The Naked Dev

TND Upscaling Framework for Unity Documentation

Live documentation: https://docs.google.com/document/d/1X4ayGIDx-7bRk4p_B4vmJNvS9W0UDUuAmkxtlerajwY

About Upscalers

Upscalers are advanced techniques that generate high-quality, high-resolution frames from lower-resolution input. By using them, projects can significantly reduce GPU requirements.

If your project is GPU-bound, our upscalers can help you achieve a higher framerate. If it's CPU-bound, they reduce the GPU workload. While this might seem like a limitation, it also means laptops and mobile devices will consume significantly less battery power.

Note: Some upscalers have specific hardware or platform requirements.

Supported Unity Render Pipelines

All of our Upscalers fully support Built-in (**BIRP**), Universal Render Pipeline (**URP**) and High-Definition Render Pipeline (**HDRP**).

Supported Unity Versions

All of our upscalers are compatible with **Unity 2021 LTS**, **2022, 2023**, as well as **Unity 6, 6.1** and **6.2**.

Note: While our Upscaling Framework is likely to work with most newer alpha and beta versions of Unity, these versions are not officially supported.

Hardware Limitations

Most of our upscalers run on all GPU brands and on every platform supported by Unity. However, there are exceptions, detailed here.

Support

Because of the advanced nature of these assets, we offer first-class support through **e-mail** or live on **discord** during our office hours between 10:30 and 18:00 CEST.

The Naked Dev Tools

Upscaling Tools

Every upscaler has its own strengths and weaknesses. Some require specific hardware, while others are designed for slower or older devices.

Below is a list of all other upscalers we offer for Unity:

FSR 3 - Upscaling for Unity

FSR 3 works on most platforms and hardware, making it the top choice for compatibility. However, its visual quality is slightly below that of DLSS 4 or XeSS 2.

XeSS 2 - Upscaling for Unity

XeSS 2 runs on Windows x64 with DX11/DX12 and supports Intel, AMD, and Nvidia GPUs. Its visual quality surpasses FSR 3 and can even rival DLSS 4 in some cases.

DLSS 4 - Upscaling for Unity

DLSS 4 supports Windows x64 with DX11/DX12 and works only on Nvidia RTX GPUs (20-series and newer). Its visual quality is higher than FSR 3 and, in some cases, better than XeSS.

SGSR 1 - Upscaling for Unity

SGSR 1 runs on nearly all platforms and hardware. Its visual quality is lower than other upscalers, but it's significantly faster, making it ideal for high-DPI platforms like mobile devices.

SGSR 2 - Upscaling for Unity

SGSR 2 is supported on nearly all platforms and hardware. It delivers much better visual quality than SGSR 1 with only slightly higher GPU cost. A fragment shader version is also available, enabling support for even older hardware, perfect for mobile devices like Android, iOS, and Nintendo Switch.

Graphics Tools

CACAO - Ambient Occlusion for Unity

Coming Soon! - 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.

Game-dev Tools

Markers!

Markers! is the easiest way to add UI screen markers to your game. Whether it is quest markers, objective indicators or waypoint icons, this tool makes it quick and intuitive. Plug and play, fully customizable and perfect for both 2D and 3D games. No setup hassle. Just mark it.

