Investigating the Feasibility of NVIDIA's Multi Frame Generation Within a Custom Engine

Alex Emeny

MSc Game Programming

Falmouth University

August 29, 2025

Statement of Originality and Intellectual Property Rights Agreement

This dissertation is submitted to fulfil the requirements for the MSc in Game Programming at Falmouth University. It is the result of my own labour, except where otherwise indicated and cited. This report may be freely copied and distributed, provided that the source is acknowledged.

Student Number: ae319035

Abstract

Image interpolation has progressed from basic digital processing to advanced frame generation, which can utilise deep learning models to raise the output frame rate by synthesising intermediate frames fast enough for real-time graphics. NVIDIA's DLSS 4 Multi Frame Generation builds upon their frame generation breakthrough in 2022; this new technology can insert up to three additional AI-generated frames from every traditionally rendered frame, allowing the doubling to quadrupling of the final output frame rate.

Currently, NVIDIA's revolutionary suite of DLSS 4 rendering technologies, such as Ray Reconstruction, Super Resolution and Multi Frame Generation, is exclusively available on RTX 50-series GPUs. These emerging technologies allow consumers to have improved performance and enjoyment from video games, enabling higher visual fidelity with lower processing power, and have the potential to become a standard within the game industry.

This dissertation evaluates whether the increasing availability of this new technology allows custom engines developed by independent developers or small teams can be utilised to achieve higher frame rates.

Using benchmarking across representative scenes, giving a testbed for stress testing, Multi Frame Generation highlighted improvements to frame time statistics and the potential to enhance other small engines.

Future work may extend this framework by exploring alternative test scenarios, a larger gameplay feature list or by extending hardware testing to other 40 or 50 series GPUs.

Keywords: Deep Learning Super Sampling; Multi Frame Generation; Streamline; NVIDIA; DLSS

Contents

1 Introduction

In the modern day, graphics processing units (GPUs) get used in a variety of applications, from powerful 3D rendering to non-graphics tasks such as machine learning and scientific applications. The computation cost of video game graphics has been increasing with developers pushing realism and graphical fidelity to new limits. To keep up with these ever-growing demands, companies like NVIDIA have started to produce groundbreaking novel approaches, including leveraging machine learning in the render pipeline.

With the new NVIDIA RTX 50 Series cards, DLSS 4 Multi Frame Generation (MFG) was announced, advancing frame generation to produce up to three additional frames per traditionally rendered frame [1]. Unlike earlier versions of DLSS that rely on traditional optical flow to interpolate frames, DLSS 4 leverages an additional AI network to more efficiently and accurately predict frame transitions [2].

Quick to gain access to DLSS were the big commercial engines, Unity and Unreal Engine; developers can find DLSS 4 within Unreal Engine through an official NVIDIA plugin, which provides access to MFG and other technologies in the suite [3]. Unity also has an official NVIDIA plugin, which is slightly behind Unreal, offering only DLSS Super Resolution and no frame generation.

Outside of NVIDIA's official plugins, they also allow support for custom engines; developers can make use of DLSS through NVIDIA's open source Streamline SDK. Streamline simplifies the integration of the latest NVIDIA technologies into applications and games [4]. Streamline sits between the game and the render API, so developers do not need to integrate each SDK manually but can identify which resources are required [4]. NVIDIA provides a variety of programming guides within its SDK, helping developers have a smoother process when integrating. With Streamline being such a new technology, it does lack external support and documentation outside of these guides.

This project is built in C++ from scratch, making use of the Vulkan API and studying the implementation of Streamline into a minimal custom engine. There are currently no other independent studies which investigate the use of DLSS 4 within this context. This study will provide contributions towards bridging this gap for other independent or small team developers interested in this technology.

1.1 Research Questions

This works contributions aim to answer the following:

- When NVIDIA's new technology becomes more widely available, can developers make use of small custom engines to utilise DLSS 4 Multi Frame Generation to increase frame rate without the need for commercial engines?
- How does DLSS 4 Multi Frame Generation affect render-to-present latency, GPU power draw and energy used per frame compared to native rendering within smaller engines?

1.2 Scope

This study examines the feasibility of integrating DLSS 4 MFG into a small custom Vulkan engine via Streamline, with a focus on performance experiences to guide others interested in using this technology. Measurements will be made across fixed stress scenes at 1080p with 4x frame multipliers on an RTX 5080 GPU. This analysis is time-limited for out-of-scope comparisons with image-quality comparisons or any other features the DLSS 4 suite has to offer.

