NVIDIA 언리얼 엔진 DLSS 초고해상도 / DLAA 플러그인

DLSS3 및 NVIDIA 언리얼 엔진 DLSS 초고해상도 플러그인

NVIDIA *DLSS 슈퍼 해상도/광선 재구성/DLAA* 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 슈퍼 샘플링 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 앨리어싱(DLAA)*은 이미지 품질을 개선하는 데 사용됩니다. *DLAA에는* NVIDIA RTX 그래픽 카드가 필요합니다. • NVIDIA *이미지 스케일링(NIS)은* NVIDIA 또는 타사 비RTX GPU에 동급 최고의 업스케일링 및 선명도를 제공합니다.
  자세한 내용은 NVIDIA *이미지 스케일링* 언리얼 엔진 플러그인을 참조하세요.
- ***출시 예정!*** - NVIDIA *광선 재구성(DLSS-RR)은* 집중적인 레이 트레이싱 콘텐츠를 위해 샘플링된 광선 사이에 고품질 픽셀을 생성하여 이미지 품질을 향상시킵니다. 이 기능을 사용하려면 GeForce RTX 그래픽 카드가 필요합니다.

NVIDIA 언리얼 엔진 DLSS-SR/DLSS-RR/DLAA 플러그인(여기에 문서화되어 있음)을 제공합니다:

- DLSS 슈퍼 해상도(DLSS-SR)
- 딥러닝 앤티 앨리어싱(DLAA)
- ***출시 예정!*** - DLSS 광선 재구성(DLSS-RR)

NVIDIA 언리얼 엔진 스트림라인 플러그인(별도 구매)이 제공합니다:

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

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

- NVIDIA 이미지 스케일링

빠른 시작

자세한 내용은 이 문서의 관련 섹션을 참조하세요.

1. 에디터에서 *NVIDIA DLSS 고해상도/광선 재구성/DLAA* 플러그인을 활성화한 다음 에디터를 재시작합니다.

2. 에디터의 DLSS-SR/DLSS-RR/DLAA: 프로젝트 플러그인 설정에서 다음 설정을 활성화합니다.

   1. 에디터 뷰포트에서 DLSS를 켜도록 설정합니다(기본값은 꺼져 있음).
   2. 뷰포트 옵션(왼쪽 상단 모서리에 있는 삼중 막대 ≡ 메뉴)에서 화면 비율 옵션을 설정하여 업스케일 비율을 제어합니다. 모든 스크린 퍼센티지가 지원되는 것은 아닙니다. DLSS의 경우 50-67 범위의 값을, DLAA의 경우 100을, DLSS-RR의 경우 50-100을 권장합니다.

3. 블루프린트에서 DLSS-SR/DLSS-RR/DLAA:

   1. EnableDLSS DLSS 블루프린트 라이브러리 함수는 DLSS 슈퍼 해상도에 대한 콘솔 변수를 편리하게 설정할 수 있는 방

법을 제공합니다. 프로젝트의 사용자 인터페이스 및 설정에 지원을 통합할 때 이 기능을 사용하는 것이 좋습니다.

2. `DLSS-RR DLSS 블루프린트 라이브러리 활성화` 기능은 DLSS 광선 재구성을 위한 콘솔 변수를 편리하게 설정할 수 있는 방법을 제공합니다. 이 기능은 프로젝트의 사용자 인터페이스 및 설정에 지원을 통합할 때 사용하는 것이 좋습니다.
3. `GetDlssModeInformation` DLSS 블루프린트 라이브러리 함수는 주어진 퀄리티 모드에 사용할 최적의 스크린 퍼센티지를 쿼리하는 데 사용할 수 있습니다. 그런 다음 화면 퍼센티지는 `ExecuteConsoleCommand` 노드에서 `r.ScreenPercentage`와 함께 사용할 수 있습니다.

4. PIE의 DLSS-SR/DLSS-RR/DLAA: 편집 -> 에디터 환경설정 -> 성능에서 "PIE의 에디터 설정으로 게임 화면 비율 설정 재정의" 끄기(기본적으로 켜져 있음)

5. 게임 내 DLSS-SR: 다음 콘솔 변수가 DLSS 슈퍼 해상도를 활성화하도록 설정되어 있는지 확인합니다:

    1. r.NGX.Enable 1(명령줄에서 -ngxenable로 재정의 가능)
    2. r.NGX.DLSS.Enable 1
    3. r.ScreenPercentage 66.7

6. 게임 내 DLAA: 다음 콘솔 변수가 DLAA를 활성화하도록 설정되어 있는지 확인합니다.

1. r.NGX.Enable 1(명령줄에서 -ngxenable로 재정의 가능)
2. r.NGX.DLSS.Enable 1
3. r.ScreenPercentage 100

7. 게임 내 DLSS-RR: DLSS-RR을 사용하려면 모든 디노이저를 비활성화해야 합니다. 프로젝트에서 레이 트레이싱을 사용할 때 가장 그렇습니다. 다음 값이 설정되어 있는지 확인하세요.

   1. r.NGX.DLSS.denoisermode 1
   2. r.Lumen.Reflections.BilateralFilter 0 - *엔진 버전 5.2 및 5.3에서 콘솔 또는 블루프린트를 통해 런타임에 이 설정을 변경하면 나중에 해상도가 변경될 경우 엔진 런타임 어설션이 발생할 수 있습니다. 이 설정은 DefaultEngine.ini와 같은 환경설정 파일만 변경해야 합니다. 따라서 DLSS-RR이 비활성화되었을 때 r.Lumen.Reflections.BilateralFilter를 다시 활성화하려면 구성 설정을 변경하고 엔진을 재시작해야 합니다.*
   3. r.Lumen.Reflections.ScreenSpaceReconstruction 0
   4. r.루멘.반사.템포럴 0
   5. r.Shadow.Denoiser 0

8. `LogDLSS 로그 확인: NVIDIA NGX DLSS 지원 1`

9. (선택 사항) 다음을 통해 화면 왼쪽 하단에 있는 DLSS 화면 표시기를 사용 설정합니다. `DLSS\Source\ThirdParty\NGX\Utils\ngx_driver_onscreenindicator.reg`로 이동하여 DLSS가 활성화되어 있는지 확인합니다.

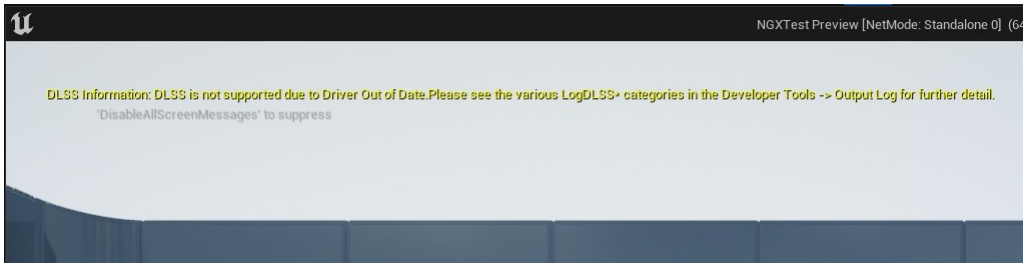
문제 해결

시스템 요구 사항

- Windows 10, 64비트
  - 버전 v1709 이상, Windows 10 2017 가을 크리에이터 업데이트 64비트.
- NVIDIA 지포스 드라이버
  - 필수: 2022년 3월 3일 이후에 발급된 드라이버(예: 512.15)
- DLSS 슈퍼 해상도를 지원하는 NVIDIA RTX GPU(GeForce, Titan 또는 Quadro)
- 다음 중 하나를 사용하는 UE 프로젝트
  - 벌칸
  - DX11
  - DX12

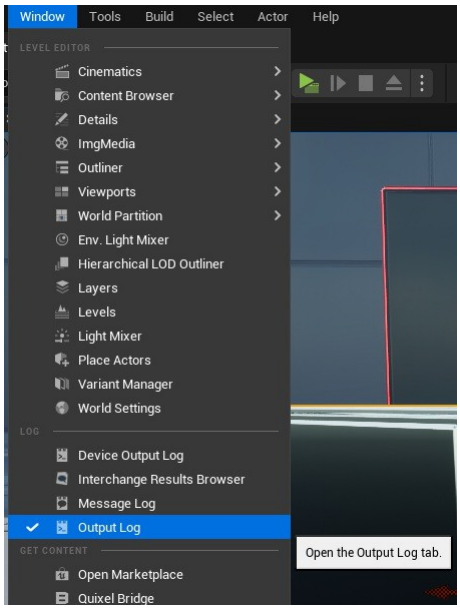
에디터에서 DLSS 문제 진단하기

DLSS 플러그인은 DLSS가 작동하지 않는 여러 가지 일반적인 이유를 화면 상단에 표시합니다(배송 빌드 구성이 아닌 경우). 이 문서의 'DLSS 플러그인 설정' 섹션에 설명된 대로 DLSS 플러그인 설정에서 이 메시지를 끌 수도 있습니다.