About Upscalers **2**
Supported Unity Render Pipelines 2
Supported Unity Versions 2
Hardware Limitations 2
Support 2
The Naked Dev Tools **3**
Upscaling Tools 3
FSR 3 - Upscaling for Unity 3
XeSS 2 - Upscaling for Unity 3
DLSS 4 - Upscaling for Unity 3
SGSR 1 - Upscaling for Unity 3
SGSR 2 - Upscaling for Unity 3
Graphics Tools 3
CACAO - Ambient Occlusion for Unity 3
Game-dev Tools 3
Markers! 3
Quick Start **6**
Renderpipeline Specifics **7**
Built-in Render Pipeline (BIRP) 7
Custom TND Post-Processing Stack v2 7
Universal Render Pipeline (URP) 7
Rendergraph 7
High-Definition Render Pipeline (HDRP) 7
Stock Unity DLSS 7
Important Information **8**
“My Performance has not improved!” 8
Multiple & Stacking Camera’s 8
Motion Vectors 8
Demo Scenes **9**
Hardware & Platform Requirements **10**
Upscaler Specific limitations 10
AMD FidelityFX Super Resolution - FSR 3 10
Intel Xe Super Sampling - XeSS 2 10
Nvidia Deep Learning Super Sampling - DLSS 4 10
Qualcomm Snapdragon Game Super Resolution 1 - SGSR 1 10
Qualcomm Snapdragon Game Super Resolution 2 - SGSR 2 10
Inspector **11**
Upscalers by Priority 11
Recommended Fallback Setup 11
Quality 12

Sharpening 12
Sharpness 12
Show Expert Settings 12
Auto Generate Reactive Mask 12
Reactive Scale 13
Reactive Threshold 13
Reactive Binary Value 13
Flags 13
Auto Texture Update - BIRP only 13
Mip Map Update Frequency 14
Mipmap Bias Override 14
Upscaler Specific Settings 14
FSR 3 14
Transparency and Composition Mask 14
Enable Debug View 14
XeSS 14
DLSS 4 14
SGSR 1 15
SGSR 2 15
Shader Quality Variant 15
Public API 16
public bool SetUpscaler(UpscalerName value) 16
public Upscaler Name GetSelectedUpscaler() 16
public Upscaler Name GetActiveUpscaler() 16
public void SetQuality(UpscalerQuality value) 16
public UpscalerQuality GetQuality() 17
public void SetSharpening(bool value) 17
public bool GetSharpening() 17
public void SetSharpness(float value) 17
public float GetSharpness() 17
public void SetAutoReactive(bool value) 17
public bool GetAutoReactive() 17
public UpscalerSettingsBaseGetUpscalerSettings(UpscalerName value) 17
public TSettings GetUpscalerSettings(UpscalerName value) 17
public void ResetCamera() 17
public static List<UpscalerName> GetSupported() 17
public static bool IsSupported() 17
Uninstall 18

