



NVIDIA Streamline 2.10

Developer Guide

Version 1.0
December 2025

Streamline 2.10 New Items - Overview

DLSS Super Resolution

- Introduces new Render Presets M (default for Performance mode), L (default for UltraPerformance Mode).
 - Preset F which is currently marked for deprecation will be removed in the next SDK.

Key Steps to a Successful DLSS Integration

1. **Integrate Streamline SDK** (without any features) and focus on manual hooking, resource state tracking, etc. - **1 day**
2. **Check for the official NVIDIA & Streamline dual signatures** on sl.interposer.dll before loading the DLL - **30 mins**
3. **Check for system (hardwares/software) support for each of the DLSS 3 features** - and show appropriate error messages to end users based on reported hw support - **1 hour**
4. **Integrate DLSS Super Resolution via Streamline** - pass in the necessary input resources and set up the upscaling pipeline (before all post-processing) - **2 days**
5. **Validate IQ and performance benefits** from DLSS Super Resolution - **1 day**
6. **Integrate DLSS Ray Reconstruction** via Streamline - **4 days**
7. **Validate IQ and performance** benefits from DLSS Ray Reconstruction - **1 day**
8. **Integrate Reflex/PCL via Streamline** - set up the appropriate markers, sleep call, etc. - **1 day**
9. **Validate that input latency gets reduced**, using FrameView SDK or GFE in game overlay or the Reflex validation HUD - **4 hours**
10. **Integrate DLSS Frame Generation** - pass in the appropriate constants, camera matrices, and input resources in addition to the ones marked for DLSS Super Resolution (e.g. hudless and ui color/alpha). Be sure to also disable DLSS Frame Generation when appropriate, such as when in-menu or scene transitions - **2 days**
11. **Integrate Reflex Frame Warp** - pass in the most up-to-date camera matrices during simulation. Pass a mask for pixels that are camera-dependent. Integrate predictive rendering by modifying the rendering pipeline to accept a predicted camera matrix during the same time as submitting the most up-to-date camera matrices. Ensure that camera dependent objects that are rendered in world space are offset by this predicted camera matrix - **2 days**
12. **Validate that inputs are correct** using Streamline ImGui plugin and buffer visualization using the development DLLs - **1 day**

Total Estimated Time: ~15 days

DLSS Super Resolution

Detailed information can be found in the [DLSS Programming Guide](#)

Basic Integration Checklist

	Item	
<input type="checkbox"/>	Game-specific Application ID is used during initialization	
<input type="checkbox"/>	Make sure that DLSS Super Resolution is integrated as close to the start of post-processing as possible (and not after)	
<input type="checkbox"/>	Default Presets are set when the Application does not explicitly choose a Preset corresponding to a Performance Mode. ePresetM is the default model for Performance Mode. ePresetL is the default model for UltraPerformance Mode. ePreset K is the default for all other Modes.	
<input type="checkbox"/>	Mip-map bias set when DLSS SR is enabled (<i>without this, textures will look blurry / smudgy / low-resolution</i>)	
<input type="checkbox"/>	Motion vectors for all scenes, materials and objects are accurate. If you are using Hi-Res Motion Vectors, please ensure they are rendered to target resolution and not dilated.	
<input type="checkbox"/>	Static scenes resolve and compatible jitter confirmed (<i>Incorrect / missing jitter can lead to poor anti-aliasing and increased flickering</i>)	
<input type="checkbox"/>	Exposure value is properly sent each frame (or auto-exposure is enabled)	
<input type="checkbox"/>	DLSS modes are queried and user selectable in the UI and/or dynamic resolution support is active and tested.	
<input type="checkbox"/>	Full production non-watermarked DLSS library (nvngx_dlss.dll) is packaged in the release build.	
<input type="checkbox"/>	Pass in Camera Reset Flag on scene changes, view changes (first person to third person), or during camera jumps in cutscenes.	
<input type="checkbox"/>	Perform NGX cleanup / shutdown procedures when DLSS is no longer needed. (<i>otherwise, you'll leak resources/memory</i>)	
<input type="checkbox"/>	DLSS Sharpness is deprecated. The recommended sharpness integration is Nvidia Image Scaling (NIS).	

IMPORTANT: *DLSS should only replace the primary upscale pass on the main render target and should not be used on secondary buffers like shadows, reflections etc.*

DLSS Ray Reconstruction

Detailed information can be found in the [DLSS RR Programming Guide](#)

Basic Integration Checklist

	Item	Reference
<input type="checkbox"/>	Game-specific Application ID is used during initialization	
<input type="checkbox"/>	DLSS-RR is an addition to DLSS Super Resolution and uses the same Perf Quality modes that have been set for DLSS Super Resolution. So make sure DLSS Super Resolution is integrated first.	
<input type="checkbox"/>	Make sure that both DLSS Ray Reconstruction and DLSS Super Resolution are integrated at the start (or as close to the start) of post-processing as possible (and not after)	
<input type="checkbox"/>	Make sure other denoisers in your pipeline such as NRD are completely disabled .	
<input type="checkbox"/>	Please set your DLSS RR Preset (DLSSDPreset) to eDefault or ePresetD or ePresetE (D being the default). Depth of Field Guide buffer is only compatible with ePresetE. ePresetA, ePresetB, ePresetC are removed.	Section 3.12
<input type="checkbox"/>	Mip-map bias set when DLSS RR is enabled (<i>without this, textures will look blurry / smudgy / low-resolution</i>)	
<input type="checkbox"/>	Ensure required buffers are provided	Section 4.1
<input type="checkbox"/>	For Ray Tracing / Path Tracing, DLSS RR assumes independent samples and requires that sampling used to generate Inputs must have minimal correlation both spatially and temporally	
<input type="checkbox"/>	For DLSS-RRsub pixel jitter, there is no reason to limit the number of samples. Use of many more jitter positions (at least 32) is highly recommended.	
<input type="checkbox"/>	DLSS RR Requires Linear Depth to be provided which is different from those provided to DLSS Super Resolution and DLSS Frame Generation <ul style="list-style-type: none">○ Please use kBufferTypeLinearDepth specifically provided for this○ Please set the Inverted Depth bit if the Depth buffer provided has inverted z ordering	
<input type="checkbox"/>	Motion vectors / Specular Motion Vectors for all scenes, materials and objects are accurate	