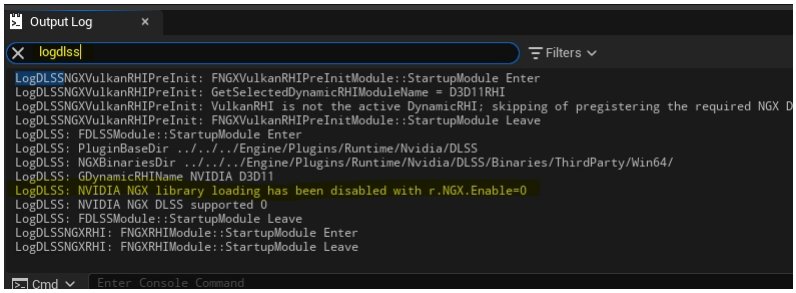


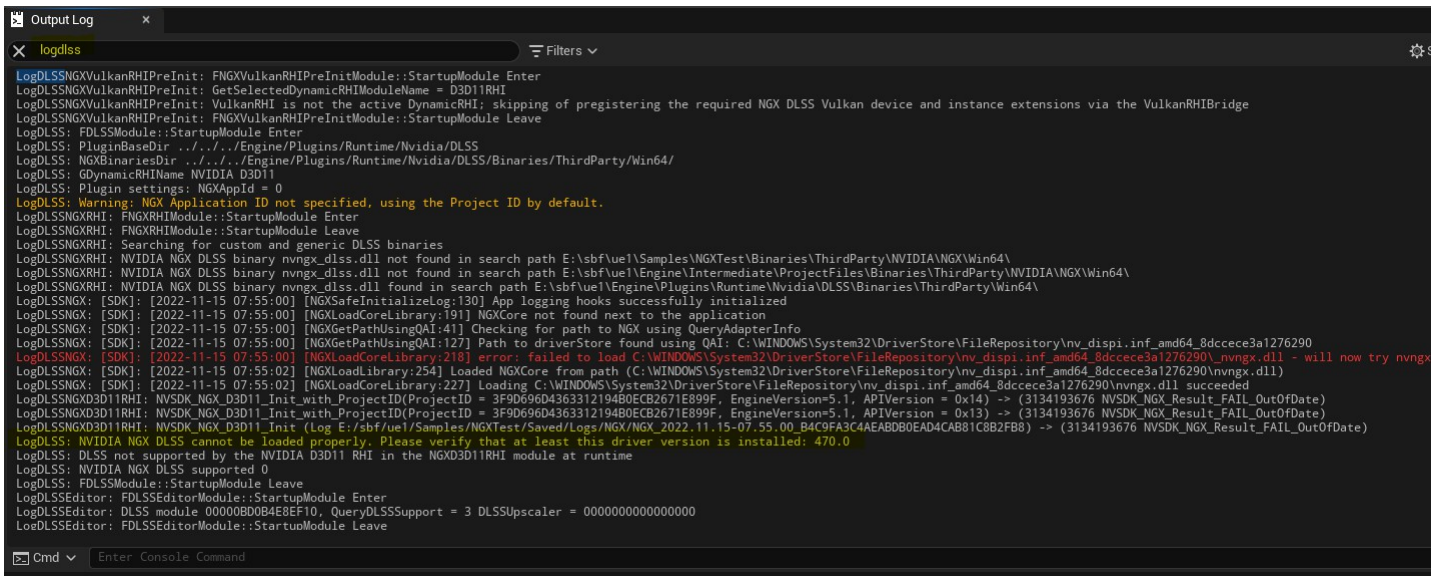
또한 DLSS 플러그인 모듈은 다양한 정보를 다음 UE 로그 카테고리에 기록합니다:
- LogDLSS
- LogDLSSEditor
- LogDLSSBlueprint
- LogDLSSNGXRHI
- LogDLSSNGXD3D11RHI
- LogDLSSNGXD3D12RHI
- LogDLSSNGXVulkanRHIPreInit
- LogDLSSNGXVulkanRHI
- LogDLSSNGX

에디터의 `창 -> 출력 로그`에서 액세스할 수 있습니다.



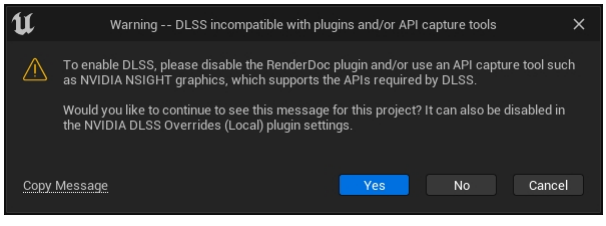
그런 다음 메시지 로그를 필터링하여 DLSS 관련 메시지만 표시하도록 하여 이 예시와 같이 DLSS가 예상대로 작동하지 않는 이유에 대한 자세한 정보를 얻을 수 있습니다.





RenderDoc과 같은 API 캡처 도구와의 비호환성

DLSS와 호환되지 않는 API 캡처 도구(예: RenderDoc) 또는 플러그인을 사용하는 경우 편집기를 시작할 때 경고가 표시됩니다. DLSS를 사용하려면 DLSS에 필요한 NGX API를 지원하는 NVIDIA NSIGHT 그래픽스와 같은 API 캡처 툴을 사용하세요.



뎁스 오브 필드와의 비호환성

DLSS 워크로드는 파이프라인에서 TAAU와 같은 위치에서 발생하므로 DoF의 시각적 차이가 예상됩니다. 이러한 차이를 최소화하려면 '품질' 또는 '울트라 품질' 모드에서 DLSS를 사용하는 것이 좋습니다. 카메라 액터에서 DoF 설정을 조정하여 차이를 보정할 수 있습니다. 이는 콘텐츠에 따라 다르다는 점에 유의하세요. 따라서 어떤 장면에서는 차이가 최소화되어 피할 수 있지만 다른 경우에는 더 어려울 수 있습니다.

엔진 측 플러그인 후크 확인

다음 값은 기본적으로 해당 값으로 설정해야 합니다:

- r.TemporalAA.Upscaler 1
- r.Reflections.Denoiser 2

최종 사용자 시스템에서 NGX DLSS 로깅 활성화하기

또한 DLSS 플러그인은 NGX DLSS 로그를 UE 로깅 시스템으로 `LogDLSSNGX` 로그 카테고리로 파이프합니다. 기본적으로 활성화되어 있으며 `r.NGX.LogLevel` 콘솔 변수를 사용하여 조정하거나 -NGXLogLevel=X 명령줄 옵션으로 설정할 수 있습니다.

이를 위해서는 NVIDIA GeForce **드라이버 버전 461.36** 이상이 필요합니다.

개발 중 NGX DLSS 로깅 활성화하기

r.NGX.EnableOtherLoggingSinks가 설정되어 있으면 자세한 내용은 DLSS 프로그래밍 가이드의 "NGX 로깅" 장에서 설명하는 대로 파일에 대한 NVIDIA NGX 소프트웨어 스택의 추가 NGX 로깅도 사용할 수 있습니다. 또한 -NGXLogFileEnable 및 -NGXLogFileDisable 명령줄 옵션을 사용하여 기본 설정을 재정의할 수 있습니다. DLSS SDK는 다음 .reg 파일로 설정할 수 있는 레지스트리 키를 제공하며, 이 레지스트리 키는 플러그인 폴더의 `\DLSS\Source\ThirdParty\NGX\Utils\`에서 찾을 수 있습니다:

- `ngx_log_on.reg`
- `ngx_log_off.reg`
- `ngx_log_verbose.reg`

DLSS 플러그인은 `$(프로젝트 디렉터리)\저장된\로그\` 아래의 하위 폴더에 다음과 같이 기록합니다.
`NGX_$(타임스탬프)_$(GUID)` 패턴

- `nvngx.log`
- `nvngx_dlss_2_1_34.log`
- `nvsdk_ngx.log`

DLSS 화면 표시기

DLSS SDK는 아래의 플러그인 폴더에서 찾을 수 있는 다음 .reg 파일로 설정할 수 있는 레지스트리 키를 제공합니다.
`\DLSS\소스\서드파티\NGX\유틸리티\`:

- `ngx_driver_onscreenindicator.reg`
- `ngx_driver_off_screenindicator.reg`

첫 번째 레지스트리 키를 설정하면 DLSS가 활성화되면 화면에 표시기가 표시되어 보다 쉽게 문제를 해결할 수 있습니다. 두 번째 레지스트리 키를 사용하여 이 표시기를 다시 비활성화할 수 있습니다.

자세한 내용은 DLSS 프로그래밍 가이드를 참조하세요.

명령줄 옵션 및 콘솔 변수 및 명령어

DLSS 활성화(엔진 측)

DLSS 플러그인은 다양한 엔진 사이드 훅을 사용하며, 다음 변수로 구성할 수 있습니다. 기본값

r.TemporalAA.Upscaler(1, 기본값)
  - DLSS 플러그인과 같은 사용자 지정 TAAU 업스케일링 플러그인 활성화

- r.Reflections.Denoiser(2, 기본값)
  - 커스텀 노이즈 제거 플러그인을 활성화합니다. DLSS 플러그인은 이를 사용하여 TAA 패스를 추가함으로써 레이트레이싱 된 리플렉션의 이미지 품질을 개선합니다.

또한 DLSS-RR을 사용할 때는 `r.Lumen.Reflections.BilateralFilter=0` 설정을 사용하여 루멘 리플렉션에 대한 내장 노이즈 제거를 비활성화하는 것이 좋습니다. 엔진 버전 5.2 및 5.3에서는 콘솔 또는 블루프린트를 통해 런타임에 이 설정을 변경하면 나중에 해상도가 변경되는 경우 엔진 런타임 어설션이 발생할 수 있습니다. 이 설정은 DefaultEngine.ini와 같은 환경설정 파일만 변경해야 합니다. 따라서 DLSS-RR이 비활성화되었을 때 r.Lumen.Reflections.BilateralFilter를 다시 활성화하려면 구성 설정을 변경하고 엔진을 재시작해야 합니다.

DLSS에 모션 벡터 활성화

DLSS가 제대로 작동하려면 올바른 모션 벡터가 필요합니다. 다음 콘솔 변수를 사용하여 동적 지오메트리가 있는 오브젝트뿐만 아니라 모든 오브젝트의 모션 벡터를 렌더링할 수 있습니다. 예를 들어 모든 메시를 고정된 상태로 변경하거나

동적.

