



NVIDIA Streamline 2.2.1

개발자 가이드

버전 1.0

2023년 10월

2.2.1 새 항목 간소화 - 개요

DLSS 광선 재구성

- 새로운 파티클 레이어 API

DLSS 프레임 생성

- 새로운 디스토션 맵 API
- DLSS 프레임 생성을 지능적으로 활성화/비활성화하여 긍정적인 성능 확장을 보장하는 동적 프레임 생성 지원

기타

- 버그 수정 및 안정성 향상

성공적인 DLSS 3.5 통합을 위한 주요 단계

1. [Streamline 2.x SDK](#)(기능 없이)를 통합하고 수동 후킹, 리소스 상태 추적 등에 집중하세요. - 1일
2. DLL을 로드하기 전에 sl.interposer.dll에서 [공식 NVIDIA & Streamline 이중 서명을 확인합니다](#).
3. [각 DLSS 3 기능에 대한 시스템\(하드웨어/소프트웨어\) 지원 여부 확인](#) - 보고된 하드웨어 지원 여부에 따라 최종 사용자에게 적절한 오류 메시지 표시 - 1시간
4. Streamline를 [통한 DLSS 슈퍼 해상도 통합](#) - 필요한 입력 리소스 전달 및 업스케일링 파이프라인 설정(모든 후처리 전) - 2일
5. DLSS 슈퍼 해상도의 [IQ 및 성능 이점 검증](#) - 1일
6. Streamline를 통한 [DLSS 광선 재구성 통합](#) - 4일
7. DLSS 광선 재구성을 통한 [IQ 및 성능 이점 검증](#) - 1일
8. [Streamline를 통한 Reflex 통합](#) - 적절한 마커, 절전 모드 통화 등을 설정합니다. - 1일
9. 게임 오버레이 또는 Reflex 유효성 검사 HUD에서 FrameView SDK 또는 GFE를 사용하여 [입력 지연 시간이 단축되는지 검증](#) - 4시간
10. [DLSS 프레임 생성 통합](#) - 적절한 상수, 카메라 매트릭스, 입력 리소스(예: 허드리스 및 UI 색상/알파)를 DLSS 초고해상도용으로 표시된 것 외에 추가로 전달합니다. 메뉴 내 또는 장면 전환 시와 같이 적절한 경우 DLSS 프레임 생성도 비활성화해야 합니다 - 2일
11. [Streamline ImGui 플러그인을 사용하여 입력이 올바른지](#) 검증하고 개발 DLL을 사용하여 버퍼 시각화 - 1일
12. DLSS 프레임 생성의 IQ 및 퍼펙트 이점이 검증되면 워터마크가 있는 DLL을 NV에서 프

로덕션 준비가 완료된 비워터마크 DLL로 교체합니다.

총 예상 시간: **~14일**

DLSS 최고 해상도

자세한 정보는 [Streamline DLSS 프로그래밍 가이드](#) 또는 전체 문서에서 확인할 수 있습니다.

[전체 DLSS 프로그래밍 가이드](#)

기본 통합 체크리스트

☐ 초기화 시 게임별 애플리케이션 ID가 사용됩니다.

- 이 기능이 없으면 향후 하드웨어 지원, 버그 수정 및 DLSS 슈퍼 해상도에 대한 OTA 업데이트가 작동하지 않을 수 있습니다.

☐ DLSS 슈퍼 해상도가 가능한 한 후처리 시작에 가깝게 통합되었는지 확인합니다.

- 후처리 후에 DLSS SR을 통합하면 블룸 등의 후처리 효과로 인해 이미지 품질 아티팩트 또는 기타 시각적 문제가 발생할 수 있습니다.

☐ DLSS가 활성화된 경우 mip맵 바이어스 설정

- 이 기능이 없으면 텍스처가 흐릿하거나 번지거나 저해상도로 보입니다.

☐ 모든 장면, 머티리얼 및 오브젝트의 모션 벡터가 정확합니다.

- 엠벡이 잘못되면 고스트 현상 및 기타 IQ 문제가 발생할 수 있습니다.

☐ 정적 장면 해결 및 호환 지터 확인([섹션 8.3](#) 참조)

- 지터가 잘못되거나 누락되면 안티앨리어싱이 제대로 작동하지 않고 깜박임이 증가할 수 있습니다.

☐ 노출 값이 매 프레임마다 올바르게 전송됨(또는 자동 노출이 활성화됨)

- 부적절한 노출은 고스트, 깜박임 및 기타 IQ 문제를 증가시킬 수 있습니다.

☐ DLSS 모드가 쿼리되고 UI에서 사용자가 선택할 수 있거나 동적 해상도 지원이 활성화되고 테스트됩니다.

- 이 모드를 통해 최종 사용자는 성능과 IQ의 절충점을 더 잘 제어할 수 있습니다. 모든 모드가 쿼리되어 표시되지 않으면 최종 사용자는 해당 모드를 선택할 수 없습니다.

☐ 워터마크가 없는 정식 프로덕션 DLSS 라이브러리(nvngx_dlss.dll)는 릴리스 빌드에 패키지로 제공됩니다.

- 게임에 엔비디아 기밀 워터마크를 포함시키지 마세요.

☐ 장면 변경, 시점 변경(1인칭에서 3인칭으로) 또는 컷신에서 카메라 점프 시 카메라 재설정 플래그를 전달합니다.

□ DLSS가 더 이상 필요하지 않은 경우 NGX 정리/종료 절차를 수행합니다.

- 그렇지 않으면 리소스/메모리가 유출됩니다.

중요: DLSS는 메인 렌더 타겟의 기본 업스케일 패스만 대체해야 하며 색도, 반사 등과 같은 보조 버퍼에는 사용해서는 안 됩니다.

DLSS 광선 재구성

자세한 정보는 *DLSS-RR 프로그래밍 간소화 가이드* 또는 *DLSS-RR 통합 가이드*에서 확인할 수 있으며, 이 두 가이드는 모두 DLSS-RR SDK 패키지의 일부로 제공됩니다. DLSS-Ray 재구성 빠른 시작 가이드는 DLSS 슈퍼 해상도 빠른 시작 가이드와 매우 유사합니다.

기본 통합 체크리스트

☐ 초기화 시 게임별 애플리케이션 ID가 사용됩니다.

- 이 기능이 없으면 향후 하드웨어 지원, 버그 수정 및 DLSS 기능에 대한 OTA 업데이트가 작동하지 않을 수 있습니다.

☐ DLSS-RR은 DLSS 슈퍼 해상도에 추가된 기능입니다. DLSS 슈퍼 해상도에 설정된 것과 동일한 화질 모드를 사용합니다. 따라서 DLSS 슈퍼 해상도가 먼저 통합되어 있는지 확인하세요.

☐ 후처리를 시작할 때(또는 시작에 가깝게) DLSS 광선 재구성 및 DLSS 슈퍼 해상도를 모두 통합해야 합니다.

- 후처리 후에 통합하면 블룸 등의 후처리 효과로 인해 이미지 품질 아티팩트 또는 기타 시각적 문제가 발생할 수 있습니다.

☐ NRD와 같은 파이프라인의 다른 디노이저가 완전히 비활성화되어 있는지 확인하세요. ☐

DLSS RR이 활성화된 경우 맵 바이어스 설정.

- 이 기능이 없으면 텍스처가 흐릿하거나 번지거나 저해상도로 보입니다.

☐ 기타 필수 버퍼는 DLSS RR 프로그래밍 가이드의 섹션 3.4에 언급된 대로 제공됩니다.

- 디퓨즈 및 스페큘러 알베도, 노멀, 러프니스, 입력 컬러, 모션 벡터, 뎀스, 스페큘러 모션 벡터(또는 해당 매트릭스가 있는 스페큘러 히트 거리)

☐ DLSS 슈퍼 해상도 및 DLSS 프레임 생성에 제공되는 것과는 다른 선형 깊이를 제공해야 합니다

- 이를 위해 특별히 제공된 `kBufferTypeLinearDepth`를 사용하세요.
- 제공된 뎀스 버퍼에 z 순서가 반전된 경우 반전된 뎀스 비트를 설정하세요.

☐ 모든 장면, 머티리얼 및 오브젝트에 대한 모션 벡터 / 스페큘러 모션 벡터가 정확합니다.

- 모션 벡터가 잘못되면 고스트 현상 및 기타 IQ 문제가 발생할 수 있습니다.

☐ 정적 장면 해결 및 호환 지터 확인(DLSS 프로그래밍 가이드의 8.3절 참조)

- 지터가 잘못되거나 누락되면 안티앨리어싱이 제대로 작동하지 않고 깜박임이 증가할 수 있습니다.

☐ DLSS 모드가 켜리되고 UI에서 사용자가 선택할 수 있거나 동적 해상도 지원이 활성화되고 테스트됩니다.

- 이 모드를 통해 최종 사용자는 성능과 IQ의 절충점을 더 잘 제어할 수 있습니다. 모든 모드가 켜리되어 표시되지 않으면 최종 사용자는 해당 모드를 선택할 수 없습니다.

☐ DLSS RR 활성화/비활성화 토글을 사용할 수 있습니다.