2 Literature Review

2.1 Introduction to AI Frame Generation

The introduction of DLSS 3 in September 2022 brought AI-powered Frame Generation with NVIDIA's RTX 40 Series GPUs. This feature brought their technology from just upscaling techniques to AI-generated pixels through their Optical MFG, introducing a new intermediate step to the rendering process. This technology was incorporated into the NVIDIA Ada Lovelace architecture on the 40 Series. It analyses two sequential in-game images and takes in generated motion vector data from the game engine, enabling it to predict a new frame that would fit between them. Combining the DLSS-generated frames with their super-resolution technology enables DLSS 3 to reconstruct seven-eighths of the displayed pixels with AI, boosting frame rates up to 4x to native rendering [5].

DLSS 3 Frame Generation executes as a post-process event on the GPU, allowing it to leverage NVIDIA's powerful 4th-generation AI Tensor Cores and boost frame rates even when a game is bottlenecked by the CPU [5]. Previous generations of RTX cards also made use of Tensor Cores, but at a significantly lower scale and efficiency than the Ada Lovelace architecture could provide. With the RTX 2080 providing 2nd generation cores to RTX 3080 3rd generation cores, the third series delivers 2.7X higher peak operation throughput compared to the second series [6]. This jump in power and efficiency in the hardware allowed NVIDIA to produce their revolutionary Frame Generation technology.

2.2 Improved Multi Frame Generation

With the release of the RTX 50 Series cards came the improvement of the DLSS technology to DLSS 4, which brings one key innovation: Multi Frame Generation. This is a technique that enables the generation of up to three additional frames per transitionally rendered frame, extending on Frame Generation from DLSS 3. With the new 5th generation tensor cores from NVIDIA's Blackwell architecture and over double the total AI TOPS (Trillion of Operations Per Second), this allowed for computational power not seen before [7].

NVIDIA notes in their in-house DLSS 4 paper that Frame Generation is hard, and developing their own benchmark, training, and testing pipeline was important to maintain quality. NVIDIA wanted to achieve a similar level of image quality to DLSS

3's Frame Generation, while also wanting these new additional frames be generate in less time. With DLSS 3, a single 4K image could be generated in around 3.25ms on an RTX 4090, in contrast to the new DLSS 4 frame Generation, creating three additional frames each in around 1ms on average. This feat was achieved through a combination of new architectural changes to split the DLSS 3's frame output to run one heavy model once per rendered frame and a small, faster model per generated frame, cutting the cost of each frame dramatically [2].

Extra enhancements towards frame generation included physical hardware-based changes, such as a new frame pacing logic called Flip Metering. This hardware was introduced to the Blackwell display engine, enabling the GPU to manage display timings with multiple frames more precisely. The introduction of this technology allowed support for the GPU when generating higher frame rates beyond what state-of-the-art monitors can display; any missteps in this dance between the neural generation process and the GPU scheduling would cause skips, stutters or hitches, which would destroy a smooth high frame rate experience [2].

2.2.1 How Multi Frame Generation works

MFG extends DLSS frame synthesis by reconstructing the render path so multiple frames can develop from a single render input. NVIDIA replaced their prior hardware optical-flow stage with a learned AI optical-flow model to reduce memory requirements. NVIDIA reports this new frame generation model is 40% faster and uses 30% less VRAM. Together with their latest AI model, the total per-frame cost is kept to a few milliseconds, even through the use of five different AI models executing each frame [1]. This updated framework still allows the same usual game data requirements, such as motion vectors, depth buffers, final colour passes, and more [1].

In development, engines integrate MFG through NVIDIA Streamline DLSS-G; by tagging and passing a variety of buffers, common constants, and a unique frame index for each render. The underlying AI model can predict and generate intermediate frames [8]. NVIDIA provides programming checklists to show the integration steps and correct frame generation behaviour. They suggest integration of NVIDIA Reflex a technology to help avoid latency issues. NVIDIA opens up state information such as "numFramesActuallyPresented" when calculating FPS, as their additional intermediate presents won't get tracked by many engines by default [9].

2.3 Technologies used in similar studies

To determine good performance, it is imperative to look towards past studies and see what has been reliable and best for the job. While there are few formal studies on MFG specificity, DLSS-focused studies have consistently relied on frame time analysis with PresentMon-based tooling such as FrameView. In academia, recent work on Energy implications of upscaling and frame generation in games [10] uses FrameView to log similar data captures for the GPU.

Many studies make use of Intel PresentMon or tools built on it, such as AMD OCAT and CapFrameX. PresentMon tools are excellent for frame-time/latency capture across both DX11/12 and Vulkan, but they do not natively support recording of GPU power consumption [11]. FrameView is a low-overhead analyser that records detailed frame-time scheduling data, supports Vulkan crucially, and leverages PresentMon for analytics so results align with industry practice.

