FrameView

Integrated Frame Benchmarking & Power Tool

BETA

USER GUIDE

7/9/2019

INTRODUCTION

FrameView is a software tool designed to capture and measure performance and power utilization of PC-based graphics hardware. It's especially useful for measuring frame rates and GPU power usage when running stressful "real world" PC gaming scenarios. FrameView captures performance and power data with minimal overhead so as not to impact frame rates or gameplay. FrameView includes an overlay that shows performance and power metrics as a game is being played. It also allows benchmarks runs to be captured and charted in detailed reports.

FrameView captures game performance metrics including average and percentile frame-per-second (FPS) data for both single- and multi-GPU configurations. Percentile FPS data is valuable for illustrating the severity and frequency of stutters that can interrupt gameplay. FrameView has been optimized particularly for detailed frame time, present, and display scheduling metrics for measuring stutter.

FrameView captures real-time power measurements for both total board power (including graphics memory) and GPU chip-only power through application programming interfaces (APIs), which is publicly-available software that communicates with the hardware and returns data. This removes the need for additional, and sometimes costly frame capture and power measurement hardware.

While FrameView reports both chip and board power for NVIDIA graphics cards, it currently only reports what appears to be something in-between chip power and TGP for AMD graphics cards because this is all AMD reports in their API.

Since FrameView captures both performance and power data, it allows users to create accurate perf-per-watt statistics to determine GPU efficiency by viewing the performance of the GPU alongside the power it uses. This metric is called performance-per-watt (PPW). The lower the power utilization and the higher the game performance, the better the perf-per-watt.

API Support: DirectX APIs (versions 9-12), OpenGL, Vulkan

Single-GPU Configs: NVIDIA GeForce, AMD, Intel

Multi-GPU Configs: NVIDIA SLI, AMD Crossfire, MSHybrid- and Optimus-based platforms

Display Support: G-SYNC, Non-G-SYNC, ASYNC (including FreeSync) single monitor setup

Screen Modes: Full Screen, Windowed, UWP apps

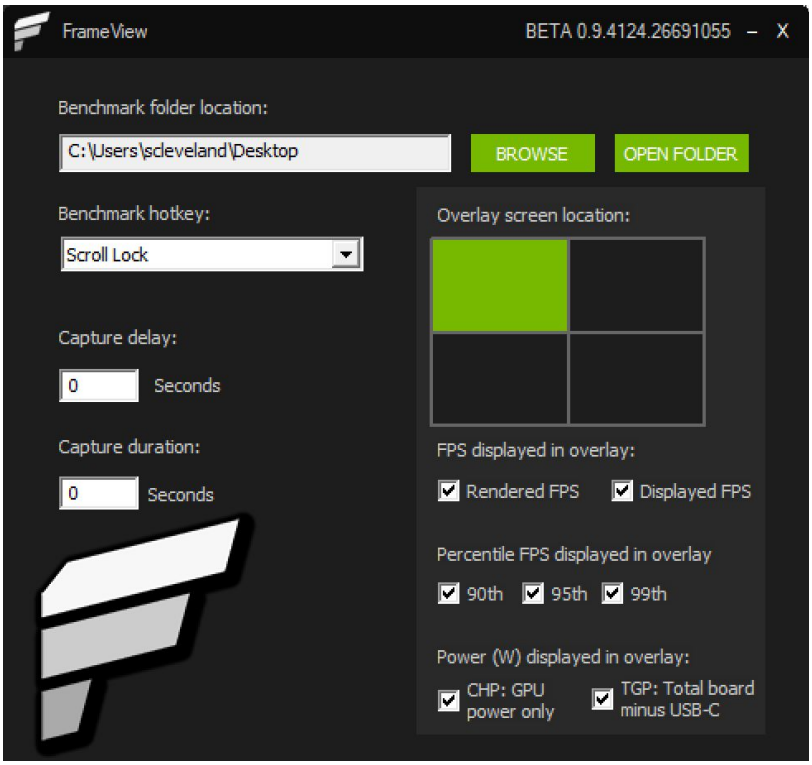
OS Support: Windows 10

Logging: Average rendered FPS, displayed FPS, frame percentages (90/95/99th percentiles), board and GPU power (AMD API appears to report something in-between chip power and board power)

Overlay: Average rendered FPS, displayed FPS, frame percentages (90/95/99th percentiles), dropped frames, performance per watt, GPU power. (Note: Overlay does not currently work for DX9/10 or Vulkan)

FrameView Interface & Settings

This section outlines the functionality of the settings provided in the FrameView interface.