- 일반적으로 레이 트레이싱 또는 패스 트레이싱을 사용할 때 이 기능을 사용할 수 있어야 합니다.
- 레이 트레이싱과 패스 트레이싱은 DX12와 별칸에서만 사용할 수 있습니다.

☐ 워터마크가 없는 정식 프로덕션용 DLSS 광선 재구성 라이브러리(nvngx_dlssd.dll)가 릴리스 빌드에 패키징되어 있습니다.

- 게임에 엔비디아 기밀 워터마크를 포함시키지 마세요.

☐ 장면 변경, 시점 변경(1인칭에서 3인칭으로) 또는 컷신에서 카메라 점프 시 카메라 재설정 플래그를 전달합니다.

☐ DLSS가 더 이상 필요하지 않은 경우 NGX 정리/종료 절차를 수행합니다.

- 그렇지 않으면 리소스/메모리가 유출됩니다.

DLSS 프레임 생성

자세한 정보는 [DLSS FG 프로그래밍 가이드](#)에서 확인할 수 있습니다.

기본 통합 체크리스트

☐ **덱스 버퍼, 모션 벡터, HUD가 없는 컬러 버퍼, UI 컬러 버퍼** 등 필요한 모든 입력이 Streamline로 전달됩니다.

- 입력 버퍼 중 하나라도 누락되거나 올바르게 표시되지 않으면 물체가 흐릿하게 보이거나 화면의 문자가 깨지거나 분리되는 현상, UI 및 HUD 요소의 프레임 증가 및 불안정성 등 심각한 IQ 아티팩트가 발생합니다.
- 디버그 시각화를 사용하여 (1) 버퍼가 전달되었는지, (2) 프레임이 정렬되었는지 확인합니다.

☐ 각 프레임에 대한 공통 상수 및 프레임 인덱스는 **슬릿셋컨스턴트** 및 **슬릿피처컨스턴트** 메서드를 사용하여 제공됩니다(Streamline 2.0.1 전용 API).

- 여기에 잘못된 값을 입력하면 FG와 반사 통합이 불일치하여 이미지가 자홍색으로 착색되거나 입력 지연 및 끊김 현상이 발생할 수 있습니다.

☐ 태그가 지정된 모든 버퍼는 현재 프레임에서 유효하며 다른 용도로 재사용되지 않습니다.

- 현재 시간 이전에 태그가 지정된 버퍼를 재사용하면 전체 이미지(또는 재사용된 리소스의 양에 따라 일부만)가 심각하게 손상될 수 있습니다.

☐ **고유 뷰포트 ID 0으로 태그할 버퍼**

- DLSS 프레임 생성에는 **SwapChain::Present** 호출 중에 사용되는 덱스, 모션 벡터 및 HUD가 없는 컬러 버퍼가 필요하므로 프레임이 표시되기 전에 이러한 버퍼를 재사용, 파괴 또는 변경하려면 수명 주기를 올바르게 지정해야 합니다.
- DLSS FG는 다중 뷰포트를 지원하지 않으므로(DLSS 슈퍼 해상도 및 SL SDK에서 지원하는 다른 기능과 달리), DLSS 프레임 생성을 위한 모든 입력 버퍼는 0의 뷰포트 ID를 사용해야 합니다.

☐ 공통 상수와 함께 제공된 프레임 인덱스가 제시된 프레임과 일치하는지 확인합니다.

- 여기에 잘못된 값을 입력하면 FG와 반사 통합이 불일치하여 이미지가 자홍색으로 착색되거나 입력 지연 및 끊김 현상이 발생할 수 있습니다.

☐ 입력은 카메라 매트릭스 및 동적 오브젝트뿐만 아니라 올바른 모양으로 스트림라인에 전달됩니다.

- 잘못된 입력은 흐릿하거나 번진 물체, 고스트 현상 증가, 분리된 캐릭터, 깜박이는 UI, 일반적으로 이미지가 흐릿해지는 등의 심각한 IQ 문제를 유발합니다.
- 동적 모션은 플레이어 카메라 모션과 독립적으로 움직이는 모든 오브젝트를 표시해야 합니다. 이 휴리스틱은 올바른 카메라 매트릭스 형식을 식별하는 데 도움이 됩니다.
- 카메라 매트릭스에 지터 정보가 없어야 합니다.
- MVEC가 지터링된 경우 MVECJittered SL 부울 또는 NGX Equivalent를 설정합니다.

☐ 애플리케이션이 sl.interposer.dll의 서명을 확인하여 정품 NVIDIA 라이브러리인지 확인합니다.

- 이 기능이 없는 경우 악의적인 사용자가 자체 버전의 sl.interposer.dll을 삽입하여 렌더링 파이프라인을 탈취할 수 있습니다(치터, 해커 등이 사용할 수 있음).

□ 동적 해상도 요구 사항 충족(게임에서 동적 해상도를 지원하는 경우)

- DLSS 프레임 생성은 MVec 및 텍스 버퍼 익스텐션의 동적 해상도를 지원합니다. 동적 해상도는 DLSS 또는 애플 방법을 통해 수행할 수 있습니다.
- DLSS 프레임 생성은 모든 후처리가 완료된 최종 색상 버퍼를 ~~생성~~ 색상 버퍼는 고정된 크기여야 하며 프레임별로 크기를 조정할 수 없습니다.
- DLSS 프레임 생성 동적 해상도 모드가 활성화되면 애플리케이션은 프레임 단위로 MVec 및 텍스 버퍼의 크기를 다르게 전달할 수 있습니다. 이를 통해 애플리케이션이 렌더링 부하를 동적으로 원활하게 변경할 수 있습니다.

□ 게임 일시정지, 로딩 중, 메뉴 및 일반적으로 게임 프레임을 렌더링하지 않을 때, 해상도 및 전체 화면 대 창 모드를 수정할 때 DLSS 프레임 생성을 비활성화합니다(sl::DLSSGOptions::모드를 eDLSSGModeOff로 설정하여).

- 이렇게 하면 전반적인 게임 안정성과 경험이 향상됩니다.

□ 모션 블러의 양을 줄이고, DLSS 프레임 생성을 활성화하면 모션 블러의 거리/크기를 절반으로 줄입니다.

- 잘못된 모션 블러는 추가 생성 프레임으로 인해 모션 블러가 본질적으로 두 배가 되는 IQ 아티팩트로 이어질 수 있습니다.

□ Reflex가 제대로 통합되었는지 확인합니다(Reflex 프로그래밍 가이드의 체크리스트 참조).

- FG의 올바른 통합은 Reflex의 올바른 통합에도 절대적으로 의존합니다. Reflex 통합이 올바르지 않으면 FG를 활성화할 때 입력 지연 시간이 크게 증가하고 마젠타 색조의 이미지가 표시됩니다.

□ DLSS 프레임 생성을 활성화/비활성화하기 위한 게임 내 UI가 구현되었으며 RTX UI 가이드라인을 따릅니다.

- 최종 사용자를 위한 직관적이고 일관성 있는 작동 방식

□ 워터마크가 없는 정식 프로덕션 라이브러리만 릴리스 빌드에 패키징됩니다.

- 그렇지 않으면 화면 왼쪽 상단에 "엔비디아 기밀 - NDA에 따라 제공 - 어떠한 방식으로도 배포하지 마세요"라는 메시지가 표시됩니다.

□ 기능을 실행하는 동안 Streamline 및 DLSS 프레임 생성 로그 파일에 오류나 예기치 않은 경

고가 표시되지 않습니다.

- 오류와 경고에 따라 충돌 가능성, IQ 아티팩트, 성능 저하 등 다양한 결과가 발생할 수 있습니다.

☐ 장면 변경, 시점 변경(1인칭에서 3인칭으로) 또는 컷신에서 카메라 점프 시 카메라 재설정 플래그를 전달합니다.

☐ 반투명 물체는 약간의 주의와 관심이 필요합니다. 물체의 움직임을 추적해야 하는지, 아니면 물체를 통해 보이는 물체의 움직임을 추적해야 하는지. 움직이는 물체(예: 총의 조준경이 눈앞까지 올라오지 않는 경우)인 경우, 해당 물체에 대한 초속도와 깊이 값을 제공해야 합니다. 이 경우 일반적으로 투명 오브젝트에는 필요하지 않은 추가 코딩이 필요할 수 있습니다.

☐ 게임 내 FPS를 계산할 때는 항상 DLSS-FG에 의해 생성된 추가 프레임을 고려하세요.

이는 프레임 단위로 쿼리할 수 있으며, 다음과 같이 반환됩니다.

`DLSSGState::numFramesActuallyPresented`는 해당 프레임에 대해 DLSS 프레임 생성이 활성화되었는지 여부에 따라 1 또는 2가 됩니다.

- 현재 호출 사이의 시간을 계산하여 실시간 FPS를 계산하는 경우 이 값으로 시간을 나누어야 합니다.
- 총 시간을 제시된 게임 프레임 수로 나누어 평균 FPS를 계산하는 경우,

`DLSSGState::numFramesActuallyPresented`는 해당 기간 동안 누적되어야 하며 실제로 제시된 총 프레임 수를 나타내게 됩니다.