2.4 The gap in related research literature

MFG is an exclusive part of the new DLSS 4 suite and is confined to RTX 50 Series GPUs. With this technology being new as of the writing of this paper, there is a lack of available literature regarding MFG as a whole. There are yet to be any independently created academic papers, or, being extremely scarce while this dissertation was conducted. This dissertation paper will fill in some of that gap by analysing DLSS 4's MFG within the context of independent developers as RTX 50 series GPUs become more widely available and considered for implementation more commonly.

3 Methodology

3.1 System overview

This MFG technology enhances user experience in complex environments, such as games. To test its capabilities, choosing the right tools is essential to achieving accurate and representative results. The program will be written in C++, using no external libraries and SDKs, except for Vulkan for graphics rendering, GLFW for window and input support, and finally Streamline SDK for DLSS 4 integration. Since this study explores the use of DLSS 4 within custom engines and will use custom-made architecture, external libraries of pre-made engines would not fit the specific approach implementation for this study. C++ was chosen for its performance as it is a compiled language and will operate fast at run time. The engine will make use of Vulkan for its lower-overhead performance cost and direct hardware control, allowing a reduction in CPU load when testing GPU performance targets.

The central component of this research is the RTX 5080 GPU, on which this project will be tested, enabling DLSS 4 capabilities. This hardware allows this paper to address the research questions of investigating its usefulness within current-day custom game engines and its potential future use as this technology becomes more widely used.

The program includes capabilities for loading a variety of scenes to test the frame generation's performance, including a variety of static and moving models with heavy instancing. The program's design mimics the basics of a rendering/game engine with pipelines slightly altered to produce inputs for frame generation. Allowing Streamline to intercept and extend the render path meant routing Vulkan calls such as *present* through proxy functions built on Vulkan's PFN pointer types.

3.2 Technologies used throughout

The project sits on top of the modern graphics Vulkan 1.2+ API, allowing the Streamline SDK to be present, enabling the engine to tag and pass the required inputs into the open-sourced framework. NVIDIA Streamline framework simplifies the integration of NVIDIA DLSS and various super-resolution technologies from other hardware vendors for easy implementation into applications [9]. Streamline's DLSS-G plugin runs the frame generation passes and allows users with the correct hardware and software to access Frame Generation and MFG.

Streamline's current SDK targets Windows with DirectX 11 and Vulkan 1.2 or higher, and brings the DLSS-G plugin as a prebuilt module while exposing the rest of the framework in source [9].

For testing and measuring the capabilities of MFG, this project will make use of NVIDIA FrameView, which captures frame rate, frame times, latency, and GPU power via on-screen stats and CSV logging with a low overhead for comparing native output to MFG [12].

3.3 Implementing Streamline SDK

Integrating MFG through NVIDIA Streamlines involved implementing and tagging a variety of parameters. The renderer produces HUD-less final output colour, depth data, and motion vector images; camera parameters, such as near/far, jitter settings, and motion vector scale, are passed as constants. Creating formats in the expected way by Streamline avoids any extra copies, minimises extra costs and keeps the measurements close to the feature test. Initialisation of Streamline requires the call of *slInit()* as early as possible, before any APIs are invoked, to attach to the render pipeline as it is created successfully. Similarly, *slShutdown()* must be called before destroying any instances, devices, or components in the engine.

3.4 System specifications

Table 3.1: Hardware configuration of the test system.

Component	**Specification**
CPU	AMD Ryzen 7 7700X (8C/16T) @ 4.5 GHz
GPU	NVIDIA GeForce RTX 5080 (16 GB)
Memory	32 GB DDR5-4800Mhz (2×16 GB)
Motherboard	TUF GAMING A620M-PLUS ASUSTeK (BIOS V1616 2023)
PSU	Corsair RM1000x SHIFT

Table 3.2: Software configuration and runtime settings.

Item	**Version / Setting**
OS	Windows 11 Home (Build 26100)
Graphics Driver	NVIDIA 581.08 (GeForce Game Ready)
Graphics API	Vulkan 1.3
Streamline / DLSS	Streamline 2.8.0
Render Settings	1920×1080; FG multipliers 2×/3×/4×; V-Sync Off

3.5 GPU performance comparison across different scenarios

To best evaluate MFG behaviour, a variety of different scenes were tested with FrameView for constant statistics on GPU performance. Each scenarios test was executed with identical parameters and settings, each testing the various capabilities of frame generation. These scenes are loaded similarly to how a game's level would be loaded, with an initial upfront cost on the CPU pushing data to the GPU, then decreasing the majority of the workload to prevent bottlenecks and ensure the GPU can reach its performance limits.

