FidelityFX Super Resolution 3.1

Unreal Engine Plugin

The AMD FidelityFX Super Resolution 3 (FSR 3) plugin for Unreal Engine provides an open source, high-quality solution for producing high resolution frames from lower resolution inputs and a frame interpolation technique which can increase the frame rate up to twice the input rate to improve smoothness of animation and frame pacing.

It uses temporal upscaling based on AMD FSR 2.2.1, an optical flow implementation, and algorithms to reproject samples from two frames to generate an additional frame in between. AMD FSR 3 also implements a proxy swap chain class that implements the `IDXGISwapChain` interface to schedule interpolation workloads and handle frame pacing.

The package also includes the **FSR3MovieRenderPipeline** plugin which enables use of AMD FSR 3 technology to accelerate rendering when using the Unreal Movie Render Queue.

FSR 3's Recommendations and Known Issues *must* be understood and addressed on a per-project basis to achieve optimal quality. See the respective section in this guide.

Contents

Known issues

Unreal 5.1.0

The pre-built binary version of the 5.1 plugin requires Unreal Engine 5.1.1 due to an ABI incompatibility between 5.1.0 and 5.1.1. Please update to Unreal Engine 5.1.1.

Plugin engine version warning

When building the Unreal Engine from source it may result in the engine displaying a warning that the FSR 3 plugin was built for a different engine version. When this occurs, shaders may not be cooked and packaged properly. Using the appropriate FSR 3 plugin build for the major and minor version of the engine and modifying the patch number in the FSR3.uplugin file (and optionally the FSR3MovieRenderPipeline.uplugin file) to match the version of engine source being used is sufficient to resolve the error and ensure the shaders are properly cooked.

World-Position-Offset on static objects

In specific circumstances static objects that use a material with World-Position-Offset do not always generate motion vectors. This affects FSR 3's ability to correctly upscale such materials and results in blurring/ghosting of the affected objects.

For Unreal 5.1 objects with WPO should render velocity by default. The default is to render velocity in the base pass and this can be changed using *`r.VelocityOutputPass`*. If it is necessary *`r.Velocity.ForceOutput`* can be used to force all primitives to emit velocity.

Animated opaque materials

Animated opaque materials which do not generate motion vectors for the animated content, such as in-world video screens, may also blur or ghost when obscured by static geometry. This can be reduced by ensuring such materials write into the Reactive Mask generated in the plugin.

For the Deferred Renderer this can be achieved by selecting a Shading Model for the *'Reactive Shading Model*' option in the FSR3 section of the project settings or using the *`r.FidelityFX.FSR3.ReactiveMaskForceReactiveMaterialValue`* console variable. Materials that use this Shading Model will be treated as reactive. This means that when this Shading Model is selected in the Material Editor that material will write either the value of the CustomData0.x channel, when exposed by the Shading Model, or the value of the console-variable *`r.FidelityFX.FSR3.ReactiveMaskForceReactiveMaterialValue`* provided it is set to a value greater than 0. Using *`r.FidelityFX.FSR3.ReactiveMaskForceReactiveMaterialValue`* overrides any material specific reactivity specified in the CustomData0.x channel.

Where selecting an existing shading model is unsuitable follow the installation instructions to install the correct version of the 'LitReactiveShadingModel' engine patch which adds a new *'LitReactive'* shading model that can be used specifically for this purpose.

This problem cannot currently be resolved in the Forward Renderer where the Shading Model cannot be determined by the plugin.

Substrate materials (Strata in Unreal 5.1) are currently not supported by the LitReactive shading model.

PIX and RenderDoc issues

When the native FSR3 backends are used within the FSR3 plugin graphics capture tools such as PIX and RenderDoc may be less stable. Disabling the native backend *(`r.FidelityFX.FSR3.UseNativeDX12`*) and using the RHI-based backend may allow the capture tool to replay captures more reliably.

UE post-processing volume screen-percentage overrides

UE developers should be aware that FSR3's quality mode (when enabled) will determine the screen-percentage and ignores any screen-percentage override present in a post-processing volume. This will result in different visual and performance results.

UE5.2 RDG debugging

Usage of some RDG debugging features can lead to GPU device removal when FSR3 and Lumen are both active.

Direct3D debug errors

When using the native Direct3D 12 backend with frame interpolation and *`r.FidelityFX.FI.OverrideSwapChainDX12`* enabled changing the display mode can cause cross-queue resource access errors when the Direct3D debug layer is enabled. These errors should not affect execution without the debug layer.

Recommendations

Optimizing translucency appearance

While the default settings for the FSR3 Reactive Mask should generate reasonable results, it is important that developers are aware that the appearance can be altered via the `*r.FidelityFX.FSR3.ReactiveMask*` console-variables. Tuning these variables to suit the content may be necessary to optimise visual results.

