LOD Group
Fade Mode None
LOD 0 100%
LOD 1 60%
LOD 2 30%
Culled 5%
32%
Active LOD bias is 1,5. Distances are adjusted accordingly.
Recalculate Bounds
Recalculate Lightmap
Add Component
Project Console
Create
Meshes
Prefabs
CellParts
Furniture
Prefab Group
Fireplace with roo
Kitchen Fridge ar
Kitchen Waishing
Nightstand and c
Office Corner Tal
Sofa Corner with
Table with chairs
Table with chairs
Washing Comple
Bed
Chair
Chandelier
Cofe Table
Cupboard
Cupboard Little
Fireplace
Fridge
Hall Closet
Table

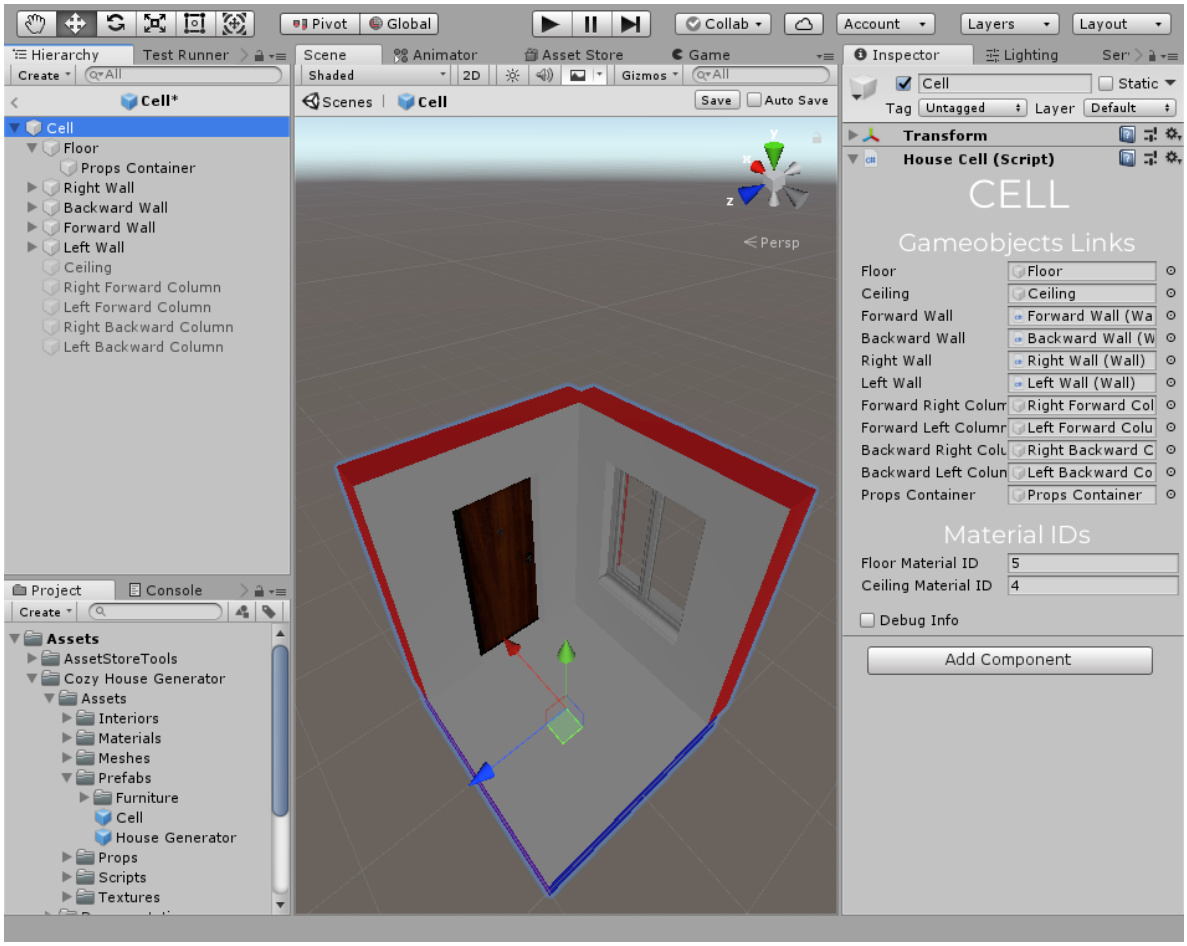

Unity 2018.3.0f2 Personal - [PREVIEW PACKAGES IN USE] - Demo WebGL.unity - Cozy House - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Asset Store Tools Tools Window Help
Pivot Global
Collab
Account
Layers
Layout
Hierarchy
Create
Table with chairs and chandel
Table with chairs and chandelier
Table
Chair 1
Chair 2
Chair 3
Chair 4
Chair 5
Chair 6
Chandelier
Game
Scene
Animator
Shaded
2D
Gizmos
Scenes
Table with chairs and chandelier
Save
Auto Save
Persp
LOD 2
Simplygon
Frame Debug
Chair 1
Tag
Untagged
Layer
Default
Prefab
Open
Select
Overrides
