Portal Kit Pro User Manual

Table of Contents

Introduction

PortalKitPro has a wide variety of applications across many game styles, leading to many possible issues; however, regardless of if you are working on a game with portals as a mechanic, a visual effect, a method of area transferral or something else entirely, this guide will hopefully answer any questions that you may have. If it does not, please do not hesitate to contact me from the store page. If you would like to report a bug, suggest a feature, or buy development time in relation to this package for a nominal fee, then please contact me in the same manner. PortalKitPro aims to be general-application, but as a developer I can only account for so much in a single release.

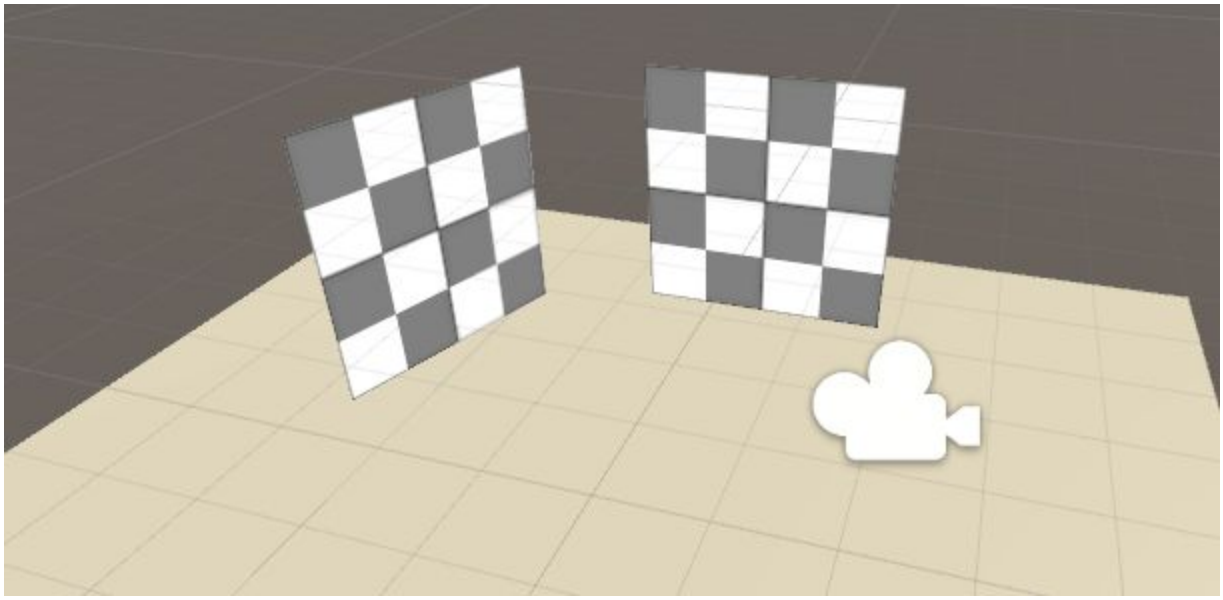
Quickstart Guide

Before beginning, please note:

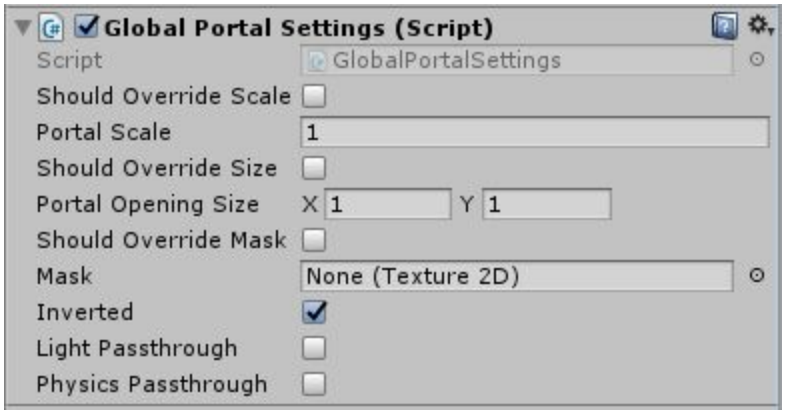
1. **If you are a VR user, please delete the NonVR folders under both /Scripts/ and Resources/Prefabs/.** If you are a nonVR user, please delete the VR folders under those same path names.
2. **You must import the tags/layers correctly for this asset to function properly.** Under the !!INSTALL THIS!! Folder in the root of the asset, there is a TagManager file. In your Operating System's file browser, you must place this in your project folder's /ProjectSettings/ folder. This will overwrite your tags and layers. If you would rather add PortalKitPro's required layers by hand, please refer to the *Installation Instructions* file in the same folder. **If you do not do this, the asset will not function**.