Translucent skyboxes and 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 FSR3. If these are rendered with the Translucent shading model they will contribute to the FSR3 translucency and 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, so they are all treated the same.

To address this issue, the FSR3 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.FSR3.ReactiveMaskTranslucencyMaxDistance*` console variable.

When using an opaque skybox or backplane, adjust the `*r.FidelityFX.FSR3.ReactiveMaskTranslucencyMaxDistance*` console variable to avoid translucency cut-outs.

Hair and dither effects

FSR 3 does not smooth dither effects in the way other upscalers do. They are retained as thin features which may not be intentional. To avoid this, especially with hair, enable the `*r.FidelityFX.FSR3.DeDither*` console variable which attempts to smooth dither effects prior to FSR 3 upscaling.

Release notes

3.0.3

- Initial release.

3.0.3a

- Updated documentation.

3.0.4

- Update the FSR3 code to version 3.0.4.
- Resolved a crash when pausing a game in the editor.
- Resolved an error that prevented packaging of games to be carried out.
- Changed the behaviour of `*r.FidelityFX.FI. OverrideSwapChainDX12*` when using the Direct3D 12 backend to render the UI onto both the interpolated and game frame rather than using the frame interpolation library's capability to extract the UI from the game frame. The proxy-swapchain is still used to provide far superior frame pacing and asynchronous present versus the RHI backend.
- Reverted the handling of translucency for CreateReactiveMask in the FSR3 Temporal Upscaler to behave the same as in the FSR2 plugin. This had been simplified in FSR3 3.0.3 due to changes in Unreal Engine 5.3 but it could result in inferior handling of Post-DOF Translucency rendered into the SeparateTranslucency texture.
- Improved the handling of debug UI – see the Integration Instructions > Debug UI section for details.

3.1

- Updated to FSR3 code to version 3.1 - Shader Model 6 is now **required**.
- Updated to support Unreal Engine version 5.4.
- Fixed support for split-screen.
- Resolved Direct3D debug layer warnings and errors.
- When forcing Slate to re-render the UI the time-delta provided will be set to 0 for the second invocation to avoid incorrect behavior in NativeTick.
- Added `*r.FidelityFX.FI.UIMode*` to control how the UI is rendered when using the Direct3D 12 backend.
- Added `*r.FidelityFX.FI.RHIPacingMode*` to enable simple frame-pacing when using the RHI to present frames.

[Public]

Setup

The FSR3 and FSR3MovieRenderPipeline plugins are intended for Unreal Engine 5.1.1* or later.

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

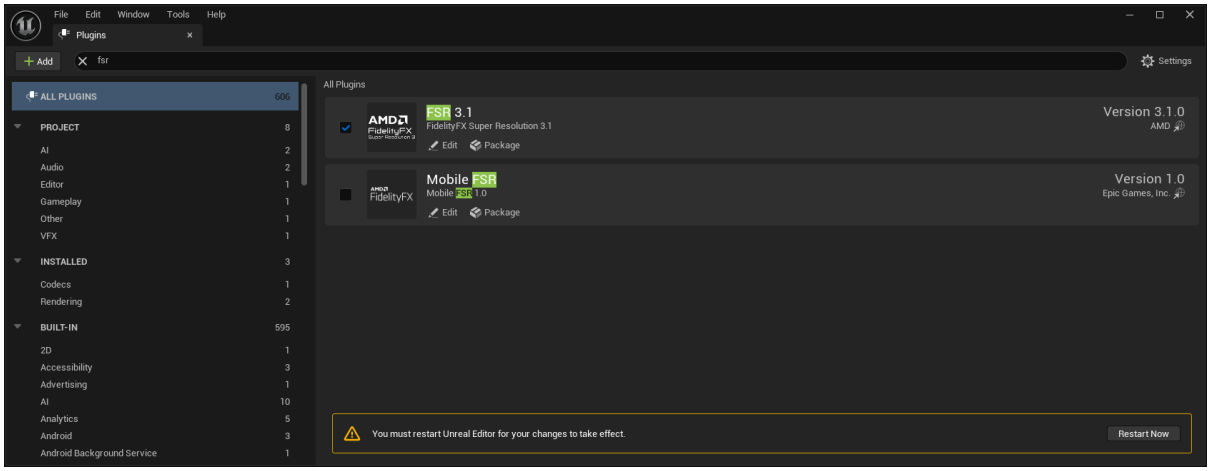
Patch the engine.

For optimal quality it is necessary to use Unreal Engine from source code and apply source code patches.

1. To improve FSR3's handling of animated opaque materials:
    a. Use: `git apply <VERS>-LitReactiveShadingModel.patch`
        i. Where `<VERS>` should be the engine-version in use.