<input type="checkbox"/>	Static scenes resolve and compatible jitter confirmed <i>(Incorrect / missing jitter can lead to poor anti-aliasing and increased flickering)</i>	
<input type="checkbox"/>	DLSS modes are queried and user selectable in the UI and/or dynamic resolution support is active and tested.	
<input type="checkbox"/>	A DLSS RR enable / disable toggle is available	
<input type="checkbox"/>	Full production non-watermarked DLSS Ray Reconstruction library (nvngx_dlssd.dll) is packaged in the release build	
<input type="checkbox"/>	Pass in Camera Reset Flag on scene changes, view changes (first person to third person), or during camera jumps in cutscenes.	
<input type="checkbox"/>	Perform NGX cleanup / shutdown procedures when DLSS is no longer needed. <i>(otherwise, you'll leak resources/memory)</i>	

DLSS Frame Generation

Detailed information can be found in the [DLSS FG Programming Guide](#)

Basic Integration Checklist

	Item	Reference
<input type="checkbox"/>	All the required inputs are passed to Streamline: depth buffers, motion vectors, HUD-less color buffers, UI color buffer	Section 5.0
<input type="checkbox"/>	Common constants and frame index are provided for each frame using <code>slSetConstants</code> and <code>slSetFeatureConstants</code> methods	Section 7.0
<input type="checkbox"/>	All tagged buffers are valid at frame present time, and they are not reused for other purposes	Section 5.0
<input type="checkbox"/>	Buffers to be tagged with unique id 0	Section 5.0
<input type="checkbox"/>	Make sure that frame index provided with the common constants is matching the presented frame	Section 8.0
<input type="checkbox"/>	For multi-frame generation, remember to set <code>numFramesToGenerate</code> should be set 1 less than the multiplier mode. For example: set it to 1 for 2x, 2 for 3x and 3 for 4x	Section 6.1
<input type="checkbox"/>	Inputs are passed into Streamline look correct as well as camera matrices and dynamic objects	SL ImGUI guide
<input type="checkbox"/>	Application checks the signature of <code>sl.interposer.dll</code> to make sure it is a genuine NVIDIA library	Streamline Programming Guide - 2.1.1
<input type="checkbox"/>	Requirements for Dynamic Resolution are met (if the game supports Dynamic Resolution)	Section 10.0
<input type="checkbox"/>	Disable DLSS Frame Generation (by setting <code>sl::DLSSGOptions::mode</code> to <code>sl::DLSSGMode::off</code>) when the game is paused, loading, in-menu, modifying resolution, switching between full-screen/windowed mode, and in general NOT rendering game frames.	Section 12.0
<input type="checkbox"/>	Swap chain is recreated every time DLSS Frame Generation is turned on or off (by changing <code>sl::DLSSGOptions::mode</code>) to avoid unnecessary performance overhead when DLSS Frame Generation is switched off	Section 18.0
<input type="checkbox"/>	Reduce the amount of motion blur; when DLSS Frame Generation enabled, halve the distance / magnitude of motion blur	

<input type="checkbox"/>	Reflex is properly integrated (see checklist in Reflex Programming Guide)	Section 8.0
<input type="checkbox"/>	In-game UI for enabling/disabling DLSS Frame Generation is implemented and follows RTX UI Guidelines	RTX UI Guidelines
<input type="checkbox"/>	Only full production non-watermarked libraries are packaged in the release build	
<input type="checkbox"/>	No errors or unexpected warnings in Streamline and DLSS Frame Generation log files while running the feature	
<input type="checkbox"/>	Ensure extent resolution or resource size, whichever is in use, for Hudless and UI Color and Alpha buffers exactly match that of backbuffer.	
<input type="checkbox"/>	Hook up the VSync setting in the menu to sl::DLSSGState::blsVsyncSupportAvailable	

Reflex Low Latency & PCL / LWL Stats

Detailed information can be found in the [Reflex and Reflex 2 Programming Guides](#)

Basic Integration Checklist

	Item	Reference
<input type="checkbox"/>	Reflex Low Latency default state is “On”	
<input type="checkbox"/>	All Reflex modes (Off, On, On + Boost) function correctly	
<input type="checkbox"/>	PC Latency (PCL) in the Reflex Test Utility is higher than zero <ul style="list-style-type: none">Zero means that some of the markers were not integrated correctlyYou can visualize marker placement in NSight Systems to help aid in debugging, including seeing the sim marker moving to JIT when Reflex is enabled vs. disabled.	
<input type="checkbox"/>	Reflex Test Utility Report has passed without warning with DLSS Frame Generation enabled <ul style="list-style-type: none">Indicates that Reflex worked correctly in On / Low Latency mode. Your input latency should have gone down by at least 20%	
<input type="checkbox"/>	Reflex Test Utility Report has passed without warning with Reflex set to “On”	
<input type="checkbox"/>	Reflex does not significantly impact average framerate (fps) more than 4% when Reflex is enabled <ul style="list-style-type: none">Reflex set to “On + Boost” is expected to have a slightly larger performance impact in order to provide the absolute lowest latency	
<input type="checkbox"/>	Reflex Test Utility Report has passed without warning with Reflex set to “On + Boost”	
<input type="checkbox"/>	PCL Markers are always sent regardless of Reflex Low Latency mode state	
<input type="checkbox"/>	Reflex Flash Indicator appears when left mouse button is pressed	
<input type="checkbox"/>	Reflex UI settings are following the UI Guidelines	
<input type="checkbox"/>	Keybinding menus work properly (no F13)	
<input type="checkbox"/>	PC Latency (PCL) is higher than zero on non-NVIDIA GPUs	

<input type="checkbox"/>	Reflex UI settings are disabled or not available on non-NVIDIA GPUs	
--------------------------	---	--

