Intel® X^{e}SS Plugin for Unreal* Engine

Introduction

This plugin integrates Intel(R) Xe Super Sampling (X^{e}SS) into Unreal Engine 4 as of 4.26 (or higher) and Unreal Engine 5.

Intel(R) X^{e}SS enables an innovative framerate boosting technology supported by Intel(R) Arc(TM) graphics cards and other GPU vendors. Using AI deep-learning to perform upscaling, X^{e}SS offers higher framerates without degrading the image. For more information visit: https://github.com/intel/xess

Installing the plugin

1. If you use source code version of Unreal, you need to build XeSS plugin locally, otherwise just skip this step.
    i. Open the terminal and go to ***<Unreal Engine install directory>\Engine\Build\BatchFiles***
    ii. Run the command line: ***RunUAT.bat BuildPlugin -plugin="<Full path to downloaded XeSS.uplugin location>\XeSS.uplugin" -package="<Full path to destination plugin location>"* -VS2019**
    > The last parameter is version of Visual Studio used to build, required by Unreal 4 only.

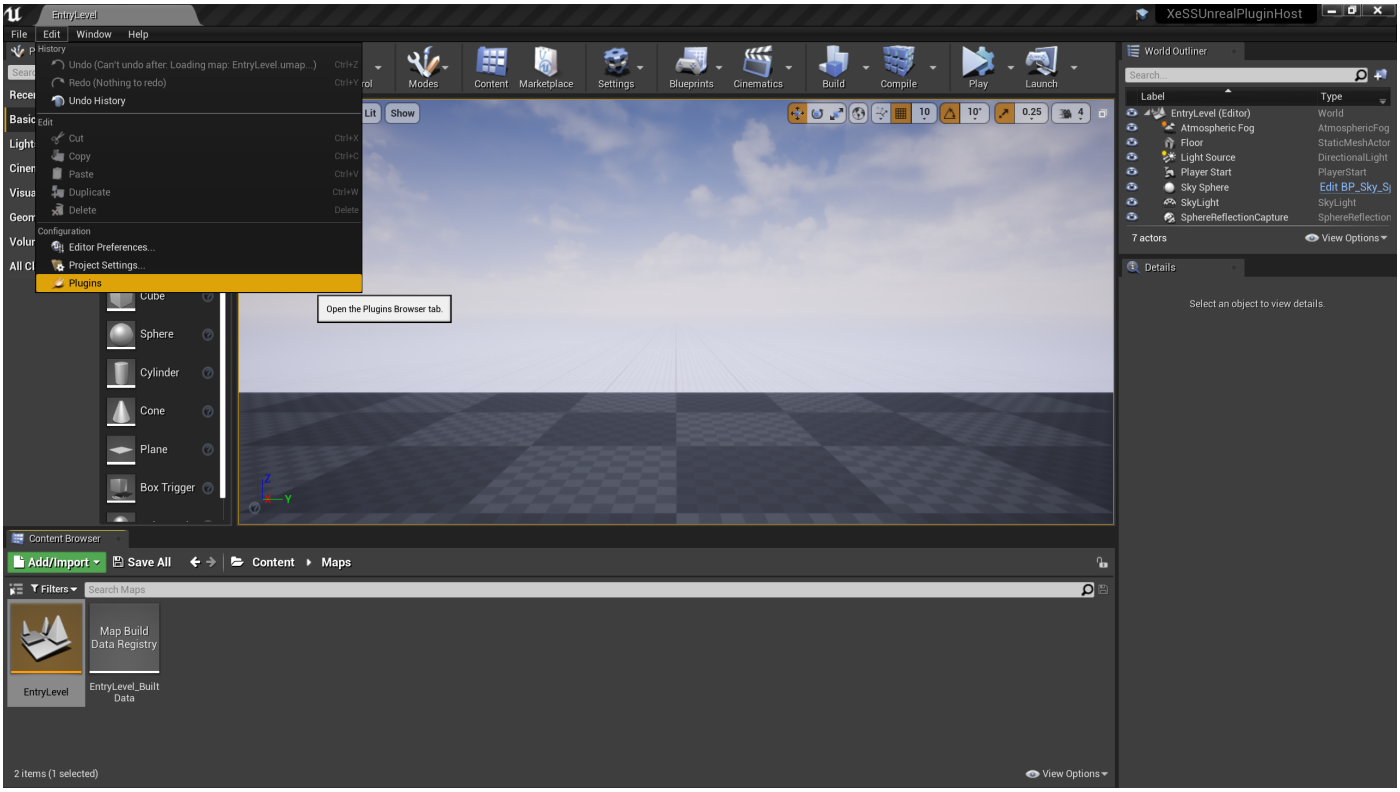
2. Copy (pre-)built files to:

