AMD FidelityFX Super Resolution 2 Upscaling for Unity Documentation

Current version: FSR 2.2.0a

About FSR 2

This Unity asset is created to interface with the open-source AMD FidelityFX-FSR2 found here: https://github.com/GPUOpen-Effects/FidelityFX-FSR2.

AMD FidelityFX-FSR2 is an upscaling technique, creating high quality and resolution frames based on lower resolution input. By using this, projects could have drastically lower GPU requirements than without.

Only if your project is limited by GPU performance, FSR 2 will gain you a higher framerate. If a project is limited by CPU performance, all it will do is make the GPU workload lower. While this may seem like a big limitation, it also means a laptop will use way less battery power when using FSR 2!

Current supported Unity Render Pipelines:

Built-in (BIRP), Universal Render Pipeline (URP) and High-Definition Render Pipeline (HDRP).

Current supported platforms:

- Windows (DX11, DX12, Vulkan)
- Linux (Vulkan)
- MacOS (Metal)
- iOS (Metal)
- Android (Vulkan)
- Playstation 4
- Playstation 5
- Xbox One
- Xbox Series X|S
- Nintendo Switch

Not tested platforms:

- VR

Quick Start

This chapter is written to add FSR 2 as fast as possible to your project. However, it is very much recommended to read the Important Information chapter. FSR 2 upscales very well when used properly, but it will take some tweaking to get the best quality possible for your project specifics.

Goto: Quick Start Built-in Render Pipeline
Goto: Quick Start Universal Render Pipeline
Goto: Quick Start High-Definition Render Pipeline

Quick Start: BIRP

Step 1: Import the AMD FSR 2 Package in your project.

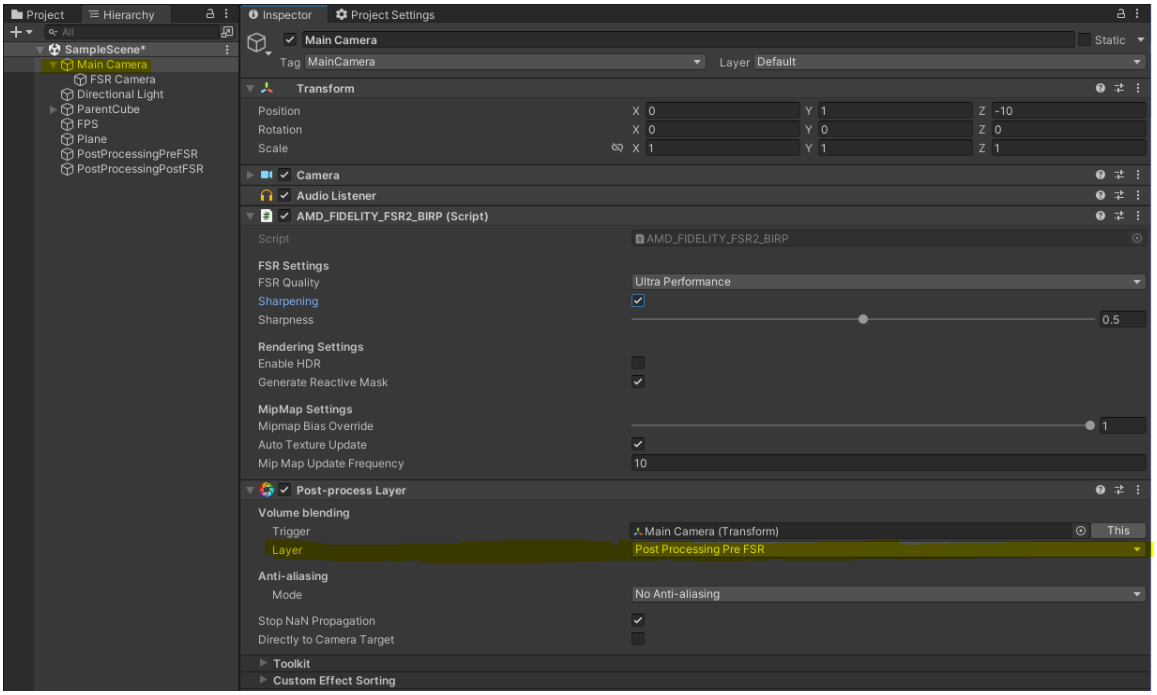
Step 2: Add AMD_FIDELITY_FSR2_BIRP.cs script to your main camera.

Hit play!

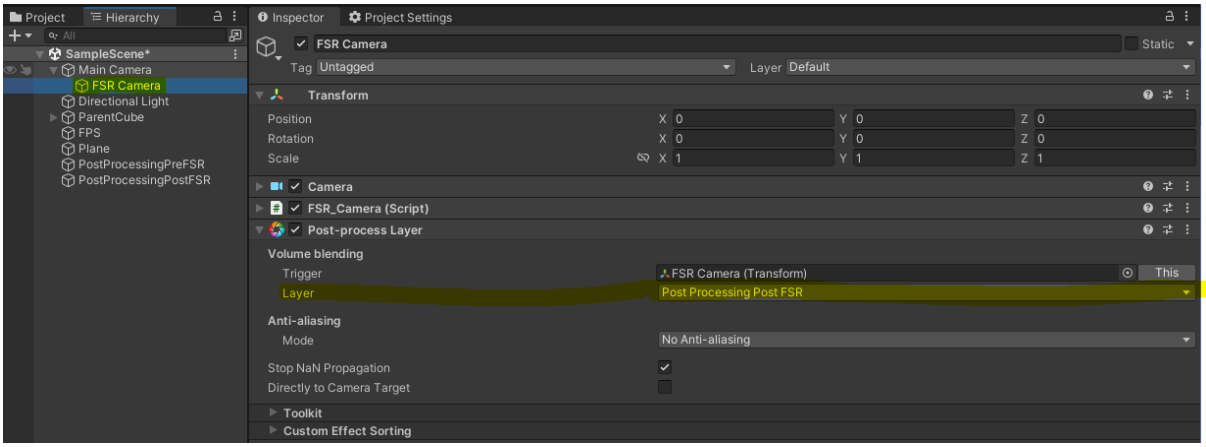
Note: If you are using Unity Post Processing. Read chapter Post-Processing or check out the Demo's for more information.