Each experiment puts MFG to the test to determine if it improves lightweight engines, showing a consistent overall increase in frames. Each test recorded the entire render pipeline, showing GPU power draw, render-to-present latency, energy used per frame and constant frame rate stats.

3.6 Differentiating performance from larger engines

Testing the capabilities of MFG at its core is challenging within the context of large, general-purpose engines; commercial engines are built mainly upon additional plugin abstractions, specialised conservative synchronisation, and render interface abstraction over several platform graphics APIs. With all these optimisations, they are not representative of lightweight engines that do not already have official support. To attribute performance statistics in real-world conditions for independent developers, the study will involve the development of a minimal Vulkan renderer where attachments, evaluations, and the current render path will be explicitly controlled.

4 Results and Analysis

This paper selects four scenes for testing the capabilities, three of which push the GPU through an intensive scene that dramatically drops the frame rate, and one that allows for fast frame times on a minimal scene.

The test scenes under load each used between 10,000 objects with a mid-range poly count of 10,000. These scenes included a static scene with little movement (Static), panoramic camera motion around static objects (Pan), and objects constantly moving and overlapping on a sine/cosine path (Move), updating on the CPU. Finally, the low-cost scene involved 500 static quads in a simple, repeating pattern (Over).

Throughout all the test scene scenarios, frame measurements were obtained from the FrameView application. Four test scenes were executed comparing performance on an RTX 5080; the tests included examining MFG on 4x generation compared to native 1x rendering.

4.1 Frame rate

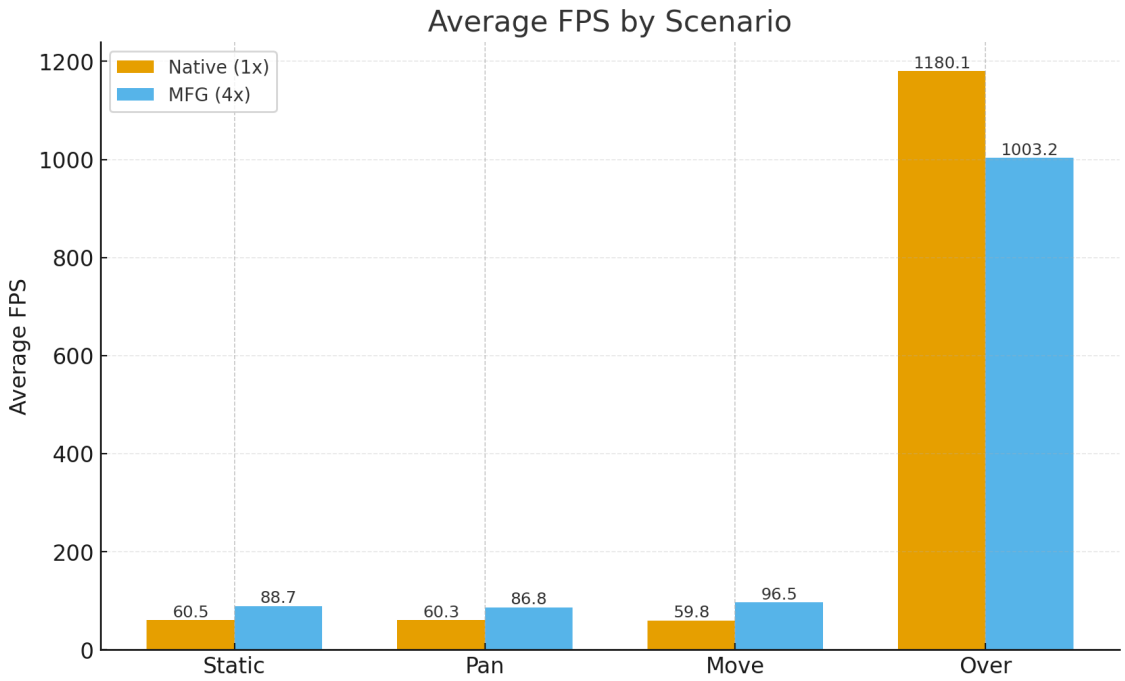

Figure 4.1: Results from 4 test scenes showing average FPS between MFG 4x and native 1x rendering.

MFG generally shows a consistent increase in frame rate compared to native rendering under load. The plot shows that an increase in frame rate between 40-60% was achieved under the intensive scenes. This test further shows that, under normal frame rate conditions, there is an increase, however, when native rendering is completed quickly, the MFG technique cannot keep up and would take longer to process the extra frame than it took to generate a native frame.