- For UE4 - ***<Engine's main directory>\Engine\Plugins\Runtime\Intel\XeSS***
- For UE5 - ***<Engine's main directory>\Engine\Plugins\Marketplace\XeSS***

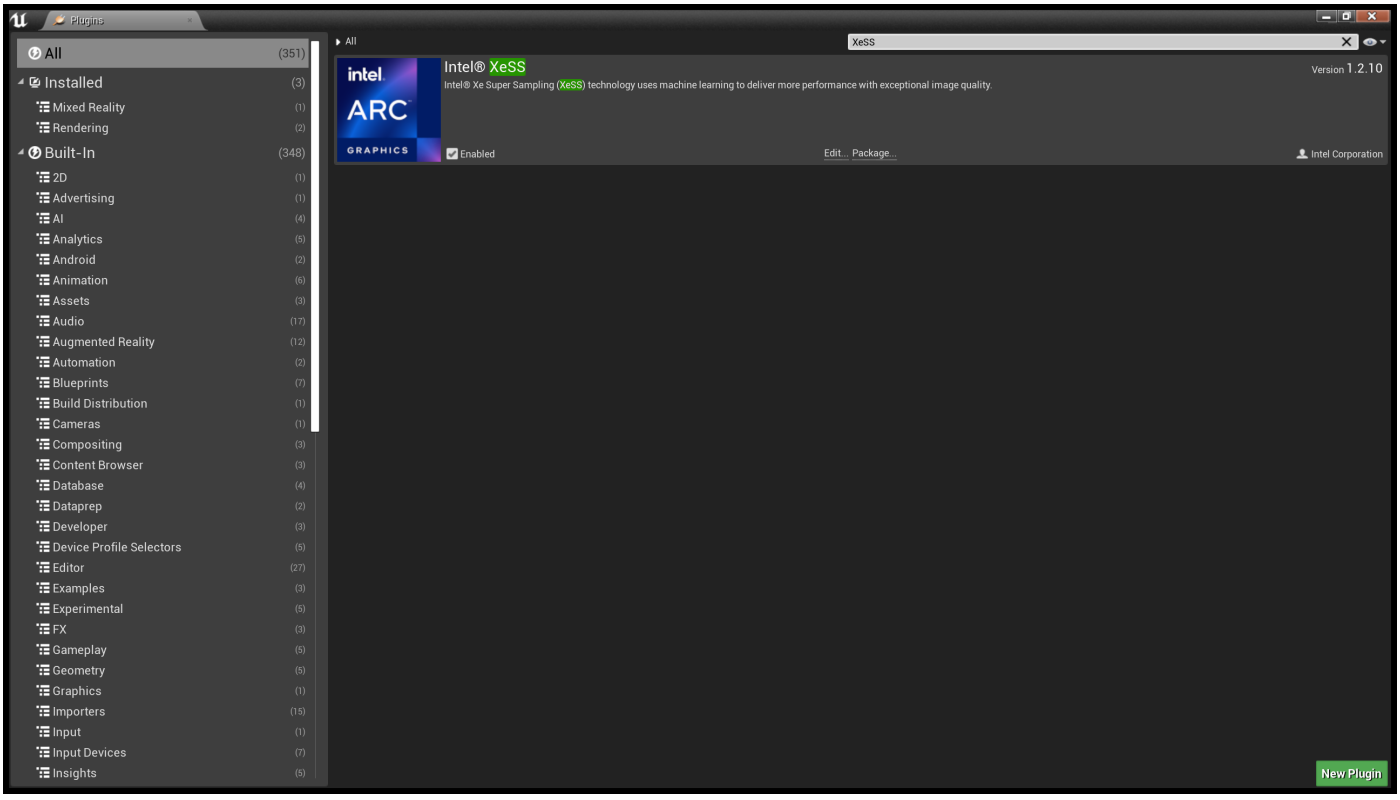
Enabling the plugin in the Editor

Note: X^{e}SS is only supported on the DX12 Renderer, please make sure that the Default RHI is set to DX12.