Benchmark folder location

This is where the benchmark logs will be saved. Use the Browse button to choose a location and the Open Folder button to access saved results in Windows File Explorer.

Benchmark hotkey

This is the button assigned to start and stop the benchmarking process. At this time, FrameView only supports **Scroll Lock** and **F10** as the benchmarking hotkeys.

Capture delay

This will delay the capture of a game by the seconds specified in the window. The default is 0 seconds.

Capture duration

This will set a capture time limit for the benchmark. The default is 0 seconds, which means the benchmark capture logging must be manually started and stopped with the hotkey. When the time limit is set to a number greater than 0, the benchmark logging must still be manually started, but it will be automatically stopped after the specified capture duration.

Overlay screen location

FPS, percentiles, and power information will be displayed by default in the upper-left corner of your monitor when running a game. To change the overlay location, click a different quadrant in the FrameView interface, represented by green blocks. More information about the overlay can be found in the Overlay section.

> ***NOTE:*** *The overlay is automatically disabled during benchmarking to ensure more accurate results. The overlay will return once the benchmark hotkey is pressed a second time.*

FPS displayed in overlay

> **Rendered FPS:** When enabled, FrameView will measure and report timestamps at the *beginning* of the graphics pipeline. This metric indicates the smoothness of the animation delivered to the GPU.
>
> **Displayed FPS:** When enabled, FrameView will measure and report timestamps at the *end* of the graphics pipeline. This metric provides an indicator of what the user actually sees displayed on screen.

Percentile FPS displayed in overlay

Selecting these will show the 90th, 95th, and 99th frame time percentile calculations in the overlay.

> **90th:** 10 frames out of 100 are slower than this frame rate. Put another way, 90% of the frames will achieve at least this frame rate.
>
> **95th:** 5 frames out of 100 are slower than this frame rate. Put another way, 95% of the frames will achieve at least this frame rate.
>
> **99th:** Only 1 frame out of 100 is slower than this frame rate. Put another way, 99% of the frames will achieve at least this frame rate.

Power (W) displayed in overlay

Selecting these will show real-time power reporting.

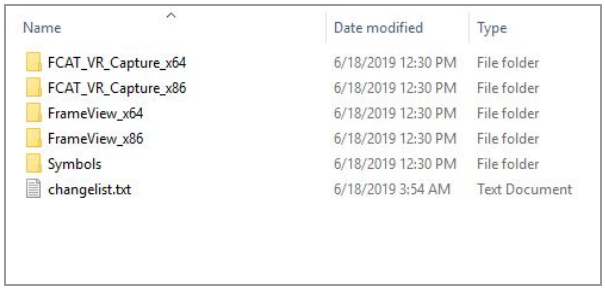
GPU Only Power: Power will be shown for the GPU (chip) only in watts, post regulator.

TGP (Total Board minus USB-C): Power will be shown for the everything (chip + board), except for the power used by USB-C devices on GeForce RTX GPUs.

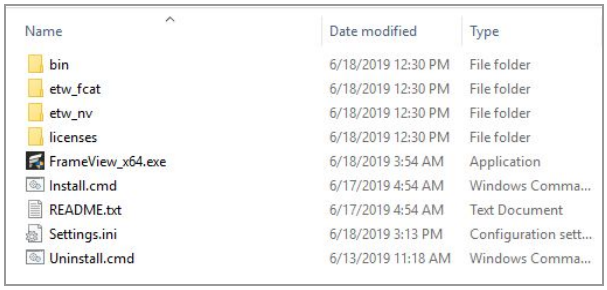
NOTE: *While FrameView accurately reports both chip and board power for NVIDIA graphics cards, the AMD API used by FrameView appears to report a value in-between chip power and board power for AMD graphics cards. Therefore it's currently not possible to use FrameView to directly compare AMD GPU power to NVIDIA GPU power. AMD power will be displayed in the overlay as AMDPWR.*

