



## NVIDIA 언리얼 엔진 DLSS 프레임 생성 플러그인(스트림라인)

### DLSS3 및 NVIDIA 언리얼 엔진 DLSS 프레임 생성 플러그인

NVIDIA DLSS 프레임 생성 플러그인은 성능 및 이미지 품질 향상과 관련된 광범위한 NVIDIA 기술 및 해당 NVIDIA 언리얼 엔진 플러그인 제품군의 일부입니다:

- NVIDIA 딥 러닝 슈퍼샘플링 프레임 생성(DLSS-FG)은 AI를 사용하여 추가 프레임을 렌더링함으로써 프레임 속도를 향상시킵니다. DLSS-FG에는 GeForce RTX 40 시리즈 그래픽 카드가 필요합니다.
- NVIDIA 딥 러닝 슈퍼샘플링 초해상도(DLSS-SR)는 더 적은 픽셀을 렌더링하고 AI를 사용하여 고해상도 프레임을 출력함으로써 프레임 속도를 향상시킵니다. DLSS-SR에는 NVIDIA RTX 그래픽 카드가 필요합니다.
- NVIDIA 딥 러닝 앤티 앤티에이싱(DLAA)은 이미지 품질을 개선하는 데 사용됩니다. DLAA에는 NVIDIA RTX 그래픽 카드가 필요합니다.
- NVIDIA 이미지 스케일링(NIS)은 NVIDIA 또는 타사 비RTX GPU에 등급 최고의 업스케일링 및 선명도를 제공합니다.

자세한 내용은 NVIDIA 이미지 스케일링/언리얼 엔진 플러그인을 참조하세요. NVIDIA DLSS 3는 DLSS 슈퍼 해상도, DLSS 프레임 생성 및 NVIDIA Reflex를 결합합니다.

NVIDIA 언리얼 엔진 DLSS 프레임 생성 플러그인(여기에서 문서화되어 있음)이 제공합니다:

- DLSS 프레임 생성(DLSS-G 또는 DLSS-FG라고도 함)
- NVIDIA Reflex

NVIDIA 언리얼 엔진 DLSS-SR 플러그인(별도 구매)은 다음과 같은 기능을 제공합니다:

- DLSS 슈퍼 해상도(DLSS-SR)
- 딥러닝 앤티 앤티에이싱(DLAA)

NVIDIA 언리얼 엔진 NIS 플러그인(별도 구매)은 다음과 같은 기능을 제공합니다:

- NVIDIA 이미지 스케일링

### 통합 권장 사항

DLSS 프레임 생성 플러그인은 에픽의 패키지 엔진 릴리스를 포함하여 언리얼 엔진 5.2 이상에서 '즉시' 지원됩니다.

5.1 및 이전 엔진 릴리스의 경우 DLSS 프레임 생성 플러그인을 지원하기 위해 엔진 자체에 추가적인 소스 변경을 수행해야 합니다. 소스 코드에서 언리얼 엔진을 리빌드하고 코드 스니펫을 엔진 코드 자체에 병합하는 것에 대해 어느 정도 이해하고 있는 엔지니어가 통합 작업을 수행할 것을 권장합니다.

### 빠른 시작 DLSS3 프레임 생성

참고: DLSS 프레임 생성과 Reflex 구현은 "Streamline" 라이브러리를 통해 함께 제공되므로 플러그인 이름은 Streamline입니다.

### 에픽게임즈 런처의 패키지 엔진(UE 5.2 이상)

DLSS 프레임 생성 플러그인은 5.2 이전의 패키지 엔진 버전은 지원하지 않습니다. GitHub의 소스 엔진을 사용하고 이전 엔진에 패치를 적용해야 하며, 자세한 지침은 아래를 참조하세요.

1. 엔진의 플러그인\마켓플레이스 폴더 또는 소스 프로젝트의 플러그인 폴더 아래에 전체 Streamline 플러그인 폴더를 복사합니다.
  - 에픽에서 패키지 엔진 릴리스를 받은 경우, 엔진의 Engine\Plugins\Marketplace 폴더 아래 플러그인을 복사하세요◦ 소스 프로젝트(블루프린트 전용 프로젝트가 아닌)가 있는 경우 프로젝트의 플러그인 폴더에 플러그인을 복사할 수도 있습니다
- 에 복사해야 합니다. 플러그인의 복사본은 하나만 허용되므로 두 위치에 복사하지 마세요.
2. 편집기(편집 -> 플러그인)에서 DLSS 프레임 생성 플러그인을 활성화합니다.
3. 편집기를 다시 시작합니다.
4. NVIDIA Streamline 지원 로그 확인 1

## GitHub의 소스 엔진

버전 5.2 이전 소스 엔진의 경우 아래의 커스텀 브랜치를 사용하여 엔진 패치를 적용해야 합니다.

5.2 버전 이후의 소스 엔진의 경우 엔진의 Engine\Plugins\Runtime 폴더 아래에 플러그인을 복사하고 엔진을 빌드할 수 있습니다. 하지만 아래의 커스텀 5.2 브랜치에 DLSS-FG에 대한 엔진 수정 사항이 포함되어 있으므로 가능하면 커스텀 브랜치를 병합하는 것이 좋습니다.