Reflex Integration Checklist Steps

1. Download [Reflex Verification Tools](#)
2. Install FrameView SDK
 - Double click the FrameView SDK Installer (**FVSDKSetup.exe**)
 - Restart system
3. Run **ReflexTestSetup.bat** from an administrator mode command prompt
 - This will force the Reflex Flash Indicator to enable, enable the Verification HUD, set up the Reflex Test framework, and start ReflexTest.exe.
4. Check Reflex Low Latency modes
 - Run game
 - Make sure game is running in fullscreen exclusive
 - Make sure VSYNC is **disabled**
 - Make sure MSHybrid mode is not enabled
 - Ensure Reflex Low Latency mode is set to “On” in game GUI and in the Verification HUD
 - Use the Reset/Default button in UI if it exists
 - Cycle through the Reflex modes in UI and confirm it matches with “Reflex Mode” within the Verification HUD
5. Running Reflex Tests
 - For titles supporting DLSS Frame Generation, **enable** DLSS-FG
 - Press **Alt + T** in game to start the test (2 beeps)
 - Analyze results after the test is done (3 beeps)
 1. Test should take approximately 5 minutes
 2. Check for warnings in ReflexTest.exe output
 - **Disable** DLSS-FG (if available) and set Reflex to “**On**”
 - Press **Alt + T** in game to start the test (2 beeps)
 - Analyze results after the test is done (3 beeps)
 1. Test should take approximately 5 minutes
 2. Check for warnings in ReflexTest.exe output
 - Set Reflex to “**On + Boost**”
 - Press **Alt + T** in game to start the test (2 beeps)
 - Analyze results after the test is done (3 beeps)
 1. Test should take approximately 5 minutes
 2. Check for warnings in ReflexTest.exe output
6. Test that markers are always sent even when Reflex is Off
 - Set Reflex to “**Off**”
 - Look at the Verification HUD and ensure the markers are still updating
7. Test the Reflex Flash Indicator
 - Verify the Reflex Flash Indicator is showing
 - Notice the gray square that flashes when the left mouse button is pressed
 - Verify that the Flash Indicator in the Verification HUD increments by 1 when the left mouse button is pressed

8. Check UI
 - Verify UI follows the Guidelines
9. Check keybinding
 - Run **capturePclEtw.bat** in administrator mode from command prompt
 - Go back to the game. Start gameplay
 - Check the Keybinding menu to make sure F13 is not being automatically applied when selecting a key
 - Go back to the command prompt and press any key to exit the bat
10. Run **ReflexTestCleanUp.bat** in administrator mode from command prompt
 - This disables the Reflex Flash Indicator and the Reflex Test framework
11. Test on other IHV (if available)
 - Install other IHV hardware
 - Install FrameView SDK and restart the system
 - Run **PrintPCL.exe** in administrator mode command prompt
 - Run game
 - Press **Alt + t** in game
 - Look at the PCL value in the command prompt. If the value is not 0.0, then PCL is working
 - Press **Ctrl + c** in the command prompt to exit **PrintPCL.exe**
 - Check to make sure Reflex UI is not available
12. Send NVIDIA Reflex Test report and Checklist results
 - Email results to NVIDIA alias: reflex-sdk-support@nvidia.com

Reflex Frame Warp & LWL Stats

Detailed information can be found in the [SL PCL Stats and Reflex 2 Programming Guides](#)

Basic Integration Checklist

	Item	Reference
<input type="checkbox"/>	Reflex Low Latency default state is “On”	
<input type="checkbox"/>	Ensure that additional markers are set on top of Reflex markers (camera constructed)	
<input type="checkbox"/>	Ensure camera matrices are properly passed	
<input type="checkbox"/>	Ensure the NoWarp mask is properly passed and contains only objects which should not be reprojected	
<input type="checkbox"/>	Ensure Predictive Rendering is active	
<input type="checkbox"/>	Ensure Frame Warp and Predictive Rendering are off during scene transitions, cutscenes, and in menus	
<input type="checkbox"/>	Ensure Camera-Dependent object correction is active with predictive rendering	
<input type="checkbox"/>	Ensure PCL Markers are always sent regardless of Reflex Low Latency mode state	
<input type="checkbox"/>	Reflex UI settings are following the UI Guidelines	
<input type="checkbox"/>	Keybinding menus work properly (no F13)	
<input type="checkbox"/>	PC Latency (PCL) is higher than zero on non-NVIDIA GPUs	
<input type="checkbox"/>	Perceived Latency (LWL) is lower than PCL for when feature is turned off	
<input type="checkbox"/>	Reflex UI settings are disabled or not available on non-NVIDIA GPUs	

FAQ

Streamline

Q: What do I need to do to ensure third party overlays (STEAM, Epic Game Store, etc) work correctly with Streamline?

A: The following rules should be followed in order for 3rd party overlays to work correctly with Streamline:

- Overlays in general **must not make assumptions about swap-chain and command queues**, when DLSS Frame Generation is active there could be multiple command queues and multiple asynchronous presents.
- Overlays should intercept `IDXGIFactory::CreateSwapChainXXX` to obtain the correct swap-chain and command queue used to present frames.
- If integrated in engine, overlays should initialize **before** `slInit` is called
- There is an optional flag `sl::PreferenceFlags::eUseDXGIFactoryProxy` which can be used to avoid injecting SL hooks in DXGI factory v-table, this might help resolve some issues with overlays.

Q: What do I do if the D3D debug layer is complaining about incorrect resource states?

A: Please provide correct state when tagging Streamline resources

Q: What do I do if I get a crash in Swapchain::Present or some similar unexpected behavior?

A: Please double check that you are **NOT** linking `dxgi.lib/d3d12.lib` together with the `sl.interposer.dll`. If this is not an issue, please view the Streamline logs for further guidance on what the cause could be.

Q: Should I enable Streamline on non-NVIDIA hardware?

A: Yes. While Streamline is an SDK developed by NV, it supports multiple features that work across all hardware vendors (for example: NVIDIA Image Sharpening (NIS), NVIDIA Real-Time Denoiser (NRD), Reflex (PCL))

Q: Streamline 2.1+ introduces OTA functionality for Streamline. What sort of things can be updated OTA and how is this different from earlier DLSS OTA?

A: Streamline 2.1+ expands over-the-air (OTA) functionality for all Streamline plugins. Previously this OTA was limited to DLSS updates only. . Significant improvements have been made, including performance, smoothness, and memory optimizations as well as bug fixes and security enhancements. Streamline OTA is supported on NVIDIA GPUs.

DLSS Super Resolution

Q: DLSS Super Resolution output does not look correct. What went wrong?

