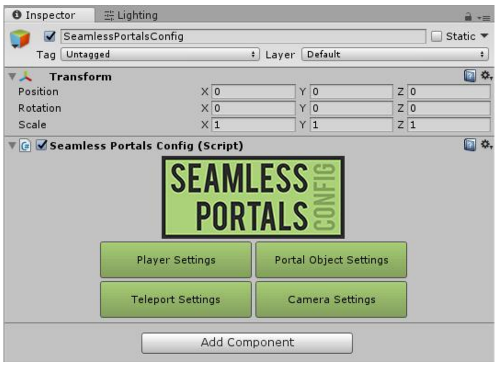
Seamless Portals for Unity – Documentation

Before configuring the portals, there are 4 steps that first need to be completed.

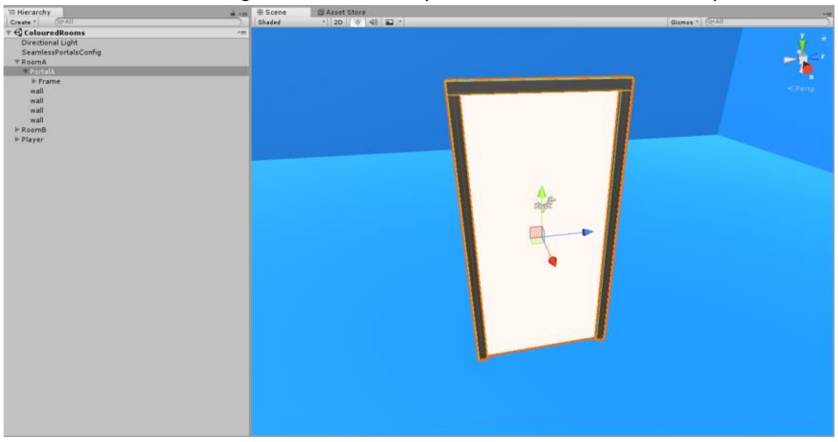
Step 1: Add the SeamlessPortalsConfig script to an empty GameObject



```
Inspector   Lighting
SeamlessPortalsConfig                     Static
Tag Untagged          Layer Default
Transform
Position   X 0   Y 0   Z 0
Rotation   X 0   Y 0   Z 0
Scale      X 1   Y 1   Z 1
Seamless Portals Config (Script)
```

Step 2: Create the two portal objects, these GameObjects are what will render as a portal, you can also add a frame or other details as shown below but these should be separate GameObjects.

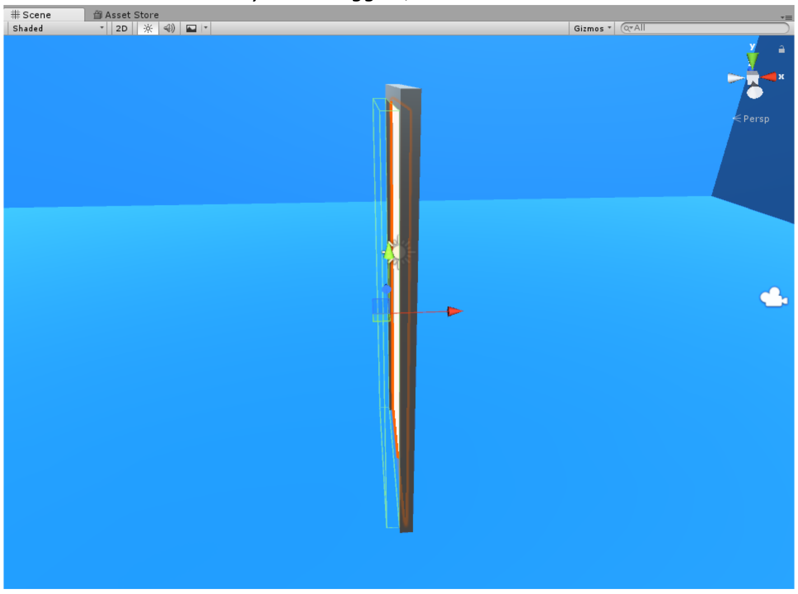
I would recommend using a cube with a very thin thickness rather than a plane so that it renders on both sides.



```
Hierarchy
ColouredRooms
  Directional Light
  SeamlessPortalsConfig
  RoomA
    PortalA
      Frame
    wall
    wall
    wall
    wall
  RoomB
  Player
```