Install the plugin

1. Locate the *Engine/Plugins* directory of your Unreal Engine installation.
2. Extract the contents of the FSR3.zip file.
3. Select the sub-folder that corresponds to the Unreal Engine version to be used.
4. Place the **FSR3** folder within your Unreal Engine source tree at: *Engine/Plugins/Marketplace* (for UE5)
    a. (Optional) Place the **FSR3MovieRenderPipeline** folder within your Unreal Engine source tree at: *Engine/Plugins/Marketplace* (for UE5).
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 3.1** plugin.
        i. (Optional) Select the **Enabled** checkbox for the **FSR3MovieRenderPipeline** plugin.
    d. When prompted, click **Restart Now** to apply changes, and restart Unreal Engine.

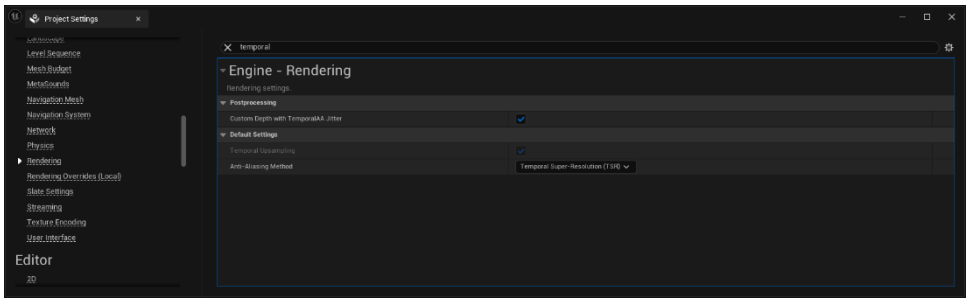


[Public]

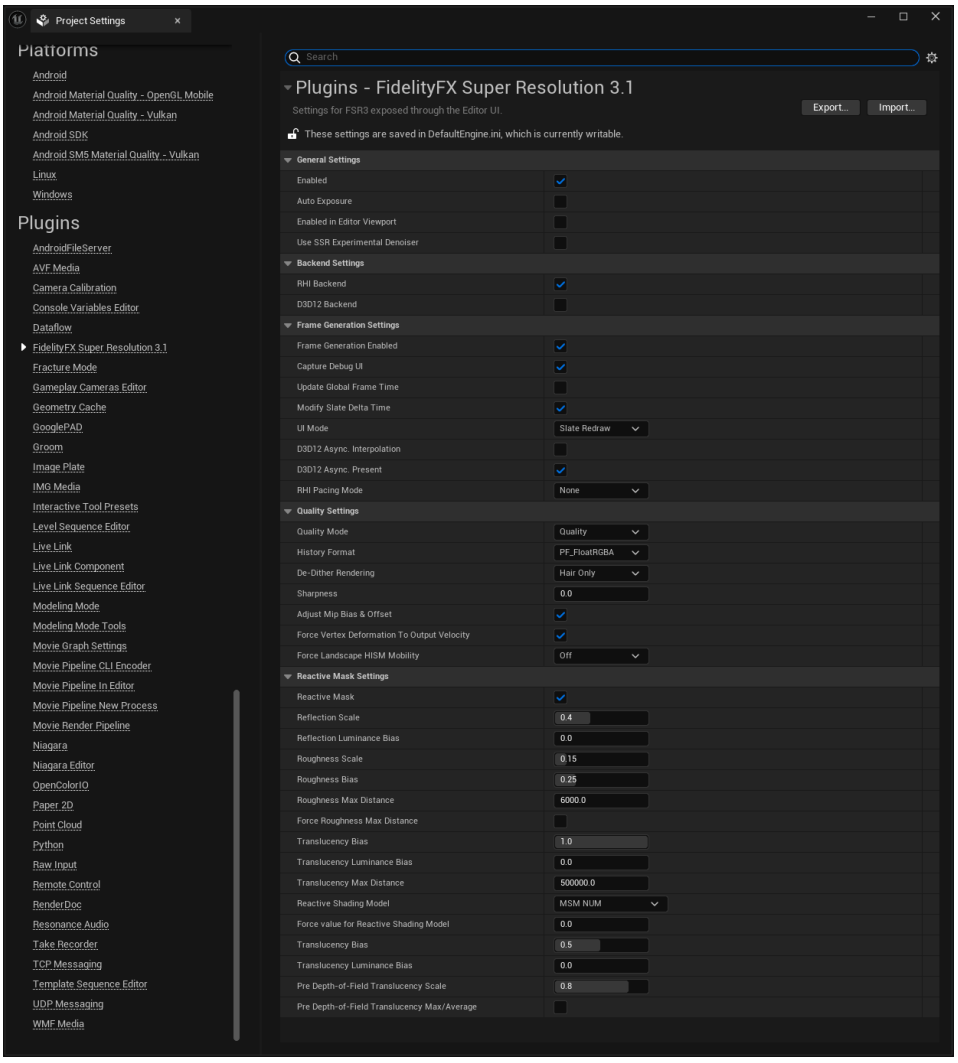
Plugin configuration

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 3.1 can be enabled or disabled via the **Enabled** option in the **Project Settings > FidelityFX Super Resolution 3.1** settings window, or with the console variable *`r.FidelityFX.FSR3.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.