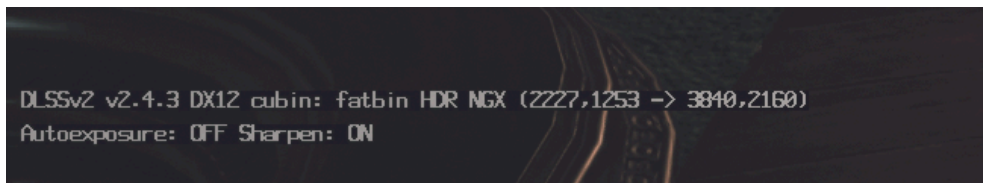
A: If the DLSS output does not look right (*ie: instability when moving the camera*) please check the following items. Also, review see [section 8](#) of the DLSS Super Resolution Programming Guide for detailed troubleshooting information:

- If your motion vectors are in pixel space then scaling factors `sl::Constants::mvecScale` should be {1 / render width, 1 / render height}
- If your motion vectors are in normalized -1,1 space then scaling factors `sl::Constants::mvecScale` should be {1, 1}
- Make sure that jitter offset values are in pixel space
- `NVSDK NGX_Parameter_FreeMemOnRelease` is replaced with `slFreeResources`
- `NVSDK NGX_DLSS_Feature_Flags_MVLowRes` is handled automatically based on tagged motion vector buffer's size and extent.

Q: How can I verify that DLSS Super Resolution is enabled?

A: You can enable the DLSS on-screen indicator utility by running “`ngx_driver_onscreenindicator.reg`” before starting your application. Once DLSS Super Resolution is enabled, you’ll see confirmation on the overlay in-game.

Once finished, you can run “`ngx_driver_onscreenindicator_off.reg`” to disable the indicator.



Q: What happened to DLSS Sharpening?

A: DLSS sharpening and softening provided in previous versions of the SDK are now deprecated. All developers are encouraged to use the Image Scaling SDK (NIS) via the DLSS SDK or directly on GitHub.

DLSS Ray Reconstruction

Q: What is DLSS Ray Reconstruction?

A: DLSS Ray Reconstruction is a new neural network for all GeForce RTX GPUs that improves the image quality of path-traced and intensive ray-traced content. Ray Reconstruction replaces hand-tuned denoisers with an NVIDIA supercomputer-trained AI network that generates higher-quality pixels in between sampled rays.

Q: I want to integrate DLSS Ray Reconstruction into my title. Is everything I need included within the Streamline SDK?

A: DLSS Ray Reconstruction functionality is limited to a NDA Streamline package, **and is not supported in the public SDK on Github**. Please contact your NVIDIA account manager for early access, or sign up to be notified when its publicly available here: <https://developer.nvidia.com/rtx/dlss/notify-me>

Q: What GeForce GPUs support DLSS Ray Reconstruction?

A: Any GeForce RTX GPU can support DLSS Ray Reconstruction as they feature the required Tensor Cores. This hardware requirement is the same as with DLSS Super Resolution.

Q: Can I enable DLSS Ray Reconstruction without enabling any other DLSS technologies?

A: DLSS Ray Reconstruction should not be enabled unless DLSS Super Resolution is enabled.

Q: Will I see a performance benefit when I enable DLSS Ray Reconstruction?

A: Game content with multiple ray traced effects, or full ray tracing, typically has several denoisers tuned for each specific type of lighting. DLSS Ray Reconstruction replaces these multiple denoisers with a single neural network, so it can run faster in heavily ray traced scenes. However, the performance benefits will vary according to the scene, settings, and GPU being used.

Q: Should Ray Reconstruction have a separate setting in game menus?

A: Please consult the RTX UI Guidelines for details.

Q: What should I do with any other denoiser I might have already integrated?

A: Please ensure that when DLSS RR is enabled, any other Denoiser (such as NRD) is completely disabled.

Q: Does Ray Reconstruction need to be integrated at multiple places in the pipeline with specialized passes?

A: Similar to DLSS Super Sampling, Ray Reconstruction is designed to be a single point of integration

Q: What kind of Ray Tracing / Path Tracing models is Ray Reconstruction compatible with?

A: DLSS Ray Reconstruction is designed to be compatible with all Path Tracing models and Ray Tracing Models (Heavier RT is preferred).

Q: Which Operating Systems is DLSS Ray Reconstruction available on?

A: DLSS Ray Reconstruction is supported on Windows, Linux and Proton (Proton Experimental is currently required to enable DLSS Ray Reconstruction).

Q: Does DLSS Ray Reconstruction work with all DLSS Super Resolution modes?

A: Yes - All DLSS Super Resolution modes are supported.

DLSS Frame Generation

Q: What are the HW/SW requirements for enabling DLSS Frame Generation?

A: The following criteria must be met to enable DLSS Frame Generation:

- GeForce RTX 40 Series GPU (Desktop or Notebook) or above
- Minimum Windows OS version of Win10 20H1 (version 2004, build 19041 or higher)
- Windows Hardware-accelerated GPU Scheduling (HWS) must be enabled via Settings : System : Display : Graphics : Change default graphics settings.

The following criteria must be met to enable DLSS Multi-Frame Generation:

- **GeForce RTX 50 Series** GPU (Desktop or Notebook)
- Minimum Windows OS version of Win10 20H1 (version 2004, build 19041 or higher)
- Windows Hardware-accelerated GPU Scheduling (HWS) must be enabled via Settings : System : Display : Graphics : Change default graphics settings.

Q: How do I confirm the requirements are met on a system?

A: In order for DLSS-FG to work correctly certain requirements regarding the OS, driver and other settings on a user's machine must be met. To obtain DLSS-FG configuration and check if all requirements are met you can use the code snippets listed in [Section 3.0 of the DLSS FG Programming Guide](#).

Q: What happens if I don't tag resources correctly?

A: If validity of tagged resources cannot be guaranteed (for example game is loading, paused, in menu, playing a video cut scene etc.) all tags should be set to null pointers to avoid stability or IQ issues.

For more information, please reference [Section 5.0 of the DLSS FG Programming Guide](#)

Q: Why am I getting random crashes when my game is loading, or if I'm in a menu screen or playing a cut scene, etc.?

A: If validity of tagged resources cannot be guaranteed (for example game is loading, paused, in menu, playing a video cut scene etc.) all tags should be set to null pointers to avoid stability or IQ issues.

For more information, please reference [Section 5.0 of the DLSS FG Programming Guide](#)

Q: Is DLSS Frame Generation compatible with HDR?