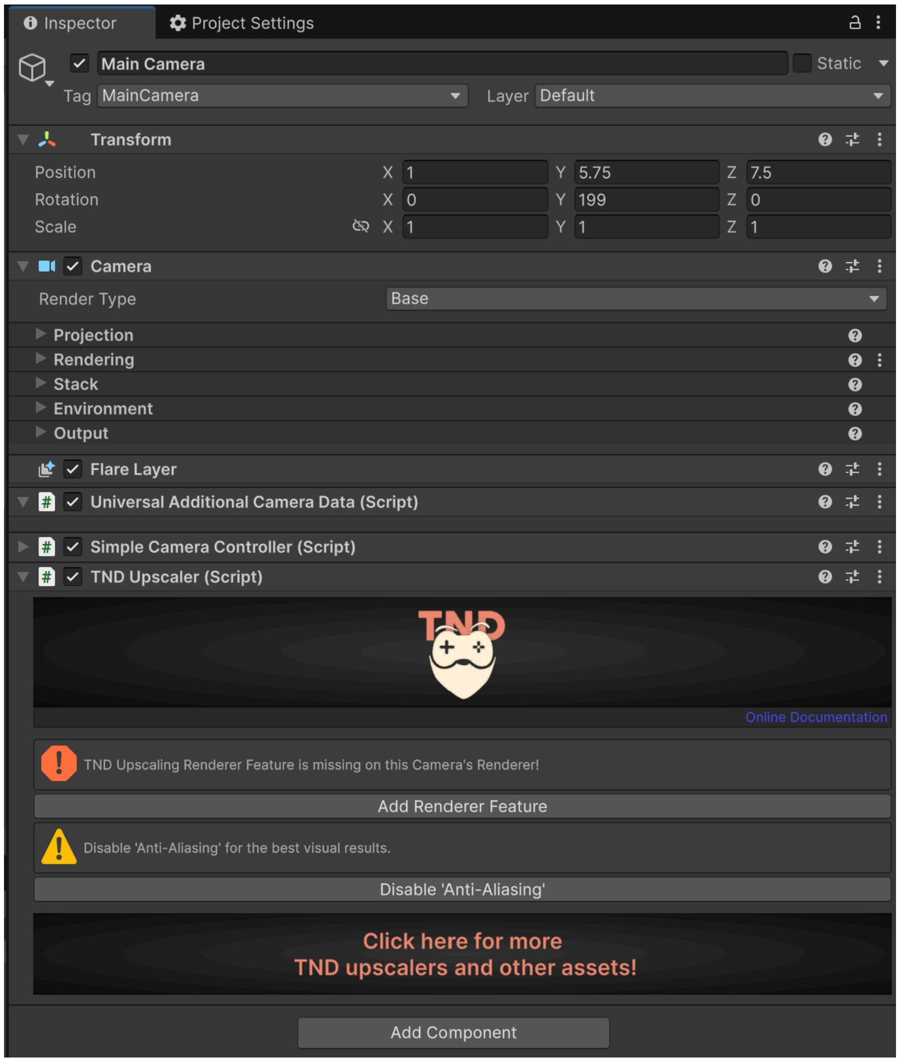
Quick Start

This chapter is designed to help you integrate our upscalers into your project as quickly as possible.
However, we strongly recommend reading the rest of the documentation to fully understand how to optimize overall upscaling quality.

Step 1: Import one of our Upscaler assets from the Unity Package Manager.
Step 2: Add the TNDUpscaler component to your Main Camera.
Step 3: Fix all of the errors shown on the TNDUpscaler component, for the best performance and quality, it is recommended to also make sure you fix the warnings.

Hit play!

Quick Start Video: https://youtu.be/1185xk8MZe0
Demo projects included in Packages/TND Upscaler Framework/Demo/



Main Camera ✓ Static
Tag MainCamera Layer Default

	X	Y	Z
Position	1	5.75	7.5
Rotation	0	199	0
Scale	1	1	1

Render Type Base

TND Upscaling Renderer Feature is missing on this Camera's Renderer!

Disable 'Anti-Aliasing' for the best visual results.

Renderpipeline Specifics

Built-in Render Pipeline (BIRP)

Custom TND Post-Processing Stack v2

TND Upscalers will work out of the box, but for best third-party asset and Post-Processing support we recommend using our upscalers together with our custom Post-Processing Stack v2, which you can get here.
Without it, it is possible that some Post-Processing effects will jitter (Bloom, Depth of Field) or seem to not render at all (Grain). Additionally, third-party assets will most likely be better supported with the custom TND Post-Processing Stack v2.

Universal Render Pipeline (URP)

Rendergraph

TND upscalers fully support Unity 6's Rendergraph, with or without 'compatibility mode'.

High-Definition Render Pipeline (HDRP)

Stock Unity DLSS

Unity already comes with DLSS in HDRP, so why should you still get our DLSS plugin? Obviously you're not forced to, our framework automatically adds Unity's DLSS if you're on HDRP. However, our TND DLSS offers quite a few more features like custom Render Presets and more!

Important Information

“My Performance has not improved!”

If you don’t see any increased performance when using our upscalers, please take the following steps.

Step 1: Always test performance in a release build.
Create a normal (non-development) build of your project and measure the frame rate there.
Testing in the editor or in a development build does not give accurate results.

Still not seeing a performance boost?

Step 2: Check if your project is CPU or GPU limited.
Create a normal build without any upscaler and check the GPU usage using a tool like MSI Afterburner or Windows Task Manager.
If GPU usage is below 99–100%, your project is likely CPU limited on your hardware. In that case, an upscaler will have little or no effect.