- r.Velocity.ForceOutput(0, 기본값)
  - 베이스 패스에 모션 벡터를 계산하도록 강제 - FPrimitiveUniformShaderParameters 에 관계없이 모션 벡터를 계산합니다.
  - 0: 비활성화
  - 1: 사용

DLSS-SR/DLAA 활성화(플러그인 측)

- r.NGX.Enable(1, 기본값)은 명령줄에서 **-ngxenable** 및 **-ngxdisable을** 사용하여 재정의할 수도 있습니다.
  - NGX 라이브러리를 로드할지 여부입니다. 이 옵션을 선택하면 DLSS 플러그인을 활성화할 수 있지만 DLSS의 드라이버 측 NGX 부분을 건너뛰어 잠재적인 비호환성을 피할 수 있습니다.
- r.NGX.DLSS.Enable(1, 기본값)
  - DLSS-SR/DLAA를 활성화/비활성화합니다.

UI에 DLSS를 표시하는 방법에는 모든 DLSS 모드를 명시적으로 나열하거나 DLSS 끄기/자동 모드를 제공하는 두 가지 대안이 있습니다. 명시적인 DLSS 모드의 경우 UI에 지원되는 모든 DLSS 모드를 나열합니다. `GetDLSSModeInformation`을 사용하여 선택한 DLSS 모드에 대한 최적의 화면 비율을 찾고, 해당 값을 사용하여 `r.ScreenPercentage`를 설정한 다음 `EnableDLSS`를 호출합니다. DLSS 자동 모드의 경우 `GetDLSSModeInformation`을 사용하여 DLSS 자동 모드에 대한 최적의 화면 비율을 찾은 다음 화면 비율이 0이 아닌 경우 해당 값을 사용하여 `r.ScreenPercentage`를 설정하고 `EnableDLSS`를 호출합니다.

플러그인에는 DLSS 및 DLAA를 활성화하기 위한 두 개의 블루프린트 매크로가 DLSSMacros 에셋에 포함되어 있는데, 하나는 명시적 DLSS 모드용이고 다른 하나는 DLSS 자동 모드용입니다. 매크로를 직접 사용하거나 프로젝트에 복사하여 작업의 시작점으로 사용할 수 있습니다. 매크로를 표시하려면 콘텐츠 브라우저 설정에서 '엔진 콘텐츠 표시' 및 '플러그인 콘텐츠 표시'를 활성화해야 할 수 있습니다.

블루프린트 함수:

- DLSS-SR 사용, DLSS-SR 사용됨
- DLSS-SR 지원 여부, DLSS-SR 지원 쿼리, DLSS-SR 최소 드라이버 버전 가져오기, 기본 DLSS 모드 가져오기
- DLSS-SR 모드 지원 여부, 지원되는 DLSS-SR 모드 가져오기, DLSS-SR 모드 정보 가져오기, DLSS-SR 화면 백분율 범위 가져오기

DLSS-RR 활성화(플러그인 측)

- r.NGX.DLSS.DenoiserMode (0, 기본값)
  - DLSS 노이즈 제거 방법을 구성합니다.
  - 0: 꺼짐, 노이즈 제거 없음(기본값)
  - 1: DLSS-RR 활성화됨
- r.NGX.DLSS.BuiltInDenoiserOverride (-1, 기본값)
  - 내장된 노이즈 제거기 변경하기
  - -1: 자동, r.NGX.DLSS.DenoiserMode(기본값)에 따라 다름
  - 0: 내장된 모든 노이즈 제거 건너뛰기
  - 1: 내장된 노이즈 제거 기능 사용

기본적으로 DLSS-RR이 활성화되면 모든 기본 제공 노이즈 제거 기능을 무시합니다.

블루프린트 함수:

- DLSS-RR 사용, DLSS-RR 사용됨
- DLSS-RR 지원 여부, DLSS-RR 지원 조회, DLSS-RR 최소 드라이버 버전 받기

DLSS-RR 지원은 향후 플러그인 릴리스에서 제공될 예정입니다. 이번 릴리스에서는 DLSS-RR이 지원되지 않는 것으로 보고됩니다.

DLSS 런타임 이미지 품질 조정

- r.NGX.DLSS.DilateMotionVectors(1, 기본값)
  - 0: 저해상도 모션 벡터를 DLSS로 전달합니다.
  - 1: 확장된 고해상도 모션 벡터를 DLSS로 전달합니다. 이렇게 하면 얇은 디테일의 이미지 품질을 개선하는 데 도움이 됩니다.◦ 참고: 모션 벡터 확대는 DLSS-RR에서 지원되지 않으며, DLSS-RR이 활성화된 경우 cvar는 영향을 미치지 않습니다.

- r.NGX.DLSS.Reflections.TemporalAA(1, 기본값)
  - 노이즈 제거된 리플렉션에 템포럴 AA 패스를 적용●

r.NGX.DLSS.WaterReflections.TemporalAA (1, 기본값)

  - 노이즈 제거된 물 반사에 템포럴 AA 패스를 적용●

r.NGX.DLSS.EnableAutoExposure

- 0: 입력 이미지에 엔진에서 계산한 노출 값을 DLSS에 사용합니다.
- 1: 애플리케이션에서 제공하는 자동 노출 대신 DLSS 내부 자동 노출을 활성화하면 어두운 장면에서 고스트 현상과 같은 효과를 완화할 수 있습니다(기본값).

- r.NGX.DLSS.PreferNISSharpen(2, 기본값)
  - 프로젝트에 NIS 플러그인도 활성화된 경우 DLSS 샤프닝 대신 추가 NIS 플러그인 샤프닝 패스를 사용하여 샤프닝하는 것을 선호합니다.
  - NIS 플러그인을 사용하도록 설정해야 하며, 그렇지 않으면 DLSS 선명화가 사용됩니다.
    - 0: DLSS 패스로 부드럽게/선명하게 처리합니다.
    - 1: NIS 플러그인으로 선명하게 합니다. 소프트닝은 지원되지 않습니다. NIS 플러그인을 활성화해야 합니다.
    - 2: NIS 플러그인으로 선명하게 하기. DLSS 플러그인으로 부드럽게(즉, 마이너스 선명도) 합니다. NIS 플러그인을 활성화해야 합니다.
  - **참고** 이 변수는 C++ 또는 블루프린트 이벤트 그래프에서 더이상 사용되지 않는 `SetDLSSSharpness` 블루프린트 함수를 사용할 때만 평가됩니다!
  - **참고** DLSS 선명하게 하기는 더 이상 사용되지 않으며, 향후 플러그인 버전에서는 DLSS 선명하게 하기가 제거됩니다. 대신 NIS 플러그인을 사용하여 선명하게 하세요.

DLSS 바이너리

- r.NGX.BinarySearchOrder(0, 기본값)
  - 0: 자동
    - 프로젝트 및 실행 폴더(*ProjectDir*)*/Binaries/ThirdParty/NVIDIA/NGX/*(플랫폼)의 커스텀 바이너리가 있는 경우▪ 플러그인 폴더의 일반 바이너리로 폴백합니다.
  - 1: 플러그인 폴더에서 일반 바이너리를 강제로 가져오고, 찾지 못하면 실패합니다.
  - 2: 프로젝트 또는 실행 폴더에서 사용자 지정 바이너리 강제 실행, 찾지 못하면 실패
  - 3: 플러그인 폴더에서 일반 개발 바이너리를 강제로 찾고, 찾지 못하면 실패합니다. 이는 비출시 빌드 구성에서만 지원됩니다.

DLSS 메모리 사용량

- 통계 `DLSS`
  - 는 DLSS가 사용하는 GPU 메모리 양과 DLSS 기능, 즉 DLSS의 인스턴스가 할당된 수를 보여줍니다.
  - 정상 상태에서는 뷰당 1개의 DLSS 기능이 할당되어야 합니다. 이 값은 일반적으로 DLSS 품질 모드를 변경하거나 창 크기를 조정한 후 일시적으로 증가할 수 있습니다. 이 값은 `r.NGX.FramesUntilFeatureDestruction` 콘솔 변수를 사용하여 구성할 수 있습니다.

DLSS 사전 설정

DLSS 플러그인은 DLSS-SR 업스케일링의 다양한 측면을 조정할 수 있는 렌더링 프리셋을 제공합니다. 일반적으로 기본 사전 설정에서 변경할 필요가 없습니다. DLSS 사전 설정의 이름은 A - G이며 DLSS 프로그래밍 가이드에 자세히 설명되어 있습니다.

특정 프리셋은 DLSS 플러그인 설정(편집 -> 프로젝트 설정 -> NVIDIA DLSS -> 일반 설정 -> 고급)에서 DLAA 및 각 DLSS 품질 모드에 대해 개별적으로 강제 적용할 수 있습니다. 또한 cvar `r.NGX.DLSS.Preset`을 0에서 7 사이의 값(0=프로젝트 설정, 1=A, 2=B, 3=C, 4=D, 5=E, 6=F, 7=G)으로 설정하여 DLSS 프리셋을 전역적으로 오버라이드할 수 있습니다.

DLSS 프리셋은 새 버전의 DLSS 및 무선(OTA) 업데이트를 통해 변경될 수 있습니다. OTA 업데이트를 선택 해제하려면 DLSS 플러그인 설정(편집 -> 프로젝트 설정 -> NVIDIA DLSS -> 일반 설정)에서 "OTA 업데이트 허용" 옵션을 선택 해제하세요. 명령줄에서 **-ngxdisableota** 및 **-ngxenableota**를 사용하여 OTA 업데이트를 재정의할 수도 있습니다. OTA 업데이트와 DLSS 프리셋의 상호 작용은 DLSS 프로그래밍 가이드에 자세히 설명되어 있습니다.

