Mirrors and Reflections

Built for VR but perfect for non-VR Desktop and Mobile projects as well

Last update: Oktober 14, 2023 - Version 2.0.0

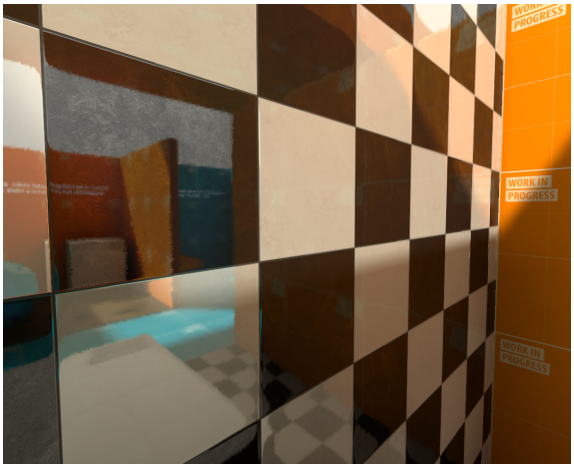
If you're looking at the pdf bundled with the asset, the documentation might have already been updated. You can find the most recent documentation here:

https://docs.google.com/document/d/1mED5MoYx9XgSbKIRe5Wz_zBv9itWDnEae0U8t1RoOCE/edit?usp=sharing

Warning!

Rendering reflections is taxing on mobile hardware. Don't expect 4 fully recursive mirrors all reflecting each other 3 recursions deep at full screen resolution to hit framerate on an Oculus Quest 1. We're essentially rendering your scene multiple times over depending on how deep you want the reflections to go.

That being said this is most likely the fastest solution to do real-time planar reflections on mobile VR hardware in existence.



Compatibility

Are you working with the **URP rendering pipeline?** Then continue reading.
Are you using the standard rendering pipeline!? Request a refund.. This is **currently** not the package for you.

"Mirrors and reflections for VR" is created trying to be a great way to render real time recursive reflections. Everything is made to look good and to be as performant as I know how to make it.

Have a look at the below table.. Find your Unity / URP combination, then check which modes are available. **If your unity version is not included but higher than the lowest version number listed here, chances are good things will work.** It just means that combination was not verified working. This table is not complete so feel free to provide testing data by emailing. Fragilem17@gmail.com

# Unity # URP	Non VR (PC and mobile games)	PC VR Single Pass Instanced	Mobile VR Multiview	Mobile & PC VR Multi Pass	Remarks
2023.01.08f1 URP 15.0.6	OK	OK	OK	OK	All Good
2022.03.11f1 URP 14.0.8	OK	OK	OK	OK	All Good
2022.03.07f1 URP 14.0.8	OK	OK	OK	OK	All Good
2022.01.05f1 URP 13.1.8	OK	OK	OK	OK	All Good
2022.01.00f1 URP 13.1.8	OK	OK	OK	OK	All Good
2021.03.31	OK	OK	OK	OK	All Good
2021.03.28	OK	OK	OK	OK	All Good
2021.03.22	OK	OK	OK	OK	All Good

2021.03.17 URP 12.1.9	OK	OK	OK	OK	All Good
2021.03.02f1	OK	OK	OK	OK	All Good, Except for Canvasses not reflecting in builds
2021.03.00f1 URP 12.1.6	OK	OK	OK	OK	All Good Except for Canvasses not reflecting in builds
2021.02.06f1 URP 12.1.2	OK	OK	OK	OK	All Good Except for Canvasses not reflecting in builds
2020.03.36f1 URP 10.9.0	OK	OK*	OK*	Fail	requires a skybox material to work in VR
2020.03.24f1 URP 10.9.0	OK	OK*	OK*	Fail	requires a skybox material to work in VR
2020.03.19f1 URP 10.6.0	OK	OK*	OK*	Fail	requires a skybox material to work in VR
2020.03.10f1 URP 10.5.1	OK	OK*	OK*	Fail	requires a skybox material to work in VR
2020.03.02f1 URP 10.4.0	OK	OK*	OK*	Fail	requires a skybox material to work in VR
2019.04.40f1 URP 7.7.1	OK	OK	OK	Fail	Use the 2019 Materials
2019.04.16f1 URP 7.3.1	OK	OK	OK	Fail	Use the 2019 Materials

***requires a skybox material to work.** **(Projection Matrices return identity Matrix when no Skybox is set.)**

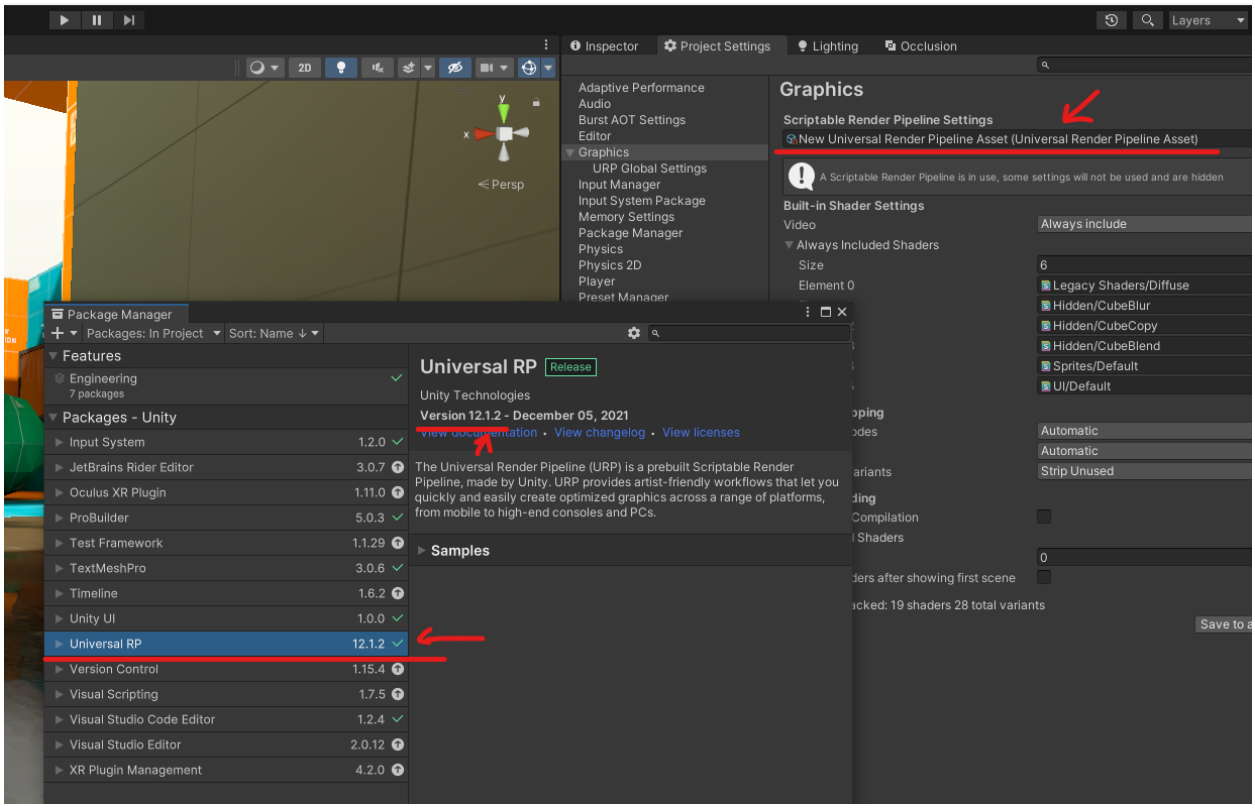
PS: make sure you switch the stereo rendering mode for desktop to your required setting even though you might be compiling for Android your development machine is using the setting for desktop during development.

