Impossible Spaces Manual

This is a guide on how to create and convert an environment into an impossible space. For the best results, keep the idea of impossible spaces in mind as you are designing your environments. Additionally, the immersion provided by impossible spaces works better with larger play areas. As the focus is on adding the impossible space aspect, this manual will work assuming that a layout has already been created.

Setup

Before we start, make sure that the project is in URP. This implementation relies on URP's Renderer Features.

Rooms

First, create an empty GameObject for each room. Under these GameObjects, you can add all the necessary objects for each room like the walls, items, obstacles, etc. Feel free to add any scripts or functionality that may be needed. It may help later on to number every object.

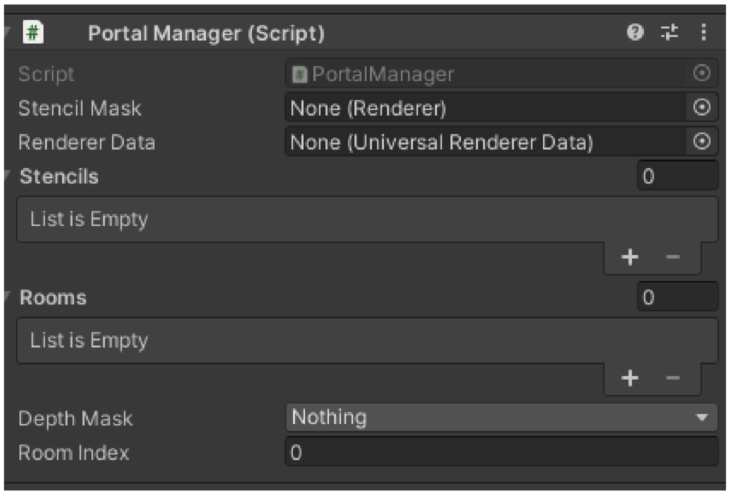
Player

Add a quad GameObject in front of the main camera as a child object. This will act as our main stencil mask. This will be covered more in-depth in the Camera section.
Also, make sure the player object has some sort of collider on it. Then assign the "Player" tag to the object.

Portal Manager

The key to impossible spaces is the use of Euclidean manifolds. To manage all the portals and triggers necessary to create the impossible space, we will use the Portal Manager script. Create an empty GameObject and add the Portal Manager script to it.

The Portal Manager should look like the image below. We will now go over each exposed field and their usages.


Portal Manager (Script)

Script	PortalManager
Stencil Mask	None (Renderer)
Renderer Data	None (Universal Renderer Data)
Stencils	0
	List is Empty
Rooms	0
	List is Empty
Depth Mask	Nothing
Room Index	0

Stencil Mask

Assign the stencil mask we created in front of the camera in the Setup step. The Portal Manager will automatically handle material switching for the mask.

Renderer Data

Drag the Universal Renderer Data settings asset you are using for this project. By default, the High Fidelity asset is used. This is mainly so we can augment the depth pass to only account for the current stencil value.

Stencils

For now, you can leave this blank. This field will be covered more in-depth in a later section.

Rooms

Drag each room GameObject created in the Setup step and add it to this field. The result should be a list of all the rooms. Note: Order doesn't matter but it helps to keep some sort of logical order to the array so that it is easier to set up the rest of the scene.

Depth Mask

Do not change this from “Nothing”. This is an exposed field with the sole purpose of debugging, though changing this value should not matter, it is best to keep it at the default: “Nothing”.

Room Index

Assign the starting room’s index to this field. This value is a 1-indexed number corresponding to the position of the starting room in the “Stencils” array mentioned above. Important: This value is 1-indexed, meaning that the first index is 1, not 0.

Stencils (Skippable)

This section will go in-depth about everything needed to properly set up stencil masks with URP. If you are using the default project, this should already be done and you can skip this step.

Preparation

Before beginning, we must determine the number of different stencils needed for this environment. To find this number, simply take the maximum number of doorways any single room contains and add 1.