NGX 프로젝트 ID

DLSS 플러그인은 기본적으로 프로젝트 식별자를 사용하여 NGX 및 DLSS를 초기화합니다. 드물게 NVIDIA에서 특별한 NVIDIA NGX 애플리케이션 ID를 제공할 수 있습니다. 다음 콘솔 변수에 따라 어떤 것이 사용되는지 결정됩니다.

r.NGX.ProjectIdentifier(0, 기본값)

- 0: 자동:
  - 0이 아닌 경우 NVIDIA NGX 애플리케이션 ID를 사용하고, 그렇지 않으면

UE 프로젝트 ID를 사용합니다.- 1: UE 프로젝트 ID 강제 적용

- 2: NVIDIA NGX 애플리케이션 ID 강제 적용(프로젝트 설정 -> NVIDIA DLSS 플러그인을 통

해 설정) 자세한 내용은 "DLSS 배포하기" 섹션을 참조하세요.

멀티 GPU 지원(실험적)

DLSS 플러그인은 다음 표와 같이 특정 상황에서 여러 개의 GPU를 지원합니다. 여기서 AFR은 대체 프레임 렌더링, 즉 SLI 또는 크로스파이어를 의미하고 SFR은 분할 프레임 렌더링을 의미합니다.

RHI	AFR	SFR

RHI AFRSFR

D3D12RH 또는

D3D11RH 또는

벌칸RHI

참고:

- D3D12RHI
  - AFR은 지원되지 않습니다.
    - 주로 상위 레벨 렌더러 코드가 동일한 GPU에서 연속되지 않은 프레임에 걸쳐 TAA(따라서 DLSS) 기록을 유지하지 않기 때문입니다.
  - SFR은 아직 5.1 엔진에서 지원되지 않

습니다- D3D11RHI

  - AFR은 드라이버 기반의 자동 SLI 지원을 통해 지원됩니

다- VulkanRHI

  - (UE 4.27 기준) VulkanRHI 는 명시적 MGPU 를 구현하지 않으므로 AFR 이나 SFR 을 사용할 수 없습니다. 다음

콘솔 변수를 사용하여 DLSS 가 GPU 노드와 상호작용하는 방식을 조정할 수 있습니다.

- r.NGX.DLSS.FeatureCreationNode(-1, 기본값)
  - DLSS 기능을- -1에서 생성할 GPU를 결정합니다: - 0: 명령 목록이 실행되는 GPU에 생성 0: GPU 노드 0에 생성
  - 1: GPU 노드 1에서 생성
- r.NGX.DLSS.FeatureVisibilityMask(-1, 기본값)
  - DLSS 기능을 표시할 GPU를 결정합니다.
  - -1: - 1: GPU 노드 0에 명령 목록이 실행 중입니다.
  - 2: GPU 노드 1에 표시
  - 3: GPU 노드 0과 GPU 노드 1에 표시됨

기타

- r.NGX.DLSS.AutomationTesting(0, 기본값)
  - GIsAutomationTesting이 true일 때 NGX 라이브러리를 로드할지 여부입니다(기본값은 false).
  - 시작하기 전에 true로 설정해야 합니다. DLSS로 자동화 테스트를 실행하려는 경우 이 옵션을 활성화할 수 있습니다.
- r.NGX.Automation.Enable (0, 기본값)
  - NGX DLSS 이미지 품질 및 성능 평가를 위한 자동화를 사용하도록 설정합

니다- r.NGX.Automation.ViewIndex (0, 기본값)

  - NGX DLSS 이미지 품질 및 성능 자동화와 함께 사용할 뷰를 선택합니다.-

r.NGX.Automation.NonGameViews (0,기본값)

  - NGX DLSS 이미지 품질 및 성능 자동화를 위해 비게임 뷰를 활성화합니다.-

r.NGX.FramesUntilFeatureDestruction (3, 기본값)
  - 사용하지 않는 NGX 기능이 파괴될 때까지의 프레임 수
- r.NGX.DLSS.MinimumWindowsBuildVersion(16299, v1709의 기본값)

- DLSS를 활성화하는 데 필요한 최소 Windows 10 빌드 버전을 설정합

니다● r.NGX.LogLevel(1, 기본값).

- NGX 구현의 최소 로깅 양을 결정합니다. 로깅 수준을 변경하는 다른 방법은 DLSS 플러그인 설명서를 참조하세요.
- 0: 꺼짐
- 1: 켜짐
- 2: 자세한 설명

● r.NGX.EnableOtherLoggingSinks(0, 기본값)

- NGX 구현이 추가 로그 싱크를 켤지 여부를 결정합니다. LogDLSSNGXRHI○ 0: 꺼짐
- 1: 켜짐

● r.NGX.RenameNGXLogSeverities(1, 기본값)

- NGX 로그 콜백이 반환한 메시지의 '오류' 및 '경고'를 UE 로그 시스템으로 전달하기 전에 'e_rror' 및 'w_arning'으로 이름을 바꿉니다.
- 0: 꺼짐
- 1: 켜짐, 초기화 중 일부 메시지의 경우○ 2:

켜짐, 모든 메시지의 경우

● r.NGX.DLSS.ReleaseMemoryOnDelete(1, 기본값)

◦ DLSS 기능이 릴리스될 때 NGX 측에서 DLSS 관련 메모리 릴리스 활성화/비활성화

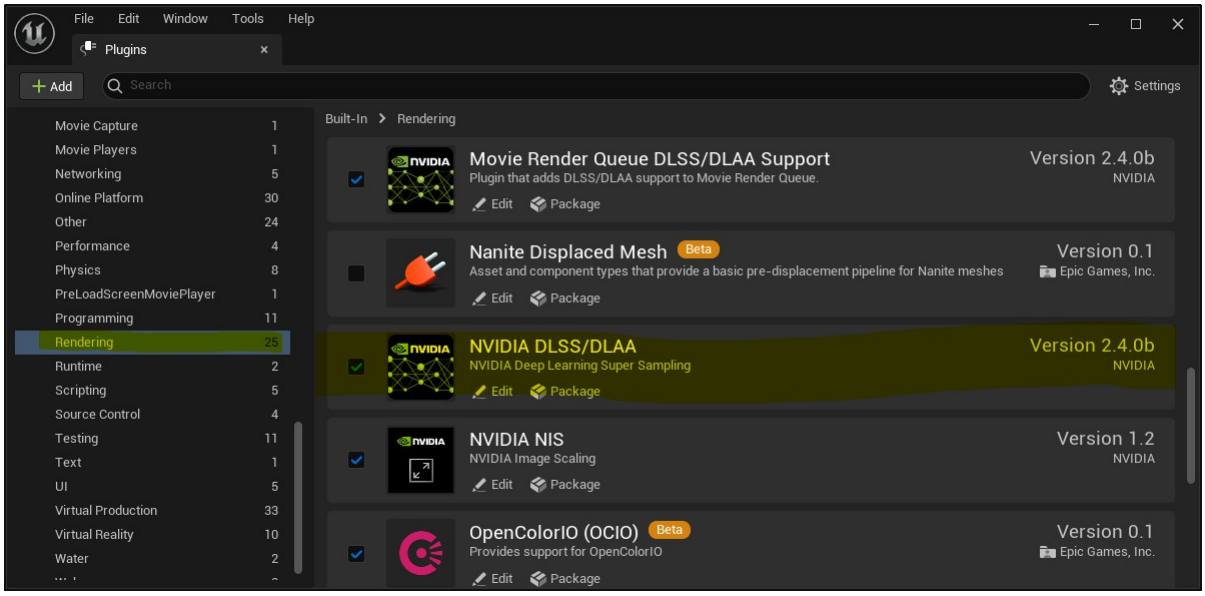
DLSS 및 NIS NVIDIA 이미지 스케일링 플러그인

동일한 프로젝트에 대해 *DLSS* 플러그인과 *NIS(*NVIDIA 이미지 스케일링) 플러그인을 함께 활성화할 수 있습니다. 권장 UI 구현은 RTX UI 개발자 가이드라인 문서를 참조하세요.

더 이상 사용되지 않는 DLSS 선명하게 하기 기능 대신 선명하게 하기용 NIS 플러그인을 사용하는 것이 좋습니다.

에디터의 DLSS/DLAA

프로젝트에 DLSS/DLAA 사용 설정하기



레벨 에디터 뷰포트에서 DLSS/DLAA 활성화하기

먼저 프로젝트 플러그인 설정 프로젝트 설정 -> 플러그인 -> NVIDIA DLSS 또는 로컬 오버라이드 프로젝트 설정 -> 플러그인 -> NVIDIA DLSS(로컬)에서 "에디터 뷰포트에서 DLSS/DLAA 켜기" 체크박스를 활성화합니다. 에디터 뷰포트 창에서 DLSS/DLAA는 기본적으로 꺼져 있습니다.