반사

자세한 정보는 [SL 리플렉스 프로그래밍 가이드에서](#) 확인할 수 있습니다.

기본 통합 체크리스트

- ☐ 반사 지연 시간 기본 상태는 "켜짐"입니다.
- ☐ 모든 반사 모드(끄기, 켜기, 켜기 + 부스트)가 올바르게 작동합니다.
- ☐ 반사 테스트 유틸리티의 PC 지연 시간(PCL)이 0보다 높습니다.
 - 0은 일부 마커가 올바르게 통합되지 않았음을 의미합니다.
 - NSight 시스템에서 마커 배치를 시각화하여 Reflex를 활성화할 때와 비활성화할 때 시뮬레이션 마커가 JIT로 이동하는 것을 확인하는 등 디버깅에 도움을 줄 수 있습니다.
- ☐ DLSS 프레임 생성이 활성화된 상태에서 반사 테스트 유틸리티 보고서가 경고 없이 통과되었습니다.
 - Reflex가 켜기/낮은 지연 시간 모드에서 올바르게 작동했음을 나타냅니다. 입력 지연 시간이 20% 이상 감소했어야 합니다.
- ☐ 반사 테스트 유틸리티 보고서가 반사 기능이 "켜짐"으로 설정된 상태에서 경고 없이 통과되었습니다.
- ☐ 리플렉스는 리플렉스 활성화 시 평균 프레임 속도(fps)에 4% 이상 큰 영향을 미치지 않습니다.
 - 리플렉스를 '켜기 + 부스트'로 설정하면 절대적으로 가장 낮은 지연 시간을 제공하기 위해 성능에 미치는 영향이 약간 더 커질 것으로 예상됩니다.
- ☐ Reflex가 "켜기 + 부스트"로 설정된 상태에서 Reflex 테스트 유틸리티 보고서가 경고 없이 통과되었습니다
- ☐ Reflex 저지연 모드 상태와 관계없이 항상 Reflex 마커가 전송됩니다.
- ☐ 마우스 왼쪽 버튼을 누르면 리플렉스 플래시 표시기가 나타납니다
- ☐ 리플

렉스 UI 설정은 UI 가이드라인을 따릅니다.

☐ 키 바인딩 메뉴가 제대로 작동합니다(F13 없음).

☐ 비 NVIDIA GPU에서 PC 지연 시간(PCL)이 0보다 높음

☐ Reflex UI 설정이 비활성화되거나 비 NVIDIA GPU에서 사용할 수 없습니다.

반사 통합 체크리스트 단계

1. [반사 확인 도구](#) 다운로드
2. FrameView SDK 설치
 - FrameView SDK 설치 프로그램(**FVSDKSetup.exe**)을 더블 클릭합니다.
 - 시스템 재시작
3. 관리자 모드 명령 프롬프트에서 **ReflexTestSetup.bat**을 실행합니다.
 - 이렇게 하면 반사 플래시 표시기가 강제로 활성화되고, 확인 HUD가 활성화되고, 반사 테스트 프레임워크가 설정되고, ReflexTest.exe가 시작됩니다.
4. 반사 저지연 모드 확인
 - 게임 실행
 - 게임이 전체 화면 전용으로 실행 중인지 확인하기
 - VSYNC가 **비활성화**되어 있는지 확인
 - MSHybrid 모드가 활성화되어 있지 않은지 확인합니다.
 - 게임 GUI와 검증 HUD에서 반사 지연 시간이 짧은 모드가 "켜짐"으로 설정되어 있는지 확인합니다.
 - UI에 재설정/기본값 버튼이 있는 경우 이를 사용합니다.
 - UI에서 반사 모드를 순환하고 검증 HUD 내의 "반사 모드"와 일치하는지 확인합니다.
5. 달리기 반사 테스트
 - DLSS 프레임 생성을 지원하는 타이틀의 경우 DLSS-FG를 **활성화**합니다.
 - 게임에서 **Alt + T**를 눌러 테스트를 시작합니다(신호음 2회).
 - 테스트 완료 후 결과 분석(신호음 3회)
 1. 테스트는 약 5분 정도 소요됩니다.
 2. ReflexTest.exe 출력에서 "통과됨"을 찾아 경고를 확인합니다.
 - DLSS-FG(사용 가능한 경우)를 비활성화하고 Reflex를 "**켜기**"로 설정합니다.
 - 게임에서 **Alt + T**를 눌러 테스트를 시작합니다(신호음 2회).
 - 테스트 완료 후 결과 분석(신호음 3회)
 1. 테스트는 약 5분 정도 소요됩니다.

2. ReflexTest.exe 출력에서 "통과됨"을 찾아 경고를 확인합니다.
 - 리플렉스를 "켜기 + 부스트"로 설정합니다.
 - 게임에서 **Alt + T**를 눌러 테스트를 시작합니다(신호음 2회).
 - 테스트 완료 후 결과 분석(신호음 3회)
 1. 테스트는 약 5분 정도 소요됩니다.
 2. ReflexTest.exe 출력에서 "통과됨"을 찾아 경고를 확인합니다.
6. 리플렉스가 꺼져 있어도 마커가 항상 전송되는지 테스트합니다.
 - 리플렉션을 "끄기"로 설정
 - 확인 HUD를 보고 마커가 계속 업데이트되고 있는지 확인합니다.
7. 반사 플래시 표시기 테스트
 - 반사 플래시 표시등이 표시되는지 확인합니다.
 - 마우스 왼쪽 버튼을 누르면 회색 사각형이 깜박이는 것을 확인하세요.

- 마우스 왼쪽 버튼을 누르면 확인 HUD의 플래시 표시기가 1씩 증가하는지 확인합니다.

8. UI 확인

- UI가 가이드라인을 따르는지 확인

9. 키 바인딩 확인

- 명령 프롬프트에서 관리자 모드로 **capturePclEtw.bat**을 실행합니다.
- 게임으로 돌아갑니다. 게임 플레이 시작
- 키 바인딩 메뉴를 확인하여 키를 선택할 때 F13이 자동으로 적용되지 않는지 확인하세요.
- 명령 프롬프트로 돌아가서 아무 키나 눌러 박쥐를 종료합니다.

10. 명령 프롬프트에서 관리자 모드로 **ReflexTestCleanUp.bat**을 실행합니다.

- 이렇게 하면 반사 플래시 표시기와 반사 테스트 프레임워크가 비활성화됩니다.

11. 다른 IHV에서 테스트(가능한 경우)

- 다른 IHV 하드웨어 설치
- FrameView SDK를 설치하고 시스템을 재시작합니다.
- 관리자 모드 명령 프롬프트에서 **PrintPCL.exe**를 실행합니다.
- 게임 실행
- 게임에서 **Alt + t**를 누릅니다.
 - 명령 프롬프트에서 PCL 값을 확인합니다. 값이 0.0이 아니라면 PCL이 작동하는 것입니다.
- 명령 프롬프트에서 **Ctrl + c**를 눌러 **PrintPCL.exe**를 종료합니다.
- Reflex UI를 사용할 수 없는지 확인합니다.

12. NVIDIA Reflex 테스트 보고서 및 체크리스트 결과 보내기

- NVIDIA 별칭으로 결과 이메일 보내기: reflex-sdk-support@nvidia.com

자주 묻는 질문

간소화

질문: 타사 오버레이(스팀, 에픽 게임 스토어 등)가 Streamline에서 올바르게 작동하도록 하려면 어떻게 해야 하나요?

A: 타사 오버레이가 스트림라인에서 올바르게 작동하려면 다음 규칙을 따라야 합니다:

- 일반적으로 오버레이는 스왑 체인 및 명령어 대기열에 대해 가정해서는 안 되며, DLSS 프레임 생성이 활성화되면 여러 명령어 대기열과 여러 비동기 프레젠테이션이 있을 수 있습니다**.
- 오버레이는 프레임을 표시하는 데 사용되는 올바른 스왑 체인 및 명령 대기열을 얻기 위해 ``IDXGIFactory::CreateSwapChainXXX``를 가로채야 합니다.
- 엔진에 통합된 경우, 오버레이는 ``slInit``이 호출되기 ****전에**** 초기화되어야 합니다.
- 선택적 플래그인 ``sl::PreferenceFlags::eUseDXGIFactoryProxy``를 사용하면 DXGI 팩토리 v-테이블에 SL hooks을 삽입하지 않을 수 있으며, 이는 오버레이 관련 일부 문제를 해결하는 데 도움이 될 수 있습니다.

질문: D3D 디버그 레이어가 잘못된 리소스 상태에 대해 불만을 제기하는 경우 어떻게 해야 하나요?

A: Streamline 리소스에 태그를 지정할 때 올바른 상태를 제공해 주세요.

질문: 스왑체인::프레젠티에서 충돌이 발생하거나 이와 유사한 예기치 않은 동작이 발생하면 어떻게 해야 하나요?

A: `dxgi.lib/d3d12.lib`를 `sl.interposer.dll`과 함께 연결하지 않았는지 다시 한 번 확인해 주세요. 문제가 없는 경우 Streamline 로그에서 원인에 대한 추가 지침을 확인하세요.