A: Yes, DLSS Frame Generation is compatible with HDR. However, DLSS Frame Generation currently does **NOT** support FP16 pixel format and scRGB color space.

Q: When I enable DLSS Frame Generation, my performance drops. Why is that?

A: DLSS Frame Generation has a finite cost to run. It provides an fps boost if the time taken to generate a frame is less than the time taken to render the same frame through the game engine.

However, if the game renders at extremely high framerates, then DLSS FG might actually slow down the game's overall framerate. FG's Auto mode (`sl::DLSSGMode::eAuto`) is designed to help with this scenario.

Q: I've integrated DLSS Frame Generation, but don't see any change in my performance when it's enabled. What could be the reason?

A: DLSS Frame Generation adds additional calls to `Present()`, which the game engine may not track by default.

In-game fps numbers, as a result may not match with numbers reported from outside the app process via tools like FrameView, Geforce Experience, CapFrameX, Windows Game Bar, etc.

The Streamline API provides information about the number of frames presented through `DLSSGState::numFramesActuallyPresented`, which should be hooked up to the engine's fps reporting function. This will align with fps reported by external tools as well as the game engine itself.

Q: How can I verify that my camera data and motion vectors are aligned?

A: Developers can also leverage the `sl.imgui.dll` overlay which was introduced with Streamline 2.0.1.

1. Confirm DLSS FG is running
2. Confirm DLSS Frame Generation is running
3. Press Debug key (CTRL+SHIFT+INS)
4. Enter "Dynamic Objects" view (cycle through views with CTRL+SHIFT+END)
5. Confirm functionality on screen

Q: How can I verify that DLSS Multi-Frame Generation is supported?

A: Multi-frame support is reported via `sl::DLSSGState::numFramesToGenerateMax`. Before enabling multi-frame, check for device support by calling `slDLSSGGetState` and checking `numFramesToGenerateMax`. If the value is 1, multi-frame is not supported. Otherwise, multi-frame is supported, up to the number of frames specified.

Q: How can I estimate the amount of VRAM required by DLSS Frame Generation?

A: SL can return a general estimate of the GPU memory required by DLSS-G via `slDLSSGGetState`. This can be queried before DLSS-G is enabled, and can be queried for resolutions and formats other than those currently active. To receive an estimate of GPU memory required, the application must:

- Set the `sl::DLSSGOptions::flags` flag, `DLSSGFlags::eRequestVRAMEstimate`
- Provide the values in the `sl::DLSSGOptions` structure include the intended resolutions of the MVecs, Depth buffer, final color buffer (UI buffers are assumed to be the same size as the color buffer), as well as the 3D API-specific format enums for each buffer. Finally, the expected number of backbuffers in the swapchain must be specified. See the `sl::DLSSGOptions` struct for details.

Q: If I enable VSYNC when DLSS Frame Generation is enabled, the game stutters. Why is that?

A: If you're using Streamline 2.0.1 or newer, you have DLSS Frame Generation support for V-SYNC (set *within the NVIDIA Control Panel*) without any requirement of having a G-SYNC compatible display. In-game V-SYNC setting support is currently not available. The application should check `sl::DLSSGState::blsVsyncSupportAvailable` and to determine whether the VSync option should stay

disabled or not (for example: a future Streamline over-the-air update could potentially deliver this support).

V-SYNC (NV Control Panel)	G-SYNC (NV Control Panel)	Result	Required Streamline Version
Off	Off	Max FPS, Low Latency, Tearing	1.3.3+
Off	On	Max FPS, Low Latency, Tearing above Refresh Rate	1.3.3+
On	Off	No Tearing, Not Smooth, High Latency	< 2.0.1
On	Off	No Tearing, Smooth, Lower Latency	2.0.1+
On	On	No Tearing, Smooth, Lowest Latency VSYNC ON option	1.3.3+

Starting with driver version 526.98 and higher, DLSS Frame Generation is compatible with VSYNC on G-SYNC and G-SYNC Compatible monitors and TVs. To enable this feature:

- Enable G-SYNC: NVIDIA Control Panel --> Display --> Setup G-SYNC
- Turn VSYNC On: NVIDIA Control Panel --> 3D Settings --> Manage 3D Settings

Note: DLSS Frame Generation is currently **not** compatible with VSYNC on displays that are not G-SYNC or G-SYNC Compatible when using Streamline 2.0.0a or earlier.

Note: DLSS Frame Generation is currently not compatible with VSync for Vulkan applications.

Q: What does the Auto Scene Change Detection feature do?

A: Auto Scene Change Detection aims to automatically prevent Frame Generation from producing difficult-to-create frames between a substantial scene change. It does this by analyzing the in-game camera orientation on every input DLSS Frame Generation frame pair. Auto Scene Change Detection simplifies integration of new DLSS 3 titles, is backwards compatible with all DLSS 3 integrations, and supports all rendering platforms.

Q: What does the Auto mode do?

A: Auto mode (Dynamic Frame Generation) can intelligently enable/disable DLSS Frame Generation to help ensure positive performance scaling.

Q: I've enabled Auto mode, but I still see performance drops. Why is that?

A: Auto mode uses a heuristic that samples CPU and GPU performance over time. If the game's engine performance varies greatly within this sampling period, then Auto mode may enter a state that results in lower instantaneous performance.

Q: How can I verify that DLSS Frame Generation is enabled?

A: You can enable the DLSS on-screen indicator utility by running “ngx_driver_onscreenindicator.reg” before starting your application. Once DLSS Frame Generation is enabled, you’ll see confirmation on the overlay in-game.

Once finished, you can run “ngx_driver_onscreenindicator_off.reg” from “scripts” folder to disable the indicator.

```
DLSSG SDK - DO NOT DISTRIBUTE - CONTACT NVIDIA TO OBTAIN DLLs FOR YOUR TITLE
NVIDIA DLSSG v3.8.0 CL 34946406 - endpoint_ada - Develop - VK - FGW: Off - Output 3440x1440 - MVec 1720x720 - 2x - Hudless: Yes - UIAlpha: Yes
Frame Count (Real, Generated) - Since Create (5226, 5226) - Since Reset (1792, 1792)
Vis Mode (ctrl+shift+v) = Stats (ctrl+shift+t) = Keybinds (ctrl+shift+k)
```

Developers can also leverage the sl.imgui.dll overlay which was introduced with Streamline 2.0.