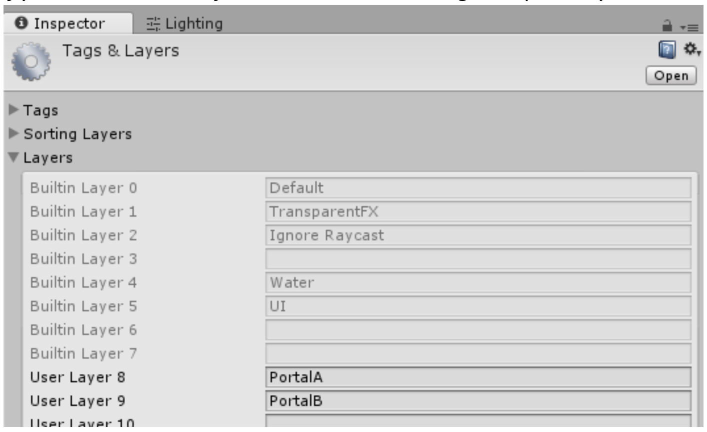
Step 3: Create the Colliders. The collider type must be a ‘BoxCollider’ and you can only have 1 collider per portal. For best results, you should offset the collider as shown below. The collider can be applied directly to the portal GameObject or to any other arbitrary GameObject.

Colliders are automatically set as triggers; The collider needs to be inside the view cube – this is explained later



Step 4: Add ‘PortalA’ and ‘PortalB’ layers

Once created you don’t need to apply these layers as it’s done automatically from the script.
If you want certain objects to be invisible through the portals you can also apply these layers to them.

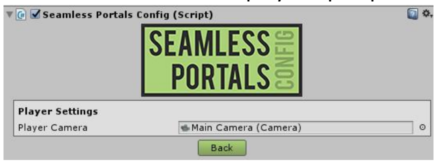


Builtin Layer 0	Default
Builtin Layer 1	TransparentFX
Builtin Layer 2	Ignore Raycast
Builtin Layer 3	
Builtin Layer 4	Water
Builtin Layer 5	UI
Builtin Layer 6	
Builtin Layer 7	
User Layer 8	PortalA
User Layer 9	PortalB
User Layer 10	

Once setup you can configure the settings through the SeamlessPortalsConfig script UI.

Player Settings

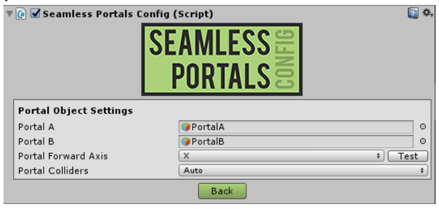
In this menu, there is just one setting – The player camera. You will need to drag in the main camera for your scene, the camera that renders the player’s perspective.



Player Settings
Player Camera: Main Camera (Camera)

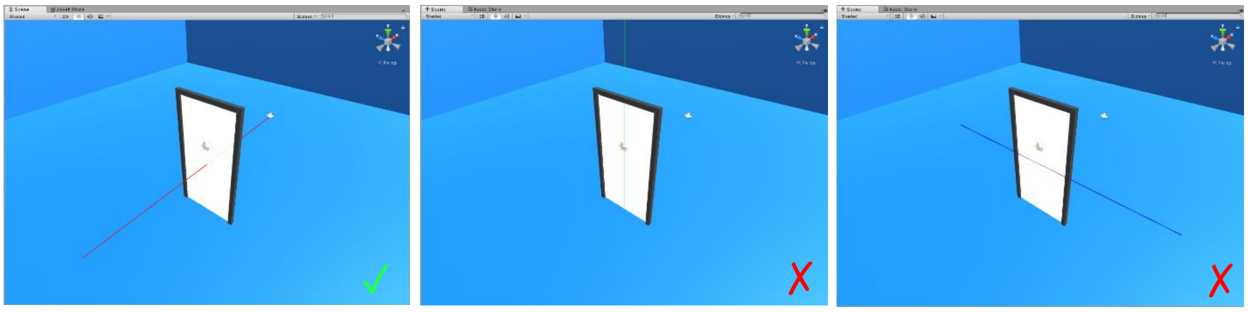
Portal Object Settings

The menu has a few settings. First you will need to drag in the two GameObjects you created earlier which are your portals.