Q: NVIDIA 하드웨어가 아닌 하드웨어에서 Streamline을 활성화해야 하나요?

A: 예. Streamline은 NV에서 개발한 SDK이지만, 모든 하드웨어 공급업체에서 작동하는 여러 기능을 지원합니다(예: NVIDIA 이미지 샤프닝(NIS), NVIDIA 실시간 노이즈 제거기(NRD), 리플렉스(PCL))

Q: Streamline 2.1에는 Streamline에 OTA 기능이 도입되었습니다. OTA로 업데이트할 수 있는 항목은 무엇이며 이전 DLSS OTA와 어떻게 다른가요?

A: Streamline 2.1은 모든 Streamline 플러그인에 대한 무선(OTA) 기능을 확장합니다. 이전에는 이 OTA가 DLSS 업데이트에만 제한되었습니다. . 성능, 부드러움, 메모리 최적화는 물론 버그 수정 및 보안 강화 등 상당한 개선이 이루어졌습니다. Streamline OTA는 NVIDIA GPU에서 지원됩니다.

DLSS 최고 해상도

질문: DLSS 고해상도 출력이 올바르게 보이지 않습니다. 무엇이 문제인가요?

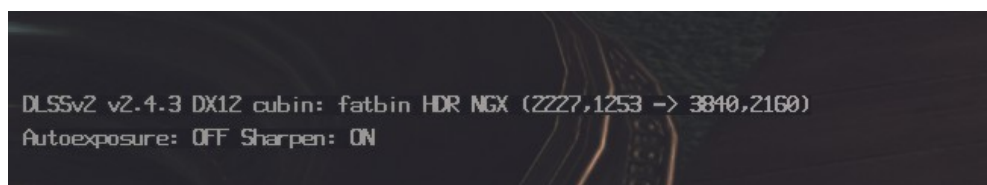
A: DLSS 출력이 제대로 보이지 않는 경우(예: *카메라를 움직일 때 불안정함*) 다음 항목을 확인하세요. 또한 자세한 문제 해결 방법은 DLSS 최고해상도 프로그래밍 가이드의 [섹션 8](#)을 참조하세요:

- 모션 벡터가 픽셀 공간에 있는 경우 스케일 인자 `sl::Constants::mvecScale` 은 다음과 같아야 합니다.
{1 / 렌더링 너비, 1 / 렌더링 높이}
- 모션 벡터가 정규화된 -1,1 공간에 있는 경우 스케일링 인자 `sl::Constants::mvecScale`은 {1, 1}이어야 합니다.
- 지터 오프셋 값이 픽셀 공간에 있는지 확인합니다.
- NVSDK NGX_Parameter_FreeMemOnRelease가 `slFreeResources`로 대체됩니다.
- 태그된 모션 벡터 버퍼의 크기와 범위에 따라 NVSDK NGX_DLSS_Feature_Flags_MVLowRes가 자동으로 처리됩니다.

질문: DLSS 슈퍼 해상도가 활성화되어 있는지 확인하려면 어떻게 해야 하나요?

A: 애플리케이션을 시작하기 전에 "ngx_driver_onscreenindicator.reg"를 실행하여 DLSS 온스크린 표시기 유틸리티를 활성화할 수 있습니다. DLSS 고해상도가 활성화되면 게임 내 오버레이에 확인 메시지가 표시됩니다.

완료되면 "ngx_driver_onscreenindicator_off.reg"를 실행하여 표시기를 비활성화할 수 있습니다.



질문: DLSS 고해상도 3.1.x는 2.1.x와 비교하여 어떤 새로운 기능을 제공하나요? 답변: DLSS

3.1.x에는 다음과 같은 기능이 추가되었습니다:

- OTA를 통한 사전 설정 업데이트([2.4절](#) 참조)
- 모델 선택 API([섹션 3.12](#) 참조)
- 기능 지원 요구 사항 쿼리하기([2.3절](#) 참조)
- 별칸 확장 요구 사항 쿼리하기([2.3.1 섹션](#) 참조)
- DLSS 선명화 사용 중단([섹션 3.11](#) 참조)

질문: DLSS 샤프닝은 어떻게 되나요?

답변: 이전 버전의 SDK에서 제공된 선명화 및 소프트닝 패스는 이제 더 이상 사용되지 않습니다. 모든 개발자는 DLSS SDK를 통해 또는 GitHub에서 직접 이미지 스케일링 SDK를 사용할 것을 권장합니다.

DLSS 광선 재구성

Q: DLSS 광선 재구성이란 무엇인가요?

A: DLSS 광선 재구성은 모든 GeForce RTX GPU를 위한 새로운 신경망으로, 경로 추적 및 고집적 광선 추적 콘텐츠의 이미지 품질을 향상시킵니다. 광선 재구성은 수작업으로 조정된 디노이저를 샘플링된 광선 사이에 더 높은 품질의 픽셀을 생성하는 NVIDIA 슈퍼컴퓨터로 학습된 AI 네트워크로 대체합니다.

질문: DLSS 광선 재구성을 제 타이틀에 통합하고 싶습니다. 필요한 모든 것이 Streamline 2.2 SDK에 포함되어 있나요?

A: DLSS 광선 재구성 기능은 NDA Streamline 2.2 패키지로 제한되며, Github의 공개 SDK에서는 지원되지 않습니다. 얼리 액세스를 이용하려면 NVIDIA 계정 관리자에게 문의하거나 여기에서 등록하여 공개 출시 시 알림을 받으세요 (<https://developer.nvidia.com/rtx/dlss/notify-me>).

Q: DLSS 광선 재구성을 지원하는 GeForce GPU는 무엇인가요?

A: 필요한 텐서 코어가 탑재된 모든 GeForce RTX GPU는 DLSS 광선 재구성을 지원할 수 있습니다. 이 하드웨어 요구 사항은 DLSS 슈퍼 해상도와 동일합니다.

질문: 다른 DLSS 기술을 활성화하지 않고 DLSS 광선 재구성을 활성화할 수 있나요?

A: DLSS 광선 재구성은 DLSS 슈퍼 해상도가 활성화되어 있지 않으면 활성화해서는 안 됩니다.

질문: DLSS 광선 재구성을 활성화하면 성능 이점을 볼 수 있나요?

A: 다중 레이 트레이스 효과 또는 풀 레이 트레이싱이 적용된 게임 콘텐츠에는 일반적으로 각 특정 조명 유형에 맞게 조정된 여러 개의 디노이저가 있습니다. DLSS 레이 재구성은 이러한 여러 노이즈 제거기를 단일 신경망으로 대체하므로 레이 트레이스가 많이 사용되는 장면에서 더 빠르게 실행할 수 있습니다. 그러나 성능 이점은 장면, 설정, 사용 중인 GPU에 따라 달라집니다.

질문: 레이 재구성에 게임 메뉴에 별도의 설정이 있어야 하나요? A: 자세한 내용은 RTX

UI 가이드라인을 참조하세요.

질문: 이미 통합한 다른 디노이저는 어떻게 해야 하나요?

A: DLSS-RR이 활성화되면 다른 모든 노이즈 제거기(예: NRD)가 완전히 비활성화되어 있는지 확인하세요.

질문: 레이 재구성을 파이프라인의 여러 위치에 특수 패스로 통합해야 하나요?

A: DLSS 슈퍼 샘플링과 마찬가지로 광선 재구성은 단일 통합 지점으로 설계되었습니다.

Q: 레이 재구성은 어떤 종류의 레이 트레이싱/패스 트레이싱 모델과 호환되나요?

A: DLSS 광선 재구성은 모든 레이 트레이싱 및 패스 트레이싱 모델과 호환되도록 설계되었습니다.

질문: DLSS 광선 재구성은 어떤 운영 체제에서 사용할 수 있나요?

A: DLSS 광선 재구성은 Windows, Linux 및 Proton에서 지원됩니다(DLSS 광선 재구성을 활성화하려면 현재 Proton Experimental이 필요합니다).

질문: DLSS 광선 재구성은 모든 DLSS 슈퍼 해상도 모드에서 작동하나요?

답변: 현재 DLSS 광선 재구성이 활성화된 경우 '초고성능' 및 'DLAA' 모드는 지원되지 않습니다. DLSS 슈퍼 해상도의 다른 모든 모드는 지원됩니다.

DLSS 프레임 생성

질문: DLSS 프레임 생성을 활성화하기 위한 HW/SW 요구 사항은 무엇인가요? A: DLSS

프레임 생성을 사용하려면 다음 기준을 충족해야 합니다:

- GeForce RTX 40 시리즈 GPU(데스크톱 또는 노트북)
- 최소 Windows OS 버전 Win10 20H1(버전 2004, 빌드 19041 이상)
- 설정 : 시스템 을 통해 **Windows 하드웨어 가속 GPU 스케줄링(HWS)을 활성화해야 합니다.** 디스플레이 : 그래픽 : 기본 그래픽 설정 변경.

질문: 시스템에서 요구 사항이 충족되었는지 확인하려면 어떻게 해야 하나요?