1. [여기](#)에 설명된 액세스 요구 사항을 따라 모든 언리얼 엔진 디포에 액세스할 수 있는지 확인하세요.
2. 커스텀 엔진 변경 사항을 커스텀 플러그인 후크, DLSS 프레임 생성, DLSS 슈퍼 해상도 플러그인을 사용하여 UE 버전과 일치하도록 소스 트리에 병합합니다:
  - ∞ <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-5.2-dlss-plugin>
  - ∞ <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-5.1-dlss-plugin>
  - ∞ <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-5.0-dlss-plugin>
  - ∞ <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-4.27-dlss-plugin>
3. 편집기에서 DLSS 프레임 생성 플러그인을 활성화합니다.
4. 편집기를 다시 시작합니다.
5. NVIDIA Streamline 지원 로그 확인 1

## DLSS 프레임 생성을 위한 시스템 요구 사항

- 최소 Windows OS 버전은 Win10 20H1(버전 2004, 빌드 19041 이상), 64비트입니다.
  - 디스플레이 하드웨어 가속 GPU 스케줄링(HWS)은 설정 : 시스템을 통해 활성화해야 합니다: 디스플레이 : 그래픽 : 기본 그래픽 설정 변경. <https://devblogs.microsoft.com/directx/hardware-accelerated-gpu-scheduling/>
  - NVIDIA Ada 아키텍처 GPU(GeForce RTX 40 시리즈, NVIDIA RTX 6000 시리즈)
- NVIDIA 지포스 드라이버
- 권장: 버전 536.99 이상
- DirectX 12를 사용하는 UE 프로젝트(프로젝트 설정의 기본 RHI)

## DLSS 프레임 생성 플러그인 지원을 위한 커스텀 엔진 변경사항 병합 및 통합 언리얼 엔진 5.2

1. 다음 두 가지 방법 중 하나를 사용하여 이 리포지토리에서 필요한 엔진 측 변경 사항을 적용합니다.
  1. 코드 트리가 git 기반인 경우 이 브랜치를 직접 병합하세요 <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-5.2-dlss-plugin>
  2. 또는 다음 링크에서 패치 파일을 다운로드하세요:
   
<https://github.com/EpicGames/UnrealEngine/compare/release...NvRTX:dlss3/sl2-5.2-dlss-plugin.patch>
    1. 다음 명령을 실행하여 패치 파일이 엔진 버전과 호환되는지 확인합니다: git apply --check release...NvRTX:dlss3/sl2-5.2-dlss-plugin.patch
    2. 이 작업은 비파괴적이며 순차적 병합이 가능한지 확인하고 문제를 표시하기 위한 테스트입니다.
    3. 문제가 발생하면 병합 중인 파일을 확인하세요. 이는 개발자가 직접 파일을 수정했을 수 있으며 git 병합 테스트가 실패할 때 발생할 수 있습니다.
    4. 다음 명령으로 패치를 적용합니다: git apply release...NvRTX:dlss3/sl2-5.2-dlss-plugin.patch.
2. 패치된 엔진 버전과 애플리케이션을 다시 빌드합니다.
3. 편집기에서 DLSS 프레임 생성 플러그인을 활성화합니다.
  1. 플러그인 브라우저로 이동합니다.
  2. 플러그인을 찾을 때까지 끝까지 아래로 스크롤합니다.
4. 편집기를 다시 시작합니다.
5. UE에서 DLSS 프레임 생성 통합 검증하기
  1. 편집기 또는 게임을 실행합니다.
  2. 디버그 출력 및/또는 \$(프로젝트 이름)/저장된/로그에서 로그를 확인하고 다음 줄을 찾습니다: NVIDIA Streamline 지원됨 1
6. 문제가 지속되면 주저하지 말고 NVIDIA 담당자에게 문의하세요.

UE DLSS 프레임 생성 플러그인 자체는 다른 UE 버전에서도 동일하지만, 각 UE 버전마다 엔진이 변경된 다른 브랜치/패치가 Github에 있습니다:

## UE 5.1

- 5.2 지침과 유사
- Branch
    - <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-5.1-dlss-plugin>
  - 패치
    - <https://github.com/EpicGames/UnrealEngine/compare/release...NvRTX:dlss3/sl2-5.1-dlss-plugin.patch>

## UE 5.0

- 5.2 지침과 유사
- Branch
    - <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-5.0-dlss-plugin>
  - 패치
    - <https://github.com/EpicGames/UnrealEngine/compare/release...NvRTX:dlss3/sl2-5.0-dlss-plugin.patch>

## **UE 4.27**

### 5.2 지침과 유사

- Branch
  - <https://github.com/NvRTX/UnrealEngine/tree/dlss3/sl2-4.27>
- dlss-plugin패치
  - <https://github.com/EpicGames/UnrealEngine/compare/release...NvRTX:dlss3/sl2-4.27-dlss-plugin.patch>

## 문제 해결

### 빠른 팁

1. 사용 중인 DirectX 버전을 확인하는 방법: 언리얼 에디터에서 출력 로그를 엽니다. "RHI"를 검색하여 사용 중인 DirectX 버전을 확인합니다. 예를 들어 DX12인 경우 "LogD3D12"와 같은 내용이 표시됩니다.
2. 프레임 생성이 올바르게 초기화되었는지 확인하는 방법: 출력 로그를 사용하여 스트림라인을 검색하면 초기화가 성공했는지 실패했는지 여기에서 확인할 수 있습니다.
3. 사용 중인 Windows 버전을 확인하는 방법 시작 또는 Windows 버튼(일반적으로 컴퓨터 화면 왼쪽 하단에 있음)을 클릭합니다. 설정을 클릭합니다. 시스템을 클릭합니다. 정보(일반적으로 화면 왼쪽 하단에 있음)를 클릭합니다. 결과 화면에 Windows 버전이 표시됩니다.

### 알려진 문제

#### DLSS 프레임 생성

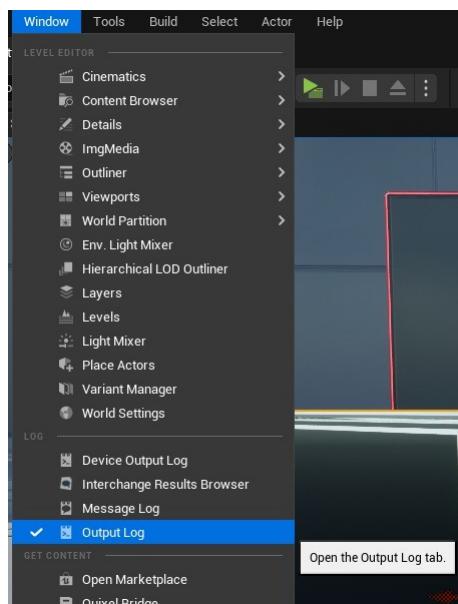
- 이전 드라이버를 사용하면 SER(세이더 실행 재정렬)이 동시에 활성화된 경우 DLSS-FG를 활성화할 때 "장치 제거" 오류가 발생할 수 있습니다. 이 문제를 방지하려면 드라이버 536.99 이상을 권장합니다.