Q: How can I verify that hudless buffers are tagged?

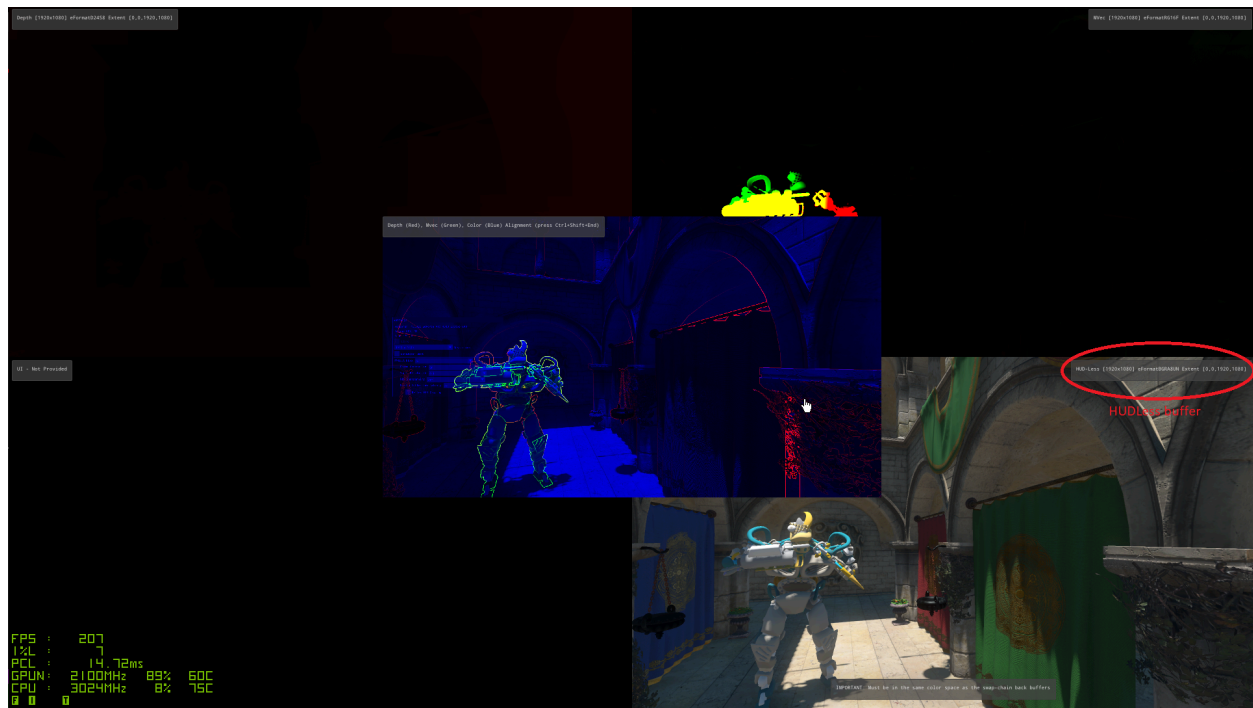
A: You can enable the DLSS on-screen indicator utility by running “ngx_driver_onscreenindicator.reg” before starting your application. Once DLSS Super Resolution is enabled, you’ll see “Use Hudless: Yes” in the overlay if hudless buffers are tagged.

Once finished, you can run “ngx_driver_onscreenindicator_off.reg” from “scripts” folder to disable the indicator.

```
DLSSG SDK - DO NOT DISTRIBUTE - CONTACT NVIDIA TO OBTAIN DLLs FOR YOUR TITLE
NVIDIA DLSSG v3.8.0 CL 34946406 - endpoint_ada - Develop - VK - FGW: Off - Output 3440x1440 - MVec 1720x720 - 2x - Hudless: Yes - UIAlpha: Yes
Frame Count (Real, Generated) - Since Create (5226, 5226) - Since Reset (1792, 1792)
Vis Mode (ctrl+shift+v) = Stats (ctrl+shift+t) = Keybinds (ctrl+shift+k)
```

Developers can also leverage the sl.imgui.dll overlay which was introduced with Streamline 2.0.1.

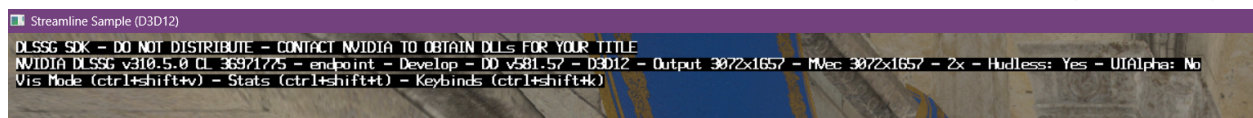
1. Confirm DLSS FG is running
2. Confirm DLSS Frame Generation is running
3. Press Debug key (CTRL+SHIFT+INS)
4. Visualizes all buffers (allows dev to confirm hudless buffer is correct)
5. Confirm functionality on screen



