[Public]

FidelityFX Super Resolution 2.0

Unreal Engine Plugin

The AMD FidelityFX Super Resolution 2.0 (FSR2) plugin for Unreal Engine is an open source, high-quality solution for producing high resolution frames from lower resolution inputs.

The package also includes the FSR2MovieRenderPipeline plugin which enables use of FSR2 to accelerate rendering when using the Unreal Movie Render Queue.

Setup

The FSR2 plugin is intended for Unreal Engine 4.26.2* or later.

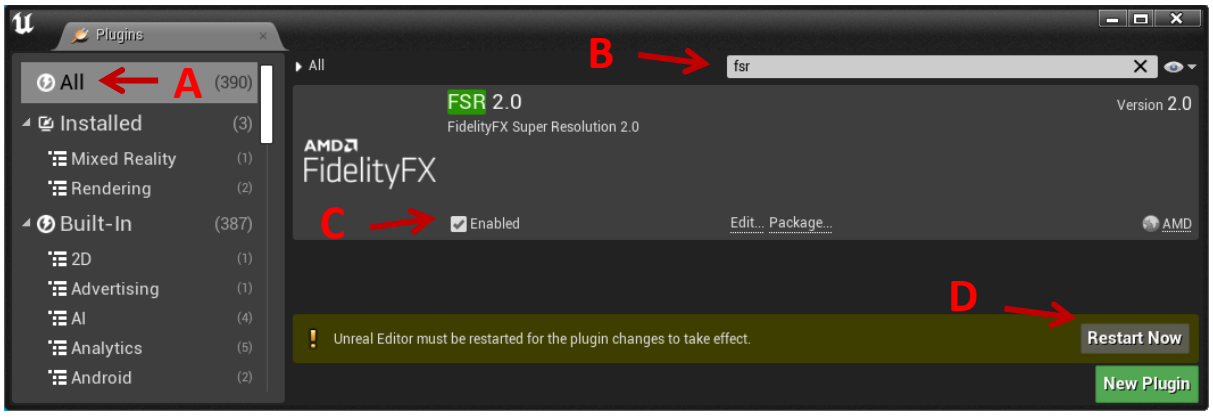
The FSR2MovieRenderPipeline plugin is intended for Unreal Engine 4.27.2* or later.

*If you are not a registered Unreal Engine developer, you will need to follow these instructions to register for access to this link.

Installation Procedure

To install the plugin:

1. Locate the *Engine/Plugins* directory of your Unreal Engine installation.
2. Extract the contents of the FSR2.zip file.
3. Select the sub-folder that corresponds to the Unreal Engine version to be used.
4. Place the **FSR2** folder within your Unreal Engine source tree at: *Engine/Plugins/Runtime/AMD* (for UE4) or *Engine/Plugins/Marketplace* (for UE5)
    a. (Optional) Place the **FSR2MovieRenderPipeline** folder within your Unreal Engine source tree at: *Engine/Plugins/Runtime/AMD*. Only available from Unreal Engine **4.27.2 and later**.
5. Open your Unreal Engine project.
6. Navigate to **Edit > Plugins** in the Unreal Engine toolbar
7. Within the plugin dialog:
    a. Ensure that All is selected on the left side.
    b. Type `fsr` into the search box in the top right corner
    c. Select the **Enabled** checkbox for the **FSR 2.0** plugin.
        i. (Optional) Select the **Enabled** checkbox for the **FSR2MovieRenderPipeline** plugin.
    d. When prompted, click **Restart Now** to apply changes, and restart Unreal Engine.