Installing & Running FrameView

1. Navigate to \FrameView_x64\ and run **Install.cmd** as **Administrator** to install. Do this by right-clicking on **Install.cmd** in File Explorer and selecting **Run as administrator**.
   It can be found in the directory where the FrameView package was downloaded/saved to.



Top-level directory structure of FrameView

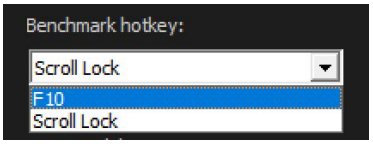


Inside FrameView_x64 folder

2. Launch **FrameView_x64.exe** on Windows x64 systems, and **FrameView_x86.exe** on Windows x32 systems. Launch by double-clicking the application executable file.

3. Select the preferred directory where benchmark results will be stored.

4. At this time, FrameView supports the **Scroll Lock** and **F10** as the benchmarking hotkeys.



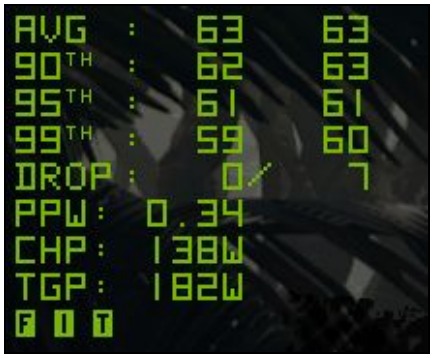
5. Launch a game. The FrameView overlay should show up in the designated area chosen in the FrameView interface.
6. Push the benchmarking hotkey (default is **Scroll Lock**) to begin benchmarking. The overlay will disappear during data collection to reduce overhead in the captured data.
7. Press the benchmarking hotkey again to stop data collection. The overlay will reappear in the designated area.
8. Exit the game and return to FrameView. Click the **Open Folder** button to view benchmark results.

9. FrameView Results will be saved as .CSV files with an application and timestamp name. Consider renaming the files or creating a directory to reflect the GPU, game, and settings tested.

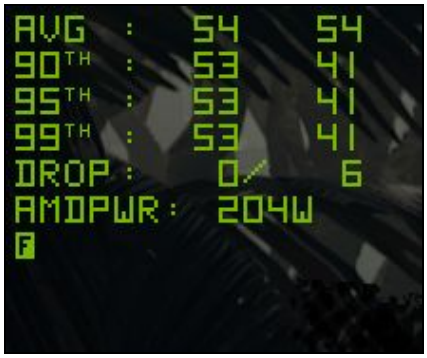
Overlay

The overlay will appear in games with proper API support when FrameView is running in the background. If the overlay does not appear, make sure to check if FrameView is running. Adjust the overlay screen location in the FrameView settings to move the overlay to a different corner of the screen.

When benchmarking is enabled through the hotkey, the overlay will disappear. Removing the overlay reduces overhead to ensure a more accurate capture of the game data. The overlay will reappear when the hotkey is pressed again and capture is stopped or capture duration expires.



FrameView Overlay when using NVIDIA GeForce GPU



FrameView Overlay when using AMD GPU

The FrameView overlay displays two columns of real-time data. These are comprised of rendered and displayed FPS data. The column on the left is shown when the Rendered FPS setting in the settings is enabled, and the column on the right is shown when the Displayed FPS setting is enabled. Please refer to the HOW FRAMEVIEW WORKS section later in this guide for more information about rendered and displayed FPS.

The overlay displays real-time data for average FPS, percentile FPS (90/95/99), dropped frames, chip power, and perf-per-watt. Please refer to the FRAMEVIEW SCAN FILES section for more information.