### 에디터에서 플러그인 문제 진단하기

UE DLSS 프레임 생성 플러그인 모듈은 다양한 정보를 다음 UE 로그 카테고리에 기록합니다. •

- LogStreamline
- LogStreamlineAPI
- 로그스트림라인 블루프린트
- LogStreamlineD3D11RHI
- LogStreamlineD3D12RHI
- LogStreamlineRHI
- LogStreamlineRHIPreInit

에디터의 창 -> 출력 로그에서 액세스할 수 있습니다.



그런 다음 메시지 로그를 필터링하여 Streamline 관련 메시지만 표시하도록 하여 플러그인이 예상대로 작동하지 않는 이유에 대한

자세한 정보를 얻을 수 있습니다.

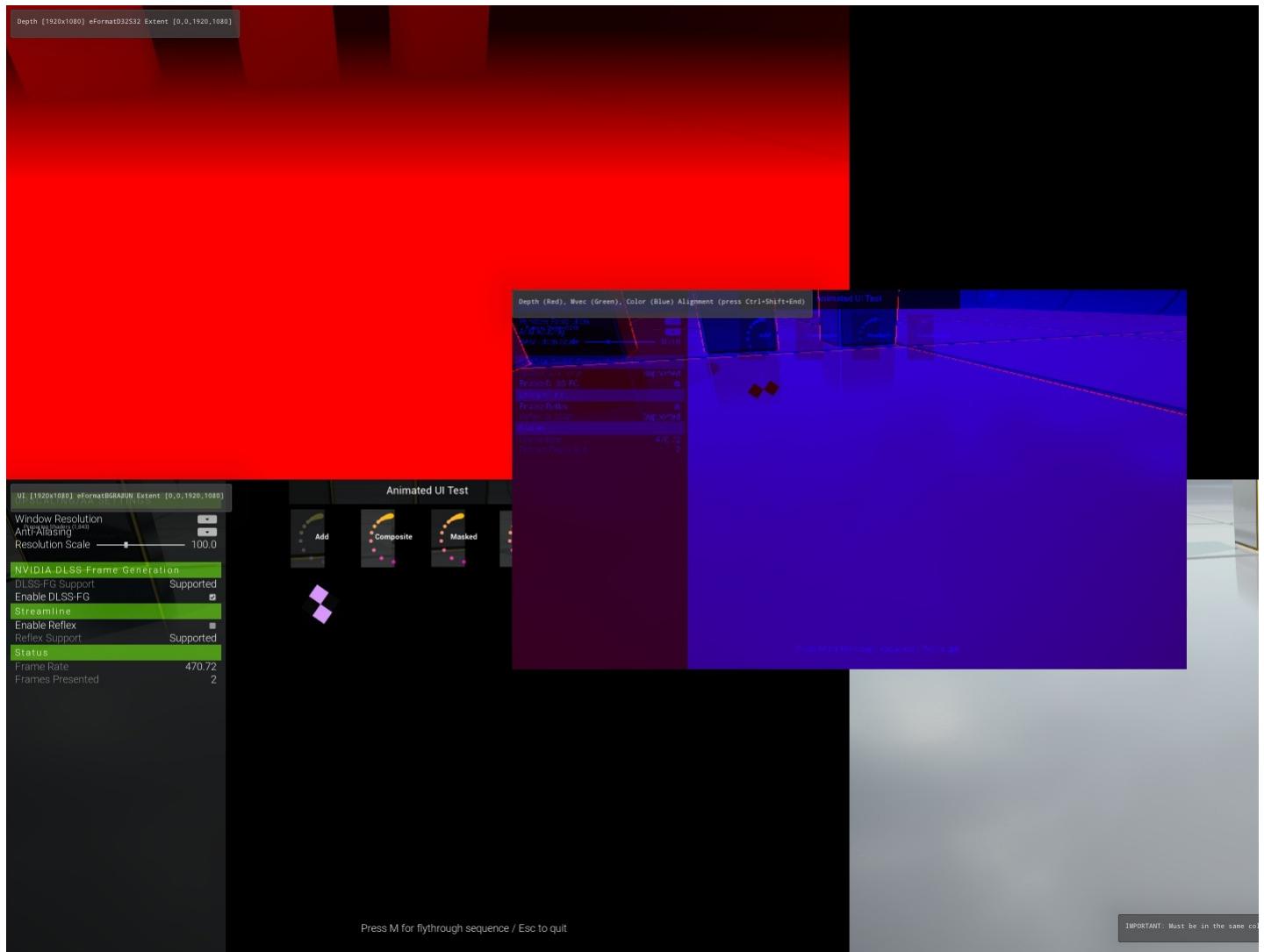
## 디버그 오버레이

비출시 UE 빌드에서 DLSS 프레임 생성 플러그인은 런타임 정보를 표시하는 디버그 오버레이를 표시할 수 있습니다. 오버레이는 비출시 빌드에서 프로젝트 세팅의 플러그인 -> NVIDIA DLSS 프레임 생성 -> 디버그 오버레이 로드 옵션으로 활성화/비활성화할 수 있습니다.

이 설정은 명령줄에서 `-s1debugoverlay` 및 `-slnodebugoverlay`를 사용하여 재정의할 수도 있습니다. 이렇게 하면 개발 바이너리 간소화가 암시적으로 선택됩니다.

오버레이는 DLSS-FG 기능이 지원되는 시스템에서만 지원됩니다.

디버그 오버레이는 DLSS-FG용 스트림라인에 전달된 다양한 텍스처를 시각화할 수 있는 방법을 제공합니다. DLSS-FG를 활성화한 경우 Ctrl+Shift+삽입하면 다음 이미지 보기 활성화됩니다.