A: DLSS-FG가 올바르게 작동하려면 사용자 컴퓨터의 OS, 드라이버 및 기타 설정과 관련된 특정 요구사항이 충족되어야 합니다. DLSS-FG 구성을 열고 모든 요구 사항이 충족되는지 확인하려면 다음 코드 스니펫을 사용할 수 있습니다:

```
Unset

sl::기능요건 요구사항{};

if (SL_FAILED(result, slGetFeatureRequirements(sl::kFeatureDLSS_G,
requirements))))
{
    // slInit에서 기능이 요청되지 않았거나 로드, 로그 확인, 오류 처리에 실패했습니다.
}
else
{
    // 피처가 로드되면, 요구사항을 확인할 수 있습니다. requirements.flags &
    FeatureRequirementFlags::eD3D11지원되는 요구사항.flag &
    FeatureRequirementFlags::eD3D12지원되는 요구사항.flag &
    FeatureRequirementFlags::eVulkan지원되는 요구사항.maxNumViewports.
    // 등등 ...
}
```

질문: 리소스에 올바르게 태그를 지정하지 않으면 어떻게 되나요?

A: 태그된 리소스의 유효성을 보장할 수 없는 경우(예: 게임이 로드 중이거나 일시 중지되었거나 메뉴에 왜 비디오 컷 장면 재생

중 등) 안정성 또는 IQ 문제를 방지하기 위해 모든 태그를 null 포인터로 설정해야 합니다.

자세한 내용은 [DLSS FG 프로그래밍 가이드](#)의 섹션 5.0을 참조하세요.

질문: 게임이 로딩 중이거나 메뉴 화면 또는 컷신 등을 재생할 때 무작위로 충돌이 발생하는 이유는 무엇인가요?

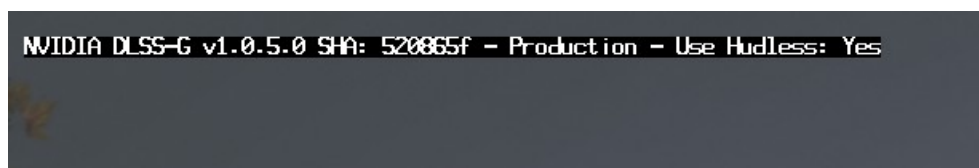
A: 태그된 리소스의 유효성을 보장할 수 없는 경우(예: 게임이 로드 중이거나 일시 중지되었거나 메뉴에 왜 비디오 컷 장면 재생 중 등) 안정성 또는 IQ 문제를 방지하기 위해 모든 태그를 null 포인터로 설정해야 합니다.

자세한 내용은 [DLSS FG 프로그래밍 가이드의](#) 섹션 5.0을 참조하세요.

질문: DLSS 프레임 생성이 활성화되어 있는지 확인하려면 어떻게 해야 하나요?

A: 애플리케이션을 시작하기 전에 "ngx_driver_onscreenindicator.reg"를 실행하여 DLSS 온스크린 표시기 유틸리티를 활성화할 수 있습니다. DLSS 프레임 생성이 활성화되면 게임 내 오버레이에 확인 메시지가 표시됩니다.

완료되면 "ngx_driver_onscreenindicator_off.reg"를 실행하여 표시기를 비활성화할 수 있습니다.




개발자는 Streamline 2.0에 도입된 sl.imgui.dll 오버레이를 활용할 수도 있습니다.



질문: 허드리스 버퍼에 태그가 있는지 확인하려면 어떻게 해야 하나요?

A: 애플리케이션을 시작하기 전에 "ngx_driver_onscreenindicator.reg"를 실행하여 DLSS 온스크린 표시기 유틸리티를 활성화할 수 있습니다. DLSS 고해상도가 활성화되면 오버레이에 "허드리스 사용: 허드리스 버퍼가 태그된 경우 오버레이에 "허드리스 사용: 예"가 표시됩니다.

완료되면 "ngx_driver_onscreenindicator_off.reg"를 실행하여 표시기를 비활성화할 수 있습니다.

A screenshot of a terminal window with a dark background. The text "NVIDIA DLSS-G v1.0.5.0 SHA: 520865f - Production - Use Hudless: Yes" is displayed in a light-colored monospace font. On the left side of the terminal, there is a small, faint, golden-colored logo that appears to be the NVIDIA logo.

```
NVIDIA DLSS-G v1.0.5.0 SHA: 520865f - Production - Use Hudless: Yes
```

개발자는 Streamline 2.0.1에 도입된 sl.imgui.dll 오버레이를 활용할 수도 있습니다.

1. DLSS FG가 실행 중인지 확인
2. DLSS 프레임 생성이 실행 중인지 확인
3. 디버그 키(CTRL+SHIFT+INS)를 누릅니다.
4. 모든 버퍼 시각화(개발자가 허드리스 버퍼가 올바른지 확인할 수 있음)
5. 화면에서 기능 확인



Q: DLSS 프레임 생성이 HDR과 호환되나요?

A: 예, DLSS 프레임 생성은 HDR과 호환됩니다. 그러나 DLSS 프레임 생성은 현재 FP16 픽셀 형식 및 sRGB 색 공간을 지원하지 않습니다.

질문: DLSS 프레임 생성을 활성화하면 성능이 저하됩니다. 왜 그런가요?

A: DLSS 프레임 생성은 실행 비용이 한정되어 있습니다. 프레임 생성에 걸리는 시간이 게임 엔진을 통해 동일한 프레임을 렌더링하는 데 걸리는 시간보다 짧으면 FPS가 향상됩니다.

그러나 게임이 매우 빠르게 렌더링되는 경우(일반적으로 200fps 이상이지만 경우에 따라 더 낮을 수 있음) DLSS FG는 실제로 게임의 전체 프레임 속도를 느리게 할 수 있습니다. 동적 프레임 생성은 이 시나리오에 도움이 되도록 설계되었습니다.

그러나 사용 가능한 비디오 메모리의 양이 DLSS 프레임 생성에 필요한 양보다 적으면 DLSS FG로 인해 게임의 전체 프레

임 속도가 실제로 느려질 수 있습니다.

질문: DLSS 프레임 생성을 통합했지만 활성화해도 성능에 변화가 없습니다. 그 이유는 무엇인가요?

A: DLSS 프레임 생성은 게임 엔진이 기본적으로 추적하지 않을 수 있는 `Present()` 호출을 추가로 추가합니다.

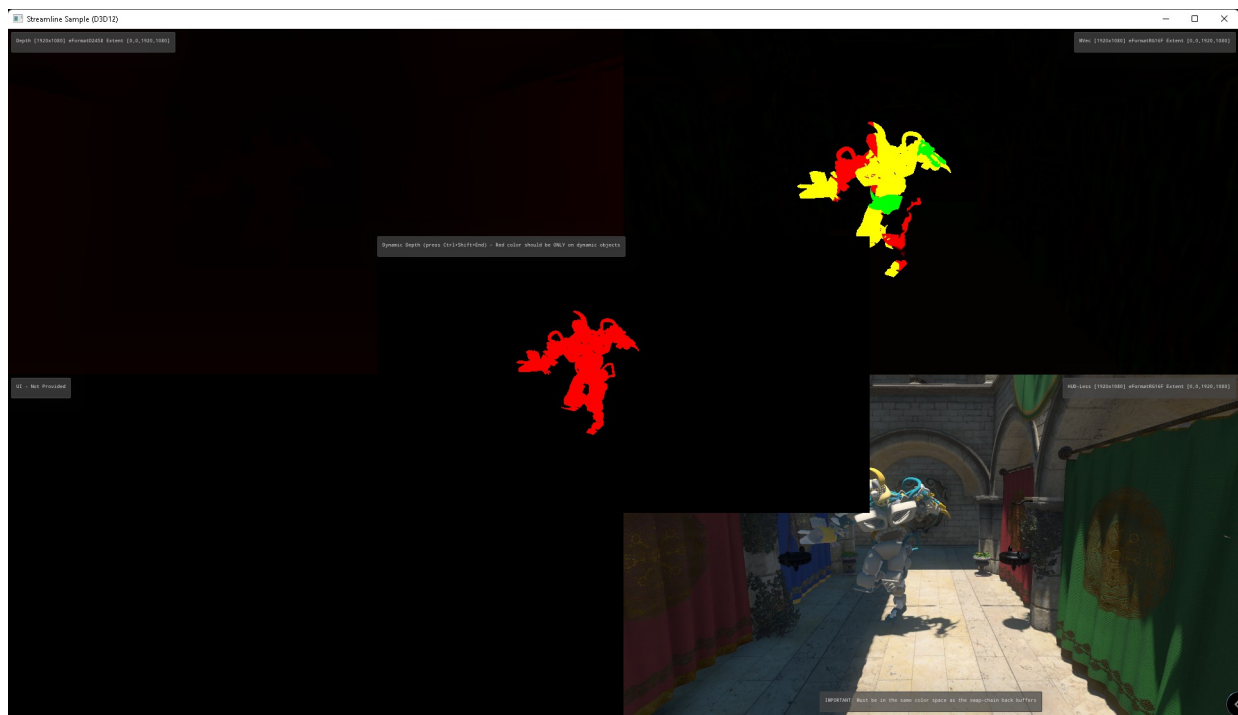
결과적으로 게임 내 FPS 수치는 프레임뷰, Geforce Experience, CapFrameX, Windows 게임 바 등과 같은 도구를 통해 앱 프로세스 외부에서 보고된 수치와 일치하지 않을 수 있습니다.