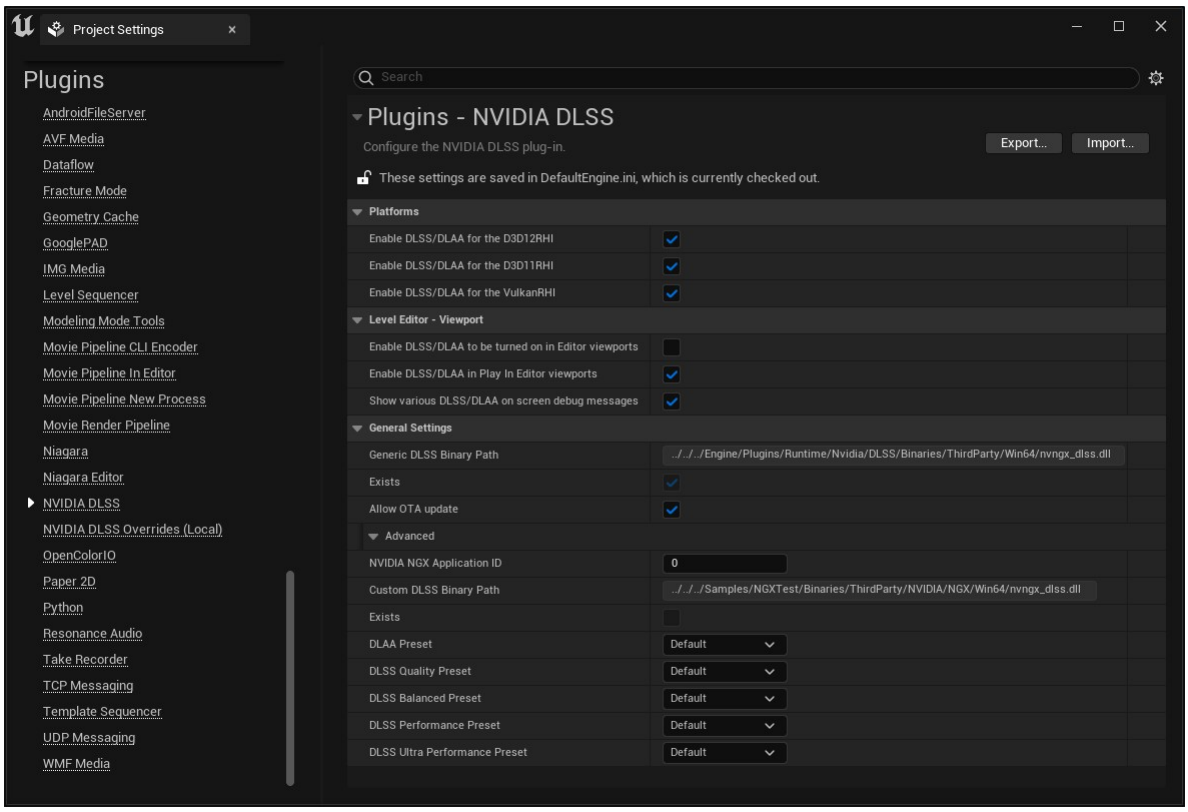
DLSS/DLAA가 활성화된 상태에서 편집기 뷰포트 창에서 화면 비율을 변경하여 업스케일링 양을 제어합니다. 스크린 퍼센티지가 100이면 DLAA가 적용됩니다. 화면 비율을 50%에서 99% 사이로 슬라이드하면 DLSS 업스케일링이 활성화됩니다.

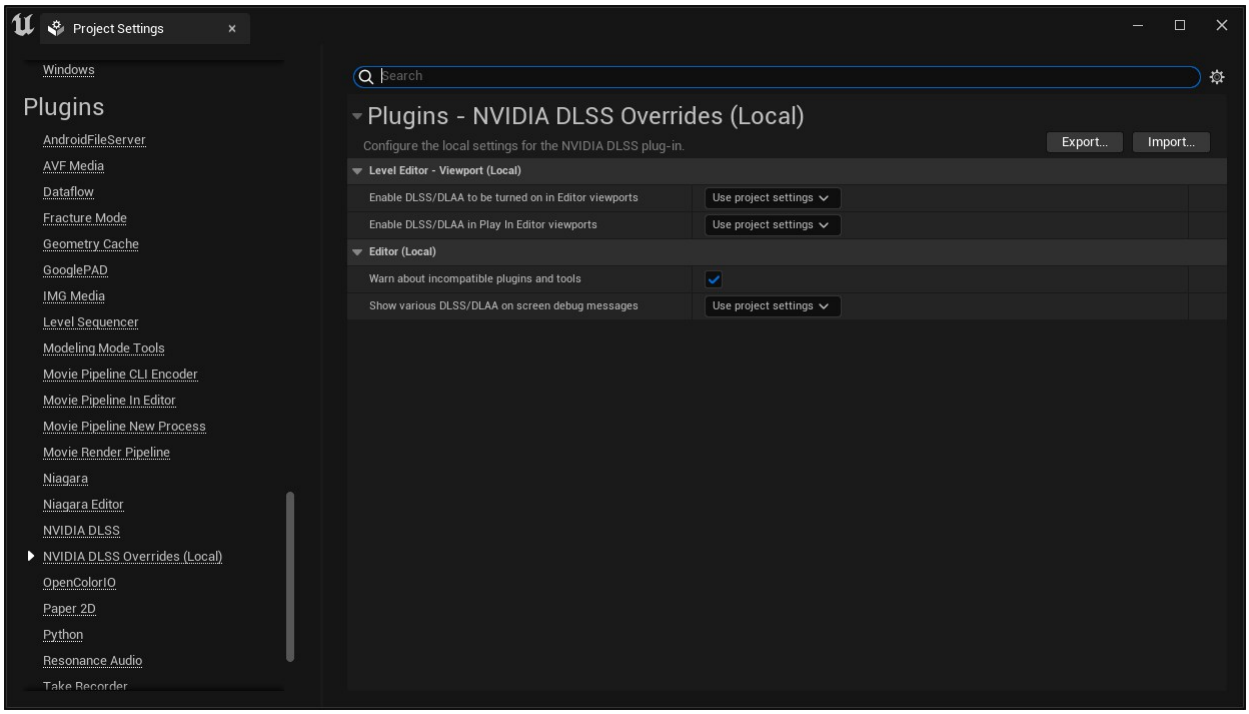
DLSS 플러그인 설정

"레벨 에디터 - 뷰포트" 세팅 중 일부는 두 개의 구성 파일과 세팅 페이지로 나뉘어 DLSS가 에디터 사용자 경험과 상호작용하는 방식에 맞게 조정할 수 있습니다.

예를 들어 크로스 플랫폼 게임 프로젝트의 경우 지원되는 다양한 플랫폼에서 일관된 콘텐츠 제작 환경을 유지하기 위해 기본적으로 '에디터 뷰포트에서 플레이' 또는 '게임 모드'에서만 DLSS를 활성화하는 것이 더 실용적일 수 있습니다. 그러나 레이트레이싱 워크로드가 많은 아키텍처 시각화 프로젝트와 같은 프로젝트에서는 콘텐츠 제작 중에 DLSS를 활성화하는 것이 더 유용할 수 있습니다. 어느 쪽이든 각 사용자는 로컬에서 해당 설정을 재정의할 수 있습니다:

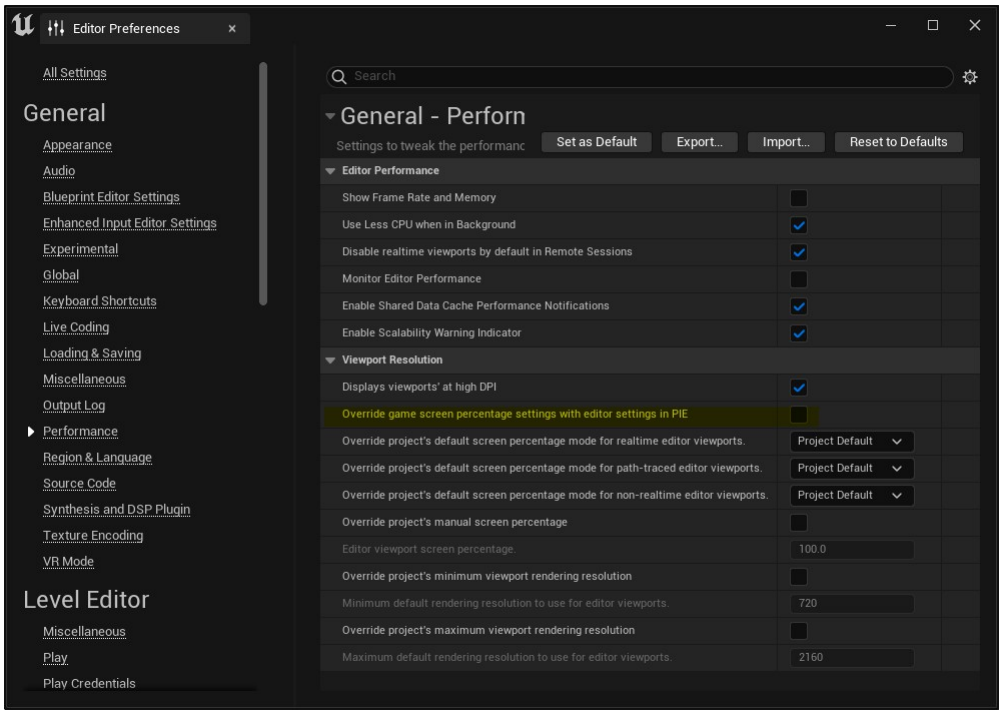
- 프로젝트 설정 -> 플러그인 -> NVIDIA DLSS
  - DefaultEngine.ini에 저장된
  - 는 일반적으로 소스 제어에 있습니다.
  - 설정은 사용자 간에 공유됩니다.
- 프로젝트 설정 -> 플러그인 -> NVIDIA DLSS(로컬)
  - 저장된 UserEngine.ini
  - 소스 제어에 체크인하지 않는 것이 좋습니다.
  - 사용자가 원하는 경우 프로젝트 전체 설정을 재정의할 수 있도록 허용합니다. 기본값은 "프로젝트 설정 사용"입니다.





