# Sci-Fi Portals

# Overview

The Sci-Fi Portals asset is your gateway to immersive interdimensional experiences! Whether you're creating a space exploration adventure or a futuristic RPG, this portal is the perfect addition to your project. With the ability to effortlessly walk through portals, players will be mesmerized by the fluidity of their teleportation experience. Craft gameplay scenarios by combining two portals seamlessly, allowing complex teleportation mechanics and dynamic level design. Whether you're transporting players to distant galaxies or alternate realities,with this asset you can get seamless teleportation by simply walking through the portal like through a simple door.

### Key features:

### Seamless teleportation

Players can effortlessly walk through portals as if they were walking through a simple door. Say goodbye to clunky transitions as users seamlessly traverse between worlds with a single step. From a third-party perspective, witness the objects appear to seamlessly transition through the portal, adding a touch of realism and immersion to your project

### Perfect portal camera alignment

Through mechanisms like perspective division and oblique projection, the portal's cameras adjust based on the player's camera, creating the most realistic portal view possible. This view mechanism is the base of the seamless teleportation, since it exactly shows the future view of the player's camera.

### Slicing Shader

With this shader, the mesh of objects passing through the portal is precisely clipped at the portal's center, creating the illusion that they are truly traversing through the dimensional gateway.

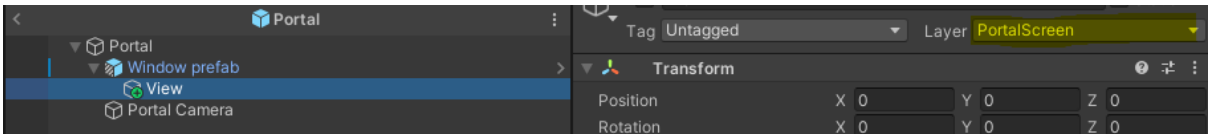
### Intuitive Integration

Integrate the portals into your Unity project with user-friendly scripts and documentation. No need to be a coding expert – this asset is designed for easy implementation.

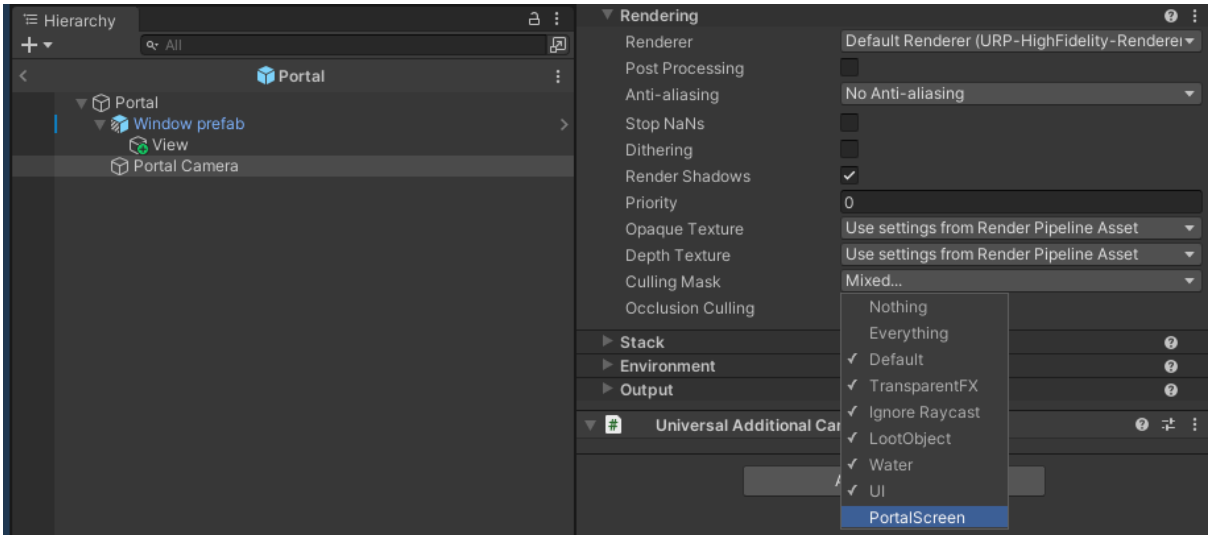
# Setting up the asset

The setup of this asset is pretty simple. We only need to ensure to add one new Layer, the *PortalScreen*. (if not already added automatically). This is needed because the portal views should be set to this layer. The portal cameras will exclude this layer for rendering, to ensure a proper portal view. We do not want to render the view object Itself, only what is seen through the portal.

So add the new layer and go to SpaceFusion→SF Portals → Prefabs → Portal.
Open the Portal prefab and select the newly added layer for the View object




Then go to the Portal camera and for the culling mask select every layer except the PortalScreen.




That’s it. Just start one of the included demo scenes and you are ready to explore the asset.

# Start the demo scene

Every demo scene has a simple character controller included.
Simply move around by following controls:

- W: move forward
- A: move left
- S: move back
- D: move right
- Space: Jump

For rotating the camera you have 2 options:
Per default you need to hold the right mouse button and then you can rotate around.
You can also disable this restriction to be able to directly rotate by moving your mouse, without needing to hold the right mouse button.
SImply disable the ”RodateOnlyOnRightMouse” flag on the player gameobject in your scene.

# Shaders

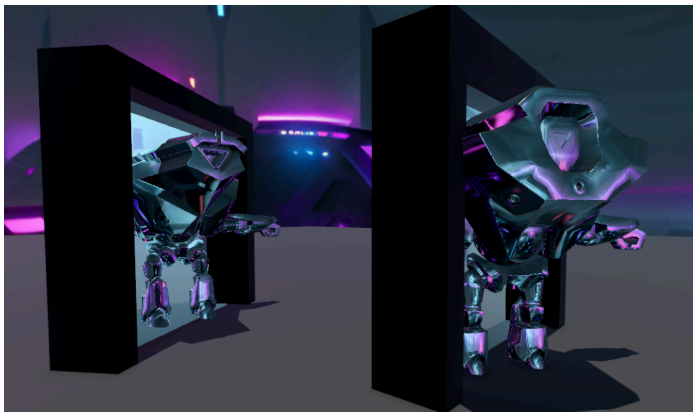
## Shader: SF Portal

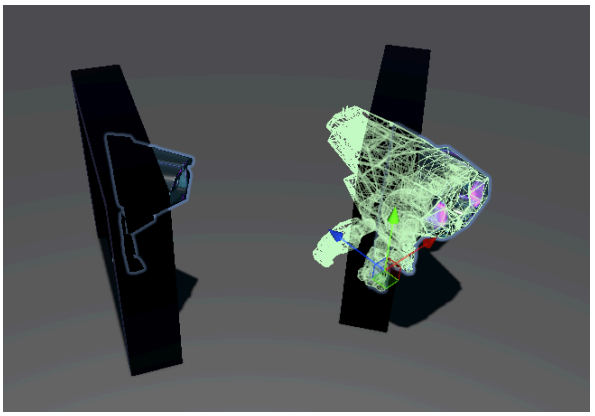
You do not need to configure anything for this shader, everything will be handled automatically. This shader is used for the Portal view and the *Portal* script will attach the according RenderTexture to the shader.

## Shader: SF Slice

Use this shader for objects that you want to be sliced when moving through the portal.
You can simply configure the basic stuff like base texture, normals, metallic map, smoothness, emission and emission strength.
All the “advanced” slicing properties will be handled directly in the Portable script

The next 2 images are showing the slicing shader in action.
The left image shows the actual game view and the right image the scene view





Basically every object with a slicing shader is sliced on the portal center. So only the part of the object that is in front of the portal is visible.
The *Portable* script will create a clone on the second portal, with inverted slicing configuration, so the hidden part of the original object will appear on the second portal.

# Portals

## Where to find the Portals?

Go to the prefabs folder and look for the Portal prefab.

## How does the portal work?

The functionality and the portal screen view is handled by the Portal script.
Every portal has one camera attached which will be responsible to capture the view of this portal. In order to be able to teleport stuff around you need to link 2 portals together.

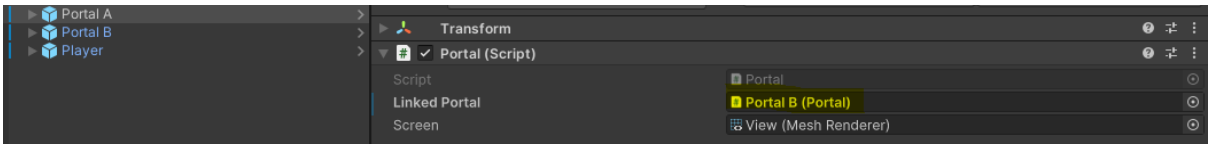
On the player camera there is a MainCamera script attached.
This script is responsible for setting the player's camera reference to all portals so the corresponding portal cameras can be updated properly. This script is also responsible for controlling when the portal views are rendered. If you are planning to integrate the assed with your own character and camera, please don't forget to add the MainCamera script to your main camera, in order to ensure a proper portalView

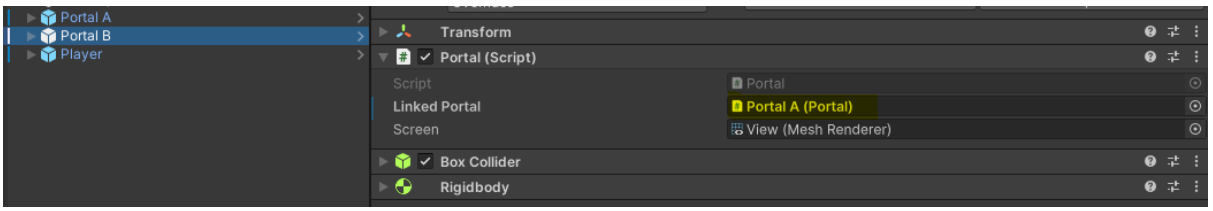
## How to set up a new portal scene?

- Create a new scene and add a simple plane for the ground.
    - E.g. scale it to 10x10x10

- Delete the default camera and drag the player prefab into the scene
    - The player has a camera attached so we do not need the default one

- Drag 2 Portal prefabs into the scene
    - Click on the first portal and drag the second one to the linkedPortal property
    - Do it also for the second one. Click on second one and link it with the first one




- Position the portals and the player as you wish, and also add some other scene objects if you want to make it more visually appealing

- You are ready to go.

## Teleporters: Portals without a portal view

If you do not care about the portal view, and want to have something else for the portal screen you can use the Teleporter prefab instead of the Portal prefab.
The setup for the teleporters works the same as for the portal. Instead of linking two portals you just link two teleporters together. The only difference is that teleporters do not have a camera attached and therefore won't render anything as a portal view. It will just stay a black screen, but you can replace the screen with whatever you want.

Here you have an example of portals and teleporters in the same scene:

On the left side in the image is a portal prefab, you can see how the objects go through the portal. The teleporter on the top has no camera to render the screen, therefore it will only show black. Objects will just fade into the black screen but the teleport functionality is the same in both cases.Slicing also works in both cases.

In general teleporters are more performance efficient than portals, since we do not have additional cameras and do not need to render anything on the portal screens. Depending on your use case you may use only teleporters, portals or a combination of both.

# Portables

## What are Portables?

Portables are gameObjects that have a *Portable* script or any script that inherits from *Portable* attached. Portables work tightly together with the *Portal* script.
They provide a Teleport function and handle entering and exiting the portal area.

You have following options of Portables:

| | |
|---|---|
| Portable.cs | Stores the basic teleport functionality |
| PlayerController | PlayerController with simple move, rotate and teleport mechanism. |
| PlayerTeleporter.cs | Inherits from Portable.cs and handles adapting position and rotation on teleport, as well as syncing the Physics properties. Can be used to attach to your own Character controller to enable the teleport feature |
| PortableRigidBody.cs | Also inherited from Portable.cs. Use this script for objects with rigidbodies |

## How do Portables work?

Portables are basically responsible to provide some basic helper functionality for teleporting, entering portal area and exiting the portal area, which will be called by the *Portal* script
They are responsible for creating the clone when entering the portal area and keep track of all the materials.

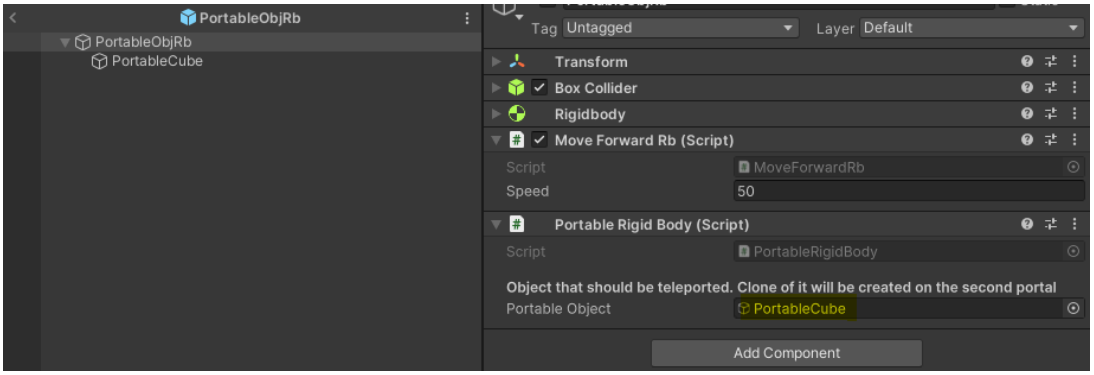
## How to create your own Portables?

You have a PortableObjRb prefab in the prefabs folder, which is a portable object that simply moves forward with a defined speed.

For creating your own portable, follow these steps:

- Create a empty gameObject as the PortableHolder

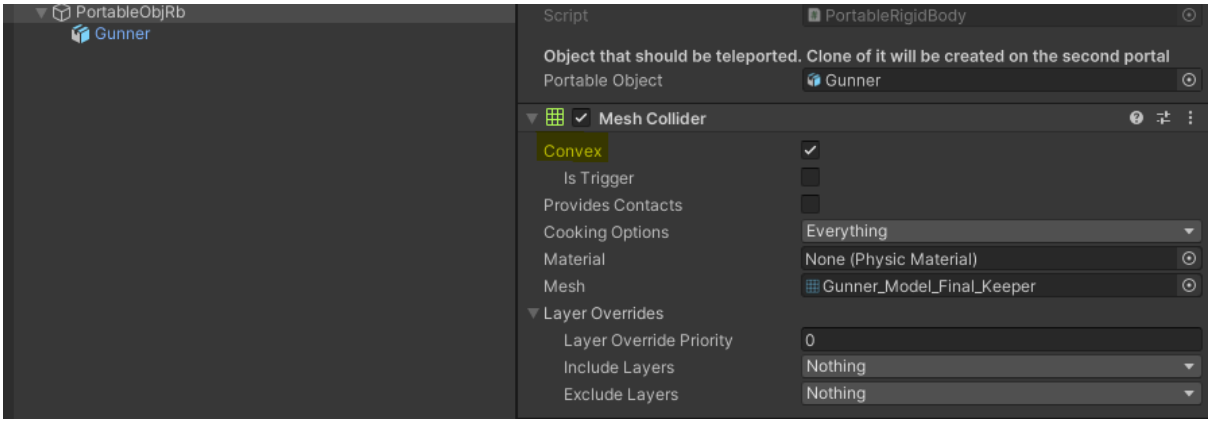
- If you want a rigid body object then attach the *PortableRigidBody.cs* to it (The rigidBody should be added automatically)
  Otherwise use the *Portable.cs* script




- Add your model as a child object and drag it to the scripts portable object property

- If you want proper slicing, make sure to use materials with the SF Slicing shader for your model. There is a *Sliced Obj* material in this asset, that you can use if you do not want to create a new one.

- Add a Mesh Collider or any other collider that fits your model
  *Note: Dynamic rigidbodies (Is Kinematic set to false) in combination with Mesh Collider require you to set the Convex option on the Mesh Collider to true.*



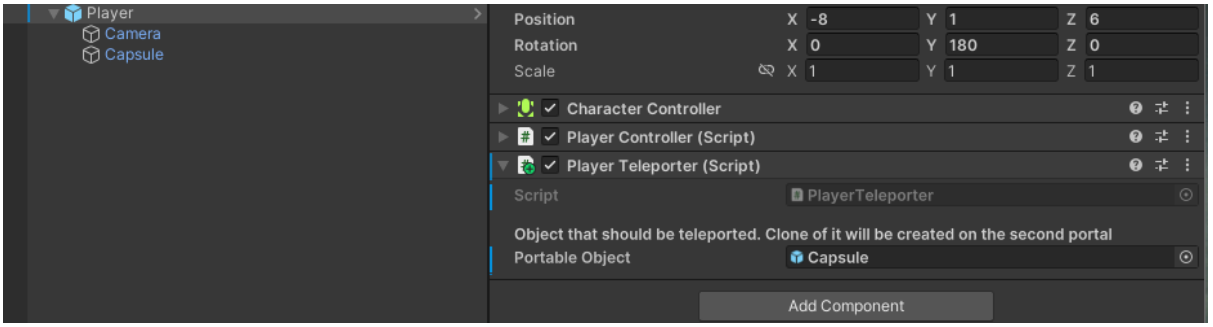

Now your portable object is finished and you should be able to teleport it through the portals.

# Integrate teleport functionality in your character controller

In each included demo scene you can find a Player gameobject, which already has the teleport functionality built in. But since you probably have an own controller and do not want to use the one of mine, here is a separate *PlayerTeleporter* script that isolates the player teleport functionality and makes the integration with your own character much easier.

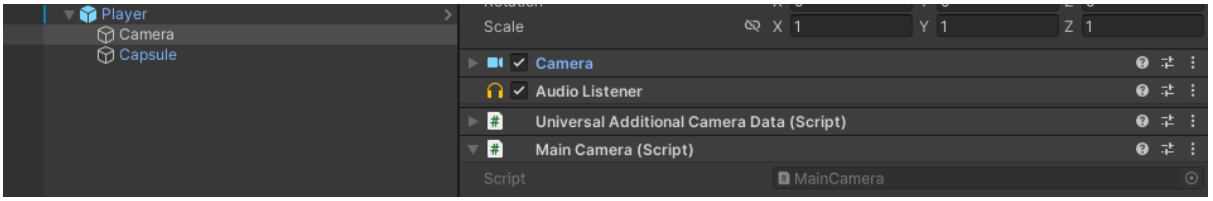
Let's assume you have the same structure as my controller:
Player (Root object with your player/CharacterController)

- Camera
- Capsule




To add the teleport feature to your character follow these steps:

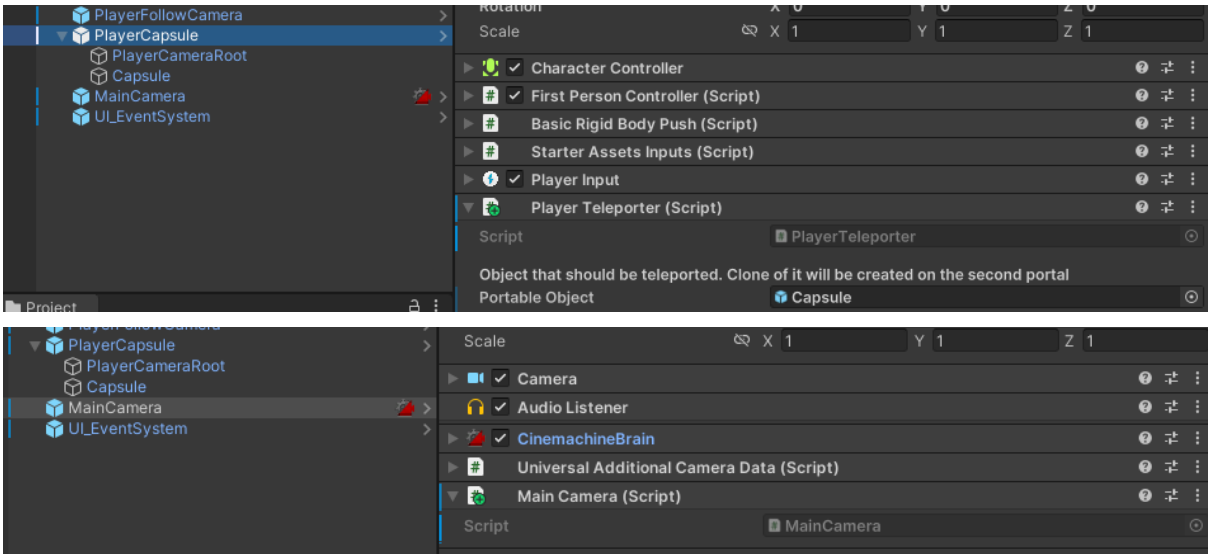
- Add PlayerTeleporter to your root object
- Assign the portable object (Object that contains the Character mesh to be cloned)
    - In our case this is the Capsule
- Go to your player camera and attach the MainCamera script to it. This will make sure that the portal views are updated and rendered properly.




And that's basically it, you have now added the teleport functionality to your own Character.

## Example: Teleport integration with Starter Assets First Person Controller

Since each Character Controller could have different structures, here you have an example of a different structure where the Player and the main camera are 2 different root objects. For this example I used the Free Package: [Starter Assets: Character Controllers | URP]

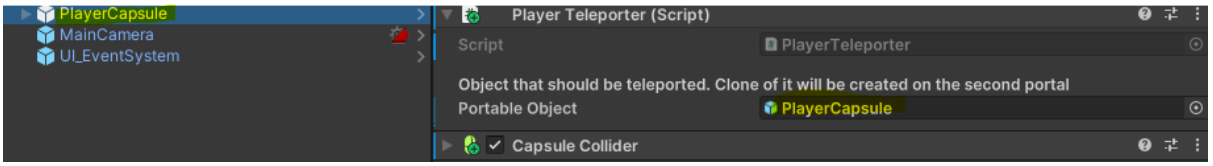



As you can see, the PlayerCapsule and MainCamera are separated, but the principle stays the same:

- Add *PlayerTeleporter* to your Player root object
- Add *MainCamera* script to your player camera

The only thing you want to make sure is that your CharacterController and your cloned mesh are not in the same GameObject. You want to have your mesh separated since otherwise cloning the portable object would clone the character controller too, and not only the character mesh.

Here is an example of what you should NOT do:




Since the controller and the mesh reside in the same object (which is also used as portable object), going through the portal will create a clone of the full PlayerCapsule object, which would lead into 2 simultaneously active PlayerControllers.

---

Hope you enjoy this asset!
For any questions/ complaints/suggestions contact me under: ***gamedevibk@gmail.com***