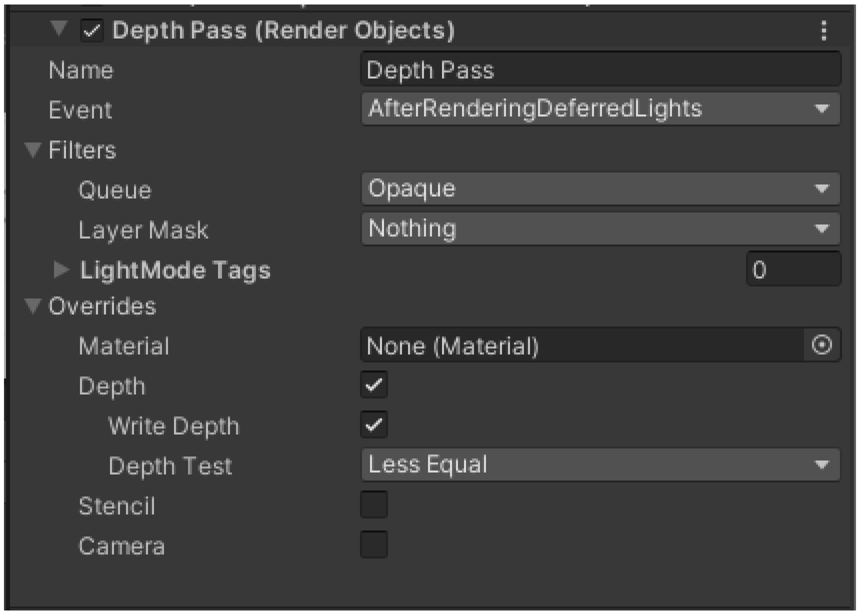
Layers

Now create N layers, where N is the number we calculated in the previous step. For simplicity, each layer’s name should be “Stencil [i]”, for the ith layer created in this manner. Again, start i=1 to keep things easier.

Renderer Features

Navigate to the URP settings asset. Under the “Renderer Features” area, click “Add Renderer Feature” and select the Render Objects option.

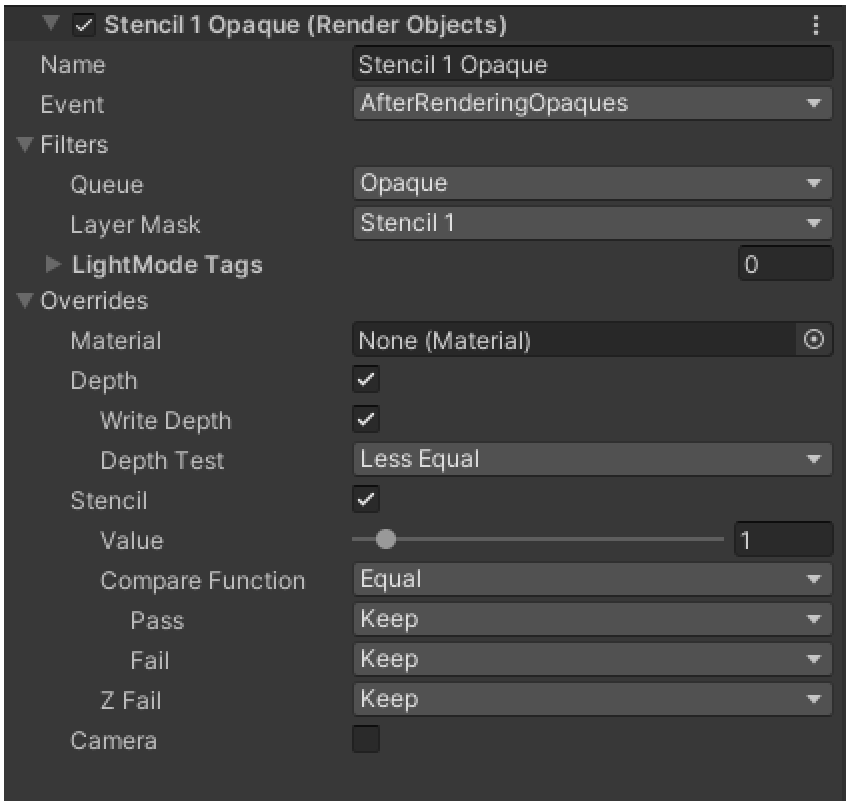
Name this first Render Object “Depth Pass” and configure the settings like so:



Depth Pass (Render Objects)