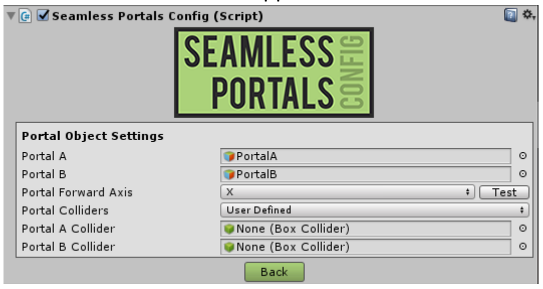


Portal Object Settings
Portal A: PortalA
Portal B: PortalB
Portal Forward Axis: X
Portal Colliders: Auto

You then need to define the forward axis of the portals, they must be the same for both. There is a ‘Test’ button in case you aren’t sure, this will draw the axis in the scene view and should pass through in the direction you would walk through the portal, as shown below.



The final setting under this menu is the colliders. If you applied the colliders directly to the portal objects then you can leave this on 'Auto' however if they are on a different GameObject then you will need to select 'User Defined' and drag them in to the fields that appear.

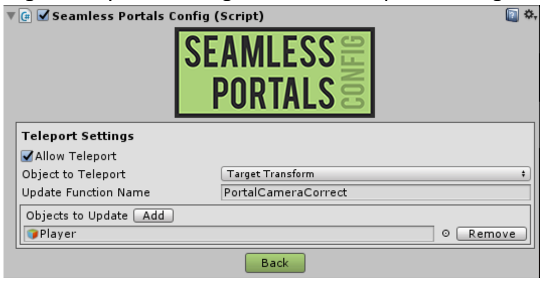


Portal Object Settings

Portal A	PortalA
Portal B	PortalB
Portal Forward Axis	X
Portal Colliders	User Defined
Portal A Collider	None (Box Collider)
Portal B Collider	None (Box Collider)

Teleport Settings

This menu allows you to configure the teleportation aspect of your portals. If you wish you can disable teleporting all together by unchecking the 'Allow Teleport' setting.



Teleport Settings

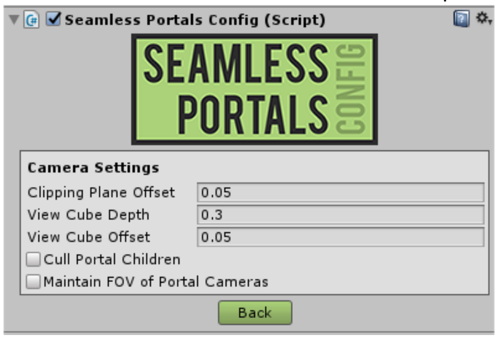
- ✓ Allow Teleport

Object to Teleport	Target Transform
Update Function Name	PortalCameraCorrect
Objects to Update	Player

'Object to Teleport' is the object that gets teleported when the 'Player Camera' enters the trigger colliders. For very simple player controllers you may be able to set this to 'Player Camera'; with this selected the camera GameObject itself will be teleported as you move into the collider. The camera GameObject will have its position and rotation set so that the transition is seamless. 'Camera Parent' works in the same way but the parent of the camera is teleported instead; the parent's position and rotation is changed instead of the camera itself. These two settings work fine for very simple payer controllers but you will most likely need to choose 'Target Transform'. For example, if your controller rotates the camera GameObject but moves the camera's parent – the position and rotation is on separate GameObjects. With 'Target Transform' it's left to you so that it should work with any player controller. You will need to create a function in your player controller with a Transform parameter; in the config, you then need to enter the name of this function and all the objects with this function that should be called. Then when you enter the collider this function will be called on all listed GameObjects and will be fed a Transform with the target position and rotation. From here you can set the position and rotation of your player, for example setting the position of the camera parent and the rotation of the camera itself.
If you are struggling to understand this then checkout the 'SimplePlayerController' script included in the demo folder, if you still need help then please email me at josephpatrick2013@hotmail.com

Camera Settings

The final menu is the Camera Settings. These settings are a little advanced, you will probably find you can leave most of these at their default values but here is an explanation of them.


