SGSR Mobile Upscaling for Unity Documentation

Current version: SGSR 1.0.0

About SGSR

This Unity asset is created to interface with the Qualcomm Snapdragon Game Super Resolution code found here: https://github.com/quic/snapdragon-gsr

Qualcomm Snapdragon Game Super Resolution is an upscaling technique created especially for mobile devices, 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, SGSR 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 mobile device will use way less battery power when using SGSR!

Current supported Unity Render Pipelines:

Built-in (BIRP) and Universal Render Pipeline (URP)

Current supported platforms:

- iOS (Metal)
- Android (Vulkan)
- Nintendo Switch
- Windows (DX11, DX12, Vulkan)
- Linux (Vulkan)
- MacOS (Metal)
- Playstation 4
- Playstation 5
- PCVR (BIRP, URP)
- Standalone VR (BIRP, URP)

Unconfirmed platforms:

- Xbox One
- Xbox Series X|S

Unsupported platforms:

- WebGL (no support for Compute Shaders)

The Naked Dev Tools

Upscaling Tools

Each and every upscaler out there has different strong and weak spots, some require specific hardware, some are specifically made for slow or older devices.

Here is the list of all our other upscalers for Unity:

FSR 3 - Upscaling for Unity

FSR 3 is supported on almost every platform and most hardware! Making it the number #1 goto upscaler. However compared with DLSS or XeSS the visual fidelity of FSR 3 is considered to be slightly lower.

XeSS - Upscaling for Unity

XeSS is limited to Windows x64, DX11 and DX12, but works just as well on Intel, AMD and Nvidia GPUs! XeSS's visual fidelity is higher than FSR 3 and in some ways even better than DLSS.

DLSS - Upscaling for Unity

DLSS is limited to Windows x64, DX11 and DX12, and will only work on Nvidia RTX GPUs (20x0 series and up). DLSS's visual fidelity is higher than FSR 3 and in some ways better than XeSS.

SGSR - Upscaling for Unity

SGSR is supported on almost every platform and most hardware! However SGSR's visual fidelity is a lot lower than the other upscalers. But it's also way faster than all other upscalers, making it perfect to use on platforms with a high DPI like mobile devices!

Fallback Setup

For the absolute best visual upscaling results we recommend using the following fallback format:

1. DLSS
2. XeSS
3. FSR 3
4. SGSR
5. FSR 1

Various Tools

CACAO - Ambient Occlusion for Unity

CACAO is an ambient occlusion technique, created for AAA games, to produce the best possible ambient occlusion visuals while also offering the best performance possible.

Quick Start

This chapter is written to add SGSR as fast as possible to your project. However, it is very much recommended to read the Important Information chapter. SGSR 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