Using it.

First, make sure you're using the URP rendering pipeline.
The asset was originally made using:

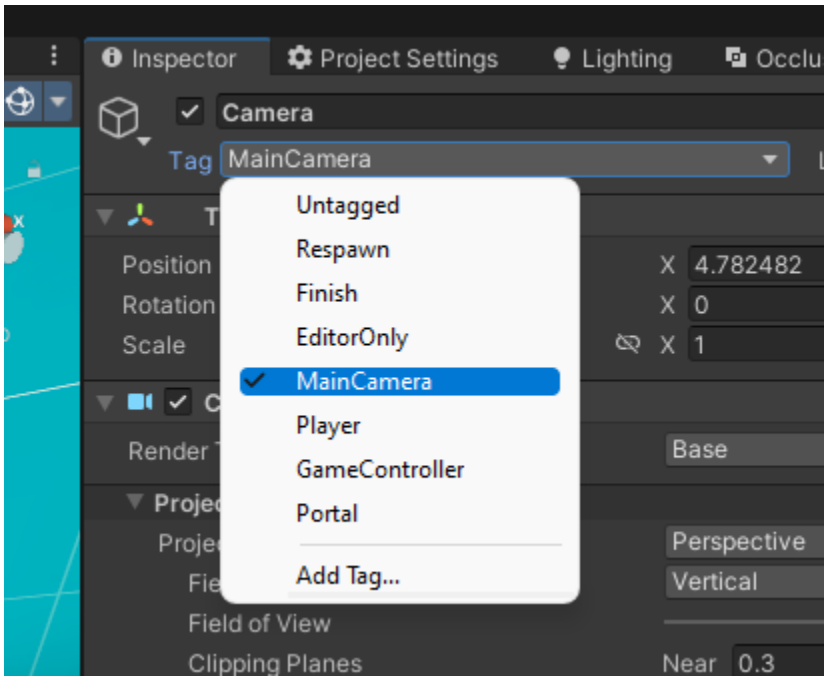
Universal RP 12.1.2
Unity 2021.2.6f1

But newer versions will of course work. Check the table above for details.



Tag your camera

Mirrors will only reflect for the scene view in edit mode and Cameras tagged as “**MainCamera**” or “**SpectatorCamera**”. So make sure your main camera has the tag “**MainCamera**”.