Name	Depth Pass
Event	AfterRenderingDeferredLights
Filters	
Queue	Opaque
Layer Mask	Nothing
LightMode Tags	0
Overrides	
Material	None (Material)
Depth	✓
Write Depth	✓
Depth Test	Less Equal
Stencil	
Camera	

Then, for each stencil layer, we created in the Layers step above, create another Render Object with the following settings:



Stencil 1 Opaque (Render Objects)

Name	Stencil 1 Opaque
Event	AfterRenderingOpaques
Filters	
Queue	Opaque
Layer Mask	Stencil 1
LightMode Tags	0
Overrides	
Material	None (Material)
Depth	✓
Write Depth	✓
Depth Test	Less Equal
Stencil	✓
Value	1
Compare Function	Equal
Pass	Keep
Fail	Keep
Z Fail	Keep
Camera	

Note: The main differentiators between each Render Object created this way are the "Layer Mask" field and "Stencil Value" fields, which should be assigned to the corresponding layer and stencil value. For example, the "Stencil 2 Opaque" Render Object should have the "Stencil 2" layer selected for the layer mask and a stencil value of 2.

Stencil Materials

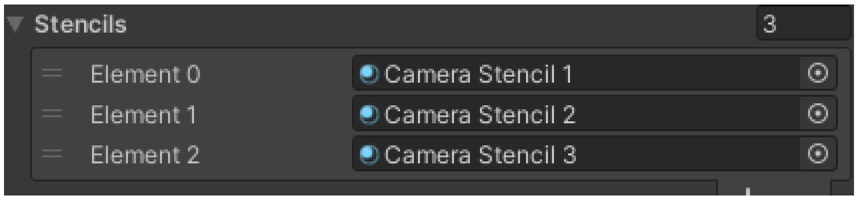
There are 2 types/copies of materials that we must use for the objects of our scene. One is for the camera, the other is for the stencil masks in doorways.

Creation (Skippable)

If no materials have been created, we must create the aforementioned materials ourselves. Simply create N number of materials (again N is calculated from the above Stencils section with the "Custom/CameraStencil" shader. Each of these materials should have a different stencil value associated with it (from 1 to N). Repeat this process but this time use the "Custom/StencilMask" shader instead. Again, make sure the stencil values are marked correctly.

Portal Manager Stencils

Now that we have the stencil materials set up, we can now address the "Stencils" field. Assign, in order, the camera stencil materials. Make sure these materials are using the CameraStencil shader as we rely on these materials being earlier in the render queue. The following is an example with N=3 stencils:



Stencils	3
Element 0	Camera Stencil 1
Element 1	Camera Stencil 2
Element 2	Camera Stencil 3

Scene Configuration

With all the setup done, we can now look to configure the scene to work with the impossible spaces. Assuming the scene was set up correctly, this should be a fairly simple process.

Room Assignment

We can now assign each room its stencil value. We do this by using the layers that we've created. Taking a room GameObject in the scene, we can assign one of the layers we made to that GameObject. When given the option, change the layer of its children as well. The room and all its objects should not be rendered – this is expected behavior. Repeat this process for all the room objects in the scene.

While not explicitly required, it is very highly recommended to ensure that adjacent rooms have different stencil values from each other and the current room. One easy method for this if the general path is linear is to assign the layers sequentially for every room.

For example, given 3 rooms A, B, and C, if the path is: A → B → C, then A would be layer Stencil 1, B would be Stencil 2, and so on. Upon reaching Stencil [N-1], just start again from Stencil 1 and repeat.