To enable the X^{e}SS plugin for a project go to the plugin selection.



Use the search box to find the X^{e}SS plugin and make sure it is enabled.



XeSS console settings

To enable the XeSS upscaler while running the game navigate to the console's command window by pressing the tilde (~) key and typing in the command:

- ***r.XeSS.Enabled 1***

To change the Quality Mode use the command:

- ***r.XeSS.Quality <quality_mode>***

Where <*quality_mode*> represents the scale factor:

Value	Quality Mode	Scale factor
0	Ultra Performance	3
1	Performance	2.3
2	Balanced (default)	2
3	Quality	1.7
4	Ultra Quality	1.5
5	Ultra Quality Plus	1.3
6	Anti-Aliasing	1

Auto exposure with XeSS SDK, which is enabled by default.

> For more details on exposure calculation, please check `Intel® XeSS Developer Guide` files in the `Documents` folder.

- ***r.XeSS.AutoExposure 1***

Checking the XeSS SDK version integrated with the plugin

Type in the following command in the console's window to query the used XeSS SDK version:

- ***r.XeSS.Version***

Project packaging

No additional steps are needed for packaging.

Verifying XeSS is enabled in a running game

The easiest way to confirm that XeSS is enabled is to open thr console's command window by pressing the tilde (~) key and typing in the command:

- ***stat gpu***

This will bring up the real-time per-frame stats. XeSS should visible as one of the rendering passes on the displayed list.

Capturing frames for debug purposes

The X^{e}SS SDK allows to capture all input resources and parameters that are passed to the X^{e}SS library, along with the resulting output upscaled image and the internal history resource. When enabled, data for each frame will be captured and saved on the local drive. To enable frame dumps:

1. Set the frame dump directory. If no path is specified the frame data will be saved in the game's binary directory.

- ***r.XeSS.FrameDump.Path <existing_directory>***

2. Make sure that X^{e}SS is enabled

- ***r.XeSS.Enabled 1***

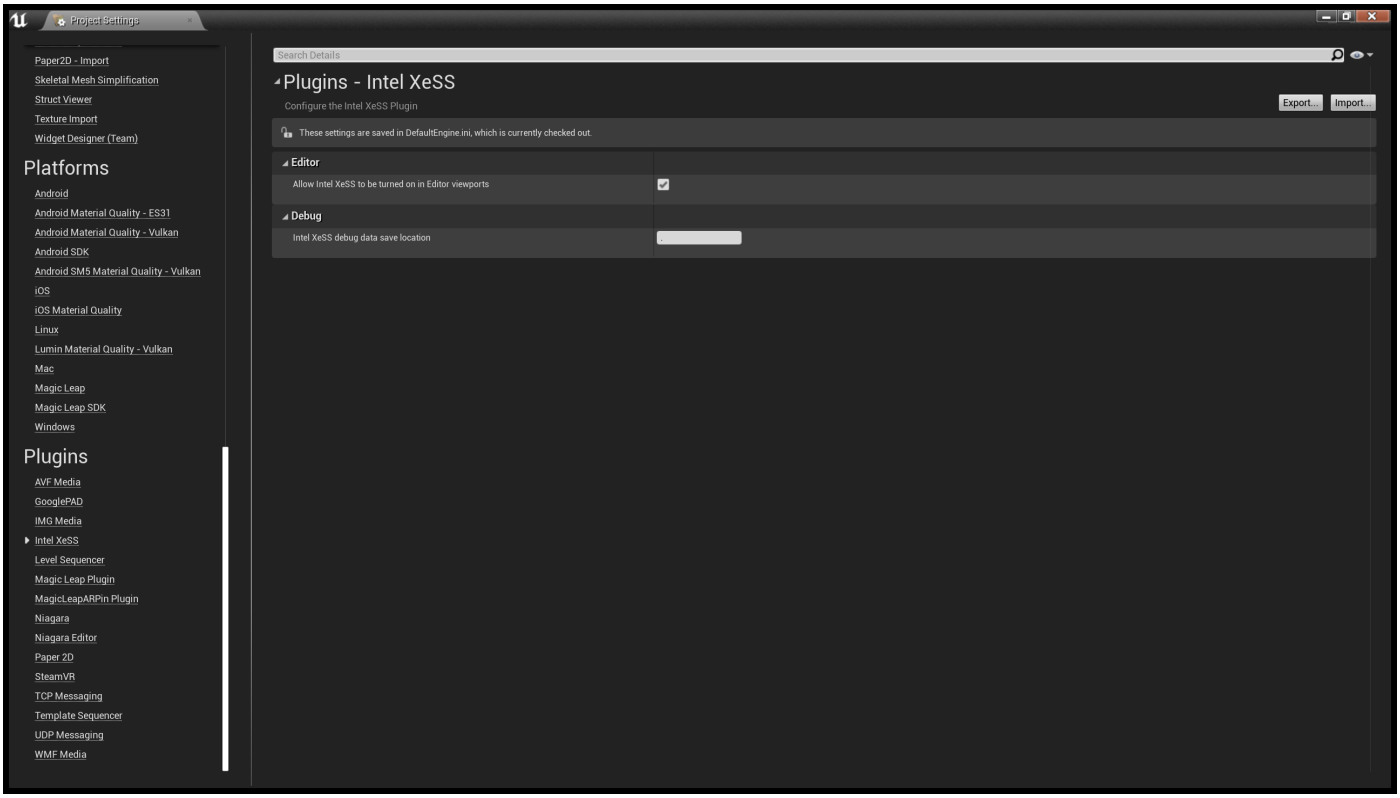
3. Specify the number of frames to be intercepted and trigger the capture, the engine's internal frame counter is used for file indexing.