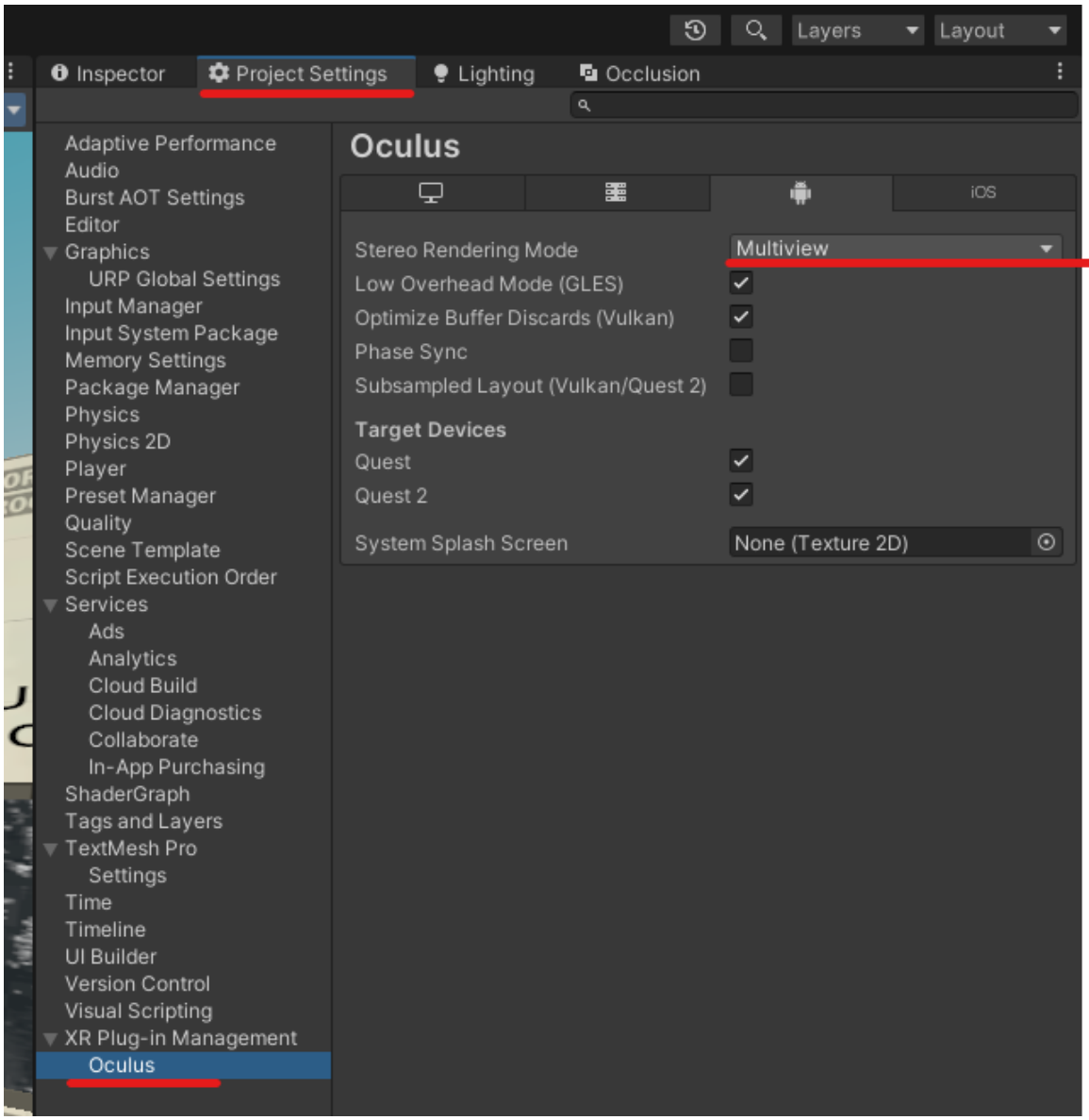
Spectator Camera

If you tag a camera as “**SpectatorCamera**” and set it to target “no” eyes then on a desktop build that camera will be used to render the view on the desktop monitor. The MainCamera will still be rendered in the VR HMD. If you make it so the Spectator Camera smoothly follows your head and has a wide FOV then people following along on the desktop monitor don’t get sick with the fast headmovements. Only when the camera has the tag “**SpectatorCamera**” will the mirrors be rendered.

For Quest or Native Mobile VR (meta):

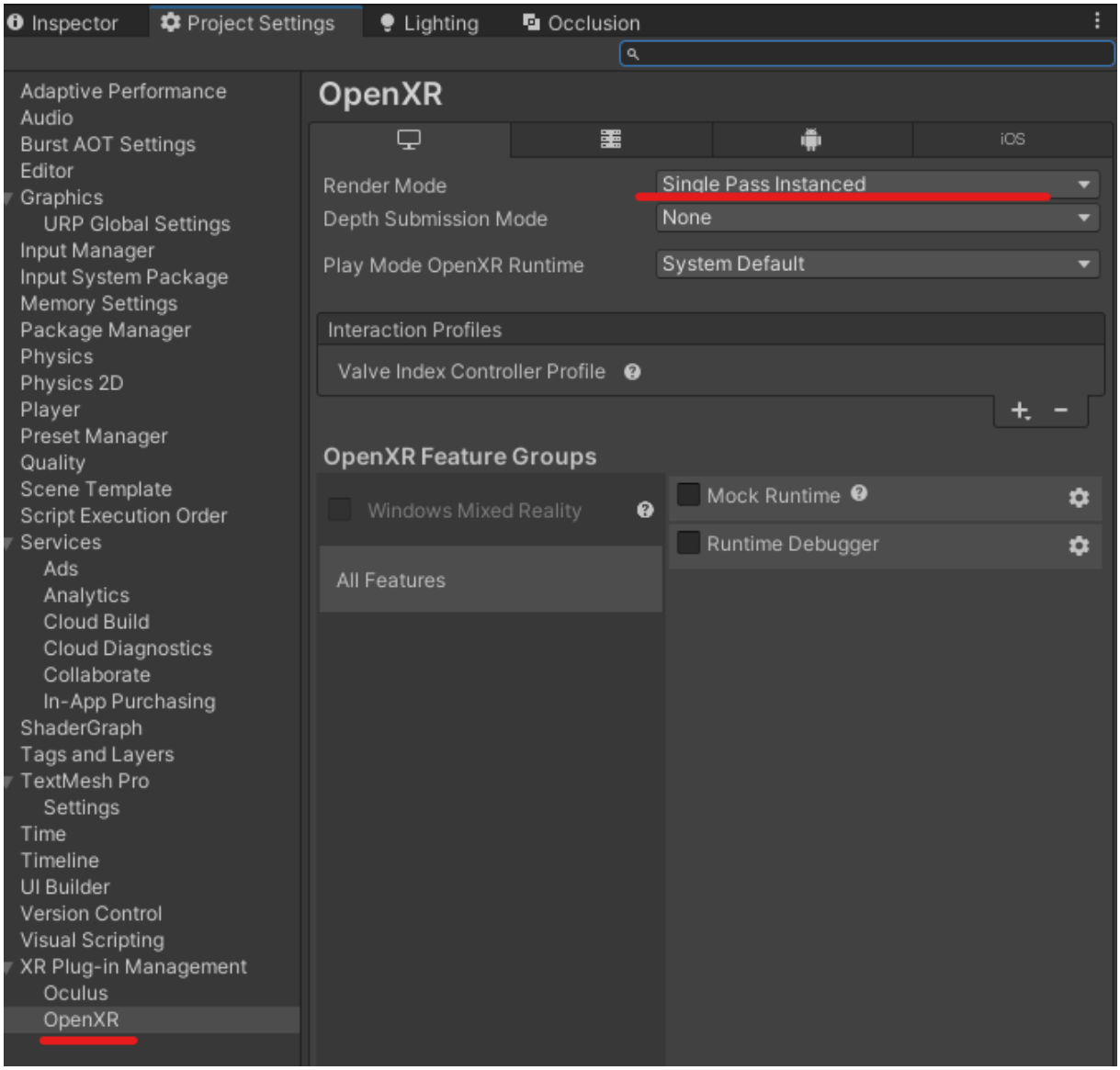
If you're using the Oculus integration, make sure you're rendering in **Multiview** mode! This was previously named **singlepass** and is significantly faster than Multipass. Even though we now **fully support Multipass** you should really look into converting to **Multiview**.