Seamless Portals Config (Script)

SEAMLESS PORTALS CONFIG

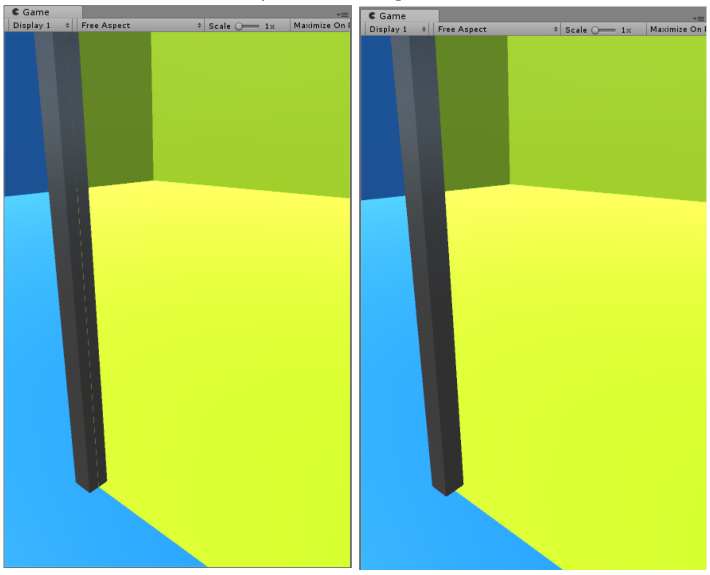
Camera Settings

Clipping Plane Offset	0.05
View Cube Depth	0.3
View Cube Offset	0.05

☐ Cull Portal Children

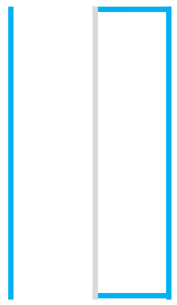
☐ Maintain FOV of Portal Cameras

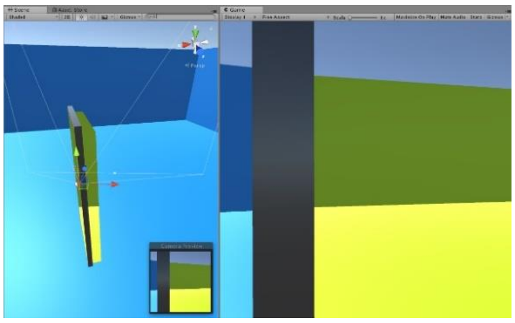
'Clipping Plane Offset requires' some explanation of how this script works to explain what this setting is for.
The portals are rendered with render textures, what is rendered on the portal is captured by a camera that moves around as the player does to give the right perspective. This means that normally objects between this camera and the portal would block the view, so they are clipped with a clipping plane. This plane is normally aligned with the portal but the offset allows you to move it back/forward so that more/less is clipped. Generally, you don't want extra clipping but you might want to reduce the clipping slightly so that the frame of the portal isn't clipped. As I said you mostly won't need to mess with this much, but if you find that the frame of your portal is clipped then try increasing this until you get the result you want. This number can be positive or negative to set the direction of the offset.



This image shows this in action, in the left image the offset is set to 0, in the right image it is set to 0.05. The difference is small but as you can see in the left image, there is slight clipping which causes a green line to run up the middle of the frame, where as this is not visible in the right image.

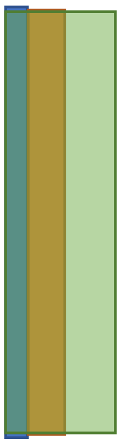
As you walk into the portal object they are prevented from rendering to prevent flickering as the cameras passes through it, in its place a 'view cube' is rendered, this is just a cube which renders the same as the portal but as the camera is inside the cube there is no flickering from passing through an object. The back face is removed so you can still look out behind you. This simple diagram show's this better, on the left the single line represents the portal; as you can see to walk through the portal the camera must pass through the object – which would cause flickering. Instead, as the camera enters the portal, the view cube (right) is rendered instead (the grey side is not rendered) so the camera can move into the cube and then the camera is teleported and the cube disappears. A screenshot also shows the view cube in scene and game view.