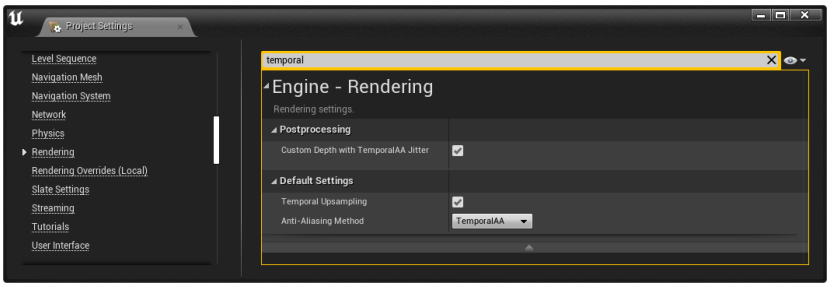


[Public]

Plugin Configuration

Usage

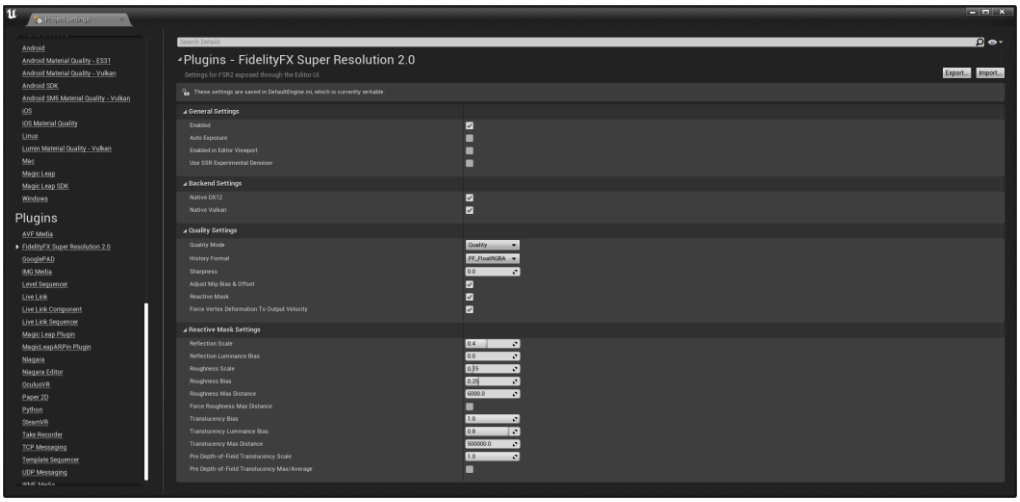
Temporal Upsampling must be enabled in the **Project Settings > Rendering** window, accessed via **Edit > Project** Settings in the Unreal Engine toolbar or via the Console Variable *`r.TemporalAA.Upsampling`*.



FSR 2.0 can be enabled or disabled via the **Enabled** option in the **Project Settings > FidelityFX Super Resolution 2.0** settings window, or with the Console Variable *`r.FidelityFX.FSR2.Enabled`* in the configuration files. The variable can be modified at runtime ***however*** this is not guaranteed to be safe when other third-party upscalers are also enabled.

The plugin will use specific quality modes specified via *`r.FidelityFX.FSR2.QualityMode`* overriding *`r.ScreenPercentage`*. The exposed modes are:

- **Quality (1.5x)**: *`r.FidelityFX.FSR2.QualityMode 1`*
  Provides an image quality equal or superior to native rendering with a significant performance gain.
- **Balanced (1.7x)**: *`r.FidelityFX.FSR2.QualityMode 2`*
  Offers an ideal compromise between image quality and performance gains.
- **Performance (2.0x)**: *`r.FidelityFX.FSR2.QualityMode 3`*
  Provides an image quality similar to native rendering with a major performance gain.
- **Ultra Performance (3.0x)**: *`r.FidelityFX.FSR2.QualityMode 4`*
  Provides the highest performance gain while still maintaining an image quality representative of native rendering.



Integration Instructions

FidelityFX Super Resolution 2.0 contains a built-in sharpening pass called **Robust Contrast Adaptive Sharpening** that can be configured through the CVar *`r.FidelityFX.FSR2.Sharpness`*, this is disabled by default. If your project has already integrated FidelityFX-CAS, it may be necessary to disable FidelityFX CAS - including any in-game menu options - while *`r.FidelityFX.FSR2.Sharpness`* is enabled to prevent over-sharpening your final renders, and improve integration results.

In order for FSR2 to process materials with World Position Offset and/or World Displacement correctly the *`r.VertexDeformationOutputsVelocity`* (UE4) or *`r.Velocity.EnableVertexDeformation`* (UE5) option must be enabled. The *`r.FidelityFX.FSR2.ForceVertexDeformationOutputsVelocity`* setting is enabled by default and when enabled FSR2 will force *`r.VertexDeformationOutputsVelocity`* (UE4) or *`r.Velocity.EnableVertexDeformation`* (UE5) on.