Quality modes

The plugin will use specific quality modes specified via *`r.FidelityFX.FSR3.QualityMode`* overriding *`r.ScreenPercentage`*. The exposed modes are:

- **Native AA (1.0x)**: *`r.FidelityFX.FSR3.QualityMode 0`*
  Provides an image quality superior to native rendering with a modest performance cost.
- **Quality (1.5x)**: *`r.FidelityFX.FSR3.QualityMode 1`*
  Provides an image quality equal or superior to native rendering with a significant performance gain.
- **Balanced (1.7x)**: *`r.FidelityFX.FSR3.QualityMode 2`*
  Offers an ideal compromise between image quality and performance gains.
- **Performance (2.0x)**: *`r.FidelityFX.FSR3.QualityMode 3`*
  Provides an image quality similar to native rendering with a major performance gain.
- **Ultra Performance (3.0x)**: *`r.FidelityFX.FSR3.QualityMode 4`*
  Provides the highest performance gain while still maintaining an image quality representative of native rendering.

Frame Generation

Frame Generation can be enabled or disabled via the **Project Settings > FidelityFX Super Resolution 3.1** settings panel by changing the setting of 'Frame Generation Enabled'. It does not affect the Editor and requires the FSR3 temporal upscaler to be enabled as it requires inputs generated by the upscaling component.

Frame Generation supports both an RHI and Direct3D 12 backend. The default is the native Direct3D 12 backend as it provides better performance and frame pacing via the FSR3 proxy IDXGISwapChain implementation. The RHI backend can be used on other platforms to provide basic frame generation support running serially after the end of the game frame and using the in-built Slate and RHI presentation framework.

Direct3D 12

When running on Windows using Direct3D 12 there are additional options to control how Frame Generation executes to take advantage of the proxy IDXGISwapChain implementation.

- **D3D12 Async. Present**: *`r.FidelityFX.FI.OverrideSwapChainDX12`*
  Enables the FSR3 proxy IDXGISwapChain implementation that provides superior frame pacing via asynchronous presentation of frames. This is a read-only setting and cannot be changed at runtime, so a restart is required to modify the setting when exposed as an option.
- **D3D12 Async. Interpolation**: *`r.FidelityFX.FI.AllowAsyncWorkloads`*
  When enabled the frame interpolation executes on the asynchronous queue provided by the FSR3 proxy IDXGISwapChain implementation which allows frame interpolation to occur concurrently with the beginning of the following game frame. This improves performance compared to running frame interpolation serially with the game rendering.
  ***NOTE**: Enabling this mode prevents the plugin from requesting Slate re-render the game UI onto each interpolated and normal game frame, instead the UI is extracted from the normal game frame by comparing the pre-UI and post-UI frame contents. This can result in inferior handling of UI with translucency on interpolated frames.*

Integration instructions

RCAS

FidelityFX Super Resolution 3.1 contains a built-in sharpening pass called **Robust Contrast Adaptive Sharpening** that can be configured through the CVar *`r.FidelityFX.FSR3.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.FSR3.Sharpness`* is enabled to prevent over-sharpening your final renders and improve integration results.

World-Position-Offset

For FSR3 to process materials with World Position Offset and/or World Displacement correctly the *`r.Velocity.EnableVertexDeformation`* option must be enabled. The *`r.FidelityFX.FSR3.ForceVertexDeformationOutputsVelocity`* setting is enabled by default and when enabled FSR3 will force *`r.Velocity.EnableVertexDeformation`* on. The plugin will also enable *'r.Velocity.EnableLandscapeGrass'* if it has been disabled.

This option also forces on the console-variable *'r.BasePassForceOutputsVelocity'* to ensure that all objects render velocity during the base-pass when a project enables *'r.BasePassOutputsVelocity'*.

Reactive mask and experimental screen space reflection denoiser

When *`r.FidelityFX.FSR3.CreateReactiveMask`* is enabled the FSR3 plugin forces *`r.SSR.ExperimentalDenoiser`* to 1 to capture the screen space reflections, to handle this the initial value of *`r.SSR.ExperimentalDenoiser`* will be applied to *`r.FidelityFX.FSR3.UseSSRExperimentalDenoiser`*. Subsequent changes to the value of *`r.FidelityFX.FSR3.UseSSRExperimentalDenoiser`* will override this.