## 프레임 속도 제한

언리얼 엔진이 프레임 속도 제한을 처리하는 방식 때문에 프레임 속도가 최소 프레임 속도에 멈춰서 회복되지 않는 문제가 발생할 수 있습니다. 콘솔 변수 `t.Streamline.Reflex.HandleMaxTickRate`를 `False`로 설정하면 엔진이 스트림라인 리플렉스 대신 최대 틱 속도를 제한하도록 하면 이러한 상황에 도움이 될 수 있습니다. 엔진은 DLSS 프레임 생성을 인식하지 못하므로 이 변수를 `False`로 설정하면 실제 그래픽 프레임 속도는 엔진의 최대 FPS가 설정된 것보다 두 배가 될 수 있습니다.

## 콘텐츠 고려 사항 UI 알파의 올바른 렌더링

### 렌더링

DLSS-FG는 UI 컬러와 알파 버퍼를 사용하여 합성된 UI의 이미지 퀄리티를 높일 수 있습니다. UE DLSS 프레임 생성 플러그인은 UI가 렌더링된 후(현재 바로 전) 알파 채널을 추출하여 백버퍼에서 이를 생성합니다.

UE는 기본적으로 시나리오 컬러의 알파를 지우지 않는데, 모든 UMG 머티리얼 블렌딩 모드는 아니더라도 대부분의 경우 들어오는 알파만 고려하고 모든 픽셀을 쓰기 때문에 정상적으로 작동합니다. 그러나 개발자 콘솔 창은 UI 요소 다음에 렌더링되어 알파 채널에 기록됩니다. 이 알파는 여러 프레임에 걸쳐 지속되므로 잘못된 UI 컬러와 알파 버퍼가 UE Streamline 플러그인에 의해 생성된 다음 Streamline/DLSS-FG로 전달되는 결과를 초래합니다.

엔진은 씬 컬러의 알파만 지울 수 있는 내장된 방법을 제공하지 않습니다. 따라서 UE 스트림라인 플러그인은 포스트 프로세싱이 끝날 때, UI 요소가 렌더링되기 전에 씬 컬러의 알파 채널을 지우는 렌더패스를 추가합니다. 이는 기본적으로 True인 `r.Streamline.ClearSceneColorAlpha`로 제어됩니다.

또한 Streamline 플러그인은 임계값을 추가하여 픽셀의 알파가 임계값보다 큰 픽셀을 UI로 결정하기 위한 임계값인 `r.Streamline.TagUIColorAlphaThreshold`를 추가합니다. 기본적으로 이 값은 0.0으로 설정되어 알파가 0보다 큰 모든 픽셀이 UI 색상 및 알파 버퍼로 추출됩니다.

참고: UI 색상 알파 지원은 독립형 게임 창에서만 지원됩니다. 에디터 PIE 팝업 창에서 UI 색상 및 알파 태그 지정

알파 채널을 지우는 것이 간단하지 않기 때문에 기본적으로 비활성화되어 있습니다

(r.Streamline.Editor.TagUIColorAlpha 참조): PIE 창은 씬을 별도의 'BufferedRT' 렌더 타깃으로 렌더링하고 백버퍼로 블릿한 다음 그 위에 UI를 그린 다음 프레젠테이션합니다. 이 'BufferedRT' 중간 단계는

r.Streamline.ClearSceneColorAlpha가 의도한 대로 작동하지 못하게 합니다. PIE 에디터 창의 이미지 품질은 대표적이지 않지만 블루프린트 라이브러리를 사용하여 UI 및 설정 로직을 개발하는 데는 충분합니다.

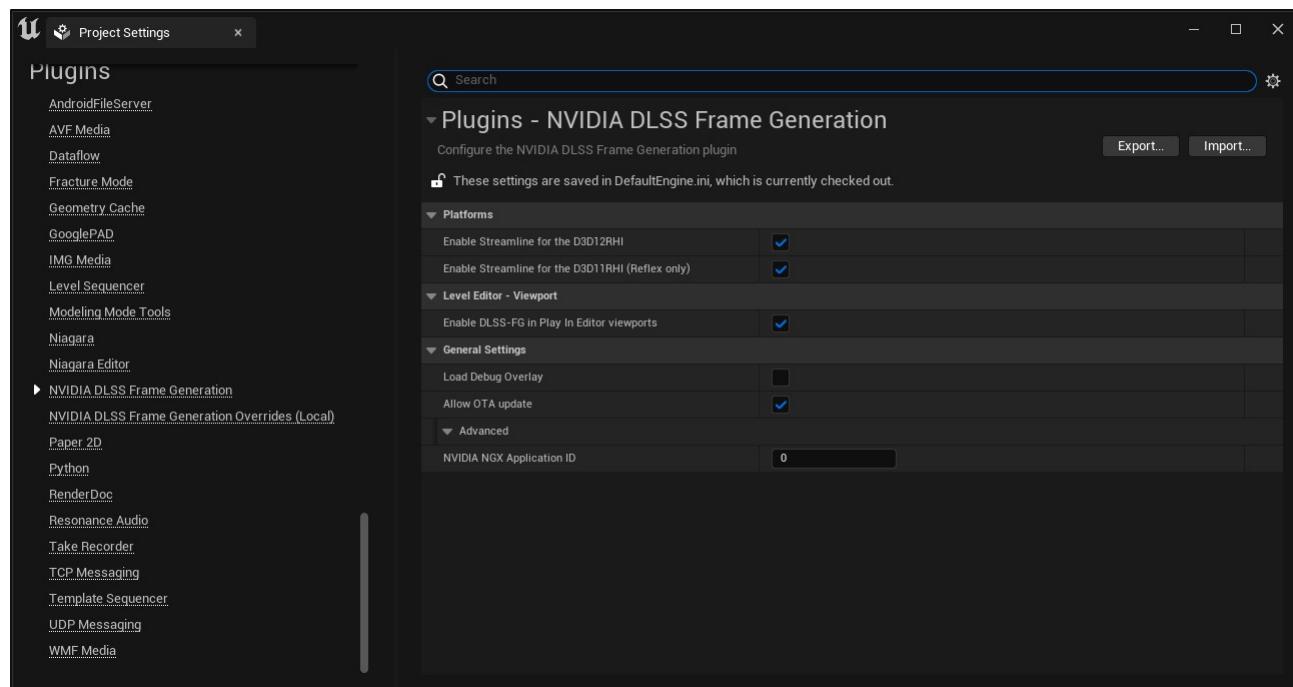
그러나 이는 애플리케이션이 예상한 방식으로 UI를 렌더링하는 경우에만 작동하므로 모든 UI 요소는 불투명 또는 투명 여부에 관계 없이 백버퍼에 알파를 써야 합니다. 실제로 이것은