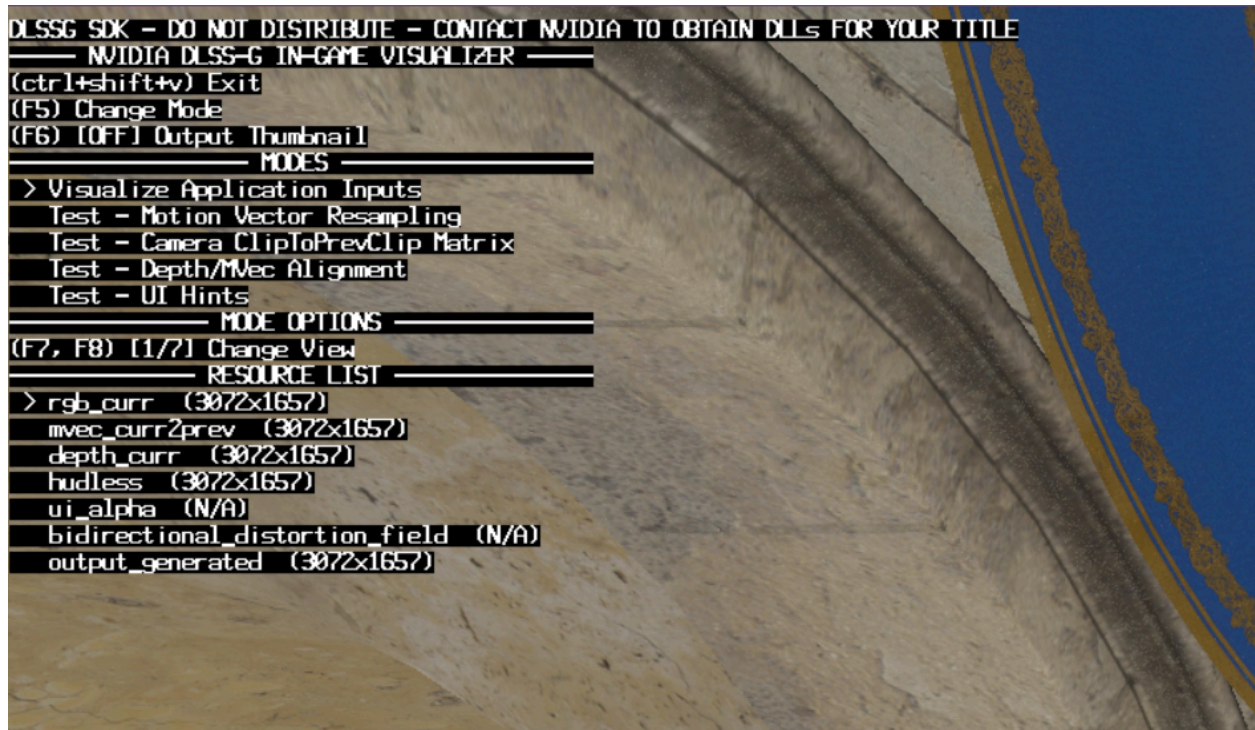
New in Streamline v2.10.0 “DLSS-G Enhanced In-Game Debug Visualization”

This new feature contains helpful in-game visualization modes to validate the inputs are correct in a more targeted way than the sl.imgui solution. This mode requires only the development nvngx_dlssg.dll binary (bin/x64/development), no other changes are required to either Streamline or any of the plugins.

SL SDK v2.10.0 NGX Snippet Overlay contains a “Vis Mode” keyboard shortcut, defaults to (ctrl+shift+v).



Vis Mode Entrypoint (ctrl+shift+v)



After pressing the keybind, you can see the in-game render text change. The visualization modes are shown in the MODES category. The independent mode options are shown below as well as a text description describing the visualization mode options. The (F5) key will change the mode.

In the case the DLSS-G optional input bidirectional distortion field is present, that can be toggled in each mode to validate the correction of the pre-render data (depth,mvecs) with the post-process distortion effect data (rgb_color).

In the first mode, **Visualize Application Inputs**, the integrator can individually inspect every input resource to the DLSS-G algorithm. Press the F7 and F8 keys to cycle through. Missing inputs are marked N/A. The F6 key will also display a preview of the output buffer, to help correlate intermediate resource alignment with the color pixels. Useful when visualizing Depth.

Visualizing Depth



Visualizing HUDless



Test - Motion Vector Resampling

Only objects not characterized by the input geometric curr2prev motion vector field are highlighted in this view.



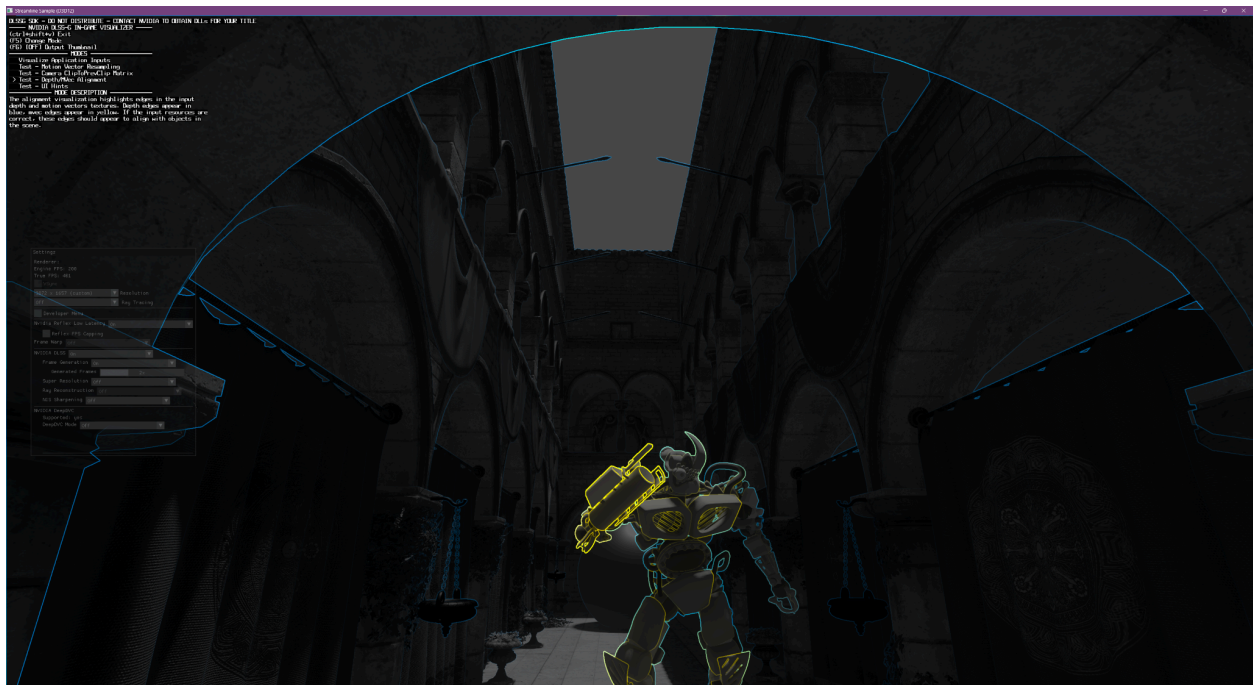
Test - Camera clipToPrevclip Matrix mode

IsDynamic visualization mode, validating camera matrices are correct w.r.t. Input depth and motion vector textures. Only objects moving independently of the camera should be annotated here.



Test - Depth/Mvec Alignment

Validates the edge alignment of the depth and motion vector input textures w.r.t. the output buffer.