- ***r.XeSS.FrameDump.Start <frame_count>***

X^{e}SS Project Settings

Once the X^{e}SS plugin is enabled it's default behavior can be modified in the Project Settings menu.

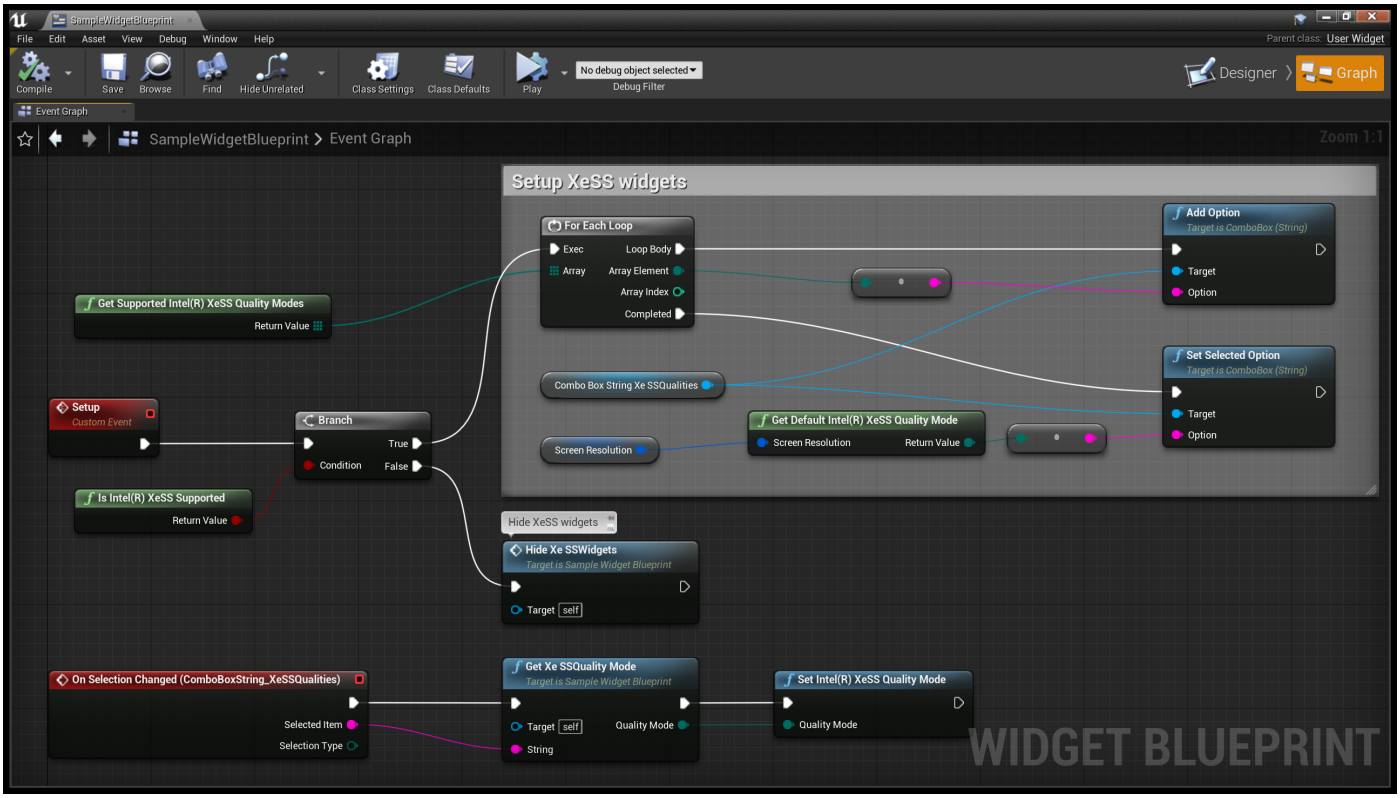
- It is possible to block/enable X^{e}SS usage in the Editor's viewports
- The default path for X^{e}SS debug dumps can be set for the Project, this value can be later overridden with the console var *r.XeSS.FrameDumpPath*