편집기 설정

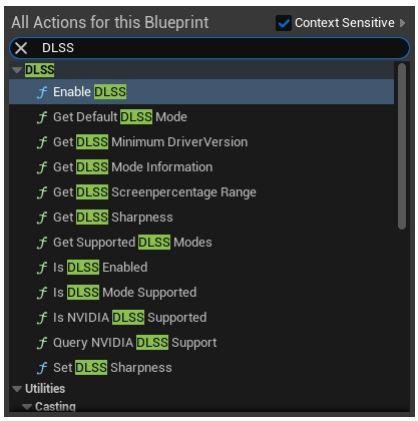
기본적으로 편집기는 PIE에서 실행할 때 화면 비율을 재정의합니다. 이로 인해 특정 DLSS 품질 수준을 설정하기 위해 화면 비율을 변경하는 애플리케이션 로직을 방해할 수 있습니다. 애플리케이션이 화면 비율을 설정할 수 있도록 하려면 편집 -> 에디터 환경 설정 -> 성능으로 이동하여 "PIE에서 에디터 설정으로 게임 화면 비율 설정 재정의"를 끕니다. 이 설정을 변경한 후 에디터를 재시작하세요.



DLSS 블루프린트

UDLSSLibrary 블루프린트 라이브러리는 DLSS 지원 여부와 지원되는 모드를 쿼리하는 기능을 제공합니다. 또한 기본 DLSS 콘솔 변수를 활성화하는 편리한 함수도 제공합니다. 각 함수의 툴팁은 추가 정보를 제공합니다.

콘솔 변수를 직접 설정하는 것보다 블루프린트 또는 C++(게임 프로젝트에 DLSSBlueprint 모듈을 포함)를 통해 UDLSSLibrary를 사용하는 것이 좋습니다. 이렇게 하면 게임 로직을 업데이트할 필요 없이 DLSS 플러그인만 업데이트하면 향후 업데이트가 적용될 수 있습니다.

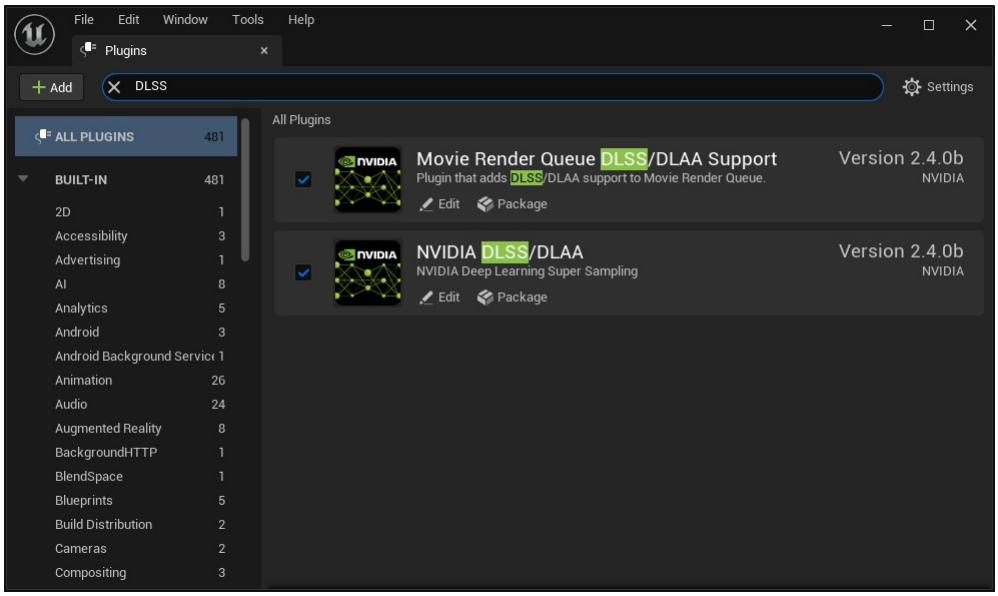


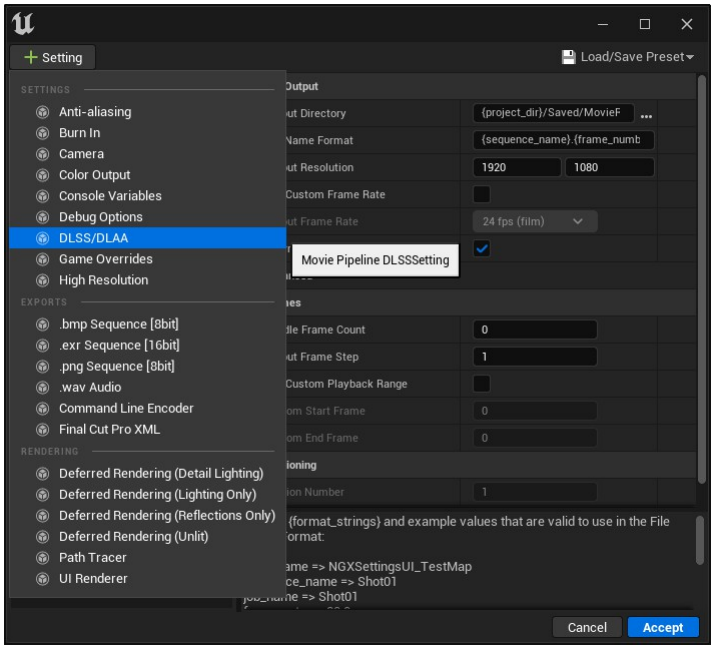
DLSS 무비 렌더 큐 지원

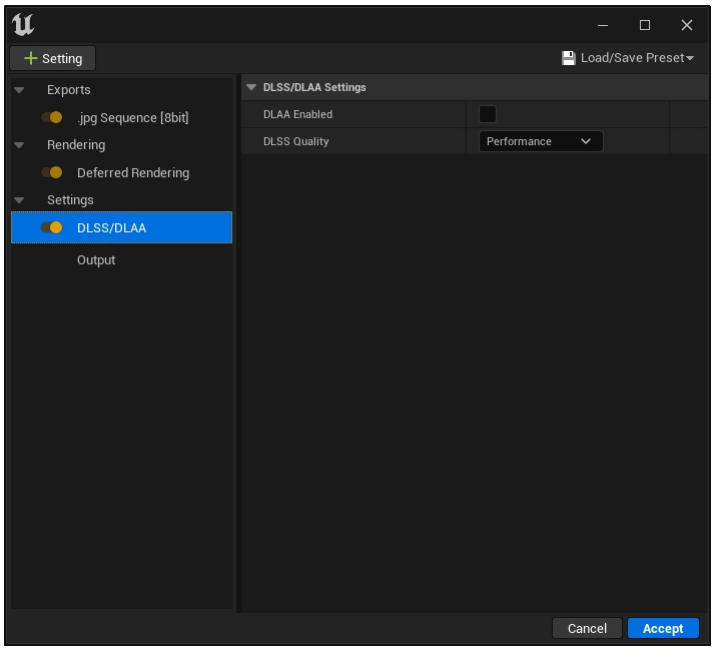
무비 렌더 대기열 플러그인으로 무비를 렌더링할 때 DLSS가 지원됩니다.

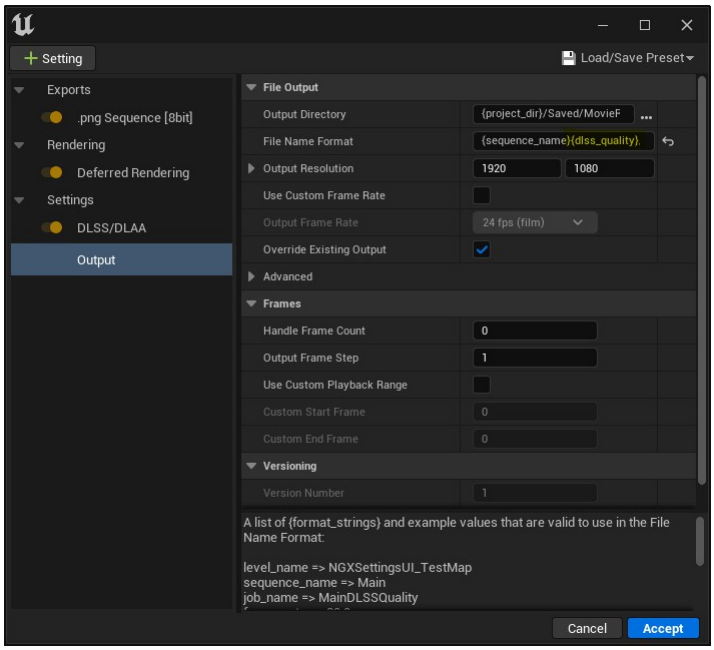
0. 에디터에서 무비 *렌더 대기열* 및 *DLSS* 플러그인을 활성화합니다.
1. 에디터에서 *무비 렌더 큐 DLSS 지원* 플러그인을 활성화한 다음 에디터를 재시작합니다.
2. 구성에서 설정 `->` `DLSS` 페이지를 추가합니다.
3. DLSS 설정 페이지에서 원하는 DLSS 품질 모드를 변경합니다.
    1. 참고: 지원되지 않는 DLSS 모드에서는 창 하단에 경고가 표시됩니다.
4. 선택 사항입니다: 설정 `->` 출력 `->` 파일 이름 형식 페이지에서 `{dlss_quality}` 형식 태그를 지원합니다.

참고: DLSS에서는 *디퍼드 렌더링* 렌더링 패스만 지원되며, 다른 모든 패스는 기본 제공 TAA를 사용합니다.