Camera

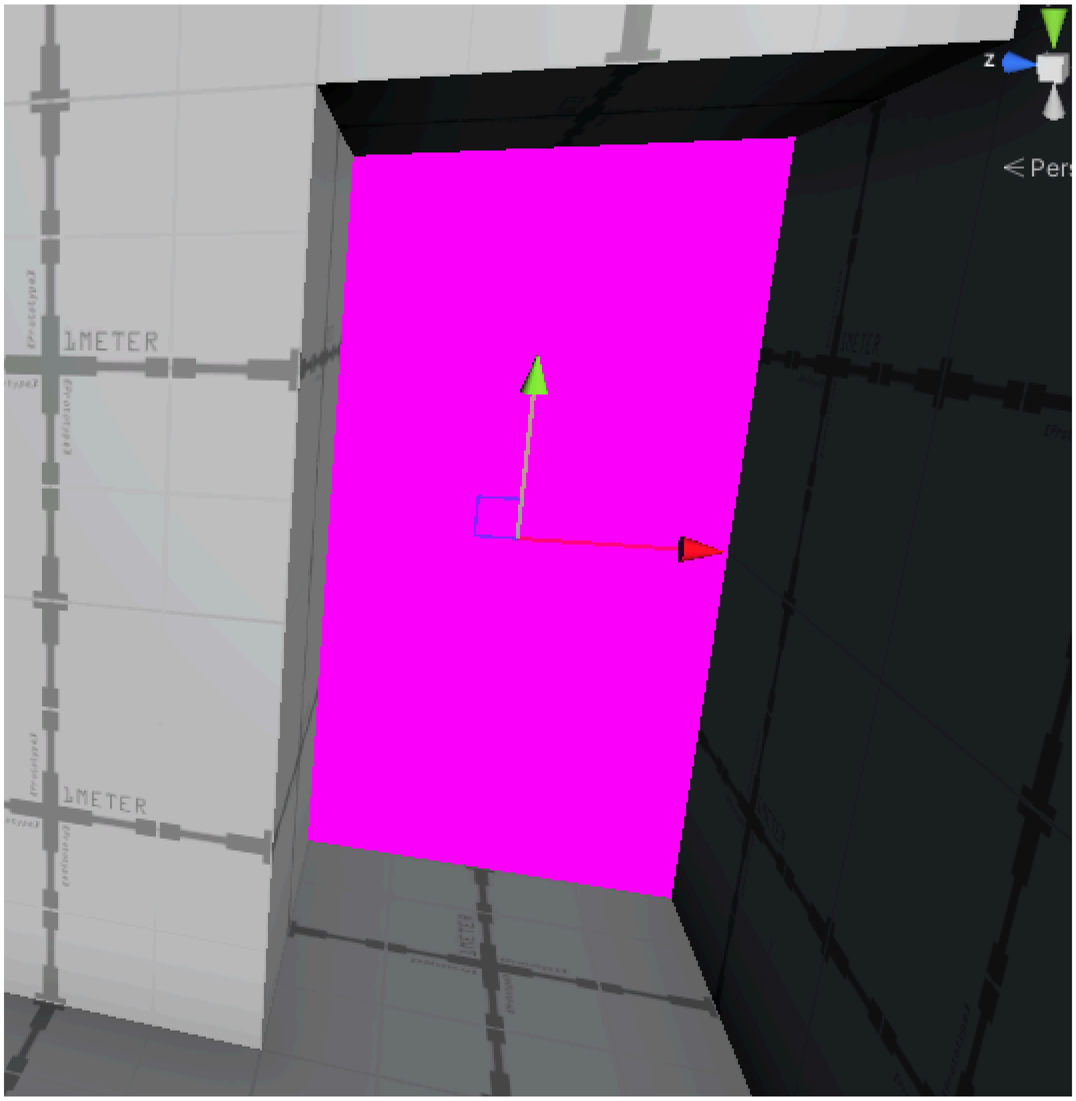
We need to adjust a few things for the camera. First, make sure the camera's near plane is something small like 0.01. This ensures that the camera will not cull stencil masks when we get close to them. Additionally, make sure that the quad object we created that acts as the main stencil mask covers the entire screen. Generally, a 2x2x2 quad with position (0,0,0.3) tends to work well. An optional step is to set the material of this object to be the StencilMask material associated with the starting room.

Doorways

There are a few more things needed for doorways specifically. This is because while we can easily use our main stencil mask for rendering the current room, there is no way to see other rooms with different stencil values. We are also missing a way to transition from one room to another. For the following two subsections, repeat the processes described for each doorway in the environment.