Examples

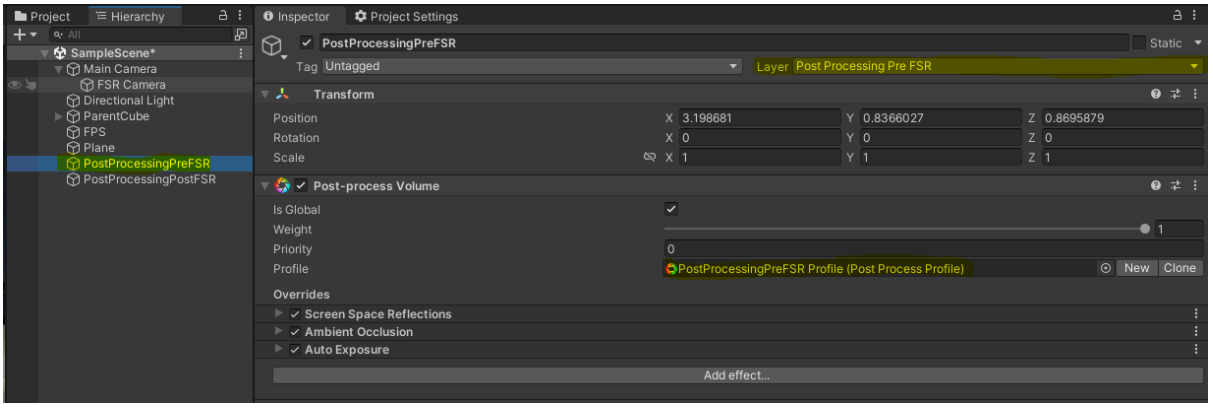
Main Camera Setup



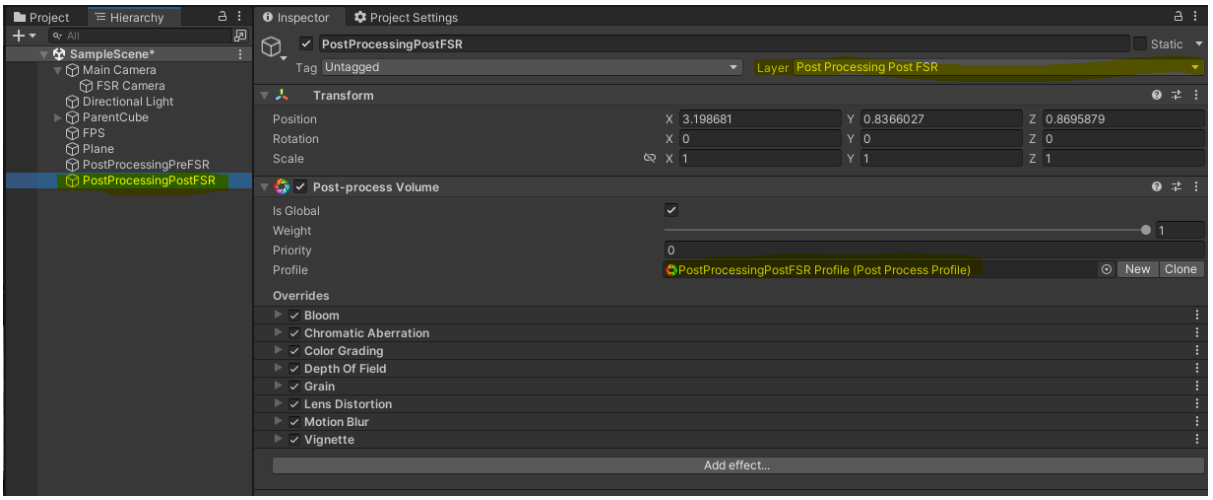
FSR Camera



Pre FSR Post-Processing Volume



Post FSR 2 Post-Processing Volume



Quick Start: URP

Step 1: Import the AMD FSR 2 Package in your project.

Step 2: Add AMD_FIDELITY_FSR2_URP.cs script to your main camera.

Step 3: Add the “FSR Scriptable Render Feature” to your Universal Renderer Data files. (Add it to all the URP data files you will use in your project, for example if you use different ones per quality setting).

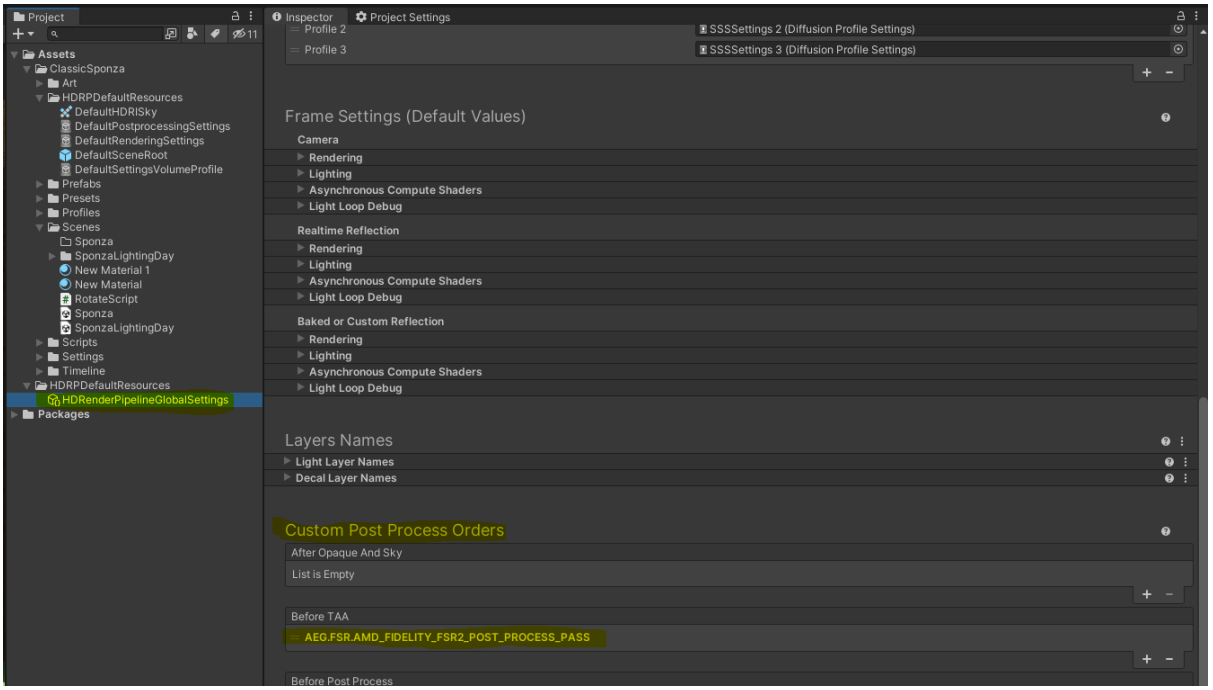
Hit play!