File
Edit
Window
Tools
Help
Plugins
Add
DLSS
Settings
ALL PLUGINS 481
BUILT-IN 481
2D 1
Accessibility 3
Advertising 1
AI 8
Analytics 5
Android 3
Android Background Servic 1
Animation 26
Audio 24
Augmented Reality 8
BackgroundHTTP 1
BlendSpace 1
Blueprints 5
Build Distribution 2
Cameras 2
Compositing 3
All Plugins
Movie Render Queue DLSS/DLAA Support
Plugin that adds DLSS/DLAA support to Movie Render Queue.
Edit
Package
Version 2.4.0b
NVIDIA
NVIDIA DLSS/DLAA
NVIDIA Deep Learning Super Sampling
Edit
Package
Version 2.4.0b
NVIDIA


Setting
Load/Save Preset
SETTINGS
Anti-aliasing
Burn In
Camera
Color Output
Console Variables
Debug Options
DLSS/DLAA
Game Overrides
High Resolution
EXPORTS
.bmp Sequence [8bit]
.exr Sequence [16bit]
.png Sequence [8bit]
.wav Audio
Command Line Encoder
Final Cut Pro XML
RENDERING
Deferred Rendering (Detail Lighting)
Deferred Rendering (Lighting Only)
Deferred Rendering (Reflections Only)
Deferred Rendering (Unlit)
Path Tracer
UI Renderer
Movie Pipeline DLSSSetting
Cancel
Accept


Setting
Load/Save Preset
Exports
.jpg Sequence [8bit]
Rendering
Deferred Rendering
Settings
DLSS/DLAA
Output
DLSS/DLAA Settings
DLAA Enabled
DLSS Quality
Performance
Cancel
Accept



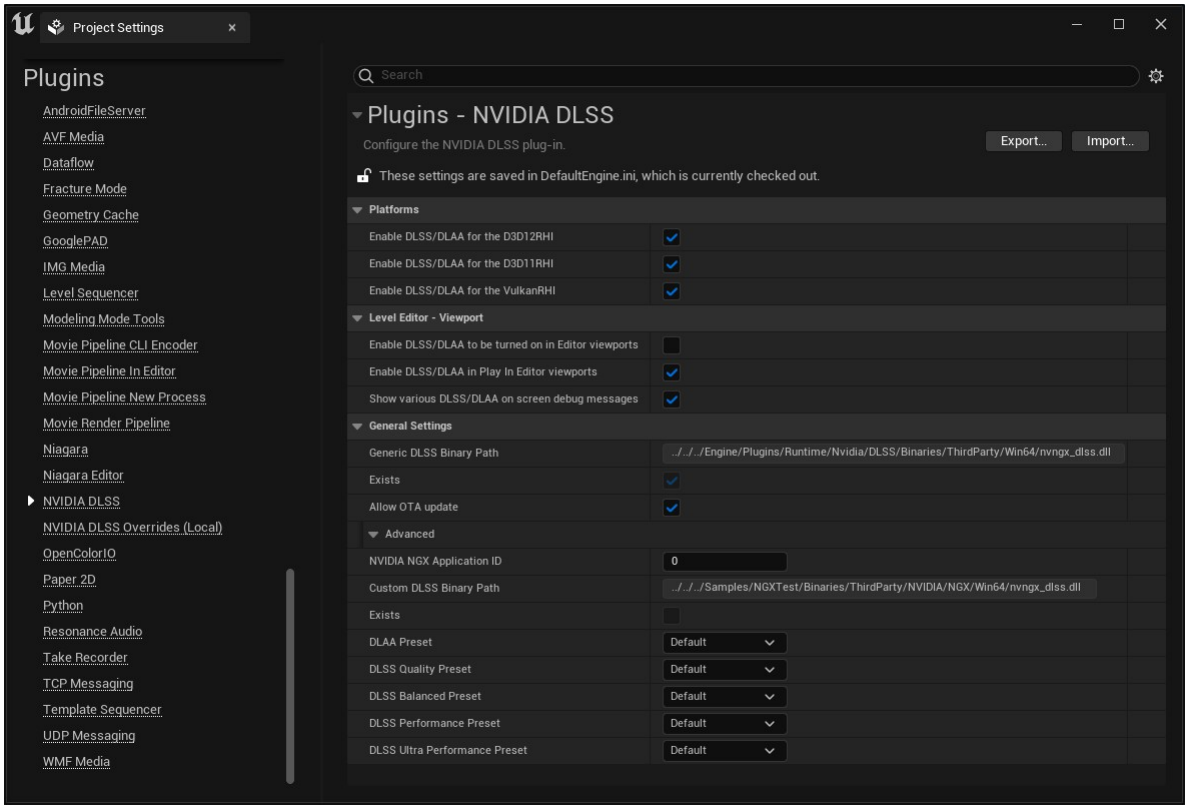
DLSS 배포

DLSS 플러그인은 바로 사용할 수 있는 프로덕션 DLSS 바이너리(워터마크 없음)와 함께 제공되며, 프로젝트 식별자를 사용하여 NGX 및 DLSS를 초기화합니다. 이는 최종 사용자에게 배포하는 일반적인 경우이며 귀사 또는 NVIDIA 측에서 추가 조치가 필요하지 않습니다. 하지만 드물게 NVIDIA에서 제공할 수 있습니다:

1. 사용자 지정 프로젝트별 DLSS 바이너리
2. NVIDIA 애플리케이션 ID

이 경우 고급 플러그인 설정에서 구성할 수 있습니다. 또한 r.NGX.ProjectIdentifier 콘솔 변수가 0(기본값) 또는 2로 설정되어 있는지 확인하시기 바랍니다. 프로젝트 플러그인 설정을 사용하여 이를 구성할 수 있습니다(위 참조).

1. 프로젝트별 커스텀 DLSS 바이너리 nvngx_dlss.dll을 프로젝트 아래에 넣어야 합니다.
   `(프로젝트 디렉터리)/바이너리/서드파티/NVIDIA/NGX/$(플랫폼)`
2. 프로젝트에 대한 NVIDIA NGX 애플리케이션 ID를 설정합니다.



자세한 내용은 DLSS 프로그래밍 가이드의 '4장 게임 내 DLSS 배포하기'를 참조하세요.

이전 엔진용 DLSS 플러그인 버전에서 업그레이드하기

일반적으로 5.1 이전 엔진 버전에서 업그레이드할 때는 DLSS 온스크린 표시기를 사용하여 애플리케이션에서 DLSS 및 DLAA가 제대로 활성화되어 있는지 확인하는 것이 좋습니다.

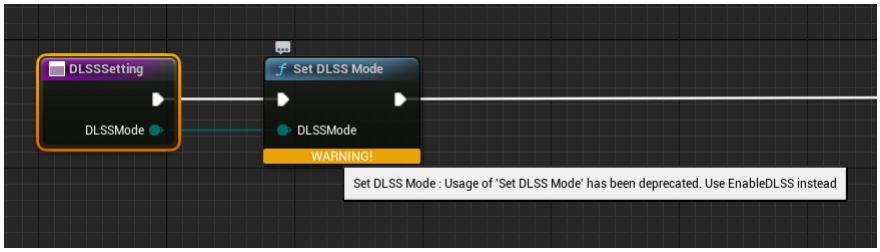
화면 비율

언리얼 엔진 5.1 이전에는 DLSS 플러그인이 직접 업스케일링 스크린 퍼센티지를 제어하여 `r.ScreenPercentage` 변수나 다른 메서드를 통해 설정된 모든 것을 덮어썼습니다. 블루프린트나 코드에서 애플리케이션이 플러그인에 특정 DLSS 퀄리티 모드를 요청하면 플러그인은 내부적으로 퀄리티 모드와 적절한 스크린 퍼센티지를 설정했습니다.

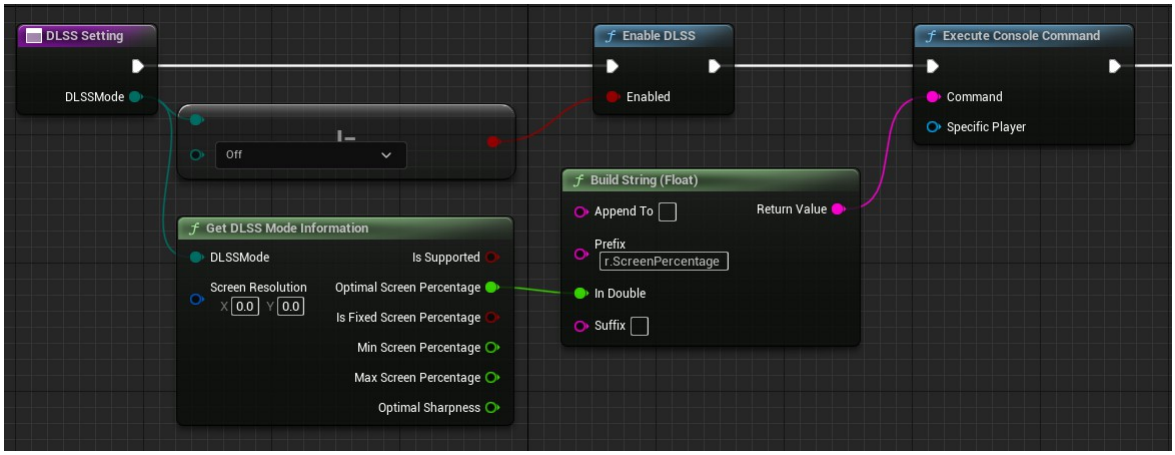
언리얼 엔진 5.1부터 플러그인은 업스케일링 스크린 퍼센티지를 오버라이드하지 않습니다. 따라서 이제 특정 DLSS 퀄리티 모드를 얻으려면 애플리케이션에서 원하는 DLSS 퀄리티 모드의 최적값에 해당하는 스크린 퍼센티지를 설정해야 합니다. DLSS가 활성화된 경우 플러그인은 내부적으로 현재 화면 비율에 가장 적합한 품질 모드를 설정합니다. 블루프린트 또는 C++에서 GetDLSSModeInformation을 사용하여 주어진 DLSS 품질 모드에 대한 최적의 스크린 퍼센티지를 찾을 수 있습니다. DLAA의 경우 스크린 퍼센티지를 100%로 설정합니다.