Getting a portal system to function at its most basic is extremely easy.

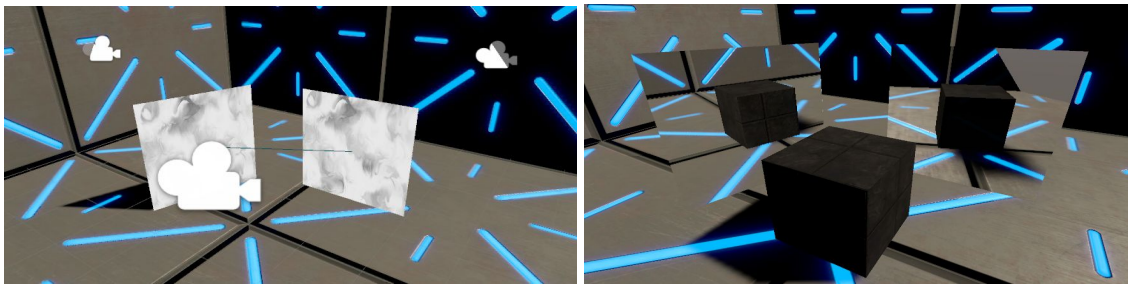
1. Open a new scene, navigate to the /Resources/Prefabs folder, and drop in two **Portal Spawner** prefabs. They should appear as checkerbox quads.



2. Go to the properties of each portal spawner and drag the other portal's game object from the hierarchy window into the **Target Initializer** field.
3. Drop a **Global Portal Settings** prefab into the scene.



4. Click play. In the inspector, the portals will spawn all of the components that they require and they will connect themselves with a colored line, while the game view should have them visually connected to one another.



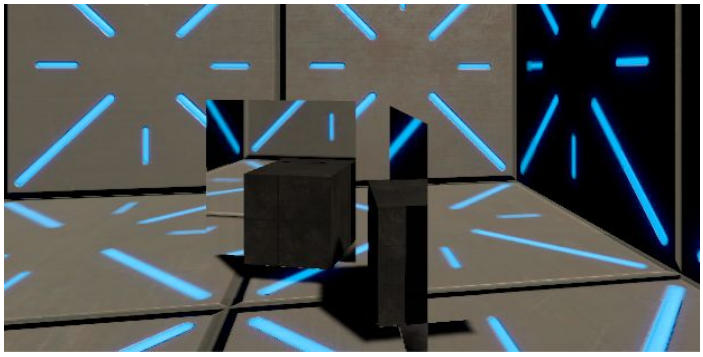
That's it for the quickstart guide! Read on for more details and application usages.

Notes

1. Ensure that "Portal only" is not visible to the camera.
2. Ensure that the camera looking through the portals is the main camera (you can edit this within PortalCamera.cs if you want other cameras to render portals for unique effects)
3. If your render result appears blacked out or upside down, tick the Inverted checkbox on the **Global Portal Settings** object you spawned into the level.

Objects and Portals

Allowing objects to teleport is fairly easy. Simply take any object which you want to teleport, and add the **Teleportable** script to it. When you do this, the object will spawn all of the necessary objects and components to teleport seamlessly. It will also replace the material with one that cuts off the object as it enters the portal. If you would like to have a custom material, please refer to the **PixelExclude** shader and copy the Clip logic into your own. Almost any object may be **Teleportable**, but it is important to note that skinned mesh renderers may lag your game for a couple of frames as they spawn in due to the large amount of transformation data that must be copied for bones. Set the **Root** field to the root of the gameobject to be teleported (if the object itself is not its own root root), Because of this, it is recommended that you use skinned mesh renderers with few bones or that you spool or prespawn your skinned mesh renderers. If you wish for the object to be able to visually enter portals but not actually cross through the threshold, then tick the **Vis Only** checkbox.



Scripts and Portals

While gameobjects will teleport as expected, scripts will not teleport until the object is entirely through the portal. If you would like to have scripts teleport as soon as their object travels through the portal, then have them inherit from **TeleportableScript.**

```
public class Grabber : TeleportableScript {
```

In addition to inheriting the script, you should also preface all monobehaviour methods with *new* and call the *base* version of them after overriding.

```
new void Start() {
    base.Start();
```

All **Teleportable Script**s MUST be within the scope of a **Teleportable** to function. A useful side effect of this is that any and all **Teleportable** data that you need for a script will be exposed, such as its root and contained objects. Examples of teleportable scripts are within the **Scripts/VR/TeleportableScripts** folder of the asset.