NOTE: Even though you're building for Android, your development PC/Mac is a desktop.. So **switch over to desktop, and make sure you've set the rendermode to singlepass instanced there as well.**



For Oculus Link / Valve Index / Vive / any steam VR OpenXR based headset

If you're using OpenXR then, make sure you're rendering in **Single Pass Instanced** mode. This is significantly faster than Multi Pass. Even though we now **fully support Multipass** you should really look into converting to **Single Pass Instanced**.

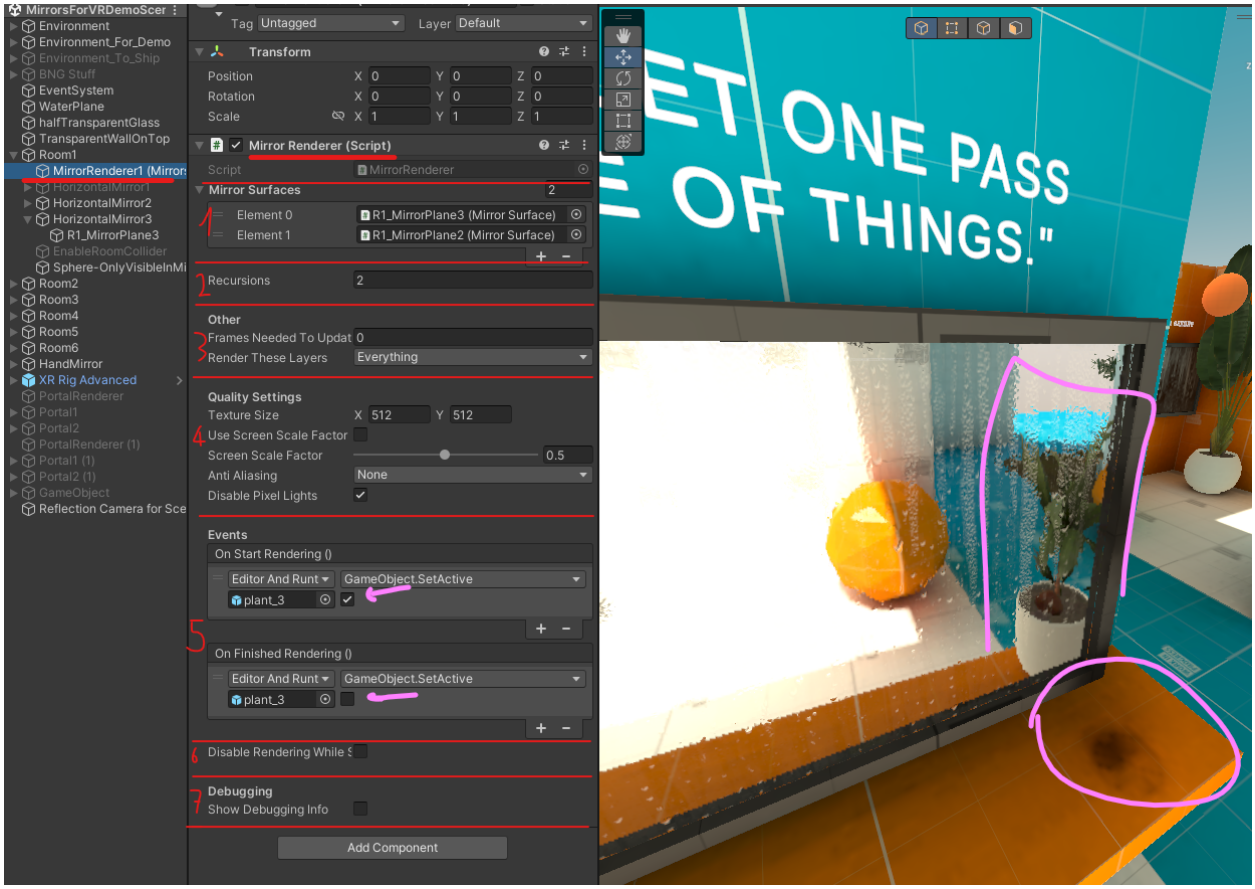


(Even though we're still stuck rendering the reflection as a pass per eye, i'm keeping an "eye" out for changes to URP to allow a rendertexture to be rendered as a double wide texture)

Open the Demo Scene

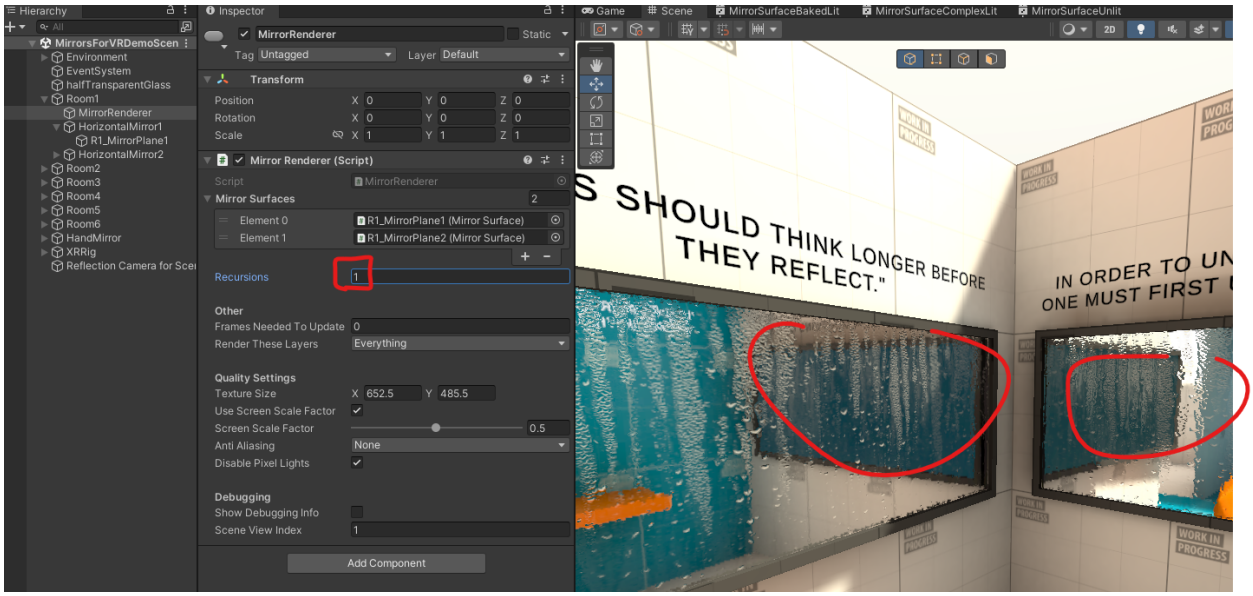
MirrorRenderer

Have a look at Room 1, Select the Gameobject “**MirrorRenderer**”