When *`r.FidelityFX.FSR2.CreateReactiveMask`* is enabled the FSR2 plugin forces *`r.SSR.ExperimentalDenoiser`* to 1 in order to capture the Screen Space Reflections, to handle this the initial value of *`r.SSR.ExperimentalDenoiser`* will be applied to *`r.FidelityFX.FSR2.UseSSRExperimentalDenoiser`*. Subsequent changes to the value of *`r.FidelityFX.FSR2.UseSSRExperimentalDenoiser`* will override this.

In order to switch to or from FSR2 to another temporal upscaler always ensure that only one external temporal upscaler is enabled at a time. Disable the current upscaler before enabling the desired upscaler.

Other Configurations

Console Variable	Default Value	Value Range	Details
r.FidelityFX.FSR2.AdjustMipBias	1	0, 1	Applies negative MipBias to material textures, improving results.
r.FidelityFX.FSR2.Sharpness	0	0.0 – 1.0	When greater than 0.0 this enables **Robust Contrast Adaptive Sharpening** Filter to sharpen the output image.
r.FidelityFX.FSR2.AutoExposure	0	0, 1	Set to 1 to use FSR2's own auto-exposure, otherwise the engine's auto-exposure value is used.
r.FidelityFX.FSR2.HistoryFormat	0	0, 1	Selects the bit-depth for the FSR2 history texture format, defaults to `PF_FloatRGBA` but can be set to `PF_FloatR11G11B10` to reduce bandwidth at the expense of quality.
r.FidelityFX.FSR2.CreateReactiveMask	1	0, 1	Enable to generate a mask from the SceneColor, GBuffer, SeparateTranslucency & ScreenspaceReflections that determines how reactive each pixel should be.
r.FidelityFX.FSR2.ReactiveMaskReflectionScale	0.4	0.0 – 1.0	Scales the Unreal engine reflection contribution to the reactive mask, which can be used to control the amount of aliasing on reflective surfaces.
r.FidelityFX.FSR2.ReactiveMaskReflectionLumaBias	0	0.0 – 1.0	Biases the reactive mask by the luminance of the reflection. Use to balance aliasing against ghosting on brightly lit reflective surfaces.
r.FidelityFX.FSR2.ReactiveMaskRoughnessScale	0.15	0.0 – 1.0	Scales the GBuffer roughness to provide a fallback value for the reactive mask when screenspace & planar reflections are disabled or don't affect a pixel.
r.FidelityFX.FSR2.ReactiveMaskRoughnessBias	0.25	0.0 – 1.0	Biases the reactive mask value when screenspace/planar reflections are weak with the GBuffer roughness to account

			for reflection environment captures.
r.FidelityFX.FSR2.ReactiveMaskRoughnessMaxDistance	6000	0.0 – INF	Maximum distance in world units for using material roughness to contribute to the reactive mask, the maximum of this value and View.FurthestReflectionCaptureDistance will be used.
r.FidelityFX.FSR2.ReactiveMaskRoughnessForceMaxDistance	0	0, 1	Enable to force the maximum distance in world units for using material roughness to contribute to the reactive mask rather than using View.FurthestReflectionCaptureDistance.
r.FidelityFX.FSR2.ReactiveMaskTranslucencyBias	1.0	0.0 – 1.0	Scales how much contribution translucency makes to the reactive mask. Higher values will make translucent materials less reactive which can reduce smearing.
r.FidelityFX.FSR2.ReactiveMaskTranslucencyLumaBias	0.8	0.0 – 1.0	Biases the translucency contribution by the luminance of the transparency. Higher values will make bright translucent materials less reactive which can reduce smearing.
r.FidelityFX.FSR2.ReactiveMaskPreDOFTranslucencyScale	1.0	0.0 – 1.0	Scales how much contribution pre-Depth-of-Field translucency color makes to the reactive mask. Higher values will make translucent materials less reactive which can reduce smearing.
r.FidelityFX.FSR2.ReactiveMaskPreDOFTranslucencyMax	0	0, 1	Toggle to determine whether to use the max(SceneColorPostDepthOfField - SceneColorPreDepthOfField) or length(SceneColorPostDepthOfField - SceneColorPreDepthOfField) to determine the contribution of Pre-Depth-of-Field translucency.
r.FidelityFX.FSR2.ReactiveMaskTranslucencyMaxDistance	500000	0.0 – INF	Maximum distance in world units for using translucency to contribute to the reactive mask. This is a way to remove sky-boxes and other back-planes