4.2 Render-to-present latency

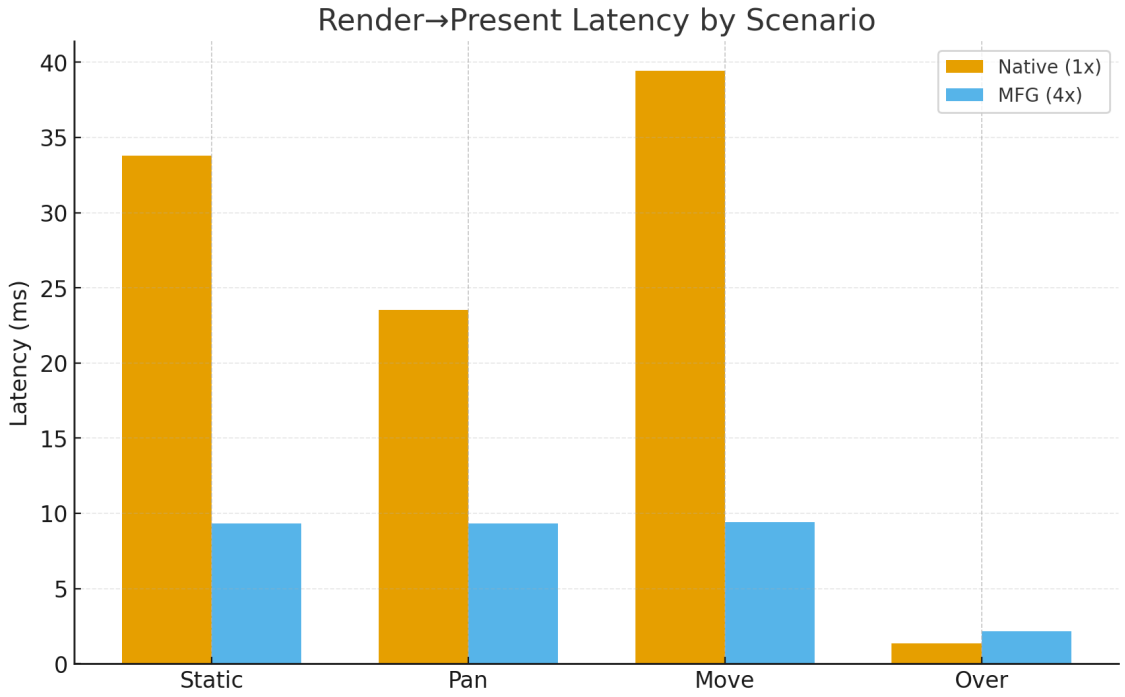

Figure 4.2: Results from 4 test scenes showing latency between rendering to presenting for MFG 4x and native 1x rendering.

Within this test, MFG showed significantly faster time between when the frame's rendering work is submitted and it is presented, except for the "Over" scene, which presents a slightly faster time due to the speed at which the frames are rasterised.