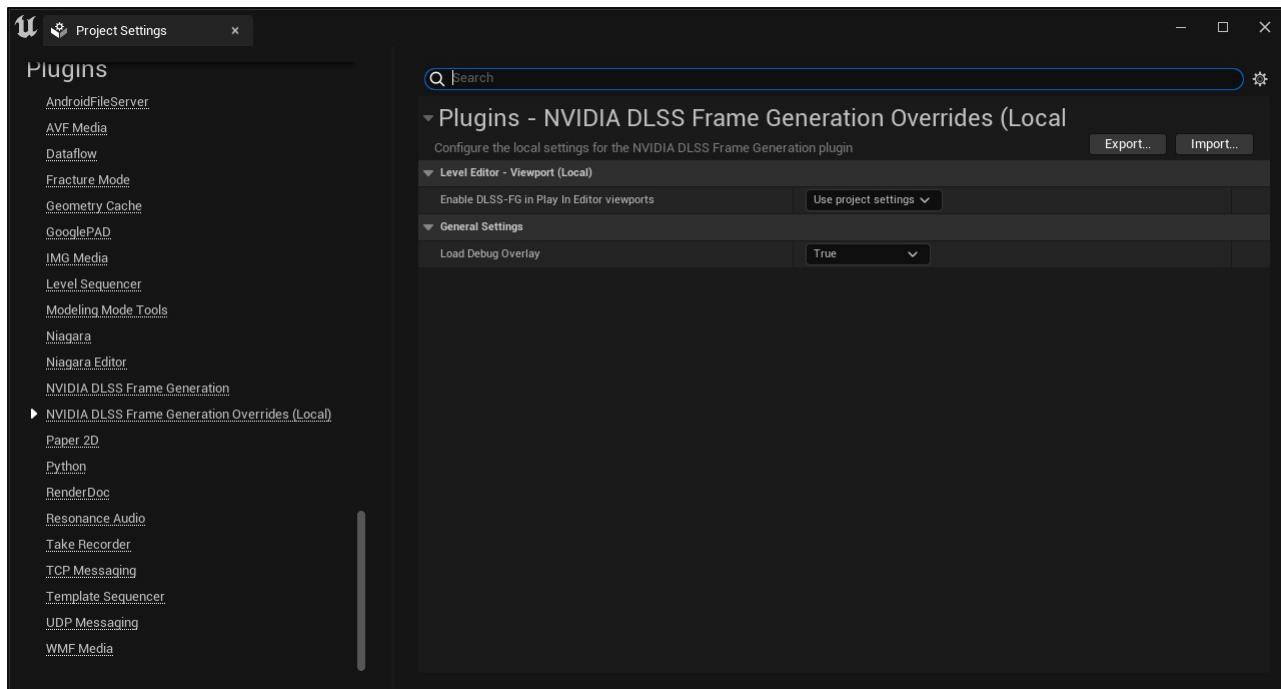
- 알파를 쓰지 않으므로 UI 텍스트를 그리는 데 UWidgetBlueprintLibrary::DrawText를 사용하지 마세요.
- UMG 위젯은 적절한 블렌딩 모드로 알파를 백버퍼에 올바르게 쓰기 때문에 대신 사용하세요.

## 플러그인 구성

일부 프로젝트 설정은 두 개의 구성 파일로 나뉘며, 로컬 재정의가 가능합니다.

- 프로젝트 설정 -> 플러그인 -> NVIDIA DLSS 프레임 생성
  - DefaultEngine.ini에 저장된
  - 는 일반적으로 소스 제어에 있습니다.
  - 설정은 사용자 간에 공유됩니다.
- 프로젝트 설정 -> 플러그인 -> NVIDIA DLSS 프레임 생성 오버라이드(로컬)
  - 저장된 UserEngine.ini
  - 소스 제어에 체크인하지 않는 것이 좋습니다.
  - 사용자가 원하는 경우 프로젝트 전체 설정을 재정의할 수 있도록 허용합니다. 기본값은 "프로젝트 설정 사용"입니다.





## 명령줄 옵션 및 콘솔 변수 및 명령어

**참고:** UE DLSS 프레임 생성 플러그인 모듈은 엔진 시작 시 콘솔 변수를 사용할 수 있기 전에 매우 일찍 로드됩니다. 따라서 다양한 디버깅 세팅은 콘솔 변수 대신 명령줄 옵션을 통해 노출됩니다.

### 간소화 바이너리 플레이어 선택

기본적으로 Streamline는 서명된 프로덕션 바이너리(예: sl.interposer.dll, sl.common.dll)를 Streamline\Binaries\ThirdParty\Win64\ 경로에서 사용합니다. 이 바이너리에는 화면상의 워터마크가 없으며 최종 사용자에게 제공되는 애플리케이션에 사용하기 위한 것입니다. 항상 UE Streamline 플러그인에 의해 패키징됩니다.

출시되지 않은 UE 빌드의 경우, -slbinaries={개발, 디버그} 명령줄 인수를 통해 대체 바이너리를 선택할 수 있으며, 각각 Streamline\Binaries\ThirdParty\Win64\개발 및 Streamline\Binaries\ThirdParty\Win64\Debug 경로에 해당합니다. 이 경로에는 온스크린 워터마크가 있으며 애플리케이션 개발 중에 사용하기 위한 것으로 Streamline 디버그 오버레이를 사용하기 위한 필수 조건입니다. 이들은 출시되지 않은 UE 빌드용 UE Streamline 플러그인에 의해서만 패키징됩니다.