Blueprint support

The X^{e}SS plugin provides support for querying and settings X^{e}SS quality modes using Blueprint. It is recommended to use these functions when creating game menus with X^{e}SS settings.

- *Is Intel(R) XeSS Supported*
- *Get Supported Intel(R) XeSS Quality Modes*
- *Get Current Intel(R) XeSS Quality Mode*
- *Get Default Intel(R) XeSS Quality Mode*
- *Set Intel(R) XeSS Quality Mode*



Basic example showing Blueprint usage

Note:

1. It is recommend to use the *Get Default Intel(R) X^{e}SS Quality Mode* Blueprint function to set the out-of-the-box X^{e}SS Quality Mode in the game's UI. Based on the passed Screen Resolution the recommended Quality Mode will be returned - for resolutions with pixel counts corresponding to 1920x1080 and lower it will be *Balanced*, for higher resolutions it will be *Performance*.
2. No internalization considered to simplify example Blueprint.

Localization support

Localized text is supported via string table "XeSSStringTable", which can be referenced in widgets, Blueprint and C++ code.

Here are cultures currently supported:

Abbreviation	Name
ar-SA	Arabic (Saudi Arabia)
da-DK	Danish (Denmark)
de-DE	German (Germany)
en	English
es-ES	Spanish (Spain)
fr-FR	French (France)
it-IT	Italian (Italy)
ja-JP	Japanese (Japan)
ko-KR	Korean (South Korea)
nl-NL	Dutch (Netherlands)
pl-PL	Polish (Poland)
pt-PT	Portuguese (Portugal)
ru-RU	Russian (Russia)
uk-UA	Ukrainian (Ukraine)
zh-Hans	Chinese (Simplified)
zh-Hant	Chinese (Traditional)

Using the Unreal High Resolution Screenshot tool

Unreal provides a high resolution screenshot console command that allows to take high resolution screen captures.

- ***HighResShot <screen_resolution>***

When using this tool while X^{e}SS is engaged please make sure to set *screen_resolution* to the currently set output resolution (can be set with *r.SetRes <screen_resolution>*). Otherwise, the capture tool will change the target resolution, which will re-initialize the X^{e}SS context and drop all temporally accumulated data. In result the captured image will not reflect the actual quality seen on the screen.

Unreal 5.1 or above support

Blueprint API is recommended to use in Blueprint or C++, for plugin loses control to upscaling screen percentage directly since Unreal 5.1 and `r.ScreenPercentage` is set in Blueprint API to keep its backward compatibility.

Known issues

- Stereo rendering for VR or split screen usage is currently not supported.
- Pre-built Unreal 5.1 package doesn't work with Unreal 5.1.0, please upgrade Unreal to 5.1.1.
- If following link error occurs with pre-built packages, please upgrade Visual Studio to the latest version.

```
error LNK2019: unresolved external symbol __std_find_trivial_4 referenced in function "int * __
```

- Some text is not localized in some cultures.
- Pre-exposure is still not used in the exposure calculation process.