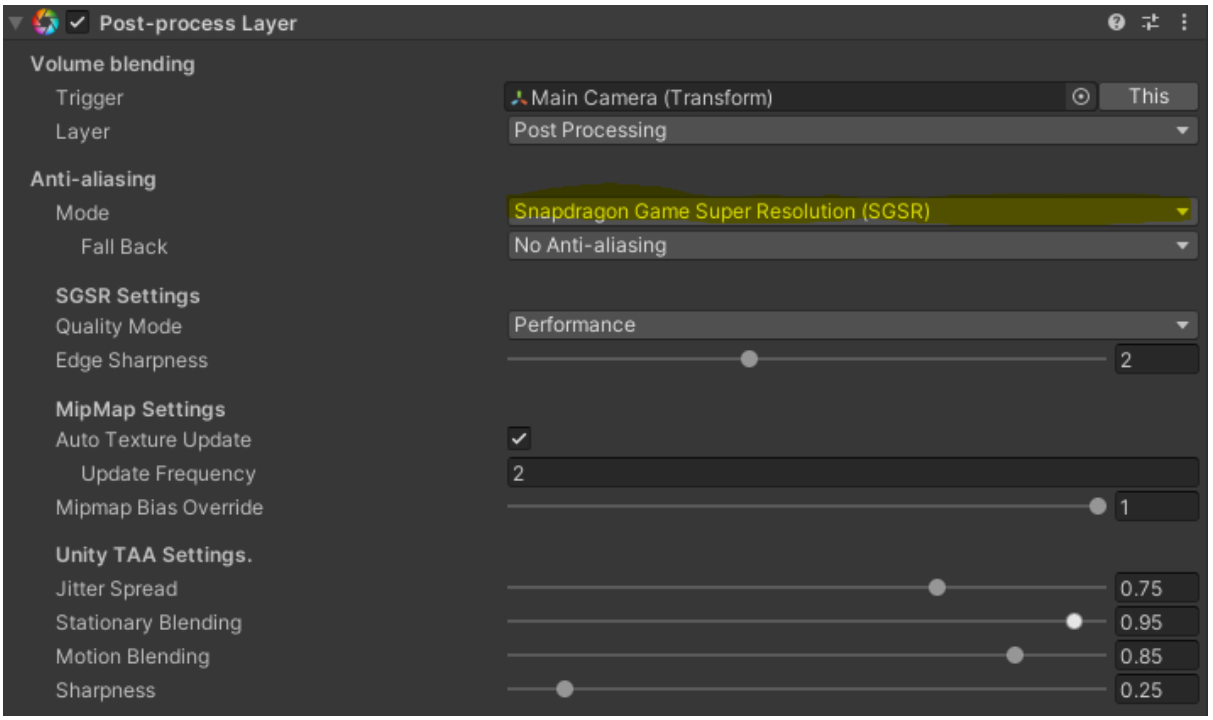
Quick Start: BIRP

Step 1: Import the SGSR Package in your project.

Now there are two options:
NOTE: Don't use both options at the same time!

Option 1: Add SGSR_BIRP.cs script to your main camera.

Option 2 [Recommended]: Import our custom Post-Processing package here and place a Post-process Layer on your main camera then enable SGSR in the Anti-Aliasing settings on the Post-Process Layer.



Hit play!

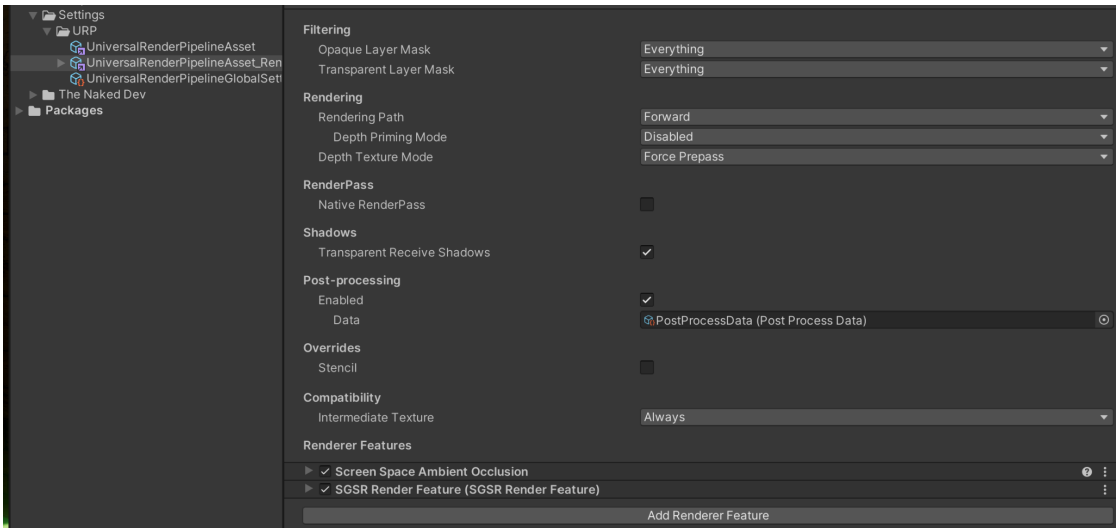
Note: Read more about using **Unity Post Processing**. Read chapter Post-Processing or check out the Demo's for more information.