Streamline API는 엔진의 fps 보고 기능에 연결해야 하는 `DLSSGState::numFramesActuallyPresented`를 통해 표시되는 프레임 수에 대한 정보를 제공합니다. 이렇게 하면 게임 엔진 자체뿐만 아니라 외부 툴에서 보고하는 프레임 수와도 일치하게 됩니다.

질문: 카메라 데이터와 모션 벡터가 정렬되었는지 확인하려면 어떻게 해야 하나요?

A: 개발자는 Streamline 2.0.1에 도입된 `sl.imgui.dll` 오버레이를 활용할 수도 있습니다.

1. DLSS FG가 실행 중인지 확인
2. DLSS 프레임 생성이 실행 중인지 확인
3. 디버그 키(CTRL+SHIFT+INS)를 누릅니다.
4. '동적 개체' 보기로 들어갑니다(CTRL+SHIFT+END로 보기를 순환).
5. 화면에서 기능 확인



질문: DLSS 프레임 생성이 활성화된 상태에서 VSYNC를 활성화하면 게임이 끊깁니다. 왜 그런가요?

A: Streamline 2.0.1을 사용하는 경우 이제 G-SYNC 호환 디스플레이가 없어도 V-SYNC에 대한 DLSS 프레임 생성 (NVIDIA 제어판에서 설정)이 지원됩니다. 게임 내

V 동기화 설정 지원은 사용할 수 없으며 DLSS 프레임 생성이 활성화된 경우 비활성화된 상태로 유지해야 합니다.

V-SYNC (NV 제어판)	G-SYNC (NV 제어판)	결과	필수 간소화 버전
꺼짐	꺼짐	최대 FPS, 짧은 지연 시간, 티어링	1.3.3+
꺼짐	켜짐	최대 FPS, 짧은 지연 시간, 재생률 이상의 티어링	1.3.3+
켜짐	꺼짐	끊임 없음, 부드러움 없음, 높은 지연 시간	< 2.0.1
켜짐	꺼짐	끊임 없음, 부드러움, 낮은 지연 시간	2.0.1+
켜짐	켜짐	티어링 없음, 부드러움, 최저 지연 시간 VSYNC 켜기 옵션	1.3.3+

드라이버 버전 526.98 이상부터 DLSS 프레임 생성은 G-SYNC 및 G-SYNC 호환 모니터 및 TV에서 VSYNC와 호환됩니다. 이 기능을 활성화하려면 다음과 같이 하세요:

- G-SYNC 활성화: NVIDIA 제어판 --> 디스플레이 --> G-SYNC 설정
- VSYNC를 켭니다: NVIDIA 제어판 --> 3D 설정 --> 3D 설정 관리

참고: 현재 DLSS 프레임 생성은 Streamline 2.0.0a 이하를 사용할 때 G-SYNC 또는 G-SYNC 호환이 되지 않는 디스플레이에서는 VSYNC와 호환되지 않습니다.

질문: 자동 장면 변경 감지 기능은 어떤 기능을 하나요?

A: 자동 장면 변경 감지 기능은 프레임 생성이 상당한 장면 변경 사이에 생성하기 어려운 프레임을 생성하지 않도록 자동으로 방지하는 것을 목표로 합니다. 이 기능은 모든 입력 DLSS 프레임 생성 프레임 쌍에서 게임 내 카메라 방향을 분석하여 이를 수행합니다. 자동 장면 변경 감지 기능은 새로운 DLSS 3 타이틀의 통합을 간소화하고, 모든 DLSS 3 통합과 역호환

되며, 모든 렌더링 플랫폼을 지원합니다.

질문: 동적 프레임 생성 기능은 어떤 기능을 하나요?

A: 동적 프레임 생성을 지능적으로 활성화/비활성화하여 긍정적인 성능 확장을 보장할 수 있습니다.

질문: 동적 프레임 생성을 활성화했지만 여전히 성능이 저하되는 것을 볼 수 있습니다. 왜 그런가요?

A: 동적 프레임 생성은 시간 경과에 따라 CPU 및 GPU 성능을 샘플링하는 휴리스틱을 사용합니다. 이 샘플링 기간 내에 게임 엔진 성능이 크게 변하면 동적 프레임 생성이 순간 성능이 저하되는 상태로 전환될 수 있습니다.

반사

질문: Reflex에 GPU 하드웨어, 공급업체 또는 드라이버 버전 확인을 추가해야 하나요?

A: GPU 하드웨어, 공급업체 또는 드라이버 버전 확인을 하지 마세요. Reflex 플러그인이 지원되는 한, 항상 모든 기능 상수를 설정하고 Reflex Streamline 플러그인에 대한 기능 호출을 평가하세요.

- Reflex Streamline 플러그인은 이러한 모든 추상화를 처리합니다. 게임에 완전히 투명합니다.
- PCL 통계는 PC 지연 시간을 제대로 측정하기 위해 상수와 마커가 필요합니다. 모든 GPU 하드웨어, 공급업체 및 드라이버 버전에서 작동합니다.
- Reflex 저지연 모드는 모든 GPU 하드웨어, 공급업체 및 드라이버 버전에서 작동하는 것은 아니지만 Reflex Streamline 플러그인에 의해 이미 추상화되어 있습니다.
- 게임에서는 아래 설명된 대로 lowLatencyAvailable 설정을 사용하여 반사 지연 시간 UI를 비활성화/활성화하기만 하면 됩니다.
- 반사 지연 시간이 없어도 지연 시간을 측정하는 것이 중요합니다.

질문: Reflex 플러그인에 대한 GPU 하드웨어, 공급업체 또는 드라이버 버전을 확인할 수 없습니다.

NVIDIA가 아닌 HW에서도 동일한 Reflex 호출이 이루어집니다. PCL이 여전히 작동하지 않는 이유는 무엇인가요?

A: PCL 통계가 초기화되고 작동을 시작하려면 최소 1개의 기능 상수 세트 호출이 있어야 합니다.

- 軒비-NV GPU에서 Reflex UI가 없으면(낮은 지연 시간 사용 가능 == 거짓이므로) 게임에서 설정된 기능 상수를 전혀 호출하지 못할 수 있습니다. 이것이 누락된 것일 수 있습니다.

질문: 리플렉스 저지연 모드가 꺼짐으로 설정된 경우 리플렉스 마커 호출을 건너뛰어야 하나요? A:

리플렉스 로우 레이턴시 모드가 꺼짐으로 설정된 경우 리플렉스 호출을 건너뛰지 마세요.

- 마커는 지연 시간을 측정하는 데도 사용됩니다.
- 반사 지연 시간이 짧은 모드가 꺼진 상태에서도 지연 시간을 측정하는 것이 중요합니다.

질문: 호스트 시스템에서 UI를 비활성화/활성화하기 위해 Reflex 저지연 기능을 사용할 수 있는지 확인하려면 어떻게 해야 하나요?

A: 게임에서 낮은 지연 시간 사용 가능 여부를 확인하여 반사 지연 시간이 짧은 UI를 표시할지 여부를 결정해야 합니다.

- 낮은 지연시간 사용 가능은 UI에만 사용하세요. **Reflex Streamline** 플러그인이 지원되는 한, 게임은 **lowLatencyAvailable**이 **false**인 경우에도 다른 모든 Reflex 관련 작업을 동일한 방식으로 수행해야 합니다.

질문: 반사 수면이 프레임 속도에 부정적인 영향을 미칩니다. 무슨 문제인가요?

A: Reflex On은 FPS에 4% 이상의 영향을 미치지 않아야 합니다. 반사광 켜기 + 부스트는 7% 이상의 FPS에 영향을 미치지 않아야 합니다.

- 영향이 약간 더 큰 경우에는 프로듀서/CM에게 연락하여 버그를 제출하고 드라이버 팀에서 추가 조사를 할 수 있도록 하세요.
- 영향이 훨씬 더 크다면 계속 읽어주세요.

커짐 + 부스트에서만 재생되나요, 아니면 커짐과 함께 재생되나요? 커짐 + 부스트에서만 재생되는 경우 RTB0(렌더 스레드 바운드 최적화)로 인한 문제일 가능성이 높습니다.

- 또 다른 증상은 FPS가 30으로 제한된다는 것입니다.
- bUseMarkersToOptimize = false로 설정하여 문제가 해결되는지 확인해 보세요.
- 반사 수면이 증가함에 따라 렌더링 제출 시작 및 렌더링 제출 종료/현재 종료 마커 내의 시간이 증가하면 이러한 문제가 발생할 수 있습니다.
- 일반적으로 렌더링 제출을 개선하고 시작/끝 마커를 표시하면 이 문제를 피할 수 있습니다. 하지만 일부 게임은 RTB0와 호환되지 않을 수 있으며 bUseMarkersToOptimize = true로 설정할 수 없습니다.