Transform
Position
Rotation
Scale
LOD Group
Fade Mode
None
LOD 0
100%
LOD 1
60%
22%
Active LOD bias is 1,5. Distances are adjus
accordingly.
Recalculate Bounds
Recalculate Lightmap
Sub Props (Script)
SUB PROP
General
Chance to place
Default Transform
Default Position
Default Rotation
Set current transform as default
Offset
Chair 1
Project
Console
Create
Meshes
Prefabs
CellParts
Furniture
Prefab Group
Fireplace with roo
Kitchen Fridge ar
Kitchen Washing
Nightstand and c
Office Corner Tal
Sofa Corner with
Table with chairs
Table with chairs
Washing Comple
Bed
Chair
Chandelier
Cofe Table
Cupboard
Cupboard Little
Fireplace
Fridge
Hall Closet

Cell

Cell is the basic component of the house, which is responsible for the construction of walls, floor, ceiling and the instantiation of furniture.



To change the walls, floor, ceiling or columns, simply replace the child objects of the Cell prefab and set the links in the **Gameobject Links** panel.

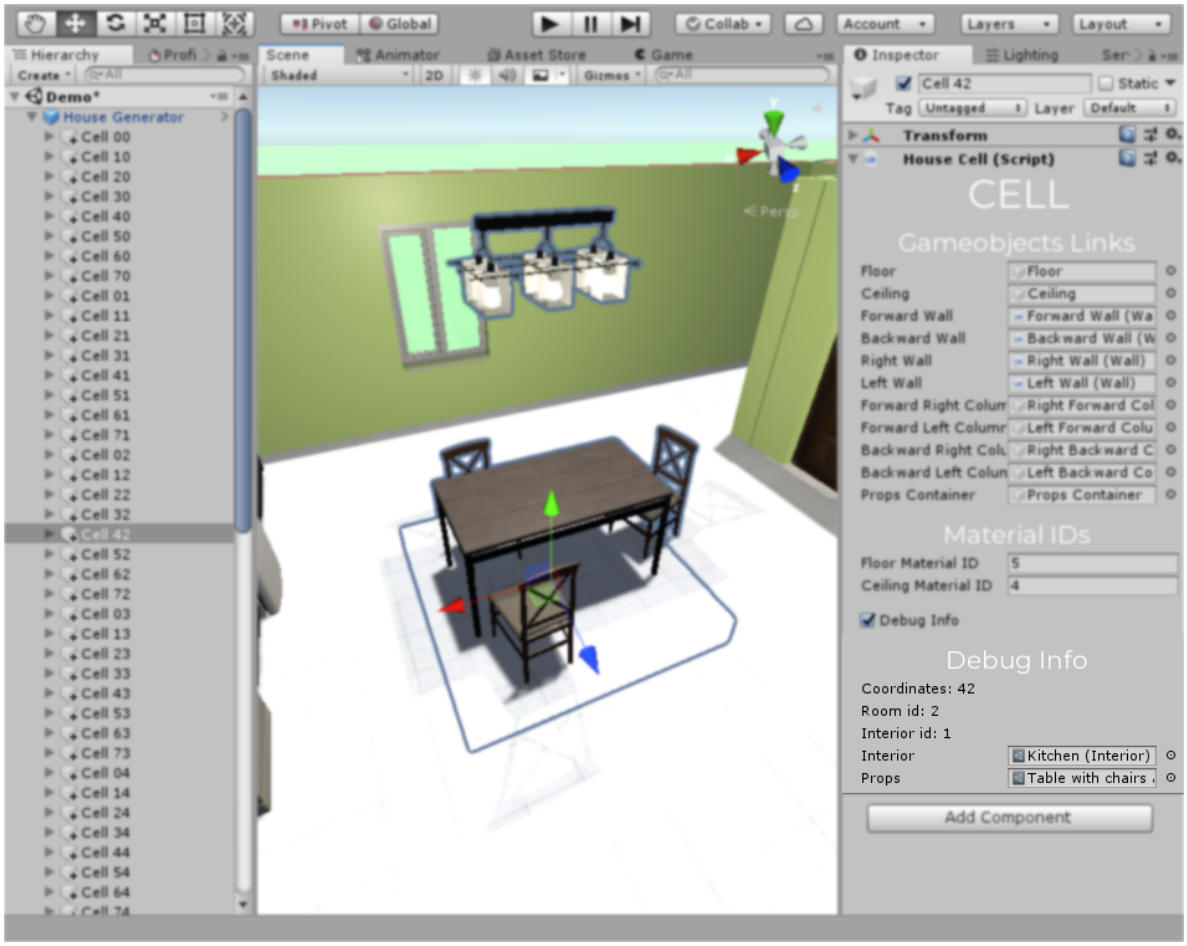


> Walls must have a **wall** component.
>
> How to configure the **wall** component will be described below.

