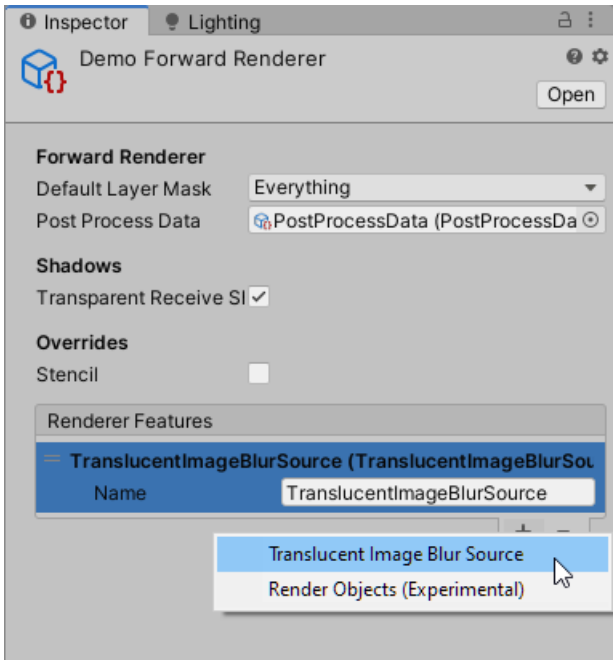
Table of Contents

Getting Started

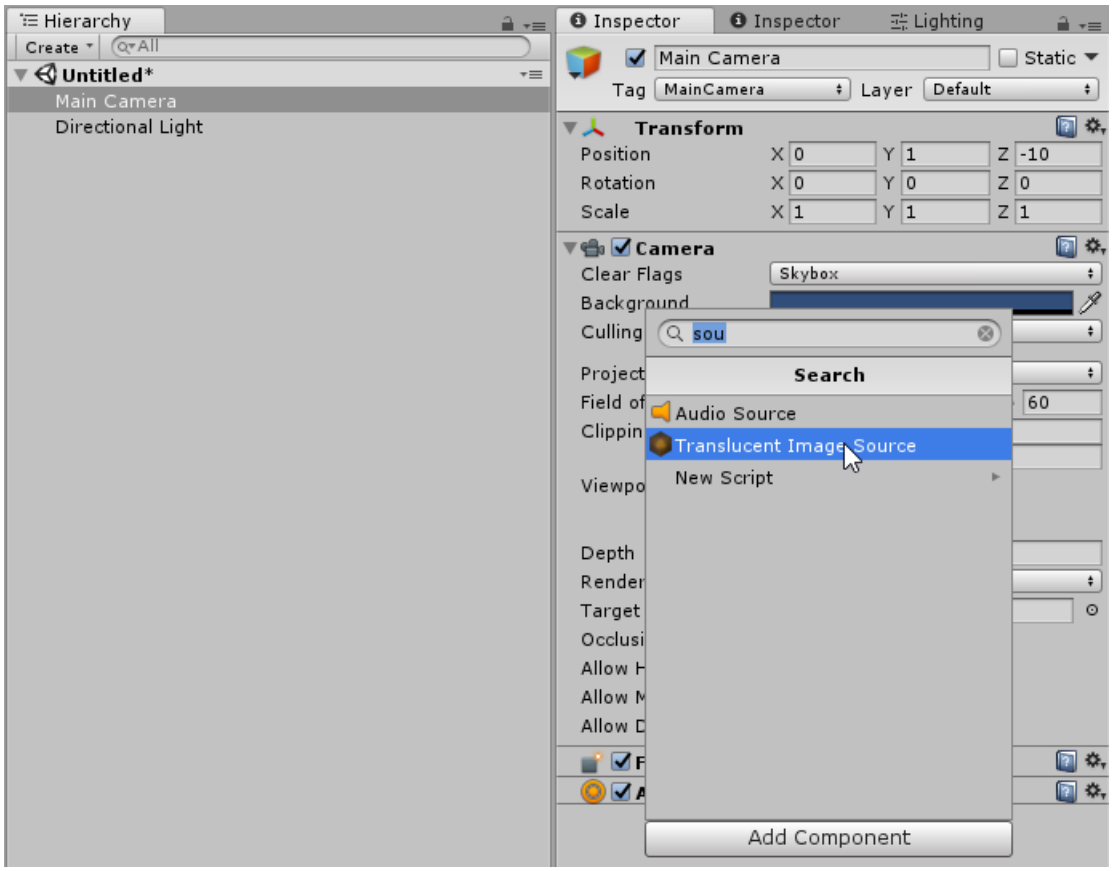
Installation

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, import the `URP Support` package at the root of the Translucent Image folder, and add the Translucent Image Blur Source renderer feature to your Scriptable Renderer setting asset. See the URP page for more detail.

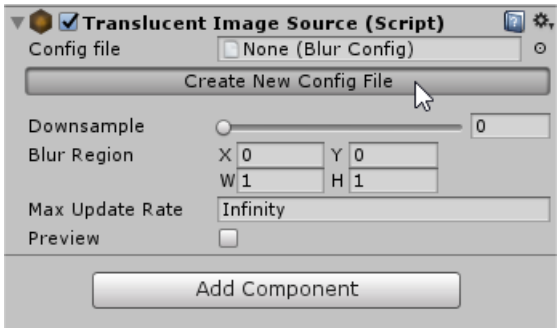


Adding Translucent Image to your scene

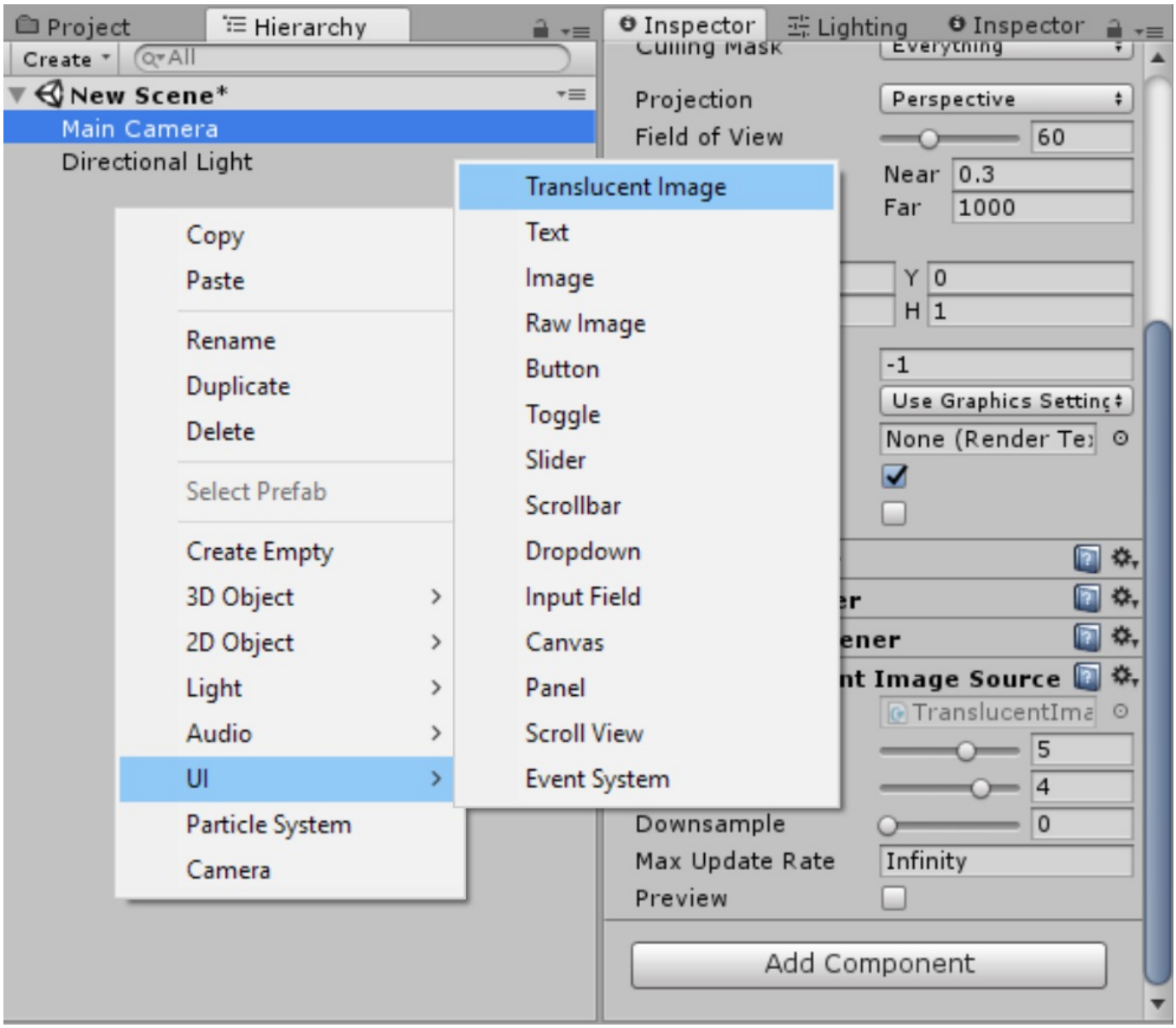
1. Add **Translucent Image Source** to your main camera.



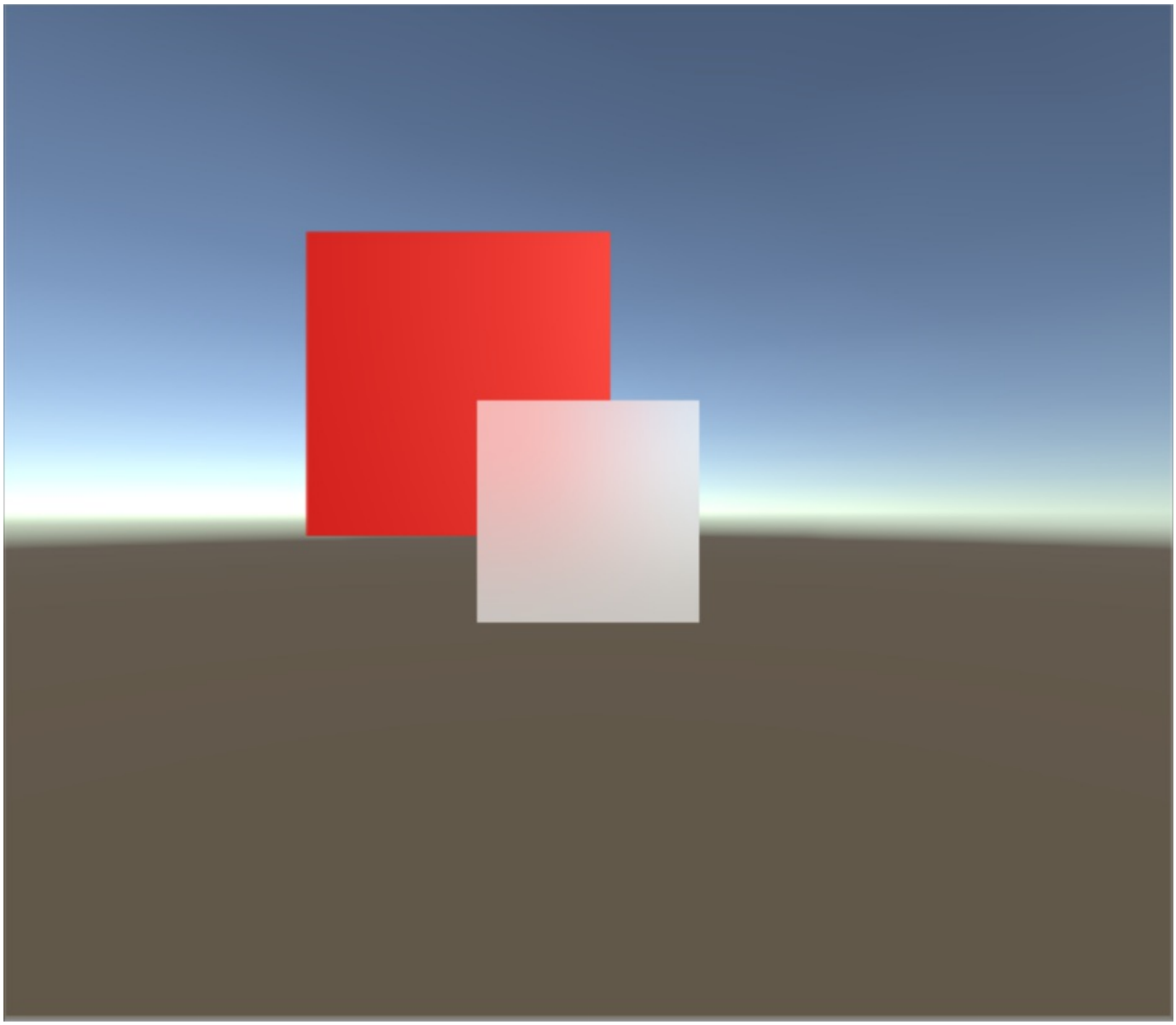
2. Create a ***Blur Config*** asset (or assign an existing one).



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.

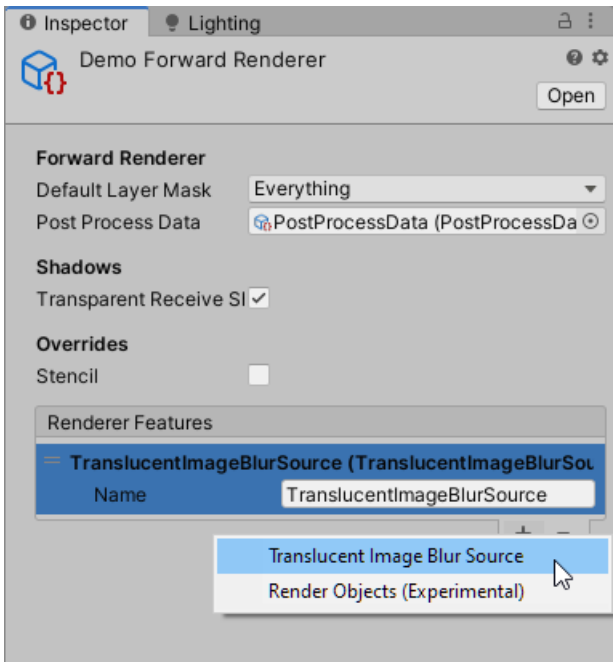
Universal Render Pipeline

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

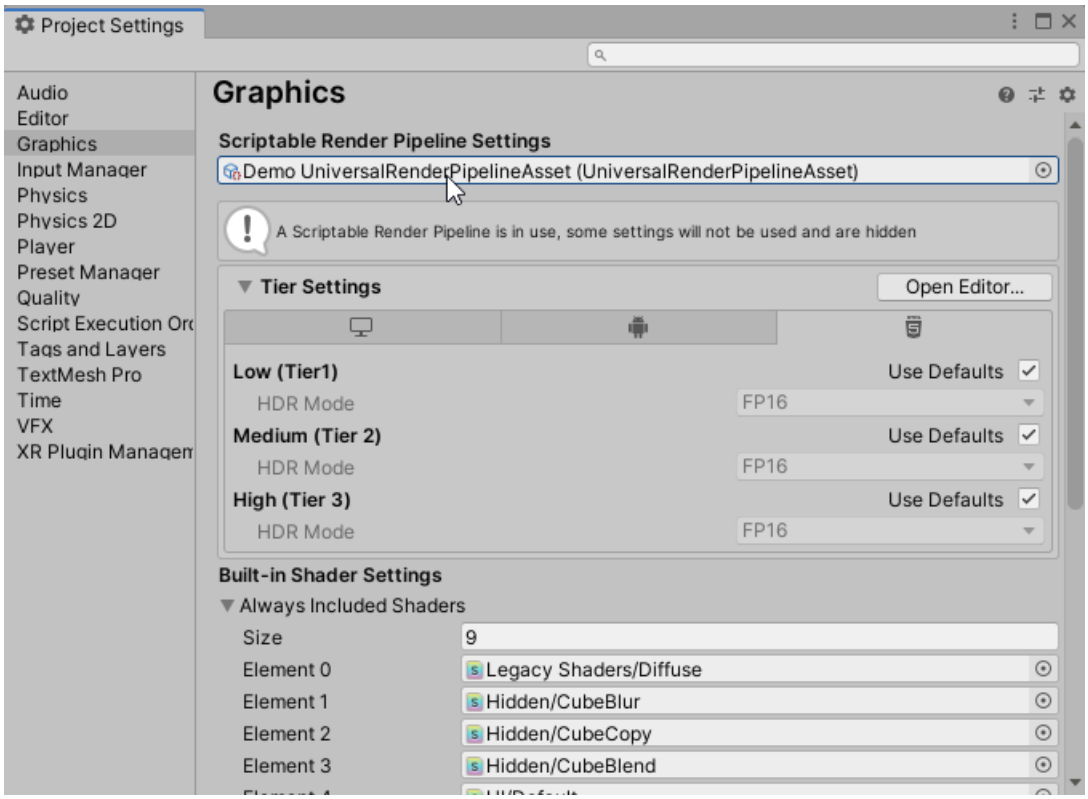
1. Import the package at `TranslucentImage/UniversalRP support`.
2. Find your Universal Renderer Assets and add Translucent Image Blur Source to its list of Renderer Features:



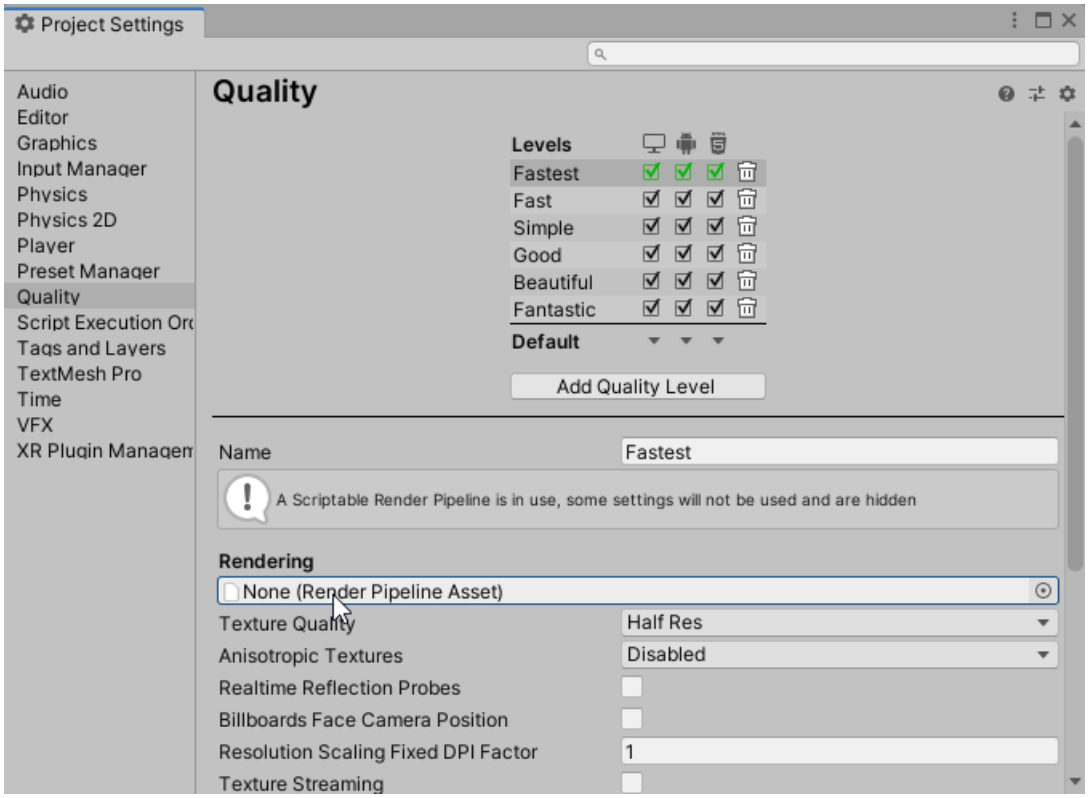
> ⚠ **WARNING**
>
> You may have multiple Universal Renderer Assets for different platforms and Quality level. You have to add Translucent Image Blur Source to any that you want to use Translucent Image on.

Finding the Universal Renderer Assets

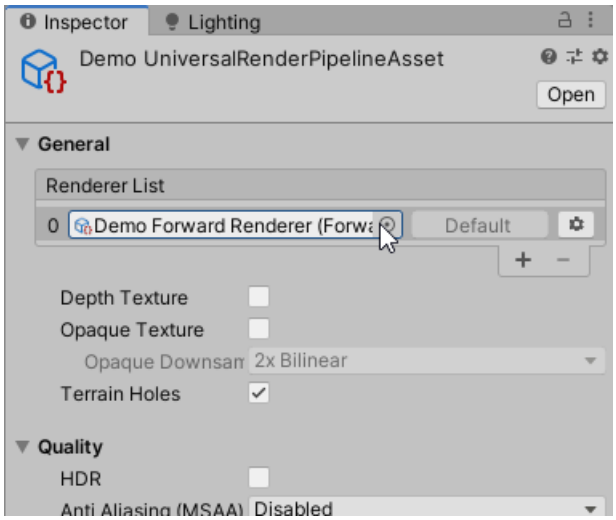
1. You can find the Universal Renderer asset(s) you're using by finding the Render Pipeline Settings asset in Graphic Settings:



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

Blur Config

Share blur settings across multiple Cameras and Scenes through a Scriptable Object. There's two mode to control the blur strength. The Simple mode work 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 small 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 visually edit this by turning on Preview. The number here works the same as the Camera component Viewport. 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.
- **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

- **Sprite, Image Type, Raycast Target, Maskable, Color**: same as built-in 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.
- **Sprite Blending**: Mix between the Source Image and the blurred background. This should be what you would use instead of the alpha channel of the Color property most of the time.
- **Material**: Multiple Translucent Images using the same material share some settings. They can also batch dynamically into a single draw call.

> **WARNING**
>
> - Materials used here must use the shader UI/TranslucentImage.
> - You should create your own Material instead of the default to avoid changes to look after asset updates.

Material settings

The following settings are shared across Translucent Images using the same material:

- **Vibrancy**: Colorfulness of the background. 0 means monochrome, and a negative value will invert the color.
- **Brightness**: Brighten or darken the background.
- **Flatten**: Reduce the background contrast. Useful when you can't predict the color of the background but want to keep the content on top of it legible.

Controls Translucent Image from scripts

You can control all of the settings available in the inspector in C# through the exposed properties. See: @LeTai.Asset.TranslucentImage.TranslucentImage and @LeTai.Asset.TranslucentImage.TranslucentImageSource . The file `MainDemoViewController.cs` contains a short demonstration of how various properties can be accessed.

```
// Change blur strength
translucentImageSource.BlurConfig.Strength = value;

// Change material property
translucentImage.materialForRendering.SetFloat(ShaderID.VIBRANCY, value);
```

Be aware that many 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.

Blurring other UI elements

To achieve the best possible performance, Translucent Image does not support blurring UI elements in arbitrary order. However, UIs in a Canvas can blur UIs in different Canvas, by using additional Cameras.

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

The idea

To understand the setup, it's useful to understand how Translucent Image works.

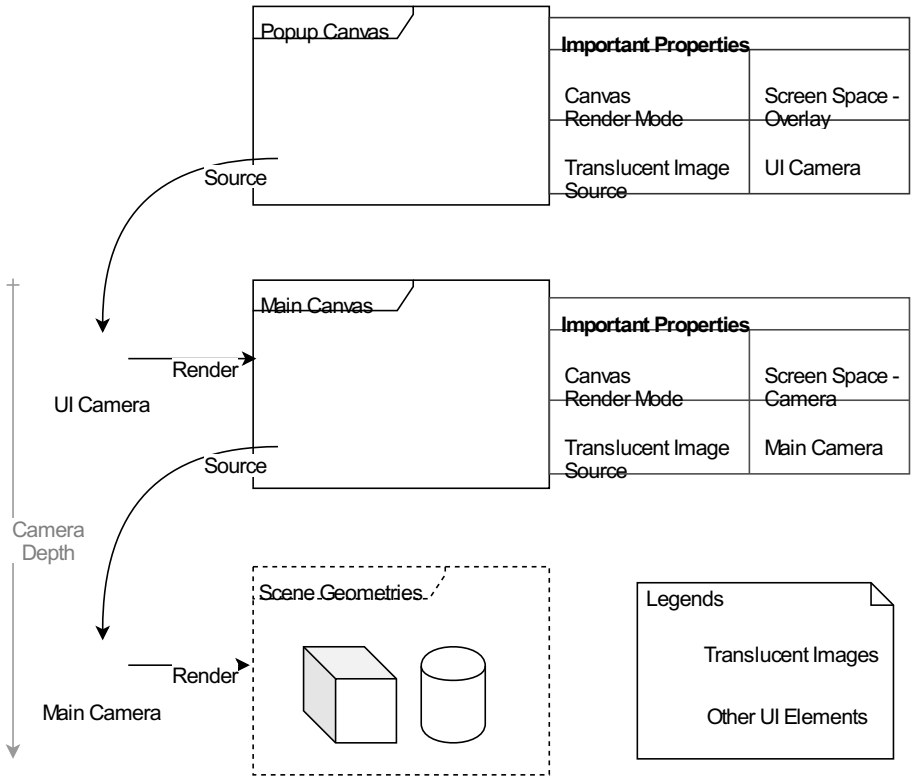
Translucent Images **do not blur** things by themselves. Instead, the Translucent Image Source component **blurs the entire screen** at once, then shares the result with multiple Translucent Images. This massively **reduces the number of pixels** that have to be blurred. It's one reason why Translucent Image is so fast.

The Translucent Image Source component "see" and blurs what the Camera they attached to render. If the Source "sees" the Translucent Images, these Images will appear in their own background, causing a feedback loop and making the Images turn **opaque over time**. It's important to make sure Translucent Images are **not "seen" by their own Source**.

So, to blur UIs, you need at least **2 Canvases**: a lower one containing the UIs to be blurred, and an upper one for the Translucent Images. Then, set the Camera with the Translucent Image Source to only render the lower Canvas.

An example setup

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.

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.

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 Setting is overriding your Render Pipeline Asset with one without the asset's renderer feature.

Compilation error after asset import/update

- If you're updating the asset, completely remove the asset first. Importing an asset does not remove old files.
- If you're using a very recently released version of Unity, the asset may not have been updated. Please let me know
- Sometime the asset download are corrupted. Completely remove the asset, then redownload and reimport it.

Can I smoothly animate the blur level?

The way the blur algorithm works makes it difficult to smoothly animate the blur amount. Changing the Strength property allows for a mostly smooth interpolation of blurriness, but 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

Support

If you need assistance regarding the asset or have a feature request, feel free to contact me by the form below or at:
https://letai.freshdesk.com/support/tickets/new

You will receive an automated confirmation email shortly after submitting a ticket. If not, double check the email address used and try again.

Including this information in your support request will help quicken the troubleshooting process:

- Your Unity version and the render pipeline used
- Any relevant error messages, or the fact that there are no errors
- Whether the demo scenes are working correctly
- If the asset doesn't look correct, please describe in which way

Support request