NOTE: *Overlay information will not be shown in DX9/10 and Vulkan-based games. However, data capture is supported and the information will be properly logged. A future version of FrameView will add overlay support for Vulkan-based games.*

Overlay Mode Tags

The overlay also includes three mode tags. These letters are used to provide information about game settings that can impact overlay data reporting and data captures.

F (Full Screen): Running the game at full screen will ensure that accurate performance results are measured at the resolution specified in the game.

W (Windowed mode): If the game runs in windowed mode, pressing the Alt+Enter keys on your keyboard while the game is running can often force the game into fullscreen mode. Check the game settings if that doesn't work.

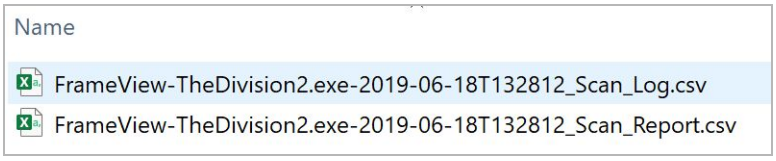
I (Independent flip): iFlip (also called Independent flip), is the mode where the app is simulating as if it was running in Full Screen Exclusive mode.

T (Tearing): When Vertical Sync is disabled, the full performance of the game can be measured (due to higher frame rates beyond the refresh rate of the monitor). However, a major artifact of disabling Vertical Sync is tearing. This is the optimal method for testing game performance.

V (Vsync ON): Vertical Sync is enabled, which forces the frame rate of the game to synchronize with the refresh rate of the monitor. However, this limits the frames that can be displayed, and will not show the full performance potential of the hardware being measured.

FRAMEVIEW SCAN FILES

Two files are created once the benchmarking is completed using the hotkeys. Both files are created in standard CSV (comma-separated values), which can be imported into Microsoft Excel, OpenOffice Calc, or Google Sheets, and are saved in the benchmark folder location that is specified in the FrameView user interface. They are named using the application process name and include date and time stamps.



FrameView Scan Log

The smaller file is called the Scan Log and it contains the high-level data from a captured run using the FrameView hotkey.

When opened, the file will look like this:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	NumRende	NumDispla	Time (ms)	RenderMin	RenderMa	RenderAvg	DisplayMin	DisplayMa	DisplayAvg	Render90	Render95	Render99	Display90	Display95	Display99
2	681	681	10781	37.5	85	66.4	52.8	67.5	66.5	62.4	60.4	55.3	66.2	65.6	60.5
3															
4															

This table explains each header and the data contained in it:

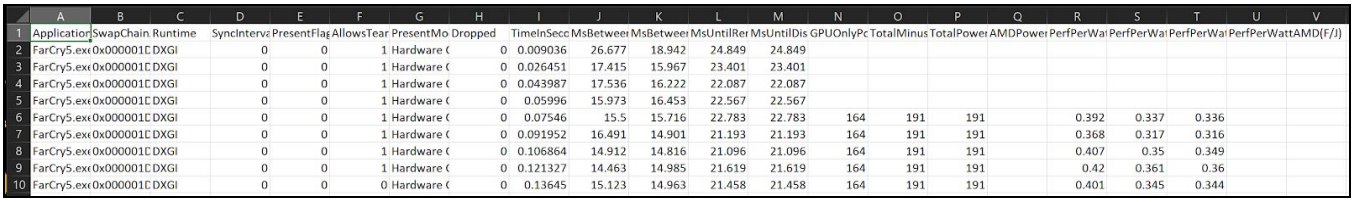
SCAN LOG HEADER	DESCRIPTION
NumRenderFrames	Total number of rendered frames captured
NumDisplayFrames	Total number of displayed frames captured
Time (ms)	The amount of time that comprises the capture
RenderMin	Uses Rendered FPS scheduling metrics to show the minimum (single lowest) FPS
RenderMax	Uses Rendered FPS scheduling metrics to show the maximum (single highest) FPS
RenderAvg	Uses Rendered FPS scheduling metrics to show the overall average FPS
DisplayMin	Uses Displayed FPS scheduling metrics to show the minimum (single lowest) FPS
DisplayMax	Uses Displayed FPS scheduling metrics to show the maximum (single highest) FPS
DisplayAvg	Uses Displayed FPS scheduling metrics to show the overall average FPS