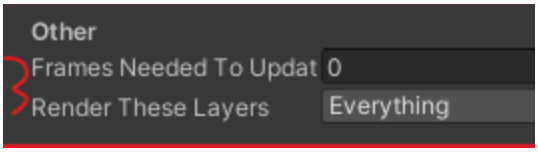
1 Here we define which surfaces are to be used to render the reflections in. All surfaces in this list will render a reflection and depending on the recursion depth will render each other's reflections too.

2 “recursions” means how many mirrors “deep” we will render.

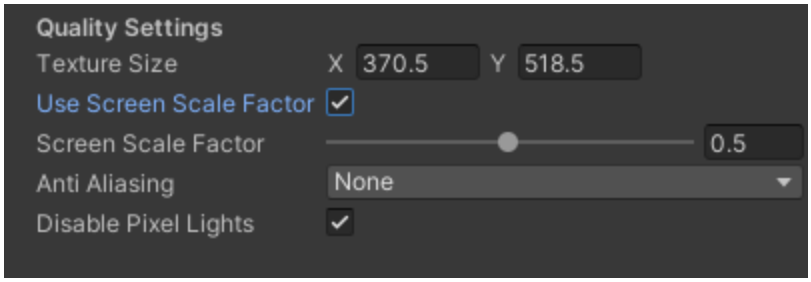


A reflection depth of 1 will only render a reflection in the first mirror, but not in the one seen inside it. Use this with caution. In most (mobile) cases a depth higher than 3 is a bad idea and will tank your performance.

3 some features to optimize for performance, render every other frame if you set this to 1 or higher.. In VR this will look bad so try to limit the things to reflect by changing the layer mask of the Reflection Camera..

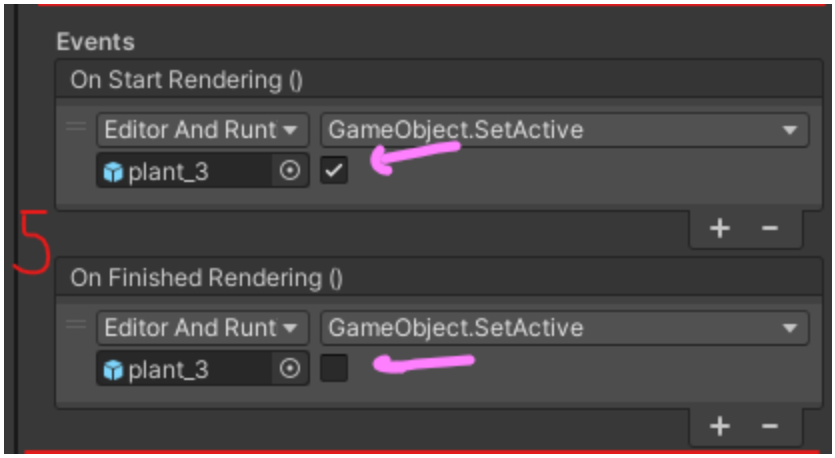


4 “Use Screen Scale Factor”, when enabled the texture size is calculated as a factor of the screensize. 0.5 is a good default value for a clean mirror, a nice normal map that blurs and refracts things allows you to go lower. Also, go easy on the AA, none or low are generally good enough if you combine it with a good normal map. “Disable pixel lights” will turn off the



realtime lights for the reflection cameras. On Quest I Feel like square fixed multiples of 16 are faster.

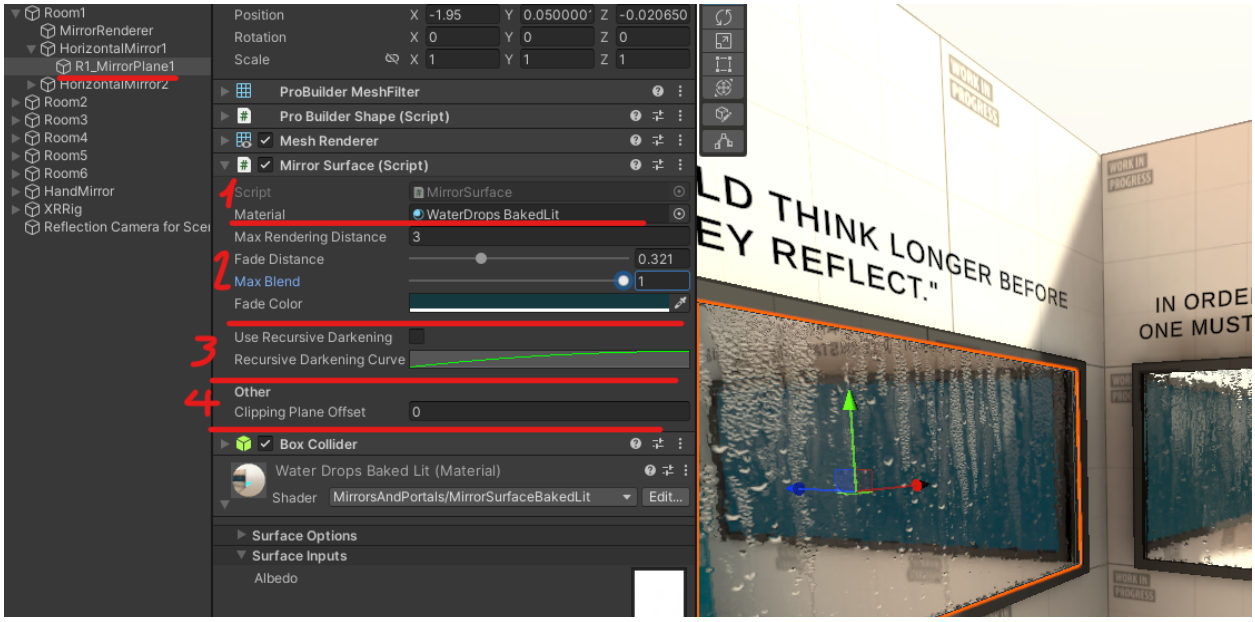
5 These 2 events allow you to change the scene just before the reflection gets rendered and then change it back. Use it for example to scale your head down in the VR camera and scale it up for the reflection camera. Can be used as an alternative for Culling masks.