### 로깅

기본적으로 Streamline은 슬래그 레벨 디폴트를 사용합니다. 이는 명령줄 인수 -slloglevel={0,1,2}를 사용하여 변경할 수 있으며, 각각 sl::eLogLevelOff, sl::eLogLevelDefault, sl::eLogLevelVerbose에 해당합니다.

기본적으로 Streamline 콘솔 창은 꺼져 있습니다. 명령줄 인수 -sllogconsole={0,1}로 이 값을 변경할 수 있습니다.

### DLSS 프레임 생성 일반 설정

DLSS 프레임 생성 플러그인은 다양한 엔진 사이드 퀴리를 사용하며, 다음 변수로 구성할 수 있습니다. 기본값은 다음과 같습니다.

- r.Streamline.ViewIdOverride
  - 0: ViewState.UniqueID 사용
  - 1: 보기 ID를 0으로 설정(기본값)
- r.Streamline.TagSceneColorWithoutHUD
  - HUD가 없는 씬 컬러를 DLSS 프레임 생성에 전달(기본값 = true)
- r.Streamline.Editor.TagSceneColorWithoutHUD
  - 에디터의 DLSS 프레임 생성에 HUD가 없는 씬 컬러를 전달합니다(기본값 = false)
- r.Streamline.ClearSceneColorAlpha
  - 후속 UI 드로콜이 알파 채널에서 올바르게 표현되도록 스트림라인 뷰 확장 끝에서 시나리오 컬러의 알파를 지웁니다(기본값 = false)

값=true).

- r.Streamline.Editor.TagUIColorAlpha

- 실험적: 에디터 PIE 창에서 UI 색상 및 알파를 Streamline에 전달(기본값 = false)

## DLSS 프레임 생성을 위한 모션 벡터 미세 조정하기

DLSS 프레임 생성이 제대로 작동하려면 올바른 모션 벡터가 필요합니다. 게임 개발 중에 값을 조정하는 데 사용할 수 있는 콘솔 변수는 다음과 같습니다.

- r.Streamline.DilateMotionVectors
  - 0: 저해상도 모션 벡터를 DLSS 프레임 생성에 전달합니다(기본값).
  - 1: 확장된 고해상도 모션 벡터를 DLSS 프레임 생성에 전달합니다. 이렇게 하면 얇은 디테일의 이미지 품질을 개선하는 데 도움이 됩니다.
- r.Streamline.MotionVectorScale
  - 이 상수에 따라 DLSS 프레임 생성 모션 벡터의 스케일과 뷰 렉 크기의 1/에 해당하는 스케일을 조정합니다. (기본값 = 1.0)

## DLSS 프레임 생성을 위한 깊이 미세 조정

- r.Streamline.CustomCameraNearPlane
  - 카메라 근처 평면까지의 커스텀 거리. 내부 DLSS 프레임 생성 목적으로 사용되며, 엔진에서 사용하는 해당 값과 일치할 필요는 없습니다. (기본값 = 0.01)
- r.Streamline.CustomCameraFarPlane
  - 카메라 원거리 평면까지의 커스텀 거리. 내부 DLSS 프레임 생성 목적으로 사용되며, 엔진에서 사용하는 해당 값과 일치할 필요는 없습니다. (기본값 = 75000.0)

## 반사 간소화

UE DLSS 프레임 생성 플러그인은 수정되지 않은 버전의 UE와 함께 제공되는 기존 UE Reflex 플러그인과 반호환되는 NVIDIA Reflex의 구현을 제공합니다.

기존 UE Reflex 플러그인을 비활성화하고 UE DLSS 프레임 생성 플러그인에서 제공하는 Reflex 구현을 사용하는 것이 좋습니다. 그러나 UE Reflex 플러그인의 블루프린트 기능을 사용하는 기존 프로젝트는 Reflex 플러그인의 블루프린트가 기본적으로 DLSS 프레임 생성 플러그인의 모듈식 기능을 호출해야 하므로 UE DLSS 프레임 생성 플러그인을 추가한 후에도 계속 작동해야 합니다.

## Reflex 블루프린트 라이브러리

Reflex (Streamline) / UStreamlineLibraryReflex 블루프린트 라이브러리는 Reflex 지원 여부를 쿼리하는 권장 방법이며, Reflex를 구성하는 함수도 제공합니다.

- IsReflexSupported, 쿼리 리플렉스 지원
- SetReflexMode, GetReflexMode, GetDefaultReflexMode
- GetGameToRenderLatencyInMs, GetGameLatencyInMs, GetRenderLatencyInMs

## 콘솔 변수(로우 레벨)

다음 콘솔 변수를 사용하여 구성할 수 있습니다.

- t.Streamline.Reflex.Enable
  - 리플렉스 간소화 확장을 활성화합니다. (기본값 = 0)
    - 0: 사용 안 함
    - 1: 사용
- t.Streamline.Reflex.Auto
  - 다른 SL 기능이 필요할 때 Streamline Reflex 확장을 활성화합니다. (기본값 =
    - 0: 사용 안 함
    - 1: 사용
- t.Streamline.Reflex.EnableInEditor
  - 편집기에서 리플렉스 간소화를 활성화합니다. (기본값 = 0)
    - 0: 사용 안 함
    - 1: 사용
- t.Streamline.Reflex.Mode
  - 간소화 반사 모드(기본값 = 1)
    - 0: 꺼
    - 1: 켜

짐

- 1: 짧은 지연 시간
- 2: 부스트를 통한 낮은 지연 시간

- t.Streamline.Reflex.HandleMaxTickRate

- 스트림라인 리플렉스가 엔진 대신 프레임 속도 제한을 처리할지 여부를 제어합니다(기본값 = true)
  - false: 엔진이 프레임 속도 제한을 처리합니다.
  - true: 리플렉스가 프레임 속도 제한을 처리합니다.