Render90	Uses Rendered FPS scheduling metrics to show 90th percentile data 10 frames out of 100 are slower than this frame rate. 90% of the frames will achieve at least this frame rate.
Render95	Uses Rendered FPS scheduling metrics to show 95th percentile data 5 frames out of 100 are slower than this frame rate. 95% of the frames will achieve at least this frame rate.
Render99	Uses Rendered FPS scheduling metrics to show 99th percentile data 1 frame out of 100 are slower than frame rate. 99% of the frames will achieve at least this frame rate.
Display90	Uses Displayed FPS scheduling metrics to show 90th percentile data 10 frames out of 100 are slower than this frame rate. 90% of the frames will achieve at least this frame rate.
Display95	Uses Displayed FPS scheduling metrics to show 95th percentile data 5 frames out of 100 are slower than this frame rate. 95% of the frames will achieve at least this frame rate.
Display99	Uses Displayed FPS scheduling metrics to show 99th percentile data 1 frame out of 100 are slower than this frame rate. 99% of the frames will achieve at least this frame rate.

FrameView Scan Report

The largest file is called the Scan Report and it contains all of the data from a captured run using the FrameView hotkey.

When opened, the file will look like this:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Application	SwapChain	Runtime	SyncInterva	PresentFlag	AllowsTear	PresentMo	Dropped	TimeInSecc	MsBetweer	MsBetweer	MsUntilRer	MsUntilDis	GPUOnlyPc	TotalMinus	TotalPower	AMDPower	PerfPerWa	PerfPerWa	PerfPerWa	PerfPerWatt	AMD(F/J)
2	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.009036	26.677	18.942	24.849	24.849									
3	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.026451	17.415	15.967	23.401	23.401									
4	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.043987	17.536	16.222	22.087	22.087									
5	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.05996	15.973	16.453	22.567	22.567									
6	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.07546	15.5	15.716	22.783	22.783	164	191	191		0.392	0.337	0.336		
7	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.091952	16.491	14.901	21.193	21.193	164	191	191		0.368	0.317	0.316		
8	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.106864	14.912	14.816	21.096	21.096	164	191	191		0.407	0.35	0.349		
9	FarCry5.exe	0x000001C	DXGI	0	0	1	Hardware (	0	0.121327	14.463	14.985	21.619	21.619	164	191	191		0.42	0.361	0.36		
10	FarCry5.exe	0x000001C	DXGI	0	0	0	Hardware (	0	0.13645	15.123	14.963	21.458	21.458	164	191	191		0.401	0.345	0.344		

This table explains each header and the data contained in it:

SCAN REPORT HEADER	DESCRIPTION
Application	Process name (if known)
SwapChainAddress	Swap chain address
Runtime	Swap chain runtime (e.g., D3D9 or DXGI)

SyncInterval	Sync interval used
PresentFlags	Present flags used
AllowsTearing	Whether tearing possible (1) or not (0)
PresentMode	Present mode
Dropped	Whether the present was dropped (1) or displayed (0)
TimeInSeconds	Time since PresentMon recording started
MsBetweenPresents	Time between this Present() API call and the previous one
MsBetweenDisplayChangeActual	Time between when this frame was displayed, and previous was displayed
MsUntilRenderComplete	Time between present start and GPU work completion.
MsUntilDisplayed	Time spent inside the Present() API call
GPUOnlyPower(W)	GPU/Chip/ASIC power, post-regulator
TotalMinusUSBCPower(W)	Board power excluding USB-C
TotalPower(W)	Board power
PerfPerWattGPUOnly(F/J)	Performance per Watt considering MsBetweenPresents for performance and GPU/Chip/ASIC power
PerfPerWattTotalMinusUSBC(F/J)	Performance per Watt considering MsBetweenPresents for performance and board power excluding USB-C
PerfPerWattTotal(F/J)	Performance per Watt considering MsBetweenPresents for performance and board power

Charting Scan Report Data

Use the information below to help determine which of the captured data that you need to chart.