Step 3: Make sure the upscaler is working
Create another build with the upscaler enabled and set it to **Ultra Performance**.
If GPU usage is now much lower than before, the upscaler is working correctly.
To get more benefit from the upscaler, consider optimizing the CPU side of your project first.

Multiple & Stacking Camera’s

Unity has officially **not** added support for multiple or stacked cameras with their in-house upscaler (STP).
We also recommend avoiding the use of multiple cameras in Unity, as this introduces significant CPU overhead, which can reduce the effectiveness of any upscaling solution.

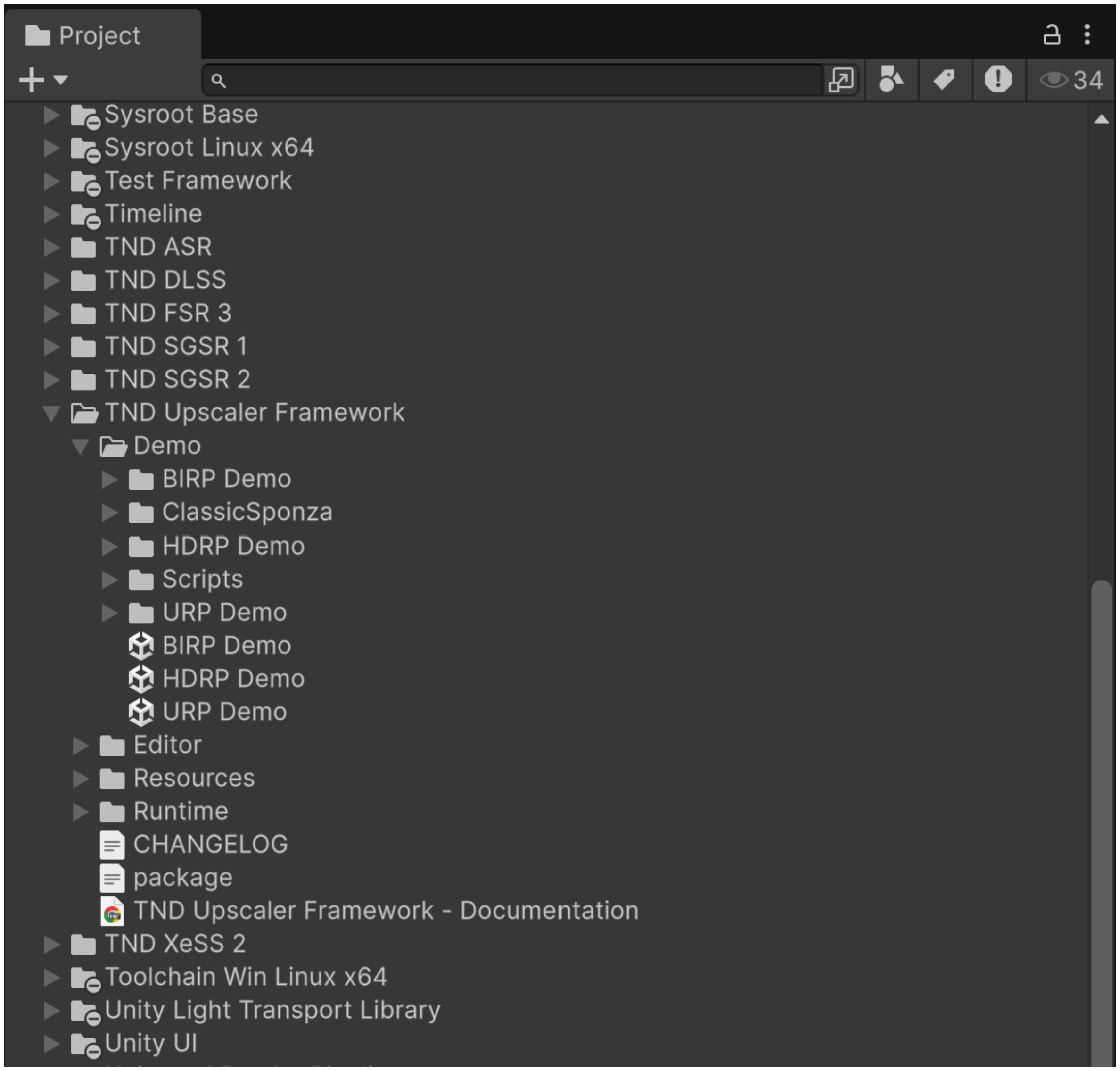
However, since using multiple cameras remains a popular development approach in Unity, we’re actively working on adding support for this feature and in cases it may already work.