Quick Start: HDRP

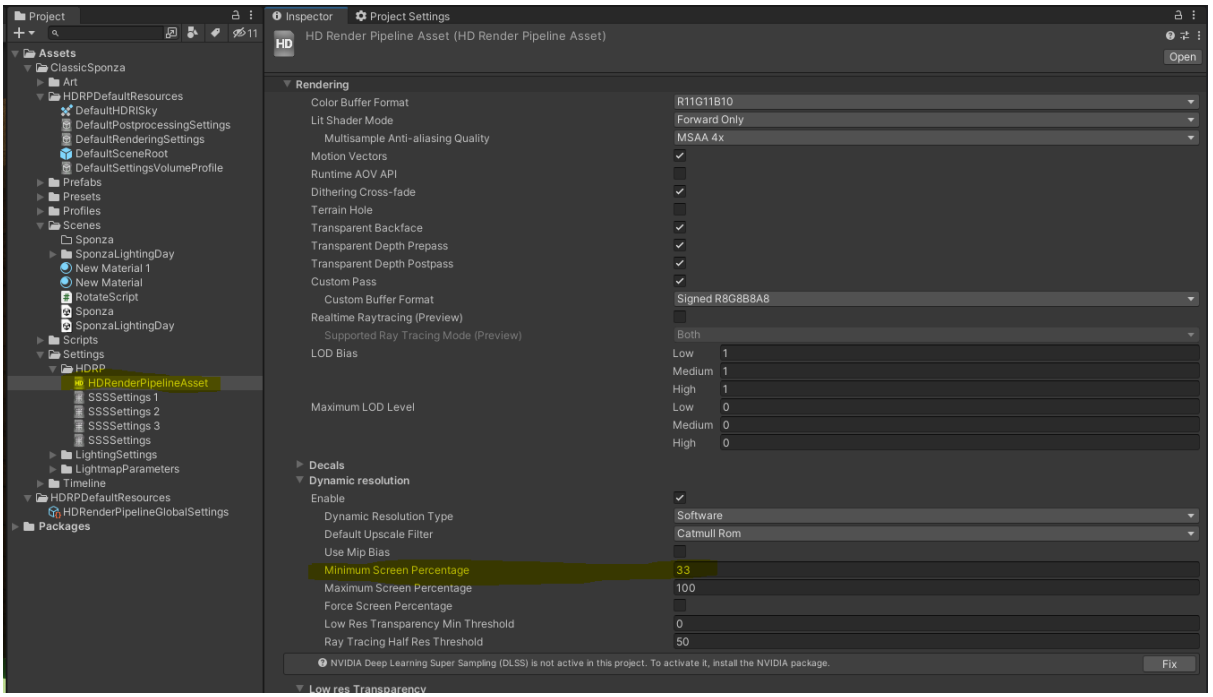
Step 1: Import the AMD FSR 2 Package in your project.

Step 2: Edit the HDRP source files, seen chapter HDRP source files

Step 3: Add the AMD_FIDELITY_FSR2_POST_PROCESS_PASS to the HDRenderPipelineGlobalSettings in “Before TAA”.



Step 4: Enable Dynamic Resolution in HDRP settings and set the “Minimum Screen Percentage to 33. Keep the Default Upscale Filter to Catmull Rom.

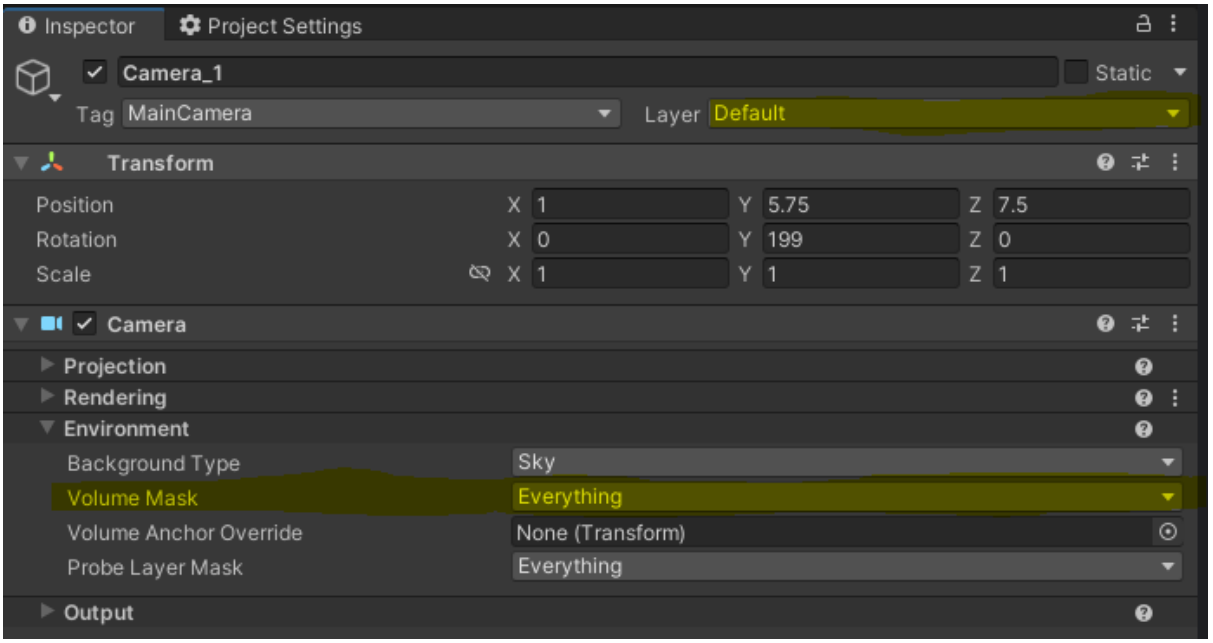


Step 4: Add AMD_FIDELITY_FSR2_HDRP.cs script to your main camera.

Hit play!

The next steps are optional, because FSR will do these steps for you if you skip them.

Optional Step 5: Make sure the “Volume Mask” of your camera includes the layer of your camera. If you skip this step, FSR will automatically do this for you.



Optional Step 6: Enable “Allow Dynamic Resolution” on the Camera. If you skip this step, FSR will automatically do this for you.

Important Information

Temporal Anti-Aliasing (TAA)

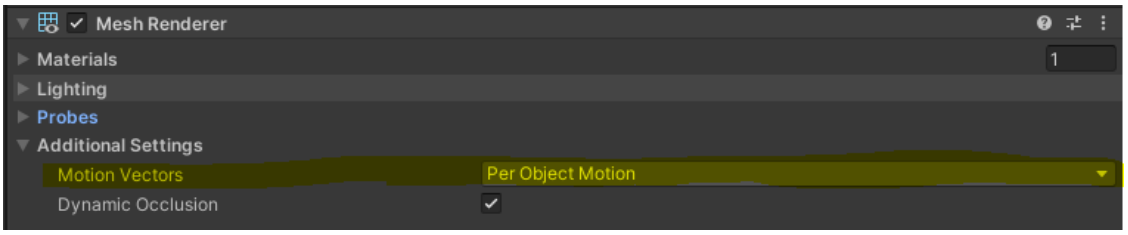
FSR 2 has a very good built-in TAA solution. This is required for FSR2 to do its work. There is no need to add any other TAA to the camera, since this will only add more blur and ghosting. Adding different types of Anti-Aliasing will also decrease the upscaling quality.