Multiple upscalers

To switch to or from FSR3 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

Frame Generation

FSR3 Frame Generation requires FSR3 Upscaling to be enabled, when Frame Generation is enabled but FSR3 Upscaling is disabled no Frame Generation will be performed and rendering will proceed normally.

Using the **D3D12 Async. Present** (`r.FidelityFX.FI.OverrideSwapChainDX12`) option is incompatible with other frame interpolation technologies. Only one may be enabled at one time and toggling between them requires a restart of the title.

If using the RHI Backend then enabling V-Sync is recommended as this backend does not provide robust frame pacing and relies on the existing Unreal Slate and RHI code to pace and present the frame. When not using V-Sync with the RHI Backend enabling `r.FidelityFX.FI.RHIPacingMode` forces V-Sync on when displaying the 'real' frame to improve pacing.

UI Rendering

There are two mechanisms to render the UI when FSR3 Frame Generation is enabled with different trade-offs and availability.

When using the RHI Backend or with the D3D12 Backend when *`r.FidelityFX.FI.UIMode`* is set to 'Slate Redraw' (0) the UI is rendered via Slate on to both the generated and normal frame. This is done by forcing Slate to re-render the window. This will invoke NativeTick on each Slate widget twice

(once for the generated and once for the normal frame) - the second invocation will have a delta-time of 0 unless *`r.FidelityFX.FI.ModifySlateDeltaTime`* is disabled. The frames are then presented using the normal RHI present mechanism.

When using the D3D12 Backend with *`r.FidelityFX.FI.OverrideSwapChainDX12`* enabled, setting *`r.FidelityFX.FI.UIMode`* to 'UI Extraction' (1) causes UI to be extracted from the normal frame rendering by comparing the frame-buffer prior to UI rendering with the frame-buffer after UI rendering and then using the result to integrate the UI onto the interpolated frame. This decouples Frame Generation from Slate, NativeTick is not invoked twice, and is automatically enabled by *`r.FidelityFX.FI.AllowAsyncWorkloads`*. This method generally works acceptably but may produce inferior results with translucent UI elements.

Debug UI

When using FSR3 Frame Generation with the RHI Backend or the DX12 backend without *'r.FidelityFX.FI.AllowAsyncWorkloads'* the Slate UI is rendered twice but this does not include Unreal's debug UI e.g. the in-game console. For Debug/Development/Test builds *'r.FidelityFX.FI.CaptureDebugUI'* is enabled to ensure that these UI elements appear but it is disabled for Shipping by default. Enable this setting explicitly to avoid flickering if debug UI elements are visible in a Shipping build.

Other Configurations

Console Variable	Default Value	Value Range	Details
r.FidelityFX.FSR3.AdjustMipBias	1	0, 1	Applies negative MipBias to material textures, improving results.
r.FidelityFX.FSR3.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.FSR3.AutoExposure	0	0, 1	Set to 1 to use FSR3's own auto-exposure, otherwise the engine's auto-exposure value is used.
r.FidelityFX.FSR3.HistoryFormat	0	0, 1	Selects the bit-depth for the FSR3 history texture format, defaults to `PF_FloatRGBA` but can be set to `PF_FloatR11G11B10` to reduce bandwidth at the expense of quality.
r.FidelityFX.FSR3.QualityMode	1	0 - 4	FSR3 Mode. Lower values yield superior images. High values yield improved performance.
r.FidelityFX.FSR3.CreateReactiveMask	1	0, 1	Enable to generate a mask from the SceneColor, GBuffer, SeparateTranslucency and ScreenspaceReflections that determines how reactive each pixel should be.
r.FidelityFX.FSR3.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.FSR3.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.FSR3.ReactiveMaskRoughnessScale	0.15	0.0 – 1.0	Scales the GBuffer roughness to provide a

			fallback value for the reactive mask when screenspace and planar reflections are disabled or do not affect a pixel.
r.FidelityFX.FSR3.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.FSR3.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.FSR3.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.FSR3.ReactiveMaskTranslucencyBias	1.0	0.0 – 1.0	Scales how much contribution translucency makes to the reactive mask. Higher values will make translucent materials more reactive which can reduce smearing.
r.FidelityFX.FSR3.ReactiveMaskTranslucencyLumaBias	0.0	0.0 – 1.0	Biases the translucency contribution to the reactive mask by the luminance of the transparency. Higher values will make bright translucent materials more reactive which can reduce smearing.
r.FidelityFX.FSR3.ReactiveHistoryTranslucencyBias	0.5	0.0 – 1.0	Scales how much translucency suppresses history via the reactive mask. Higher values will make translucent

			materials more reactive which can reduce smearing.
r.FidelityFX.FSR3.ReactiveHistoryTranslucencyLumaBias	0.0	0.0 – 1.0	Biases how much the translucency suppresses history via the reactive mask by the luminance of the transparency. Higher values will make bright translucent materials more reactive which can reduce smearing.
r.FidelityFX.FSR3.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 more reactive which can reduce smearing.
r.FidelityFX.FSR3.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.FSR3.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.FSR3.ReactiveMaskReactiveShadingModelID	MSM_NUM	0-MSM_NUM	Treat the specified shading model as reactive, taking the CustomData0.x value as the reactive value to write into the mask.