Notes

1. Objects will not “actually” teleport until they are all the way through the portal, although they will appear to seamlessly travel.

2. TeleportableScripts should generally be on their own game objects with no children (unless the script directly requires use of said children as they will be unparented and moved to the doppleganger as they enter the portal.

Physics

The default settings for physics should operate as expected. As a **Teleportable** physics object nears a portal, the objects directly behind that portal are ignored for said object's collision (and its children's collisions), while the **Portal Edge Wall**'s collisions are enabled for the object. If you have a portal with a custom shape, be sure to assign the parent of all edge wall objects to the **Portal Edge Wall** field of the prefab's **Portal Script**, residing in the hierarchy directory of [prefabname]/PortalRenderer/BackWall. If you want physical objects to be able to collide with colliders on the other side of the portals, enable the **Physics Passthrough** checkbox on your **Global Portal Settings** object.

Raycasting

Raycasting through portals is possible via the **TeleportableRaycast** method inside **PortalUtils**. Item1 of the tuple will be the final ray that was cast before the ray hit its target, and item2 of the tuple will be the RayCastHit. This is useful in several cases, such as if you wish to get the directional information from the raycast for applying force (I.E. a gun)

```
Tuple<Ray, RaycastHit> hitInfo = PortalUtils.TeleportableRaycast(ray, 100, ~0);
if (hitInfo.Item2.transform) {
```

Player Physics

Player physics can end up being a little tricky, depending on your setup. If you're lucky, you will be able to add **Teleportable** to the collider that your player is using, set its **Root** to the root of your player, and have it function perfectly. However, this may not be the case. For instance, if you look at **Movement.cs**, you will see my implementation of physics-based VR movement teleportation, which was a gigantic pain. The simpler your camera system the better. If you need guidance, refer to that script as starting point. **Movement.cs** is implemented in the **[CameraRig]** prefab located in /Resources/Prefabs/VR folder.

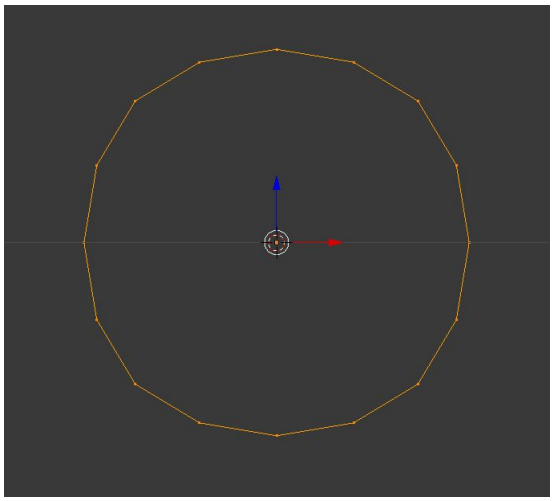
Notes

1. It is highly likely that objects with physics constraints will not behave as expected if one of the objects has a **TeleportableScript** on it. In these instances, consider having a node child on the physics object with the script on it instead.
2. This release does not have physics passthrough, so if an object enters a collider on the other side before it travels all the way through it will not collide with it. This is a planned feature.

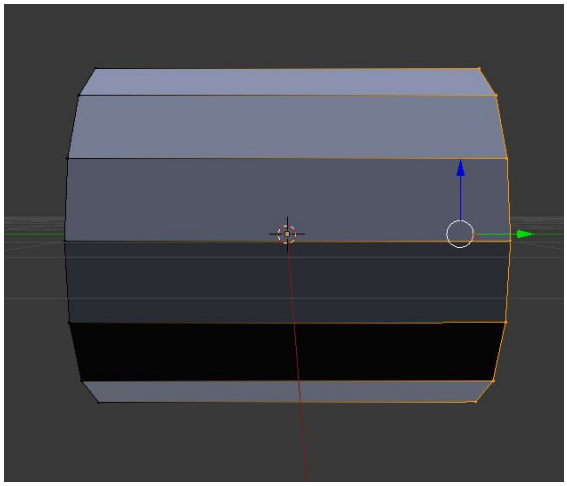
Custom portals

It is possible to create custom portal definitions, if your game has the need for it. To create a custom portal, follow these steps:

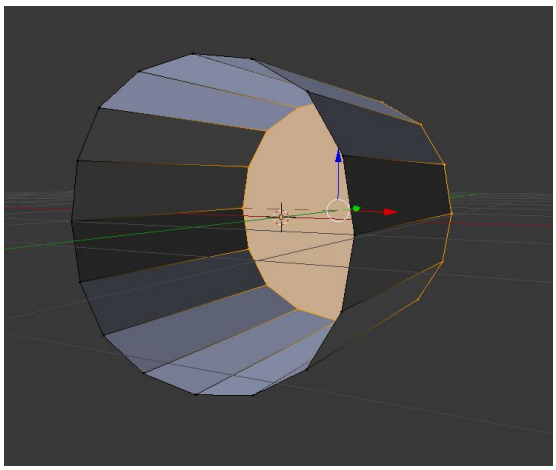
1. Draw a two-dimensional edge mesh of your portal's desired shape in your 3d modeling program of choice, approximately one by one unit



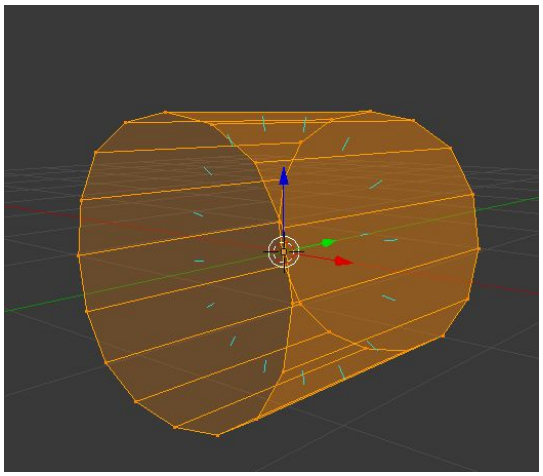
2. Extrude the edge mesh by one unit



3. Fill the back of the mesh



4. Ensure that all normals are facing inwards



5. Duplicate an existing Portal Prefab, and apply your new mesh to the **BackWall** mesh filter component. In addition, offer a proper replacement for the **Placeholder** object.
6. Open your new prefab's **Portal Edge Wall** game object and add colliders to define a shield around your portal's borders
7. Apply the new portal prefab to a **Portal Spawner** and/or **Global Portal Settings**

Notes:

These are for standard portals with seamless transitions. For 3d portals for the purpose of visual effects, assign a 3d mesh to your BackWall object and tick the **Is 3d** checkbox on your portal controller script.

Visual effects

Portal Placeholders

When viewing a portal through another portal, a replacement material will be rendered. To change this effect, go to your portal prefab and change the **Portal Placeholder**. Portal view through is a feature planned in a future update.

Lighting

-Feature coming in an update very soon-

Optimization

Graphics Optimization was the one of the largest focus targets of this asset package, and for good reason: Portals are extremely visually expensive. While looking through a portal, a typical game must render the scene twice, while a VR Game must render it four times. For VR in particular this is a problem, as dropped frames are incredibly noticeable. To solve this problem, the portals only render what is absolutely necessary to render, significantly reducing render time as the player gets farther away from the portals or as the portals reduce in size. It is absolutely imperative that the following are followed for acceptable performance:

1. Occlusion culling must be baked, and used for best performance. In the current version of Unity, Occlusion Matrices are generated incorrectly which is exacerbated with oblique culling. This bug will be fixed soon according to the Unity Staff, but in the meantime please either disable oblique culling or Occlusion culling.
2. Dynamic lights should be used sparingly
3. Dynamic lights with passthrough enabled should be used even more sparingly
4. Dynamic lights should have the lowest acceptable quality settings
5. Critically examine whether or not extremely expensive shaders should be on a layer that portals can render

Non-VR Usage

If you are a VR user, please delete the NonVR folders under both /Scripts/ and Resources/Prefabs/. If you are a nonVR user, please delete the VR folders under those same path names.

Troubleshooting

My portals appear to be upside down!

Try again with the **Inverted** checkbox on your **Global Portal Settings** prefab in the opposite position

My portals are invisible!

Ensure that you have a **Mask** applied to the **Mask** slot on your **Portal Spawner** prefab. Also ensure that your camera is tagged with the **MainCamera** tag.

My portals look like a strange white effect / are pure black!

Ensure that the camera rendering the portal has **PortalOnly** removed from its visible layers

Passthrough isn't seamless!

Ensure that your main camera has a box collider and a kinematic rigidbody on it. Additionally, ensure that there is a collider which your camera is parented to called **PlayerExtents**.

My game runs slowly!

Ensure that you are following the steps in the **Optimization** section of the manual, as well as Unity's optimization recommendations. Additionally, try to avoid having both your scene and your game views open while playing if you can avoid it. Finally, if you run it and you get TERRIBLE performance, restart the run.

Everything is still broken!

Please, don't be afraid to contact us if you have persistent issues. If you would like to contact me personally, please do so at **chunkworksstudios@gmail.com**