			from the reactive mask, at the expense of nearer translucency not being reactive.
r.FidelityFX.FSR2.ForceVertexDeformationOutputsVelocity	1	0, 1	Force enables materials with World Position Offset and/or World Displacement to output velocities during velocity pass even when the actor has not moved.
r.FidelityFX.FSR2.UseSSRExperimentalDenoiser	0	0, 1	Enable to use *r.SSR.ExperimentalDenoiser* when FSR2 is enabled. This is required when *r.FidelityFX.FSR2.CreateReactiveMask* is enabled as the FSR2 plugin overrides *r.SSR.ExperimentalDenoiser* in order to capture reflection data to generate the reactive mask.
r.FidelityFX.FSR2.UseNativeDX12	1	0, 1	True to use FSR2's native & optimised D3D12 backend, false to use the fallback implementation based on Unreal's RHI.
r.FidelityFX.FSR2.UseNativeVulkan	1	0, 1	True to use FSR2's native & optimised Vulkan backend, false to use the fallback implementation based on Unreal's RHI.
r.FidelityFX.FSR2.EnabledInEditorViewport	0	0, 1	Enable FidelityFX Super Resolution for Temporal Upscale in the Editor viewport by default.

Recommendations

Optimizing Translucency Appearance

While the default settings for the FSR2 Reactive Mask should generate reasonable results it is important that developers are aware that the appearance can be altered via the *'r.FidelityFX.FSR2.ReactiveMask'* console-variables. Tuning these variables to suit the content may be necessary to optimise visual results.

Translucent Skyboxes & Background Planes

When using a skybox or a distant background plane it is beneficial for this to be rendered with the Opaque or Masked shading model when using FSR2. If these are rendered with the Translucent shading model they will contribute to the FSR2 translucency & reactive masks which can result in unnecessary artefacts. This is especially noticeable when other translucent materials are rendered over the top of the skybox/background-plane and the camera moves. This occurs because the plugin cannot distinguish the purpose of individual translucent materials, they are all treated the same.

To address this issue the FSR2 plugin assumes that a translucent skybox or background-plane is used and will fade out translucency contribution based on reconstructed distance from the camera. This will cut-out all translucency rendered over distant opaque geometry and can be controlled with the *'r.FidelityFX.FSR2.ReactiveMaskTranslucencyMaxDistance'* console variable.

When using an opaque skybox or backplane, adjust the *'r.FidelityFX.FSR2.ReactiveMaskTranslucencyMaxDistance'* console variable to avoid translucency cut-outs.

Static Foliage

For best results with Foliage assets that use World Position Offset and/or World Displacement select Moveable mobility. This ensures the assets will render the necessary motion vectors that are required for FSR2 to produce the best results.

UE4 Post-Processing Volume Screen-Percentage Overrides

For UE4 developers should be aware that when enabled FSR2's quality mode will determine the screen-percentage, ignoring any screen-percentage override present in a post-processing volume. This will result in different visual and performance results.

PIX & RenderDoc Issues

Then the native FSR2 backends are used within the FSR2 plugin graphics capture tools such as PIX & RenderDoc may be less stable. Disabling the native backend *('r.FidelityFX.FSR2.UseNativeDX12'* or *'r.FidelityFX.FSR2.UseNativeVulkan'*) and using the RHI-based backend may allow the capture tool to replay captures more reliably.

Known Issues

Split-Screen

Split-screen is known to render incorrectly in Unreal Engine 4 and to crash with the native DirectX 12 & Vulkan backends in Unreal Engine 5. This will be fixed in a future release.

