
Portal Kit Pro User Manual For Version 5.53

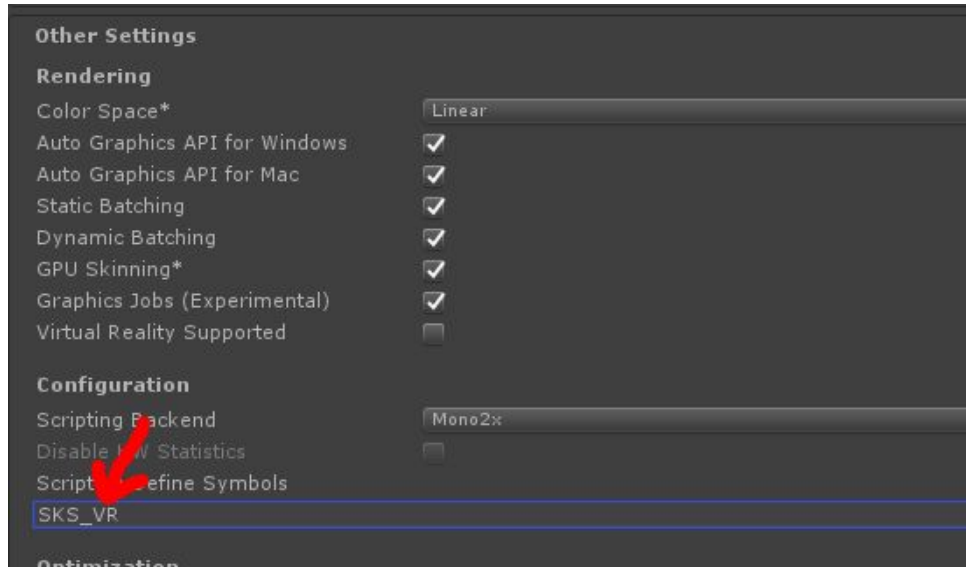
Table of Contents

Table of Contents	2
Extremely important update PSA	4
Introduction	5
Setup and Quickstart	6
Notes	8
Additional Setup: The Portal Controller Inspector	9
Advanced VRTK Setup Guide	10
Menu Location and Function Overview	11
Setup	11
Global Portal Settings	11
Detect RequireComponent Dependencies	12
Objects and Portals	13
Scripts and Portals	13
Notes	14
Physics	15
Raycasting	15
Player Physics	15
Notes	16
Custom portals	16
Notes:	18
Visual effects	18
Portal Placeholders	18
Optimization	19
VR and Non-VR Usage	20
Support for Locomotion Systems	21
VRTK	21
UFPS	21
Troubleshooting	22

My portals are invisible!	22
My portals look like a strange white effect / are pure black!	22
My portals look strange and far away!	22
My portals don't look properly 3d while in VR mode!	22
Passthrough isn't seamless!	23
Teleportable objects behave strangely and my console contains errors!	23
My game runs slowly!	23
The Change VR Mode dialog does nothing!	23
None of these fixed my problem, and my portals are still having issues!	24
Something/Everything is still broken!	24

Extremely important update PSA

When updating the asset, please remove and reinstall the asset after update. If the asset is mission-critical, PLEASE make a backup before doing this. Additionally, if you are using VR portals, ensure that you delete the compile directive before reimport (under File > Build Settings > Player Settings). Delete the SKS_VR directive and hit enter, then click outside of the text box and ensure that it is gone.



Failure to do this can and will result in a botched import or a broken asset.

Introduction

PortalKitPro has a wide variety of applications across many game styles, leading to many possible edge cases; 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.