Material ID is used to set materials for walls, floor, ceiling, etc.

Material ID is the id of the material that will be overwritten during generation. Information about the new material is taken from the **interior** assigned to this **cell**.

For the plugin to work correctly, the new meshes must have a separate material for each **Material ID**.

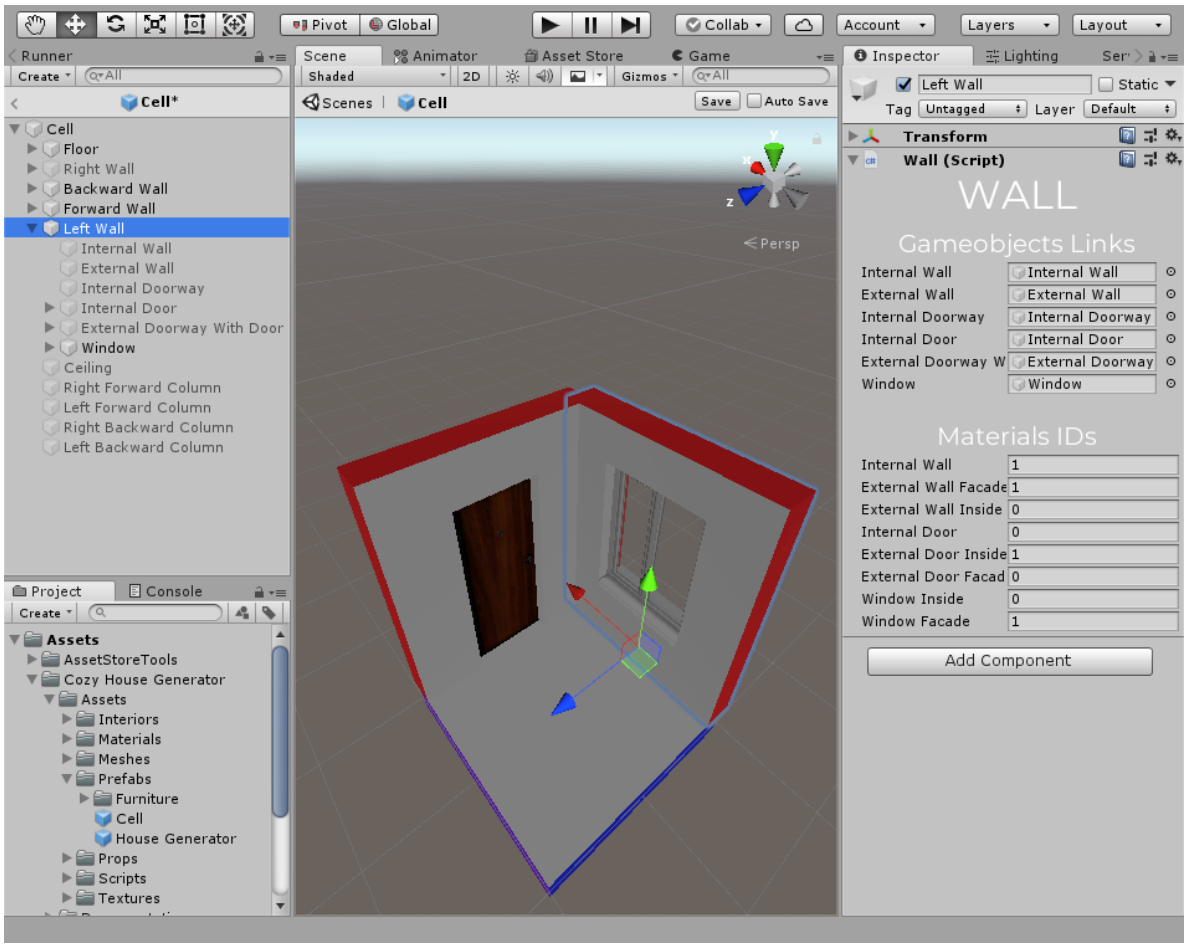


Debug Info provides information for debugging.

- **Coordinates:** coordinates of the generated cell along the axes I and J.
- **Room id:** id of the room the cell belongs to.
- **Interior id:** id of the assigned interior room.
- **Interior:** reference to the interior.
- **Props:** reference to the that was generated on the cell.

Wall

The **wall** component provides wall construction.



Setting up a wall is very similar to cell, you have to set up links to different wall variations (a simple wall, a wall with a doorway, etc.), and specify an ID for materials.

To change wall meshes, replace the **Wall** child objects and set the links in the **Gameobject Links** panel.

> Note that the **wall component** should only be on the parent object.