Motion Vectors

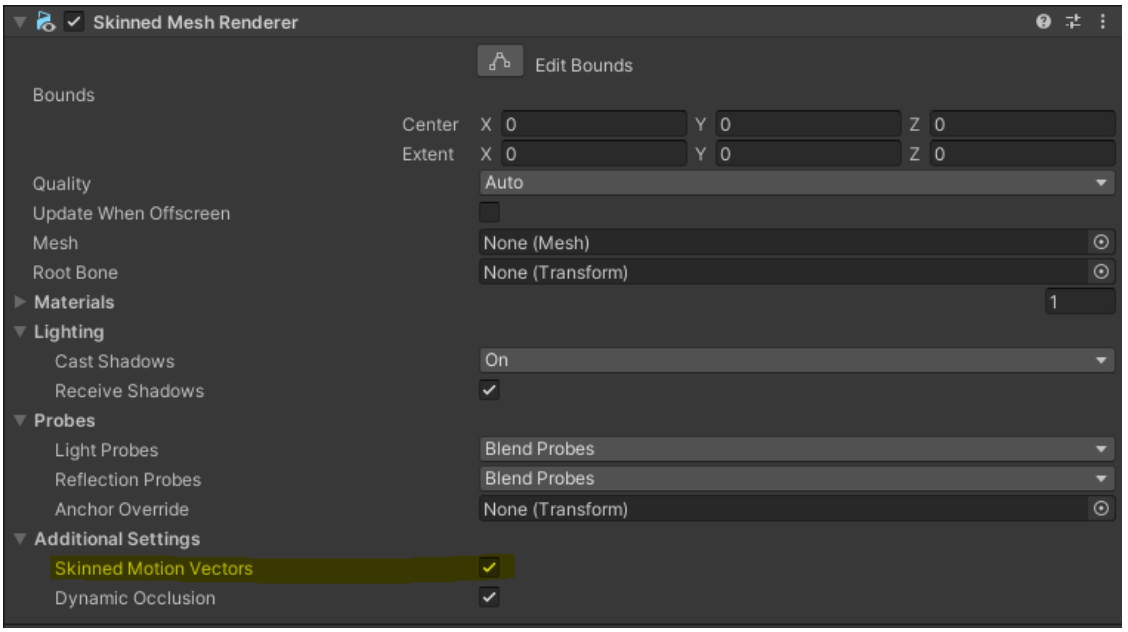
FSR 2 uses the motion-vector buffer to calculate how much an object moves, to improve upscaling. If you want opaque objects to be upscaled in FSR 2, always enable Motion Vectors.

Examples

Mesh Renderer



Skinned Mesh Renderer



Alpha-Blended / Transparency

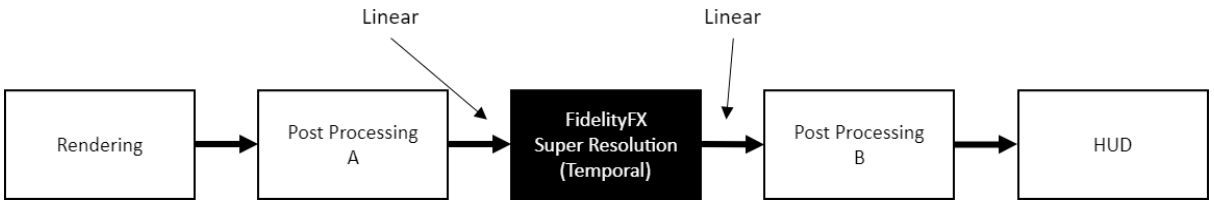
In Unity, alpha-blended and transparent objects are often not added to Unity's motion vector pass. For this FSR 2 has a feature called a Reactive Mask, by default enabled in the inspector. Enabling this setting will drastically improve Alpha-Blended / Transparent objects, without the need to add them to the motion-vector buffer.

Camera Transform

Since FSR 2 uses multiple frames to upscale, this can cause rendering artefacts when you have very fast camera transform changes. To improve this, call OnResetCamera on the same frame you change your camera transform, to let FSR 2 know it needs to drop previous rendered frames.

Post Processing Effects

For every upscaling technique, is it very important to know how to use Post Processing effects. Some effects will need to be added before FSR 2, others afterwards. Check out our demo's for examples.



Post Processing order suggestions by AMD:
https://github.com/GPUOpen-Effects/FidelityFX-FSR2#placement-in-the-frame

In BIRP this gives us a bit of an issue.

The camera renders the scene at a lower resolution, which gets upscaled by FSR, we then order the camera to resize back to its original resolution and then we blit the upscaled image back. This is the most performant way, however it will break many Post Processing effects because to make this work, we need to change the PPV2 CameraEvent from BeforeImageEffects to AfterForwardAlpha.

For this, we have edited a version of Unity's Post Processing 3.2.2, which you can download here, that inserts the Post Processing stack into an earlier CameraEvent (AfterForwardAlpha) when FSR is enabled.

The downside of this process is that all Post Processing is done before FSR 2 instead of after. Better ways to do this are in the making.

Note: If your BIRP project uses other methods of Post Processing, make sure you also change the injection point to AfterForwardAlpha.

Mipmaps - BIRP Only

When FSR 2 downscales the source rendering, it is recommended to add a negative mipmap bias to all textures in your scene for improved quality. Textures will look very washed out otherwise. In the Inspector chapter, we explain how we created a way to update all mipmap biases on the textures automatically.

Demo Scenes

General

With the Quality.cs script you can toggle between quality modes by pressing the spacebar.

BIRP

Download BIRP Demo Project here

FSR 2 Demo Scene

The setup in this demo scene shows how FSR 2 can be implemented with Unity's Post-Processing Stack v2. Read more about it here.

URP

Download URP Demo Project here

HDRP

Download HDRP Demo Project here

Note, the HDRP demo has been created in Unity 2021.3 LTS, and HDRP 12, the demo project might break on older or newer Unity versions.

Inspector

Quality

Off: Disables FSR 2.
Quality: 1.5x scaling
Balanced: 1.7x scaling
Performance: 2.0x scaling
Ultra Performance: 3.0x scaling

Example: If the native resolution is 1920x1080, selecting the **Performance** option will change the rendering resolution to 960x540, which will then be upscaled back to 1920x1080. Changing the quality setting will update the “FSRScaleFactor” variable.

Sharpening

Off - On

All Temporal Anti-Aliasing solutions add some blur, to prevent this, FSR 2 features a sharpening filter. With these settings you can enable/disable this filter.

Sharpness

0.0 - 1.0

With this slider you can change the amount of sharpening.

Generate Reactive Mask

Off - On

Unless you do not use any Transparent or Alpha Blended objects in your project, we recommend keeping this enabled. Otherwise, disabling it will free up some additional memory.

Reactive Scale

0.0 - 1.0

Larger values result in more reactive pixels. Recommended default value is 0.9f

Reactive Threshold

0 - 1.0

Smaller values will increase stability at hard edges of translucent objects. Recommended default value is 0.05f.