Quick Start: URP

Step 1: Import the SGSR Package in your project.

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

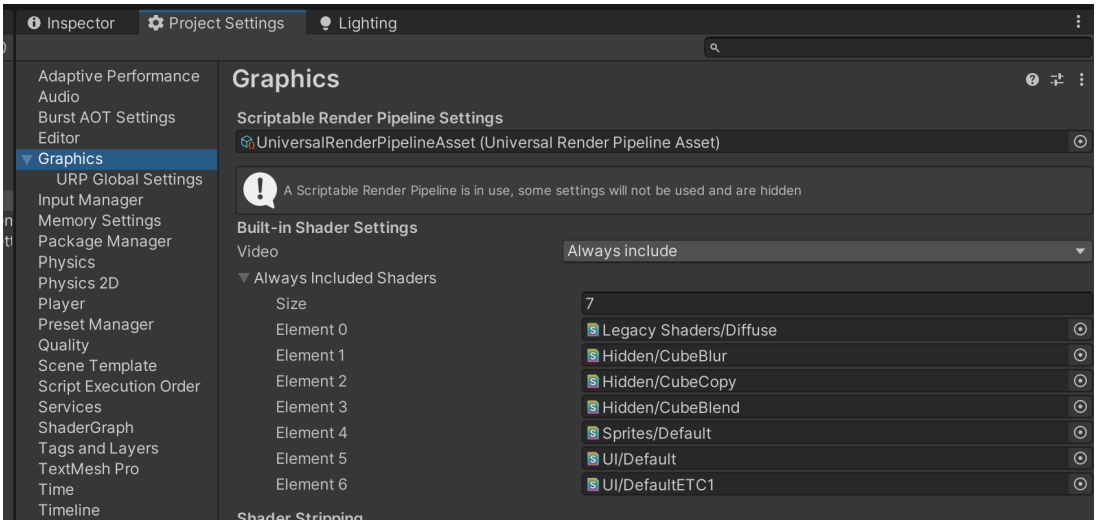
Step 3: Add the “SGSR Scriptable Render Feature” to your Universal Renderer Data files. (Add it to all the URP data files you will use in your project, for example if you use different ones per quality setting).



Step 4 [Optional]: For the best quality settings, use SGSR together with good TAA (Temporal Anti-Aliasing)!

Hit play!

Note: If you can’t add the SGSR_URP component to your Main Camera, make sure you have a Scriptable Render File in the Scriptable Render Pipeline Settings, SGSR uses this to check which render pipeline is active



Demo Scenes

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

[Download Demo Projects here]