FPS가 15로 제한되어 있나요? 그렇다면 Reflex 컨트롤러 문제일 가능성이 높으므로 드라이버 팀에서 조사해야 합니다.

게임에서 프레임당 한 번 이상 반사 절전을 호출하면 FPS에 큰 영향을 줄 수 있습니다. 반사 절전은 프레임당 한 번 이상 호출해서는 안 됩니다.

- 반사 절전 후 시뮬레이션이 시작되기 전 어느 시점에서 게임이 프레임을 처음부터 다시 시작하나요? 결정하면 이런 문제가 발생할 수 있습니다. 이 경우 반사 절전이 두 번 이상 호출되었을 수 있으므로 FPS에 큰 영향을 줄 수 있습니다.
- 반사 절전 모드가 없으면 프레임을 처음부터 다시 시작하고 입력 메시지와 다른 단계를 반복한 후 확인해도 문제가 없습니다. 하지만 반사 절전을 사용하면 프레임을 다시 시작하면 프레임을 호출하고 여러 번 절전 상태가 됩니다. 이것이 문제입니다.
- 이 경우 한 가지 가능한 해결책은 시뮬레이션 시작 전에 반사 절전을 두 번 이상 호출하지 않도록 하는 것입니다.

질문: Reflex 기능에 필요한 HW/SW 요구 사항은 무엇인가요? A: 다

음 표를 참조하세요:

기능	GPU IHV	GPU HW	드라이버 버전	지원 확인	키 설정 / 마커
----	---------	--------	---------	-------	-----------

리플렉스 낮은 지연 시간	NVIDIA Only	GeForce 900 시리즈 이상	456.38 또는 더 높은	sl::ReflexSettings::lowLatencyAvailable	sl::ReflexConstants::모드
자동 구성 반사 분석기	NVIDIA Only	GeForce 900 시리즈 이상	521.60 또는 더 높은	sl::ReflexSettings::플래시 인디케이터 드라이버 제어	sl::ReflexMarker::eReflexMarkerTriggerFlash
프레임 속도 제한기	NVIDIA Only	모두	모두	항상	sl::ReflexConstants::frameLimitUs
PC 지연 시간 통계	모두	모두	모두	항상	sl::ReflexMarker::eReflexMarkerPCLatencyPing

참고: 하위 기능은 상호 종속성 없이 서로 구별됩니다. 모든 것이 플러그인 내에서 추상화되어 있으며 애플리케이션에 투명하게 표시됩니다. 애플리케이션은 GPU HW 공급업체 및 드라이버 버전을 명시적으로 확인하지 않아야 합니다. 애플리케이션은 하위 기능 지원 및 활성화 여부에 관계없이 모든 작업을 동일하게 수행해야 합니다. 단 두 가지 예외는 Reflex UI와 Reflex Flash Indicator(RFI)입니다. 애플리케이션은 `sl::ReflexSettings::lowLatencyAvailable`에 따라 Reflex UI를 비활성화해야 합니다. 그리고 애플리케이션은 `sl::ReflexSettings::flashIndicatorDriverControlled`가 거짓일 때 `sl::ReflexMarker::eReflexMarkerTriggerFlash`를 보내지 않아야 합니다.

질문: Reflex가 활성화되어 있고 NVIDIA 하드웨어에서 작동하는지 확인하려면 어떻게 해야 하나요?

A: [NVIDIA Reflex SDK](#)에 있는 간단한 테스트를 실행하여 Reflex가 활성화되어 있고 제대로 작동하는지 빠르고 쉽게 확인할 수 있습니다.

1. "Reflex_Verification" 폴더로 이동합니다.
2. 명령 프롬프트(관리자 권한으로 실행)를 열고 FVSKSetup.exe를 실행합니다.
3. 명령 프롬프트(관리자 권한으로 실행)에서 ReflexTestSetup.bat을 실행합니다.
4. 게임 시작 - 게임 내 반사 확인 HUD에 반사 활성화 여부가 표시됩니다.
 - 반사 확인 HUD에는 드라이버 531.18 이상이 필요합니다.
 - 앱 프로필에 게임을 추가해야 할 수도 있습니다: NVIDIA 제어판 -> 3D 설정 -> 프로그램 설정 -> 사용자 지정할 프로그램 선택 -> 추가 -> 프로그램 선택 -> 선택한 프로그램 추가 -> 적용
5. 'ALT+T'를 눌러 테스트를 시작합니다(신호음이 두 번 울립니다).
6. 약 5분 후 테스트가 완료됩니다(신호음이 3번 울립니다).
7. 명령 프롬프트를 참조하여 "통과"를 확인합니다.

```
Reflex_Mode = Disabled  App_Called_Sleep = 1  Flash_Indicator = 4
Total_Render_Time      = 1449
Simulation_Interval    = 2003  Simulation_End_Time = 157
Render_Start_Time      = 158   Render_End_Time   = 1928
Present_Start_Time     = 1930  Present_End_Time  = 2104
Driver_Start_Time      = 1709  Driver_End_Time   = 2063
OsQueue_Start_Time     = 1714  OsQueue_End_Time  = 3513
Gpu_Start_Time         = 2053  Gpu_End_Time      = 3513
```

```
Administrator: Command Prompt - ReflexTest.exe curve
Trying 2003086 kHz... GPCCLK = 1575, DRAMCLK = 810, Util = 98.0, FPS = 15.8, PC Latency = 108.293
uring: Reflex ON, No Frame Gen...
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 15.9, PC Latency = 108.140
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 15.9, PC Latency = 104.768
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.0, PC Latency = 103.562
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.0, PC Latency = 102.733
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.0, PC Latency = 102.782
uring: Reflex OFF, No Frame Gen...
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.1, PC Latency = 143.385
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.1, PC Latency = 174.637
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.2, PC Latency = 202.342
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.0, PC Latency = 218.619
      GPCCLK = 2010, DRAMCLK = 810, Util = 99.0, FPS = 16.1, PC Latency = 215.291
.0 FPS: PCL 104.4 ms is PASS at 1.67 FT! -0.8% FPS impact. 45.3% latency reduction. {0.53 0.10 1.03 0.00}

Summary [64e3b0da] - Reflex ON, No Frame Gen, I>S S>Q Q>R R>D
.0 FPS: PCL 17.8 ms is PASS at 2.48 FT! 0.4% FPS impact. 28.1% latency reduction. {0.50 0.76 1.20 0.02}
.7 FPS: PCL 18.3 ms is PASS at 2.39 FT! 0.1% FPS impact. 31.6% latency reduction. {0.47 0.71 1.19 0.02}
.9 FPS: PCL 20.0 ms is PASS at 2.36 FT! -0.3% FPS impact. 32.8% latency reduction. {0.53 0.64 1.18 0.02}
.8 FPS: PCL 20.5 ms is PASS at 2.19 FT! -0.0% FPS impact. 37.1% latency reduction. {0.43 0.58 1.16 0.02}
.3 FPS: PCL 22.8 ms is PASS at 2.22 FT! -0.2% FPS impact. 36.8% latency reduction. {0.53 0.53 1.14 0.01}
.2 FPS: PCL 25.4 ms is PASS at 2.01 FT! -0.4% FPS impact. 42.6% latency reduction. {0.44 0.44 1.12 0.01}
.2 FPS: PCL 29.5 ms is PASS at 2.01 FT! 1.8% FPS impact. 42.7% latency reduction. {0.51 0.38 1.10 0.01}
.6 FPS: PCL 31.2 ms is PASS at 1.98 FT! -0.1% FPS impact. 43.7% latency reduction. {0.52 0.36 1.10 0.01}
.9 FPS: PCL 58.5 ms is PASS at 1.80 FT! -0.7% FPS impact. 46.5% latency reduction. {0.53 0.18 1.09 0.01}
.0 FPS: PCL 104.4 ms is PASS at 1.67 FT! -0.8% FPS impact. 45.3% latency reduction. {0.53 0.10 1.03 0.00}
cycle #1 has completed successfully.

o the game and press hot key [Alt-t] to start cycle #2. Come back and press [Ctrl-c] when all cycles are done.
```

질문: Reflex가 활성화되어 있고 NVIDIA가 아닌 하드웨어에서 작동하는지 확인하려면 어떻게 해야 하나요?

A: [NVIDIA Reflex SDK](#) 내에서 간단한 테스트를 실행하여 Reflex가 활성화되어 있고 제대로 작동하는지 빠르고 쉽게 확인할 수 있습니다.

1. "Reflex_Verification" 폴더로 이동합니다.
2. 명령 프롬프트(관리자 권한으로 실행)를 열고 PrintPCL.exe를 실행합니다.
3. 게임 시작
4. 테스트를 시작하려면 "ALT+T"를 누르세요.
5. 명령 프롬프트를 참조하여 PCL 값을 확인합니다. 값이 0.0이 아니라면 PCL이 작동하는 것입니다.
6. 테스트를 종료하려면 "Ctrl + C"를 누르세요.

질문: GPU 사용률이 0%로 보고되는 이유는 무엇인가요?

A: 이 문제는 그래픽 카드 드라이버 또는 FVSDK를 방금 업데이트한 후에 발생할 수 있습니다. 설치가 완료된 후 시스템을 다시 시작한 다음 테스트를 실행하세요.