Motion Vectors

Most upscalers heavily rely on motion vectors. If you notice visual artifacts such as ghosting or trailing, it’s likely that motion vectors are disabled, either in the rendering asset settings (URP or HDRP) or on individual Renderers.

Demo Scenes

We’ve included demo scenes for each render pipeline, allowing you to test our upscaling framework before integrating it into your own project.
You can find them under: **“Packages/TND Upscaler Framework/Demo/”**



```
▶ Sysroot Base
▶ Sysroot Linux x64
▶ Test Framework
▶ Timeline
▶ TND ASR
▶ TND DLSS
▶ TND FSR 3
▶ TND SGSR 1
▶ TND SGSR 2
▼ TND Upscaler Framework
  ▼ Demo
    ▶ BIRP Demo
    ▶ ClassicSponza
    ▶ HDRP Demo
    ▶ Scripts
    ▶ URP Demo
      BIRP Demo
      HDRP Demo
      URP Demo
  ▶ Editor
  ▶ Resources
  ▶ Runtime
    CHANGELOG
    package
    TND Upscaler Framework - Documentation
▶ TND XeSS 2
▶ Toolchain Win Linux x64
▶ Unity Light Transport Library
▶ Unity UI
```

Hardware & Platform Requirements

In general, all of our upscalers run on every platform and have no hardware limitations, they are compatible with all GPU brands.

Unless stated otherwise below, our upscalers are compatible with:

- Windows (DX11, DX12, Vulkan, GLCore & OpenGL ES 3.0)
- Linux (Vulkan)
- MacOS (Metal)
- iOS (Metal)
- Android (Vulkan, OpenGL ES 3.0)
- Playstation 4
- Playstation 5
- Xbox One
- Xbox Series
- Nintendo Switch 1
- WebGL
- PC VR (OpenXR & OpenVR)
- Standalone VR (Quest)

Note: Even if an upscaler supports a specific platform, it doesn't always guarantee a performance gain. On some older platforms, the upscaler's cost may outweigh its benefits. For such cases, we recommend using SGSR 1 or SGSR 2.

Upscaler Specific limitations

AMD FidelityFX Super Resolution - FSR 3

- Does not support OpenGL ES 3
- Does not support WebGL

Intel Xe Super Sampling - XeSS 2

- Only works on Windows x64 Standalone (DirectX11, DirectX12)

Nvidia Deep Learning Super Sampling - DLSS 4

- Only works on Nvidia RTX GPU's (20x0 and up)
- Only works on Windows x64

Qualcomm Snapdragon Game Super Resolution 1 - SGSR 1

- SGSR 1 has no limitations on hardware or platform.

Qualcomm Snapdragon Game Super Resolution 2 - SGSR 2

- SGSR 2 has no limitations on hardware or platform, though it will automatically fallback to a simpler variant of upscaling on some platforms.

Inspector

This chapter explains all the Inspector settings of our Upscaler Framework.
Everything you see, except for the Expert Settings, can also be configured at runtime via our public API.

Upscalers by Priority

In this list, you can add and (re)order your preferred upscalers.
The Upscaler Framework will attempt to activate the topmost option. If that upscaler isn't supported on the current hardware or platform, it will automatically fall back to the next one in the list, continuing down until a compatible upscaler is found, or none remain.