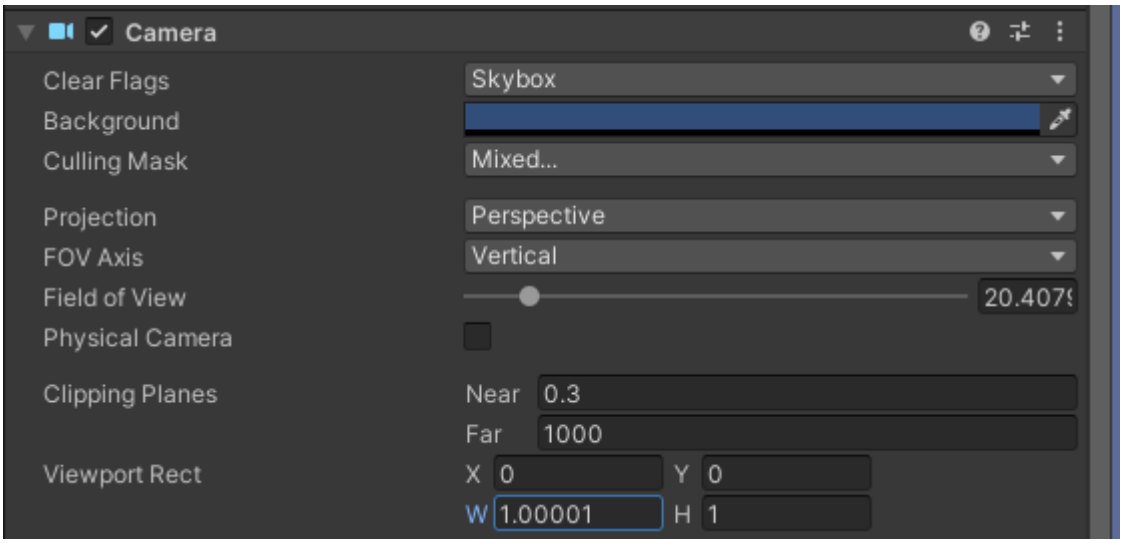
Important Information

Multiple (stacking) Camera's

BIRP

Multiple camera's is only supported when using the custom Post-Processing package. Only one camera can make use of SGSR.

Note: Unity "links" multiple cameras automatically, which is bad because the other cameras will also downscale, making everything look blurry. To prevent this, make sure the "Viewport Rect" values of all cameras are slightly different.

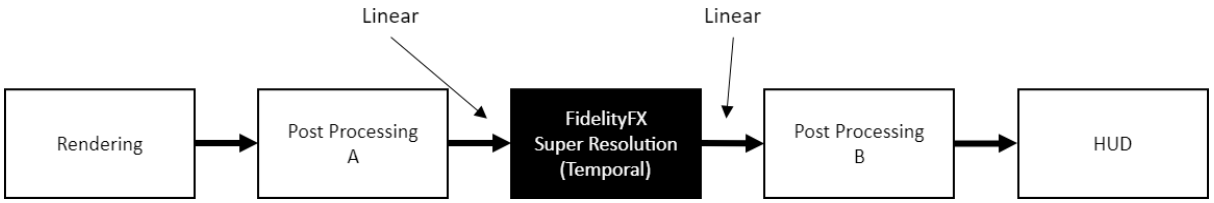


URP

Using multiple Base camera's is not supported, stacking camera's is also not supported!

Post Processing Effects (BIRP)

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

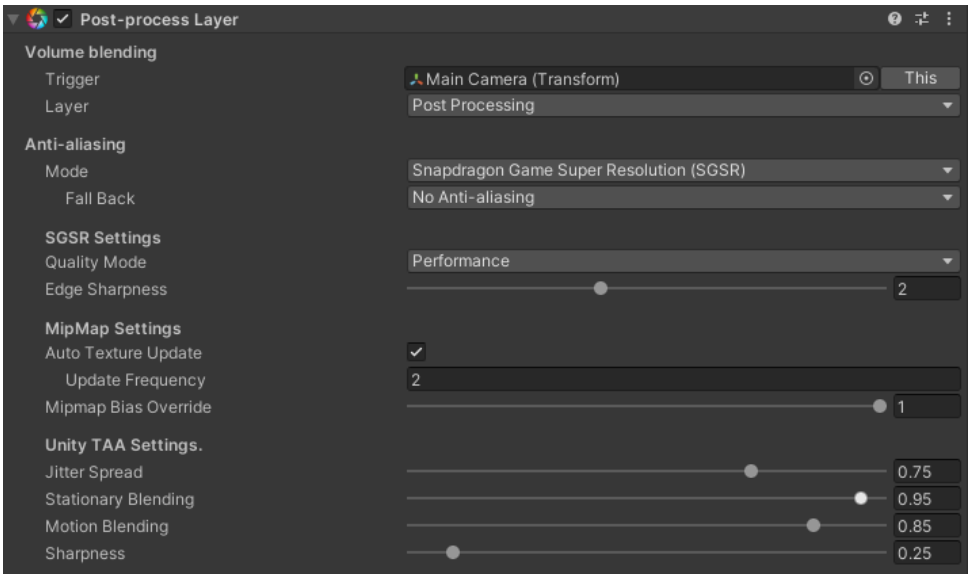


Post Processing order suggestions by AMD:
https://github.com/GPUOpen-Effects/FidelityFX-FSR2#placement-in-the-frame

In BIRP this gives us a bit of an issue.