r.FidelityFX.FSR3.ReactiveMaskForceReactiveMaterialValue	0.0	0.0-1.0	Force the reactive mask value for Reactive Shading Model materials, when > 0 this value can be used to override the value supplied in the Material Graph.
r.FidelityFX.FSR3.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.FSR3.ForceLandscapeHISMMobility	0	0, 1, 2	Allow FSR3 to force the mobility of Landscape actors Hierarchical Instance Static Mesh components that use World-Position-Offset materials so they render valid velocities. Setting 1 is faster on the CPU, 2 is faster on the GPU.
r.FidelityFX.FSR3.UseSSRExperimentalDenoiser	0	0, 1	Enable to use *r.SSR.ExperimentalDenoiser* when FSR3 is enabled. This is required when *r.FidelityFX.FSR3.CreateReactiveMask* is enabled as the FSR3 plugin overrides *r.SSR.ExperimentalDenoiser* to capture reflection data to generate the reactive mask.
r.FidelityFX.FSR3.UseNativeDX12	1	0, 1	True to use FSR3's native and optimised D3D12 backend, false to use the fallback implementation based on Unreal's RHI.
r.FidelityFX.FSR3.UseRHI	0	0, 1	True to enable FSR3's default RHI backend, false to disable in which case a native backend must be enabled.
r.FidelityFX.FSR3.QuantizeInternalTextures	0	0, 1	Setting this to 1 will round up the size of some internal texture to ensure a specific divisibility. This is only intended for

			compatibility if required. Default is 0.
r.FidelityFX.FSR3.EnabledInEditorViewport	0	0, 1	Enable FidelityFX Super Resolution for Temporal Upscale in the Editor viewport by default.
r.FidelityFX.FSR3.DeDither	2	0, 1, 2	Enable an extra pass to de-dither rendering before handing over to FSR3 to avoid over-thinning. Can be set to Full for all pixels or to just around Hair for Deferred Renderer. Defaults to Hair Only.
r.FidelityFX.FI.Enabled	1	0, 1	Enable FidelityFX Frame Interpolation.
r.FidelityFX.FI.CaptureDebugUI	1 (Debug/Development/Test), 0 (Shipping)	0, 1	Force FidelityFX Frame Interpolation to detect and copy any debug UI which only renders on the first invocation of Slate's DrawWindow command. **Disabled in Shipping builds.**
r.FidelityFX.FI.OverrideSwapChainDX12	1	0, 1	True to use FSR3's D3D12 swap-chain override that improves frame pacing, false to use the fallback implementation based on Unreal's RHI.
r.FidelityFX.FI.AllowAsyncWorkloads	0	0, 1	True to use async. execution of Frame Interpolation, 0 to run Frame Interpolation synchronously with the game.
r.FidelityFX.FI.UpdateGlobalFrameTime	0	0, 1	True to update the GAverageMs/GAverageFPS globals with average frame time/FPS calculated by the frame interpolation code.
r.FidelityFX.FI.ShowDebugTearLines	1	0, 1	Show the debug tear lines when running Frame Interpolation. **Not available in Test/Shipping builds.**
r.FidelityFX.FI.ShowDebugView	0	0, 1	Show the debug view when running Frame Interpolation. **Not**

			available in Test/Shipping builds.
r.FidelityFX.FI.UIMode	0	0, 1	The method to render the UI when using Frame Generation. Slate Redraw (0): will cause Slate to render the UI on to both the real & generated images. UI Extraction (1): will compare the pre- & post- UI frame to extract the UI and copy it on to the generated frame.
r.FidelityFX.FI.ModifySlateDeltaTime	1	0, 1	Set the FSlateApplication delta time to 0.0 when redrawing the UI for the 'Slate Redraw' UI mode to prevent widgets' NativeTick implementations updating incorrectly, ignored when using 'UI Extraction'.
r.FidelityFX.FI.RHIPacingMode	0	0, 1	Enable pacing frames when using the RHI backend. None (0) : No frame pacing. Custom Present VSync (1) : enable VSync for the second presented frame.