Charting FPS Data

Average Rendered FPS

MsBetweenPresents (Rendered FPS) should be used to chart the average rendered FPS. This data is captured from the beginning of the graphics pipeline and indicates the smoothness of the animation delivered to the GPU. This is the data that is typically provided by other benchmarking capture tools.

Average Displayed FPS

MsBetweenDisplayChangeActual (Displayed FPS) should be used to chart the average displayed FPS. This data is captured from the end of the graphics pipeline and is an indicator of what the user actually sees displayed on screen.

Charting Percentile Data

FrameView Scan Logs provide percentiles which illustrate the frame rates that given percentages of frames can achieve, and FrameView Scan Reports provide frametime data to calculate the frametimes below which a given percentage of frametimes will fall. A 95th percentile frametime is the value below which 95% of the frametimes are found. Example: if the dataset has a rendered frametime 95th percentile of 16.67ms, then 95% of the frames were rendered faster than 16.67ms.

Rendered and Displayed Percentiles

MsBetweenPresents (Rendered FPS) should be used to understand the percentile distribution of frametimes for frames that have been *rendered*. Once again, this data is captured from the beginning of the graphics pipeline and is a metric that captures how smooth the game animation was.

MsBetweenDisplayChangeActual (Displayed FPS) should be used to understand the percentile distribution of frametimes for frames that have been *displayed*. Once again, this data is captured from the end of the graphics pipeline and is an indicator of what the user actually sees displayed on screen.

NOTE: *Percentile data is highly sensitive to stutter. In order to obtain the most meaningful results, exclude loading screens, menus, and large frametime spikes when selecting a dataset.*

Charting Power Data

Chip Power Consumption

GPUOnlyPower(W) should be used for charting the average GPU (chip) power consumption. It is sampled at intervals of 100msec.

Chip Perf Per Watt (PPW)

PerfPerWattGPUOnly(F/J) should be used for charting performance per watt data for the GPU (chip) where F is frames and J is joules (one joule is the equivalent of one watt of power radiated or dissipated for one second). So F/J would be frames per second (F/S) divided by watts (J/S).

F/J = (F/S) / (J/S)

For more details on measuring power of GPUs, please refer to the **NVIDIA GeForce GPU Power Primer**.

Board Power Consumption

TotalMinusUSBCPower(W) should be used for charting Total Graphics Power (TGP). TGP is the maximum power in watts that a power supply should provide to the graphics board. TGP is also defined as the average power consumed by the entire graphics board subsystem while executing a very stressful "real world" application. TBP or Total Board Power is essentially the same as TGP. Using this data will be more accurate since it does not include the power used by devices that may be connected to the USB-C connector on NVIDIA GeForce RTX graphics cards. It is sampled at intervals of 100msec.

Therefore it's currently not possible to use FrameView to directly compare AMD GPU power to NVIDIA GPUs, because AMD's API calls currently report a single power value that appears to more than chip-only power, but less than full board power. It may be useful to ask AMD if they can report chip-only and full board power with the APIs, similar to NVIDIA.

IMPORTANT NOTES: *While FrameView accurately reports both chip and board power for NVIDIA graphics cards, the AMD API used by FrameView currently only reports a value that appears to be in-between chip power and board power for AMD graphics cards.* ***Therefore it's currently not possible to use FrameView to directly compare AMD GPU power to NVIDIA GPU power.*** *FrameView will be updated to capture total board power and chip power for AMD if they make such information publicly available in their API.*

Also note that ***FrameView is not as accurate as interposer/riser card techniques for measuring idle chip or idle board power.*** *It is accurate for load testing, so it is best to use FrameView when running real-world applications that stress the GPU.*

Board Perf Per Watt (PPW)

PerfPerWattTotalMinusUSBC(F/J) should be used for charting performance per watt data for Total Graphics Power (TGP) where F is frames and J is joules (one joule is the equivalent of one watt of power radiated or dissipated for one second). So F/J would be frames per second (F/S) divided by watts (J/S).