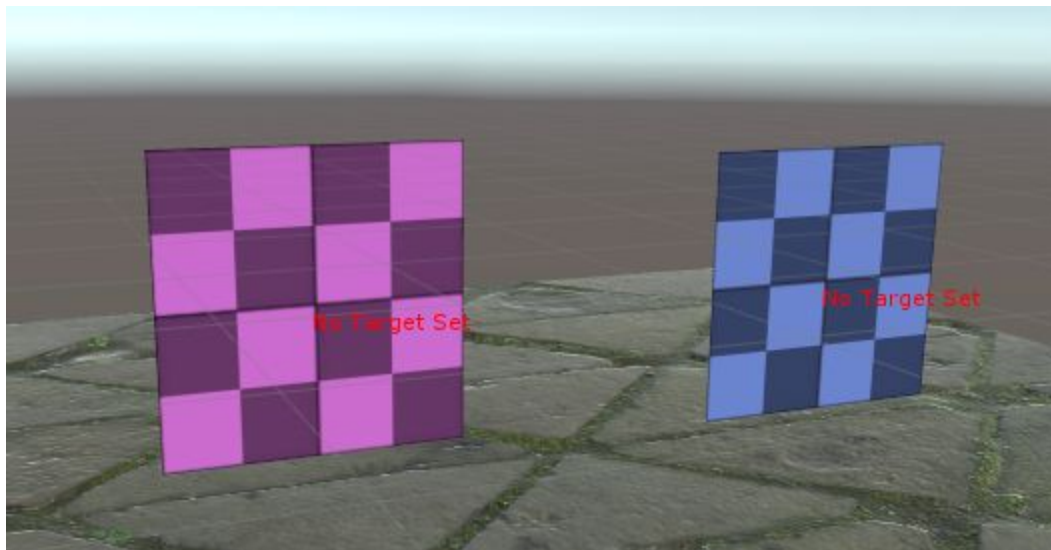
Setup and Quickstart

Before beginning, please note:

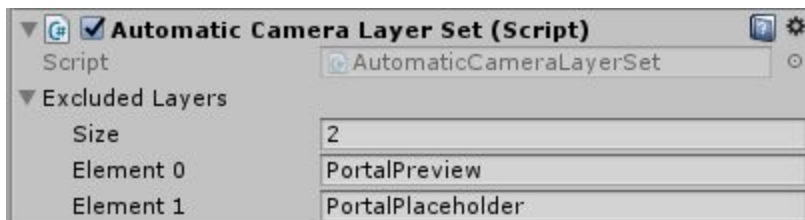
1. Be sure to set up your project on import through the dialog option. If you selected the wrong option or wish to change your project type, navigate to **Tools/SKStudios/PortalKit Pro/Setup** and reset it. If you have selected VR by accident, you will want to delete the automatically-imported VRTK content.
2. The asset will automatically create all the layers that you need and assign the prefabs to the appropriate layers. This does not delete existing layers.

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 checkerboard quads. These may turn invisible when selected if you have portal previews enabled, but that is not a problem.

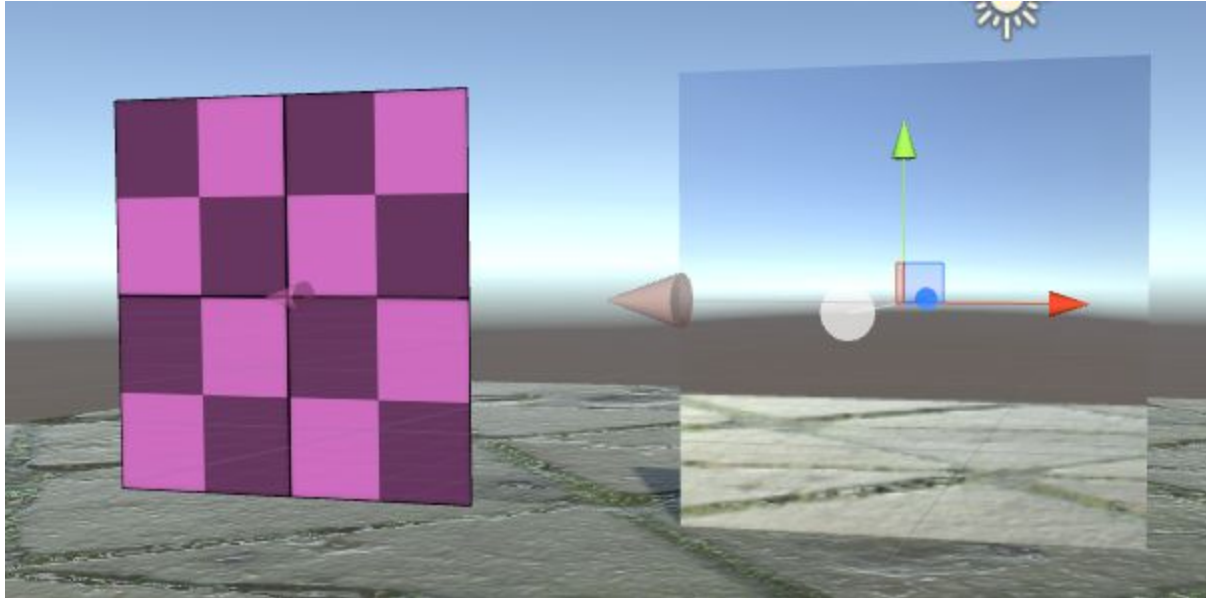


2. Next, we will setup the camera. Please, ensure that your camera has the **MainCamera** tag and has **PortalOnly**, and **PortalPlaceholder** ignored in its layer mask. Alternatively, you may add a **AutomaticCameraLayerSet** script to the camera, and set it up as follows:

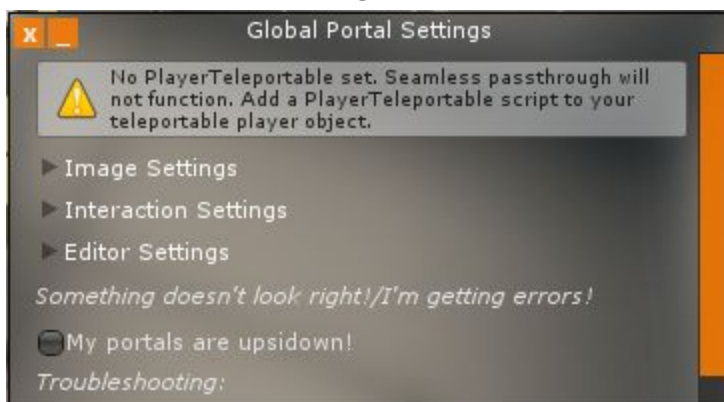


3. Go to the properties of each portal spawner and drag the other portal's game object from the hierarchy window into the **Target Controller** field. Colored arrows will begin traveling

from one portal to the other, signifying a connection. If you have Portal Previews enabled, selecting one of the portals will result in a preview through that portal. Please note that previews are experimental, and may not always function as expected.



4. Ensure that your RequireComponent dependencies have been detected underneath **Tools/SKStudios/Detect RequireComponent Dependencies**. If it has not automatically scanned, press the Detect all RequireComponent dependencies button.
5. Check the **Global Settings** window to ensure that your settings are as you want them. There are several sub-headings, and some options are to taste. If you have closed this window, you can open it with **Tools/SKStudios/PortalKit Pro/Dock GlobalPortalSettings Window**

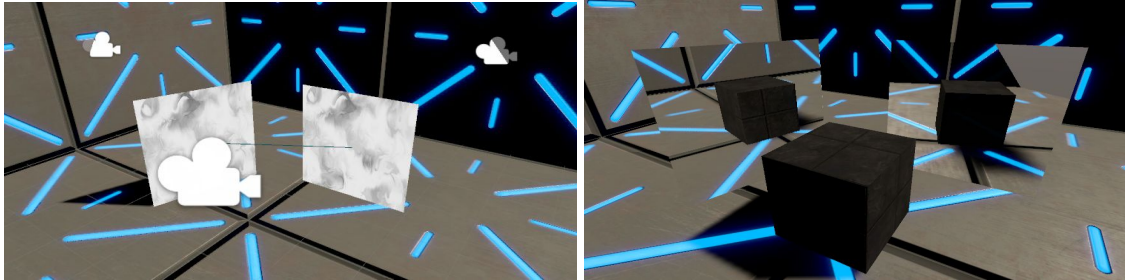


6. Click play. In the inspector, the portals will spawn all of the components that they require and they will connect themselves with a colored line, while the game view should have them visually connected to one another.
7. If you want seamless camera passthrough to function, add a **Teleportable** script to the

camera, and put a collider on it. Also, add a **PlayerTeleportable** script to the object. When you do this, your Global Portal Settings window will update automatically:



Now, when your camera moves through portals, it will appear seamless.



7. Please make sure that every single time that you add or remove a package or scripts that include the [RequireComponent] annotation within them, you re-scan for RequireComponent dependencies. Failure to do so will result in strange behavior from teleportable objects and will possibly crash Unity.