이 콘솔 변수를 사용하여 DLSS 프레임 생성 플러그인이 제공하는 Reflex와 UE Reflex 플러그인 간의 상호작용을 구성할 수 있습니다:

- r.Streamline.UnregisterReflexPlugin

- 기존 NVAPI 기반 UE Reflex 플러그인은 DLSS 프레임 생성 기반 구현과 호환되지 않습니다. 이 변수는 Reflex 플러그인을 엔진에서 등록 해제할지 여부를 제어합니다.
- 0: Reflex 플러그인 모듈식 기능 등록 유지

- 1: Reflex 플러그인 모듈 기능 등록을 해제합니다. Reflex 블루프린트 라이브러리는 DLSS 프레임 생성 플러그인 모듈식 기능과 함께 작동해야 합니다(기본값).

## 반사 통계

Reflex 또는 DLSS 프레임 생성이 활성화된 경우 Reflex는 게임 스레드 지연을 처리하여 t.MaxFPS 또는 프레임 속도 스무딩을 통해 요청된 프레임 속도 제한을 적용할 수 있습니다. 통계 스레딩의 "게임 스레드 대기 시간(Reflex)" 통계는 Reflex가 게임 스레드를 지연시키는 데 소요된 시간을 보여줍니다.

## DLSS 프레임 생성

UE DLSS 프레임 생성 플러그인은 NVIDIA DLSS 프레임 생성(DLSS-FG)의 구현을 제공합니다. DLSS-FG는 패키지 빌드(또는 -게임 모드에서 에디터 실행)에서 완벽하게 지원됩니다.

DLSS-FG는 편집기에서 부분적으로 지원되며 다음과 같은 제한 사항이 있습니다:

- 기본 편집기 창은 DLSS-FG를 지원하지 않습니다. 따라서 '선택한 뷰포트에서 재생'은 지원되지 않습니다.
- PIE(새 창)는 단일 PIE 창에 대해 지원됩니다. 네트워크 디버깅을 위해 여러 개의 PIE 창을 여는 경우(예: 네트워크 디버깅) 첫 번째 창에만 DLSS-FG가 지원됩니다.
  - 이 옵션을 활성화/비활성화하려면 프로젝트 세팅의 '에디터 뷰포트에서 플레이 시 DLSS-FG 활성화' 옵션을 사용합니다.

참고: 최적의 성능을 위해 DLSS-FG도 Reflex를 사용하도록 설정해야 합니다. 이 작업은 자동으로 수행됩니다. 자세한 내용은 t.Streamline.Reflex.Auto를 참조하세요.

## DLSS-FG 블루프린트 라이브러리

DLSS-FG (Streamline) / UStreamlineLibraryDLSSG 블루프린트 라이브러리는 DLSS-FG 지원 여부를 쿼리하는 데 권장되며, DLSS-FG를 구성하는 기능도 제공합니다.

- IsDLSSGSupported, QueryDLSSGSupport, GetDLSSGMinimumDriverVersion
- IsDLSSGModeSupported, GetSupportedDLSSGModes
- SetDLSSGMode, GetDLSSGMode, GetDefaultDLSSGMode, GetDLSSGFrameTiming

## 콘솔 변수(로우 레벨)

다음 콘솔 변수를 사용하여 구성할 수 있습니다.

- r.Streamline.DLSSG.Enable
  - DLSS-FG 모드(기본값 = 0)
    - 0: 꺼짐
    - 1: 항상 켜짐
    - 2: 자동 모드(프레임 속도에 도움이 되는 경우)
  - 예만 켜짐)
- r.Streamline.DLSSG.AdjustMotionBlurTimeScale
  - DLSS-FG가 활성화된 경우, 생성된 프레임을 기준으로 모션 블러 타임스케일을 조정합니다(기본값 = 1)
    - r.Streamline.TagUIColorAlpha
      - UI 색상 및 알파를 DLSS 프레임 생성에 전달합니다(기본값 = true).

## DLSS-FG 자동 모드

"DLSS-FG 모드 설정" 블루프린트 노드로 DLSS-FG를 활성화할 때는 자동 모드를 사용하는 것이 좋습니다. 자동 모드에서는 프레임 속도를 감소시킬 수 있는 경우 DLSS-FG가 자동으로 비활성화됩니다. 애플리케이션에서 자동 모드를 처음 활성화할 때 활성화 직후 약 2초 동안 DLSS-FG가 비활성화되는 것을 볼 수 있습니다. 이는 초기 자동 모드 보정 중 정상적인 동작입니다. 콘솔 명령 r.Streamline.DLSSG.Enable 2를 사용하여 자동 모드를 활성화할 수도 있습니다.

## 통계

엔진은 프레임 생성으로 인해 생성된 추가 프레임을 직접 인식하지 못하므로 기본 제공 프레임 속도 및 프레임 시간 지표에 정보가 누락될 수 있습니다. "DLSSG" 통계 그룹은 추가 프레임을 고려한 프레임 속도 통계를 제공합니다. 화면 정보를 보려면 콘솔 명령 stat dlssg를 사용하세요.