Stencil Masks

To see through doorways into other rooms, we must set up stencil masks in the doorways with stencil values different from the main mask. To do this, we can create a quad whose size matches the shape of the doorway as closely as possible. Make sure this quad is a child of the current room. Then, move and orient the quad so that it lies on the outer side of the doorway and faces toward the interior of the room. Refer to the following as an example:



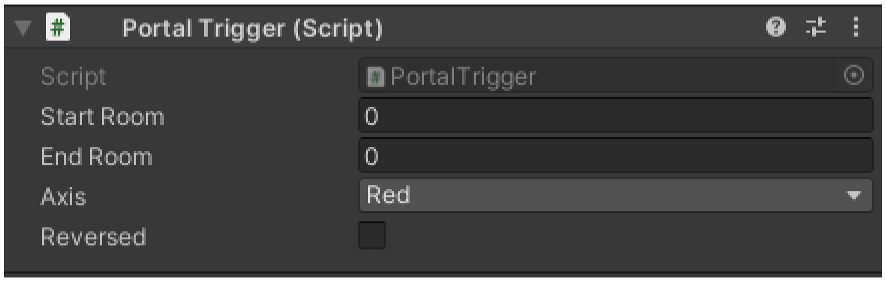
We can then change the material of this quad to be the material with the StencilMask shader whose stencil value is equal to [k], the stencil number of the layer associated with the room that this doorway leads to (not the current room).

For example, if I am in a room with layer Stencil 1, and the doorway leads to a room with layer Stencil 2, I would assign the StencilMask material with stencil value = 2.

If done correctly, you should be able to look through the quad to see the room the doorway goes to. If you find it hard to visualize the current room, feel free to temporarily change the room's layer to "Default", just make sure to change it back after. IMPORTANT: make sure that the stencil mask itself (the quad GameObject) is still on the Default layer and not on Stencil [i].

Portal Triggers

Now, to move between rooms and have our main stencil mask switch values for rendering the proper room, we have to set up some collisions. To this end, we will create an empty GameObject with a Box Collider component. Set the center of the collider to be (1,1,0). Move this GameObject to the center of the doorway and configure the Box Collider's size and bounds so that it fills the entire doorway's shape. Feel free to use other types of colliders, but Box Colliders are generally more convenient and they're quite consistent. For the best results, the thickness of the collider should be relatively small. NOTE: Make sure the GameObject has zero rotation. Now, we can attach the "Portal Trigger" script component to this GameObject. As seen below, there are a few fields we need to fill out:


Portal Trigger (Script)

Script	PortalTrigger
Start Room	0
End Room	0
Axis	Red
Reversed	☐

First, let's look at the second-to-last field: Axis. Axis gives you 3 options: Red, Blue, and Green. These correspond to the three axes Transform.right, Transform.forward, and Transform.up. To figure out which one to choose, select your GameObject with the "Portal Trigger" script component and look at the object in the Scene view. You should be able to see the 3 axes color-coded with the same colors Red, Green, and Blue. Choose the color axis that is parallel to the direction of movement through the doorway (i.e. normal to the quad).

Now, we can look at the other fields. First, we must determine the room numbers for the two rooms that share the doorway. These room numbers are just their index position in the array noted in the Room Assignment section.

For example, if my room GameObject is at index position 2 of the Portal Manager's "Rooms" array, its associated room number would be 3. Basically, it's 1-indexed instead of 0-indexed.

Now, determine which room will be the start and which will be the end. The order doesn't really matter all too much, but if the player must travel through one room before the other then we can assign them accordingly to "Start Room" and "End Room".

Now, refer back to the object in the Scene view. Look at the direction the axis arrow is pointing. If it is pointing at the "Start Room", then keep the "Reversed" field unchecked. Or else, check the "Reversed" field.

Room Transitions

In order to be more efficient with our use of stencil masks and not interfere with currently rendered rooms, we will turn off rooms that shouldn't be visible. We can do this by creating another empty collider GameObject as a child of the room object with the "Room Toggle" script component attached to it. Notice there are two lists: "Room On" and "Room Off". This directly correlates to the Start and End rooms in the Portal Triggers section. Room On will be the list of rooms that are turned on upon exiting the collider from the Start Room. Typically, we only want 1 room turned on and 1 room turned off, though different circumstances will require different needs. This GameObject should be positioned such that when the player passes through the collider, the rooms that are turned off will not be visible after passing it.

If the collider will poke outside of the room's walls, make sure to mark its tag as "Collider". If you do so, make sure it comes BEFORE any Portal Trigger GameObjects in the doorways.

Extras

This section will primarily cover the technical aspects of this impossible spaces system and how this implementation actually works.