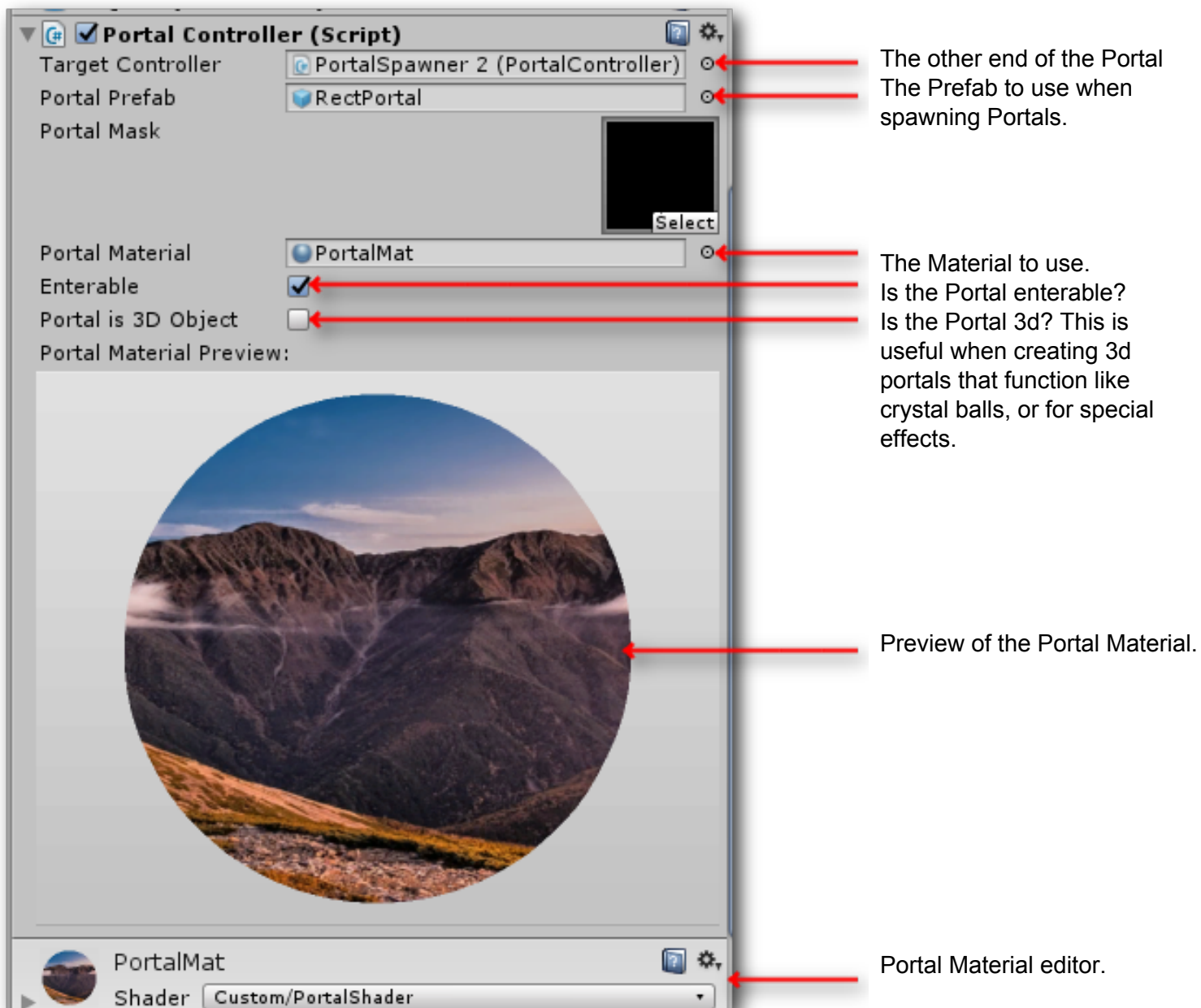
That's it for the quickstart guide! Read on for more details and application usages.

Notes

1. Ensure that "Portal only" and "portal placeholder" are both not in the camera's culling mask.
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)

Additional Setup: The Portal Controller Inspector

You will likely want to familiarize yourself with this editor, as it controls the majority of the settings that you will be using after you have set up the import and Global settings to your liking.



The screenshot shows the 'Portal Controller (Script)' Inspector window. It has several fields and checkboxes. Red arrows point from text labels to specific UI elements:

- Two arrows point to the 'Target Controller' and 'Portal Prefab' dropdown menus, which are set to 'PortalSpawner 2 (PortalController)' and 'RectPortal' respectively. The label is: "The other end of the Portal The Prefab to use when spawning Portals."
- An arrow points to the 'Portal Material' dropdown menu, which is set to 'PortalMat'. The label is: "The Material to use."
- An arrow points to the 'Enterable' checkbox, which is checked. The label is: "Is the Portal enterable?"
- An arrow points to the 'Portal is 3D Object' checkbox, which is unchecked. The label is: "Is the Portal 3d? This is useful when creating 3d portals that function like crystal balls, or for special effects."
- A large circular preview of a mountain landscape is shown. An arrow points to it with the label: "Preview of the Portal Material."
- An arrow points to the 'PortalMat' material editor at the bottom, which shows a 'Shader' dropdown set to 'Custom/PortalShader'. The label is: "Portal Material editor."

Advanced VRTK Setup Guide

Getting VRTK functioning can be a bit of a hassle, but thankfully their active community leads to a robust wiki, with helpful contributors willing and able to answer most questions. If you have any questions about getting VRTK working, please refer to this page: [VRTK Start Wiki](#)

As VRTK is constantly changing, maintaining up-to-date support can be quite challenging. As VRTK is totally open source and has a very active community, builds tend to be temporal, and there is no guarantee of the newest build functioning. Realizing this, PortalKit Pro comes bundled with the latest known functioning version of VRTK. If you select “VR” from the import selector, VRTK’s core scripts as well as several VRTK-related scripts from PortalKit Pro will automatically install themselves into your asset folder.

When getting started with VRTK, the first thing that you will want to do is place the **Resources/Prefabs/VR/PortalKitPro VRTK Starter 3.0** prefab into your scene. When expanded, you will find all of the required scripts installed as per the guide above. Should you want an SDK other than the Simulator or SteamVR, please follow the VRTK quick start guide for guidance.

Menu Location and Function Overview

Setup

Accessible from: **Tools/SKStudios/PortalKit Pro/Setup.**

Function: Setup tools

Buttons:

- Documentation: Opens the Docs.
- VR: Enable VR Mode.
- NonVR: Enable NonVR Mode.
- Detect RequireComponent Dependencies: Required to be clicked after VR or NonVR is selected to fully set up the asset.
- Reassign Layers: Used to reassign layers on all of the asset’s prefabs. Used when changing the order of layers that the asset uses.

Global Portal Settings

Accessible from: **Tools/SKStudios/PortalKit Pro/Dock GlobalPortalSettings Window**

Function: Settings that affect all portals globally

Buttons:

- Image Settings
 - Override Masks on all PortalSpawners: When enabled, allows the setting of a global mask for all portals in the project.
 - SinglePass Enabled: Reflects whether or not Single-Pass stereo rendering is being utilized while using SteamVR.
- Interaction Settings
 - Enable Physics Passthrough: Enables collision with objects on the other side of portals.
 - Enable Physics Model B: Enables realistic simulation of conservation of momentum with portals. More realistic, but not necessarily more fun.
- My portals are upsidown!: Used to fix different UV alignment issues on other platforms that cannot be automatically detected.

Detect RequireComponent Dependencies

Accessible from: **Tools/SKStudios/Detect RequireComponent Dependencies**

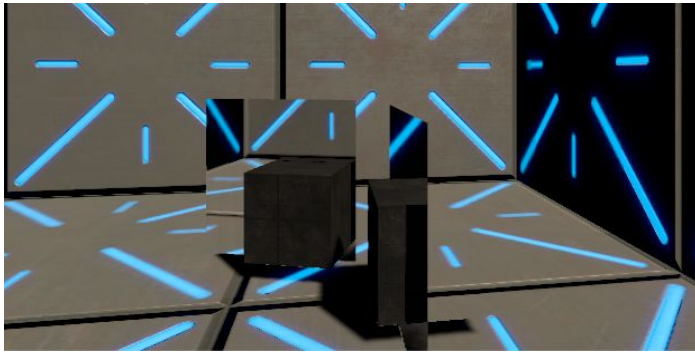
Function: Allows for the guaranteed deletion of components. Very important for doppleganger spawning. This is necessary, as Unity will refuse to delete any component marked with the RequireComponent annotation. If you have circularly-defined requirecomponent dependencies, this will cause a crash. Please avoid doing this, for this asset and as a general practice (try to delete them individually yourself- it is impossible).

Buttons:

- Detect All RequireComponent dependencies: Populate the RequireComponent graph
- Delete dependency data: Used to clear the data. If the data will not clear, open Dependencies.cs and clear it by hand (this only happens if large amounts of classes are deleted at the same time and the asset can't catch it)

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 objects with large amounts of

children may cause slight lag as they are instantiated due to the large amount of data that must be copied. 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 Scripts** 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/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. VRTK is also included, and is recommended to be used as a first option. To use VRTK, place the **PortalKitPro VRTK Starter** in your scene.

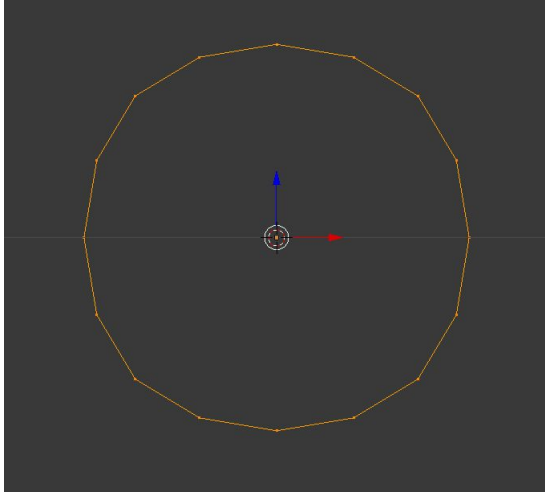
Notes

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.

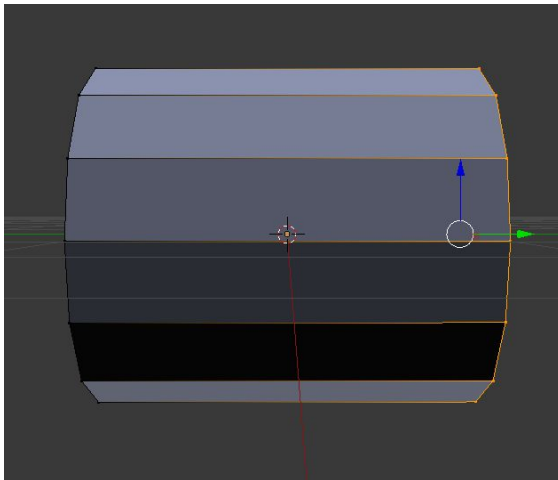
Custom portals

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

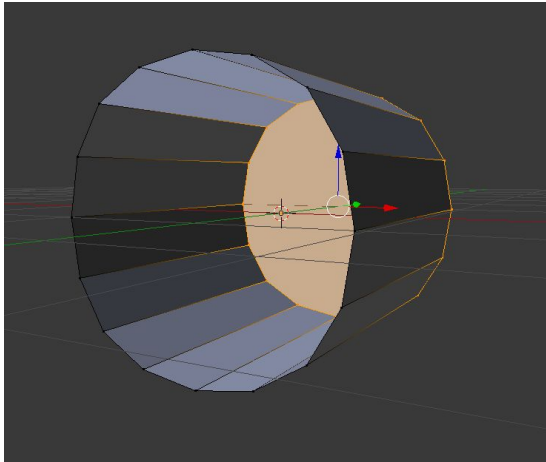
1. Draw a two-dimensional edge mesh of your portal's desired shape in your 3d modeling program of choice, approximately one by one unit



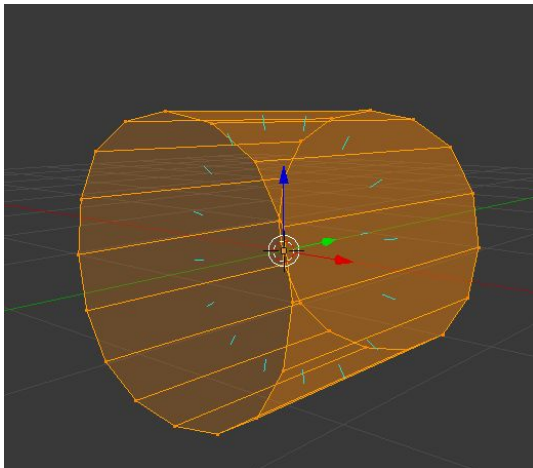
2. Extrude the edge mesh by one unit



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**

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 and maximum recursion has been reached, a replacement material will be rendered. To change this effect, go to your portal prefab and change the **Portal Placeholder**.

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 on the portal's cameras, but still use it on the main scene camera.
2. While using VR portals, set your player stereo mode to Single Pass if your game supports it. You will get a much higher frame rate!
3. Dynamic lights should be used 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.

VR and Non-VR Usage

Be sure to set up your project on import through the dialog option. If you selected the wrong option or wish to change your project type, navigate to **Tools/SKStudios/PortalKit Pro/Setup** and select the correct option.

Support for Locomotion Systems

VRTK

The current build of the asset includes support for all non-teleportation VRTK movement scripts. Support for more features is still being worked on (there is a lot to do), but as it stands it functions for non-teleportation locomotion, and it works with the various grabbing and interaction scripts. Use the **Resources/Prefabs/VR/PortalKitPro VRTK Starter 3.0** prefab or load the **Advanced VRTK Example** scene to try it out!

UFPS

UFPS was... incredibly hard to make function well, to say the least. Due to how all of the methods are privatized and the asset is proprietary, there was only so much support that was able to be added. With that being said, UFPS works with non-accelerative movement for portals that do not change the player's roll. To get UFPs functioning, uncomment **Scripts/Non VR/UFPS Fix** and add it to the base of your FPS controller. Then, select the head camera, and add a box collider to it with **IsTrigger** enabled.

Troubleshooting

My portals are invisible!

Ensure that you have a **Mask** applied to the **Mask** slot on your **PortalSpawner** prefab. Also ensure that your camera is tagged with the **MainCamera** tag. If these don't work, try reassigning your layers from the button in the **Tools/SKStudios/PortalKit Pro/Setup** dialog.

Reassign layers (used for custom layer setups for existing projects)

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

Ensure that the camera rendering the portal has **PortalOnly** and **PortalPlaceholder** removed from its visible layers. Ensure that you have the correct project type set from the **Tools/SKStudios/PortalKit Pro/Setup** window. Additionally, try changing the following settings on the GlobalPortalSettings window, restart play mode, and check if your issue has been fixed.

Something doesn't look right!/I'm getting errors!

☐ My portals look strange! (restart playmode)

☐ My portals are upsidown! (restart playmode)

My portals look strange and far away!

Be sure to set up your project on import through the dialog option. If you selected the wrong option or wish to change your project type, navigate to **Tools/SKStudios/PortalKit Pro/Setup** and reset it.


My portals don't look properly 3d while in VR mode!

Please ensure that you have a **PKProVRTracking** component added to your camera rig. Take a look at the **VRTK Starter** prefab, if you haven't already.

Passthrough isn't seamless!

Ensure that your main camera has a box collider with **IsTrigger** set to true, and a kinematic rigidbody on it. Additionally, ensure that there is a collider which your camera is parented to with a **Teleportable** and **PlayerTeleportable** script attached to it. Check the GlobalPortalSettings window to verify that it is detected.

Player Teleportable

 PlayerExtents

Teleportable objects behave strangely and my console contains errors!

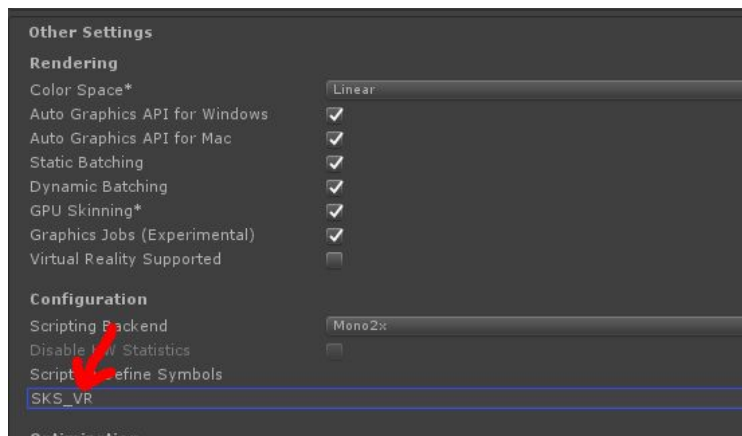
Open the **Tools/SKStudios/Detect RequireComponent Dependencies** window and ensure that your RequireComponent dependencies have been properly detected.

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, by selecting **Maximize on play** from the game window. Finally, if you run it and you get TERRIBLE performance, close and re-open unity. The problem should disappear.

The Change VR Mode dialog does nothing!

For some reason, the Change VR Mode dialog does not work for some users. To remedy this, simply go to File > Build Settings > Player Settings > Other Settings. Then, set the keyword **SKS_VR** to enable VR mode, or remove it to disable VR Mode.



None of these fixed my problem, and my portals are still having issues!

In some builds of Unity, the Scene view camera will undetectably render portals. Try running your game with "Maximize on play" selected in the game view and see if that helps.

Something/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 **support@skstudios.zendesk.com**