Note: If this list is left empty, or if none of the listed options are supported and the Upscaler Framework falls back to nothing, rendering will still look downscaled.

Recommended Fallback Setup

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

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

We highly recommend always including **SGSR 2** or **SGSR 1** and setting it as final fallback, as they are supported on all platforms and hardware.

Quality

***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*

If the native resolution is 1920x1080, selecting the ***Performance*** option will lower the rendering resolution to 960x540. This lower-resolution image is then upscaled back to 1920x1080.

Note: When using the Native AA option, expect performance to be less than when using no upscaler.

Sharpening

Off - On

During the process of rendering at a lower resolution and upscaling to a higher output, some loss of visual clarity may occur. To mitigate this, we've included an optional sharpening filter.

Sharpness

0.0 - 1.0

Use this slider to adjust the intensity of the sharpening effect.

Show Expert Settings

Off - On

This setting reveals the expert options. By default, all of these are configured with optimal values for most projects. Some of these options are universal for all upscalers, others are Upscaler specific.

Note: None of the Expert settings are accessible through script.

Auto Generate Reactive Mask

Off - On

In our upscalers, reactivity determines how much influence the current frame has on the final upscaled image. For transparent objects, using a reactive mask is recommended, it tells the upscaler to rely less on historical data and more on the current frame, improving quality in complex visual areas.

Enabling this feature will automatically include everything rendered in Unity's transparency pass in the Reactive Mask.

Reactive Scale
0.0 - 1.0

The difference between color values will be multiplied by this factor before further processing. Higher values make pixels more reactive, the recommended default is 0.9f.

Reactive Threshold
0 - 1.0

When using the Apply Threshold flag, this determines how large the difference between color values has to be for a pixel to be drawn to the reactive mask. Smaller values improve stability at hard edges of translucent objects, the recommended default is 0.05f.

Reactive Binary Value
0 - 1.0

When using the Apply Threshold flag, this is the value that is written to the reactive mask for pixels with a large enough color difference. Larger values increase pixel reactivity, the recommended default is 0.5f.

Flags
Nothing | Everything, Apply Tonemap | Apply Inverse Tonemap | Apply Threshold | Use Components Max

Apply Tonemap: apply a simple HDR tonemap to the color values to get a more accurate comparison.
Apply Inverse Tonemap: apply an inverse tonemap to the color values before comparison.
Apply Threshold: use a cut-off value to determine whether pixels should be drawn to the reactive mask or not. When this is off, the color difference values will be written directly to the reactive mask.
Use Components Max: use the maximum difference of each individual color channel for comparing pixel values. Otherwise, the overall distance between color values is used for comparison.

Auto Texture Update - BIRP only
Off - On

We recommend adjusting the MipMap Bias of all used textures. Since Unity only allows this via script, texture by texture, we've added a feature to automatically update all textures currently loaded in memory. In large real-world and released projects, this had no noticeable performance impact.

Note: In URP and HDRP, mipmap biasing is handled automatically, so this feature is disabled by default.

Mip Map Update Frequency
0.0 - Infinite

This setting controls how often the Auto Texture Update checks for newly loaded textures to apply the MipMap Bias. We recommend a value between 2 and 10 seconds.

Mipmap Bias Override
0.0 - 1.0

When using upscalers in BIRP and adjusting MipMap bias, texture flickering may occur. If flickering is too noticeable, lower this setting to improve stability. When there's no flickering, keep it at 1 for optimal visual quality.

Upscaler Specific Settings
Some upscalers feature **Advanced Parameters** in the inspector. These are not covered in this documentation, as they are highly specialized and rarely applicable.

FSR 3

Transparency and Composition Mask
Texture2D/RenderTexture

In addition to the Reactive Mask, this Upscaler also provides for the application to denote areas of other specialist rendering which should be accounted for during the upscaling process. Examples of such special rendering include areas of raytraced reflections or animated textures.