Again, you don't really need to know how it works. 'The 'View Cube Depth' setting is the depth of the cube and the 'View Cube Offset' is the positional offset of it. You should set the offset so that the cube is aligned with the edge of the portal as opposed to the centre; setting the offset the half the width of the portal (frame) should work.

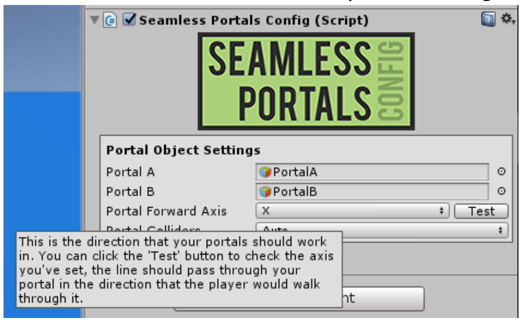
Going back to the colliders, they should be contained in the view cube as this diagram shows. Blue is the portal itself, orange is the collider and green is the view cube



The last two settings are 'Cull Portal Children' and 'Maintain FOV of Portal Cameras'. If you want the children of the portal objects to be invisible when looking through the other portal then check this, this is the same as applying the 'PortalA'/'PortalB' layers to these objects. If the FOV of your player camera is not constant then you should check the Maintain FOV box otherwise the portals will not look right.

Additional Info

You can hover over the labels of any of the settings and a tooltip will show to remind you what it does.



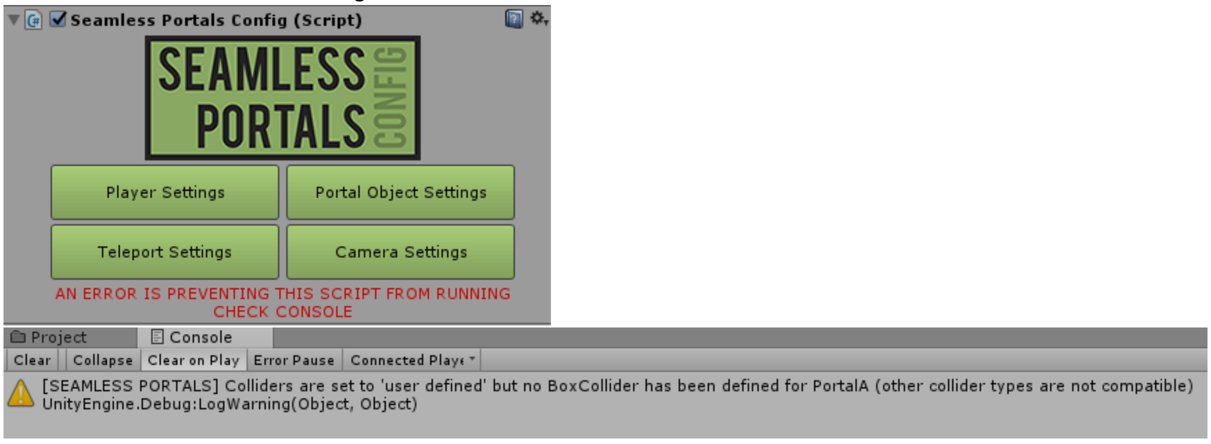
Seamless Portals Config (Script)

Portal Object Settings

Portal A	PortalA
Portal B	PortalB
Portal Forward Axis	X

This is the direction that your portals should work in. You can click the 'Test' button to check the axis you've set, the line should pass through your portal in the direction that the player would walk through it.

If there is an error with your setup then when you press play the config menu should show a warning and more information will be in the console log.



Seamless Portals Config (Script)

AN ERROR IS PREVENTING THIS SCRIPT FROM RUNNING
CHECK CONSOLE

```
[SEAMLESS PORTALS] Colliders are set to 'user defined' but no BoxCollider has been defined for PortalA (other collider types are not compatible)
UnityEngine.Debug:LogWarning(Object, Object)
```

And finally, if you get stuck with anything please feel free to email me at josephpatrick2013@hotmail.com and I will get back to as soon as possible.