## 팁 및 모범 사례

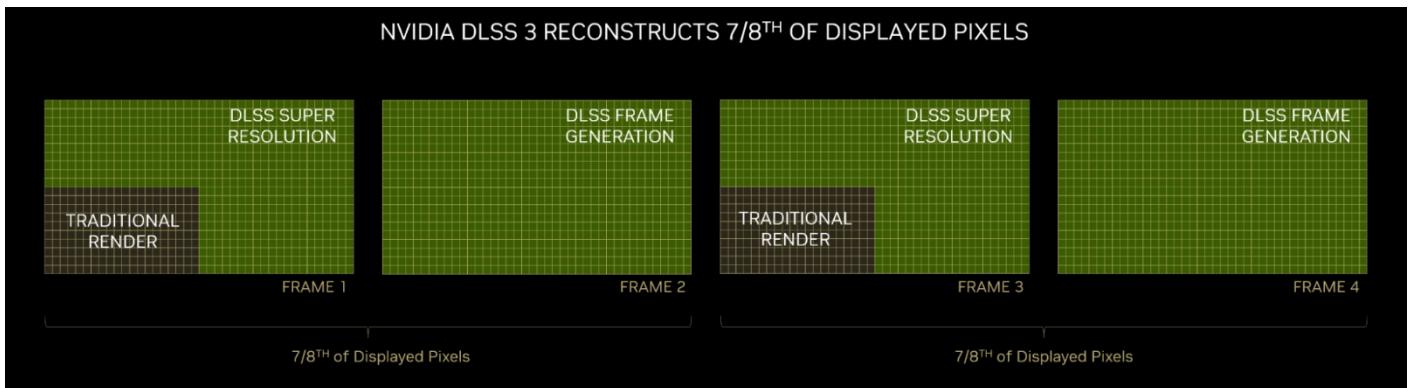
1. 스트림라인/프레임 생성이 활성화되면 에디터 내 PIE 또는 독립형 창에서 r.streamline.dlssg.enable 1 명령을 입력하거나 블루프린트 스크립트 함수("streamline"로 검색하여 함수 목록을 확인)를 사용하여 플레이 시작 시 활성화할 수 있습니다.
2. 프로젝트 설정으로 이동한 다음 "NVIDIA DLSS 프레임 생성" 플러그인에 대한 기본 설정으로 이동합니다. 여기에서 '로드 디버그 오버레이' 옵션을 토글하여 실시간 통계와 함께 DLSS 프레임 생성이 작동하는지 빠르고 편리하게 확인할 수 있습니다.
3. 동일한 설정 창에서 "OTA 업데이트 허용" 옵션이 활성화되어 있는지 확인하면 Streamline도 자동으로 업데이트됩니다.

를 최신 개선 사항이 적용된 DLSS의 AI 알고리즘으로 사용합니다.

4. 프레임 생성을 위한 디버그 오버레이는 에디터에서 작동하며 개발/디버그 빌드에는 표시될 수 있지만, 출시 빌드에는 표시되지 않는다는 점에 유의하세요.
5. 언리얼 에디터에서 프레임 생성은 새 에디터 창(PIE) 또는 독립형 모드에서만 작동하며, 선택된 뷰포트나 편집 도구에는 작동하지 않습니다.
6. 프레임 생성이 켜져 있을 때는 애플리케이션에서 Vsync를 끄는 것이 좋습니다. DLSS 3 플러그인이 활성화되면 Vsync가 잘 못 작동하도록 설정할 수 있습니다. r.vsync 0 콘솔 명령으로 Vsync를 비활성화할 수 있습니다.
7. NVIDIA DLSS 프레임 제너레이션 언리얼 엔진 플러그인에는 현재 언리얼 엔진에 내장된 Reflex보다 최신 버전인 최신 NVIDIA Reflex 기술이 포함되어 있습니다. 이전 플러그인을 활성화한 상태로 유지하고 이전 Reflex 블루프린트 스크립트를 사용할 수도 있지만, 이전 Reflex 플러그인을 비활성화하고 대신 DLSS 프레임 생성에 번들로 제공되는 새 버전을 사용할 것을 권장합니다.
8. UI 메뉴에서 플러그인을 편리하게 활성화하고 사용자 환경 설정을 할 수 있으므로 모든 NVIDIA 플러그인은 블루프린트 스크립트를 통해 설정하는 것이 좋습니다. 그러나 DLSS 고해상도 콘솔 명령에 액세스해야 하는 경우 'ngx' 아래에서 찾을 수 있으며, DLSS 프레임 생성 명령은 'streamline' 아래에서 찾을 수 있습니다. DLSS 고해상도 콘솔 명령어 사용에 대한 자세한 내용은 DLSS 3 플러그인 다운로드에 포함된 DLSS\_Super\_Resolution\_Quick\_Start\_Guide.pdf를 참조하세요.

## 성능 이점에 대한 기대

DLSS 3은 성능 승수입니다. 이 기능을 활성화하면 여러 상황에서 프레임 속도를 크게 향상시킬 수 있습니다.



하지만 여기에는 몇 가지 주의 사항과 주의사항이 있으므로 다음 사항에 유의하세요:

1. DLSS 슈퍼 해상도는 언리얼 엔진에서 자체적으로 프레임 속도를 향상시키지 못할 수 있습니다. 그 이유는 UE의 기본 동작이 업스케일링을 사용하고 창 크기에 따라 자동으로 업스케일링된 해상도를 관리하기 때문입니다. 따라서 DLSS 슈퍼 해상도를 켜면 AI 향상된 업스케일링의 이점을 누릴 수 있지만, 기본값과 동일한 입력 해상도를 사용합니다.
2. 즉, DLSS SR은 더 낮은 입력 해상도를 사용하여 동일한 유효 화질을 얻을 수 있으며, DLSS는 33%(울트라 성능) 모드까지 입력 해상도를 지원합니다.
3. 장면과 입력 해상도에 따라 1.5배에서 3배 이상의 성능 배율을 기대할 수 있습니다.
4. 100% 기본 해상도, 4K, 16fps로 실행되는 복잡한 장면



5. 33% DLSS 해상도, 4K, 60fps로 실행되는 동일한 장면



6. CPU가 제한된 상황에서는 GPU가 작업을 위해 고갈되기 때문에 DLSS SR의 프레임 속도가 크게 향상되지 않을 수 있습니다.  
DLSS SR에서 최대 성능을 얻으려면 최대 용량으로 작동하지 못하게 할 수 있는 CPU 관련 병목 현상을 완화해야 합니다.
7. DLSS 프레임 생성은 많은 CPU 및 GPU 제약 상황을 완화하는 데 도움이 될 수 있으며, 성능 향상은 다음과 같이 다양할 수 있습니다.  
1.5배에서 2.2배 이상.
8. 33% DLSS 해상도 및 프레임 생성, 4K, 100fps로 실행되는 동일한 장면



9. DLSS SR과 FG 모두 설정에 약간의 초기 비용이 발생하며, 이 비용은 장면과 해상도에 따라 달라집니다. 일반적으로 이러한 성능 비용은 0.5밀리초에서 2밀리초 이상일 수 있지만, 목표는 순 성능 향상이 비용보다 더 크다는 것입니다.