You'll need to create your own Texture2D or RenderTexture and assign it here.

Enable Debug View
This feature provides a screen overlay displaying debug buffers.

XeSS
No Specific Upscaler Settings

DLSS 4
Render Presets
Here you can change the selected Render Presets per Quality mode. The default mode is different per Quality mode so we highly recommend changing this values to what you prefer.

SGSR 1

Note: Qualcomm suggests combining SGSR 1 with a high quality AA. When using SGSR 1, you should ignore the *Disable Anti-Aliasing* warning of the TND Upscaler component.

Use Edge Direction
Improves visual quality by using the edge direction when calculating the weights. Adds a small additional cost to the shader and might lower wave occupancy for some GPUs.

Edge Threshold
Qualcomm suggests keeping its value as 8.0 for mobile applications but changing it to 4.0 if targeting VR.

Edge Sharpness
By default this value is set to 2, but any number in the range 1.0 - 2.0 can be used.

SGSR 2

Shader Quality Variant

3-Pass Compute (Highest Quality), 2-Pass Compute (Balanced Quality vs Performance), 2-Pass Fragment (Best Performance)

This Upscaler has a few different Shader Options, ranging from Quality, for the best visual quality to Performance for the best possible performance on older hardware.

Note: On Android (OpenGL ES 3) and WebGL, this setting is forced to **2-Pass Fragment** for compatibility reasons.

Public API

To use our public API, simply create a script that references the TNDUpscaler component and call the public methods as described in this chapter.

Example:

```
using TND.Upscaling.Framework;
using UnityEngine;

public class ExampleScript: MonoBehaviour
{
  private TNDUpscaler _upscalerReference;

  void Start(){

    //Getting the list of currently supported Upscalers (NOTE, this is a
static function so doesn’t need the _upscalerReference!
    List<string> supportedUpscalers = TNDUpscaler.GetSupported();

    //Getting the reference to the TND Upscaler component
    _upscalerReference = GetComponent<TNDUpscaler>();

    //Setting the upscaling quality level to Balanced
    _upscalerReference.SetQuality(QualityMode.Balanced);
  }
}
```

Note: Place this script on the Main Camera, next to the TNDUpscaler component.

public bool SetUpscaler(*UpscalerName* value)

Use this method to override the current active upscaler.
Returns if it was successful or not.

public Upscaler Name GetSelectedUpscaler()

Use this method to get the current selected upscaler.

Note: this may differ from GetActiveUpscaler(), as the selected upscaler may not be supported on the current hardware or platform and the fallback system will change to another upscaler.

public Upscaler Name GetActiveUpscaler()

Use this method to get the current active upscaler.

public void SetQuality(*UpscalerQuality* value)

Use this method to adjust the upscaler quality level.

public UpscalerQuality GetQuality()
Use this method to get the current upscaler quality level.

public void SetSharpening(*bool* value)
Use this method to enable or disable sharpening.

public bool GetSharpening()
Use this method to see whether sharpening is on or off.

public void SetSharpness(*float* value)
Use this method to set the sharpening intensity.

public float GetSharpness()
Use this method to get the current sharpness.

public void SetAutoReactive(bool value)
Use this method to toggle Auto Reactive Mask on or off.

public bool GetAutoReactive()
Use this method to see whether Auto Reactive Mask is on or off.

public UpscalerSettingsBaseGetUpscalerSettings(UpscalerName value)
Retrieve the upscaler-specific settings for the given upscaler, cast to a specific type.

public TSettings GetUpscalerSettings(UpscalerName value)
Use this method to retrieve a list of all currently supported upscalers.

public void ResetCamera()
Use this method to reset the camera for the next frame by clearing all buffers from previous frames, preventing visual artifacts.
Use this on camera cuts, or when the camera teleports or moves abruptly.