6 “Disable Rendering While Still Updating Materials” When checked, the reflection will stop rendering but the materials will still update their position and blending.

7 “Show Debugging info” will start drawing lines of the reflections camera's view direction.

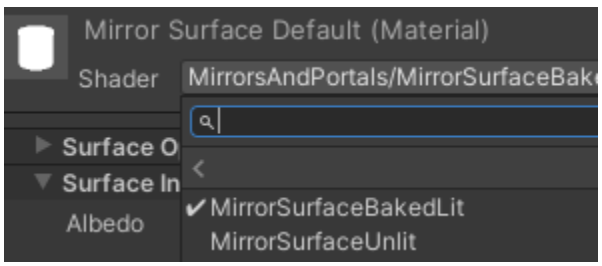
MirrorSurface

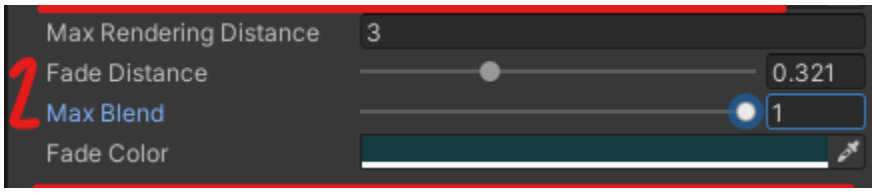


(continue reading if you're looking for the new "My Forward Transform" and "Child Surfaces")

1 Select the GameObject "R1_MirrorPlane", this is one of the surfaces in the above MirrorRenderer. A surface needs a Material. Start by making a material and dragging it into the slot.

There are 2 shaders: a real simple unlit shader for simple mirrors and a bakedLit advanced shader that allows you to do refractions and other nice effects.





2 "Max Rendering Distance" defines at what distance from the nearest corner of the bounding box of your mirror the reflection stops rendering. As you move further away from the mirror the mirror will change color towards the "Fade Color".

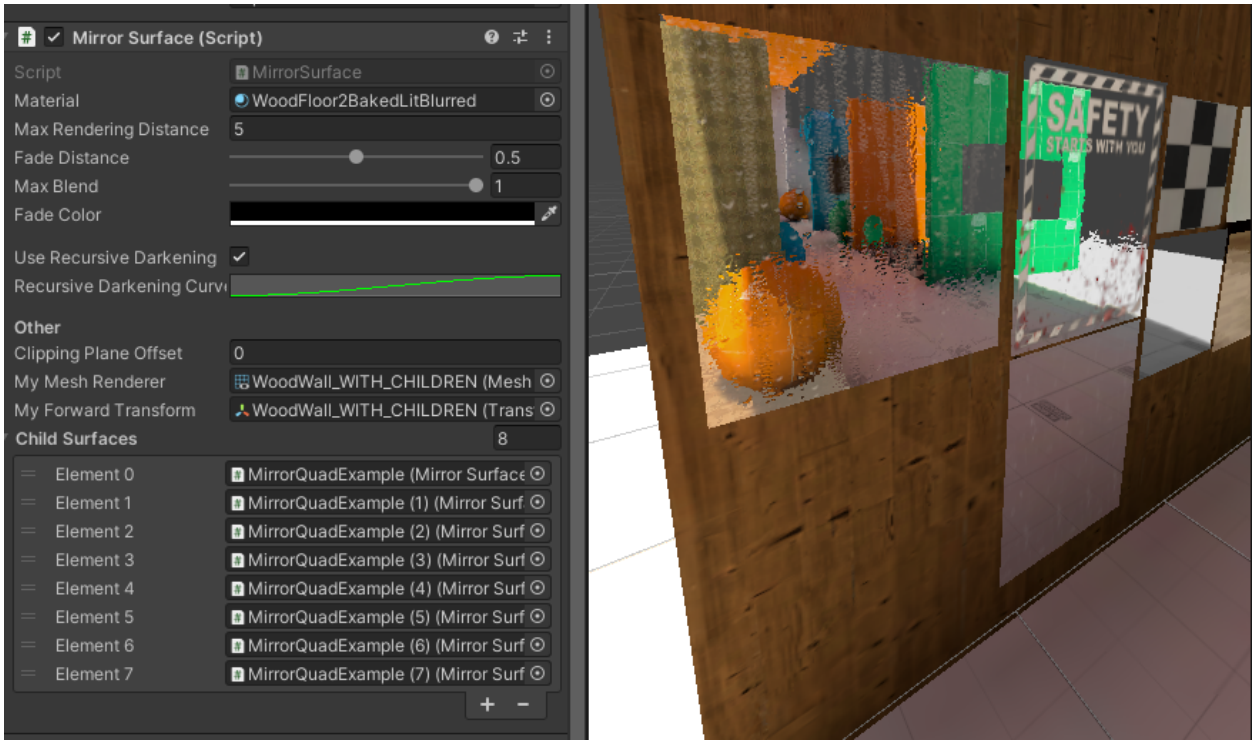
"Fade Distance" controls over what percentage of the max rendering distance the fade occurs. (0.5 means it will start blending at half the max distance away from the mirror, and be completely blended at the max distance)

"Max Blend" allows you to always have a percentage of "Fade Color" in your mirror. A cheap and nice way to make your mirror a bit darker for example.

3 "recursive darkening" if you have really deep mirrors (recursions of 3 or more) you can control how fast the mirror goes to the blend color besides the max rendering distance. This only has limited functionality on mobile as you most likely won't be going over more than 3 recursions.

When disabled the darkening will happen based on the fade distance. Using low distances will result in the recursion stopping early.