F/J = (F/S) / (J/S)

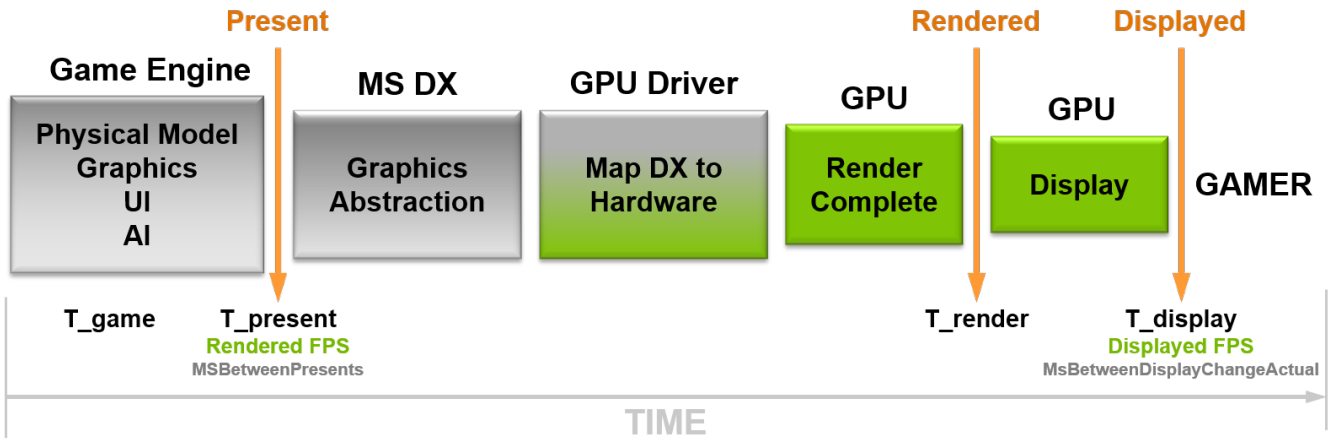
For more details on measuring power of GPUs, please refer to the **NVIDIA GeForce GPU Power Primer.**

HOW FRAMEVIEW WORKS

To provide performance data for an accurate comparative analysis of GPUs, FrameView measures timestamps at the beginning of the graphics pipeline to provide a metric indicating the smoothness of the animation delivered to the GPU, and at the end of the pipeline to provide an indicator of what the user actually sees displayed on screen.

Frame Rendering Pipeline

The diagram below shows how game frames are created at the beginning of the pipeline and their path to the display.



FrameView provides performance data that is captured in the Present and Displayed portions of the game/graphics pipeline. Data from these two areas will always be reported in the logs. You can choose which you would like shown in the overlay using the FrameView settings in the interface. They are called Rendered FPS and Displayed FPS.

> **Rendered FPS (MsBetweenPresents)** measures timestamps from the beginning of the graphics pipeline and is a metric indicating the smoothness of the animation delivered to the GPU. This is the data that is typically provided by other benchmarking capture tools.
>
> **Displayed FPS (MsBetweenDisplayChangeActual)** measures timestamps at the end of the game pipeline and is an indicator of what the user actually sees displayed on screen.

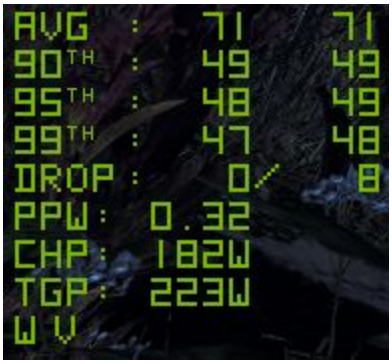
Stutter is the variation between T_game and T_display. This data is also reported by FrameView in the logs. The header is called MsUntilRenderComplete and it measures the time between present start and GPU work completion.

TROUBLESHOOTING

Frames are capped at 30fps, 60fps, 75 fps (or any other framerate) in a game

The game may have a frame rate cap “framecap” (internal frame limiter) that prevents rendering faster than a specified rate. Check the game settings to see if a framecap is set.