4.3 GPU power draw

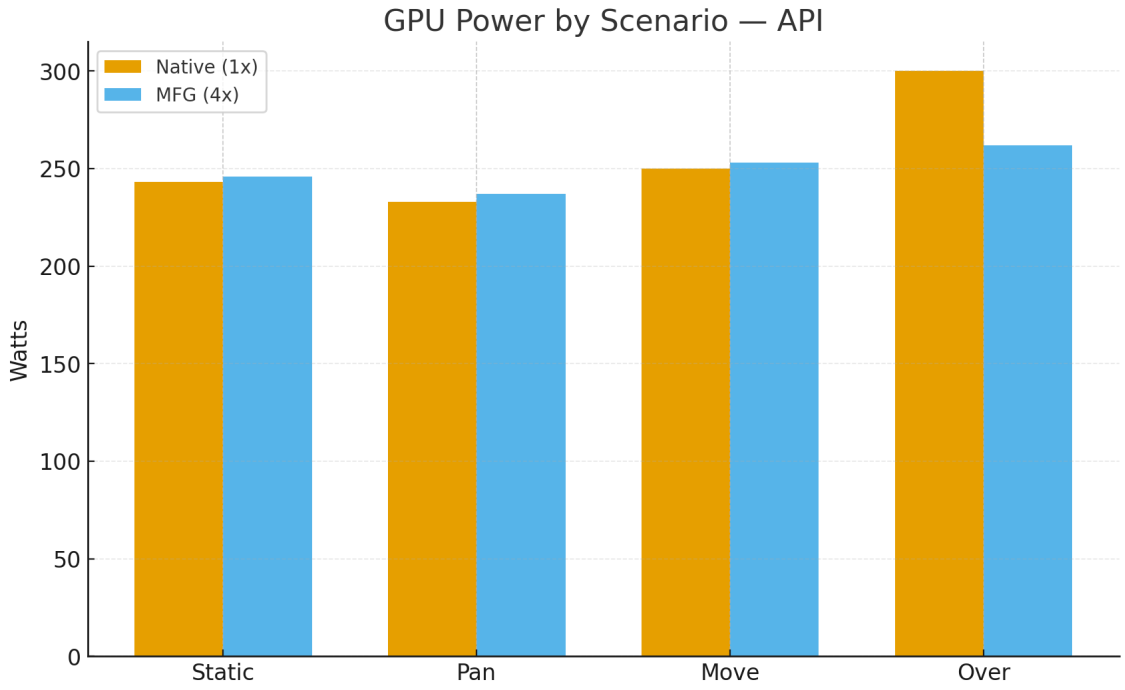

Figure 4.3: Results from 4 test scenes showing GPU power draw between MFG 4x and native 1x rendering.

The GPU power draw test showed only slight variations in power consumption across all tests. The first three scenarios showed little difference between MFG and native rendering, but consistently showed an extra power draw during frame generation. Within the final test, similarly, native rendering was an outlier, pulling more when not capped by frame generation.

4.4 Energy used per frame

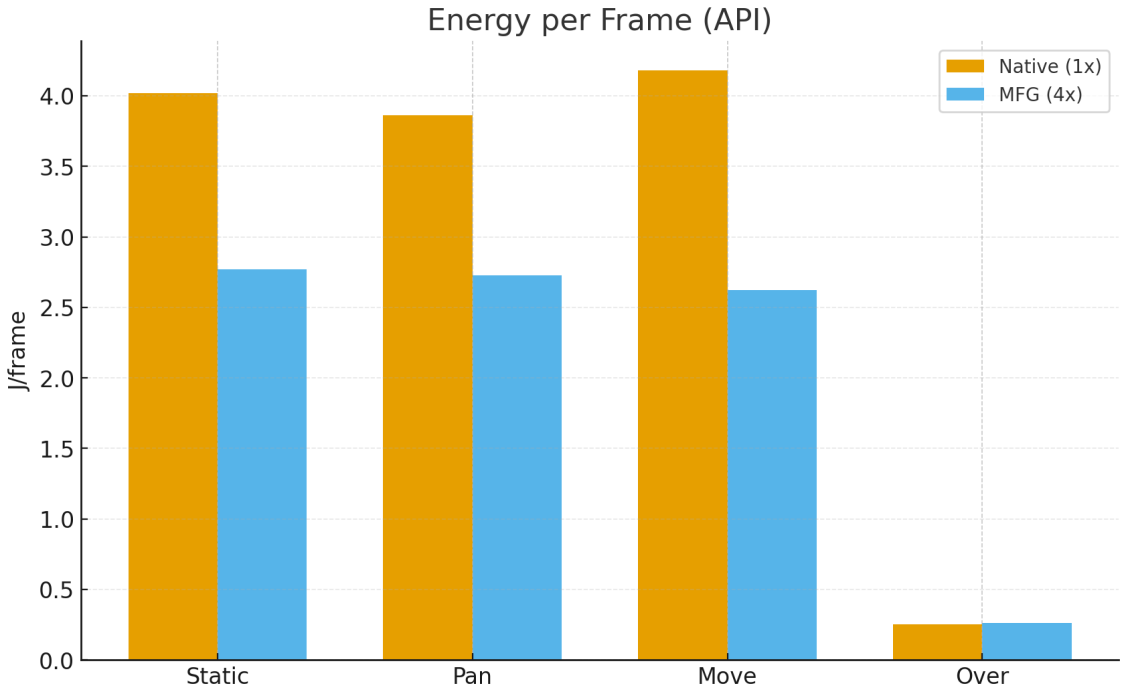

Figure 4.4: Results from 4 test scenes showing energy used per frame between MFG 4x and native 1x rendering.

Differentiating from the overall average power draw, each frame tells a different story: native rendering, which requires significant power for the entire render pipeline, versus MFG's additional frames, which average out to lower power consumption. In the lighter-weight scenario, with less work per frame, a dramatic drop in power usage is observed, as native rendering proves more efficient than the AI frame generation technique.

4.5 Findings

MFG reveals consistent patterns across each of the generic high-pressure scenes, generally being more performant and positively improving the frame statistics values. Conversely, under fast processing times, frame generation harmed the frame statistic values.