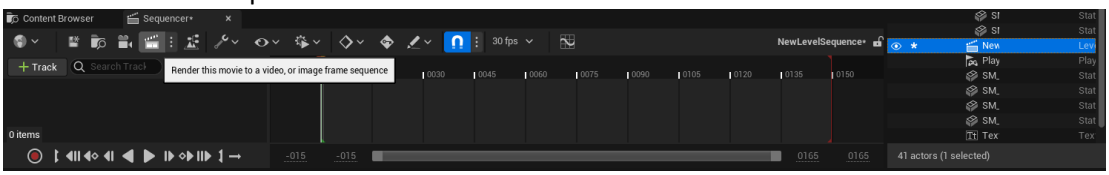
[Public]

Movie Render Pipeline plugin

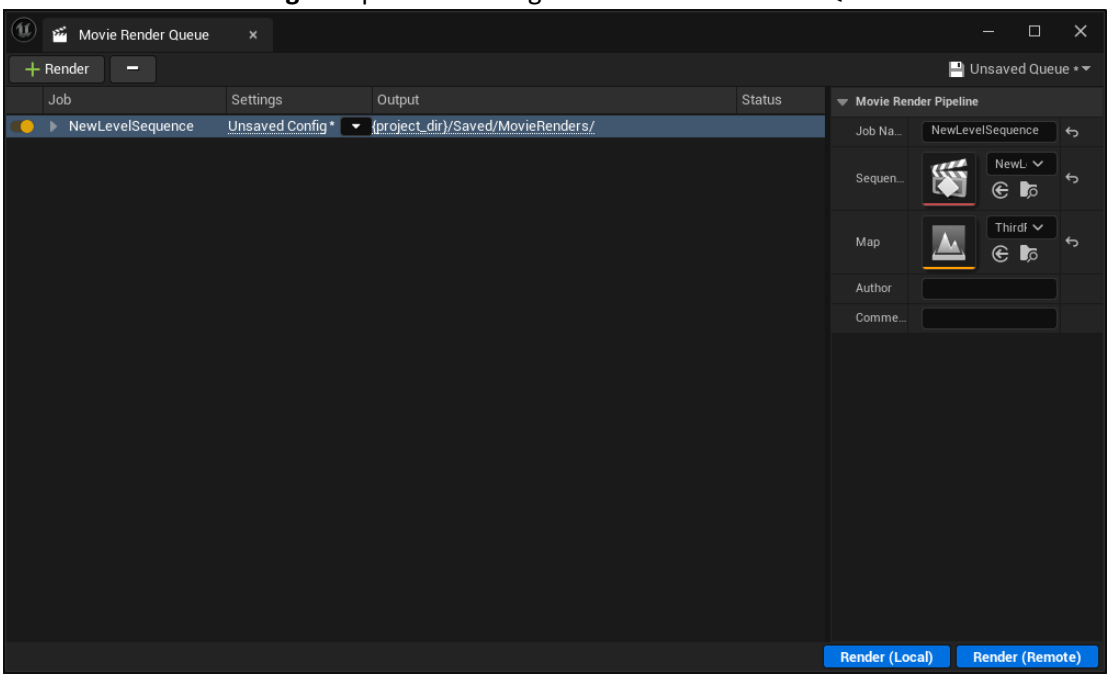
The FSR3MovieRenderPipeline plugin once enabled allows using FSR3 to accelerate rendering of Sequencer cinematics using Unreal's Movie Render Queue.

To use the plugin:

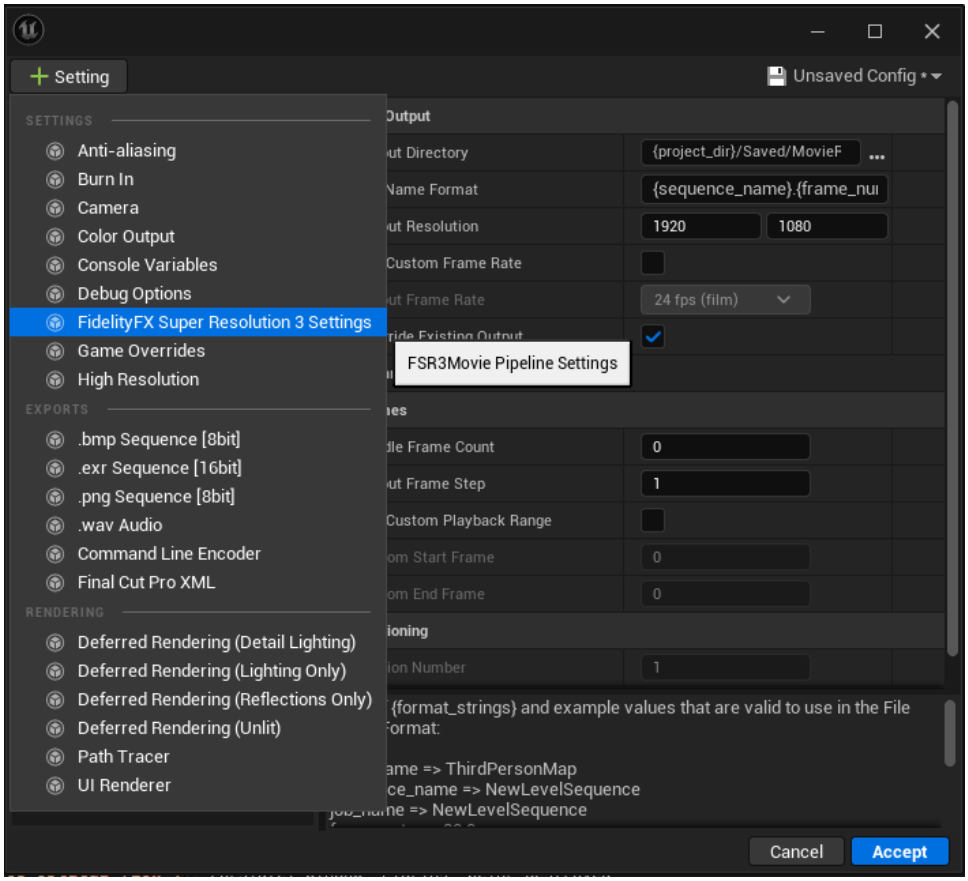
1. Open your Unreal project.
2. Open a **Sequencer** cinematic through the Editor.
3. Select the movie output in the toolbar.



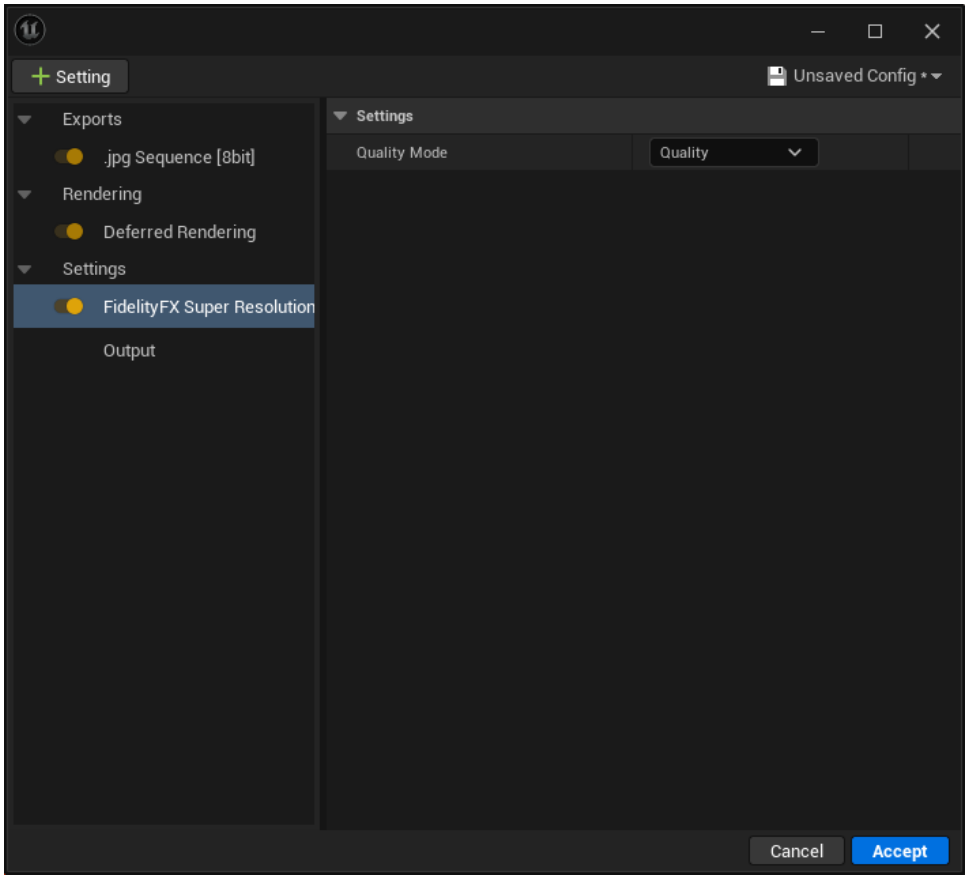
4. Click on **'Unsaved Config'** to open the settings for the Movie Render Queue.



5. Select the **‘+ Setting’** option and enable **‘FidelityFX Super Resolution 3.1 Settings’**.



6. Then select the new **‘FidelityFX Super Resolution 3.1 Settings’** in the list and choose the desired quality mode for rendering.



7. Click Accept and then Render (Local).
8. The output will be rendered using FSR3 to upscale the output to the target resolution.