Reactive Binary Value

0 - 1.0

Larger values result in more reactive pixels. Recommended default value is 0.5f

Auto Texture Update - BIRP only
Off - On

As previously explained, it is recommended to update the MipMap Bias of all used textures. In Unity, the only way to do this is by script, texture by texture. This is less than ideal, so we added a feature to automatically update all textures currently loaded in memory. In real-world projects, we saw no noticeably extra CPU cost.
Note: It seems URP and HDRP already automatically do mipmap biassing, so here we disabled this feature by default.

Mip Map Update Frequency
0.0 - Infinite

This settings determines how often the Auto Texture Update checks for new loaded textures to update the Mipmap Bias for.

Mipmap Bias Override - BIRP only
0.0 - 1.0

When using FSR 2, and changing the MipMap bias, it is possible that there will be additional texture flickering. If you notice too much texture flickering, try lowering this setting until you have the desired balance between quality and texture stability. If you have no texture flickering, keep this to 1 for best quality.

Public API

Generic

public void OnResetCamera()
Resets the camera for the next frame, clearing all the buffers saved from previous frames in order to prevent artefacts.
Should be called in or before PreRender oh the frame where the camera makes a jumpcut.
It is automatically disabled the frame after.

BIRP Only

public void OnMipmapSingleTexture(Texture texture)
Updates a single texture to the set MipMap Bias.
Should be called when an object is instantiated, or when the ScaleFactor is changed.
Use this function if you are not making use of the Auto Texture Update feature.

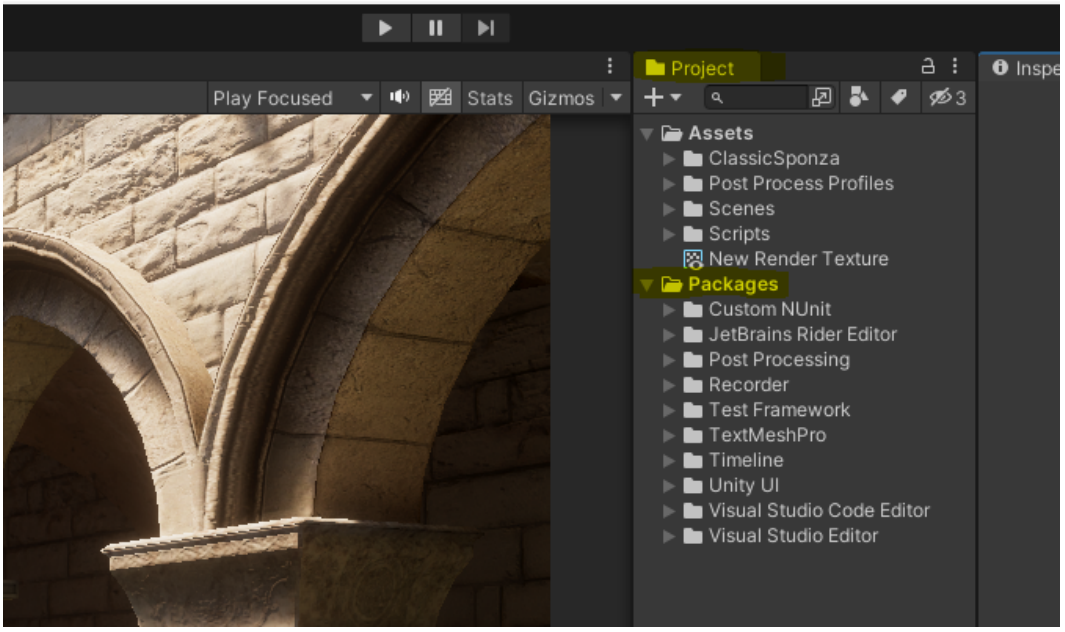
public void OnMipMapAllTextures()
Updates all textures currently loaded to the set MipMap Bias.
Should be called when a lot of new textures are loaded, or when the ScaleFactor is Changed.
Use this function if you are not making use of the Auto Texture Update feature.

public void OnResetAllMipMaps()
Resets all currently loaded textures to the default MipMap Bias.

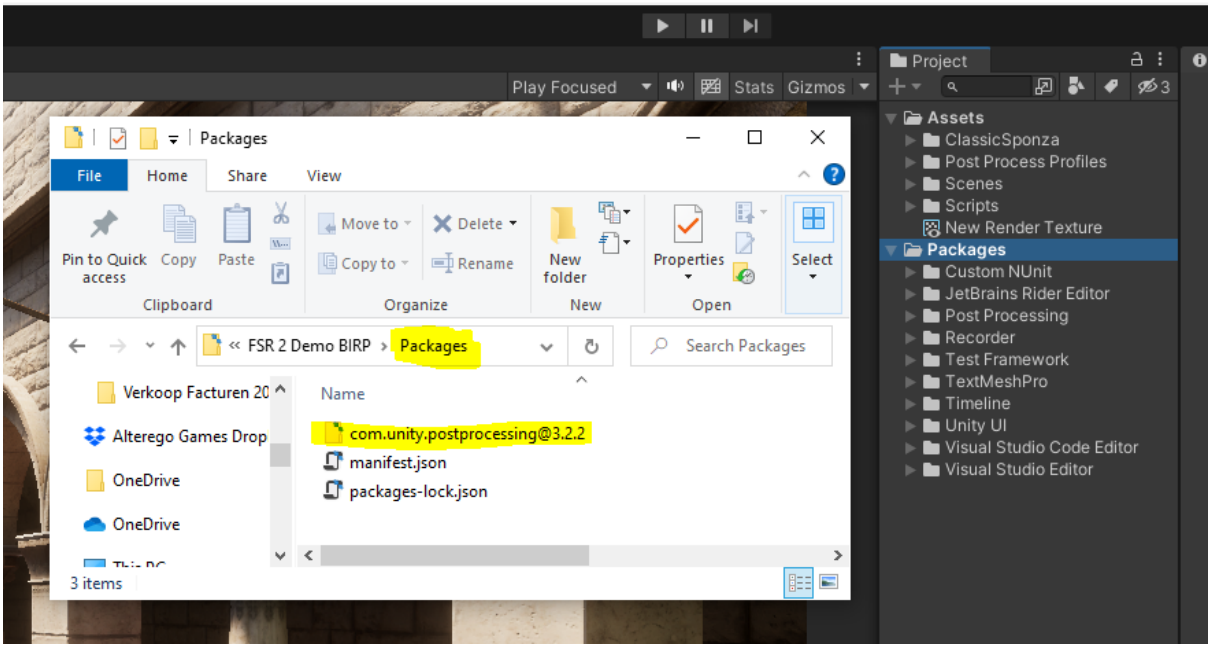
Editing Unity Post Processing package

First of all, download our edited Unity Post Processing package here. Use this edited package only when you're planning to use BIRP and want to be able to use Unity's Post Processing Stack V2.