During the first test of frame rate, there was seen to be a general increase in frame rate for the heavy load scenarios; these tests showed that MFG's extra frames positively had an impact and had a consistent 40-60% uplift. This uplift is evident in the

calculation, where generated frames take less time to produce than native frames. This was seen to have an adverse effect on the lightweight scene; AI-generated frames took extra time to calculate and interpolate compared to the speed at which natively rendered frames could be produced overall decreasing the amount of frames per second.

Render-to-present time is the GPU timing from when a frame finishes rendering work and is submitted until it is sent off for presentation. Throughout this test, MFG showed a significant decrease in render-to-present times, except for the lightweight scene. With frame generation, not every frame has to be fully rasterised; as a result, the GPU work per presented frame should be lighter, naturally shortening the time required for the extra frames. This decrease is precisely what is observed in the first three tests. With lightweight frames, the used native generation is faster, being comparable to the frame rate test, allowing for faster presentation speeds and a lower score than MFG.

During the testing for GPU power draw, only slight power variations were observed, as the frame rate was uncapped. This allowed the GPU to run at full capacity, consistently consuming around the same amount of power. However, as seen in the graphs, MFG showed a general increase in power consumption anyway. This can be theorised as with MFG, the card could utilise more specialised AI Tensor cores alongside the rendering CUDA cores. The outlier test of the lightweight scene yielded the most power from the tests. This could be because, in less intensive scenes, there is less calculation and waiting for data to be passed around, compared to the other scenarios, allowing for more consistent power usage. A mention should be made of the example of MFG, which uses less power than native rendering; however, this comes with a lower frame rate and positively a lower computation cost per frame.

Finally, the energy used per frame was evaluated, which showed differentiation from GPU power usage by flipping MFG to achieve an overall lower usage per frame. As stated earlier, the generated AI frames exhibit lower power consumption compared to native frames actually improving power usage on average.

Overall, MFG improved performance in the high-pressure scenes. Delivering a consistent frame uplift and lower render-to-present times by avoiding full rasterisation on every generated frame. Within the lightweight scene, the frame generation added overhead, which outweighed the benefits by reducing frame rate and slightly increasing render-to-present time relative to native rendering.

With GPU power draw, this idea stayed relatively similar, with small MFG increasing

consumption mainly from extra AI workloads, but with the positive of an overall lower per-frame power cost.

To summarise these results would be to say that MFG is a handy and powerful tool that, under normal game circumstances, only uplifts and improves GPU performance. MFG has seen a downside effect on the frame statistics when being outpaced by natively rendered frames in the more unlikely event of a game having a very lightweight run time.

5 Discussion

5.1 Answering the research questions

This study assessed whether the RTX 5080 GPU could make use of MFG for a positive impact on frame statistics within the context of a small custom engine. This paper explores when NVIDIA's new technology becomes more widely available, could developers utilise it positively without the need for commercial engines.

Within the test scenarios, the results of MFG showed an uplift effect on frame statistics under heavier load scenes. Under load, MFG was seen to increase frame rate, reduce average render-to-present time and decrease the amount of GPU power per frame compared to native rendering. This was only seen to have an adverse effect on efficient light load scenes, where native rendering actively outpaced the AI-generated frames' speed and had the opposing effect on frame time statistics. The use of a lightweight scene was an outlier that may not refer to a generic game scene with more complexity, but rather to more basic scenarios where it is worth deactivating frame generation.

On the question of whether developers could use this technology in custom engines rather than needing to make use of commercial engines, the results indicate yes. This technology is still very new, with DLSS 4 coming out at the start of 2025; with DLSS 4 being locked to a 50 series GPU, the number of users that the developer would be targeting could be small. With smaller development teams having limits to less development time and costs, implementation might not be viable for small engines or development teams yet. With technology advancing and newer GPUs becoming cheaper over time, this can definitely become a standardised feature for engines, especially those with high-fidelity scenes.

5.2 Limitations

The findings in this paper must be interpreted with regard to some limitations and constraints of the methodology. The evaluation uses multiple controlled scenes in Vulkan rather than a complete game with a variety of features and events. As such, the results are not influenced by gameplay systems such as AI, physics, UI frameworks or networking, which means the results do not capture the content and variability that these extra systems would introduce.

A further limitation is that measurements were taken on a single RTX 5080 system; outcomes for the results may differ across other 50 series cards and individual systems across different architectures.

Finally, the study did not investigate the use of all MFG multipliers and presents the use in context with other DLSS 4 technologies and how they may perform together. This may limit conclusions on stability and performance at higher multiplier or when combined with multiple architectures.

DLSS 4 is a closed-source, rapidly evolving technology where model updates or driver changes can shift behaviour over time and change the output of results.

5.3 Future work

Expanding hardware coverage beyond a single RTX 5080 to include multiple different Blackwell architecture GPUs, with potential work in Ada Lovelace single Frame Generation on the 40 series. Test all MFG multipliers and presets under matching conditions across one or multiple devices to expand on which mode might have the best impact.

Measurements taken can be further strengthened with end-to-end latency for response time, with extra frames testing for scenarios that require fast response time, such as competitive games. Future work can look towards testing VRAM usage throughout to combat NVIDIA's in-house statistics on DLSS 3 to DLSS 4, or similar to this study, with comparisons to native rendering.

To expand on the methodology, gameplay features previously mentioned, such as AI, physics, UI frameworks or networking, should be incorporated to test the limits in varying scenarios. Studies against alternate vendor solutions from NVIDIA's MFG, such as AMD's FSR or Intel's XeSS frame generation, under identical tests to clarify which behaviours affect different frame time statistics.

6 Conclusion

This study set out to determine whether MFG can be integrated into a lightweight, custom Vulkan engine with performance and stability comparable to that of larger, commercial engines.

The evaluation showed that enabling MFG yields an output FPS uplift over native rendering across the heavier load scenes. Further, scenario observations indicated that a heavy lifting scene, which more accurately represents a game's environment, would increase frame rate, decrease render-to-present time and improve GPU power usage per frame.

Conversely, in the light workload scene there was a negative impact on frame rate performance with MFG enabled. This is likely caused because this scene's natively rendered frames outpacing the time taken for MFG to compute the intermediate frame and, in turn, negatively affect frame statistics.

This study showed evaluation of render-to-present timing, average frame rate, GPU power draw and finally energy used per frame to help evaluate the differences between 4x MFG and 1x native rendering. These tests showed an insight into the working of NVIDIA's technology and the promising contributions it can have on the industry.

Ultimately, MFG emerges as a feasible, practical option for small engines which are looking to improve the frame performance of high-intensity games. Currently, with limited access to the technology for the users, this may not be a good option for developers to spend their time on while reaching smaller audiences. Despite this, it shows promising use for these developers in the future to consider implementation.

References

[1] NVIDIA Corporation (2025a). *NVIDIA DLSS 4 Introduces Multi Frame Generation & Enhancements For All DLSS Technologies*. URL: https://www.nvidia.com/en-us/geforce/news/dlss4-multi-frame-generation-ai-innovations (accessed 20/07/2025).

[2] — (2025b). *DLSS 4: Transforming Real-Time Graphics with AI*. URL: https://research.nvidia.com/labs/adlr/DLSS4 (accessed 28/06/2025).

[3] — (2025c). *NVIDIA DLSS*. URL: https://developer.nvidia.com/rtx/dlss (accessed 05/07/2025).

[4] — (2025d). *Streamline*. URL: https://developer.nvidia.com/rtx/streamline (accessed 15/06/2025).

[5] — (2022a). *NVIDIA Introduces DLSS 3 With Breakthrough AI-Powered Frame Generation for up to 4x Performance*. URL: https://nvidianews.nvidia.com/news/nvidia-introduces-dlss-3-with-breakthrough-ai-powered-frame-generation-for-up-to-4x-performance (accessed 21/07/2025).

[6] — (2021). *NVIDIA AMPERE GA102 GPU ARCHITECTURE*. URL: https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.1.pdf (accessed 21/08/2025).

[7] — (2025e). *Compare GeForce Graphics Cards*. URL: https://www.nvidia.com/en-gb/geforce/graphics-cards/compare (accessed 23/07/2025).

[8] — (2024). *Download NVIDIA Streamline SDK*. URL: https://developer.nvidia.com/rtx/streamline/get-started (accessed 02/07/2025).

[9] NVIDIA RTX (2025). *Streamline - DLSS-G*. URL: https://github.com/NVIDIA-RTX/Streamline/blob/main/docs/ProgrammingGuideDLSS_G.md (accessed 02/07/2025).

[10] Berg, Mikael (2024). *Energy implications of upscaling and frame generation in games*. URL: https://www.diva-portal.org/smash/record.jsf?dswid=935&pid=diva2%3A1888392 (accessed 02/08/2025).

[11] Intel (2025). *Optimize Performance with PresentMon's Precision*. URL: https://presentmon.com (accessed 02/08/2025).

[12] NVIDIA Corporation (2022b). *Integrating Frame Benchmarking & Power Tool*. URL: https://images.nvidia.com/content/geforce/technologies/

frameview/frameview-1-4-user-guide-web-version.pdf (accessed 19/07/2025).