Test - UI Hints

Validates the input UI hint buffers are accurately annotating UI related pixels. The user can toggle the usage of the HUDless or UI Color and Alpha buffer using the appropriate keybinds to explore if the resource is helping.



For more information, [please reference Section 15.0 of the DLSS FG Programming Guide](#)

Reflex

Q: Should I add GPU hardware, vendor, or driver version check for Reflex?

A: Do NOT do any GPU hardware, vendor, or driver version check. As long as the Reflex plugin is supported, always make all set feature constants and evaluate feature calls to the Reflex Streamline plugin.

- The Reflex Streamline plugin handles all such abstractions. It is completely transparent to the game.
- Reflex Low Latency mode does not work for all GPU hardware, vendors, and drivers versions, but it is already abstracted by the Reflex Streamline plugin.
- The game only needs to disable/enable the Reflex Low Latency UI using the `lowLatencyAvailable` setting, as detailed below.

Q: Should the Reflex marker calls be skipped when Reflex Low Latency mode is Off?

A: Do NOT skip any Reflex calls when Reflex Low Latency mode is set to Off.

- The markers are also used to measure latency.
- It is important to measure latency, even when Reflex Low Latency mode is Off.

Q: How to check if Reflex Low Latency is available on the host system to disable/enable UI?

A: The game should check `lowLatencyAvailable` to determine whether to show the Reflex Low Latency UI or not.

- Use `lowLatencyAvailable` for UI only. **As long as the Reflex Streamline plugin is supported, the game should do everything else Reflex related the same way, even when `lowLatencyAvailable` is false.**

Q: Reflex sleep is negatively impacting framerate. What is going on?

A: Reflex On should not impact more than 4% FPS. Reflex On + Boost should not impact more than 7% FPS.

- If the impact is slightly higher, then please contact your Producer/CM so they can file a bug and have the driver team investigate further.
- If the impact is significantly higher, please read on.

Does it repro with On + Boost only, or does it repro with On as well? If it is On + Boost only, then it is likely caused by RTBO (Render Thread Bound Optimization).

- Another symptom is that FPS is capped to 30.
- Try setting `bUseMarkersToOptimize = false` to verify if the issue goes away.
- This can happen if the time within render submit start and render submit end/present end markers increases as Reflex sleep increases.
- The problem can usually be avoided by having better render submit and present start/end markers. But some games may just not be compatible with RTBO and cannot set `bUseMarkersToOptimize = true`.

Is FPS capped to 15? If so, it is likely a Reflex controller issue that the driver team should investigate.

Huge FPS impact is likely caused by the game calling Reflex sleep more than once per frame. Reflex sleep should not be called more than once per frame.

- It can happen if the game at some point after Reflex sleep and before simulation starts decides to restart the frame from the beginning. In that case, the Reflex sleep might have been called twice or more, and therefore impact the FPS greatly.
- Without Reflex sleep, restarting the frame from the beginning and repeating the input message and other steps before the check would have no problem. But with Reflex sleep, restarting a frame would call it and sleep multiple times. That's a problem.
- One possible solution in this case is to make sure to not call Reflex sleep more than once before simulation starts.

Q: What are the HW/SW requirements for Reflex features?

A: Please refer to the following table:

Feature	GPU IHV	GPU HW	Driver Version	Support Check	Key Setting / Marker
Reflex Low Latency	NVDA Only	GeForce 900 Series and newer	456.38+	sl::ReflexSettings::lowLatencyAvailable	sl::ReflexConstants::mode
Auto-Configure Reflex Analyzer	NVDA Only	GeForce 900 Series and newer	521.60+	sl::ReflexSettings::flashIndicatorDriverControlled	sl::PCLMarker::eTriggerFlash
Frame Rate Limiter	NVDA Only	All	All	Always	sl::ReflexConstants::frameLimitUs

NOTE: The sub-features are distinct to each other without any cross-dependencies. Everything is abstracted within the plugin and is transparent to the application. The application should not explicitly check for GPU HW, vendor, and driver version. The application should do everything the same regardless of sub-feature support and enablement. The only 2 exceptions are Reflex UI and Reflex Flash Indicator (RFI). The application must disable Reflex UI based on `sl::ReflexSettings::lowLatencyAvailable`. And the application must not send `sl::PCLMarker::eTriggerFlash` when `sl::ReflexSettings::flashIndicatorDriverControlled` is false.

Q: How can I confirm that Reflex is enabled and working on NVIDIA hardware?

A: You can quickly and easily verify Reflex is enabled and functioning properly by running a simple test found within the [NVIDIA Reflex SDK](#).

1. Browse to "Reflex_Verification" folder
2. Open Command Prompt (run as Administrator) and run FVSKSetup.exe
3. From Command Prompt (run as Administrator), run ReflexTestSetup.bat

4. Start game - you should see the Reflex Verification HUD in-game which will show whether Reflex is enabled
 - Reflex Verification HUD requires driver 531.18 or higher
 - You may need to add the game to the app profile: NVIDIA Control Panel -> 3D Settings -> Program Settings -> Select Program to Customize -> Add -> Select a program -> Add Selected Program -> Apply
5. Press "ALT+T" to start test (you'll hear 2 beeps)
6. After ~5min, test should be done (you'll hear 3 beeps)
7. Reference the Command Prompt and ensure "passed"

```

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.

```

Q: Why is GPU Utilization being reported as 0%?

A: This issue can arise after you've just updated your graphics card drivers or FVSDK. Please be sure to restart the system after the installations are completed before you run the test.

PCL Stats

Q: What are the GPU hardware, vendor, or driver version requirements for PCL Stats?

A: PCL has no GPU hardware, vendor, or driver dependency. It is always supported.

Q: Where is the checklist for PCL Stats?

A: There is 1 checklist for validating both Reflex and PCL Stats together.

Q: How can I confirm that PCL is working on non-NVIDIA hardware?

A: You can quickly and easily verify Reflex is enabled and functioning properly by running a simple test within the [NVIDIA Reflex SDK](#).

1. Browse to "Reflex_Verification" folder
2. Open Command Prompt (run as Administrator) and run PrintPCL.exe
3. Start game
4. Press "ALT+T" to start test
5. Refer to the command prompt and confirm the PCL value. If the value is not 0.0, then PCL is working.
6. Press "Ctrl + C" to exit test