Separate Translucency Screen Percentage

The current FSR2 plugin release relies on Separate Translucency to generate the reactive mask and assumes that the Separate Translucency texture resolution matches the render resolution. Using *'r.SeparateTranslucencyScreenPercentage'* to specify a different resolution will result in incorrect rendering.

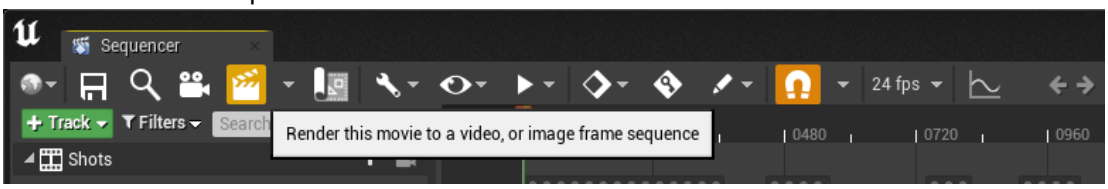
[Public]

Movie Render Pipeline Plugin

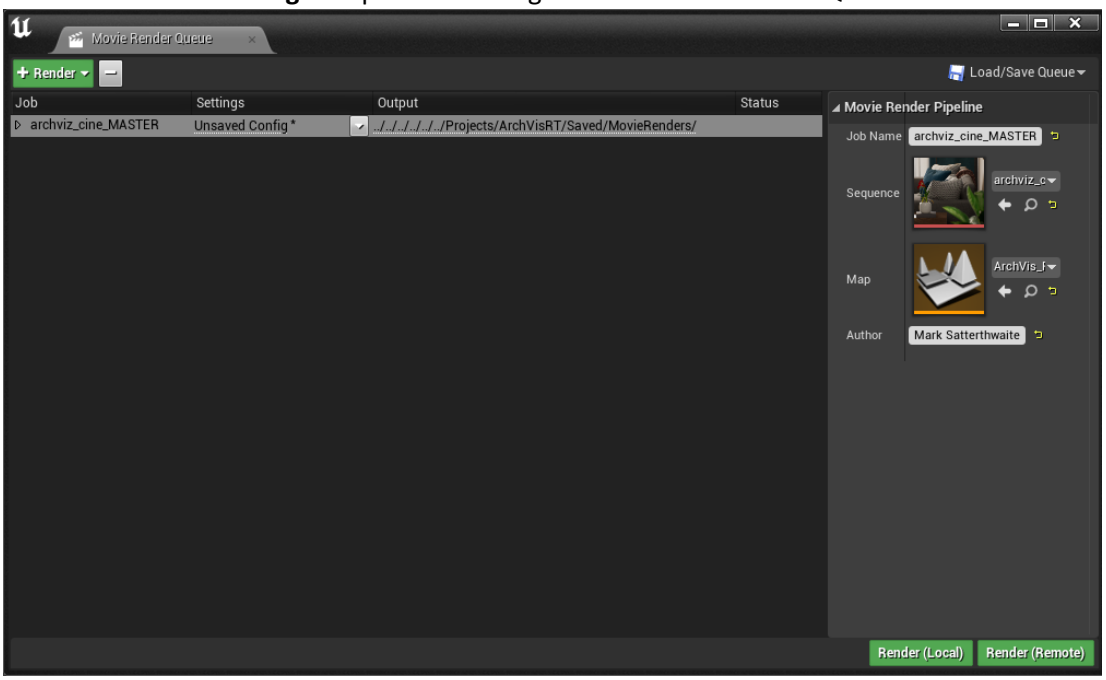
The FSR2MovieRenderPipeline plugin once enabled allows using FSR2 to accelerate rendering of Sequencer cinematics using Unreal's Movie Render Queue.