이전 SetDLSSMode 및 EnableDLAA 블루프린트 함수는 더 이상 사용되지 않으며 EnableDLSS로 대체해야 하며, 블루프린트 또는 코드에서 `r.ScreenPercentage` 콘솔 명령을 실행하여 적절한 스크린 퍼센티지를 설정할 수 있습니다.

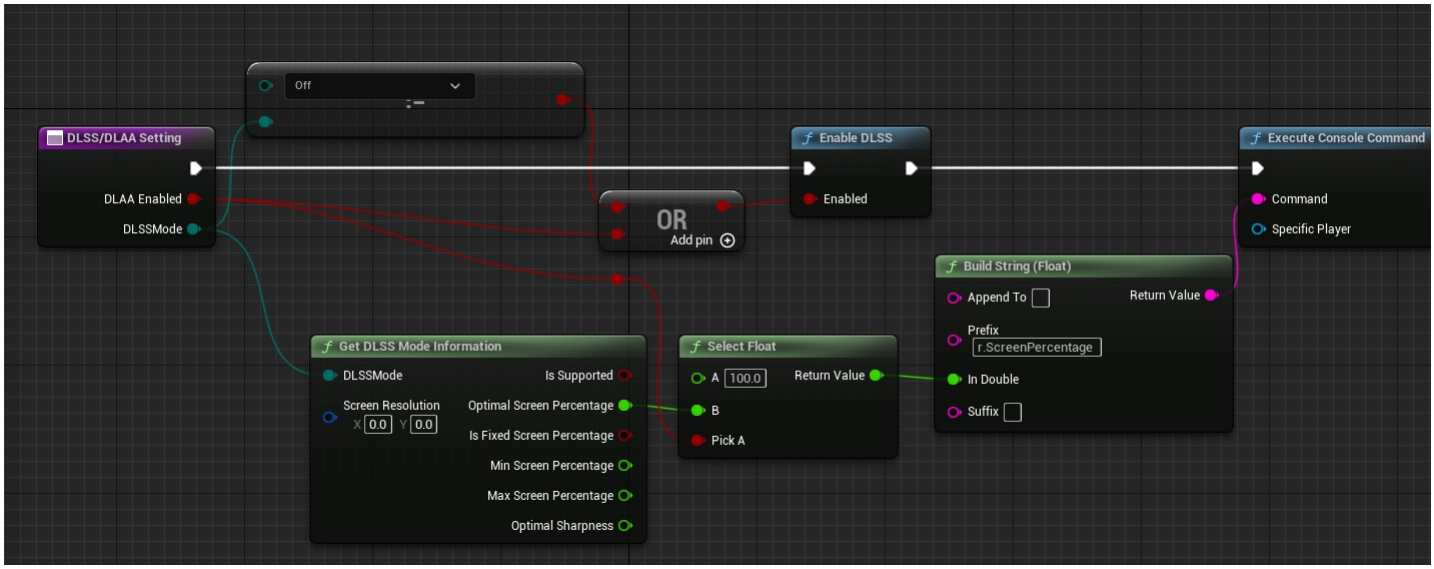
업데이트하기 전에 이전 블루프린트 기능에 대한 사용 중단 경고가 표시됩니다.



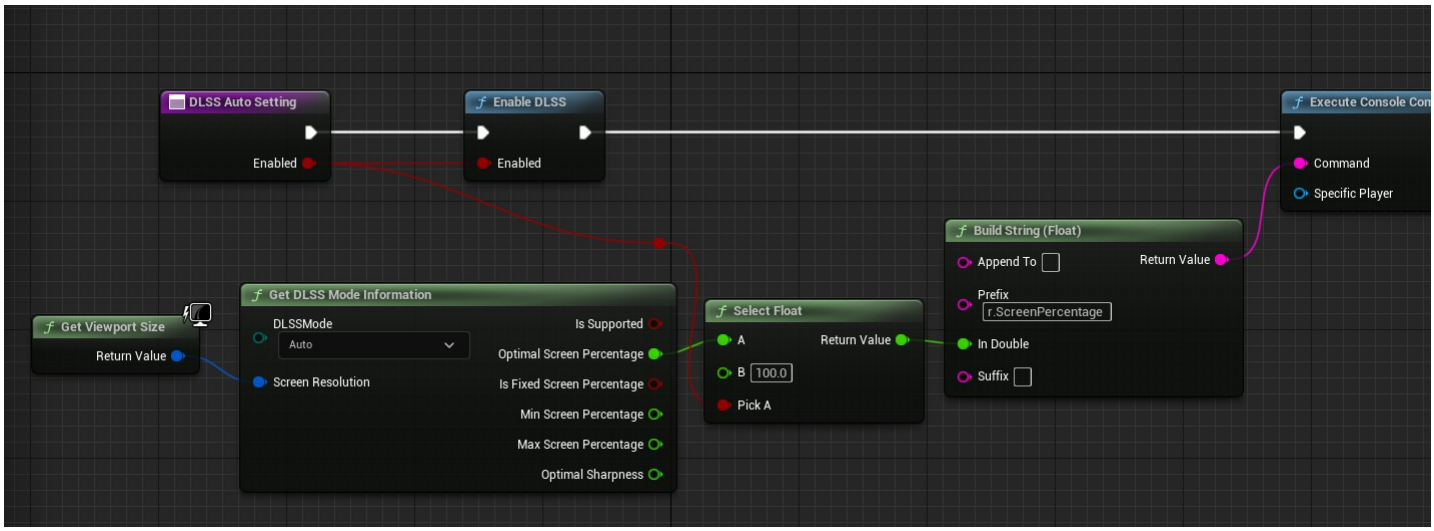
이전과 동일한 DLSS 품질 모드 옵션을 제공하려면 품질 모드에 대한 최적의 업스케일링 화면 비율을 확인하고 "r.ScreenPercentage" 콘솔 명령을 사용하여 설정하세요.



화면 비율을 100으로 설정하면 DLAA가 활성화됩니다.


Off
DLSS/DLAA Setting
DLAA Enabled
DLSSMode
OR
Add pin
Enable DLSS
Enabled
Execute Console Command
Command
Specific Player
Build String (Float)
Append To
Return Value
Prefix
r.ScreenPercentage
In Double
Suffix
Get DLSS Mode Information
DLSSMode
Is Supported
Screen Resolution
X 0.0 Y 0.0
Optimal Screen Percentage
Is Fixed Screen Percentage
Min Screen Percentage
Max Screen Percentage
Optimal Sharpness
Select Float
A 100.0
Return Value
B
Pick A

애플리케이션에서 개별 DLSS 품질 모드를 선택하는 대신 DLSS 끄기/자동 옵션을 제공하려는 경우 화면 해상도와 함께 가짜 "자동" 품질 모드를 GetDLSSModeInformation에 제공하면 권장 화면 비율이 제공됩니다.



언리얼 엔진 5.1 이상으로 업그레이드하기 전에 애플리케이션에서 `r.ScreenPercentage`를 변경한 경우, 업그레이드 후 DLSS/DLAA가 다르게 작동하는 것을 확인할 수 있습니다. 이는 엔진 버전 5.1 이전에는 DLSS 또는 DLAA가 활성화된 경우 DLSS 플러그인이 화면 퍼센티지 값을 오버라이드할 수 있었기 때문입니다. DLSS 온스크린 표시기를 사용할 때 DLSS 모드를 변경할 때 업스케일 해상도가 변경되지 않는다면 `r.ScreenPercentage`를 직접 설정하는 블루프린트나 코드가 있는지 확인하세요.

DLSS 선명도 향상

DLSS 선명화 기능은 더 이상 사용되지 않습니다. 대신 NIS(NVIDIA 이미지 스케일링) 플러그인을 사용하여 선명하게 하는 것이 좋습니다.

DLSS-SR의 성능 이점

언리얼 엔진 5는 렌더링된 씬의 입력 해상도를 현재 출력 해상도에 따라 50~100% 범위에서 자동으로 스케일 조정합니다. 모든 업스케일러는 동일한 자동 입력 해상도를 사용하므로, 사용 가능한 옵션에 따라 퍼포먼스가 비슷해야 합니다.
DLSS-SR은 출력을 업스케일링하고 향상시키는 데 사용되는 AI 및 딥러닝 기술 덕분에 전체적으로 최상의 이미지 품질을 제공한다는 장점이 있습니다. 그 결과 DLSS-SR은 50% 미만의 해상도를 33%의 입력 해상도 '울트라 성능' 모드로 지원하여 이미지 품질을 유지하면서 상당한 성능 향상을 제공할 수 있습니다.

DLSS API 및 UI 문서

DLSS 프로그래밍 가이드는 플러그인에서 DLSS를 구현하는 데 사용되는 NVIDIA NGX API에 대한 자세한 내용을 제공합니다.

RTX UI 개발자 가이드라인(중국어)에서는 DLSS에 권장되는 게임 설정 및 UI에 대한 세부 정보를 제공합니다.

NVIDIA 개발자 블로그 팁: DLSS 언리얼 엔진 4 플러그인 최대한 활용하기에서는 언리얼 엔진 게임과 애플리케이션에서 NVIDIA DLSS를 사용하기 위한 모범 사례와 기타 팁과 요령을 제공합니다.