4 The "clipping plane offset" allows you to move the reflection plane forward or backward from the actual mirror plane.



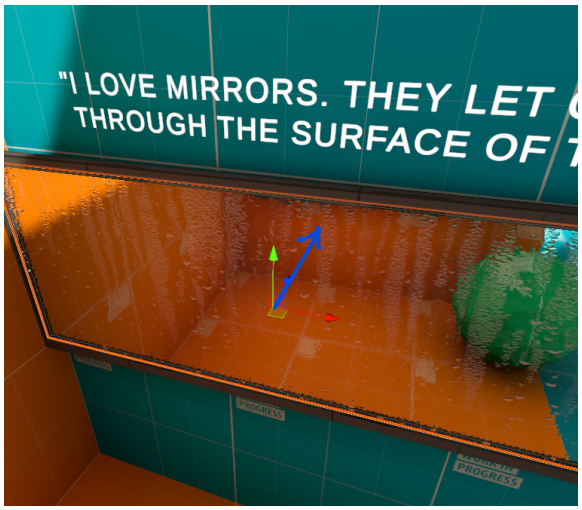
“My Forward Transform” (v1.1.0 and higher) you can set a custom forward transform, meshes that do not have their forward (blue Z arrow) in the correct rotation can use a child transform that defines the way the mirror will reflect.

“Child Surfaces” is a list of MirrorSurfaces. Instead of adding them to a MirrorRenderer you can add them to a MirrorSurface that is already in a MirrorRenderer itself. The materials of the childs will get the same reflection textures. This way you only incur the rendering cost once for all surfaces. Best used on surfaces in the same plane.

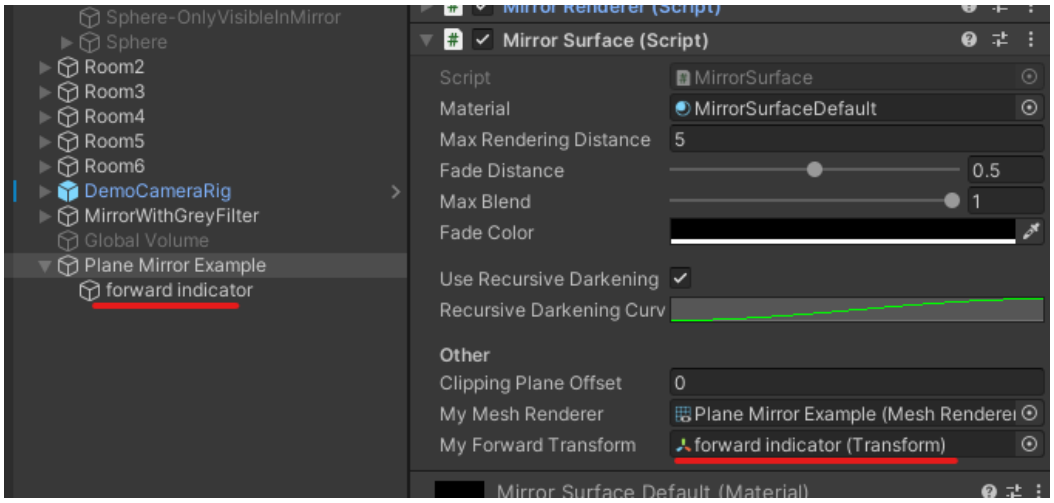
Creating your own mirror

Forward direction

The reflection will be calculated looking back from the blue arrow of the model. So make sure to be looking at your local coordinates and that the blue arrow is looking away from the visible side of the plane. This is the default for the Unity 3D Plane.

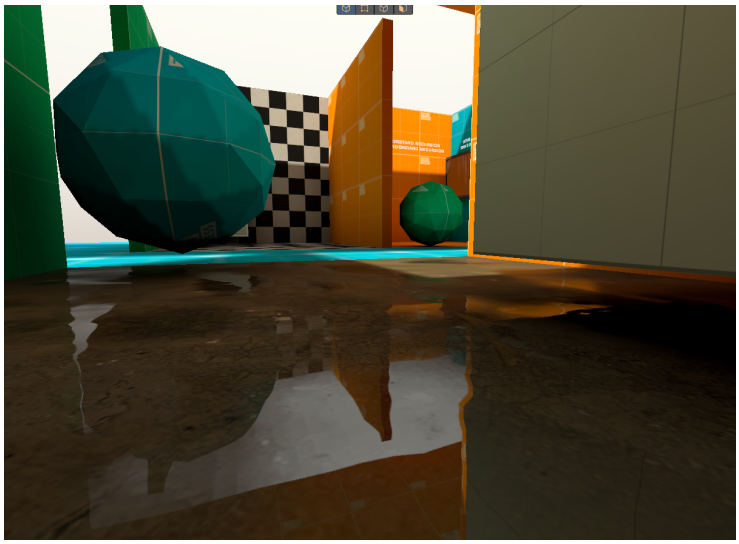


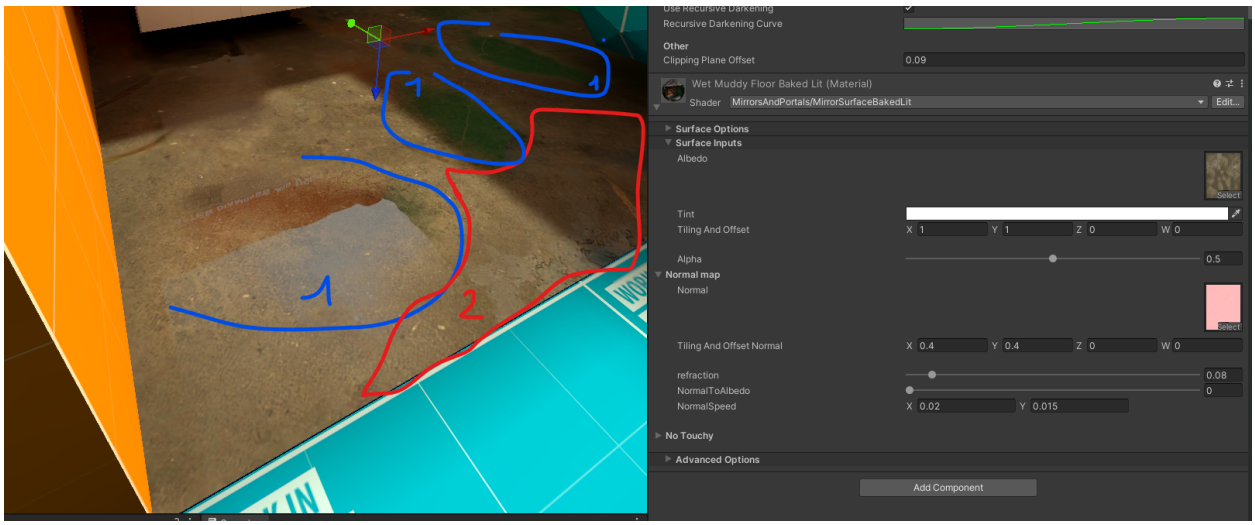
If your model does not have its pivotpoint set correctly you can create an empty Transform. Rotate and position that transform correctly with its blue arrow pointing away from the surface. Then drag the transform in the “My Forward Transform” property.



The Shaders

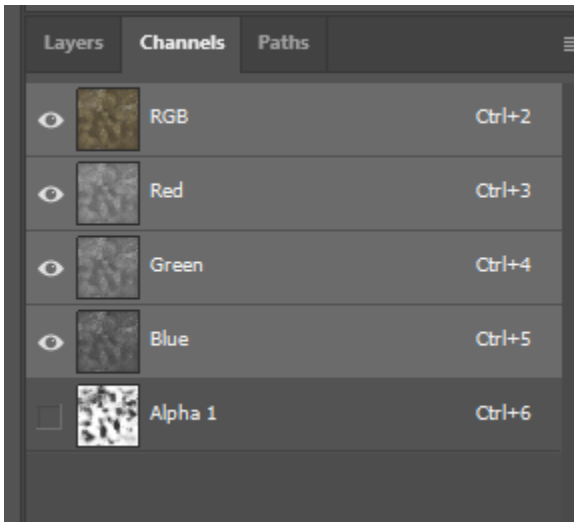
MirrorSurfaceBakedLit





Notice how the water moves and some parts are sand (1) and others (2) are reflecting water?

You control which parts show the albedo texture and which part is reflection by adding an alpha mask in your texture. The easiest way I know of is by using Photoshop and adding an alpha mask there. The specular property controls how much of the alpha mask is used for reflection



(https://www.instructables.com/How-to-use-Photoshop-to-Create-Textures-With-Alpha/)

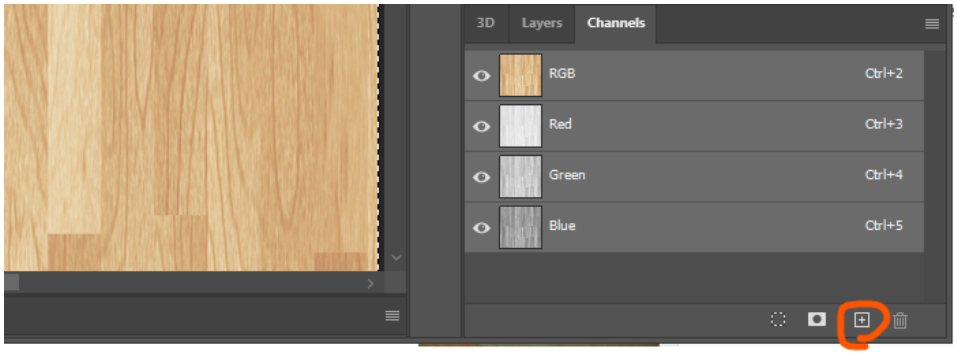
The rest of the shader is pretty self-explanatory. Don't forget to mark your normals as normal maps.

Converting a PBR texture

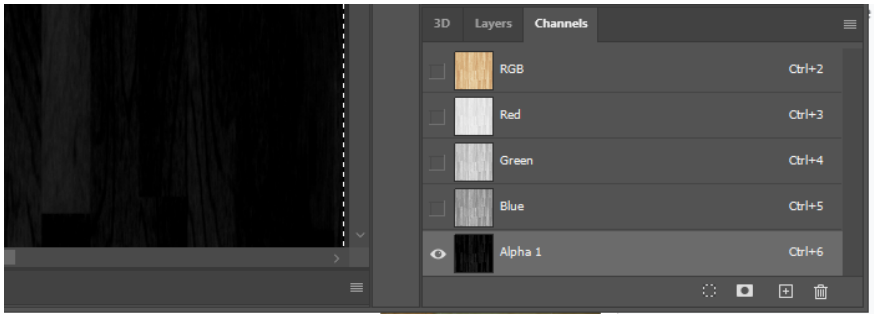
Use photoshop or any other graphics program. (using Photoshop for the tutorial below)

Open the specular texture, press Ctrl+A to select everything and press Ctrl+C to copy the texture. (You can close the texture now, we no longer need it)

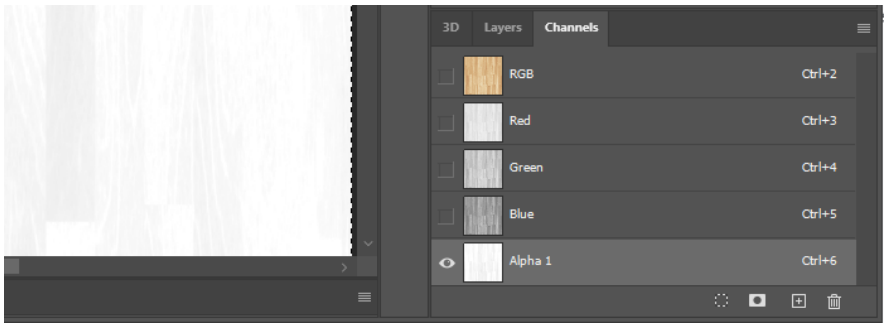
Open the Albedo texture, then go to channels and click the + icon.



Press Ctrl+V to paste our specular texture in the alpha channel.



Press Ctrl+I to invert the values.



Now save the PSD, you can directly use it as the texture in Unity.

Thanks!

If you have any questions, don’t hesitate to send me a mail at

Fragilem17@gmail.com

I hope you create beautiful things with this tool and don’t forget. Framerate is life! Kill your darlings and if you can’t hit framerate it’s better to not use it.

I’ve spent an enormous amount of personal time creating this so thanks for your support in buying this asset! It is very much appreciated.

Tom