This can also be caused by having V-sync enabled (ON), which will synchronize the frame rate to the refresh rate of the monitor. Check the game settings and disable V-sync to ensure that the frame rate is no longer tied to the monitor refresh rate. The FrameView overlay will show an “T” when V-sync is OFF (for tearing) and will show a “V” when V-sync is ON.



Power results are not showing in the FrameView overlay

You may need to rerun Install.cmd again. Navigate to \FrameView_x64\ and run **Install.cmd** as **Administrator** to install. Do this by right-clicking on **Install.cmd** in File Explorer and selecting **Run as administrator**. It can be found in the directory where the FrameView package was downloaded/saved to.

The FrameView overlay is not being displayed over a game

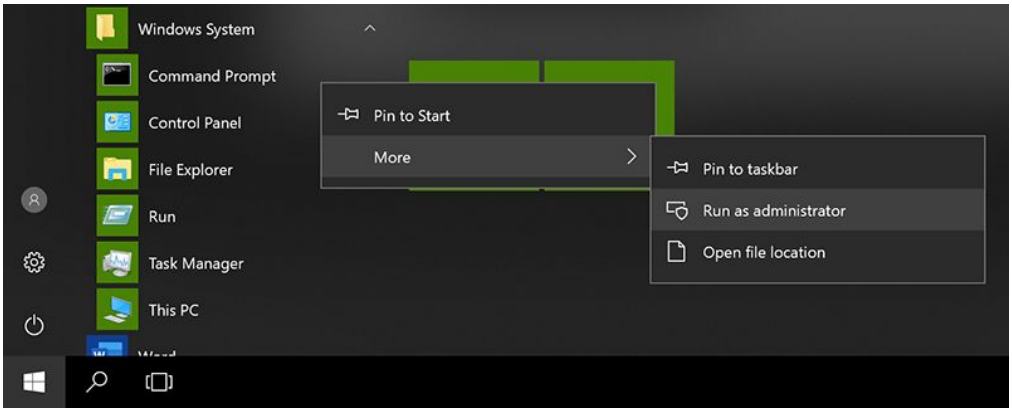
Overlay information will not be shown in DX9/10 and Vulkan-based games. However, data capture is supported and the information will be properly logged. A future version of FrameView will add overlay support for Vulkan-based games.

If the overlay is not being displayed over DX12 or OpenGL games, please follow the steps below for manually stopping and restarting the overlay application process in Windows.

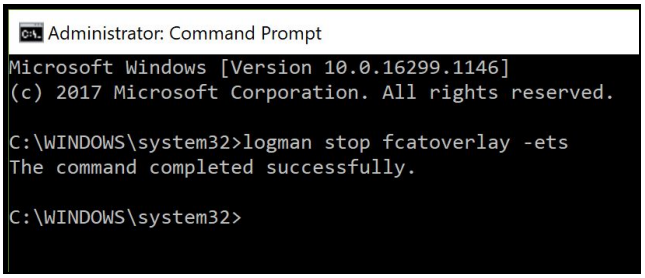
Scan Report and Scan Log files are not being created after capture

You may need to manually stop the overlay to restart the application process fresh in Windows.

Open a Command Prompt with administrator privileges. Click the Start button, scroll down on the apps list to the Windows System folder and then click it to open the contents. Right-click on Command Prompt, click More, and then click Run as Administrator (as shown in the image below).



In the Command Prompt, type the following: **logman stop FCATOverlay -ets**



Close FrameView and relaunch and try again.

Running FrameView and FRAPS Concurrently

Since FrameView and FRAPS both are both hooking into application processes, FrameView might not work with x64-bit applications if FRAPS is already running in the background.

1. Launch FrameView
2. Launch Game
3. Let FrameView overlay appear
4. Launch FRAPS
5. Close FRAPS before closing the game, and then follow steps 1-4 for the next run

Notice

ALL INFORMATION PROVIDED IN THIS WHITE PAPER, INCLUDING COMMENTARY, OPINION, NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, and GeForce are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright