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

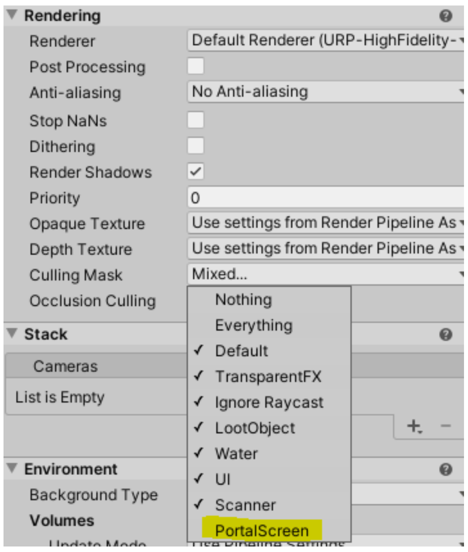
Perfect portal camera alignment
Through mechanisms like perspective division and oblique projection, the portal's cameras adjust based on the player's camera, creating the most realistic portal view possible. This view mechanism is the base of the seamless teleportation, since it exactly shows the future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



Rendering

Renderer	Default Renderer (URP-HighFidelity-
Post Processing	☐
Anti-aliasing	No Anti-aliasing
Stop NaNs	☐
Dithering	☐
Render Shadows	✓
Priority	0
Opaque Texture	Use settings from Render Pipeline As
Depth Texture	Use settings from Render Pipeline As
Culling Mask	Mixed...
Occlusion Culling	

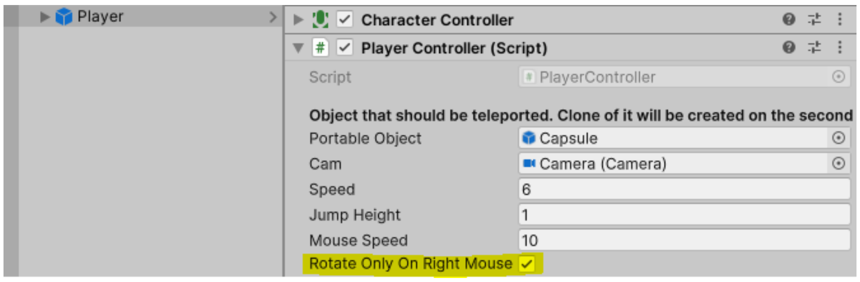
That's it. Just start one of the included demo scenes and you are ready to explore the asset.

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 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.



Script	PlayerController

Object that should be teleported. Clone of it will be created on the second

Portable Object	Capsule
Cam	Camera (Camera)
Speed	6
Jump Height	1
Mouse Speed	10
Rotate Only On Right Mouse	✓

Shaders

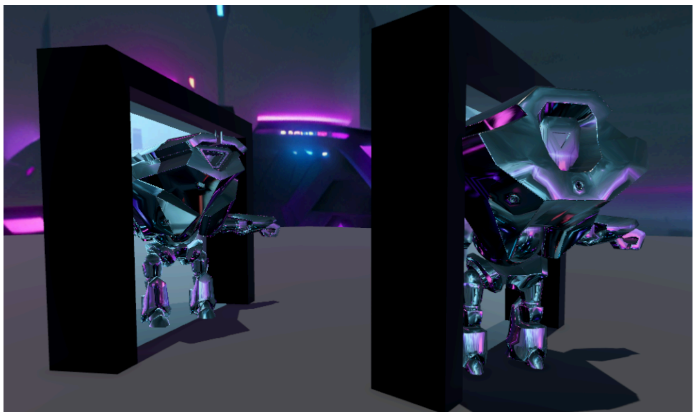
Shader: SF Portal

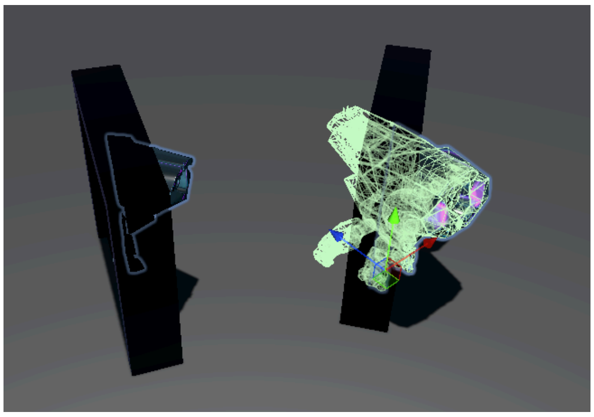
You do not need to configure anything for this shader, everything will be handled automatically. This shader is used for the Portal view and the *Portal* script will attach the according RenderTexture to the shader.

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.

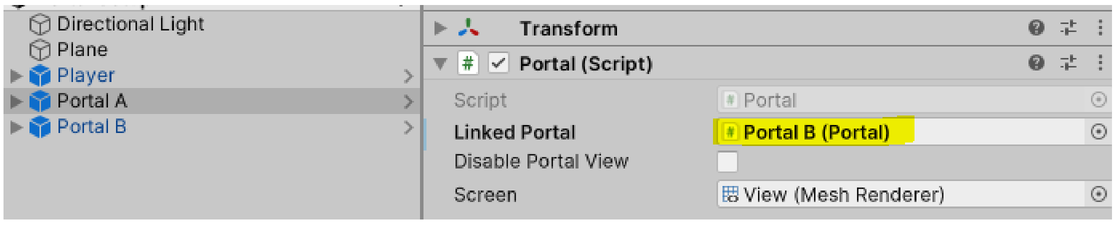
On the player camera there is a MainCamera script attached.
This script is responsible for setting the player's camera reference to all portals so the corresponding portal cameras can be updated properly. This script is also responsible for controlling when the portal views are rendered. If you are planning 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?

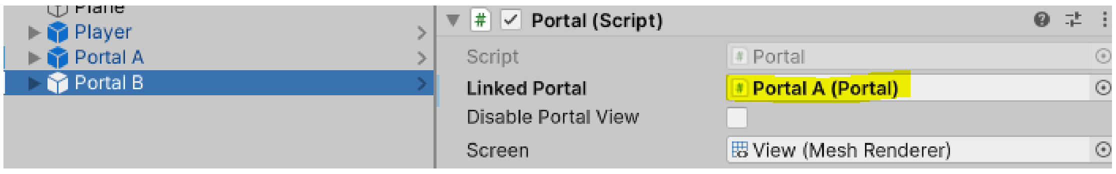
- Create a new scene and add a simple plane for the ground.
    - E.g. scale it to 10x10x10

- 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



Script: Portal
Linked Portal: Portal B (Portal)
Disable Portal View: ☐
Screen: View (Mesh Renderer)



Script: Portal
Linked Portal: Portal A (Portal)
Disable Portal View: ☐
Screen: View (Mesh Renderer)

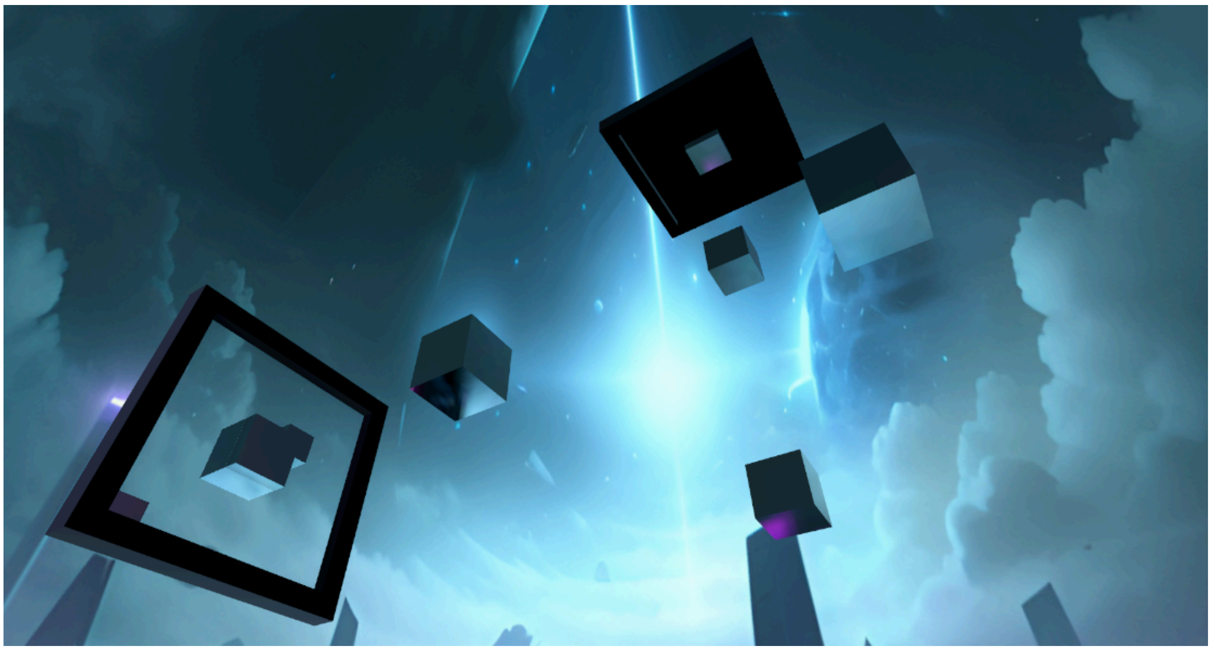
- Position the portals and the player as you wish, and also add some other scene objects if you want to make it more visually appealing

- 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	Simple script with basic functionality
PlayerController.cs	Inherits from Portable.cs and handles adapting velocity, position and rotation on teleport.
PortableRigidBody.cs	Also inherits 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.

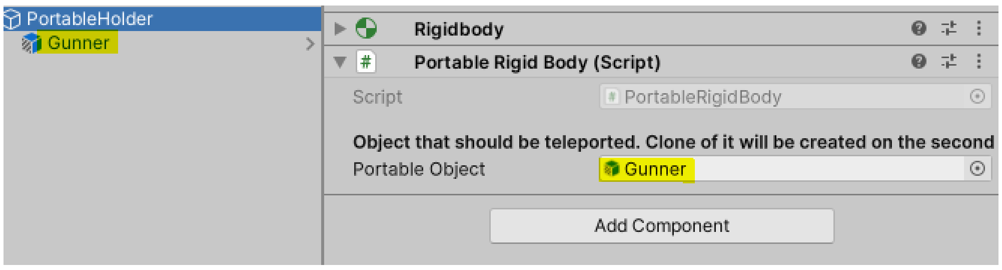
How to create your own Portables?

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



Script: PortableRigidBody

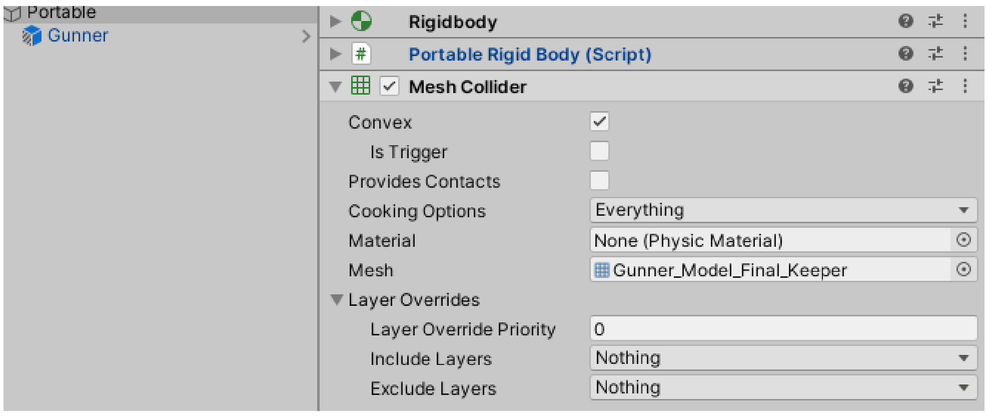
Object that should be teleported. Clone of it will be created on the second

Portable Object: Gunner

- 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.



Convex	✓
Is Trigger	
Provides Contacts	
Cooking Options	Everything
Material	None (Physic Material)
Mesh	Gunner_Model_Final_Keeper
Layer Overrides	
Layer Override Priority	0
Include Layers	Nothing
Exclude Layers	Nothing

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

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