The camera renders the scene at a lower resolution, which gets upscaled by SGSR, 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 CameraEvent from BeforeImageEffects to AfterForwardAlpha.

For this, we have edited a version of Unity's Post Processing 3.2.3, which you can download here, that fully inserts SGSR to the Post Processing layer allowing it to fully support SGSR.



Inspector

Quality

Off: Disables SGSR.
Native AA: 1.0x scaling (Native, AA only!)
Ultra Quality: 1.2x scaling
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 “SGSRScaleFactor” variable.

Edge Sharpness

0.0 - 5.0

Use this setting to make things look a bit softer or sharper based on your preference. A value of 2 is default.

FallBack - BIRP only

The current Anti-Aliasing will be changed to the fallback option when SGSR is not supported.

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 already automatically does 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 SGSR, 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

When using the SGSR_URP.cs camera component, you can call the following API functions on those components. When using the custom PostProcessing package for BIRP, you can change the values of the Post-processing Layer just like you'd normally would when changing settings on it.

Generic

public bool OnIsSupported()
Use this function to check if SGSR is supported on the current hardware. It's recommended to use this function before enabling SGSR.

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.

BIRP Custom Post Processing Layer

To change any of the settings of the Post-process Layer you'll need a reference to it, afterwards you can address the upscaler settings like this:

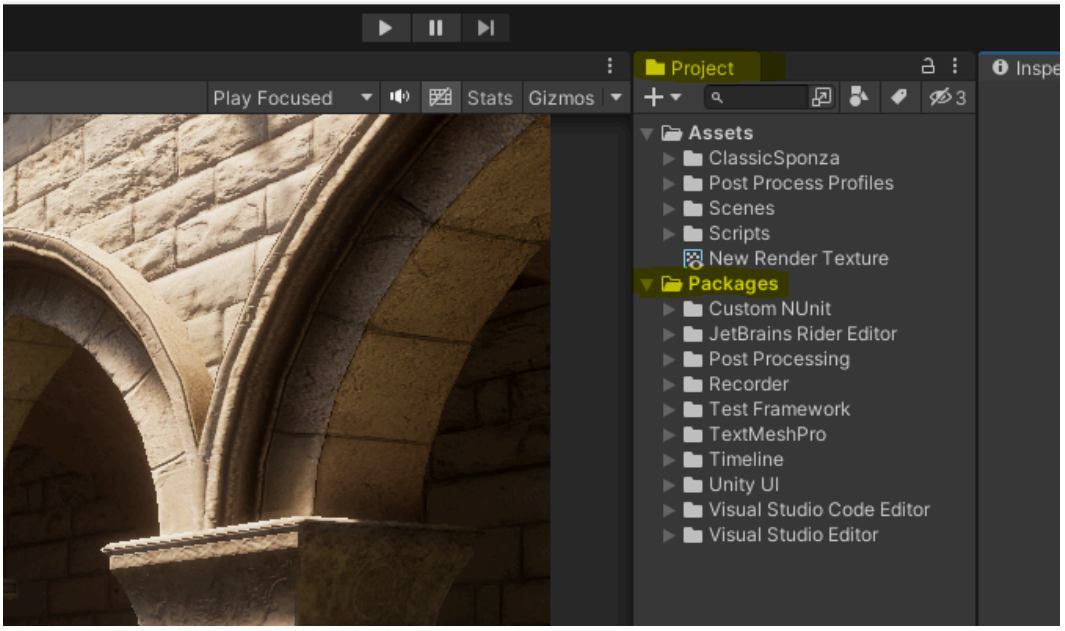
```
using TND.SGSR;

public class ChangeQuality : MonoBehaviour
{
    void Start()
    {
        _postProcessingLayer.sgsr.qualityMode = SGSR_Quality.Quality;
    }
}
```

Editing Unity Post Processing package

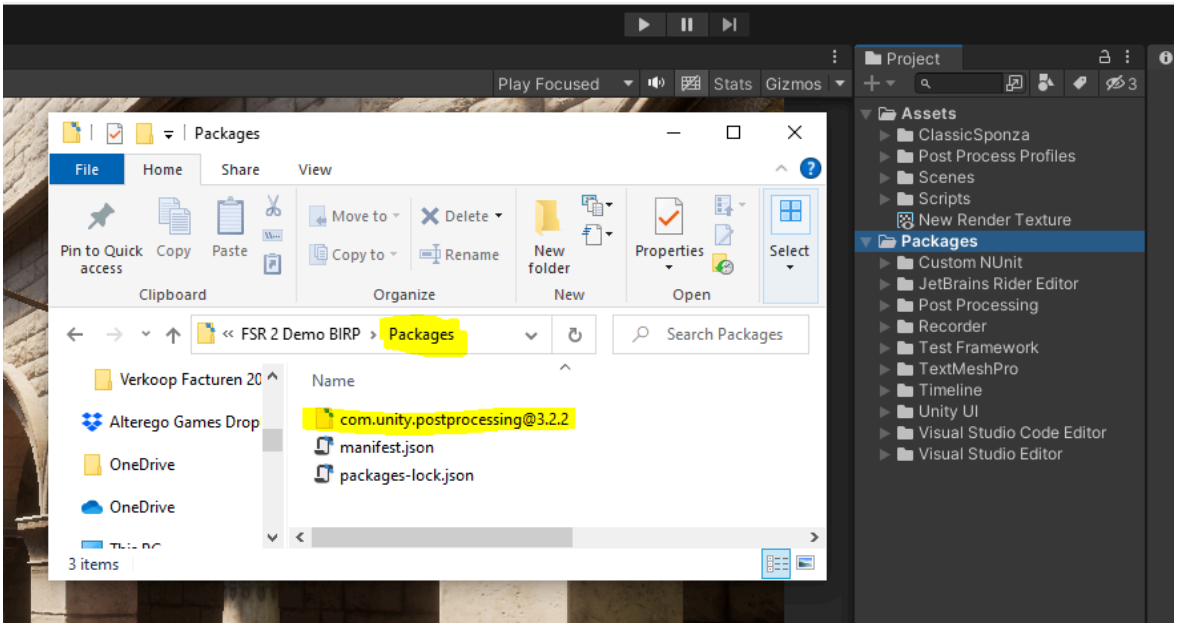
Download

First of all, download our edited Unity Post Processing package:Download Link or add https://github.com/DominicdeGraaf/Unity-Post-Processing-Stack.git to the Unity Package Manager.
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.

VR

The latest update of SGSR Mobile - Upscaling for mobile has added support for PCVR and Standalone VR for BIRP and URP. If you do run into any issues, please get in contact!

Current Limitations:

For now only Multi-pass is supported. Single-pass support is still under investigation.

FAQ

Q: Does SGSR offer free performance?
A: Almost, in almost all cases it does. SGSR is specifically engineered to be a very, very lightweight upscaler. However if your project is CPU bound, you will not likely see much performance gains, just a lower GPU usage.

Q: Will SGSR work for every kind of project?
A: No, in projects that are CPU bound, SGSR will only make sure the GPU has to draw less power.

Q: SGSR is not working or I am having an issue, help!
A: If you encounter any issues, you can contact us by emailing to info@thenakeddev.com, joining our discord in the “Unity Tools” channel or on the Unity Forum.

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

Q: SGSR is amazing!
A: Indeed SGSR is a great upscaling technique. It's not as good as FSR 3 or DLSS in visual fidelity, but it's better than FSR 1 in both visual fidelity and performance!

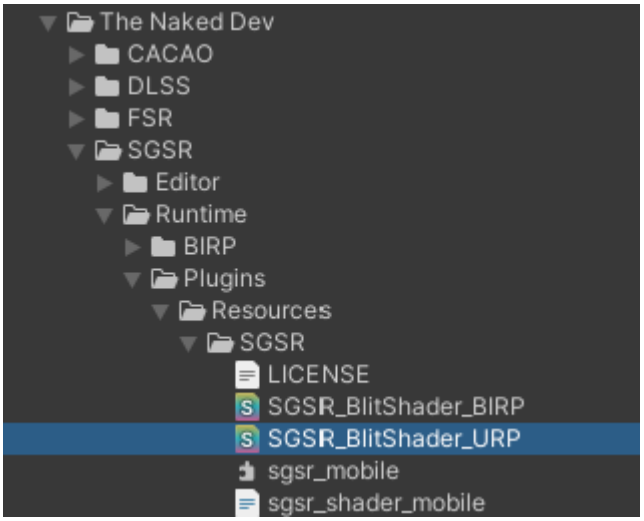
Known Issues & Limitations

General

-

BIRP

- Sometimes the shader files for URP and HDRP will interfere with the building process. When this happens, it is safe to delete them.
-



URP

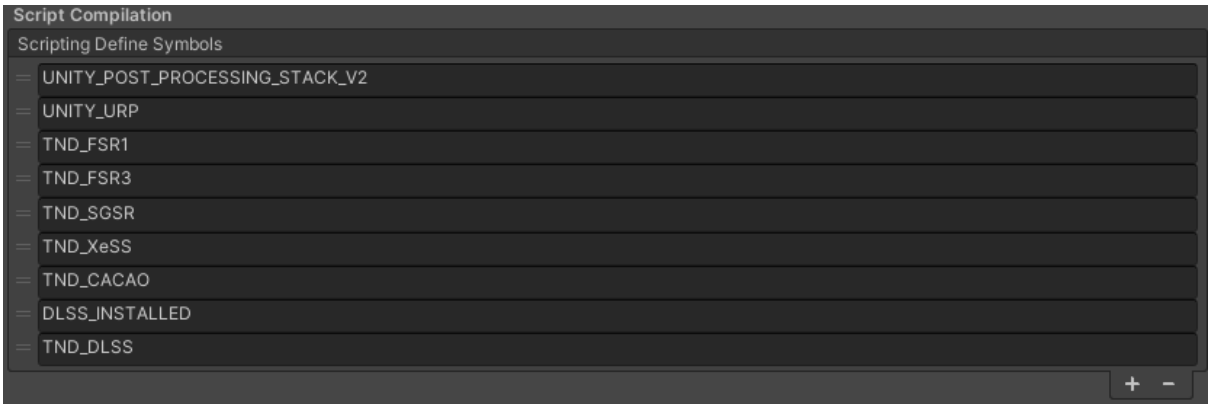
- Unity 2022.1 is not supported. **Older and newer versions are!**

Uninstall

Step 1: Delete the “SGSR” folder in“The Naked Dev” Folder

Optional Step 2 (BIRP ONLY): Delete the Custom Post-Processing Package folder from the “Packages” folder

Optional Step 3: If you are still getting compile errors after deleting the asset, go to the Scripting Define Symbols in the Player settings and manually delete the define symbols starting with “**TND_**” and “**UNITY_URP/UNITY_BIRP/UNITY_HDRP**”.



If you are still getting compile errors, make sure you are not referencing the upscaling asset in a component of your own.

Support

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

Wishlist

- VR Support

Licence

SGSR Mobile - Upscaling for Unity