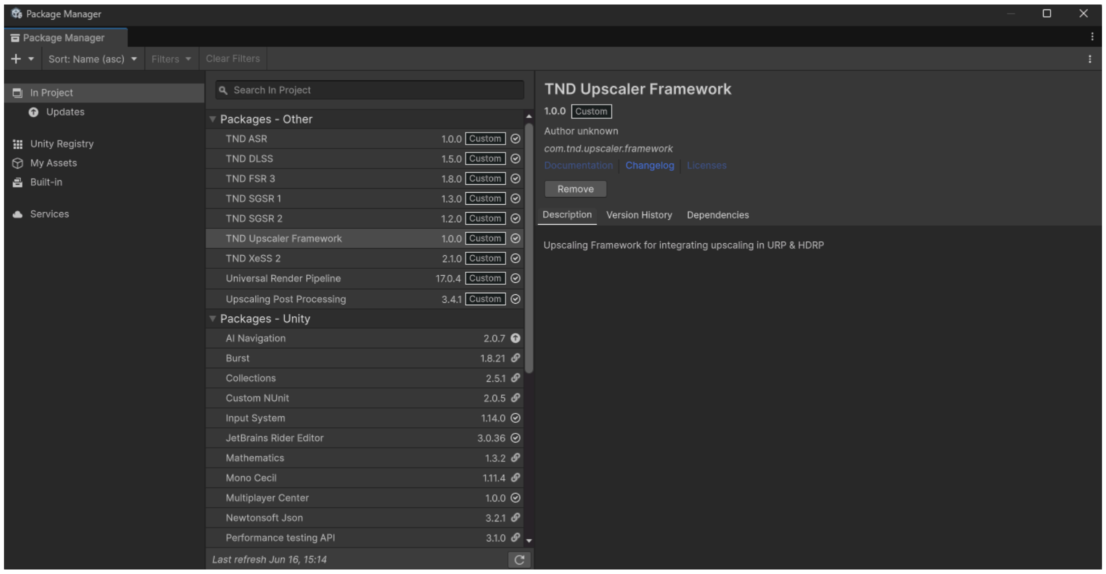
public static List<*UpscalerName*> GetSupported()
Use this method to retrieve a list of all currently supported upscalers. This way you can easily add the currently supported upscalers to your in-game menu!

public static bool IsSupported()
Returns if the given upscaler is available and supported on the current hardware.

Uninstall

Step 1: Open Unity's Package Manager
Step 2: Select the Upscaler you want to remove.
Step 3: Click **Remove**.

[Optional] Step 4: If you want to remove all upscalers **and** the Upscaler Framework, first remove all the upscaler packages (i.e. **TND SGSR 2**, **TND FSR 3**), and lastly remove the **TND Upscaler Framework**.
[Optional] Step 5 BIRP ONLY: Uninstall our custom TND Post-Processing Package v2 package from the Package Manager.



Packages - Other		
TND ASR	1.0.0	Custom
TND DLSS	1.5.0	Custom
TND FSR 3	1.8.0	Custom
TND SGSR 1	1.3.0	Custom
TND SGSR 2	1.2.0	Custom
TND Upscaler Framework	1.0.0	Custom
TND XeSS 2	2.1.0	Custom
Universal Render Pipeline	17.0.4	Custom
Upscaling Post Processing	3.4.1	Custom
Packages - Unity		
AI Navigation	2.0.7	
Burst	1.8.21	
Collections	2.5.1	
Custom NUnit	2.0.5	
Input System	1.14.0	
JetBrains Rider Editor	3.0.36	
Mathematics	1.3.2	
Mono Cecil	1.11.4	
Multiplayer Center	1.0.0	
Newtonsoft Json	3.2.1	
Performance testing API	3.1.0	

Last refresh Jun 16, 15:14

TND Upscaler Framework
1.0.0 Custom
Author unknown
com.tnd.upscaler.framework

Upscaling Framework for integrating upscaling in URP & HDRP