Step 1: Locate the Packages folder in your project, press Right-Mouse Button on it and select "Show in Explorer"



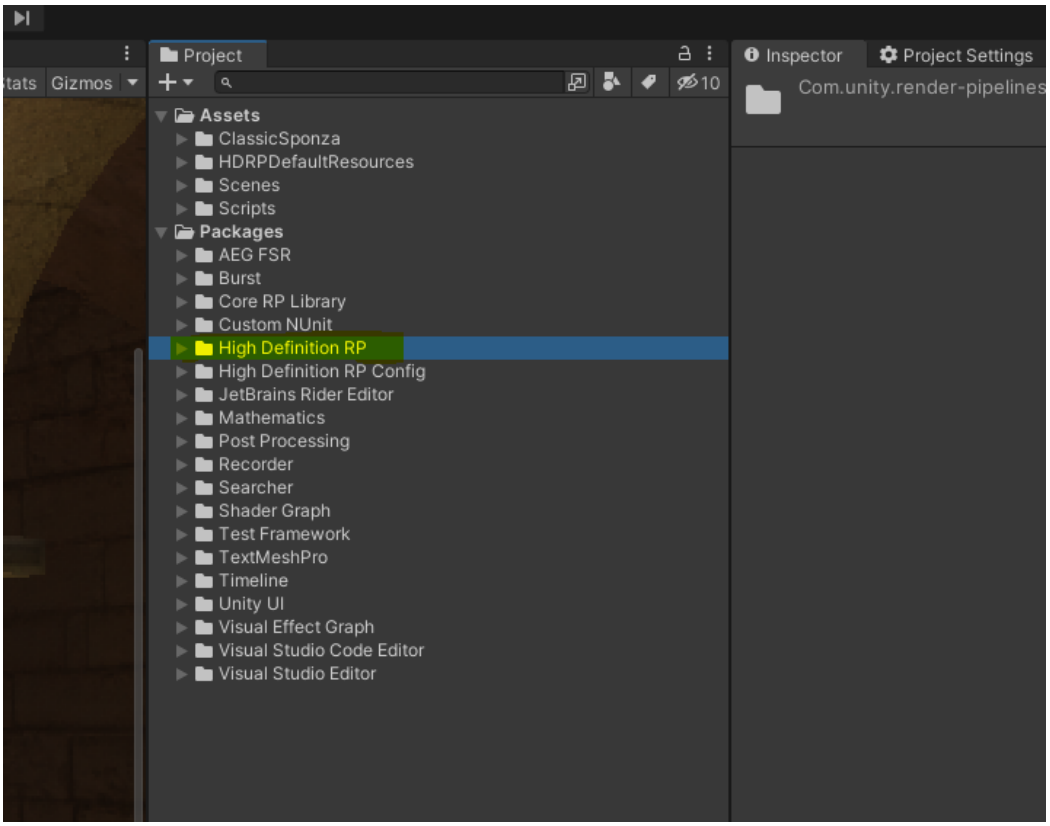
Step 2: Unzip the downloaded package into the Packages folder



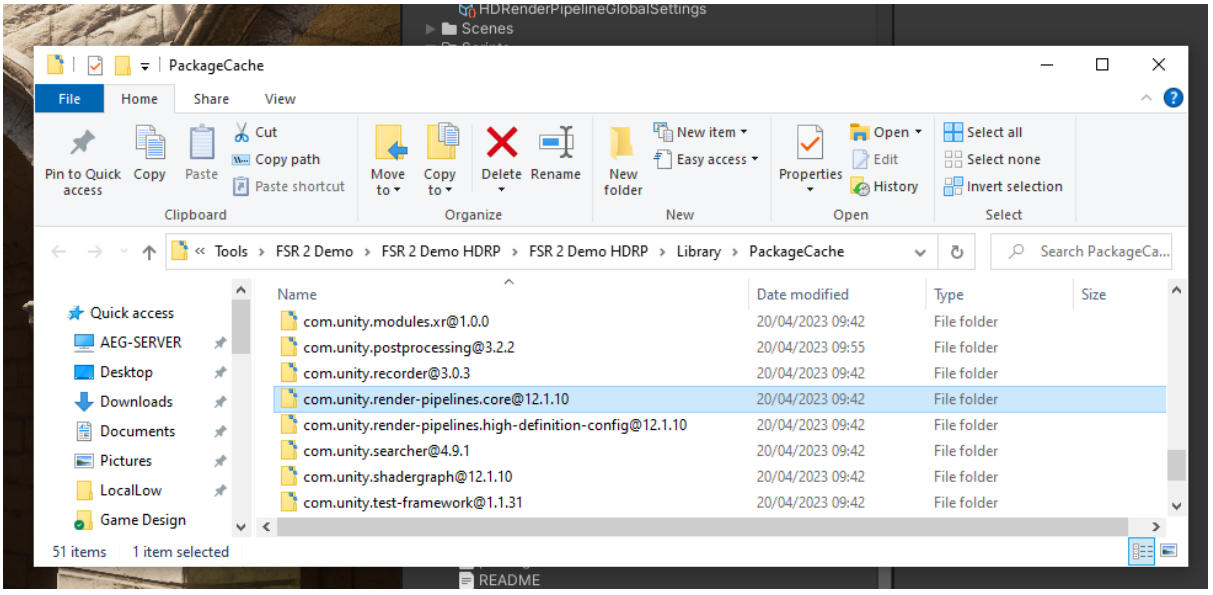
All done, now you should be able to use the single camera setup in BIRP.

Editing Render Pipeline source files

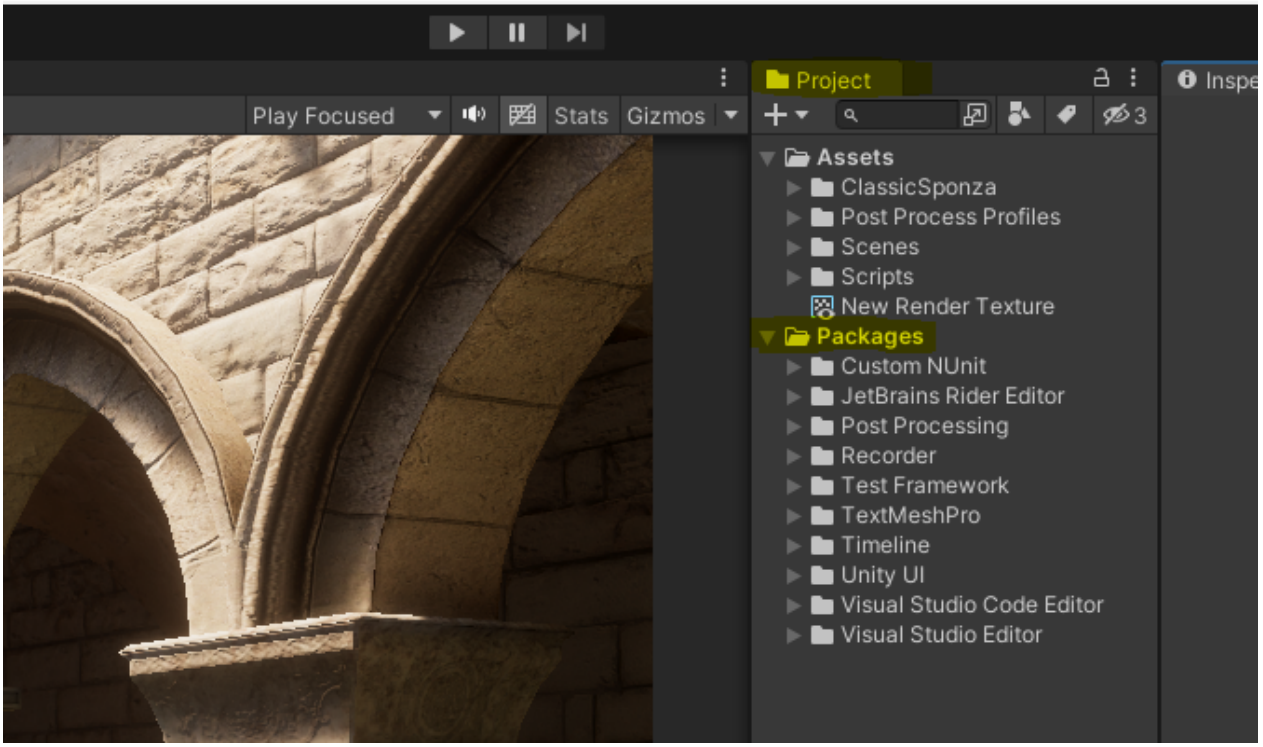
Step 1: Locate the Packages folder in your project, unfold it and press Right-Mouse Button on the “High Definition RP” and select “Show in Explorer”



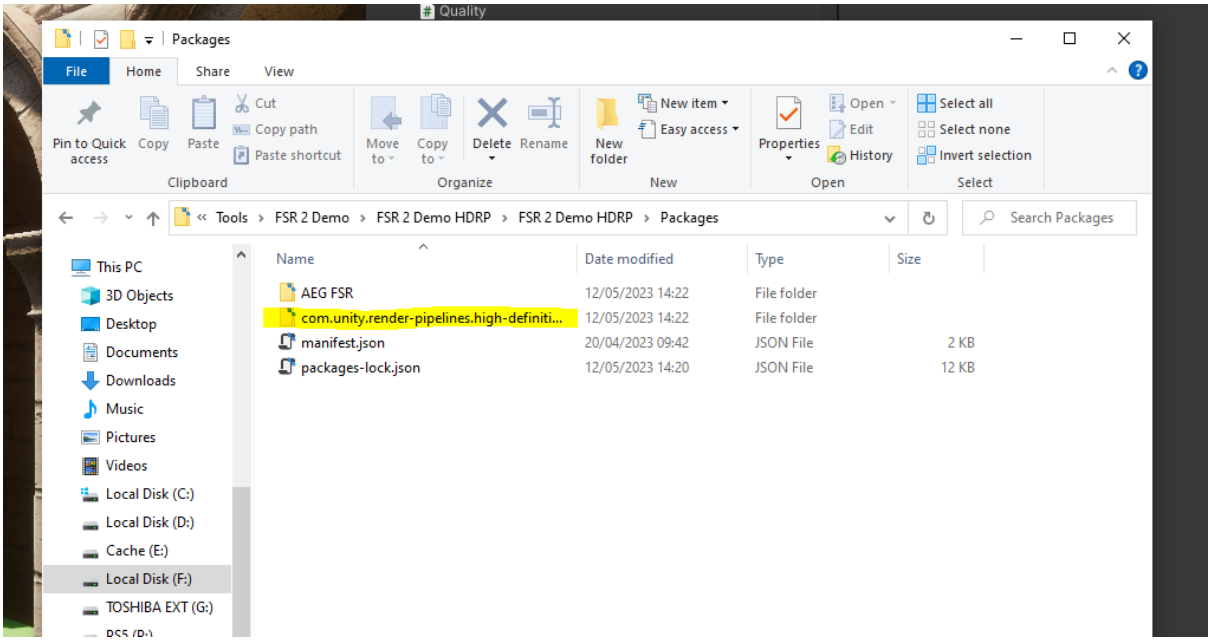
Step 2: Copy the com.unity.render-pipeline.core@xx.x.xx



Step 3: Locate the Packages folder in your project, press Right-Mouse Button on it and select “Show in Explorer”



Step 4: And paste the previously copied folder into this Packages folder!

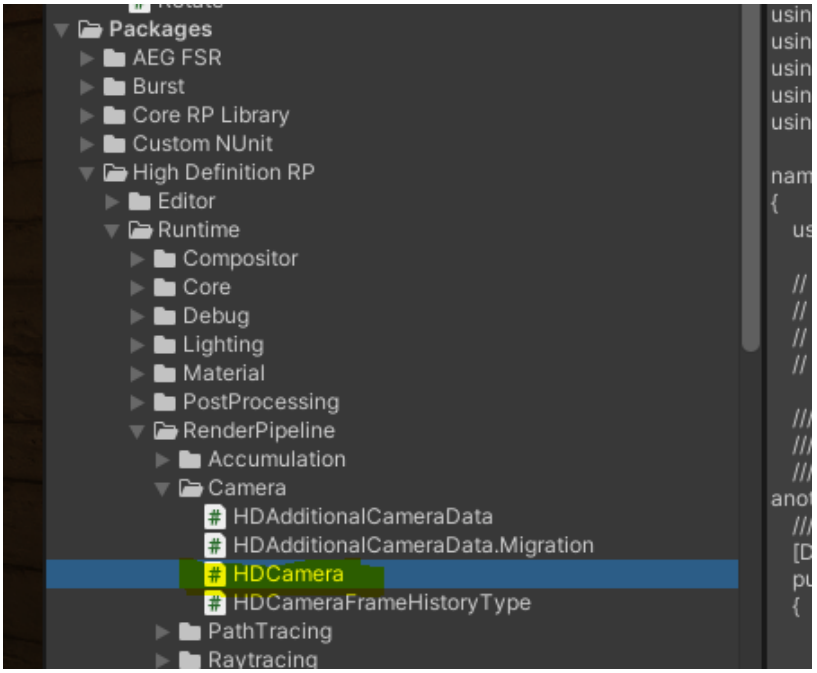


Now the source files of the High-Definition Render Pipeline are editable!

HDRP source files

To make FSR 2 work in HDRP we need to edit two files. If you are unsure on how to edit Render Pipeline Source files, [read about it here](#).

HDCamera.cs



In the script, locate:

```
internal bool UpsampleHappensBeforePost()
{
    return IsDLSSEnabled() || IsTAAUEnabled();
}
```

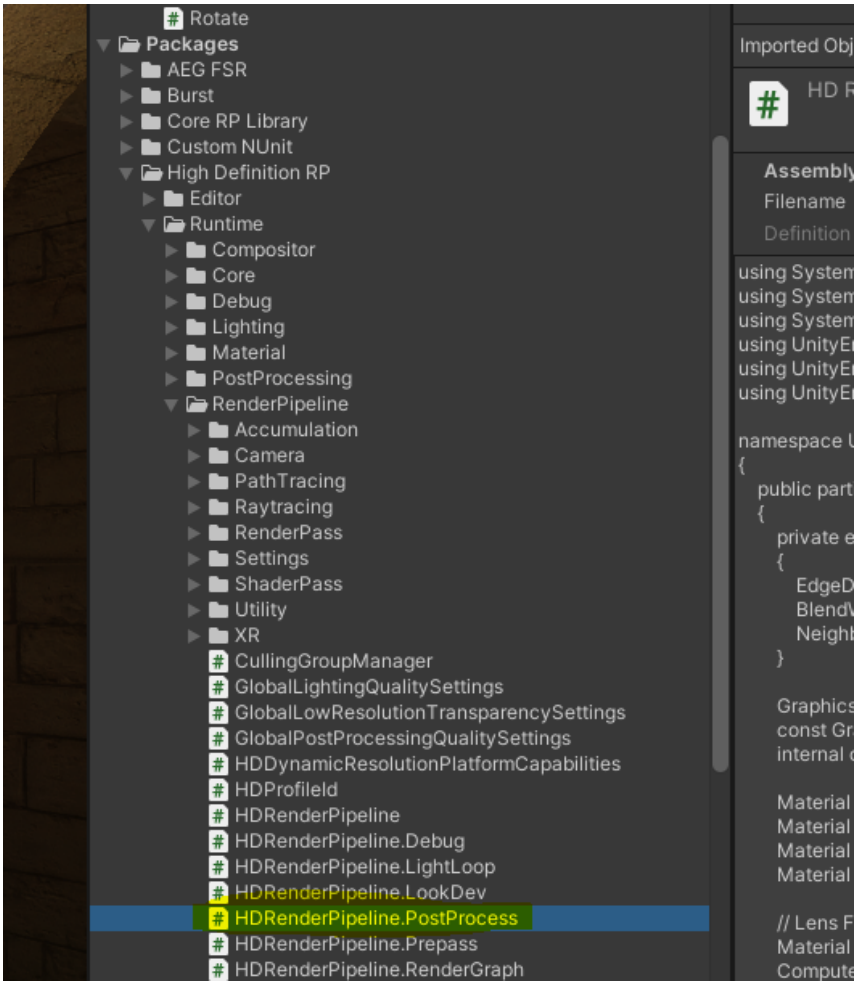
And replace the whole function with

```
public bool fsr2Enabled = false;

internal bool IsFSR2Enabled()
{
    return fsr2Enabled;
}

internal bool UpsampleHappensBeforePost()
{
    return IsFSR2Enabled() || IsDLSSEnabled() || IsTAAUEnabled();
}
```

HDRenderPipeline.PostProcess



In the script locate:

```
source = CustomPostProcessPass(renderGraph, hdCamera, source,
depthBuffer, normalBuffer, motionVectors,
m_GlobalSettings.beforeTAACustomPostProcesses,
HDProfileId.CustomPostProcessBeforeTAA);
```

And replace the line with:

```
source = CustomPostProcessPass(renderGraph, hdCamera, source,
depthBuffer, normalBuffer, motionVectors,
m_GlobalSettings.beforeTAACustomPostProcesses,
HDProfileId.CustomPostProcessBeforeTAA);
if (hdCamera.IsFSR2Enabled())
{
    SetCurrentResolutionGroup(renderGraph, hdCamera,
ResolutionGroup.AfterDynamicResUpscale);
}
```

Known Issues & Limitations

BIRP

- To make Unity's Post Processing Stack work correctly we need source changes which can be downloaded here.
- Ideally most Post Processing is rendered before FSR 2, due to the nature of BIRP, this is not possible as of yet. In future updates this might be added for better quality Post Processing.

URP

- Ideally most Post Processing is rendered before FSR 2, due to the nature of each different URP version, this is not possible as of yet. In future updates this might be added for better quality Post Processing.

HDRP

- Requires changes to the source of the HDRP package.

FAQ

Q: Does FSR 2 offer free performance?
A: Almost, in many cases it does. However FSR 2 does use a small bit of GPU performance, so when using Quality mode the gains can be little. However if you project is CPU bound, you will not likely see much performance gains, just a lower GPU usage.

Q: My screen is black!
A: Disable HDR in the FSR settings.

Q: When I disable the "Post Processing Camera", I lose my Post Processing effects.
A: Make sure you are using our custom Unity Post Processing package, or disable the "Post Process Camera" boolean.

Q: When using FSR, some UI elements are in a different location than they should be.
A: Check out chapter BIRP Camera ScreenPointToRay & WorldToScreenPoint

Q: Your FSR asset gives profiling spikes, yikes!
A: Unfortunately to make FSR 2 work stable in the Unity Editor, we need to do some additional updates which cost a bit of CPU power. Not to worry though, in a build we don't do this!

Q: My Vegetation is blurry when using FSR 2.
A: This is an issue that the vegetation shaders do not add their vertex displacement (movement) to the motion vectors. Unfortunately this is an issue that should be fixed by the creators of the vegetation. Assets like the "Nature Renderer" or "Nature Shaders" have this functionality.

Q: FSR 2 makes my game look jittery.
A: Try changing the Reactive Mask values, this should eliminate the jitters. A small jitter is expected behaviour on lower quality settings, just like normal TAA.

Q: FSR 2 flips out when adding the FSR script to a another camera
A: FSR 2 Upscaling currently only supports 1 upscaled camera!

Q: Will this asset also work on Linux and MacOS?
A: Not yet, but we're planning to add Vulkan and Metal support.

Q: FSR 2 is amazing!
A: Indeed FSR 2 is a great upscaling technique, thank AMD for making it publicly available and do leave us a review!

Support

If you encounter any issues, you can contact us by emailing to info@alteregogames.com, joining our discord in the “Unity Tools” channel or on the Unity Forum

Wishlist

- Multiple upscaling camera support for all render pipelines.
- No more HDRP source changes.
- VR Support
- Nvidia Deep Learning Super Sampling (DLSS).
- Intel Arc - Xe Super Sampling (XeSS)

Licence

FSR 2 Upscaling for Unity

AMD FidelityFX Super Resolution 2.2

Special Thanks

Special thanks to Nico de Poel for allowing us to use his FSR 2 backend to make sure our asset supports so many platforms!