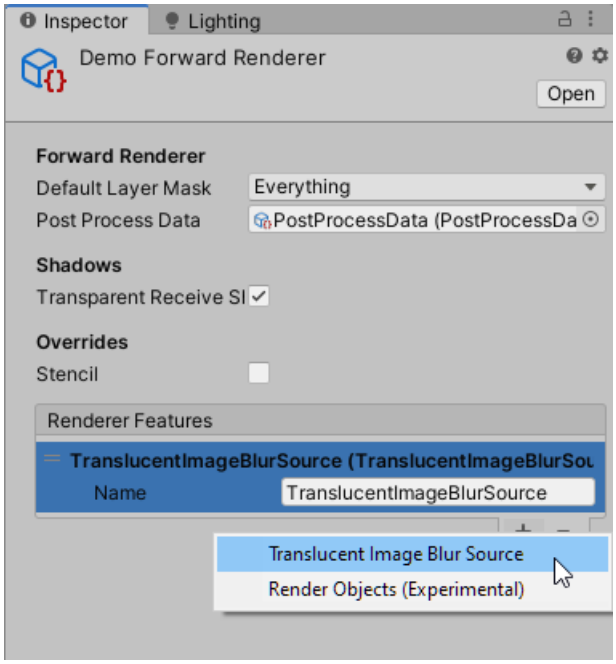
# Table of Contents

# Getting Started

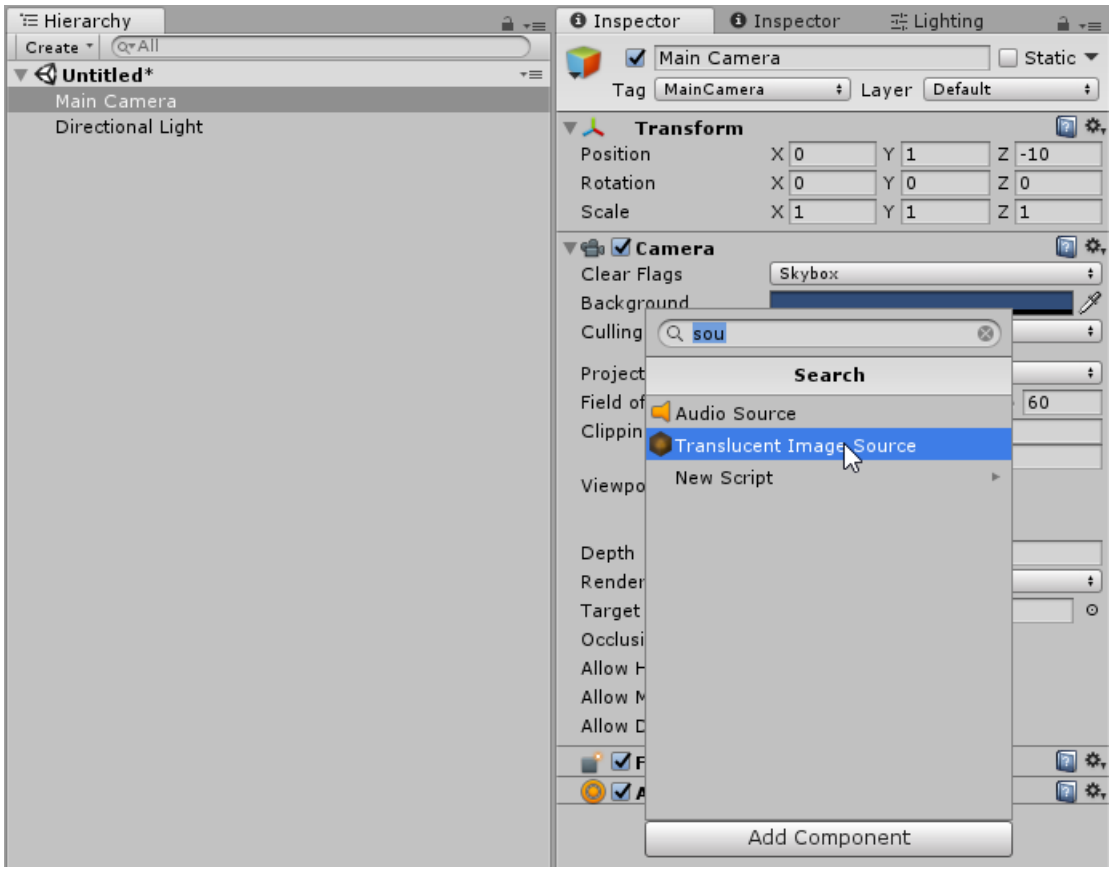
## Installation

1. Import the asset, accept API upgrade and package manager dependencies if prompted.
    - Translucent Image specifies very early versions of package manager packages to maintain compatibility with older Unity versions. Feel free to install higher versions of these packages if you want
2. If you are using URP, add the `Translucent Image Blur Source` renderer feature to your Scriptable Renderer setting asset. See the URP page for more detail.

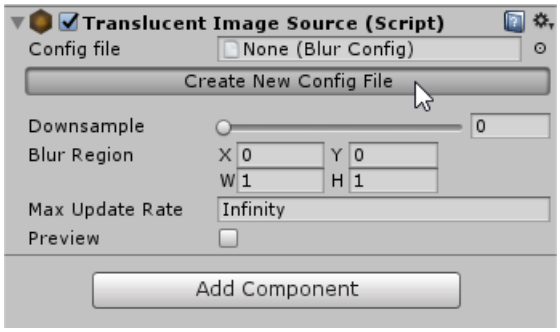



## Adding Translucent Image to your scene

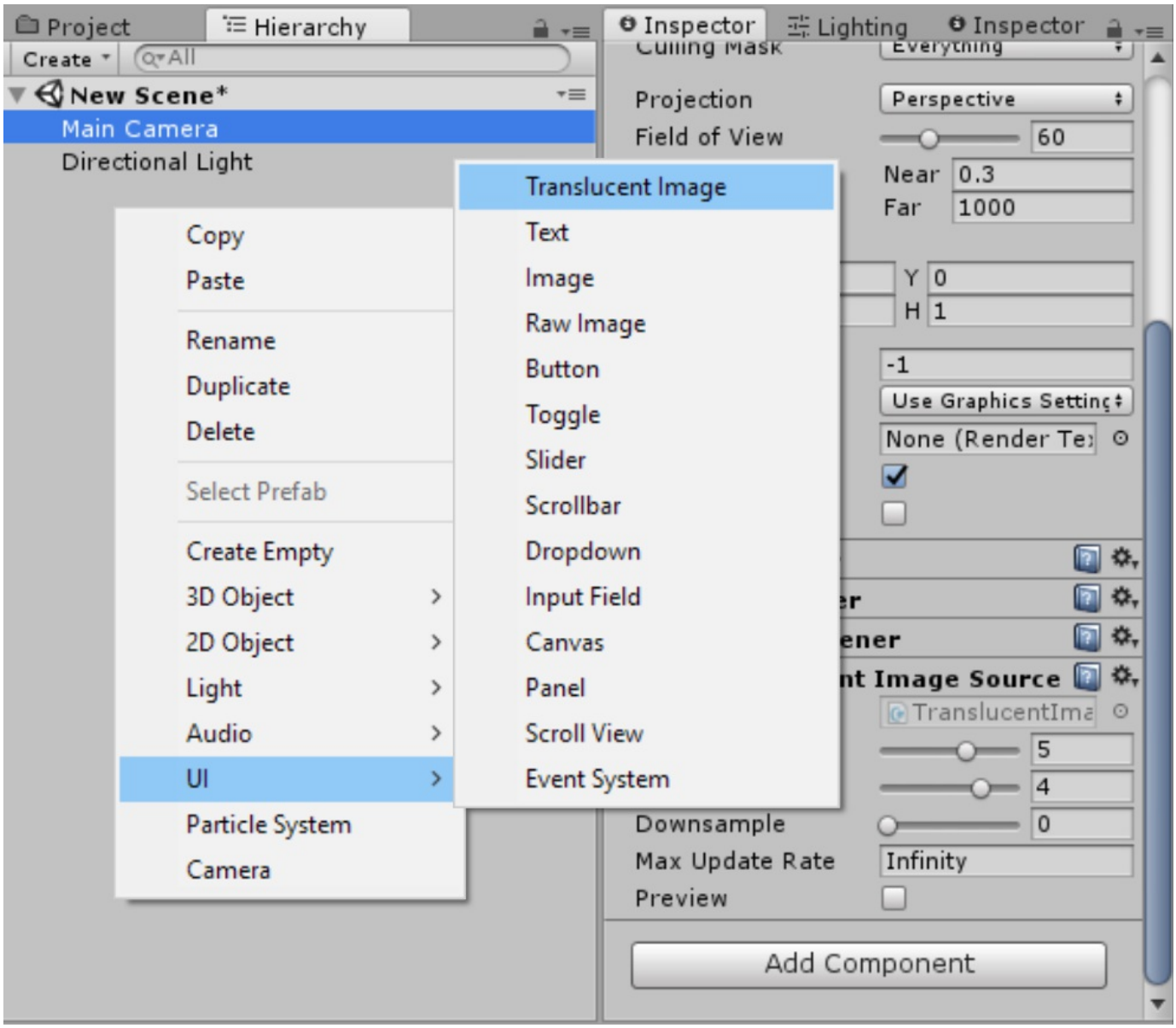
1. Add **Translucent Image Source** to your main camera.



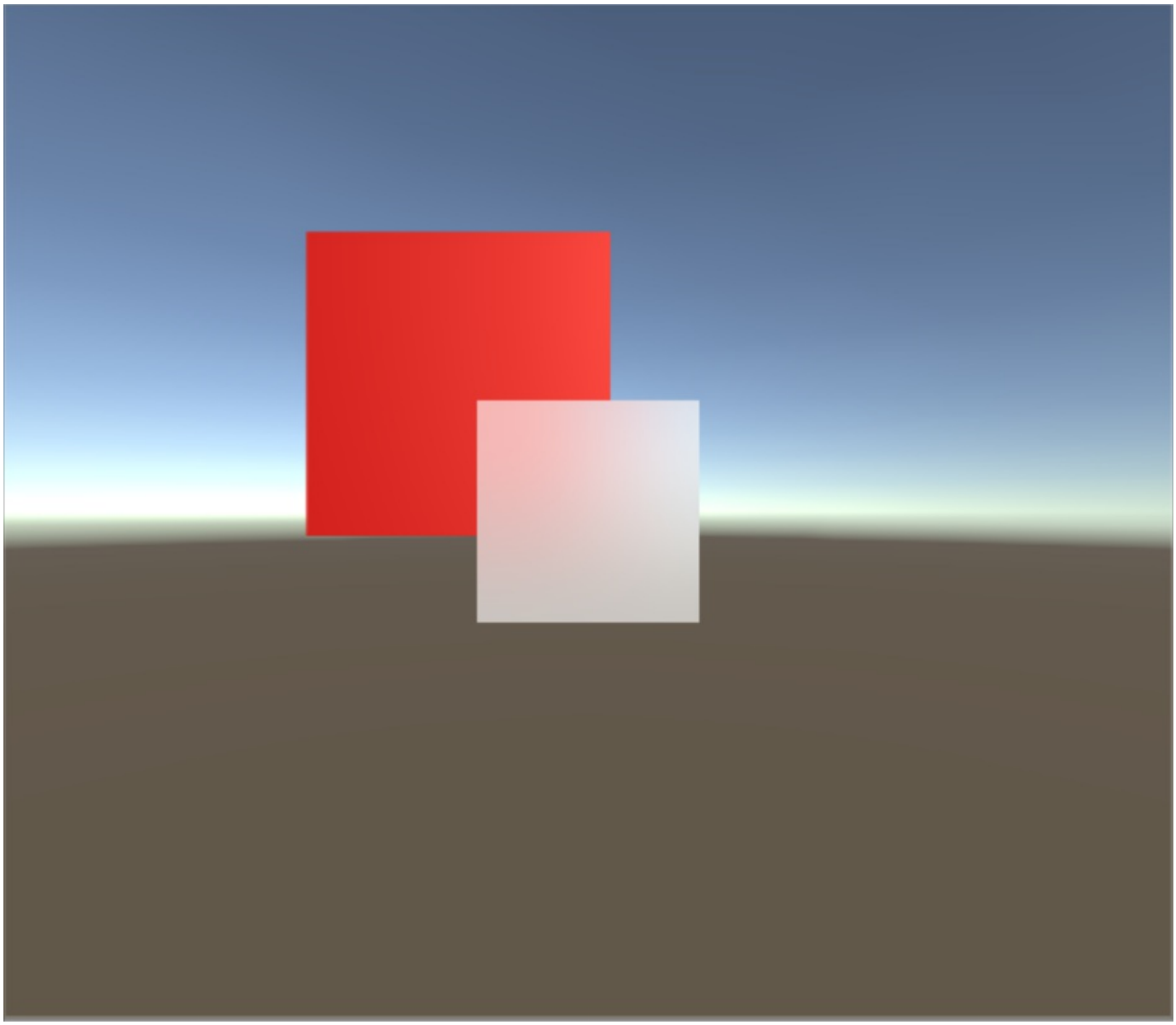

2. Create a ***Blur Config*** asset (or assign an existing one).




3. Create a ***UI > Translucent Image***, as you would with normal Image.


Project
Hierarchy
Inspector
Lighting
Inspector
Create
All
New Scene*
Main Camera
Directional Light
Culling Mask
Everything
Projection
Perspective
Field of View
60
Near
0.3
Far
1000
Y
0
H
1
-1
Use Graphics Setting
None (Render Te
Copy
Paste
Rename
Duplicate
Delete
Select Prefab
Create Empty
3D Object
2D Object
Light
Audio
UI
Particle System
Camera
Translucent Image
Text
Image
Raw Image
Button
Toggle
Slider
Scrollbar
Dropdown
Input Field
Canvas
Panel
Scroll View
Event System
ener
nt Image Source
TranslucentIma
5
4
Downsample
0
Max Update Rate
Infinity
Preview
Add Component

4. That's it!



> **⚠ WARNING**
>
> By default, Translucent Image will use a default Material. To make sure your Translucent Image are not affected by asset update, create your own Material. See Customize section for more info.

## Take a look at the demo scenes

The demo scenes are the best way to learn the asset's capabilities. Check the `Demo/00_Scenes/00_Minimal` scene for a simple reference setup. The `Demo/BiRP` and `Demo/URP` folders contain more scenes to showcase more complex setups for each render pipeline.

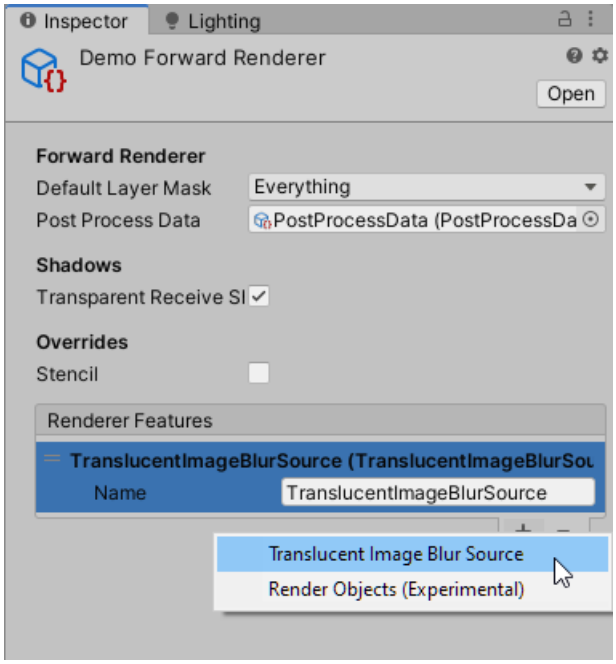
# Universal Render Pipeline

## Requirements

Only non-preview, non-beta versions of URP and Unity will be supported. The 2D renderer may break in certain configurations before 2023.3, if the built-in `Full Screen Pass Renderer Feature` doesn't work, Translucent Image is unlikely to work either.

## Setup

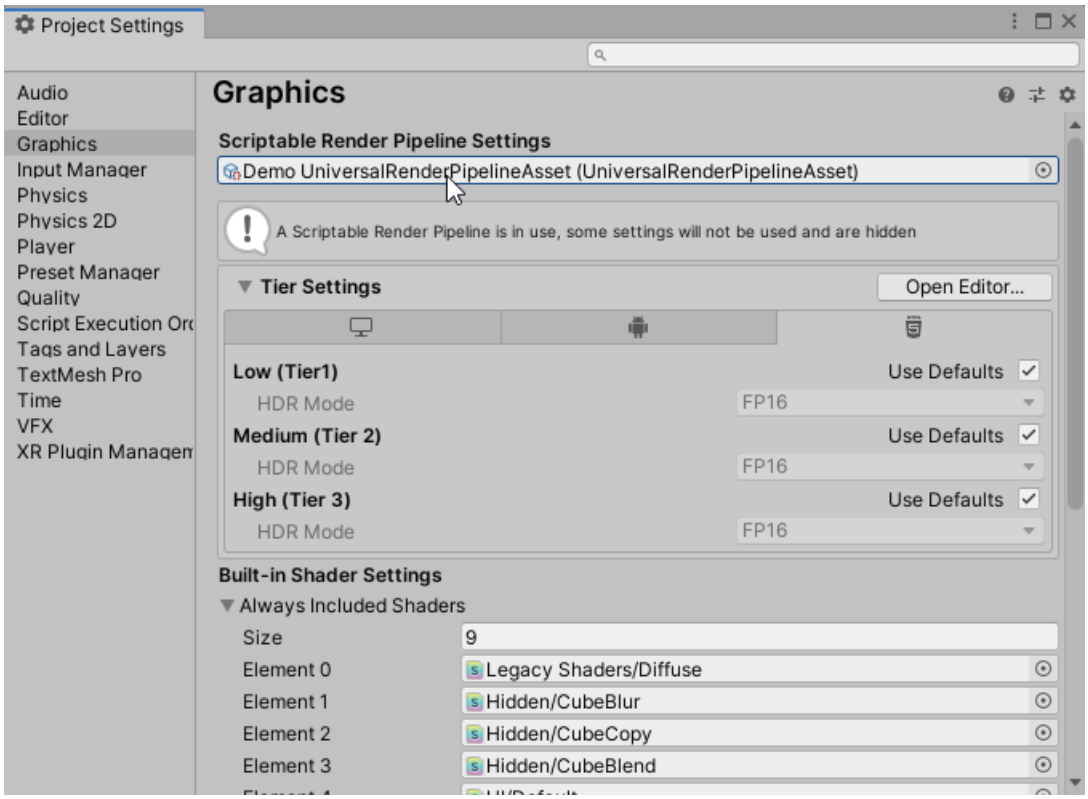
2. Find your Universal Renderer Assets and add Translucent Image Blur Source to its list of Renderer Features:



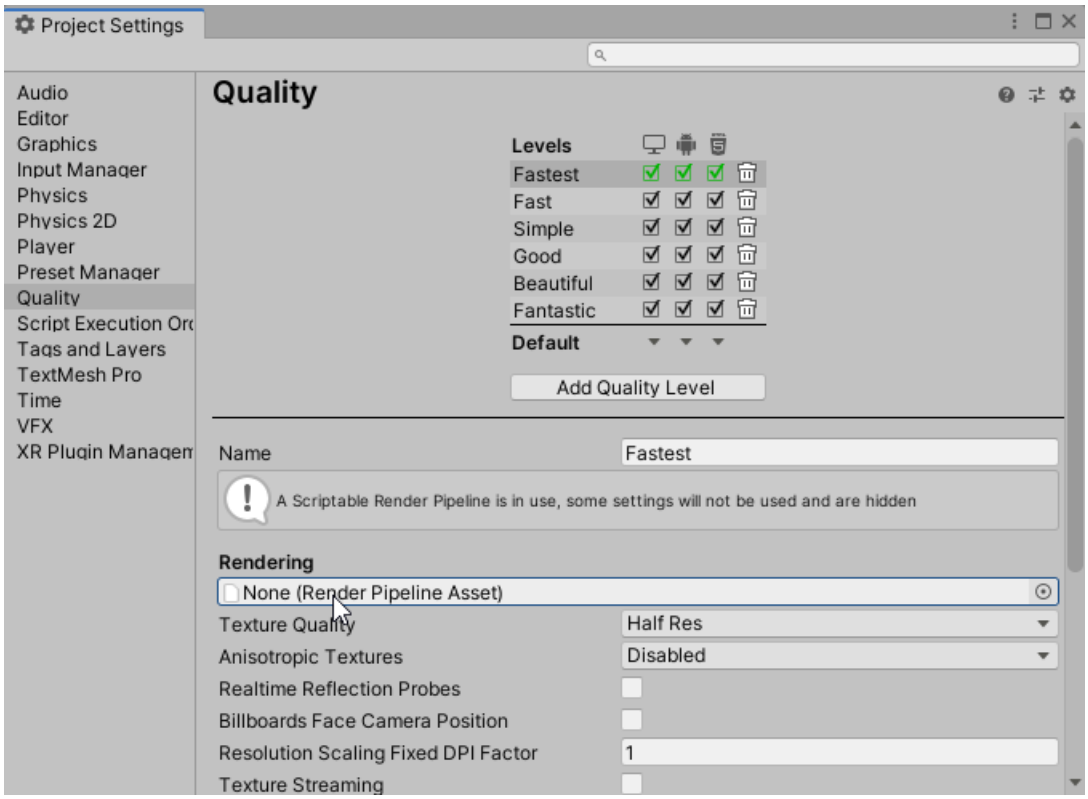

> **⚠ WARNING**
>
> You may have multiple Universal Renderer Assets for different platforms and Quality level. You have to add Translucent Image Blur Source to any that you want to use Translucent Image on.

### Finding the Universal Renderer Assets

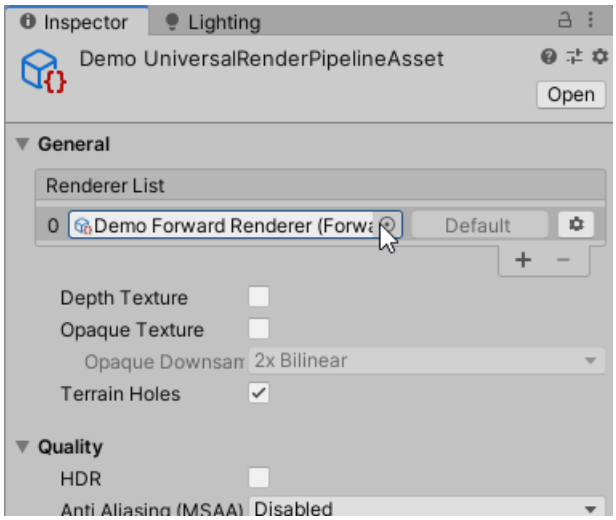
1. You can find the Universal Renderer asset(s) you're using by finding the Render Pipeline Settings asset in Graphic Settings:




2. You may also have more Quality Setting. Be sure to check all Quality Levels that you use:




3. Double-click the field under the cursor in the above images will take you to the Render Pipeline Settings asset, where you can find your Universal Renderer asset(s) in the list of Renderer:


Inspector
Lighting
Demo UniversalRenderPipelineAsset
Open
General
Renderer List
0
Demo Forward Renderer (Forwa
Default
Depth Texture
Opaque Texture
Opaque Downsam
2x Bilinear
Terrain Holes
Quality
HDR
Anti Aliasing (MSAA)
Disabled

# Customize

Translucent Image requires 2 components in a scene: 1 or more Sources that control the blur amount, and multiple Translucent Images that replace the built-in Image component.

## Translucent Image Source

Translucent Image Source generates the blurred background. It lets you control how much blur is present, the quality of the blur, and thus the performance trade-off.

### Blur Config

Share blur settings across multiple Cameras and Scenes through a Scriptable Object. There are 2 modes to control the blur strength. The Simple mode works well most of the time except at very low Strength. In Advanced mode, you can use the Radius property to gain fine control over very low blur Strength.

> **TIP**
>
> Unlike most blur solutions, with Translucent Image, blur strength have very little effect on performance. Step on that gas!

- **Downsample**: Reduce internal textures resolution. Larger values reduces memory usage and compute time, at the cost of quality.
- **Blur Region**: Limit the blur effect to a region of the screen. If your UI does not span the entire screen, it is a good idea to use this to increase performance and reduce power usage.
  - You can visualize and edit this interactively by turning on Preview. This field works the same as the Camera component Viewport field. It's easier to wield if you change `x` and `y` before `w` and `h`.
- **Max Update Rate**: How many times the screen is blurred per second. Use this to improve performance and decrease power usage.
  - Setting this to 0 will pause the effect completely. This can reduce power usage and prevent overheat when you don't need a dynamically updating background, for example, in a pause menu.
- **Cull Padding**: This setting is only visible while limiting Max Update Rate. Expand the blurred area to avoid gaps around moving UIs when using a low Max Update Rate.
  - Use higher value for lower Max Update Rate or faster UI movement. Use 0 for infinite Update Rate or static UIs.
  - For the best culling effectiveness, set this at runtime while UIs are moving, and reset to 0 while they're static.
  - Unit: fraction of the screen's shorter side.
- **Background Fill**: Fill the background where the frame buffer alpha is 0. Useful for VR Underlay and Passthrough, where these areas would otherwise be black.
- **Preview**: Preview the effect in full-screen. It also shows the Blur Region as a resizable rectangle.
- **Skip Culling**: Disable the culling system to avoid it's (minor) CPU cost, in case you know your UIs always cover the whole screen

## Translucent Image

Translucent Image shares many of the same properties as a normal Image, like controlling different types of Sprites, the Color, or how raycasting works and whether masking is applied.

The following explains controls specific to Translucent Image.

- **Source**: a Translucent Image Source component. Will be automatically set to the first one found, so you should make sure there's one in your scene before creating any Translucent Image. You can change this to select which camera will provide the background.

**Paraform**

Paraform is an addon to Translucent Image that provide procedural shape and glass refraction effect.

- **Corner Radii**: Indiviual corner radii. You can drag the rounded corner symbols to fine tune each radius. Click the link button to keep all corners the same.
- **Corner Curvature**: 0 is a flat diagonal corner. 1 is the perfect circle. Value > 1 increase curvature continuity target: 2 for G2 continuity, 3 for G3, and so on. Note that curvature continuity requires increased transition length, and is not guaranteed if the corner radius is too large compared to side length.
- **Fillet Curvature**: Same as Corner Curvature, but for the 3d bevel.
- **Edge Width**: Bevel width or thickness.
- **Elevation**: Distance to the below surface, affecting refraction.

**Appearance**

- **Foreground Opacity**: Mix between the Source Image and the blurred background. Use this instead of the Color's alpha channel. Alpha should only be used for fading the UI in and out.
- **Vibrancy**: Make the background more or less colorful. 0 means monochrome, and a negative value will invert the color.
- **Brightness**: In Normal Background Mode, brighten or darken the background. In Colorful mode, set the overall background brightness.
- **Flatten**: Reduce the background color contrast.

### Material Settings

**Background Mode**:

- **Normal**: Typical alpha blending. The foreground Sprite and Color is overlaid on top of the background, like thin cloth.
- **Colorful**: The background brightness is equalized by the Brightness property, while retaining colors as much as possible. The foreground Sprite and Color colors the background like tinted glass.
- **None**: For Paraform only. Turn off blurred background.

**Paraform**

- **Refraction**
    - **Refraction Mode**: Disable or enable refraction, without and with chromatic dispersion. More complex effects are more expensive.
    - **Refractive Index**: Higher values bend light more w.r.t. surface angle. https://en.wikipedia.org/wiki/List_of_refractive_indices
    - **Chromatic Dispersion**: How much different colors of light bend differently. This is 1/Abbe Number, so higher value disperse more. To use real abbe numbers, you can type in `1/Abbe Number` directly.
- **Edge Glint** : Control direction, strength, and shape of the edge glint.

# Controls Translucent Image from scripts

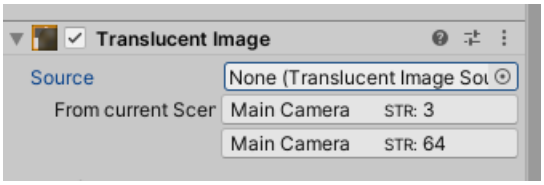
You can control all of the settings available in the inspector in C# through the exposed properties.The file `MainDemoViewController.cs` contains a short demonstration of how various properties can be accessed.

Be aware that some settings are stored in Scriptable Object and Material, which persist after exiting Play Mode. If you need to change these properties at runtime, consider making a copy of the Scriptable Object or Material.

# Multiple blur strengths on the same canvas

You can add multiple Sources to the same Camera, each with a different config and blur strength.

You can select them by clearing the Source field on the Translucent Image with Delete or Backspace. A list of Sources in the scene will be shown.




Alternatively, you can drag the Camera into the Translucent Image, a selector will then let you swap between Sources on the same Camera.

# Blurring other UI elements

Unity provides a few fixed points in the rendering process to acquire the screen for blurring, such as `AfterTransparent` or `AfterPostProcessing`. UIs are all rendered in one stage; as such, it is not possible to blur the background of any arbitrary UI elements without a significant performance cost.

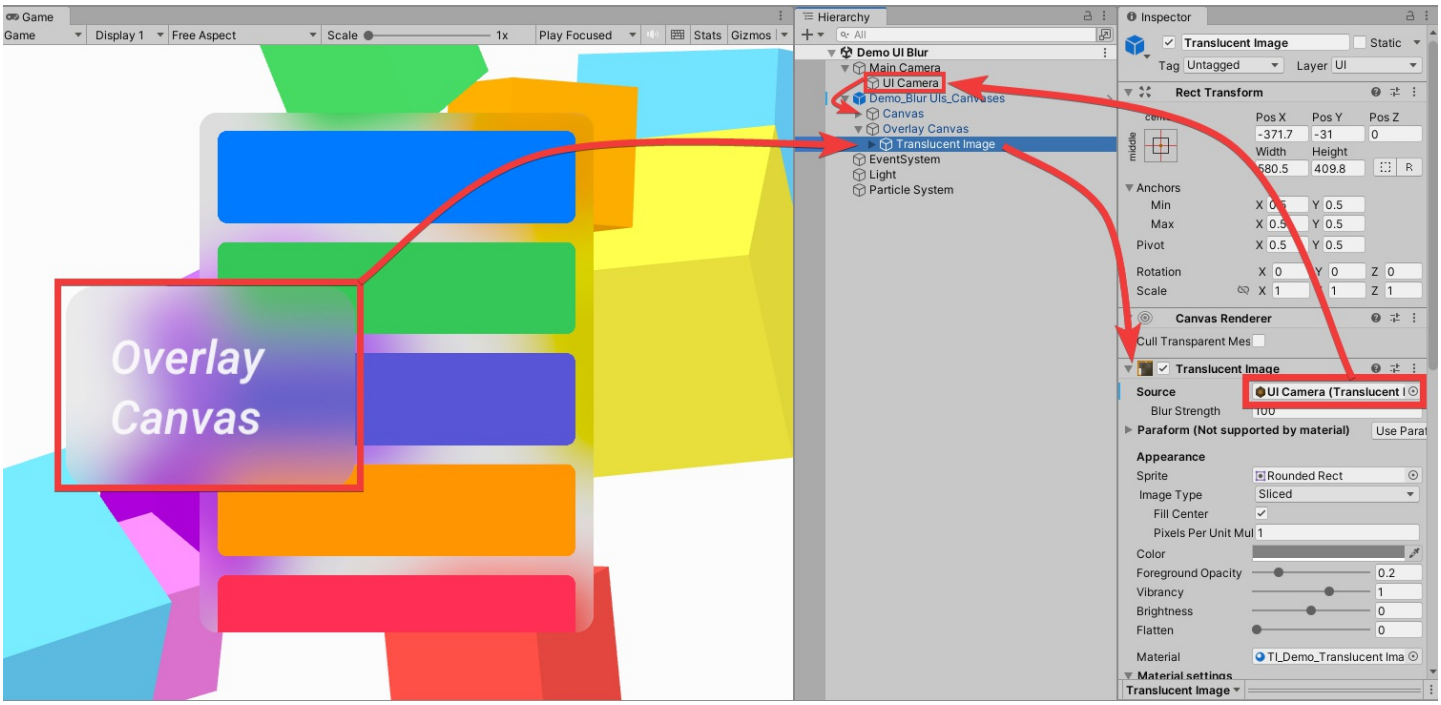
> **NOTE**
>
> Do not think of this as a limitation. Rendering UIs and blurring them all at once opens up a lot of optimizations. Allowing inserting pass everywhere, like the deprecated GrabPass, can degrade performance significantly.

We can work around this by separating the UIs into multiple Cameras and Canvases. Each camera allows us to capture all UIs it renders for blurring.

> **TIP**
>
> The best way to learn is by example! Check out the scene at `Demo/Render Pipeline Name/00_Scenes/Demo UI Blur.unity`

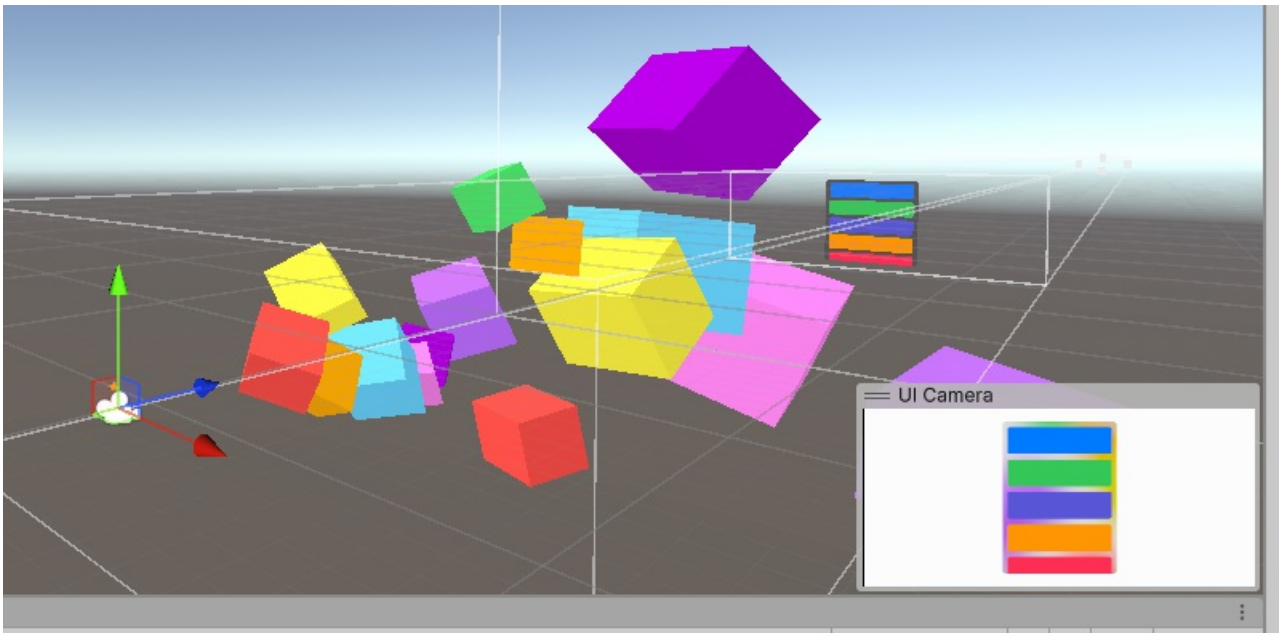
## Overview

For every layer of UI blurring you need, create an additional Camera & Canvas. Configure them so the Camera only renders its own canvas, using the Camera's `Culling Mask` and the Canvas render mode `Screen Space - Camera`.




*A Translucent Image in a `Screen Space - Overlay` Canvas referencing a Source on a Camera Rendering a `Screen Space - Camera` Canvas. Open in another tab for a bigger view*

What this camera sees will be in the background of Translucent Images that use it as Source. Ensure that these Translucent images do not appear in their own background. The Camera preview window is helpful in determining if you've set them up correctly.



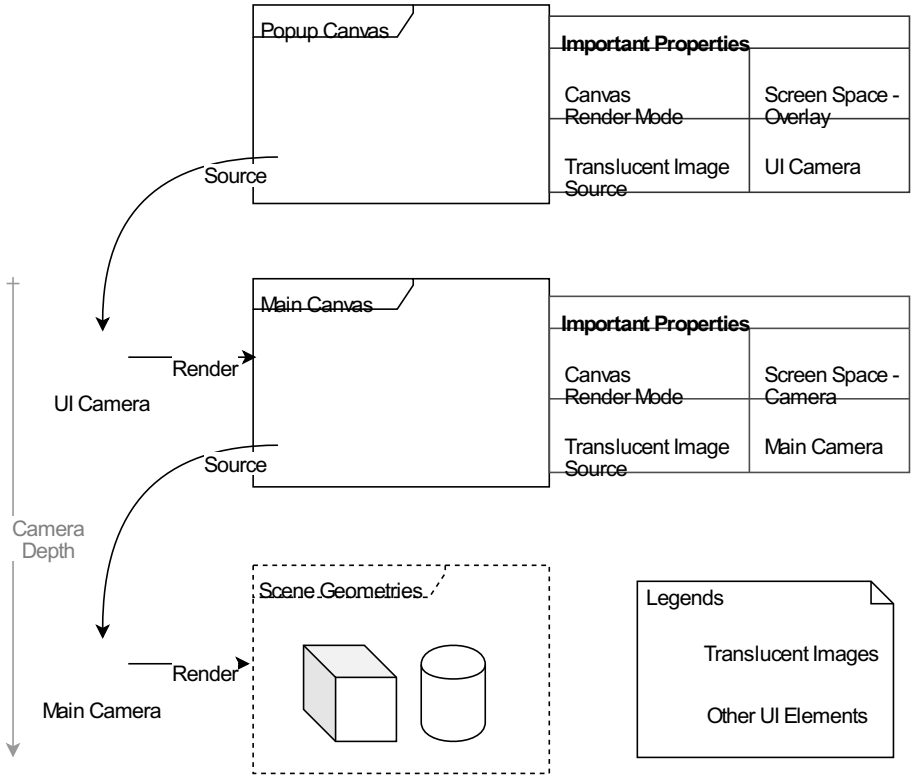

*The UI Camera only renders the* `Screen Space - Camera` *Canvas and nothing else*

# URP

# A detailed example setup

> **NOTE**
>
> This section uses BiRP wording. In URP, you can stack cameras by using the Stack section. Clear Flags: Depth only is called Background Type: Uninitialized.
>
> If you're using the Deferred Rendering Path, create an additional Renderer Asset with the Forward/Forward+ render path for the UI Camera. As it only renders UIs, the Deferred path brings overhead without benefits, and prevents camera stacking.

A common use of UI blurring is for pop-up panels. The following setup lets the Translucent Images in the Main Canvas blur the scene geometries. Those in the Popup Canvas will blur both the scene geometries, as well as UIs in the Main Canvas.



*An example setup. What a Camera render can be controlled using its Culling Mask property*

For reference, these are the important objects and settings:

```
├── Popup Canvas
│   ├─ Layer: UI
│   └─ Render Mode: Screen Space - Overlay
│
├── Main Canvas
│   ├─ Layer: UI
│   ├─ Render Mode: Screen Space - Camera
│   └─ Render Camera: UI Camera
│
├── UI Camera
│   ├─ Clear Flags: Depth only
│   ├─ Culling Mask: UI
│   └─ Depth: 2
│
└── Main Camera
    ├─ Clear Flags: any
    ├─ Culling Mask: Everything but UI
    └─ Depth: 1
```

# Performance implication

You can stack as many cameras and canvases as you like. However, with each extra Translucent Image Source you use, the GPU will have to do more work. A workaround is to disable the Source that is not the top-most. In fact, both Windows 10 and macOS do this:

*Windows 10 only uses blur on the top-most UI*

# World Space UI

World Space UI faces the same problem as blurring other UIs. Putting Translucent Images in world space will cause them to continuously blur themselves, becoming more opaque over time. See the linked page for reasoning.

The solution is also the same: use a separate Camera for the Translucent Images, an example of this setup is in the scene: `Le Tai Asset/TranlucentImage/Demo/World Space UI`. Particularly, the World UI Camera should:

- Have a higher Depth than your Main Camera (or order in URP).
- Have Culling Mask set to UI layer only.
- Have `Depth only` clear Flags.
- Other properties should match your Main Camera setting.
- Be in the same position as your Main Camera - setting it as children with position and rotation of (0,0,0) is the easiest way.

Also, your Main Camera should have its Culling Mask set to *exclude* the UI layer.

Now, your Translucent Images will always appear on top of scene geometry, even if they are further away. While this is not ideal in some situations, it still satisfies many use cases, and allows for significantly better performance.

# Troubleshooting common problems

## Compile error after updating the asset

- Always remove the asset folder entirely before importing. Just importing doesn't remove old files
- Shader in particular can sometime be incorrectly cached. Right click the asset's folder and click Reimport
- Sometime, the asset download can be corrupted. Completely remove the asset, then redownload and reimport it
- If the version of Unity you're using is very new, there's a chance it is not yet supported. Let me know!

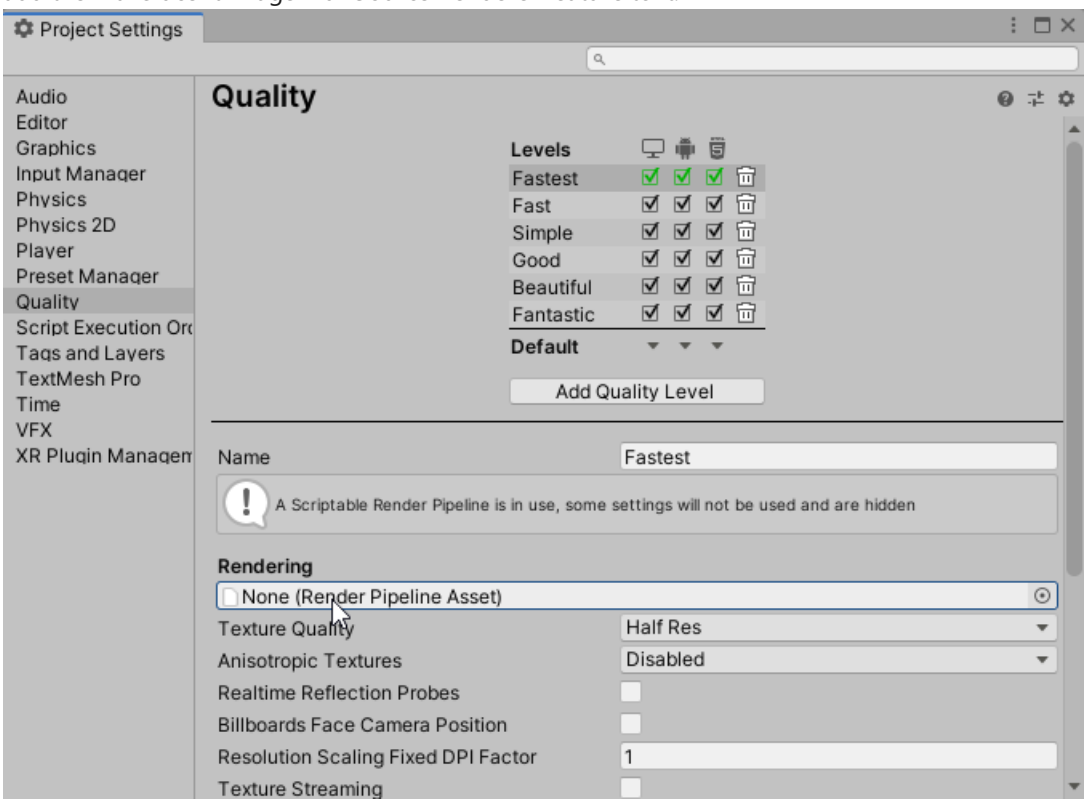
## The blur does not work. UI in the demo scene is just all white

This is usually due to one of the following:

- If you're using URP, you have to do some setup, as detailed in the Universal Render Pipeline section first.
- Sometime Unity's import process breaks some random things. Try to delete the whole folder and re-import the asset.

## The asset work in the Editor, but not in build

The most common issue when using URP is the build platform Quality Settings is overriding your Render Pipeline Asset with one without the asset's renderer feature. Check if any Quality level is overriding the Render Pipeline Asset, then either remove it or add the Translucent Image Blur Source Renderer feature to it.




## Blur look pixelated at low Strength

Due to the complex way the algorithm work, the Strength property is only an approximation. It is more accurate at a large amount. If you want finer control of very small blur, use the Advance mode and adjust Radius instead.

## Artifact around the edge of fast moving UIs

When limiting Max Update Rate, you should increase Cull Padding or disable Culling.

## Still not working? Have another question?

Contact me

# Frequently Asked Questions

## Will this asset work well on my device?

The asset should run on any device. Performance-wise, it depends on your project's existing GPU consumption, but here are some general rules of thumb:

- PC/Mac/Console: Should run well on almost everything except very old integrated GPU.
- Android: There are too many of them with too much difference in capability. The only way to know for sure is to test the demo on your target devices. On a very old flagship, the Samsung Galaxy S7 Edge, the demo runs at 60FPS with any setting. However, even the latest low-end phone may not run at full 60FPS well since they usually skim on GPU.
- IOS:
    - Iphone: Apple A8 and later should hit 60FPS. A7 can hit 30FPS.
    - Ipad: Because of the higher pixel count, you'll need to use the resolution scale features to hit 60fps on A9 and below.

## Does the asset support VR?

Yes. The asset support VR in the Single Pass Instanced / Multiview Rendering Mode, which should be the default.

## Are transparent objects blurred?

Yes

## Are UI blurred?

UI can be blurred with specific multi Canvases setup. See this page for more details.

## Can I smoothly animate the blur level?

The way the blur algorithm works makes it difficult to smoothly animate the blur amount. The Strength property allows for a smooth interpolation of blurriness in most case. There will still be some abrupt jump that is noticeable when interpolating slowly at lower blur amounts.

If you just need to fade in and out, you can use the alpha component of the Color property. You can also use Canvas Group as with normal Images.

## Still not working? Have another question?

Contact me