To use the plugin:

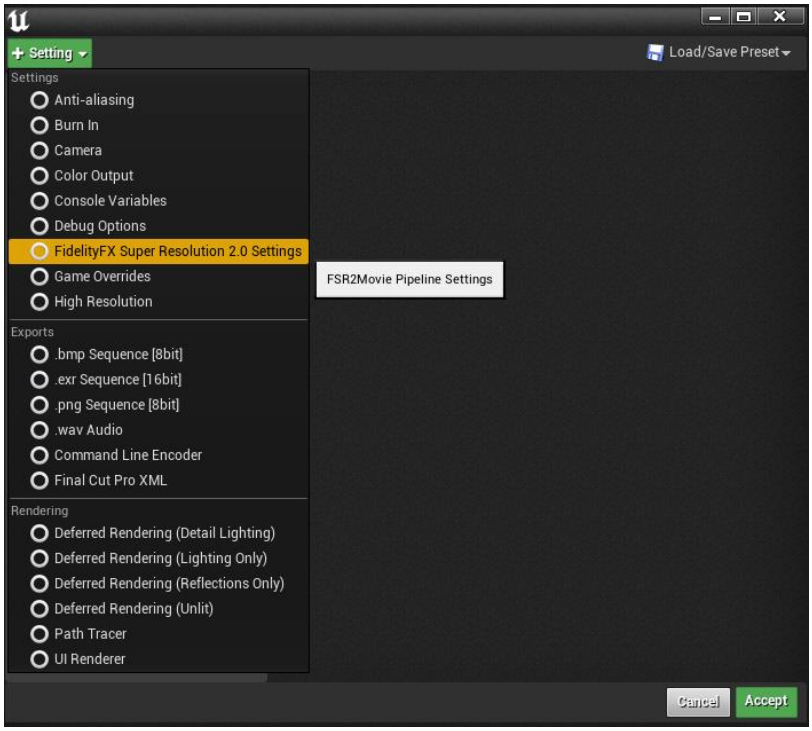
1. Open your Unreal project.
2. Open a **Sequencer** cinematic through the Editor.
3. Select the movie output in the toolbar.



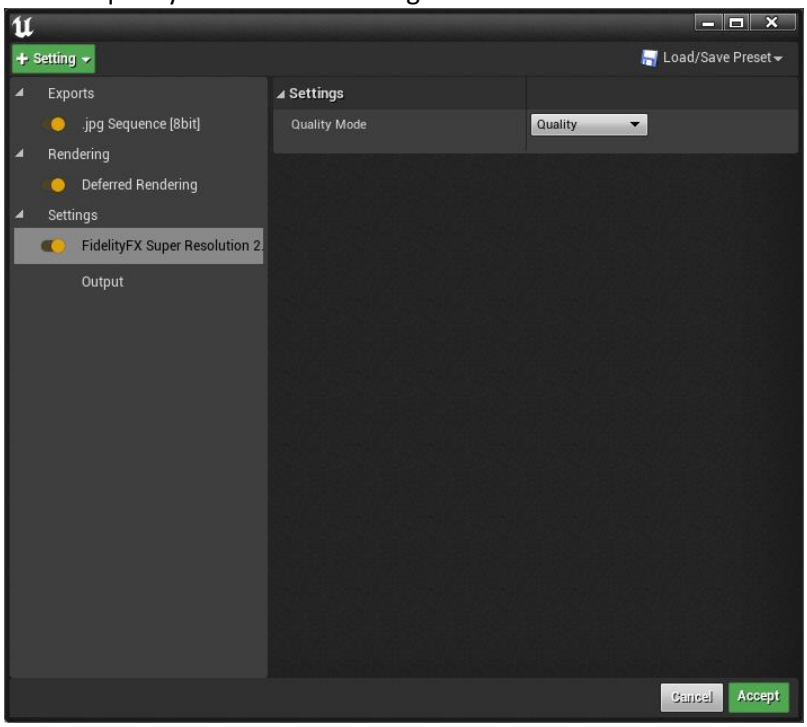
4. Click on **'Unsaved Config'** to open the settings for the Movie Render Queue.



5. Select the **‘+ Setting’** option and enable **‘Fidelity FX Super Resolution 2.0 Settings’**.



6. Then select the new **‘Fidelity FX Super Resolution 2.0 Settings’** in the list and choose the desired quality mode for rendering.



7. Click Accept & then Render (Local).
8. The output will be rendered using FSR2 to upscale the output to the target resolution.