2.2.1 새 항목 간소화 - 개요

DLSS 광선 재구성

- 새로운 파티클 레이어 API

DLSS 프레임 생성

- 새로운 디스토션 맵 API
- DLSS 프레임 생성을 지능적으로 활성화/비활성화하여 긍정적인 성능 확장을 보장하는 동적 프레임 생성 지원

기타

- 버그 수정 및 안정성 향상

(Streamline 2.2.0에 도입된 다음 기능 포함)

DLSS 광선 재구성

- DLSS 광선 재구성은 경로 추적 및 집약적인 광선 추적 콘텐츠의 이미지 품질을 개선하는 모든 GeForce RTX GPU 용 새로운 신경망입니다. 레이 재구성은 수작업으로 조정된 디노이저를 샘플링된 광선 사이에 더 높은 품질의 픽셀을 생성하는 NVIDIA 슈퍼컴퓨터로 훈련된 AI 네트워크로 대체합니다. 플러그인 "sl.dlss_d"

DLSS 프레임 생성을 위한 자동 장면 변경 감지

- 빠른 장면 전환 및 컷 중에 잘못된 프레임을 삭제하고 올바른 프레임만 남겨 일부 사용자가 발견한 아티팩트의 찰나적 현상을 제거합니다. 자세한 내용은 '프로그래밍 가이드DLSS_G.md'의 섹션 21.0 "자동 장면 변경 감지"를 참조하세요.

기타

- sl.nis 플러그인에 다중 뷰포트 지원 추가
- 별칸 서브리소스 범위 정보를 지정하기 위한 새로운 구조체 'SubresourceRange'를 추가했습니다.
- SL 기능 ID 정의를 'sl.h'의 중앙 위치로 이동했습니다.
- 비교 연산자에 'sl::StructType' 자손 클래스의 처음 8 바이트만 사용되던 버그를 수정했습니다.

(Streamline 2.1.0에 도입된 다음 기능 포함)

OTA 업데이트

- 이제 무선 업데이트를 통해 플러그인을 업데이트할 수 있습니다.

DLAA

- 이제 DLSS 슈퍼 해상도 옵션으로 DLAA가 지원됩니다.

DLSS 프레임 생성

- 다양한 성능, 이미지 품질 및 메모리 최적화 (**Streamline 2.0.0에 도입된**)

다음 기능 포함 기능 ID

- 더 이상 열거형이 아닌 상수 표현식 부호 없는 정수로 제공됩니다.
- 핵심 기능 ID는 `sl.h`에서 선언되고 정의되며 특정 기능 ID는 이제 각각의 헤더 파일에 있습니다.

열거형 이름 지정

- 모든 열거형은 열거형 클래스 `Name::eValue` 형식으로 변환되었습니다.

오류 보고

- 이제 모든 SL 함수가 `sl::Result`를 반환합니다(새 헤더 `sl_result.h`).
- 여전히 오류 로깅 콜백을 모니터링하여 가능한 모든 오류를 적시에 포착해야 하지만 가장 일반적인 오류의 처리가 개선됩니다.
- 헬퍼 매크로 `SL_FAILED`를 도입했습니다.
- `sl::결과`를 문자열로 변환하는 헬퍼 메서드가 추가되었습니다.

버전 관리

- 새로운 API `slGetFeatureVersion`은 SL 및 NGX(있는 경우) 버전을 모두 반환합니다.

요구 사항 보고

- 이브에 항상 편리하지 않기 때문에 JSON을 반환하던 `slGetFeatureConfiguration`을 제거했습니다.
- OS, 드라이버, HWS 렌더링 API, VK 확장 및 기타 요구 사항에 대한 정보를 제공하기 위해 새로운 API `slGetFeatureRequirements`가 추가되었습니다.
- 새로운 `sl_helpers_vk.h` 헤더에 `getVkPhysicalDeviceVulkan12Features` 및 `getVkPhysicalDeviceVulkan13Features` 헬퍼 함수를 추가했습니다.

상수 대 설정

- 일반 슬렛피처콘스트 및 슬렛피처설정 API가 제거되고 새로운 API 슬렛피처평선이 추가되었습니다.
- 각 SL 기능은 기능별 작업을 수행하는 데 사용되는 함수 집합을 내보냅니다.
- 새로운 헬퍼 매크로 `SL_FEATURE_FUN_IMPORT` 및 관련 `sl_$feature.h` 헤더의 헬퍼 함수.
- 각 기능별 헤더에 도우미 기능이 추가되어 가져오기를 쉽게 할 수 있습니다.

기능이 지원되고 어댑터 비트 마스크

- `slIsFeatureSupported`는 `DXGIAdapterDesc` 또는 VK 물리적 장치에서 쉽게 구할 수 있는 어댑터 LUID를 사용하도록 수정되었습니다.
- 엔진은 이미 어댑터를 열거하고 있으므로 기존 코드와 잘 맞을 것입니다.
- VK를 사용할 때 호스트는 필요한 경우 LUID 대신 `VkPhysicalDevice`를 제공할 수 있습니다.

진정한 FPS

- 실제 프레임 시간 및 시간 간 프레젠테이션을 제거하고 정수 값인 `numFramesActuallyPresented`로 대체했습니다.

DLSS-FG에 대한 향상된 뷰포트 지원

- 호스트는 항상 뷰포트 0을 강제로 지정하는 대신 `slDLSSetOptions`를 호출할 때 임의의 뷰포트 ID를 지정할 수 있습니다.

"게임 프레임 렌더링 안 함" 플래그

- 중복되므로 완전히 제거되었습니다.
- 이제 호스트는 `slDLSSGSetOptions`를 사용하여 DLSS-FG를 켜거나 꺼야 합니다.

버퍼 복사 및 태그 지정

- 명령 목록과 리소스 수명 주기 정보를 포함하도록 `slSetTag` API가 확장되었습니다.
- 필요한 경우 리소스가 SL에 의해 내부적으로 자동으로 복사됩니다.
- 슬레벨레이트피처를 사용하여 리소스에 로컬로 태그를 지정할 수 있습니다(태그는 특정 평가 호출에만 유효합니다. 자세한 내용은 프로그래밍 가이드 참조).

프레임 ID

- 호스트가 다음 프레임의 프레임 핸들을 가져올 수 있는 새로운 API `slGetNewFrameToken`이 추가되었습니다.
- 그런 다음 모든 SL 호출에 동일한 핸들이 전달됩니다.
- 호스트가 프레임 카운팅을 완전히 제어하려는 경우 프레임 인덱스를 입력으로 제공할 수 있습니다.

여러 장치

- 새로운 API인 `slSetD3DDevice`와 `slSetVulkanInfo`를 사용하여 SL이 사용해야 할 디바이스를 지정할 수 있습니다.

네이티브 인터페이스 확보

- 타사 라이브러리(NVAPI 포함)를 사용하는 경우 SL 프록시를 입력으로 전달하는 것은 바람직하지 않습니다.
- 필요에 따라 네이티브 인터페이스에 액세스할 수 있는 새로운 API `slGetNativeInterface`가 추가되었습니다.

수동 후킹

- `slGetHook*` API 제거

- 이제 **슬루업그레이드인터페이스**와 결합된 새로운 API `slGetNativeInterface`를 사용하여 수동 후킹을 대폭 간소화할 수 있습니다.

(Streamline 1.5.x에 도입된 다음 기능 포함)

리소스 상태 추적

- SL은 더 이상 리소스 상태를 추적하지 않으며, 호스트는 리소스에 태그를 지정할 때 올바른 상태를 제공할 책임이 있습니다.

명령 목록 상태 추적

- SL은 기본적으로 모든 명령 목록(CL) 추적을 비활성화하며, 호스트는 각 `slEvaluateFeature` 호출 후 CL 상태를 복원할 책임이 있습니다.

인터포저 비활성화

- SL은 지원되는 플러그인 하나 이상에서 명시적으로 요청하지 않는 한 인터포저(DXGI/D3D 프록시)를 비활성화합니다.

SL DLL 검증

- SL은 이제 `sl_security.h` 헤더를 기본 SDK의 일부로 포함합니다(더 이상 SL 샘플 SDK를 다운로드할 필요가 없음).

OS 버전 감지

- 다른/부정확한 버전을 반환하는 다양한 MS API를 사용하는 대신 `kernel32.dll` 제품 설명에서 올바른 버전을 가져오도록 전환했습니다.
- 플래그 `sl::sl_flags::eBypassOSVersion`을 `sl::환경설정`에 체크하면 OS 감지를 오프아웃할 수 있습니다(기본값은 꺼져 있으며 사용을 권장하지 않음).

OTA 옵트인

- OTA 프로그램(SL 및 NGX)에서 자동으로 옵트인할 수 있도록 `PreferenceFlags::eAllowOTA`를 포함하도록 `sl::Preferences`를 확장했습니다. 이 플래그는 기본적으로 설정되어 있습니다.

DLSS 사전 설정

- 호스트는 `sl::DLSSOptions`를 수정하여 DLSS의 DL 네트워크(사전 설정)를 변경할 수 있습니다.